Worm Detection without Knowledge Base in Industrial Networks

Huayang Cao[1], Jinjing Zhao[2,3], Peidong Zhu[1], Xicheng Lu[1], and Chonglun Zhao[1]
[1] National University of Defense Technology, Changsha, 410073, China
[2] Beijing Institute of System Engineering, Beijing, 100029, China
[3] National Key Laboratory of Science and Technology on Information System Security, Beijing, 100029, China
Email: huayang.cao@gmail.com; misszhaojinjing@hotmail.com; {pdzhu, xclu, clzhao}@nudt.edu.cn

***Abstract*—**A sophisticated worm, namely Stuxnet, attacked Iran nuclear facilities in 2010. This incident, together with newly found similar worms, e.g., Duqu, Flame, Gauss, highlight the cyber threat in industrial networks. These worms are highly-targeted and are carefully tested before being released. They are difficult to be detected by current security products, as there is no knowledge about them when they are spreading. We introduce a worm detection mechanism in this paper, which doesn't need any knowledge of known worms. This mechanism maintains a worm propagation model and traces the spread of suspicious files and triggers alerts based on the model. The experiment of detecting Stuxnet shows its efficiency. We also give a performance analysis at the end of this paper.

***Index Terms*—**industrial network, Stuxnet, worm detection, colored petri net.

I. Introduction

Industrial networks are critical infrastructures of our society. This kind of network was thought to be reliable and safety due to the complex architecture and specialized network technologies which act as a barrier in front of adversaries. However, with the incremental connection with public network and adoption of standard network technologies, such thought is not applicable anymore. Traditional cyber-attacks may also appear in the industrial networks. Stuxnet has been an impressive example. In 2010, a worm named Stuxnet invaded in Iran nuclear power station and damaged many centrifuges without being detected[1], this worm is considered as a real start of cyber warfare[2].

Current worm detection measures can be classified into two categories, say signature-based approach and behavior-based approach. The former method relies on signature extracted by human experts and uses these signatures to find known malwares in the local file system. The high accuracy in detecting known malwares makes it widely adopted in Anti-Virus security softwares. In contrast, the latter method judges an executable process or network flow by monitoring its behaviors, i.e., calling critical APIs or aggressively scanning ports. This method can deal with unknown threat and is often used in IDS.

We can see from these newly found industrial worms that these worms are not opportunistic, but highly-targeted. These worms adopt technologies that are different from that of well-known Internet worms. They exploit some quite-new or even zero-day vulnerabilities, so there are no signatures of such worms in the Anti-Virus software database. Besides, the industrial network environment is not the same as the Internet, in which the Anti-Virus database is not always up to date. On the other side, as the worm is highly-targeted, it behaves in-offensive on non-target hosts, which also makes it difficult to be detected by behavior-based manners. In addition, Stuxnet contains many industrial process codes which beyond the understanding of COTS security products.

To secure the industrial network, we need a defense-in-depth technique which is distinguished from that in the Internet circumstance. This technique should focused on defending against new, never-before-seen, highly-targeted threats, than against well-known, widely distributed, and opportunistic threats[3]. Here we define *Highly-target* and *New* as follows*:*

Highly-target: the worm sneaks from a publicly accessible network zone to the core part of the industrial network and unfold its destructive power here. During the propagation, the worm carefully hides its behaviors on the infected non-target host.

New: the worm or its polymorphic forms have never been seen before, thus there are no signatures in security software's database, and no usable malware samples for any knowledge training process that often used in detecting unknown polymorphic worms.

In this paper, we propose a novel mechanism that comprises detection server and distributed agents to detect highly-targeted new worms in industrial networks in a cooperation way. This mechanism doesn't need any knowledge base or training phase, but maintains a worm propagation model and monitors the penetration behaviors of suspicious files in the industrial network. By performing the monitoring over the whole network, this mechanism provides a defense-in-depth measure that against worm-based intrusion.

Manuscript received June 10, 2013; revised October 14, 2013.
Corresponding author email: huayang.cao@gmail.com
doi:10.12720/jcm.8.11.716-723

II. Related Work

Worm detection has been a hotspot in computer security research area, as various worms pose serious threats to our information systems. There are two trends in worm detection technologies, which are signature-based approach and behavior-based approach. The former method detects worms based on the signatures that extracted from known worm samples, and focuses on how to accurately and efficiently extract signatures from these worms. Ref. [4] proposed a general virus scanner that based on regular expression matching. A pattern matching algorithm was used to compare all known viruses with a suspicious file to see if it matches a known virus. This kind of detection heavily depends on the virus knowledge base and can be bypassed with polymorphic viruses. Based on computer measurements extracted from the operating system, Ref. [5] proposed a technique to detect the presence of unknown worms. A series of experiments were designed to find proper feature set that can be used to detect unknown worms. An active learning approach was also used to maintain a low false-positive rate. This method is efficient on detecting polymorphic worms, but cannot deal with new worms which are described above. Ref. [6] proposed a new worm signature, namely the position-aware distribution signature, to improve the detection accuracy. The new signature was a collection of position-aware byte frequency distributions, and is thought to be more flexible than the traditional signatures of fixed strings while it is more precise than the position-unaware statistical signatures. Basically speaking, the signature-based measures have low accuracy on detecting polymorphic worms, and have nothing to do with new worms as there are no signatures for them. On the contrary, the behavior-based approaches mainly focus on detecting polymorphic worms, which detect worms based on monitoring malicious behaviors and doesn't depend on the content-signatures. *vEye* [7] was a novel mechanism that detects worms based on a behavior sequence of common self-propagate worms. Various worms are analyzed and the authors proposed that there are shared behavior sequence among self-propagate worms, that is, target selection/probing, exploitation and replication. However, a sophisticated worm like Stuxnet behaves quite in-offensive on non-target hosts, and several zero-day vulnerabilities are exploited, which makes its behaviors difficult to be sensed by any monitoring measures. Authors of [8] proposed an Internet worm monitoring system. This system monitored the trend of illegitimate traffic to detect a worm at its early propagation stage. *Kalman* filter estimation was used to against the background noise. The authors conducted experiment on detecting Code Red[9] worm to show the efficiency of the proposed system. Ref. [10] and [11] also detect fast worms by monitoring anomaly traffics. Current behavior-based measures based on the hypothesis that while polymorphic worms have different signatures from original ones, they share the same malicious behaviors on the infected hosts. This hypothesis may be true for the polymorphic worms, but is not true for a specialized and highly-targeted new worm, such as Stuxnet.

Generally speaking, current signature-based and behavior-based worm detection measures are both knowledge-based measures, as they need either collected knowledge or training knowledge. This kind of measure can detect known and polymorphic worms, but not highly-targeted new worm as stated previously.

III. System and Threat Model

In a typical industrial network, there are several inter-connected network zones. According to security requirements, they can be divided as public zone, enterprise control zone, support zone, wireless communication zone, manufacturing operations zone, perimeter zone, process control zone, etc. These zones have different safety protection levels that some of these zones are connected publicly and some only can be accessed by authorities. In the core part of an industrial network, such as a group of process controller computers, the network components are isolated from other network zones with gap technology. However, in most scenarios, even the core zone is not absolutely isolated because of maintenance or corrective requirements. And that's why there are worms that can invade into the network from a peripheral network zone and find a way to sneak into the core part of network.

We assume that an adversary cannot directly put a worm in the core zone due to the high level protection here. In the contrary, the worm is introduced into the industrial network from an edge network, e.g., the public network. Due to the zoning mechanism and air-gap, the core zone cannot be directly accessed from outside. So the worm cannot just exist in memory like Slammer [12], it must exist as disk files and propagate via various ways, otherwise the worm may be easily erased by a reboot of infected host.

The main motivation behind this work is the observation that while the worm may be able to exploit zero-day vulnerabilities and hide their attempts on infected host, they cannot hide their propagation routine that approaches the critical hosts, only on which can it get access to industrial facilities. In contrast, a valid file normally doesn't have copies across multiple security level zones.

In industrial network, there are only a group of hosts that directly control the physical facilities. We don't try to find the worm when it begins to infect the first host in the whole network, because it is difficult, if not impossible. Instead, we alert the administrator to the worm before it causes serious accidents on the underlying industrial facilities. We think that a proper warning which can lead to a deep inspection of critical facilities is of great help of saving the facilities from a potential catastrophe.

We will give a worm propagation model in industrial network as a Colored Petri Net. Colored Petri Nets are a powerful modeling technique for complex systems[13], which are very good at representing complex processes. Traditional Petri Nets can only model processes of single kind of resources by token. Colored Petri Nets extend the traditional Petri Nets model and combine state and action into a single diagram by labeling the tokens with various colors that reside in places (or states). Colors can indicate various attributes of resources in the system, and thus the Colored Petri Nets can model multiple processes at the same time. Tokens move from one place to another through transitions. A transition allows tokens to pass if it is fired. Normally, a transition can fire if all input arcs are enabled, which means that tokens are available in places for each input arc. The quantity of tokens can decrease as tokens from multiple places may be merged (or unified) at transitions. It can also increase when tokens left transitions and are duplicated to multiple destination places. Colored Petri Nets can be organized in hierarchical structure to allow reuse and top-down or bottom-up development, just the same as in computer programming languages.

Petri Nets are usually represented in graphical representation. In an illustration of Colored Petri Nets, places are denoted by ovals or circles, Transitions are represented by rectangles or squares, and lines with arrows indicate arcs. If a predicate or tuple is written next to an arc, a token must satisfy the predicate or unify with the tuple before it may pass through the arc. Token colors are defined at the top of each Colored Petri Net in terms of tuples of standard values, such as booleans, integers, strings, or even data structures.

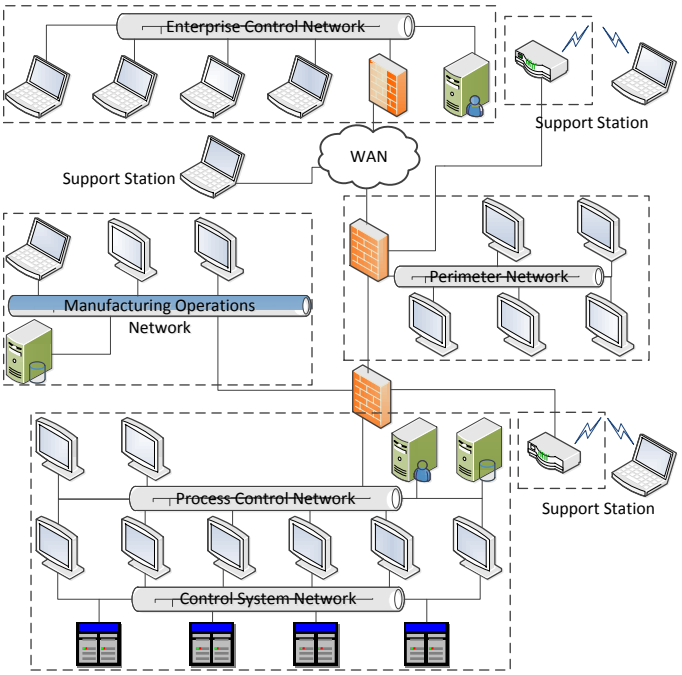

Figure 1. A typical industrial network.

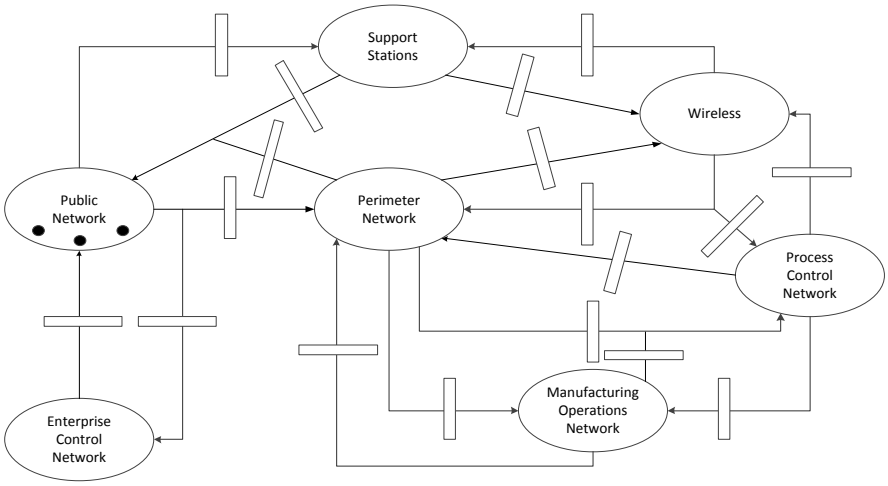

Figure 2. Colored Petri Net model of an industrial network prior to worm infection.

Fig. 1 is a typical industrial network [14], in which the whole network is divided as multiple network zones. We give a worm propagation model for this network with Colored Petri Net in Fig. 2. In Petri Nets, the ovals are called as places, here they are network zones. The rectangles are transitions, which are the worm propagation channel. The dots in a place are called as tokens, which represents that there are worms in this place. Colored Petri Net extends classic Petri Net that the tokens can have attributes, thus we can model multiple worms by introducing more tokens in the same model.

We define this Colored Petri Net model as follow:

Definition 1. The model is a quintuple $N = (P, T, F, C, M_0)$ where:

- P is place set. According to Fig. 1, Table I gives some typical facilities in each place.
- T is transition set. In Petri Nets, a transition may fire if there are sufficient tokens in all of its input places, and certain preconditions are met.
- F is the set of arcs. Arcs run between a place and a transition, but never between places or between transitions.
- C is color set. Each color stands for a kind of worm in this model. We assume there are three worms can propagate in this industrial network and define them as follows:

$$Color\ worm_1 = bool\ with\ (True, False). \quad (1)$$

$$Color\ worm_2 = bool\ with\ (True, False). \quad (2)$$

$$Color\ worm_3 = bool\ with\ (True, False). \quad (3)$$

We use subscript to indicate different worms. The Boolean returns a True or a False if there is or is not a worm in the place.

- M_0 is the initial configuration. We assume the three worms come from the public network, then:

$$\mathrm{M_0(Public\ Network)} = (1,1,1), \quad (4)$$

$$\begin{aligned} &M_0(Enterprise\ Control\ Network) \\ &= M_0(Support\ Stations) \\ &= M_0(Manufacturing\ Operations\ Network) \\ &= M_0(Perimeter\ Network) \\ &= M_0(Process\ Control\ Network) \\ &= M_0(Wireless) \\ &= (0,0,0) \end{aligned} \quad (5)$$

where the triple stands for occurrence of the three worms.

Definition 2. Transition rules:

- A transition $t \in T$ can fire if ①all input places of t have *token*②propagation channel between input and output places of t has vulnerabilities.
- When a transition fires, it doesn't consume tokens from *input* places, but creates one token in each of its output places. The attribute of created token is the same as the one that fires this transition. If the out place already has such a token, then no new tokens will be created.

As described above, in classic Petri Nets, when a transition fires, it consumes tokens from input places, and creates tokens in its output places. Since we know that worms normally don't remove their original copy after infecting new targets, we define the transition rule No.2. And in this paper, we don't care worm propagation among hosts in the same network zone, so a token only represents whether a network zone contains such kind of worms, but not the worm quantity in the zone.

TABLE I. TYPICAL FACILITIES IN INDUSTRIAL NETWORK PLACES

Places	Typical facilities in places
Public Network	Wide Area Networks, e.g., the Internet
Enterprise Control Network	WinCC Web Client, OS Web Client, Data Monitor Web Client, Historian Web Client, etc.
Support Stations	Perform maintenance and corrective operations
Wireless	Wireless connections
Manufacturing Operations Network	Historian Web Client, SIMATIC IT Server, SIMATIC IT SQL-Server
Perimeter Network	Virus scan Server, WSUS Server, Web Navigator Server, PCS7 OS Web Server, CAS Server
Process Control Network	WinCC Server, PCS7 OS Server, Maintenance Server, Engineering Station. This area controls industrial facilities.

According to this Petri Net model, in a worm-based intrusion, the worm comes from an adversary and chooses a publicly accessible network zone as the initial landing point. Then, the worm propagates over various channels, such as network connection (including file/print share channels), removable media, etc. Due to the aforementioned reasons, the worm is not discovered by current security products when it is spreading. And after a long-time-hiding, it reaches the most critical part of the network, says Process Control Network in our model, and performs attacks. In contrast, an ordinary file is more likely to move inside the same network zone only, and not likely to appear everywhere in the industrial network, especially for the files that reside in the most critical network zone.

IV. WORM DETECTION MECHANISM

A. Framework

The detection mechanism is based on the worm propagation model. As shown in Fig. 3, this mechanism maintains the above model and comprises a detection server and many monitoring agents. These agents are deployed on hosts in the industrial environment and monitor file systems of these hosts.

The mechanism detects worms based on the following criteria:

1) Unknown files

There are numbers of files in the industrial network, and most of them are valid. Our mechanism only detects

worms from suspicious unknown files. A reliable method to differentiate valid and unknown files is manual recognition conducted by human experts. However, identifying all files in the industrial network is somewhat an impossible mission. In practice, we assume that the initial state of industrial network is secure, and ignore all files that already exist before the deployment of agents. That means only new files which occur after the deployment of our mechanism will be monitored.

2) Anomaly based

Our mechanism monitors the propagation of unknown files. If a file propagates along a path in the Petri Net model and finally reaches the most critical zones, our mechanism will consider it as a malicious file and fire alerts. The more security levels and network zones the file transmitted, the more severe it is considered by our mechanism. Notice that if an unknown file only occurs in a core network zone, no alerts will be fired, because we assume that the worm cannot be directly put into the core zone and those newly occurred files may be manually created by administrator or may be the system log/cache/temp files. This paper doesn't consider a malicious inner administrator who directly put a malware in the core zone.

The workflow of the detection mechanism can be simply described as follows:

Step1: Set up the detection server, and deploy monitoring agents on each host in the infrastructure network.

Step2: Each agent scans local file system periodically to find newly created files, gets fingerprints of these files and submits this information to detection server using a one-way transmission channel. The adoption of one-way transmission is to avoid adding a bypass channel between different security level zones.

Step3: Detection server traces these files. If finding any propagation path that coincides with the worm propagation model, it will alert administrators.

B. One-way Transmission

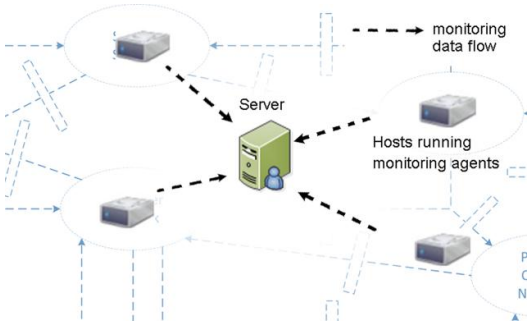

Figure 3. Worm detection framework.

As shown in Fig. 3, our mechanism employs a detection server that receives monitoring data from agents located in different network zones. This measure actually introduces a potential bypass channel among different security level zones. To remove such threat, we need to set up a one-way transmission channel that only accepts data transmission from agents to server, but not vice versa. By doing this, no data can be exchanged among different security level zones via these channels.

Since industrial networks normally deployed over large area and comprise of multiple network zones, setting up a transmission channel from every agent to server is difficult. We set up the channels as follow:

As shown in Fig 3, we introduce a proxy agent in each network zone. In each zone, monitoring agents sends data to the proxy agent over existing network links. Then several relays are deployed in the industrial network, which receive collected data from proxy agents over wireless channels, and forward these data to the detection server. The links and arrows in Fig. 4 only indicate data transmission of our mechanism, but not existing network flows. To meet the requirement of one-way transmission, the input and output ports on relays and proxy agents should be implemented as independent hardware. The output port on agents, proxy agents and relays can only send data to specific destination addresses, and input port on proxy agents, relays and detection server can only receive data from specific source addresses. The transmission protocol can use IP and UDP as underlying protocols. We assume that an adversary can put worms in the industrial network, but cannot capture or replace industrial devices. Otherwise, the adversary can sabotage the industrial network without designing the sophisticated worms. Due to time constraints, we have not implemented the one-way transmission hardware, but realize their function via software.

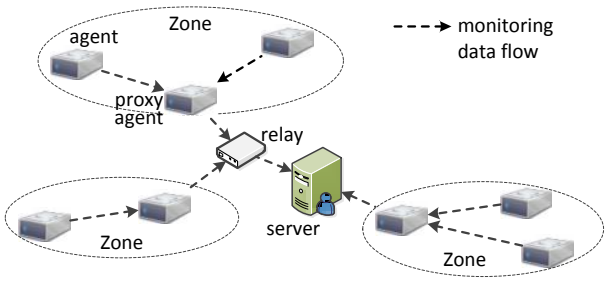

Figure 4. One-way transmission with proxy agents and relays.

C. Detection of polymorphic worms

As stated above, the main focus of our mechanism is detecting highly-targeted new worms in industrial network environment. However, this mechanism still faces the task of detecting polymorphic worms. A worm may update and modify itself while propagation, which poses an obstacle for our mechanism to trace its propagation.

A worm normally has two ways to change its appearance, that is, online update and self-modification. If a worm changes itself via online update, it can be appeared totally different from its original copy because it can receive any information from the worm writer. But if it changes via self-modification, there must be some similarities between resulted worm and original one, at least they share the same code of self-modification, and can be detected by content matching.

In industrial network, most network zones are disconnected from the Internet, thus a polymorphic worm is most likely to be the result of self-modification. We introduce a content matching algorithm in our mechanism as shown in Fig. 5 to find same content patterns in two files. When finished, the *pattern_set* stores all patterns that are shared by both files.

```
FUNCTION File_Matching(file f1, file f2)
    FOR pattern_length in [len_min, len_max]
        FOR EACH pattern p with pattern_length, appeared in f1
            IF(Find p in f2)
                Record p in pattern_set
            ENDIF
        ENDFOR
    ENDFOR
    FOR EACH pattern q in pattern_set
        IF(q is sub_pattern of p)
            Remove q
        ENDIF
    ENDFOR
ENDFUNCTION
```

Figure 5. Process of the proposed file matching.

On an agent host, the monitoring agent firstly finds all new files. Then these files are sent to detection server. The server checks these files by the file matching function to extract shared content patterns. A threshold is set to determine if two files are the same. In this scenario, the detection server traces content patterns instead of file to find worm penetration.

V. Validation

A. Case Study

To validate our mechanism, we deploy a prototype of our mechanism and evaluate it by detecting the Stuxnet worm. The experiment is conducted in the LAN environment of our lab as shown in Fig. 6. We roughly divide the LAN into three zones. Computers in zone A and B belong to two projects respectively, which are considered as non-critical zones. Zone C is considered as a critical zone, as computer VII doesn't belong to any project and is occasionally used. Computer VIII is our detection server. We also establish LAN_2 environment which has the same structure as LAN_1, but without people using it. Computers in LAN_2 all run Windows XP operation system with the service pack 2. Given this simple experimental environment, we don't install *relay* devices.

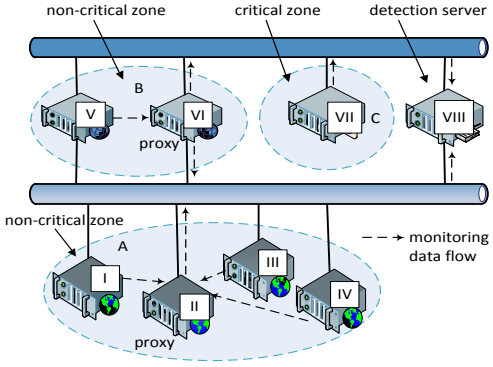

Figure 6. LAN environment of worm detection experiment.

We deploy our mechanism in LAN_1 to see if our normal working can trigger alerts, as this LAN is a daily working environment. And then we deploy LAN_2 and inject Stuxnet into this LAN to evaluate the ability of our detection mechanism. The Stuxnet worm is injected into computer I by using an USB disk. To avoid the self-deactivation [1], we adjust the system date back to May 02, 2012. To ease the worm propagation, we turn off firewalls, uninstall AV software and don't install any security patch for these computers.

TABLE II. Detection Results of Normal Worms

Worms	Detected files
Worm.Nimaya.	spoclsv.exe
W32.HLLW.Kilonce	Killonce.exe Run32.exe Regedit.exe.sys Riched20.dll Shdocvw.dll
W32.HLLW.Gaobot.gen	Csrrs.exe Scvhost.exe System.exe
W32.viking	rundl132.exe logo_1.exe _desktop.ini vdll.dll

As Stuxnet doesn't modify itself while propagation, we only use file name as the file-fingerprint during this experiment. Host II and host VI act as proxy agents in zone A and B respectively. Host VII sends data to detection server directly. In the experiment, detection server in LAN_1 finds some duplicate files that appear in multiple computers. However, no alert is fired as computer VII doesn't hold these files. In contrast, detection server in LAN_2 triggered six alerts, because six new files are found both in zone A, B and C. The six files are mdmcpq3.PNF, mdmeric3.PNF, oem6C.PNF, oem7A.PNF, mrxcls.sys, mrxnet.sys, which are the same as reported malicious files of Stuxnet[15]. We can also see from this experiment that Stuxnet has an effective propagation mechanism. Only one hour after the initial infection, it infects computer II, III, IV, V and VI. And after about another one hour, it infects computer VII and triggers alerts.

From the experiment we can see that the detection server doesn't make false alert in the scenario of normal working environment and successfully alert us to the Stuxnet worm in the infected environment.

We also test the framework with some normal self-propagation worms. Table II shows the detection results.

B. Performance

The storage and computation cost is evaluated in this subsection.

As for the storage cost, on the agent side, we need to maintain two file-lists that contain the disk file information in adjacent scanning intervals. Both of the lists are sorted at first, and then are compared to get the new files in the latter interval. Fig. 7 is a schematic

diagram of the process of finding new files on agent hosts. The rectangles indicate file fingerprints. Our mechanism doesn't need manual intervention. All of the new files found on agent hosts are considered as unknown files and their fingerprints are sent to detection server for tracing purpose.

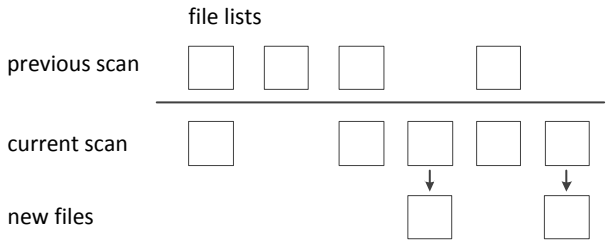

Figure 7. Process of finding new files on agent hosts.

We assume each host stores Y files on disk and creates X new files at each interval on average. And each piece of file information occupies L bytes. If the agent works for duration of T hours and each interval is t hours, then the storage cost on agent host is:

$$S_a = \left[Y + \frac{T}{t} \times X + Y + \left(\frac{T}{t} - 1\right) \times X\right] \times L$$
$$= \left[2Y + \left(\frac{2T}{t} - 1\right) \times X\right] \times L. \quad (6)$$

On the server side, given that there are N hosts in the industrial network. We need to maintain the Petri Net model which can be stored as an adjacent matrix which consumes N^2 bytes. We can organize the received file information as Fig. 8. The host information is arranged into lists and is attached to corresponding file fingerprint. Every time when receiving file information, the server scans corresponding host lists to decide whether to send alert. It should be noted that there may be same hosts in host lists of different file fingerprint, but this will not occupy redundant disk space, as they are just pointers to the original host information.

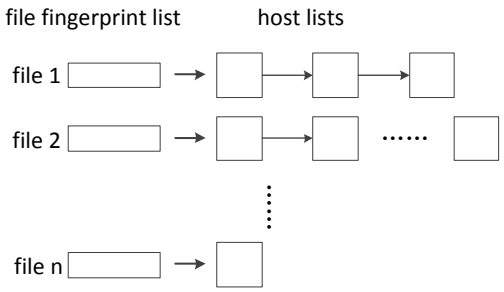

Figure 8. Structure of hosts list of each traced file information.

If each piece of host information needs M Bytes space, and each file appears on Z hosts on average, then the storage cost on server is:

$$S_s = N^2 + \frac{N}{Z} \times \frac{T}{t} \times X \times L + N \times \frac{T}{t} \times X \times M$$
$$= N\left[N + \frac{T \times X}{t} \times \left(\frac{L}{Z} + M\right)\right] \quad (7)$$

Given an industrial network with one thousand hosts, if we set (X, Y, Z, T, t, L, M) as (5,200000,3,8760,1,50,10), then the mechanism occupies about 24MB and 1170MB free space on agent host and server respectively. If we set a time threshold on server that the server can discard file information that exceeds the threshold, the storage cost can be further reduced.

As for the computation cost, the agent performs one sort and comparison operation during each interval, so the computation cost C_a is $O(n \times \log n)$, where n is the number of files on disk. On the server side, each time the server receives file information, it checks the file and host lists and decide whether an alert should be fired. The file and host lists can be sorted with binary insertion sort, so the computation cost C_s can achieve $O(\log n)$, where n is the number of hosts in the industrial network.

VI. Conclusion

The sophisticated, highly-targeted worms pose serious threat to industrial network, as current security products are not effective against them. In this paper, we propose a novel mechanism for detecting worms in industrial network without knowledge base. Our mechanism based on worm propagation model and monitors the penetration of unknown files. While the penetration can be viewed as a kind of malicious behavior, the proposed measure can be classified as a behavior-based approach. However, traditional measures only consider worm behaviors in a single host, but our proposal looks over worm behaviors at a higher level, that is, the whole network level. By maintaining the worm propagation model, our measure doesn't require a training phase or knowledge base of known worms. The experiment has shown its potential capacities for detecting unknown worm. Since no training or knowledge base is required, this measure is generally applicable to protect industrial network against worm-based intrusions. However, in the Internet environment, due to the large scale and complex network connections, this technique may be unpractical. Our next step will be further investigation of the detection of hidden worms.

At last, we suggest that industrial network owners adopt COTS security products to defense against known worms, as they are effective against traditional malwares. However, COTS security products are mainly designed for desktop environment and may be deeply inspected by organized malware writers, e.g., the terrorists or opponent organizations, and as a necessary complement, customized security measures should be set up.

Acknowledgment

This work was supported by the National Natural Science Foundation of China under Grant 61170285 and 61100223. Hunan Provincial Innovation Foundation for Postgraduate under Grant CX2010B030, and NUDT Innovation Foundation for Postgraduate under Grant B100605.

References

[1] Stuxnet. Wikipedia. (June 2013). [Online]. Available: http://en.wikipedia.org/wiki/Stuxnet
[2] T. M. Chen, "Stuxnet, the real start of cyber warfare?" *IEEE Network*, vol. 24, pp. 2-3, November, 2010.
[3] The Stuxnet Worm and Defenses for Advanced Threats. (June 2013). *Industrial Defender, Inc*. [Online]. pp. 10. Available: http://www.industrialdefender.com/advisory/stuxnet/tech_paper/2011_01_05_stuxnet_defenses.pdf
[4] S. Kumar and E. H. Spafford, "A generic virus scanner in c++," in *Proc. 8th Computer Security Applications Conf.*, 1992, pp. 210-219.
[5] N. Nissim, R. Moskovitch, L. Rokach, and Y. Elovici, "Detecting unknown computer worm activity via support vector machines and active learning," *Pattern Analysis and Applications*, vol. 15, pp. 459-475, November 2012.
[6] Y. Tang and S. Chen, "An automated signature-based approach against polymorphic internet worms," *IEEE Trans. on Parallel And Distributed Systems*, vol. 18, pp. 879-892, July 2007.
[7] X. Jiang and X. Zhu, "vEye: Behavioral footprinting for self-propagating worm detection and profiling," *Knowledge and Information Systems*, vol. 18, pp. 231-262, February 2009.
[8] C. C. Zou, W. Gong, D. Towsley, and L. Gao, "The monitoring and early detection of internet worms," *IEEE/ACM Trans. on Networking*, vol. 13, pp. 961-974, October 2005.
[9] Code Red (Computer worm). Wikipedia. (June 2013). [Online]. Available:http://en.wikipedia.org/wiki/Code_Red_(computer_worm)
[10] Y.-C. Jhi, P. Liu, L. Li, *et al.*, "PWC: A proactive worm containment solution for enterprise networks," *Security and Communication Networks*, vol. 3, pp. 334-354, July 2010.
[11] S. Chen, X. Wang, L. Liu, X. Zhang, and Z. Zhang, "WormTerminator: An effective containment of unknown and polymorphic fast spreading worms," in *Proc. ACM/IEEE Symposium on Architecture for Networking and Communications Systems*, 2006, pp. 173-182.
[12] SQL Slammer. Wikipedia. (June 2013). [Online]. Available: http://en.wikipedia.org/wiki/SQL_Slammer
[13] G. Helmer, J. Wong, M. Slagell, *et al*., "Software fault tree and colored petri net based specification, design and implementation of agent-based intrusion detection systems," *International Journal of Information and Computer Security*, vol. 1, pp. 109-142, January 2007.
[14] Security Concept PCS 7 and WinCC - Basic document. (June 2013). *White Paper, Siemens, Simatic*. [Online]. pp. 47. Available: https://a248.e.akamai.net/cache.automation.siemens.com/dnl/jE/jE2MjIwNQAA_26462131_HB/wp_sec_b.pdf
[15] N. Falliere, L. O. Murchu, and E. Chien. (June 2013). W32. Stuxnet Dossier. *White paper, Symantec Corp., Security Response*. [Online]. pp. 18. Available: http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/w32_stuxnet_dossier.pdf

Huayang Cao received his B.S. degree in Network Engieering and M.S. degree in Computer Science from Department of Computer Science, National University of Defense Technology (NUDT), Changsha, Hunan, China, in 2007 and 2009 respectively. He is now a PhD student of NUDT. His research is focused on Internet routing security, and security for cyber-physical systems and social networks.
He visited Helsinki Institute for Information Technology(HIIT), Finland as a visiting Ph.D. student during December 2011–November 2012.
Mr. Cao is now a student member of IEEE.

Jinjing Zhao received her Ph.D. degrees in School of Computer from NUDT, Changsha, China, in 2007. She is currently an associate professor at Beijing Institute of System Engineering. Her major research interests include computer networks, and information security.

Peidong Zhu is a professor with School of Computer Science of NUDT, China. He received his PhD degree in computer science from NUDT in 1999. His research interests include network routing, network security and architecture design of the Internet and various wireless networks.
He was the visiting professor at St Francis Xavier University, Canada, from December 2008 to December 2009.
Prof. Zhu is now a member of IEEE.

Xicheng Lu received his B.Sc. degree in computer science from Harbin Military Engineering Institute, China, in 1970. He is currently a professor in the School of Computer Science, NUDT. His research interests include distributed computing, computer networks, and parallel computing.
He was a visiting scholar at the University of Massachusetts between 1982 and 1984.
Prof. Lu has served as a member of editorial boards of several journals and has co-chaired many professional conferences. He is an academician of the Chinese Academy of Engineering.

Chonglun Zhao received his B.Sc. degree in School of Computer from Northeastern University, Shenyang, China, in 2011. He is now a Master student of NUDT. His research is focused on security of cyber-physical systems.