Scalable Approach to Enhancing ICS Resilience by Network Diversity

Tingting Li
Cardiff University
lit29@cardiff.ac.uk

Cheng Feng
Siemens Corporate Technology
cheng.feng@siemens.com

Chris Hankin
Imperial College London
c.hankin@imperial.ac.uk

***Abstract*—Network diversity has been widely recognized as an effective defense strategy to mitigate the spread of malware. Optimally diversifying network resources can improve the resilience of a network against malware propagation. This work proposes a scalable method to compute such an optimal deployment, in the context of upgrading a legacy Industrial Control System with modern IT infrastructure. Our approach can tolerate various constraints when searching for optimal diversification, such as outdated products and strict configuration policies. We explicitly measure the *vulnerability similarity* of products based on the CVE/NVD, to estimate the infection rate of malware between products. A *Stuxnet*-inspired case demonstrates our optimal diversification in practice, particularly when *constrained* by various requirements. We then measure the improved resilience of the diversified network in terms of a well-defined *diversity metric* and *Mean-time-to-compromise (MTTC)*, to verify the effectiveness of our approach. Finally, we show the competitive scalability of our approach in finding optimal solutions within a couple of seconds to minutes for networks of large scales (up to 10,000 hosts) and high densities (up to 240,000 edges).**

***Keywords*-ICS/SCADA Security, Network Diversity, Optimal Diversification, Malware Propagation**

I. Introduction

Industrial Control Systems (ICS) are cyber-physical systems that are responsible for maintaining normal operation of industrial plants such as water treatment, gas pipelines, power plants and industrial manufacture. Modern industrial organizations perform an increasing large amount of operations across IT and Operational technology (OT) infrastructures, resulting in inter-connected ICS. It also creates new challenges for protecting such integrated industrial environments, and makes cyber-physical security threats even more difficult to mitigate [1]. Therefore, many industrial organizations started looking for methods to converge IT and OT infrastructures in more secure and resilient ways.

In this paper, we consider software diversification as a way of deploying products across ICS and improving the resilience of the integrated systems. However, there are various real-world constraints we might encounter when finding an optimal diversification strategy, for instance, limited flexibility of diversification for legacy systems, strict configuration policies and other (un)desirable configuration requirements. Therefore, our approach particularly considers these constraints into optimization and evaluates the impact of these constraints on the optimal diversification. Although the proposed approach was demonstrated in the domain of ICS, the approach can also be applied to find optimal diversification plans for other systems in which there are constraints on diversification of some system components.

Software mono-culture has been recognized as one of the key factors that promote and accelerate the spread of malware. It is widely acknowledged that diversifying network resources (e.g. software packages, hardware, protocols, connectivity etc.) significantly mitigates the infection of malware between similar products and reduces the likelihood of repeating application of single exploits [2]. When facing attacks using zero-day exploits (i.e. unknown exploits), the situation becomes even worse as there are no available defense countermeasures to stop them. *Stuxnet*, as the first cyber weapon against ICS, leveraged four zero-day vulnerabilities. Until September 2010, there were about 100,000 hosts over 155 countries infected by Stuxnet [3]. The invariability or high similarity of products used in most affected hosts accounts for the rapid infection and prevalence of Stuxnet. Therefore, diversity-inspired countermeasures have been introduced to improve the resilience of a network against malware propagation. However, it is not very clear about (i) how much diversification is required to reach an optimal/maximal resilience, (ii) how exactly to deploy diverse resources across a network, and (iii) how configuration constraints would harm the optimal diversification. This paper aims to mitigate stuxnet-like worm propagation by optimally diversifying resources. We consider a variety of off-the-shelf products to provide services at each host, and find the optimal assignment of them to maximize the network resilience.

There are two main trends of research investigating diversity as an effective defense mechanism. One trend seeks for solutions from software development such as n-version programming [4], program obfuscation [5] and code randomization [6]. The other trend studies diversity-inspired defense strategies from the perspective of security management. Specifically, the goal of this trend is to find an optimal assignment of diverse products for each host in a network.Detailed related work are provided in Section II.

Our work lies in the second trend of research. Most of the existing work has made three critical assumptions: (i) It was assumed in most existing work that there was *no* configuration constraints when searching for an optimal assignment of products; (ii) Currently only one vulnerable product was modelled for each node in a network, which is not realistic; (iii) Most existing work assumed that each individual product shared *no* vulnerability with any other, which implied that unique exploits are necessary to compromise different products. We contend that these assumptions in earlier work are unrealistic, and thus we drop these assumptions in this work. We specifically defined configuration constraints into the process of optimization. Also we considered a more realistic infection model of malware. In the following subsection, a simple example demonstrates how these assumptions prevent us from modelling the actual infection model of malware.

In this paper, we start with formally defining the similarity of vulnerabilities between products to reflect the similar exploitability of products. We conduct a statistical study to estimate such vulnerability similarities by using public vulnerability databases such as Common Vulnerabilities and Exposures (CVE) [7] and the National Vulnerability Database (NVD) [8]. Furthermore, we represent each host in a network by a multi-label node, which can be formally mapped to a discrete Markov Random Field (MRF) model. By combining the similarity metric and the MRF model, we can construct the corresponding infection model of potential zero-day exploits across a network with a given product assignment. We then focus on computing an optimal product assignment to minimize the prevalence of zero-day exploits. Before we give our main contributions in Section I-B, we present an illustrative example in Section I-A to further explain the motivation.

A. Motivational Example

We use a simple example in Fig. 1 to explain the motivation of this work, where a simplified network with 8 hosts is presented. Most of the existing work models the network as in Fig. 1(a), where each host is represented by a single-label node. A zero-day exploit breaks into the network from the entry node. In order to prevent the exploit (which exploits circle labels) from infecting more hosts, most existing work suggests to diversify all hosts in the way indicated by triangle and circle labels respectively in Fig. 1(a). The illustrated configuration is effective because the spread of the exploit is stopped after it compromises the entry node and hence the chance of the target node being infected is 0.

Nevertheless, it is assumed that different products share *no* vulnerabilities between each other, which is however not always the case Therefore, we improve the model by considering the vulnerability similarities between different products. Fig. 1(b) demonstrates the zero-day propagation when the two products (circle and triangle labels) have a 0.5 vulnerability similarity between each other, namely there is a 50% chance that the same zero-day vulnerability exploited at circle labels can also be exploited at triangle labels, and *vice versa*. In this case, the probability of the target node being breached is increased to approximately 0.125.

In most realistic scenarios, a host is supposed to deliver multiple services (e.g. operating systems, web servers, email servers, databases, etc), each of which is potentially vulnerable to zero-day attacks. That means each host actually offers several alternative attack vectors, and as a result, sophisticated attackers can choose the vulnerability with higher success rate to exploit the host. Therefore, we represent each host by multiple labels corresponding to different services on the host. As shown in Fig. 1(c), we add another shape of labels (i.e. red squares) to some hosts, and introduce a sophisticated attacker in possession of *two* zero-day exploits (one for round labels and the other for square labels). It can be seen from Fig. 1(c), the attacker uses the square label exploit (rather than the round label exploit) to infect its adjacent node, which gives a greater chance of success. Consequently, with the collaboration of two zero-day exploits, the risk of the target being compromised is further increased to approximately 0.5.

B. Main Contributions

From the example above, we learn that in order to find the optimal way to diversify network resources, we first need to model the resources accurately, based on which we can determine the optimal assignment of products to minimize the prevalence of exploits. We summarize the main contributions of this paper as follow:

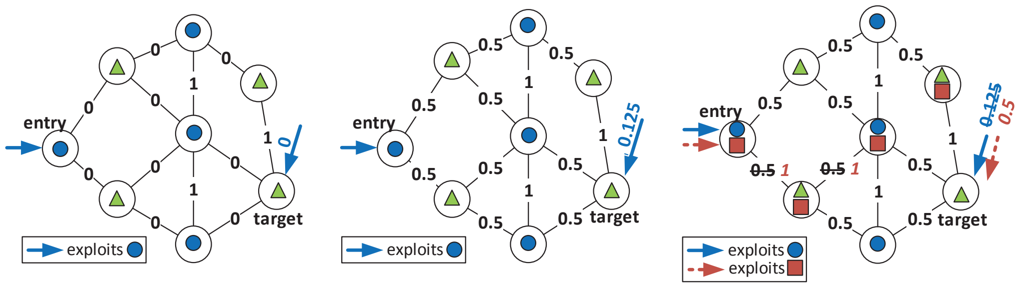
(i) Our approach can be directly applicable in practice to find the optimal diversification strategy when integrating ICS with modern IT infrastructure. We use a real-world case study inspired by *Stuxnet*, to find optimal diversification to IT-OT convergence of ICS, particularly accommodating configuration constraints and limited flexibility of diversification in legacy areas.

(ii) To the best of our knowledge, our work is the first attempt to explicitly consider the *vulnerability similarity* between products when finding the optimal diversification solutions. Specifically, we represent each host by a *multi-labelling model* with each label corresponding to a product of the host.

(iii) In order to compute the *optimal assignment* of products, we model the network by a *discrete Markov Random Field* (MRF), which then can be optimized by an efficient *sequential tree-reweighted message passing* (TRW-S) algorithm [9]. In this way, our approach can scale up well to analyze large-scale networks.

II. Related Work

Software diversity has long been recognized as a mechanism for improving resilience and security of networked computing systems [2], [10], [11]. The rationale is that it forces attackers to develop an unique exploit to compromise an individual product at each node in a network, thus



(a) diversifying single-label hosts

(b) diversifying single-label hosts with *similarity*

(c) diversifying *multi-label* hosts with *similarity*

Figure 1: Motivational example about diversifying products in a network

substantially increasing the attacking time and cost needed to penetrate a networked system at a massive scale.

A variety of methodologies for software diversity have been studied in the literature, among which the first direction of research focuses on software development diversity. Examples include n-version programming [4], execution environment diversity [12] and code randomization [6].

The second direction, which is also the focus of this paper, is the strategies for diversified deployment of resources in a networked system. For instance, based on the assumption that different variants of products share no common vulnerabilities, O'Donnell and Sethu [13] proposed to assign diverse software packages in a communication network by a distributed coloring algorithm to limit the total number of nodes an attacker can compromise using a limited attack toolkit. Newell *et al.* [14] found an efficient approach to compute the optimal solution for placing diverse software/OS variants on routing nodes to improve overall network resilience given the assumption that each variant is compromised independently with some probability metrics. Besides, there were some work defining formal security metrics for software diversity. For example, Wang *et al.* [15] defined a network security metric, k-zero day safety, for measuring the risk of unknown vulnerabilities based on the number of unknown vulnerabilities required for compromising network assets. Furthermore, Zhang *et al.* [16] introduced three security metrics to evaluate the resilience against zero-day attacks using different diversity strategies based on the number and distribution of distinct resources inside a network, the least attacking effort required for compromising certain important resources, and the average attacking effort required for compromising critical assets, respectively. Borbor *et al.* [17] explicitly considered cost constraints on optimizing software diversity strategies. It is noticed that most existing work assumes that there is a very limited attack surface provided at each host, namely there is only one vulnerable product at each host. By contrast, we explicitly model various attack vectors (offered by multiple products) at each host.

Vulnerability databases such as CVE/NVD can provide statistical evidence for measuring software diversity. For example, Garcia *et al.* [18] presented a study on the overlap of vulnerabilities in 11 different OSes with OS vulnerability data from NVD. In [19], Bozorgi *et al.* trained classifiers to predict whether and how soon a vulnerability is likely to be exploited by applying machine learning techniques on CVE data. On the validity issue of CVE/NVD, Johnson *et al.* conducted the assessment of several well-known vulnerability databases and concluded that NVD was actually the most trustworthy database [20]; we used NVD in this paper.

Some existing work [15] [16] [21] studied malware propagation based on attack graphs to assess the risk of malware along with specific attack paths and network topology. Attack graphs have been extensively studied in the community to express the exploitation conditions of vulnerabilities. However, due to the unknown nature of zero-day vulnerabilities, we contend that such approaches are not always feasible to model zero-day malware. In contrast to existing work using attack graphs, our work focuses on the speed of zero-day exploits across a network configured by similar products. Highly similar configurations (in terms of potential vulnerabilities) would accelerate the prevalence of zero-day exploits. Instead of producing specific propagation paths, we use more general undirected edges to symbolize the connections (rather than directed information flow) between different hosts. We then use the proposed similarity metric to estimate the infection rate on each edge and find optimal diversification solutions.

III. Similarity of Product Vulnerability

In this section, we formally introduce the notion of vulnerability similarity between a pair of products, namely the likelihood of an exploit compromising both products.

Definition 1 (Similarity of Product Vulnerability): Let x_i, x_j be a pair of products, $\mathbf{V}_{x_1}$ and $\mathbf{V}_{x_j}$ are sets of vulnerabilities of x_i and x_j respectively. The vulnerability similarity between x_i and x_j can be obtained by the *Jaccard similarity coefficient [22]*: $sim(x_i, x_j) = \frac{|\mathbf{V}_{x_i} \cap \mathbf{V}_{x_j}|}{|\mathbf{V}_{x_i} \cup \mathbf{V}_{x_j}|}$
Given a pair of products, the vulnerability similarity is estimated by the ratio of the number of shared vulnerabilities between the two products to the total number of vulnerabil-

ities. The rationale for this is to capture statistically how similar the vulnerabilities found on two products are.

To provide a more realistic sense, we can use the data from the NVD database [8] to calculate the similarity metric for any pair of off-the-shelf products. An example of an NVD entry is given in Table I, where the affected products of a vulnerability are sorted by Common Platform Enumerations (CPEs). CPE provides a well-formed naming scheme for IT systems, platforms and packages.

Table I: Simplified NVD Summary for `CVE-2016-7153`

CVE-ID	CVE-2016-7153

Vulnerable software &Versions	`cpe:/a:microsoft:edge:-`
	`cpe:/a:microsoft:internet_explorer:-`
	`cpe:/a:google:chrome:-`
	`cpe:/a:apple:safari`
	`cpe:/a:mozilla:firefox`
	`cpe:/a:opera:opera_browser:-`

Given the large number of vulnerabilities in NVD, CPE serves the role of sorting vulnerabilities according to their affected products. We developed a program based on CVE-SEARCH [23] to fetch necessary data from NVD, filter out vulnerabilities for each studied product, and calculate the similarity of vulnerabilities between products. The pairwise similarities are stored as *Similarity Tables*. In this way, we can calculate the similarity of vulnerabilities between *any* pair of products listed in NVD. Note that the effectiveness of the vulnerability similarity metric is subject to an assumption that the metric that is based on past zero-day attacks is a good predictor with respect to future zero-day attacks. We believe that this is currently the best way available to predict the future zero-day vulnerabilities.

For the purpose of illustration, here we use operating systems and web browsers as examples. We collect vulnerabilities for 9 common OS products and 8 common web browsers reported in the period between 1999 and 2016. Table II enumerates the pairwise similarity between the chosen OS products and Table III for the chosen web browsers. The reason for choosing these products is mainly because they have been ranked as most vulnerable products by *CVE Details* [24]. Each entry of the two tables contains the pairwise similarity calculated by Def.(1) and the number of shared vulnerabilities between products in brackets. The diagonal entries in tables are the total number of vulnerabilities of the row/column product. As the pairwise similarity is symmetric, the other half of a similarity table is omitted. For preserving the generality and flexibility of our study, we implicitly consider each different release/version of a product as a distinct *product* to compare. For instance, *Windows 8.1* and *Windows 7* are treated as two individual products and sorted by two different CPE queries `cpe:/o:microsoft:windows_7` and `cpe:/o:microsoft:windows_8.1`.

Based on the statistical studies in both tables, we conclude that a single vulnerability is likely to affect multiple products across different versions, different vendors and different platforms, which implies that a single zero-day vulnerability could probably be exploited on heterogeneous hosts in a network. Therefore, to maximize the resilience of a network against zero-day exploits, it is desirable to use the *up-to-date* products from *diverse* vendors across a network. For instance, *Windows 10* has much lower similarities of vulnerabilities with the other Windows OS, and even shares no vulnerability with *Windows XP*. However, it is not always feasible to deploy the latest and diverse products due to their incompatibilities with other services. For instance, SIMATIC WinCC is one of the most widely applied SCADA systems, but it can only operate on Windows OS, and most releases of WinCC do not fully support *Windows 10* yet [25]. It is one of the key constraints addressed in our work when finding optimal deployment for ICS.

In this section, we demonstrated the usage of CVE data to calculate the vulnerability similarity. The NVD database is one of the most well-known publicly accessible vulnerability databases, which also covers most off-the-shelf products and up-to-date vulnerability information. It is worth mentioning that the versions of selected software in both tables are constrained by the granularity of CPE search engine. The CPE entries for many vulnerabilities in NVD are not complete or of different granularities.

IV. Diverse Product Assignment

In this section, we present the formal model of a product assignment for a given network, which is to find a diversification solution to assigning products to each host such that the malware propagation between similar products can be effectively mitigated.

Each host has to provide a set of services S, such as an operating system, a web browser and a database server. Each service can be provided by a range of diverse products P. Therefore, we formally define a network in terms of hosts, links, services and products as below.

*Definition 2 (**Network**):* Let $N = \langle H, L, S, P \rangle$ be a network, $H = \{h_0, \ldots, h_n\}$ is a set of hosts. L captures the links between a pair of hosts, $L \subseteq H \times H$. $S = \{s_1, \ldots, s_m\}$ is a set of services, and $S_{h_i} \in 2^S$ denotes a set of services provided by a host h_i. $S_{h_i} = \{s_1, \ldots, s_k\}$, where $S_{h_i} \in 2^S, k \leqslant m$. P denotes a set of products, and hence each service s_j can be provided by a range of diverse products, $p(s_j) = \{p^1_{s_j}, \ldots, p^l_{s_j}\}$, where $p^x_{s_j} \in P$.

Assigning one product for each service on a host is termed as *an assignment of products* for a host.

*Definition 3 (**Product Assignment**):* Given a network $N = \langle H, L, S, P \rangle$, an assignment of products is captured by $\alpha' : H \times S \to P$, such that $\alpha'(h_i, s_j)$ is the product assignment for a service $s_j \in S_{h_i}$ at the host h_i: $\alpha'(h_i, s_j) = p^x_{s_j}$. The assignment for all services at a host

Table II: Similarity Table for Common OS Products from CVE/NVD

	WinXP2	Win7	Win 8.1	Win10	Ubt14.04	Deb8.0	Mac10.5	Suse13.2	Fedora
WinXP2	1.00 (479)								
Win7	0.278 (328)	1.00 (1028)							
Win8.1	0.009 (10)	0.228 (298)	1.00 (572)						
Win10	0 (0)	0.124 (164)	0.697 (421)	1.00 (453)					
Ubt14.04	0 (0)	0 (0)	0 (0)	0 (0)	1.00 (612)				
Deb8.0	0 (0)	0 (0)	0 (0)	0 (0)	0.208(195)	1.00 (519)			
Mac10.5	0 (0)	0.081 (109)	0 (0)	0 (0)	0 (0)	0 (0)	1.00(424)		
Suse13.2	0 (0)	0 (0)	0 (0)	0 (0)	0.170(161)	0.112 (102)	0 (0)	1.00(492)	
Fedora	0 (0)	0 (0)	0 (0)	0 (0)	0.083(75)	0.049 (41)	0.001(1)	0.116 (89)	1.00(367)

Table III: Similarity Table for Common Web Browser from CVE/NVD

	IE8	IE10	Edge	Chrome	Firefox	Safari	SM	Opera
IE8	1.0 (349)							
IE10	0.386 (240)	1.0 (513)						
Edge	0.014 (7)	0.121 (73)	1.0 (194)					
Chrome	0 (0)	0 (0)	0.001 (2)	1.0 (1661)				
Firefox	0 (0)	0 (0)	0.001 (2)	0.005 (15)	1.0 (1502)			
Safari	0 (0)	0 (0)	0.002 (2)	0.009 (21)	0.003 (6)	1.0 (766)		
SeaMonkey	0 (0)	0 (0)	0 (0)	0.001 (3)	0.450 (683)	0.001(1)	1.0(492)	
Opera	0 (0)	0 (0)	0.003 (1)	0.003 (6)	0.004 (7)	0.004(4)	1.00(492)	1.00(225)

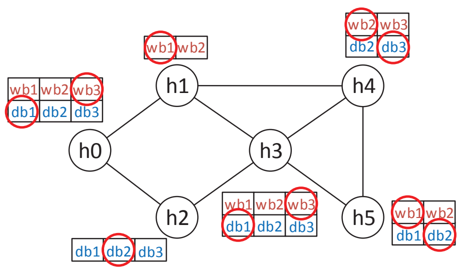


Figure 2: A network with an assignment α by red circles

$h_i \in H$ can be derived by $\alpha : H \times 2^S \rightarrow 2^P$:

$$\begin{aligned} \alpha(h_i, S_{h_i}) &= (\alpha'(h_i, s_1), \ldots\ \alpha'(h_i, s_k)) \\ &= (p^m_{s_1}, \ldots, p^n_{s_k}) \\ \text{where}\ \ & p^m_{s_1} \in p(s_1), \ldots,\ p^n_{s_k} \in p(s_k) \end{aligned}$$

Therefore α allocates products to all services running on a host, whilst α' assigns a product to a specific service of a host. An example network is illustrated in Fig. 2, where a network consisting of 6 hosts $H = \{h_0, \ldots h_5\}$ is modelled. Each host provides up to two essential services *web browser* and *database*. Three diverse web browser products $\{wb_1, wb_2, wb_3\}$ and three database products $\{db_1, db_2, db_3\}$ are available to choose. Each host might have different ranges of products to choose. A *possible* product assignment α is highlighted by red circles in Fig. 2

Now the problem is to find an optimal assignment which allocates *most diverse* products for each pair of connected hosts, so that the likelihood of a malware propagation between two hosts can be minimized. Nevertheless, some configuration requirements might hinder us from choosing the most optimal product assignment in practice. Therefore, we formally introduce *local* and *global* constraints to represent those requirements into the optimization process.

A *local constraint* indicates that for a particular host, a product p_j is required to either configure with another product p_l (expressed by c_y), or avoid the product p_k (expressed by c_x). Such requirements can also be applied to all hosts by using *global constraints*.

*Definition 4 (**Configuration Constraints**):* Given a network $N = \langle H, L, S, P \rangle$, a set of constraints $\mathcal{C}$ expresses any (un)desirable product combinations in the solution. A constrained solution $\alpha_{\mathcal{C}}$ allocates products subject to $\mathcal{C}$.

- a *local constraint* is applied to a specific host $h_i \in H$ in the form of: $c_x := \langle h_i, s_m, s_n, +p_j, -p_k \rangle$ or $c_y := \langle h_i, s_m, s_n, +p_j, +p_l \rangle$ such that the constrained solution $\alpha_{\mathcal{C}}$ satisfies:

$$\begin{aligned} &\forall c_x \in \mathcal{C} : \alpha'_{\mathcal{C}}(h_i, s_m) = p_j \ \wedge\ \alpha'_{\mathcal{C}}(h_i, s_n) \neq p_k \\ &\forall c_y \in \mathcal{C} : \alpha'_{\mathcal{C}}(h_i, s_m) = p_j \ \wedge\ \alpha'_{\mathcal{C}}(h_i, s_n) = p_l \end{aligned}$$

- a *global constraint* is applied to all hosts in H in the form of: $c_x := \langle \mathsf{ALL}, s_m, s_n, +p_j, -p_k \rangle$ or $c_y := \langle \mathsf{ALL}, s_m, s_n, +p_j, +p_l \rangle$ such that $\alpha_{\mathcal{C}}$ satisfies:

$$\begin{aligned} &\forall c_x \in \mathcal{C}\ , \forall h_i \in H : \alpha'_{\mathcal{C}}(h_i, s_m) = p_j \wedge \alpha'_{\mathcal{C}}(h_i, s_n) \neq p_k \\ &\forall c_y \in \mathcal{C}\ , \forall h_i \in H : \alpha'_{\mathcal{C}}(h_i, s_m) = p_j \wedge \alpha'_{\mathcal{C}}(h_i, s_n) = p_l \end{aligned}$$

The usage of constraints is demonstrated in the case study (Section VII-B). We next define the optimal assignment of products $\widehat{\alpha}$ and the constrained optimal assignment $\widehat{\alpha}_{\mathcal{C}}$.

*Definition 5 (**Optimal Diversification**):* Given a network $N = \langle H, L, S, P \rangle$, an *optimal* assignment of products is captured by $\widehat{\alpha} : H \times 2^S \rightarrow 2^P$, such that $\widehat{\alpha}(h_i, S_{h_i})$ is the optimal product assignment for a host $h_i \in H$.

A *constrained optimal* solution is denoted by $\widehat{\alpha}_{\mathcal{C}}$ which provides an optimal product assignment subject to a set of local and global constraints $\mathcal{C}$.

We adopt the following notation convention throughout this paper. α denotes an assignment of products for a network in general. $\widehat{\alpha}$ is for an optimal assignment *without* constraints, and $\widehat{\alpha}_{\mathcal{C}}$ is for a constrained optimal assignment. Specifically, $\alpha(h_i, S_{h_i})$ includes the products assigned to a host h_i, and $\alpha'(h_i, s_m)$ is the product assigned to a particular service s_m at the host h_i.

In the next section, we focus on finding such an optimal assignment of products $\widehat{\alpha}$ for a given network, as well as computing constrained optimal solutions in Section V-A.

V. Finding the Optimal Diversification

We need a model to represent a network in which each host has multiple services and each service can be provided by a range of products. Besides, this model has to offer sufficient flexibility, allowing each host to run a customized set of services and even the same service can have various selections of products at different hosts. Furthermore, we have to consider whether there is any existing efficient optimization algorithm to such a model. For these purposes, we choose to model the problem as a discrete Markov Random Field (MRF), which is converted into an optimal assignment problem of MRF that can be solved by an efficient message passing algorithm.

Specifically, we model this problem as a discrete MRF where each host has up to $|S|$ services, and there are up to $|P|$ products for each service $s_k \in S$. The optimization assigns up to $|S|$ products – one for each service on each host – to reach the global minima of the propagation. Given a network $N = \langle H, L, S, P \rangle$, we derive the optimization function $\mathbf{E}$ to denote the *unary cost* for each host and *pairwise cost* between a pair of connected hosts.

$$\mathbf{E}(N) = \sum_{\substack{h_i \in H \\ s_k \in S_{h_i}}} \phi(h_i, s_k) + \sum_{(h_i,h_j) \in L} \psi(\alpha(h_i, S_{h_i}), \alpha(h_j, S_{h_j})) \tag{1}$$

where $\phi(\cdot)$ denotes how likely a product is preferred by a host h_i to deliver the service s_k, and $\psi(\cdot,\cdot)$ is a pairwise cost between the products assigned to a pair of connected hosts, which in our context would be the pairwise similarity between products.

A. Unary Cost $\phi(\cdot)$ with Constraints

The unary cost is derived from the preference of a specific product for a host. By considering one product being assigned to each host, our unary cost $\phi(\cdot)$ is expressed as

$$\sum_{h_i \in H} \sum_{s_k \in S_{h_i}} Pr(\alpha'(h_i, s_k) | h_i) \tag{2}$$

where $Pr(\cdot)$ presents the probability that a product is assigned to h_i. If there is no specific preference amongst available products for each host to deliver a service, this term can be replaced by a small constant Pr_{const} for optimization. However, we considered real-world networks that are constrained by practical requirements. Therefore, the unary cost is further refined subject to any constraints. For a local constraint $c \in \mathcal{C}$ expressing an undesirable requirement $c := \langle h_i, s_m, s_n, +p_j, -p_k \rangle$ or a desirable requirement $c := \langle h_i, s_m, s_n, +p_j, +p_k \rangle$, our unary cost $\phi(\cdot)$ can be represented as follows:

$$\begin{aligned} & Pr(\alpha'(h_i, s_j) | h_i) \\ & \quad = \begin{cases} P_c(\alpha | \alpha'(h_i, s_m) = p_j, \alpha'(h_i, s_n) = p_k) & \text{if } s_j = s_m \\ Pr_{const} & \textit{otherwise} \end{cases} \end{aligned}$$

For the constrained services (when $s_j = s_m$), the unary cost is given by $P_c(\cdot)$:

$$\begin{aligned} & P_c(\alpha | \alpha'(h_i, s_m) = p_j, \alpha'(h_i, s_n) = p_k) \\ & \propto \begin{cases} 0 & \text{if } \ c := \langle h_i, s_m, s_n, +p_j, +p_k \rangle \\ \infty & \text{if } \ c := \langle h_i, s_m, s_n, +p_j, -p_k \rangle \end{cases} \end{aligned}$$

where the desirable constraint contributes no additional cost whilst the undesirable constraint introduces a large cost. The optimization is then induced to reach desirable assignments, but avoid undesirable ones. Note that such customized unary cost can also be applied for any global constraint, which is equivalent to applying a local constraint to all hosts.

B. Pairwise Cost $\psi(\cdot,\cdot)$

The pairwise cost is derived from the similarity between the assigned products. As mentioned previously, a pair of connected hosts being assigned with more similar products would have greater infection rate. When defining the pairwise cost, we penalize such similarities in order to provide a more diverse product assignment for the network. To achieve that, we define the pairwise cost term $\psi(\cdot,\cdot)$ as:

$$\sum_{(h_i,h_j) \in L} \sum_{s_k \in S_{h_i} \cap S_{h_j}} sim(\alpha'(h_i, s_k), \alpha'(h_j, s_k)) \tag{3}$$

where h_i and h_j denote a pair of connected hosts, and $sim(\cdot,\cdot)$ presents the similarity between two products providing the same service on a pair of connected hosts. It serves as a strong regularization on the product assignment as it ideally prevents the same product from being assigned to connected hosts.

C. Optimization

Based on the unary cost and pairwise cost, we can determine the optimal assignment $\widehat{\alpha}$ for N by minimizing the optimization function as below:

$$\begin{aligned} \widehat{\alpha} &= \operatorname*{argmin}_{\alpha} \mathbf{E}(N) \\ &= \operatorname*{argmin}_{\alpha} \sum_{h_i \in H} \sum_{s_j \in S_{h_i}} Pr(\alpha'(h_i, s_j) | h_i) \\ &\quad + \sum_{(h_i,h_j) \in L} \sum_{s_k \in S_{h_i} \cap S_{h_j}} sim(\alpha'(h_i, s_k), \alpha'(h_j, s_k)) \end{aligned}$$

Solving such a problem is NP-Hard [9], and the alternative way is to use an approximate minimization algorithm to achieve a solution. The well-known techniques for solving such problems are based on *graph-cuts* and *belief propagation (BP)*. The former is currently considered as the most accurate minimization approach for optimization functions arising in many complex scenarios but it can be applied to a limited range of optimization function forms. If the form is outside the class, like our optimization function in Eq. 1, BP is the common alternative. However, BP might not converge when applying to a wide range of convex functions. Instead, we employ a *sequential tree-reweighted message passing algorithm (TRW-S)* [9]. Similar to BP, TRW-S can be applied to the type of problems with the function form in Eq. 1. It is also guaranteed to give an *optimal* MAP solution in most cases [9]. TRW-S outperforms BP and graph-cuts on many heavy tasks. It also demonstrates a great potential for the cases of labeling of nearly flat probabilities, as well as the cases of large-scale networks.

Our optimization scheme mainly follows [9], which is also extended to a multi-level fashion to better fit our problem. Specifically, we enable the possibility of the parallel computation and even GPU acceleration. In addition, the optimization with constraints is also straightforward because the constraints are efficiently encoded into the unary cost by manipulating the cost for specific hosts and assignments. More details about the scalability analysis are given in Section VIII. A case study using our optimization approach in practice can be found in the later Section VII.

VI. Evaluation of Network Diversity

This section evaluates how much diversity a specific assignment can bring about into a network, and we achieve this by using a network diversity metric based on BN [16]. Given a network N and a specific assignment α, we first construct its corresponding BN to estimate the infection rate on each edge between hosts, based on which we can evaluate the network diversify by calculating the value of the metric.

We first need a way to capture the impact of the attacker's behaviour on malware propagation. From an attack entry host, there are different ways to reach the final target by continuously exploiting a number of stepping-stone hosts. At each attack step from one host to another, there are often more than one vulnerable products to exploit and induce further spread of the malware. Therefore, we extended the conventional attack paths by introducing a set of *attack nodes* to capture which exact product is chosen to exploit between a pair of connected hosts.

Attackers can choose one of the products to exploit or keep silent. Different choices lead to different propagating rates to the destination host. Without considering the similarity, the probability of a host being infected $P'_{h_k=T}$ only depends on the products being chosen to exploit at the host and the infection rate is set to the average zero-day propagation rate P_{avg}.

In this paper, we adapt the diversity metric used in this paper to evaluate product assignments for a network. The network diversity metric was proposed by Zhang *et. al.* [16] to evaluate a diversified network by measuring the *average* attacking effort needed to compromise the network. We adapt the metric to fit our model considering the vulnerability similarity of products.

*Definition 6 (**BN-based Diversity Metric** d_{bn}):* Given a BN B constructed for a diversified network N, and a specific target host h_t, the network diversity based on B can be defined as below in term of the probability of the target host being compromised: $d_{bn} = \frac{P'_{h_t=T}}{P_{h_t=T}}$ where $P_{h_t=T}$ ($P'_{h_t=T}$) is the probability of h_t being infected with (without) considering the vulnerability similarity of products.

The probabilistic metric d_{bn} estimates the average attacking effort by combining all valid attack paths. Naturally, the diversity metric d_{bn} is always less than 1.0 and the greater value indicates higher diversity. With the help of BN, $P_{h_t=T}$ captures the risk of the target host when the vulnerability similarity of products is considered. $P_{h_t=T}$ reflects the *current* robustness of the network, which is provided by the given assignment. $P'_{h_t=T}$ indicates the *maximum* potential of the network diversity. More explanations are in [16].

VII. Case Study - Upgrading Legacy ICS with Modern Industrial Networks

In this section, we present a case study on upgrading legacy control systems with interconnected IT systems, to achieve the convergence of IT and OT in modern industrial networks. Such an integration can facilitate highly interconnected Industrial Internet of Things (IIoT) applications, but also leave ICS more vulnerable by introducing more attack vectors, i.e. as the control networks are no longer isolated, malware can propagate itself across IT systems to breach the core control units causing physical damage.

Therefore, in this case study, we demonstrate the usage of our approach in finding an optimal diversification strategy to improve the resilience of the integrated systems. Particularly we consider three main constraints that might arise when applying the approach in practice:

(i) Most hosts in OT networks run legacy software, which have *no* flexibility to diversify or upgrade.
(ii) Some hosts in various networks are required to run specific software and hence cannot be diversified.
(iii) Some desirable and undesirable product combinations should be taken into account.

We start with a brief description of the case study in Section VII-A. An optimal solution and two constrained optimal solutions are then computed and illustrated in Section VII-B. In Section VII-C we evaluated the produced optimal solutions in terms of (i) the *diversity metric* discussed in

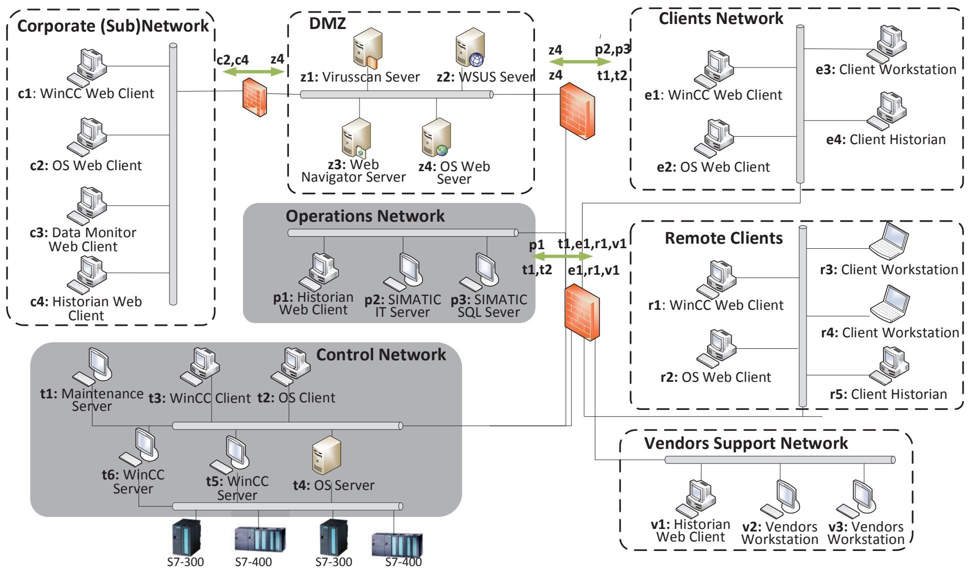


Figure 3: A typical structure of integrated modern ICS

Section VI, and (ii) the *Mean-time-to-compromise (MTTC)* obtained from NetLogo simulations we have developed.

A. Experiment Configuration

The example is adapted from the Stuxnet-like worm propagation analysis in [26]. There are newer ICS-targeted attacks such as German steel mill [27] in 2014 and Ukrainian power outage [28] in 2015, but Stuxnet is still the one with the most detailed disclosure and hence used as a case study here. Fig. 3 depicts a typical ICS architecture integrating *existing* OT zones (e.g. *Operation Network, Control Network*) with *new* IT zones (e.g. *Corporate Sub-Network, Clients Network, Vendors Support Network*). We use gray shading to indicate that OT zones have *no* flexibility to diversify or upgrade deployed software. Specific firewall white-list access rules are given in Fig. 3 to provide perimeter protection between different zones.

We use the example to demonstrate the Stuxnet worm propagation across an ICS. The primary intrusion can be introduced from the *Corporate Network, Clients Network* or *Vendors Support Network*. Once a host has been exploited as a foothold, the worm can continue scanning other connected hosts for similar vulnerabilities, by which the worm can propagate itself through the network. Stuxnet eventually breached the hosts in *Control Network*, such as t4, t5 and t6 in Fig. 3, which can access field devices.

In the following experiments, we consider the optimal assignment of products to provide three key services, i.e. an *Operating System (OS)*, a *Web Browser(WB)* and a *Database Server(DB)*. These services are distributed across all the hosts in the network according to the key role each host plays (indicated in Fig. 3). For instance, the host c1 in the *Corporate Network* is configured as a WinCC Web Client, which runs WinCC V7.x as the main application. The essential requirements for this application are a Windows OS and an IE web browser [25], and hence a range of available products that we can choose to install on c1 is provided in Table IV. The host z2 in DMZ requires a Windows OS and a Microsoft Database Server to run the WSUS server, which is reflected accordingly in the table. As a result, Table IV lists essential services required at each host and the corresponding selections of products for each service.

We highlight the legacy hosts in grey in Table IV, which run outdated software and cannot be diversified (e.g. the host p2, p3 in the *Operations Network*). The example also includes outdated versions of software running on legacy hosts such as Windows XP, MS SQL 2008. All of these introduce extra constraints in finding the optimal diversification strategy. The other chosen products in Table IV are either frequently suggested in WinCC manuals or rated as one of the most vulnerable products by *CVE Details* [24].

The similarities of web browsers and operating systems refer to Table II and III, and the similarities for DB are obtained in the same way as described in Section III. Given the products for each host in Table IV, we can compute the optimal assignment to the networked ICS by the approach discussed in Section V. It is worth noticing that our approach offers high generality and flexibility, by which each host can have a customized range of services, and each service can have various ranges of products to deploy.

B. Optimal Assignment of Products

The optimal assignment $\widehat{\alpha}$ for the case study is computed by the approach introduced in Section V and illustrated in Fig. 4(a). The assignment indicates the optimal strategy to deploy the software in IT networks when integrating with OT systems. The solution attempts to minimize the vulnerability similarity between each pair of connected hosts. From the

Table IV: Available products for essential services in the case study

Serv.	Products	c1	c2	c3	c4	z1	z2	z3	z4	p1	p2	p3	t1	t2	t3	t4	t5	t6	e1	e2	e3	e4	r1	r2	r3	r4	r5	v1	v2	v3
s_1	WIN XP										✓	✓	✓		✓	✓	✓	✓												
	WIN 7	✓	✓	✓	✓	✓	✓	✓	✓										✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ubuntu 14.04		✓		✓	✓			✓	✓				✓					✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Debian 8.0		✓		✓	✓			✓										✓		✓	✓	✓		✓	✓	✓	✓	✓	✓
s_2	IE8										✓		✓		✓	✓	✓	✓	✓				✓					✓		
	IE10	✓	✓	✓	✓			✓	✓										✓	✓	✓		✓	✓	✓	✓		✓	✓	✓
	Chrome 50		✓		✓				✓	✓				✓					✓		✓		✓	✓	✓	✓		✓	✓	✓
s_3	MS SQL 08										✓	✓	✓		✓															
	MS SQL 14			✓	✓		✓												✓			✓	✓				✓	✓		
	MySQL 5.5				✓					✓									✓			✓	✓				✓	✓		
	MariaDB 10				✓														✓			✓	✓				✓	✓		

figure, we can find that each pair of connected hosts is generally assigned with different products from each other.

As mentioned in the beginning of this section, the second type of constraints we might encounter is that some hosts are required to run specific software according to certain company policies. For this case study, we specify that the host z4, e1, r1 and v1 are required to run specific products. We outline fixed choices for these hosts in Table IV in grey. Having adding those constraints into the optimisation, we now compute the constrained optimal assignment $\widehat{\alpha}_{C_1}$, which is given in Fig. 4(b). It can be seen that whilst we fixed the products of the four hosts, the new solution accordingly updates assignments of products for several hosts to find a new optimal diversification, as highlighted by red squares.

We can also specify undesirable product combinations to avoid during optimisation. For instance, the solution $\widehat{\alpha}_{C_1}$ in Fig. 4(b) uses the IE10 on Ubuntu14.04 at host v2. If we want to eliminate such undesirable assignments, we can specify and embed *product constraints* $\mathcal{C}_2$ in the computation of optimal solutions, as introduced in Definition 4. Then we can compute the constrained optimal solution $\widehat{\alpha}_{C_2}$ that is illustrated in Fig. 4(c). It can be found that the web browsers at c2 and v2 are changed to *Chrome* as required.

The optimal solution $\widehat{\alpha}$ is produced by minimizing the optimization function presented in Section V-C, and hence it guarantees the minimal infection rate of the worm and the most diverse product assignment possible. In order to accommodate the host and product constraints, the constrained solutions $\widehat{\alpha}_{C_1}$ and $\widehat{\alpha}_{C_2}$ have to sacrifice a certain amount of diversity. In the next section, we evaluate all these optimal solutions and quantify the compromised diversity of the constrained solutions in terms of the diversity metric proposed in Section VI and *MTTC* by our NetLogo simulation.

C. Case study analysis

1) Evaluation by Network Diversity Metric: We first construct a BN for the case study with a given assignment of products, to estimate the propagation of malware. In the following experiments, we consider an attacker breaks into the system from c4 in Corporate Network, and hence we set c4 to be the root being infected with a prior probability 1.0. The final target of is set to the host t5 which has the direct access to controlling the critical field devices. Therefore, the probability of the target t5 being infected becomes the key element to calculate the network diversity metric $d_{bn} = P'_{t5=T}/P_{t5=T}$, as defined in Definition 6.

Given an assignment of products (e.g. the optimal one $\widehat{\alpha}$), we can determine the possible infection rate of malware at each edge with the help of the constructed BN. As we investigate the infection of multiple zero-day exploits, we assume that the attacker is in possession of three unique zero-day exploits, each of which exploits a particular type of product respectively (i.e. OS, WB and DB). Once a host is infected, attackers search for similar products/vulnerabilities to exploit amongst the connected hosts and proceed. When multiple exploits are feasible, attackers evenly choose one to use. The similarity between the source and chosen product decides the likelihood of infecting the chosen product.

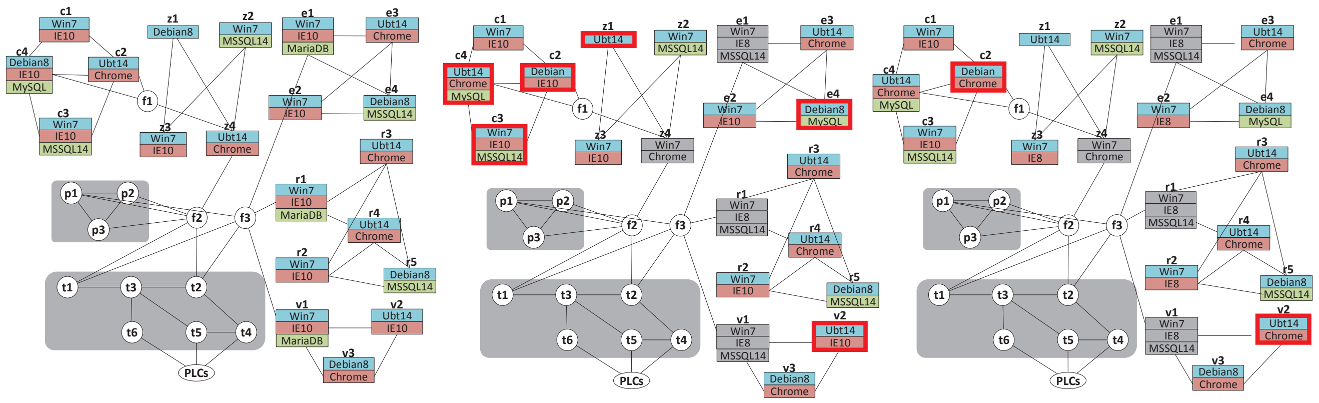
Table V: Diversity metric d_{bn} of different assignments

Label	Description	$\log P'_{t5=T}$	$\log P_{t5=T}$	$d_{bn} = \frac{P'_{t5=T}}{P_{t5=T}}$
$\hat{\alpha}$	optimal assign.	-3.151	-3.062	0.81457
$\hat{\alpha}_{C_1}$	host constr.	-3.151	-2.838	0.48590
$\hat{\alpha}_{C_2}$	product constr.	-3.151	-2.833	0.48119
α_r	random assign.	-3.151	-2.576	0.26622
α_m	mono assign.	-3.151	-1.978	0.06709

The first row of Table V is the evaluation of the optimal assignment $\widehat{\alpha}$ which reaches a very high diversity $d_{bn} = 0.81457$. The constrained optimal solutions $\hat{\alpha}_{C_1}$ and $\hat{\alpha}_{C_2}$ produce lower diversities as the two solutions are required to accommodate certain constraints.

For the purpose of comparison, we also generate a homogeneous assignment α_m which generally allocates the *same* operating system, the *same* web browser and the *same* database server for all non-constrained hosts. Such mono-assignment provides the *worst* possible diversity for the ICS case. It also shows how vulnerable the network would become if we use homogeneous products. Besides, a randomly diversified assignment α_r is also provided, which delivers a limited diversity that is significantly lower than our optimal solution.

The notation $P'_{t5=T}$ denotes the probability of the target t5 being infected *without* considering the vulnerability similarities between products. Therefore, $P'_{t5=T}$ has a constant value for all different assignments. When we take similarities into consideration, the probability of t5 being infected $P'_{t5=T}$ increases with less diverse assignments of products.



(a) Optimal assignment of products $\hat{\alpha}$ (b) Optimal assignment of products $\hat{\alpha_{C_1}}$ with *host constraints* (c) Optimal assignment of products $\hat{\alpha_{C_2}}$ with *product constraints*

Figure 4: Optimal Assignment of products for the case study

2) Evaluation by NetLogo Simulation: NetLogo is an agent-based modelling tool that enables a programmable modelling environment for simulating behaviours of complex systems [29]. We use NetLogo to construct the networked ICS as shown in Fig. 3 and simulate the propagation of malware. After breaking into the system from a host, attackers can further spread the malware across the network. Given an assignment of products (e.g. the one in Fig. 4(a)), we can determine the possible infection rate of exploits at each edge. By deploying the simulation with a given product assignment, we can determine how much time is required by attackers to penetrate the diversified network, which implies the average effort required to compromise the network. More optimal assignment should provide more resilience to the network against the penetration.

To test the resilience provided by the diversification, we designed five sets of experiment to simulate the malware propagation from five different entry points respectively – c1 and c4 from the *Corporate Network*, e3 from the *Clients Network*, r4 from the *Remote Clients*, and v1 from the *Vendors Support Network*. Once the entry host is infected, attackers search for similar products/vulnerablities to exploit from the connected hosts. We looked at the sophisticated attackers who conduct reconnaissance activities before launching attacks, and hence at each step this type of attackers always chooses the exploits with the highest success rate.

Table VI: MTTC (in ticks) against different assignments

Assignment	MTTC from **c1**	MTTC from **c4**	MTTC from **e3**	MTTC from **r4**	MTTC from **v1**
$\hat{\alpha}$	45.313	37.561	52.663	52.491	24.053
$\hat{\alpha}_{C_1}$	28.041	16.812	44.359	48.472	15.243
$\hat{\alpha}_{C_2}$	14.549	15.817	45.118	46.257	14.749
α_m	14.345	12.654	19.338	18.865	15.916

We deployed the network according to the three optimal assignments $\hat{\alpha}$, $\hat{\alpha}_{C_1}$ and $\hat{\alpha}_{C_2}$ respectively, as well as the mono-assignment α_m. Each experiment ran the simulation for 1,000 times. The average MTTC for each test is given in Table VI. The MTTC is the time steps (i.e. ticks in NetLogo) consumed by attackers to successfully reaching the final target. The results show that the optimal $\hat{\alpha}$ provides the strongest resilience to the network, as it requires the longest period of time to be compromised across all five scenarios, while the other two constrained optimal assignments can be compromised in a shorter period of time. The mono-assignment provides the weakest resilience to the network.

VIII. Scalability Analysis

We run the optimization against a series of randomly generated networks to analyse the scalability of this approach. Our optimizer is implemented using C++ and enables the multi-threading mechanism to provide high convergence speed in multi-level optimization. We apply a GPU-friendly compute unified device architecture to gain extra efficiency on complex matrix operation. All the experiments run on a mid-range computer with an Intel i5 2.8GHz CPU, a 8GB RAM and an Nvidia GTX 750. The optimization in all the following experiments can be achieved within a reasonably short time from a couple of seconds to minutes.

Table VII provides the computational time of optimizing networks with the middle-scale (20 degrees and 15 services per host) and high density (40 degrees and 25 services per host). We observe that the number of hosts has a major impact on the computational time, but our method still finds the optimal solution within 3 minutes for large-scale (6000 hosts) high-density network. Moreover, we run experiments on mid-scale and large-scale networks with various densities and the results in Table VIII show that the degree has less influence on the computational time than the number of hosts. Finally, we vary the number of services for each host on both mid-scale and large-scale networks in Table IX. For a large-scale network of 6000 hosts with up to 240,000 edges and 30 services per host, our method still performs well and

Table VII: Computational time (in seconds) for networks of various densities over different # hosts

	# deg.	# serv.	# hosts								
			100	**200**	**400**	**600**	**800**	**1000**	**2000**	**4000**	**6000**
mid-density	**20**	**15**	0.239	0.438	1.099	1.478	1.944	2.784	6.706	16.517	33.392
high-density	**40**	**25**	0.640	1.766	3.553	5.881	8.135	10.999	27.484	82.500	151.110

Table VIII: Computational time (in seconds) for various sizes of networks over different # degrees

	# hosts	# serv.	# degree									
			5	**10**	**15**	**20**	**25**	**30**	**35**	**40**	**45**	**50**
mid-scale	**1000**	**15**	0.759	1.577	1.954	2.693	3.294	4.040	4.652	5.174	5.758	6.309
large-scale	**6000**	**25**	21.239	40.940	59.216	77.583	95.750	117.810	144.470	152.040	167.190	189.710

Table IX: Computational time (in seconds) for various sizes of networks over different # services

	# hosts	# deg.	# edges	# services					
				5	**10**	**15**	**20**	**25**	**30**
mid-scale	**1000**	**20**	**$\sim$ 20,000**	0.603	1.608	2.709	4.008	5.253	6.974
large-scale	**6000**	**40**	**$\sim$ 240,000**	10.306	27.214	51.587	90.407	134.340	188.050

converges at about 3 minutes.

IX. Discussion and Conclusion

Moving towards integrated ICS enables an efficient way to operate, but also provides new attack vectors. It is now a challenging and urgent issue for many industrial organizations to find a secure way to converge OT and IT systems to provide an efficient and also resilient industrial environment. Furthermore, there are other constraints hindering us from finding an optimal solution, such as outdated legacy systems, strict company policies and other configuration requirements. In this paper, we proposed an approach based on software diversification to increase the system resilience of the integrated ICS against malware propagation.

We introduced the similarity metric to capture how similar the vulnerabilities of two products are, which was then applied in a statistical study on CVE/NVD databases. The study showed that most vulnerabilities could affect multiple products, even from different vendors. Therefore, when finding the diverse assignment of products, we explicitly considered such vulnerability similarities of products. The similarity metric can estimate the likelihood of a zero-day exploit successfully propagating itself between two products. By assigning diverse products for a pair of connected hosts, such propagation can be effectively reduced. Unlike most existing work, we do not assume that there is only one vulnerable product on each host, and instead we adopted a multi-label model to represent various attack vectors on each host, offered by different products. Such a model is of great help to investigate the collaboration of multiple exploits.

We formally represented a network by a MRF model with different services and products for each host. Such a model can be efficiently optimized by the TRW-S algorithm. We can then obtain an optimal assignment of products for the given network. The optimal solution maximizes the defense strength of the network against malware propagation. Compared to random diversification, the optimal solution is more effective in cutting off valid attack paths. In the scalability analysis, we illustrated that our method scaled well in large-scale high-density networks.

We contend that our approach has great value and potential in practical applications, by which we can advise on the best diversification strategy for a system operator to decide the most robust way to upgrade an existing ICS. We also demonstrated the practical usage of our optimization approach in a realistic case study. Furthermore, we provided a way to specify configuration constraints that we might encounter in practice. Constrained optimal solutions can be produced to accommodate these constraints.

There are several promising lines of research to carry on. The vulnerability similarity of products in this work is estimated by data from CVE/NVD. We are aware of the potential "publication bias" of CVE/NVD. However, as discussed in [20], NVD is currently the most trustworthy database, compared to the others. Besides, a more systematic way is needed to estimate the vulnerability similarity, such as (i) from the perspective of software engineering [30]; or (ii) by estimating how diverse two products are [31]. Another future direction is to evaluate the diversified network from an adversarial perspective, subject to different level of attacker's knowledge about the network configuration and vulnerabilities that can be leveraged. In such a way, we can further evaluate the results to prove that the proposed approach can provide a more resilient network against zero-day exploits.

Acknowledgment

We thank all reviewers and our shepherd Prof. Yair Amir for their insightful reviews. This research was funded by the EPSRC grant RITICS (EP/L021013/1). We acknowledge the hardware donation of NVIDIA Corporation.

References

[1] K. Stouffer, V. Pillitteri, S. Lightman, M. Abrams, and A. Hahn, "Guide to industrial control systems (ics) security," *NIST Special Publication*, vol. 800, p. 82, 2015.

[2] K. J. Hole, "Diversity reduces the impact of malware," *IEEE Security & Privacy*, vol. 13, no. 3, pp. 48–54, 2015.

[3] N. Falliere, L. O. Murchu, and E. Chien, "W32. stuxnet dossier," *White paper, Symantec Corp., Security Response*, vol. 5, 2011.

[4] A. Avizienis, "The n-version approach to fault-tolerant software," *IEEE Transactions on software engineering*, no. 12, pp. 1491–1501, 1985.

[5] S. Bhatkar, D. C. DuVarney, and R. Sekar, "Address obfuscation: An efficient approach to combat a broad range of memory error exploits." in *USENIX Security Symposium*, vol. 12, no. 2, 2003, pp. 291–301.

[6] V. Pappas, M. Polychronakis, and A. D. Keromytis, "Smashing the gadgets: Hindering return-oriented programming using in-place code randomization," in *Security and Privacy (SP), 2012 IEEE Symposium on*. IEEE, 2012, pp. 601–615.

[7] MITRE, *Common vulnerabilities and exposures*, available at https://cve.mitre.org/, last acceessed on February 09, 2018.

[8] NIST, *National Vulnerability Database*, available at https://nvd.nist.gov/, access date: February 09, 2018.

[9] V. Kolmogorov, "A new look at reweighted message passing," *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 5, pp. 919–930, 2015.

[10] P. Larsen, A. Homescu, S. Brunthaler, and M. Franz, "Sok: Automated software diversity," in *Security and Privacy (SP), 2014 IEEE Symposium on*. IEEE, 2014, pp. 276–291.

[11] B. Baudry and M. Monperrus, "The multiple facets of software diversity: Recent developments in year 2000 and beyond," *ACM Computing Surveys (CSUR)*, vol. 48, no. 1, p. 16, 2015.

[12] P. Pal, R. Schantz, A. Paulos, and B. Benyo, "Managed execution environment as a moving-target defense infrastructure," *IEEE Security & Privacy*, vol. 12, no. 2, pp. 51–59, 2014.

[13] A. J. O'Donnell and H. Sethu, "On achieving software diversity for improved network security using distributed coloring algorithms," in *Proceedings of the 11th ACM conference on Computer and communications security*. ACM, 2004, pp. 121–131.

[14] A. Newell, D. Obenshain, T. Tantillo, C. Nita-Rotaru, and Y. Amir, "Increasing network resiliency by optimally assigning diverse variants to routing nodes," *IEEE Transactions on Dependable and Secure Computing*, vol. 12, no. 6, pp. 602–614, 2015.

[15] L. Wang, S. Jajodia, A. Singhal, and S. Noel, "k-zero day safety: Measuring the security risk of networks against unknown attacks," in *Computer Security–ESORICS 2010*. Springer, 2010, pp. 573–587.

[16] M. Zhang, L. Wang, S. Jajodia, A. Singhal, and M. Albanese, "Network diversity: a security metric for evaluating the resilience of networks against zero-day attacks," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 5, pp. 1071–1086, 2016.

[17] D. Borbor, L. Wang, S. Jajodia, and A. Singhal, "Diversifying network services under cost constraints for better resilience against unknown attacks," in *IFIP Annual Conference on Data and Applications Security and Privacy*. Springer, 2016, pp. 295–312.

[18] M. Garcia, A. Bessani, I. Gashi, N. Neves, and R. Obelheiro, "Os diversity for intrusion tolerance: Myth or reality?" in *Dependable Systems & Networks (DSN), 2011 IEEE/IFIP 41st International Conference on*. IEEE, 2011, pp. 383–394.

[19] M. Bozorgi, L. K. Saul, S. Savage, and G. M. Voelker, "Beyond heuristics: learning to classify vulnerabilities and predict exploits," in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2010, pp. 105–114.

[20] P. Johnson, R. Lagerstrom, M. Ekstedt, and U. Franke, "Can the common vulnerability scoring system be trusted? a bayesian analysis," *IEEE Transactions on Dependable and Secure Computing*, 2016.

[21] T. Li and C. Hankin, "Effective defence against zero-day exploits using bayesian networks," in *International Conference on Critical Information Infrastructures Security*. Springer, 2016.

[22] S.-S. Choi, S.-H. Cha, and C. C. Tappert, "A survey of binary similarity and distance measures," *Journal of Systemics, Cybernetics and Informatics*, vol. 8, no. 1, pp. 43–48, 2010.

[23] P.-J. Moreels and A. Dulaunoy, CVE-SEARCH, gitHub repository at https://github.com/cve-search/cve-search, access date: February 09, 2018.

[24] CVE-Details, *Top 50 Products By Total Number Of "Distinct" Vulnerabilities*, available at http://www.cvedetails.com/top-50-products.php and http://www.cvedetails.com/top-50-versions.php, access date: February 09, 2018.

[25] SIEMENS, *WinCC v7.4: General information and installation*, available at https://cache.industry.siemens.com/dl/files/216/109736216/att_879785/v1/WinCC_GeneralInfo_Installation_Readme_en-US_en-US.pdf, access date: February 09, 2018.

[26] E. Byres, A. Ginter, and J. Langill, *How Stuxnet Spreads – A Study of Infection Paths in Best Practice Systems*, available at https://www.tofinosecurity.com/how-stuxnet-spreads, access date: February 09, 2018.

[27] R. Lee, M. Assante, and T. Connway, "Ics cyber-to-physical or process effects case study paper–german steel mill cyber attack," *Sans ICS, Dec*, 2014.

[28] ICS-CERT. (2016) Cyber-attack against Ukrainian critical infrastructure. "www.ics-cert.us-cert.gov/alerts/IR-ALERT-H-16-056-01".

[29] U. Wilensky, *NetLogo*, available at http://ccl.northwestern.edu/netlogo/., access date: February 09, 2018.

[30] A. Calleja, J. Tapiador, and J. Caballero, “A look into 30 years of malware development from a software metrics perspective,” in *International Symposium on Research in Attacks, Intrusions, and Defenses*. Springer, 2016, pp. 325–345.

[31] K. Nayak, D. Marino, P. Efstathopoulos, and T. Dumitraş, “Some vulnerabilities are different than others,” in *International Workshop on Recent Advances in Intrusion Detection*. Springer, 2014, pp. 426–446.