International Journal of Applied Engineering Research ISSN 0973-4562 Volume 13, Number 12 (2018) pp. 10861-10870
© Research India Publications. http://www.ripublication.com

An Enhanced Framework for Identification and Risks Assessment of Zero-Day Vulnerabilities

Chanchala Joshi*, Umesh Kumar Singh and Dimitris Kanellopoulos*****

**Institute of Computer Science, Vikram University, Ujjain, M.P. India*

***Institute of Computer Science, Vikram University, Ujjain, M.P. India*

****Department of Mathematics, University of Patras, Greece*

Abstract

Nowadays highly-skilled attackers can find the vulnerabilities of many networked applications. Meanwhile, the risk of a data breach increases dramatically as a software or application vulnerability always remains without a patch. By exploiting such vulnerability (called zero-day), hackers gain entry to the target network and can steal sensitive data. It is challenging to detect zero-day with traditional defenses because signature information in zero-day attacks is unknown. Consequently, a novel security solution is required that will discover zero-day attacks and estimate the severity of identified zero-day vulnerability. In our previous work [1], we proposed an approch for discovery of unknown vulnerabilities. This paper enhances the previous approch by presenting a framework that constitutes an integrated approach for detection and prioritization (based on likelihood) of zero-days attacks. The proposed framework follows a probabilistic approach for identification of the zero-day attack path and further to rank the severity of identified zero-day vulnerability. It is a hybrid detection-based technique that detects unknown flaws present in the network that are not detected yet. To evaluate the performance of the proposed framework, we adopted it in the network environment of Vikram university campus, India. The framework is very promising as experimental results showed detection rate of 96% for zero-day attacks with 0.3% false positive rate.

Keywords: Zero-day attacks • Exploit • Vulnerability analysis • Intrusion Detection • Attack graphs • AttackRank

INTRODUCTION

Current organizations and enterprises have taken great care to secure their networks. However, they are still at risk even with responsible and sustained investment in their defenses. This happens because attackers can bypass organization's security through unknown vulnerabilities, which are not listed by security persons. In a well-guarded network, a loophole may be revealed by the persistent probing of a determined hacker. Attackers can leverage vulnerabilities which are present in network configuration to penetrate the target network. Zero-day (0-day) is a software or application vulnerability that can be exploited by a threat actor to gain entry to a target network. In this way, hackers or attackers can steal sensitive information such as legal documents and enterprises data. Cyber criminals are increasing the success rate of attacks by finding and exploiting zero-day vulnerabilities. Regularly, the information about vulnerability is not available until zero-day attacks have already taken place. As a result, it is difficult to identify and analyze attacks that use zero-day exploits. With zero-day vulnerability in hand, a hacker has two possible choices:

- He may help the software vendor by providing him with information about the discovered vulnerability;
- He may sell the crucial information to the black market broker, who may further sell the identified 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. In particular, 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 its defense that provides them access to secretly execute their code. This is how a network is breached through zero-day attack.

Actually, there is no protection against zero-day when the attacks were first observed. Traditional security approaches discover the vulnerabilities by generating signatures, but in the case of zero-day, signature information is unknown. So, it is extremely difficult to detect zero-day with traditional defenses [2]. Attackers are highly-skilled and the discovered vulnerability can remain unknown to the public for months or even years. This fact provides plenty of time to attackers to cause irreparable harm [3, 4]. According to FireEye [5], a typical zero-day attack may last for 310 days on average. Therefore, dealing with zero-day is clearly a challenging task. Fig. 1 shows the timeline of zero-day vulnerability from discovery to patch.

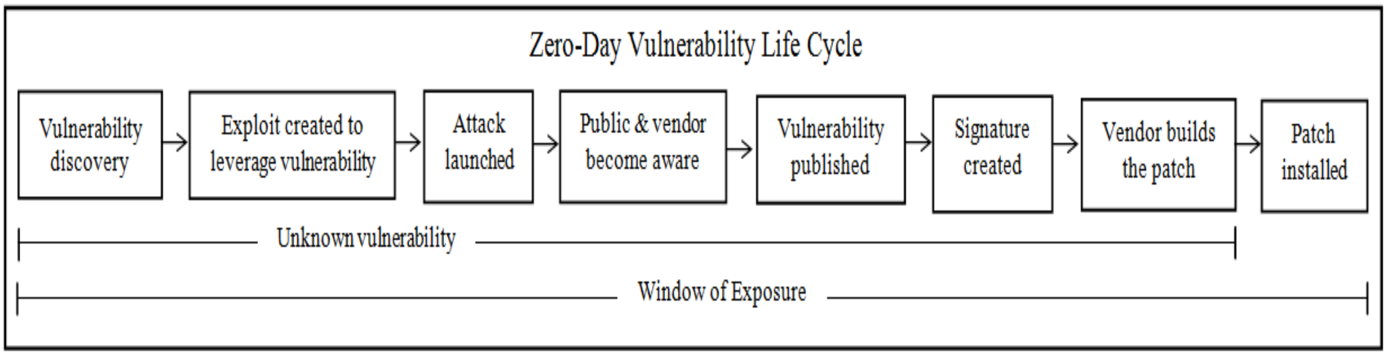


Figure 1. Timeline of zero-day attack

In our previous work, we presented an approch towards discovery of unknown vulnerabilities [1]. In this paper we are enhancing the previous proposed approch by presenting a layered architecture for detection and risks analysis of zero-day attacks. The architecture consists of three layers:

(1) Zero-day attack path generator: The first layer is liable to detect the unknown vulnerability.

(2) Risk analyzer: The second layer is assigned to analyze the generated attack.

(3) Physical layer: The third layer consists of a centralized database and a centralized server that are used during the information processing of the first two layers.

This framework was designed to block attackers by examining known malware samples. The proposed framework follows a probabilistic approach for identification of the zero-day attack path and further to rank the severity of identified zero-day vulnerability. Actually, it is a hybrid detection-based technique that detects unknown flaws present in network that are not detected yet. In particular, our framework performs the following tasks during two phases:

- During the 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 attack conditions represent the abnormal system and network activities that are noticed by security persons or security sensors (e.g., Intrusion Detection Systems).
- During the second phase, the nodes of the generated attack graph with higher risks of zero-day vulnerabilities are discovered. This task is accomplished by using an intelligent algorithm (called AttackRank) that ranks the severity of identified zero-day vulnerability.

The contributions of the proposed framework are:

(1) This framework is the first integrated approach for detection and prioritization (based on likelihood) of zero-days attacks.

(2) Known malicious packets are detected and filtered at initial stage. This task significantly manages the heavy network traffic and further avoids the unnecessary payload.

(3) The proposed AttackRank algorithm measures the risks of unknown, which helps in designing remediation plans.

The rest of this paper is organized as follows: In Section 2 we discuss related work, while in Section 3 we present our framework. In Section 4 we present the experimental setup, and in Section 5 we discuss the performance evaluation of our framework. Finally, Section 6 concludes the paper and gives directions for further research.

RELATED WORK

Vulnerabilities are the flaws in system configuration or in network by which an attacker may gain entry to the target network [6]. Many framework have developed for identification and risks assessment of known vulnerabilities [7]. However, discovery of unknown vulnerabilities is a big issue that is still unsolved. Some of the prominent methods to protect against zero-day attacks are classified as statistical-based, signature-based, behavior-based, and hybrid detection-based techniques [8].

- Statistical-based techniques generate attack profiles from past exploits that are now publically known. From these known exploits, the historical exploit's profile parameters are updated to detect new attacks [9]. Statistical-based zero-day detection approaches [10] cannot be applied at real-time instantaneous detection and protection. They are relying on static attack profiles, and thus they require a manual modification of detection settings.
- Signature-based detection method compiles a library of different malware signatures. These signatures are cross-referenced with local files, network files, email or web downloads depending on settings chosen by the user. These libraries are constantly being updated for new signatures that often represent the signatures

of new exploited vulnerabilities [11]. Signatures-based techniques are broadly used yet, and they need an improvement to generate high-class signatures.

- Behavior-based techniques sniff essential characteristics of worms to predict the future behavior of a web server, server or victim machine in order to deny any behaviors that are not expected [12]. These techniques can predict the flow of network traffic.
- Hybrid-based techniques overcome the weaknesses of the above mentioned techniques by using various combinations of them [13]. It is noteworthy that Kaur and Singh [14] proposed a hybrid approach for identification of zero-day although it is applicable only for polymorphic warm detection.

In works [15], [16], the authors introduced an approach for measuring the risk level of vulnerabilities using Hazard metric with the involvement of frequency [17] and impact [18] factors. However, the measurement of zero-day attacks risks level is like “measuring an immeasurable”. Obviously, we cannot measure the severity of vulnerability, while it is not known. Therefore, we must consider the degree of exploitability, while measuring the risks of zero-day attack.

The proposed framework provides a method for zero-day detection and estimates the likelihood of system being intruded by attacker. Our framework assumes that “an attacker can only advance his attack position to a node that has connectivity and vulnerability to be exploited”.

Furthermore, our framework is based on the link analysis algorithm [19] used for personalized web, that measures the probability of visiting a web page by an anonymous web surfer. The idea behind link analysis algorithm is that pages visited more often are more important and having high probability of visit. We follow the same approach while defining AttackRank algorithm to rank nodes in the attack graph in order of their intrusion likelihoods. The ranking determines which attack path is more vulnerable or requires more immediate attention for network protection.

ENHANCED FRAMEWORK FOR RISKS ASSESSMENT OF ZERO-DAY VULNERABILITIES

The architecture proposed in our previous work, consists of three layers [1]: (1) Zero-day attack path generator; (2) Risk analyzer; and (3) Physical layer. Once the generated attack graph is generated, these layers execute dedicated functionality in parallel. Parallel work of each layer improves the performance of our approach.

In this paper, we analyze the functionality of each layer.

Zero-day Attack Path Generator Layer

The aim of Zero-day Attack Path Generator layer is to identify aberrant network behavior, in order to detect unknown vulnerabilities which are rare to find and have high value. It detects unknown attacks and generates signatures for the Snort by analyzing the incoming traffic.

In particular, it includes four main components:

(1) Snort anomaly detector;

(2) Attack-graph generator;

(3) Detection engine; and

(4) Zero-day attack path generator.

The network is initially filtered by *Snort* Intrusion Detection/ Prevention System, to block known attacks based on the signatures. In order to lure attackers, we placed honeypot in our network with many common services running (like Apache, share folders etc.) in order to seem more valuable. Snort analyzes the traffic and performs filtering based on its signatures, the traffic is either dropped or passed as it is malicious or not. The filtered traffic is then logged by the honeypot; honeypot only captures the information and doesn’t do any processing.

To capture intrusion propagation, the *Attack Graph* is built 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 (node) being infected in the attack graph.

Detection engine is implemented on another protected machine which retrieves the information saved on the honeypot through a secure channel. The function of Detection Engine is to sense malicious packets that may cause exploit and whose signature is not defined previously in Snort IDS/IPS. The 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.

Fig. 3 shows how the zero-day attack path generator works.

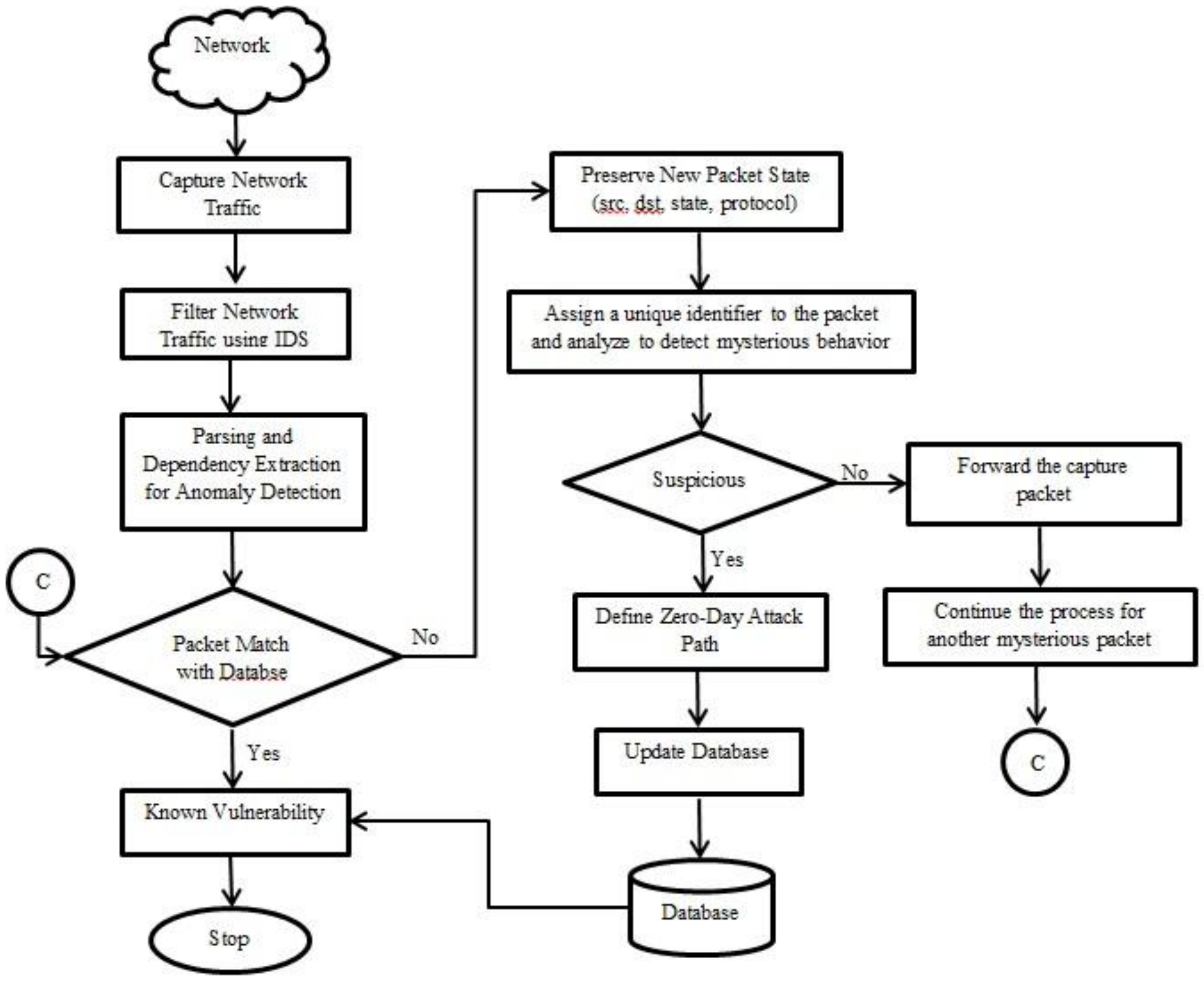

Network
Capture Network Traffic
Filter Network Traffic using IDS
Parsing and Dependency Extraction for Anomaly Detection
C
Packet Match with Databse
No
Yes
Known Vulnerability
Stop
Preserve New Packet State (src, dst, state, protocol)
Assign a unique identifier to the packet and analyze to detect mysterious behavior
Suspicious
No
Yes
Forward the capture packet
Continue the process for another mysterious packet
C
Define Zero-Day Attack Path
Update Database
Database

Figure 3. Functioning of zero-day attack path generator layer of proposed framework

Parsing and Dependency Extraction through Snort anomaly detector: The purpose of Snort anomaly detector is to detect and filter known attacks from the captured network scenario. This task is achieved 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 [20, 21]. When traffic flows through the Snort, it is analyzed by Snort rule-set; if the traffic turns out to be malicious the Snort filters it and does not let it pass. The packets that match with the Snort profile are known attacks, and after storing their information in a centralized database, these packets are discarded. If the traffic doesn’t match (packets that are partially matched or not matched) with the rule-set defined using Snort, it flows through the network.

We placed honeypot in our network alongside other systems like several workstations that run different operating systems, servers and others hosts. The honeypot system logs information about filtered traffic, the log is used by Detection Engine to detect suspicious activity.

Attack Graph Generation: After filtering the known attacks through parsing, further analysis is done by Snort tagger on extracted mysterious packets which did not trigger any alert. Tagging packets is a way to continue logging packets from a session or host that generated an event in Snort [22]. Tagged traffic is logged to allow analysis of response codes and post-attack traffic. The function of Snort tagger is to monitor the traffic, to tag the packets and send them to the 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 based on six 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 (with mysterious conditions) is generated in this module.

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 (e.g., an attack or unauthorized activity). The Detection Engine receives the parsed packets, compares them with existing "good" traffic, and detects unknown observations. The "good" traffic is the collection of traffic generated by 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 safe machines, based upon the past experience. Algorithm 1 explains the operation of the detection engine.

Algorithm 1

```
1: Procedure zero_day_detection
2: for network_scenario in Network do
3:  if (equals(packet_content,snort_rules)) then
4:      drop current_packet;
5:  else
6:      preserve filtered_packet := current_packet;
7:  end if
8: tag:= hash (preserve filtered_packet (arvl_time,
          source_ip, destination_ip, source_port,
          destination_port, protocol));
9:  update_database(tag);
10: tagged_packet:= preserve filtered_packet;
11: if ( NOT ( isMalicious (tagges_packet)) ) then
12:     Capture good traffic network scenario from safe
         systems
13:     Extract features and update Snort NIDS database
14: else
15:     unknown:= tagges_packet;
16:     insert unknown;
17:     update zero_day_database(unknown);
18: end if
19: end for
20: end procedure
```

Zero-day Attack Graph Generator: An attack graph estimates the probability of an attacker reaching his goal (a vulnerable host) in a network, i.e. it ranks the intrusion likelihoods. The ranking determines which attack path is more vulnerable or requires more immediate attention for network protection.

Risk Analyzer Layer

In our framework, in the first phase, an attack graph is built as chains of possible vulnerability exploits. An attack graph can help security persons to locate security flaws. In the second phase, we focus on ranking the nodes of the attack graph, based on likelihood of an attacker reaching these states. Such ranking determines the most vulnerable attack paths that require immediate attention for security reasons. The proposed approach is based on link analysis algorithm used for personalized Web [19]. The idea behind using link analysis is that link analysis algorithm measures the probability of visiting a web page by an anonymous web surfer. We are following this approach while measuring the likelihood of vulnerability to be exploited. Conceptually, the risk of vulnerability exploitability is depending upon two factors: likelihood of exploit and impact. However, in case of zero-day we can't measure the impact of vulnerability because it was unknown and not exploited previously; hence we are focusing on measurement of likelihood of exploit which actually defines a potential vulnerability can be successfully exploited. In order to find probability of a node to be vulnerable we developed an AttackRank algorithm which is described in the next section.

Along with AttackRank, for determining the exploitability of vulnerability, we are examining three prominent attributes: Attack Vector, Attack Complexity, and Authentication of severity matrix [23].

Attack Vector: The Attack vector includes the difficulty values required to exploit certain vulnerability from various access location points. For example, a hacker may exploit certain vulnerability by accessing the target local network remotely. In this case, the value of exploitability vulnerability will be high.

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

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

AttackRank Algorithm:

The proposed AttackRank algorithm measures the likelihood of exploit in the generated attack graph. AttackRank is based on the PageRank algorithm [24]. 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. In this situation attacker follows backtracking, i.e. he will start over from one of the set of initial states to find an alternative path. On the other hand, a web surfer can randomly pick a web page to visit via its URL while an attacker does not have the same freedom. Also, a web surfer can directly type URL to reach at destination page but an attacker requires multiple steps to advance to an attack state that the target system is completely down. However, we can simulate the structure of web containing web pages as nodes with the network graph containing hosts as nodes. The PageRank measures the rank of page to be visited, in the same way by network graph we are measuring the likelihood of vulnerability to be exploited.

AttackRank algorithm is based on the assumption:

Assumption: "*If the attacker dumps the current attack path, he will find an alternative path by back-tracking (from one of the set of previous states), and if he continues attacking, he will attempt to each of the possible navigational states with a probability, based on how hard its vulnerabilities can be exploited".*

Based on this assumption, the AttackRank algorithm finds the frequency of exploit.

Algorithm 2: *AttackRank algorithm for likelihood detection of exploit*

```
1: procedure AttackRank G(V,E)
2: I: set of initial states Є V
3: for each u, v Є V do
4: if u Є in_link(v) then
5:   attackrank(v):=attackrank(in(u))/ out(v)
6:   else if v Є out(v) then attackrank(v):=1-
       (attackrankin((u))/ out (v))
7: else
8:  attackrank(v):=1
9: end if
10:end for
11:end procedure
```

Physical Layer

The physical layer contains a centralized database and a centralized server used during the information processing involved in the first two layers. All of the information along with attack path, malicious or non-malicious activities, known or unknown exploits is stored in the database server of the physical layer. This database is continuously updated by the records in the audit network data repository that do not yet have any sort of context profile. The centralized database primarily contains the following six tables:

(1) Host table (tbl_host): This table contains details of all hosts present in the network (either up or down). The general structure of host table is *tbl_host (host_ip, host_name, mac_addr, prt_addr, status)*. Here, host_ip is the unique identifier for each host. In our network, we are using IPv4 format. Along with ip, each host present in the network has a unique name, MAC address, port address and host status information. The status information contains the binary value that represents that the host is either alive or not.

(2) Snort Rule-set table (tbl_snort_rule-set): Snort rules logically have two parts: *(rule_header, rule_option).* The rule_header contains information about action and criteria for matching a rule against data packets. The general structure of Snort Rule-set table is: *tbl_snort_rule-set (action, protocol, source_ip, source_port, destination_ip, destination_port)*. The rule_option part usually contains an alert message and information about which part of the packet should be used to generate the alert message.

(3) WireShark Result table (tbl_packet_detail): After Snort parsing, the transferred packets traffic is analyzed by WireShark. Information about packet obtained by WireShark packet sniffers is stored in this table. The general structure of this table is *tbl_packet_detail (no, arrival_time, src_ip, dest_ip, protocol, length, info)*.

(4) Malicious Packet Detail table (tbl_malicious_packet): After analysis of tbl_packet_detail, good packets are dropped while suspicious packets information filters and stored in temporary table tbl_malicious_packet. Information stored in this table is used further to update Snort table.

(5) Vulnerable Host Path table (tbl_vuln_host): This table contains information about vulnerable host route, traced by Nmap.

(6) Zero-day Vulnerability table (tbl_unknown_vuln): It contains the information about identified suspicious packets. The structure of this table is *tbl_unknown_vuln (id, attack_vector, attack_complexility, privilege)*.

In our framework, database server utilized MySQL database.

EXPERIMENTAL SETUP

To test the performance of our framework, we selected a group of 8 hosts playing miscellaneous roles. These hosts constitute the test-bed for our case study. Fig. 3 shows the structure of the test-bed that is comprised of these hosts in diverse physical locations. In particular, the test-bed includes: a network server located at academic block within the contact range of firewall (208.91.191.121); a server located at School of Engineering and Technology (128.168.1.4), and other machines.

Fig. 4 shows the experimental setup of attack scenario. The attacker uses Kali Linux run on virtual machine; Ubuntu Metasploitable Server is used for honeypot system which has many holes available for potential attackers; the web server run on windows platform.

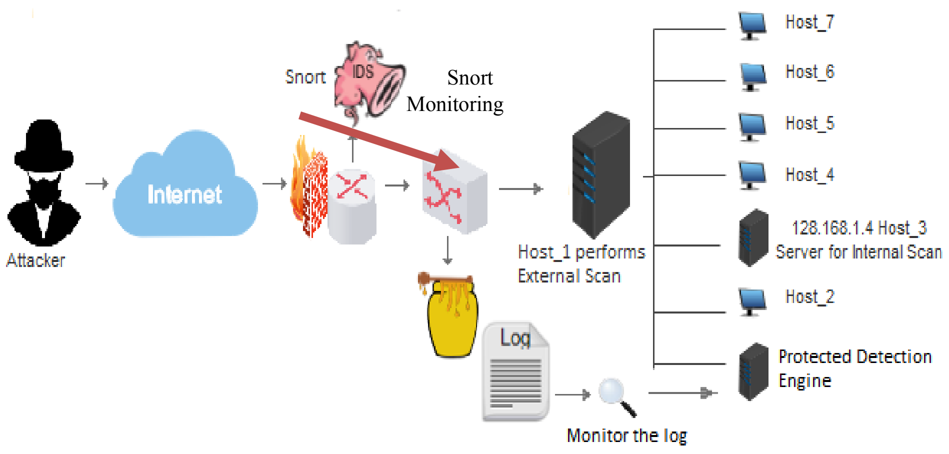


Figure 4. Experimental setup of the test-bed

In the developed network architecture, packet enters the system from network and goes first through the firewall rules, if it does not matches to any of those rules it is logged. Snort analyzes the traffic pass through the firewall and performs filtering based on its signatures, the traffic is either dropped or passed as it is malicious or not. The filtered traffic is then logged by the honeypot. Honeypot only captures the information and doesn't do any processing. The detection engine then analyses the log, identify malicious IPs and block them; every incoming packet from these IPs will be then forwarded to the Honeypot by *Activating IP forwarding*.

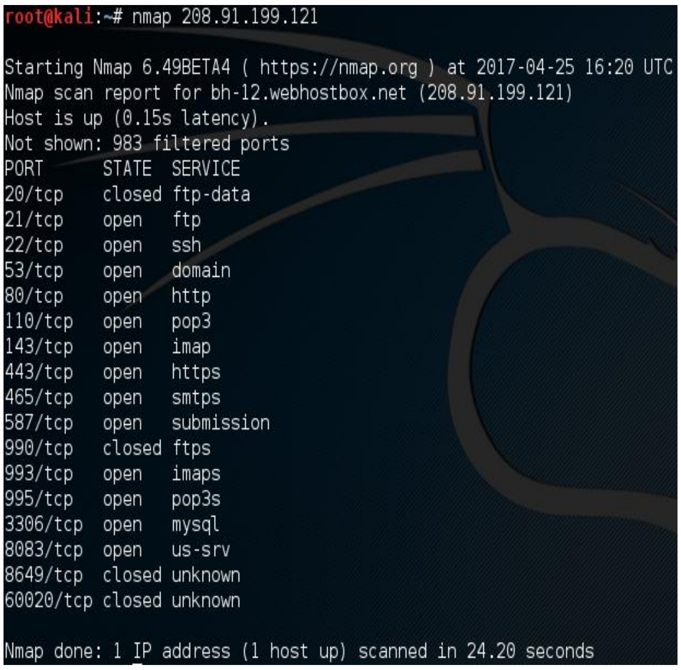
To detect vulnerabilities present in our network we perform external and internal scan. The External scan is done through a router or firewall by the means of Nmap [25] and Nessus [26] vulnerability scanner. The internal scan took place at the School of Engineering and Technology (SoET) location, and was plugged into a server that resides inside university's network. As shown in Fig. 5, the placement of the *blue* scanner is inside the firewall, so it can scan internal vulnerabilities. On the contrary, the *red* scanner is used for external vulnerabilities scan. Both internal and external vulnerability scans are used to collect data in order to assess the effectiveness of current security measures taken at the campus network. The objective of the internal scan is to avoid external security countermeasures to get a detailed view at system configurations. The external scan determines 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's network [27].

PERFORMANCE EVALUATION

In this section we present the experiments performed in order to confirm the accuracy and efficiency of our framework. We built a test-bed network and launched an attack towards it. Since zero-day exploits are not readily available, we emulated zero-day vulnerabilities with known vulnerabilities. For example, we treated 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-day" vulnerabilities can be available to verify the correctness of our experiment results.

The basic components of the test-bed (Fig 4) are two servers for network vulnerabilities scan. 208.91.199.121 performs the external scanning through a router or firewall, by the means of the Nmap and Nessus vulnerability scanner. Nmap placed within contact range of University, and generates details about active services, credentials and successful attacks. Fig. 5 shows the scanning result generated by Nmap at particular timestamp.


```
root@kali:~# nmap 208.91.199.121

Starting Nmap 6.49BETA4 ( https://nmap.org ) at 2017-04-25 16:20 UTC
Nmap scan report for bh-12.webhostbox.net (208.91.199.121)
Host is up (0.15s latency).
Not shown: 983 filtered ports
PORT      STATE  SERVICE
20/tcp    closed ftp-data
21/tcp    open   ftp
22/tcp    open   ssh
53/tcp    open   domain
80/tcp    open   http
110/tcp   open   pop3
143/tcp   open   imap
443/tcp   open   https
465/tcp   open   smtps
587/tcp   open   submission
990/tcp   closed ftps
993/tcp   open   imaps
995/tcp   open   pop3s
3306/tcp  open   mysql
8083/tcp  open   us-srv
8649/tcp  closed unknown
60020/tcp closed unknown

Nmap done: 1 IP address (1 host up) scanned in 24.20 seconds
```

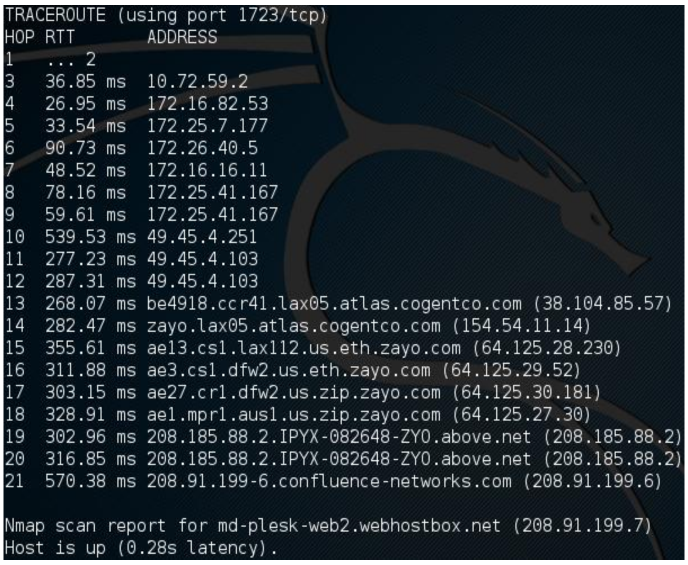
Figure 5. Nmap external port scan result

Scanning activities result that the server 208.91.199.121 has 13 open ports including 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 three 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. CVE-2014-6271 allows remote attackers to execute arbitrary code via a crafted environment.

The HTTP service has two 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 HTTP service vulnerabilities are present in the Apache HTTP Server.

To capture intrusion propagation, an attack graph was built by capturing the network scenario at specific timestamp. Fig. 6 shows the TraceRoute result generated by Nmap, with this information of captured network scenario at any time stamp, an attack graph is built. Nodes of attach graph represent the hosts within network, while edges represent the favorable attack conditions.



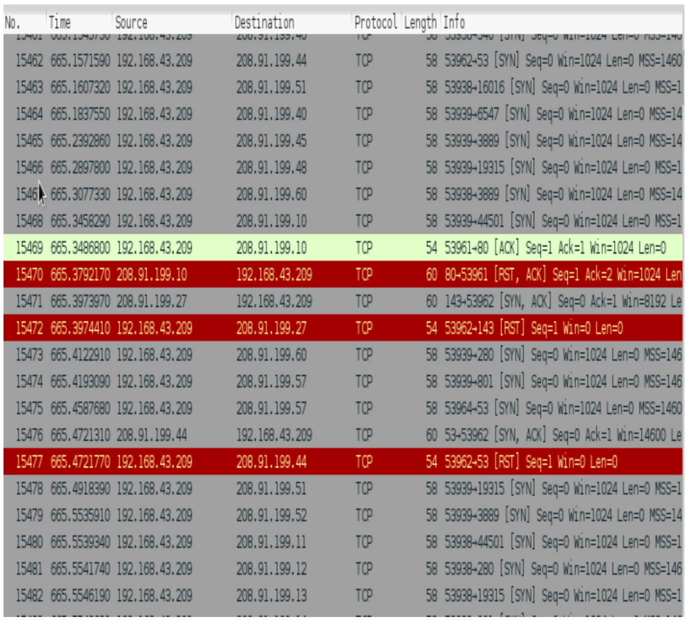
```
TRACEROUTE (using port 1723/tcp)
HOP RTT       ADDRESS
1    ... 2
3   36.85 ms  10.72.59.2
4   26.95 ms  172.16.82.53
5   33.54 ms  172.25.7.177
6   90.73 ms  172.26.40.5
7   48.52 ms  172.16.16.11
8   78.16 ms  172.25.41.167
9   59.61 ms  172.25.41.167
10  539.53 ms 49.45.4.251
11  277.23 ms 49.45.4.103
12  287.31 ms 49.45.4.103
13  268.07 ms be4918.ccr41.lax05.atlas.cogentco.com (38.104.85.57)
14  282.47 ms zayo.lax05.atlas.cogentco.com (154.54.11.14)
15  355.61 ms ae13.cs1.lax112.us.eth.zayo.com (64.125.28.230)
16  311.88 ms ae3.cs1.dfw2.us.eth.zayo.com (64.125.29.52)
17  303.15 ms ae27.cr1.dfw2.us.zip.zayo.com (64.125.30.181)
18  328.91 ms ae1.mpr1.aus1.us.zip.zayo.com (64.125.27.30)
19  302.96 ms 208.185.88.2.IPYX-082648-ZYO.above.net (208.185.88.2)
20  316.85 ms 208.185.88.2.IPYX-082648-ZYO.above.net (208.185.88.2)
21  570.38 ms 208.91.199-6.confluence-networks.com (208.91.199.6)

Nmap scan report for md-plesk-web2.webhostbox.net (208.91.199.7)
Host is up (0.28s latency).
```

Figure 6. Input for Attack path generation through Nmap

The attack graph was 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.

After Snort filtering of the known attacks, Wireshark sniffs the filtered traffic in order to detect unknown malicious packets having unexpected behavior. Fig. 7 shows the snapshot of Wireshark. Wireshark sniffs essential characteristics of a packet in order to deny any behaviors that are not expected. It predicts the flow of network traffic.



```
No.   Time        Source          Destination     Protocol Length Info
15462 665.1571590 192.168.43.209  208.91.199.44   TCP      58 53962→53 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
15463 665.1607320 192.168.43.209  208.91.199.51   TCP      58 53938→16016 [SYN] Seq=0 Win=1024 Len=0 MSS=1
15464 665.1837550 192.168.43.209  208.91.199.40   TCP      58 53939→6547 [SYN] Seq=0 Win=1024 Len=0 MSS=14
15465 665.2392860 192.168.43.209  208.91.199.45   TCP      58 53939→3889 [SYN] Seq=0 Win=1024 Len=0 MSS=14
15466 665.2897800 192.168.43.209  208.91.199.48   TCP      58 53939→19315 [SYN] Seq=0 Win=1024 Len=0 MSS=1
1546  665.3077330 192.168.43.209  208.91.199.60   TCP      58 53938→3889 [SYN] Seq=0 Win=1024 Len=0 MSS=14
15468 665.3458290 192.168.43.209  208.91.199.10   TCP      58 53939→44501 [SYN] Seq=0 Win=1024 Len=0 MSS=1
15469 665.3486800 192.168.43.209  208.91.199.10   TCP      54 53961→80 [ACK] Seq=1 Ack=1 Win=1024 Len=0
15470 665.3792170 208.91.199.10   192.168.43.209  TCP      60 80→53961 [RST, ACK] Seq=1 Ack=2 Win=1024 Len
15471 665.3973970 208.91.199.27   192.168.43.209  TCP      60 143→53962 [SYN, ACK] Seq=0 Ack=1 Win=8192 Le
15472 665.3974410 192.168.43.209  208.91.199.27   TCP      54 53962→143 [RST] Seq=1 Win=0 Len=0
15473 665.4122910 192.168.43.209  208.91.199.60   TCP      58 53939→280 [SYN] Seq=0 Win=1024 Len=0 MSS=146
15474 665.4193090 192.168.43.209  208.91.199.57   TCP      58 53939→801 [SYN] Seq=0 Win=1024 Len=0 MSS=146
15475 665.4587680 192.168.43.209  208.91.199.57   TCP      58 53964→53 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
15476 665.4721310 208.91.199.44   192.168.43.209  TCP      60 53→53962 [SYN, ACK] Seq=0 Ack=1 Win=14600 Le
15477 665.4721770 192.168.43.209  208.91.199.44   TCP      54 53962→53 [RST] Seq=1 Win=0 Len=0
15478 665.4918390 192.168.43.209  208.91.199.51   TCP      58 53939→19315 [SYN] Seq=0 Win=1024 Len=0 MSS=1
15479 665.5535910 192.168.43.209  208.91.199.52   TCP      58 53939→3889 [SYN] Seq=0 Win=1024 Len=0 MSS=14
15480 665.5539340 192.168.43.209  208.91.199.11   TCP      58 53938→44501 [SYN] Seq=0 Win=1024 Len=0 MSS=1
15481 665.5541740 192.168.43.209  208.91.199.12   TCP      58 53938→280 [SYN] Seq=0 Win=1024 Len=0 MSS=146
15482 665.5546190 192.168.43.209  208.91.199.13   TCP      58 53938→19315 [SYN] Seq=0 Win=1024 Len=0 MSS=1
```

Figure 7. Snapshot of WireShark packet sniffer

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 were used to evaluate performance and accuracy of the proposed framework. Fig. 8 and Fig. 9 represent true detection rate and false positive rate of zero-day attack correspondingly.

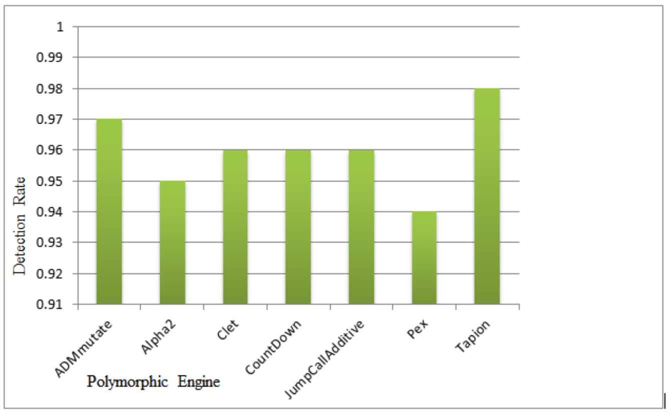


Figure 8. True Positive Rate

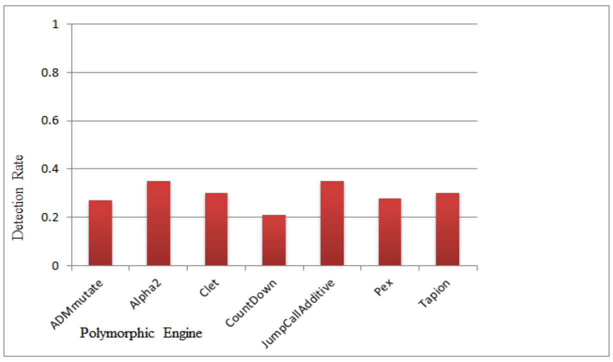


Figure 9. False Positive Rate

Fig. 10 shows ROC curve that is drawn by taking the average value of TPR. In Fig. 10, it is clearly shown that ROC is closer to 1, which proves the efficiency of our proposed approach.

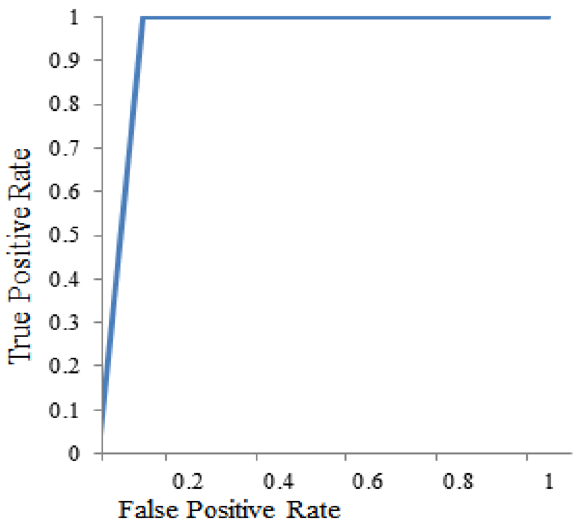


Figure 10. Average value of ROC curve

CONCLUSION

In this paper we proposed a novel security approach that assesses the security risks of zero day vulnerabilities for compromising a network asset and measures the capability of hardening a network against zero day vulnerabilities. The proposed method developed a probabilistic attack graphs which encode probabilistic and temporal knowledge of the attacker's behavior and determine the risks of exploit. 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 96% and the false positive rate was 0.3%. The proposed algorithm results efficient performance for both detection and prediction of zero-day vulnerabilities.

REFERENCES

[1] Singh, U. K., Joshi, C.: Scalable Approach towards Discovery of Unknown Vulnerabilities, International Journal of Network Security, Vol. 20, No. 5, pp. 827-835, (2018).

[2] Joshi, C., Singh, U. K., Singh, S. K.: ZDAR system: defending against the unknown. International Journal of Computer Science and Mobile Computing. 5(12), 143-149 (2016).

[3] Yang, Y., Zhu, S., Cao, G.: Improving sensor network immunity under worm attacks: a software diversity approach. In: Proceedings of the 9th ACM International Symposium on Mobile Ad Hoc Networking and Computing. ACM, (pp.149–158) (2008).

[4] Caballero, J., Kampouris, T., Song, D., Wang, J.: Would diversity really increase the robustness of the routing infrastructure against software defects?. In: Proceedings of the Network and Distributed System Security Symposium (2008).

[5] White Paper: ZERO-DAY DANGER: a survey of zero-day attacks and what they say about the traditional security model. FireEye Security Raimagined (2015).

[6] Singh, U. K., Joshi, C.: Comparative Study of Information Security Risk Assessment Frameworks, International Journal of Computer Application (IJCA), ISSN 2250-1797, UGC No. 64190, Vol. 8, No. 2, April 2018, pp.82-89, impact factor: 3.12

[7] Joshi, C., Singh, U. K.: Layered Architecture for Security Vulnerabilities Assessment - A Cornerstone of Effective Security Planning, International Journal of Creative Research Thoughts, Vol 6, N0 2, pp. 668-700, April 2018.

[8] Singh, U. K., Joshi, C., Singh, S. K: Zero day attacks defense technique for protecting system against unknown vulnerabilities. International Journal of Scientific Research in Computer Science and Engineering. 5(1), 13-18, (2017).

[9] Singh, A. P.: A study on zero day malware attack. International Journal of Advanced Research in Computer and Communication Engineering. 6(1), 391-392 (2017).

[10] Kaur, R., Singh, M.: Automatic evaluation and signature generation technique for thwarting zero-day attacks. Second International Conference, SNDS 2014, India, (pp.298-309), March 13-14 (2014).

[11] Holm, H.: Signature based intrusion detection for zero-day attacks: (not) a closed chapter?, 47th International Conference on System Science, Hawaii (2014).

[12] Hammarberg, D.: The best defenses against zero-day exploits for various-sized organizations. SANS Institute Reading Room, (Sep. 2014).

[13] Kaur, R., Singh, M.: Efficient hybrid technique for detecting zero-day polymorphic worms. Advance Computing Conference (IACC), 2014 IEEE International, pp.95-100, 21-22 Feb. (2014).

[14] Kaur, R., Singh, M.: A survey on zeroday polymorphic worm detection techniques. IEEE Communication Surveys & Tutorials. 16(3), 1520-1549 (2014).

[15] Joshi, C., Singh, U. K.: Quantifying security risk by critical network vulnerabilities assessment. International Journal of Computer Applications. 156(13), 26-33 (2016).

[16] Joshi, C., Singh, U. K.: Information security risk management framework for university computing environment. International Journal of Network Security. 19(5), 742-751 (2017).

[17] Joshi, C., Singh, U. K.: Quantitative security risk evaluation using CVSS metrics by estimation of frequency and maturity of exploit. The World Congress

on Engineering and Computer Science (WCECS 2016) San Francisco, USA.

[18] Joshi, C., Singh, U. K.: Information security assessment by quantifying risk level of network vulnerabilities. International Journal of Computer Application. 156(2), 6-10, (2016).

[19] Joshi, C., Singh, U. K.: 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. 128(11), 1-5 (2015).

[20] Roesch, M.: Snort - lightweight intrusion detection for networks. In: Proceedings of LISA '99: 13th Systems Administration Conference Seattle, Washington, USA, pp. 229-238, (Nov 1999).

[21] Liu, X., Ye, Y.: Intrusion detection system based on snort. In: Proceedings of the 9th International Symposium on Linear Drives for Industry Applications, Volume 3, Lecture Notes in Electrical Engineering 272, doi: 10.1007/978-3-642-40633-1_82, Springer-Verlag, Berlin Heidelberg (2014).

[22] Patel, N., Shah, V., Pancholi, K.: An analysis of network intrusion detection system using SNORT. International Journal for Scientific Research & Development. 1(3), 410-412, (2013).

[23] Mell, P., Scarfone, K., Romanosky, S.: CVSS: a complete guide to the common vulnerability scoring system version 2.0. Forum of Incident Response and Security Teams (FIRST), (2007).

[24] Page, L., Brin, S., Motwani, R., Winograd, T.: The Pagerank Citation Ranking: bringing order to the Web. Technical Report, Stanford Dig. Lib. Tech. Project, pp.1-17, (1998).

[25] Joshi, C., Singh, U. K.: Security testing and assessment of vulnerability scanners in quest of current information security landscape. International Journal of Computer Applications. 145(2), 1-7 (2016).

[26] Nessus Vulnerability Scanner. http://www.tenable.com /products/nessus-vulnerability-scanner

[27] Joshi, C., Singh, U. K.: Performance evaluation of web application security scanners for more effective defense. International Journal of Scientific and Research Publications, 6(6), 660-667 (2016).