Information Processing and Management xxx (xxxx) xxx–xxx

Contents lists available at ScienceDirect

Information Processing and Management

journal homepage: www.elsevier.com/locate/infoproman

Prediction of drive-by download attacks on Twitter

Amir Javed*, Pete Burnap, Omer Rana

School of Computer Science and Informatics, Cardiff University, CF243AA, United Kingdom

ARTICLE INFO

Keywords:
Cyber security
Drive-by download
Malware
Machine learning
Web security

ABSTRACT

The popularity of Twitter for information discovery, coupled with the automatic shortening of URLs to save space, given the 140 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 to click 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*. 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 this paper we build a machine learning model using machine activity data and tweet metadata to move beyond post-execution classification of such URLs as malicious, to predict 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 s 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.

1. Introduction

Online social networks (OSNs) have emerged as powerful tools for disseminating information. Among these, Twitter, a micro-blogging website that allows its users to express themselves in 140 characters, has emerged as a go-to source for current affairs, entertainment news and to seek information about global events in real-time. For example, Twitter has been used to study public reaction to events such as natural disasters (Sakaki, Okazaki, & Matsuo, 2010), political elections (Tumasjan, Sprenger, Sandner, & Welpe, 2010) and terrorist attacks (Burnap et al., 2014). 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 (Rogers, 2016). This high volume of users around a popular trending event and Twitter's inbuilt feature of shortening a URL due to its 140 character restriction provides cybercriminals with an opportunity to obfuscate links to malicious Web pages within *tweets* and carry out a drive-by download attack. In a *drive-by download* (Cova, Kruegel, & Vigna, 2010; Moshchuk, Bragin, Gribble, & Levy, 2006) an attacker attempts to lure users to malicious Web pages so that they can hijack the user's system by exploiting a system vulnerability. By successfully carrying out these attacks an attacker is able to, for example, obtain remote access, steal user information, or make the computer part of a botnet (Provos, McNamee, Mavrommatis, Wang, & Modadugu, 2007).

The more popular OSNs become, the more attractive a platform they become for cybercriminals to conduct their attacks (ZeroFox, 2017). Microsoft acknowledged this fast growing threat of malicious Web pages as one the top threats in their security and intelligence report published in 2013 (Microsoft, 2013) and the detection of drive-by download attacks remains an important topic of

* Corresponding author.
E-mail address: javeda7@cardiff.ac.uk (A. Javed).

https://doi.org/10.1016/j.ipm.2018.02.003
Received 8 August 2017; Received in revised form 6 February 2018; Accepted 12 February 2018

research. The problem of detecting these drive-by download attacks on Twitter has been broadly investigated from a number of perspectives including: (i) characteristics of OSN user accounts (e.g. posting behaviours (Cao & Caverlee, 2015) and social network links (Yang, Harkreader, & Gu, 2011)); (ii) characteristics of URLs (e.g. lexical features (Ma, Saul, Savage, & Voelker, 2009) and endpoint activity (Lee & Kim, 2013; Lee & Stokes, 2011)); 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.

In our earlier work we recorded system-level machine activity for five minutes to capture behavioural interactions with Web servers (Burnap, Javed, Rana, & Awan, 2015). This was used to build a machine classifier that was able to distinguish between malicious and benign URLs with an *F*-measure of 0.72 when we tested the model on an unseen dataset. The main contribution in our previous work was to build a machine classifier to classify a URL at the end of a 5 minute interaction.

In this paper we extend our previous work by adding more behavioural features to improve classifier performance and reducing the classification period to 10 s *to predict a drive-by download attack based on early-stage machine activities observed before the attack is complete*. By capturing machine activity metrics (e.g. CPU use, RAM use, Network I/O for full list see Appendix A) and tweet attributes, we are now 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 s 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.

2. Related work

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 (Robinson, 2015).

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 (Berkowitz, 2015) and former Apple Macintosh evangelist Guy Kawasaki (McMillan, 2009). 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 (SANS Institue, 2017). Such cyber attacks could also be used as an entry point to carry out more wide-spreading attacks such as Ransomware for instance, a Crypto Locker attack that originated from a drive-by download attack locked down a small city in Washington, USA for four days (Kumar, 2017).

In this section, we discuss the related work on the topic of detecting malicious content in Online Social Networks (OSN). This is presented in two sub-sections - we first look at detecting such content using OSN user account and URL characteristics, and then study the use of static and dynamic code analysis. Using tweet meta-data Kristina and Rumi followed various top stories and used various tweet attributes to demonstrate how rapidly information (e.g. malicious URLs) can be disseminated in Twitter (Lerman & Ghosh, 2010), making it the core focus for existing work in this area - so the majority of the related work focuses on Twitter and tweet meta-data. It should be noted though that malicious URLs and spam are a significant issue on all OSNs. Twitter is very active for breaking-news and real-world events, hence it provides an environment that is particularly attractive to cybercriminals - but all OSNs include the sharing of hyperlinks so are susceptible to these issues. Table 1 provides a summary of related work and the methods used at a high level for comparison.

2.1. Detecting malicious content based on OSN account and URL characteristics

Previous research has aimed to identify tweets that are classified as spam or contain a URL pointing to a malicious Web server based on tweet meta-data. The rationale being that it is possible to differentiate between a 'normal' user and that of a cybercriminal based on user account characteristics extracted from meta-data such as the number of followers, number of people they follow, their posting behaviour etc. This research identified tweet attributes can be used to detect accounts that exhibit abnormal behaviour (e.g. posting spam or malicious URLs). Cao and Caverlee analysed the behaviour of Twitter users to detect tweets classified as spam, using meta-data from the user account posting the spam or URL, as well as the user account clicking the URL (Cao & Caverlee, 2015). Their hypothesis was based on the assumption that it is difficult to manipulate such behavioural signals. Chen, Zhang, Xiang, Zhou, and Oliver (2016) used a Finite State Machine based spam template, demonstrating that a cybercriminal can create 2000 tweets from a single template. They discovered that such users were using multiple accounts to post spam in a coordinated manner to avoid detection.

They were exhibiting "load balancing" - a technique frequently used to prevent denial of service attacks - but in this case, posting from multiple accounts to prevent being detected. Stringhini et al. created honey-profiles on the top three OSNs and recorded the content and interactions made to these profiles to identify tweet attributes contributing to malware propagation (Stringhini, Kruegel, & Vigna, 2010). Benevenuto et al. focused on identifying spam centred around Twitter using twenty three tweet attributes (Benevenuto, Magno, Rodrigues, & Almeida, 2010). Grier et al. analysed spam behaviour and the effectiveness of using a blacklist of URLs to detect spam on Twitter (Grier, Thomas, Paxson, & Zhang, 2010). Yang et al. (2011) used features based on timing and automation to detect spam on Twitter. Their research was focused on social network relationships such as betweenness centrality and bidirectional link ratio between spam nodes and their neighbouring nodes. The same authors collaborated with Yang, Harkreader, Zhang, Shin, and Gu (2012) to analyse the cybercriminal ecosystem on Twitter studying inner and outer social relationships. The

Table 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 (Benevenuto et al., 2010; Cao & Caverlee, 2015; Yang et al., 2012; Chen et al., 2016; Cresci et al., 2015; Grier et al., 2010; Lerman & Ghosh, 2010; Faghani & Saidi, 2009; Stringhini et al., 2010; Yang et al., 2011)	✓	✓		✓		✓
URL characteristics (Lee & Kim, 2013; Lee & Stokes, 2011; Ma et al., 2009)		✓	✓			
Detect by analysing static code (Canali et al., 2011; Kapravelos et al., 2013; McGrath & Gupta, 2008)		✓	✓			
Detect by analysing dynamic code (Bartos et al., 2016; Burnap et al., 2014; Cao et al., 2016; Cova et al., 2010; Jayasinghe et al., 2014; Kim et al., 2017; Wressnegger et al., 2016)	✓				✓(network only)	
Our model	✓	✓		✓	✓	

inner social relationship hypothesised that criminal accounts are interconnected. The outer social relationship experiment highlighted that accounts that follow and support criminal accounts are well hidden in the network. Similarly, a feature based approach was employed by Cresci, Di Pietro, Petrocchi, Spognardi, and Tesconi (2015) by building a classifier to detect fake accounts created by cybercriminals to inflate the number of followers. To date, the research has been focused on studying OSN accounts and URL characteristics to identify those tweets or accounts that are exhibiting deviant behaviour (posting spam or malicious URLs). Providing evidence that OSN accounts or URLs may be malicious can be beneficial but given the frequency and volume at which new accounts emerge, the only way to determine actual malicious behaviour is occurring is to observe it. Once malicious activity occurs it is currently not possible to flag it and stop it. None of the methods published to date allows us to observe malicious activity and block it to minimise the damage. Thus we propose to build on the existing literature that uses characteristics as features and include them in a predictive model that will incorporate tweet attributes to predict that the URL is likely to perform malicious activity during the early stages of interaction, providing a novel enhancement to the research field whereby we can observe malicious behaviour, including that of newly created accounts with limited account history, and block it before maximum damage occurs.

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. Static analysis looks at the code that drives the page, looking for recognised malicious code and methods. Dynamic analysis executes the code by interacting with the Web page and observes the behaviour on 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 to detect phishing URLs (McGrath & Gupta, 2008).

In a similar approach, an automated classification model was built based on lexical and host-based features to detect malicious URLs using statistical models (Ma et al., 2009). Canali et al. developed a filter called *Prophiler* (Canali, Cova, Vigna, & Kruegel, 2011) that uses features derived from URLs and Web page code to determine whether a drive-by download will occur. In another approach Kapravelos et al. compared similarities between various JavaScript programs to detect malicious Web pages (Kapravelos et al., 2013).

Dynamic analysis: A system was develop by Cova et al. to detect malicious Web pages in two stages (Cova et al., 2010). In the first stage various features such as URL redirects, length of dynamic code, number of dynamic executions etc. were used to detect an anomaly. In the second part, they used a custom built browser to open the URL and record the events used to detect malicious behaviour. Building on the principle of detecting malware by analysing dynamic execution of code, Kim et al. proposed a model to systematically explore possible execution paths in order to reveal malicious behaviours (Kim et al., 2017). This is achieved by analysing function parameters that could expose suspicious DOM injection and reveal malicious behaviour. In a similar approach, Javasinghe et al. used the dynamic behaviour of a Web page to detect a drive-by download attack (Jayasinghe, Culpepper, & Bertok, 2014). Adobe Flash animations are a well-known entry point for Web-based attacks and these have been studied at various levels during the interpreter loading, and execution process to detect malicious code (Wressnegger, Yamaguchi, Arp, & Rieck, 2016). Research has also been undertaken to build a machine classifier based on network activity to detect malware. In one approach Bartos and Sofka looked at network traffic to build the classifier from data captured in the form of proxy logs generated by 80 international companies (Bartos, Sofka, & Franc, 2016). By doing so, they were able to detect both known as well as previously unseen security threats based on network traffic. Similarly, Burnap et al. built a real-time classifier specific to drive-by downloads originating from Twitter based on network activity and machine activity (Burnap et al., 2014). Looking at the dynamic redirection of Web pages has been proposed to detect phishing and spamming webpages in Lee and Kim (2013) and Lee and Stokes (2011). This was extended to using forward and graph based features in Cao, Li, Ji, He, and Guo (2016).

In summary, while excellent results have been achieved by studying the static or dynamic activity of a Web page, the focus has been on *detection*. As stated at the end of the previous section, to identify malicious activity in OSN it must be observed, and generally, once it is observed, it is a problem that needs to be remedied. As with the research in the previous section, none of the research to date that focuses on Web page activity has proposed a model capable of observing and potentially blocking malicious activity. Thus, in this paper we focus on *prediction*, proposing a model that can classify a URL into malicious or benign based on OSN account attributes (as per the previous section) 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.

3. Experimental setup

3.1. Data collection

We collected data on two popular sporting events. The rationale for choosing sporting events is that they attract a large number of users, thus increasing the chances of a malicious link being clicked. For example in 2015 the Copa America recorded 14 billion impressions alone (Laird, 2015) and the 2016 Rio Olympics was the top topic that year - surpassing even the US presidential election (Kottasova, 2016).

For our experiments, we identified the European Football Championships (#Euro2016) and the Olympics (#Rio2016) in 2016. Both generated some of the largest volumes of tweets in 2016 (Kottasova, 2016). Tweets containing a URL and hashtags relating to these events were captured via the Twitter streaming API. The rationale behind selecting two events was to the determine whether

+/-	File Access	Process Name	File Path
+	Write	C:\\WINDOWS\\system32\\wuauclt\.exe	C:\\WINDOWS\\WindowsUpdate\.log
-	Write	:*	.+\.bat
-	Write	:*	.+\.exe

Fig. 1. File exclusion list.

our predictive model would generalise beyond a single event and be applicable for use on URLs posted around other events. 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 7500 unique tweets to identify 975 malicious URLs for European Football Championships dataset and, around 5000 tweets were randomly chosen to identify around 525 unique malicious tweets for Olympics 2016 dataset by using a high interaction client side honeypot.

High interaction honeypots perform dynamic analysis of interaction behaviour between a client machine and that of a Web server. For our experimental results we used Capture HPC toolkit (Seifert, 2017). Capture HPC operates by visiting each URL that is passed to it through a virtualised sandbox environment - interacting with the Web page for a pre-defined amount of time. 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 (Puttaroo, Komisarczuk, & de Amorim, 2014). The classification is based on three *exclusion lists* (see Fig. 1) that are created based on known file, process or registry entries that are targeted by drive-by download attacks. Fig. 1 gives a typical example of rules from a file exclusion list, where each positive symbol indicates that system activity is allowed and a negative symbol means that it is not allowed and is flagged as malicious. For example any *exe* file that is written or created during the visitation of a Web page is not allowed. This exclusion list is updated every 14 days to reflect the most recent actions that have been observed in drive-by download attacks. These exclusion lists are created by formalising rules while visiting malicious or benign Web pages. A URL is classified as malicious if, while visiting the website, a system performs certain activity or activities that violate the rules.

Capture HPC therefore gives us a label we can use for supervised learning and a set of activity logs we can use to train a system to recognise the 'early warning signals' that are present *before* the exclusion list flag would have been raised. 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, we will not obtain 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 we can mitigate this limitation with regular updates.

3.2. Architecture of the predictive model

The predictive model has three main components (see Fig. 2): *feature extraction, persistent storage* and *machine learning*. The main function of feature extraction is to create a timeline of measurable observations on the client system based on 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 interval *'t'* for a period of *'p'*. For our experiment, $t = 1$ s and the observation period is defined as $p = 10$ s. The first snapshot is generated when a URL is 'clicked' at $t = 1$ s, and then subsequently at an interval of t. Each snapshot is written to a database for persistence as the sandbox environment is wiped clean after each URL has been visited. Each database insert includes (i) machine activity and (ii) metadata of the tweet containing the URL. For machine activity, we log 54 metrics including network activity, file, process, registry, RAM use, CPU usage (see Appendix A for a longer list and associated Pearson correlation scores with the malicious/benign class). While recording machine activity we have defined peak activities as the maximum number of activities observed while visiting the Web site in the given 10-s window, and irregular activities are defined by a set of activities that occur after a machine is infected. These irregular activities are activities not observed while visiting a website as defined in the exclusion list. We also use 24 pieces of metadata from the tweet, including username, user screen name, user id, follower count, friends count, and age of account (see Appendix A for longer list). This produces 78 attributes every second for a period of p. During the training phase we know whether the URL is malicious or benign based on the results from Capture HPC. 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 230 configuration with a consistent baseline.

The third component is the machine learning phase. For our predictive model we trained four different machine algorithms to determine the best method for class prediction using these data. We used the Weka toolkit 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 (J48 Decision Tree) and build multiple models to map

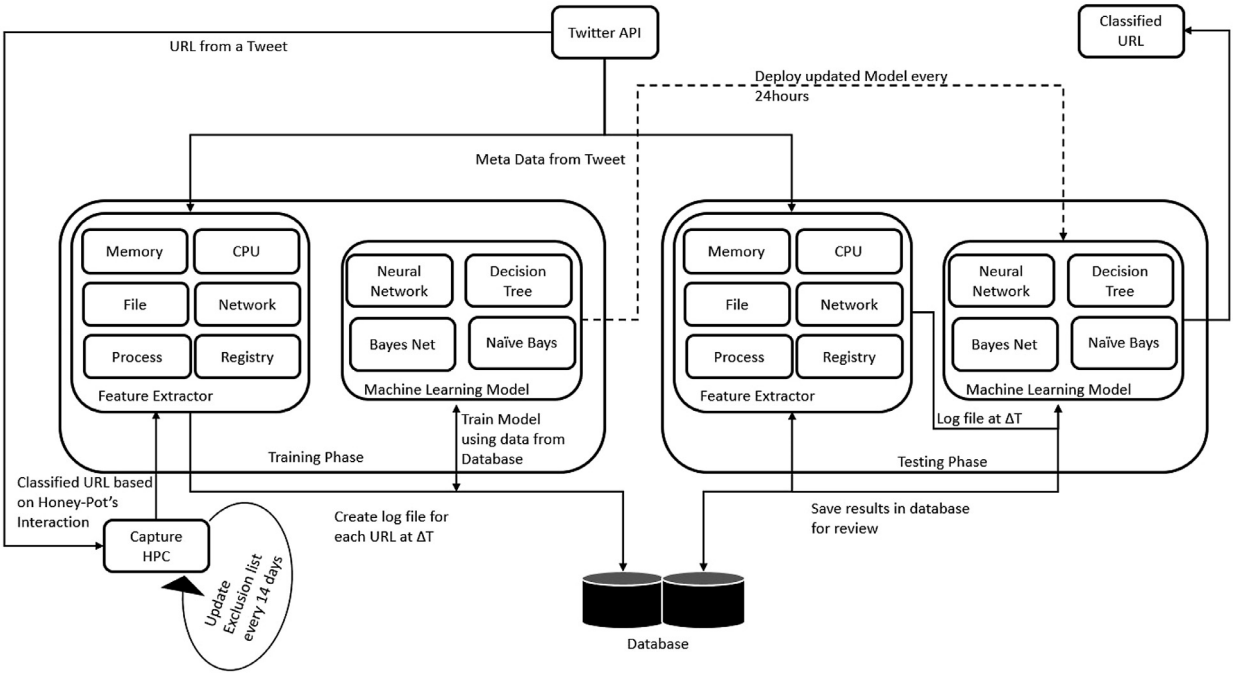


Fig. 2. Architecture of predictive model.

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 time interval. Each snapshot was passed onto 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 stops and the URL is classified as malicious, killing the connection to the Web page.

The framework is designed to be adaptive to an ever-changing environment by periodically updated the labelling method used to train and test the classifier so that new malware behaviour is reflected in the labels. This is achieved by periodically updating the exclusion list of the honeypot. The exclusion list is updated once every 14 days by running URLs in CaptureHPC after executing them in known malware labelling Web sites like Virustotal, which provide labels based on the leading commercial anti-virus tools. Based on the machine activity observed in terms of files/process/registry we update the exclusion list (Puttaroo et al., 2014).

4. Results

4.1. Training on data from Euro 2016

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 ten fold cross validation using the Weka toolkit. The results were calculated using standard classification metrics in which we define –

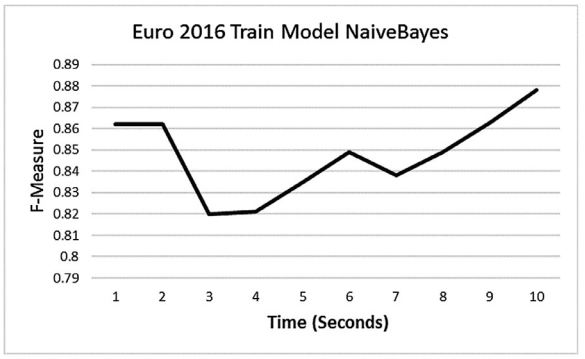
$$Precision = \frac{\text{TruePositive}}{\text{TruePositive} + \text{FalsePositive}}$$

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

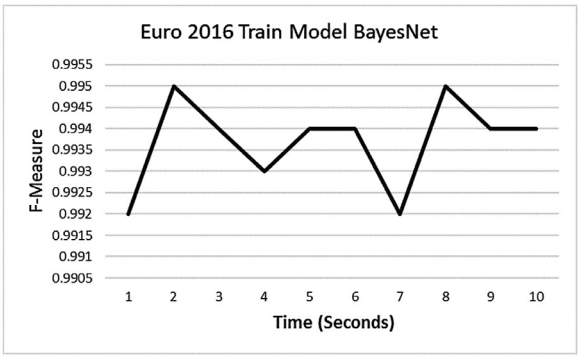
$$F - Measure = 2.\frac{\text{Precision. Recall}}{\text{Precision} + \text{Recall}}$$

We have also included False positive rate as one of metrics while testing our unseen dataset.

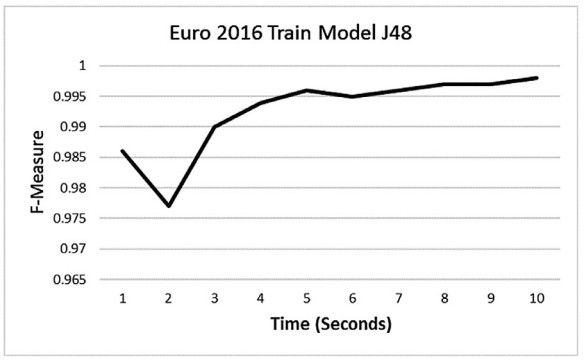
$$FalsePositiveRate = \frac{\text{FalsePositive}}{\text{FalsePositive} + \text{TrueNegative}}$$



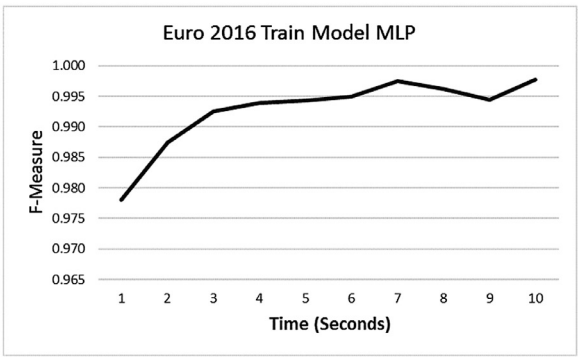
(a) F-Measure of Naive Bayes over time during training phase



(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

Fig. 3. *F*-Measure of all machine learning algorithm over time during training phase.

The results for each classifier are presented in Fig. 3. In each sub-figure, the machine learning model is trained and tested on the metrics derived using the Euro2016 data-set. Time in each table 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 1 s has elapsed since the URL has been 'clicked' (URL clicked at $t = 1$). Models built using the Naive Bayes and J48 algorithms (see Fig. 3a and c) exhibit similar behaviour - they both have a dip in accuracy from the starting point and then it gradually continues to rise up. One explanation for this could be that during early seconds there is a lack of system activity (see Fig. 4), leaving the algorithm struggling to differentiate between benign and malicious activity. We define system activities as the range of activities happening while visiting a Web page. These include process running, read/write operations happening 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 Naive Bayes, suggesting interdependencies between attributes. This is

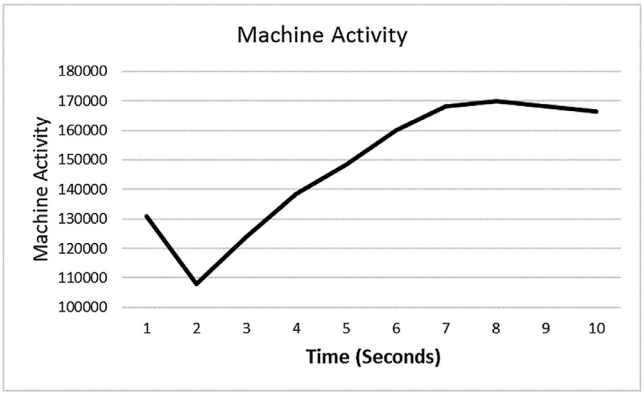


Fig. 4. Machine activity over time.

Table 2
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

logical as, for instance, when malicious network activity occurs is likely that CPU and RAM use will also spike due to additional resource being required for the activity. Looking at the results of the MLP model (see Fig. 3d) we see the model is able to better weight the machine activity and tweet meta-data to control for the lack of machine activity at the start of the interaction. The *F*-measure rises smoothly from 1 s, suggesting it is making better use of the Twitter metadata to improve accuracy in the early stages of activity. In terms highest *F*-measure achieved, the J48 and MLP models perform best with 0.998 at 10 s. At 3 s the results are almost identical. The key difference between models being a slight improvement in MLP at 2 s, but this is countered by the speed at which J48 returns a result. The MLP result takes longer than a second to be returned, whereas the J48 takes milliseconds. Thus, in practical application, the J48 model is most likely to be favourable.

4.2. Training model without online social network platform attributes

A lot of research has been done in the past to detect malicious/spam tweets propagating on Twitter based on tweet attributes (Benevenuto et al., 2010; Yang et al., 2012; Cresci et al., 2015; Grier et al., 2010; Stringhini et al., 2010; Yang et al., 2011). Thus we included tweet metadata as part of the feature set for prediction in the previous section. 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. Thus, we aimed to determine the impact of removing these features and 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 provided us the best performance - the J48 algorithm that displayed apparent correlation with machine activity. We retrained the model using only the machine activity - no tweet metadata. Table 2 and Fig. 5 show performance of the model over time.

Fig. 5 shows the *F*-measure metrics for the J48 model when trained with and without tweet metadata. When we compare the results of both 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 $t = 1$ s. This suggests that Twitter's idiosyncratic attributes such as number of followers significantly contribute to accurate classification of malicious URLs but that the model is still highly accurate when using machine activity alone, making it likely that the approach would work 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

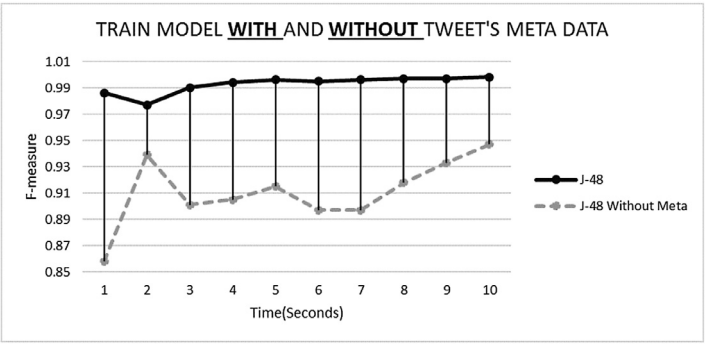


Fig. 5. Train J48 model without OSN metadata.

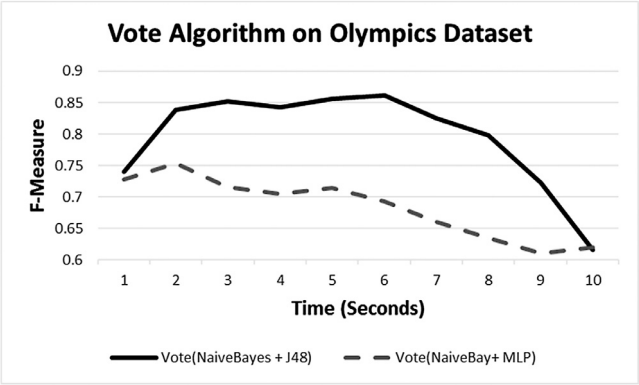


Fig. 6. Testing on Olympics data using model built earlier.

interesting as this is the opposite of the situation when metadata were 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 in future our aim will be to try and retain user account characteristics where possible when applied to 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.

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 with this experiment is that cyber attack methods vary over time. For instance, in a second unrelated event we may see a new collection of individuals spreading malicious URLs, and indeed a different behavioural profile exhibited by the URLs. We therefore now introduce an unseen dataset from the Olympics 2016. This dataset has played no part in training the model so is completely unseen, testing the generality of the approach to some degree. Given that J48, MLP and Naive Bayes (NB) models performed best on the Euro 2016 data, we combined these 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 down two combinations of methods that produced the best classification performance: J48 & Naive Bayes and Naive Bayes & MLP. Fig. 6 shows the *F*-measure for both. The combination of J48 with Naive Bayes 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 the test machine launches the URL so there is a lag of 1 s, meaning $t = 3$ is actually 2 s after the URL is clicked. The Naive Bayes and MLP combination reaches a maximum *F*-measure of 0.75. Thus there is a significant performance difference when combining the Naive Bayes and J48 models. This is somewhat counter intuitive given the MLP and J48 algorithms were almost indistinguishable at 3 s in the previous experiments, and that J48 is a rule-based model. We would expect a rule-based model to overfit to a single event (i.e. the CPU, RAM and network traffic would have a large variance between events as demonstrated by Burnap et al. (2014)). 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 15.2% False Positive rate at only 2 s into the interaction (Table 3).

Table 3
Test model on Olympics 2016 dataset.

Test Olympic- vote algorithm (Naive Bayes and J48)				
Interaction time (S)	FP rate	Precision	Recall	*F*-measure
1	0.149	0.836	0.723	0.729
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

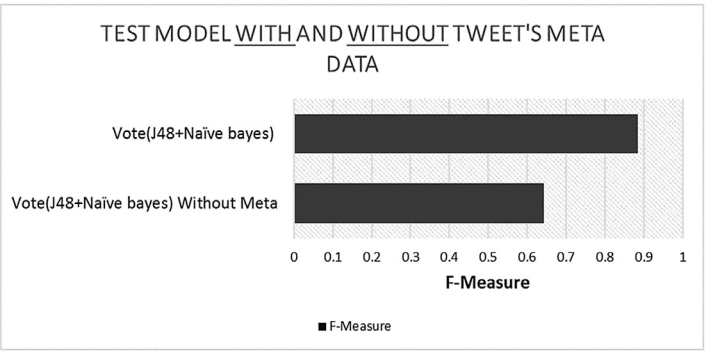


Fig. 7. Comparison of results on unseen data with and without tweet metadata.

We next rebuilt the Vote model with and without tweet metadata. Fig. 7 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 *F*-measure of the classifier when tweet attributes were added to machine data. This suggests that even though there is a similarity in tweet attributes across events they are not 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 network characteristics of the individuals posting malicious URLs, and machine activity recorded while interacting with URLs, remain fairly stable - showing a drop in *F*-measure from 0.977 to 0.833 at 2 s between events. Our model may therefore not be limited to a single case, but could be applied to multiple events that attract large users on Twitter maintaining reasonably low error rates when predicting malicious URLs just 2 s into the interaction.

4.4. Adaptive nature of the predictive model

To make our predictive model adaptive, a feed-forward architecture was implemented (see Fig. 2). 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, are 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%. Fig. 8 displays the results of these experiments. We found that training the model with only 1% of total sample size, using 10 cross fold technique, produced an *F*-Measure of 0.89. However, when we tested the model on an unseen dataset we found 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 as new data is added to the model over time and the feature diversity increases. We

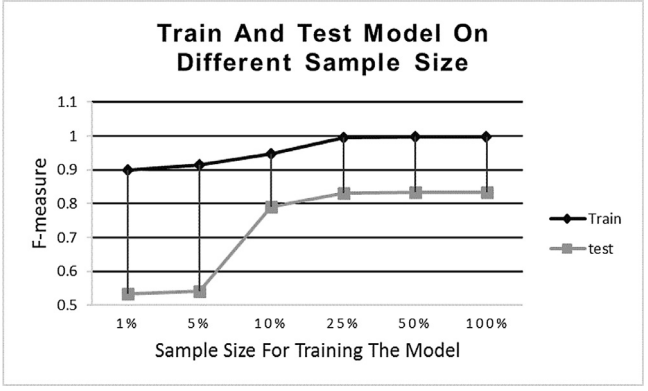


Fig. 8. Comparing classifier accuracy in terms of *F*-measure when data set is changed.

observed that the *F*-measure did indeed increase with increases in dataset size during the training phase as well as with the testing phase, showing the model to be adaptive when observing more diverse machine behaviour. We saw 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 we increased the sample size from 25% to 100%, suggesting that 25% of data representing machine activity is enough to build a model that will give us over 0.83 *F*-measure and 15% False Positive Rate. After this point more data does not appear to improve prediction accuracy.

5. Conclusions

As Online Social Networks (OSNs) become a crucial source of information publication and propagation following global events, it has become an environment that is particularly vulnerable to cyber attack via the injection of shortened URLs that take the user to a malicious server from which a 'drive-by download' attack on the local machine is launched.

In this paper, we aimed to build on a body of work that has developed methods to identify malicious URLs in OSNs in an effort to combat the problem. Existing work has developed methods to provide evidence that OSN accounts or URLs may be malicious, which can be beneficial, but given the frequency and volume at which new accounts emerge, the only way to determine actual malicious behaviour is occurring is to observe it. Once malicious activity occurs it was previously not possible to flag it and stop it. None of the methods published prior to this work allowed us to observe malicious activity and block it to minimise the damage. The main focus of our research was therefore to develop a method capable of identifying a URL as malicious or benign based on machine activity metrics generated and logged during interaction with a URL endpoint, and OSN user account attributes (in this case Twitter users) associated with the URL. Furthermore, the aim was 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 captured tweets 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. We trained the classification model 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 s 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 Twitter metadata exhibited by cybercriminals carrying out drive-by download at425 tacks 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 s - that is 1 s after launching the URL. The highest *F*-measure achieved on the unseen event was 0.862 at 5 s 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 s 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 s, 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 s 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. Future work includes increasing the granularity further by creating log files at shorter intervals to determine if we can detect malicious URLs even earlier in the execution cycle, to avoid the key limitation which is that a cybercriminal can evade detection if the connection is dropped within one second. We have used two different sporting events in this paper because of their reported popularity and therefore attractiveness as a target event. Other types of events could be included in future. From a real-world scenario, it could be possible that our proposed predictive system could be implemented to monitor tweets around ongoing events that generate large volumes of traffic to identify malicious Web servers and 445 remove them before users can click on links that interact with them.

Acknowledgement

This work was supported by the Engineering and Physical Sciences Research Council (EPSRC) Global Uncertainties Consortia for Exploratory Research in Security (CEReS) programme, grant number: EP/K03345X/1.

Appendix A

Table A1
Feature selection of attributes using Pearson's R correlation between attributes and its class (Malicious/benign).

Sr no	Pearsons correlation	Attribute name	Sr no	Pearsons correlation	Attribute name
1	0.4059	Process create time	32	0.0184	Retweet user screen name
2	0.2950	Disk io counter write bytes	33	0.0175	User time zone
3	0.2915	Disk memory free	34	0.0172	Retweet user verified
4	0.2915	Disk memory used	35	0.0151	Disk io counter read times
5	0.2914	Disk memory percent	36	0.0138	User language
6	0.2621	CPU	37	0.0137	Process id net
7	0.1125	User verified	38	0.0133	Age
8	0.0981	Virtual memory percent	39	0.0132	Retweet user timezone
9	0.0974	Virtual memory available	40	0.0103	Retweet favourite tweet count
10	0.0974	Virtual memory free	41	0.0091	Retweet user id
11	0.0974	Virtual memory used	42	0.0082	Disk io counter write times
12	0.0939	Packets received	43	0.0074	Process username
13	0.0935	Bytes received	44	0.0072	Retweet user favourites count
14	0.0891	Disk io counter read bytes	45	0.0069	Memory percent
15	0.0885	Swap memory free	46	0.0063	Process path
16	0.0885	Swap memory used	47	0.0062	Process name
17	0.0874	Swap memory percentage	48	0.0061	Process status
18	0.0799	Packets Sent	49	0.0061	Remote ip
19	0.0647	Disk io counter write count	50	0.0061	Connection Establish listen
20	0.0638	User friends count	51	0.0061	User coordinates
21	0.0627	Disk io counter read count	52	0.0047	Process id
22	0.0617	Bytes Sent	53	0.0043	Source path
23	0.0548	User name	54	0.0036	CMD line statement
24	0.0495	Retweet user name	55	0.0036	Process exe path
25	0.0368	Retweet count	56	0.0036	CPU time user
26	0.0292	User screen name	57	0.0034	Retweet user friends count
27	0.0261	User location	58	0.0023	Retweet user followers count
28	0.0234	User followers count	59	0.0010	CPU time system
29	0.0214	Type	60	0.0006	Port number
31	0.0186	Retweet user location	61	0.0000	Swap memory swap in

References

Bartos, K., Sofka, M., & Franc, V. (2016). Optimized invariant representation of network traffic for detecting unseen malware variants. *USENIX security symposium* (pp. 807–822). .

Benevenuto, F., Magno, G., Rodrigues, T., & Almeida, V. (2010). Detecting spammers on twitter. *Collaboration, electronic messaging, anti-abuse and spam conference (CEAS). Vol. 6. Collaboration, electronic messaging, anti-abuse and spam conference (CEAS)* (pp. 12–). .

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

Burnap, P., Javed, A., Rana, O. F., & Awan, M. S. (2015). Real-time classification of malicious URLs on twitter using machine activity data. *Proceedings of the 2015 IEEE/ACM international conference on advances in social networks analysis and mining 2015, ASONAM '15* (pp. 970–977). New York, NY, USA: ACM.

Burnap, P., Williams, M. L., Sloan, L., Rana, O., Housley, W., Edwards, A., et al. (2014). Tweeting the terror: Modelling the social media reaction to the woolwich terrorist attack. *Social Network Analysis and Mining, 4*(1), 1–14.

Canali, D., Cova, M., Vigna, G., & Kruegel, C. (2011). Prophiler: A fast filter for the large-scale detection of malicious web pages categories and subject descriptors. *Proceedings of the international World wide web conference (WWW)* (pp. 197–206). .

Cao, C., & Caverlee, J. (2015). Detecting spam URLs in social media via behavioral analysis. *Advances in Information Retrieval, 9022*, 703–714.

Cao, J., Li, Q., Ji, Y., He, Y., & Guo, D. (2016). Detection of forwarding-based malicious URLs in online social networks. *International Journal of Parallel Programming, 44*(1), 163–180.

Chen, C., Zhang, J., Xiang, Y., Zhou, W., & Oliver, J. (2016). Spammers are becoming" smarter" on twitter. *IT Professional, 18*(2), 66–70.

Cova, M., Kruegel, C., & Vigna, G. (2010). Detection and analysis of drive-by-download attacks and malicious JavaScript code. *Proceedings of the 19th international conference on World wide web - WWW '10* (pp. 281). .

Cresci, S., Di Pietro, R., Petrocchi, M., Spognardi, A., & Tesconi, M. (2015). Fame for sale: Efficient detection of fake twitter followers. *Decision Support Systems, 80*, 56–71.

Faghani, M. R., & Saidi, H. (2009). Malware propagation in online social networks. *4th international conference on malicious and unwanted software (MALWARE)* (pp. 8–14). .

Grier, C., Thomas, K., Paxson, V., & Zhang, M. (2010). @ spam: The underground on 140 characters or less, categories and subject descriptors. *Proceedings of the 17th ACM conference on computer and communications security* (pp. 27–37). .

Jayasinghe, G. K., Culpepper, J. S., & Bertok, P. (2014). Efficient and effective realtime prediction of drive-by download attacks. *Journal of Network and Computer Applications, 38*, 135–149.

Kapravelos, A., Shoshitaishvili, Y., Cova, M., Kruegel, C., & Vigna, G. (2013). Revolver: An automated approach to the detection of evasive Web-based malware. *USENIX Security Symposium* (pp. 637–652). .

Kim, K., Kim, I. L., Kim, C. H., Kwon, Y., Zheng, Y., Zhang, X., et al. (2017). J-force: Forced execution on javascript. *Proceedings of the 26th international conference on World Wide Web* (pp. 897–906). International World Wide Web Conferences Steering Committee.

Kottasova, I. Twitter reveals the top tweeted events of 2016. (2016). December 6, http://money.cnn.com/2016/12/06/technology/twitter-top-events-hashtags-2016/

index.html 2016, (Accessed on 11/29/2017).
Kumar, M (2017). How a drive-by download attack locked down entire city for 4 days, 2017,. https://thehackernews.com/2017/10/drive-by-download-ransomware.html.
Laird, S. (2015). The top 15 sporting events that blew up twitter in 2015, http://mashable.com/2015/12.//2015-top-sports-events-twitter/#7TVsYNhLQSqN (2015).
Lee, S., & Kim, J. (2013). Warningbird: A near real-time detection system for suspicious URLs in twitter stream. *IEEE Transactions on Dependable and Secure Computing, 10*(3), 183–195.
Lee, W., & Stokes, J. W. (2011). ARROW: Generating signatures to detect drive-by downloads. *WWW 2011* (pp. 187–196). .
Lerman, K., & Ghosh, R. (2010). Information contagion: An empirical study of the spread of news on digg and twitter social networks. Association for the Advancement of Artificial Intelligence. (pp. 90-97).
Ma, J., Saul, L. K., Savage, S., & Voelker, G. M. (2009). Beyond blacklists: Learning to detect malicious web sites from suspicious URLs. *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 1245–1254). ACM.
McGrath, D., & Gupta, M. (2008). Behind phishing: An examination of phisher modi operandi. *Usenix workshop on large-scale exploits and emergent threats (LEET)* (pp. 4). .
McMillan, R (2009). High profile twitter hack spreads porn trojan — pcworld, 2009,. https://www.pcworld.com/article/167253/article.html.
Microsoft (December 2013) Microsoft security intelligence report, Technical report.
Moshchuk, A., Bragin, T., Gribble, S. D., & Levy, H. M. (2006). A crawler-based study of spyware in the web. *NDSS. Vol. 1. NDSS* (pp. 2–). .
Provos, N., McNamee, D., Mavrommatis, P., Wang, K., Modadugu, N., (2007). The ghost in the browser: Analysis of web-based malware. *HotBots, 7*, 4–4.
Puttaroo, M., Komisarczuk, P., & de Amorim, R. C. (2014). Challenges in developing capture-HPC exclusion lists. *Proceedings of the 7th international conference on security of information and networks* (pp. 334). ACM.
Robinson, W (2015). Russia hacked pentagon's joint chiefs of staff and shut down its email system — daily mail online, 2015. http://www.dailymail.co.uk/news/article-3187344/Russia-hacked-Joint-Chiefs-Staff-shut-email-4-000-defence-department-employees-ELEVEN-DAYS.html.
Rogers, C (2016). Euro 2016 most tweeted TV of the year, 2016. https://www.marketingweek.com/2016/12/14/euros-tweeted-tv-2016/.
Sakaki, T., Okazaki, M., & Matsuo, Y. (2010). Earthquake shakes twitter users: Real-time event detection by social sensors. *Proceedings of the 19th international conference on World wide web* (pp. 851–860). ACM.
SANS Institue (2017). 2017 threat landscape survey: Users on the front line, 2017. https://www.sans.org/reading-room/whitepapers/threats/2017-threat-landscape-survey-users-front-line-37910.
Seifert, R. S. C (2017). Capture-HPC, 2017. https://projects.honeynet.org/capture-hpc.
Stringhini, G., Kruegel, C., & Vigna, G. (2010). Detecting spammers on social networks. *Proceedings of the 26th annual computer security applications conference* (pp. 1–9). ACM.
Tumasjan, A., Sprenger, T., Sandner, P., & Welpe, I. (2010). Predicting elections with Twitter: What 140 characters reveal about political sentiment. *Proceedings of the fourth international AAAI conference on weblogs and social media* (pp. 178–185). .
Virustotal - Free online virus, malware and URL scanner, https://www.virustotal.com/en/, (Accessed on 07/21/2017).
Wressnegger, C., Yamaguchi, F., Arp, D., & Rieck, K. (2016). *Comprehensive analysis and detection of flash-based malware. Detection of intrusions and malware, and vulnerability assessment.* Springer101–121.
Yang, C., Harkreader, R., & Gu, G. (2011). *Die free or live hard? Empirical evaluation and new design for fighting 475 evolving twitter spammers. Recent advances in intrusion detection.* Springer318–337.
Yang, C., Harkreader, R., Zhang, J., Shin, S., & Gu, G. (2012). Analyzing spammer's social networks for fun and profit a case study of cyber criminal ecosystem on twitter. *The international World wide web 515 conference committee* (pp. 71–80).
ZeroFox (November 2017) The social media security timeline, https://www.zerofox.com/security-timeline/.