Understanding Malware Behaviour in Online Social Networks and Predicting Cyber Attacks

A dissertation submitted for the partial fulfilment

for the degree of doctor of philosophy by

Amir Javed

School of Computer Science and Informatics

Cardiff University

2019

STATEMENT 1
This thesis is being submitted in partial fulfilment of the requirements for the degree of PhD

Signed	**Date**
[illegible]	April 6, 2020

STATEMENT 2
This work has not been submitted in substance for any other degree or award at this or any other university or place of learning, nor is it being submitted concurrently for any other degree or award (outside of any formal collaboration agreement between the University and a partner organisation)

Signed	**Date**
[illegible]	April 6, 2020

STATEMENT 3
I hereby give consent for my thesis, if accepted, to be available in the University's Open Access repository (or, where approved, to be available in the University's library and for inter-library loan), and for the title and summary to be made available to outside organisations, subject to the expiry of a University-approved bar on access if applicable.

Signed	**Date**
[illegible]	April 6, 2020

DECLARATION
This thesis is the result of my own independent work, except where otherwise stated, and the views expressed are my own. Other sources are acknowledged by explicit references. The thesis has not been edited by a third party beyond what is permitted by Cardiff University's Use of Third Party Editors by Research Degree Students Procedure.

Signed	**Date**
[illegible]	April 6, 2020

Word Count **45,440**

(Excluding summary, acknowledgements, declarations, contents pages, appendices, tables,diagrams and figures, references, bibliography, footnotes and endnotes)

Dedication

To my parents, for their support and patience.

Acknowledgements

I would like to thank my supervisors Professor Pete Burnap and Professor Omer Rana for their continued support and guidance. I would also like to thank the staff and PhD students of the School of Computer Science and Informatics, both past and present, for their support and feedback.

Abstract

The popularity of Twitter for information discovery, coupled with the automatic shortening of URLs to save space given the 280 character limit, provides cybercriminals with an opportunity to obfuscate the URL of a malicious Web page within a tweet. Once the URL is obfuscated, the cybercriminal can lure a user into clicking on it with enticing text and images before carrying out a cyber attack using a malicious Web server. This is known as a *drive-by download* and has been reported to account for 48% of web-based attacks. In a *drive-by download* a user's computer system is infected while interacting with the malicious endpoint, often without them being made aware the attack has taken place. An attacker can gain control of the system by exploiting unpatched system vulnerabilities, and this form of attack currently represents one of the most common methods employed. In order to counter drive-by download attacks on Twitter, this thesis contributes to the existing literature on detecting malware on online social networks by shifting the focus towards the effects of malware on user machines, and away from the malware signature and dynamic behaviour, which can be obfuscated. Initially we developed a drive-by download detection model for Twitter that was successful in classifying URLs into malicious and benign with an F-measure of 0.81 during training, and 0.71 while testing on an unseen dataset. The model was then extended into a predictive model that was able to predict whether the URL was pointing to a malicious Web page with 0.99 F-measure (using 10-fold cross-validation) and 0.833 F-measure (using an unseen test set) at *1 second into the interaction with a URL*. These provide a novel contribution with which it is possible to kill the connection to the server before an attack has completed - thus proactively blocking and preventing an attack, rather than reacting and repairing at a later date. This thesis also contributes to the broader literature on malware propagation by uncovering both social and content-based factors that aid in the propagation of a tweet containing a link to a malicious Web server. This was achieved by gathering data from seven different and diverse sporting events over a period of three years. The data were then analysed to answer questions including: why are certain Tweets retweeted more than others? is virality partly driven by psychological arousal? and, is the act of retweeting affected by the tweet content and the emotions it evokes? Experimental results showed a strong association towards content-driven features, such as emotions and the choice of words associated with emotions that were used to compose a tweet or create hashtags. Tweets that contain malicious links were associated with negative emotions, particularly the emotion fear, for their retweet likelihood (virality) and survival (longevity of propagation). Whereas, in tweets that were classified as benign, it was positive sentiment and high arousal emotions such as surprise that were associated with the size and survival of Web links.

List of Abbreviations

API	Application Programming Interface
BAT	Batch file
BIOS	Basic Input/Output System
BN	BayesNet
BPB	BIOS parameter Block
CLSID	Class Identification
CMD	Command File
CNN	Convolution Neural Network
CNN-LSTM	Convolution Neural Network and Long and Short Term Memory
CPU	Central Processing Unit
CSS	Cascading Style Sheets
CTI	Cyber Threat Intelligence
DOM	Document Object Model
DRM	Digital Rights Management
DT	J48 Decision Tree
EURO	The European Football Championships
EXE	Executable File
FIFA	The Fédération Internationale de Football Associations
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IE	Internet Explorer
INF	Setup information file
I-RNN	Identity Recurrent Neural Network

Log	Logarithm
LSASS	Local service Authentication Server
LSTM	Long and Short Term Memory
MLP	Multi-Layer Perceptron
MTF	Master File Table
NB	Naive Bayes
OpCode	Operation Code
OSN	Online Social Network
PDF	Portable Document Format
PIF	Program Information File
RAM	Random Access Memory
REG	Registry File
RNN	Recurrent Neural Network
SCR	System Configuration Repository
SCT	Scitex Continuous Tone File
SHS	Microsoft Scrap File
SVM	Support Vector Machine
TCP/IP	Transmission Control Protocol (TCP) and the Internet Protocol (IP)
URL	Unique Resource Locator
USA	United States of America
WBEM	Web-Based Enterprise Management
WMI	Windows Management Instrumentation
WSC	Windows Script Components
WSF	Windows Script File
WSH	Windows Script Host
XML	eXtensible Markup Language

List of Publications

- Burnap, P., Javed, A., Rana, O.F. and Awan, M.S., 2015, August. Real-time classification of malicious URLs on Twitter using machine activity data. In Advances in Social Networks Analysis and Mining (ASONAM), 2015 IEEE/ACM International Conference on (pp. 970-977). IEEE.
- Javed, A., Burnap, P. and Rana, O., 2018. Prediction of drive-by download attacks on Twitter. Information Processing & Management.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

”True Cyber security is preparing for what’s next, not what was last”— Neil Rerup

1.1 Introduction

In this digital era, it is estimated that 4.02 billion people actively use the Internet [37], and each user, simply by virtue of being online, is vulnerable to a cyber attack. In 2018, 12 billion records [47] containing personal and confidential information fell victim to cyber attacks, and it is estimated that this number will continue to rise, no matter what security measures are taken. A report by Juniper Research Labs estimated that by 2023, 33 billion records containing personal and confidential information would be stolen by cybercriminals [106], resulting in substantial financial losses to both individual and companies. On an average, £3.03 millions were lost, due to data breaches by companies worldwide, and it was observed that the average time taken to detect a data breach was 196 days [158].

Though cyber attacks can be carried out on the Internet in numerous ways, cybercriminals have in the past largely preferred platforms that are widely used [55]. Thus, with the growing popularity of online social networks (OSNs), attacks such as Phishing [137], Distributed Denial of service [121], Cross-site scripting [86], Man in the middle, Trojan attacks [153] and drive-by downloads [102] that deploy viruses, worms [184] are continually being modified so that they can be delivered through OSNs. The introduction of OSNs as another medium through which an attack can be delivered has increased the complexity and the number of cyber attacks. The growing popularity of OSNs has also attracted infamous cyber groups such as Lurk [74] and Patchwork [53] (or Dropping Elephant) and others like them, which are dedicating time and effort to the design of OSN-specific attacks.

Though OSNs have brought millions of users together and created a new venue where users can interact online, they have also posed a serious threat to information security [77]. They present a highly connected network of users who can communicate with each other by sending direct messages or by sharing content by means of posts. This is because the rate at which people react to topics on OSN can escalate very rapidly giving a very short response time in which to counter an attack. For example, 6,000 tweets per second is the average rate at which a tweet is posted on Twitter [145], a rate which can increase to reach 143,199 tweets per second [205].

In the current digital environment, the Windows operating system is the most commonly adopted [88] and, thus, most often targeted by cybercriminals [47]. This makes preserving the security and privacy of individuals' and companies' data a challenging task for security analysts and forensics investigators. Even though substantial amounts are spent on researching ways to fight cyber attacks and developing malware detection models, the rate at which cyber attacks occur increases [106]. As technology changes so do the attacking strategies of cyber criminals and, even though counter-measures are taken, such as educating users [93, 171], installing anti-virus software [139], restricting user privileges in security policy [143], deploying malware detection models based on static/dynamic analysis [100, 210, 137, 135, 31, 125, 65, 103, 108, 114, 11, 116, 149, 22], the use of anti-malware [169] and anti-exploitation tools [170], they continue to pose a threat to cybersecurity. In order to counter cyber attacks, researchers have proposed a systematic method in which data on security threats, threat actors, exploits, malware, vulnerabilities and compromise indicators [182] are collected, assessed and applied. This is referred to as ”Cyber Threat Intelligence” (CTI). The aim of CTI is to help security practitioners to identify the indicators of a cyber attack by analysing the information about attacking strategies and formalising counter-measures to respond in time.

In this study, we looked at cyber threats arising from OSNs, focusing mainly on drive-by download attacks on Twitter. Here a methodological approach in CTI was followed, conducting behavioural analysis on malware by executing malicious code. The study then developed a machine learning model to predict drive-by download attacks on Twitter and used statistical models to uncover the social and content-based factors that exist as part of a tweet containing a malicious links that help it to propagate. However, the model had to surmount such challenges as (i) finding suitable sources to gather this intelligence from; (ii) gathering data from multiple events so that the model generalises beyond a single event; and (iii) sifting the collected data for information that would be useful for predicting such attacks. These challenges were met by (i) gathering data around seven popular events on Twitter; (ii) extracting URLs from Tweets and opening them in a honeypot to identify the occurrence of a drive-by download attack; and (iii) revisiting the Web page in a sandboxed environment to observe the machine's behaviour and identify the requisite set of features; (iv) using the information gathered to develop a model that was capable of predicting a drive-by download attack; and (v) understanding malware propagation behaviour by uncovering the social and content-based factors that aid this propagation.

1.2 Problem Definition

In 2018, Symantec reported in their Internet Threat Security technical report that an average of 953,800 Web-based attacks occurred every day [47], and that 1 in 10 unique resource locators (URLs) points to a malicious Web server. Among the different types of Web-based attacks, the present study focuses on drive-by download attacks, particularly those that use OSNs as the delivery method. In a drive-by download attack, a user's system is infected with malware merely by visiting a compromised Web page. During this visit, the malware executes a malicious java-script code and the vulnerability of the system is exploited to infect the user's system. This is referred to as a ”drive-by download” because the user is directly linked to a malicious file which is downloaded without the user's consent. Such attacks are considered more dangerous than attacks where a user has

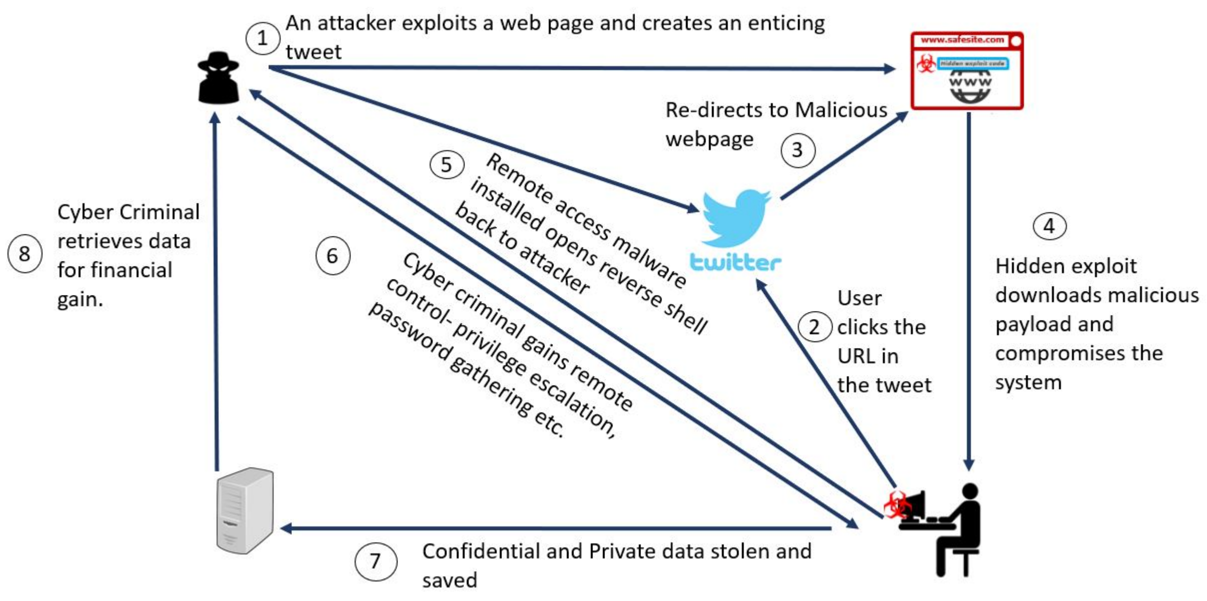

1 An attacker exploits a web page and creates an enticing tweet
www.safesite.com
Hidden exploit code
www
Re-directs to Malicious webpage
3
5 Remote access malware installed opens reverse shell back to attacker
twitter
8 Cyber Criminal retrieves data for financial gain.
6 Cyber criminal gains remote control- privilege escalation, password gathering etc.
2 User clicks the URL in the tweet
4 Hidden exploit downloads malicious payload and compromises the system
7 Confidential and Private data stolen and saved

Figure 1.1: Typical Sequence of events for a drive-by download attack

to click to download a compromised software, which infects the system when installed. Microsoft also acknowledges these attacks as one of its top threats in their security and intelligence report [140]. Also, in a survey conducted by SANS Institute to identify the most frequent methods employed by cybercriminals to launch cyber attacks on organisations, it was shown that drive-by downloads accounted for 48% of attacks by exploiting Web-based vulnerabilities [175].

Twitter is growing in popularity and the 280-character restriction imposed on a tweet automatically shortens a URL and gives cybercriminals the opportunity to create an enticing tweet and obscure the URL that points to a malicious Web server inside it. Figure 1.1 above outlines the typical sequence of events through which an attacker entices a user to open a malicious Web page and admit a malware on his/her system. This gives the cyber criminal access to the system and exposes his/her private and confidential information to theft for monetary gain. In order to counter malware attacks, numerous detection models have been proposed.

These were built using OSN characteristics [12, 183, 227, 126, 144, 43, 192, 30, 187, 19], by doing static [100, 210, 137, 135, 31, 125, 65] or dynamic code analysis [103, 108, 114, 11, 116, 149, 22]. Models built using OSN characteristics help identify those accounts that are posting malicious URLs by identifying similarities with an account already classified as malicious or by comparing URLs against publicly published blacklists of malicious URLs. Even though OSN characteristics help identify accounts exhibiting deviant behaviour, the OSNs features used change with time or can be masked by cybercriminals to avoid being detected [6]. To overcome this limitation, additional investigation of the Web page's code or machine behaviour while visiting the Web page was required to classify the URL as malicious. Typically, a honeypot is set up to observe machine state changes while visiting a Web page [161, 24] or static/dynamic analysis is conducted to investigate a Web page's code.

Models have been built using static code analysis by analysing the code of the malware without actually running it. The aim of these models is to identify known malware signatures based on characteristics such as function calls, headers, etc. extracted from the malware code [232]. One of the major drawbacks to these models is that they are slow and require a considerable amount of human interaction [52], making them unsuitable for handling large volumes of data quickly.

In contrast to static analysis, models built using dynamic code analysis identify the code by executing the malware while observing the changes made by the code. However, these models require many resources and present only a limited viewing period in which to observe the malware, thus restricting their usefulness against the malware in the Polymorph category [152]. Moreover, these models fail if the malware becomes familiar with the execution conditions, such as in a sandboxed environment, because of the strategically placed triggers in the code [123]. These make the malware behave benignly and thus escape detection.

One of the common methods used by security practitioners to protect their network is to deploy honeypots [180]. The aim is to lure the cybercriminal to a system away from the network and allow a cyber attack to take place, to reveal the malicious nature of the application downloaded or Web page visited. The detection is performed on the basis of system activities observed post an attack rather than by dissecting a malware code. By doing so they gain an advantage over static/dynamic analysis by focusing on the effects of the malware infecting the system than analysing the malware itself. The assumption is that no matter how well hidden the code was, upon successful execution, its effect on the user's system would eventually reveal its malicious nature. Thus, detection based on observing machine activities provides an alternative approach where malware code is not analysed, overcoming the need to decrypt, reassemble code or identify triggers in it. Even though detection through honeypot by observing changes in machine activity removes the need to analyse malware, they require high maintenance and if they were not properly deployed they could act as access points to the network [180]. Cybercriminals continue to develop new ways to overcome Twitter's policy and beat the detection models proposed to counter them. There is a requirement to develop detection models by shifting the focus from existing malware code/behaviour features to keep pace with cybercriminals.

To summarise, OSN characteristics were used to identify accounts posting malicious URLs and malware detection models which focused on identifying malware from activities which had already taken place or from existing malware signatures. Most importantly, the focus of the research so far has been on **detecting malware** based on its code or its behaviour alone or alternatively by observing machine activities through a honeypot **after** an attack has taken place. To the best of our knowledge, no work conducted yet has **achieved early-stage prediction** to identify malware **before** an attack, allowing security practitioners to take appropriate countermeasures and stop the malware from infecting a system. This *predictive* security feature, could potentially save millions of pounds that are spent towards repairing a network post a malware attack is detected.

In order to launch a successful drive-by download attack via Twitter, a cybercriminal aims to create a tweet containing a URL, tempting enough that a user clicks on the URL pointing to a malicious Web server, and that it gets retweeted to a larger number of users. The more the tweet gets retweeted, the more it is exposed to a significant number of users, thus, increasing the chances of infecting a larger number of people. Even with a high processing capability of the detection models, there is a good chance that a large number of users would have been exposed to malware by the action of a retweet, even if the user that started spreading it was identified and removed. In order to carry out a successful drive-by download attack via Twitter, the cybercriminal has to ensure that the tweet is 'interesting' enough to catch a user's attention and also contains content that encourages the user to share it within its network. This leads to questions such as: (i) why are certain post shared more than others? and (ii) is virality driven by physiological arousal? Content-based features such as emotions have been used to understand the virality of posts on online social networks [17, 211]. Particularly, content containing negative emotions have a higher chance to be shared on OSN than positive emotions [17]. This tactic of using negative emotions to convey a message to a larger audience has been seen in advertisements as well [16]. In order to investigate if emotion were a contributing factor behind propagation of malicious URLs, a novel study to link emotions and sentiments to malware propagation was conducted. Once the factors are identified, they can be used in developing strategies to stop the malware from spreading. Thus, in order to successfully stop an attack both the source of infection (user and URL used to spread the malware) and the factors aiding its propagation need to be identified.

In order to identify accounts posting malware on OSN, their characteristics need to be observed. What is required is the **anticipation** of the attack and its probability through a series of machine activities taking place, while **identifying** the factors that aid its propagation. This raises the following research question:

Is it possible to construct a predictive model based on machine behaviour and social features that will identify malicious URLs and their propagation behaviour on OSN?

1.3 Contribution

To answer this question, this thesis adopts the detection principle used in honeypots i.e. observing machine state changes to detect malware, but shifts the focus away from fixed and possibly missing features such as API calls, to dynamic behaviour based on the effects of malware on user machines (e.g. CPU, RAM, network use) that are much more difficult to obfuscate and will be present in every piece of malware. This led to the development of a novel detection model capable of classifying an URL into a malicious or benign category with an F-measure of 0.81 during training and 0.71 while testing on an unseen dataset. It then combines social features, that act as indicators to identify accounts exhibiting deviant behaviour with system activities leading to a drive-by download attack to build a novel predictive model capable of identifying a malicious URL on Twitter with 0.99 F-measure (using 10-fold cross-validation) and 0.833 (using an unseen test set) at 1 second into the interaction with the URL. Thus, giving the model the ability to identify a malicious URL based on system activities leading to an attack, providing a basis from which to kill the connection to the server before an attack has completed and proactively blocking and preventing an attack, rather than reacting and repairing at a later date. Though identifying malicious URLs on Twitter is important, it is equally important to understand the factors that aid in its propagation in order to stop it from spreading. A malicious URL propagates on Twitter by the action of retweets. The more interesting the tweet is, the higher the chances are that it would be shared. In this thesis, the contents of the tweets were analysed to understand why certain tweets were retweeted more than others. In doing so it conducts the first study that links content based factors such as emotion and sentiment to malware propagation.

The thesis is composed of several individual chapters, listed below. It should be noted that the chapters in this thesis have been published as peer-reviewed papers:

Paper 1 Burnap, P., Javed, A., Rana, O.F. and Awan, M.S., 2015, August. Real-time classification of malicious URLs on Twitter using machine activity data. In Advances in Social Networks Analysis and Mining (ASONAM), 2015 IEEE/ACM International Conference on (pp. 970-977). IEEE.

Paper 2 Javed, A., Burnap, P. and Rana, O., 2019. Prediction of drive-by download attacks on Twitter. Information Processing & Management, 56(3), pp.1133-1145.

1.3.1 Contributions

C1 To propose a predictive model capable of identifying a malicious URL within seconds, based on machine activities and social features, thus providing new capability to kill the connection and curtail the sequence of malicious actions rather than depending on detection and repair at significant cost and inconvenience.

C2 To understand propagation behaviour by uncovering the social and content features, particularly *emotions* that influence the propagation of tweets containing malicious URLs.

Contributions C1 is relevant to Papers 1 and 2;

1.4 Thesis Structure

The chapters of this thesis have the following content:

Chapter 2 This chapter surveys the work related to the detection and propagation of drive-by download attacks on Twitter, with the aim of identifying the limitations and research gaps in the current work. Open research questions, based on the research gaps, are proposed in the course of the discussion. This chapter contributes to C1 and C2.

Chapter 3 The emergence of Twitter as one of the most popular OSNs for updates on current affairs has in the past attracted cybercriminals, who exploit system vulnerabilities to carry out cyber attacks. This chapter investigates cyber attacks on Twitter and finds evidence of drive-by download attacks around popular sporting events. It contributes to C1 and C2

Chapter 4 In this chapter, a novel detection model is proposed that classifies a URL as malicious or benign based on the system's activities leading to an attack. Thus shifts the focus from deriving features from malware code/behaviour to observing the changes in the user system made by the malware. The associated chapter contributes to C1.

Chapter 5	In this chapter we develop a world-first predictive model for drive-by download attacks on Twitter. Further, we develop evidence that social features contribute to the prediction of drive-by download attacks on Twitter and can adapt to changing techniques of malware delivery. By developing a machine learning model using machine activity data and tweet metadata, the study moves beyond the post-execution classification of such URLs as malicious ones, to predict whether a given URL will be malicious or not. The associated chapter contributes to C1.
Chapter 6	This chapter investigates the propagation of malware through the action of a retweet. We develop evidence that social and content-based factors that form part of a tweet containing a malicious URL aid its propagation. This chapter contributes to C2.
Chapter 7	This chapter reflects on the contributions and highlights the future effect of the thesis on the behaviour analysis of drive-by downloads on Twitter.

Chapter 2

Background

A survey on work related to the detection and propagation of drive-by download attacks on Twitter is conducted in this chapter, with the aim of identifying limitations and research gaps in the current work. Open research questions, based on the research gaps, are proposed and the scope of the thesis is defined in the course of the discussion.

2.1 Introduction

In January 2019, it was estimated that there were around 4.02 billion Internet users, of whom 3.19 billion were active users on OSNs [37], defined as services or Web pages that facilitate social interaction among online users as a means of exchanging information. An OSN consists of active users whose connection to each other allows a relationship to form. This relationship may be based on common interests, friendship or a search for information. Over the years OSNs such as Facebook [66], Instagram [98], Tumblr [203], Pinterest [156], Snapchat [186], YouTube [81], Whatsapp [97], Twitter [207], etc. have grown considerably and engage around 80% of all Internet users [37]. Amongst the popular OSN platforms, we chose Twitter, a micro-blogging online social platform, for our research, since it has around 335 million active users [118] and has become one of the habitual places for getting updates on news of current affairs and entertainment. It also makes 1% of its daily data publicly available for research. Its popularity, unfortunately, continues to attract cyber attacks, such as distributed denial of service [121], cross-site scripting [86], Trojan attacks [153] and drive-by downloads [102] which have become major threats to it.

This research is focused on **drive-by download attacks** on Twitter because they have been acknowledged by Microsoft as one of its top threats in their security and intelligence report [140]. Also, in a survey conducted by SANS Institute to identify the most frequent methods employed by cybercriminals to launch cyber attacks on organisations, it was shown that drive-by downloads accounted for 48% of attacks by exploiting Web-based vulnerabilities [175].

Typically, in a drive-by download attack, a cybercriminal creates a client-side scripting code that targets a Web browser's vulnerability or a Web-browser's plugin vulnerability. The client-side code is either added to the code of the Web page or hosted on a server. When a user visits any of these Web pages they are injected with the malicious code, transferring the client side-script on the user's machine, exploiting the browser's vulnerability and thus compromising the user's machine. Once the machine is compromised, the cybercriminal can steal confidential information such as files and passwords or can even use secondary devices such as a camera or a microphone to spy on the user [20]. These cyber attacks can also be used as an entry point to carry out more widespread attacks such as Ransomware [36] – where a CryptoLocker attack that originated from a drive-by download attack locked down a small city in Washington, USA for four days [122]. Twitter is particularly susceptible to drive-by download attacks, due to the 280-character restriction imposed on tweets [102]. This automatically shortens a unique resource locator (URL), giving cybercriminals the opportunity to obfuscate a URL that points to a malicious Web server.

Today, a company that has an online presence spends on average around £1.89 million per year on security defence against malware and Web-based attacks [1]. Yet 48% of Web-based attacks launched on companies were contributed to drive-by download attacks [175] and 1 in every 13 Web-based requests made points towards malicious Web servers [46]. Despite a security policy in place by Twitter and different malware detection models proposed to counter malware on OSNs cybercriminals continue to develop new ways to overcome them leading to a year by year increase in cyber attacks [106]. Also, it is estimated that by 2023, 33 billion records containing personal and confidential information would be stolen by cybercriminals [106]. Thus, there is a need to change the counter-attacking strategies in order to better defend against malware attacks. This chapter is divided into three sections on (i) definition of general terms used on Twitter, (ii) discussion of work related to the detection of malicious content on OSNs, (iii) identifying the underlying factors behind the propagation of malware.

The second section focusing on the detection of malware is further divided into three subsections, namely, (i) detection based on OSN characteristics,(ii) detection based on static and dynamic analysis (iii) detection based on Twitter's policy. Through these we look into the detection techniques based on content, using OSN user accounts, and URL characteristics; study the detection techniques based on static and dynamic code analysis; and finally, look at Twitter's policy for detecting spam and malware. In the third section we look at work to combat malware propagation on Twitter. In each section the typical design choices, key findings and limitations are discussed, research gaps are identified and research questions are presented.

2.1.1 Definition of Terms Used on Twitter

Tweet: Twitter allows users to share their opinions, make status updates, or discuss news in posts, which it restricts to 280 characters, commonly known as tweets. This restriction automatically applies a URL shortening service to facilitate the posting of a URL in a tweet. This service acts as re-direction service that creates a short URL of 8 characters from the original length of the URL.

Followers: the followers of a user's account are defined by the set of users who choose to receive a tweet when the user has posted it.

Friends: Unlike popular OSNs such as Facebook, relationships between users in Twitter are not bidirectional. Twitter lets users receive tweets from a friend without revealing their own tweets. Friends are a set of users that an account subscribes to, in order to obtain access to status updates.

Mentions: Twitter allows users to be addressed specifically by prefixing @ to their unique screen name in the tweet, which is a way of identifying a user. If a user has a public timeline, mentions appear in this timeline whether or not they are following the user that initially posted the tweet. This allows users on Twitter to quickly identify tweets directed at them (which are broadcast to the sender's followers). A typical example of this would be

TwitterUser[1]*:"@**Superman** keep Doomsday away"*,

where the user, Superman, will receive the tweet on her/his timeline, even though s/he is not following the user.

Retweets: Retweets on Twitter are a form of attribution, where **RT** @*username* denotes that the tweet text originally appeared on another user's profile.

TwitterUser[1]*:"**RT** @Batman: where did you park the #batmobile."*

is an example of a retweet, where RT signifies that a user has retweeted the original tweet. Retweets are used to increase the volume of users who see a tweet.

Hashtags: Hashtags are created by users to give context to their post and make them stand out from others, in order to reach a targeted audience, increase traffic to their post and in turn increase traffic to their profile. Any word preceded by # on Twitter is termed a hashtag; typical examples are #superman, #batman, etc. An example of a tweet using a hashtag is

TwitterUser[1]*:"**#TGIF** movies and popcorn"*

One of the unique advantages that Twitter provides is that it highlights the most talked about topics (using hashtags) by creating a list of trending topics.

[1] Tweets examples given are not taken from any user's account on Twitter, but are merely created as examples for explanation purpose.

2.2 Related Work for Detecting Malicious Content in Online Social Networks

In this section we look not only at work related to drive-by download attacks on Twitter but also at work detecting malicious content more generally on OSNs. Any content on Twitter that has malicious intent, and that may be used to harm a user's computer system, steal personal/confidential information or in any way disrupt a user's activity is categorised as malicious content. In addition to malicious content, spam circulating on Twitter, which is considered a significant issue on OSN, is also considered. The rationale in looking at spam detection is that it has revealed similar detection methods to those combating malicious content. The remainder of this section is divided into three parts, which deal with the detection of the malware problem from three different perspectives, namely detection based on:

1. Social features.
2. Static and Dynamic behaviour.
3. Twitter's own policy to counter malware.

2.2.1 Detecting Malicious Content Based on OSN Accounts and URL Characteristics

Previous research has aimed to identify tweets that were classified as spam or contained a URL pointing to a malicious Web server, based on a tweet's meta-data. This was because it is possible to distinguish a 'normal' user from a 'malicious' user on the basis of account characteristics extracted from the tweet's meta-data, such as the number of followers, number of people s/he follows, posting behaviour, etc. Evidence of this is seen in earlier works, where tweet attributes were used as key features to detect accounts exhibiting abnormal behaviour (e.g. posting spam or malicious URLs). For example, in order to identify the social features of accounts spreading spam, Benevenuto et al. [14] collected 1.8 billion tweets from 54 million Twitter accounts discussing three trending topics on Twitter. They then manually annotated the dataset and identified 355 accounts that were spreading spam and 7,852 accounts that were categorised as regular accounts. Accounts that were labelled as spam were further analysed to identify patterns and features that could be used to build spam detection models. They found around 23 tweet attributes that could be used to flag an account or a tweet as spam. The resultant model was able to detect accounts spreading spam with an accuracy of 70%. However, much time was taken up by the manual annotation of data, making it difficult to add features from new accounts identified as spam to the detection model and thus making the model restrictive.

Grier et al. [83] analysed account behaviour and included blacklists of URLs as one of the features in the malware detection model. The model was built by capturing 200 million tweets and investigating 125 million URLs. These authors used three well known sites (Google Safebrowsing [80], URIBL [209], and Joewein [218]) that published blacklists to identify malicious URLs. By means of account posting, these URLs were analysed to identify features that could be used to detect malicious accounts. The authors also evaluated the effectiveness of blacklists to curtail the propagation of malware on Twitter. However, their work was limited to URLs already identified in blacklists and they had little means of identifying the URLs that were evading detection because they were not included in the list. Similarly, focusing only on the behaviour of Twitter users, Cao and Caverlee [32] proposed a spam detection model based on tweet attributes and users' posting behaviour. The robustness of the spam detection model was based on the assumption that it is difficult to manipulate a user's posting behaviour. For their study, they investigated the posting behaviour of Twitter users by collecting from a publicly accessible Bitly API the URL clicking statistics for those URLs that were embedded in the posts marked as spam. From the collected data a user's behavioural features were extracted. Features included frequency of posting, frequency of the link being clicked, and the post sharing behaviour such as observing if the post was shared in a burst or gradually over time. A model was then constructed using 7 million tweets that contained a URL. In terms of per-

formance the model recorded a precision of 0.86 and a recall of 0.86. However, the model relied on Bitly's application programming interface (API) to provide statistics on users' posting behaviour. The same statistics are generated by Twitter or other URL shorting services but are not easily available; hence, they limit the model's efficiency, depending on Bitly's API.
Building on the idea that the same information could be perceived differently by different people, Shen et al [183] proposed a generalised spammer detection framework based on multiple views of information. When these multiple views of the same information were taken, the authors proposed that the information be incorporated from various perspectives and not be limited to a single view (such as hashtags, URLs, etc.) as in previous models. In order to build the proposed model information about a user was gathered from three different perspectives – text, hashtag and URL. The analysis by these authors showed that different information has the ability to characterise users differently. They assigned different weightings to each view and then combined them to build the classification model. The model was built using information gathered from 10,080 Twitter users' accounts and the final model was able to classify tweets as spam with an F-measure of 0.879. Though the model showed different views of information could be used to detect account that were showing deviant behaviour it was limited to identify account that were spreading spam only.

Alghamdi et al [6] conducted experiments to identify the features that could be used to build a reliable spam detection model. Experiments were conducted on 6,509 tweets with a mix of spam and legitimate content, segregating them on the basis of publicly available blacklists of URLs marked as spam. Features based on the tweet content and user demographic information were used to show a close relationship between them and the tweets that were classified as spam. The authors conducted content analysis to characterise user behaviour by comparing the content of a tweet around a particular topic (based on hashtags) with that of both legitimate and spam containing tweets using the Frequency Pattern [2] and Term Frequency. By doing so, they were able to segregate user features and behaviour into sets of distinct patterns. Then, using Cosine similarity, they compared similarities between the user's profile description and the tweets posted. Experimental results showed that spammers tend to show either lower similarity or high divergence between their profile description and their tweets.

Instead of observing an adversary's behaviour, Lee et al [126] created a trap to attract cybercriminals. Social features were captured from every cybercriminal that fell into the trap to construct a spam detection model that was capable of identifying similar accounts on OSN. In their approach these authors created a legitimate profile and attached a social bot on two online social platforms, Myspace and Twitter. The aim was to attract and identify spammers to this profile. The account was classified as spam on the basis of interactions - for example, through content in direct messages or friend requests that were made. When an account was identified as spam, features such as the number of friends, presence of URL in a post, age of the account, etc., were extracted and were used to build a spam detection model. In order to validate the model, it was tested on two online social platforms. For Twitter an F-measure of 0.992 was observed and for Myspace an F-measure of 0.88 was observed. One of the key features of this model was that it could adapt to a changing environment. This model was implemented by the manual validation of its results and its design was based on the validation results. The model was rebuilt to prevent future mistakes. In a similar approach, Stringhini et al created honey-profiles on the top three OSNs (Twitter, Facebook and Myspace) and recorded the content and interactions made to these profiles in order to identify tweet attributes that could be used to detect accounts which spread spam [192]. Using these honey-profiles the authors were able to identify accounts that spread spam by using the friend requests and messages that they received. Behavioural analysis of these accounts led to the identification of important online social account features that could be used to detect irregularities. Using these features the authors built a machine learning model and tested it against unseen datasets. They were successful in identifying around 15,857 accounts on Twitter that were spreading spam. However, in both cases, the spam detection model performance depended on the honey-profile. This meant that if the profile did not attract enough spammers it would not be able to capture enough attributes from spam accounts and its performance would decline, thus limiting its efficiency.

Exploring the idea of using a blacklist - a list of suspended accounts that had been classified as spreading spam - Thomos et al. [199] analysed content and user account attributes to highlight features that could be used in identifying other accounts spreading spam. The dataset that they used for their analysis contained around 1.1 million suspended accounts and around 1.2 billion posts from them. The aim of the experiment was to come up with a competent blacklist that could be used to remove all accounts that were spreading spam. One of the key findings of this research was uncovering emerging *"spam as a service"* APIs. These APIs were used by cybercriminals to evade detection, to decouple themselves from the process of distributing spam, to save on resources by using APIs, and to launch a very specialised spam attack on OSN. However, the experiment was limited to the data that had already been detected by Twitter as spam and the blacklist that was generated reflected only the accounts that had features similar to accounts published on Twitter's blacklist. The model failed to capture information about accounts that had been evading Twitter's spam detection filter and would always fail to detect these and similar accounts on Twitter. Similarly, Yang et al [226] collected data from around 500,000 Twitter accounts and applied a blacklist filter. URLs were extracted from the results and passed on to a honeypot to identify those who were spreading malicious content. The honeypot results were also validated by a manual check before using the URLs to construct a spam detection model. By studying the characteristics of the OSN accounts they identified ten new detection features, including graph-based, neighbour based, timing-based and automation based features, that were later used to build the model. The model in this research outperformed previous spam detection models by 12%, giving an accuracy of 85%. These authors collaborated with Zhang and Shin [228] to analyse the cybercriminals' ecosystem on Twitter, studying inner and outer social relationships. Inner relationships were defined as accounts associated with cybercriminals which were interconnected, while the outer relationships looked at accounts which supported accounts conducting malicious activities.

Similarly, a feature-based approach was employed by Cresci et al [50], who built a classifier to detect fake accounts that cybercriminals had created to inflate the number of followers. A baseline dataset was created by manually annotating the data and by using public lists of accounts that spread spam. This was followed by running numerous experiments to identify the best spam/bot detection model to date for detecting accounts spreading spam and accounts that are classified as bots. The most effective features were identified by testing those used in previous studies to identify spam/bots. Based on the features found most effective, a machine learning model was built to detect accounts that were spreading spam or were classified as bots. The results showed that the proposed model had an F-measure of 0.991. Shifting focus form social attributes Concone et al. [44] developed a real-time malware discovery system, based on the contents of the post. The proposed model was able to analyse the content of a tweet and generate alerts signifying the presence of malware online. However, the system did not detect the malware but only confirmed the presence of malware circulating on the online social network based on social chatter on Twitter. Chen et al [38] used a Finite State Machine based spam template, demonstrating that a cybercriminal can create 2,000 tweets from a single template. They discovered that such users were using multiple accounts to post spam in a coordinated manner chosen to avoid detection. This is called ”load balancing” - a technique frequently used to prevent denial of service attacks – but in this case, tweets were posted from multiple accounts to prevent detection.

So far, the research has been focused on studying OSN accounts and URL characteristics to identify the tweets or accounts that exhibit deviant behaviour (posting spam or malicious URLs). Table 2.1 gives a list of the most common features used to detect malicious accounts, but are these features enough to identify accounts posting malicious URLs? It is evident from models built using OSN characteristics, that it was necessary to include them to identify deviant accounts but they may not be the only features that a detection model should be composed of. The reason being, Twitter spammers quickly modify their behaviour in order to evade existing spam detection techniques [227] and subsequently OSN features that are required to detect them change with time [6]. Furthermore, this problem is not specific to Twitter and other OSNs may have different user characteristics.

Table 2.1: List of most frequent Meta features used in Previous Research

Sr. No.	Features Most Frequently Used
1	Followers And Friends Ratio [192][12][183]
2	Age Of The Account [227] [126][144][43][183]
3	Number Of Tweets Posted By Account[144][43] [192][30][187]
4	Profile Has A Name [30]
5	Friends Count For An Account [43] [192][187][19]
6	A Default Image Set After 2 Months [187]
7	Account Part Of A Blacklist [83][30][50][199]
8	Account Has A Image [30]
9	The Word Bot In Bio Description[12]
10	Followers Count For An Account [12][43][3][183]
11	An Address Provided [43]
12	Based On Biography [30][187]

Thus, we propose features derived by conducting additional analysis are required to identify malware. We explore this question in Chapter 3.

While evidence exists in related work of OSN users exhibiting malicious behaviour, no significant pattern has been observed to show how cybercriminals were using OSN to spread malware or spam on a large scale. It was not clear if specific people were targeted or whether popular events used to deliver a cyber attack. This leads us to our first research question:

RQ1 *Are popular events that attract millions of users on online social networks exploited by cybercriminals carrying out drive-by download attacks?*

2.2.2 Detecting Malicious Content by Analysing the Static or Dynamic Activity of a Web Page

There are two ways to analyse the activity of a Web page, one by static analysis and the other by dynamic. Static analysis examines the code that loads the Web page, looking for recognisable malicious codes and methods, whereas, dynamic analysis executes the code by interacting with the Web page and observes the behaviour at the endpoint and the local system, also looking for evidence of known malicious activity.

Static Analysis

McGrath and Gupta analysed the anatomy of phishing URLs, studying the patterns of characters and domain length in URLs to develop a filter that would detect phishing URLs [137]. For their experiment they collected around 7,500 phishing URLs from Phishtank [155] and MarkMonitor [136]. Their results showed that phishing URLs tend to be longer than regular ones, they contain the name of the brand that they are targeting, and exploit URL shortening services to evade being detected. Furthermore, they found that a phishing website is *live* for only about 3 days before being detected. Their work gives an insight into the techniques used by cybercriminals to carry out phishing attacks, and the features identified were able to be used to build a malware detection model. A time trigger based malicious code was constructed that behaves maliciously only if a certain condition is fulfilled. But these attacks cannot be detected by lexical analysis of URLs only, and hence they restrict the model's detection capabilities. In a similar approach, an automated classification model was built, based on lexical and host-based features, to detect malicious URLs using statistical models [135]. Experiments were conducted on around 20,000 malicious URLs collected from Phishtank [155] and SpamScatter [8] to train and validate the model in this study. The authors used lexical features such as the length of the URL, types and number of delimiters in the URL, and so on. In terms of host-based features they looked at IP address, WHOIS, the domain name and geographical properties. The model reported an accuracy of 95-99%, however since the model was build on static analysis it failed against cyber attacks that were carried out once a trigger was set off.
Using a filter based approach, Canali et al [31] developed a filter called *Prophiler* that used features derived from URLs and Web page code to determine the likelihood of a drive-by download. The filter used a number of Hypertext Markup Language (HTML), JavaScript, URL and host-based features to determine whether the URL was malicious or not. A HoneyClient, WepaWet [49] was included in the design to further check the URLs that had been classified as benign as a way of checking whether any malicious URL had been missed. One of the key advantages of models built using static code analysis was

that they tended to be fast because once they found a similarity to the malicious code on the Web page they flagged the URL as malicious. The same was seen in Prophiler, which had a classification time of 0.237 seconds per Web page. However one of the limitations of this approach was that it failed to cope with complex behavioural attacks. For example, if a URL has multiple redirects and the final landing page has malicious code, the filter flags the URL as benign due to the features of the first URL and does not consider the features of the landing page.

Similarly, Lin et al [131] proposed a lightweight malware detection filter based on the lexical properties of the URL. The rationale for implementing the URL based filter was to reduce the load on the more complex malware detection model. Their results showed that a 75% load reduction was observed on a malware detection model, while 90% of the URLs were retained. However, the filter considered only the URL characteristics. These may change with time, letting the malware evade detection, and leading to the model becoming outdated.

Focusing only on the embedded JavaScript code in a Web page, Kapravelos et al [109] compared similarities between various JavaScript programs to detect malicious Web pages. The detection model was built using a drive-by download detection tool called *"oracle"*, that can access both malicious and benign Web pages. Once the database of malicious and benign was populated, an abstract syntax tree of the Java code was computed to identify malware with similar code signatures. Based on signature matching, a Web page was classified as either malicious or benign. For their experiment these authors collected around 6.4 million Web pages, out of which around 250 thousand were classified as malicious. In addition, using abstract syntax trees they detected the Web pages that used JavaScript injection, data-dependencies and multiple evasion techniques. However, their model was ineffective towards server-side evasion and attacks that involved code fragmentation.

Looking at browser plugin specific attacks, Laskov and Šrndic [125] used static code analysis to detect malware targeting a browser using JavaScript based on Portable Document Format (PDF) exploits. The proposed model extracted embedded JavaScript from the PDF file. The extracted JavaScript was then compared with a malicious JavaScript code to determine if the PDF was infected with malware or not. The model outperformed WepaWet [49], a Web-based service to detect malware based on JSand, giving a true positive rate of 85%. However, the model was limited to detecting PDF-based exploits only.

Eshete and Venkatakrishnan [65] proposed a malware detection model that determined whether a URL was pointing to a malicious Web server, by checking if the URL was hosting an exploit kit. The detection required a machine learning model that used features derived from the attacking and defending mechanisms of the exploit. The machine learning model was built using 2,629 malicious exploit kits. Of these, 1000 exploit kits were used to train the model, and the remainder to test it. The model gave a true positive rate of 99.9% for training and a rate of 99.4% was observed while testing the model. Even though the model gave a high accuracy of 99.4%, features that were derived were dependent on Web page's code and its URL.

Shifting the focus to system calls made while executing malware, Naval et al [151] proposed a malware detection model immune to system-call injection attack in which irrelevant and independent calls were inserted during the regular execution flow of malware binaries, to avoid being detected. In order to counter these attacks, the proposed model characterised program semantics using the property of asymptotic equipartition. This property allows the model to extract information-rich call sequences that are then used to construct semantically relevant paths. These paths describe the program behaviour and are eventually used to detect malicious binaries. Even though the model gave an accuracy of 95.42% on the sample dataset, it did not account for malicious binaries that were used to carry out shadow attacks. In a shadow attack the system-call sequences are divided and exported to discrete shadow processes. These processes individually appear to be benign but jointly they carry out malicious attacks.

Focusing on the results of a search engine, Invernizzi and Comparetti [100] proposed a malware detection filter based on search engine queries. The tool takes a known malicious Web page as a starting point to generate numerous Web page search queries. The retrieved Web pages are then compared with known malicious Web pages and honeyclients to determine if the retrieved Web page is malicious or not. This filter successfully identified around 230,000 malicious URLs from only 2400 seeds. However, the performance of their model was dependent on the results of the search engine. If the search engine did not retrieve the malicious URL then the Web page could not be identified.

Applying deep learning models to detect malware, Thanbi et al [210] proposed a malware detection model based on deep learning techniques. They collected a sample of around 110,000 malicious and benign URLs from different known resources such as Phishtank [155], MalwareDomains, MalwareDomainList, etc. Subsequently, characters from each URL were encoded and a unique ID was assigned. After the encoding of the characters, the collected dataset was divided into two parts, for training and testing. These were later used to build various deep learning models such as a recurrent neural network (RNN), an identity-recurrent neural network (I-RNN), a long- and short-term memory (LSTM), a convolution neural network (CNN), and a convolution neural network long and short term

Table 2.2: List of most frequent used for Static Analysis in Previous Research

Sr. No.	Features used Static Analysis
1	URL Characteristics [100] [210] [137][135][31]
2	Domain Name[31]
3	JavaScript Code[31][109]
4	Malware Code(exploit kit)[65]
5	HTML[100][31]
6	CSS[31]
7	Browser extension [125]

memory (CNN-LSTM) to identify the best performing model.

Under static code analysis, malware detection models were primarily built using features derived by conducting the lexical analysis of a URL and by analysing the embedded code in a Web page. In the lexical analysis, features such as its length, the words used in forming it, domain names, etc.[100, 210, 137, 135, 31] were derived to build detection models. In analysing the embedded code of a Web page, features were derived by looking at HTML, Cascading Style Sheets (CSS), JavaScript, Octet-Stream (e.g., long byte pattern), command, plain text, compressed content (e.g., *. zip, *.gz, *.tar), eXtensible Markup Language (XML), PDF, and Postscript content [31, 109, 151] to build malware detection models. In addition to lexical analysis of the URL and embedded code analysis of a web page, exploit kits were also analysed to identify various malware signatures [65]. Table 2.2 gives a summary of the features used for static code analysis.

The fundamental principle behind models developed using static analysis are that malware code needed to be examined where the security practitioners were reversing the malware code to get a deeper understanding of the malware [111] and which can be time consuming [96] . This process requires much human interpretation and can be a slow process [52]. Furthermore, these models fail against techniques like encryption and polymorphism that were often used by cybercriminals to obfuscate their code [150]. Thus, models built using static analysis alone might not be sufficient enough to detect malware, and alternative methods to malware detection need to be explored.

Dynamic Analysis

In the following section various malware detection models are presented and their limitations are highlighted in order to identify the research gaps in earlier works. Unlike static models, which focus on the code used in a Web page to identify known malware signatures, dynamic analysis focuses on the run-time behaviour of the code. By observing this behaviour, researchers have proposed them as malicious.

In order to analyse the run-time behaviour of a Web page, Cova et al [49] proposed a malware detection model. This model was developed in two stages. In the first stage, the redirecting of URLs, length of the dynamic code, number of dynamic executions and similar features were used to detect anomalies. In the second part, a custom built browser was used to open the URL and record the process of detecting malicious behaviour. This is achieved by low interaction honeyclients, which imitate the action of a browser based on the events that identify malware. However, one of the key limitations of models based on browser emulators was that they provide limited access to a small number of internet protocol and services [134]. Due to their inability to give the adversary full access to system they cannot capture complex cyber attacks [134].

In a dynamic malware detection analysis, path exploration is a common technique. Here the malware code execution is forced to go down multiple possible conditional branches which prevents the malware from hiding; hence it is easily detected. Building on this technique of detecting malware by analysing the dynamic execution of a code, Moser et al [149] proposed executing the code through multiple paths, and in certain instances even forcing it to go down specific paths. The rationale of these authors was to detect malware that hid their signature and revealed their malicious nature in the execution of the code itself. For their experiment they executed the malware code in an emulator QEMU [13], and recorded the system calls made during its execution. By tainting the inputs given in the program, the system was able to track whenever input values were used to make control decisions about the flow. The program could then be reanalysed using different inputs so as to lead to the exploration of alternative paths of execution. In their evaluation of 308 malware samples from 92 families, the authors found from at least one tainted input source that 229 had acted conditionally, and 172 (55.8%) exhibited additional behaviour under forced path exploration.

Moving from system calls to bit-level operations, Brumley et al [22] proposed a malware detection model called *MINESWEEPER* that could handle more complicated formulae and bit-level operations than the model proposed by Moser et al [149]. *MINESWEEPER* could detect the existence of a trigger based malware, through which those conditions were identified that trigger hidden behaviour, along with the input triggers (e.g., timing, network input, or keyboard input). MINESWEEPER could identify trigger based malware by executing the code both symbolically and concretely. It provides inputs to potential triggers and then executes the code symbolically whereas the part of the code that did not require the trigger is executed concretely. It could determine the input values that triggered the execution down each path, and could then force code execution using different trigger inputs in order to let the code's behaviour be observed. The authors tested their model on four malware samples and identified many conditional branches based on trigger inputs.

The technique of forcing the code to be executed through a chosen path is also seen in a model proposed by Kim et al [114]. This model was designed to systematically explore possible execution paths in order to reveal malicious behaviours. The proposed model J-Force forced path exploration by recording branch outcomes and mutating them into Document Object Model (DOM) elements. When an execution is forced down a certain path, it may crash due to the missing reference to DOM elements, which J-Force overcomes by dynamically creating new DOM elements as and when required so that execution is not interrupted. By doing so, the writers uncovered hidden codes behind the evasion checks in Exploit Kit exploits, and discovered injection vulnerabilities in Chrome browser extensions. Using a path exploration technique, Kolbitsch et al [116] proposed a model, called *ROZZLE*, that sought to identify Web malware by using multi-path JavaScript exploration. *ROZZLE* detects environmentally sensitive Web malware specifically targeting browser configurations. Even though path-exploratory technique helps to identify malware that mask its identity and reveals its malicious nature through certain triggers or inputs, this technique has certain limitations. It made the model fragile because forceful execution of the code could compel the program to access part of a memory that might be corrupt, with the result that the program crashes. Furthermore, a cybercriminal could design a malware using anti-symbolic execution obfuscation [10] or anti-fuzzing [84] techniques that could evade detection by models based on path exploration techniques. Using the same principle of forceful execution of the code but moving away from path exploring techniques, Kang et al [108] proposed a malware detection model that focused on system state changes. Their model analyses changes that were made to the program execution state, such as registry changes, memory changes, etc., in order to identify evasion tactics used by cybercriminals. Once the root point of evasion is identified, dynamic state changes are made to forcefully execute the program. The execution of the program reveals its malicious nature and the program is classified as malicious. This model also has the limitations that the models using path exploration strategies have - forcing the code to be executed may point to corrupt memory and the program may crash. In a similar approach Das et al [56] proposed hardware-enhanced architecture to detect malware at runtime based on system call made during malware execution. The system call were observed for eight different types of malware from a collective sample of 472 malware.

Based on the system call recorded numerous machine learning model were build, among which MLP overperformed other and have a true positve rate of 97.6% and an false positive rate of 1.2%. Similarly, a system call detection engine was created to detect malware by intercepting all system calls made during execution of a program [96].

Jayasinghe et al used the dynamic behaviour of a Web page to detect a drive-by download attack [103]. In order to detect malware, they first executed the embedded JavaScript of a Web page and then created a log file log file was used to create supervised machine learning models. Using the technique of 10-fold cross validation they used three different machine learning algorithms and found that Support Vector Machine (SVM) outperformed NaiveBayes and Decision tree, giving an F-Measure of 0.96. However the model is highly dependent on the successful execution of the JavaScript code; thus, if the cybercriminal adds event triggers, then in their JavaScript the malicious script may not be executed and the malicious Web page will not be detected. Research has also been undertaken to build a machine classifier that uses network activity to detect malware. In one approach Bartos and Sofka looked at network traffic and URL characteristics to build the classifier from data captured in the form of proxy logs generated by 80 international companies [11] to detect malware. They evaluated their classifier by testing it on the companies' network and on real HyperText Transfer Protocol (HTTP) network traffic with approximately 15 million samples, out of which around 43 thousand were detected as malicious. The model gave a precision of 90% and around 67% of the malware that were detected were previously unseen variants. However, it failed to detect malware sent over Hypertext Transfer Protocol Secure (HTTPS). This was because it used features extracted from URLs or flow field which were not available over encrypted HTTPS traffic. It also failed against the dynamic changing strategies of an attacker [11].

Adobe Flash animation has been widely used on Web pages to enrich the multimedia content. However, numerous vulnerabilities have been discovered by means of which it has been used as an entry point for Web-based attacks. Focusing on attacks specific to Adobe Flash, Wressnegge et al [222] proposed an Adobe Flash specific malware detection model called *Gordon*. The model was capable of analysing Flash animation at various levels, starting from the interpreter's loading of the code to its execution. The authors combined structural analysis of the code with path exploration techniques where the code was forced to be executed along certain paths to maximise the coverage of the indicative code regions. By doing so, they revealed the malicious nature of the code. When they applied their model to 26,600 Flash samples that contained around 1900 malicious samples, an accuracy of 95.5% was observed. However, the model was limited to Flash based malware alone and it shares the limitations of the path exploration methods.

Zhang et al [128] developed *Arrow*, a malicious URL detection tool, which correlates URL redirect chains to generate signatures of a drive by download attacks. It used a honeyclient to detect a drive-by download attacks and generated a log file containing an HTTP redirect request from malicious Websites. Arrow was specially designed to detect drive-by download attack, while *Warning Bird* [127] was developed by Lee and Kim to detect spam, drive-by downloads and phishing Web pages based on URL redirects. One of the biggest issues that their model tackled was scalability but they showed that a URL could be classified into malicious or benign in 28.06 (ms) by using 100 concurrent connections for crawling. Overcoming the limitations of malware detection models based on URL redirect, Cao et al [33] proposed a graph based malware detection model, which was built using features selected by studying 100,000 messages from Sina Wiebo, one of China's biggest OSN websites. The writers built their detection model by analysing the length of the re-directional chain, forward comment ratio, forward following ratio, following-follower ratio, etc. These features were chosen because a malicious URL is redirected many times before it lands on a malicious Web page. However, these models relied on the URL redirect and on the assumption that cybercriminals share resources or intermediary links, which may not hold true, since the cost of hosting resources constantly goes down and such resources are taken down regularly by law enforcement.

The fundamental principle behind models built using dynamic analysis was that malware was executed and its run-time behaviour was observed to determine if it was malicious or not. Popular techniques such as URL redirection [127, 128, 49, 11] and path exploration [116, 149, 22] have been used to build previous malware detection models. Table 2.3

Table 2.3: List of most frequently used for Dynamic Analysis in Previous Research

Sr. No.	**Features Used Dynamic Analysis**
1	URL Characteristics and Redirection[127, 128, 49, 11]
2	JavaScript Code[116, 149, 22]
3	Network Activity[11]
4	Proxy Logs[103]
5	Document Object Model [114]
6	Operation Code[103, 108]

highlights the most frequent features used in building them. However, these models have limitations; for instance, the path exploration models are fragile, as they force the malware code to be executed on specific paths and models based on URL redirection, and rely on the assumption that cybercriminals have domains in common. They do not account for the changing environment and do not add features or a component that constantly updates the model in line with the current situation.

To summarise, different techniques have been used to detect malware on OSNs, which are collated in Table 2.4. Models built using OSN characteristics identified accounts that were behaving abnormally. However, in order to be effective, these models needed to be updated at regular intervals, as Twitter spammers quickly modify their behaviour to evade existing spam detection techniques [227]. Though features derived from OSN characteristics were useful in identifying deviant accounts, they cannot be solely relied upon, and there was a need to have more features that were derived by observing malware code/behaviour to detect malicious accounts. In order to do so, the models were built by focusing on the malware itself, conducting either static or dynamic analysis. Detection models built using static analysis analysed the malware code to get a deeper understanding of the malware [111]. However, these models failed against techniques like encryption, polymorphism, etc. that were used to hide the malicious code [150]. In order to overcome the limitation of static code analysis, models were built using dynamic analysis, where the code was executed in a sandboxed environment and *post execution* its behaviour was observed in order to reveal its malicious nature. Typically, the code was forcefully executed along a path by modifying various parameters of the code, to reveal the malicious nature of the program. Even though the behaviour of a program is hard to modify, cybercriminals use techniques like anti-symbolic execution obfuscation [10] or anti-fuzzing [84], strategically placing triggers in code, etc. to avoid detection. In both cases, the fundamental principle was analysing the malware code, whether it was analysed without execution (statically) or after execution (dynamically).

In order to shift the focus from malware code analysis to observing the effects of malware on the user system, security practitioners have used honeypots to detect malware [180]. The fundamental principle behind a honeypot is that it mimics a user by providing computing resources that could be scanned allowing it to interact with a cybercriminal. By doing so it allows a cybercriminal to execute a cyber-attack successfully. By deploying a honeypot a security practitioner can gather evidence from the log file that was created to document the attacks, and also monitor system state changes - giving them an insight into the vulnerabilities exploited and files targeted by the adversary. Honeypots can be

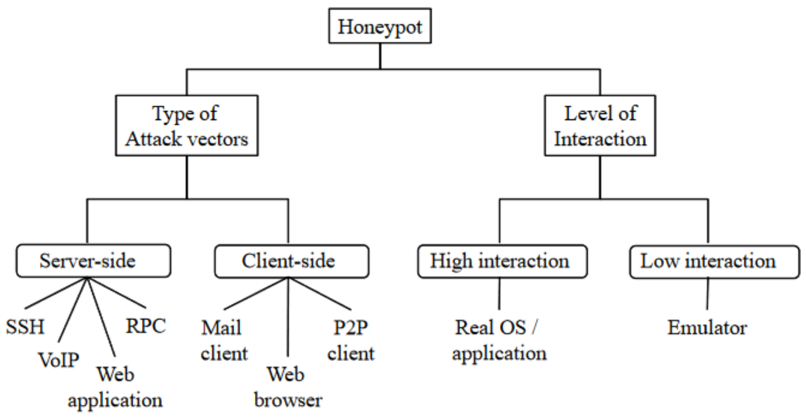


Figure 2.1: Taxonomy of Honeypot [4]

categorised based on the attack vector and the level of interactions [4]. The attack vectors are further divided into client and server-side attacks, whereas the level of interaction is divided into high and low level (see figure 2.1). As the focus of this research was drive-by download attacks that target client machines, only client-side honeypots will be considered for discussion. In terms of interaction, a low interaction honeypot performed static analysis on a Web page code, looking for evidence of malicious scripts [49, 109]. These were typical emulators but gave limited access to the adversary [134]. Thus, due to

Table 2.4: Summary of Existing work on malware detection on OSN

Year	**Author**	**Method**	**Forced Path Exploration**	**State Changes**	**Lexical Analysis of URL**	**Java Script Code**	**Trigger Based**	**Social Features**	**Blacklist**	**Relationship based**	**Plugin Based**	**Network Traffic**	**Exploit Kit**	**System Calls**
2007	Moser et al	DC	✓			✓								
2007	Idika,N and Mathur, A	DC				✓								✓
2008	Brumley et al	DC	✓			✓	✓							
2009	McGrath and Gupta	SC			✓									
2009	Ma et al	SC			✓									
2009	Kang et al	DC	✓	✓		✓								
2010	Lee et al	OSN-AC			✓			✓		✓				
2010	Stringhini	OSN-AC						✓						
2010	Benevenuto et al	OSN-AC						✓		✓				
2010	Grier et al.	OSN-AC						✓						
2010	Cova et al	DC			✓	✓								
2011	Thomas et al	OSN-AC						✓						
2011	Yang et al	OSN-AC						✓		✓				
2011	Canali et al	SC			✓	✓								
2011	Laskov and Šrndic	SC				✓					✓			
2011	Zhang et al.	DC			✓									
2012	Yang et al	OSN-AC						✓		✓				
2012	Invernizzi and Comparetti	SC				✓								
2012	Kolbitsch et al	DC	✓			✓								
2013	Kapravelos et al	SC				✓					✓			
2013	Lin et al	SC			✓									
2014	Eshete and Venkatakrishnan	SC				✓							✓	
2014	Jayasinghe et al	DC	✓			✓								
2015	Cheng Cao and James Caverlee	OSN-AC						✓		✓				
2015	Cresci et al	OSN-AC			✓			✓		✓				
2015	Naval et al	SC	✓			✓								
2015	Das et al	DC												✓
2016	Chen et al	OSN-AC						✓						
2016	Bartos and Sofka	DC			✓							✓		
2016	Wressnegge et al	DC				✓					✓			
2017	Shen et al	OSN-AC			✓			✓		✓				
2017	Alghamdi et al	OSN-AC						✓						
2017	Concone et al	OSN-AC						✓						
2017	Kim et al	DC	✓			✓								
2018	Thambi et al	SC			✓									
DC- Dynamic Code, SC- Static Code ,OSN-AC -OSN Account Characteristics														

their inability to give the adversary full access to the system they cannot capture complex cyber attacks [134]. Whereas, high interaction honeypots use dynamic analysis of the interaction behaviour between the Web page and the client system, looking for evidence of malicious actions [214, 219]. Capture HPC [27], Threat analyser [200], and Winpooch [15] are typical example of high interaction honeypots that are capable of detecting malicious attacks. Where Capture HPC has been favoured over others in the detection of drive-by download attacks [5, 160].

Honeypots gain an advantage over models built by analysing malware code because they focus on the effect of the malware on the system. The assumption is that no matter how well hidden the code is, upon successful execution, its effect on the user's system would eventually reveal its malicious nature. However, to the best of our knowledge, none of the models presented so far has adopted the principle of a honeypot that observes the effects of malware on the system to detect malware on OSN. This leads us to the following research questions.

RQ2 *Is it possible to construct a model to detect drive-by download attacks that is easily reproducible and is independent of the malware code signature, by using system-based activity extracted from a honeypot?*

RQ3 *Which machine learning models are well suited for detecting drive-by download attacks, based on machine activity?*

2.2.3 Twitter Spam and Malware Detection Policy

On an average around 10 million Web attacks per month were reported in 2018 [195]. Twitter's popularity continues to attract cybercriminals to carry out cyber attacks. Cyber attacks on Twitter such as distributed denial of service [121], cross-site scripting [86], Trojan attacks [153] and drive-by downloads [102] continue to be major threats. Since Twitter is being used to propagate and carry out drive-by download attacks [102, 103], it too has taken measures to identify tweets containing malicious URLs and users who are spreading malware. However, for security reasons Twitter has safeguarded the details of the malware detection model and has published certain rules as part of its terms and condition and policy [206] to detect fraudulent accounts or malicious content on Twitter. These broad rules forbid a user from:

- Stealing or copying the profile bio of other users.
- Accessing searches on Twitter by means other than provided by Twitter.
- Using Twitter to send fraudulent or misleading information by modifying TCP/IP headers or any part of a header message.
- Intentionally misleading people by adding fraudulent information, including a false geo-location.
- Aggressively start following people.
- Posting either tweets or direct messages that contain only links.
- Posting of duplicate or similar content, replies or user mentions across multiple accounts.
- Accounts that post multiple updates in order to manipulate a trending topic or to divert the traffic to unrelated accounts, products or services.
- Aggressively adding users to a list.
- Sending bulk replies.
- To randomly or aggressively engage in tweeted replies that aim to divert the traffic to an account, product or service.
- To use or promote third-party services or apps that claim to get you more followers, retweets, or likes.

The above rules are among the terms and conditions set by Twitter and used to identify accounts that are spreading spam or malicious content. Though the details of Twitter's actual spam/malicious tweet detection model have not been made public for security reasons, it can be inferred from other defined rules that Twitter analyses content, posting behaviour and user behaviour (the adding of friends/followers) to identify malicious users [206]. Cybercriminals have found ways to evade detection and continue to use Twitter to spread spam and carry out cyber attacks. Furthermore, due to its large active user base Twitter is constantly probed for vulnerabilities [112], which if found are used to further attacks leading to financial losses, thus limiting the efficiency of the detection model.

To curtail malicious attacks, research has focused on building malware detection models based on analysing OSN characteristics, static code and dynamic code, or by observing machine activities through a honeypot **after** an attack has taken place. Most importantly, the focus of the research so far has been on **detecting malware** on OSNs. Honeypot based detection gains an advantage over detection models based on code because they give an insight into the techniques used and vulnerabilities exploited during a drive-by download attack. However, a honeypot needs to execute the malware in a sandboxed environment and observe it for a user-defined period, before making a classification decision. This means the attack has already completed. If an alarm was raised **before** the malicious activity was complete, the connection to the malicious server could potentially be killed, and thus reduce the exposure of the network to additional risk. To the best of our knowledge, no work conducted to date has achieved **early-stage prediction** to identify malware **before** an attack is complete, allowing security practitioners to take appropriate countermeasures and stop the malware from infecting a system. Limitations observed in the work published to date include:

- Detection models are specific to one type of vulnerability (e.g. Adobe Flash, or Adobe PDF plugin).
- Detection Models are dependent on malicious code signatures (static analysis) or code behaviour (dynamic analysis) at run-time and these models fail against code obfuscating techniques such as encryption, polymorphism, etc.
- Detection models proposed are incapable of detecting malware based on early-stage interaction, meaning machine activities lead to an attack.

These give rise to the following research questions :

RQ4 *Is it possible to predict drive-by download attacks on Twitter by observing machine activities?*

RQ5 *Do social features that have been shown to help with identifying abnormal accounts (posting spam/malicious URL), contribute to the prediction of drive-by download attacks on Twitter?*

2.3 Related Work for Malware Propagation on Twitter

In the previous section we analysed the research developed to detect malware on OSN. No matter how accurate the detection model was, the features used to detect it suggest there is a good chance that a malware would evade detection because (i) Twitter spammers quickly modify their behaviour to evade existing spam detection techniques [227], and (ii) these models fail against techniques like anti-symbolic execution obfuscation [10], anti-fuzzing [84], encryption [150], polymorphism [150], etc. that were used to hide the malicious code or its behaviour. If undetected or not removed from the network, malware in an online social platform which has millions of active users has potential to infect thousands of users at great speed. In this section we review some techniques used to curb malware propagation.

2.3.1 Curbing of Malware Propagation on Traditional Networks

Researchers have studied malware propagation using a range of different methods. Malware propagation models fall into two categories, namely, control theoretical models and epidemiology models. The aim of control theoretical models is to detect and contain malware, whereas epidemiological models count the number of compromised hosts and evaluate the distribution of malware from these compromised hosts. While both are equally important, propagation models based on epidemiological theory have been favoured over other models because they give us a sense of the damage that a malware can cause to a network.

Zou et al proposed a model based on epidemiology to detect the propagation of worms on the Internet [235]. The detection model favoured by these authors focused more on identifying network trends than on looking at bursts in the network. The underlying assumption for their model was that at the early infection stage a malware propagates rapidly with exponential growth. They used the *Kalman filter* estimation model as one of the key components to identify early trends and in order to identify outbreaks of malware in the network. The filter was strategically placed to monitor the network and was activated when a surge of illegitimate scan activities was encountered. If the infection rate estimated by the model stabilised and oscillated slightly around a constant positive value, the model flagged the presence of malware. The rationale was that non-malware noise will not grow exponentially. The model did not consider a scenario where multiple malware might spread at the same time, or a single malware used multiple vulnerabilities to propagate itself.

Similarly, Yu et al proposed a two-layer malware propagation model for large networks based on epidemiological principles [229]. Using the two-layer approach, these authors used the upper layer to focus on a large scale network, while the lower layer focused on the hosts of a given network. To validate their model they used Android based [105] malware and the Conficker worm [184], an Internet based Botnet. The researchers observed that during the early stages malware follows exponential distribution, in late stages of propagation they follow power law distribution with a short exponential tail and in their final propagation they follow power law distribution. However, one of the limitations observed was that they did not consider multiple malware propagating at the same time, which in real life is a more likely scenario.

Ganesh et al combined epidemiology and graph theory to understand malware propagation on networks [76]. They observed that if the spectral radius of the graph of the network studied is higher than the ratio of cure to infection, then the average epidemic lifetime is of order *log n*, where n is the number of nodes. However, if the same ratio is greater than the isoperimetric constant of the graph then the average epidemic lifetime is represented by e^{n^a}, where a is a positive constant. Liu et al combined an epidemic model with transmission theory in order to observe malware propagation in wireless ad hoc networks [132]. For their experiments they used two different malware propagation schemes in two different network modes. They studied features such as the mobility of nodes, the number of nodes in the network and the transmission range for each node and formulated mitigation strategies. The study gives an insight into the ways that malware propagates on wireless ad hoc networks and describes the highlighted features that can be used to observe malware propagation. Others that used epidemic models to understand malware propagation in ad hoc wireless networks used techniques such as removing infected nodes [234], applying an immunisation defence technique [213] or studying the topology of the network [233] in order to identify propagation factors. The epidemic models were also used to understand malware propagation using Bluetooth as the propagation medium, where models were built using features derived from Bluetooth transmission protocols [75, 34, 193, 224] emphasising short-range and short communication protocols.

A considerable amount of research has sought to understand malware propagation using epidemiological models in traditional networks [164, 154], including wireless and Bluetooth. However, while building these epidemic models to understand malware propagation, the assumption was that the malware is propagated using traditional local area networks, ad hoc wireless networks or more recent Bluetooth technology. The types of malware considered here are self replicating and do not require much human interaction. However, malware on OSN, particularly drive-by downloads require human interactions (visits to the Web page) for infection. The malware after infection can be self-replicating when downloaded to the user machine and may use traditional networks such as wireless, Bluetooth or local area network to further infect other users. However, in order to give a more realistic view, features derived from OSN (the point where the attack originated) must be incorporated into the propagation model to make it most effective.

2.3.2 Curbing of Malware Propagation on Online Social Networks

Earlier research on malware propagation related to social networks focused on the communication medium such as a mobile device [70], Bluetooth[41, 75, 34, 193, 224], wireless networks [132] or emails[220]. However, with the emergence of online social networks, new techniques have been developed to exploit the social relationship between users in order to propagate malware. To counter such propagation, researchers wanting to know more about it have incorporated techniques such as basing design on epidemiology or on user relationships.

Fleizach et al [70] proposed a model to observe malware propagation in a mobile network, where the model would create a social network topology using the contacts saved on the user's device to evaluate the speed and severity of random contact worms. Even with the mobile network constraints and limited address books that the malware exploits, the experimental results in this study showed that aggressive malware are capable of launching distributed denial of service attacks and of preventing users from using services such as Voice over IP and multi-media services. Their propagation model gives an insight into the speed of propagation and help to formulate defensive rules to contain the malware infection. However, one of the key elements that was missing in the propagation model was user behaviour that could influence malware propagation. Even though the social networks created by using contact address books may share traits with OSN graphs, they differ regarding the amount of data generated and the amount of time a user spends on the network. These abundant data on a user's behaviour open doors to understanding malware propagation using various techniques that incorporate social behaviour defined by user relationships.

Having compared virus propagation through emails with the propagation of viruses using messages exchanged on Facebook, Fan et al went on to propose malware propagation models based on the application network of Facebook [67]. They investigated two malware propagation strategies, one where the cybercriminal would develop applications designed to carry out attacks or contain vulnerabilities for subsequent exploitation; and the other where a malware is distributed by means of direct messages to users. Their experimental results regarding malicious application showed that even if the malicious application is less popular on the OSN platform, it still has the potential to spread rapidly. Cybercriminals exploit the trust relationships between users whereby the installation by a user of a malicious application may installs the same malicious software in the computers, etc. of her/his friends who trust any application their friends install. The propagation model based on malware that is spread through direct messages showed that the propagation is directly proportional to the amount of time that a user spends on OSN. Furthermore, the chances of spreading increase since users are more likely to click on the messages and hence on re-directional links that they receive, which are used to carry out attacks.

Experimenting with user relationships, Sun et al proposed a human behaviour model based on game theory to describe the propagation of network worms on social networks [194]. For their experiment the writers created an artificial network and looked at malware attacks that were carried out using direct messages. They created two malware propagation models, one based on user behaviour, which is predicted in terms of whether the user will succumb to opening a message containing malware, and the other, which characterises the dynamics of network users in order to identify those who are most vulnerable to malware attacks. Even though the model captures user behaviour it is yet to be tested on a real sample where unexpected obstacles may be met, or the change in malware behaviour which may affect the propagation model and thus reduce its efficiency. Sanzgriri et al successfully applied epidemiology theory to understand malware propagation on Twitter and showed that even a low degree of connectivity and the probability of user clicking links could cause wide malware dissemination [177]. Similarly, Giri et al. [107] proposed a mathematical model based on epidemic theory to understand malware propagation on Twitter. They conducted multiple simulations on NetLogo, and their results showed that even with a small number of infected users with low connectivity, malware on Twitter has the potential to infect a large number of users. One of the drawbacks of this models is that it lacks scalability and accuracy due to the assumptions made. To overcome the drawbacks of the epidemic models, Wang et al [216] presented a discrete-time absorbing Markov process to characterise virus propagation. The proposed model was capable of evaluating virus lifetimes in large networks.

Yan et al [225] analysed user activity patterns and OSN structure to narrow down the characteristics of malware propagation in OSNs. For their experiments they used real-world locations based on OSN data and conducted analysis from the perspectives of user friendship and activity. Furthermore, they conducted trace driven simulation to observe the initial infection impact, user click probability, social structures and user activity patterns on malware propagation. However, their research assumed that users were active only if they were engaging in certain activities, such as location based check ins, photo updating or posting. Furthermore, they assumed that each user has the same probability of clicking on a malicious URL, which may not be the case in real life, some necessarily being more educated than others. Focusing on user behaviour on OSN, Wang et al proposed a malware propagation model based on user behaviour, mainly looking at user mobility and temporal message processing [215]. One of the key features that they introduced was user mobility as one of the main factors to estimate malware propagation. They incorporated the idea that a user can be mobile and hence the infection rate can change endlessly. In many OSNs, a message recall function was introduced to tackle the malware propagation problem. This feature allowed a user to delete any post that contains a malicious link so that it is no longer accessible to other connected users. However, users were still in the network that had been infected before the message was recalled and they might continue to spread the malware to their connected users. In view of this message recall mechanism, Chen et al proposed a model based on epidemic theory to measure the propagation of infections in a message-recallable OSN [40].

2.3.3 Propagation of User Posts Based on Content

Gupta et al.'s [85] work gave an in-depth characterisation of factors that influence the virality of malicious post or fake content on Twitter. For their analysis, they gathered data around the Boston Marathon bombing incident and found that 29% of content that became viral after the incident was either rumours or was fake. They discovered that a large number of malicious or dubious account were created after the event to propagate fake news. Furthermore, they used statistical models to demonstrate that the overall impact of users who propagate rumours could be used to estimate the growth of fake content. Sentiment analysis is concerned with detecting positive, negative or neutral content in written text, whereas emotion analysis is concerned with detecting discrete emotions (e.g. anger, fear, joy, and so on). Current research has focused on the detection of spam [212] or predicting malware based on machine activities [24, 102]. Wang et al. proposed a spam detection model that uses sentiment as one of the features in detection on Twitter [212]. They showed that by using only four features, one could achieve results that were satisfactory compared to previous tools. Similarly, Hu et al. used a network topology to detect spam showing that the performance of the model increased by the addition of sentiment data [95]. Focusing on content only, Berger et al. studied emotions expressed in a tweet to identify a relationship between retweeting and emotions [17]. They found that content that evokes high arousal, such as positive emotion (awe) or negative emotion (anger or anxiety) has a higher probability of propagation than 'deactivating emotions' such as sadness. In a similar approach, Vosoughi et al used emotions to explain the propagation of news on social media [211]. They found that news that was false and reflected fear, disgust, and surprise was more likely to be retweeted than actual stories that reflected anticipation, sadness, joy and trust. This leads us to the following research questions:

RQ6 *Do social features aid the propagation of posts containing malicious URLs?*

RQ7 *Do features derived from emotive content, such as fear or joy, help in the propagation of malware?*

2.4 Conclusion

The main contribution of the chapter is the survey of existing techniques and methods used to detect malware on OSNs. Previous work was discussed and research gaps were identified, leading to the development of seven research questions. These are summarised within the remaining chapter descriptions below.

RQ 1 Chapter 3 contributes to addressing RQ1, where evidence of consistent use of Twitter to deliver drive-by download attacks has been found, particularly around popular events that attract millions of users. The evidence was gathered by opening captured URLs in a sandboxed environment and based on system changes observed the URL was classified as benign or malicious.

RQ 2, 3 Chapter 4 contributes towards RQ2 and RQ3, where a novel drive-by download detection model is proposed. The model adopts the fundamental principle on which honeypots are built by observing system activity, and uses this as input to machine learning algorithms that use the effect of malware on a user's system to classify URLs as malicious or benign.

RQ 4, 5 Chapter 5 contributes towards RQ4, and RQ5, where a novel drive-by download prediction model is proposed, which identifies malicious URLs within seconds of the interaction starting based on machine activities. This provide a basis from which to kill the connection to the server before an attack has completed and proactively blocking and preventing an attack, rather than reacting and repairing at a later date.

RQ 6 and 7 In Chapter 6 we present the first study to link emotions and sentiment to the propagation of tweets containing malicious URLs on Twitter. In earlier studies it has been established that emotions could be transferred between people through OSNs [89] and they can affect the posting behaviour of users [120]. On Twitter, a drive-by download attack is propagated by the action of retweeting the tweet containing the link to a malicious Web server. The higher the number of retweets, the larger the number of people exposed to it. Experimental results of the study showed emotions, particularly negative emotions, were associated with retweet likelihood (virality) and its survival.

Chapter 3

Popular Events as a Medium to Deliver Drive-by Downloads Attacks

Are popular events that attract millions of users on online social network being exploited to carry out drive-by download attacks by cybercriminals? The emergence of Twitter as one of the most popular OSNs for updates on current affairs has in the past attracted cybercriminals, who exploit system vulnerabilities to carry out cyber attacks. This chapter investigates cyber attacks on Twitter and finds evidence of drive-by download attacks around popular sporting events.

3.1 Introduction

Over the years online social networking platforms have attracted billion of Internet users. These numbers have grown tremendously from 0.97 billion recorded in 2010 to 3.19 billion users recorded in January 2019 [37]. Their popularity has tempted cybercriminals to carry out cyber attacks on the computer systems of the network users. Numerous attacks from social media are recorded every year (see Figure 3.1) compelling security experts to teach people about the dangers related to OSN. By observing the trust exhibited between a user and others in their online social network, security experts have suggested that a user is more likely to fall for a malicious attack through an online social platform than a malware attached through email [73]. Exploiting the trust between the users on online social network platforms, cybercriminals have used various techniques to compromise the confidentiality, integrity and availability of systems. Attackers have used techniques

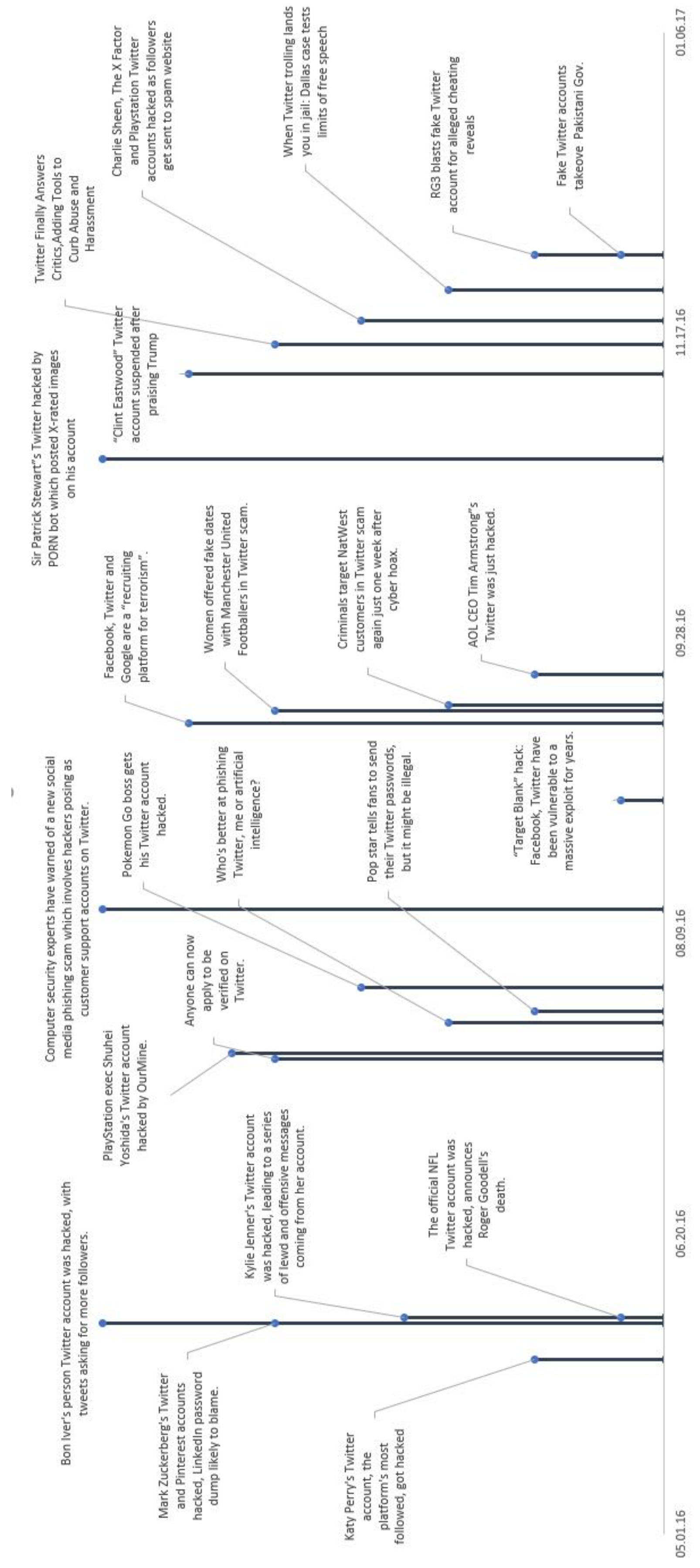


Figure 3.1: Time line of cyber attacks on Online Social Platform

Table 3.1: Top ten cyber attacks carried out using online social networks

Ten worst social media attack				
Sr. No.	**Target**	**Type of Technique**	**Time of Attack**	**Description of Attack**
1	10k US Government Employees Spearphished with Malware-Laced Posts [28]	Targeted Phishing/Malware, Fraudulent Accounts	Early 2017	Custom phishing messages via social media were sent to targeted government employees Each post contained a link laced with malware enabling the attacker to access and control the victim's device.
2	Fake Social Media Persona Sends Malware to Employees Via Social Media [48]	Trojan APP	Jul-17	Attackers created an incredibly compelling fake profile to connect with corporate employees. Later they disseminated a Remote Access Trojan (RAT), called PupyRAT, via these social media honeypot accounts to hijack the controls of victims' devices.
3	3rd Party App Leads to Hundreds of High-Profile Account Compromises [173]	Account Takeover	Mar-17	Vulnerability exploited in 3rd party app called Twitter Counter enabling the attacked to take over the users' account.
4	HAMMERTOSS Malware Uses Social Media for Command & Control [99]	Malware/Data Ex-filtration	Jul-15	Malware automatically searched social networks for commands posted by attacker profiles, allowing cybercriminals to control the malware via social media posts.
5	Financial Crime Runs Rampant on Social Networks [45]	Fraud & Scams	Aug-16	Instagram was used by scammers to prey on the followers of verified banks with fraudulent financial services offerings, including card cracking and money flipping
6	AP's Social Accounts Hijacked, $136 Billion Lost in Stock Market Value [69]	Account Takeover	Apr-13	Attackers compromised the account of the Associated Press, posting fake breaking news that bombs had gone off in the White House. The Dow subsequently dropped 150 points before rebounding; an economic value of $136 billion
7	LinkedIn Hacked, Exposing 117 Million Credentials [72]	Data Breach, Account Takeover	May-16	The 2016 LinkedIn data dump was the 7th largest in history by sheer number of compromised credentials,
8	Enigma's Slack and website hacked, a half million in Ether coin stolen [172]	Fraud & Scams, Impersonation, Account Takeover	Aug-17	The Slack community channel of Enigma, a start-up exchange for the cryptocurrency Ethereum, was breached by attackers. The attackers impersonated the executives of the company and instructed the community members to send their Ethereum coin to a specific coin wallet, stealing roughly a half million worth of the cryptocurrency
9	Phishing Direct Message Sent to Customers from Compromised Brand Account [119]	Account Takeover, Targeted Phishing & Malware	Sep-11	Australian bank's account was taken over by cybercriminals that requested users to give sensitive financial details over direct messages.
10	Vevo Hacked Via Targeted LinkedIn Phishing Attack, 3.12TB Ex-filtrated [29]	Targeted Phishing & Malware	Sep-17	Streaming service Vevo suffered a breach when one of its employees was phished via LinkedIn. Hackers were able to obtain and publicly release 3.12TB worth of the company's sensitive internal data

such as Phishing, Trojan malware, taking over users' accounts, exploiting vulnerabilities in mobile applications and luring users to malicious websites.

Table 3.1 lists the ten most harmful cyber attacks that were carried out using OSNs. In each attack, the cybercriminals have taken advantage of the large number of active online users on the online social network being used. These attacks varied from stealing financial data [45] to targeting government employees in order to steal confidential information [28]. Furthermore, techniques such as fake offers (for example, like-jacking, fake plugins or app tricks and manual video sharing) are used to trick users into installing malware. A *fake offer* is one where a user is lured to join a fake group or take part in a fake event to earn gift cards. *Like jacking* tricks the user into redirecting other users to a malicious web-page whenever they like a page. *Fake plugins* and *fake apps* trick the user into installing malicious software, which will later be used to steal confidential information. In addition to these, posts containing links to malicious web servers were manually shared by creating the post interesting enough for the users to share it and enticing enough for the users to click on it.

In order to expose a large number of users on OSNs, cybercriminals have either overtaken an account that had a significantly large following, such as that of celebrities [221], either through data breaches [72], or through the distributing malicious applications [48] or through content sharing that contained links to malicious web servers. By using each technique the adversary has aimed to maximise exposure, so as to infect a maximum number of people to malware. However, the drawback of these approaches has been that they get detected quickly and thus get stopped, for example the tone of communication used by cybercriminals lets the fans of the celebrities quickly recognise that the accounts have been taken over and thus report the accounts [221].

Popular global events that attract and engage millions of people to one topic are an alternate to accounts of famous people that have the potential to expose malware to a broader audience. Thus giving cybercriminal an alternate medium to launch an attack that could potentially expose millions of users to malware. By overtaking an account a cybercriminal would quickly gain access to the number of people following the user and any post shared from it would potentially be seen by all the followers of that account. Whereas, if the cybercriminal was to spread the malware using popular events, they would have to piggyback on the popularity of the event by creating a post (using event-specific hashtags) related to the event, enticing enough to attract attention by means of being shared among users. Evidence of accounts of famous people being taken over to spread malware have been found [221]. However, to the best of our knowledge, no association between popular events that gather millions of users and drive-by download attacks has been seen.

In order to gather evidence of drive-by download attacks being carried out using popular events on Twitter, data from seven diverse popular global events were collected, from which URLs were extracted and further analysed to detect malicious behaviour. Typically, there are two ways through which a drive-by download attack can be detected, (i) by the construction of detection models based on malware code/behaviour, and (ii) by observing the effect of malware on user's system by opening the web page inside a honeypot. These are discussed in detail in Chapter 2. However, the models constructed by analysing malware code/behaviour often fail against code/behaviour obfuscating techniques such as anti-symbolic execution obfuscation [10], anti-fuzzing [84], encryption [150], and polymorphism [150]. To overcome these, the latter approach was chosen to detect drive-by download attacks on Twitter. The underlying assumption is that no matter how well the code was hidden, once executed it would infect the system and so would reveal its malicious nature.

3.1.1 Contribution

While all OSN are vulnerable to drive-by download attacks, due to its 280 character restriction, Twitter is particularly susceptible to it. This chapter aims to further motivate the development of cybersecurity methods to support safer OSNs by answering the following research question

RQ1 *Are popular events that attract millions of users on online social networks exploited by cybercriminals carrying out drive-by download attacks?*

The research carried out in this chapter contributes to the growing literature on countering malware on OSN as follows:

1. It demonstrates how Twitter is vulnerable to drive-by download attack.
2. It shows how malware (drive-by download attacks) can be detected.
3. It lays the groundwork for a predictive online malware detection system by identifying key indicators from the characteristics of tweets or Twitter accounts.

3.2 Experimental Setup

In order to gather evidence that popular events on Twitter were being used to deliver drive-by download attacks, the experiment is divided into two main phases:

1. Data Collection and Annotation— In this phase tweets containing URLs were captured from Twitter around a global event using event specific hashtags.
2. Identification of Malicious URLs— In this phase, the captured URLs were connected to via a honeypot to detect the presence of a drive-by download attack.

3.2.1 Data Collection

For phase one, a Python-based data collection script using Tweepy [166] was created. It connected to Twitter using its programmatically available streaming API. For our experimental purposes, we chose sporting events around which tweets were collected. Sporting events were specifically favoured over any other trending events such as the American presidential election, because they were reported to generate a large volume of Twitter traffic. For example, the Copa America in 2015 alone recorded 14 billion impressions [124] and the 2016 Rio Olympics was the top subject that year - more popular than the United States of America (U.S.A) presidential election [118]. In 3 years, we collected data from seven sporting events:

1. The Fédération Internationale de Football Associations (FIFA) World Cup of 2014
2. The American Football Superbowl 2015
3. The Cricket World Cup 2015
4. The Rugby World Cup 2015
5. The American Football Superbowl 2016

6. The European Football Championships (EURO) 2016
7. The Olympics 2016

Table 3.2: Description of Tweets collected during sporting event

Sporting Event	Year	Location	Hashtag Used	Number of Tweets Captured	Malicious Tweets Identified	Number of Unique Tweets
Federation Internationale de Football Association (FIFA) World Cup	2014	Brazil	*#FIFA2014*	95,000	46,481	2,039
Circket World Cup	2015	Australia & New Zealand	*#CWC15*	7,961	4,238	891
Rugby World Cup	2015	United Kingdom	*#RWC2015*	127,393	3,836	627
SuperBowl 2015	2015	USA	*#SB50, #SuperBowlSunday #superbowlXLIX*	122,542	2,293	1,230
European Football Championship	2016	France	*#Euro2016*	3,154,605	21,559	975
Olympics	2016	Rio de Janeiro (Brazil)	*#Rio2016*	6,111	3,359	525
SuperBowl	2016	USA	*#SuperBowlSunday #NFL*	57,572	23,876	582

Tweets were captured for each event using event-related topics such as *#FIFA2014, #superbowlXLIX, #CWC15, #Euro2016, #Rio2016, #RWC2015, #NFL, #SB50, #SuperBowlSunday*. Figure 3.2 gives the distribution of tweets captured for each sporting event. The

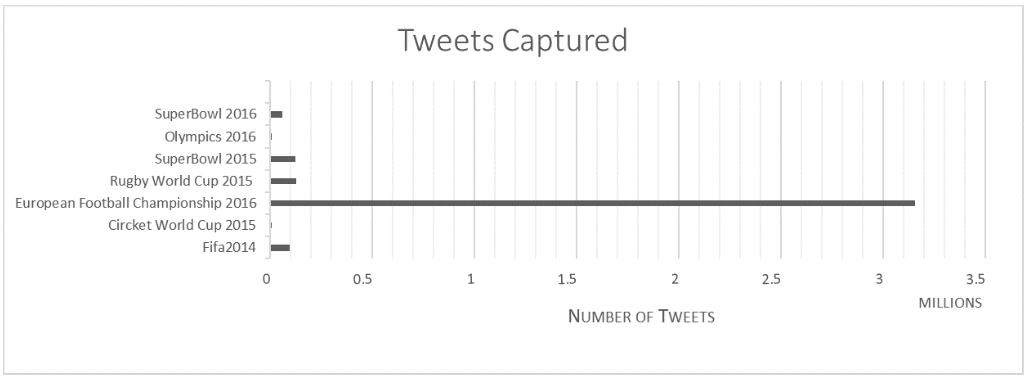


Figure 3.2: Number of Tweet's captured for each sporting event over a period of four year

minimum sample of 6,111 tweets was collected for the Olympics 2016 opening ceremony and the maximum sample of 3.1 million tweets was captured for the European Football Championships 2016. The rationale behind selecting diverse popular events was to gather evidence of these events being used to deliver drive-by downloads, as well as the subsequent statistical findings, that would generalise beyond a single event.

3.2.2 Identifying Malicious URLs

The second stage of the experiment required visiting the Web pages from the URLs that were captured during the sporting events. A sandboxed environment in which websites were visited in an isolated subnetwork was used to protect the connected networks. These protective measures were taken because there was a possibility a drive-by download attack may occur during the visitation of these websites in which case the system could act as an entry point to infiltrate and compromise the connected network.

A honeypot was setup in a sandboxed environment to systematically and automatically visit the websites. Two types of honeypots exist - a high interaction and a low interaction client-side honeypot. Low interaction honeypots perform static analysis on a Web page code, looking for evidence of malicious scripts [49, 109]. High interaction honeypots use dynamic analysis of the interaction behaviour between the Web page and the client system, looking for evidence of malicious actions [214, 219]. Also, these honeypots may act as active defence systems that visit potentially malicious URLs and log the system state during the interaction.

Since the aim of this chapter is to identify associations between popular events on Twitter and cyber attacks, and build the foundations of an approach to predict malware behaviour, a low interaction honeypot based on static analysis was ruled out. First reason being cybercriminals take measures, such as encrypting the malware code to avoid being detected. Secondly, because the high interaction honeypot detection of malware is based on behavioural analysis that fits with our novel approach to using machine activity in the detection of malware i.e. overcoming dynamically changing feature limitations such as code signatures and URLs/domain names of malicious sites. Three high interaction honeypots, Capture HPC[27], Threat analyser [200], and Winpooch [15] were considered to support the capture and observation of behaviour of a Web application and to flag suspicious activity. Transparency, portability and confidence in the report generated were the three parameters used to evaluate the effectiveness of a honeypot. Transparency reveals the inner working of a honeypot, allowing the end-user to understand the working of the honeypot and interpret the outputs. Portability enables a honeypot to be used in different environments and allows it to be configured as desired. Finally, there should be a high level of confidence in the correctness of the report generated outlining application behaviour. If an infected or malicious application manipulates the system using low-level system calls, honeypots observing behaviour based on high-level function calls would not be able to detect the malicious nature of such an application. Hence, a honeypot should observe application behaviour in terms of state changes at the lowest possible level, so there is high confidence that the report generated is correct.

Capture HPC, Winpooch and Threat Analyser were judged useful, based on transparency, portability and confidence in the correctness of the system activity generated report while observing application behaviour (see Table 3.3). Threat Analyser only provides portability, meaning it can be easily used in a different environment. Winpooch that is designed especially to prevent malware infection was not portable but transparent and observed application behaviour at the lowest possible level resulting in high confidence in the report generated detailing the application behaviour as correct. Capture HPC being open-source and designed especially for Win32 operating system exhibited all the three features, because of which it was chosen for all our experiments.

Table 3.3: Comparison matrix between various High interaction Honeypots

Tools Used	**Confidence in Report**	**Portability**	**Transparency**
Capture HPC	✓	✓	✓
Winpooch	✓	**X**	✓
Threat Analyser	**X**	✓	**X**

3.3 Capture HPC

The capture HPC honeypot system is designed on a client-server model, where the server provides instructions for the client to execute and, in return, the client executes the instruction and passes on the results to the server. Since the honeypot would be visiting a malicious website, both the server and the client of the honeypot are set up in a sandboxed

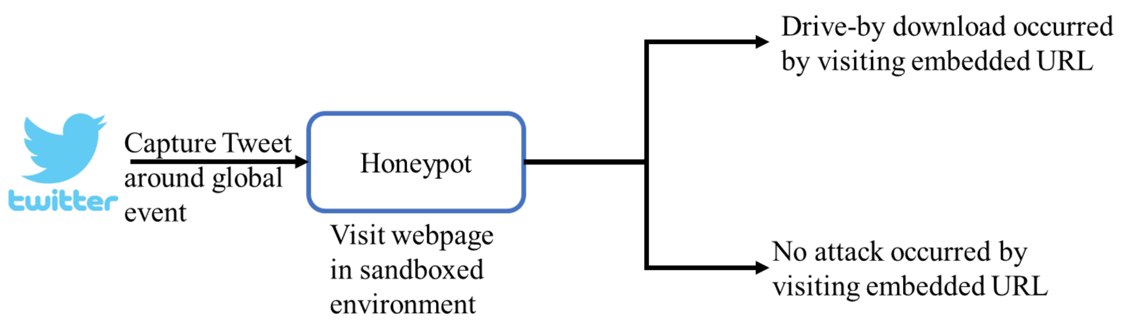

twitter
Capture Tweet around global event
Honeypot
Visit webpage in sandboxed environment
Drive-by download occurred by visiting embedded URL
No attack occurred by visiting embedded URL

Figure 3.3: Experimental setup to detect drive-by download attacks

environment to prevent the entire network from being infected. Figure 3.3, highlights the experimental setup used to detect a malicious URL, where URLs that were captured during popular events were opened inside the honeypot. Keeping in mind that a drive-by download attack could occur while visiting the Web page, a sandboxed environment was created using VirtualBox, an open source hosted hypervisor for *x86* computers, and inside this the Capture HPC server on the Debian operating system was configured. Two capture HPC clients were configured, one running Win7 and other on WinXP. These operating systems were chosen for their popularity: 48% of desktop personal computers in the world still use Win 7 and even though Microsoft discontinued updating WinXP it is still in use by 7% of all Desktop users [104].

Furthermore, each client machine was equipped with Java version 7, Flash player 12, Shockwave player 12, Internet Explorer 8.0 and Firefox 29.01. Upon bootup, the Capture server was initiated, which in turn invoked all the Capture clients via the VMWare Vix library within the virtual environment. Figure 3.4 gives an overview of the interactions between a Capture HPC server and a client. For the communication between client and server to be successful, the Capture client established a socket communication channel between the server and the client in order to exchange various files and information. Once a communication channel was established, the Capture server directed the Capture client to visit a particular website. While visiting the website the system activities in terms of changes to the *file, registry* and *processes* were captured and reported back. These changes were analysed, and a URL was classified as malicious if the configuration rules (the exclusion list for the file, process and registry, see 3.4) were violated.

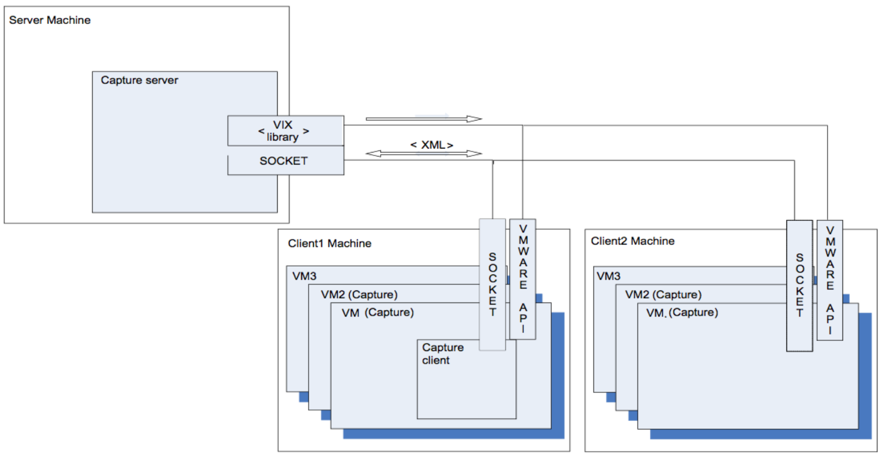


Figure 3.4: Overview of the Capture HPC system [91]

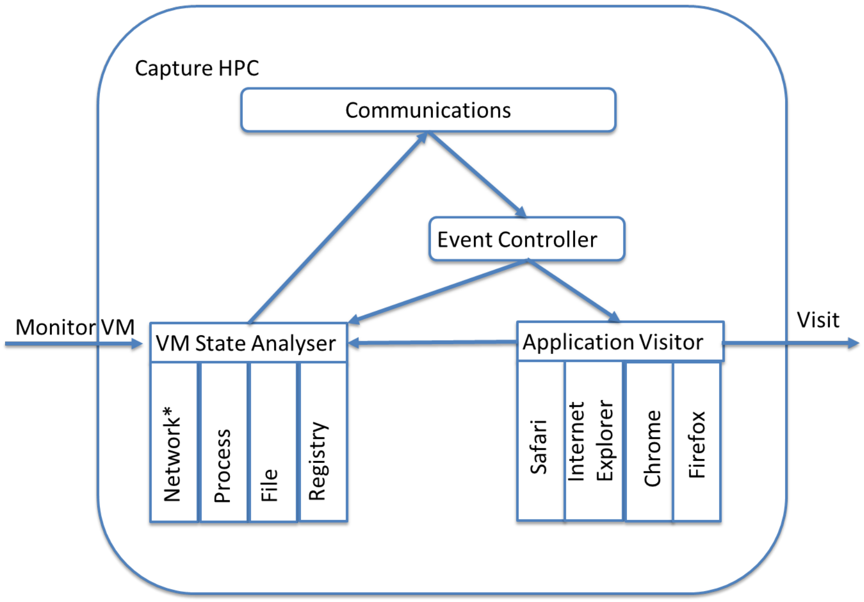


Figure 3.5: Capture client components [91]

Figure 3.5 gives an overview of the architecture of the capture client that is composed of four subsystems :

1. Client/Server communication.
2. Visitor.
3. Analyser.
4. Event controller.

3.3.1 Client/Server Communication

This is responsible for establishing a communication channel between the client and the server. It has two components, transmitter and receiver, which are responsible for sending XML-based messages. Once the client side receives the message, it passes it to the Event Controller, who later decides which event to initiate.

3.3.2 Visitor

When the Capture server requests it via the event controller, the visitor subsystem is responsible for directing a Web browser to open a URL sent by the Capture server. It is also responsible for initiating all the plugins required by the Web browser. One of the primary responsibilities of the visitor is to keep the analyser informed of every activity it engages in, from initiating the plugins to any errors that may occur during the visit to the website.

3.3.3 Event Controller

This is responsible for handling requests from the server and communication between the sub-components of the visitor and the analyser. It decodes server requests and initiates the visitor component and the analyser.

3.3.4 Analyser

The analyser is responsible for maintaining various monitoring subsystems, whereas a monitoring system itself is responsible for monitoring and analysing the system activities. The three monitoring systems that Capture HPC has are as follows:

The File System Monitor

Once initiated, it records all the read or write events from all the current files mounted in the operating systems. It records all the relevant information for the triggered event, such as the process that triggered the event, the file and the directory name related to the event.

The Registry Monitor

Similarly once it is initiated, it starts to monitor the entire Windows registry. A Windows registry holds all the critical information for an operating system, which may be anything from the configuration of the operating system to the information about the programs installed on this operating system. The registry monitors and records the time of the event and the process by which the registry was accessed; it edits the path of the key and the action performed on the key.

The Process Monitor

Like the other monitors it records activities - in this case it is concerned with process activity. It captures all the processes that were either created or destroyed while the URL was being accessed, but ignores the processes that are already running. Like other log files generated by system monitors, this also records the file name that represents the process. It also captures the parent process for the triggering process.

3.4 Exclusion List

The primary operation of Capture HPC is to observe system activities with regard to *file*, *process* and *registry*. During the observation, an operating system is capable of generating hundreds of machine state events. For instance, on an idle machine that had Windows XP SP2 system, 530 registry entries and 60 file entries were observed in one minute [181].

In order to differentiate the malicious events from the benign ones, Capture HPC relied on rules that were defined while configuring the Capture HPC server. These rules were configured in the server as three files called the exclusion list, namely, *FileSystemMonitor.exl*, *RegistryMonitor.exl* and *ProcessMonitor.exl*.

Exclusion lists are simple text-based files that are easily amendable to add more rules that aid in detecting malicious behaviour. Also, these files could easily be ported by merely copying these files from one machine to another (see appendix for detailed exclusion list). By default, no rules were defined in each exclusion list, giving the end-user the flexibility to define and update their own malware detection rules. Without any rules Capture HPC flags every activity as malicious, meaning by default everything that was observed for an application aided in classifying the application as malicious. In order to avoid this, rules were defined that instructed Capture to ignore those activities that are benign and flag those that were malicious. The process of identifying those activities that were benign is highlighted in Figure 3.6, where a website which is most probably a benign website is chosen, it is then checked using HoneySpider 2 [197], a static analyser that determines whether a website is malicious or not. The website is then counter checked by Virus Total [201], an online website maintained by Google to check whether a website is malicious or not. The results of both tools are combined, and only those websites that are benign are then opened in a Capture client and the activities recorded while visiting them are included in the exclusion list for the omission. Many benign websites are used to create a comprehensive exclusion list.

For malicious rules, activities that were without any doubt malicious and should not occur while visiting a Web page were created in the exclusion list, such as the automatic download of an executable file, execution of a batch file, modification in Win32 folder. In addition to these, any activity that occurred and was not defined in the exclusion list would result in classifying the URL as malicious. Also, activities recorded, against which rules are not defined were reviewed to check if these were correctly mapped to malicious activities. In order to do so, the log file that was generated was inspected to check if another process triggered the activity or not. These were then checked using Windows

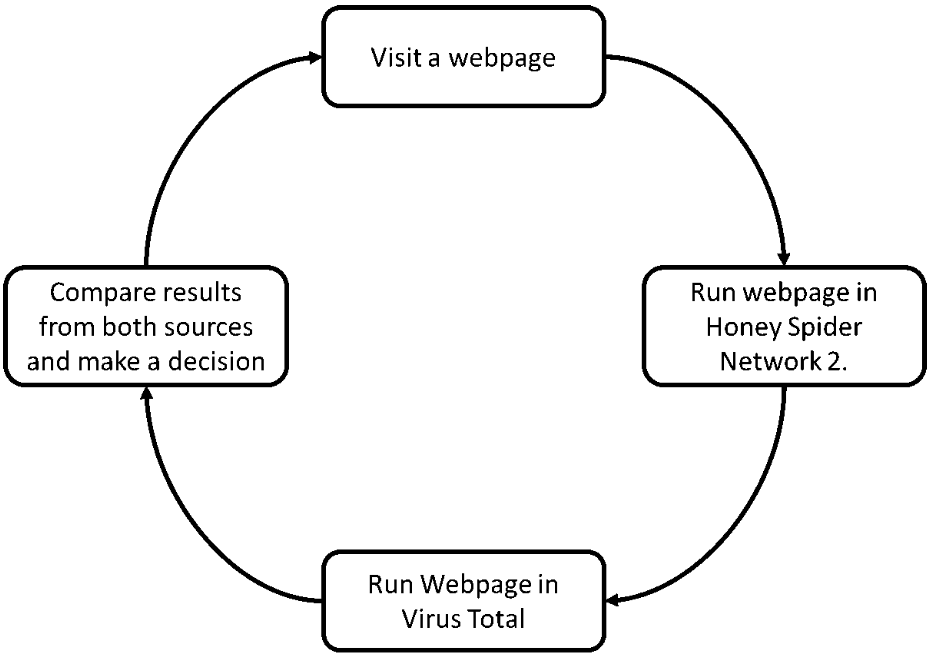

Visit a webpage
Run webpage in Honey Spider Network 2.
Run Webpage in Virus Total
Compare results from both sources and make a decision

Figure 3.6: Determining a website to be benign leading to formation of exclusion list.

internals book and the operating system help pages [141] to map the activities with respective processes and files. With the help of these documents, it could be determined whether the excepted function that generated the activity was behaving normally or not. Finally, VirusTotal [201] was used and based on the results the URLs were either marked as benign or malicious activities.

An exclusion list is a file that contains symbols indicating (i) the type of event, (ii) the process name, and (iii) the file path of the event. Figure 3.7 shows the content of an exclusion list, in which each row specifies a rule that was created for the honeypot to follow. An omission rule has the prefix of a plus sign '+', and an inclusion rule has a prefix of

```
# [Evt Type] [Process Name]  [File Path]
+ Read       .*              .*
+ Write      .*              C:\\WIN\\.+
- Write      .*              C:\\WIN\\sys\\.+
+ Write      C:\\BROWSER.EXE C:\\CACHE\\.+
```

Figure 3.7: Contents of file exclusion list [181]

'-' sign. In Figure 3.7, line 2 specifies that the writing in the folder *c:\win\* should be omitted whereas any writing in the folder *c:\win\sys\* should be flagged. This feature of

Capture HPC gives the user a great deal of flexibility and provides transparency in classifying an activity in a malicious or a benign category. The main exclusion rules in each list are discussed in sections 3.4.1, 3.4.2 and 3.4.3, to clarify this categorisation of the URL. Each exclusion list is tailored to capture the activities for both the Capture client, namely, the WinXP and Win 7 operating system.

3.4.1 File Exclusion List

This exclusion file contains lists of all the files, those which should be ignored and those which should be highlighted as malicious. The main rules that are added to the file exclusion list are as follows:

1. **Capture files.** The honeypot imitates the program that will monitor the system state changes and any triggers raised by this program are to be ignored. The rule mentioned under this heading concerns the files that are accessed by the Capture client to carry out its processes and are to be ignored. The rules define that any executable (exe) file executed or accessed from the Capture directory in program files should be ignored, and that the writing of the log file, which is further communicated to the server, should be ignored as a malicious activity.

2. **Prefetch Files.** Each time a computer is turned on, Windows keeps track of the way that it starts and which programs is generally opened first. Windows saves this information as a number of small files in the prefetch folder. The next time the computer is turned on, Windows refers to these files to accelerate the start process. However, if the file is not a part of Windows but is from any other source it will automatically raise an alarm and be flagged as malicious, but will ignore any writing in the Windows prefetch and *svchost* folders.

3. **New Technology File System (NTFS) Meta Data.** To run efficiently, the NTFS contains several files that define and organise the file system. Like any other operat-

ing system, these files are structured in the same way as any other user file ($Volume being the most peculiar) but are not of direct interest to file system clients. These meta-files define files, back up critical data in the file system, buffer file system changes, manage free space allocation, satisfy basic input/output system (BIOS) expectations, track bad allocation units, and store security and disk space usage information. Because the Master File Table (MFT) stores information about itself, NTFS reserves the first 16 records of the MFT for meta-data files (approximately 16 KB), which are used to describe the MFT. Meta-data files that begin with a dollar sign ($) are described in the Metadata Files table and are stored in the Master File Table. As access to these files is always a routine activity, an exception was created in the list in order to ignore any access to these files. These files are as follows:

- $*mft* –contains one base file for each file and folder on the drive.
- $*mftmirr*- is the mirror image in case mft file fails.
- $*logfile*- contains information for recovery in case of any crashes.
- $*volume*- contains volume information.
- $*directory*- contains directory information.
- $*AttrDef*- contains list of attributes such as name, number and description.
- $boot- includes the BIOS parameter block (BPB) used to mount the volume and additional bootstrap loader code used if the volume is boot table.
- $*bitmap*- contains information for free and unused cluster.
- $*badclus*- contains information about bad clusters.
- $*quota*- keep tracks of file quota.
- $*upcase*- is an uppercase table used to convert characters from lower to upper case.
- *$ReplaceAttribute2* and *1*- to optimise the storage and input or output overhead.

4. **Performance**- uses certain files such as Windows Management Instrumentation (WMI) Performance Adaptor (wmiadap) and those in the system32 folder to enhance the performance of the operating system. Access to these files is benign and should be ignored.

5. **System Log Files**- In Windows XP, an event is added to a log if it is of any significant occurrence in the system or in a program that requires users to be notified, or is the subject of an entry. The system log contains events logged by the Windows XP system components. Hence, writing system log files should be ignored

6. **Windows update**- activities responsible for Windows updates should be ignored, because a Windows update sometimes starts in a way that could falsely be picked up as a malicious activity. The exception list here is configured to ignore any write activity on *wuauclt*, *windowsupdate* and *softwaredistribution* directories.

7. **System events**- In Windows XP, if an event is of any significant occurrence in the system or in a program that requires users to be notified, or an entry has been made which is of significance, it is added to a log. Any activity related to the event should be ignored; however, exceptions to *AppEvent*, *SysEvent* and *SecEvent*, in the exception list, have been made to prevent the false triggering of malicious activities.

8. **Mapping** – In this, we are mainly concerned with Web-Based Enterprise Management (WBEM), which is a set of systems management technologies developed to unify the management of the distributed computing environments. In this case, an exception has been made to ignore all activities related to this from the *wbem* folder.

9. **Cataloguing**- A security catalogue is the part of Windows that handles digital signatures for updates, system file protection and other functions. Hence any activity related to this should be ignored. The main folders where this exception is to be applied are *Catroot*, *svchost*, *winevnt* and *lastalive*.

10. **System Restore**- As the name suggests, System Restore is a feature in Microsoft Windows that allows users to revert to a former state of their computer (including

system files, installed applications, Windows Registry, and system settings).This can be used to recover from system malfunctions or other problems. The exception list contains rules to ignore all activities related to this event.

11. **User Date**-A user profile describes the desktop computing configuration for a specific user, including the user's environment and preference settings. Consequently, any activity in the *UserClass* folder should be handled in accordance with the rule in the exception list.

12. **Internet Explorer**- Our Honeypot Capture client accesses each URL using Internet Explorer (IE), so our exception list, based on system activities, has designed exception rules for each subcategory, as detailed below.

    - *IE temporary files/ Internet Cache*- Temporary Internet Files is the name of a folder on Microsoft Windows, which holds browser caches. This folder usually holds multimedia content and Web pages for quick access. Not only do Web browsers have access to the directory to read or write, but also have access to the Window Explorer and Windows Desktop Search. Our exception list ignores any access to the files of this folder.

    - *History* – Internet Explorer keeps track of all the websites that have been visited in the past, and our exception list ignores any access to this folder.

    - *IE cookies*- In the cookie folder, we created an exception for the *index.dat* file. This is a database file, which is a repository of information such as Web URLs, search queries and recently opened files. Its purpose is to enable the data used by the Internet to be quickly accessed.

    - *User Data*-This folder also contains information which is used to improve performance. Hence, the exception list contains rules to ignore access to this data.

    - *Plugins (Flash players)*- Plugins such as Flash will be accessed in visiting a URL; hence, any such access is ignored.

- *Digital rights management (DRM) related*- Internet Explorer accesses the DRM folder to check the information on various digital rights and thus access to this folder should not be flagged as malicious.
- *Msg ActiveX*- Internet Explorer may use this folder for content download from the Internet. Our exception rule prevents the wrongful marking of any website as malicious if this folder is accessed.

Just as Internet Explorer uses various folders to function efficiently, there are other browsers such as Chrome and FireFox which also use similar folders to perform efficiently. Though our honeypot client may open Internet Explorer by default, the Capture client's exclusion list has made provisions to ignore the files and folders that are accessed by other browsers. For instance, as listed below, there are certain folders for which rules are designed for Firefox.

1. To check Start-up cache.
2. Profile file access.
3. Web-apps.
4. Temporary files.
5. Firefox cache.
6. History.
7. Cookies.
8. Safe browsing malware (inbuilt feature).
9. Certificates.
10. Session data.
11. Firefox Preferences.

12. URL classifiers.

In addition to these exceptions, there are rules which determine whether the activity should be flagged as malicious or not. These are sub-categorised as follows:

1. Alerts about executable and scripts that are written on to disk; any scripts or files such as
    - Batch file **‘bat’**
    - Command file **‘cmd’**
    - Executable file **‘exe’**
    - Setup information file **‘inf’**
    - Windows installer file **‘msi’**
    - Windows installer path file **‘msp’**
    - Program information file **‘pif’**
    - Registry file **‘reg’**
    - Scitex Continuous Tone file **‘sct’**
    - Microsoft scrap file **‘shs’**
    - System Configuration Repository **‘scr’**
    - Windows Script Components **‘wsc’**
    - Windows script file **‘wsf’**
    - Windows script host **‘wsh’**
2. Alerts about modifications to startup locations – any changes in the Windows start-up folder, *win.ini*, or the Windows task folder, are flagged as malicious activity.

3.4.2 Process Exclusion List

Once the Capture HPC client has booted up, the process exclusion list is loaded in the client's machine along with the file exclusion list. The process exclusion list contains a list of the processes which should be ignored if they are started. Like the file exclusion list, the process exclusion list has also been sub-divided into the categories listed below.

1. **Capture script** – Our process exclusion list should include the capture process that is running in the background and it should not be flagged as malicious activity. The processes that are checked are *CaptureClient.exe, CaptureClient.bat* and the files which are executed in the capture folder.

2. **Windows Update**- At times, even if a Windows update is disabled while running, the operating system calls up a process. Hence, our exclusion list will ignore the running of wuauclt.exe and savedump.exe as processes.

3. **Standard screen-saver**- It is possible that the screen saver is started while the Capture client is being analysed. Therefore, the *logon.scr* process is included in the exclusion list and is to be ignored.

4. **Defragmenter**- While the Capture client is running the operating system, the process of defragmentation may start. Thus have included *dfrgntfs.exe* and *defrag.exe* as items are to be ignored if picked up by our monitors.

5. **7za**- the 7za process that would be running since it is used to unzip files for the Capture client.

6. **Mapping**- This activity is carried out for the efficient management of distributed computing environments, usually by invoking *wmiadap.exe* and *wmiprvse.exe*. Therefore, our exclusion list includes instructions to ignore such processes.

7. **VMWare tools**- VMWare would be running so processes which belong to VMWare, namely *VMWareUser.exe* should be ignored.

8. **Capture Client**- For the Win 7 Capture client we have configured the exclusion list of items to ignore:

    - Svchost is a process that contains or hosts other services required by the operating system to carry out its process.
    - Win 7 Task scheduler is a process in which we have made an exception to allow *googleupdate.exe* to run.
    - Win 7 built-in search process, which starts on bootup, namely, *SearchProtocolHost.exe*, *SearchFilterHost.exe* and *SearchIndexer.exe* should be ignored.
    - Windows login scripts namely *userinit.exe* and *winlogon.exe* should be ignored.
    - Critical system processes like *csrss.exe* should be ignored.
    - Conhost which is responsible for fixing bugs from previous operating systems, is started when the system is booted. Processes *conhost.exe* and *mobsync.exe*, which are responsible for this activity, should be ignored.

9. **Internet Explorer** – processes running to support Internet Explorer, namely *iexplore.exe, agentsvr.exe, msmsgs.exe, rundll32.exe* and *imapi.exe* should be ignored.

10. **Other Browsers** - As with the file exclusion list, provisions have been made to ignore processes which would run if any other browser such as Firefox was running instead of Internet Explorer.

3.4.3 Registry Exclusion list

So far we have worked on the process and file exclusion list, but must also bear in mind that a malicious website will also try to make changes in the registry. We have configured an exclusion list to highlight malicious activity and also to ignore any registry entry which could have resulted from routine operations while visiting a website. First, we look into the registry keys which we would ignore as benign since they are accessed during regular operations by various processes running whenever a website is opened for evaluation. Then we discuss the registry keys that are highlighted as malicious. As with process and file exclusion lists we also divide our registry list into various categories and elaborate on each point.

1. **Capture Client**- There will be a value set in the registry when the Capture client is running. The keys that would be affected and have been configured to be ignored are as follows:

    - Whenever an application is started the program count value in the registry is incremented to show the number of programs running. This value, as a benign activity, should be ignored. The registry key is HKU\SessionInformation \ProgramCount

    - Microsoft uses a set of registry keys to maintain the dimensions, icons and position of a folder using Explorer. This activity is part of a regular operation and is to be ignored. The key value set in registry is HKCU\Software \Microsoft\Windows\ShellNoRoam

    - Windows will update the information for all Windows installer products and components that are installed for the user once logged on. Registry entry to this would be considered as benign; the value in the registry accessed is HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Installer \User Data\

    - Windows sets a registry key to hold an entropy value, which is the randomness collected by the operating system. This is further required for cryptography purposes and the registry key that is set is HKLM\software \Microsoft\ Cryptography\RNG\Seed

    - Changes made by Capture Client from path C:\Program Files\Capture \CaptureClient.exe

    - System configuration information, such as drivers, is updated by setting the value to a registry key that should again be ignored. The value that is updated is HKLM\SYSTEM\ControlSet001\

2. **Internet Explorer** - Every URL will be accessed using Internet Explorer. While the browser is running registry values for that program will be set as part of the regular operation and these should be ignored. They are as follows. Similar to the Capture Client process we also allow fields such as SessionInformation, Shell-noRoam, Internet Setting (how the browser should connect to the Internet), group policy, System certificates, interface list and driver setting. Registry editing was allowed through only those processes which were permitted in the process list, such as *svchost*, *wmiadap*, *wmiprvse* and *lsass*.

3. **De-fragmentation**- Registry modification, if defragmentation process is started, should be ignored, subsequently any changes in the HKLM\SOFTWARE\Microsoft \Dfrg should be ignored.

4. **Windows Update** - A registry entry is created when windows update is taking place, hence any setting up of value for HKLM\SYSTEM\ControlSet001\Services \EventlogV Application\ESE NT \should be ignored.

5. **Capture Client** - While running a Capture Client process in a Win 7 environment provisions were made to ignore all modifications made by legitimate applications. The list of keys that have been ignored while this application is running is as follows:

   - *Internet Explorer*- To ignore any registry updates made by Explorer such as home group owner setting, driver information, and so on.
   - *Search Indexer*-Any update made by process searchindexer.exe, which is responsible for indexing of your files, should be ignored.
   - *Local service Authentication server (lsass.exe)* Changes made in this process from Windows system32 folder should be ignored.

  - *Task scheduler* – modification to the key HKLM\SOFTWARE\Microsoft \Windows NT\CurrentVersion\Schedule\Handshake\should be ignored as it points to task scheduler being loaded.
  - *Svchost*- contains a number of individual services that perform a range of functions such as win defender services and can have multiple instances hence should be ignored. See appendix for complete list.
  - *System setting* – registry being updated for printers, fax, etc should be ignored.

6. **Additional rules** have been added which are specific to Internet Explorer 6 Sp2 and Internet Explorer 8 as well.A few are mentioned below:
   - *Registry entry*- for layout and notifications such as those mentioned below are to be ignored.
     - HKCU\Software\Microsoft\Internet Explorer\Main\Window_Placement
     - HKCU\Software\Microsoft\Internet Explorer\Main\Fullscreen
     - HKCU\Software\Microsoft\
       Internet Explorer\Main\NotificationDownloadComplete
     - HKCU\Software\Microsoft\Internet Explorer\TypedURLs
   - *Plugins*- Registry being updated due to plugin's in the Internet Explorer such as media player, multi-media player, and so on.

The list of rules to be ignored also included rules that would capture malicious activity if any is observed while the Capture Client is running. Any modification in the registry, which would affect the Windows start or boot up sequence was highlighted and flagged as malicious activity. For this, it is necessary to look at registry keys mentioned below. These rules are as follows:

1. *Run and RunOnce keys from registry* – these keys cause the program to run once the user has logged into the system. Any addition and changes to this key should be flagged as malicious [142].

2. *Userinit*- specifies the program that should start once the user logs into the system; hence any modification to this key should be recorded [140].

3. *BootExecute*- specifies the application, services and command executed during startup of the operating system [140].

4. *ShellServiceObjectDelayLoad*- This is similar to the Run Key; The only difference is that instead of pointing to the file itself, this points to the class id's (CLSID) InProcSever, which contains information about the DLL file that are to be used.

Capture HPC iterates through the list, visiting each URL, and keeping the connection open for five minutes, logging machine activity on the client machine and identifying whether the URL is malicious or benign via reference to its exclusion list. The 5 minute interval is currently a heuristic to ensure that a large number of sites can be visited – it makes the significant assumption that any malicious activity will be triggered within the first 5 minutes of the visit.

3.5 Results and Discussion

Table 3.4: Total Tweets Captured

Event	**Number of Tweets**
Fifa2014	95,000
Cricket World Cup 2015	7,961
European Football Championship 2016	3,154,605
Rugby World Cup 2015	127,393
SuperBowl 2015	122,542
Olympics 2016	6,111
SuperBowl 2016	57,572

We collected 3,571,184 tweets containing URLs across seven sporting events to gather evidence of malware presence around popular events. The total number of captured tweets for each event is given in Table 3.4. The captured tweets were pre-processed by removing the tweets containing the same URL, that is retweets or the same tweet posted again.

From these individual tweets, URLs were stripped and were passed on to the Capture HPC honeypot. Due to the Web-page visitation time allocated (5 minutes), Capture HPC was limited to process only 500 URL per day, a limitation that does not allow the entire dataset to be checked for malicious behaviour in a short period of time. A random sample of 500 unique URLs from the collected tweets for each event were selected and passed to Capture HPC for the next two weeks. The results shown in Table 3.5 show that the maximum number of tweets containing a malicious URL were found during the FIFA 2014 football event, which generated around 672 million tweets [59] during the one-month period and the least number of tweets (525 - see Table3.5) containing malicious URLs were found during the Olympic 2016 opening ceremony. In terms of percentage, the highest percentage of tweets (19.26%) that were reported as malicious were in FIFA 2014 and lowest being reported for Superbowl 2016 (8.31%). Events related to football (FIFA 2014 and EURO 2016) were used the most to deliver drive-by download attacks (see Table3.5). Whereas, a substantial drop in number in malicious URLs identified were seen for Superbowl.

The URLs were classified as malicious based on the system changes they made during the

Table 3.5: Number of Malicious URL Captured

Event	**Number of Unique Tweets**	**Percentage of Malicious Tweets**
FIFA 2014	2,039	19.26
SuperBowl 2015	1,230	17.57
Cricket World Cup 2015	891	11.19
Rugby World Cup 2015	627	08.96
SuperBowl 2016	582	08.31
European Football Championship 2016	975	13.93
Olympics 2016	525	10.12

visit. These changes included modifying files in Window's System 32 folder or modifying the Internet Explorer file or modifying the dynamic link library files. They were also labelled as malicious due to the processes started as a result of the execution of malicious executable files from the Internet.

Twitter allows each user to follow other users, which allows them to see what others

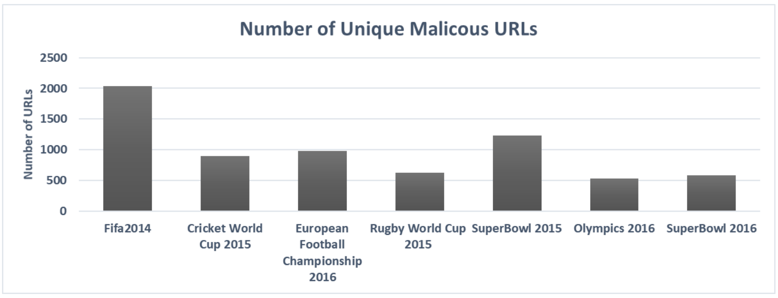


Figure 3.8: Number of Unique Tweets captured containing Malicious URLs

Table 3.6: Ratio of Followers by Following for accounts posting malicious URLs

Event	**Followers by Friend Ratio greater than**						
	>10,000	10,000-1,000	1,000-100	100-10	10-1	0-1	Only Followers
Fifa2014	0.002%	0.039%	0.706%	10.434%	81.044%	7.855%	0.022%
Cricket World Cup 2015		2.360%	19.821%	27.230%	35.087%	15.432%	0.118%
European Football Championship 2016	0.028%	0.148%	2.129%	6.355%	77.494%	13.660%	0.232%
Rugby World Cup 2015		0.052%	3.128%	38.921%	34.176%	22.993%	0.704%
SuperBowl 2015		0.000%	2.268%	24.858%	43.611%	29.132%	0.131%
Olympics 2016	0.060%	1.250%	18.339%	40.161%	27.538%	12.474%	0.268%
SuperBowl 2016		0.000%	0.046%	0.846%	43.140%	55.901%	0.071%

are posting. Users who are *Following* someone on Twitter will be subscribing to (someone else's) tweets as a follower and their updates will appear in the follower's *Home timeline*; that person will also be able to send *Direct Messages*, these are also referred as *Friends* of a user. *Followers*, in contrast, are people who receive others' Tweets. If people follow a user, they will show up in the followers' list, and they will see users' Tweets in their Home timeline. Also, whenever they log in to Twitter, they can start a private conversation by sending a direct message. The following and friends characteristics on the accounts of users who posted tweets containing malicious URL were investigated. The aim was to observe and identify patterns emerging out of OSN accounts that have been used to post malicious URLs.

In order to successfully deliver a drive-by download attack, a cybercriminal needs to have a potential list of users. On Twitter, this is represented by the number of followers a cyber criminal's account have who will view the malicious post. One of the challenges the cybercriminal faces is to increase the number of followers. One way to increase the number of followers is to constantly follow users, with the hope that they will reciprocate the gesture by following them back. Evidence of this reciprocal relationship has been observed among users on Twitter [94] .

In order to identify similar behaviour between accounts posting malicious content, seven categories were created. Each category represented the ratio between the followers and friends. The ratio was calculated by dividing the number of *Followers* by *Friends* an account had. The higher the ratio was, the more likely it was that the account will post spam [192]. We found 0.12% accounts (121 accounts in total) across seven events posting malicious tweets that were following people but had no followers themselves. At the other extreme, seven accounts were identified (see table 3.6) that had a ratio higher than 10,000. Accounts that had a follower by following ratio between 10,000 and 10 posted 12.8% of the malicious URLs. We also found an average 65% of the malicious tweets were posted by accounts having a ratio between 1 and 10 across all events. The result showed that accounts that were used to post malicious content were following more people than they were being followed by others, a tactic used to gain followers. By increasing the number of followers a cybercriminal could increase the number of users exposed to malware, and using the reciprocal technique it is possible s/he could increase the number of followers by following people. Thus these features are frequently used in constructing spam [192, 12, 183] or malware [127] detection models. An interesting observation was that there were malicious accounts that exhibited ratios between 0 and 1, a ratio that was usually associated with normal accounts [192].

Table 3.7: Age of User Account Including Those Re-Tweeting Malicious URL

Event	**Percentage of User Account Posting Malicious URLs including Re-Tweets based on their age**			
	<1 month	**1-3 month**	**3-6 month**	**>6 month**
European Football Championship 2016	5.4%	6.0%	5.9%	82.7%
Olympics 2016	2.7%	3.6%	6.0%	87.7%
SuperBowl 2015	3.0%	1.5%	3.4%	92.2%
Cricket World Cup 2015	12.1%	5.1%	5.5%	77.2%
Rugby World Cup 2015	5.1%	3.8%	6.1%	85.1%
SuperBowl 2016	51.1%	3.3%	1.3%	44.3%
Fifa 2014	4.2%	4.6%	5.6%	85.6%
Total Percentage	15.3%	4.5%	4.7%	75.6%

In addition to OSN characteristics such as friends and followers, age of an account has been used to detect those accounts that were posting spam [227, 126, 144, 43, 183]. An investigation was conducted with an aim to observe the effect of account age posting malicious content across all the seven events. In order to do that, the age for each account that posted malicious tweets was calculated as the number of days between the day that the malicious tweet was posted and the day that the account was created. The results showed 21% of the accounts across all events had been created 6 months or more before the event. 15% of the accounts were created under one month before the sporting event. Interestingly for Superbowl 2016, 51% of the accounts that were posting malicious tweets were created one month before the event (see table 3.7). These results showed a significant association between accounts posting malicious content and their age and should therefore be included in a malware detection/prediction model.

3.6 Conclusion

In the beginning of the chapter a question was posed *"Are popular events that attract millions of users on online social network being exploited to carry out drive-by download attacks by cybercriminals?"* In answer to this, we can say that there is substantial evidence to show that there is an association between popular events on Twitter and drive-by download attacks. The evidence was gathered by collected 3,571,184 tweets from seven diverse sporting events in three years. From these, we randomly selected a sample of 48,991 unique tweets that were passed on to the high interaction honeypot, Capture HPC. Of the URLs processed through Capture HPC, 6,879 (individual tweets) and around 105,642 (tweets including retweets) were established to be malicious through an automated analysis of each URL. This process involved a virtual machine interacting with an endpoint of the URL for a fixed period, using an exclusion list of known malicious activity to detect malicious behaviour.

In this process, results showed that popular sporting events on Twitter have continually been used by cybercriminals to carry out drive-by download attacks. A maximum of 19.26% accounts during FIFA 2014 were identified as posting malicious URLs and a minimum of 8.31% for Superbowl 2016. Even though the number of accounts posting malicious URLs have gone down from 19.26% (FIFA) in 2014 to 10.12% (Olympics) in 2016, there were still a significant number of accounts that were detected as malicious.

This is likely due to the popularity of an event attracting millions of users generating thousands of tweets per second. For example on average 6,000 tweets [18] are tweeted every second and a record of 143,199 tweets were tweeted during a popular event (airing of Castle in the Sky in Japan) [205]. With so many tweets being generated a user acts quickly to investigate the tweet. In addition to this a 280 character restriction imposed by Twitter gives cybercriminals the opportunity to obfuscate a malicious URL in an exciting tweet. Furthermore, with less reaction time and obfuscated URLs, an interesting tweet acts as a click bait to lure users to visit the Web-page so that upon visitation the malicious code is executed on their systems.

The tweets identified as malicious were investigated through the characteristics of Twitter accounts to identify any patterns exhibited by accounts posting malicious tweets. Results showed that accounts that had a higher number of follower and followee ratio indicated that malicious account were following more people than they were followed by. The trend whereby accounts that were following a large number of people with fewer people following them back was seen across all seven sporting events, this is consistent with other areas where a user by means of reciprocal relationship had aimed to gain followers by following many users on Twitter [94] .

By using a honeypot, Capture HPC, evidence of popular events being used to spread drive-by download attacks was gathered. Accounts that were posting malicious URLs were identified by opening the URLs inside the honeypot and observing changes made during visitation. However, there were limitations to his approach, Capture HPC only observes changes made in terms of file, process and registry, it required high maintenance, and if not properly deployed it could act as access points to the network [180]. Detection models have been proposed based on malware code/behaviour, but these may fail against code/behaviour obfuscating techniques such as anti-symbolic execution obfuscation [10], anti-fuzzing [84], encryption [150], polymorphism [150]. This brings us to the question, is it possible to construct a novel detection model to detect drive-by download attacks on Twitter, that overcomes these challenges?

Chapter 4

Classification of Drive-by Downloads on Twitter

Popular events that attract millions of users on OSNs are being used to deliver cyber attacks. Existing literature documents the development of detection models by analysing malware code and by comparing with online resources such as blacklists. However, these models fail against code obfuscating techniques such as encryption and polymorphism. In this chapter, a novel detection model is proposed that classifies a URL into malicious/benign based on system activities, thus shifting the focus from analysing malware code to observing changes to a user's system by the malware. The proposed model was successful in classifying URLs into malicious and benign with a F-measure of 0.81 during training and 0.71 while testing on an unseen dataset.

4.1 Introduction

Online social networks have emerged as powerful tools for disseminating information. Among these, Twitter, a micro-blogging website that allows its users to express themselves in 280 characters, has emerged as a go-to source for current affairs, entertainment news and getting information about global events in real time. For example, Twitter has been used to study public reaction to events such as natural disasters [174], political elections [202] and terrorist attacks [25]. The growing popularity of Twitter and events that attract millions of active users has tempted cybercriminals to conduct attacks [231]. Also, evidence of popular events being used to carry out drive-by downloads has been found as discussed in Chapter 3. This evidence was gathered using a high interaction honeypot, Capture HPC, in which an attack was recorded if any unauthorised changes were made during the visitation of a URL. Even though Twitter has a policy to curb malicious content and successful models have been proposed by researchers, there is an 8% chance a Web request made on the Internet will point to a malicious Web-server [46].

Cyber criminals continue to develop new ways to overcome Twitter's policy and beat the detection models proposed to counter them. There is a requirement to develop detection models by shifting the focus from existing malware code/behaviour features to keep pace with cybercriminals. One such approach would be to focus on the effect of malware on user systems, by observing machine activities leading to an attack. After all, malware will reveal itself eventually to harm a system even if it is using different evading techniques such as polymorphism, encryption, triggers, etc. to hide the malicious code. In this chapter we propose a malware detection model that classifies a URL into malicious/benign based on machine activities, thus shifting the focus from malware code/behaviour to system behaviour during malware execution.

4.2 Background

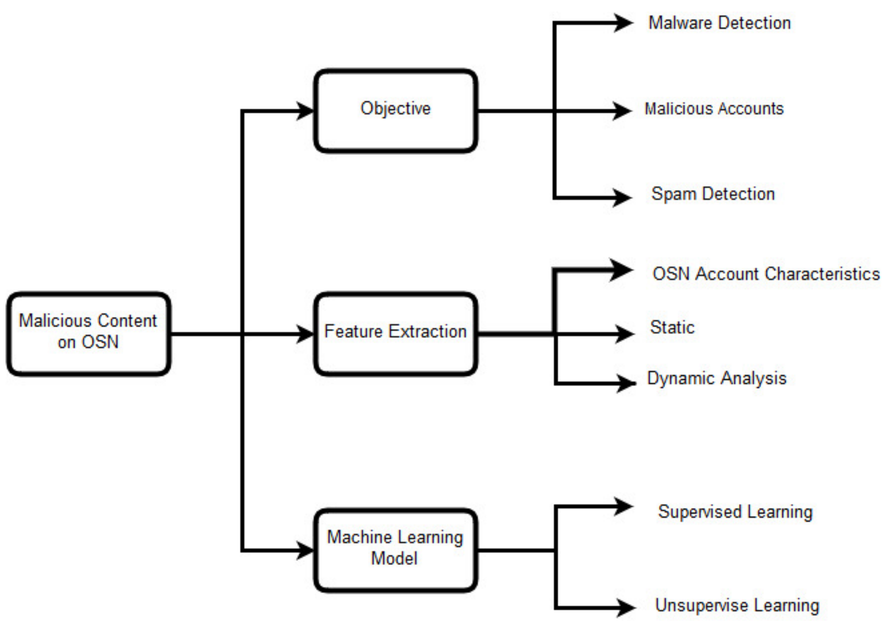


Figure 4.1: Malicious Content Detection on OSN

Figure 4.1 gives an overview of malware detection models for OSNs, in which research on detecting malicious content on OSN was conducted with three primary objectives, (i) Detecting malware on OSN, (ii) Detecting accounts that are spreading malicious content and (iii) Detecting Spam. Models that were developed to achieve these objectives relied on features that were derived from OSN account characteristics [12, 183, 227, 126, 144, 43, 192, 30, 187, 19], by performing static code analysis [100, 210, 137, 135, 31, 125, 65] or dynamic analysis [103, 108, 114, 11, 116, 149, 22] of the code embedded in a Web page. Subsequently, these features were used to either create a filter or a machine learning model to detect malicious content on OSN. Models that used only OSN characteristics could not be relied upon as Twitter spammers quickly modify their behaviour in order to evade existing spam detection techniques [227] and subsequently OSN features that are required to detect them change with time [6]. This led to the development of models based on static and dynamic analysis, where the models focused on malware code/behaviour to detect malicious URLs on OSNs. However, these models failed against code/behaviour obfuscating techniques such as anti-symbolic execution obfuscation [10], anti-fuzzing [84], encryption [150], polymorphism [150] (see chapter 2 for detailed discussion of these models). Alternatively, security practitioners have deployed high interaction honeypots, particularly Capture HPC [161], to detect a drive-by download attack by observing changes made on the client's machine in terms of file, process, and registry to identify the URL as malicious or benign by referring to its exclusion list (see chapter 3 for details on Capture HPC and exclusion lists). However, there were limitations to this approach. A high interaction honeypot observes changes only to file, process and registry [180]. Thus in this chapter we propose a novel drive-by download detection model, that shifts the focus of detection from analysing malware code/behaviour through specific (discrete) file, process and registry activity to observing effect of malware to client's machine using (continuous) machine activity. To the best of our knowledge this is the first study where a detection model was built based on actual machine activity to detect drive-by download attacks on Twitter.

4.2.1 Contribution

The main contributions of this chapter are to develop evidence to answers to two research questions posed in Chapter 2 namely;

RQ2 *Is it possible to construct a model to detect drive-by download attacks that is easily reproducible and is independent of the malware code signature, by using system-based activity extracted from a honeypot?*

RQ3 *Which machine learning models are well suited for detecting drive-by download attacks, based on machine activity?*

Here we aim to develop a novel machine learning-based method that will be able to detect a drive-by download attack on the basis of system-level recorded machine activities. The underlying assumption here is that behaviour changes to the system are more difficult to manipulate than the hiding of the malicious code. The main contributions of this chapter are:

1. We developed a novel machine learning model based on system behaviour collected while visiting the websites that can distinguish between malicious and benign URLs within five minutes of the URL being clicked. Thus, shifting the focus to the analysis of system behaviour rather than analysis of code. This overcomes limitations of models built based on malware code, such as encryption, polymorphism, anti-symbolic execution obfuscation [10] or anti-fuzzing [84].

2. We examine and interpret the learned model and trace the relationship between machine activity and malicious behaviour exhibited by drive-by-downloads on Twitter.

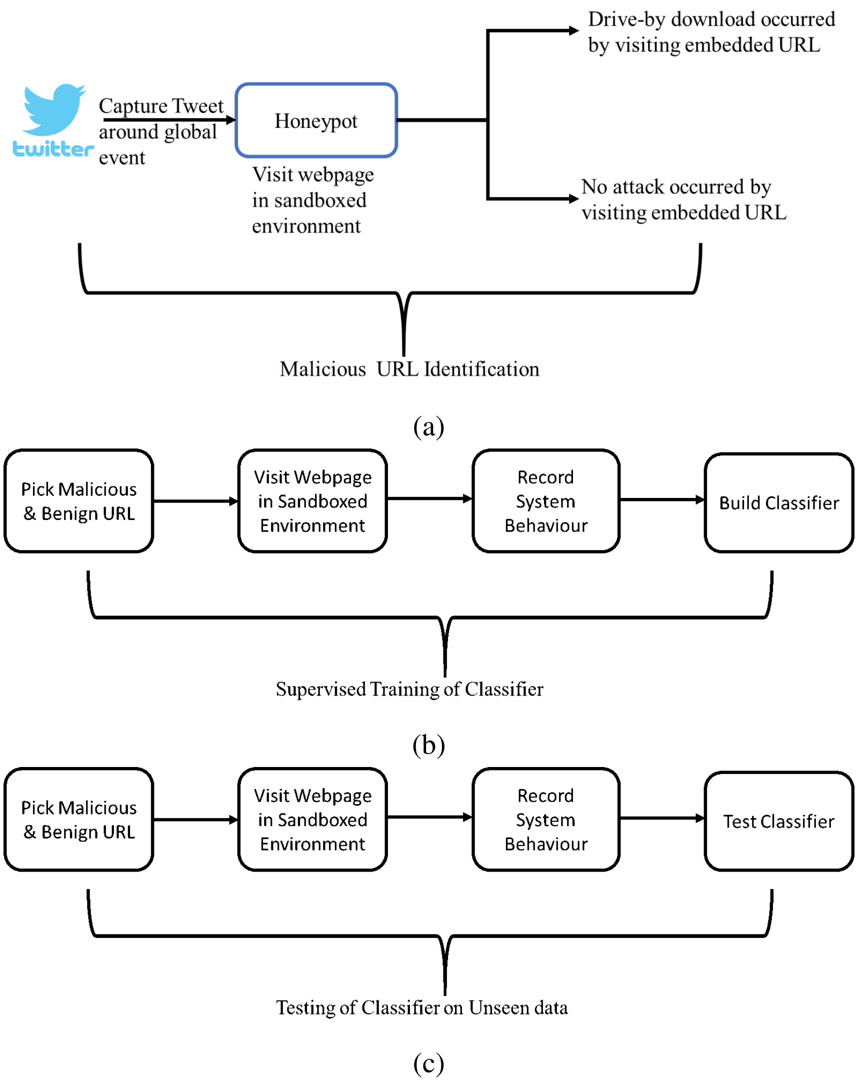

Capture Tweet around global event
Honeypot
Visit webpage in sandboxed environment
Drive-by download occurred by visiting embedded URL
No attack occurred by visiting embedded URL
Malicious URL Identification
(a)
Pick Malicious & Benign URL
Visit Webpage in Sandboxed Environment
Record System Behaviour
Build Classifier
Supervised Training of Classifier
(b)
Pick Malicious & Benign URL
Visit Webpage in Sandboxed Environment
Record System Behaviour
Test Classifier
Testing of Classifier on Unseen data
(c)

Figure 4.2: (a) Identifying malicious URLs which can carry out drive-by download attacks on Twitter (b) Revisiting malicious and benign URLs in a sandboxed environment to record system behaviour and build a classifier using the Weka toolkit (c) Testing the classifier on unseen data from a different sporting event

4.3 A Drive-by Download Attack Classifier

4.3.1 Data Annotation and Pre-processing

Figure 4.2 outlines the experimental setup that was used to build and validate the drive-by download classification model. The detection model was constructed in three stages, namely: (i) using a honeypot to annotate URLs into malicious/benign (ii) revisiting the URLs and observing behavioural changes made for the purposes of developing a machine learning model, (iii) validating the detection model.

The aim of the experimental setup is to build a machine classifier that can distinguish between malicious and benign URLs on the basis of the machine activities recorded over a period of five minutes. The 5-minute interval is currently a heuristic to ensure that a large number of sites can be visited – it makes the significant assumption that any malicious activity will be triggered within the first 5 minutes of the visit.

This was achieved by training machine classification models, from both generative and discriminative categories (see section 4.3.3 on building the model) using machine activity logs generated while interacting with URLs extracted from Twitter data that had been collected during two popular sporting events – Superbowl 2015 and the Cricket World Cup 2015. TThe experimental setup was divided into three parts. In the first part (see Figure 4.2a) URLs were classified into malicious or benign by Capture HPC (a detailed discussion on how classification is performed can be found in Chapter 3). The result of the first stage is an annotated list of URLs, where if a URL is classified as malicious it signifies a drive-by download attack occurred during its visitation, and a benign label signifies no attack. However, Capture HPC only observes changes made to the system by observing changes made to files, registry and process. We have introduced the novel concept of observing machine activity. Thus, in the second phase, an activity log was created at 30 second interval capturing all the relevant activities while interacting with known malicious and benign URLs (see section 4.3.3 for detail). Next, a range of machine learning models were developed as part of the investigation based on cutting edge approaches in literature. Two popular generative (NaiveBayes[179, 117, 68, 208, 110] and Baynet[110, 208, 176, 223]) and discriminative (Decision Tree[117, 68, 110, 176, 208] and Multi-Layer Perceptron[68, 178, 51]) classifiers were chosen to build the detection models (see section 4.3.3). The best performing method was then chosen for the detection model. In the final phase the classifier was tested on unseen data (data from a completely different sporting event) and the performance of the classifier was evaluated (see section 4.3.4).

4.3.2 Malicious URL Identification

For the study data was collected from Twitter via its programmatically accessible streaming API using Tweepy [166] and a total of 122,542 Tweets were collected for the Super Bowl final while 7,961 Tweets that contained a URL including retweets were captured for the cricket World Cup using event-specific hashtags (#CWC15, #RWC2015, #SB50, #SuperBowlSunday, #superbowlXLIXend). Figure 4.2a shows how a high interaction honeypot was set up to analyse the activity of all the URLs collected from the Twitter stream and to determine whether to annotate each of one them as either malicious or benign. For this purpose we used Capture HPC, a client-side honeypot[160] system, that interacted with each URL for period of 5 minutes. At the end of the interaction period Capture HPC determines if any system-level operations have occurred including file, process and registry changes made to the system. Based on these changes it classifies the URL as malicious or benign (for details regarding the configuration of Capture HPC, see Chapter 3)

The tasks that are summarised in Figure 4.2a are as follows:

1. Connect to the Twitter Streaming API and send the search term to collect on (#superbowlXLIX), specifying that only posts containing URLs should be returned. This returns Tweets as they are posted. Write and send the details of the Tweets to a database.
2. Expand the shortened URLs and remove duplicates.
3. For every 500 (new) URLs, upload a text file using a (clean) Capture HPC virtual machine.
4. Capture HPC iterates through the list, visiting each URL, and keeping the connection open for 5 minutes and at the same time, observes changes made on the client machine in terms of file, process, and registry to identify the URL as malicious or benign by referring to its exclusion list.

4.3.3 Building a Machine Classifier

From the collection of annotated Tweets made in the previous phase 2,000 Tweets were sampled containing unique URLs. The training data consisted of samples of machine activity at regular intervals of 30 seconds throughout the Super Bowl data collection period. The *training set* contained 1,000 URLs identified by Capture HPC as malicious, and 1,000 as benign. The unseen dataset used as the *test set* was collected around the time of the cricket World Cup; it consisted of 891 malicious URLs and 1,100 benign (sampling 80% from the day of the final and 10% from each of the semi-finals).

Feature Selection and Preprocessing

To identify features that were predictive of malicious behaviour machine log data was collected while revisiting the malicious and benign URLs in a sandboxed environment. The details of the metrics that we measured are as follows:

1. CPU usage (number).
2. Connection established/listening (yes/no).
3. Port Number (yes for port 80/no for other port).
4. Remote IP (established or not).
5. Network Interface (type e.g. Wifi, Eth0).
6. Bytes Sent (number).
7. Bytes received (number).
8. Packets Sent (number).
9. Packets Received (number).
10. Time since interaction started.

From the log file created only ten attributes from the recorded machine activities were considered to build the machines learning models. The attributes were categorised into two broad categories, one presenting the load on the machine, such as the CPU usage during the visitation of the website and other presenting network data. In the log file generated to build machine learning models, the value for CPU was represented as a numeric value representing CPU usage during visitation. While for attributes represented network statistics had to undergo a pre-processing stage before they could be written in the log file. In the pre-processing stage, attributes such as connection, ports, network interface, and remote IP were transformed into nominal value. Where for connection and remote IP a value *one* represented a presence of remote IP and connection established and *zero* represent an absence. Whereas for ports a binary value 1 represented the use of port 80 and 0 represented the usage of any other port other than 80. The rationale for focusing on port 80 was because it is the most commonly used for the internet communication protocol, Hypertext Transfer Protocol (HTTP). It is the port from which a computer sends and receives Web client-based communication and messages from a Web server and is used to send and receive HTML pages or data. For the network interface attribute, each number presented the network interface that was used while visiting the website. Once the data was transformed log file representing each attribute and their respective values at every 30-second interval were generated for both dataset.
In total a 5.5 million observations were recorded from interacting with 2,000 Tweets (1,000 malicious and 1000 benign). Each observation represented a feature vector containing metrics which indicated whether the URL was annotated by Capture HPC as malicious or not.

Classifier Model Selection

The data contained logs of machine activity, which occurred even when the system was idle, so it was likely that any log would contain a great deal of 'noise' as well as malicious behaviour. Table 4.1 presents a comparison between the training and testing datasets with respect to the mean and standard deviation of recorded machine activity. It illustrates the high variance in the mean recorded values of CPU usage, bytes/packets and sent/received used between the two datasets, which suggested that it would be challenging to identify similar measurements between datasets for prediction purposes. The standard deviation in both datasets was very similar, which suggested that the variance is common to both datasets, while the deviation is high, suggesting a great deal of ’noise’ in the data.

In addition to the ‘noise’ in the data – although the training and testing datasets contain

Table 4.1: Descriptive statistics for train and test datasets at T=60 for numeric attributes

Attribute	**Mean**		**Std. Dev**	
	Train	**Test**	**Train**	**Test**
Cpu	1.255354	6.26	2.144828	2.31
Connection	0.86	0.88	0.34	0.32
Portnumber	0	0.37	0.01	0.19
Remoteip	0.86	0.88	0.34	0.32
Network	4	4	2	2
Bytessent	1.01E+08	3.59E+08	2.06E+08	9.50E+08
Bytesrecd	2.87E+08	3.12E+08	8.47E+08	8.90E+08
Packetssent	470821.5	2442275	1472258	6659166
Packetsrecd	539358	2849133	1843365	7742467

a well-balanced number of malicious and benign activity logs – the behaviours in both logs are largely benign, creating a large skew in log activity towards the benign type. The noise and skewness may have an impact on the effectiveness of a discriminative classifier in identifying the decision boundaries in the space of inputs (i.e. the inputs may not be linearly separable, which could cause problems in using a perceptron-type classifier) even after great many iterations (for instance, if a multilayer perceptron were used that had been developed using multiple layers of logistic regression).

It could be argued that for more complex relationships, such as multiple sequential activities leading to a malicious machine exploit, a generative model would more appropriately generate a full probabilistic model for all the variables (possible behaviours), giving a training dataset of machine logs. For example, a Bayesian approach could effectively capture the dependencies between variables over time [190]. Or a Naive approach to Bayesian modelling might be more suitable in that it assumes that there are no dependencies, but that the probabilistic value of individual variables will be enough to determine the likely behaviour [115]. The first phase of data modelling was therefore to conduct a number of baseline experiments to determine which of the two models would predict more accurately. We used the Weka toolkit to compare the predictive accuracy of:

1. Generative models that consider conditional dependencies in the dataset (BayesNet) or assume conditional independence (Naive Bayes).
2. Discriminative models that aim to maximise information gain (the J48 Decision Tree) and build multiple models to map input to output via a number of connected nodes, even if the feature space is hard to linearly separate (Multi-layer Perceptron)

While developing each machine learning model using Weka toolkit, default configuration for each algorithm had been selected. The default setting for each are listed below

1. NaiveBayes- Batch Size=100, number of Decimal place =2
2. BayesNet-Batch size=100, Estimator Algorithm= Simple Estimator and Search Algorithm= K2-P 1-S Bayes
3. J48- Batch Size=100, confidence factor=0.25, min number of Obj=2, number of folds=3, unpruned=False.
4. MLP- Batch size=100, Decay=False, Hidden layers= (number of attributes+ classes)/2, Learning rate=0.3, Momentum=0.2, Training time =500, Validation threshold =20,Normalise Attribute=True, Normalise Numeric class=true.

4.3.4 Classifier Training and Testing Results

The results are presented using standard classification metrics, Precision (a measure of false positives), Recall (a measure of false negatives) and F-measure (a harmonised mean of P and R). We define these as follows:

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive}$$

Table 4.2: Bayesnet Result

BayesNet	**Precision**		**Recall**		**F-Measure**	
Time	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**
0	0.83	0.67	0.83	0.67	0.83	0.67
30	0.87	0.66	0.86	0.66	0.85	0.66
60	0.87	0.67	0.86	0.66	**0.86**	0.66
90	0.87	0.68	0.86	0.67	**0.86**	0.66
120	0.86	0.70	0.86	0.69	0.85	0.69
150	0.86	0.70	0.85	0.68	0.85	0.67
180	0.86	0.68	0.85	0.67	0.85	0.66
210	0.85	0.69	0.85	0.68	0.85	0.67
240	0.85	0.69	0.84	0.68	0.84	**0.68**
270	0.85	0.69	0.85	0.68	0.85	0.67

,

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative}$$

$$F - Measure = 2.\frac{Precision.Recall}{Precision + Recall}$$

The tables of results for two classifiers belonging to each type of model (generative and discriminative) is presented below. Each table presents the results while training and testing the model using machine activity logs split into incremental time windows (0-270 seconds) with aggregated log results. Note that t=0 is not actually 0 seconds but 5 seconds after the connection is opened. The classifiers that were used are BayesNet (BN), Naive Bayes (NB), J48 Decision Tree (DT) and Multi Layer Perceptron (MLP). It can be seen from the training performance data , that was performed using ten fold cross validation technique, (see Tables 4.2-4.5) that each model exhibits optimal performance between t=30 and t=60, with very little improvement after this time.

The low error rates at t=60 in models that consider conditional dependencies in the training phase suggest three things:

1. The features we are using to build the models are predictive of malicious behaviour.
2. Malicious activity probably occurs within the first 60 seconds of the interaction.
3. There are conditional dependencies between the measured variables. This is a logical result as we would expect certain machine activities to have some conditional

dependencies, for example, CPU usage and Bytes sent/received.

Table 4.3: Naive Bayes Results

Naïve	**Precision**		**Recall**		**F-Measure**	
Time	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**
0	0.51	0.50	0.51	0.45	0.42	0.40
30	0.59	0.51	0.62	0.50	**0.57**	0.49
60	0.59	0.52	0.62	0.51	**0.57**	0.51
90	0.58	0.60	0.62	0.55	**0.57**	0.50
120	0.58	0.62	0.61	0.57	0.56	0.54
150	0.57	0.64	0.61	0.57	0.55	0.54
180	0.58	0.63	0.60	0.56	0.54	0.52
210	0.58	0.64	0.60	0.58	0.54	0.54
240	0.53	0.63	0.55	0.53	0.47	0.53
270	0.58	0.63	0.56	0.59	0.55	**0.56**

Table 4.4: J48 Results

J48	**Precision**		**Recall**		**F-Measure**	
Time	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**
0	0.83	0.69	0.83	0.65	0.83	0.65
30	0.87	0.66	0.86	0.67	0.85	0.66
60	0.87	0.66	0.86	0.66	**0.86**	0.65
90	0.87	0.69	0.86	0.69	**0.86**	**0.69**
120	0.86	0.68	0.86	0.68	0.85	0.68
150	0.86	0.67	0.86	0.67	0.85	0.67
180	0.86	0.68	0.86	0.68	0.85	0.68
210	0.86	0.68	0.85	0.68	0.85	0.68
240	0.85	0.69	0.85	0.69	0.85	**0.69**
270	0.85	0.70	0.85	0.70	0.85	**0.69**

Table 4.5: MLP Results

MLP	**Precision**		**Recall**		**F-Measure**	
Time	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**
0	0.72	0.69	0.72	0.68	0.72	0.67
30	0.83	0.70	0.82	0.68	0.81	0.68
60	0.79	0.70	0.79	0.68	0.77	0.67
90	0.81	0.63	0.80	0.58	0.79	0.56
120	0.80	0.74	0.80	0.71	0.79	**0.71**
150	0.81	0.717	0.80	0.687	0.80	0.679
180	0.81	0.738	0.81	0.703	0.80	0.694
210	0.74	0.72	0.74	0.69	0.74	0.68
240	0.76	0.69	0.76	0.69	0.76	0.69
270	0.80	0.72	0.79	0.70	0.79	**0.70**

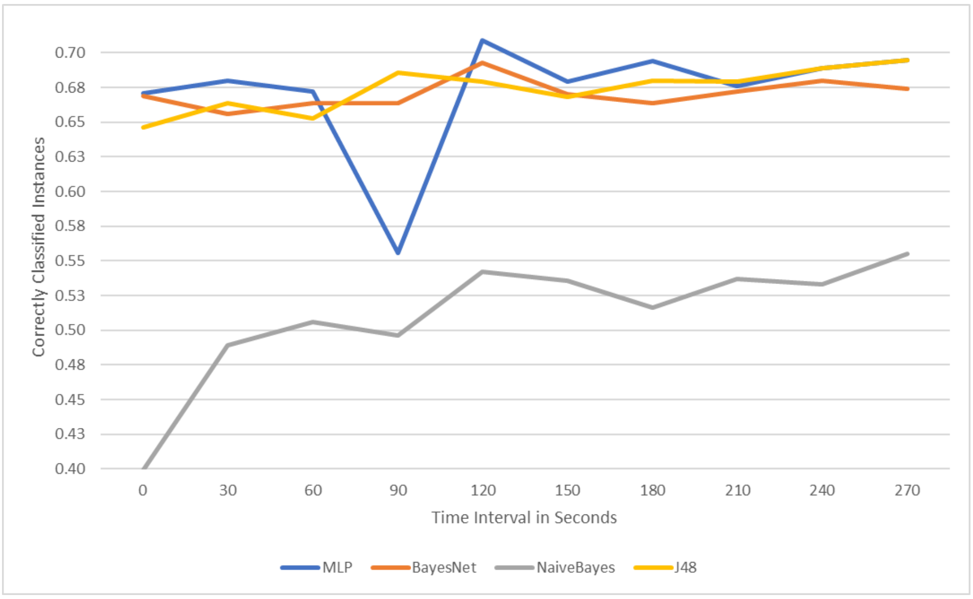


Figure 4.3: Classifier performance over time for test dataset

With reference to Figure 4.3, a chart of the correctly classified instances over time for the test dataset, the first point to note is that the model that does not consider dependencies between input variables (the NB model) performs much worse than the other models. This model improves over time, but takes until t=270 to reach peak performance, while the other models begin to plateau or decline in performance around t=120 (see Figure 4.3). The second point is that the discriminative models outperform the generative models, suggesting that there are distinct malicious activities that are linearly separable from benign behaviour. This means that over time and, most importantly, across different events, it is possible to monitor and measure specific machine behaviours that can be indicative of malicious activity when the URLs are clicked. Strong claims cannot be made about this, given that our optimal F-Measure performance was only 0.71 at time t=120 using the MLP model (see Table 4.5). However, it is encouraging to see that the MLP model exhibited a precision performance of 0.740, only slightly above its optimum level, at time t=30. This demonstrates the model's ability to reduce false positives fairly early on in the interaction.

Model Analysis

Table 4.7 presents the weightings assigned to each attribute used as a predictive feature in the MLP model. These results were extracted from the best performing iteration of the MLP model during the training phase (t=60) to examine how the model represents a learned weighting between features. The model produced 8 hidden nodes and the weighting given to each node for each class (malicious or benign) is shown in Table 4.6, with 4 of the 8 nodes having values above the threshold value, and nodes 3, 5 and 7 having high weightage towards a particular class. Node 1 stands out as the most discriminative positive weighted node for malicious URLs. If we look further into this (see Table 4.7) we can identify that the Bytes Sent variable has the highest weighting. If we compare Node 3, which is more heavily weighted towards the benign class, we can see that Bytes Sent/Received have similar weighting but that the Packets Sent/Received are negatively weighted in Node 3 and positively weighted in Node 8. This is an interesting finding, because Web endpoints will almost always send data to the machine visiting them. This model demonstrates that there are measurable 'norms' for the inflow of packets from Web pages and that there are measurable deviations from this that can predict malicious behaviour, as is happening in Nodes 3 and 8.

Table 4.6: Node by Weight Class

Inputs	**Weights (Benign)**	**Weights (Malicious)**
Threshold	1.99	-1.99
Node1	-11.31	11.31
Node2	2.45	-2.45
Node3	13.79	-13.79
Node4	-8.90	8.9
Node5	10.55	-10.55
Node6	-9.34	9.34
Node7	24.65	-24.65
Node8	-2.61	2.61

CPU has weightings higher than the threshold for the node in Nodes 1 which suggests that this is also a predictive feature.

Table 4.7: MLP Analysis

Attribute	Node 1	Node 2	Node 3	Node 4	Node 5	Node 6	Node 7	Node 8
Threshold	51.75	-7.47	44.34	59.95	-2.14	11.62	0.36	2.48
CPU	1.36	-86.36	-0.04	0.02	-0.39	0.20	-1.60	-60.51
Connection	0.14	-2.39	-0.01	0.08	0.04	-0.08	0.00	-0.03
port number	-56.20	12.19	-45.24	-60.90	2.16	-11.84	-0.28	-19.34
remoteIP	0.12	-2.37	0.02	0.16	-0.02	-0.06	0.00	-0.01
Network_Interface=1	-31.12	19.79	-8.82	-1.79	1.93	-11.74	1.92	-1.36
Network_Interface=2	-26.56	-9.39	-44.23	-59.96	2.17	-12.26	-0.44	6.58
Network_Interface=3	-26.23	-9.81	-43.65	-59.15	2.09	-12.21	-0.43	5.87
Network_Interface=4	-30.85	19.94	-8.19	-0.81	1.82	-11.30	2.06	-1.27
Network_Interface=5	-87.26	1.32	-63.24	-116.64	-1.23	13.03	-6.52	-27.81
Network_Interface=6	-31.01	24.48	-8.26	-0.82	1.83	-11.82	2.06	-1.02
Network_Interface=7	-25.73	-9.20	-45.22	-60.65	1.86	-12.09	-0.57	6.45
Bytes_Sent	484.93	9.01	484.23	593.43	50.37	14.02	22.54	62.36
Bytes_received	-62.91	1.21	-59.96	-12.24	140.94	91.83	-3.19	-10.23
Packets_Sent	-51.65	3.36	-49.40	-48.90	18.52	18.72	-2.20	8.15
Packets_received	-51.51	5.55	-48.18	-49.28	10.78	15.26	-0.91	5.15

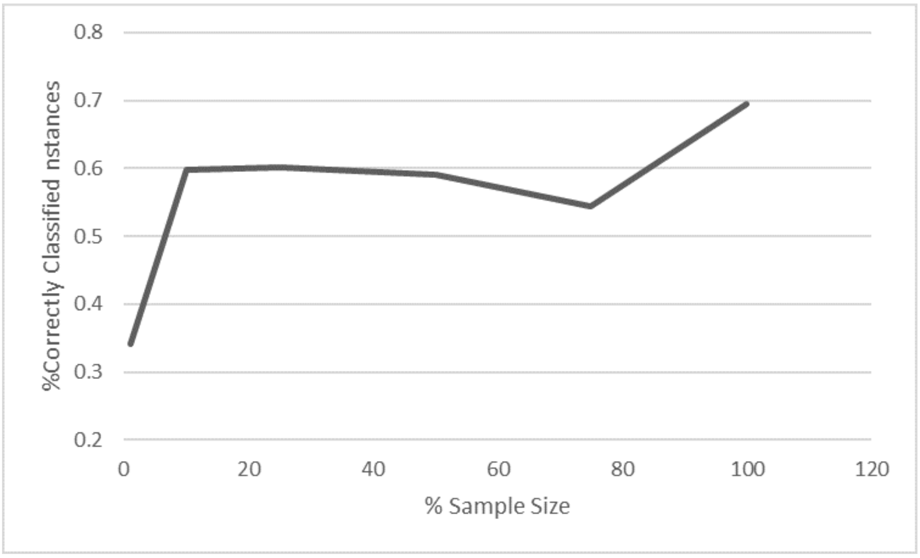


Figure 4.4: Correctly classified instances with sampled training data

Sampled Learning

Finally, we investigated how much training data was required to train the MLP model. Storing Twitter data around ongoing real-world events is not straightforward, given that events can last several weeks. If the system is deployed it could run on a daily basis to monitor malware and retrain learned models. Fewer data means less storage space and less computational time required to extract model features and run models. In addition, interacting with URLs is a time-intensive process. It may be asked whether the training set is missing a significant proportion of malicious activity, since not all URLs can be visited in real time with the relatively low level of computer resources available to academic researchers. Malicious endpoints are also frequently taken down and are no longer accessible. Thus, demonstrating that a small training sample achieves a similar performance as a full sample would, to some extent, alleviate the disadvantage of missing data, in that it would demonstrate that most of the explanatory features are present in smaller samples. To test this theory, we retained the full test dataset and sampled (using no replacement) from the training data at 1%, 10%, 25% and increments of 25% up to 100. Figure 4.4 illustrates the percentage of correctly classified instances with a 100% sample, down to 1% and shows that a sample of 10%, 25% and 50% yields a performance of 60%, which is 10% lower than the optimal performance of 70%.

4.3.5 Experiment Summary and Key Findings

Data were collected from Twitter, using event-specific hashtags to sample the Tweets posted at the time of two real-world sporting events. Having found evidence of drive-by download attack during other sporting events (see Chapter 3) we sought to develop a machine classifier that could classify a URL as malicious or benign within 5 minutes with classification being performed at every 30 second interval. We used machine activity log data for this purpose, such as CPU usage, network traffic and network connection statistics to discriminate between malicious and benign URLs. We aimed to explicate the relationship between machine activity and the malicious URLs posted to Twitter. Finally, given the large volumes of Tweets surrounding events and potential issues with sampling from all malicious sites due to their short lifespan and the time intensiveness of interacting with all of them, we aimed to understand the impact on classification performance when using much smaller samples.

We built a number of machine classifiers and identified that a Bayesian model, a Decision Tree and a Multi Layer Perceptron approach all worked extremely well during training, achieving over 0.70 in the F-Measure, up to 0.86 for the DT and 0.81 MLP. Of these, the discriminative models performed better than the generative models in testing, and the MLP model performed best overall with an F-Measure of up to 0.71 using previously unseen data (logs created during the cricket World Cup 2015). The Bayesian approach performed best in the early stages of the interaction, achieving an F-measure of 0.66 when the model had the least information available. The high training scores suggest that the features used are indeed predictive of malicious behaviour for a single event. The drop in performance on a new event suggested that the attack vectors were slightly different across events, but with a reasonably high degree of F-measure some independence between predictive features and events could be claimed. Upon inspecting the decision-making process within the MLP model, evidence was identified to suggest that the key predictive machine activity metric was network activity – particularly packets sent and received. CPU use and process IDs also had a clearly raised and correlated weighting in the model, as did the bytes sent from the network when correlating with new connections to remote endpoints, suggesting that data ex-filtration exercises could be distinguished from general data transfer.

On data sampling, a learning curve produced using a range of samples from the training dataset, while still being tested on the full testing dataset, revealed only a drop (10%, 25%, 50% of sample = 60% and 100% of sample yields a performance of 70%) in classification performance below that of the full training sample. This suggested that machine log data can be predictive of malicious system behaviour even with small samples, alleviating some concerns over the appropriate sampling mechanisms, the lack of a complete log of all Twitter URL activity, and the requirement for large amounts of data storage.

4.4 Conclusion

As Online Social Networks (OSNs) become a crucial source of information publication and propagation following global events, an environment has been created that is particularly vulnerable to cyber attacks via the injection of shortened URLs that take the user to a malicious server launching a 'drive-by download' attack on the local machine. In this chapter, we have sought to combat the problem by building on a body of work that has developed methods to identify malicious URLs in OSNs. Existing work has developed methods to provide evidence that OSN accounts or URLs may be malicious, which is helpful; but given the frequency and volume at which new accounts emerge, the only way to determine that actual malicious behaviour is occurring is to observe it. The existing literature encourages users to classify URLs by means of all the data generated through observing malware code/behaviour. This provides a post-hoc result, without actually observing the malicious activity; it makes a decision on the basis of previously seen malware signature. However, these approaches relied on obsolete or published blacklist of users no longer in the network, and on statically or dynamically analysing the malware code that can easily evade detection by using encryption or polymorphism.

The proposed classification model presented as a result of our experiment observes machine activities during visitation of a Web page instead of statically/dynamically analysing malware code. It was trained by using data collected over popular sporting events from Twitter for its training and testing. None of the methods published before the present study allowed us to observe malicious activity and classify an URL as malicious. The main focus of our research was therefore to develop a method capable of identifying a URL as malicious or benign on the basis of the machine activity metrics generated, which were logged during interaction with a URL endpoint. The model gave a F-measure 0.81 for MLP during training the model and a F-Measure of 0.71 during testing the model on unseen dataset. As the detection model was built by observing system activities at every 30 second interval, it created a series of time bounded events that showed different system state changes, from its benign state (when the URL is clicked), to attacking state (when the vulnerability is exploited) and to infection state (when the system is infected). This bring us to the next question, can we use these system state changes to predict an attack based on initial machine activities thus proactively blocking and preventing an attack, rather than reacting and repairing at a later date ?

Chapter 5

Prediction of drive-by Downloads on Twitter

Current work on malware detection models on Twitter have used static/dynamic techniques and analysed OSN account characteristics to detect malware, where, the focus has been on ***detection****. In this chapter we answer research questions like: is it possible to predict a drive-by download attack on Twitter?; and do social features contribute to the prediction of drive-by download attack on Twitter? A machine learning model using machine activity data and tweet metadata was built to move beyond post-execution classification of such URLs as malicious, to predict if a URL will be malicious with 0.99 F-measure (using 10-fold cross-validation) and 0.833 (using an unseen test set) at 1 second into the interaction with the URL. Thus providing a basis from which to kill the connection to the server before an attack has completed and proactively blocking and preventing an attack, rather than reacting and repairing at a later date.*

5.1 Introduction

In Chapter 3 an association between drive-by download attacks and popular events that attract millions of user on Twitter was identified. The evidence was gathered by analysing tweets posted during popular sporting events. Sporting events were particularly chosen due to high volume of users they attract. For example, the England versus Iceland football match at the European Football Championships (Euro 2016) was one of the most tweeted about events of 2016 - attracting 2.1 million users[167]. The more popular OSNs become, the more attractive a platform they become for cybercriminals to conduct their attacks [231]. Current research has broadly investigated the problem of detecting these drive-by download attacks on Twitter from a number of perspectives including: (i) characteristics of OSN user accounts (e.g. posting behaviours [32] and social network links [226]); (ii) characteristics of URLs (e.g. lexical features [135] and endpoint activity [127, 128]); and (iii) analysing the code of a Web page in a static or dynamic manner to study its intended or actual behaviour when interacting with the underlying system on which the OSN user is accessing the Web page [167]. However, the models developed focused on malware code and failed against code obfuscating techniques such as encryption, polymorphism etc.

In Chapter 4, a novel malware detection model was proposed focusing on system behaviour rather than malware code signatures, recording system-level machine activity for five minutes to capture behavioural interactions with Web servers [23]. This was used to build a machine classifier that was able to distinguish between malicious and benign URLs with an F-measure of 0.71 when the model was tested on an unseen dataset. The main contribution in previous chapter was to build a machine classifier to *classify a URL at the end of a 5 minute interaction.*

In this chapter we extend the work by adding more behavioural features to improve classifier performance on unseen data across different events, and reducing the classification period to 10 seconds *to predict a drive-by download attack based on early-stage machine activities observed before the attack is complete*. The choice of adding more features, was taken to increase the diversity of data and incorporate features that may change over time to test adaptability as cyber criminals change attack behaviour over time. In addition to increasing the number of features, observation time was also reduced to a maximum 10 seconds and snapshots of machine activity were taken at each second, instead of every 10 seconds - meaning the predictive model could make a second by second prediction (at 1,2,3,4...10 seconds) to identify and remove malicious URLs from the network as early as possible. This not only reduces the amount of time the user's system is exposed to the malicious URL, but also increases the URL processing capacity of the model. As a result of the work in this chapter, a single deployment of the predictive model is now capable of daily processing a minimum of 15,000 (with full 10 second observation time) and a maximum of 150,000 (with predicting at 1 second) URLs compared to 500 URLs (with 5 minutes observation time) for Capture HPC.

Features are derived by capturing machine activity metrics (e.g. CPU use, RAM use, Network I/O etc) as per Chapter 4, and we now introduce tweet attributes. Through the developments in this chapter, the prediction model is able to predict whether the URL is pointing to a malicious Web page with 0.99 F-measure (using 10-fold cross-validation) and 0.833 F-measure (using an unseen test set) at *1 second into the interaction with a URL*. This provides a novel contribution with which it is possible to kill the connection to the server before an attack has completed - thus proactively blocking and preventing an attack, rather than reacting and repairing at a later date. To the best of our knowledge, this is the first study to proactively predict a drive-by download attack by classifying a URL during interaction, rather than requiring the malicious payload to complete before classification.

5.2 Background

Twitter has been used to carry out a broad range of cyber attacks. For instance, in 2015 the US Pentagon's email servers were targeted by Russian hackers using Twitter [165]. Cybercriminals have targeted popular people who have a large number of followers to propagate malware or spam by hacking their accounts, for instance, Twitter's CFO Anthony Noto [18] and former Apple Macintosh evangelist Guy Kawasaki [138]. In a survey conducted by SANS Institute to identify the most frequent methods employed by cybercriminals to launch cyber attacks on organisations, it was shown that drive-by downloads accounted for 48% of attacks by exploiting Web-based vulnerabilities [175]. Such cyber attacks could also be used as an entry point to carry out more wide-spreading attacks such as Ransomware - for instance, a CryptoLocker attack that originated from a drive-by download attack locked down a small city in Washington, USA for four days [122].

Current work related to the topic of detecting malicious content in Online Social Networks is divided in two parts, in the first detection models are built using static and dynamic analysis. For static analysis OSN user account, URL characteristics, and malware code were analysed to detect malware on OSN. Whereas, in dynamic analysis the behaviour of the malware or its effect on client machine was observed by executing it.Table 5.1 provides a summary of related work and the methods used at a high level for comparison.

To date the focus has been on *detecting malware* on OSNs by using either static or

Table 5.1: Malware or Spam detection techniques used

Techniques used to detect Spam/Malware on Twitter						
Methods Used by Researchers	**Tweet Attributes**	**Blacklist cross check**	**Lexical analysis of URL**	**HoneyPot or Honey profiles**	**Machine Behaviour (Network, File,Process, Memory,CPU etc)**	**User Behaviour on Twitter**
OSN Account Characteristics[129] [32] [146] [39] [192] [14] [83] [228] [226][50]	✓	✓		✓		✓
URL characteristics[127][128][135]		✓	✓			
Detect By analysing Static Code[137][31] [109]		✓	✓			
Detect By analysing Dynamic Code[49][114][103][222] [11][25][33]	✓				✓(network only)	
Our Model	✓	✓		✓	✓	

dynamic techniques. However, considering 335 million active users [118] post at an average rate of 6,000 tweets per second [145], with the highest retweet count of 4.1 million recorded for a single tweet [130], a drive-by download attack has the potential to expose malware to millions of users before a malware detection model detects it. For example, a cyber attack that exploited a vulnerability in TweetDeck [204], a social media dashboard for managing Twitter accounts, infected 84,700 Twitter accounts in a matter of hours and it even infected the BBC News account exposing the malware to its 10.1 million followers [79]. In such a volatile environment, we need to adopt a proactive strategy to defend against malware attacks, such as anticipating a cyber attack and subsequently devising a means to counter it.

Thus, in this chapter we focus on *prediction*, proposing a novel model that can classify a URL into malicious or benign at *every second* based on the combination of URL and OSN account attributes and also dynamic machine behaviour - activity observed when the URL is clicked, and the Web page is being loaded. The aim is to predict that behaviour observed in the early stages of loading a Web page is likely to lead to malicious activity at a later stage - providing new capability for a user to block the completion of the malicious actions rather than depend on detection and repair at a significant cost and inconvenience.

5.2.1 Contribution

In this chapter we aim to develop a novel machine learning-based method that will be able to predict a drive-by download attack at every one second interval, on the basis of system-level recorded machine activities and URL/OSN account attributes. The contributions of this chapter develops evidence to provide answers to two research questions that were identified in Chapter 2:

RQ4 *Is it possible to predict drive-by download attacks on Twitter by observing machine activities?*

RQ5 *Do social features that have been shown to help with identifying abnormal accounts (posting spam/malicious URL), contribute to the prediction of drive-by download attacks on Twitter?*

The contributions can be summarised as:

1. A novel predictive machine learning model based on OSN account attributes and system behaviour observed while visiting websites, which can predict a drive-by download attack at every second, with maximum prediction time of 10 seconds.

Research question 4 will be answered in Section 5.3 and 5.4 where the architecture of the predictive model is introduced and data is collected from Twitter around sporting events to train and test the model. Research question 5 will be answered in section 5.4.2.

5.3 Experimental Setup

5.3.1 Data Collection and Annotation

Based on the evidence gathered regarding popular events being used to deliver drive-by download attacks in Chapter 3, two popular sporting events were chosen for training and validating the model. Behind the decision to select two events was the desire to determine whether our classification model would generalise beyond a single event and be applicable for use on URLs posted around other events. However, data collected so far could not be re-used to construct the predictive model proposed in this Chapter because of the reduction in observation time and an increase in number of features required to build the predictive model. Data collected so far recorded only 8 Tweet attributes and log files that were generated for building detection model were generated at 30 second intervals, much less granular than the proposed 1 second intervals we now aimed to study. Furthermore, it was not possible to use the older URLs to recreate the log files at shorter intervals since most of the URLs that were identified as malicious had been taken down.

The European Football Championships (#Euro2016) and the Olympics (#Rio2016) in 2016 were identified for training and testing the predictive model. Both events generated some of the largest volumes of Tweets in 2016 [118], because of which were considered appropriate for the experiment. Tweets containing a URL and hashtags relating to these events were captured via the Twitter streaming API, similar to ones described in Chapter 3. For Euro 2016 we captured Tweets from the period of 10 June to 14 July 2016 using the hashtag #Euro2016. We harvested 3,154,605 Tweets that contained a URL. During the opening ceremony that marked the opening of the Olympics in 2016 (the peak of public interest), we captured 148,881 Tweets that contained a URL using the hashtag #Rio2016. From the captured Tweets we randomly created a sample of 7,500 unique Tweets to identify 975 malicious URLs for the European Football Championships dataset; in addition, around 5,000 Tweets were randomly chosen. Using a high interaction client side honeypot, we gathered 525 unique malicious Tweets for the Olympics 2016 dataset. As in our previous experiments, Capture HPC was used as the high interaction honeypot, with an updated exclusion list.

The process of updating each exclusion list was conducted every 14 days to reflect the most recent actions that had been observed in drive-by download attacks. These exclusion lists were created by formalising rules when visiting malicious or benign Web pages. A URL is classified as malicious if during a visit to a website a system performs a certain activity or activities that violate the rules (see Chapter 3 for details of the exclusion list). As in previous experiments, Capture HPC gives us a label that we can use for supervised learning and a set of activity logs that we can use to train a system to recognise the 'early warning signals' that are present before the exclusion list flag is raised. However, the reliance on Capture HPC to provide us with a labelled data set for training our model is a limitation of our predictive model, in that, if the URL behaviour varies beyond what has been previously flagged as malicious, it will not create a malicious label for the URL. However, there are millions of flagged malicious URLs made available every day online for continuously updating Capture HPC's exclusion lists, so this limitation can be mitigated through regular updates.

5.3.2 Architecture of the Predictive Model

Even though the malware detection model that was proposed in Chapter 4 addressed limitations observed in previous detection models (discussed in Chapter 2), such as the focus on malware code signatures. However, due to the observation time of 5 minutes limited the processing capability of the model to 500 URLs per day. This is a limitation to scaling the model without multiple parallel models. When the observation time was reduced to 10 seconds, we saw a drop in performance - the F-measure improved with time in the detection model for both training and testing sample. This poses the question, *is it possible to make the predictive model by reducing the observation time without compromising on its performance?*

The architecture of the predictive model is laid out in Figure 5.1, where the predictive

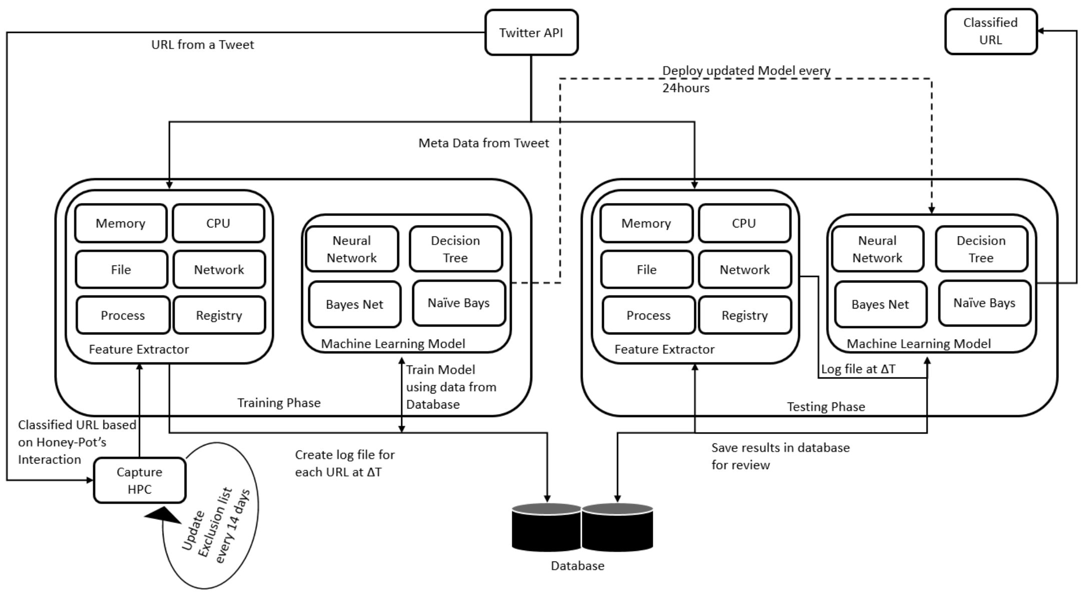


Figure 5.1: Architecture of Predictive Model

model has three main components (see Figure 5.1): *feature extraction*, *persistent storage* and *machine learning*.

Feature Extraction

The main function of feature extraction is to create a timeline of measurable observations on the client system on the basis of machine activity and Tweet attributes from the time a URL is opened to the point at which a drive-by download is carried out, or the system becomes idle. The feature extractor opens each URL that is passed to it in a sandbox environment and starts creating snapshots of machine activity at time intervals *'t'* for a period of *'p'*. For our experiment, *t=1* second and the observation period is defined as *p=10* seconds. The first snapshot is generated when a URL is 'clicked' at *t=1* second, and then subsequently at intervals of t. Each snapshot is written to a database for the sake of persistence, since the sandbox environment is wiped clean after each URL has been visited. Each database insert includes (i) metadata of the Tweet containing the URL and (ii) machine activity. Building on existing literature, features such as number of friends, followers, favourite count, retweet count, verification of user, etc that have proven effective in detecting malware on OSN were extracted from a tweets meta data. These features were extracted from two perspectives, first from the user that is posting the tweet and the other on the person that was retweeting or sharing the tweet to a larger audience.

1. Details pertaining to person tweeting
    - User name.
    - Screen name.
    - Number of followers.
    - Number of Friends (people following back the user).
    - Favourite count.
    - Geographical location in terms of latitude and longitude of the place where the tweet originated.
    - Verification of user.
    - Default Language set for user.
    - Time zone.
    - Location by name.
2. Details about the account.
    - Age of the account - calculated from the day of the event.
3. Details about people retweeting a tweet.
    - Retweet favourite tweet count.
    - Retweet retweet count.
    - Retweet user followers count.
    - Retweet user favourites count.
    - Retweet user friends count.
    - Retweet user location.
    - Retweet user timezone.

- Retweet user name.
- Retweet user screen name.
- Retweet user verified.

For machine activity, we log the metrics including file, process, CPU usage, network activity and memory changes. The number of observational parameters originally used to detect a malware while visiting a Web page were increased from 10 to 54 to enrich the training data for better performance and to observe system changes from a social content and machine activity perspective. Also, by including a more diverse set of features, it is possible the model will be able to observe and identify relationships among different features, even if in future the cyber criminal start using different attacking strategies. A complete list of features are mentioned below.

1. Activities related to File changes
   - File created.
   - Files modified.
   - Files deleted.
   - Files moved.
   - Source Path of the file.
2. Activities related to Process changes
   - Process Name.
   - Process executable path.
   - Process status.
   - Process created by user.
   - Process creation time.
3. Activities related to Central Processing Unit

  - CPU usage in percentage.
  - User CPU time -Amount of time the processor worked on a specific program.
  - System CPU time- Amount of time the processor worked on operating system's functions connected to a specific program.

4. Activities related to Network changes
   - Port Number
   - Bytes sent out.
   - Bytes received.
   - Packets sent.
   - Packets received.
   - Error in sending messages.
   - Error in receiving.
   - Packets drop while receiving.
   - Packets drop while sending.
5. Activities related to memory changes.
   - Virtual memory total.
   - Virtual memory available.
   - Virtual memory percentage.
   - Virtual memory used.
   - Virtual memory free.
   - Swap memory total available.
   - Swap memory used.
   - Swap memory free.
   - Swap memory percentage.

- Swap memory swap-in.
- Swap memory swap-out.
- Disk memory total available.
- Disk memory used.
- Disk memory free.
- Disk memory percent.
- Disk input output counter read count.
- Disk input output counter write count.
- Disk input output counter read bytes.
- Disk input output counter write bytes.
- Disk input output counter read times.
- Disk input output counter write times.

Generating Input File for Machine Learning Models

A log file containing both social information extracted from a tweet and machine activities recorded during visitation of a Web page was created for each URL classified as malicious and benign. A program was created to read the data from all logfiles' (containing both social attribute and machine activities for each URL), to combine them together to generate an *arff* file that was later used to build the machine learning models. While reading from the log file containing information regarding social attributes, data for those attributes that contained string/identifier values, such as location, default language, user verified, geographical location, etc. were transformed into nominal values for the logfile created as input to the Weka toolkit. A numeric nominal value was mapped to the default language of the user and data recorded for this field was transformed in the following way.

1. English-1

2. Chinese-2
3. Hindi-3
4. Spanish-4
5. For any other categories-5

Where nominal value to top four languages spoken in the world [198] were given a number between 1 and 4 and for any other language a value 5 was assigned. For the location, eleven categories were created to represent countries that have the biggest number of twitter users. The nominal values assigned to these were:-

1. USA-1
2. Japan-2
3. United Kingdom -3
4. Saudi Arabia-4
5. Turkey-5
6. Brazil-6
7. India-7
8. Mexico-8
9. Indonesia-9
10. Spain-10
11. Other-11

The geographical location in terms of latitude and longitude was transformed into a nominal attribute that contained a value '1' for coordinates present and '0' representing absence of coordinates. For the user and the retweet user's time zone, a value '1' meant the presence of a time zone value and value '0' represented absence of a value in this field. Similarly, for verification of the user's and the retweet user's account, nominal attributes containing binary values were created. Where, '1' meant that the user's/retweeter's account had been verified by Twitter and '0' meant the account was not verified. Attributes like the user's/retweet user's username was transformed into nominal values and these could have 4 possible value (1,2,3,4). Where a value '1' was assigned if the user's/re-tweeter's username had previously (from data collected in previous events) been used to post only malicious tweets. A value '2' was assigned if it was previously seen posting only benign URLs. A value '3' was assigned if it was used to post both malicious and benign dataset. A value '4' was assigned if the username was not found in the previously stored database. Similar transformation technique was used to the user's/re-tweeter's screen-name and the attribute could have four possible values (1,2,3,4). Furthermore, attributes representing fields such as the number of friends, followers, retweet count, favourite count were not transformed, and the actual numerical value that was captured was set. For example, if an account that posted the tweet had 40 followers, then the attribute representing the number of followers would contain the value 40. Age of the account was calculated using the date when the account was created and the date when the event took place. It represented the age of the account and contained a numeric value representing the number of days that have passed since the account was created.

A similar transformation was done on attributes that contained string values or identifier information while reading data from the log file containing machine activities and writing it into an *'Arff'* file. For attributes explaining file changes, an attribute *file type* was created that had four nominal values (created, modified, deleted, moved) representing the action done on a file. The attributes relating to the path, such as the source path of the file and the executable process path were categorised as nominal and could have three possible value (1,2,3). Value '1' was set if the process started from *System32* folder, '2' was set if it was executed from *Program Files* folder and for everything else a value '3' was set. A total of 32 process names that occurred in 99% of the dataset (training and testing) were identified, and a numeric nominal value was given to each process name (see table 5.2). Furthermore, a nominal value of 33 was given to all process names that

Table 5.2: Nominal value for Process Name

Process Name	**Nominal Value**
Svchost	1
Csrss	2
Iexplore	3
Searchprotocolhost	4
Audiodg	5
Cmd	6
Conhost	7
Dwm	8
Explorer	9
Lsass	10
Lsm	11
Python	12
Searchindexer	13
Services	14
Smss	15
Spoolsv	16
System	17
System Idle Process	18
Taskhost	19
Wininit	20
Winlogon	21
Wmpnetwk	22
Searchfilterhost	23
Wmiprvse	24
Dllhost	25
Userinit	26
Taskeng	27
Sppsvc	28
Wmpnscfg	29
Rundll32	30
Mobsync	31
Wsqmcons	32
OtherProcess	33

were not in the list of 32 identified process names. Process status was set as nominal, with '1' representing process running and '0' for any other value set to this field.

Similarly, an attribute representing the user that started the process was created that could have two values (0,1). Where a value '1' represented the process started by the system and

0 represented that the user had started it. The attribute process creates time stores the difference between the time at which the process was started and the machine started. For a port number, a nominal value was created where a value '1' is assigned if the port number has value *80* else '0' was assigned. Furthermore, attributes such as CPU usage, User CPU time, System CPU time, Bytes sent out, Bytes received, Error in sending messages, Error in receiving, Packets drop while receiving, Packets drop while sending, Virtual memory total, Virtual memory available, Virtual memory percentage, Virtual memory used, Virtual memory free, Swap memory total available, Swap memory used, Swap memory free, Swap memory percentage, Swap memory swap-in, Swap memory swap-out, Disk memory total available, Disk memory used, Disk memory free, Disk memory percent, Disk input-output counter read times, Disk input-output counter write times were kept as numeric fields, and they represented the actual values that were captured. Table 5.3 presents a comparison between the training and testing datasets with respect to the mean and standard deviation of recorded numerical value of both attributes representing machine activities and social features.

While recording machine activity we defined peak activities as the maximum number of activities observed while visiting the website in the given 10-second window, and irregular activities as a set of activities that occur when a machine is infected. These irregular activities are activities not observed while visiting a website as defined in the exclusion list (see Chapter 3 for the creation of an exclusion list and rules used to define malicious and benign behaviour). We also used attributes derived from metadata on the Tweet, including username, user screen name, user id, follower count, friends count and age of account. This produced a set of attributes every second for a period of *'p'*.

Persistence Storage

This component is responsible for storing information captured in the snapshots each second. Each snapshot contains information about tweet attributes that were extracted from the tweet meta data and machine activities that were observed during visitation of

Table 5.3: Descriptive statistics for train and test datasets at T=10

Statistics for train and test datasets at T=10				
Attributes	**Train**		**Test**	
Social	Mean	Std Dev	Mean	Std Dev
User Followers Count	47332.94	377276.25	21074.805	134614.78
User Friends Count	1905.27	7415.42	1874.259	12691.045
Age of the Account	1393.62	845.97	1569.297	898.162
Retweet Favourite Tweet	16.35	122.42	59.592	295.427
Retweet Count	30.45	538.31	31.982	134.885
Retweet User Followers Count	210673.32	1275254.77	434220.734	2575403.175
Retweet User Favourites Count	1263.86	6176.86	1967.987	8401.988
Retweet User Friends Count	2536.99	25326.51	7726.806	59492.778
Machine				
CPU Time User	0.134	0.184	0.119	0.167
CPU Time System	1.177	4.066	1.142	3.785
Memory Percent	0.249	0.196	0.225	0.179
CPU	71.25	29.319	94.935	9.702
Bytes Sent	51044.89	14405.899	39970.356	4903.951
Bytes Received	273618.84	350108.233	206786.318	133748.574
Packets Sent	147.78	164.66	86.361	38.861
Process Create Time	2.4	1.2	2.3	1.1
Packets Received	325.05	296.65	249.33	118.675
Virtual Memory Available	3636543218	39928656.57	3682304035	28862771.89
Virtual Memory Percent	20.23	0.875	19.224	0.632
Virtual Memory Used	922198286.3	39928565.57	876437469.4	28862771.89
Virtual Memory Free	3636543218	39928656.57	3682304035	28862771.89
Swap Memory Used	946574660.1	40582773.59	900921630.9	28601859.81
Swap Memory Free	8169003708	40582773.59	8214656737	28601859.81
Swap Memory Percentage	10.384	0.447	9.884	0.315
Disk Memory Used	26911543580	2708262348	30369623824	1815692572
Disk Memory Free	6723624676	2708262348	3265544432	1815692572
Disk Memory Percent	80.01	8.053	90.291	5.398
Disk IO Counter Read Count	5377.316	648.72	5082.231	430.637
Disk IO Counter Write Count	528.373	142.885	447.443	122.634
Disk IO Counter Read Bytes	218401106.3	34698593.93	210012597.6	24258661.5
Disk IO Counter Write Bytes	15711208.5	1747620.21	14883931.16	1461488.181
Disk IO Counter Read Times	5816878.119	4772238.097	5216764.111	2370010.96
Disk IO Counter Write Times	40668.949	45238.507	37052.043	35896.811

a Web page. Also, during the training phase we knew whether a URL was malicious or benign on the basis of the results from Capture HPC (see Chapter 3 for the configuration of Capture). This label is inserted into the database with each snapshot. Once the observation time is complete, the sandbox environment is reset to a malware-free state so that each new URL can be opened in a known malware-free configuration with a consistent baseline and new information can be stored.

Machine Learning

The third component is the machine learning phase. For the predictive model four different machine algorithms were trained to determine the best method of class prediction using these data. As in previous experiment the Weka toolkit was used to compare the predictive accuracy of (i) generative models that consider conditional dependencies in the dataset (BayesNet) or assume conditional independence (Naive Bayes), and (ii) discriminative models that aim to maximise information gain (the J48 Decision Tree) and build multiple models to map input to output via a number of connected nodes, even if the feature space is hard to linearly separate (Multi-layer Perceptron). To test the models, we used the feature extractor and the learned machine learning model from the training phase. Tweets from the testing dataset (in the first instance using 10-fold cross validation, and later using a holdout testing dataset) were passed into the feature extractor, which opened the URL in the sandbox environment and created the machine activity and Tweet meta-data snapshots at every timed point. Each snapshot was passed on to the learned model which classified the snapshot as malicious or benign. If the result was benign, the process continued to the next snapshot. The first time the outcome was malicious, the process stopped and the URL was classified as malicious, killing the connection to the Web page.

5.3.3 Relationship between features attributes and its class

To understand which attributes were contributing to the classification, we did the Pearson's correlation analysis (see table 5.4) to identify those attributes that were contributing the most. Six attributes had the Pearson's correlation coefficient higher than 0.10. Five of those were related machine activities, where process create time (time difference between

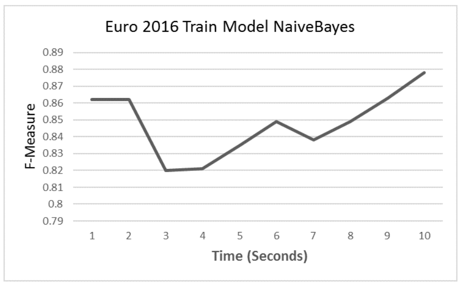
Table 5.4: Feature Selection of Attributes using Pearson's R Correlation between attributes and its class (Malicious)

Sr No	**Pearson's Correlation Coefficient**	**Attribute Name**	**Sr No**	**Pearson's Correlation Coefficient**	**Attribute Name**
1	0.35960	Process Create Time	31	0.01495	User Time Zone
2	0.26686	Disk Memory Used	32	0.01435	Retweet User Verified
3	0.26686	Disk Memory Free	33	0.01222	Disk I/O Counter Read Times
4	0.26682	Disk Memory Percent	34	0.01056	Retweet User Favourites Count
5	0.20133	Disk I/O Counter Write Bytes	35	0.01007	Disk I/O Counter Read Count
6	0.11725	User Verified	36	0.00997	Disk I/O Counter Write Times
7	0.08071	Bytes Received	37	0.00984	Retweet User ID
8	0.07416	Packets Received	38	0.00821	CPU
9	0.06412	User Friends Count	39	0.00744	Retweet User Friends Count
10	0.05818	Packets Sent	40	0.00696	User Language
11	0.05411	User-name	41	0.00438	User Coordinates
12	0.04440	Retweet User Name	42	0.00349	Process Using Network Yes/No
13	0.03877	Virtual Memory Percent	43	0.00212	Action Done on File
14	0.03806	Virtual Memory Used	44	0.00184	Process Status
15	0.03806	Virtual Memory Available	45	0.00180	Retweet Favourite Tweet Count
16	0.03806	Virtual Memory Free	46	0.00145	CPU Time User
17	0.03529	User Screen Name	47	0.00135	Source Path
18	0.03325	Retweet Count	48	0.00114	Process Name
19	0.03038	Disk I/o Counter Read Bytes	49	0.00089	Connection Establish Listen
20	0.03010	Disk I/O Counter Write Count	50	0.00089	Remote IP
21	0.02942	User Followers Count	51	0.00080	Process Username
22	0.02464	Swap Memory Used	52	0.00067	CPU Time System
23	0.02464	Swap Memory Free	53	0.00064	Process Path
24	0.02408	Swap Memory Percentage	54	0.00057	Memory Percent
25	0.02368	Bytes Sent	55	0.00028	Command Line Statement
26	0.02234	Retweet User Screen Name	56	0.00028	Process Executable Path
27	0.02233	User_location	57	0.00024	Retweet User Followers Count
28	0.01596	Age of account	58	0.00009	Port Number
29	0.01571	Retweet User Location	59	0.00000	Swap Memory Swap In
30	0.01554	Retweet User Timezone	60	0.00000	Disk Memory Total

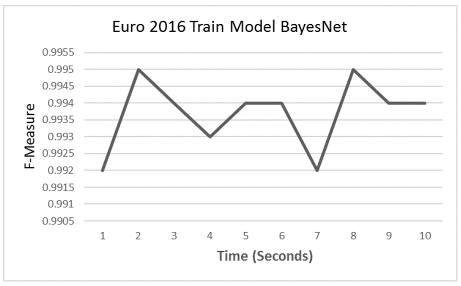
the process created and time the machine was booted) contributed the most to the malicious classification. The other attributes corresponding to machine activities were related to disk memory (used, free and percentage) and disk I/O counter writes signifying a distinction between malicious and benign activities while visiting a website. One rationale could be that during visiting of malicious web-servers, malicious files were downloaded into the client's system, which was eventually executed to carry out malicious activities such as stealing confidential information. One of the attributes that had correlation greater than 0.10 was user verified, a feature that has been used in previous studies to identify malicious users on Twitter [162]. Four features had the Pearson's correlation coefficient greater than 0.05 but less than 0.10, out of which three represented machine activities and one represented social attribute of the account that posted the tweet. Features such as bytes received, and packet send/received were contributing to the classification of the URL and have also been used as indicators to identify malicious URLs in previous studies [24]. Whereas, the number of friends an account has, was statistically significant and was contributing towards the classification of malicious URLs that were posted on Twitter. This attribute has also been used in earlier studied to identify both accounts that were posting spam and malicious URLs [192, 12, 183].

Furthermore, among the social attributes that had a correlation coefficient greater than 0, age of the account and number of followers a user account had, have also been previously used [192, 12, 183, 227, 126, 144, 43, 183] to detect accounts posting spam/malicious URLs. Out of the 58 features that had Pearson's correlation coefficient greater than 0, there were in total of 36 attributes related to machine activities and 22 attributes representing social features of an account. This showed a combination of attributes representing both machine activities and social features were used in the classification of the URL into malicious and benign.

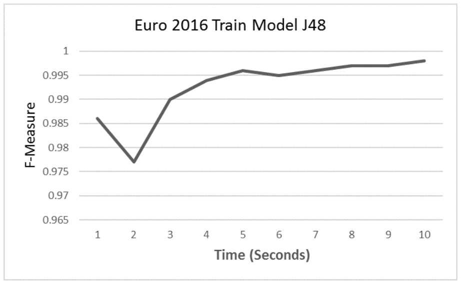
5.4 Results



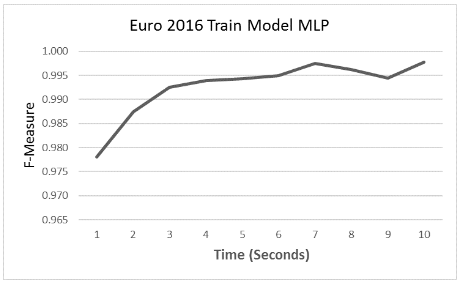
(a) F-Measure of Naive Bayes over time during training phase



(b) F-Measure of BayesNet over time during training phase



(c) F-Measure of J48 over time during training phase



(d) F-Measure of MLP over time during training phase

Figure 5.2: F-Measure of all machine learning algorithms over time during the training phase

Table 5.5: Training Model On Euro 2016 log file using NaiveBays Algorithm

Euro 2016 Train Model -NaiveBays			
Time	**Precision**	**Recall**	**F-Measure**
1	0.815	0.984	**0.891**
2	0.815	0.985	**0.891**
3	0.842	0.829	0.825
4	0.832	0.823	0.821
5	0.851	0.837	0.835
6	0.863	0.850	0.848
7	0.853	0.840	0.837
8	0.860	0.850	0.849
9	0.871	0.864	0.862
10	0.883	0.878	0.878

Table 5.6: Training Model On Euro 2016 log file using BayesNet Algorithm

Euro 2016 Train Model-BayesNet			
Time	**Precision**	**Recall**	**F-Measure**
1	0.997	0.987	0.992
2	0.995	0.995	**0.995**
3	0.994	0.994	0.994
4	0.993	0.993	0.993
5	0.994	0.994	0.994
6	0.994	0.994	0.994
7	0.992	0.992	0.992
8	0.995	0.995	0.995
9	0.994	0.994	0.994
10	0.994	0.994	0.994

Table 5.7: Training Model On Euro 2016 log file using J48 Algorithm

Euro 2016 Train Model -J48			
Time	**Precision**	**Recall**	**F-Measure**
1	0.990	0.986	0.986
2	0.978	0.977	0.977
3	0.990	0.990	0.990
4	0.994	0.994	0.994
5	0.996	0.996	0.996
6	0.995	0.995	0.995
7	0.996	0.996	0.996
8	0.997	0.997	0.997
9	0.997	0.997	0.997
10	0.998	0.998	**0.998**

Table 5.8: Training Model On Euro 2016 log file using MLP Algorithm

Euro 2016 Train Model -MLP			
Time	**Precision**	**Recall**	**F-Measure**
1	0.988	0.975	0.978
2	0.987	0.987	0.987
3	0.993	0.993	0.993
4	0.994	0.994	0.994
5	0.994	0.994	0.994
6	0.995	0.995	0.995
7	0.998	0.998	0.998
8	0.996	0.996	0.996
9	0.995	0.995	0.995
10	0.998	0.998	**0.998**

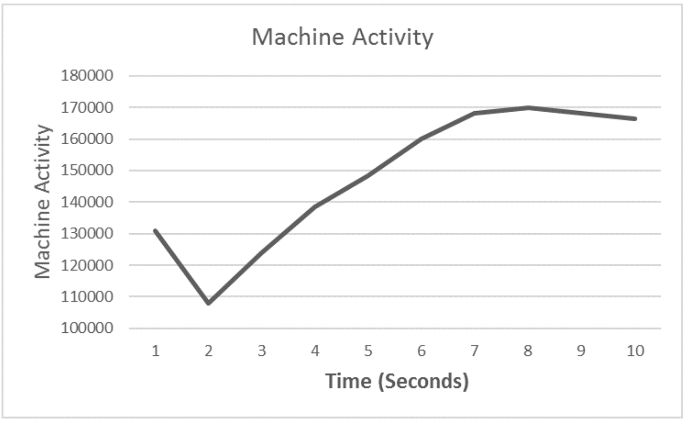


Figure 5.3: Machine activity over time

5.4.1 Training and Testing the Predictive Model

To determine which models provide the best predictive power – not just overall classification accuracy on all data – each model was trained and tested using data from sequential, cumulative time intervals. That is, at each time interval t from $t = 1$ to $t = p$ where p is the total number of time intervals (in this case $p = 10$), each model was trained and tested using data from *t=1-to-p* where $p = p + 1$. Each interval was evaluated with a ten fold cross validation using the Weka toolkit. The results were calculated using standard classification metrics: Precision, Recall and F-Measure. We also included a False positive rate as one of the metrics used in testing our unseen dataset.

$$FalsePositiveRate = \frac{FalsePositive}{FalsePositive + TrueNegative}$$

The results for each classifier are presented in Figure 5.2 and Tables 5.5-5.8. In each sub-figure, the machine learning model is trained and tested on the metrics derived using the Euro 2016 data-set. In each table, time represents the time in seconds elapsed from the time the URL was clicked, and the starting point is defined as $t = 1$. For example *Time=2* means that 1 second had elapsed since the URL was 'clicked' (URL clicked at $t = 1$).
Models built using the Naive Bayes and J48 algorithms (see Figures 5.2a and 5.2c) exhibit similar behaviour – they both have a dip in accuracy from the starting point and then the accuracy steadily rises. One explanation for this may be that in the opening seconds the system is inactive (see Figure 5.3), leaving the algorithm struggling to differentiate between benign and malicious activity. We define system activities as the range of activities that ensue in a visit to a Web page. These include process running, read/write operations on a file or registry entry, CPU usage, etc. The F-measure of the J48 machine learning model follows the trend of machine activity and continues to rise as more activity is recorded. When we compare the generative probabilistic models (Naive Bayes and BayesNet), we find that BayesNet outperforms NaiveBayes, suggesting inter-dependencies between the attributes. This is logical because when malicious network activity occurs it is likely that, for instance, CPU and RAM use will also spike, owing to the additional resource required for the activity. Looking at the results of the MLP model (see Figure 5.2d) we see the model is able to better balance the utility of the machine activity and Tweet meta-data features, and to control for the lack of machine activity at the start of the interaction. The F-measure rises smoothly from 1 second, suggesting it is making better use of the Twitter metadata to improve accuracy in the early stages of its activity.
In terms of the highest F-measure achieved, the J48 and MLP models perform best with 0.998 at 10 seconds. At 3 seconds the results are almost identical. The key difference between models is a slight improvement in MLP at 2 seconds, but this is countered by the speed at which the J48 returns a result. The MLP result takes longer than a second to return, whereas the J48 takes milliseconds. Thus, in practical applications, the J48 model is the most likely to be favoured.

5.4.2 Impact of Online Social Network Platform Attributes

Much research has been done in the past on the basis of Tweet attributes to detect malicious/spam Tweets propagating on Twitter [192, 14, 83, 228, 226, 50]. Thus in the previous section we included Tweet metadata as part of the feature set for prediction. However, these features are quite idiosyncratic and not consistent across different OSNs. For instance, if we wanted to predict a drive-by download via other OSNs such as Facebook, Tumblr or Instagram, we would get a slightly different set of user characteristics from the metadata available. Hence, we aimed to determine the impact of removing these features, using machine activity data alone to determine the applicability of our method across different OSNs. To conduct this experiment we selected the model from the previous experiment that performed best – the J48 algorithm that displayed apparent correlation with machine activity. We retrained the model using the machine activity but no Tweet metadata. Table 5.9 and Figure 5.4 show the performance of this model over time.

Table 5.9: Training Model On Euro 2016 log file using J48 Algorithm without Tweet metadata

Euro 2016 Train Model -J48 (Without Tweet metadata)			
Time	**Precision**	**Recall**	**F-Measure**
1	0.89	0.863	0.858
2	0.945	0.94	0.939
3	0.909	0.9	0.901
4	0.92	0.904	0.905
5	0.928	0.916	0.915
6	0.914	0.899	0.897
7	0.915	0.899	0.897
8	0.929	0.918	0.918
9	0.941	0.933	0.933
10	0.952	0.947	0.947

Figure 5.4 shows the F-measure metrics for the J48 model when trained with and without Tweet metadata. When we compare the results of the two J48 models we observe that the model built solely on machine activity data fluctuates over time. The model F-measure drops by around 13% at t=1 second. This suggests that Twitter's idiosyncratic attributes,

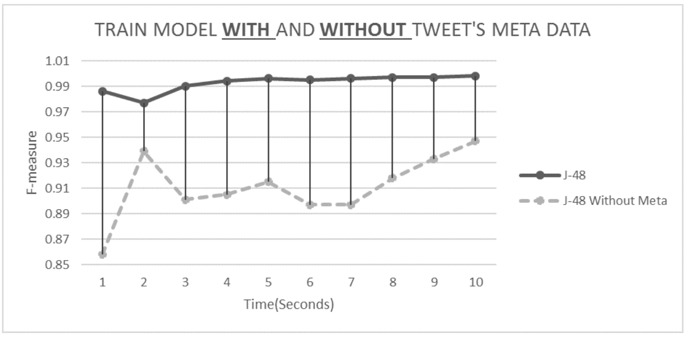


Figure 5.4: Train J48 Model without OSN metadata

such as number of followers, significantly contribute to the accurate classification of malicious URLs, but that the model is still highly accurate when using machine activity alone, making it likely that the approach will be able to detect drive-by downloads on other OSNs. Without the OSN metadata, the model seems able to cope with the low rate of activity at the start of the interaction, which is interesting, because this is the opposite of the situation when metadata were being used to train the model. The key finding here is that including the OSN metadata improves the prediction of the classifier by 12.98%. Thus our future aim will be to retain the characteristics of the user account wherever possible when they come from OSNs outside of Twitter. Nevertheless, our model still provides a high predictive performance even without these idiosyncratic data, providing promising results for the application of machine activity models for predicting malicious behaviour in URLs on multiple OSN platforms.

Table 5.10: Test Model On Olympics 2016 Dataset

Test Olympic- Vote Algorithm (NaiveBayes and J48)				
Interaction Time(Sec)	**FP Rate**	**Precision**	**Recall**	**F-Measure**
1	0.149	0.836	0.723	0.730
2	0.152	0.861	0.834	0.833
3	0.147	0.867	0.846	0.845
4	0.160	0.856	0.834	0.832
5	0.157	0.881	0.859	0.856
6	0.164	0.884	0.866	**0.862**
7	0.202	0.860	0.837	0.831
8	0.195	0.855	0.837	0.832
9	0.192	0.854	0.837	0.833
10	0.185	0.855	0.837	0.833

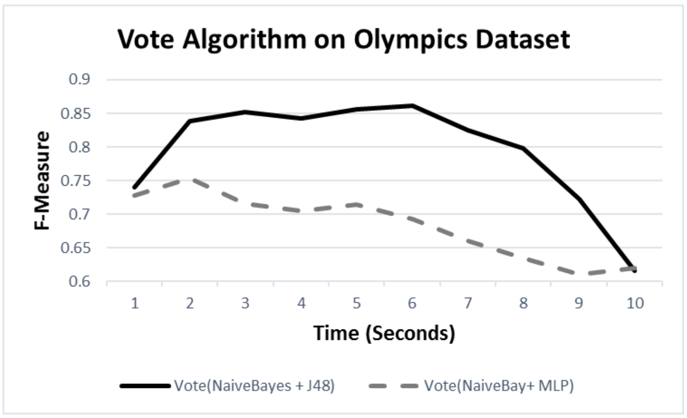


Figure 5.5: Testing on Olympics Data using model built earlier

5.4.3 Testing Using Unseen Data from Olympics 2016

In the previous two experiments we validated our predictive models using a single dataset from Euro 2016 and obtained promising results. One possible limitation of this experiment comes from the variance in cyber attack methods over time. For instance, in a second unrelated event we may see a new collection of individuals spreading malicious URLs, and indeed the URLs themselves exhibiting different behavioural profiles. We therefore now introduce an unseen dataset from the Olympics 2016. This dataset has played no part in training the model and is thus completely unseen, to some extent testing the generality of the approach .

Given that J48, MLP and Naive Bayes (NB) models performed best on the Euro 2016 data, we combined them using a *Vote* meta-classifier. The Vote algorithm allows two or more machine learning algorithms to be combined in such a way that the label likelihood from each model is used to provide the classification label for each test instance. In our case we used the average probability as the decision point. Through experimentation we narrowed the field down to two combinations of methods that produced the best classification performance: J48 & NaiveBayes and NaiveBayes & MLP. Figure 5.5 shows the F-measure for both. The rationale for choosing two classifiers through the vote algorithm was that we wanted the final classifier to have a mix of generative and discriminative machine learning properties. The combination of J48 with NaiveBayes reaches an F-measure of 0.85 after just two seconds into the interaction with a Web page. Note again that t=1 is the time when the test machine launches the URL so there is a lag of 1 second, meaning that t=3 is actually 2 seconds after the URL was clicked. The combination of NaiveBayes and MLP reaches a maximum F-measure of 0.75. Thus a significant performance difference results from combining the Naive Bayes and J48 models.

This is somewhat counterintuitive considering that in the previous experiments the MLP and J48 algorithms were almost indistinguishable at 3 seconds, and that J48 is a rule-based model. We would expect a rule-based model to overfit a single event (i.e. the CPU, RAM and network traffic would have a large variance between events as demonstrated by [25]). This was not the case, and in fact this combination produced a model that is capable of detecting malicious URLs in an unseen dataset with 0.83 F-measure and a 15.2% False Positive rate only 2 seconds into the interaction. We next rebuilt the Vote model with and without Tweet metadata. Figure 5.6 shows the result of the classifier when we tested this model on the Olympics 2016 (unseen) dataset. We see a significant increase (on average an increase of 24% was observed) in the F-measure of the classifier when Tweet attributes were added to machine data. This suggests that even though Tweet attributes are similar across events they are not alike enough to accurately classify a URL on their own, and we still require machine data to improve our classification across events. Note also that the results of the same models based on Tweet metadata alone using the Olympics 2016 dataset gave an F-measure of only 0.16 (full results not shown, for brevity). We can see that while the attack vectors as measured by system activity are changing between events (hence the drop in performance when we remove the Twitter metadata), the combination of the network characteristics of the individuals posting malicious URLs, and machine activity recorded while interacting with URLs remains fairly stable, showing a drop in F-measure from 0.977 to 0.833 at 2 seconds between events (see Table 5.10). Our model may therefore not be limited to a single case, but could be applied to multiple events that attract large numbers of users on Twitter, where they would maintain reasonably low error rates when predicting malicious URLs just 2 seconds into the interaction.

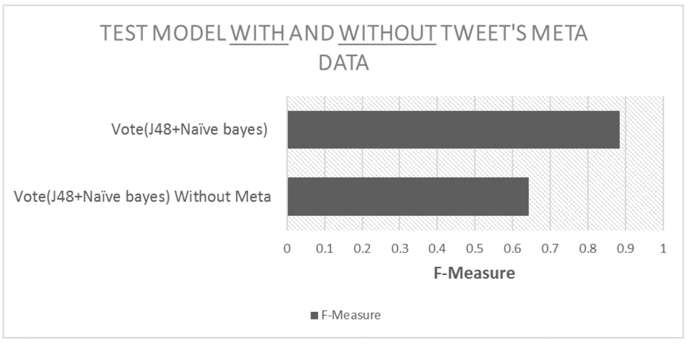


Figure 5.6: Comparison of Results on Unseen Data with and without tweet metadata

5.4.4 Adaptive Nature of the Predictive Model

To make our predictive model adaptive, a feed-forward architecture was implemented (see Figure 5.1). The rationale was to ensure that new techniques employed by cybercriminals to carry out a drive-by download attack, as captured in the form of machine activity, would be continually captured and considered while training the model. In order to check the effectiveness of the feed-forward architecture in achieving this, we conducted a further experiment. We trained the model on the Euro 2016 dataset with varying sample sizes, and tested using 10-fold cross validation. We then tested the model on an unseen dataset (Olympics 2016), with the hypothesis that increasing the size of a dataset would capture new machine behaviour that would increase the diversity of features seen by the model and improve the overall F-measure of the predictive model. We used a range of sample sizes for model training - 1%, 5%, 10%, 25%, 50% and 100%. Figure 5.7 displays the results of these experiments.

We found that training the model with only 1% of total sample size, using a 10-fold cross technique, produced an F-Measure of 0.89. However, when we tested the model on an unseen dataset we found that the F-measure dropped to 0.533. By increasing the size of the training dataset from 1% to 100% in various stages, we aimed to simulate how the model would behave when new data were added to it over time with increased feature diversity. It was observed that the F-measure did indeed increase with increases in the dataset size during the training phase as well as in the testing phase, showing the model to be adaptable when presented with more diverse machine behaviour. It was seen

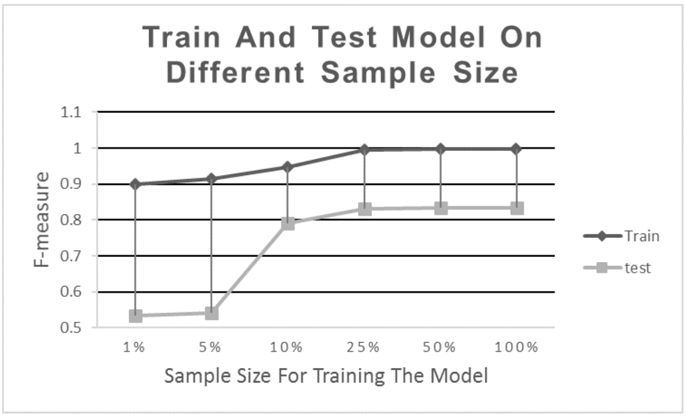


Figure 5.7: Comparing classifier accuracy in terms of F-measure when data set is changed

there was a significant jump in the F-measure (from 0.54 to 0.80) when the sample size was increased to 10%. However, little change in the F-measure was observed when the sample size from 25% to 100%, suggesting that 25% of data representing machine activity was enough to build a model that would give an F-measure over 0.83 and a 15% False Positive Rate. After this point more data does not appear to improve prediction accuracy.

5.5 Conclusions

In this chapter, we aimed to build on Chapter 4 by moving beyond *detection* of malicious URLs to *predict* that the URL was likely to be malicious within seconds of opening the interaction - before the drive-by download attack could complete the execution of its payload. This is the first time a method has been tested to predict a malicious outcome before it takes place - existing literature always classified URLs using all the data generated throughout an interaction period - so provided a post-hoc result, or without actually observing the malicious activity; making a decision based on previously seen behaviour. We also aimed to improve classifier performance by including OSN and URL features to support the increase of diversity in the machine activity feature set.

Tweets were captured containing URLs around two global sporting events. Our system produced a second-by-second time series of system-level activity (e.g. CPU use, RAM use, network traffic etc.) during the visitation of a Web page. The predictive model was trained using four different types of machine learning algorithm on log files generated from one event (Euro 2016). The model was then validated using tweets captured during another event (Olympics 2016). The rationale was to determine if similar machine activity and tweet attributes were exhibited in two completely different events (i.e. does the model generalise beyond a single event). A ten-fold cross-validation was performed to train the model, and an F-measure of 0.99 was achieved by using the log files generated at 1 second into the interaction with a Web server. One of the interesting observations during the training phase was that by using tweet attributes we can increase the accuracy by 12.98% during training and around 24% during testing phase when compared to machine activity alone, demonstrating that the machine activities carrying out drive-by download attacks were relatively stable, while the URL behaviour changed.

When tested using an unseen dataset (Olympics 2016) we achieved an F-measure of 0.833 from log files generated at 2 seconds - that is 1 second after launching the URL. The highest F-measure achieved on the unseen event was 0.862 at 5 seconds from the time the URL was launched. Our model may therefore not be limited to a single case but could be applied to multiple events on Twitter maintaining reasonably low error rates when predicting malicious URLs just 1 second into the interaction. The model allows us to reduce the detection time of a malicious URL from minutes - the time taken to run the URL in a secure sandbox environment - to 5 seconds, with F-measure of 0.86 on an unseen dataset. Furthermore, it allows us to stop the execution process with 0.833 F-measure just 1 second after clicking the URL, preventing the full execution of the malicious payload, rather than detecting the malicious action retrospectively and having to repair the system.

Even though the proposed model is capable of processing a minimum of 15,000 URLs and a maximum of 150,000 URLs per day per deployment, this number could be increased further by deploying the predictive model in parallel. However, even with multiple deployments of the predictive model, the processing of URLs per day might not be enough to handle the volatile load of tweets generated during the popular event and 1 in 10 URLs points to a malicious Web server [47]. Thus, there is a need to understand the infection behaviour and the factors that contribute towards the propagation of post containing malicious URLs. This would help devise a strategy to curtail the infection even when the users that were spreading are removed from the network. Which leads us to the next research question, do social and content-based factors that exist as part of a tweet containing a malicious URL aid its propagation?

Chapter 6

Fear me not, retweet me

Why are certain Tweets retweeted more than others? Virality is partly driven by psychological arousal. Is the act of retweeting affected by the tweet content and the emotions it evokes? Having identified sporting events as being used by cyber criminals to infect a user's system, we investigated propagation of malware through the action of a retweet. In this chapter we uncover the social and content-based factors that exist as part of a tweet containing a malicious URL that aid its propagation

6.1 Introduction

Twitter's popularity continues to tempt cybercriminals to carry out a myraid of cyber attacks of various kinds. Cyber attacks, such as distributed denial of service [121], cross-site scripting [86], Trojan attacks [153] and drive-by downloads [102], continue to be major threats on Twitter. Once a user's system is infected, sensitive information is exposed to unauthorised users whose machines can be used to carry out further attacks. Even though various detection [100, 210, 137, 135, 31, 125, 65, 103, 108, 114, 11, 116, 149, 22] have been proposed to identify malware, there is still a chance that a large number of users will be infected before the malware is detected and removed from the network. This is because current detection models are not capable of processing a large number of tweets in real time, as the frequency of tweets varies from an average of 6,000 [145] to a high of 143,199 tweets per second [205]. The evasion of existing detection methods is also evidenced by a recent report that suggests drive-by downloads account for 48% of attacks by exploiting Web-based vulnerabilities [175]. Thus, in addition to detection/prediction models to counter malware on OSN, there is a need to understand its propagation behaviour and factors that influence its propagation.

In the existing research, propagation models for malware are based either on scanning techniques or the topology of a network, where the focus of the research has been on the communication medium [41, 70], social network topology [70] and the relationship [194] exhibited among users. The underlying assumption while building propagation models has been that the malware is self-propagating. However, the content sharing capability of an OSN adds another medium through which malware could be distributed, where an OSN presents a highly interconnected network that could propagate malicious links on the scale of thousands in a matter of minutes. Malware in the form of malicious links, that were hidden in a posts could easily be propagated by means of sharing these posts among users in OSN. This leads to questions such as: (i) why are certain post shared more than others?; (ii) is virality driven by physiological arousal?; and (iii) do emotion and sentiment represented in a post aid in its propagation?

In this chapter we identify both social and content based factors, such as emotion and sentiment, and evaluate their influence on the propagation of posts containing malicious URLs. Emotion and sentiment were particularly chosen because it has been well established that people transfer positive and negative moods to one another in a form of emotional contagion [89]. Through experiments, researchers have shown that emotions representing long-lasting moods such as happiness and unhappiness can be diffused through online social networks [89, 71]. Also, one person's emotional state can be used to predict the emotional states of connected friends. Kramer et al. [120] show how the posting behaviour of a user varied according to the emotional content received.

In order to uncover different factors that cybercriminals use to entice users to retweet content containing a malicious URL, a corpus of circa 3.5 million tweets were collected around seven different sporting events in 3 years; a sub-sample was created and passed to a high interaction honeypot (see Chapter 3 for details) to identify those tweets that contained malicious URLs. Capture HPC identified 105,642 tweets containing malicious URLs and 186,244 tweets containing benign URLs. These data were entered into statistical models to estimate the size and survival of *'malicious'* information flows. To build statistical models that help explain the correlation between emotion and propagation we derived two dependent variables, *size* and *survival*. Size was defined by the number of retweets a tweet receives in the study window, and survival represents the time between the first and the last retweet in the same window. Independent variables derived from tweets included social characteristics (those of users) and content factors (those of sentiment and emotion). To the best of our knowledge, this is the first study that links emotions and sentiment to the propagation of tweets containing malicious URLs on Twitter.

6.2 Background

As technology has evolved over time, so has malware. The increasing complexity of users' communication on the Internet has given cybercriminals several methods by which to propagate malware. The current work on malware propagation is broadly divided into two categories: propagation on traditional networks and that on online social networks. Research on traditional networks focuses either on studying malware propagation based on the transmission medium [164, 132] or based on the topology of the network [76]. Furthermore, malware propagation on traditional networks has frequently applied epidemiology theory [235, 229, 154, 132] to understand how the damage spreads.
However, with the growing popularity of online social networks among cybercriminals, traditional networks were augmented by a new medium to propagate malware, shifting the research focus from understanding malware propagation on traditional networks to online social networks. The degree of complexity has also increased with the introduction of online social networks, since they were overlaid on top of traditional networks. Furthermore, these platforms introduced new communication methods and social relationships among active users on online social networks. Unfortunately, this social relationship or connection between users gave cybercriminals another medium through which they could propagate malware. The malware propagation models that were subsequently built included mobile devices [70], Bluetooth [41], email [220], *user-friends* relationships [70],

and online social network platforms [194] such as Facebook [67] and Twitter [177]. These propagation models analysed user behaviour [225, 215] and investigated the user relationships [40] to shed light on the infection rate and reach of malware [177, 216]. These are covered in more detail in Chapter 2.

Technology and malware are constantly evolving. Models investigating malware propagation have to incorporate new features to tackle the new techniques used by cybercriminals for malware propagation. Content based features such as sentiment have been more often used in models to detect spam [212, 95] than to understand their propagation. Features such as emotions are used to understand the virality of posts on online social networks [17, 211]. Particularly, content containing negative emotions have a higher chance to be shared on OSN than positive emotions [17]. This tactic of using negative emotions to convey a message to a larger audience have been seen in advertisements as well. For example, a series of short films called "The Hire" that evoked negative emotions by including a story line that involved a car chase were created by BMW in order to gain millions of views [16]. This provoked the question of whether cyber criminals were employing similar tactics by creating contagious posts containing negative emotions on OSN to propagate malware. We cannot of course presume to know the the mindset or deliberate actions employed by attackers but we can provide evidence to measure the success of emotions in increasing size and survival of malicious URLs on OSNs

Even though content-based features are being considered in research, they are limited to understanding the flow of information related to news or to detecting spam. The analysis conducted to understand content sharing on OSN use basic emotions as defined by Ekman [63] that identifies more negative emotion than positive in its six basic emotions (Anger, Fear, Disgust, Sadness, Surprise and Joy). This presents an imbalanced set of positive and negative emotions for analysis. A research gap exist that links emotions and sentiment to malware propagation on online social networks, where the set of emotions used for analysis are balanced in term of positive and negative emotions such as defined by Plutchik [157] (Anger, Fear, Disgust, Sadness, Anticipation, Surprise, Trust and Joy).

To the best of our knowledge this is the first study that correlates emotions expressed in a tweet containing malicious URL with its propagation.

6.2.1 Contribution

The contributions of this chapter develop evidence to answer two research questions that were identified in Chapter 2:

RQ7 *Do social features aid the propagation of posts containing malicious URLs?*

RQ8 *Do features derived from content, such as emotion or sentiment, help in the propagation of malware?*

In this chapter we address the research gap and investigate the factors that contribute to the retweeting of malicious URLs during popular sporting events. It contributes to the broader literature on malware propagation by:

- Determining if tweets that contain negative emotions are statistically associated with the size and survival of the information flow, and at the same time, determining what discrete negative emotions emerge as most significant;
- Determining if the tweets that contain positive emotions are statistically associated with the size and survival of information flow;
- Determining if social factors, such as the number of followers a user has, are statistically positively associated with the size and survival of the information flow;

6.3 Data Collection and Predictive Measures

6.3.1 Data Collection

Over 3 years, data from seven different sporting events was collected using the Twitter's streaming API and is summarised in Table 6.1 (the detail regarding data collection is discussed in Chapter 3). The seven sporting events that were used are;

Table 6.1: Description of Tweets collected during sporting event

Sporting Event	**Year**	**Location**	**Hashtag Used**	**Number of Tweets Captured**	**Malicious Tweets Identified**	**Number of Unique Tweets**
Federation Internationale de Football Association (FIFA) World Cup	2014	Brazil	*#FIFA2014*	95,000	46,481	2,039
Circket World Cup	2015	Australia & New Zealand	*#CWC15*	7,961	4,238	891
Rugby World Cup	2015	United Kingdom	*#RWC2015*	127,393	3,836	627
SuperBowl 2015	2015	USA	*#SB50, #SuperBowlSunday #superbowlXLIX*	122,542	2,293	1,230
European Football Championship	2016	France	*#Euro2016*	3,154,605	21,559	975
Olympics	2016	Rio de Janeiro (Brazil)	*#Rio2016*	6,111	3,359	525
SuperBowl	2016	USA	*#SuperBowlSunday #NFL*	57,572	23,876	582

1. Fédération Internationale de Football Association (FIFA) World Cup of 2014 : was the 20^{th} FIFA World Cup for men's national football team. It took place in Brazil from 12 June to 13 July 2014 and 32 teams from different countries participated in the event. During this period 642 million tweets were posted on Twitter related to the event and Brazil vs Germany semi-final was the most tweeted about event generating 35.6 million Tweets [185].

2. The American Football Superbowl 2015: was an American football game played between the American Football Conference (AFC) champion *New England Patriots* and the National Football Conference (NFC) champion *Seattle Seahawks*, to determine the champion of the National Football League (NFL) for the 2014 season. A total of 28.4 million tweets were recorded during the event and it was the most talked about event on Facebook, with 1.36 million people commenting every minute during the event [78].

3. The Cricket World Cup 2015 : was the 11^{th} men's Cricket World Cup, jointly hosted by Australia and New Zealand from 14 February to 29 March 2015. A total of 14 teams from different countries participated in the event and 3.5 million tweets were recorded during the period event occured [113]. During which the India vs Pakistan match was the most talked about match that generated 1.7 million tweets. [189]

4. Rugby World Cup 2015 was the eighth Rugby World Cup, hosted by England from

18 September to 31 October. A total of 20 teams from different nations participated in the event. The final between New Zealand vs Australia recorded 560,000 tweets with a highest frequency of 2,900 tweets per second [188].

5. The American Football Superbowl 2016 : was an American football game played between the AFC champion *Denver Broncos* and the NFC champion *Carolina Panthers*, to determine the champion of the NFL for the 2015 season. A total of 27 million tweets were reported during the event by Twitter and an engagement of 60 million users related to the event was reported by Facebook [82].

6. The European Football Championships 2016 : was the 15^{th} International men's football championship of Europe organised by The Union of European Football Associations (UEFA). It was held in France from 10 June to 10 July 2016 and a total of 24 teams participated in it. Where England vs Iceland was reported to be the most tweeted about programme, generating 2.1 million tweets during the match [168].

7. The Olympics 2016 : was an international multi-sport event that was held from 5 to 21 August 2016 in Rio de Janeiro, Brazil. A total of 207 nations participated in the event and it was the most talked about event of 2016, even surpassing the U.S Presidential election [118].

6.3.2 Dependent Measure

Two dependent variables were generated to measure the effect of emotion represented in text on malware propagation, namely, *size* and *survival*. Size is defined as the number of times a tweet is retweeted. This is a measure of virality and therefore we assume that the more a tweet is retweeted, the greater the risk to other network users if the tweet is malicious. Survival is defined as the length of time over which a tweet continues to receive retweets. Again, if the tweet is malicious, the longer it continues to be retweeted, the longer the risk to network users remains.

Table 6.2: Description of Malicious Sample Data *N=1,137*

Variable	**Range**	**Mean**	**Std. Dev**
Dependent			
Size	5-22,614	86.53	817.19
Survival	0-2,850,416	218,556.40	483,290.60
Independent			
Social Factors			
Hashtag	0-13	1.90	1.66
Mentions	0-6	1.22	0.93
Friends	0-784,471	3652.78	28614.25
Followers	0-928,4012	168519.50	618537.10
Age of Account	562-1,321	918.25	287.17
Emotion			
Anger	0-3	0.21	0.48
Anticipation	0-7	1.01	1.18
Disgust	0-5	0.27	0.60
Fear	0-5	0.39	0.67
Joy	0-7	0.71	0.95
Sadness	0-3	0.28	0.57
Surprise	0-10	0.44	0.76
Trust	0-5	0.50	0.75
Sentiment			
Negative	0-3	0.19	0.46
Positive	0-4	0.37	0.67

In order to identify the number of retweets a tweet received, all the retweets were filtered

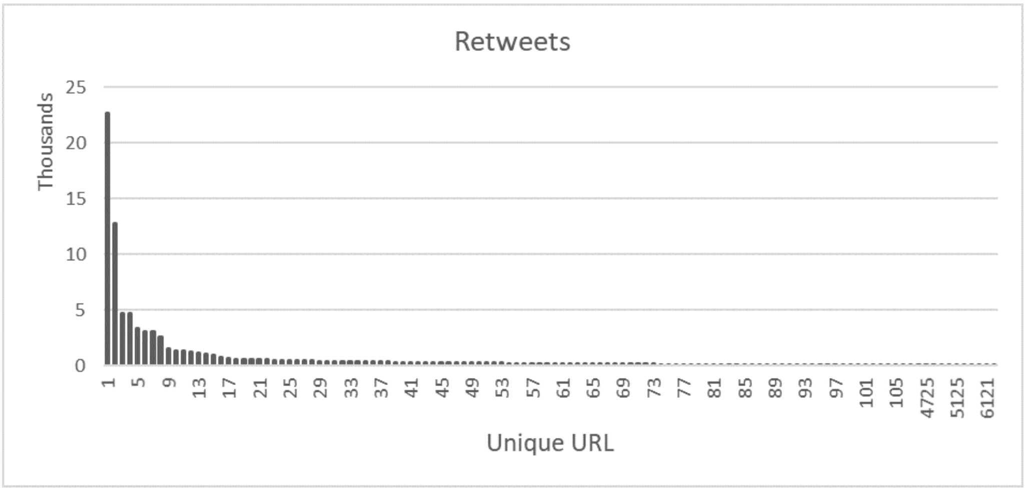


Figure 6.1: Number of Tweet's captured with malicious URLs for each sporting event

by looking at tweets that had RT as the prefix, as one of Twitter's features is to prefix each retweet with *RT*. All the unique tweets in the dataset were identified and we counted the number of times each unique tweet had been retweeted. The count for each retweet gave

Table 6.3: Description of Benign Sample Data *N=1,862*

Variable	**Range**	**Mean**	**Std. Dev**
Dependent			
Size	5-48,875	90.86	1,160.23
Survival	0-2,896,989	291,628.00	616,751.50
Independent			
Social Factors			
Hashtag	0-12	2.02	1.81
Mentions	0-7	1.00	0.97
Friends	0-481194	2,973.49	16,787.49
Followers	0-12,700,000	143,917.60	761,002.80
Account Age	933-1635	1,200.09	306.50
Emotion			
Anger	0-3	0.07	0.28
Anticipation	0-6	0.21	0.51
Disgust	0-2	0.05	0.23
Fear	0-3	0.09	0.32
Joy	0-5	0.17	0.46
Sadness	0-2	0.08	0.29
Surprise	0-3	0.09	0.32
Trust	0-5	0.22	0.51
Sentiment			
Negative	0-3	0.14	0.39
Positive	0-5	0.34	0.64

us the size of each tweet's information flow. The sample showed a positive skew (see Figure 6.1) where unique tweets retweeted fewer than five times were found on the left-hand side of the distribution. Since the main aim of the study was to understand the propagation factors, tweets with fewer than five retweets were removed from the sample, following research that indicates this to be a reasonable cut-off point for non-trivial information flows [9, 26]. The resulting dataset contained *N=1,137* unique malicious tweets and *N=1,862* unique benign tweets that had been retweeted at least five times. Tables 6.2 and 6.3 give details of both dependent variables, where the range of Size for malicious tweets was 5 - 22,614 retweets with a mean of 87.53 retweets and the Size for benign tweets was 5 - 48,875 with a mean of 90.86 retweets. For survival, we found that malicious tweets had a range of 0 - 2,850,416 seconds with a mean of 218,556.40 seconds, and benign tweets had a range of 0 - 2,896,989 seconds with a mean of 291,628 seconds. The minimum of zero represents rounding down to the nearest second where retweets happened within milliseconds. Furthermore, an average mean of 0.48 for emotions was recorded in the malicious sample, a figure that is significantly higher from average mean of emotions (0.12) in a benign sample. This descriptive statistic suggests that malicious tweets that were identified across all the seven events contained more keywords that represent emotions than in those identified as benign. The presence of more keywords associated with emotions in tweets across all the seven events may suggests that those words that were associated with emotions were intentionally added to increase the probability that the tweet would be shared. We cannot be certain of this but this is a tactic that is commonly used in creating an advertisement to gain a large number of views [16].

6.3.3 Independent Features

Social Features

Based on previous research ([102, 32, 38, 192, 14, 83, 50]), where social features were used to detect malware in OSNs, the number of friends, number of followers and age of the Twitter account that posted the initial tweet were extracted from the meta-data of the captured tweet. The number of hashtags and user mentions that were included in the tweets, were also calculated. See Tables 6.2 and 6.3 for details regarding the social features extracted from the tweets that were captured.

Content Features

In addition to the social factors, the aim was to study the emotions and sentiments reflected in the captured tweets. The rationale behind extracting emotions and sentiment was to identify the association between different emotions and (i) the frequency of retweets (size), and (ii) the duration of continuous retweeting (survival). The emotional aspect of content circulating in OSNs shows its impact based on whether it is being shared or not [90]. Emotions can even be the common link between two people who find common

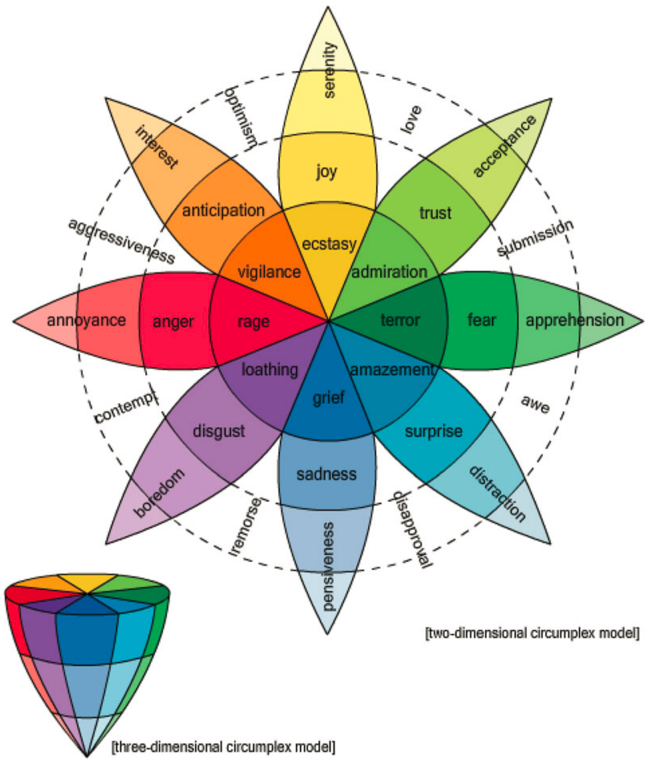


Figure 6.2: Plutchik's wheel of emotions. Similar emotions are placed next to each other. Contrasting emotions are placed diametrically opposite to each other. Radius indicates intensity [157].

ground between them. It has also been known for two culturally different people, who had absolutely no contact in the past, to display the same facial expression to reflect a particular emotion [64], so it is possible that in an online social network people follow similar patterns of retweeting when reading text exhibiting specific emotions. Furthermore, the effect of emotions has been studied by biologists and psychologists to establish a link between one's survival and the emotion expressed [54]. Numerous methods of classifying human emotion have been proposed. Some have categorised emotions into two categories - instinctive (emotions that we can sense and perceive) and cognitive (emotions that result from thinking or reasoning) [230]. However, some have argued against this categorisation of emotions by claiming that emotions do not precede cognition [230].

Plutchik proposed a resolution of this debate by suggesting that the problem lies in the way the categories are defined. Basic and instinctive emotions can be correlated and so can complex and cognitive ones [157]. This leads to definition of basic and complex emotions. However, in the present research we focus on the basic emotions. A number of theories have been proposed to define the basic emotions perceived by humans, among which the models proposed by Ekman [64] and Plutchik [157] has been found to be more popular than others. Ekman [63] identifies six basic emotions (Anger, Fear, Disgust, Sadness, Surprise and Joy), whereas Plutchik [157] identifies eight (Anger, Fear, Disgust, Sadness, Anticipation, Surprise, Trust and Joy) that are more balanced in term of positive and negative emotions and also contain the 6 basic emotions identified by Ekman. Figure 6.2 shows the eight primary or basic emotions identified by Plutchik, along with other identified emotions depicted in a circular diagram that was later named the *wheel of emotion*. The eight basic emotions are positioned in such a way that each is diagonally opposite a contrasted emotion; for example, joy-sadness, anger-fear, trust-disgust and anticipation-surprise (see Figure 6.2). Plutchik also categorised a few emotions as *primary dyads*, shown in Figure 6.2 in the white spaces between the primary emotions. He argued that these emotions are a result of two primary emotions; for instance Love is a result of Joy and Trust. The radius lines in the circle at the centre of the figure represent the intensity of the emotion: the closer one gets to the radius lines the higher the intensity. Interestingly, he states that the emotions in general are not found in isolation and do not have clear boundaries. Hence, in deriving the emotions in a tweet, we identified all the emotions expressed by it. For example a tweet might contain both anger and sadness.

In earlier studies an imbalanced set of basic emotions taken from Ekman[64] model were used to identify the relationship between emotions expressed in a post and the content shared. For example Kramer et al.[120] showed through a series of experiment that emotions can be transferred to others via content shared, leading people to experience the same emotions without their awareness. However, no study has been conducted that links emotions to propagation of tweets containing malicious URL using basic emotions from Ekman [64] or any other model. In order to address the gap in the research, Plutchick's eight primary emotions [157] were chosen for the experiment, demonstrating a balanced mix of positive and negative emotions. We do not claim that Plutchik's eight emotions are better than other categorisations; however, we adopted them because they are well-founded in psychological empirical research, and unlike some other choices, for example, that of Ekman, they are composed of a balanced set of positive and negative emotions. The eight basic emotions as defined by Plutchik are used as independent features and are derived from the tweets. In addition to the eight emotions, positive and negative sentiments were included.

Architecture of the Sentiment And Emotion Extraction Tool

A Java-based script (see Figure 6.4 for the flowchart) was developed to extract the emotion and sentiment of each tweet. The script identifies the emotion and sentiment based on keywords, following the principle that it is possible to infer emotional properties associated with words [60]. To start with, each captured tweet was pre-processed by removing any stop words and non-ascii characters except those representing *emoji* or emoticon - symbols representing emotions. Next emoticons were identified and the emotions associated with them were noted. Once an emoticon was identified and the emotions were noted, the emoticon was stripped out of the tweet and passed on to the next stage. We used multiple dictionaries containing words associated with each of the eight emotions. The three dictionaries that we used are listed below.

1. ***WordNet Affect Lexicon*** was manually created and is composed of words that evoke emotions. These words were collected from various sources such as dictionaries [217], newspapers and literary texts [191]. The words were then used to mark all their synonyms with the same emotion in the WordNet dictionary. The emotions in Wordnet Affect are organised in a hierarchical structure with the root having positive, negative, ambiguous and neutral emotions. Each category is then sub-divided and the related emotions are mapped to the root. For example positive emotions are divided into thirteen different types of positive emotion (joy, love, affection, liking, enthusiasm, gratitude, self-pride, levity, calmness, fearlessness, positive-expectation, positive-fear and positive-hope). These are then again divided into sub categories. Due to limitations of space, in Figure 6.3 only the complete

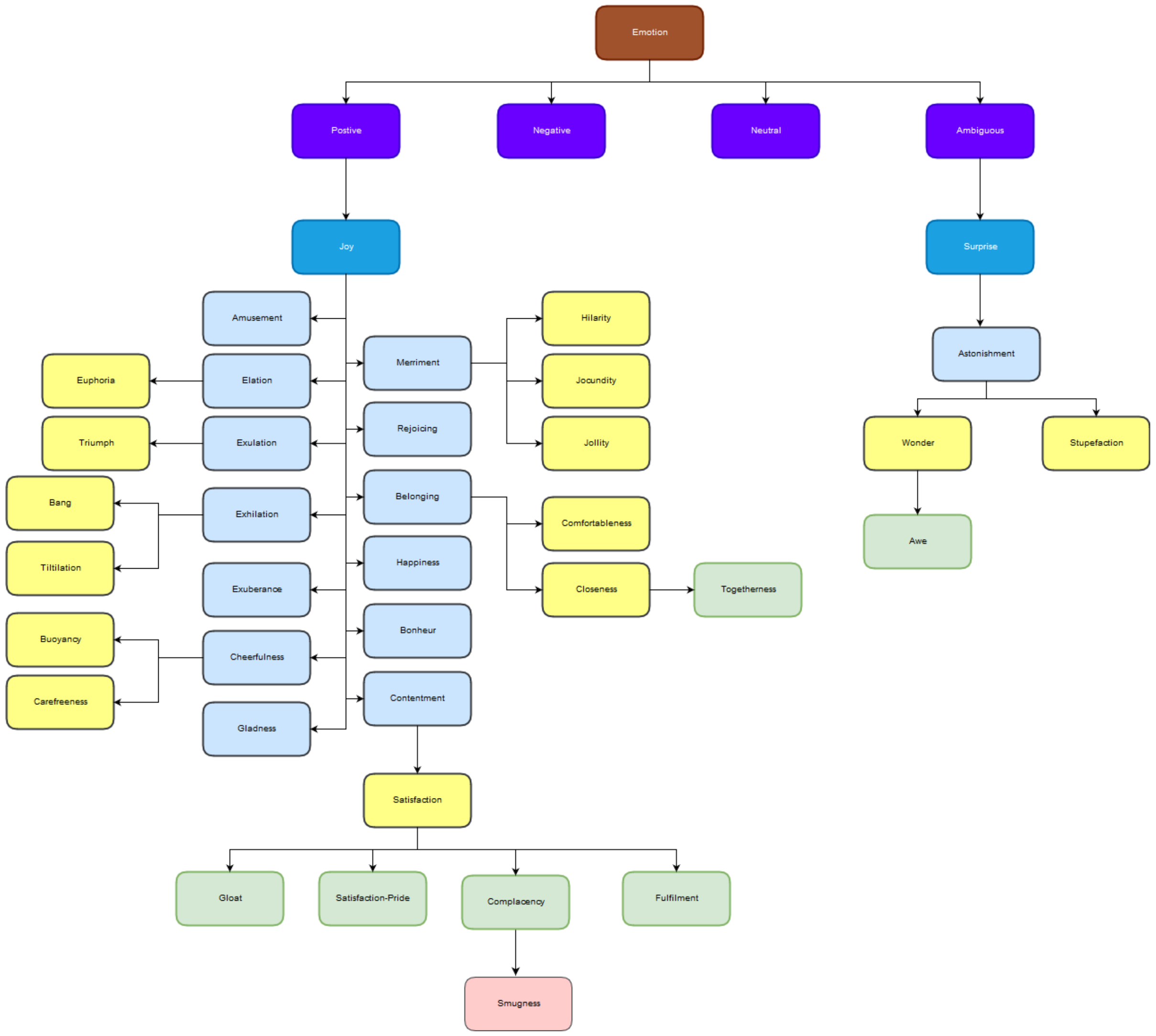


Figure 6.3: Hierarchical structure of Wordnet Affect [35]

mapping of the emotions Joy and Surprise in Wordnet are shown, where each level is represented by the same colour. Similar mappings for other emotions exist but are not shown due to space restriction.

2. ***NRC-Emolex*** is another word-emotion based dictionary that was created by the National Research Council of Canada [148]. The words from the Macquarie Thesaurus [57] were filtered by their frequency of use in the Google n-gram corpus [21]. Words that were part of Ekman's subset in the WordnetAffect were also included. These terms and phrases were then annotated using the crowd-sourcing platform Mechanical Turk. For each question that was asked to identify the emotion associated with the word, a test question was asked to eliminate the users who did not understand the term or word. The test question asked the user to name a word with similar meaning from a list of four options, where one in the list was a synonym and others were distractions. Only those results where the participants got the answers right were included. Along with the emotions, sentiments for each word were also identified using the same approach as that used to identify associated emotions.

3. ***Hashtag Emotion Corpus*** - a hashtag based dictionary was created where the association between a hashtag and an emotion was recorded [147].

In addition to these dictionaries, we used the emoticons contained in a tweet to identify appropriate additional emotions. Figure 6.4 provides an overview of the core program in the form of a flowchart. The program first reads a tweet from the dataset and then pre-processes it by removing stop words and punctuation. It then checks whether any emoticons are present and identifies those that it finds with the associated emotions. In the next step the tweet was split into multiple tokens; for each token (word) the associated emotion was identified using the three dictionaries. As the WordNet affect lexicon represents emotions in a hierarchical structure (see Figure 6.3), a backward mapping of granular emotions was performed on Plutchik's eight emotions. In addition to the eight primary emotions that were identified for a tweet, sentiments (positive or negative) were also identified as each word of the tweet was being iterated.

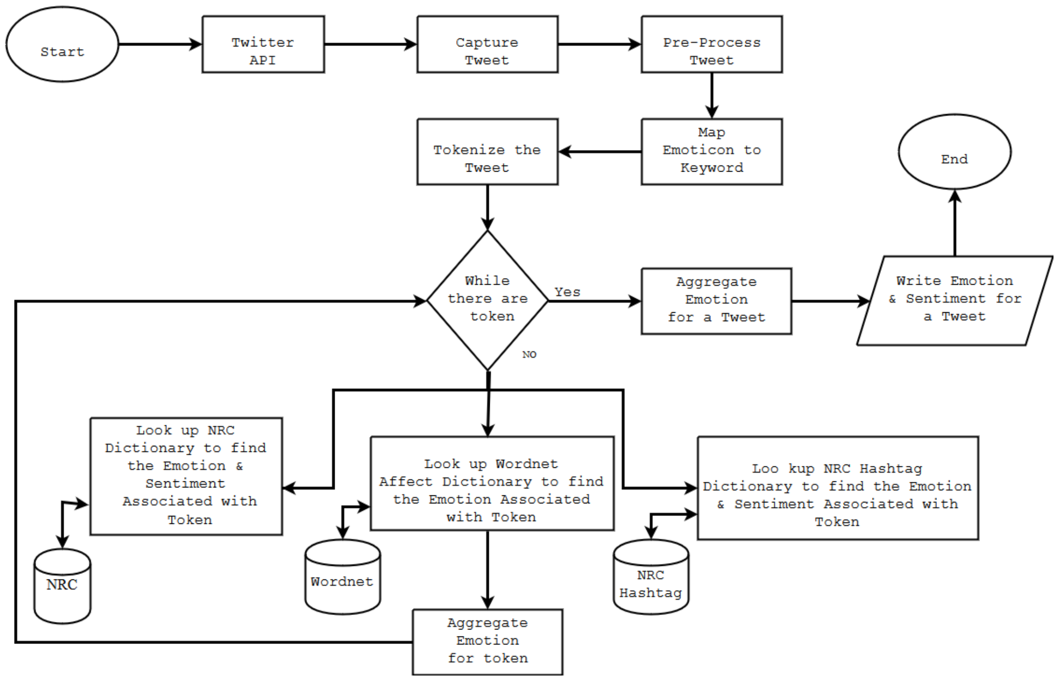

Start
Twitter API
Capture Tweet
Pre-Process Tweet
Map Emoticon to Keyword
Tokenize the Tweet
While there are token
Yes
NO
Aggregate Emotion for a Tweet
Write Emotion & Sentiment for a Tweet
End
Look up NRC Dictionary to find the Emotion & Sentiment Associated with Token
Look up Wordnet Affect Dictionary to find the Emotion Associated with Token
Loo kup NRC Hashtag Dictionary to find the Emotion & Sentiment Associated with Token
NRC
Wordnet
NRC Hashtag
Aggregate Emotion for token

Figure 6.4: Flow Chart of the Emotion and Sentiment Extraction

6.4 Model Selection

6.4.1 Model selection for Dependent variable

The dependent variable *size*, which represented the number of retweets, is a positive non-zero number (due to the imposed cutoff of 5). Thus we considered a number of statistical models suitable for count data to help explain the rate of an event, which in the given case was the number of times the tweet had been retweeted during the observation period. Due to the data being positively skewed (see Figure 6.1) we would typically use a Poisson model, where the model predicts the number of occurrences of an event. However, on inspecting the data it was observed that the dependent variable Size mean and variance were 86.53 and 667,804 respectively for the malicious sample, and 90.86 and 1,160.23 respectively for the benign sample, both indicating over-dispersion. Therefore a negative binomial model was selected. Given the lack of zeros in the dependent variable we used the zero-truncated variant of the negative binomial model (ZTNB).

$$Pr(y_i|y_i > 0) = \frac{(\Gamma(y_i + \alpha^{-1})/y_i!\Gamma(\alpha^{-1}))(\alpha^{-1}/(\alpha^{-1} + \mu_i))^{\alpha^{-1}}(\mu_i/(\alpha^{-1} + \mu_i))^{y_i}}{1 - (1 + \alpha\mu_i)^{\alpha^{-1}}} \tag{6.1}$$

$$E(y_i|y_i > 0) = \frac{\mu_i}{Pr(y_i > 0)} = \frac{\mu_i}{1 - (1 + \alpha\mu_i)^{\alpha^{-1}}} \tag{6.2}$$

$$Var(y_i|y_i > 0) = \frac{E(y_i|y_i > 0)}{Pr(y_i > 0)^{\alpha}}[1 - Pr(y_i = 0)^{\alpha+1}E(y_i|y_i > 0)] \tag{6.3}$$

$$L = \prod_{i=1}^{N} Pr(y_i|y_i > 0) \tag{6.4}$$

$$= \prod_{i=1}^{N} \frac{(\Gamma(y_i + \alpha^{-1})/y_i!\Gamma(\alpha^{-1}))(\alpha^{-1}/(\alpha^{-1} + \mu_i))^{\alpha^{-1}}(\mu_i/(\alpha^{-1} + \mu_i))^{y_i}}{1 - (1 + \alpha\mu_i)^{\alpha^{-1}}} \tag{6.5}$$

$$log(\mu_i) = \beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i} + \cdots + \beta_k X_{ki} \tag{6.6}$$

The zero-truncated models calculate the probability of response variable (size- number of retweets a tweet got) based on positive count data using Bayes's Theorem [87, 133, 92]. Above shows the probability mass function (see equation 6.1), mean (see equation 6.2), variance (see equation 6.3), likelihood function (see equation 6.4) and response surface (see equation 6.6) of a zero- truncated negative binomial model. Where $Pr(y_i|y_i > 0)$ is the probability mass function of the zero truncated negative binomial distribution, $E(y_i|y_i > 0)$ is the expectation of zero-truncated negative binomial distribution, $Var(y_i|y_i > 0)$ is the variance of zero-truncated negative binomial distribution, α is the over-dispersion parameter, **L** is the likelihood function, μ_i is the estimated retweet count for the i^{th} observation, y_i is the observed retweet count for the i^{th} observation, *k* is the parameter coefficient of the k^{th} predictor variable ($k = 0$ for intercept), X_{ki} is the value of the k^{th} predictor variable (Hashtag, Mentions, Friends, Followers, Account Age, Anger, Anticipation, Disgust, Fear, Joy, Sadness, Surprise, Trust, Negative, and Positive) for the i^{th} observation.

6.4.2 Survival Model - Cox Proportional Hazard Regression

Having identified the model to understand the factors associated with the retweet volume, the next step was to identify the factors that were associated with the Survival of a tweet containing a malicious URL. Survival analysis involves the modelling of time to event data; in this context, failure of a tweet to be retweeted in a time-limited window was considered an ”event”. Survival analysis was conducted to establish the factors that could be associated with the duration of time until people stopped retweeting a malicious or benign URL. This allowed those factors to be understand that increased or decreased the *hazards of survival*.

For instance, the interest was in identifying if emotions (e.g. anger or fear) had any effect on the hazard rate for information flow survival. Put another way, does the lifetime of a tweet (lifetime defined as the time from the first tweet to the last retweet) increase when anger is expressed in the text content? In order to identify the explanatory factors Cox's proportional hazards model [7] was chosen. Cox's model produces a survival function that predicts the probability that a retweet is made in the present time-frame for the given values of the predictor variables. Given the predictor variables *X* at a given time *t*, the survival function can be defined as $\lambda(t|X)$ where

$$\lambda(t|X) = \lambda_0(t)exp(\beta_1 X_1 + \beta_2 X_2 + ...\beta_n X_n) \tag{6.7}$$

Based on this, the partial likelihood for X can be calculated using:

$$L(\beta) = \sum_{i:C_i=1} \frac{\theta_i}{\sum_{j:Y_j \geq Y_i} \theta_i} \tag{6.8}$$

where, for a given tweet i, C_i is an indicator of the time corresponding to the tweet and $\theta_i = exp(\beta_1 X_1 + \beta_2 X_2 + ...\beta_n X_n)$

6.4.3 Kaplan–Meier Estimation

In order to show the impact of the survival rate when an emotion or sentiment is detected, we used the Kaplan-Meier estimation model to plot the survival function. The generated plot consisted of a declining horizontal step. The estimator could be represented by

$$\hat{S}(t) = \Pi_{t_i<t}\frac{n_i - d_i}{n_i} \tag{6.9}$$

where *ni* is the number of tweets that were retweeted and *di* is the number of tweets that failed to be retweeted at time t_i.

6.5 Results

The dataset used for the analysis was collected during seven popular sporting events making the analysis not specific to one event but generalising it beyond a single event. The seven sporting event that were used are namely (i) Fédération Internationale de Football Association (FIFA) World Cup of 2014, (ii)Superbowl 2015, (iii) The Cricket World Cup 2015, (iv) Rugby World Cup 2015, (v) Superbowl 2016, (vi)The European Football Championships 2016 and (vii) The Olympics 2016. This made the dataset geographically diverse with each event catering to a different audience. Tables 6.4 and 6.5 give a summary of the results for the zero-truncated negative binomial models built for our dependent variable Size using the malicious and the benign datasets. Tables 6.6 and 6.7 summarise the results of the Cox proportional hazard regression models for the dependent variable Survival.

The independent or predictor variables were divided into three categories: social, emotion and sentiment factors. Several statistically significant associations were observed between the predictor and the dependent variables. For count data models in place of coefficients the incident rate ratio (IRR) is shown for each predictor. The IRR is derived by the exponentiation of the zero truncated negative binomial regression coefficients, allowing for the interpretation of retweet incidence rates (as opposed to logs of expected retweet counts). We can therefore use the IRR to report the strength of causal associations between certain factors and the information flow size, enabling us to identify quantitatively which factors are more important than others. In terms of the results, the magnitude of the effect of the variables of interest (emotions, sentiment and social content) on the retweets was expressed as a percentage change in the incident rate of a retweet when all the other factors in the model were held constant. An IRR of more than one indicated that the percentage change in the incidence of a retweet had increased, whereas an IRR of less than one indicated the reverse. For example, in Table 6.4, IRR for Surprise is 1.7851390, which is greater than one. So the percentage change in the IRR is calculated as:

$$\%increase = (IRR - 1) * 100 = (1.7851390 - 1) * 100 = 78.51390\% \qquad (6.10)$$

Which is interpreted as, by holding all other factors constant tweets that contained more words associated with the emotion Surprise were more likely to be retweeted by 78.51390%. Similarly, in Table 6.4 percentage change in the incident rate for Hashtag (IRR=0.893171), where IRR is less than one, is calculated in the following way

$$\%decrease = (1 - IRR) * 100 = (1 - 0.8931710) * 100 = 10.6829\% \qquad (6.11)$$

Which is interpreted as, by holding all other factors constant tweets that contain higher number of hashtags were less likely to be retweeted by 10.6829%

6.5.1 Dependent Variable- Size

Social Factors

Benign Dataset

Table 6.4: Size model Results for Tweets containing Benign URL

Predictors	IRR	**Std. Err**	**Z**	**Sig.**
Social Factors				
Hashtag	0.8931710	0.0205727	-4.900	0.000
Mentions	0.6581943	0.0333803	-8.250	0.000
Friends	0.9999995	0.0000045	-0.120	0.907
Followers	1.0000010	0.0000001	9.370	0.000
Age of User Account	0.9993696	0.0001616	-3.900	0.000
Emotion				
Anger	0.9553148	0.2145193	-0.200	0.839
Anticipation	0.7157636	0.1030828	-2.320	0.020
Disgust	0.8145451	0.2109367	-0.790	0.428
Fear	0.8445490	0.1555862	-0.920	0.359
Joy	0.8073432	0.1528402	-1.130	0.258
Sadness	1.0266900	0.2111274	0.130	0.898
Surprise	1.7851390	0.3412885	3.030	0.002
Trust	0.8126954	0.0880218	-1.910	0.056
Sentiment				
Negative	0.5349507	0.0989301	-3.380	0.001
Positive	1.0844390	0.1233625	0.710	0.476

Four social factors were statistically significantly associated with the dependent variable size. The number of hashtags and mentions in a tweet both emerged as negatively associated with size. For hashtag the IRR was 0.8932, *Z* was -4.9 and p<0.00 (see Table 6.4). The results show that for every increase in hashtags within a tweet, the tweet is approximately 11% less likely to be retweeted. Similar results were observed for mentions and the age of the user account. For mentions, the IRR was 0.6582, $Z = -8.25$, p<0.00, and for the age of the user account the IRR was 0.9994, $Z = -3.9$ and p<0.00. Thus, for each increase in user mentions, the tweets were 34% less likely to be retweeted. Every increase in the age of an account made it 0.0006% less likely that tweets posted by the account would be retweeted. The older the account, the less likely its benign tweets are to be retweeted. The number of followers a user had was also statistically significant: with an IRR of 1.0000010, Z equalled 9.37 and p<0.00, thus showing that for every increase in follower numbers, the likelihood of retweet increased. This follows expectations and is due to increased social capital; therefore the tweet is exposed to more people.

Malicious Dataset

Two variables from the set of independent variables in social factors were statistically sig-

Table 6.5: Size model Results for Tweets containing Malicious URL

Predictors	IRR	**Std. Err**	**Z**	**Sig.**
Social Factors				
Hashtag	1.2493	0.0783	3.5500	0.0000
Mentions	0.9678	0.0892	-0.3500	0.7230
Friends	1.0000	0.0000	-0.8600	0.3880
Followers	1.0000	0.0000	2.7300	0.0060
Age of User Account	0.9999	0.0003	-0.4500	0.6540
Emotion				
Anger	0.6585	0.1471	-1.8700	0.0610
Anticipation	1.2971	0.1122	3.0100	0.0030
Disgust	0.9395	0.1492	-0.3900	0.6950
Fear	2.4397	0.3817	5.7000	0.0000
Joy	0.9353	0.0840	-0.7400	0.4560
Sadness	1.3274	0.1731	2.1700	0.0300
Surprise	0.6941	0.0471	-5.3800	0.0000
Trust	0.6883	0.0699	-3.6800	0.0000
Sentiment				
Negative	1.0526	0.2703	0.2000	0.8420
Positive	0.9531	0.1278	-0.3600	0.7200

nificantly associated with the dependent variable size. For hashtags, the IRR was 1.2493, Z 3.55 and p<0.00 (see Table 6.5). This association showed that for every increase in hashtags in a tweet, the chances of a retweet increased by around 25%. We also observed that the number of followers that the tweeter had was also statistically significant and it was observed an IRR of 1.00, Z equal to 2.73 and p<0.00. It was also observed that, even though IRR for *followers* was statistically significant, it did not affect the retweeting behaviour (IRR=1).

Emotion and Sentiment

Benign Dataset

Two positive emotions out of the eight primary emotions used as the independent variable were found to be statistically significant. Results showed an IRR of 0.7157636, Z -2.320 and p<0.02 for anticipation. For every increase in words relating to anticipation, the tweet was 28% less likely to be retweeted. However, for every increase in the emotion surprise, tweets were 78% more likely to be retweeted (IRR=1.7851390,Z=3.030 and p<0.02) than tweets not containing surprise (see Table 6.4). Among the independent variables in the sentiment set, negative sentiment (IRR=0.5349507, Z=-3.38 and p<0.001) was statistically significant, showing that for benign tweets posted during a sporting event, every increase in words containing negative sentiment is associated with a 47% reduction in the likelihood of being retweeted.

Malicious Dataset

Five out of the eight emotions were found to be statistically significant, whereas no significant association was found for sentiment. From the set of negative emotions, fear and sadness were positively associated with size, with fear having an IRR of 2.4397, Z of 5.7 and p<0.00, and sadness having an IRR of 1.3284, Z of 2.17 and p<0.03 (see Table 6.5). In malicious tweets, for every increase in the words relating to fear, the likelihood of retweet increased by 143%. For sadness, each increase was associated with a 33% increase in the likelihood of being retweeted. Anticipation evokes the feeling of excitement, whereas surprise is the reaction to what is unexpected and these were both found to be statistically significant. For anticipation, the IRR was 1.2971, Z 3.01 and p¡0.03 and for surprise the IRR was 0.6941, Z -5.38 and p<0.00 (see Table 6.5).

This means that malicious tweets which expressed anticipation were 30% more likely to be retweeted for each additional word containing this emotion, and those that expressed surprise were 31% less likely to be retweeted. Anticipation was positively associated with the number of retweets, whereas surprise was negatively associated, compared to the results from the benign dataset sample. From the set of positive emotions, trust was observed to be statistically significant. It was observed to have a negative association with information flow, having an IRR of 0.6883, Z of -3.68 and p<0.00.

We interpret this as meaning that for every increase in a text representative of trust, malicious URLs were 31% less likely to be propagated. The Bayesian information criterion for the full model was observed to be 10,660.43 and the log-likelihood to be -5270.41, suggesting a good fit to the data.

6.5.2 Dependent Variable -Survival

Tables 6.6 and 6.7 show the results obtained from the Cox proportional hazards model for both malicious and benign tweets. Considering the diversity of each sporting event regarding the length of playing time (from 90 minutes for a football game to around 480 minutes for a cricket match), we aimed to understand which tweets had information that survived longer than 24 hours after the end of the sporting event. Therefore, a Cox proportional hazard model was created for a 24-hour time window. The results from using the model indicated several statistically significant associations between the dependent variable (survival) and the predictive factors. When the model was used to explain the proportional hazards, a positive β indicated an increase in the hazard of survival, meaning that it reduced the chances of survival for the information flow and a negative β indicated the reverse.

Table 6.6: Survival Model Results for Tweets containing Benign URL

24 hours Window				
Predictors	**Coef**	**Std. Err**	**z**	**Sig**
Social Factors				
Hashtag	0.0505660	0.0167934	3.010	0.003
Mentions	-0.1661053	0.0357883	-4.640	0.000
Friends	0.0000012	0.0000018	0.660	0.509
Age of User Account	0.0012092	0.0001006	12.020	0.000
Followers	-0.0000002	0.0000001	-3.480	0.000
Emotion				
Anger	0.1982328	0.1531408	1.290	0.196
Anticipation	-0.0820958	0.0956012	-0.860	0.390
Disgust	0.1259461	0.1607216	0.780	0.433
Fear	-0.0869308	0.1282319	-0.680	0.498
Joy	0.3235211	0.1259961	2.570	0.010
Sadness	0.2598578	0.1410289	1.840	0.065
Surprise	-0.2424558	0.1399941	-1.730	0.083
Trust	0.0393096	0.0743090	0.530	0.597
Sentiment				
Negative	-0.1533467	0.1219457	-1.260	0.209
Positive	-0.2803509	0.0833525	-3.360	0.001

Social Factors

Benign Dataset

Holding all the factors constant, we found hashtags ($\beta = 0.05056, z = 3.01$) and the age of the account in question ($\beta = 0.0012092, z = 12.02$) to be statistically significant and positively associated with hazards of survival. The results showed that benign tweets that contained a higher number of hashtags or were posted by accounts that were recently created had less chance of survival i.e. they would be retweeted for a shorter period than those with less hashtags or with older accounts. We also found that user mentions ($\beta = -0.166, z = -4.64$) and the number of followers that a user had ($\beta = -0.0000002, z = -3.48$) were statistically significant and negatively associated with hazards to survival. Moreover, we found positive sentiment to be statistically significant and associated with decreased hazards to survival. Figure 6.5 illustrates that, using

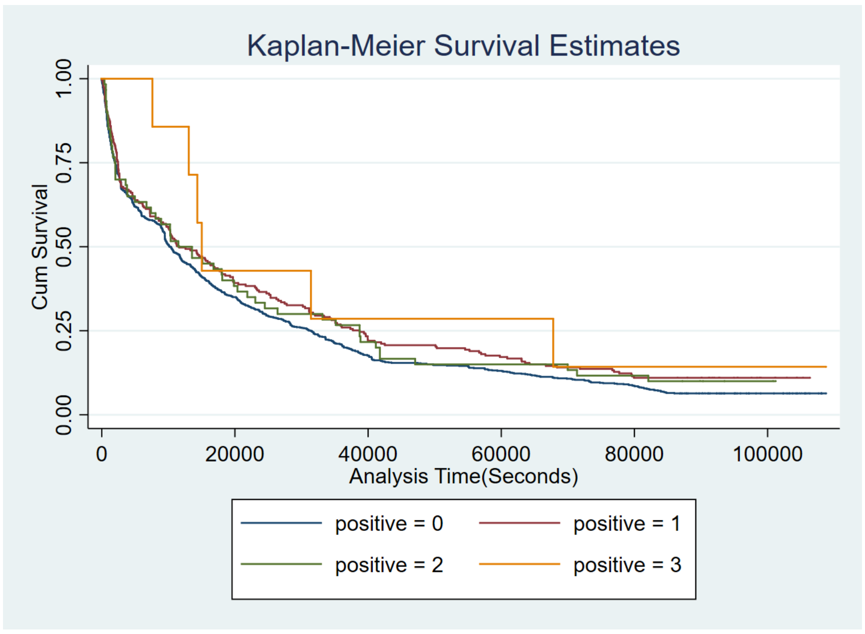


Figure 6.5: Survival rate for positive sentiment in benign tweet sample

Kaplan-Meir survival estimates, benign tweets with higher numbers of positive words have an increased chance of survival over longer periods.

Malicious dataset

Holding all the factors constant, we found user mentions ($\beta = 0.13513360, z = 3.17$)

Table 6.7: Survival Model Results for Tweets containing Malicious URL

24 hours Window				
Predictors	**Coef**	**Std. Err**	**z**	**Sig**
Social Factors				
Hashtag	-0.02988310	0.02786000	-1.070	0.283
Mentions	0.13513360	0.04256820	3.170	0.002
Friends	-0.00001060	0.00000539	-1.970	0.049
Age of User Account	-0.00008420	0.00014620	-0.580	0.565
Followers	0.00000017	0.00000006	2.760	0.006
Emotion				
Anger	0.35190140	0.09892910	3.560	0.000
Anticipation	-0.02911340	0.04077730	-0.710	0.475
Disgust	-0.00215150	0.07319090	-0.030	0.977
Fear	-0.19202340	0.06798030	-2.820	0.005
Joy	0.01330880	0.04505960	0.300	0.768
Sadness	0.04871910	0.07660280	0.640	0.525
Suprise	0.00820430	0.05620190	0.150	0.884
Trust	-0.04157660	0.06243850	-0.670	0.505
Sentiment				
Negative	0.02373590	0.11119810	0.210	0.831
Positive	0.10034690	0.07497770	1.340	0.181

and the number of followers a user had ($\beta = 0.00000017, z = 2.76$) to be statistically significant in predicting hazards to survival. Both of them were found to be positively associated with the hazards to survival. Results showed that the more a cybercriminal uses user mentions in a tweet (e.g. in trying to target users) or has an extensive social network (often a sign of bots who buy followers), the more it reduced the chance of survival. A negative association was observed by the number of friends that a user had ($\beta = -0.00001060, z = -1.970$), showing that the more friends a user has, the higher the chances of survival of a tweet.

Emotion and Sentiment

Benign dataset

The independent variables representing emotion and sentiment in a tweet were derived from the contents of the tweet, based on the words expressing them. Survival models were built using these independent variables. The results showed that only the positive emotion of joy and positive sentiment were statistically significant in predicting hazard of the survival of the information flow. We observed a positive association for joy ($\beta = 0.3235211, z = 2.570$), indicating that the chances of survival were reduced if the tweet reflected joy. However, tweets that reflected a positive sentiment showed a negative association $\beta = -0.2803509, z = -3.360$). Results showed that the more positive tweets had the greater chance of survival.

Malicious dataset

Sentiment alone was not found to be statistically significant. However, we found two emotions to be statistically significant in predicting hazard for the survival of the information flow. The results showed that anger ($\beta = 0.352, z = 3.56$) was positively significantly associated, indicating that the chances of survival were reduced if the tweet reflected anger. However, tweets that reflected fear ($\beta = -0.192, z = -2.82$) showed a negative association, indicating that the chance of survival of a malicious tweet increases if a cybercriminal posts an intimidating tweet. Figure 6.6 illustrates the Kaplan-Meir

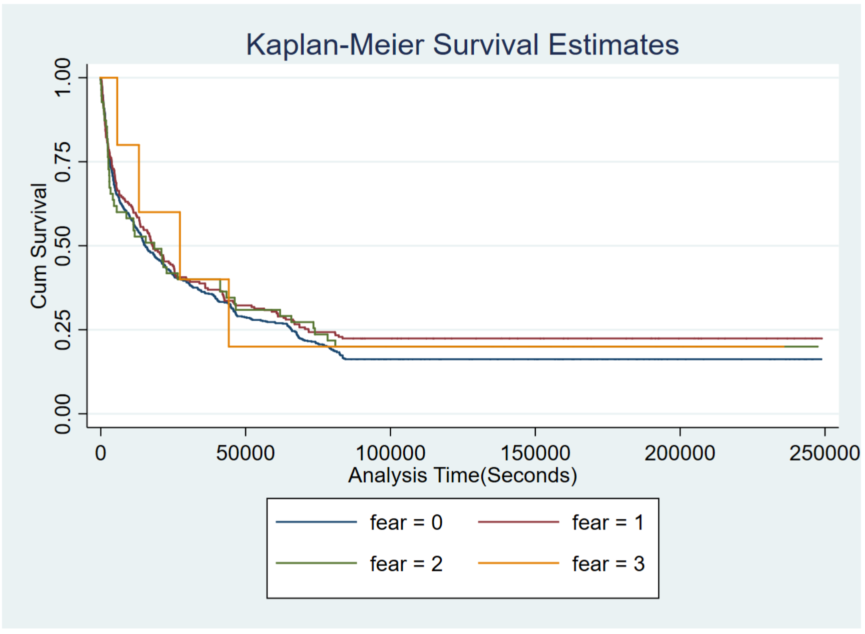


Figure 6.6: Survival rate for fear in malicious dataset

survival estimates for those tweets that reflect the emotion fear. The levels of fear (0-3) represented the number of words related to this emotion in the tweet. The results showed that the more often words associated with fear appeared in a malicious tweet, the higher were the chances of its survival. Fear spreads for a longer time than any other emotion in malicious tweets.

6.6 Discussion

We collected Twitter data around seven sporting events using event-specific hashtags. Sporting events had been chosen specifically because they attracted a large number of users which gave cybercriminals an opportunity to spread drive-by download attacks by obfuscating malicious URLs in tweets. A subsample of 270,00 tweets was randomly created from the collected data sample of around 3.5 million tweets, that were pre-processed to remove the retweeted tweets before the unique URL was extracted from them. The resultant sample of around 31,171 unique URLs was later passed on to Capture HPC, a high interaction honeypot, that was set up to distinguish malicious tweets from benign. Once a URL was passed to the Capture HPC server, it interacted with the Web server for a limited period and, based on changes made to the client machine, it classified the URL into malicious (drive-by download attack occurring from the endpoint of a URL) or benign. The changes made to the client machine clarified that Capture HPC identified around 6,122 malicious and 25,049 benign URLs. Across all the seven events we identified all the tweets that contained these malicious and benign URLs, including retweets, to understand the propagation of a drive-by download attack on Twitter. The research aimed to identify social and content factors, such as the emotion and sentiment in the tweet, that was associated with the propagation of malicious URLs. With the nature of the sample data in mind, we chose the zero-truncated negative binomial method to model *size* and the Cox proportional hazard model to calculate the *survival* as the dependent variables - and social and content features as the independent variables.

In line with previous research on the virality of news [211, 17] and spam detection [95] in online social networks, several significant associations for emotions and sentiment were revealed between the information size and survival. Among the social features, hashtags stood out, being created by users to give context to their post and link them to related topics to reach a targeted audience, increase traffic to their post, and in turn increase interactions and the probability that their tweet will be retweeted. A recent report on the engagement of users on online social platforms revealed that though hashtags provide context and are thus important elements in a post, they reduce user engagement if their number increases beyond a certain point [163]. Interestingly, our results for tweets that contained benign URLs were in line with the report [163]. The results showed that tweets classified as benign do not engage users' interest and are 11% less likely to be retweeted if they contain more hashtags. This contradicted our finding for those tweets that were classified as malicious. Our results showed that malicious tweets were 24% more likely to be retweeted if they contained more hashtags. One explanation for this could be that hashtags which trigger emotional responses are added to tweets to gain popularity or to engage users, as has been shown in earlier work where a relationship was identified between emotion and content sharing [17].

For benign tweets the number of followers of the posting user was positively associated with the chances of them being retweeted (see Table 6.6). However, though followers were statistically significant in the propagation of malicious tweets, the effect size was very small suggesting cybercriminals do not depend heavily on their followers to propagate malware and may seek other techniques for this purpose, including the use of content features (sentiment or emotion), embedding hashtags that highlight the tweet, or using techniques such as paying for retweets [61, 42] from black market services such as Like4Like [62] or YouLikeThis[159] .

Information that is novel attracts people [101] and things that are attractive are worth sharing on online social platforms [17]. Information that is found novel is also considered valuable and surprising. Based on this principle, Berger et al showed that content on online social networks that evokes high arousal emotions such as *'awe'* have a higher probability of being propagated by the action of sharing [17]. Since the emotion *'awe'* can be derived from the root emotion surprise [191] (see Figure 6.2), we found that tweets which were classified as benign and contained keywords associated with surprise were

78% more likely to be retweeted. Even though a higher number of tweets (33% malicious compared to 8% benign) contained one or more keywords associated with the emotion surprise in a malicious dataset, it was not statistically significant for the size of the information flow. This suggests that it was not one of the driving factors behind a high retweet count. Rather, it was negative emotions such as fear and sadness that were found to be statistically significant for the size of the information flow. The results showed fear to have the highest incident rate ratio (2.4), meaning that each tweet that reflected fear was 114% more likely to be retweeted. A comparison of sample size revealed that 30% of the tweets from the malicious data sample contained one or more keywords associated with fear compared to 8% from the benign data sample, suggesting that a higher number of words associated with the emotion fear were used in constructing tweets with a malicious link. Even though negative emotion was present in the benign dataset it was not found statistically significant for the size of the information flow in the dataset categorised as benign.

To investigate further the choice of keywords used that helped a tweet to gain popularity or be retweeted, a world cloud (see Figure 6.7 and Figure 6.8) was created from the tweets that were categorised as malicious and and tweets that were categorised as benign . On

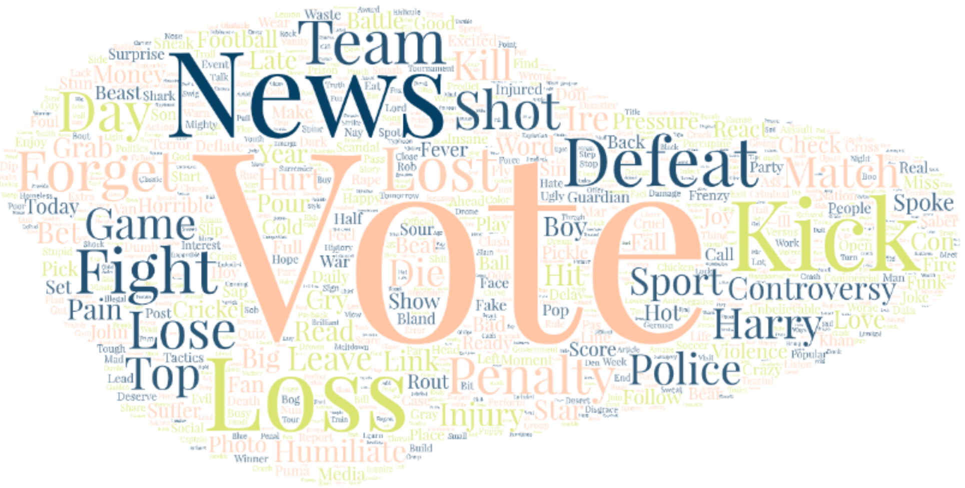


Figure 6.7: Word Cloud of Malicious Tweet

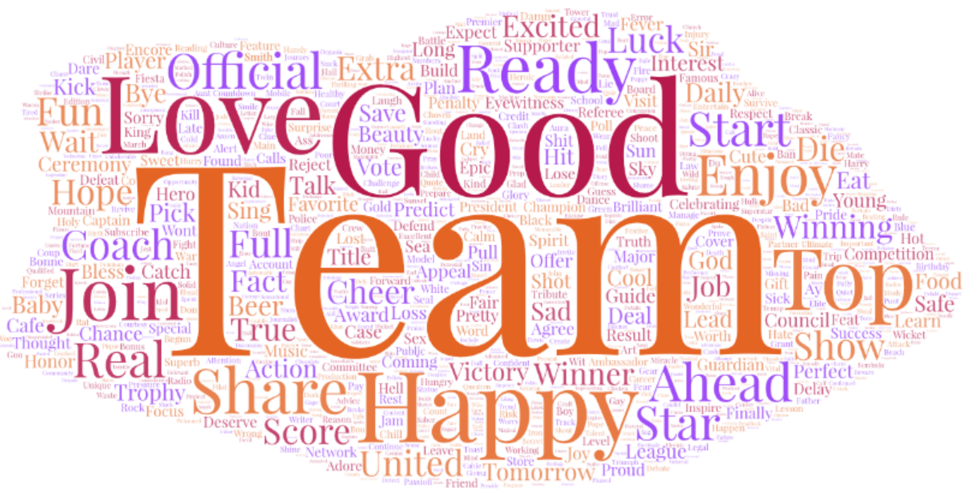


Figure 6.8: Word Cloud of Benign Tweet

closer inspection words such as ”kill”, ”fight”, ”shot”, ”controversy” etc. were frequently observed in tweets that contained malicious URLs. Whereas words such ”Team”, ”love”, ”happy”,”good”, ”enjoy”,”fun” were found in benign tweets. This suggests that carefully selected words were being used in the formation of these tweets, where a keyword could trigger emotional arousal using negative emotions such as fear, anger, or sadness that could encourage the propagation of malicious tweets. [17, 211]. We further investigated

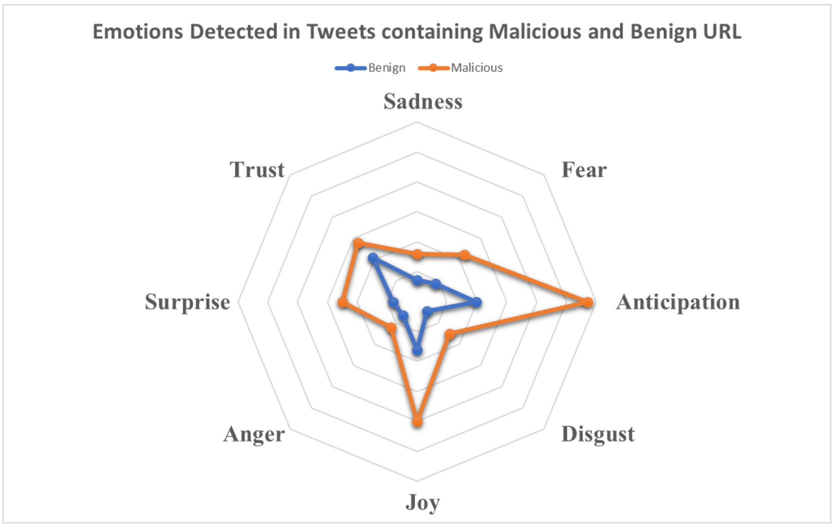


Figure 6.9: Emotions captured in those Tweets categorised as malicious (N=1137) and benign (N=1862)

the number of words used in tweets that created emotional arousal and found that tweets that were classified as malicious contained more words associated with emotions than did the tweets classified as benign (see Figure 6.9).

Even though the collective intensity (total number of words associated with emotions) of positive emotions such as anticipation, surprise, trust and joy were higher than those of the negative emotions, it was the negative emotion such as fear that were associated with size and the survival of information flow for tweets classified as malicious. This association implies that the associative factors were higher for emotion rather than on *followers* for the propagation of tweets containing malicious links. A similar association was found between content of false news and emotions, where a study showed it was negative emotions that assisted more than positive in propagating fake news [211].

In addition to the ZTNB model built to predict the number of retweets (size) we built a Cox proportional hazard model for both benign and malicious tweets to measure the lifetime of the information flow. A number of social factors were found statistically significant, where the number of hashtags used in a tweet showed similar traits in the size of the information flow. That is, as the number of hashtags in a tweet increased, the continued engagement of users reduced. This is supported by a report on user engagement, that suggested an inverse relationship between number of hashtags in a post and user engagement with that post [163]. However, this trait was seen only in tweets classified as benign and no statistical significance was seen between the survival of a tweet categorised as malicious and the number of hashtags used. Whereas, user mentions were statistically significant in both datasets and had an opposite relationship. In the malicious tweets it was positively associated, suggesting that with an increase in user mentions the chances of survival decreased. This is possibly because cybercriminals may misuse the user mention option by mentioning popular users to attract attention to their tweet.

However, the survival of a tweet classified as benign is linked to user mentions and the number of followers, suggesting that users who were mentioned shared a trust relationship with the person mentioning them, thus attracting the retweet as demonstrated in related work where authors showed cybercriminals have exploited the *trust* relationship [73] between users by using accounts that appear trustworthy or by mentioning users in a tweet to aid their propagation. The results for malicious datasets showed a negative association with friends (the more the friends, the higher the chances of survival) and a positive association with followers (fewer followers implying a higher chance of survival), suggesting that malicious tweets from accounts that have a low ratio of followers to friends have a higher chance of survival. This could be one of the tactics employed by cybercriminals to prevent detection, since the follower to friend ratio is identified as one of the key features for flagging an account as malicious [192, 227, 183]. Experimental results show that a tweet categorised as malicious will survive longer if posted by an account that has a many friends and uses fewer user mentions. However, these features were not statistically significant for tweets classified as benign.

In terms of emotions that were associated with the survival of a tweet, the results were similar to those on the size of the information flow. For a benign dataset, positive sentiment, which is the emotional effect of the tweet on its reader, was found statistically significant for its survival. This was similar to the size of information flow for benign tweets, where tweets containing emotions with positive associations were likely to be retweeted. Similarly, it was the negative emotions that influenced the survival of a tweet classified as malicious, where, like the size of the information flow, fear stood out from the other negative emotions. Tweets that contained keywords associated with fear were more likely to survive the twenty-four hour window after the sporting event and more likely to be retweeted.

6.7 Conclusion

This study has analysed malware propagation across seven different sporting events covering a diverse group of users. Our results show that there is a statistically significant association between the social and emotional factors derived from a tweet captured during a sporting event. In this chapter, it was observed that malware propagation was not strongly associated with the number of followers that a user had. The stronger association was towards content driven features, such as emotions and the choice of words associated with emotions that were used to compose a tweet or create hashtags. Even though the malicious dataset had lower numbers, the cumulative intensity of emotions (see Figure 6.9) was much higher than in the tweets containing benign tweets. The results showed that tweets that contain malicious links are associated with negative emotions, particularly the emotion fear, for their retweet likelihood (virality) and survival. Whereas, in tweets that are classified as benign, it was the positive sentiment and high arousal emotions such as surprise that were associated with the size and survival of the information flow.

Chapter 7

Conclusion and Future Work

7.1 Thesis Summary

With 80% of Internet users now active in OSNs, we have seen a 56% rise in Web based attacks reported by Symantec [196]. Year by year the increase in cyber attacks continue to pose serious threats to information security [106], resulting in a loss of millions [158]. Cyber criminals who earlier had been known to favour popular platforms such as emails for their nefarious activities [55], have shifted their focus towards OSNs, which was evident by the growing number of web based attacks being reported on them [137, 121, 86, 153, 102, 184]. The motivation of this thesis was to contribute to the growing literature on fighting these web-based attacks, focusing particularly on drive-by download attacks, that accounted for 48% of web-based attacks [175].

In order to counter these attacks, researchers have proposed malware detection models based on OSN accounts characteristics [12, 183, 227, 126, 144, 43, 192, 30, 187, 19], malware code/behaviour [100, 210, 137, 135, 31, 125, 65, 103, 108, 114, 11, 116, 149, 22] and by deploying honeypots [5, 160]. In spite of numerous detection models developed to counter such attacks, cybercriminals continue to develop new ways to overcome Twitter's policy and beat the detection models proposed to counter them. Obfuscating techniques such as anti-symbolic execution obfuscation [10], anti-fuzzing [84], encryption [150], polymorphism [150] were used to evade detection from models developed on malware code/behaviour. Whereas, cybercriminals evaded models developed on OSN accounts characteristics by modifying their attacking behaviour [227].

The focus of research prior to this thesis had been on *detecting malware* on OSNs, however, considering 335 million active users on Twitter [118], an average rate of 6,000 tweets per second [145] and the highest retweet of 4.1 million recorded for a single tweet [130], drive-by downloads attacks have the potential to expose the malware to millions of users before a malware detection model could detect it. In such a volatile environment we proposed the need to adopt a proactive strategy to defend against malware attacks.

In order to keep pace with cybercriminals and overcome challenges faced by models developed by analysing malware code/behaviour or OSN characteristics, first of all, evidence of popular events being used as a delivery medium was gathered in chapter 3. Data from Twitter were collected from seven popular and diverse events over a period of 3 years. From the collected tweets, a sub-sample was randomly created, and the extracted URLs were passed into the Capture HPC [27], a high interaction honeypot set up to identify malicious URLs. A malicious URL is defined as one which points to a malicious server or website from which a drive-by download is carried out. Results from the experiment showed that 12.76% of total URLs processed by using data from all the seven events were classified as malicious, meaning a drive-by download attack occurred while visiting these websites. Using the list of malicious and benign URLs, in chapter 4, a novel detection model was proposed that classifies a URL into malicious/benign based on system activities, thus shifting the focus from analysing malware code to observing changes to a user's system by the malware. The proposed model was successful in classifying URLs into malicious and benign with an F-measure of 0.81 during training and 0.71 while testing on an unseen dataset.

In order to extend the detection model so that it could anticipate a drive-by download attack, a novel prediction model was proposed in chapter 5, to predict drive-by download attacks on Twitter. The proposed model was based on OSN characteristics and machine activities leading to a drive-by download attack. Thus, moving beyond the post-execution classification of such URLs as malicious, to predict that a URL will be malicious with 0.99 F-measure (using 10-fold cross-validation) and 0.833 (using an unseen test set) at 1 second into the interaction with the URL. This finding provides a basis from which to kill the connection to the server before an attack has completed and proactively blocking and preventing an attack, rather than reacting and repairing at a later date.

Considering the aim of the cybercriminal is to increase damage through maximum exposure, Chapter 6 focuses on uncovering social and content-based factors that aid in the propagation of drive-by download attacks by the action of retweeting. We answered questions including (i) why are certain post shared more than others?; (ii) is virality driven by physiological arousal?; and (iii) do emotion and sentiment represented in a post aid in its propagation? Emotion and sentiment were particularly chosen because it has been well established that people transfer positive and negative moods to one another in a form of emotional contagion [89] and the posting behaviour of a user varied according to the emotional content received [120]. The findings of Chapter 6 provide the first study that links emotions and sentiment to the propagation of tweets containing malicious URLs on Twitter. The experimental results showed that there was a statistically significant association between the social and emotional factors derived from a tweet captured during a sporting event and tweets that contain malicious links are associated with negative emotions, particularly the emotion fear, for their retweet likelihood (virality) and survival.

7.1.1 Contributions

The research conducted and explained in this thesis contributes towards the evolving field of detecting and managing drive-by download attacks on Twitter and can be summarised in two contributions. Existing research that has been conducted to fight against cyber attacks and to stop them from spreading on OSNs has been studied from three perspectives: (i) analysis of social features of the accounts posting malicious URLs, (ii) constructing models based on static code analysis, and, (iii) constructing models built using dynamic code analysis. Chapter 2, focused on analysing existing models developed to detect malware on OSN and highlighted the limitations. It further studied models/techniques that have been proposed to curb the infection on OSN. The discussion on existing literature and on identifying research gaps led to open research questions. The research questions that were identified in Chapter 2, helped shape the structure of the thesis, where the primary focus has been in proposing a model to predict a drive-by download attack that can identify malicious URLs on Twitter and to identify factors that help propagation of posts that contain malicious URLs.

In order to carry out a successful drive-by download attack, cybercriminals need to expose the malware to a large number of users on OSNs. However to best of our knowledge, no research finding presented any measure of association between popular events and the scale of malicious URLs posted to Twitter. In Chapter 3 evidence that popular events on Twitter were being used to deliver drive-by download attacks was gathered.The evidence was gathered by collecting tweets containing URLs using event-specific hashtags and passing them through a high interaction client side honeypot, Capture HPC. The honeypot classified a URL as malicious or benign on the basis of client machine changes in terms of file, process and registry. Using seven diverse sporting events, 3,571,184 tweets were collected over a period of three years. From these, a random sample of 48,991 unique tweets were passed on to Capture HPC, out of which 6,879 (individual tweets) and around 105,642 (tweets including retweets) were identified to be malicious through an automated analysis of each URL.

In order to detect and counter malware on OSNs numerous models have been proposed. However, cyber criminals continue to develop new ways to overcome Twitter's policy and beat the detection models proposed to counter them. The models were primarily built using OSN accounts characteristics and by observing malware code/behaviour. Yet cybercriminals tend to modify their behaviour on OSNs, hence evade detection from models build on OSN characteristics, while models based on malware code/behaviour succumb to techniques such as anti-symbolic execution obfuscation [10], anti-fuzzing [84], encryption [150], polymorphism [150] used to obfuscate malware code/behaviour. Thus, there was a requirement to develop detection models by shifting the focus from existing malware code/behaviour features to keep pace with cybercriminals. We proposed one such approach would be to focus on the effect of malware on user systems, by observing machine activities leading to an attack. After all, malware will reveal itself eventually to harm a system even if it is using different evading techniques such as polymorphism, encryption, triggers, etc. to hide the malicious code. In Chapter 4 we proposed a novel malware detection model that classifies a URL into malicious/benign based on machine activities, thus shifting the focus from malware code/behaviour to system behaviour during malware execution for the first time. The proposed detection model presented as a result of our experiment, observes machine activities during visitation of a webpage instead of statically/dynamically analysing malware code. It was trained and tested by using data collected over popular sporting events from Twitter. The model gave an F-measure 0.81 for MLP during training the model and an F-Measure of 0.71 during testing the model on unseen dataset. However, the malware detection model could only handle 500 URLs per day and classified a URL as malicious based on activities *after an attack had taken place*. In order to increase the number of URLs to be processed without increasing the number of resources a predictive model was proposed in Chapter 5. Thus, giving our first contribution:

C1 To propose a predictive model capable of identifying a malicious URL within seconds, based on machine activities and social features, thus providing new capability to kill the connection and curtail the sequence of malicious actions rather than depending on detection and repair at significant cost and inconvenience.

The prediction model was developed using data around two global sporting events. The model was trained using a second-by-second time series of system-level activity (e.g. CPU use, RAM use, network traffic etc.) during the visitation of a Web page and that of social features derived from the meta-data of the tweet. During training, a ten-fold cross-validation was performed to train the model, and an F-measure of 0.99 was achieved by using the log files generated at 1 second into the interaction with a Web server. When tested using an unseen dataset (Olympics 2016) we achieved an F-measure of 0.833 from log files generated at 2 seconds - that is 1 second after launching the URL. The highest F-measure achieved on the unseen event was 0.862 at 5 seconds from the time the URL was launched. Furthermore, it allows us to stop the execution process with 0.833 F-measure just 1 second after clicking the URL, preventing the full execution of the malicious payload, rather than detecting the malicious action retrospectively and having to repair the system.

Even though the predictive model was capable of processing a maximum of 150,000 URLs per day per deployment, the rate at which the tweets are posted could spike during a popular event on Twitter. The processing power of the model could be increased by simultaneously deploying multiple images of the predictive model. However, even with multiple deployments of the predictive model, the processing of URLs per day might not be enough to handle the volatile load of tweets generated during the popular event. Thus, there is a need to understand the infection behaviour and the factors that contribute towards the propagation of posts containing malicious URLs.

In Chapter 6, the social and content-based factors within a tweet containing a malicious URL that aid its propagation were uncovered. In order to identify these factors, we developed evidence to answer questions including: why are certain post shared more than others?; is virality driven by physiological arousal?; and do emotion and sentiment reflecting in a post aid in its propagation? Answers to these questions could help in the future to devise a strategy to curtail the infection even when the users that were spreading are removed from the network. The factors that aid in the propagation of post containing malicious URL produces our final contribution.

C2 To understand propagation behaviour by uncovering the social and content features, particularly emotions that influence the propagation of tweets containing malicious URLs.

In order to identify the factors that aid in malware propagation, data across seven different sporting events covering a diverse group of users was collected. Experimental results show that there is a statistically significant association between the social and emotional factors derived from a tweet captured during a sporting event. It was observed that malware propagation had a stronger association towards content driven features, such as emotions and the choice of words associated with emotions that were used to compose a tweet or create hashtags. The results showed that tweets that contain malicious links are associated with negative emotions, particularly 30% of the tweets were associated with the emotion fear. Whereas, in tweets that are classified as benign, it was the positive sentiment and high arousal emotions such as surprise that were associated with the size and survival of the information flow.

7.2 Future Work

The contributions made through this thesis can also be used as basis for future work both in terms of improving the performance of the predictive model to identify tweets containing malicious URLs and to further investigate malware propagation.

7.2.1 Real world application

The future direction of the proposed predictive model lies in the development of real world application that is capable of identifying malicious URLs being posted during popular events on Twitter in real time. However, one of the main challenges while dealing with Twitter data is handling large number of tweets in real time. Also, considering the volatility of the tweets being posted during popular events the model should be either capable of adding resources or creating multiple instances of itself to cope with the load. One of the ways to overcome the dynamic scalability issue is to deploy the model on a cloud environment using an overlay framework, such as, CometCloud [58] that allows creating dynamically multiple instances of the predictive algorithm. CometCloud is open source and is specifically focused on supporting integration of distributed computer platforms, enabling it to create multiple instances of the predictive model to quickly classify a URL. It achieves this through the use of the Comet coordination ”spaces” — an abstraction based on the availability of a distributed shared memory that all users and providers can access and observe, enabling information sharing by publishing requests/offers to information to this shared memory.

CometCloud deployment consists of a Master and number of Workers. The Master is responsible for managing the interaction between workers and tasks that have been sub-

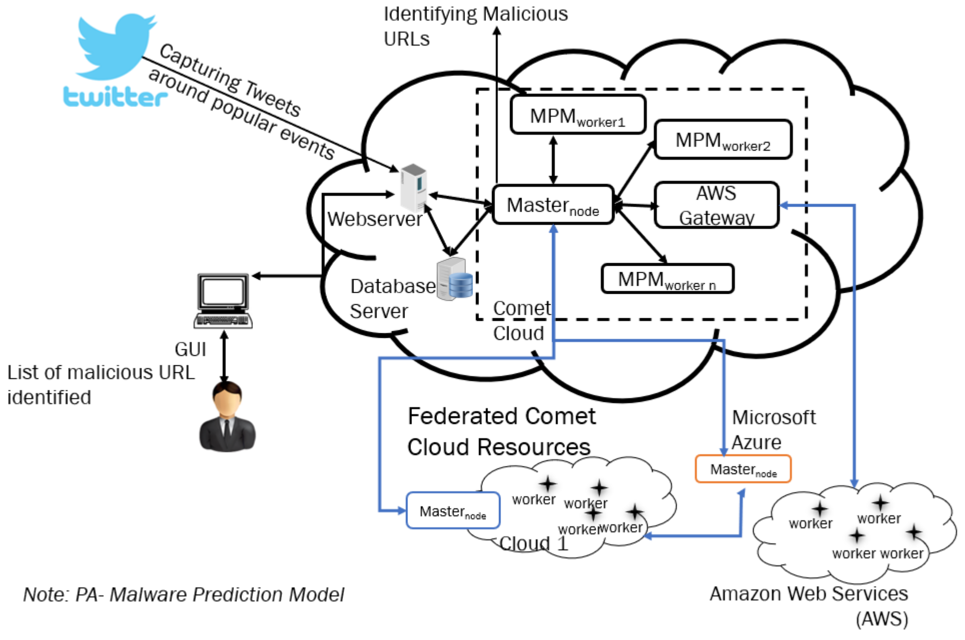


Figure 7.1: Deploying of Malware Prediction Model on Cloud

mitted to the system by one or more users. A worker can directly execute a task, or act as a gateway to a third party system. Execution spaces can be created in the context of a single site for provision, such as local resources or to support a cloudburst (i.e. when additional capacity is needed to respond to a sudden increase in tweet posting rate) to public clouds or external high performance computing systems. Moreover, they can be used to create a private sub-federation across several sites. This case can be useful when several sites have some common interest (e.g. to defend against cyber attacks) and they decide to jointly target certain types of tasks as a specialised community.

Figure 7.1 shows the proposed architecture of deploying the malware prediction model on cloud using the CometCloud framework. The master is responsible for receiving the request containing the details about the tweet captured and then to aggregating the results received from workers. The malware prediction algorithm will be deployed as worker and will be responsible for visiting the web page, creating a log file and classifying the URL into malicious or benign. Once the tweet is captured from the Twitter streaming API, it is forwarded to the master, the master then publishes the task, which in turn can be picked up by any free worker or by another master in a different cloud connected via

CometCloud framework. Once the classification is complete, the result are displayed to the user on a graphical user interface, possibly a dashboard displaying statistics of URLs identified, accounts posting those URL, events used etc., giving the security practitioner a complete picture of the attacks defended against.

7.2.2 Curbing propagation of Malware

Chapter 5 focuses on the development of a predictive model to identify malicious URLs on Twitter and Chapter 6 identifies those factors that aid in propagation of tweets containing malicious URLs. A tweet that is captured and identified as containing malicious link may not be the original tweet, but could be part of a retweet chain. Removing the tweet may not necessarily stop the malware propagation as users that have retweeted the tweet have already exposed it to their followers. However, a strategy to selectively removing those users that have a high degree of centrality from the retweet chain could be employed to curb the malware propagation. Research in identifying these key users or minimum number of users that can stop the propagation until the infection is removed from the network could be another future direction of this work.

7.2.3 Generalising model for other OSN platforms

The research was conducted using Twitter as a case study, where data were collected from seven different diverse popular sporting events. However, the model and techniques can potentially be adapted to other platforms, as the predictive model relied greatly on the machine activities leading to a drive-by download attack. The capturing of machine activities was independent of the OSNs, but the social feature vector may be different from platform to platform. However, these features could easily be added when the social features are merged with the machine activities. Making the algorithm adaptable to different OSNs.

7.3 Conclusion

The contribution of this research has been towards countering drive-by downloads attacks on Twitter by developing a predictive model to identify URLs pointing to malicious servers and to identify factors aiding its propagation. It contributes to existing literature on detecting malware on OSNs, by shifting the focus towards the effects of malware on user machines away from the malware signature and dynamic behaviour, which can be obfuscated. This was achieved by developing a novel model to detect drive-by downloads on Twitter. The model was built by observing changes to the user's system in terms of machine activities observed while visiting a web page. The model was successful in classifying an URL into malicious or benign with an F-measure of 0.81 during training and an F-Measure of 0.71 while testing the model on an unseen dataset. This was then extended to develop a novel prediction model by using machine activity data and tweet's metadata, thus moving beyond post-execution classification of such URLs as malicious, to predicting a URL will be malicious with 0.99 F-measure (using 10-fold cross-validation) and 0.833 (using an unseen test set) at 1 second into the interaction with the URL. This provides a new basis from which to kill the connection to the server before an attack has completed and proactively blocking and preventing an attack, rather than reacting and repairing at a later date.

The extent of damage caused due to a cyber attack on Twitter depends on the number of people that were exposed to it. Earlier studies have established that emotion and sentiments could be transferred to one another through OSNs [89] and they can affect the posting behaviour of users [120]. Thus, in order to investigate the relationship between emotion, sentiment and tweets containing malicious links, our first study that links emotions and sentiment to the propagation of tweets containing malicious URLs on Twitter was conducted. The experimental results showed that there was a statistically significant association between the social and emotional factors derived from a tweet captured during a sporting event. Furthermore, tweets that contain malicious links were associated with negative emotions, particularly with the emotion fear (30% of total tweets contained words associated with fear), for their retweet likelihood (virality) and survival.

The work conducted in the thesis aimed to address the problem of drive-by download on Twitter, that accounted for 48% of web-based attacks, by predicting whether a URL will be pointing to a malicious web server or not within one second of interaction with the URL, and by identifying the association of negative emotion with retweetability of a malicious tweet. The outcome of the thesis provides hitherto unavailable means through which malicious URLs could be proactively identified and factors affecting its propagation could be used to devise a strategy to curtail its propagation.

Bibliography

[1] Accenture. *Cybertech Europe 2017 I Accenture*. [Online; accessed 3. Apr. 2019]. Apr. 2019. URL: `https://www.accenture.com/us-en/event-cybertech-europe-2017?src=SOMS#block-insights-and-innovation`.

[2] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. "Mining association rules between sets of items in large databases". In: *Acm sigmod record*. Vol. 22. 2. ACM. 1993, pp. 207–216.

[3] Faraz Ahmed and Muhammad Abulaish. "A generic statistical approach for spam detection in online social networks". In: *Computer Communications* 36.10-11 (2013), pp. 1120–1129.

[4] Mitsuaki Akiyama. "Study on High Interaction Client Honeypot for Infiltrative Intrusion Detection". In: (2013).

[5] Mitsuaki Akiyama et al. "Design and implementation of high interaction client honeypot for drive-by-download attacks". In: *IEICE transactions on communications* 93.5 (2010), pp. 1131–1139.

[6] Bandar Alghamdi, Yue Xu, and Jason Watson. "Malicious behaviour analysis on twitter through the lens of user interest". In: *Australasian Conference on Data Mining*. Springer. 2017, pp. 233–249.

[7] Per Kragh Andersen and Richard David Gill. “Cox’s regression model for counting processes: a large sample study”. In: *The annals of statistics* (1982), pp. 1100–1120.

[8] David S Anderson et al. “Spamscatter: Characterizing internet scam hosting infrastructure”. PhD thesis. University of California, San Diego, 2007.

[9] Lars Backstrom et al. “Characterizing and curating conversation threads: expansion, focus, volume, re-entry”. In: *Proceedings of the sixth ACM international conference on Web search and data mining*. ACM. 2013, pp. 13–22.

[10] Sebastian Banescu et al. “Code obfuscation against symbolic execution attacks”. In: *Proceedings of the 32nd Annual Conference on Computer Security Applications*. ACM. 2016, pp. 189–200.

[11] Karel Bartos, Michal Sofka, and Vojtech Franc. “Optimized Invariant Representation of Network Traffic for Detecting Unseen Malware Variants.” In: *USENIX Security Symposium*. 2016, pp. 807–822.

[12] Bas van den Beld. *How to recognize Twitterbots: 7 signals to look out for.* 2012. URL: `http://goo.gl/YZbVf`.

[13] Fabrice Bellard. “QEMU, a fast and portable dynamic translator.” In: *USENIX Annual Technical Conference, FREENIX Track*. Vol. 41. 2005, p. 46.

[14] F Benevenuto et al. “Detecting spammers on twitter”. In: *Collaboration, electronic messaging, anti-abuse and spam conference (CEAS)* 6 (2010), p. 12.

[15] Blanchon Benoît. *Winpooch | Open Source Alternative - osalt.com*. [Online; accessed 18. Nov. 2018]. Aug. 2017. URL: `https://www.osalt.com/winpooch`.

[16] Jonah Berger and Katherine L Milkman. “Emotion and virality: what makes online content go viral?” In: *GfK Marketing Intelligence Review* 5.1 (2013), pp. 18–23.

[17] Jonah Berger and Katherine L Milkman. “What makes online content viral?” In: *Journal of marketing research* 49.2 (2012), pp. 192–205.

[18] Ben Berkowitz. *Twitter says aware CFO's account was hacked; working to remove content*. https://www.cnbc.com/2015/02/10/twitter-says-aware-cfos-account-was-hacked-working-to-remove-content.html. 2015.

[19] Sajid Yousuf Bhat and Muhammad Abulaish. "Using communities against deception in online social networks". In: *Computer Fraud & Security* 2014.2 (2014), pp. 8–16.

[20] Kim Boatman. *Are Hackers Using Your Webcam to Watch You?* [Online; accessed 31. May 2019]. May 2019. URL: http://ca.norton.com/yoursecurity%20resource/detail.jsp?aid=webcam_hacking.

[21] Thorsten Brants and Alex Franz. *Web 1t 5-gram version 1,(Linguistic Data Consortium, Philadelphia)*. Tech. rep. Technical report, 2006.

[22] David Brumley et al. "Automatically identifying trigger-based behavior in malware". In: *Botnet Detection*. Springer, 2008, pp. 65–88.

[23] Pete Burnap et al. "Real-time Classification of Malicious URLs on Twitter Using Machine Activity Data". In: *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015*. ASONAM '15. Paris, France: ACM, 2015, pp. 970–977. ISBN: 978-1-4503-3854-7.

[24] Pete Burnap et al. "Real-time classification of malicious URLs on Twitter using machine activity data". In: *Advances in Social Networks Analysis and Mining (ASONAM), 2015 IEEE/ACM International Conference on*. IEEE. 2015, pp. 970–977.

[25] Pete Burnap et al. "Tweeting the terror: modelling the social media reaction to the Woolwich terrorist attack". In: *Social Network Analysis and Mining* 4.1 (2014), pp. 1–14. ISSN: 18695469.

[26] Pete Burnap et al. “Tweeting the terror: modelling the social media reaction to the Woolwich terrorist attack”. In: *Social Network Analysis and Mining* 4.1 (2014), p. 206.

[27] R. Steenson. C. Seifert. *Capture-HPC*. `https://projects.honeynet.org/capture-hpc`. 2017.

[28] Massimo Calabresi. *Inside Russia’s Social Media War on America*. [Online; accessed 18. Nov. 2018]. Nov. 2018. URL: `http://time.com/4783932/inside-russia-social-media-war-america`.

[29] Dell Cameron. “Welp, Vevo Just Got Hacked”. In: *Gizmodo* (Sept. 2017). URL: `https://gizmodo.com/welp-vevo-just-got-hacked-1813390834`.

[30] M. Camisani-Calzolari. *Analysis of Twitter followers of the US Presidential Election candidates: Barack Obama and Mitt Romney*. 2012. URL: `http://digital%20evaluations.com`.

[31] Davide Canali et al. “Prophiler : A Fast Filter for the Large-Scale Detection of Malicious Web Pages Categories and Subject Descriptors”. In: *Proc. of the International World Wide Web Conference (WWW)* (2011), pp. 197–206.

[32] Cheng Cao and James Caverlee. “Detecting Spam URLs in Social Media via Behavioral Analysis”. In: *Advances in Information Retrieval* 9022 (2015), pp. 703–714. ISSN: 16113349.

[33] Jian Cao et al. “Detection of Forwarding-Based Malicious URLs in Online Social Networks”. In: *International Journal of Parallel Programming* 44.1 (2016), pp. 163–180. ISSN: 08857458.

[34] Luca Carettoni, Claudio Merloni, and Stefano Zanero. “Studying bluetooth malware propagation: The bluebag project”. In: *IEEE Security & Privacy* 5.2 (2007), pp. 17–25.

[35] Strapparava Carlo and Valitutti Alessandro. *WN-Affect Taxonomy Specification*. [Online; accessed 28. Jun. 2019]. July 2015. URL: `http://www.gsi.dit.upm.es/ontologies/wnaffect`.

[36] US-CERT. *Ransomware | US-CERT*. [Online; accessed 31. May 2019]. May 2019. URL: https://www.us-cert.gov/Ransomware.

[37] Dave Chaffey. *Global social media research summary 2019 | Smart Insights*. [Online; accessed 2. Apr. 2019]. Feb. 2019. URL: https://www.smartinsights.com/social-media-marketing/social-media-strategy/new-global-social-media-research.

[38] Chao Chen et al. "Spammers Are Becoming" Smarter" on Twitter". In: *IT professional* 18.2 (2016), pp. 66–70.

[39] Chao Chen et al. "Spammers Are Becoming" Smarter" on Twitter". In: *IT professional* 18.2 (2016), pp. 66–70.

[40] Yijin Chen et al. "Malware propagation analysis in message-recallable online social networks". In: *Communication Technology (ICCT), 2017 IEEE 17th International Conference on*. IEEE. 2017, pp. 1366–1371.

[41] Shin-Ming Cheng et al. "On modeling malware propagation in generalized social networks". In: *IEEE Communications Letters* 15.1 (2011), pp. 25–27.

[42] Aditya Chetan et al. "CoReRank: Ranking to Detect Users Involved in Blackmarket-Based Collusive Retweeting Activities". In: *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. ACM. 2019, pp. 330–338.

[43] Zi Chu, Indra Widjaja, and Haining Wang. "Detecting social spam campaigns on twitter". In: *International Conference on Applied Cryptography and Network Security*. Springer. 2012, pp. 455–472.

[44] Federico Concone et al. "Twitter analysis for real-time malware discovery". In: *2017 AEIT International Annual Conference*. IEEE. 2017, pp. 1–6.

[45] Kate Conger. "Tracking Instagram's money-flipping scammers". In: *TechCrunch* (Aug. 2016). URL: https://techcrunch.com/2016/08/26/tracking-instagrams-money-flipping-scammers.

[46] Symantec Coorp. “Symantec Internet Security Threat Report Volume XXIII”. In: *Whitepaper, Apr* 23 (2008).

[47] Symantec Corporation. *10 cyber security facts and statistics for 2018*. [Online; accessed 17. May 2019]. May 2019. URL: `https://us.norton.com/internetsecurity-emerging-threats-10-facts-about-todays-cybersecurity-landscape-that-you-should-know.html`.

[48] Counter Threat Unit Research Team. *The Curious Case of Mia Ash - COBALT GYPSY Threat Intelligence*. [Online; accessed 18. Nov. 2018]. Nov. 2018. URL: `https://www.secureworks.com/research/the-curious-case-of-mia-ash`.

[49] Marco Cova, Christopher Kruegel, and Giovanni Vigna. “Detection and analysis of drive-by-download attacks and malicious JavaScript code”. In: *Proceedings of the 19th international conference on World wide web - WWW ’10* (2010), p. 281.

[50] Stefano Cresci et al. “Fame for sale: efficient detection of fake Twitter followers”. In: *Decision Support Systems* 80 (2015), pp. 56–71.

[51] George E. Dahl et al. *Large-scale malware classification using random projections and neural networks*. IEEE, May 2013. DOI: `10.1109/ICASSP.2013.6638293`.

[52] Mohsen Damshenas, Ali Dehghantanha, and Ramlan Mahmoud. “A survey on malware propagation, analysis, and detection”. In: *International Journal of Cyber-Security and Digital Forensics* 2.4 (2013), pp. 10–30.

[53] Cedric Pernet Daniel Lunghi Jaromir Horejsi. *Untangling the Patchwork Cyberespionage Group - TrendLabs Security Intelligence Blog*. [Online; accessed 3. Apr. 2019]. Dec. 2017. URL: `http://tinyurl.com/y58z53oq`.

[54] C. Darwin, P. Ekman, and P. Prodger. *The Expression of the Emotions in Man and Animals*. Oxford University Press, 1998. ISBN: 978-019515806-9.

[55] Farid Daryabar, Ali Dehghantanha, and Nur Izura Udzir. "Investigation of bypassing malware defences and malware detections". In: *2011 7th International Conference on Information Assurance and Security (IAS)*. IEEE. 2011, pp. 173–178.

[56] Sanjeev Das et al. "Semantics-based online malware detection: Towards efficient real-time protection against malware". In: *IEEE transactions on information forensics and security* 11.2 (2015), pp. 289–302.

[57] Arthur Delbridge et al. *The macquarie dictionary*. Vol. 1. Macquarie Library North Ryde, NSW,, Australia, 2001.

[58] Javier Diaz-Montes et al. "Cometcloud: Enabling software-defined federations for end-to-end application workflows". In: *IEEE Internet Computing* 19.1 (2015), pp. 69–73.

[59] Stuart Dredge. "World Cup was biggest event yet for Twitter with 672m tweets". In: *the Guardian* (July 2014). URL: `https://www.theguardian.com/technology/2014/jul/15/twitter-world-cup-tweets-germany-brazil`.

[60] Valentina D'Urso and Rosanna Trentin. *Introduzione alla psicologia delle emozioni*. Laterza, 2007.

[61] Hridoy Sankar Dutta et al. "Retweet us, we will retweet you: Spotting collusive retweeters involved in blackmarket services". In: *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. IEEE. 2018, pp. 242–249.

[62] Edgefluence. *Like4Like - Get FREE real Instagram likes!* [Online; accessed 25. Mar. 2019]. Mar. 2019. URL: `https://like4like.com`.

[63] Paul Ekman. "An argument for basic emotions". In: *Cognition & emotion* 6.3-4 (1992), pp. 169–200.

[64] P. Ekman et al. *Emotion in the Human Face*. Elsevier Science, 2013. ISBN: 978-148314763-5.

[65] Birhanu Eshete and VN Venkatakrishnan. “Webwinnow: Leveraging exploit kit workflows to detect malicious urls”. In: *Proceedings of the 4th ACM conference on Data and application security and privacy*. ACM. 2014, pp. 305–312.

[66] Facebook. *Facebook – log in or sign up*. [Online; accessed 2. Apr. 2019]. Apr. 2019. URL: https://www.facebook.com.

[67] W Fan and KH Yeung. “Online social networks—Paradise of computer viruses”. In: *Physica A: Statistical Mechanics and its Applications* 390.2 (2011), pp. 189–197.

[68] Ivan Firdausi, Alva Erwin, Anto Satriyo Nugroho, et al. “Analysis of machine learning techniques used in behavior-based malware detection”. In: *2010 second international conference on advances in computing, control, and telecommunication technologies*. IEEE. 2010, pp. 201–203.

[69] Max Fisher. “Syrian hackers claim AP hack that tipped stock market by 136 billion. Is it terrorism?” In: *Washington Post* (Apr. 2013). ISSN: 0190-8286. URL: https://www.washingtonpost.com/news/worldviews/wp/2013/04/23/syrian-hackers-claim-ap-hack-that-tipped-stock-market-by-136-billion-is-it-terrorism.

[70] Chris Fleizach et al. “Can you infect me now?: malware propagation in mobile phone networks”. In: *Proceedings of the 2007 ACM workshop on Recurring malcode*. ACM. 2007, pp. 61–68.

[71] James H Fowler and Nicholas A Christakis. “Dynamic spread of happiness in a large social network: longitudinal analysis over 20 years in the Framingham Heart Study”. In: *Bmj* 337 (2008), a2338.

[72] Lorenzo Franceschi-Bicchierai. “Another Day, Another Hack: 117 Million LinkedIn Emails And Passwords - Motherboard”. In: *Motherboard* (May 2016). URL: https://motherboard.vice.com/en_us/article/78kk4z/another-day-another-hack-117-million-linkedin-emails-and-password.

[73] Sheera Frenkel. "Hackers Hide Cyber attacks in Social Media Posts". In: *N. Y. Times* (May 2017). ISSN: 0362-4331. URL: https://www.nytimes.com/2017/05/28/technology/hackers-hide-cyberattacks-in-social-media-posts.html.

[74] Vladimir Kropotov Fyodor Yarochkin. *Lurk: Retracing the Group's Five-Year Campaign - TrendLabs Security Intelligence Blog*. [Online; accessed 3. Apr. 2019]. Feb. 2017. URL: https://blog.trendmicro.com/trendlabs-security-intelligence/lurk-retracing-five-year-campaign/.

[75] Antonio Galante, Ary Kokos, and Stefano Zanero. "Bluebat: Towards practical bluetooth honeypots". In: *2009 IEEE International Conference on Communications*. IEEE. 2009, pp. 1–6.

[76] Ayalvadi Ganesh, Laurent Massoulié, and Don Towsley. "The effect of network topology on the spread of epidemics". In: *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*. Vol. 2. IEEE. 2005, pp. 1455–1466.

[77] Shantanu Ghosh. *Top seven social media threats*. [Online; accessed 23. May 2019]. May 2019. URL: https://www.computerweekly.com/tip/Top-seven-social-media-threats.

[78] Alexandra Gibbs. "Super Bowl XLIX smashes Twitter records". In: *CNBC* (Feb. 2015). URL: https://www.cnbc.com/2015/02/02/super-bowl-xlix-and-social-media-most-tweeted-nfl-game-ever.html.

[79] Dan Goodin. *Powerful worm on Twitter unleashes torrent of out-of-control tweets*. [Online; accessed 11. Jun. 2019]. Nov. 2014. URL: https://arstechnica.com/information-technology/2014/06/powerful-worm-on-twitter-unleashes-torrent-of-out-of-control-tweets.

[80] Google. *Google Safe Browsing | Google Developers*. [Online; accessed 28. Feb. 2019]. Jan. 2019. URL: https://developers.google.com/safe-browsing.

[81] Google. *YouTube*. [Online; accessed 2. Apr. 2019]. Apr. 2019. URL: https://www.youtube.com.

[82] Nelson Granados. "Super Bowl Underperforms In TV Audience And Social Media Chatter". In: *Forbes* (Feb. 2016). URL: https://www.forbes.com/sites/nelsongranados/2016/02/09/super-bowl-underperforms-in-tv-audience-and-social-media-chatter/#2a7611a02be3.

[83] Chris Grier et al. "@ spam : The Underground on 140 Characters or Less , Categories and Subject Descriptors". In: *Proceedings of the 17th ACM conference on Computer and communications security* (2010), pp. 27–37. ISSN: 15437221.

[84] NCC Group. *Introduction to anti-fuzzing: A defence in depth aid.* 2015. URL: https://www.nccgroup.trust/uk/about-us/newsroom-and-events/blogs/2014/january/introduction-to-anti-fuzzing-a-defence-in-depth-aid.

[85] Aditi Gupta, Hemank Lamba, and Ponnurangam Kumaraguru. "$1.00 per rt# bostonmarathon# prayforboston: Analyzing fake content on twitter". In: *2013 APWG eCrime researchers summit*. IEEE. 2013, pp. 1–12.

[86] Shashank Gupta and Brij Bhooshan Gupta. "Cross-Site Scripting (XSS) attacks and defense mechanisms: classification and state-of-the-art". In: *International Journal of System Assurance Engineering and Management* 8.1 (2017), pp. 512–530.

[87] Shiferaw Gurmu. "Tests for detecting overdispersion in the positive Poisson regression model". In: *Journal of Business & Economic Statistics* 9.2 (1991), pp. 215–222.

[88] Alex Guy. *Infection rates and end of support for Windows XP*. [Online; accessed 19. May 2019]. May 2019. URL: https://blogs.technet.microsoft.com/uktechnet/2013/10/29/infection-rates-and-end-of-support-for-windows-xp.

[89] Elaine Hatfield, John T Cacioppo, and Richard L Rapson. “Emotional contagion”. In: *Current directions in psychological science* 2.3 (1993), pp. 96–100.

[90] Chip Heath, Chris Bell, and Emily Sternberg. “Emotional selection in memes: the case of urban legends.” In: *Journal of personality and social psychology* 81.6 (2001), p. 1028.

[91] Radek Hes et al. *The Capture-HPC client architecture*. Tech. rep. Citeseer, 2009.

[92] Joseph M Hilbe. *Negative binomial regression*. Cambridge University Press, 2011.

[93] Chris Hoffman. *How to keep your PC secure when Microsoft ends Windows XP support*. [Online; accessed 19. May 2019]. Feb. 2014. URL: https://www.pcworld.com/article/2102606/how-to-keep-your-pc-secure-when-microsoft-ends-windows-xp-support.html.

[94] John Hopcroft, Tiancheng Lou, and Jie Tang. “Who will follow you back?: reciprocal relationship prediction”. In: *Proceedings of the 20th ACM international conference on Information and knowledge management*. ACM. 2011, pp. 1137–1146.

[95] Xia Hu et al. “Social spammer detection with sentiment information”. In: *Data Mining (ICDM), 2014 IEEE International Conference on*. IEEE. 2014, pp. 180–189.

[96] Nwokedi Idika and Aditya P Mathur. “A survey of malware detection techniques”. In: *Purdue University* 48 (2007), pp. 2007–2.

[97] WhatsApp Inc. *WhatsApp*. [Online; accessed 2. Apr. 2019]. Apr. 2019. URL: https://www.whatsapp.com.

[98] Instagram. *Instagram*. [Online; accessed 2. Apr. 2019]. Apr. 2019. URL: https://www.instagram.com/?hl=en.

[99] Fire Eye Threat Intelligence. “HAMMERTOSS: stealthy tactics define a Russian cyber threat group”. In: *Milpitas, CA: FireEye, Inc* (2015).

[100] Luca Invernizzi et al. “Evilseed: A guided approach to finding malicious web pages”. In: *2012 IEEE symposium on Security and Privacy*. IEEE. 2012, pp. 428–442.

[101] Laurent Itti and Pierre Baldi. “Bayesian surprise attracts human attention”. In: *Vision research* 49.10 (2009), pp. 1295–1306.

[102] Amir Javed, Pete Burnap, and Omer Rana. “Prediction of drive-by download attacks on Twitter”. In: *Information Processing & Management* (2018).

[103] Gaya K Jayasinghe, J Shane Culpepper, and Peter Bertok. “Efficient and effective realtime prediction of drive-by download attacks”. In: *Journal of Network and Computer Applications* 38 (2014), pp. 135–149.

[104] Business Insider Jeff Dunn. “A huge number of PCs still use ancient Windows software that puts them at risk”. In: *Business Insider* (May 2017). URL: `http://uk.businessinsider.com/how-many-people-use-windows-xp-chart-2017-5`.

[105] Xuxian Jiang and Yajin Zhou. “Dissecting android malware: Characterization and evolution”. In: *2012 IEEE symposium on security and privacy*. IEEE. 2012, pp. 95–109.

[106] Juniper Research Lab. *Juniper Research: Cybersecurity Breaches to Result in Over 146 Billion Records Being Stolen by 2023*. [Online; accessed 17. May 2019]. May 2019. URL: `https://www.businesswire.com/news/home/20180808005033/en/Juniper-Research-Cybersecurity-Brea%20ches-Result-146-Billion`.

[107] S Jyothi, Chandra Sekhar Vorugunti, et al. “Epidemic model based evaluation of malware propagation in Twitter”. In: *2017 9th International Conference on Communication Systems and Networks (COMSNETS)*. IEEE. 2017, pp. 407–408.

[108] Min Gyung Kang et al. “Emulating emulation-resistant malware”. In: *Proceedings of the 1st ACM workshop on Virtual machine security*. ACM. 2009, pp. 11–22.

[109] Alexandros Kapravelos et al. “Revolver: An Automated Approach to the Detection of Evasive Web-based Malware”. In: *Usenix security* (2013).

[110] Naoto Kawaguchi and Kazumasa Omote. *Malware Function Classification Using APIs in Initial Behavior*. IEEE, May 2015. ISBN: 978-1-4799-1989-5. DOI: `10.1109/AsiaJCIS.2015.15`.

[111] Kris Kendall and Chad McMillan. “Practical malware analysis”. In: *Black Hat Conference, USA*. 2007, p. 10.

[112] Swati Khandelwal. *Twitter Discloses Suspected State-Sponsored Attack After Minor Data Breach*. [Online; accessed 14. Mar. 2019]. Mar. 2019. URL: `https://thehackernews.com/2018/12/twitter-data-breach.html`.

[113] Apalak Khatua and Aparup Khatua. “Cricket World Cup 2015: Predicting User’s Orientation through Mix Tweets on Twitter Platform”. In: *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*. ACM. 2017, pp. 948–951.

[114] Kyungtae Kim et al. “J-Force: Forced Execution on JavaScript”. In: *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee. 2017, pp. 897–906.

[115] Matthew Kirk. *Thoughtful machine learning: A test-driven approach*. ” O’Reilly Media, Inc.”, 2014.

[116] Clemens Kolbitsch et al. “Rozzle: De-cloaking internet malware”. In: *2012 IEEE Symposium on Security and Privacy*. IEEE. 2012, pp. 443–457.

[117] J Zico Kolter and Marcus A Maloof. “Learning to detect and classify malicious executables in the wild”. In: *Journal of Machine Learning Research* 7.Dec (2006), pp. 2721–2744.

[118] Ivana Kottasova. *Twitter reveals the top tweeted events of 2016 - Dec. 6, 2016*. (Accessed on 11/29/2017). 2016.

[119] Eduard Kovacs. *Bank of Melbourne Twitter Account Hacked*. [Online; accessed 18. Nov. 2018]. Nov. 2018. URL: `https://news.softpedia.com/news/Bank-of-Melbourne-Twitter-Account-Hacked-222511.shtml`.

[120] Adam DI Kramer, Jamie E Guillory, and Jeffrey T Hancock. "Experimental evidence of massive-scale emotional contagion through social networks". In: *Proceedings of the National Academy of Sciences* (2014), p. 201320040.

[121] Brian Krebs. "Ddos on dyn impacts twitter, spotify, reddit". In: *Krebs on Security.(October 2016). Retrieved June* 1 (2016), p. 2017.

[122] Mohit Kumar. *How A Drive-by Download Attack Locked Down Entire City for 4 Days*. `https://thehackernews.com/2017/10/drive-by-download-ransomware.html`. 2017.

[123] Lastline Labs. *The threat of evasive malware. white paper*. [Online; accessed 28. Feb. 2019]. Feb. 2013. URL: `https://www.infosecurityeurope.com/__novadocuments/357218?v=636295321401130000`.

[124] Sam Laird. *The top 15 sporting events that blew up Twitter in 2015*. 2015. URL: `%5Curl%7Bhttp://mashable.com/2015/12/07/2015-top-sports-events-twitter/#7TVsYNhLQSqN%7D`.

[125] Pavel Laskov and Nedim Šrndić. "Static detection of malicious JavaScript-bearing PDF documents". In: *Proceedings of the 27th annual computer security applications conference*. ACM. 2011, pp. 373–382.

[126] Kyumin Lee, James Caverlee, and Steve Webb. "Uncovering social spammers: social honeypots+ machine learning". In: *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*. ACM. 2010, pp. 435–442.

[127] Sangho Lee and Jong Kim. "Warningbird: A near real-time detection system for suspicious urls in twitter stream". In: *IEEE transactions on dependable and secure computing* 10.3 (2013), pp. 183–195.

[128] Wenke Lee and Jack W Stokes. "ARROW : Generating Signatures to Detect Drive-By Downloads". In: *Www 2011* (2011), pp. 187–196.

[129] Kristina Lerman and Rumi Ghosh. "Information Contagion: an Empirical Study of the Spread of News on Digg and Twitter Social Networks". In: *Association for the Advancement of Artificial Intelligenc*. 2010, pp. 90–97.

[130] Johnny Lieu. "The most retweeted tweet ever is a billionaire's giveaway of free money". In: *Mashable* (Jan. 2019). URL: `https://mashable.com/video/cold-weather-autonomous-bus-robot`.

[131] Min-Sheng Lin et al. "Malicious URL filtering—A big data application". In: *2013 IEEE international conference on big data*. IEEE. 2013, pp. 589–596.

[132] Bo Liu et al. "Malware propagations in wireless ad hoc networks". In: *IEEE Transactions on Dependable and Secure Computing* 1 (2016), pp. 1–1.

[133] J Scott Long. "Regression models for categorical and limited dependent variables (Vol. 7)". In: *Advanced quantitative techniques in the social sciences* (1997).

[134] *Low, Medium and High Interaction Honeypot Security | GuardiCore*. [Online; accessed 6. Jun. 2019]. Jan. 2019. URL: `https://www.guardicore.com/2019/1/high-interaction-honeypot-versus-low-interaction-honeypot`.

[135] Justin Ma et al. "Beyond Blacklists : Learning to Detect Malicious Web Sites from Suspicious URLs". In: *World Wide Web Internet And Web Information Systems* (2009), pp. 1245–1253.

[136] MarkMonitor. *Protecting brands in the digital world*. [Online; accessed 4. Mar. 2019]. Mar. 2019. URL: `https://www.markmonitor.com`.

[137] D.K. McGrath and Minaxi Gupta. "Behind phishing: an examination of phisher modi operandi". In: *Usenix Workshop on Large-Scale Exploits and Emergent Threats (LEET)* (2008), p. 4. ISSN: 15692558.

[138] Robert McMillan. *High Profile Twitter Hack Spreads Porn Trojan — PCWorld*. https://www.pcworld.com/article/167253/article.html. 2009.

[139] Microsoft. *Microsoft Security Essentials*. [Online; accessed 19. May 2019]. May 2019. URL: https://www.microsoft.com/en-gb/download/details.aspx?id=5201.

[140] Microsoft. *Microsoft Security Intelligence Report*. Tech. rep. Dec. 2013.

[141] Microsoft. *Resources and Tools for IT Professionals | TechNet*. [Online; accessed 9. Jun. 2019]. June 2019. URL: https://technet.microsoft.com/en-us/default.aspx.

[142] Microsoft. *Run and RunOnce Registry Keys*. [Online; accessed 20. Nov. 2018]. Nov. 2018. URL: https://docs.microsoft.com/en-us/windows/desktop/setupapi/run-and-runonce-registry-keys.

[143] Microsoft. *Set Application-specific Access Permissions*. [Online; accessed 19. May 2019]. May 2019. URL: https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2008-R2-and-2008/cc731858(v=ws.11).

[144] Zachary Miller et al. "Twitter spammer detection using data stream clustering". In: *Information Sciences* 260 (2014), pp. 64–73.

[145] G. Mohamed Sikandar. "100 Social Media Statistics You Must Know [2018]". In: *Statusbrew Blog* (Dec. 2018). URL: https://blog.statusbrew.com/social-media-statistics-2018-for-business.

[146] Hossein Saidi Mohammad Reza Faghani. "Malware Propagation in Online Social Networks". In: *4th International Conference on Malicious and Unwanted Software (MALWARE)*. 2009, pp. 8–14.

[147] Saif M. Mohammad and Svetlana Kiritchenko. "Using Hashtags to Capture Fine Emotion Categories from Tweets". In: *Computational Intelligence* 31.2 (2015),

pp. 301–326. ISSN: 1467-8640. DOI: 10.1111/coin.12024. URL: http://dx.doi.org/10.1111/coin.12024.

[148] Saif M Mohammad and Peter D Turney. "Crowdsourcing a word–emotion association lexicon". In: *Computational Intelligence* 29.3 (2013), pp. 436–465.

[149] Andreas Moser, Christopher Kruegel, and Engin Kirda. "Exploring multiple execution paths for malware analysis". In: *2007 IEEE Symposium on Security and Privacy (SP'07)*. IEEE. 2007, pp. 231–245.

[150] Andreas Moser, Christopher Kruegel, and Engin Kirda. "Limits of static analysis for malware detection". In: *Twenty-Third Annual Computer Security Applications Conference (ACSAC 2007)*. IEEE. 2007, pp. 421–430.

[151] Smita Naval et al. "Employing program semantics for malware detection". In: *IEEE Transactions on Information Forensics and Security* 10.12 (2015), pp. 2591–2604.

[152] Hamed Haddad Pajouh et al. "A two-layer dimension reduction and two-tier classification model for anomaly-based intrusion detection in IoT backbone networks". In: *IEEE Transactions on Emerging Topics in Computing* (2016).

[153] Danny Palmer. *Is your Android phone being controlled by a rogue Twitter account? Botnet is first to receive commands via tweets — ZDNet*. https://www.zdnet.com/article/is-your-android-phone-being-controlled-by-a-rogue-twitter-account-botnet-is-first-to-receive/. (Accessed on 09/18/2018). 2016.

[154] Romualdo Pastor-Satorras and Alessandro Vespignani. "Epidemic spreading in scale-free networks". In: *Physical review letters* 86.14 (2001), p. 3200.

[155] PhishTank. *PhishTank | Join the fight against phishing*. [Online; accessed 4. Mar. 2019]. Mar. 2019. URL: https://www.phishtank.com.

[156] Pinterest. *Pinterest (United Kingdom)*. [Online; accessed 2. Apr. 2019]. Apr. 2019. URL: https://www.pinterest.co.uk.

[157] Robert Plutchik. *Emotions and life: Perspectives from psychology, biology, and evolution.* American Psychological Association, 2003.

[158] Ponemon Institute. *Cost of a Data Breach Study*. [Online; accessed 17. May 2019]. Apr. 2019. URL: https://www.ibm.com/security/data-breach.

[159] PorcelainSky LLC. *Get Twitter Followers, YouTube Views, Subscribers - YouLike-Hits*. [Online; accessed 25. Mar. 2019]. Mar. 2019. URL: https://www.youlikehits.com.

[160] Mohammad Puttaroo, Peter Komisarczuk, and Renato Cordeiro de Amorim. *Challenges in developing Capture-HPC exclusion lists*. ACM, 2014.

[161] Mohammad Puttaroo, Peter Komisarczuk, and Renato Cordeiro de Amorim. “On drive-by-download attacks and malware classification”. In: *Fifth International Conference on Internet Technologies & Applications (ITA), Wrexham, Wales*. Vol. 10. 2013.

[162] Praveen Rao et al. “Methods to detect cyberthreats on twitter”. In: *Surveillance in Action*. Springer, 2018, pp. 333–350.

[163] Clement René. *Instagram Engagement Report 2019: The more hashtags, the less engagement*. [Online; accessed 25. Mar. 2019]. Mar. 2019. URL: https://mention.com/blog/hashtags-engagement-instagram.

[164] MG Roberts and JAP Heesterbeek. *Mathematical models in epidemiology*. EOLSS, 2003.

[165] Wills Robinson. *Russia hacked Pentagon’s Joint Chiefs of Staff and shut down its email system — Daily Mail Online*. http://www.dailymail.co.uk/news/article-3187344/Russia-hacked-Joint-Chiefs-Staff-shut-email-4-000-defence-department-employees-ELEVEN-DAYS.html. 2015.

[166] Joshua Roesslein. *Tweepy*. 2009. URL: http://www.tweepy.org/.

[167] Charlotte Rogers. *Euro 2016 most tweeted TV of the year*. https://www.marketingweek.com/2016/12/14/euros-tweeted-tv-2016/. 2016.

[168] Charlotte Rogers. *Euro 2016 most tweeted TV of the year*. [Online; accessed 10. Dec. 2018]. Dec. 2016. URL: https://www.marketingweek.com/2016/12/14/euros-tweeted-tv-2016.

[169] Seth Rosenblatt. "Malwarebytes: With Anti-Exploit, we'll stop the worst attacks on PCs". In: *CNET* (June 2014). URL: https://www.cnet.com/news/malwarebytes-finally-unveils-freeware-exploit-killer.

[170] Neil J. Rubenking. *The Best Antivirus Protection for 2019*. Ed. by PCMag. [Online; accessed 19. May 2019]. May 2019. URL: https://uk.pcmag.com/antivirus/8141/the-best-antivirus-protection.

[171] Paul Rubens. "10 Ways to Keep Windows XP Machines Secure". In: *CIO* (May 2014). URL: https://www.cio.com/article/2376575/10-ways-to-keep-windows-xp-machines-secure.html.

[172] Jon Russell. "Hackers nab 500,000 as Enigma is compromised weeks before its ICO". In: *TechCrunch* (Aug. 2017). URL: https://techcrunch.com/2017/08/21/hack-enigma-500000-ico.

[173] Jon Russell. "Prominent Twitter accounts compromised after third-party app Twitter Counter hacked". In: *TechCrunch* (Mar. 2017). URL: https://techcrunch.com/2017/03/15/twitter-counter-hacked.

[174] Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo. "Earthquake shakes Twitter users: real-time event detection by social sensors". In: *Proceedings of the 19th international conference on World wide web*. ACM. 2010, pp. 851–860.

[175] SANS Institue. *2017 Threat Landscape Survey: Users on the Front Line*. https://www.sans.org/reading-room/whitepapers/threats/2017-threat-landscape-survey-users-front-line-37910. 2017.

[176] Igor Santos et al. "OPEM: A Static-Dynamic Approach for Machine-Learning-Based Malware Detection". In: *SpringerLink* (2013), pp. 271–280. DOI: `10.1007/978-3-642-33018-6_28`.

[177] Ameya Sanzgiri, Jacob Joyce, and Shambhu Upadhyaya. "The early (tweet-ing) bird spreads the worm: An assessment of twitter for malware propagation". In: *Procedia Computer Science* 10 (2012), pp. 705–712.

[178] Joshua Saxe and Konstantin Berlin. "Deep neural network based malware detection using two dimensional binary program features". In: *2015 10th International Conference on Malicious and Unwanted Software (MALWARE)* (Oct. 2015), pp. 11–20. DOI: `10.1109/MALWARE.2015.7413680`.

[179] Matthew G Schultz et al. "Data mining methods for detection of new malicious executables". In: *Proceedings 2001 IEEE Symposium on Security and Privacy. S&P 2001*. IEEE. 2001, pp. 38–49.

[180] *Secure your Network by setting up a Honeypot - Loginsoft - Cybersecurity, Software Development, Offshore Services*. [Online; accessed 3. Jun. 2019]. Nov. 2018. URL: `https://www.loginsoft.com/blog/2018/11/16/secure-your-network-by-setting-up-a-honeypot`.

[181] Christian Seifert et al. "Capture–A behavioral analysis tool for applications and documents". In: *digital investigation* 4 (2007), pp. 23–30.

[182] Dave Shackleford. "Who's using Cyberthreat Intelligence and how?" In: *SANS Institute. Retrieved January* 24 (2015), p. 2018.

[183] Hua Shen et al. "Discovering social spammers from multiple views". In: *Neurocomputing* 225 (2017), pp. 49–57.

[184] Seungwon Shin et al. "A large-scale empirical study of conficker". In: *IEEE Transactions on Information Forensics and Security* 7.2 (2012), pp. 676–690.

[185] smfrogers. *Insights into the #WorldCup conversation on Twitter*. [Online; accessed 14. May 2019]. May 2019. URL: `https://blog.twitter.com/`

en_us/a/2014/insights-into-the-worldcup-conversation-on-twitter.html.

[186] Snapchat. *Snapchat – the fastest way to share a moment!* [Online; accessed 2. Apr. 2019]. Apr. 2019. URL: https://www.snapchat.com/l/en-gb.

[187] Socialbakers. *AI-Powered Social Media & Digital Marketing Solution*. [Online; accessed 4. Mar. 2019]. Mar. 2019. URL: https://www.socialbakers.com.

[188] Spotcal. *Healthy TV audiences for final as 2015 Rugby World Cup hailed as 'biggest and best' yet | Featured News| News | Sportcal*. [Online; accessed 14. May 2019]. May 2019. URL: https://www.sportcal.com/News/FeaturedNews/39963.

[189] CricketCountry Staff. "ICC Cricket World Cup 2015: India-Pakistan a Twitter hit, 1.7 million tweets". In: *Cricket Country* (Feb. 2015). URL: https://www.cricketcountry.com/criclife/icc-cricket-world-cup-2015-india-pakistan-a-twitter-hit-1-7-million-tweets-500296.

[190] James V Stone. *Bayes' rule: A tutorial introduction to Bayesian analysis*. Sebtel Press, 2013.

[191] Carlo Strapparava, Alessandro Valitutti, et al. "Wordnet affect: an affective extension of wordnet." In: *Lrec*. Vol. 4. Citeseer. 2004, pp. 1083–1086.

[192] Gianluca Stringhini, Christopher Kruegel, and Giovanni Vigna. "Detecting spammers on social networks". In: *Proceedings of the 26th annual computer security applications conference*. ACM. 2010, pp. 1–9.

[193] Jing Su et al. "A preliminary investigation of worm infections in a bluetooth environment". In: *Proceedings of the 4th ACM workshop on Recurring malcode*. ACM. 2006, pp. 9–16.

[194] Xin Sun et al. “Mathematical model for spreading dynamics of social network worms”. In: *Journal of Statistical Mechanics: Theory and Experiment* 2012.04 (2012), P04009.

[195] Symantec. *Internet Security Threat Report*. Tech. rep. 2018.

[196] Symantec. *Internet Security Threat Report*. Tech. rep. 2018.

[197] The CERT Polska. *Honeyspider Network 2.0 - CERT Polska*. [Online; accessed 19. Nov. 2018]. Jan. 2013. URL: `https://www.cert.pl/en/news/single/honeyspider-network-2-0`.

[198] *The World’s Top 20 Languages—And The Words English Has Borrowed From Them*. [Online; accessed 31. Mar. 2020]. Aug. 2015. URL: `https://www.mentalfloss.com/article/67766/worlds-top-20-languages-and-words-english-has-borrowed-them`.

[199] Kurt Thomas et al. “Suspended accounts in retrospect: an analysis of twitter spam”. In: *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*. ACM. 2011, pp. 243–258.

[200] ThreatTrack. *Malware Analysis Tool, Dynamic Malware Sandbox - ThreatAnalyzer - ThreatTrack*. [Online; accessed 23. Oct. 2018]. Oct. 2018. URL: `https://www.threattrack.com/malware-analysis.aspx`.

[201] Virus Total. *VirusTotal - Free Online Virus, Malware and URL Scanner*. `https://www.virustotal.com/en/`. (Accessed on 07/11/2017). 2018.

[202] Andranik Tumasjan et al. “Predicting elections with Twitter: What 140 characters reveal about political sentiment”. In: *Proceedings of the Fourth International AAAI Conference on Weblogs and Social Media* (2010), pp. 178–185. ISSN: 00219258.

[203] Tumblr. *Sign up | Tumblr*. [Online; accessed 2. Apr. 2019]. Apr. 2019. URL: `https://www.tumblr.com`.

[204] Twitter. *How to use TweetDeck*. [Online; accessed 11. Jun. 2019]. June 2019. URL: https://help.twitter.com/en/using-twitter/how-to-use-tweetdeck.

[205] Twitter. *New Tweets per second record, and how!* [Online; accessed 29. Apr. 2019]. Apr. 2019. URL: https://blog.twitter.com/engineering/en_us/a/2013/new-tweets-per-second-record-and-how.html.

[206] Twitter. *The Twitter Rules*. [Online; accessed 27. Feb. 2019]. Feb. 2019. URL: https://help.twitter.com/en/rules-and-policies/twitter-rules.

[207] Twitter. *Twitter*. [Online; accessed 2. Apr. 2019]. Apr. 2019. URL: https://twitter.com.

[208] Dolly Uppal et al. "Malware detection and classification based on extraction of API sequences". In: *2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. IEEE. 2014, pp. 2337–2342.

[209] URIBL. *URIBL.COM - Realtime URI Blacklist*. [Online; accessed 28. Feb. 2019]. Feb. 2019. URL: http://uribl.com.

[210] R Vinayakumar, KP Soman, and Prabaharan Poornachandran. "Evaluating deep learning approaches to characterize and classify malicious URL's". In: *Journal of Intelligent & Fuzzy Systems* 34.3 (2018), pp. 1333–1343.

[211] Soroush Vosoughi, Deb Roy, and Sinan Aral. "The spread of true and false news online". In: *Science* 359.6380 (2018), pp. 1146–1151.

[212] Bo Wang et al. "Making the most of tweet-inherent features for social spam detection on Twitter". In: *arXiv preprint arXiv:1503.07405* (2015).

[213] Chenxi Wang, John C Knight, and Matthew C Elder. "On computer viral infection and the effect of immunization". In: *Proceedings 16th Annual Computer Security Applications Conference (ACSAC'00)*. IEEE. 2000, pp. 246–256.

[214] Yi-Min Wang et al. “Automated web patrol with strider honeymonkeys: Finding web sites that exploit browser vulnerabilities”. In: *IN NDSS*. Citeseer. 2006.

[215] Tianbo Wang et al. “The Spatial–Temporal Perspective: The Study of the Propagation of Modern Social Worms”. In: *IEEE Transactions on Information Forensics and Security* 12.11 (2017), pp. 2558–2573.

[216] Xu Wang et al. “Virus propagation modeling and convergence analysis in large-scale networksyan2011malware”. In: *IEEE Transactions on Information Forensics and Security* 11.10 (2016), pp. 2241–2254.

[217] Edda Weigand. *Contrastive lexical semantics*. Vol. 171. John Benjamins Publishing, 1998.

[218] Joe Wein. *joewein.de LLC - fighting spam and scams on the Internet.* [Online; accessed 28. Feb. 2019]. Dec. 2017. URL: `https://joewein.net`.

[219] Ian Welch, Xiaoying Gao, Peter Komisarczuk, et al. “Two-stage classification model to detect malicious web pages”. In: *Advanced Information Networking and Applications (AINA), 2011 IEEE International Conference on*. IEEE. 2011, pp. 113–120.

[220] Sheng Wen et al. “Modeling and analysis on the propagation dynamics of modern email malware”. In: *IEEE transactions on dependable and secure computing* 11.4 (2014), pp. 361–374.

[221] May Wilkerson. “10 times celebrities got hacked and sh$*$t got weird. | Someecards Celebrities”. In: *Someecards* (June 2019). URL: `https://www.someecards.com/entertainment/celebrities/celebrities-social-media-twitter-hack`.

[222] Christian Wressnegger et al. “Comprehensive analysis and detection of flash-based malware”. In: *Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 2016, pp. 101–121.

[223] Tobias Wüchner, Martın Ochoa, and Alexander Pretschner. "Robust and Effective Malware Detection Through Quantitative Data Flow Graph Metrics". In: *SpringerLink* (July 2015), pp. 98–118. DOI: 10.1007/978-3-319-20550-2_6.

[224] Guanhua Yan and Stephan Eidenbenz. "Modeling propagation dynamics of bluetooth worms (extended version)". In: *IEEE transactions on mobile computing* 8.3 (2009), pp. 353–368.

[225] Guanhua Yan et al. "Malware propagation in online social networks: nature, dynamics, and defense implications". In: *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*. ACM. 2011, pp. 196–206.

[226] Chao Yang, Robert Harkreader, and Guofei Gu. "Die free or live hard? empirical evaluation and new design for fighting evolving twitter spammers". In: *Recent Advances in Intrusion Detection*. Springer. 2011, pp. 318–337.

[227] Chao Yang, Robert Harkreader, and Guofei Gu. "Empirical evaluation and new design for fighting evolving twitter spammers". In: *IEEE Transactions on Information Forensics and Security* 8.8 (2013), pp. 1280–1293.

[228] Chao Yang et al. "Analyzing spammers' social networks for fun and profit: a case study of cyber criminal ecosystem on twitter". In: *Proceedings of the 21st international conference on World Wide Web*. ACM. 2012, pp. 71–80.

[229] Shui Yu et al. "Malware propagation in large-scale networks". In: *IEEE Transactions on Knowledge and Data Engineering* 27.1 (2015), pp. 170–179.

[230] Robert B Zajonc. *On the primacy of affect*. 1984.

[231] ZeroFox. *The Social Media Security Timeline*. https://www.zerofox.com/security-timeline/. Nov. 2017.

[232] Zongqu Zhao. "A virus detection scheme based on features of control flow graph". In: *2011 2nd International Conference on Artificial Intelligence, Management Science and Electronic Commerce (AIMSEC)*. IEEE. 2011, pp. 943–947.

[233] Cliff C Zou, Don Towsley, and Weibo Gong. "Modeling and simulation study of the propagation and defense of internet e-mail worms". In: *IEEE Transactions on dependable and secure computing* 4.2 (2007), pp. 105–118.

[234] Cliff Changchun Zou, Weibo Gong, and Don Towsley. "Code red worm propagation modeling and analysis". In: *Proceedings of the 9th ACM conference on Computer and communications security*. ACM. 2002, pp. 138–147.

[235] Cliff C Zou et al. "The monitoring and early detection of internet worms". In: *IEEE/ACM Transactions on Networking (TON)* 13.5 (2005), pp. 961–974.

Appendix

The rules for file, process and registry that were defined to identify drive-by downloads attacks and were used to configure Capture HPC are defined below.

File exclusion list rules

```
#[+,-]	[File Access]	[Process Name]	[File Path]

####################################################
### Clean Windows 7 SP 1 System		###
####################################################
+	Read	.*	.*
+	Create	.*	.*
+	Open	.*	.*
#issue in the way process path information is communicated to capture client
+	Write	UNKNOWN	.*
+	Delete	UNKNOWN	.*
#capture
+	Write	.*	C:\\program files\\capture\\logs\\.+
+	Delete	C:\\program Files\\capture\\captureclient\.exe	C:\\program files\\capture\\.+\.zip
+	Delete	C:\\program Files\\capture\\captureclient\.exe	C:\\program files\\capture\\logs
+	Delete	C:\\program Files\\capture\\captureclient\.exe	C:\\program files\\capture\\logs\\.*
+	Write	C:\\program Files\\capture\\captureclient\.exe	C:\\program
files\\capture\\capture\.log
+	Write	C:\\program Files\\capture\\7za\.exe	C:\\program files\\capture\\capture\.log
+	Write	C:\\program Files\\capture\\7za\.exe	C:\\program files\\capture\\.+\.zip
+	Delete	C:\\program Files\\capture\\7za\.exe	C:\\program files\\capture\\.+\.zip
+	Write	C:\\program Files\\capture\\7za\.exe	C:\\progra~1\\capture\\capture\.log
+	Write	C:\\program Files\\capture\\7za\.exe	C:\\progra~1\\capture\\.+\.zip
+	Delete	C:\\program Files\\capture\\7za\.exe	C:\\progra~1\\capture\\.+\.zip
#Prefetch
+	Write	C:\\WINDOWS\\system32\\svchost\.exe	C:\\WINDOWS\\Prefetch\\.+
+	Write	System	C:\\WINDOWS\\Prefetch\\.+
#NTFS Metadata
+	Write	.*	c:\\\$mft
+	Write	.*	c:\\\$mftmirr
+	Write	.*	c:\\\$logfile
+	Write	.*	c:\\\$volume
+	Write	.*	c:\\\$directory
+	Write	.*	c:\\\$AttrDef
+	Write	.*	c:\\\$boot
+	Write	.*	c:\\\$bitmap
+	Write	.*	c:\\\$badclus
+	Write	.*	c:\\\$quota
+	Write	.*	c:\\\$upcase
+	Write	.*	c:\\\$ReplaceAttribute2
+	Write	.*	c:\\\$converttononresident
#Performance
+	Write	C:\\WINDOWS\\system32\\wbem\\wmiadap\.exe
	C:\\WINDOWS\\system32\\wbem\\Performance\\.+
+	Write	C:\\WINDOWS\\system32\\wbem\\wmiadap\.exe	C:\\WINDOWS\\system32\\Perf.*
+	Write	C:\\WINDOWS\\system32\\svchost\.exe	C:\\WINDOWS\\Prefetch\\.+
+	Write	System	C:\\WINDOWS\\Prefetch\\.+
#System Log Files
+	Write	System	C:\\Documents and Settings\\.+\\.+\.LOG
+	Write	System	C:\\WINDOWS\\system32\\config\\.+\.LOG
+	Write	System	C:\\WINDOWS\\Debug\\UserMode\\userenv\.log
+	Write	System	C:\\WINDOWS\\SoftwareDistribution\\ReportingEvents\.log
```

```
+           Write     C:\\WINDOWS\\system32\\winlogon\.exe
            C:\\WINDOWS\\Debug\\UserMode\\userenv\.log
+           Write     C:\\WINDOWS\\system32\\svchost\.exe      C:\\WINDOWS\\.+\.log
+           Write     C:\\WINDOWS\\system32\\lsass\.exe        C:\\WINDOWS\\system32\\config\\.+\.LOG
+           Write     C:\\WINDOWS\\system32\\lsass\.exe        C:\\WINDOWS\\system32\\config\\SAM
+           Write     C:\\WINDOWS\\system32\\lsass\.exe        C:\\WINDOWS\\system32\\config\\system
+           Write     C:\\WINDOWS\\system32\\lsass\.exe        C:\\WINDOWS\\system32\\config\\SECURITY
+           Write     C:\\WINDOWS\\system32\\wbem\\wmiprvse\.exe
            C:\\WINDOWS\\system32\\wbem\\Logs\\wmiprov\.log
#Windows update
+           Write     C:\\WINDOWS\\system32\\wuauclt\.exe
            C:\\WINDOWS\\SoftwareDistribution\\DataStore\\Logs\\.+
+           Write     C:\\WINDOWS\\system32\\wuauclt\.exe      C:\\WINDOWS\\WindowsUpdate\.log
+           Write     C:\\WINDOWS\\system32\\wuauclt\.exe
            C:\\WINDOWS\\SoftwareDistribution\\DataStore\\DataStore\.edb

+           Delete    C:\\WINDOWS\\system32\\wuauclt\.exe
            C:\\WINDOWS\\SoftwareDistribution\\DataStore\\Logs\\.+
+           Delete    C:\\WINDOWS\\system32\\wuauclt\.exe      C:\\WINDOWS\\WindowsUpdate\.log
+           Delete    C:\\WINDOWS\\system32\\wuauclt\.exe
            C:\\WINDOWS\\SoftwareDistribution\\DataStore\\DataStore\.edb
#System Events
+           Write     C:\\WINDOWS\\system32\\services\.exe
            C:\\WINDOWS\\system32\\config\\AppEvent\.Evt
+           Write     C:\\WINDOWS\\system32\\services\.exe
            C:\\WINDOWS\\system32\\config\\SysEvent\.Evt
+           Write     C:\\WINDOWS\\system32\\services\.exe
            C:\\WINDOWS\\system32\\config\\SecEvent\.Evt
#Mapping
+           Write     C:\\WINDOWS\\system32\\svchost\.exe      C:\\WINDOWS\\system32\\wbem\\.+
#Cataloging
+           Write     C:\\WINDOWS\\system32\\svchost\.exe      C:\\WINDOWS\\system32\\CatRoot2\\.+
+           Write     C:\\WINDOWS\\system32\\svchost\.exe      C:\\WINDOWS\\system32\\CatRoot\\.+
+           Write     C:\\WINDOWS\\system32\\svchost\.exe      C:\\Windows\\System32\\winevt\\Logs\\.+
+           Write     C:\\WINDOWS\\system32\\svchost\.exe
            C:\\Windows\\ServiceProfiles\\LocalService\\AppData\\Local\\lastalive0\.+
#System restore
+           Write     C:\\WINDOWS\\system32\\svchost\.exe
            C:\\WINDOWS\\SoftwareDistribution\\WuRedir\\.+
+           Write     C:\\WINDOWS\\system32\\svchost\.exe      C:\\System Volume Information\\_restore.*
#snapshots and crypt url
+           Write     C:\\Windows\\System32\\svchost\.exe
            C:\\Windows\\System32\\wdi\\.+\\.+\\snapshot.etl
+           Delete    C:\\Windows\\System32\\svchost\.exe
            C:\\Users\\.+\\AppData\\LocalLow\\Microsoft\\CryptnetUrlCache\\Content\\.*
+           Delete    C:\\Windows\\System32\\svchost\.exe
            C:\\Users\\.+\\AppData\\LocalLow\\Microsoft\\CryptnetUrlCache\\Content\\.+
+           Delete    C:\\Windows\\System32\\svchost\.exe
            C:\\Users\\mp\\AppData\\LocalLow\\Microsoft\\CryptnetUrlCache\\Content\\.+
+           Delete    C:\\Windows\\System32\\svchost\.exe
            C:\\Users\\mp\\AppData\\LocalLow\\Microsoft\\CryptnetUrlCache\\Content\\.*
+           Delete    C:\\Windows\\System32\\svchost\.exe
            C:\\Users\\mp\\AppData\\LocalLow\\Microsoft\\CryptnetUrlCache\\MetaData\\.*
+           Delete    C:\\Windows\\System32\\svchost\.exe
            C:\\Users\\mp\\AppData\\LocalLow\\Microsoft\\CryptnetUrlCache\\MetaData\\.+
+	Delete	C:\\Windows\\System32\\svchost\.exe
	C:\\Users\\mp\\AppData\\LocalLow\\Microsoft\\CryptnetUrlCache\\Content\\8059E9A0D314877E4
0FE93D8CCFB3C69_298C7C05A76CF4F87B7E48888C7B12A9
+	Write	C:\\Windows\\System32\\svchost\.exe
	C:\\Users\\mp\\AppData\\LocalLow\\Microsoft\\CryptnetUrlCache\\Content\\.+
+	Write	C:\\Windows\\System32\\svchost\.exe
	C:\\Users\\mp\\AppData\\LocalLow\\Microsoft\\CryptnetUrlCache\\Content\\.*
+	Write	C:\\Windows\\System32\\svchost\.exe
	C:\\Users\\mp\\AppData\\LocalLow\\Microsoft\\CryptnetUrlCache\\MetaData\\.*
+	Write	C:\\Windows\\System32\\svchost\.exe
	C:\\Users\\mp\\AppData\\LocalLow\\Microsoft\\CryptnetUrlCache\\MetaData\\.+
+	Write	C:\\Windows\\System32\\svchost\.exe
	C:\\Windows\\System32\\wdi\\ShutdownPerformanceDiagnostics_SystemData\.bin
+	Write	C:\\Windows\\System32\\svchost\.exe
	C:\\Windows\\System32\\wdi\\BootPerformanceDiagnostics_SystemData\.bin
+	Write	C:\\Windows\\System32\\svchost\.exe	C:\\Windows\\System32\\wdi\\.+\\.+
+	Write	C:\\Windows\\System32\\svchost\.exe	C:\\Windows\\System32\\wdi\\.+\\.*
#user data

+	Write	System	C:\\Documents and Settings\\.+\\Local Settings\\Application
Data\\Microsoft\\Windows\\UsrClass\.dat
####################################################
### Win 7 specific files	(part of OS)	###
####################################################
#Google chrome updating task scheduler
+	Write	System	C:\\Windows\\Tasks\\GoogleUpdateTaskMachineCore\\.job
#+	Write	System	C:\\Windows\\Tasks\\.+
#+	Write	System	C:\\Windows\\Tasks\\.*
+	Write	C:\\Windows\\System32\\svchost\.exe
	C:\\Windows\\Tasks\\GoogleUpdateTaskMachineCore\\.job
#+	Write	C:\\Windows\\System32\\svchost\.exe	C:\\Windows\\Tasks\\.+
#+	Write	C:\\Windows\\System32\\svchost\.exe	C:\\Windows\\Tasks\\.+\\.*
+	Write	C:\\Windows\\System32\\svchost\.exe	C:\\Windows\\System32\\config\\SYSTEM
+	Write	C:\\Windows\\System32\\svchost\.exe	C:\\Windows\\System32\\config\\SYSTEM.LOG1
# Some sort of windows 7 logging feature

+	Write	C:\\Windows\\System32\\svchost\.exe
	C:\\Windows\\System32\\winevt\\Logs\\Microsoft-Windows-HomeGroup Provider
Service%4Operational\.evtx
#Win 7 search indexer these are index files created by search indexer
+	Write	C:\\Windows\\System32\\SearchIndexer\.exe
	C:\\ProgramData\\Microsoft\\Search\\Data\\Applications\\Windows\\GatherLogs\\SystemIndex\\.+\
\.*
+	Write	C:\\Windows\\System32\\SearchIndexer\.exe
	C:\\ProgramData\\Microsoft\\Search\\Data\\Applications\\Windows\\GatherLogs\\SystemIndex\\.*
+	Write	C:\\Windows\\System32\\SearchIndexer\.exe
	C:\\ProgramData\\Microsoft\\Search\\Data\\Applications\\Windows\\GatherLogs\\SystemIndex\\Sy
stemIndex.4\.gthr
+	Write	C:\\Windows\\System32\\SearchIndexer\.exe
	C:\\ProgramData\\Microsoft\\Search\\Data\\Applications\\Windows\\GatherLogs\\SystemIndex\\Sy
stemIndex.4\.Crwl
+	Write	C:\\Windows\\System32\\SearchIndexer\.exe
	C:\\ProgramData\\Microsoft\\Search\\Data\\Applications\\Windows\\MSS\.log
+	Write	C:\\Windows\\System32\\SearchIndexer\.exe
	C:\\ProgramData\\Microsoft\\Search\\Data\Applications\\Windows\\.*
+	Write	C:\\Windows\\System32\\SearchIndexer\.exe
	C:\\ProgramData\\Microsoft\\Search\\Data\Applications\\Windows\\MSS\.chk
```

```
+	Write	C:\\Windows\\System32\\SearchIndexer\.exe
	C:\\ProgramData\\Microsoft\\Search\\Data\Applications\\Windows\\tmp\.edb
+	Write	C:\\Windows\\System32\\SearchIndexer\.exe
	C:\\ProgramData\\Microsoft\\Search\\Data\Applications\\Windows\\Windows\.edb
#WSB = Opened by microsoft works quite possibly log files
+	Write	C:\\Windows\\System32\\SearchIndexer\.exe
	C:\\ProgramData\\Microsoft\\Search\\Data\\Applications\\Windows\\Projects\\SystemIndex\\Index
er\\CiFiles\\.+wsb
+	Write	C:\\Windows\\System32\\SearchIndexer\.exe
	C:\\ProgramData\\Microsoft\\Search\\Data\\Applications\\Windows\\Projects\\SystemIndex\\Index
er\\CiFiles\\.+wid
+	Write	C:\\Windows\\System32\\SearchIndexer\.exe
	C:\\ProgramData\\Microsoft\\Search\\Data\\Applications\\Windows\\Projects\\SystemIndex\\Index
er\\CiFiles\\.*
#Networking for windows 7 id files
+	Write	C:\\Windows\\System32\\svchost\.exe
	C:\\Windows\\ServiceProfiles\\LocalService\\AppData\\Roaming\\PeerNetworking\\idstore\.sst\.ne
w
+	Write	C:\\Windows\\System32\\svchost\.exe
	C:\\Windows\\ServiceProfiles\\LocalService\\AppData\\Roaming\\PeerNetworking\\idstore\.+
+	Write	C:\\Windows\\System32\\svchost\.exe
	C:\\Windows\\ServiceProfiles\\LocalService\\AppData\\Local\\.+
#Readyboot
+	Delete	C:\\Windows\\System32\\svchost\.exe	C:\\Windows\\Prefetch\\ReadyBoot\\.+
#wfpdiag- network trace files
+	Write	C:\\Windows\\System32\\svchost\.exe	C:\\Windows\\System32\\wfp\\wfpdiag\.etl
+	Write	C:\\Windows\\System32\\svchost\.exe
	C:\\ProgramData\\Microsoft\\Crypto\\RSA\\MachineKeys\\4943bf62402c78e35e6a41d057394ac7_0
e17101e-bd8f-4859-ad29-a2b0529004a5
+	Write	C:\\Windows\\System32\\services\.exe
	C:\\Windows\\System32\\LogFiles\\Scm\\28e6b4d9-ed90-4c80-b08b-814efb653eff
+	Write	C:\\Windows\\System32\\svchost\.exe
	C:\\Windows\\ServiceProfiles\\LocalService\\AppData\\Roaming\\PeerNetworking\\661081597c497
31a0cbf909a3a1a816bcd7d3e00.HomeGroupClassifier\\473194124b68995eabe9d1640fa2be41\\grouping\\t
mp\.edb
+	Write	C:\\Windows\\System32\\svchost\.exe
	C:\\Windows\\ServiceProfiles\\LocalService\\AppData\\Roaming\\PeerNetworking\\661081597c497
31a0cbf909a3a1a816bcd7d3e00.HomeGroupClassifier\\473194124b68995eabe9d1640fa2be41\\grouping\\t
mp\.edb
####################################################
### Internet Explorer 6.0 SP2(browser activities)###
####################################################
#somehow VMwareService & System accesses the same files when IE is browsing.
+	Write	C:\\Program Files\\VMware\\VMware Tools\\VMwareService\.exe	.*
+	Write	System	.*
# IE Temporary Files/Internet Cache.
+	Write	C:\\Program Files\\Internet Explorer\\iexplore\.exe	C:\\WINDOWS\\Temp\\.+
+	Write	C:\\Program Files\\Internet Explorer\\iexplore\.exe	C:\\Documents and Settings\\.+\\Local
Settings\\Temporary Internet Files\\Content\.IE5\\.+
+	Write	C:\\Program Files\\Internet Explorer\\iexplore\.exe	C:\\Documents and Settings\\.+\\Local
Settings\\Temp\\.+tmp
+	Delete	C:\\Program Files\\Internet Explorer\\iexplore\.exe	C:\\WINDOWS\\Temp\\.+
+	Delete	C:\\Program Files\\Internet Explorer\\iexplore\.exe	C:\\Documents and Settings\\.+\\Local
Settings\\Temporary Internet Files\\Content\.IE5\\.+
+	Delete	C:\\Program Files\\Internet Explorer\\iexplore\.exe	C:\\Documents and Settings\\.+\\Local
Settings\\Temp\\.+tmp
# History
```

```
+	Write	C:\\Program Files\\Internet Explorer\\iexplore\.exe	C:\\Documents and Settings\\.+\\Local
Settings\\History\\History.IE5\\.+
+	Delete	C:\\Program Files\\Internet Explorer\\iexplore\.exe	C:\\Documents and Settings\\.+\\Local
Settings\\History\\History.IE5\\.+
# IE Cookies
+	Write	C:\\Program Files\\Internet Explorer\\iexplore\.exe	C:\\Documents and
Settings\\.+\\Cookies\\.+
+	Write	C:\\Program Files\\Internet Explorer\\iexplore\.exe	C:\\Documents and
Settings\\.+\\Cookies\\index.dat
+	Delete	C:\\Program Files\\Internet Explorer\\iexplore\.exe	C:\\Documents and
Settings\\.+\\Cookies\\.+
+	Delete	C:\\Program Files\\Internet Explorer\\iexplore\.exe	C:\\Documents and
Settings\\.+\\Cookies\\index.dat
# User data
+	Write	C:\\Program Files\\Internet Explorer\\iexplore\.exe	C:\\Documents and
Settings\\.+\\Application Data\\Microsoft\\CryptnetUrlCache
+	Write	C:\\Program Files\\Internet Explorer\\iexplore\.exe	C:\\Documents and
Settings\\.+\\UserData\\.+
+	Delete	C:\\Program Files\\Internet Explorer\\iexplore\.exe	C:\\Documents and
Settings\\.+\\Application Data\\Microsoft\\CryptnetUrlCache
+	Delete	C:\\Program Files\\Internet Explorer\\iexplore\.exe	C:\\Documents and
Settings\\.+\\UserData\\.+
# Plug ins (like Flash player)
+	Write	C:\\Program Files\\Internet Explorer\\iexplore\.exe	C:\\Documents and
Settings\\.+\\Application Data\\.+
+	Delete	C:\\Program Files\\Internet Explorer\\iexplore\.exe	C:\\Documents and
Settings\\.+\\Application Data\\.+
# DRM related stuff
+	Write	C:\\Program Files\\Internet Explorer\\iexplore\.exe	C:\\Documents and
Settings\\.+\\DRM\\.+
+	Delete	C:\\Program Files\\Internet Explorer\\iexplore\.exe	C:\\Documents and
Settings\\.+\\DRM\\.+
# msg activeX
+	Write	C:\\Program Files\\Messenger\\msmsgs\.exe		C:\\Documents and
Settings\\.+\\NTUSER.DAT.LOG
+	Delete	C:\\Program Files\\Messenger\\msmsgs\.exe		C:\\Documents and
Settings\\.+\\NTUSER.DAT.LOG
####################################################
### Internet Explorer 8.0  updated for win7  ###
####################################################
#System level log files/cache files (some regarding certificates of websites) Generally written after the visit of
any malicious website.
#User Data for win 7's updated file system
+	Write	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	C:\\Users\\.+\\AppData\\LocalLow\\Microsoft\\CryptnetUrlCache\\.+\\.*
#IE Temporary Files
+	Write	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	C:\\Users\\.+\\AppData\\Local\\Temp\\.+tmp
+	Write	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	C:\\Users\\mp\\AppData\\Local\\Temp\\.+tmp
+	Delete	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	C:\\Users\\mp\\AppData\\Local\\Temp\\.+tmp
+	Delete	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	C:\\Users\\.+\\AppData\\Local\\Temp\\.+tmp
+	Write	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	C:\\Users\\mp\\AppData\\Local\\Microsoft\\Windows\\Temporary Internet Files\\Content.IE5\\.+
+           Delete    C:\\Program Files\\Internet Explorer\\iexplore\.exe
            C:\\Users\\mp\\AppData\\Local\\Microsoft\\Windows\\Temporary Internet Files\\Content.IE5\\.+
######################################## is this really safe? some temp files can be ok; others cannot
=[  #################################################################
+           write     C:\\Program Files\\Internet Explorer\\iexplore\.exe
            C:\\Users\\mp\\AppData\\Local\\Microsoft\\Windows\\Temporary Internet Files\\.+
+           Delete    C:\\Program Files\\Internet Explorer\\iexplore\.exe
            C:\\Users\\mp\\AppData\\Local\\Microsoft\\Windows\\Temporary Internet Files\\.+
# IE Internet Cache.
+           Write     C:\\Program Files\\Internet Explorer\\iexplore\.exe
            C:\\Users\\.+\\AppData\\Local\\Microsoft\\Windows\\Temporary Internet Files\\Content.IE5\\+.txt
+           Write     C:\\Program Files\\Internet Explorer\\iexplore\.exe
            C:\\Users\\.+\\AppData\\Local\\Microsoft\\Internet Explorer\\Recovery\\.+dat
#IE Cookies
+           Write     C:\\Program Files\\Internet Explorer\\iexplore\.exe
            C:\\Users\\mp\\AppData\\Roaming\\Microsoft\\Windows\\Cookies\\.+txt
+           Write     C:\\Program Files\\Internet Explorer\\iexplore\.exe
            C:\\Users\\.+\\AppData\\Roaming\\Microsoft\\Windows\\Cookies\\.+txt
+           Write     C:\\Program Files\\Internet Explorer\\iexplore\.exe
            C:\\Users\\.+\\AppData\\Roaming\\Microsoft\\Windows\\Cookies\\.+
+           Delete    C:\\Program Files\\Internet Explorer\\iexplore\.exe
            C:\\Users\\.+\\AppData\\Roaming\\Microsoft\\Windows\\Cookies\\.+
+           Write     C:\\Program Files\\Internet Explorer\\iexplore\.exe
            C:\\Users\\mp\\AppData\\Roaming\\Microsoft\\Windows\\Cookies\\.+
+           Delete    C:\\Program Files\\Internet Explorer\\iexplore\.exe
            C:\\Users\\mp\\AppData\\Roaming\\Microsoft\\Windows\\Cookies\\.+
#IE History ###Volatile
+           Write     C:\\Program Files\\Internet Explorer\\iexplore\.exe
            C:\\Users\\mp\\AppData\\Local\\Microsoft\\Windows\\History\History.IE5\\.+
+           Delete    C:\\Program Files\\Internet Explorer\\iexplore\.exe
            C:\\Users\\mp\\AppData\\Local\\Microsoft\\Windows\\History\History.IE5\\.+
+           Write     C:\\Program Files\\Internet Explorer\\iexplore\.exe
            C:\\Users\\mp\\AppData\\Local\\Microsoft\\Windows\\History\History.IE5\\.+\\.*
+           Delete    C:\\Program Files\\Internet Explorer\\iexplore\.exe
            C:\\Users\\mp\\AppData\\Local\\Microsoft\\Windows\\History\History.IE5\\.+\\.*
+           Write     C:\\Program Files\\Internet Explorer\\iexplore\.exe
            C:\\Users\\mp\\AppData\\Local\\Microsoft\\Windows\\History\History.IE5\\.+\\index\.dat
+           Write     C:\\Program Files\\Internet Explorer\\iexplore\.exe
            C:\\Users\\mp\\AppData\\Local\\Microsoft\\Windows\\History\\History.IE5\\MSHist0120140224201
40225\\.*
+           Write     C:\\Program Files\\Internet Explorer\\iexplore\.exe
            C:\\Users\\mp\\AppData\\Local\\Microsoft\\Windows\\History\\History.IE5\\MSHist0120140224201
40225\\index\.dat
+           Write     C:\\Program Files\\Internet Explorer\\iexplore\.exe
            C:\\Users\\mp\\AppData\\Local\\Microsoft\\Windows\\History\\History.IE5\\.+\\index\.dat
+           Delete    C:\\Program Files\\Internet Explorer\\iexplore\.exe
            C:\\Users\\mp\\AppData\\Local\\Microsoft\\Windows\\History\\History.IE5\\.+\\index\.dat
#frame icon cache
+           Write     C:\\Program Files\\Internet Explorer\\iexplore\.exe
            C:\\Users\\mp\\AppData\\Local\\Microsoft\\Internet Explorer\\frameiconcache.dat
#GoogleUpdateFiles
#+          Write     C:\\Windows\\System32\\svchost\.exe       C:\\Windows\\Tasks\\.*
#Morelogfiles
+           Write     C:\\Windows\\System32\\services\.exe      C:\\Windows\\System32\\LogFiles\\.*
+           Write     System   C:\\Windows\\Tasks\\.*
#IE WEB STORAGE (DOMSTORE) downloads and saves cookies etc - no sign of malicious activity previously put
there.
```

```
+           Write     C:\\Program Files\\Internet Explorer\\iexplore\.exe
            C:\\Users\\mp\\AppData\\Local\\Microsoft\\Internet Explorer\\DOMStore\\.+
#flash macromedia C:\Users\mp\AppData\Roaming\Macromedia\Flash
Player\macromedia.com\support\flashplayer\sys\#s.ytimg.com\settings.sxx#
+           Write     C:\\Program Files\\Internet Explorer\\iexplore\.exe
            C:\\Users\\mp\\AppData\\Roaming\\Macromedia\\Flash
Player\\macromedia.com\\support\\flashplayer\\sys\#s.ytimg.com\\settings.sxx
+           Delete    C:\\Program Files\\Internet Explorer\\iexplore\.exe
            C:\\Users\\mp\\AppData\\Roaming\\Macromedia\\Flash
Player\\macromedia.com\\support\\flashplayer\\sys\#s.ytimg.com\\settings.sxx
+           Write     C:\\Program Files\\Internet Explorer\\iexplore\.exe
            C:\\Users\\mp\\AppData\\Roaming\\Macromedia\\Flash
Player\\macromedia.com\\support\\flashplayer\\sys\\settings.sxx
+           Delete    C:\\Program Files\\Internet Explorer\\iexplore\.exe
            C:\\Users\\mp\\AppData\\Roaming\\Macromedia\\Flash
Player\\macromedia.com\\support\\flashplayer\\sys\\settings.sxx
+           Write     C:\\Program Files\\Internet Explorer\\iexplore\.exe
            C:\\Users\\mp\\AppData\\Roaming\\Macromedia\\Flash
Player\\macromedia.com\\support\\flashplayer\\sys\\settings.sol
+           Delete    C:\\Program Files\\Internet Explorer\\iexplore\.exe
            C:\\Users\\mp\\AppData\\Roaming\\Macromedia\\Flash
Player\\macromedia.com\\support\\flashplayer\\sys\\settings.sol
+           Write     C:\\Program Files\\Internet Explorer\\iexplore\.exe
            C:\\Users\\mp\\AppData\\Roaming\\Macromedia\\Flash
Player\\macromedia.com\\support\\flashplayer\\sys\#s.ytimg.com\\settings.sol
+           Delete    C:\\Program Files\\Internet Explorer\\iexplore\.exe
            C:\\Users\\mp\\AppData\\Roaming\\Macromedia\\Flash
Player\\macromedia.com\\support\\flashplayer\\sys\#s.ytimg.com\\settings.sol
+           Write     C:\\Program Files\\Internet Explorer\\iexplore\.exe
            C:\\Users\\mp\\AppData\\Roaming\\Macromedia\\Flash
Player\\macromedia.com\\support\\flashplayer\\sys\#s.ytimg.com\\.*
+           Write     C:\\Program Files\\Internet Explorer\\iexplore\.exe
            C:\\Users\\mp\\AppData\\Roaming\\Macromedia\\Flash
Player\\macromedia.com\\support\\flashplayer\\sys\\.*

+           Delete    C:\\Program Files\\Internet Explorer\\iexplore\.exe
            C:\\Users\\mp\\AppData\\Roaming\\Macromedia\\Flash
Player\\macromedia.com\\support\\flashplayer\\sys\#s.ytimg.com\\.*
+           Delete    C:\\Program Files\\Internet Explorer\\iexplore\.exe
            C:\\Users\\mp\\AppData\\Roaming\\Macromedia\\Flash
Player\\macromedia.com\\support\\flashplayer\\sys\\.*
+           Write     C:\\Program Files\\Internet Explorer\\iexplore\.exe
            C:\\Users\\mp\\AppData\\Roaming\\Macromedia\\Flash
Player\\#SharedObjects\\GV7ZM8GP\\s.ytimg.com\\restore.sxx
+           Delete    C:\\Program Files\\Internet Explorer\\iexplore\.exe
            C:\\Users\\mp\\AppData\\Roaming\\Macromedia\\Flash
Player\\#SharedObjects\\GV7ZM8GP\\s.ytimg.com\\restore.sxx
#+          Write     C:\\Program Files\\Internet Explorer\\iexplore\.exe
            C:\\Users\\mp\\AppData\\Roaming\\Macromedia\\Flash Player\\.+\\GV7ZM8GP\\s.ytimg.com\\.*
#+          Delete    C:\\Program Files\\Internet Explorer\\iexplore\.exe
            C:\\Users\\mp\\AppData\\Roaming\\Macromedia\\Flash Player\\.+\\.*
+           Write     C:\\Program Files\\Internet Explorer\\iexplore\.exe
            C:\\Users\\mp\\AppData\\Roaming\\Macromedia\\Flash Player\\.+\\.+\\s.ytimg.com\\videostats.sxx
+           Write     C:\\Program Files\\Internet Explorer\\iexplore\.exe
            C:\\Users\\mp\\AppData\\Roaming\\Macromedia\\Flash
Player\\.+\\.+\\s.ytimg.com\\soundData.sxx
```

```
+	Write	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	C:\\Users\\mp\\AppData\\Roaming\\Macromedia\\Flash Player\\.+\\.+\\s.ytimg.com\\restore.sxx"
+	Delete	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	C:\\Users\\mp\\AppData\\Roaming\\Macromedia\\Flash Player\\.+\\.+\\s.ytimg.com\\videostats.sxx
+	Delete	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	C:\\Users\\mp\\AppData\\Roaming\\Macromedia\\Flash
Player\\.+\\.+\\s.ytimg.com\\soundData.sxx
+	Delete	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	C:\\Users\\mp\\AppData\\Roaming\\Macromedia\\Flash Player\\.+\\.+\\s.ytimg.com\\restore.sxx"
+	Delete	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	C:\\Users\\mp\\AppData\\Roaming\\Macromedia\\Flash Player\\.+\\.+\\.*
+	Write	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	C:\\Users\\mp\\AppData\\Roaming\\Macromedia\\Flash
Player\\.+\\4WCZFSR2\\s.ytimg.com\\restore.sxx
+	Delete	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	C:\\Users\\mp\\AppData\\Roaming\\Macromedia\\Flash
Player\\.+\\4WCZFSR2\\s.ytimg.com\\restore.sxx
####################################################
### firefox v26  updated for win7                ###
####################################################
#Relatively safe file no sign of maliciousness
+	Write	C:\\Program Files\\Mozilla Firefox\\firefox\.exe
	C:\\Users\\mp\\AppData\\Local\\Mozilla\\Firefox\\Profiles\\rg570dm1.default\\startupCache\\start
upCache.4.little
+	Delete	C:\\Program Files\\Mozilla Firefox\\firefox\.exe
	C:\\Users\\mp\\AppData\\Local\\Mozilla\\Firefox\\Profiles\\rg570dm1.default\\startupCache\\start
upCache.4.little
+	Write	C:\\Program Files\\Mozilla Firefox\\firefox\.exe
	C:\\Users\\.+\\AppData\\Local\\Mozilla\\Firefox\\Profiles\\rg570dm1.default\\startupCache\\startu
pCache.4.little
+	Delete	C:\\Program Files\\Mozilla Firefox\\firefox\.exe
	C:\\Users\\.+\\AppData\\Local\\Mozilla\\Firefox\\Profiles\\rg570dm1.default\\startupCache\\startu
pCache.4.little
#startup cache
+	Write	C:\\Program Files\\Mozilla Firefox\\firefox\.exe
	C:\\Users\\mp\\AppData\\Local\\Mozilla\\Firefox\\Profiles\\rg570dm1.default\\startupCache
+	Delete	C:\\Program Files\\Mozilla Firefox\\firefox\.exe
	C:\\Users\\mp\\AppData\\Local\\Mozilla\\Firefox\\Profiles\\rg570dm1.default\\startupCache
#first time run of firefox creates and stores a profile there
+	Write	C:\\Program Files\\Mozilla Firefox\\firefox\.exe
	C:\\Users\\mp\\AppData\\Roaming\\Mozilla\\Firefox\\Profiles\\rg570dm1.default\\compatibility.ini
+	Delete	C:\\Program Files\\Mozilla Firefox\\firefox\.exe
	C:\\Users\\mp\\AppData\\Roaming\\Mozilla\\Firefox\\Profiles\\rg570dm1.default\\compatibility.ini
+	Write	C:\\Program Files\\Mozilla Firefox\\firefox\.exe
	C:\\Users\\.+\\AppData\\Roaming\\Mozilla\\Firefox\\Profiles\\rg570dm1.default\\compatibility.ini
+	Delete	C:\\Program Files\\Mozilla Firefox\\firefox\.exe
	C:\\Users\\.+\\AppData\\Roaming\\Mozilla\\Firefox\\Profiles\\rg570dm1.default\\compatibility.ini
#More firefox profiles
+	Write	C:\\Program Files\\Mozilla Firefox\\firefox\.exe
	C:\\Users\\mp\\AppData\\Roaming\\Mozilla\\Firefox\\Profiles\\rg570dm1.default\\webapps\webap
ps-1.json
+	Delete	C:\\Program Files\\Mozilla Firefox\\firefox\.exe
	C:\\Users\\mp\\AppData\\Roaming\\Mozilla\\Firefox\\Profiles\\rg570dm1.default\\webapps\webap
ps-1.json
#Always created at first launch of firefox
```

```
+	Write	C:\\Program Files\\Mozilla Firefox\\firefox\.exe
	C:\\Users\\mp\\AppData\\Roaming\\Mozilla\\Firefox\\Profiles\\rg570dm1.default\\marionette.log
+	Delete	C:\\Program Files\\Mozilla Firefox\\firefox\.exe
	C:\\Users\\mp\\AppData\\Roaming\\Mozilla\\Firefox\\Profiles\\rg570dm1.default\\marionette.log
#Firefox temp files
+	Write	C:\\Program Files\\Mozilla Firefox\\firefox\.exe
	C:\\Users\\mp\\AppData\\Local\\Microsoft\\Windows\\Temporary Internet Files\\desktop.ini
+	Delete	C:\\Program Files\\Mozilla Firefox\\firefox\.exe
	C:\\Users\\mp\\AppData\\Local\\Microsoft\\Windows\\Temporary Internet Files\\desktop.ini
#desktopini C:\Users\mp\AppData\Local\Microsoft\Windows\Temporary Internet
Files\Content.IE5\desktop.ini
+	Write	C:\\Program Files\\Mozilla Firefox\\firefox\.exe
	C:\\Users\\mp\\AppData\\Local\\Microsoft\\Windows\\Temporary Internet
Files\\Content.IE5\desktop.ini
+	Delete	C:\\Program Files\\Mozilla Firefox\\firefox\.exe
	C:\\Users\\mp\\AppData\\Local\\Microsoft\\Windows\\Temporary Internet
Files\\Content.IE5\desktop.ini
+	Write	C:\\Program Files\\Mozilla Firefox\\firefox\.exe
	C:\\Users\\mp\\AppData\\Local\\Microsoft\\Windows\\Temporary Internet Files\\desktop.ini
+	Delete	C:\\Program Files\\Mozilla Firefox\\firefox\.exe
	C:\\Users\\mp\\AppData\\Local\\Microsoft\\Windows\\Temporary Internet Files\\desktop.ini
+	Write	C:\\Program Files\\Mozilla Firefox\\firefox\.exe
	C:\\Users\\mp\\AppData\\Local\\Microsoft\\Windows\\Temporary Internet
Files\\Content.IE5\\desktop.ini
+	Delete	C:\\Program Files\\Mozilla Firefox\\firefox\.exe
	C:\\Users\\mp\\AppData\\Local\\Microsoft\\Windows\\Temporary Internet
Files\\Content.IE5\\desktop.ini
+	Write	C:\\Program Files\\Mozilla Firefox\\firefox\.exe
	C:\\Users\\mp\\AppData\\Local\\Temp\\mozilla-temp-files\\mozilla-temp-12941
+	Write	C:\\Program Files\\Mozilla Firefox\\firefox\.exe
	C:\\Users\\mp\\AppData\\Local\\Temp\\mozilla-temp-files\\.+
#Firefox cache
+	Write	C:\\Program Files\\Mozilla Firefox\\firefox\.exe
	C:\\Users\\mp\\AppData\\Local\\Mozilla\\Firefox\\Profiles\\rg570dm1.default\\Cache\\_CACHE_MA
P_
+	Delete	C:\\Program Files\\Mozilla Firefox\\firefox\.exe
	C:\\Users\\mp\\AppData\\Local\\Mozilla\\Firefox\\Profiles\\rg570dm1.default\\Cache\\_CACHE_MA
P_
+	Write	C:\\Program Files\\Mozilla Firefox\\firefox\.exe
	C:\\Users\\mp\\AppData\\Local\\Mozilla\\Firefox\\Profiles\\rg570dm1.default\\Cache\\.*
+	Delete	C:\\Program Files\\Mozilla Firefox\\firefox\.exe
	C:\\Users\\mp\\AppData\\Local\\Mozilla\\Firefox\\Profiles\\rg570dm1.default\\Cache\\.*
+	Write	C:\\Program Files\\Mozilla Firefox\\firefox\.exe
	C:\\Users\\mp\AppData\\Local\\Mozilla\\Firefox\\Profiles\\rg570dm1.default\\Cache\\.+\F1\.*
+	Write	C:\\Program Files\\Mozilla Firefox\\firefox\.exe
	C:\\Users\\mp\\AppData\\Local\\Mozilla\\Firefox\\Profiles\\rg570dm1.default\\Cache\\.*
+	Write	C:\\Program Files\\Mozilla Firefox\\firefox\.exe
	C:\\Users\\mp\AppData\\Local\\Mozilla\\Firefox\\Profiles\\rg570dm1.default\\Cache\\.+\F1\.*
+	Write	C:\\Program Files\\Mozilla Firefox\\firefox\.exe
	C:\\Users\\mp\\AppData\\Local\\Mozilla\\Firefox\\Profiles\\rg570dm1.default\\Cache\\.*
+	Write	C:\\Program Files\\Mozilla Firefox\\firefox\.exe
	C:\\Users\\mp\\AppData\\Local\\Mozilla\\Firefox\\Profiles\\rg570dm1.default\\Cache\\.+
+	Write	C:\\Program Files\\Mozilla Firefox\\firefox\.exe
	C:\\Users\\mp\\AppData\\Local\\Mozilla\\Firefox\\Profiles\\rg570dm1.default\\_CACHE_CLEAN_
+	Write	C:\\Program Files\\Mozilla Firefox\\firefox\.exe
	C:\\Users\\mp\\AppData\\Local\\Mozilla\\Firefox\\Profiles\\rg570dm1.default\\.*
```

```
#places.sqlite stores the annotations, bookmarks, favorite icons, input history, keywords, and browsing
history.
+           Write     C:\\Program Files\\Mozilla Firefox\\firefox\.exe
            C:\\Users\\mp\\AppData\\Roaming\\Mozilla\\Firefox\\Profiles\\rg570dm1.default\\places.sqlite-wal
+           Delete    C:\\Program Files\\Mozilla Firefox\\firefox\.exe
            C:\\Users\\mp\\AppData\\Roaming\\Mozilla\\Firefox\\Profiles\\rg570dm1.default\\places.sqlite-wal
+           Write     C:\\Program Files\\Mozilla Firefox\\firefox\.exe
            C:\\Users\\mp\\AppData\\Roaming\\Mozilla\\Firefox\\Profiles\\rg570dm1.default\\places.sqlite
+           Delete    C:\\Program Files\\Mozilla Firefox\\firefox\.exe
            C:\\Users\\mp\\AppData\\Roaming\\Mozilla\\Firefox\\Profiles\\rg570dm1.default\\places.sqlite
#cookies.sqliteHolds all of your cookies, including login information, session data, and preferences.
+           Write     C:\\Program Files\\Mozilla Firefox\\firefox\.exe
            C:\\Users\\mp\\AppData\\Roaming\\Mozilla\\Firefox\\Profiles\\rg570dm1.default\\cookies.sqlite-
wal
+           Delete    C:\\Program Files\\Mozilla Firefox\\firefox\.exe
            C:\\Users\\mp\\AppData\\Roaming\\Mozilla\\Firefox\\Profiles\\rg570dm1.default\\cookies.sqlite-
wal
+           Write     C:\\Program Files\\Mozilla Firefox\\firefox\.exe
            C:\\Users\\mp\\AppData\\Roaming\\Mozilla\\Firefox\\Profiles\\rg570dm1.default\\cookies.sqlite
+           Delete    C:\\Program Files\\Mozilla Firefox\\firefox\.exe
            C:\\Users\\mp\\AppData\\Roaming\\Mozilla\\Firefox\\Profiles\\rg570dm1.default\\cookies.sqlite
#places.sqlite stores the annotations, bookmarks, favorite icons, input history, keywords, and browsing
history.

+           Write     C:\\Program Files\\Mozilla Firefox\\firefox\.exe
            C:\\Users\\mp\\AppData\\Roaming\\Mozilla\\Firefox\\Profiles\\rg570dm1.default\\permissions.sqlit
e-journal
+           Delete    C:\\Program Files\\Mozilla Firefox\\firefox\.exe
            C:\\Users\\mp\\AppData\\Roaming\\Mozilla\\Firefox\\Profiles\\rg570dm1.default\\permissions.sqlit
e-journal
+           Delete    C:\\Program Files\\Mozilla Firefox\\firefox\.exe
            C:\\Users\mp\\AppData\\Roaming\\Mozilla\\Firefox\\Profiles\\rg570dm1.default\\.*
+           Write     C:\\Program Files\\Mozilla Firefox\\firefox\.exe
            C:\\Users\\mp\\AppData\\Roaming\\Mozilla\\Firefox\\Profiles\\rg570dm1.default\\permissions.sqlit
e
#safebrowsing malware/pshish etc files (built in firefox security feature)
#safebrowsing malware
+           Write     C:\\Program Files\\Mozilla Firefox\\firefox\.exe
            C:\\Users\\mp\\AppData\\Local\\Mozilla\\Firefox\\Profiles\\rg570dm1.default\\safebrowsing\\.*
+           Delete    C:\\Program Files\\Mozilla Firefox\\firefox\.exe
            C:\\Users\\mp\\AppData\\Local\\Mozilla\\Firefox\\Profiles\\rg570dm1.default\\safebrowsing\\test-
malware-simple.cache
+           Write     C:\\Program Files\\Mozilla Firefox\\firefox\.exe
            C:\\Users\\mp\\AppData\\Local\\Mozilla\\Firefox\\Profiles\\rg570dm1.default\\safebrowsing\\test-
malware-simple.pset
+           Delete    C:\\Program Files\\Mozilla Firefox\\firefox\.exe
            C:\\Users\\mp\\AppData\\Local\\Mozilla\\Firefox\\Profiles\\rg570dm1.default\\safebrowsing\\test-
malware-simple.pset\
+           Write     C:\\Program Files\\Mozilla Firefox\\firefox\.exe
            C:\\Users\\mp\\AppData\\Local\\Mozilla\\Firefox\\Profiles\\rg570dm1.default\\safebrowsing\\test-
malware-simple.sbstore
+           Delete    C:\\Program Files\\Mozilla Firefox\\firefox\.exe
            C:\\Users\\mp\\AppData\\Local\\Mozilla\\Firefox\\Profiles\\rg570dm1.default\\safebrowsing\\test-
malware-simple.sbstore
#safebrowsing phish
+           Write     C:\\Program Files\\Mozilla Firefox\\firefox\.exe
            C:\\Users\\mp\\AppData\\Local\\Mozilla\\Firefox\\Profiles\\rg570dm1.default\\safebrowsing\\.*
```

```
+           Delete    C:\\Program Files\\Mozilla Firefox\\firefox\.exe
            C:\\Users\\mp\\AppData\\Local\\Mozilla\\Firefox\\Profiles\\rg570dm1.default\\safebrowsing\\test-
phish-simple.pset
+           Write     C:\\Program Files\\Mozilla Firefox\\firefox\.exe
            C:\\Users\\mp\\AppData\\Local\\Mozilla\\Firefox\\Profiles\\rg570dm1.default\\safebrowsing\\test-
phish-simple.sbstore
+           Delete    C:\\Program Files\\Mozilla Firefox\\firefox\.exe
            C:\\Users\\mp\\AppData\\Local\\Mozilla\\Firefox\\Profiles\\rg570dm1.default\\safebrowsing\\test-
phish-simple.sbstore
+           Write     C:\\Program Files\\Mozilla Firefox\\firefox\.exe
            C:\\Users\\mp\\AppData\\Local\\Mozilla\\Firefox\\Profiles\\rg570dm1.default\\safebrowsing\\test-
phish-simple.cache
#Safebrowsing to delete folder
+           Delete    C:\\Program Files\\Mozilla Firefox\\firefox\.exe
            C:\\Users\\mp\\AppData\\Local\\Mozilla\\Firefox\\Profiles\\rg570dm1.default\\safebrowsing-
to_delete\\.*
+           Delete    C:\\Program Files\\Mozilla Firefox\\firefox\.exe
            C:\\Users\\mp\\AppData\\Local\\Mozilla\\Firefox\\Profiles\\rg570dm1.default\\safebrowsing-
to_delete
#+          Delete    C:\\Program Files\\Mozilla Firefox\\firefox\.exe
            C:\\Users\\mp\\AppData\\Local\\Mozilla\\Firefox\\Profiles\\rg570dm1.default\\safebrowsing-
to_delete\\test-phish-simple.sbstore
#thumbails
+           Write     C:\\Program Files\\Mozilla Firefox\\firefox\.exe
            C:\\Users\\mp\\AppData\\Local\\Mozilla\\Firefox\\Profiles\\rg570dm1.default\\thumbnails\\.+
#Certificates
+           Write     C:\\Program Files\\Mozilla Firefox\\firefox\.exe
            C:\\Users\\mp\\AppData\\Roaming\\Mozilla\\Firefox\\Profiles\\rg570dm1.default\\cert8.db
#Session data (saves session data including open win and tabs)
+           Write     C:\\Program Files\\Mozilla Firefox\\firefox\.exe
            C:\\Users\\mp\\AppData\\Roaming\\Mozilla\\Firefox\\Profiles\\rg570dm1.default\\sessionstore.js.t
mp
#Firefox Preferances
+           Write     C:\\Program Files\\Mozilla Firefox\\firefox\.exe
            C:\\Users\\mp\\AppData\\Roaming\\Mozilla\\Firefox\\Profiles\\rg570dm1.default\\prefs-1.js
#URL Classifier
+           Write     C:\\Program Files\\Mozilla Firefox\\firefox\.exe
            C:\\Users\\mp\\AppData\\Roaming\\Mozilla\\Firefox\\Profiles\\rg570dm1.default\\urlclassifierkey3.
txt
#webappstore
+           Write     C:\\Program Files\\Mozilla Firefox\\firefox\.exe
            C:\\Users\\mp\\AppData\\Roaming\\Mozilla\\Firefox\\Profiles\\rg570dm1.default\\webappsstore.sq
lite-wal

+           Delete    C:\\Program Files\\Mozilla Firefox\\firefox\.exe
            C:\\Users\\mp\\AppData\\Roaming\\Mozilla\\Firefox\\Profiles\\rg570dm1.default\\webappsstore.sq
lite-wal
+           Write     C:\\Program Files\\Mozilla Firefox\\firefox\.exe
            C:\\Users\\mp\\AppData\\Roaming\\Mozilla\\Firefox\\Profiles\\rg570dm1.default\\.*
####################################################
### Minus List - General Malicious Activity   ###
####################################################
# Alert about executables or scripts that are written to disk
-           Write     .*          .+\.bat
-           Write     .*          .+\.cmd
-           Write     .*          .+\.exe
#-          Write     .*          .+\.inf
```

```
#-	Write	.*	.+\.lnk
#-	Write	.*	.+\.msi
#-	Write	.*	.+\.msp
#-	Write	.*	.+\.pif
#-	Write	.*	.+\.reg
#-	Write	.*	.+\.sct
#-	Write	.*	.+\.shs
#-	Write	.*	.+\.scr
#-	Write	.*	.+\.wsc
#-	Write	.*	.+\.wsf
#-	Write	.*	.+\.wsh
#commented out for IE because \.com cache files and \.vb script files are very common
#-	Write	.*	.+\.vb
#-	Write	.*	.+\.com
# Alert about modifications to startup locations
-	Write	.*	C:\\Documents and Settings\\.+\\Start Menu\\Programs\\Startup.+
-	Write	.*	C:\\WINDOWS\\win.ini
-	Write	.*	C:\\WINDOWS\\Tasks\\.+
```

Process exclusion list

```
#[+,-]	[Process Created][Parent Process]	[Process Path]
######################################################
### Clean Windows 7 SP 1 System	done			###
######################################################
#issue in the way process path information is communicated to capture client
+	UNKNOWN	.*	UNKNOWN
#capture client itself
+	CaptureClient.exe	.*	C:\\Program Files\\Capture\\CaptureClient\.exe
+	CaptureClient.bat.*	C:\\Program Files\\Capture\\CaptureClient\.bat
+	7za.exe	.*	C:\\Program Files\\Capture\\7za\.exe
#
#Windows update (it runs even if disabled)
+	wuauclt.exe	.*	C:\\WINDOWS\\system32\\wuauclt\.exe
#
+	savedump.exe	.*	C:\\WINDOWS\\system32\\savedump\.exe
#Standard screensaver
+	logon.scr	.*	C:\\WINDOWS\\system32\\logon\.scr
#
#defragmenter
+	dfrgntfs.exe	.*	C:\\WINDOWS\\system32\\dfrgntfs\.exe
+	defrag.exe	.*	C:\\WINDOWS\\system32\\defrag\.exe
#
#7za
+	7za.exe	.*	C:\\program Files\\capture\\7za\.exe
#Line 25
#mapping
+	wmiadap.exe	.*	C:\\WINDOWS\\system32\\wbem\\wmiadap\.exe
+	wmiprvse.exe	.*	C:\\WINDOWS\\system32\\wbem\\wmiprvse\.exe
#
#vmware tools
+	VMwareUser.exe .*	C:\\Program Files\\VMware\\VMware Tools\\VMwareUser\.exe
#
#Windows 7 processes - beneign process for windows running services
+	svchost.exe	.*	C:\\Windows\\System32\\svchost\.exe
#DLLHOST manages dll based apps- care there is incertainty around 30% say it can be infectedcomment out if desired
+	dllhost.exe	.*	C:\\Windows\\System32\\dllhost\.exe
+	taskhost.exe	.*	C:\\Windows\\System32\\taskhost\.exe
#
#Services is commented out: it's a windows process but it's been shown to be prone to attacks
#+	services.exe	.*	C:\Windows\System32\services\.exe
#
#Windows 7 task scheduler - according to reviews ol seems harmless C:\Program Files\Google\Update\GoogleUpdate.exe
+	taskeng.exe	.*	C:\\Windows\\System32\\taskeng\.exe
#
#Windows 7 Search processes built in
+	SearchProtocolHost.exe	.*	C:\\Windows\\System32\\SearchProtocolHost\.exe
+	SearchFilterHost.exe	.*	C:\\Windows\\System32\\SearchFilterHost\.exe
+	SearchIndexer.exe	.*	C:\\Windows\\System32\\SearchIndexer\.exe
# Line 49
#User termi process
+	winlogon.exe	.*	C:\\Windows\\System32\\winlogon\.exe
+	userinit.exe	.*	C:\\Windows\\System32\\userinit\.exe
#csrss important win process
```

```
+	csrss.exe	.*	C:\\Windows\\System32\\csrss\.exe
#Conhost fixes some console bugs from win vista
+	conhost.exe	.*	C:\\Windows\\System32\\conhost\.exe
+		.*	C:\\Windows\\System32\\mobsync\.exe
####################################################
### Microsoft Internet Explorer 8.0		###
####################################################
+	iexplore.exe	.*	C:\\Program Files\\Internet Explorer\\iexplore.exe
+	IEXPLORE.EXE	.*	C:\\Program Files\\Internet Explorer\\IEXPLORE.EXE
#agent server is an activeX control that starts upon displaying multimedia content
+	agentsvr.exe	.*	C:\\WINDOWS\\msagent\\agentsvr.exe
#messenger activeX
+	msmsgs.exe	.*	C:\\Program Files\\Messenger\\msmsgs.exe
+	rundll32.exe	.*	C:\\WINDOWS\\system32\\rundll32.exe
#imapi
+	imapi.exe	.*	c:\\WINDOWS\\system32\\imapi\.exe

####################################################
### Firefox	v27			###
####################################################
+	firefox.exe	.*	C:\\Program Files\\Mozilla Firefox\\firefox.exe
####################################################
### Google update				###
####################################################
#Google Update processes
+	GoogleUpdate.exe	.*	C:\\Program Files\\Google\\Update\\GoogleUpdate\.exe
+	UNKNOWN	.*	C:\\Program Files\\Google\\Update\\GoogleUpdate\.exe
+	GoogleUpdate.exe	.*	C:\\Windows\\System32\\taskeng\.exe
```

Registry Exclusion list

```
#[+,-]     [Registry Event]   [Process Name]    [Registry Path]
######################################################
### Microsoft Windows XPSP2 Updated for win 7sp1###
######################################################
+          OpenKey            .*          .*
+          CreateKey          .*          .*
+          CloseKey           .*          .*
+          EnumerateKey       .*          .*
+          EnumerateValueKey          .*          .*
+          QueryValueKey      .*          .*
+          QueryKey           .*          .*
#issue in the way process path information is communicated to capture client
+          SetValueKey        UNKNOWN          .*
+          DeleteValueKey     UNKNOWN          .*
+          SetValueKey        .*          HKU\\.+\\SessionInformation\\ProgramCount
+          SetValueKey        .*          HKCU\\Software\\Microsoft\\Windows\\ShellNoRoam.*
+          SetValueKey        .*
           HKLM\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Installer\\UserData\\.+
+          SetValueKey        .*          HKLM\\SOFTWARE\\Microsoft\\Cryptography\\RNG\\Seed.*
+          SetValueKey        C:\\Program Files\\Capture\\CaptureClient.exe
           HKLM\\SYSTEM\\ControlSet001\\Services\\nm\\Parameters\\.+
+          SetValueKey        C:\\WINDOWS\\explorer.exe          HKCU\\SessionInformation\\.+
+          SetValueKey        C:\\WINDOWS\\explorer.exe
           HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Explorer\\.+
+          SetValueKey        C:\\WINDOWS\\explorer.exe
           HKLM\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Explorer\\.+
+          SetValueKey        C:\\WINDOWS\\explorer.exe
           HKU\\.+\\Software\\Microsoft\\Windows\\CurrentVersion\\Explorer\\.+
+          SetValueKey        C:\\WINDOWS\\explorer.exe
           HKLM\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Internet Settings\\Cache\\Paths\\.+
+          SetValueKey        C:\\WINDOWS\\explorer.exe
           HKU\\.+\\Software\\Microsoft\\Windows\\ShellNoRoam\\BagMRU\\.+
+          SetValueKey        C:\\WINDOWS\\system32\\winlogon.exe
           HKLM\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Group Policy\\State\\.+
+          SetValueKey        C:\\WINDOWS\\system32\\svchost.exe          \\REGISTRY\\USER\\.+
+          SetValueKey        C:\\WINDOWS\\system32\\svchost.exe          HKU\\.+
+          SetValueKey        C:\\WINDOWS\\system32\\svchost.exe
           HKCU\\Software\\Microsoft\\SystemCertificates\\Root\\.+
+          SetValueKey        C:\\WINDOWS\\system32\\svchost.exe
           HKLM\\SOFTWARE\\Microsoft\\EAPOL\\Parameters\\General\\InterfaceList
+          SetValueKey        C:\\WINDOWS\\system32\\svchost.exe
           HKLM\\SOFTWARE\\Microsoft\\SystemCertificates\\AuthRoot\\.+
+          SetValueKey        C:\\WINDOWS\\system32\\svchost.exe
           HKLM\\SOFTWARE\\Microsoft\\PCHealth\\.+
+          SetValueKey        C:\\WINDOWS\\system32\\svchost.exe
           HKLM\\SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\.+
+          SetValueKey        C:\\WINDOWS\\system32\\svchost.exe
           HKLM\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\.+
+          SetValueKey        C:\\WINDOWS\\system32\\svchost.exe
           HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\.+
+          SetValueKey        C:\\WINDOWS\\system32\\svchost.exe          HKLM\\SYSTEM\\ControlSet001\\.+
+          SetValueKey        C:\\WINDOWS\\system32\\services.exe         HKLM\\SYSTEM\\ControlSet001\\.+
+          SetValueKey        C:\\WINDOWS\\system32\\lsass.exe            HKLM\\SECURITY\\.+
+          SetValueKey        C:\\WINDOWS\\system32\\lsass.exe            HKLM\\SYSTEM\\ControlSet001\\.+
+           SetValueKey         C:\\WINDOWS\\system32\\lsass.exe            HKCU\\Software\\Microsoft\\Protected
Storage System Provider\\.+
+           SetValueKey         C:\\WINDOWS\\system32\\wbem\\wmiadap.exe
            HKLM\\SOFTWARE\\Microsoft\\WBEM\\.+
+           SetValueKey         C:\\WINDOWS\\system32\\wbem\\wmiadap.exe
            HKLM\\SYSTEM\\ControlSet001\\Services\\WmiApRpl\\Performance\\.+
+           SetValueKey         C:\\WINDOWS\\system32\\wbem\\wmiadap.exe
            HKLM\\SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Perflib\\.+
+           SetValueKey         C:\\WINDOWS\\system32\\wbem\\wmiprvse.exe
            HKLM\\SOFTWARE\\Microsoft\\WBEM\\WDM\\.+
+           DeleteValueKey   .*          HKU\\.+\\SessionInformation\\ProgramCount
+           DeleteValueKey   .*          HKCU\\Software\\Microsoft\\Windows\\ShellNoRoam.*
+           DeleteValueKey   .*
            HKLM\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Installer\\UserData\\.+
+           DeleteValueKey   .*          HKLM\\SOFTWARE\\Microsoft\\Cryptography\\RNG\\Seed.*
+           DeleteValueKey   C:\\Program Files\\Capture\\CaptureClient.exe
            HKLM\\SYSTEM\\ControlSet001\\Services\\nm\\Parameters\\.+

+           DeleteValueKey   C:\\WINDOWS\\explorer.exe          HKCU\\SessionInformation\\.+
+           DeleteValueKey   C:\\WINDOWS\\explorer.exe
            HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Explorer\\.+
+           DeleteValueKey   C:\\WINDOWS\\explorer.exe
            HKLM\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Explorer\\.+
+           DeleteValueKey   C:\\WINDOWS\\explorer.exe
            HKU\\.+\\Software\\Microsoft\\Windows\\CurrentVersion\\Explorer\\.+
+           DeleteValueKey   C:\\WINDOWS\\explorer.exe
            HKU\\.+\\Software\\Microsoft\\Windows\\ShellNoRoam\\BagMRU\\.+
+           DeleteValueKey   C:\\WINDOWS\\system32\\winlogon.exe
            HKLM\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Group Policy\\State\\.+
+           DeleteValueKey   C:\\WINDOWS\\system32\\svchost.exe       \\REGISTRY\\USER\\.+
+           DeleteValueKey   C:\\WINDOWS\\system32\\svchost.exe       HKU\\.+
+           DeleteValueKey   C:\\WINDOWS\\system32\\svchost.exe
            HKCU\\Software\\Microsoft\\SystemCertificates\\Root\\.+
+           DeleteValueKey   C:\\WINDOWS\\system32\\svchost.exe
            HKLM\\SOFTWARE\\Microsoft\\SystemCertificates\\AuthRoot\\.+
+           DeleteValueKey   C:\\WINDOWS\\system32\\svchost.exe
            HKLM\\SOFTWARE\\Microsoft\\PCHealth\\.+
+           DeleteValueKey   C:\\WINDOWS\\system32\\svchost.exe
            HKLM\\SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\.+
+           DeleteValueKey   C:\\WINDOWS\\system32\\svchost.exe
            HKLM\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\.+
+           DeleteValueKey   C:\\WINDOWS\\system32\\svchost.exe
            HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\.+
+           DeleteValueKey   C:\\WINDOWS\\system32\\svchost.exe
            HKCU\\Software\\Microsoft\\SystemCertificates\\Root\\.+
+           DeleteValueKey   C:\\WINDOWS\\system32\\svchost.exe       HKLM\\SYSTEM\\ControlSet001\\.+
+           DeleteValueKey   C:\\WINDOWS\\system32\\services.exe      HKLM\\SYSTEM\\ControlSet001\\.+
+           DeleteValueKey   C:\\WINDOWS\\system32\\lsass.exe           HKLM\\SECURITY\\.+
+           DeleteValueKey   C:\\WINDOWS\\system32\\lsass.exe           HKCU\\Software\\Microsoft\\Protected
Storage System Provider\\.+
+           DeleteValueKey   C:\\WINDOWS\\system32\\wbem\\wmiadap.exe
            HKLM\\SOFTWARE\\Microsoft\\WBEM\\.+
+           DeleteValueKey   C:\\WINDOWS\\system32\\wbem\\wmiadap.exe
            HKLM\\SYSTEM\\ControlSet001\\Services\\WmiApRpl\\Performance\\.+
+           DeleteValueKey   C:\\WINDOWS\\system32\\wbem\\wmiadap.exe
            HKLM\\SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Perflib\\.+
+	DeleteValueKey	C:\\WINDOWS\\system32\\wbem\\wmiprvse.exe
	HKLM\\SOFTWARE\\Microsoft\\WBEM\\WDM\\.+
#defrag
+	SetValueKey	C:\\WINDOWS\\system32\\dfrgntfs.exe	HKLM\\SOFTWARE\\Microsoft\\Dfrg.*
+	DeleteValueKey	C:\\WINDOWS\\system32\\dfrgntfs.exe	HKLM\\SOFTWARE\\Microsoft\\Dfrg.*
#windows update
+	SetValueKey	C:\\WINDOWS\\system32\\wuauclt.exe
	HKLM\\SYSTEM\\ControlSet001\\Services\\Eventlog\\Application\\ESENT\\.+
+	DeleteValueKey	C:\\WINDOWS\\system32\\wuauclt.exe
	HKLM\\SYSTEM\\ControlSet001\\Services\\Eventlog\\Application\\ESENT\\.+
####################################################
### Microsoft Windows 7 SP1 Updates			###
####################################################
#Win7 System specific reg changes
#Explorer
+	SetValueKey	C:\\Windows\\explorer.exe
	HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\HomeGroup\\UIStatusCache\\.+
+	SetValueKey	C:\\Windows\\explorer.exe	HKCR\\Local Settings\\MuiCache\\.+
+	DeleteValueKey	C:\\Windows\\explorer.exe
	HKLM\\SYSTEM\\ControlSet001\\services\\NlaSvc\\Parameters\\Internet\\ManualProxies
#Search Indexer
+	SetValueKey	C:\\Windows\\System32\\SearchIndexer.exe
	HKLM\\SOFTWARE\\Microsoft\\Windows Search\\Gather\\Windows\\SystemIndex\\.+
+	SetValueKey	C:\\Windows\\System32\\SearchIndexer.exe
	HKLM\\SOFTWARE\\Microsoft\\Windows Search\\.*
#lsass- Note seems "volatile" keep an eye on this please~
+	SetValueKey	C:\\WINDOWS\\system32\\lsass.exe
	HKU\\.DEFAULT\\Software\\Classes\\Local Settings\\MuiCache\\.+
+	SetValueKey	C:\\WINDOWS\\system32\\lsass.exe
	HKU\\.DEFAULT\\Software\\Classes\\Local Settings\\MuiCache\\.+\\.+\\LanguageList.*
+	SetValueKey	C:\\WINDOWS\\system32\\lsass.exe
	HKU\\.DEFAULT\\Software\\Classes\\Local Settings\\MuiCache\\.+\\.+\\.*
+	SetValueKey	C:\\WINDOWS\\system32\\lsass.exe
	HKU\\.DEFAULT\\Software\\Classes\\Local Settings\\MuiCache\\.*

+	SetValueKey	C:\\WINDOWS\\system32\\lsass.exe
	HKU\\.DEFAULT\\Software\\Classes\\Local Settings\\MuiCache\\.+\\.+\\LanguageList
#taskend
+	SetValueKey	C:\\Windows\\System32\\taskeng.exe
	HKLM\\SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Schedule\\Handshake\\.+
#SVCHOST
+	SetValueKey	C:\\WINDOWS\\system32\\svchost.exe
	HKLM\\SOFTWARE\\Microsoft\\WBEM\\Transports\\Decoupled\\Client\\.+\\.+
+	SetValueKey	C:\\WINDOWS\\system32\\svchost.exe
	HKLM\\SOFTWARE\\Microsoft\\WBEM\\Transports\\Decoupled\\Client\\.*
+	SetValueKey	C:\\WINDOWS\\system32\\svchost.exe	HKCU\\Software\\Microsoft\\Internet
Explorer\\LowRegistry\\Audio\\PolicyConfig\\PropertyStore\\.+\\.*
+	SetValueKey	C:\\WINDOWS\\system32\\svchost.exe	HKCU\\Software\\Microsoft\\Internet
Explorer\\LowRegistry\\Audio\\PolicyConfig\\PropertyStore\\ee00d86b_0\\.*
#SearchProtocolHost
+	SetValueKey	C:\\Windows\\System32\\SearchProtocolHost.exe
	HKU\\.DEFAULT\\Software\\Classes\Local Settings\\.*
#SVCHOST- contains a number of individual services that perform a range of functions such as win defender
services - can have multiple instances of this
+	SetValueKey	C:\\WINDOWS\\system32\\svchost.exe
	HKLM\\SOFTWARE\\Microsoft\\WBEM\CIMOM\\ConfigValueEssNeedsLoading
```

```
+	SetValueKey	C:\\WINDOWS\\system32\\svchost.exe
	HKLM\\SOFTWARE\\Microsoft\\WBEM\\CIMOM\\.+
+	SetValueKey	C:\\WINDOWS\\system32\\svchost.exe
	HKLM\\SOFTWARE\\Microsoft\\WBEM\\CIMOM\\.*
+	SetValueKey	System	HKLM\\SYSTEM\\ControlSet001\\Control\\Nsi\\.*
+	SetValueKey	System	HKLM\\SYSTEM\\ControlSet001\\Control\\Nsi\\.+
#System setting and deleting value keys- practically unknown but seems to be actively being picked up
+	SetValueKey	System
	HKLM\\SYSTEM\\ControlSet001\\Enum\\UMB\UMB\\1&841921d&0&PrinterBusEnumerator\\Capabi
lities
+	DeleteValueKey	System
	HKLM\\SYSTEM\\ControlSet001\\Enum\\UMB\\UMB\\1&841921d&0&PrinterBusEnumerator\\UINu
mber
+	SetValueKey	System
	HKLM\\SYSTEM\\ControlSet001\\Enum\\UMB\\UMB\\1&841921d&0&PrinterBusEnumerator\\.*
+	SetValueKey	System
	HKLM\\SYSTEM\\ControlSet001\\Enum\\UMB\\UMB\\1&841921d&0&PrinterBusEnumerator\\.+

+	DeleteValueKey	System
	HKLM\\SYSTEM\\ControlSet001\\Enum\\UMB\\UMB\\1&841921d&0&PrinterBusEnumerator\\.*
+	DeleteValueKey	System
	HKLM\\SYSTEM\\ControlSet001\\Enum\\UMB\\UMB\\1&841921d&0&PrinterBusEnumerator\\.+
+	SetValueKey	.*	HKLM\\SYSTEM\\.+\\Control\\DeviceClasses\\.+\\.+
+	DeleteValueKey	.*	HKLM\\SYSTEM\\.+\\Control\\DeviceClasses\\.+\\.+
#System logs?
+	SetValueKey	.*	HKLM\\SYSTEM\\ControlSet001\\Control\\WMI\\Autologger\\Circular
Kernel Context Logger\\Status
+	DeleteValueKey	.*	HKLM\\SYSTEM\\ControlSet001\\Control\\WMI\\Autologger\\Circular
Kernel Context Logger\\Status
#SpoolSV- Windows printing and fax services -usually docs sent to this service before reaching the printer.
+	SetValueKey	C:\\Windows\\System32\\spoolsv.exe
	HKLM\\SYSTEM\\ControlSet001\\Control\\Print\\Printers\\.+
+	SetValueKey	C:\\Windows\\System32\\spoolsv.exe
	HKLM\\SYSTEM\\ControlSet001\\Control\\Print\\.+
+	SetValueKey	C:\\Windows\\System32\\spoolsv.exe
	HKLM\\SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Print\\Printers\\.+
+	SetValueKey	C:\\Windows\\System32\\spoolsv.exe
	HKLM\\SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Ports\\.+
+	SetValueKey	C:\\Windows\\System32\\spoolsv.exe	HKU\\S-1-5-
19\\Software\\Microsoft\\Windows NT\\CurrentVersion\\Devices\\.+
+	SetValueKey	C:\\Windows\\System32\\spoolsv.exe	HKU\\S-1-5-
19\\Software\\Microsoft\\Windows NT\\CurrentVersion\\PrinterPorts\\.+
+	SetValueKey	C:\\Windows\\System32\\spoolsv.exe
	HKU\\.+\\Software\\Microsoft\\Windows NT\\CurrentVersion\\Devices\\.+
+	SetValueKey	C:\\Windows\\System32\\spoolsv.exe
	HKU\\.+\\Software\\Microsoft\\Windows NT\\CurrentVersion\\PrinterPorts\\.+
+	SetValueKey	C:\\Windows\\System32\\spoolsv.exe	HKU\\S-1-5-
19\\Software\\Microsoft\\Windows NT\\CurrentVersion\\Devices\\Fax
+	SetValueKey	C:\\Windows\\System32\\spoolsv.exe	HKU\\S-1-5-
19\\Software\\Microsoft\\Windows NT\\CurrentVersion\\PrinterPorts\\Fax
+	SetValueKey	C:\\Windows\\System32\\spoolsv.exe
	HKU\\.+\\Software\\Microsoft\\Windows NT\\CurrentVersion\\Devices\\Fax
+	SetValueKey	C:\\Windows\\System32\\spoolsv.exe
	HKU\\.+\\Software\\Microsoft\\Windows NT\\CurrentVersion\\PrinterPorts\\Fax
+	SetValueKey	C:\\Windows\\System32\\spoolsv.exe
	HKU\\.DEFAULT\\Software\\Microsoft\\Windows NT\\CurrentVersion\\Devices\\.+
+	SetValueKey	C:\\Windows\\System32\\spoolsv.exe
	HKU\\.DEFAULT\\Software\\Microsoft\\Windows NT\\CurrentVersion\\PrinterPorts\\.+
+	SetValueKey	C:\\Windows\\System32\\spoolsv.exe
	HKU\\.DEFAULT\\Software\\Microsoft\\Windows NT\\CurrentVersion\\Devices\\Fax
+	SetValueKey	C:\\Windows\\System32\\spoolsv.exe
	HKU\\.DEFAULT\\Software\\Microsoft\\Windows NT\\CurrentVersion\\PrinterPorts\\Fax
+	SetValueKey	C:\\Windows\\System32\\spoolsv.exe	HKCU\\Software\\Microsoft\\Windows
NT\\CurrentVersion\\Devices\\Microsoft XPS Document Writer
+	SetValueKey	C:\\Windows\\System32\\spoolsv.exe	HKCU\\Software\\Microsoft\\Windows
NT\\CurrentVersion\\PrinterPorts\\Microsoft XPS Document Writer
+	SetValueKey	C:\\Windows\\System32\\spoolsv.exe	HKCU\\Software\\Microsoft\\Windows
NT\\CurrentVersion\\Devices\\Fax
+	SetValueKey	C:\\Windows\\System32\\spoolsv.exe	HKCU\\Software\\Microsoft\\Windows
NT\\CurrentVersion\\PrinterPorts\\Fax
#####explorer
+	SetValueKey	C:\\Windows\\explorer\.exe
	HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Action Center\\Checks\\.+\\CheckSetting
+	SetValueKey	C:\\Windows\\explorer\.exe
	HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Action Center\\Checks\\.+\\CheckSetting
+	SetValueKey	C:\\Windows\\explorer\.exe
	HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Action Center\\Checks\\.+\\CheckSetting
+	SetValueKey	C:\\Windows\\explorer\.exe
	HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Action Center\\Checks\\.+\\CheckSetting
################################################################
##########Very volatile can change need a fix perm ###########
################################################################
+	SetValueKey	.*
	HKLM\\SYSTEM\\ControlSet001\\Enum\\UMB\\UMB\\1&841921d&0&PrinterBusEnumerator\\Capa
bilities
+	DeleteValueKey	.*
	HKLM\\SYSTEM\\ControlSet001\\Enum\\UMB\\UMB\\1&841921d&0&PrinterBusEnumerator\\UINu
mber
+	DeleteValueKey	.*
	HKLM\\SYSTEM\\ControlSet001\\Enum\\UMB\\UMB\\1&841921d&0&PrinterBusEnumerator\\LogC
onf\BasicConfigVector
+	SetValueKey	.*
	HKLM\\SYSTEM\\ControlSet001\\Enum\\UMB\\UMB\\1&841921d&0&PrinterBusEnumerator\\Hard
wareID
+	SetValueKey	.*
	HKLM\\SYSTEM\\ControlSet001\\Enum\\UMB\\UMB\\1&841921d&0&PrinterBusEnumerator\\Comp
atibleIDs
+	SetValueKey	.*
	HKLM\\SYSTEM\\ControlSet001\\Enum\\UMB\\UMB\\1&841921d&0&PrinterBusEnumerator\\Conta
inerID
+	SetValueKey	.*
	HKLM\\SYSTEM\\ControlSet001\\Enum\\UMB\\UMB\\1&841921d&0&PrinterBusEnumerator\\Devic
e Parameters\\NodeID
+	SetValueKey	.*
	HKLM\\SYSTEM\\ControlSet001\\Enum\\UMB\\UMB\\1&841921d&0&PrinterBusEnumerator\\Devic
e Parameters\\Identity
+	DeleteValueKey	.*
	HKLM\\SYSTEM\\ControlSet001\\Enum\\UMB\\UMB\\1&841921d&0&PrinterBusEnumerator\\LogC
onf\\BootConfig
+	SetValueKey	.*
	HKLM\\SYSTEM\\ControlSet001\\Enum\\UMB\\UMB\\1&841921d&0&PrinterBusEnumerator\\Capa
bilities
```

```
+	DeleteValueKey	.*
	HKLM\\SYSTEM\\ControlSet001\\Enum\\UMB\\UMB\\1&841921d&0&PrinterBusEnumerator\\UINu
mber
+	SetValueKey	.*	HKLM\\SYSTEM\\ControlSet001\\services\\umbus\\Enum\\1
+	SetValueKey	.*	HKLM\\SYSTEM\\ControlSet001\\services\\umbus\\Enum\\Count
+	SetValueKey	.*	HKLM\\SYSTEM\\ControlSet001\\services\\umbus\\Enum\\NextInstance
+	SetValueKey	.*
	HKLM\\SYSTEM\\ControlSet001\\Enum\\UMB\\UMB\\1&841921d&0&PrinterBusEnumerator\\Capa
bilities
+	DeleteValueKey	.*
	HKLM\\SYSTEM\\ControlSet001\\Enum\\UMB\\UMB\\1&841921d&0&PrinterBusEnumerator\\UINu
mber
####################################################
### Internet Explorer 6.0 SP2				###
####################################################
+	OpenKey		.*	.*
+	CreateKey	.*	.*
+	CloseKey	.*	.*
+	EnumerateKey	.*	.*
+	EnumerateValueKey	.*	.*

+	QueryValueKey	.*	.*
+	QueryKey	.*	.*
+	SetValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe   HKCU\\EUDC\\.+
+	SetValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	HKCU\\Software\\Microsoft\\Internet Explorer\\Main\\Window_Placement
+	SetValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	HKCU\\Software\\Microsoft\\Internet Explorer\\Main\\Fullscreen
+	SetValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	HKCU\\Software\\Microsoft\\Internet Explorer\\Main\\NotificationDownloadComplete
+	SetValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe
+	SetValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	HKCU\\Software\\Microsoft\\Internet Explorer\\Toolbar\\Locked
+	SetValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	HKCU\\Software\\Microsoft\\Internet Explorer\\International\\.+
+	SetValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	HKCU\\Software\\Microsoft\\Internet Explorer\\Security\\P3Global\\Enabled
+	SetValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	HKCU\\Software\\Microsoft\\Internet Explorer\\Extensions\\CmdMapping\\.+
+	SetValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	HKCU\\Software\\Microsoft\\Internet Explorer\\PageSetup\\.+
+	SetValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Explorer\\MenuOrder\\.+
+	SetValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Explorer\\MountPoints2\\.+
+	SetValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Explorer\\UserAssist\\.+
+	SetValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Explorer\\Shell Folders\\.+
+	SetValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Explorer\\CabinetState\\.+
+	SetValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Internet
Settings\\ZoneMap\\UNCAsIntranet
+	SetValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	HKCU\\Software\\Microsoft\\Internet Explorer\\Toolbar\\WebBrowser\\.+
+	SetValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Internet
Settings\\ZoneMap\\IntranetName
+	SetValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Internet Settings\\ZoneMap\\AutoDetect
+	SetValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Internet Settings\\ZoneMap\\ProxyBypass
+	SetValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Internet
Settings\\ZoneMap\\UNCAsIntranet
+	SetValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Internet Settings\\MigrateProxy
+	SetValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Internet Settings\\ProxyEnable
+	SetValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Internet Settings\\ProxyServer
+	SetValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Internet Settings\\Cache.+
+	SetValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Internet Settings\\Connections\\.+
+	SetValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Internet Settings\\5.0\\Cache\\.+
+	SetValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Ext\\Stats\\.+
+	SetValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	HKCU\\Software\\Microsoft\\Windows\\ShellNoRoam\\BagMRU.+
+	SetValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	HKCU\\Software\\Microsoft\\Windows\\ShellNoRoam\\Bags.+
+	SetValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Ext\\Stats\\.+\\iexplore\\(Count|Time|Typ
e)
+	SetValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	HKCU\\Software\\Microsoft\\Windows NT\\CurrentVersion\\Winlogon\\ParseAutoexec
+	SetValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Internet Settings\\.+
+	SetValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	HKCU\\Software\\Microsoft\\SystemCertificates\\.+
+	SetValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	HKLM\\SOFTWARE\\Classes\\.+
+	SetValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	HKLM\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Internet Settings\\Cache.+
+	SetValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	HKLM\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Internet Settings\\5.0\\Cache.+
+	SetValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	HKLM\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Internet Settings\\Cache.+
+	SetValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	HKLM\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Internet Settings\\Passport\\.+
+	SetValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	HKLM\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Explorer\\Shell Folders\\.+
+	SetValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	HKLM\\SOFTWARE\\Microsoft\\Direct3D.+
+	SetValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	HKLM\\SOFTWARE\\Microsoft\\DirectDraw.+
+	SetValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	HKLM\\SOFTWARE\\Microsoft\\Cryptography\\RNG\\Seed
+	SetValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	HKLM\\SOFTWARE\\Microsoft\\AudioCompressionManager\\.+
+           SetValueKey       C:\\Program Files\\Internet Explorer\\iexplore\.exe
            HKLM\\SOFTWARE\\Microsoft\\SystemCertificates\\.+
+           SetValueKey       C:\\Program Files\\Internet Explorer\\iexplore\.exe
            HKLM\\Software\\Microsoft\\Windows\\CurrentVersion\\Internet Settings\\Cache.+
+           SetValueKey       C:\\Program Files\\Internet Explorer\\iexplore\.exe
            HKLM\\Software\\Microsoft\\Windows\\CurrentVersion\\Internet Settings\\5.0\\Cache.
+           SetValueKey       C:\\Program Files\\Internet Explorer\\iexplore\.exe
            HKLM\\SYSTEM\\ControlSet001\\Hardware
Profiles\\0001\\Software\\Microsoft\\windows\\CurrentVersion\\Internet Settings\\ProxyEnable
+           SetValueKey       C:\\Program Files\\Internet Explorer\\iexplore\.exe
            HKLM\\SYSTEM\\ControlSet001\\Services\\EventLog\\.+
+           DeleteValueKey    C:\\Program Files\\Internet Explorer\\iexplore\.exe   HKCU\\EUDC\\.+
+           DeleteValueKey    C:\\Program Files\\Internet Explorer\\iexplore\.exe
            HKCU\\Software\\Microsoft\\Internet Explorer\\Main\\Window_Placement
+           DeleteValueKey    C:\\Program Files\\Internet Explorer\\iexplore\.exe
            HKCU\\Software\\Microsoft\\Internet Explorer\\Main\\Fullscreen
+           DeleteValueKey    C:\\Program Files\\Internet Explorer\\iexplore\.exe
            HKCU\\Software\\Microsoft\\Internet Explorer\\Main\\NotificationDownloadComplete
+           DeleteValueKey    C:\\Program Files\\Internet Explorer\\iexplore\.exe
            HKCU\\Software\\Microsoft\\Internet Explorer\\TypedURLs
+           DeleteValueKey    C:\\Program Files\\Internet Explorer\\iexplore\.exe
            HKCU\\Software\\Microsoft\\Internet Explorer\\Toolbar\\Locked
+           DeleteValueKey    C:\\Program Files\\Internet Explorer\\iexplore\.exe
            HKCU\\Software\\Microsoft\\Internet Explorer\\International\\.+
+           DeleteValueKey    C:\\Program Files\\Internet Explorer\\iexplore\.exe
            HKCU\\Software\\Microsoft\\Internet Explorer\\Security\\P3Global\\Enabled
+           DeleteValueKey    C:\\Program Files\\Internet Explorer\\iexplore\.exe
            HKCU\\Software\\Microsoft\\Internet Explorer\\Extensions\\CmdMapping\\.+
+           DeleteValueKey    C:\\Program Files\\Internet Explorer\\iexplore\.exe
            HKCU\\Software\\Microsoft\\Internet Explorer\\PageSetup\\.+
+           DeleteValueKey    C:\\Program Files\\Internet Explorer\\iexplore\.exe
            HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Explorer\\MenuOrder\\.+
+           DeleteValueKey    C:\\Program Files\\Internet Explorer\\iexplore\.exe
            HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Explorer\\MountPoints2\\.+
+           DeleteValueKey    C:\\Program Files\\Internet Explorer\\iexplore\.exe
            HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Explorer\\UserAssist\\.+
+           DeleteValueKey    C:\\Program Files\\Internet Explorer\\iexplore\.exe
            HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Explorer\\Shell Folders\\.+
+           DeleteValueKey    C:\\Program Files\\Internet Explorer\\iexplore\.exe
            HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Explorer\\CabinetState\\.+
+           DeleteValueKey    C:\\Program Files\\Internet Explorer\\iexplore\.exe
            HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Internet
Settings\\ZoneMap\\UNCAsIntranet
+           DeleteValueKey    C:\\Program Files\\Internet Explorer\\iexplore\.exe
            HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Internet
Settings\\ZoneMap\\IntranetName
+           DeleteValueKey    C:\\Program Files\\Internet Explorer\\iexplore\.exe
            HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Internet Settings\\ZoneMap\\AutoDetect
+           DeleteValueKey    C:\\Program Files\\Internet Explorer\\iexplore\.exe
            HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Internet Settings\\ZoneMap\\ProxyBypass
+           DeleteValueKey    C:\\Program Files\\Internet Explorer\\iexplore\.exe
            HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Internet
Settings\\ZoneMap\\UNCAsIntranet
+           DeleteValueKey    C:\\Program Files\\Internet Explorer\\iexplore\.exe
            HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Internet Settings\\MigrateProxy
+           DeleteValueKey    C:\\Program Files\\Internet Explorer\\iexplore\.exe
            HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Internet Settings\\ProxyEnable
```

```
+           DeleteValueKey   C:\\Program Files\\Internet Explorer\\iexplore\.exe
            HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Internet Settings\\ProxyServer
+           DeleteValueKey   C:\\Program Files\\Internet Explorer\\iexplore\.exe
            HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Internet Settings\\Cache.+

+           DeleteValueKey   C:\\Program Files\\Internet Explorer\\iexplore\.exe
            HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Internet Settings\\Connections\\.+
+           DeleteValueKey   C:\\Program Files\\Internet Explorer\\iexplore\.exe
            HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Internet Settings\\5.0\\Cache\\.+
+           DeleteValueKey   C:\\Program Files\\Internet Explorer\\iexplore\.exe
            HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Ext\\Stats\\.+
+           DeleteValueKey   C:\\Program Files\\Internet Explorer\\iexplore\.exe
            HKCU\\Software\\Microsoft\\Windows\\ShellNoRoam\\BagMRU.+
+           DeleteValueKey   C:\\Program Files\\Internet Explorer\\iexplore\.exe
            HKCU\\Software\\Microsoft\\Windows\\ShellNoRoam\\Bags.+
+           DeleteValueKey   C:\\Program Files\\Internet Explorer\\iexplore\.exe
            HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Ext\\Stats\\.+\\iexplore\\(Count|Time|Typ
e)
+           DeleteValueKey   C:\\Program Files\\Internet Explorer\\iexplore\.exe
            HKCU\\Software\\Microsoft\\Windows NT\\CurrentVersion\\Winlogon\\ParseAutoexec
+           DeleteValueKey   C:\\Program Files\\Internet Explorer\\iexplore\.exe
            HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Internet Settings\\.+
+           DeleteValueKey   C:\\Program Files\\Internet Explorer\\iexplore\.exe
            HKCU\\Software\\Microsoft\\SystemCertificates\\.+
+           DeleteValueKey   C:\\Program Files\\Internet Explorer\\iexplore\.exe
            HKLM\\SOFTWARE\\Classes\\.+
+           DeleteValueKey   C:\\Program Files\\Internet Explorer\\iexplore\.exe
            HKLM\\Software\\Microsoft\\Windows\\CurrentVersion\\Internet Settings\\Cache.+
+           DeleteValueKey   C:\\Program Files\\Internet Explorer\\iexplore\.exe
            HKLM\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Internet Settings\\5.0\\Cache.+
+           DeleteValueKey   C:\\Program Files\\Internet Explorer\\iexplore\.exe
            HKLM\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Internet Settings\\Cache.+
+           DeleteValueKey   C:\\Program Files\\Internet Explorer\\iexplore\.exe
            HKLM\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Internet Settings\\Passport\\.+
+           DeleteValueKey   C:\\Program Files\\Internet Explorer\\iexplore\.exe
            HKLM\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Explorer\\Shell Folders\\.+
+           DeleteValueKey   C:\\Program Files\\Internet Explorer\\iexplore\.exe
            HKLM\\SOFTWARE\\Microsoft\\Direct3D.+
+           DeleteValueKey   C:\\Program Files\\Internet Explorer\\iexplore\.exe
            HKLM\\SOFTWARE\\Microsoft\\DirectDraw.+
+           DeleteValueKey   C:\\Program Files\\Internet Explorer\\iexplore\.exe
            HKLM\\SOFTWARE\\Microsoft\\Cryptography\\RNG\\Seed
+           DeleteValueKey   C:\\Program Files\\Internet Explorer\\iexplore\.exe
            HKLM\\SOFTWARE\\Microsoft\\AudioCompressionManager\\.+
+           DeleteValueKey   C:\\Program Files\\Internet Explorer\\iexplore\.exe
            HKLM\\SOFTWARE\\Microsoft\\SystemCertificates\\.+
+           DeleteValueKey   C:\\Program Files\\Internet Explorer\\iexplore\.exe
            HKLM\\Software\\Microsoft\\Windows\\CurrentVersion\\Internet Settings\\Cache.+
+           DeleteValueKey   C:\\Program Files\\Internet Explorer\\iexplore\.exe
            HKLM\\Software\\Microsoft\\Windows\\CurrentVersion\\Internet Settings\\5.0\\Cache.+
+           DeleteValueKey   C:\\Program Files\\Internet Explorer\\iexplore\.exe
            HKLM\\SYSTEM\\ControlSet001\\Hardware
Profiles\\0001\\Software\\Microsoft\\windows\\CurrentVersion\\Internet Settings\\ProxyEnable
+           DeleteValueKey   C:\\Program Files\\Internet Explorer\\iexplore\.exe
            HKLM\\SYSTEM\\ControlSet001\\Services\\EventLog\\.+
+           DeleteKey        .*          .*
#Plugins
```

```
+	SetValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	HKCU\\Software\\Microsoft\\Scrunch\\.+
+	SetValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	HKCU\\Software\\Microsoft\\MediaPlayer\\.+
+	SetValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	HKCU\\Software\\Microsoft\\Windows Media\\.+
+	SetValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	HKCU\\Software\\Microsoft\\Multimedia\\ActiveMovie\\.+
+	SetValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	HKCU\\Software\\Microsoft\\ActiveMovie\\.+
+	SetValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	HKCU\\Software\\Microsoft\\MPEG2Demultiplexer\\.+
+	SetValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	HKCU\\Software\\Microsoft\\Multimedia\\msacm.imaadpcm\\.+
+	SetValueKey	C:\\WINDOWS\\msagent\\agentsvr\.exe
	HKLM\\SOFTWARE\\Microsoft\\AudioCompressionManager\\DriverCache\\msacm.msadpcm\\.+
+	SetValueKey	C:\\WINDOWS\\msagent\\agentsvr\.exe
	HKLM\\SOFTWARE\\Microsoft\\Microsoft Agent\\.+
+	SetValueKey	C:\\Program Files\\Messenger\\msmsgs\.exe
	HKCU\\AppEvents\\Schemes\\Apps\\MSMSGS.*
+	SetValueKey	C:\\Program Files\\Messenger\\msmsgs\.exe
	HKCU\\AppEvents\\EventLabels\\MSMsgs.+

+	SetValueKey	C:\\Program Files\\Messenger\\msmsgs\.exe
	HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Run\\MSMSGS
+	SetValueKey	C:\\Program Files\\Messenger\\msmsgs\.exe
	HKLM\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Internet Settings\\.+
+	SetValueKey	C:\\Program Files\\Messenger\\msmsgs\.exe
	HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Explorer\\Shell Folders\\.+
+	SetValueKey	C:\\Program Files\\Messenger\\msmsgs\.exe
	HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Internet Settings\\.+
+	SetValueKey	C:\\WINDOWS\\system32\\svchost\.exe
	HKLM\\SOFTWARE\\Microsoft\\EventSystem\\.+\\Subscriptions\\.+
+	DeleteValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	HKCU\\Software\\Microsoft\\Scrunch\\.+
+	DeleteValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	HKCU\\Software\\Microsoft\\MediaPlayer\\.+
+	DeleteValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	HKCU\\Software\\Microsoft\\Windows Media\\.+
+	DeleteValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	HKCU\\Software\\Microsoft\\Multimedia\\ActiveMovie\\.+
+	DeleteValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	HKCU\\Software\\Microsoft\\ActiveMovie\\.+
+	DeleteValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	HKCU\\Software\\Microsoft\\MPEG2Demultiplexer\\.+
+	DeleteValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	HKCU\\Software\\Microsoft\\Multimedia\\msacm.imaadpcm\\.+
+	DeleteValueKey	C:\\WINDOWS\\msagent\\agentsvr\.exe
	HKLM\\SOFTWARE\\Microsoft\\AudioCompressionManager\\DriverCache\\msacm.msadpcm\\.+
+	DeleteValueKey	C:\\WINDOWS\\msagent\\agentsvr\.exe
	HKLM\\SOFTWARE\\Microsoft\\Microsoft Agent\\.
+	DeleteValueKey	C:\\Program Files\\Messenger\\msmsgs\.exe
	HKCU\\AppEvents\\EventLabels\\MSMsgs.+
+	DeleteValueKey	C:\\Program Files\\Messenger\\msmsgs\.exe
	HKCU\\AppEvents\\Schemes\\Apps\\MSMSGS.*
+	DeleteValueKey	C:\\Program Files\\Messenger\\msmsgs\.exe
	HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Run\\MSMSGS
```

```
+	DeleteValueKey	C:\\Program Files\\Messenger\\msmsgs\.exe
	HKLM\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Internet Settings\\.+
+	DeleteValueKey	C:\\Program Files\\Messenger\\msmsgs\.exe
	HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Explorer\\Shell Folders\\.+
+	DeleteValueKey	C:\\Program Files\\Messenger\\msmsgs\.exe
	HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Internet Settings\\.+
+	DeleteValueKey	C:\\WINDOWS\\system32\\svchost\.exe
	HKLM\\SOFTWARE\\Microsoft\\EventSystem\\.+\\Subscriptions\\.+
####################################################
### Internet Explorer 8.0 SP1                    ###
####################################################
#May need more investigation - Current research shows a bit of uncertainty regarding what these actually do
+	SetValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	HKLM\\SOFTWARE\\Microsoft\\Rpc\\UuidSequenceNumber
+	SetValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	HKCU\\Software\\Microsoft\\Internet Explorer\\Main\\CompatibilityFlags
+	SetValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	HKCU\\Software\\Microsoft\\Internet Explorer\\Recovery\\AdminActive\\.+

+	DeleteValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.ex
	HKLM\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Internet
Settings\\ZoneMap\\ProxyBypass
+	DeleteValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	HKLM\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Internet
Settings\\ZoneMap\\IntranetName
+	DeleteValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	HKLM\\SYSTEM\\ControlSet001\\services\\NlaSvc\\Parameters\\Internet\\ManualProxies
+	SetValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	HKCU\\Software\\Microsoft\\Internet Explorer\\Main\\WindowsSearch\\Version
+	SetValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	HKCU\\Software\\Microsoft\\Direct3D\\MostRecentApplication\\Name
#Language list
+	SetValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe	HKCR\\Local
Settings\\MuiCache\\A\\52C64B7E\\LanguageList
+	SetValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe	HKCR\\Local
Settings\\MuiCache\\.+\\.+\\LanguageList
+	SetValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe	HKCR\\Local
Settings\\MuiCache\\.*
#DOMSTORAGE
+	SetValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	HKCU\\Software\\Microsoft\\Internet Explorer\\DOMStorage\\Total

+	SetValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	HKCU\\Software\\Microsoft\\Internet Explorer\\DOMStorage\\.+
#Windows search
+	SetValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	HKCU\\Software\\Microsoft\\Internet Explorer\\Main\\WindowsSearch\\UpgradeTime
#Linksbar (recently searched/typed addresses in iE)
+	SetValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	HKCU\\Software\\Microsoft\\Internet Explorer\\LinksBar\\ItemCache\\.+
+	DeleteValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	HKCU\\Software\\Microsoft\\Internet Explorer\\LinksBar\\ItemCache\\.+
#flashplayer
+	SetValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	HKCU\\Software\\Macromedia\\FlashPlayer\\FlashPlayerVersion
+	DeleteValueKey	C:\\Program Files\\Internet Explorer\\iexplore\.exe
	HKCU\\Software\\Macromedia\\FlashPlayer\\FlashPlayerVersion
```

```
#Mob Sync (windows process to keep compornments updated
+         SetValueKey        C:\\Windows\\System32\\mobsync\.exe      HKCR\\Local
Settings\\Software\\Microsoft\\Windows\\CurrentVersion\\SyncMgr\\HandlerInstances\\.+
+         SetValueKey        C:\\Windows\\System32\\mobsync\.exe      HKCR\\Local
Settings\\Software\\Microsoft\\Windows\\CurrentVersion\\SyncMgr\\HandlerInstances\\.+\\.+
+         SetValueKey        C:\\Windows\\System32\\mobsync\.exe      HKCR\\Local
Settings\\Software\\Microsoft\\Windows\\CurrentVersion\\SyncMgr\\HandlerInstances\\.+\\SyncTime
+         SetValueKey        C:\\Windows\\System32\\mobsync\.exe      HKCR\\Local
Settings\\Software\\Microsoft\\Windows\\CurrentVersion\\SyncMgr\\StartAtLogin
+         SetValueKey        C:\\Windows\\System32\\mobsync\.exe      HKCR\\Local
Settings\\Software\\Microsoft\\Windows\\CurrentVersion\\SyncMgr\\HandlerInstances\\.+\\Enabled
+         SetValueKey        C:\\Windows\\System32\\mobsync\.exe      HKCR\\Local
Settings\\Software\\Microsoft\\Windows\\CurrentVersion\\SyncMgr\\HandlerInstances\\.+\\connected
##################################################
### Firefox v26                                   ###
##################################################
#Firefox browser
+         SetValueKey        C:\\Program Files\\Mozilla Firefox\\firefox\.exe
          HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Internet Settings\\ProxyEnable
+         SetValueKey        C:\\Program Files\\Mozilla Firefox\\firefox\.exe
          HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Internet
Settings\\Connections\\SavedLegacySettings
+         DeleteValueKey   C:\\Program Files\\Mozilla Firefox\\firefox\.exe
          HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Internet Settings\\ProxyServer
+         DeleteValueKey   C:\\Program Files\\Mozilla Firefox\\firefox\.exe
          HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Internet Settings\\ProxyOverride
+         DeleteValueKey   C:\\Program Files\\Mozilla Firefox\\firefox\.exe
          HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Internet Settings\\AutoConfigURL
##################################################### UNKNOWN"?
+         SetValueKey        .*          HKLM\\SOFTWARE\\Microsoft\\Windows
NT\\CurrentVersion\\Schedule\\Handshake\\.+
##################################################
### Minus List - General Malicious Activity   ###
##################################################
#Any modification to start/bootup sequence
-         SetValueKey        .*          HLKM\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run.*
-         DeleteValueKey   .*          HLKM\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run.*
-         SetValueKey        .*          HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Run.*
-         DeleteValueKey   .*          HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Run.*
-         SetValueKey        .*          HKCU\\Software\\Microsoft\\Windows
NT\\CurrentVersion\\Windows\\Run.*
-         DeleteValueKey   .*          HKCU\\Software\\Microsoft\\Windows
NT\\CurrentVersion\\Windows\\Run.*
-         SetValueKey        .*          HKCU\\Software\\Microsoft\\Windows
NT\\CurrentVersion\\Windows\\Load.*
-         DeleteValueKey   .*          HKCU\\Software\\Microsoft\\Windows
NT\\CurrentVersion\\Windows\\Load.*
-         SetValueKey        .*          HLKM\\SOFTWARE\\Microsoft\\Windows
NT\\CurrentVersion\\Winlogon\\Userinit.*
-         DeleteValueKey   .*          HLKM\\SOFTWARE\\Microsoft\\Windows
NT\\CurrentVersion\\Winlogon\\Userinit.*
-         SetValueKey        .*          HLKM\\SOFTWARE\\Microsoft\\Windows
NT\\CurrentVersion\\Winlogon\\Shell.*
-         DeleteValueKey   .*          HLKM\\SOFTWARE\\Microsoft\\Windows
NT\\CurrentVersion\\Winlogon\\Shell.*
-         SetValueKey        .*
          HLKM\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Policies\\Explorer\\Run.*
```

- DeleteValueKey .* HLKM\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Policies\\Explorer\\Run.*
- SetValueKey .* HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Policies\\Explorer\\Run.*
- DeleteValueKey .* HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Policies\\Explorer\\Run.*
- SetValueKey .* HLKM\\SYSTEM\\CurrentControlSet\\Control\\Session Manager\\BootExecute.*
- DeleteValueKey .* HLKM\\SYSTEM\\CurrentControlSet\\Control\\Session Manager\\BootExecute.*
- SetValueKey .* HLKM\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\ShellServiceObjectDelayLoad\\.*
- DeleteValueKey .*