Scalable Approach Towards Discovery of Unknown Vulnerabilities

Umesh Kumar Singh[1], and Chanchala Joshi[2]
(Corresponding author: Chanchala Joshi)

School of Engineering and Technology, Vikram University Ujjain[1]
Institute of Computer Science, Vikram University Ujjain[2]
University Road, Madhav Bhavan, Near Vikram Vatika, Ujjain, Madhya Pradesh 456010, India
(Email: chanchala.joshi@gmail.com)
(Received Apr. 8, 2017; revised and accepted July 2, 2017)

Abstract

Of all the hazards confronting enterprise IT systems, zero-day vulnerabilities are among the most harmful. Zero-day vulnerabilities are flaws that leave users exposed to network attacks before a patch or work around is available. Every day an exploit remains unpatched, our risk of a data breach increases dramatically. Only a multi-layered approach that fully integrates with organization's IT defense stands a chance of stopping them. This paper presented a novel hybrid three layer architecture framework for zero-day attack detection and risk level assessment with respect to likelihood of exploits. The first layer of the proposed framework is liable to detect the unknown vulnerability which is based on statistical, signature and behavior based techniques; the second layer focuses on risk measurement; and the third physical layer contains centralized database and centralized server that are used during processing of first two layers. The proposed framework is analyzed in network environment of Vikram University Ujjain, India in order to evaluate the performance; experimental results show detection rate of 89% with 3% false positive rate.

Keywords: Attack Graphs; Attackrank; Intrusion Detection; Vulnerability Analysis; Zero-day Attacks

1 Introduction

In today's network system, organizations have taken great care to secure their network but even with responsible and sustained investment in defenses, they're still at risk. Attackers can bypass organization security through unknown vulnerabilities that are not listed by security persons. In a well-guarded network, a loophole may reveal by the persistent probing of a determined hacker. Attackers can leverage vulnerabilities present in network configuration to penetrate the target network [11, 21]. Besides known vulnerabilities, attackers find a zero-day through hours, weeks or months of painstaking effort through lines of code, to find some weakness, some flaw that methodically barrages the target application, for which even developers are not aware of. Attackers can force the network to reveal a small crack in the defense that provides them access to secretly execute their code. This is how a network is breached trough zero-day.

Zero-day is a vulnerability that has previously unknown and unpatched and therefore can be exploited by a threat actor to gain entry to a target network. Cyber criminals are increasing the success rate of attacks by finding and exploiting Zero-day vulnerabilities. In most of the cases, information about vulnerability is not available until attacks have already taken place. As a result, attacks using zero-day exploits are hard to identify and analyze. With zero-day vulnerability in hand, the hacker has a choice that, he may either help the software vendor by providing them information about the discovered vulnerability or sell it to the black market broker who may further sell the invented exploit at highest rate. Zero-day exploits have an element of surprise as they are previously unrevealed; an attacker incorporates the zero-day exploit into their charted list of vulnerabilities and once the penetration program process and payload is concocted, attack is launched.

There is actually no protection against zero-day when the attacks were first observed. Traditional security approaches discover the vulnerabilities by generating signatures, but in case of zero-day this information is unknown, so it is extremely difficult to detect zero-day with traditional defenses [18]. Attackers are highly skilled therefore their discovered vulnerability remains unknown to public for months or even years, which provides plenty of time to attacker to cause irreparable harm [20, 22]. According to FireEye [2], a typical zero-day attack may last for 310 days on average. Therefore, dealing with zero-day is clearly a challenging task.

This paper presents a three-layer, two-phase architecture framework for zero-day attack detection and analysis.

The framework consists of three layers: zero-day attack path generator, risk analyzer and physical layer. The first layer is liable to detect the unknown vulnerability; the second layer is an analyzer layer, which is assigned to analyze the generated attack and the third layer is physical layer which consists of centralized database and centralized server that are used during processing of first two layers.

The proposed framework performs two-phase working and follows a probabilistic approach for identification of the zero-day attack path and further to rank the severity of identified zero-day vulnerability. During first phase an attack graph is built from captured network scenario at any time stamp by levering the favorable attack conditions collected from various information sources. These conditions represent the abnormal system and network activities that are noticed by security persons or security sensors such as Intrusion Detection Systems. Based on the generated attack graph during first phase, second phase discovered the most probabilistic hosts by the proposed AttackRank algorithm to rank the severity of discovered vulnerability. In the proposed layered approach, once the basic network graph is generated, layers are supposed to execute dedicated functionality in parallel. Parallel work of each layer improves the performance of our proposed approach.

2 Related Work

Many research groups have proposed various methodologies to protect against Zero-day attacks. These methods are classified as statistics, signatures and behavior techniques [19]. Statistical based zero-day detection approaches [7] cannot be applied for real-time instantaneous detection and protection. They relying on static attack profiles therefore require a manual modification of detection settings. Signatures based techniques are broadly used yet they need an improvement to generate high class signatures. Kaur and Singh [8] proposed a hybrid approach for identification of zero-day although it is applicable only for polymorphic warm detection. In this paper we proposed a probabilistic approach for detection of zero-day attacks. The proposed approach integrates the signature based and behavior based methods of zero-day identification. The next part of our work focuses on measuring the risk level of identified malicious activity.

In the field of vulnerabilities's risk level calculation many researcher have made attempts to measure security risks of vulnerabilities [6, 17]. In our previous work we provide an approach for measuring the risk level of vulnerabilities using Hazard metric [15] with the involvement of frequency [16] and impact [14] factors. However, zero-day attacks risks level measurement is like measuring an immeasurable. We cannot measure the severity of vulnerability while it is not known. Therefore, we are considering the degree of exploitability, while measuring the risks of zero-day. The approach is based on link analysis algorithm [3] used for personalized web. We made an assumption that, an attacker can only advance his attack position to a node that has connectivity and vulnerability to be exploited. The proposed framework provides a method for zero-day detection and estimates the likelihood of system being intruded by attacker.

3 Proposed Approach

A three-phase architecture of Zero-day attack analysis is proposed in the paper which is shown in Figure 1. The architecture consists of three layers: zero-day attack path generator and risk analyzer. The first layer is liable to detect the unknown vulnerability, the second layer is an analyzer layer, which is assigned to analyze the generated attack graph in order to measure the risk of vulnerability and the third layer is physical layer, consists of centralized database and centralized server.

In the proposed three-layer architecture layers are supposed to execute dedicated functionality in parallel. Parallel work of each layer improves the performance of proposed approach.

3.1 Zero-day Attack Path Generator Layer

The first layer of proposed framework is liable to detect the unknown vulnerability. The main components of first layer are: Snort anomaly detector, attack-graph generator, detection engine and zero-day attack path generator. To capture intrusion propagation, we first build the attack graph by capturing the network scenario at specific timestamp. The attack graph is generated by sensing the anomalous behavior or abnormal activities of network that are noticed by security persons or security sensors such as IDS. These anomalies represent the probability of host being infected in an attack graph. Detection engine analyses the mysterious anomalous activities in parsed attack-graph that could be an attack and suspicious activities are preserved as zero-day exploit.Zero-day attack path generator layer consist of four modules:

3.1.1 Parsing and Dependency Extraction trough Snort

The purpose of this module is to detect and filter known attacks from the captured network scenario, which is done through parsing by defining a set of malicious behavior rules that are set up or configured by an administrator. By establishing a good network profile, it is easier to identify anonymous bad behavior. For this purpose, Snort 2.9.7.6 is programmed as an anomaly detector. Snort is used to detect and filter the known attacks, by implementing a good network setup [9,12]. The packets that match with the Snort profile are known attacks and after storing their information in centralized database these packets are discarded. The packets that are partially matched or not matched are forwarded to next step for further analysis.

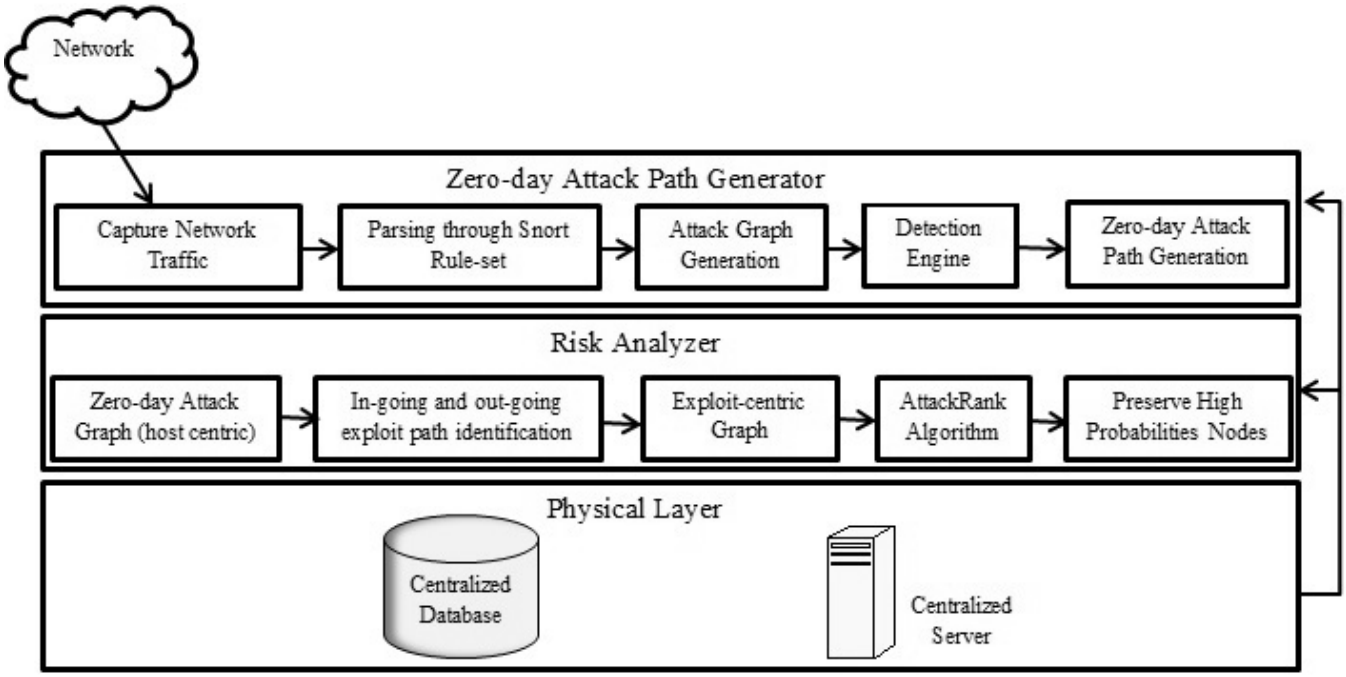

Figure 1: Proposed three layers architecture for zero-day attack detection and analysis

3.1.2 Attack Graph Generation

After filtering the known attacks through parsing, further analysis is done by Snort tagger on extracted mysterious packets which didn't trigger any alert. Tagging packets is a way to continue logging packets from a session or host that generated an event in Snort [13]. Tagged traffic is logged to allow analysis of response codes and post-attack traffic. The function of tagger is to monitor the traffic, tag the packets and send it to detection engine for further analysis. The tagger creates a new identifier based on 16-bit hash of a packet. The tag value and label for the filtered packet is stored in a table $< Tag, Label >$ for later use. The Tag value is calculated for the 6 attributes (arvl_time, source_ip, destination_ip, source_port, destination_port, protocol). The tag value is stored for later use and an attack graph of extracted nodes and mysterious conditions is generated in this module.

3.1.3 Detection Engine

The parsing module is not able to respond to an unknown attack; therefore run-time analysis is performed by Snort NIDS (Network Intrusion Detection System) to monitor network traffic in order to detect suspicious activity, which could be an attack or unauthorized activity. Detection engine receives the parsed packets, compares them with existing good traffic and detect unknown observations. The good traffic is the collection of safe machines on which all possible security mechanisms are applied. Security privileges and policies are defined for these safe systems and they do not participate in any malicious activity. A trust value has been assigned to these machines based upon the past experience. Algorithm 1 explains the procedure of detection engine.

Algorithm 1 zero_day_detection

```
1: Begin
2: Capture network_scenario.
3: while Not end of packet in network_scenario do
4:    if (equals(packet_content,snort_rules)) then
5:       drop current_packet;
6:    else
7:       preserve filtered_packet ⇐ current_packet;
8:    end if
9:    tag ⇐ hash(preserve filtered packet (arvl_time,
      source_ip, destination_ip, source_port, destina-
      tion_port, protocol))
10:   update_database(tag);
11:   tagged_packet ⇐ preserve filtered_packet
12:   if ( NOT ( is Malicious (tagges_packet)) ) then
13:      Capture good traffic network scenario from safe
         systems
14:      Extract features and update Snort NIDS
         database
15:   else
16:      unknown ⇐ tagges_packet;
17:      insert unknown;
18:      update zero_day_database(unknown);
19:   end if
20: end while
21: End
```

3.1.4 Zero-day Attack Graph Generator

In an attack graph, each node represents a host behavior at specific timestamp. For example, let us assume that the network scenario, captured in a time window T[$tbegin, tend$] is denoted as $\sum T$ and the set of hosts involved in $\sum T$ is denoted as O_T , then the attack graph is a directed graph G(V, E),where:

- V is the set of nodes that represent hosts in a given time window.
- E is the set of directed edges that represent conditions or dependencies.
- If an attacker at given timestamp navigates in between two hosts $out_i, in_j, i, j >= 1$, and a dependency relation $dep_c : out_i -> in_j$, where out_i is the i^{th} instance of host $out \in O_T$, and in_j is the j^{th} instance of host $in \in O_T$, then $V = V \cup \{out_i, in\}, E = E \cup \{dep_c\}$.

Figure 2 depicts an attack graph.

3.2 Risk Analyzer Layer

The first phase of our proposed methodology built an attack graph as chains of possible vulnerability exploits, which can help security persons to locate security flaws. The second phase of the proposed framework focuses to rank the nodes of attack graph based on likelihood of an attacker reaching these states. The ranking determines more vulnerable attack paths which require more immediate attention for network security. The proposed approach is based on link analysis algorithm used for personalized web [3]. We are considering three prominent attributes: Attack Vector, Attack Complexity and Authentication, of severity matrix [10] while determining the exploitability of vulnerability.

Attack Vector: Access vector represents difficulty from the access location (e.g. local, network accessible or remote) required to exploit the vulnerability. The more remotely an attacker can exploit the vulnerability, the greater the exploitability value will be.

Attack Complexity: It indicates the level of effort required to exploit the vulnerability after an access to the target point is gained. It's range in between low, medium and high. For example, a Denial of Service in a network has low complexity since the vulnerability can be exploited once an attacker gains access of the network. The lower the complexity is, the higher the exploitability will be.

Authentication: Authentication is defined to measure the number of authentications required (e.g., multiple instances, single instance or no instance) before network vulnerability can be exploited.

The assumption behind measurement of likelihood is that a highly exploitable vulnerability is more likely to be misused by attackers and consequently should have a higher frequency.

3.2.1 AttackRank Algorithm

To reveal attack paths of the higher risks zero-day vulnerabilities from the massive attack graph, the nodes with high probabilities are to be preserved, while the link between them should not be broken. We implemented an AttackRank algorithm based on the PageRank algorithm [1] to tag each node in the attack graph as either possessing high probability itself, or having both an ancestor and a descendant with high probabilities. The tagged nodes are the ones that actually propagate the infection through the network, and thus should be preserved in the final graph. Proposed AttackRank algorithm is based on PageRank and measures the likelihood of exploit in an attack graph. However network attacker behavior is different than web surfer behavior in a manner as during an attack, an attacker has options to continue or quit attacking on a current path (because of security privileges and policies it is too hard to lead to his goal). AttackRank algorithm made an assumption as if the attacker dump the current attack path then he will find an alternative path by backtracking (from one of the set of previous states) and if he continues attacking then he will attempt to each of the possible navigational states with a probability based on how hard its vulnerabilities can be exploited. With this assumption we proposed an AttackRank algorithm to find frequency of exploit.

Algorithm 2 AttackRank algorithm for likelihood detection of exploit

```
Begin
Initialize the Graph AttackRank G(V,E)
I: set of initial states V
while u, v ∈ V do
  if u ∈ inlink(v) then
    attackrank(v)⇐ attackrank(in(u))/ out(v)
  else
    if v ∈ out(v) then
      attackrank(v)⇐ 1 - (attackrankin((u))/ out(v))
    else
      attackrank(v)⇐ 1
    end if
  end if
end while
End
```

3.3 Physical Layer

The third layer of the proposed framework, the physical layer contains centralized database and centralized server that are used during processing of first two layers. All of the information (malicious or non-malicious activities, known or unknown exploits) is stored in the database server of physical layer. Database is continuously updated by the records in the audit network data repository that do not yet have any sort of context profile.

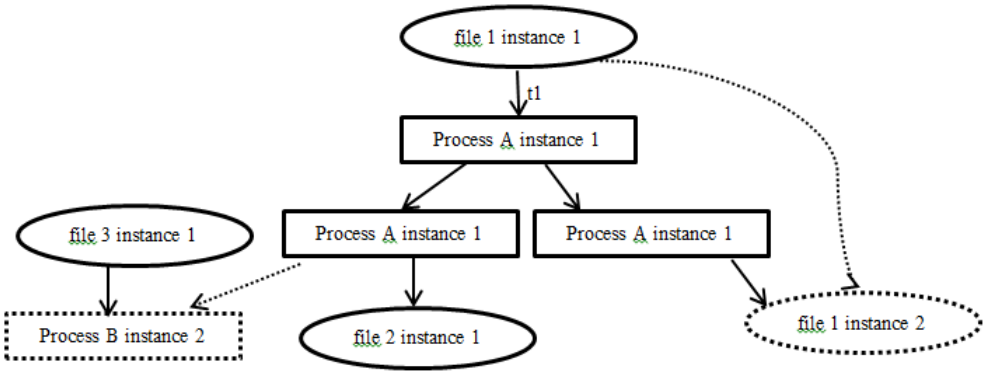

Figure 2: An attack graph

4 Experimental Setup

The experimental analysis of the proposed model has been conducted in the network of Vikram University, India campus. Vikram University campus consists of diverse multi-disciplinary departments, has a network of more than 500 computers, providing connectivity to different users in various institutes and hostels. Overall structure of the campus network is shown in Figure 3,

To test performance of proposed framework, a group of 7 hosts, playing miscellaneous roles are selected from the campus network that forms the testbed for this case study. Testbed is comprised of hosts in diverse physical locations (as shown in Figure 4 that includes network server located at academic block within the contact range of firewall (208.91.191.121), server located at School of Engineering and Technology (128.168.1.4), and other machines. The structure of testbed is shown in Figure 4.

The External scan is done through a router or firewall by the means of Nessus [4] vulnerability scanner. The internal scan took place at the School of Engineering and Technology (SoET) location, and was plugged into a server that resides inside Vikram University network. In Figure 4 the placement of the blue scanner is inside the firewall, so it can scan internal vulnerabilities and the red scanner is used for external vulnerabilities scan. These internal and external vulnerability scans are used to collect data to assess the effectiveness of current security measures taken at the Vikram Universitys network. The internal scan took place at the School of Engineering and Technology (SoET) location, and was plugged into a server that resides inside Vikram University network. The objective is to avoid external security counter measures to get a detailed view at system configurations. The external scan is for determining the security posture through Internet users view. The point behind external scanning is to identify what a hacker would see if he were trying to probe Vikram University network.

To measure the performance of proposed work four standard terminologies TP, FP, FN and TN were used, True Positive (TP) means the number of correctly identified malicious codes; True Negative (TN) refers to the number of correctly identified benign codes, means the non-malicious code that is classified as genuine code. FN and FP refers to misjudgments: False Positive (FP) shows that the alarm is generated when there is no actual attack, means the number of incorrectly identified trusted code as malicious code. False Negative (FN) is when the system fails to detect the malware activity due to it being similar to expected activity or no signature being available in the database. Table 1 summarizes the possible cases of classification scheme.

The rates of TP, FP, FN and TN will be computed by using four standard metrics to evaluate the performance of our technique: *True Positive Rate (TPR):* This is the percentage of correctly identified malicious codes which is measured as the ratio between the number of events that have accurately classified as positive and the total number of events that can be classified as positive, which is given by:

$$TPR = (|TP|)/(|TP| + |FN|)$$

False Positive Rate (FPR): This is the percentage of wrongly identified benign codes, measured as the ratio between the numbers of events that were considered positive on the number of events that should have been negative, which is given by:

$$FPR = (|FP|)/(|FP| + |TN|)$$

False Negative Rate (FNR): This is the rate of incorrectly rejected malicious code.

$$FNR = (|FN|)/(|TP| + |FN|)$$

True Negative Rate (TNR): This is the percentage of correctly identified benign codes.

$$TNR = (|TN|)/(|FP| + |TN|)$$

Accuracy, Precision and Recall will be used to evaluate the filtering accuracy of zero-day attack path. Accuracy is used to evaluate the accuracy of the classification results,

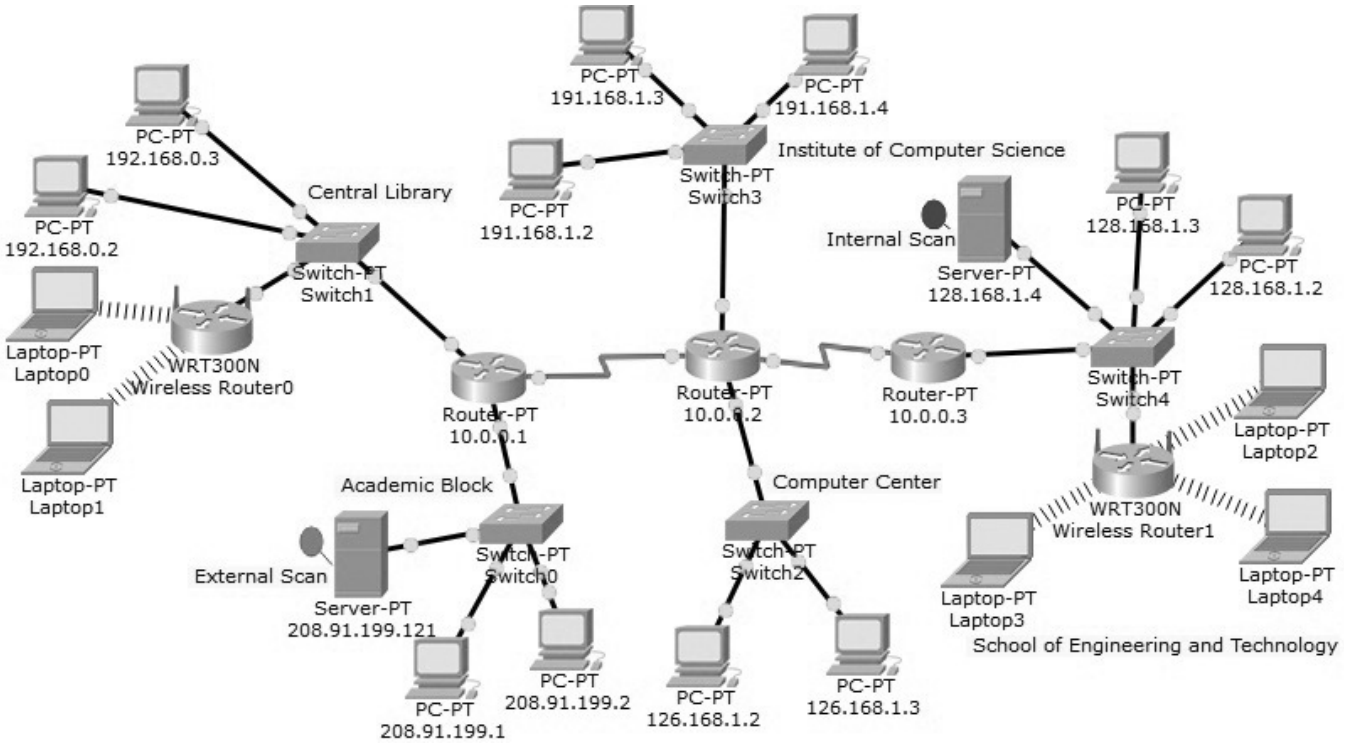

Figure 3: Network architecture of Vikram University, India campus

Table 1: Judgment cases

Classification	Malicious Code	Non-Malicious Code
Malicious Code	True Positive (TP)	False Positive (FP)
Non-Malicious Code	True Negative (TN)	False Negative (FN)

namely, the proportion of the malicious codes that are accurately classified to their own categories. It is calculated through the following formula:

$$Accuracy = (|TP| + |TN|)/(|TP| + |FP| + |FN| + |TN|)$$

Precision is the positive detection value which measures the effectiveness of zero-day detection system. Precision is used to evaluate the proportion of malicious codes among all the network activities that are judged to be malicious in nature. It is calculated through the following formula:

$$Precision = (|TP|)/(|TP| + |FP|)$$

Recall is used to evaluate the proportion of the malicious code that are accurately classified as malicious

$$Recall = (|TP|)/(|TP| + |FN|)$$

F-measure, is the harmonic mean of precision and recall, is adopted as one of the measuring indexes of the filtering mechanism, and calculated as:

$$Fmeasure = 2\times(Precision\times Recall)/(Precision+Recall)$$

F-measure is thus a means of evaluation that could combine precision and recall effectively.

Receiver Operating Characteristics (ROC) Curve is a very popular technique for measuring the relationship between the TP and FP rates of the anomaly detection system. The ROC curve uses a function of the FP rate on which the TP rate is plotted for different points. Closer the value off ROC area is to 1, it is good and when it is closer to 0.0, it is poor.

5 Evaluation of Proposed Model

In this section, we designed performed experiments to confirm the accuracy and efficiency of the proposed method. We built a test-bed network and launched an attack towards it. Since zero-day exploits are not readily available, we emulate zero-day vulnerabilities with known vulnerabilities. E.g., we treat CVE-2016-5387 as zero-day vulnerabilities by assuming the current time is Dec 31, 2015. The strategy of emulation also brings another benefit. The information for these known zero-days vulnerabilities can be available to verify the correctness of our experiment results.

The basic components of the testbed (Figure 4) are 2 servers for network vulnerabilities scan. 208.91.199.121 performs the external scanning through a router or fire-

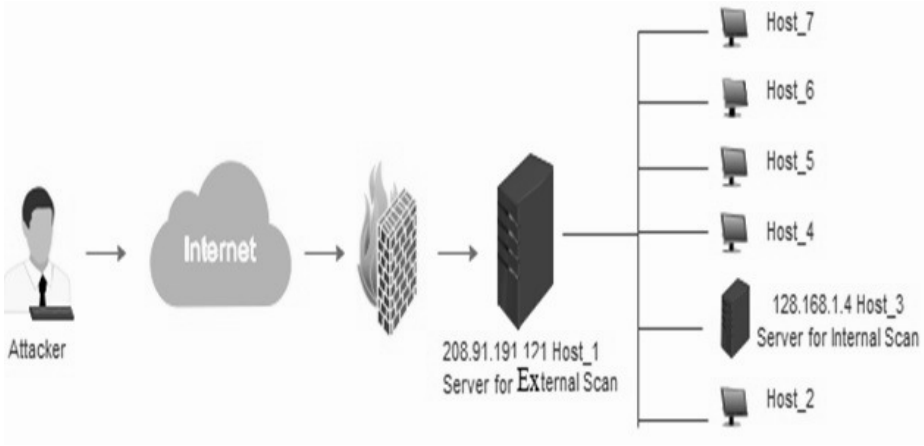

Figure 4: Experimental setup of the testbed

wall, by the means of the Nessus vulnerability scanner. Nessus placed within contact range of University, and generates details about active services, credentials and successful attacks. Scanning activities result that the server 208.91.199.121 has two open ports, tcp80 listening to HTTP traffic and tcp22 listening to SSH traffic. The SSH connection allows system administrators to do maintenance work remotely from within the subnet administration. The SSH service has vulnerabilities CVE-2012-5975, CVE-2014-6271 and CVE-2015-5600. CVE-2012-5975 allows remote attackers to bypass authentication via a crafted session involving entry of blank passwords; CVE-2015-5600 does not properly restrict the processing of keyboard-interactive devices within a single connection, which makes it easier for remote attackers to conduct brute-force attacks or cause a denial of service; and CVE-2014-6271 allows remote attackers to execute arbitrary code via a crafted environment. HTTP service has vulnerabilities CVE-2016-5387 and CVE-2015-3183. CVE-2016-5387 allows remote attackers to redirect an application's outbound HTTP traffic to an arbitrary proxy server via a crafted Proxy header in an HTTP request; and CVE-2015-3183 allows remote attackers to conduct HTTP request smuggling attacks via a crafted request. Both of these HTTP service vulnerabilities are present in the Apache HTTP Server.

We have also generated an attack environment that contains real exploit code as well as normal network traffic to web servers. Strong efforts were undertaken to make environment as realistic as possible. Acunetix, Nexpose and Metasploit [5] were used for internal scanning of vulnerabilities. Polymorphic engines ADMmutate, clet, Alpha2, CountDown, JumpCallAdditive and Pex were applied to the unencrypted exploits. True Positive Rate (TPR), False Positive Rate (FPR) and Receiver Operating Characteristics (ROC) Curve parameters are used to evaluate performance and accuracy of proposed layered architecture. Figure 5 and Figure 6 represent true detection rate and false positive rate of Zero-day attack.

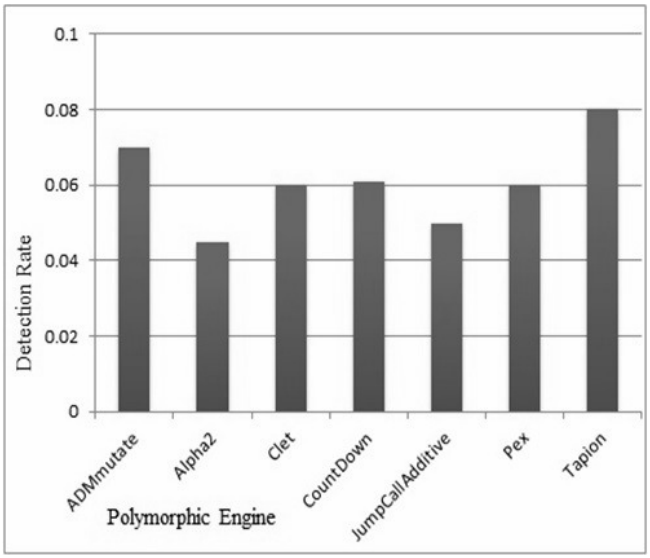

Figure 5: True positive rate

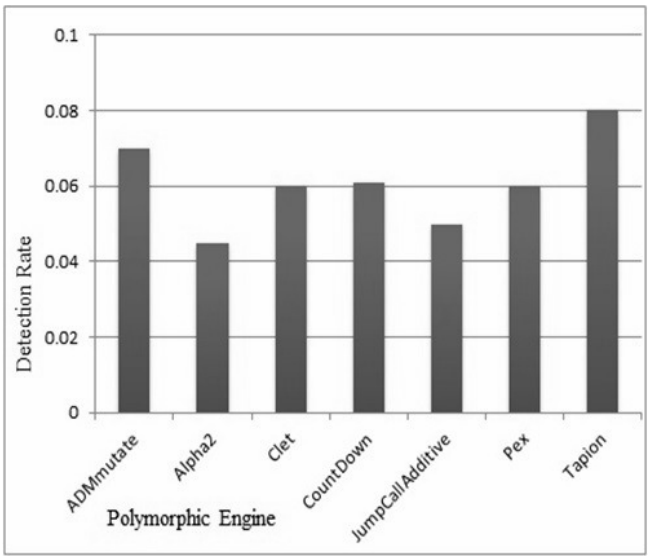

Figure 6: False positive rate

Figure 7 represents ROC curve. It is drawn by taking the average value of TPR, and in the figure, it is clearly shown that ROC is closer to 1 which proves the efficiency of our proposed approach.

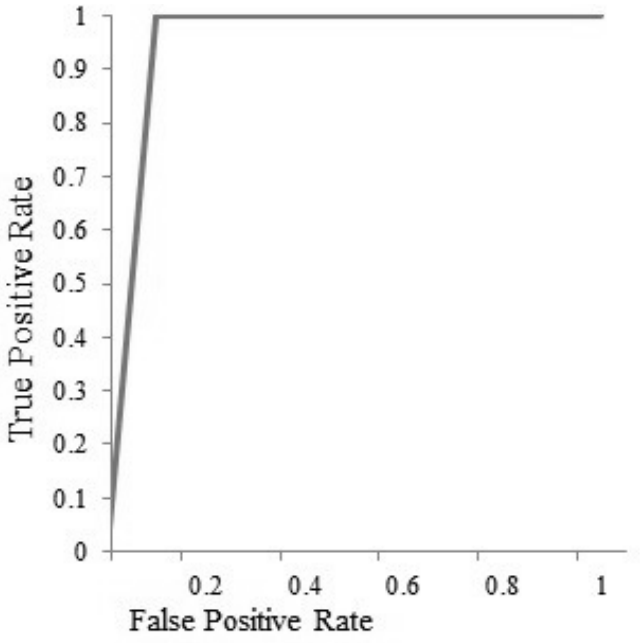

Figure 7: Average value of receiver operating characteristics curve

6 Conclusions

Zero-day vulnerability refers to the hole in the software or network system that is unknown to user and has no patch available. An attacker may take advantage of zero-day vulnerability by creating an exploit to gain access to the target network for stealing sensitive data, legal documents and other crucial information. This paper analyzed the lifecycle of zero-day vulnerabilities and proposed a three layer architecture framework for zero-day exploit identification and assessment; all three layers of proposed architecture have assigned specific functionalities to operate and execute in parallel to increase the performance. The first layer identifies the malicious activity that is not previously known, the second layer rank the identified vulnerability with respect to frequency of exploit, the third layer consists of database server and the centralized server. We designed our experiments to verify the efficiency of our proposed approach by using various standard parameters. In our experiments, it was observed that the best or truest detection rate was 89% and the false positive rate was 3%.

Acknowledgments

Authors are thankful to MP Council of Science and Technology, Bhopal, for providing support and financial grant for the research work.

References

[1] S. Brin, L. Page and R. Motwani, "The pagerank citation ranking: bringing order to the web," Technical Report, 1998.

[2] FireEye, "Zero-day danger: A survey of zero-day attacks and what they say about the traditional security model," Technical Report, 2015.

[3] C. Joshi and U. K. Singh, "A novel approach towards integration of semantic web mining with link analysis to improve the effectiveness of the personalized web," *International Journal of Computer Application*, vol. 128, pp. 1–5, Oct. 2015.

[4] C. Joshi and U. K. Singh, "Analysis of vulnerability scanners in quest of current information security landscape," *International Journal of Computer Application*, vol. 145, pp. 1–7, July 2016.

[5] C. Joshi and U. K. Singh, "Performance evaluation of web application security scanners for more effective defense," *International Journal of Scientific and Research Publications*, vol. 6, pp. 660–667, June 2016.

[6] C. Joshi and U. K. Singh, "Information security risks management framework- a step towards mitigating security risks in university network," *Journal of Information Security and Applications*, vol. 35, p. 128–137, June 2017.

[7] R. Kaur and M. Singh, "Automatic evaluation and signature generation technique for thwarting zero-day attacks," in *Proceedings of the Second International Conference (SNDS'14)*, pp. 298–309, Mar. 2014.

[8] R. Kaur and M. Singh, "Efficient hybrid technique for detecting zero-day polymorphic worms," in *Proceedings of IEEE International Advance Computing Conference (IACC'14)*, pp. 95–100, Feb. 2014.

[9] X. Liu and Y. Ye, "Intrusion detection system based on snort," in *Proceedings of the 9th International Symposium on Linear Drives for Industry Applications*, Lecture Notes in Electrical Engineering, vol. 272, Springer-Verlag, 2014.

[10] P. Mell and K. Scarfone, "CVSS: A complete guide to the common vulnerability scoring system version 2.0," Technical Report, 2007.

[11] A. A. Orunsolu, A. S. Sodiya, A. T. Akinwale, B. I. Olajuwon, M. A. Alaran, O. O. Bamgboye, and O. A. Afolabi, "An empirical evaluation of security tips in phishing prevention: A case study of nigerian banks," *International Journal of Electronics and Information Engineering*, vol. 6, no. 1, pp. 25-39, 2017.

[12] M. Roesch, "Snort- lightweight intrusion detection for networks," in *Proceedings of 13th Systems Administration Conference Seattle (LISA'99)*, pp. 229–238, Nov. 1999.

[13] V. Shah, N. Patel and K. Pancholi, "An analysis of network intrusion detection system using snort," *International Journal for Scientific Research and Development*, vol. 1, no. 3, pp. 410–412, 2013.

[14] U. K. Singh and C. Joshi, "Information security assessment by quantifying risk level of network vulnerabilities," *International Journal of Computer Application*, vol. 156, pp. 37–44, Dec. 2016.

[15] U. K. Singh and C. Joshi, "Quantifying security risk by critical network vulnerabilities assessment," *International Journal of Computer Application*, vol. 156, pp. 26–33, Dec. 2016.

[16] U. K. Singh and C. Joshi, "Quantitative security risk evaluation using cvss metrics by estimation of frequency and maturity of exploit," in *Proceedings of the World Congress on Engineering and Computer Science (WCECS'16)*, San Francisco, USA, Oct. 2016.

[17] U. K. Singh and C. Joshi, "Information security risk management framework for university computing environment," *International Journal of Network Security*, vol. 19, pp. 742–751, Sep. 2017.

[18] U. K. Singh, C. Joshi, and S. K. Singh, "Zdar system: Defending against the unknown," *International Journal of Computer Science and Mobile Computing*, vol. 5, pp. 143–149, Dec. 2016.

[19] U. K. Singh, C. Joshi, and S. K. Singh, "Zero day attacks defense technique for protecting system against unknown vulnerabilities," *International Journal of Scientific Research in Computer Science and Engineering*, vol. 5, pp. 13–18, Feb. 2017.

[20] D. Song, J. Caballero, T. Kampouris and J. Wang, "Would diversity really increase the robustness of the routing infrastructure against software defects?," in *Proceedings of the Network and Distributed System Security Symposium*, 2008.

[21] D. Stiawan, M. Y. Idris, A. H. Abdullah, M. AlQurashi, R. Budiarto, "Penetration testing and mitigation of vulnerabilities windows server," *International Journal of Network Security*, vol. 18, no. 3, pp. 501-513, 2016.

[22] S. Zhu Y. Yang and G. Cao, "Improving sensor network immunity under worm attacks: a software diversity approach," in *ACM Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing*, pp. 149–15, 2008.

Biography

Umesh Kumar Singh received his Doctor of Philosophy (Ph.D.) in Computer Science from Devi Ahilya University, Indore(MP)-India. He is currently Associate Professor in Computer Science and Director in School of Engineering & Technology, Vikram University, Ujjain(MP)-India. He has authored 6 books and his about 150 research papers are published in national and international journals of repute. He was awarded Young Scientist Award by M.P. council of Science and Technology, Bhopal in 1997. He is reviewer of various International Journals and member of various conference committees. His research interest includes Computer Networks, Network Security, Internet & Web Technology, Client-Server Computing and IT based education.

Chanchala Joshi received her Master of Science in Computer Science and Master of Philosophy in Computer Science from Vikram University, Ujjain(MP)-India. Currently, she is Junior Research Fellow and doctoral student in Institute of Computer Science, Vikram University, Ujjain(MP)-India. Her research interest includes network security, security measurement and risk analysis.