Wireless Pers Commun (2015) 83:551–570
DOI 10.1007/s11277-015-2408-0

Wireless Rogue Access Point Detection Using Shadow Honeynet

Neha Agrawal · Shashikala Tapaswi

Published online: 24 February 2015

Abstract Network security is becoming a great challenge as the popularity of wireless network is increasing. On account of open medium, insignificant software implementation, potential for hardware deficits, and improper configuration; Wi-Fi network is vulnerable to rogue access point (RAP). RAP is an unauthorized access point which can be installed by end-users without the knowledge of security administrator. When this rogue device is connected to the Internet, it can be used by an attacker to breach the security of the network. Attackers can also install RAP to lure other users for sniffing sensitive data. In this paper, a method called "Shadow Honeynet" has been proposed for the detection and prevention of RAP. The concept of Shadow Honeynet arrives from Shadow Honeypot that integrate the best features of anomaly detection system (ADS) and Honeypot. The shadow is an instance of protected software that share all internal states with the regular ("production") instance of the application to detect potential attacks. The proposed architecture improves the overall performance of the system by diminishing false positives rate generated by ADS and can be able to sustain the overall workload of honeypot.

Keywords Rogue access point · Anomaly detection system · Wired equivalent privacy · Wi-Fi protected access · Tree transformation language · Honeypot

1 Introduction

Detection of RAP is most important, as it is responsible for various security threats. Security protocol used in Wireless Local Area Network (WLAN) is Wired Equivalent Privacy (WEP) but fails to achieve security goals of confidentiality, integrity and

N. Agrawal (✉) · S. Tapaswi
ABV-Indian Institute of Information Technology and Management, Gwalior, India
e-mail: nehaiiitm345@gmail.com

S. Tapaswi
e-mail: stapaswi@iiitm.ac.in

availability. To overcome the flaws of WEP, Wi-Fi Protected Access (WPA) has been developed. Since WPA still relies on Ron's Code (RC) 4 encryption algorithm, the access points can be easily compromised [1–4].

Rogue access point can be placed into four categories [5]:

1. *Improperly configured access point:* Because of minor configuration mistakes, a legitimate access point may suddenly convert into improperly configured device. The reasons behind this type of APs are: network administrator with insufficient security knowledge (e.g, choose inappropriate encryption and authentication techniques), use of faulty AP's driver, and sometimes after software updating a properly configured AP becomes vulnerable.
2. *Unauthorized access point:* For flexibility and scalability, employees install APs without the knowledge of administrator. Hackers can connect to the internal organization, sniff sensitive data, steal bandwidth and use the network to attack other users with the help of these APs. Another possibility for the origin of unauthorized APs is the neighborhood WLAN. As the clients always prefer to connect with the AP having high signal strength, due to this sometimes authorized clients of one organization may connect with the AP in close vicinity. This neighboring AP can expose sensitive data.
3. *Phishing access point:* Outside the wireless network, if the malicious user installs an AP to obtain users' credentials like usernames and passwords by masquerading as a legitimate user, termed as phishing AP. This allows the assailant to conduct Man-in-the-Middle (MITM) attack in the wireless network that does not require mutual authentication between client-to-server and server-to-client. Since WEP enabled network does not enforce mutual authentication, so it is easy to launch MITM attack in such networks.
4. *Compromised access point:* If an attacker cracks the key that is being used in WEP and WPA Pre-Shared Key (WPA-PSK) enabled network, then it will become compromised. Once the attacker discovers a key, all APs those are using the same credentials will become rogue AP. It is easy for an attacker to masquerade as a legitimate user in a compromised network.

It is easy to detect first three classes of rogue access point. However, it is difficult to deal with the fourth one. The overall security of the network can be significantly affected by compromised AP because it is more dangerous and difficult to detect. There are several methods for the detection of rogue access points, but still there is a need to develop an approach that can deal with all types of rogue access point efficiently and effectively.

The paper is further organized as follows: Sect. 2 elaborate previously proposed methods to detect and prevent rogue access point and its associated problems, Sect. 3 explains the shadow honeypot architecture and its utility, Sect. 4 describes the proposed approach, implementation is described in Sects. 5, 6 illustrates experimental evaluation, Sect. 7 concludes the paper and presents future work.

2 Related Work

Various methods have been introduced for detecting RAP, but each technique has its own pros and cons. Traditional RAP detection rely on enumeration tools like NetStumbler, etc, which runs on handheld devices and laptops to gather information [6]. This method is timeconsuming, expensive and unreliable. Further it fails to detect if an assailant spoofs Medium Access Control (MAC) address and Service Set Identifier (SSID) of a legitimate access point.

A number of commercial products have been developed for continuous monitoring like AirDefence [5, 7]. To capture, process and correlate network events, it uses the combination of radio frequency sensors and intrusion detection and prevention servers.

Radio Frequency (RF) monitoring [5, 8] is used for the detection of wireless rogue access points. In this approach, multiple APs and mobile clients were used to perform RF monitoring. Flaws include: difficulty in proper installation of APs for sniffing, attacker can easily turn off the AP during scanning, MAC spoofing is undetected.

Distributed monitoring infrastructure, Dense Array Inexpensive Radios (DAIR) [5, 9] provides comprehensive traffic capturing ability. This is not very effective approach as it depends on the APs functionality that can be easily turned off.

The paper [5, 6] proposed a method for wired network. This approach was based on inter-packet spacing, since the spacing between the wireless packets is more than that of wired packets because of one hop count increment (due to involvement of wireless router). The network becomes vulnerable after installing wireless access point in it. The client (a laptop) was connected to the 100 Mbps shared hub (at central location) where sniffing took place. If packet spacing is large, it was considered as rogue access point.

Detection of rogue access point and eavesdropper, can be performed using multiple sniffers. Each sniffer has three network cards and intrusion detection capabilities. This method fails to detect MAC address spoofing [5, 10].

A novel approach called RAP has been introduced that targets commodity Wi-Fi network [5]. It consists of three main elements: a packet collector, a detection engine, and a preemption engine. This method has the following advantages: (1) it neither requires specialized hardware nor modification to existing networks; (2) can be integrated as a plug-in enhancement to Wi-Fi networks; (3) protect the network from adversaries capable of violating WLAN standard; (4) can be used with the security protocols like WEP and WPA.

Organizations have started to deploy Intrusion Prevention System (IPS), another method to detect and prevent newer attacks, as the malevolent activities is on high spirit on today's Internet. Since current Internet Protocol Security (IPSec) use rule-based intrusion detectors like SNORT [11] that can detect only already known attacks. The flaws associated with Intrusion Detection System (IDS) include high false positive and false negative alerts. If the IDS system is more sensitive to attacks then it may misclassify legitimate users as malicious i.e, increasing false positives, and if the IDS system is insensitive to attacks then false negatives increases i.e., missing some real attacks. To tune the system for low false positives, Shadow Honeypot [12] was introduced.

3 Shadow Honeypot Architecture

To handle network-based attacks Shadow Honeypot [12] architecture is used. The architecture consist of three components: *filtering engine*, *array of anomaly detection sensors* and *shadow honeypot* as shown in Fig. 1.

The filtering component is used to filter unauthorized traffic flowing towards the secured network. It uses an authenticated list and by comparing the traffic against this list, it rejects malicious packet. Traffic that passes the filtering stage is passed to an array of anomaly detection sensors otherwise drops the request by short-circuiting the detection heuristics and shadow testing. Anomaly detection sensors analyze the packets' characteristics. After

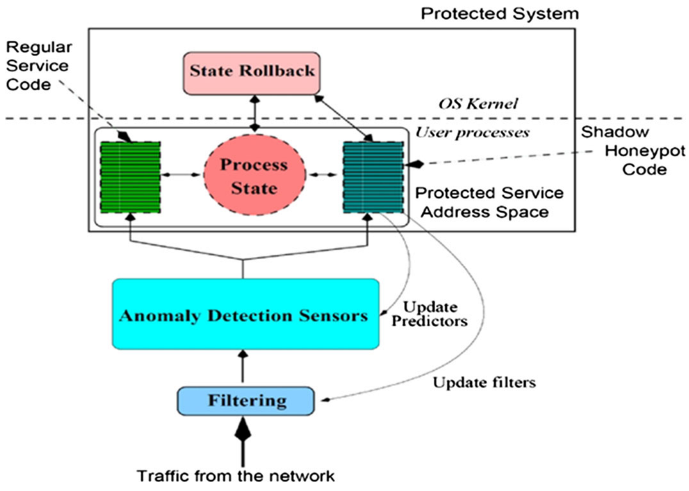


Fig. 1 Shadow honeypot architecture [12]

passing second stage, traffic is forwarded to the shadow code. It is preferable to tune the detectors towards high sensitivity to increase the false positives. The shadow and regular application fully share internal states to avoid attacks that exploit differences between the two. The result of shadow honeypot is reflected to the low level stages to filter out same malevolent packets immediately if it is detected in future.

3.1 Utility of Shadow Honeypot

"Honeypots is an information system resource whose value lies in unauthorized or illicit use of that resource". Honeypot instrumented to detect active attacks but fails to detect passive attacks where an attacker lures a victim user to download malicious file. For only server side applications, honeypot can be used.

Unlike traditional honeypots that remain idle while waiting for an active attacker to probe, shadow honeypot is able to detect passive attack that lures a victim [13].

A Honeynet [14] is a network, which captures all inbound and outbound traffic to/from the reverse firewall. The reverse firewall limits the amount of malicious traffic that can leave the Honeynet. Honeynet is placed behind the reverse firewall. To provide the feel of a real system to hacker, standard production systems are used on the Honeynet.

For successful completion of Honeynet, there are two principles: *Data Capture* and *Data Control*. Data Capture relate to the gathering of all traffic that enters and leaves the Honeynet. It must be collected for further analysis without the knowledge of the individuals who is performing malicious activity. The data must be kept at different location so that hacker should not be able to destroy it, even if the system gets compromised.

Data Control concern to protect other networks from getting attacked and compromised by the Honeynet's systems.

Each and every activity that is performed either by an attacker or by an authorized user is logged by honeypot and honeynet. This increases the overhead, and reduces efficiency. For providing better performance shadow honeypot was introduced.

The advantages of shadow honeypot are given below:

- Improves system's performance by reducing false positive rate.
- Handled attacks against specific site with a particular internal state.
- Suitable for protecting attacks against client-side like for Web-browser and P2P file-sharing client.
- Additional detection mechanism can be easily integrated.
- Capable to deceive the attackers to a greater extent.

4 Proposed Approach

4.1 Architecture

The architecture that combines the best features of honeypot and anomaly detection system is "Shadow Honeypot". The paper proposes a method for the detection and prevention of rogue access points using shadow honeypot.

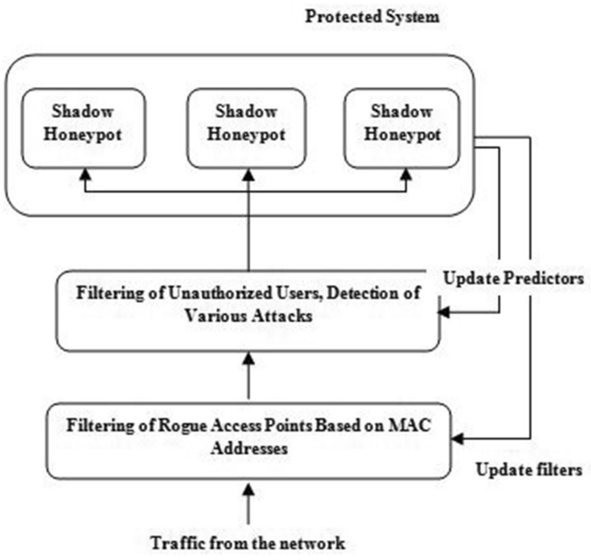
Considering Fig. 1, the filtering component is used to filter unauthorized access points (rogue access points). It uses a list, which contains the MAC addresses of all authorized access points. At this stage, the MAC addresses of all visible wireless routers is matched against the list, if it finds any router whose MAC address does not exist in the list then it will be treated as rogue access point. But there may be a possibility that either an attacker is using our authorized AP (improperly configured and compromised access point) or spoofs the MAC address of legitimate access point (phishing access point). In this case, the packet will be passed through the filtering stage and forwarded to the anomaly detection sensors. At this stage, ettercap [15], wireshark [16], snort [17] and anomaly detection heuristic payload sifting [18] are used to filter out unauthorized hosts and to detect various attacks.

Ettercap is used to scan all hosts connected to our wireless network and display their IP and MAC addresses; so that the assailants can be easily filtered out based on the unauthorized IP addresses. If an attacker is using ARP spoofing, then he/she easily evade the ettercap detection tool. For this, wireshark is used to detect Man-in-the-Middle and Denial-of-Service (DoS), Distributed Denial-of-Service (DDoS), smurf like attacks. Thereafter the packet is forwarded to the shadow honeypot for validation. Based on the results of anomaly detection system and shadow honeypot, shadow honeypot indicate measured false positive and false negative rate. This result is sent to the filtering and detection stage, so that if any packet is flowing from these malicious nodes will be rejected at the initial stage in future. Shadow and regular application fully share internal states to avoid attacks that exploit the differences between two.

For providing the illusion of real system environment to the attackers, the method used multiple shadow honeypots called " shadow honeynet" as shown in Fig. 2.

Considering Figs. 1 and 2 the shadow is an instance of the protected software (e.g a web server or client) that is instrumented to detect all potential attacks. This instance share all internal states with the regular (production) instance of the application. Shadow catch the attacks against it, and discard the changes that have been occurred in the shared states. Signal handler installed automatically by the tool indicates operating system (OS) that an attack has been detected. The OS roll back all the changes occured while servicing the malevolent request. The traffic that was misclassified as malicious will be validated by shadow, and transparently handled by the system (e.g. an HTTP request that was mistakenly classified as suspicious will be handled correctly). The filters and anomaly

Fig. 2 Shadow honeynet


Protected System
Shadow Honeypot
Shadow Honeypot
Shadow Honeypot
Update Predictors
Filtering of Unauthorized Users, Detection of Various Attacks
Filtering of Rogue Access Points Based on MAC Addresses
Update filters
Traffic from the network

detectors will be updated by the shadow to prevent future attacks. The proposed architecture improves the performance by reducing false positives. Since the system is designed sensitive to attacks; the attacks that were misclassified as malicious by anomaly detection sensors will be handled carefully by the production software transparently to the end users.

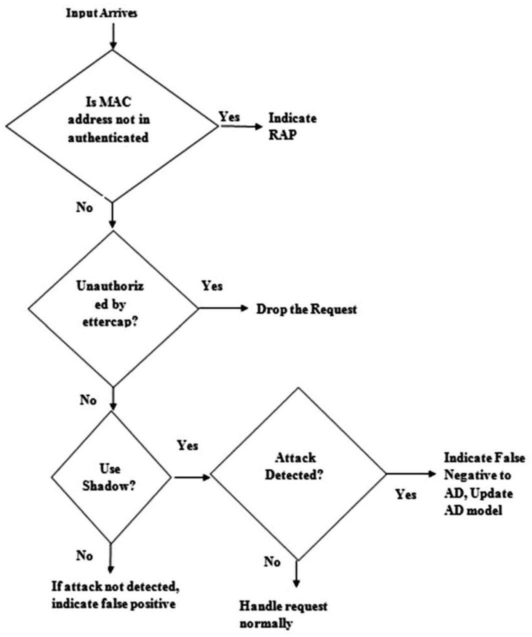
Specifically a tool "TXL compiler/interpreter" [19–21] is used to create the shadow version of the existing code. If attack is detected by the shadow then changes in the state will be rolled back and the outcome will be transferred to the filters and anomaly detectors to prevent future attacks. The shadow can be an entire separate process, running on other machine (loose coupling) or can be a different thread running on the same address space (tight coupling).

4.2 System Work Flow

Figure 3 shows the processing logic of the system. The steps involved are:

1. Filter check the MAC addresses of all showing wireless access points. If it find any router having unauthorized MAC address, then classify it as rogue access point and drop the request.
2. After passing first stage, the request goes to anomaly detector (AD) where the contents of the packets and its characteristics is to be considered. Here, three tools ettercap, wireshark, snort and a detection hueristic payload sifting are used.
   (a) Ettercap is used to scan the whole wireless network and detect the assailants. All the hosts having unauthorized IP addresses (attackers) are indentified here and drop the request. But there may be a case that the attacker is using authorized IP address (IP spoofing), then wireshark and snort is used to detect various attacks like MITM, DoS, DDoS, Smurf, etc,.
   (b) Using Wireshark, the packet flow rate is analysed. If an attacker performs MITM attack then packet spacing is increases due to his/her involvement

Fig. 3 System work flow


Input Arrives
Is MAC address not in authenticated
Yes
Indicate RAP
No
Unauthoriz ed by ettercap?
Yes
Drop the Request
No
Use Shadow?
Yes
Attack Detected?
Yes
Indicate False Negative to AD, Update AD model
No
If attack not detected, indicate false positive
No
Handle request normally

between authorized communicating entities. In DoS, DDoS, Smurf attacks packet spacing decreases. Suppose for normal wireless packets it is 10-15 ms (threshold) then for MITM it is greater than threshold and for other attacks it is less than threshold. Snort is used to provide more detailed information about packets as well as generating alarms.

(c) Payload sifting identify popular substrings in the packets' payload and mark them as fingerprints of rapidly spreading worms. These fingerprints are forwarded to the shadow honeypot for searching them in the network traffic. If the generated fingerprint is actually belong to the malicious activity then it indicates hit to ADS and filtering component, otherwise indicates false positive and upadate low level stages to reduce false positive rate.

For (a) and (b) two conditions will arise:

3. If classified as authorized, pass to the third stage:
   (a) If attack is detected then indicate false negative and update the Anomaly Detection (AD) and filtering components.
   (b) If attack is not detected, indicate true negative and handle the request carefully.
4. If classified as unauthorized then also pass to the third stage:
   (a) If attack is detected, indicate true positive and pass the result to below stages.
   (b) If attack is not detected then indicate false positive, and update the Anomaly Detection (AD) and filtering components.

5 Implementation

Figure 4 shows the experimental setup used for implementation in which ettercap, wireshark and snort have been used. The setup considered a small wireless network connected to one authorized *ciscob* wireless router. Three authorized hosts having IP addresses 192.168.38.27, 192.168.38.40 and 192.168.38.65 and one system having all required softwares named shadow honeypot are part of this home network. The attacker having IP address 192.168.38.212 connected to our network through wireless router and trying to breach the network security.

5.1 Filtering

Filtering component filter unauthorized access points by comparing their MAC addresses against white list. To do this, firstly calculate the MAC addresses of authorized APs. The paper consider only one authorized AP *(ciscob router)* in home network. Following are the steps to calculate the MAC address:

1. Find out the default router (gateway IP address), enter: **$ /sbin/route**
2. This gives the router's IP address. For ciscob, it is: **192.168.38.254**
3. Calculate router's MAC address, enter: **$ arp -a**
4. The router's MAC address is: **00.04.96.1e.56.e0**

Any AP not having this MAC address is indicated as rogue access point.

5.2 Anomaly Detection System

The packet after passing the first phase (not identified as attacker) enters into this phase. There are two possibilities; either the packet is coming from the authorized host or from unauthorized host. If the packet is coming from malicious user, it will be filtered using ettercap. Ettercap display the IP and MAC addresses of all hosts connected to wireless network as shown in Fig. 5. If the attacker is not using ARP spoofing then it will be easily filtered here.

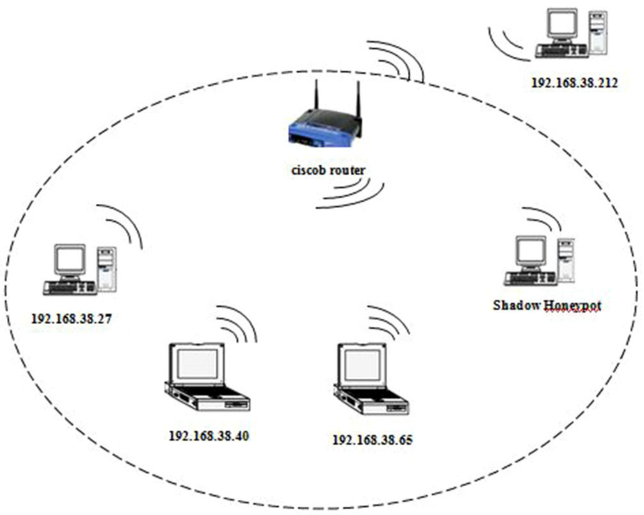
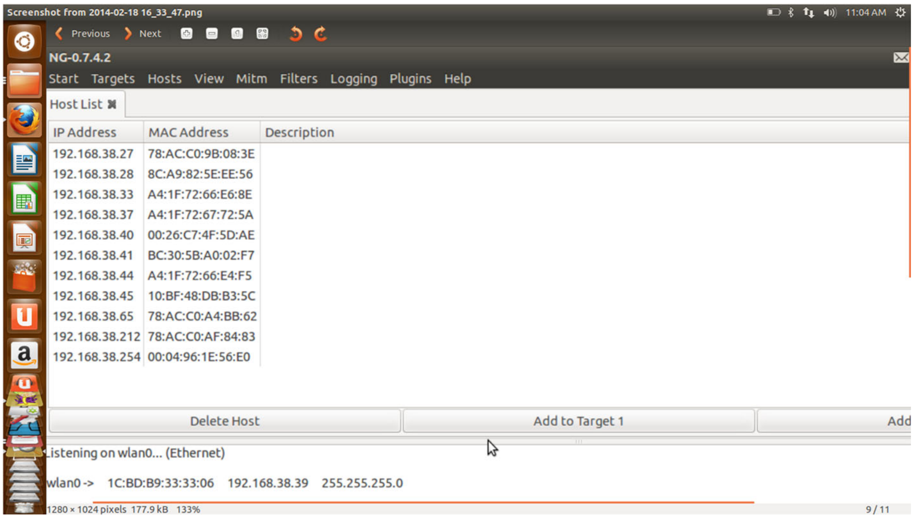


Fig. 4 Home network


Screenshot from 2014-02-18 16_33_47.png
11:04 AM
Previous
Next
NG-0.7.4.2
Start Targets Hosts View Mitm Filters Logging Plugins Help
Host List
Delete Host
Add to Target 1
Add
1280 × 1024 pixels 177.9 kB 133%
9 / 11

IP Address	MAC Address	Description
192.168.38.27	78:AC:C0:9B:08:3E	
192.168.38.28	8C:A9:82:5E:EE:56	
192.168.38.33	A4:1F:72:66:E6:8E	
192.168.38.37	A4:1F:72:67:72:5A	
192.168.38.40	00:26:C7:4F:5D:AE	
192.168.38.41	BC:30:5B:A0:02:F7	
192.168.38.44	A4:1F:72:66:E4:F5	
192.168.38.45	10:BF:48:DB:B3:5C	
192.168.38.65	78:AC:C0:A4:BB:62	
192.168.38.212	78:AC:C0:AF:84:83	
192.168.38.254	00:04:96:1E:56:E0	

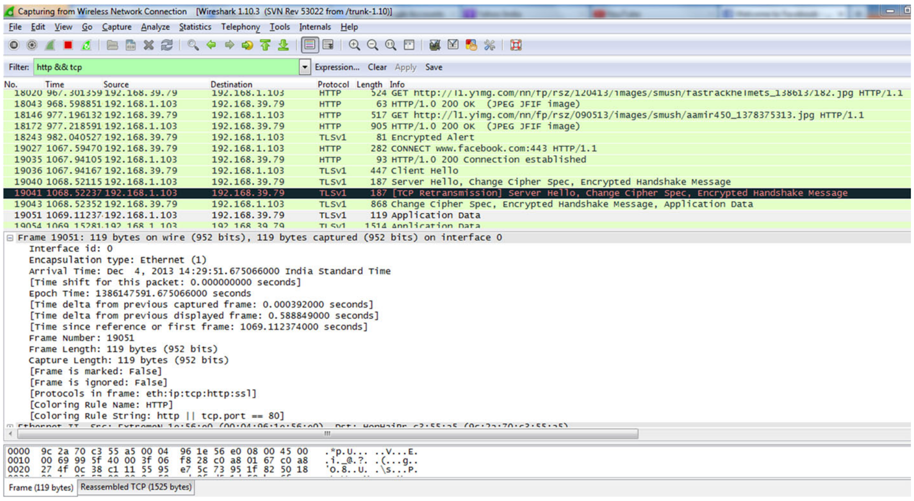
```
Listening on wlan0... (Ethernet)

wlan0 ->    1C:BD:B9:33:33:06    192.168.38.39    255.255.255.0
```

Fig. 5 IP and MAC addresses of all hosts using ettercap

Considering Figs. 4 and 5, the IP addresses except 192.168.38.27, 192.168.38.40 and 192.168.38.65 all are unauthorized hence filtered out. But if the attacker is using ARP spoofing then it will not be filtered here. To detect some attacks, wireshark is used to find out the packets' arrival time. The implementation of such attacks may be harmful for router, so the the paper calculated packet flow rate for wired and wireless network. The packet flow rate for wired network is more than that of wireless network, due to the increment of one hop count in wireless network (considering Figs. 7, 8). Figure 6 shows the packets captured using wireshark and its arrival time.

Figures 6 and 8 shows the packet's arrival time and flow rate for wireless network respectively. MITM attack is to be signaled if packet spacing is large i.e., less flow rate. If

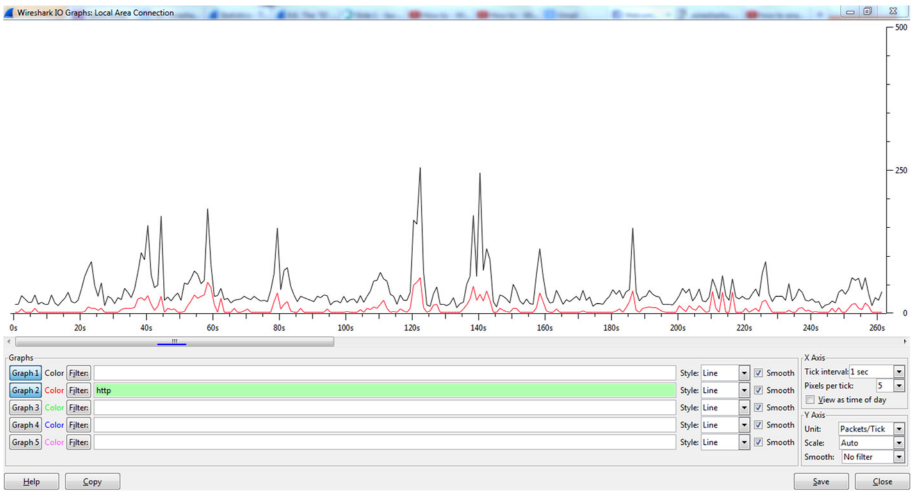

Capturing from Wireless Network Connection [Wireshark 1.10.3 (SVN Rev 53022 from /trunk-1.10)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter: http && tcp
Expression... Clear Apply Save
Frame (119 bytes)
Reassembled TCP (1525 bytes)

```
No.   Time       Source         Destination     Protocol Length Info
18020 967.301359 192.168.39.79  192.168.1.103   HTTP      524 GET http://l1.yimg.com/nn/fp/rsz/120413/images/smush/fastrackhelmets_138613/182.jpg HTTP/1.1
18043 968.598851 192.168.1.103  192.168.39.79   HTTP       63 HTTP/1.0 200 OK  (JPEG JFIF image)
18146 977.196132 192.168.39.79  192.168.1.103   HTTP      517 GET http://l1.yimg.com/nn/fp/rsz/090513/images/smush/aamir450_1378375313.jpg HTTP/1.1
18172 977.218591 192.168.1.103  192.168.39.79   HTTP      905 HTTP/1.0 200 OK  (JPEG JFIF image)
18243 982.040527 192.168.39.79  192.168.1.103   TLSv1      81 Encrypted Alert
19027 1067.59470 192.168.39.79  192.168.1.103   HTTP      282 CONNECT www.facebook.com:443 HTTP/1.1
19035 1067.94105 192.168.1.103  192.168.39.79   HTTP       93 HTTP/1.0 200 Connection established
19036 1067.94167 192.168.39.79  192.168.1.103   TLSv1     447 Client Hello
19040 1068.52115 192.168.1.103  192.168.39.79   TLSv1     187 Server Hello, Change Cipher Spec, Encrypted Handshake Message
19041 1068.52237 192.168.1.103  192.168.39.79   TLSv1     187 [TCP Retransmission] Server Hello, Change Cipher Spec, Encrypted Handshake Message
19043 1068.52352 192.168.39.79  192.168.1.103   TLSv1     868 Change Cipher Spec, Encrypted Handshake Message, Application Data
19051 1069.11237 192.168.1.103  192.168.39.79   TLSv1     119 Application Data
19054 1069.15281 192.168.1.103  192.168.39.79   TLSv1    1514 Application Data

⊟ Frame 19051: 119 bytes on wire (952 bits), 119 bytes captured (952 bits) on interface 0
    Interface id: 0
    Encapsulation type: Ethernet (1)
    Arrival Time: Dec  4, 2013 14:29:51.675066000 India Standard Time
    [Time shift for this packet: 0.000000000 seconds]
    Epoch Time: 1386147591.675066000 seconds
    [Time delta from previous captured frame: 0.000392000 seconds]
    [Time delta from previous displayed frame: 0.588849000 seconds]
    [Time since reference or first frame: 1069.112374000 seconds]
    Frame Number: 19051
    Frame Length: 119 bytes (952 bits)
    Capture Length: 119 bytes (952 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: eth:ip:tcp:http:ssl]
    [Coloring Rule Name: HTTP]
    [Coloring Rule String: http || tcp.port == 80]
⊞ Ethernet II, Src: ExtremeN_1e:56:e0 (00:04:96:1e:56:e0), Dst: HonHaiPr_c3:55:a5 (9c:2a:70:c3:55:a5)

0000  9c 2a 70 c3 55 a5 00 04  96 1e 56 e0 08 00 45 00   .*p.U... ..V...E.
0010  00 69 99 5f 40 00 3f 06  f8 28 c0 a8 01 67 c0 a8   .i._@.?. .(...g..
0020  27 4f 0c 38 c1 11 55 95  e7 5c 73 95 1f 82 50 18   'O.8..U. .\s...P.
```

Fig. 6 Packet capturing using wireshark

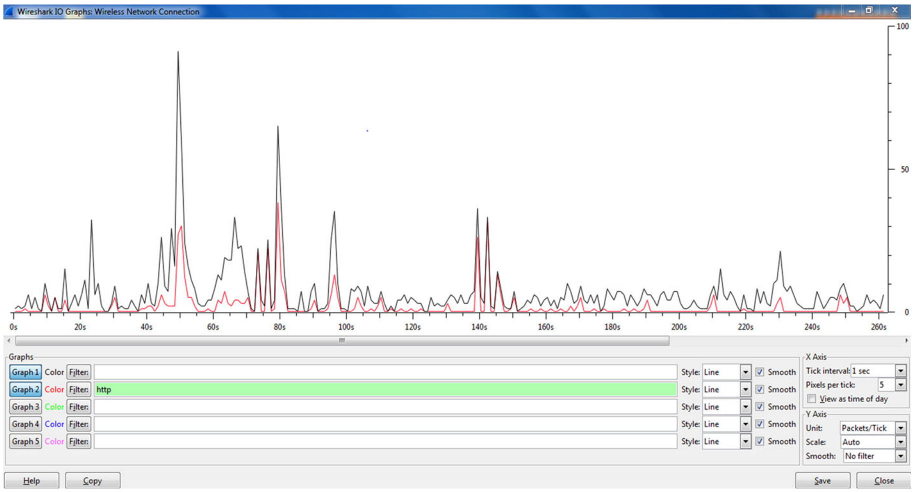


Graphs

Graph 1	Color	Filter:		Style: Line	☑ Smooth
Graph 2	Color	Filter:	http	Style: Line	☑ Smooth
Graph 3	Color	Filter:		Style: Line	☑ Smooth
Graph 4	Color	Filter:		Style: Line	☑ Smooth
Graph 5	Color	Filter:		Style: Line	☑ Smooth

X Axis
Tick interval: 1 sec
Pixels per tick: 5
☐ View as time of day
Y Axis
Unit: Packets/Tick
Scale: Auto
Smooth: No filter

Fig. 7 Packet flow rate for wired network



Graphs

Graph 1	Color	Filter:		Style: Line	☑ Smooth
Graph 2	Color	Filter:	http	Style: Line	☑ Smooth
Graph 3	Color	Filter:		Style: Line	☑ Smooth
Graph 4	Color	Filter:		Style: Line	☑ Smooth
Graph 5	Color	Filter:		Style: Line	☑ Smooth

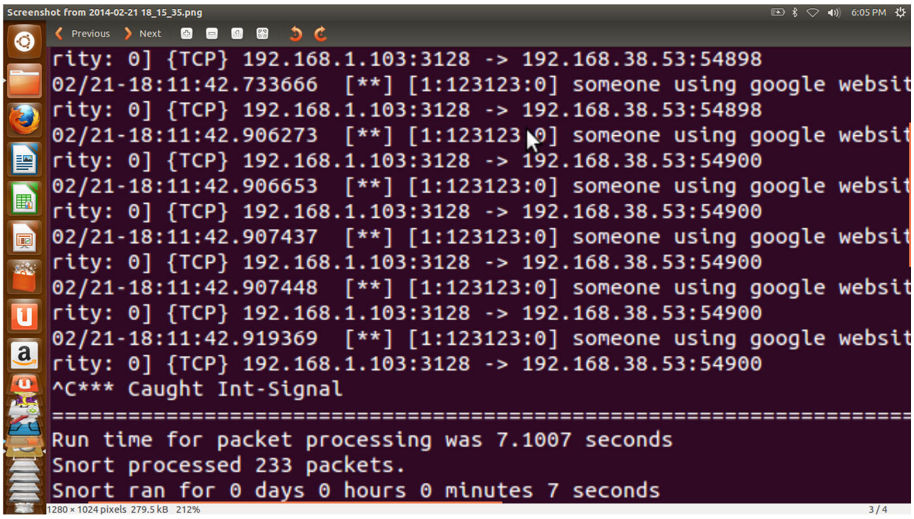
X Axis
Tick interval: 1 sec
Pixels per tick: 5
☐ View as time of day
Y Axis
Unit: Packets/Tick
Scale: Auto
Smooth: No filter

Fig. 8 Packet flow rate for wireless network

packet spacing is too less i.e., flow rate is too high, then DoS, DDoS and Smurf like attacks is detected. Using snort, paper captured more information about packets as well as generating alerts based on specified rules (considering Fig. 9). The figure show alert message that IP address 192.168.38.53 is accessing google within wireless network.

5.3 Overview of TXL

The tool which is used to create the shadow of the normal application is Tree Transformation Language (TXL) [19–21] is a rapid prototyping system and programming language



```
rity: 0] {TCP} 192.168.1.103:3128 -> 192.168.38.53:54898
02/21-18:11:42.733666  [**] [1:123123:0] someone using google websit
rity: 0] {TCP} 192.168.1.103:3128 -> 192.168.38.53:54898
02/21-18:11:42.906273  [**] [1:123123:0] someone using google websit
rity: 0] {TCP} 192.168.1.103:3128 -> 192.168.38.53:54900
02/21-18:11:42.906653  [**] [1:123123:0] someone using google websit
rity: 0] {TCP} 192.168.1.103:3128 -> 192.168.38.53:54900
02/21-18:11:42.907437  [**] [1:123123:0] someone using google websit
rity: 0] {TCP} 192.168.1.103:3128 -> 192.168.38.53:54900
02/21-18:11:42.907448  [**] [1:123123:0] someone using google websit
rity: 0] {TCP} 192.168.1.103:3128 -> 192.168.38.53:54900
02/21-18:11:42.919369  [**] [1:123123:0] someone using google websit
rity: 0] {TCP} 192.168.1.103:3128 -> 192.168.38.53:54900
^C*** Caught Int-Signal
=====================================================================
Run time for packet processing was 7.1007 seconds
Snort processed 233 packets.
Snort ran for 0 days 0 hours 0 minutes 7 seconds
```

Screenshot from 2014-02-21 18_15_35.png
6:05 PM
Previous
Next
1280 × 1024 pixels 279.5 kB 212%
3 / 4

Fig. 9 Generating alerts using snort

used for rule based source-to-source transformation. Source structure that is to be transformed is in extended Backus–Naur Form (BNF) using unrestricted ambiguous context free grammar, from which a structure parser is automatically generated. Below is the transformation rule, which convert two scalar assignments that are independent of each other into a single vector assignment.

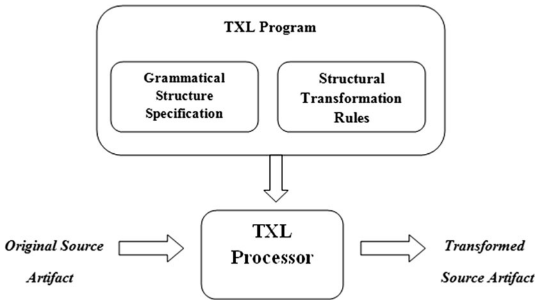
```
rule vectorizeScalarAssignments
    replace [repeat statement]
        V1 [variable] := E1 [expression];
        V2 [variable] := E2 [expression];
        RestOfScope [repeat statement]
    where not
        E2 [references V1]
    where not
        E1 [references V2]
    by
        < V1, V2 > := < E1, E2 >;
        RestOfScope
end rule
```

The *replace* clause gives the pattern for which the rule searches in actual source text. The *by* clause gives the transformed result in similar style. The *where* clause specifies additional semantic constraints on which the rule can be applied.

5.3.1 The TXL Processor

Figure 10 shows that TXL program consist of *Grammar Structure Specification* and Transformation Rules. Grammar Structure Specification defines the lexical and syntactic forms of the input and in transformation rules actual input-to-output source transformation is specified. The TXL processor is a compiler and run time system that converts the one source code into other source code [22].

Fig. 10 The TXL processor

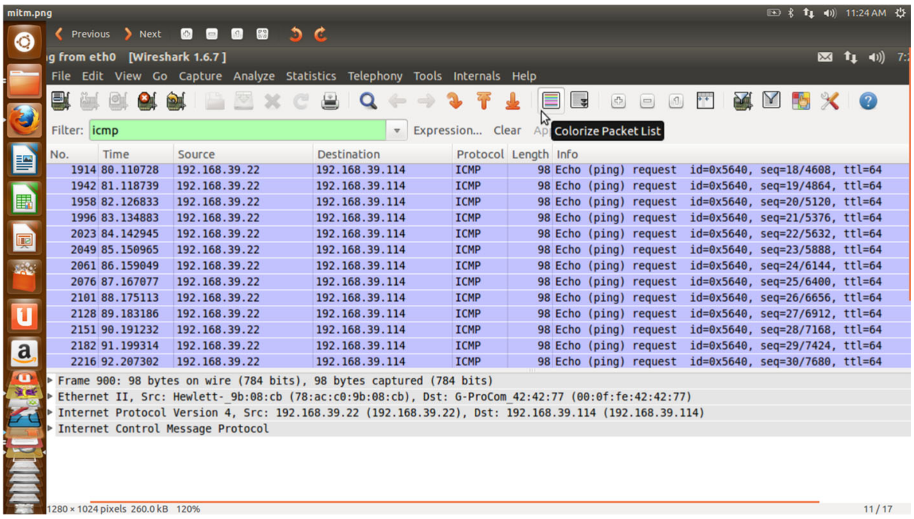

TXL Program
Grammatical Structure Specification
Structural Transformation Rules
Original Source Artifact
TXL Processor
Transformed Source Artifact

6 Experimental Evaluation

6.1 ARP Cache Poisoning Attack

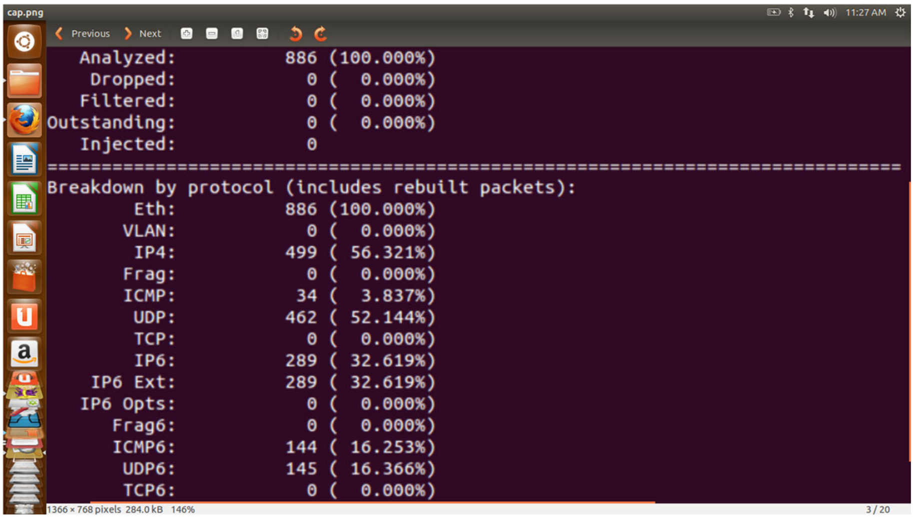
ARP Cache Poisoning is a form of Man-in-the-Middle attack in which an attacker sends his/her MAC address with the authenticated IP address in ARP_REPLY message. This attack has been implemented using IP addresses 192.168.39.22, 192.168.39.114 and 192.168.39.76. IP address 192.168.39.76 treated as an eavesdropper between authenticated IP addresses 192.168.39.22 and 192.168.39.114 (considering Figs. 11, 12). All the traffic flowing between 192.168.39.22 and 192.168.39.114 is captured by 192.168.39.76. Figure 11 is the screenshot of attacker's system showing all the captured packets when 192.168.39.22 pinged 192.168.39.114.

At second stage, ARP soofing attack is detected using snort. Figure 12 show the packets captured by snort. Since ping supports ICMP protocol, the number of ICMP packets captured by snort are 34.


mitm.png
11:24 AM
Previous
Next
g from eth0 [Wireshark 1.6.7]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter: icmp
Expression...
Clear
Ap
Colorize Packet List
1280 × 1024 pixels 260.0 kB 120%
11 / 17

```
No.   Time      Source         Destination     Protocol Length Info
 1914 80.110728 192.168.39.22  192.168.39.114  ICMP         98 Echo (ping) request  id=0x5640, seq=18/4608, ttl=64
 1942 81.118739 192.168.39.22  192.168.39.114  ICMP         98 Echo (ping) request  id=0x5640, seq=19/4864, ttl=64
 1958 82.126833 192.168.39.22  192.168.39.114  ICMP         98 Echo (ping) request  id=0x5640, seq=20/5120, ttl=64
 1996 83.134883 192.168.39.22  192.168.39.114  ICMP         98 Echo (ping) request  id=0x5640, seq=21/5376, ttl=64
 2023 84.142945 192.168.39.22  192.168.39.114  ICMP         98 Echo (ping) request  id=0x5640, seq=22/5632, ttl=64
 2049 85.150965 192.168.39.22  192.168.39.114  ICMP         98 Echo (ping) request  id=0x5640, seq=23/5888, ttl=64
 2061 86.159049 192.168.39.22  192.168.39.114  ICMP         98 Echo (ping) request  id=0x5640, seq=24/6144, ttl=64
 2076 87.167077 192.168.39.22  192.168.39.114  ICMP         98 Echo (ping) request  id=0x5640, seq=25/6400, ttl=64
 2101 88.175113 192.168.39.22  192.168.39.114  ICMP         98 Echo (ping) request  id=0x5640, seq=26/6656, ttl=64
 2128 89.183186 192.168.39.22  192.168.39.114  ICMP         98 Echo (ping) request  id=0x5640, seq=27/6912, ttl=64
 2151 90.191232 192.168.39.22  192.168.39.114  ICMP         98 Echo (ping) request  id=0x5640, seq=28/7168, ttl=64
 2182 91.199314 192.168.39.22  192.168.39.114  ICMP         98 Echo (ping) request  id=0x5640, seq=29/7424, ttl=64
 2216 92.207302 192.168.39.22  192.168.39.114  ICMP         98 Echo (ping) request  id=0x5640, seq=30/7680, ttl=64
▸ Frame 900: 98 bytes on wire (784 bits), 98 bytes captured (784 bits)
▸ Ethernet II, Src: Hewlett-_9b:08:cb (78:ac:c0:9b:08:cb), Dst: G-ProCom_42:42:77 (00:0f:fe:42:42:77)
▸ Internet Protocol Version 4, Src: 192.168.39.22 (192.168.39.22), Dst: 192.168.39.114 (192.168.39.114)
▸ Internet Control Message Protocol
```

Fig. 11 Captured packets during ARP spoofing attack using wireshark



```
    Analyzed:          886 (100.000%)
     Dropped:            0 (  0.000%)
    Filtered:            0 (  0.000%)
 Outstanding:            0 (  0.000%)
    Injected:            0
================================================================================
Breakdown by protocol (includes rebuilt packets):
         Eth:          886 (100.000%)
        VLAN:            0 (  0.000%)
         IP4:          499 ( 56.321%)
        Frag:            0 (  0.000%)
        ICMP:           34 (  3.837%)
         UDP:          462 ( 52.144%)
         TCP:            0 (  0.000%)
         IP6:          289 ( 32.619%)
     IP6 Ext:          289 ( 32.619%)
    IP6 Opts:            0 (  0.000%)
       Frag6:            0 (  0.000%)
       ICMP6:          144 ( 16.253%)
        UDP6:          145 ( 16.366%)
        TCP6:            0 (  0.000%)
```

Fig. 12 Captured packets during ARP spoofing attack using snort

Since the IP addesses 192.168.39.22, 192.168.39.114 and 192.168.39.76 are authorized, so attacker could not be detected by ettercap. According to the recieved ICMP supports packets, MITM attack is signaled by snort. After that the request has been forwarded to the shadow honeypot for validation. Calculated false positive rate in both cases is given below.

False positive rate is calculated using *Specificity*. Specificity (sometimes called the true negative rate) measures the proportion of correctly identified authorized users.

$$FalsePositiveRate = 1 - Specificity$$

$$Specificity = \frac{NumberofTrueNegatives}{NumberofTrueNegatives + NumberofFalsePositives} = \frac{NormalUserIdentifiedasNormal}{NormalUserIdentifiedasNormal + NormalUserIdentifiedasAttacker}$$

Case I: False positive rate using anomaly detection system:

Using ADS, the users having IP addresses 192.168.39.22 and 192.168.39.114 has been identified correctly and the IP address 192.168.39.76 misclassified as attacker.

$$\begin{aligned} Specificity &= \frac{5}{5+3} = 0.62 \\ FalsePositiveRate &= 1 - 0.62 \\ &= 0.38 \\ &= 38\,\% \end{aligned}$$

There were total eight systems in network on which ARP Spoofing attack is performed, out of which the paper mentioned only three (192.168.39.22, 192.168.39.114 and 192.168.39.76).

Case II: False positive rate using shadow honeypot:

The proposed method improves overall performance by reducing false positive rate. Since the system is designed sensitive to attacks, ADS will misclassify legitimate users as attacker. Misclassified authorized users is then forwarded to shadow honeypot, where it will be handled carefully, thus minimizing false positive rate.

Suppose using proposed architecture, the user having IP address 192.168.39.76 that was misclassified as attacker using ADS, identified correctly by shadow and carefully handled by production software.

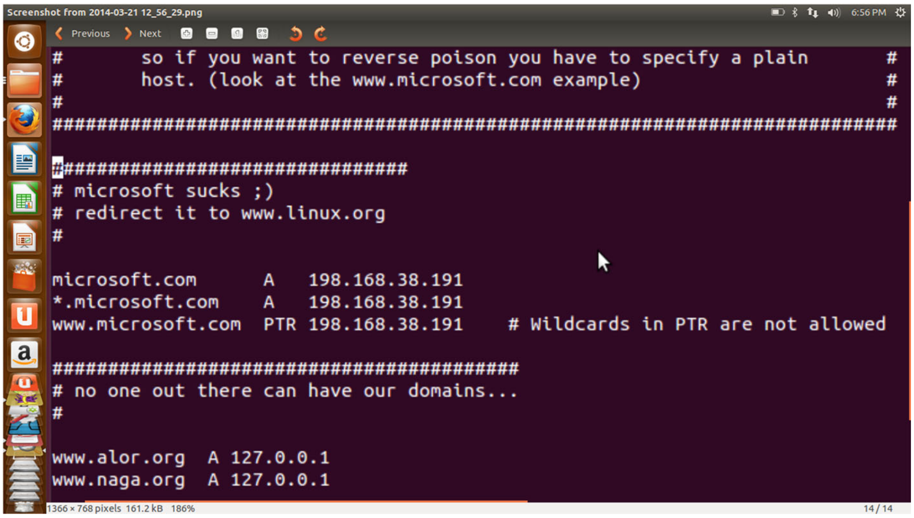
$$\begin{aligned} Specificity = \frac{6}{6+2} &= \quad 0.75 \\ FalsePositiveRate &= 1 - 0.75 \\ &= 0.25 \\ &= 25\,\% \end{aligned}$$

Thus, false positive rate decreases.

6.2 DNS Spoofing Attack

DNS Spoofing is also a form of Man-in-the-Middle attack which is used to provide fake IP address when servicing the DNS_REQUEST. So that when the requested host browse for a website at IP address W.X.Y.Z, the request is forwarded to fake site created by attacker at IP address A.B.C.D, so that an attacker can be able to steal users' credentials.

The paper implemented this attack using two systems having IP addresses 192.168.38.38 and 192.168.38.191. The system 192.168.38.191 behaves as an attacker and system 192.168.38.38 is a victim. Attacker spoofs victims's IP address using ettercap to capture all traffic flowing from his/her machine, and enable *dns_spoof* plug-in on his/her own machine using ettercap. Attacker performs DNS spoofing for microsoft website and redirect it to itself. So that whenever victim's machine browse for that site, the request is directed towards attacker's machine instead. To do this, some changes is to be done in *etter.dns* file. Figure 13 shows the modified etter.dns file where the IP address corresponding to microsoft.com is changed to the attacker's IP address.


Screenshot from 2014-03-21 12_56_29.png
6:56 PM
Previous
Next
1366 × 768 pixels 161.2 kB 186%
14 / 14

```
#       so if you want to reverse poison you have to specify a plain      #
#       host. (look at the www.microsoft.com example)                     #
#                                                                         #
###########################################################################

################################
# microsoft sucks ;)
# redirect it to www.linux.org
#

microsoft.com      A   198.168.38.191
*.microsoft.com    A   198.168.38.191
www.microsoft.com  PTR 198.168.38.191    # Wildcards in PTR are not allowed

##########################################
# no one out there can have our domains...
#

www.alor.org  A 127.0.0.1
www.naga.org  A 127.0.0.1
```

Fig. 13 Adding a spoofed DNS record to etter.dns

After making the changes in *etter.dns* file, attacker runs the command "***ettercap -T -q -M arp -i eth0 -P dns_spoof // //***" for activating DNS spoofing. After that when victim's machine browse for microsoft website it will redirect to attacker's machine. Figure 14 shows the dns spoofing where victim's machine spoofed to IP address 192.168.38.191.

At second stage Snort is used to detect this type of attack using below signature.

alert udp $ EXTERNAL_NET 53 - > $ HOME_NET any (msg:"DNS SPOOF query response PTR with TTL 1 min. and no authority"; content:"|85800001000100000000|"; content:"|c00c000c000100 00003c000f|"; classtype:bad-unknown; sid:253; rev:2;)

The signature detects DNS cache poisoning attempts, packets with large TTL values and DNS responses coming from malevolent DNS servers.

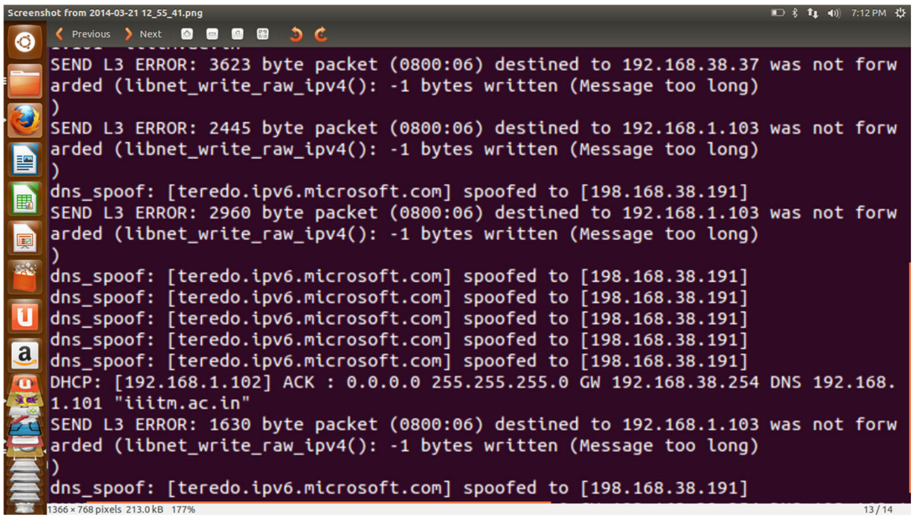
After that the request is sent to third stage i.e., towards shadow honeypot for further analysis. The paper performed dns_spoofing for three websites, microsoft.com, facebook.com and google.com, but IDS signaled alerts for five sites after accessing six sites (microsoft, facebook, google, yahoo, youtube and rediff), since the system is sensitive to attacks. Honeypot signaled fake IP address for four sites, and false positive for one site. The result of shadow honeypot is reflected to ADS and filtering component for preventing future attacks. The calculated false positive rate in both cases using above formula are:

$$\mathit{False\,Positive\,Rate\,without\,Shadow\,Honeypot} = 0.67\,\%$$
$$\mathit{False\,Positive\,Rate\,with\,Shadow\,Honeypot} = 0.34\,\%$$

Result shows the reduction in false positive rate.

6.3 Payload Sifting

Payload Sifting [12, 18] identify similar contents in the payload of several packets heading to numerous destination hosts, and consider them as fingerprints of rapidly spreading worms. The generated fingerprints are disseminated to shadow honeypot for matching network traffic against it. The pseudocode for the same is given below:


Screenshot from 2014-03-21 12_55_41.png
7:12 PM
Previous
Next
1366 × 768 pixels 213.0 kB 177%
13 / 14

```
SEND L3 ERROR: 3623 byte packet (0800:06) destined to 192.168.38.37 was not forw
arded (libnet_write_raw_ipv4(): -1 bytes written (Message too long)
)
SEND L3 ERROR: 2445 byte packet (0800:06) destined to 192.168.1.103 was not forw
arded (libnet_write_raw_ipv4(): -1 bytes written (Message too long)
)
dns_spoof: [teredo.ipv6.microsoft.com] spoofed to [198.168.38.191]
SEND L3 ERROR: 2960 byte packet (0800:06) destined to 192.168.1.103 was not forw
arded (libnet_write_raw_ipv4(): -1 bytes written (Message too long)
)
dns_spoof: [teredo.ipv6.microsoft.com] spoofed to [198.168.38.191]
dns_spoof: [teredo.ipv6.microsoft.com] spoofed to [198.168.38.191]
dns_spoof: [teredo.ipv6.microsoft.com] spoofed to [198.168.38.191]
dns_spoof: [teredo.ipv6.microsoft.com] spoofed to [198.168.38.191]
dns_spoof: [teredo.ipv6.microsoft.com] spoofed to [198.168.38.191]
DHCP: [192.168.1.102] ACK : 0.0.0.0 255.255.255.0 GW 192.168.38.254 DNS 192.168.
1.101 "iiitm.ac.in"
SEND L3 ERROR: 1630 byte packet (0800:06) destined to 192.168.1.103 was not forw
arded (libnet_write_raw_ipv4(): -1 bytes written (Message too long)
)
dns_spoof: [teredo.ipv6.microsoft.com] spoofed to [198.168.38.191]
```

Fig. 14 Adding a spoofed DNS record to etter.dns

```
for each recieved packet
    for each fingerprint in packet
        if fingerprint in cache
            if packet destination not recorded
                mark as new
                record destination
                if destinations >threshold
                report
        else
            create new entry for fingerprint
            record destination
for ward fingerprint to shadow honeypot
    analyze network traffic against fingerprints
        measured false positive
```

Figure 15 show the false positives by varying destination threshold. Increasing the threshold means for detecting attack more instances are required. As the destination threshold increases, detection delay increases and false positive reduces.

6.4 Buffer Overflow Attack

Presence of long sequence of valid instructions in the network traffic signaled buffer overflow attack. Since, these sequences can appear in authorized data, this is where shadow honeypot is used for validation [12, 23]. Attacker uses sequence of *[NOP][Shell code][Return address]* to implement buffer overflow attack. Snort is used to looking for this sled in network traffic. To diminish false positie rate the packet is forwarded towards shadow honeypot. Below is the shadow code for buffer overflow attack; the decision on whether the shadow code or the normal code should be executed depends on the *shadow_enable()* macro that simply checks the status of shared-memory variable.

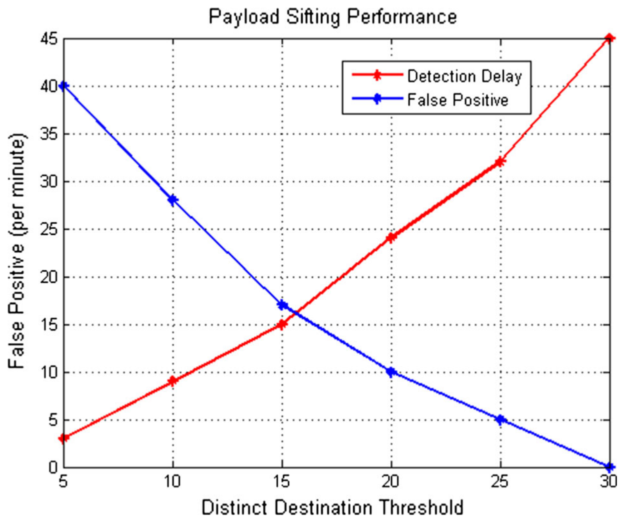


Fig. 15 False positive for payload sifting

Original Code

```
int func ()
{
    char buffer[50];
...
    func1(buffer, sizeof(buffer));
...
    return 0;
}
```

Modified Code

```
int func ()
{
    if shadow_enable()
        char *buffer = pmalloc(50);
    else
        char *buffer = buffer1[50];
...
    func1(buffer, sizeof(buffer));
...
    if shadow_enable()
        pfree(buffer);
    return 0;
}
```

Figure 16 show the effects of number of sled instructions on false positive rate. False positive rate decreases by increasing number of sled instructions.

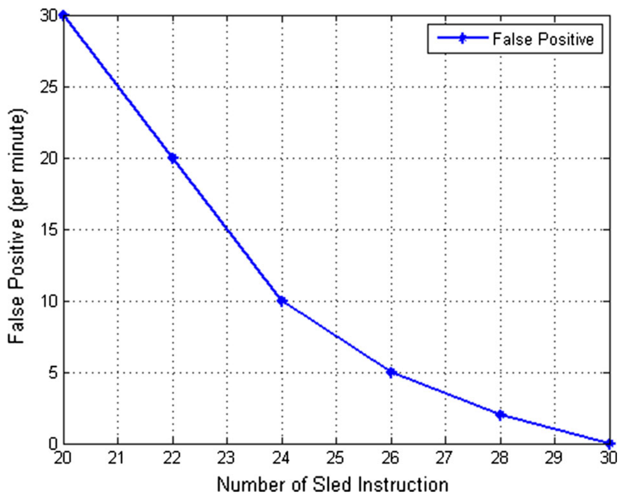


Fig. 16 False positive for buffer overflow attack

Table 1 Attacks with their false positives in both cases

Attack	Time for which attack is performed	Time taken by SH	False positive without SH %	False positive with SH %
ARP cache poisoning	75	600	38	25
DNS spoofing	35	432	67	34
Payload sifting	980	1200	37	15
Buffer overflow	600	1350	31	17
Dos, DDoS smurf	20	27	35	29

Time is in seconds

SH shadow honeypot, *FP* false positive

6.5 Denial-of-Service, Distributed Denial-of-Service, Smurf Attacks

If the inter-packet spacing is too less, then these attacks is signaled by ADS. There may be a case that packet-spacing is less for non-malevolent data. To reduce false positive alerts and carefully handling of such request shadow honeypot is used. The pseudocode is given below:

```
analyze packet flow (PF) rate using wireshark;
  if PF <threshold
    report attack by ADS and forward to shadow honeypot;
      if attack detected
        signal hit to ADS and filtering component
      else
        indicate false positive;
  else
    use shadow anyway;
      if attack detected
        indicate false negative;
      else
        signal hit to ADS and filtering component;
```

The request that was misclassified as malicious by ADS will be handled carefully. Thus, false positive rate decreases.

Table 1 shows the attacks, the time for which the particular attack is performed, the time taken by shadow honeypot to detect it along with their false positives in both cases: without shadow honeypot and using shadow honeypot.

7 Conclusion

During the past several years, installation of wireless LAN (802.11) in military and organization is tremendously increasing. To secure such networks it is necessary to detect and prevent the installation of rogue access point. In this paper, a novel hybrid approach that combines the best features found in today's honeypots and anomaly detection system has been proposed. The system improves the performance by minimizing false positive rate because all the activities that were misclassified as malicious by the anomaly detection sensors will be handled carefully by the production software transparently to the end users. Various anomaly detectors are used which analyze the traffic directed towards the protected network. Despite this, the system is capable to sustain the overall workload of honeypots.

Enhance the performance by detecting more attacks at second stage and implementation of multiple shadow honeypots to provide real system environment to attacker.

References

1. Fluhrer, S., Mantin, I., & Shamir, A. (2001). *Weaknesses in the key scheduling algorithm of RC4*. Berlin: Springer.
2. Borisov, N., Goldberg, I., & Wagner, D. (2001). Intercepting mobile communications: The insecurity of 802.11. In *Proceedings of the seventh annual international conference on mobile computing and networking*, ACM.
3. Bittau, A., Handley, M., Lackey, J. (2006). The final nail in WEPs coffin. In *IEEE symposium on security and privacy*.
4. Tews, E., Weinmann, R. P., & Pyshkin, A. (2007). *Breaking 104 bit WEP in less than 60 seconds*. Berlin: Springer.
5. Ma, L., Teymorian, A. Y., & Cheng, X. (2007). RAP: Protecting commodity Wi-Fi networks from rogue access points. In *The fourth international conference on heterogeneous networking for quality, reliability, security and robustness and workshops*, ACM.
6. Beyah, R., Kangude, S., Yu, G., Strickland, B., & Copeland, J. (2004). Rogue access point detection using temporal traffic characteristics. In *GLOBECOM*.
7. Motorola Solutions. (2011). *AirDefense enterprise: A wireless intrusion prevention*.
8. Adya, A., Bahl, P., Chandra, R., & Qiu, L. (2004). Architecture and techniques for diagnosing faults in ieee 802.11 infrastructure networks. In *MobiCom 04* (pp. 30–44).
9. Bahl, P., Chandra, R., Padhye, J., Ravindranath, L., Singh, M., Wolman, A., et al. (2006). Enhancing the security of corporate wi-fi networks using dair. In *MobiSys 06* (p. 114), ACM Press.
10. Chirumamilla, M. K., & Ramamurthy, B. (2003). Agent based intrusion detection and response system for wireless LANs. In *ICC 03* (pp. 492–496).
11. Gomez, Z., Gil, C., Padilla, N., Banos, R., & Jimenez, C. (2009). *Design of SNORT based hybrid intrusion detection sysytem*. Berlin: Springer.
12. Anagnostakis, K. G., Sidiroglou, S., Akritidis, P., Polychronakis, M., Keromytis, A. D., & Markatos, E.P. (2010). Shadow Honeypots. *International Journal of Computer and Network Security, 2*(9), 1–16.
13. Mustapha, Y. B., Debar, H., & Jacob, G. (2012). *Limitation of honeypot/honeynet databases to enhance alert correlation*. Berlin: Springer.
14. Levine, J., LaBella, R., Owen, H., Contis, D., & Culve, B. (2003). The use of honeynets to detect exploited systems across large enterprise networks. In *Proceedings of the 2003 IEEE workshop on information assurance*. West Point, NY: United States Military Academy.
15. http://ettercap.github.io/ettercap/
16. http://www.wireshark.org/
17. http://www.snort.org/
18. Singh, S., Estan, C., Varghese, G., & Savage, S. (2004). Automated worm fingerprinting. In *Proceedings of the 6th symposium on operating systems design and implementation (OSDI)*.
19. http://www.txl.ca/txldocs.html
20. http://www.txl.ca/learningtxl.html
21. http://www.txl.ca/txlworld.html
22. Cordy, J. R. (2001). *A practical introduction to TXL*.
23. Sidiroglou, S., Giovanidis, G., & Keromytis, A. D. (2005). A dynamic mechanism for recovering from buffer overflow attacks. In *Proceedings of the 8th international conference, ISC 2005, Singapore*. Berlin: Springer.

Neha Agrawal is currently persuing her M.Tech. Computer Science and Engineering 9 Specialization Information Security) from ABV-Indian Institute of Information Technology and Management, Gwalior. She completed her B.E. from Madhav Institute of Technology and Science, Gwalior in information security.

Shashikala Tapaswi is a Professor in Information Technology Department of Atal Bihari Vajpayee—Indian Institute of Information Technology and Management, Gwalior, Madhya Pradesh, India. Her primary research areas of interest are Computer Networks, Network Security,Mobile Adhoc Networks, Artificial Intelligence, Neural Network, Fuzzy Logic, Digital Image Processing.