Real-World ARP Attacks and Packet Sniffing, Detection and Prevention on Windows and Android Devices

Blagoj Nenovski and Pece Mitrevski
Faculty of Information and Communication Technologies
"St. Clement of Ohrid" University
Bitola, Macedonia
cyberbaze@gmail.com, pece.mitrevski@fikt.edu.mk

***Abstract*—This paper explains the purpose and the need of the Address Resolution Protocol (ARP) and different types of attacks that can occur due to its stateless nature. We exhibit a real world example of a Man-in-the-Middle (MitM) attack by sniffing http logins of a Windows PC and an Android device, and then suggest methods of both detecting and preventing the attack.**

Keywords—ARP spoofing; ARP poisoning; DoS; MitM; packet capture; network sniffing; WireShark; network forensics

I. INTRODUCTION

Every device connected to a LAN network has two addresses: a 32-bit IP address which can be static or dynamic and a 48-bit MAC address that is static. When a device with its own IP address sends packets to another device in the network the sender needs to know the receiver's MAC address so the communication continues from IP (Layer 3) to MAC (Layer 2). This created the need of an Address Resolution Protocol (ARP).

ARP was defined by RFC 826 in 1982 [1] and has the role to broadcast a message across the network and to determine the MAC address (Layer2) of a host with a set IP (Layer 3) both static or dynamic by DHCP. This message is replied with an ARP packet that contains the MAC address. The ARP protocol has four message types:

- ARP Request – When a device asks for MAC address by sending an IP address;
- ARP Reply – When a device replies with its IP and providing its MAC address;
- Reverse ARP request – When a device asks who has a certain MAC address;
- RARP Reply – When a device replies with its MAC and providing its IP address.

In order to minimize network traffic devices remember the MAC addresses in an Arp cache. This way, devices that already had previous communication eliminate the need to repeat the ARP requests and replies. Fig. 1 illustrates arp packets that devices send to acknowledge the MAC addresses of other devices connected to the network. ARP is a stateless protocol and does not include any authentication which makes it vulnerable to various attacks.

Noteworthy is to mention that Wi-Fi Protected Access II (WPA2), the Wi-Fi Alliance branded version of the newest and most rigorous security to implement into WLAN's today, 802.11i, has still been found to have at least one security vulnerability, nicknamed "Hole196", which uses the shared key among all users to launch attacks on other users of the same Basic Service Set identification (BSSID).

II. RELATED WORK

Tripunitara and Dutta [2] discuss the Address Resolution Protocol and the problem of ARP cache poisoning and present a solution and implementation aspects of it in a Streams based networking subsystem. They also present the algorithm that is executed in the module and application to prevent ARP cache poisoning where possible, and detect and raise alarms otherwise. They discuss some limitations and present some preliminary performance numbers for their implementation.

Zdrnja [3] explains ARP attack fundamentals and analyzes recent attacks that used ARP poisoning against Web hosting companies to let attackers insert malicious code into virtually

	Time	Source	Destination	Protocol	Length	Info
239	18:01:19.897624000	IntelCor_59:8c:e7	SamsungE_1b:a4:65	ARP	42	Who has 192.168.100.3? Tell 192.168.100.1
240	18:01:19.897754000	IntelCor_59:8c:e7	Cisco-Li_aa:78:c6	ARP	42	Who has 192.168.100.1? Tell 192.168.100.3
241	18:01:19.913130000	IntelCor_59:8c:e7	AsustekC_0d:78:f2	ARP	42	Who has 192.168.100.2? Tell 192.168.100.1
242	18:01:19.913264000	IntelCor_59:8c:e7	Cisco-Li_aa:78:c6	ARP	42	Who has 192.168.100.1? Tell 192.168.100.2
243	18:01:19.927417000	IntelCor_59:8c:e7	SamsungE_1b:a4:65	ARP	42	Who has 192.168.100.3? Tell 192.168.100.1
244	18:01:19.927560000	IntelCor_59:8c:e7	Cisco-Li_aa:78:c6	ARP	42	Who has 192.168.100.1? Tell 192.168.100.3
245	18:01:19.941570000	IntelCor_59:8c:e7	AsustekC_0d:78:f2	ARP	42	Who has 192.168.100.2? Tell 192.168.100.1
246	18:01:19.941744000	IntelCor_59:8c:e7	Cisco-Li_aa:78:c6	ARP	42	Who has 192.168.100.1? Tell 192.168.100.2
250	18:01:25.749990000	IntelCor_59:8c:e7	Broadcast	ARP	42	Who has 192.168.100.1? Tell 192.168.100.4
251	18:01:25.751637000	Cisco-Li_aa:78:c6	IntelCor_59:8c:e7	ARP	60	192.168.100.1 is at 00:1d:7e:aa:78:c6

Fig. 1. WireShark capture of ARP packets

thousands of Web sites.

Kołodziejczyk and Ogiela [4] describe few, popular security mechanisms used by network protocols. In an attempt to create some basic rules and requirements which should be met by secure network protocols.

Similarly, Abad and Bonilla [5] analyze several schemes to mitigate, detect and prevent ARP spoofing attacks that have been proposed, identify their strengths and weaknesses, and propose guidelines for the design of an alternative and (arguably) better solution to the problem of ARP cache poisoning.

Lootah et al. [6] introduce the Ticket-based Address Resolution Protocol (TARP). TARP implements security by distributing centrally issued secure MAC/IP address mapping attestations through existing ARP messages. They conclude by exploring a range of operational issues associated with deploying and administering ARP security.

Bruschi et al. [7] present a secure version of ARP that provides protection against ARP poisoning. Each host has apublic/private key pair certified by a local trusted party on the LAN, which acts as a Certification Authority. As a proof of concept, the proposed solution was implemented on a Linux box.

Preventing ARP Spoofing Attacks through Gratuitous Decision Packet System (GDPS) is a mechanism suggested by Salim et al. [8] which seeks to achieve two main goals: (1) Detection of suspicious ARP packets, by implementing a real-time analyzing for received ARP packets. (2) The distinction between a legitimate and malicious host through sending a modified request packet of the gratuitous ARP packets.

Qadeer et al. [9] focuses on the basics of a packet sniffer and its working, development of the tool on Linux platform and its use for Intrusion Detection. They also discuss ways to detect the presence of such software on the network and to handle them in an efficient way. Focus has also been laid to analyze the bottleneck scenario arising in the network, using this self-developed packet sniffer. Observation has been made on the working behavior of already existing sniffer software such as wireshark (formerly known as ethereal), tcpdump, and snort, which serve as the base for the development of the sniffer software.

Pansa and Chomsiri [10] present a design of "architecture and protocols" for the LAN security preventing the process of MAC Address spoofing, ARP Spoof and MITM. In the presented system, the operation in Protocol level of DHCP is modified, and the use of ARP Protocol is canceled in order to suit and correspond with the new architecture.

Radhakishan and Selvakumar [11] address the drawbacks of TrueIP and propose a new architecture to overcome them. In addition, all sorts of man-in-the-middle attacks (MIMA) are eliminated in their TrueIP spoofing prevention technique using Identity based cryptography in which a signature scheme is used to achieve better security.

Intrusion Detection Systems (IDS) are able to prevent MitM attacks. Belenguer and Calafate [12] present a low-cost embedded IDS which, when plugged into a switch or hub, is able to detect and/or prevent MitM attacks automatically and efficiently. Since their system is limited to a micro-controller and a network interface, it can be produced at a very low cost, which is attractive for large scale production and deployment.

Callegati et al. [13] discuss Man-in-the-Middle Attacks to the HTTPS Protocol and show that it is possible to attack even Web-based connections secured via HTTPS by exploiting some properties of common LANs as well as typical behaviors of inexperienced users.

Finally, in the paper of Anaya et al. [14] a forensics network model is proposed, which allows for obtaining the existing evidence in an involved TCP/IP network. This Model uses Fuzzy Logic and Artificial Neural Networks to detect Network flows that realize suspicious activities, minimizing cost and time to process the information in order to discriminate between normal network flows and flows that have been subjected to attacks and intrusions.

Whereas Roy et al. [15] focus on using Ettercap as the tool for the attack and suggest using arpwatch to detect and prevent the attacks (both Ettercap and arpwatch are Linux based tools), this paper focuses on Windows tools that can be used for the attack and tools that can detect and prevent the attack both on Windows PCs and Android devices.

III. Types of Attack

A. *Denial of Service Attack*

Denial of service attack or DoS attack (Fig. 2) can be leveraged by sending an ARP reply to either:

- Host – by changing the gateways MAC to a non-existing MAC (ex. FF:FF:FF:FF:FF:00).

This way every time the host sends packets to the gateway, they are sent to FF:FF:FF:FF:FF:00 which results in the host not being able to communicate with the gateway.

- Gateway – by changing the victims MAC to a non-existing MAC (ex. 00:00:00:00:00:FF).

This way every time the gateway sends packets to the victim, they are sent to 00:00:00:00:00:FF which results the victim host not receiving any of the packets.

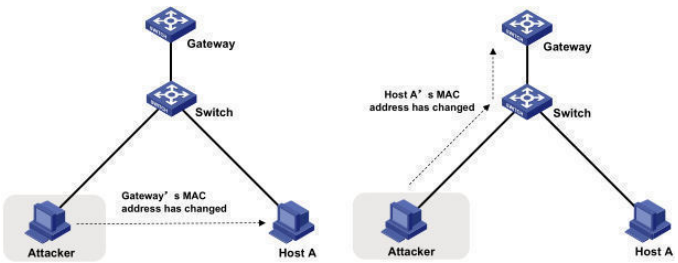

Fig. 2. DoS attack of a host (left) and gateway (right)

There are other techniques that can cause DoS of a host like ARP spoofing a vast amount of IP addresses to a single MAC address. This type of attack can be more effective on larger LAN networks and will cause the victim to be overloaded with traffic. One of the most commonly used tools to perform DoS

attacks is arpspoof, distributed in the dsniff package and Cain & Abel by oxid.

B. *Man in the Middle Attack*

Man in the middle attacks (MitM) are achieved by ARP poisoning both the victim host and the gateway (Fig. 3). The attacker, using ARP poison, sends its MAC address to the victim overwriting the gateway's MAC with its own, and an additional ARP poison to the gateway with the victim's IP its own MAC address.

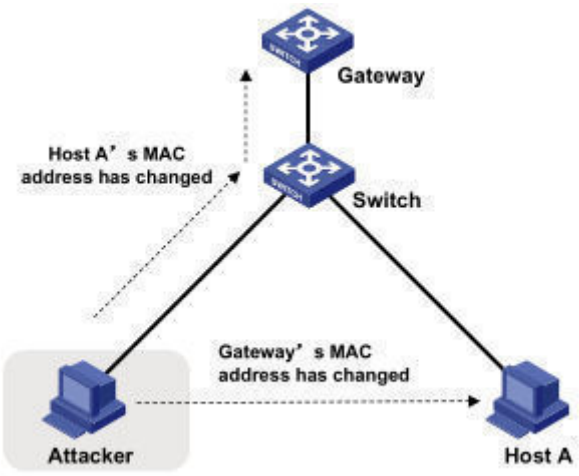

Fig. 3. MitM attack with ARP poison

TABLE I. ARP CACHE BEFORE THE MITM ATTACK

	Attacker	Gateway	Victim Host
Gateway	Attacker's MAC		Victim's MAC
Victim Host	Attacker's MAC	Gateway's MAC	

TABLE II. ARP CACHE AFTER THE MITM ATTACK

	Attacker	Gateway	Victim Host
Gateway	Attacker's MAC		Attacker's MAC
Victim Host	Attacker's MAC	Attacker's MAC	

The ARP cache after the MitM attack defines further communication between the devices in the network, meaning that any communication between the gateway and the victim host is intercepted and forwarded through the attacker's device.

Man in the middle attacks can lead to session hijacking which can grant the attacker access to all the services that victim has been logged in including bank accounts, email, social networks, forums etc. MitM attack additionally allows the attacker while intercepting and capturing the data for further examination, to modify the data in real time before it is forwarded to the victim/gateway. At first, due to the security of SSL/TLS protocol https services were immune to this type of attacks, but soon it was proven that SSL/TLS is vulnerable, too [16][17][18].

SSL/TLS protection is out of the scope of this paper and detailed information on the attack and prevention of SSLStrip can be found in [19].

IV. REAL WORLD ARP ATTACK

A. *Overview of the Network Devices*

With the use of Fing, a network scanner tool running on an Android device, we get an overview of all the devices connected on the network. At the moment the following devices are connected:

TABLE III. DEVICES CONNECTED TO THE GATEWAY

IP	MAC	Hostname	Description
192.168.100.1	00:1D:7E:AA:78:C6	Linksys	Gateway
192.168.100.2	00:1F:C6:0D:78:F2	PC	Windows PC
192.168.100.3	F0:25:B7:1B:A4:65	Samsung	Android Device
192.168.100.4	68:5D:43:59:8C:E7	Thinkpad	Notebook

At this state all of the connected devices share the same ARP cache. Before the attack we define the roles on both the attacker and the victims:

Gateway:	Linksys	IP: 192.168.100.1
Victim 1:	Windows PC	IP: 192.168.100.2
Victim 2:	Samsung	IP: 192.168.100.3
Attacker:	Thinkpad	IP: 192.168.100.4

B. *ARP Poisoning of a Windows PC and an Android Device*

The ARP attack can be conducted using either arpspoof or Ettercap on a Linux machine. In our scenario the attacker was a Windows Notebook and the ARP poisoning was achieved using Cain & Able.

C. *Wireshark Capture of the ARP Attack*

With a WireShark capture one can filter the ARP packets and inspect the ARP communication between the devices in the network (Fig. 4).

At Packet 11893 one can see the Attacker ARP poisoning the Windows PC telling that 192.168.100.1 (Gateways IP) is at 68:5D:43:59:8C:E7 which in fact is the Attackers MAC. To complete the MitM attack the attacker tells the Gateway that the Windows PC's IP 192.169.100.2 is at 68:5D:43:59:8C:E7 which can be seen at packet 11894. On packet 12043 the attacker changes the Gateway's MAC with its own in the ARP cache of the Samsung Smartphone and completes the attack with changing the Smartphone's MAC with its own on the Gateway (packet 12044).

D. *Attack Confirmation by Victim 1*

If we check Victim 1's ARP cache before the attack, using the arp –a command in cmd.exe gets us the following result (Fig. 5).

```
11893 18:05:05.773486000   IntelCor_59:8c:e7          AsustekC_0d:78:f2  ARP          42 192.168.100.1 is at 68:5d:43:59:8c:e7
11894 18:05:05.773664000   IntelCor_59:8c:e7          Cisco-Li_aa:78:c6  ARP          42 192.168.100.2 is at 68:5d:43:59:8c:e7
12043 18:05:35.760009000   IntelCor_59:8c:e7          SamsungE_1b:a4:65  ARP          42 192.168.100.1 is at 68:5d:43:59:8c:e7
12044 18:05:35.760153000   IntelCor_59:8c:e7          Cisco-Li_aa:78:c6  ARP          42 192.168.100.3 is at 68:5d:43:59:8c:e7
```

Fig. 4. WireShark capture of the ARP poisoning of Victim 1 and Victim 2

After the attack has been done, one can confirm the attack by reusing the arp –a command, which would result with the recurrent MAC address as both the Attacker and the Gateway (Fig. 6).

```
C:\Documents and Settings\b>arp -a

Interface: 192.168.100.2 --- 0x2
  Internet Address      Physical Address      Type
  192.168.100.1         00-1d-7e-aa-78-c6     dynamic
  192.168.100.3         f0-25-b7-1b-a4-65     dynamic
  192.168.100.4         68-5d-43-59-8c-e7     dynamic
```

Fig. 5. Windows PC (Victim 1) ARP cache before the attack

```
C:\Documents and Settings\b>arp -a

Interface: 192.168.100.2 --- 0x2
  Internet Address      Physical Address      Type
  192.168.100.1         68-5d-43-59-8c-e7     dynamic
  192.168.100.3         f0-25-b7-1b-a4-65     dynamic
  192.168.100.4         68-5d-43-59-8c-e7     dynamic
```

Fig. 6. Windows PC (Victim 1) ARP cache after the attack

After a successful ARP attack the ARP cache on the Victim 1's side has been altered and changed the Gateway's MAC 00:1D:7E:AA:78:C6 with the Attacker's MAC 68:5D:43:59:8C:E7. Victim 1 has 192.168.100.1 as its Gateway and therefore any further communication would continue through the Attacker's computer that has 68:5D:43:59:8C:E7 as its MAC address. At this point the Attacker can perform MitM attack by sniffing, capturing or filtering the data packets the Victim 1 both sends and receives. After the attack, the Attacker sends ARP packets that restore the MAC address for IP 192.168.100.1 to the Routers MAC.

E. Attack Confirmation by Victim 2

To check the ARP cache on an Android Smartphone we use Net Status which can be downloaded from the Play Store. Using the Get Arp Cache command in Net Status prior the attack gets the same ARP cache as all other devices (Fig. 7).

IP address	HW type	Flags	HW address	Mask	Device
192.168.100.2	0x1	0x2	00:1f:c6:0d:78:f2	*	wlan0
192.168.100.1	0x1	0x2	00:1d:7e:aa:78:c6	*	wlan0
192.168.100.4	0x1	0x2	68:5d:43:59:8c:e7	*	wlan0

Fig. 7. Android device (Victim 2) ARP cache before the attack

After the attack the ARP cache has been altered and getting the ARP cache results with the Gateways's MAC address being changed to the Attacker's MAC (Fig. 8). This way one can confirm the ARP attack, as well as who the attacker is.

IP address	HW type	Flags	HW address	Mask	Device
192.168.100.2	0x1	0x2	00:1f:c6:0d:78:f2	*	wlan0
192.168.100.1	0x1	0x2	68:5d:43:59:8c:e7	*	wlan0
192.168.100.4	0x1	0x2	68:5d:43:59:8c:e7	*	wlan0

Fig. 8. Android device (Victim 2) ARP cache after the attack

V. Sniffing Packets with WireShark

Once the ARP attack is complete the data sent and received by the users can be intercepted. This action can be done using WireShark, a free and open source packet analyzer. Wireshark is cross platform and can run on Windows, Linux, Unix and OSX. It uses pcap to capture the packets.

WireShark captures all the data transmitted to and received from the host. The MitM ARP attack makes all the communication between the Victim 1 (PC) and Victim 2 (Smartphone) and the Gateway visible in WireShark.

Capturing data can be achieved with two methods [20]:

- "Catch it as you can" – a method where all the packets are being captured and written to the storage subsequently, in batch mode. The downside of this method is the large amount of storage needed to store the data.
- "Stop, look and listen" – a method where each packet is analyzed and written to the storage only if it meets the defined criteria. The downside of this method is that is CPU intensive.

A. Capturing a Login Form

For demonstration purposes we are going to capture a login form. The login form used is the first google result on "login form" [21]. The Victim 1 will login using the login "arp-pc" and password "test" and the Victim 2 will login using the login "arp-phone" and password "test".

Between logins, WireShark was set to capture the data for 6 minutes and 9 seconds and that resulted in 14758 packets. Going through all the packets one by one can be time consuming. That leads us to filtering the packets using specific filers. User authentication is typically done using the POST

Filter: http.request.method == "POST" Expression... Clear Apply Save

No.	Time	Source	Destination	Protocol	Length	Info
1565	18:02:04.360333000	192.168.100.2	54.231.2.233	HTTP	590	POST /sn
1566	18:02:04.362581000	192.168.100.2	54.231.2.233	HTTP	590	[TCP Ret
2100	18:02:26.493112000	192.168.100.3	54.231.8.185	HTTP	761	POST /sn
2101	18:02:26.496166000	192.168.100.3	54.231.8.185	HTTP	761	[TCP Ret
2338	18:02:52.872205000	192.168.100.2	5.45.62.65	HTTP	919	POST /F/
2339	18:02:52.873455000	192.168.100.2	5.45.62.65	HTTP	919	[TCP Ret
12021	18:05:31.137572000	192.168.100.2	5.45.62.65	HTTP	919	POST /F/
12022	18:05:31.138729000	192.168.100.2	5.45.62.65	HTTP	919	[TCP Ret
12457	18:06:31.263589000	192.168.100.3	192.168.100.4	HTTP/XML	783	POST /up
12527	18:06:31.575692000	192.168.100.3	192.168.100.4	HTTP/XML	971	POST /up
12589	18:06:32.072276000	192.168.100.3	192.168.100.4	HTTP/XML	971	POST /up

Fig. 9. WireShark filter of the http packets using the POST method

method which dramatically reduces the packets displayed in WireShark. The filter used to narrow to only the packets that used the POST method is: http.request.method == "POST" (Fig. 9). To inspect the packet 1565 we will use the command "Follow TCP Stream". We need to focus on the POST paragraph which provides us with the following information:

```
POST /snippets/8-login-form/index.html HTTP/1.1
Host: s3.cssflow.com
User-Agent: Mozilla/5.0 (Windows NT 5.1; rv:34.0) Gecko/20100101 Firefox/34.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://s3.cssflow.com/snippets/8-login-form/index.html
Cookie: _ga=GA1.2.1069828863.1418392244; _gat=1
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 39
```

This info shows us that the POST method is used to send data to: /snippets/8-login-form/index.html.

The Referrer or the side that Victims used to login is: http://s3.cssflow.com/snippets/8-login-form/index.html. At this point we can confirm that we have the right packet. The next step is to expand "HTML Form URL Encoded" in the Packet details (Fig. 10).

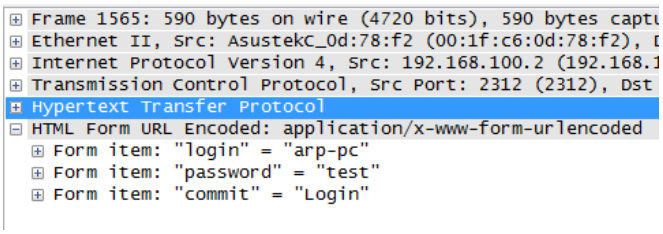

Fig. 10. Packet containing the login and password of Victim 1

Expanding the HTML Form URL encoded provides with the information of the login, password and commit form items. One can see that that login of Victim 1 is "arp-pc" and the password is "test". The Login details of Victim 2 can be inspected on packet 2100 (Fig. 11).

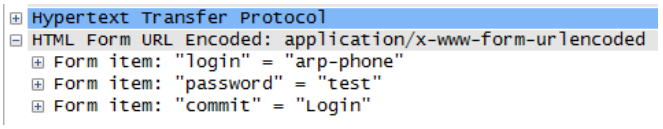

Fig. 11. Packet containing the login and password of Vitim 2

VI. Protecting from ARP Attacks

A. Detecting and Protecting Windows PCs

DecaffeinatID [16] is a Windows applications dedicated to monitoring the ARP changes in the local network. This application notifies the user for new devices that are connected to the network with a popup and stores the new events in a log. The log contains new devices that connected to the network and changes made to the ARP cache: the first and the second record identify that the Android Smartphone (Victim 2) and ThinkPad (Attacker) have connected to the network. The third record identifies that the MAC address for 192.168.100.1 (Gateway) has been changed and that the user was displayed with a warning, i.e.:

```
New IP in cache: 192.168.100.3 with MAC of f0-25-b7-1b-a4-65
New IP in cache: 192.168.100.4 with MAC of 68-5d-43-59-8c-e7
IP 192.168.100.1 changed MAC from 00-1d-7e-aa-78-c6 to 68-5d-43-59-8c-e7
WARNING!!! This IP is a Gateway!!! Something weird is up!!!
IP 192.168.100.1 changed MAC from 68-5d-43-59-8c-e7 to 00-1d-7e-aa-78-c6
WARNING!!! This IP is a Gateway!!! Something weird is up!!!
```

As one can see on the last record, DecaffeinatID does not keep a record of the original MAC of the Gateway and presents the user with a warning although the attack has stopped and the Gateway's MAC has been restored. In case the user misses the notifications the log stored as idslog.txt can be used for further network forensics.

One of the simplest methods to prevent ARP attacks is to set a static entry to the ARP cache with the Gateway's IP and MAC address. Using CMD as administrator the user can input: arp -s 192.168.100.1 00-1d-7e-aa-78-c6. This way the ARP cache will have a static entry for the Gateway which can't be altered by attackers. Static entries expire once the system reboots. To access this problem one can create a simple .bat script with the arp –s command and put the script in the "Startup" folder so the command executes every time the PC boots.

For the less experienced users, ARPFreeze [22] is an automated script that uses the ARP command to detect the devices connected on the same network and asks the user which of the entries he would like to make static.

B. Detecting and Protecting Android Devices

One of the methods in detecting and protecting Android devices is to scan the network using Fing. On Fig. 12 one can see that router's field is changed with a new MAC address and contains (+1) which indicates that another device connected on the network has the same MAC address.

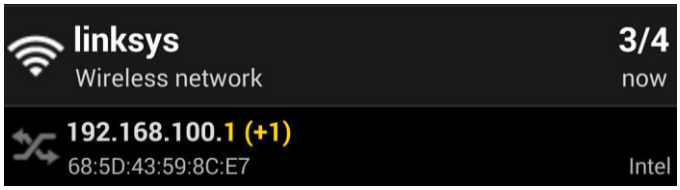

Fig. 12. Fing scan identifying additional MAC entry

Selecting the Gateway field gets that additional information so the user can identify who is that attacked, in this case the computer with the 192.168.100.4 IP and with the First seen and Last change he can easily conclude when that device has first connected on the network and more importantly when the attack occurred. Another indicator is that the previous IP and MAC address of the Gateway will be a grayed out field at the end of the list. We have to note here that the Fing network scanner has to be regularly used in order to get this specific information.

Wifi Protector [23] and WiFi ARP Guard [24] are both Android applications that automatically detect ARP spoofing

and ARP poisoning and therefore can protect the user from DoS and MitM attacks. There are many advantages of using these dedicated applications versus the Net Status and Fing methods, mainly for their automation process that checks the ARP cache of the Android device multiple times each second. Wifi Protector and WiFi ARP Guard have the option to automatically start when the wireless LAN is activated. When an attack is detected these applications can notify the user with a sound, vibration or notification. These notifications contain the MAC address of the attacker that help identify the attacker. Wifi Protector and WiFi ARP Guard store logs of the attacks and can be really useful for further network forensics. The protection that comes with these applications consists of disabling the wireless LAN in a case of attack. If the device is rooted the user can add a static ARP entry for their gateway. The disadvantage of running these types of applications is the effect on the battery life of the device.

C. *Network Based Solutions*

When it comes to small home or business networks or generally networks that are not for public use, disabling the DHCP and setting up static IP addresses for each device, bound with their unique MAC, can prove to be the best solution. Lone and Ataullah [25] have conducted a survey on various solutions of ARP attacks. The survey consists of 8 solutions including systems that use cryptography to secure the ARP protocol such as S-ARP and TARP, middleware devices installed on each host on the network to block unsolicited ARP replies and a hardware solution that consists of a server with an invite-accept and request reply protocol. The paper provides a comparative analysis of the solutions.

VII. Conclusion and Future Work

This paper shows that despite having modern applications communicating on modern hardware and software, users on local networks are vulnerable on DoS MitM attacks due to the vulnerability of the ARP protocol. We created a real world scenario of an ARP attack that serves as a platform for a packet analyzer to monitor all the data from and to the victims. Using WireShark filters we filtered the communication down to http POST methods which displayed us with the victim's login and password on a test page. The most important part, however, is protecting from ARP attacks, that detailed the detection and protection options for Windows and Android users.

Future work can complement this paper with methods and applications to detect and prevent ARP attacks. It will discuss the techniques of detecting network sniffers using ARP packets, crafting custom ping with hping3, using nmap script sniffer-detect, as well as comparative analysis and success rate of Windows applications Cain & Abel and Promqry; and Linux applications: arpwatch, Nast, ptool and sniffdet, in identifying a network card working in promiscuous mode [26].

References

[1] http://tools.ietf.org/html/rfc826

[2] M. Tripunitara and P. Dutta, "A middleware approach to asynchronous and backward compatible detection and prevention of ARP cache poisoning", 15th Annual Computer Security Applications Conf., 1999, pp. 303-309.

[3] B. Zdrnja, "Malicious JavaScript Insertion through ARP Poisoning Attacks", IEEE Security & Privacy, May/June 2009, pp. 72-74.

[4] M. Kolodziejczyk and M.R. Ogiela, "Security Mechanisms in Network Protocols", 2010 Intl. Conf. on Intelligent Systems, Modeling and Simulation, 2010, pp. 427-430.

[5] C.L. Abad and R.I. Bonilla, "An Analysis on the Schemes for Detecting and Preventing ARP Cache Poisoning Attacks", 27th Intl. Conf. on Distributed Computing Systems Workshops (ICDCSW'07), 2007, p. 60.

[6] W. Lootah, W. Enck and P. McDaniel, "TARP: Ticket-based address resolution protocol", 21st Annual Computer Security Applications Conf. (ACSAC 2005), 2005, 9 pp. – 116.

[7] D. Bruschi, A. Ornaghi and E. Rosti, "S-ARP: a Secure Address Resolution Protocol", 19th Annual Computer Security Applications Conf. (ACSAC 2003), 2003, pp. 66-74.

[8] H. Salim, Z. Li, H. Tu and Z. Guo, "Preventing ARP Spoofing Attacks through Gratuitous Decision Packet", 11th Intl. Symp. on Distributed Computing and Applications to Business, Engineering & Science, pp. 295-300.

[9] M.A. Qadeer, N. Zahid, A. Iqbal and M.R. Siddiqui, "Network Traffic Analysis and Intrusion Detection using Packet Sniffer", Second Intl. Conf. on Communication Software and Networks, pp. 313-317.

[10] D. Pansa and T. Chomsiri, "Architecture and Protocols for Secure LAN by Using a Software-level Certificate and Cancellation of ARP Protocol", Third Intl. Conf. on Convergence and Hybrid Information Technology, 2008, pp. 21-26.

[11] V. Radhakishan and S. Selvakumar, "Prevention of Man-in-the-Middle Attacks using ID Based Signatures", Second Intl. Conf. on Networking and Distributed Computing, pp. 165-169.

[12] J. Belenguer and C.T. Calafate, "A low-cost embedded IDS to monitor and prevent Man-in-the-Middle attacks on wired LAN environments", Intl. Conf. on Emerging Security Information Systems and Technologies, 2007, pp. 122-127.

[13] F. Callegati, W. Cerroni and M. Ramilli, "Man-in-the-Middle Attack to the HTTPS Protocol", IEEE Security & Privacy, May/June 2009, pp. 78-81.

[14] E.A. Anaya, M.N. Miyatake and H.M.P. Meana, "Network Forensics with Neurofuzzy Techniques", 52nd IEEE Intl. Midwest Symp. on Circuits and Systems, MWSCAS '09, 2009, pp. 848-852.

[15] http://web2.uwindsor.ca/courses/cs/aggarwal/cs60564/Assignments2Project1/RoyMoazzamiSingh_Assignment2.doc

[16] M. Marlinspike "New Tricks for Defeating SSL in Practice", https://www.blackhat.com/presentations/bh-dc-09/Marlinspike/BlackHat-DC-09-Marlinspike-Defeating-SSL.pdf

[17] D. Peifer, "SSL Spoofing", https://www.owasp.org/images/7/7a/SSL_Spoofing.pdf

[18] J. Franklin, B. Mijanovich and G. Pothier, "SSLStrip Man-In-The-Middle", http://fit.hcmup.edu.vn/~hienlth/COMP1049/Labs/04%20-%20SSL%20Spoofing/sslstrip2.pdf

[19] Y. Zhao, Y. Lei, T. Yang and Y. Cui, "A new strategy to defense against SSLStrip for Android", 15th IEEE Intl. Conf. on Communication Technology (ICCT), 2013, pp. 70-74.

[20] R. Sira, "Network Forensics Analysis Tools: An Overview of an Emerging Technology", http://www.giac.org/paper/gsec/2478/network-forensics-analysis-tools-overview-emerging-technology/104303

[21] http://www.cssflow.com/snippets/login-form/demo

[22] http://www.irongeek.com/i.php?page=security/arpfreeze-static-arp-poisoning

[23] https://play.google.com/store/apps/details?id=com.gurkedev.wifiprotector

[24] https://play.google.com/store/apps/details?id=com.mdl.arpguard

[25] I.A. lone and Md. Ataullah, "A survey on various solutions of ARP attacks", International Journal of Advanced Research in Computer Science and Software Engineering, Vol. 3, No. 2, 2013, pp. 299-303.

[26] D. Sanai, "Detection of Promiscuous Nodes Using ARP Packets", http://www.securityfriday.com/promiscuous_detection_01.pdf