International Journal of Hybrid Information Technology
Vol. 10, No. 7 (2017), pp.1-10
http//dx.doi.org/10.14257/ijhit.2017.10.7.01

APT Detection with Concolic Execution

Yunfei Su, Mengjun Li, Chaojing Tang and Rongjun Shen

School of Electronic Science and Engineering, National University of Defense Technology, Changsha 410073, China
suyunfei@nudt.edu.cn

Abstract

Advanced persistent threat (APT) is sophisticated cyber-attack and has attracted lots of attention of security researchers in cybersecurity. Traditional defense measures based on signature matching such as antivirus products and IDS/IPS are insufficient to detect APT. Concolic(a portmanteau of CONCrete and SymbOLIC) execution is a hybrid software verification technique that performs symbolic execution which could be used for APT detection. In this paper, we proposed a framework of APT detection which includes network traffic redirection module, user agent, reconstruction module, dynamic analysis module and response module. With the help of concolic execution in dynamic analysis module, the framework could effectively and accurately detect APT attacks compared with current defense systems. We provide a detailed example to illustrate how the framework works against APT attacks especially passive attacks.

***Keywords**: advanced persistent threat, concolic execution, cyber security, symbolic execution, APT detection.*

1. Introduction

At present, cyber-attacks which lead to too much cost are more sophisticated and common in cyberspace. Original network attacks in cyberspace have evolved into APT attacks, which are more complicated, stealthy and have profound effects. Advanced persistent threat has attracted lots of attention in recent years. The term of advanced persistent threat may originate from two main sources [1]. One is the security company Mandiant [2], which provided a detailed report about APT. The other is the U.S. Department of Defense [3]. Generally, an advanced persistent threat is a set of stealthy and continuous network hacking processes, often orchestrated by human(s) targeting a specific entity. An APT usually targets organizations and/or nations for business or political motives. APT processes require a high degree of covertness over a long period of time. The “advanced” process signifies sophisticated techniques using malware to exploit vulnerabilities in systems. The “persistent” process suggests that an external command and control system is continuously monitoring and extracting data from a specific target. The “threat” process indicates human involvement in orchestrating the attack [4].

Stuxnet, Operation Aurora, Duqu, Flame, Red October, Miniduke are typical examples of APT [5] [6]. According to these examples, we could get the common characteristics: a) the objective is to sabotage key infrastructures or exfiltrate information including intellectual property, political activities schedule and so on, b) the stage is multistep and kept for a long time, and sometimes it may last for several years, c) APT attacks are sophisticated, more than one zero-day vulnerabilities are usually used, d) the targets of APT are not widespread but specific.

APT attack is difficult to detect with current commercial security products, such as Anti-virus products, intrusion detection system and so on. Present network security systems generally include firewall, network intrusion/prevention detection system and virus scanners, but APT attacks can easily evade such defense mechanisms without being award just like Stuxnet did.

The main contribution of this paper is proposing a general network security framework with the aim of detecting APT attacks, which could effectively and accurately detect APT attacks compared with current defense systems. The main idea of the framework is to reconstruct application dataflow from network dataflow, which then could be used to dynamically detect the APT without affecting the using of computers on the network. We introduced the concolic execution to malicious behavior detection and automatic signature extraction, which could significantly enhance the accuracy and effectiveness of APT detection.

The remainder of the paper is structured as follows: In Section 2, we introduce current attack detection systems and the deficiencies to detect APT. In Section 3, we introduce concolic execution and its usage in APT detection. In Section 4, we explain in detail the framework of APT detection system, In Section 5, an example of APT detection using this framework is shown, and the related works present in Section 6 following with the conclusion in Section 7.

2. Traditional Measures of APT Detection

The main countermeasures against Cyber-attacks are generally deploying anti-virus products/host intrusion detection system (HIDS) and network intrusion detection system (NIDS).

All of them mainly use a mechanism of blacklist, which keeps a set of specific data segments or rules for judging. For example, when the network traffic contains a specific string “\xEA\x82\x63\xAE\xA3\x8C\x66\x49\”in the blacklist, the NIDS will issue an alert, similar to the others. Obviously this mechanism also called signature detection has a vital weakness: it needs the pre-defined signature extracting from the known cyber-attacks, then signature matching effects. So anomaly detection had been proposed in NIDS, which can be used to detect unknown attacks. But this method has a natural weakness, i.e. a relative high false positive which seriously degrades the availability.

APT attack generally exploits zero-day vulnerability to gain the access privilege of the targets, such as Stuxnet [5] exploited four zero-day vulnerabilities in windows operating system including Windows print spooler (MS10-061), LNK format (MS10-046), Win32k Keyboard Layout (MS10-073), and task scheduler (MS10-092). These zero-day vulnerability exploits are nearly impossible to be detected by detector based on signature matching.

Otherwise, original cyber-attacks usually use positive attacks to gain full access privilege by directly sending several crafted packets to the target without interacting with the user. The positive attacks can be blocked easily by the IPS deployed in the network. But in APT attacks, more passive attacks emerge, which need to interact with the users such as open a specific URL, download the email attachment and open it or insert the USB stick to the computer and so on. Spear phishing being used widely in APT attacks is one type of the passive attacks. Attackers exploit the crafted attachment (doc, xls, pdf, etc.) of normal Email sent by a personate companion or other persons who the victim trusts, once the victim opens the crafted attachment, arbitrary code could be run in the victim’s computer through the embedded shellcode in the attachment. This kind of attacks is even more difficult to detect by the traditional detection measures using at present, especially the files or URLs accessed by the user are encoded while transmission in the network.

3. Concolic Execution

Symbolic execution [7] is a means of analyzing a program to determine what inputs cause each part of a program to execute. An interpreter follows the program, assuming symbolic values for inputs rather than obtaining concrete inputs as normal execution of the program would. It thus arrives at expressions in terms of those symbols for expressions and variables in the program, and constraints in terms of those symbols for the possible outcomes of each conditional branch. Symbolic execution is mainly used in bug detection.

```
 1  void foo( ){
 2      int a, b;
 3      a=read( );
 4      b=a*3+1;
 5      if(b>100){
 6          exploit( );
 7      }else{
 8          ok( );
 9      }
10  }
```

Figure 1. An Example Code

Consider the program in Figure 1, which read in a value and can be exploited when the input is greater than 33. During a normal execution ("concrete" execution), the program would read a concrete input value (*e.g.*, 10) and assign it to variable a. Execution would then proceed with the calculation in 4 and the conditional branch in 5, which would evaluate to false and execute function ok().

During symbolic execution, the program reads a symbolic value (*e.g.*, X) and assigns it to variable a. The program would then proceed with the calculation and assign X * 3 + 1 to b. When reaching the if statement, it would evaluate X * 3 + 1 > 100. At this point of the program, X could take any value, and symbolic execution can therefore proceed along both branches, by forking two paths. Each path get assigned a copy of the program state at the branch instruction as well as a path constraint. In this example, the path constraint is X * 3 + 1 > 100 for the then branch and X * 3 + 1 <= 100 for the else branch. Both paths can be symbolically executed independently. When paths terminate, symbolic execution computes a concrete value for X by solving the accumulated path constraints on each path with constraint solver such as Z3, STP and so on. These concrete values can be thought of as concrete test cases that can, *e.g.*, help developers reproduce bugs. In this example, the constraint solver would determine that in order to reach the exploit() statement, X would need to be greater than 33. Symbolically executing all feasible program paths does not scale to large programs.

While symbolic execution the number of feasible paths in a program grows exponentially with an increase in program size which finally leads to path explosion [8]. Another problem degrading symbolic execution is environment interactions. Programs interact with their environment by performing system calls, receiving signals, etc. Consistency problems may arise when execution reaches components that are not under control of the symbolic execution tool [9].

Concolic execution[10-12] is a hybrid program execution that performs symbolic execution, along with a concrete execution (with particular inputs) path. Starting with a concrete input, concolic execution symbolically executes the program, gathering input constraints from conditional statements encountered along the way. Consider the program in Figure 1, with a malicious input (*e.g.*, 50) that want to trigger function exploit(), we will concretely execute the statements from 2 to 6, meanwhile path constraint X * 3 + 1 > 100 are collected by symbolic execution, then we can get the accurate character of malicious input X > 33.

Traditional signature are in forms of regular expressions like "/^RCPT TO\x3a\s*\x3c?[^\n\x3e]{256}/im", with concolic execution the signature can be described in constraint form such as " buf[0-1]=='PK' && buf[120-123]*buf[124-127]<0 " which is more accurate.

We implement the malicious behavior detection and automatic signature extraction by extending S2E platform. S2E is a platform for analyzing the properties and behavior of software systems [9].The S2E platform reuses parts of the QEMU virtual machine, the KLEE symbolic execution engine, and the LLVM tool chain. S2E currently runs on Mac OS X, Microsoft Windows, and Linux, it can execute any guest OS that runs on x86, and can be easily extended to other CPU architectures, like ARM or PowerPC. In S2E the path explosion and environment interactions problems are alleviated by selective symbolic execution. We currently extend S2E with a malicious behaviors monitor plugin, which dynamically monitor the stack frame and heap allocation state when concolic execution. With a sample executed by S2E, the malicious behaviors monitor plugin will issue an alert when a return address in the stack frame or the header of heap block is overwritten by symbolic variables, at the same time, we get the constraint of input that could lead to the malicious memory address overwritten. The constraint then can be used as a signature by IPS.

4. Proposed framework

The main objective of the framework is to detect and block APT attacks automatically in vivo. To achieve this, it is important that the attacks recognized within the framework are real attacks, especially passive attacks. With the help of concolic execution in dynamic analysis module, the framework could effectively and accurately detect APT attacks compared with current defense systems. The proposed invivo framework is coupled loosely and lightweight to the computers of users, the deployment is easy and it has low overheads to the users in the network after deployment.

The proposed framework is shown in Figure 2 which includes five components: network traffic redirection module, user agent, reconstruction module, dynamic analysis module and response module.

A. Network Traffic Redirection Module

This module copies the actual network traffic and redirects it to the Reconstruction module. This can be easily done by a switch with mirror ports or a server with more than one network interface card.

B. User Agent

The main functionality of user agent is to providing auxiliary information about the host to the Reconstruction and Decision modules. The auxiliary information includes some basic operating information such as whether the desktop is active, the process information of the top window and so on which are useful to the Reconstruction and Response modules. The user agent is similar to lightweight host monitor software.

C. Reconstruction Module

Using the reconstruction module we can reconstruct the application flow according to the network flow together with reducing the dependency to the host. The functionality of this module is to restore the data which may contain malicious contents and then send the data to the dynamic analysis module. The form of the data may be segments of HTML text, documents (doc, xls, pdf, etc.) or executable files, it also can be several packets sent to the application running in the host.

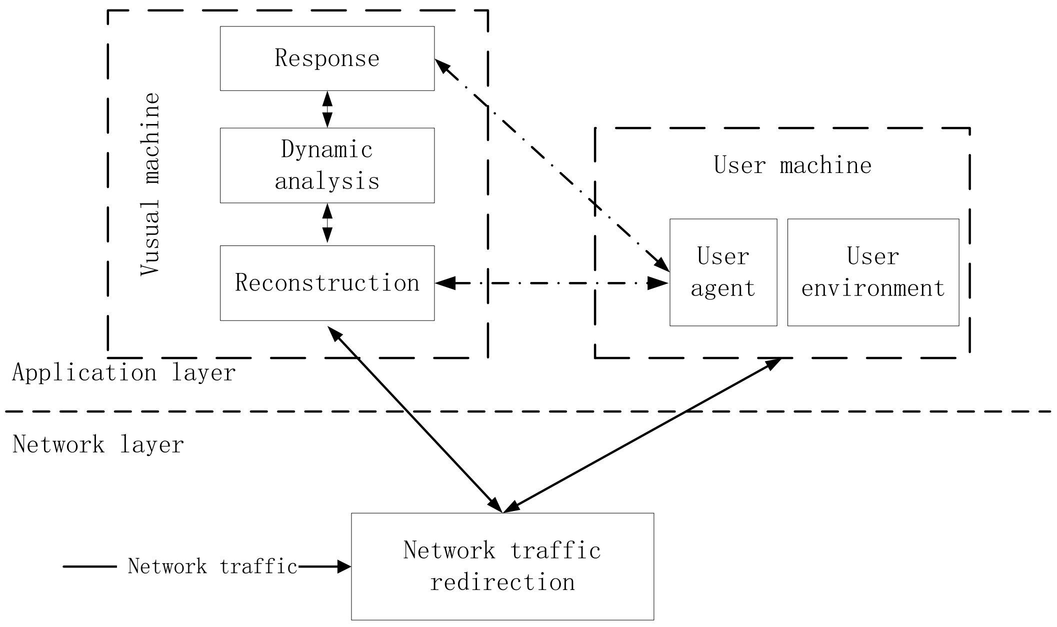

Figure 2. The Framework of APT Detection

D. Dynamic Analysis Module

With the auxiliary information provided by User agent module and the application data provided by the Reconstruction module, we can get a virtual environment whose context is similar to any host in the inner-network. The main difference compared with the actual host is that the visual environment is heavily armed by dynamic analysis system which could effectively find out the malicious behaviors in the application data.

Unknown attacks can be effectively revealed by dynamic analysis due to every behavior they do will be logged. So the exploitation with zero-day in APT can be always detected theoretically. For example, after the .doc document opened by WinWord.exe which is one of the components of Microsoft Office, if the request to one remote address is launched, we should log these suspicious behaviors and send them to the response module. A second example, the exploitation of vulnerabilities has obvious behavior features, taking the exploitation of buffer overflow as an example, to get arbitrary code execution the input data should overwrite the memory address whose value could be passed to EIP, these addresses called sensitive-address includes Return Address, Function pointer, VT pointer, etc. So we can monitor these addresses in runtime, once sensitive-address covering happens we will issue an alert. We currently use S2E as our dynamic analysis platform, all the samples received are executed in QEMU visual machine with concolic execution mode.

E. Response Module

This module integrates former information and makes a decision according to pre-defined criteria. Once malicious behavior determined, the correspond user agent will notify the user with a message, at the same time, the signature extracted by concolic execution will be sent to IPS to block similar attacks.

APT attack can be divided into three main stages: Reconnaissance, Exploitation and Command & Control (C&C). In stage 1 reconnaissance, information gathering for guiding the next attack is the primary work. Exploitation and C&C are the main stages in APT attack which lead to real destruction.

The proposed framework focuses on the latter two stages, i.e. stage 2 exploiting phase, stage 3 interacting with C&C server. The detail will be presented in next section.

5. A Case of Framework in Detecting APT

Figure 3 shows a typical deployment of the framework in actual network. In the network, firewall, NIPS, NIDS and honeypot together with our Reconstruction and Dynamic Analysis Server (RDAS) are deployed against network attacks especially APT attacks.

We assume a simplified APT attack scenario: attacker's objective is to get some classified files stored in the computers that locating in the organization's inner-network. The inner-network is protected by firewalls, NIPS, NIDS, honeypot and anti-virus software in every computer in the organization.

According to the three stages we divided into, the attackers should first gather enough information to make an attack plan. After the reconnaissance stage, the attackers may get information as follows: the organization may have two subnetworks, one is called demilitarized zone (DMZ) which provides www, email and other business services, and the other may be inner-network protected by another firewall which forbids any access initiated from the outside; there are several email addresses belong to the employees of the organization; the web server in DMZ has several critical vulnerabilities that can be exploited to control the web server. With the information above, attackers find out that getting the classified files directly by positive attacks is difficult, and then make two passive attack plans: Plan A, exploit the web services in DMZ and tamper the web page in the server, the browser will run arbitrary code such as downloading a Trojan into their computers and executing it (exploit zero-day vulnerability of browser) once the employees access the tampered web page; Plan B, personate someone's companions and send an email attached a crafted attachment, for example, a .doc document, the Winword process will run arbitrary code such as releasing a Trojan and executing it (exploit zero-day vulnerability of Winword) once the employee open the .doc document.

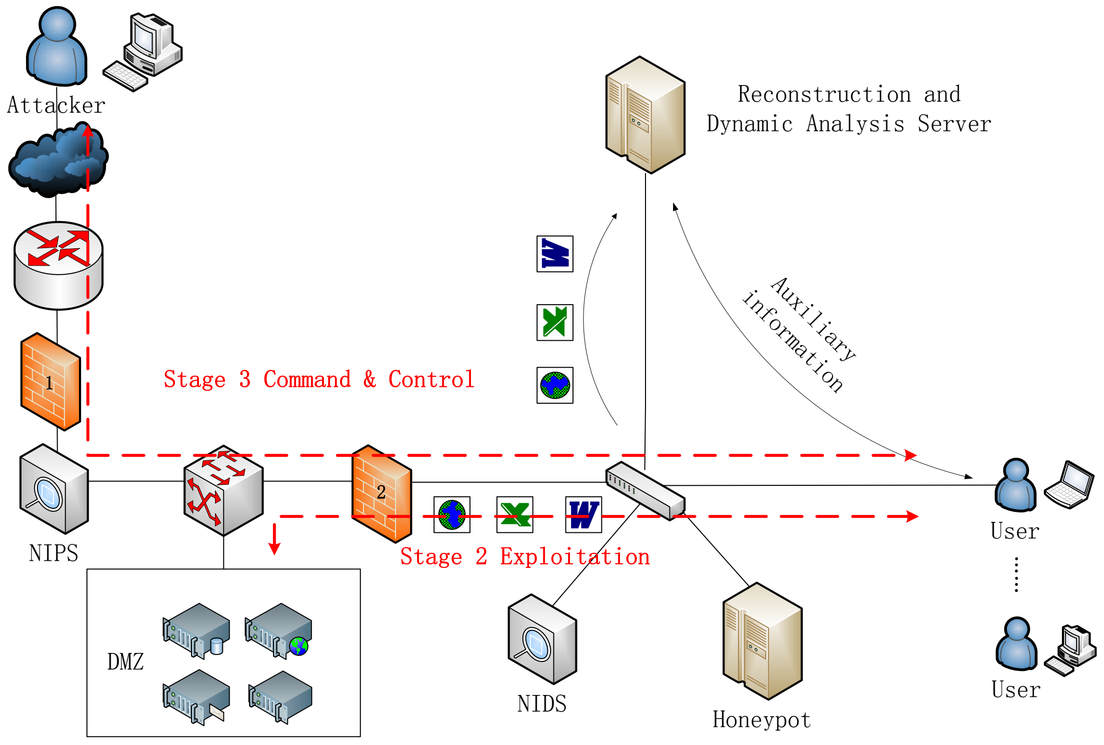

Figure 3. Detailed Deployment of the Framework in Actual Network

At stage 2, the attackers will start their plan. Choosing Plan A, the attackers may exploit SQL injection vulnerability to gain the administrator privilege of web application, and then tamper the index page with inserting elaborate code. Due to the web server is frequently accessed by the employees of the organization, the attack will easily succeed. Choosing Plan B, the attackers need send an email with malicious content to a specific employee. Through society engineering, the employee may open the document attached in the email which could lead to the malware embedded in the document being execurted. The attack process in stage 2 can bypass all defense measures deployed: 1.the network packet sent to the web server is permitted by the first firewall, 2.the SQL injection can bypass the NIPS through code obfuscation, 3.the request to the url is initiated by employees, so it has nothing to do with the honeypot and the second firewall, and exploitation in Plan A belongs to unknown attack with normal behaviors which NIDS is not able to detect, 4.the exploitation in Plan B also belongs to unknown attack which NIDS and anti-virus products are not good at,5 the malware downloaded from the Internet or released from crafted documents can be obfuscated skillfully to bypass any anti-virus installed in employees' hosts.

At stage 3, the malware such as RAT has already run in the host, the RAT may connect to the C&C to get commands or upload sensitive information through HTTP tunnel which can pass through two firewalls and evade the detection of NIDS/NIPS, honeypot again has nothing to do in this stage.

From the above, we can conclude that traditional defense measures do a little effort against APT attacks, that's why Stuxnet, etc. still can succeed in heavily armed network environment ignoring all of defense measures. With the proposed framework in this pater, we can detect these attacks. As last section say, we focus on the latter two stages. The main functionality is integrated into the Reconstruction and Dynamic Analysis Server (RDAS).RDAS is a super server which can keep the mirror of all active hosts in the inner-network.

Stage 2 Detection

In stage 2, when web application been accessed, the malicious html page is delivered to RDAS except for been delivered to the employee host. With the help of user agent module in the employee host, RDAS can simulate the context of employee host. If the malicious html page is explained by Internet explorer 11, the corresponding mirror in RDAS will use the same browser to explain the malicious html page. Similarly, after malicious .doc document in the email was delivered, if the .doc document is opened by Microsoft word 2013, the same as the mirror do. The translation from network traffic to application dataflow is done by the reconstruction module in RDAS, then dynamic analysis module will work. As said in section 4 dynamic analysis can identify the malicious behaviors hidden in the network traffic. After decision, RDAS may issue an alert and send signature of attack extracted by conclolic execution to NIPS.

Stage 3 Detection

In stage 3, RDAS can inspect the network traffic to identify suspicious network connections. Normal network connections generally initiate by users, if there are lots of network traffic between a host and an outside address, RDAS will check if there exists user activities in the host the user agent will return the relational connection information. If the network connection is hidden in user mode or the process which the connection affiliates is hidden, the connection is very suspicious. For example, the attacker installs RAT in the host after exploitation, the communication between attacker and the host masquerades http request and response whose contents are encrypted, so that the network connection can pass through the firewalls and bypass the NIDS/NIPS, but when RDAS receives these network packets, it finds that (through user agent module) the http network packets don't come from the browser processes, or they come from browser process which can't be seen on the desktop, we can get a conclusion that the host is very likely under control by the attacker.

6. Related Works

Most of the research is focus on malware analysis, including static analysis [13-15], dynamic analysis [16-20] and hybrid analysis [21,22]. For APT detection, in the paper [22], the authors extend the malware target recognition architecture initially proposed in [21] to an operational model for organization self-discovery of malware with low effective scan times and low false positive rates through successive data reduction and analysis. But in actually detecting malicious files is the key point in APT detection instead of detecting malware, which we are focus on.

In [23,24], the authors proposed an analytical security model considering the security analytics using Big Data. Their architecture is directed towards dealing with operational concerns in security organizations that aim to use existing security tools with Big Data analytics. Since their work is aimed towards operational side of security analytics therefore, it does not demonstrate any methodology of practical analysis of security threats as compared to our framework.

Ussath, Martin, Feng Cheng, and Christoph Meinel [25] proposed a Security Investigation Framework (SIF) that needs to process all investigation relevant information from different sources like prefiltered log files or forensic reports with novel correlation algorithms and rules. But this approach is prone to lead false alarm without semi-manual creation and the utilization of multiple information sources which could be a hard work.

7. Conclusion

Stuxnet, Operation Aurora, Duqu, Flame, Red October, Miniduke have a much greater impact on network defense systems due to their sophisticated exploitation and their ability to evade detection. Traditional defense measures could not effectively detect the unknown attacks and passive attacks which are common in APT attacks.

The proposed framework in this paper is much more effective in detecting APT compared with traditional defense measures, especially in detecting passive attacks. There are five main components in the framework: network traffic redirection module, user agent, reconstruction module, dynamic analysis module and response module. In the framework, dynamic analysis is the key module to detect known/unknown attacks. Dynamic analysis has a natural advantage in detecting malicious behaviors which can reveal the real intension in the files. Using the concolic execution in dynamic analysis module, we can accurately detect the attack and extract signature of the attack automatically.

In the example illustrated in Figure 3, we provide a typical APT attack process together with corresponding detecting process, i.e. how RDAS works. Through the example, we demonstrate the efficiency of the framework proposed to detect APT attacks.

References

[1] J. Andress, “Advanced Persistent Threat, Attacker Sophistication Continues to Grow?” ISSA Journal. **(2011)**.
[2] Mandiant, M Trends, “The Advanced Persistent Threat”. **(2010)**.
[3] National Institute of Standards and Technology, Managing Information Security Risk: Organization, Mission, and Information System View. Gaithersburg, National Institute of Standards and Technology, http://fismapedia.org/index.php?title=NIST_SP_800-39_Appendix_B.
[4] https://en.wikipedia.org/wiki/Advanced_persistent_threat.
[5] N. Virvilis, D. Gritzalis, “The Big Four-What We Did Wrong in Advanced Persistent Threat Detection?”, Availability, Reliability and Security (ARES), Eighth International Conference on IEEE, **(2013)**.
[6] F. Li, A. Lai, Ddl Ddl , “Evidence of Advanced Persistent Threat: A Case Study of Malware for Political Espionage”, 6th International Conference on Malicious and Unwanted Software, **(2011)**.
[7] James C. King, “Symbolic execution and program testing”, Communications of the ACM. vol. 19, no. 7, **(1976)**, pp. 385-394
[8] C. Cadar, D. Dunbar, and D. R. Engler. “KLEE: Unassisted and Automatic Generation of High-Coverage Tests for Complex Systems Programs”, OSDI. vol. 8, **(2008)**, pp. 209-224.
[9] C. Vitaly, V. Kuznetsov, and G. Candea. “S2E: a platform for in-vivo multi-path analysis of software systems”, ACM SIGPLAN Notices 46.3, **(2011)**, pp. 265-278.
[10] G. Patrice, N. Klarlund, and K. Sen. “DART: directed automated random testing”, ACM Sigplan Notices. vol. 40, no. 6, **(2005)**, pp. 213-223.

[11] K. Sen, D. Marinov, and G.Agha. "CUTE: a concolic unit testing engine for C", ACM SIGSOFT Software Engineering Notes. vol. 30, no. 5, **(2005)**, pp. 263-272.
[12] P. Godefroid, M. Y. Levin, and D. A. Molnar. "Automated Whitebox Fuzz Testing", NDSS. vol. 8, **(2008)**, pp. 151-166.
[13] T. Abou-Assaleh, N. Cercone, V. Keselj, and R. Sweidan, "N-gram based detection of new malicious code", Computer Software and Applications Conference, **(2004)**.
[14] O. Henchiri and N. Japkowicz, "A feature selection and evaluation scheme for computer virus detection", IEEE 6th Int. Conf. Data Mining, **(2006)**.
[15] J. Kolter and M. Maloof, "Learning to detect and classify malicious executables in the wild", Journal of Machine Learning Research. **(2006)**, pp. 2721-2744.
[16] M. Bailey, J. Oberheide, J. Andersen, and Z. Mao, "Automated classification and analysis of Internet malware", International Workshop on Recent Advances in Intrusion Detection, **(2007)**.
[17] M. Christodorescu, S. Jha, and C. Kruegel, "Mining specifications of malicious behavior", the 1st India Software Engineering Conference, **(2008)**.
[18] A. Dinaburg, P. Royal, M. Sharif, and W. Lee, "Ether: Malware analysis via hardware virtualization extensions", 15th ACM Conf. Comput. Commun. Security, **(2008)**.
[19] T. Lee and J. J. Mody, "Behavioral classification", EICAR, **(2006)**.
[20] A. Moser, C. Kruegel, and E. Kirda, "Limits of static analysis for malware detection", ACSAC, **(2007)**.
[21] T. Dube, R. Raines, G. Peterson, K. Bauer, M. Grimaila, and S. Rogers, "Malware target recognition via static heuristics", Comput.er & Security, vol. 31, **(2011)**, pp. 137-147.
[22] Thomas E. Dube, Richard A. Raines, Michael R. Grimaila, "Malware Target Recognition of Unknown Threats", IEEE Systems Journal. vol. 30, no. 5, **(2013)**, pp. 467-477.
[23] J. Howes, J. Solderitsch, I. Chen & J. Craighead, "Enabling trustworthy spaces via orchestrated analytical security", CSIIRW, **(2013)**.
[24] S.-H. Ahn, N.-U. Kim, and Tai-Myoung Chung. "Big data analysis system concept for detecting unknown attacks", ICACT, **(2014)**.
[25] U. Martin, F. Cheng, and C. Meinel. "Concept for a security investigation framework", NTMS, **(2015)**.