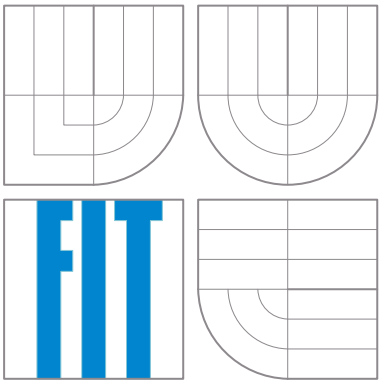

Data Communications, Computer Networks and Protocols

L2 MitM

April 23, 2017

Authors: Ján Profant, xprofa00@stud.fit.vutbr.cz

Faculty of Information Technology
Brno University of Technology

Contents

Assignment **2**

Address Resolution Protocol **3**
Network scanning 4
Network spoofing 4
Network interception 4

Neighbor Discovery Protocol **5**

Demonstration **6**
pds-scanner 6
pds-chooser 7
pds-spoof and pds-massspoof 7
pds-intercept 8

Bibliography **9**

Assignment

Study topics about transparent man in the middle (denoted as MitM for the rest of paper) attacks. Implement application in C++ which realizes MitM over IPv4 and IPv6 traffic using spurious IP-to-MAC log in following steps:

1. **Scanning and Reconnaissance** - automatically find hosts in attacker's network.
2. **Spoofing** - ARP and NDP cache poisoning.
3. **Interception** - resend network traffic, so victims have no idea they are under attack.

Address Resolution Protocol

We can divide into two main parts - Address Resolution Protocol (denoted as ARP for the rest of paper), which is used for resolution of Internet layer addresses into link layer addresses, a critical function in computer networks and Neighbor Discovery Protocol (denoted as NDP for the rest of paper) which is used in IPv6. In this chapter, we will look closer at ARP. ARP was defined by **RFC 826** in 1982 [2]. In figure 1 we can see structure of ARP protocol.

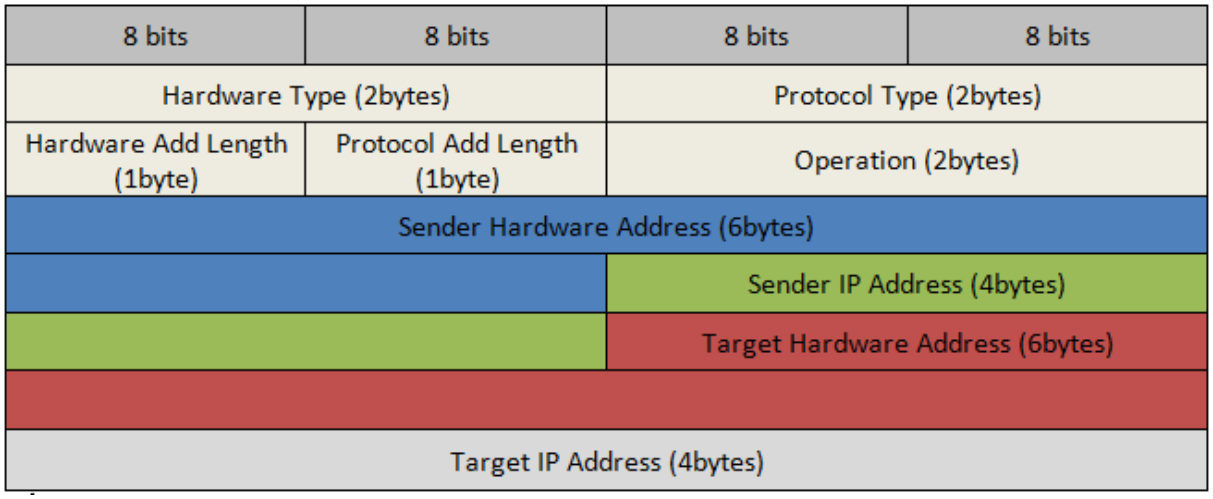

Figure 1: ARP header defined by RFC 826. Taken from [8].

Within Ethernet ARP there are four types of messages.[7]

- ARP request – this is a request for the destination hardware address, it is typically sent to all hosts.
- ARP reply – in response to a request, this tells the host the hardware address of the destination host.
- RARP request – known as Reverse ARP request, this requests the IP address of a known MAC address.
- RARP reply – the response for a RARP request, this gives the IP address from a requested hardware address.

The ARP poisoning attack targets to modify the IP/MAC address mapping in the ARP cache of a remote machine maliciously. This ARP poisoning is usually used to mount other types of attacks such as DoS or MitM attacks. [4]

Network scanning

ARP scanner sends ARP packets to hosts on the local network and catches any responses that are received. The network interface must be specified. The process of scanning is done by two threads, where one of them is generating packets for each possible IPv4 address (defined by interface IPv4 address and mask) and second one is waiting for possible responses.

Network spoofing

ARP spoofing or ARP cache poisoning is a technique by which an attacker sends (spoofed) ARP messages onto a local area network. Generally, the aim is to associate the attacker's MAC address with the IP address of another host. The victim's machine that receives the spoofed ARP replies cannot distinguish them from legitimate ones. Moreover, the ARP tables usually use the result of the last ARP reply only.

Also, in this case the bonus application which uses XML file as input and runs spoofing on independent threads for victim's pairs is implemented.

Network interception

When speaking about network interception in ARP, it is basically based on fact, that the attacker takes the role of man in the middle; any traffic directed to the legitimate resource is sent through the attacking system. The attacker reads the packet looking for sensitive or any other data; it may modify that data, and then passes it to the designated destination. [1]

Neighbor Discovery Protocol

The Neighbor Discovery Protocol is a protocol in the Internet protocol suite used with IPv6. It operates in the Link Layer of the Internet model, and is responsible for address auto configuration of nodes, discovery of other nodes on the link, determining the addresses of other nodes, duplicate address detection, finding available routers and Domain Name System (DNS) servers, address prefix discovery, and maintaining reachability information of other active neighbor nodes. [5] Due to lack of basic human needs, e.g. sleep, time the NDP was not successfully implemented. There were some attempts to implement IPv6 scanning, which consumed tons of time, but with no luck. Despite this fact, application API is prepared for IPv6 implementation and some parts are working even for IPv6 (for example *pds-intercept*, XML parsing).

Demonstration

For demonstration purposes, we used network scheme as shown in figure 2. All application can be easily compiled using *make*, or using *make debug*, if user wants to see output for debugging purposes.

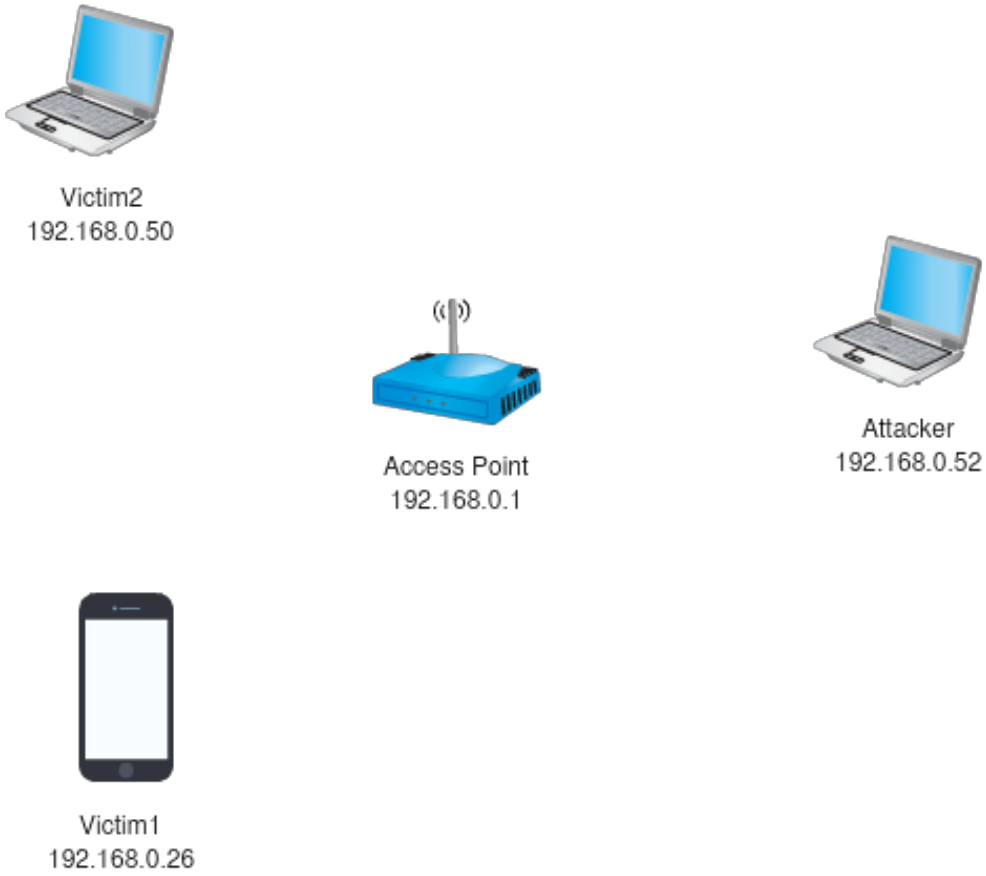

Figure 2: Network scheme which was used for demonstration of applications.

pds-scanner

For ARP network scanning, you need to specify interface and output XML file. In following XML file, we can see result of ARP scan.

```
<?xml version="1.0" encoding="UTF-8"?>
<devices>
    <host mac="88f7.c7cb.0628">
        <ipv4>192.168.0.1</ipv4>
    </host>
    <host mac="849f.b586.32ad">
        <ipv4>192.168.0.26</ipv4>
    </host>
    <host mac="e094.67a4.a100">
        <ipv4>192.168.0.50</ipv4>
    </host>
</devices>
```

According to our experiments, we found out that some of the hosts sometimes do not reply (we can see similar behavior in *arp-scan* [6]), so according to this fact, we are using iterative approach and we repeat sending 10 times.

No.	Time	Source	Destination	Protocol	Length	Info
39	4.845939331	IntelCor_67:22:a8	Broadcast	ARP	64	Who has 192.168.0.1? Tell 192.168.0.5
40	4.848335925	Technico_cb:06:28	IntelCor_67:22:a8	ARP	64	192.168.0.1 is at 88:f7:c7:cb:06:28
41	4.869767540	IntelCor_67:22:a8	Broadcast	ARP	64	Who has 192.168.0.2? Tell 192.168.0.5
42	4.871921862	IntelCor_67:22:a8	Broadcast	ARP	64	Who has 192.168.0.3? Tell 192.168.0.5
43	4.874122714	IntelCor_67:22:a8	Broadcast	ARP	64	Who has 192.168.0.4? Tell 192.168.0.5
44	4.876331972	IntelCor_67:22:a8	Broadcast	ARP	64	Who has 192.168.0.5? Tell 192.168.0.5
45	4.878500681	IntelCor_67:22:a8	Broadcast	ARP	64	Who has 192.168.0.6? Tell 192.168.0.5

Figure 3: Example of ARP traffic during scanning in wireshark.

pds-chooser

Bonus application is also implemented and allows user's input to specify victim pairs. The result is shown in XML file.

```
<?xml version="1.0" encoding="UTF-8"?>
<devices>
        <host mac="88f7.c7cb.0628">
                <ipv4>192.168.0.1</ipv4>
        </host>
        <host mac="849f.b586.32ad" group="victim-pair-1">
                <ipv4>192.168.0.26</ipv4>
        </host>
        <host mac="e094.67a4.a100" group="victim-pair-1">
                <ipv4>192.168.0.50</ipv4>
        <host>
</devices>
```

pds-spoof and pds-massspoof

These tools are used for ARP cache poisoning. User has to specify interface, time interval for sending sending ARP packets and tuple of victims. Even here, bonus *pds-massspoof* was

implemented and actually, classic spoofing is using interface of *MassSpoofer* class. It is very easy to check functionality of this application - at the victim's device we can use command *arp -a*, which prints status of ARP table, we can see example in figure 4. ARP spoofing in wireshark is shown in figure 5. When SIGINT signal is received, the ARP tables are returned to previous state. There is also defined time interval, which prevents overloading.

```
u0_a89@FRD-L02:/data/data/com.arachnoid.sshelper/home $ arp -a
? (192.168.0.50) at cc:3d:82:67:22:a8 [ether]  on wlan0
? (192.168.0.1) at 88:f7:c7:cb:06:28 [ether]  on wlan0
? (192.168.0.52) at cc:3d:82:67:22:a8 [ether]  on wlan0
```

Figure 4: Example of ARP table on android device.

No.	Time	Source	Destination	Protocol	Length	Info
3	0.494065395	IntelCor_67:22:a8	IntelCor_a4:a1:00	ARP	42	Who has 192.168.0.50? Tell 192.168.0.26
4	0.494147614	IntelCor_67:22:a8	HuaweiTe_86:32:ad	ARP	42	Who has 192.168.0.26? Tell 192.168.0.50
5	0.542665178	HuaweiTe_86:32:ad	IntelCor_67:22:a8	ARP	42	192.168.0.26 is at 84:9f:b5:86:32:ad
24	5.494373280	IntelCor_67:22:a8	IntelCor_a4:a1:00	ARP	42	Who has 192.168.0.50? Tell 192.168.0.26
25	5.494409442	IntelCor_67:22:a8	HuaweiTe_86:32:ad	ARP	42	Who has 192.168.0.26? Tell 192.168.0.50
26	5.559914266	HuaweiTe_86:32:ad	IntelCor_67:22:a8	ARP	42	192.168.0.26 is at 84:9f:b5:86:32:ad
34	10.494697873	IntelCor_67:22:a8	IntelCor_a4:a1:00	ARP	42	Who has 192.168.0.50? Tell 192.168.0.26
35	10.494739675	IntelCor_67:22:a8	HuaweiTe_86:32:ad	ARP	42	Who has 192.168.0.26? Tell 192.168.0.50

Figure 5: ARP spoofing sent from attacker to both victims.

pds-intercept

When using network interception, user has to specify interface and input XML file with group attributes. The application just receives communication, check if it belongs to the victim specified in XML and resends it with modified MAC addresses. We used same setup as shown before, we tried to send ICMP [3] packet (ping) from one device (victim1) to second device (victim2). When we have just *pds-massspoof* running, we are not able to ping from one device to another - attacker is receiving ICMP messages. When we run also *pds-intercept*, we can clearly see, that ping request comes to attacker's interface and he resends it to destination. Then, the second victim receives ping request and replies, the payload is again flowing through attacker, and he successfully sends ping reply to ping initiator, as shown in figure 6.

No.	Time	Source	Destination	Protocol	Length	Info
55	5.107833015	192.168.0.26	192.168.0.50	ICMP	98	Echo (ping) request id=0x0085, seq=1
56	5.107960768	192.168.0.26	192.168.0.50	ICMP	98	Echo (ping) request id=0x0085, seq=1
60	5.110586158	192.168.0.50	192.168.0.26	ICMP	98	Echo (ping) reply id=0x0085, seq=1
61	5.110760554	192.168.0.50	192.168.0.26	ICMP	98	Echo (ping) reply id=0x0085, seq=1
73	6.121042805	192.168.0.26	192.168.0.50	ICMP	98	Echo (ping) request id=0x0085, seq=2
74	6.121141183	192.168.0.26	192.168.0.50	ICMP	98	Echo (ping) request id=0x0085, seq=2
75	6.123641254	192.168.0.50	192.168.0.26	ICMP	98	Echo (ping) reply id=0x0085, seq=2

Figure 6: Example of MitM - resending ICMP packets in wireshark.

Bibliography

[1] Ghazi Al Sukkar, Ramzi Saifan, Sufian Khwaldeh, Mahmoud Maqableh, and Iyad Jafar. Address resolution protocol (arp): Spoofing attack and proposed defense. *Communications and Network*, 8(03):118, 2016.

[2] David C. Plummer. An Ethernet Address Resolution Protocol, 1982. https://tools.ietf.org/html/rfc826.

[3] J. Postel. INTERNET CONTROL MESSAGE PROTOCOL, 1981. https://tools.ietf.org/html/rfc792.

[4] Seung Yeob Nam, Dongwon Kim, Jeongeun Kim, et al. Enhanced arp: preventing arp poisoning-based man-in-the-middle attacks. *IEEE communications letters*, 14(2):187–189, 2010.

[5] R. Braden. Requirements for Internet Hosts - Communication Layers, 1989. https://tools.ietf.org/html/rfc1122.

[6] Roy Hills. arp-scan. https://linux.die.net/man/1/arp-scan.

[7] Robert Wagner. Address resolution protocol spoofing and man-in-the-middle attacks. *The SANS Institute*, 2001.

[8] Web. MITM 101: ARPSpoofing. https://toastersecurity.blogspot.cz/2015/12/man-in-middle-understanding-and.html.