International Cybersecurity and Digital Forensic Academy

PROGRAMME: BAZE UNIVERSITY INTERNSHIP

ASSIGNMENT

PRESENTED BY

PHILIP AKISANNI AKWUCHI
IDEAS/24/29921

COURSE CODE: INT302
COURSE TITLE: Kali Linux Tools and System Security

COURSE FACILITATOR: AHMED BUKAR

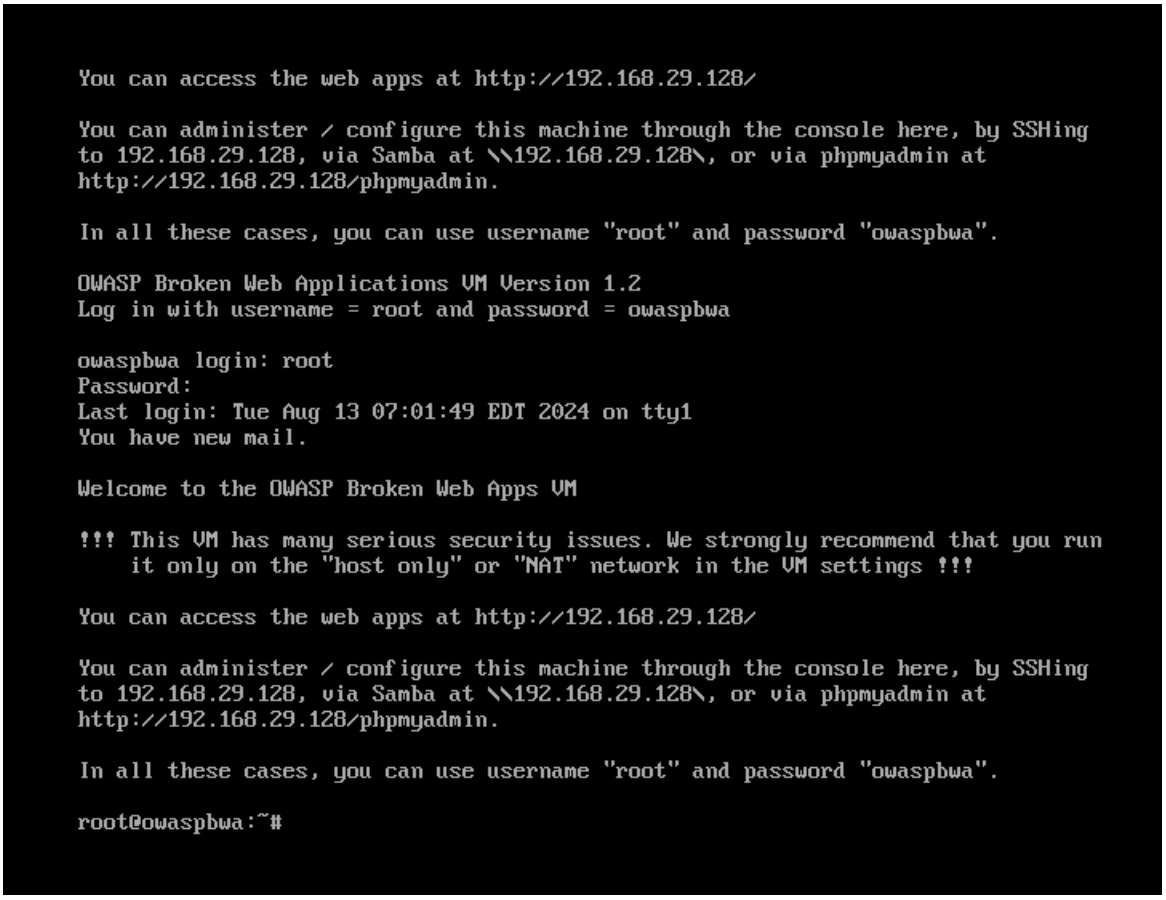
NOVEMBER, 2024

Lab 4: Basic Port Scanning

Step 1: Gather the IP Address of Your OWASP VM

Record the IP Address:

• OWASP VM IP Address**:** 192.168.29.128



Step 2: Basic Port Scanning with nmap

Exercise 1:

Perform a basic port scan on your OWASP VM IP address and record your findings:

• Open Ports:

PORT STATE SERVICE

22/tcp open ssh

25/tcp closed smtp

80/tcp open http

110/tcp open pop3

139/tcp open netbios-ssn

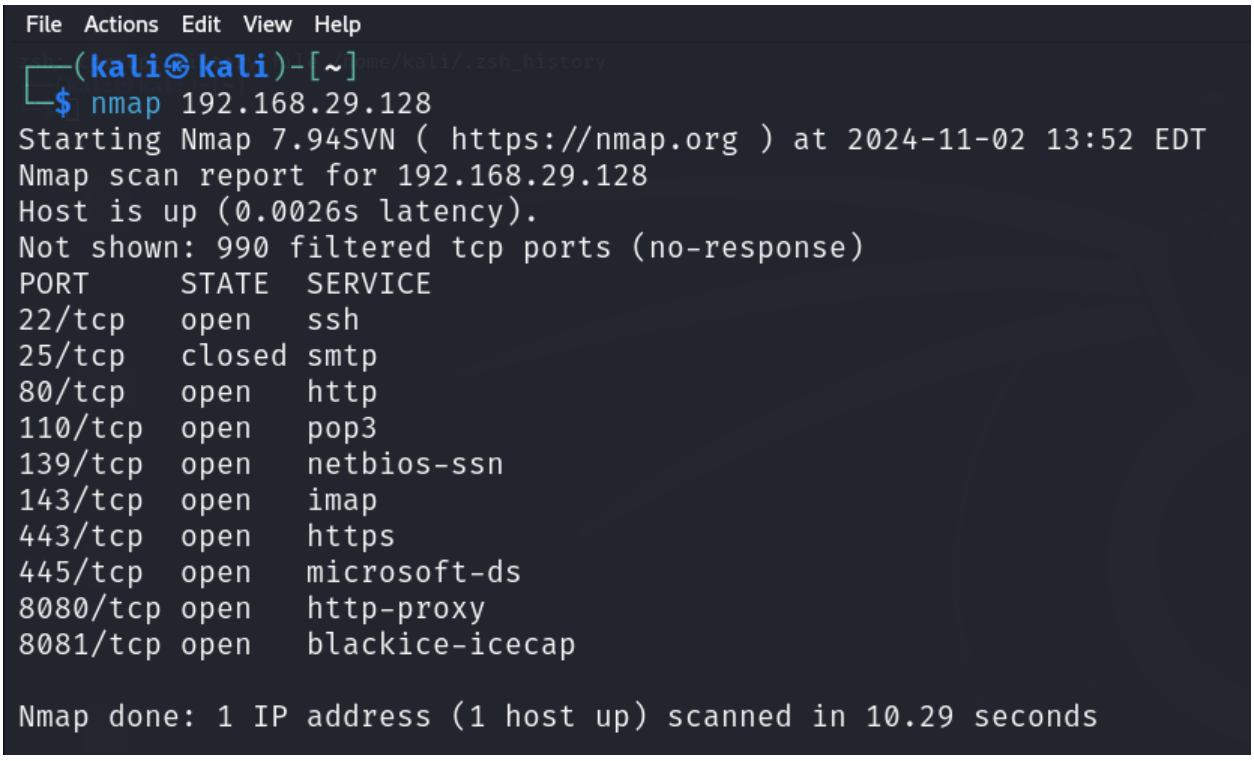
143/tcp open imap

443/tcp open https

445/tcp open microsoft-ds

8080/tcp open http-proxy

8081/tcp open blackice-icecap



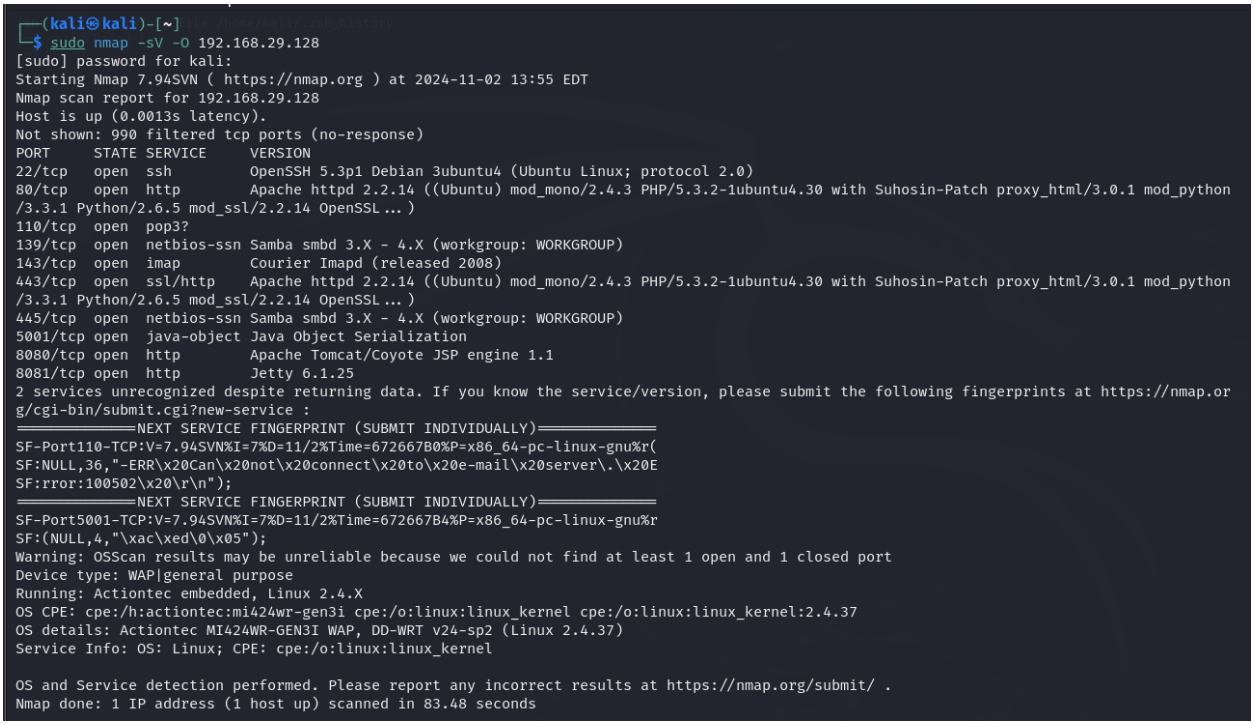
Step 3: Aggressive Scanning with nmap

Exercise 2:

Perform an aggressive scan on your OWASP VM IP address and record your findings:

• Service Versions:

• Operating System:



Step 4: Vulnerability Scanning with nmap

Exercise 3:

Conduct a vulnerability scan on your OWASP VM IP address and record your findings:

• Vulnerabilities:

Starting Nmap 7.94SVN (https://nmap.org) at 2024-11-02 14:02 EDT

Nmap scan report for 192.168.29.128

Host is up (0.0054s latency).

Not shown: 991 filtered tcp ports (no-response)

PORT STATE SERVICE

22/tcp open ssh

80/tcp open http

|_http-stored-xss: Couldn't find any stored XSS vulnerabilities.

|_http-dombased-xss: Couldn't find any DOM based XSS.

| http-sql-injection:

| Possible sqli for queries:

|_ http://192.168.29.128:80/railsgoat/assets/jquery.js?body=1%27%20OR%20sqlspider

| http-cross-domain-policy:

| VULNERABLE:

| Cross-domain and Client Access policies.

| State: VULNERABLE

| A cross-domain policy file specifies the permissions that a web client such as Java, Adobe Flash, Adobe Reader,

| etc. use to access data across different domains. A client acces policy file is similar to cross-domain policy

| but is used for M$ Silverlight applications. Overly permissive configurations enables Cross-site Request

| Forgery attacks, and may allow third parties to access sensitive data meant for the user.

| Check results:

| /crossdomain.xml:

| <?xml version="1.0"?>

| <!DOCTYPE cross-domain-policy SYSTEM "http://www.macromedia.com/xml/dtds/cross-domain-policy.dtd">

| <cross-domain-policy>

| <allow-access-from domain="*" />

| </cross-domain-policy>

| Extra information:

| Trusted domains:*

|

| References:

| http://gursevkalra.blogspot.com/2013/08/bypassing-same-origin-policy-with-flash.html

| https://www.owasp.org/index.php/Test_RIA_cross_domain_policy_%28OTG-CONFIG-008%29

| https://www.adobe.com/devnet-docs/acrobatetk/tools/AppSec/CrossDomain_PolicyFile_Specification.pdf

| https://www.adobe.com/devnet/articles/crossdomain_policy_file_spec.html

| http://sethsec.blogspot.com/2014/03/exploiting-misconfigured-crossdomainxml.html

|_ http://acunetix.com/vulnerabilities/web/insecure-clientaccesspolicy-xml-file

| http-cookie-flags:

| /mono/:

| ASP.NET_SessionId:

|_ httponly flag not set

| http-enum:

| /wordpress/: Blog

| /test/: Test page

| /mono/: Mono

| /crossdomain.xml: Adobe Flash crossdomain policy

|_ /phpmyadmin/: phpMyAdmin

| http-internal-ip-disclosure:

|_ Internal IP Leaked: 127.0.1.1

|_http-trace: TRACE is enabled

| http-vuln-cve2011-3192:

| VULNERABLE:

| Apache byterange filter DoS

| State: VULNERABLE

| IDs: CVE:CVE-2011-3192 BID:49303
| The Apache web server is vulnerable to a denial of service attack when numerous
| overlapping byte ranges are requested.
| Disclosure date: 2011-08-19
| References:
| https://www.tenable.com/plugins/nessus/55976
| https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2011-3192
| https://www.securityfocus.com/bid/49303
|_ https://seclists.org/fulldisclosure/2011/Aug/175
| http-fileupload-exploiter:
|
| Failed to upload and execute a payload.
|
| Failed to upload and execute a payload.
|
| Failed to upload and execute a payload.
|
| Failed to upload and execute a payload.
|
| Failed to upload and execute a payload.
|
| Failed to upload and execute a payload.
|
| Failed to upload and execute a payload.
|

| Failed to upload and execute a payload.
|
| Failed to upload and execute a payload.
|
| Failed to upload and execute a payload.
|
| Failed to upload and execute a payload.
|
| Failed to upload and execute a payload.
|
| Failed to upload and execute a payload.
|
| Failed to upload and execute a payload.
|
| Failed to upload and execute a payload.
|
| Failed to upload and execute a payload.
|
| Failed to upload and execute a payload.
|
| Failed to upload and execute a payload.
|
| Failed to upload and execute a payload.
|
| Failed to upload and execute a payload.
|
|_ Failed to upload and execute a payload.
| http-csrf:
| Spidering limited to: maxdepth=3; maxpagecount=20;
withinhost=192.168.29.128

| Found the following possible CSRF vulnerabilities:
|
| Path: http://192.168.29.128:80/WackoPicko/
| Form id: query2
| Form action: /WackoPicko/pictures/search.php
|
| Path: http://192.168.29.128:80/WackoPicko/
| Form id:
| Form action: /WackoPicko/pic' + 'check' + '.php
|
| Path: http://192.168.29.128:80/railsgoat/
| Form id:
| Form action: /railsgoat/signup
|
| Path: http://192.168.29.128:80/railsgoat/
| Form id:
| Form action: /railsgoat/login
|
| Path: http://192.168.29.128:80/railsgoat/
| Form id: show_creds_btn
| Form action: #myModalLabel1
|
| Path: http://192.168.29.128:80/wordpress/
| Form id: searchform
| Form action: http://192.168.29.128/wordpress/

|

| Path: http://192.168.29.128:80/phpBB2/

| Form id:

| Form action: login.php?sid=77e5279dcfc5c2c54938ccf9b0395889

|

| Path: http://192.168.29.128:80/AppSensorDemo/login.jsp

| Form id:

|_ Form action: Login

110/tcp open pop3

139/tcp open netbios-ssn

143/tcp open imap

443/tcp open https

| http-cross-domain-policy:

| VULNERABLE:

| Cross-domain and Client Access policies.

| State: VULNERABLE

| A cross-domain policy file specifies the permissions that a web client such as Java, Adobe Flash, Adobe Reader,

| etc. use to access data across different domains. A client acces policy file is similar to cross-domain policy

| but is used for M$ Silverlight applications. Overly permissive configurations enables Cross-site Request

| Forgery attacks, and may allow third parties to access sensitive data meant for the user.

| Check results:

| /crossdomain.xml:

| <?xml version="1.0"?>

| <!DOCTYPE cross-domain-policy SYSTEM "http://www.macromedia.com/xml/dtds/cross-domain-policy.dtd">
| <cross-domain-policy>
| <allow-access-from domain="*" />
| </cross-domain-policy>
| Extra information:
| Trusted domains:*
|
| References:
| http://gursevkalra.blogspot.com/2013/08/bypassing-same-origin-policy-with-flash.html
| https://www.owasp.org/index.php/Test_RIA_cross_domain_policy_%28OTG-CONFIG-008%29
| https://www.adobe.com/devnet-docs/acrobatetk/tools/AppSec/CrossDomain_PolicyFile_Specification.pdf
| https://www.adobe.com/devnet/articles/crossdomain_policy_file_spec.html
| http://sethsec.blogspot.com/2014/03/exploiting-misconfigured-crossdomainxml.html
|_ http://acunetix.com/vulnerabilities/web/insecure-clientaccesspolicy-xml-file
| ssl-ccs-injection:
| VULNERABLE:
| SSL/TLS MITM vulnerability (CCS Injection)
| State: VULNERABLE
| Risk factor: High
| OpenSSL before 0.9.8za, 1.0.0 before 1.0.0m, and 1.0.1 before 1.0.1h
| does not properly restrict processing of ChangeCipherSpec messages,
| which allows man-in-the-middle attackers to trigger use of a zero

| length master key in certain OpenSSL-to-OpenSSL communications, and
| consequently hijack sessions or obtain sensitive information, via
| a crafted TLS handshake, aka the "CCS Injection" vulnerability.
|
| References:
| https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-0224
| http://www.cvedetails.com/cve/2014-0224
|_   http://www.openssl.org/news/secadv_20140605.txt
| http-vuln-cve2011-3192:
| VULNERABLE:
| Apache byterange filter DoS
| State: VULNERABLE
| IDs: CVE:CVE-2011-3192 BID:49303
| The Apache web server is vulnerable to a denial of service attack when numerous
| overlapping byte ranges are requested.
| Disclosure date: 2011-08-19
| References:
| https://www.tenable.com/plugins/nessus/55976
| https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2011-3192
| https://www.securityfocus.com/bid/49303
|_   https://seclists.org/fulldisclosure/2011/Aug/175
| http-cookie-flags:
| /mono/:
| ASP.NET_SessionId:
| secure flag not set and HTTPS in use

|_ httponly flag not set
| http-sql-injection:
| Possible sqli for queries:
|
https://192.168.29.128:443/redmine/?C=N%3BO%3DD%27%20OR%20sqlspider
|
https://192.168.29.128:443/redmine/?C=D%3BO%3DA%27%20OR%20sqlspider
|
https://192.168.29.128:443/redmine/?C=M%3BO%3DA%27%20OR%20sqlspider
|
https://192.168.29.128:443/redmine/?C=S%3BO%3DA%27%20OR%20sqlspider
|
https://192.168.29.128:443/railsgoat/?C=N%3BO%3DD%27%20OR%20sqlspider
|
https://192.168.29.128:443/railsgoat/?C=M%3BO%3DA%27%20OR%20sqlspider
|
https://192.168.29.128:443/railsgoat/?C=D%3BO%3DA%27%20OR%20sqlspider
|_
https://192.168.29.128:443/railsgoat/?C=S%3BO%3DA%27%20OR%20sqlspider
| ssl-poodle:
| VULNERABLE:
| SSL POODLE information leak
| State: VULNERABLE
| IDs: CVE:CVE-2014-3566 BID:70574
| The SSL protocol 3.0, as used in OpenSSL through 1.0.1i and other
| products, uses nondeterministic CBC padding, which makes it easier
| for man-in-the-middle attackers to obtain cleartext data via a
| padding-oracle attack, aka the "POODLE" issue.

| Disclosure date: 2014-10-14
| Check results:
| TLS_RSA_WITH_AES_128_CBC_SHA
| References:
| https://www.imperialviolet.org/2014/10/14/poodle.html
| https://www.openssl.org/~bodo/ssl-poodle.pdf
| https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-3566
|_ https://www.securityfocus.com/bid/70574
|_http-trace: TRACE is enabled
| http-csrf:
| Spidering limited to: maxdepth=3; maxpagecount=20; withinhost=192.168.29.128
| Found the following possible CSRF vulnerabilities:
|
| Path: https://192.168.29.128:443/ghost/
| Form id:
| Form action: submit.php
|
| Path: https://192.168.29.128:443/WackoPicko/
| Form id: query2
| Form action: /WackoPicko/pictures/search.php
|
| Path: https://192.168.29.128:443/WackoPicko/
| Form id:
| Form action: /WackoPicko/pic' + 'check' + '.php
|

```
|     Path: https://192.168.29.128:443/phpBB2/
|     Form id:
|     Form action: login.php?sid=59a08b1b9c0f1e2717beaff32c9d3de0
|
|     Path: https://192.168.29.128:443/wordpress/
|     Form id: searchform
|     Form action: https://192.168.29.128/wordpress/
|
|     Path: https://192.168.29.128:443/AppSensorDemo/login.jsp
|     Form id:
|_    Form action: Login
| http-enum:
|   /wordpress/: Blog
|   /test/: Test page
|   /mono/: Mono
|   /crossdomain.xml: Adobe Flash crossdomain policy
|   /phpmyadmin/: phpMyAdmin
|   /wordpress/wp-login.php: Wordpress login page.
|   /icons/: Potentially interesting folder w/ directory listing
|_  /images/: Potentially interesting folder w/ directory listing
| http-fileupload-exploiter:
|
|     Failed to upload and execute a payload.
|
|     Failed to upload and execute a payload.
```

|
| Failed to upload and execute a payload.
|
| Failed to upload and execute a payload.
|
| Failed to upload and execute a payload.
|
| Failed to upload and execute a payload.
|
| Failed to upload and execute a payload.
|
| Failed to upload and execute a payload.
|
| Failed to upload and execute a payload.
|
|_ Failed to upload and execute a payload.
| ssl-dh-params:
| VULNERABLE:
| Diffie-Hellman Key Exchange Insufficient Group Strength
| State: VULNERABLE
| Transport Layer Security (TLS) services that use Diffie-Hellman groups
| of insufficient strength, especially those using one of a few commonly
| shared groups, may be susceptible to passive eavesdropping attacks.
| Check results:
| WEAK DH GROUP 1

| Cipher Suite: TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA

| Modulus Type: Safe prime

| Modulus Source: mod_ssl 2.2.x/1024-bit MODP group with safe prime modulus

| Modulus Length: 1024

| Generator Length: 8

| Public Key Length: 1024

| References:

|_ https://weakdh.org

|_http-stored-xss: Couldn't find any stored XSS vulnerabilities.

|_http-dombased-xss: Couldn't find any DOM based XSS.

445/tcp open microsoft-ds

5001/tcp open commplex-link

8080/tcp open http-proxy

| http-cookie-flags:

| /manager/html/upload:

| JSESSIONID:

| httponly flag not set

| /manager/html:

| JSESSIONID:

|_ httponly flag not set

| http-enum:

| /examples/: Sample scripts

| /manager/html/upload: Apache Tomcat (401 Unauthorized)

| /manager/html: Apache Tomcat (401 Unauthorized)

|_ /docs/: Potentially interesting folder

| http-slowloris-check:

| VULNERABLE:

| Slowloris DOS attack

| State: LIKELY VULNERABLE

| IDs: CVE:CVE-2007-6750

| Slowloris tries to keep many connections to the target web server open and hold

| them open as long as possible. It accomplishes this by opening connections to

| the target web server and sending a partial request. By doing so, it starves

| the http server's resources causing Denial Of Service.

|

| Disclosure date: 2009-09-17

| References:

| https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2007-6750

|_ http://ha.ckers.org/slowloris/

Host script results:

|_smb-vuln-ms10-061: Could not negotiate a connection:SMB: ERROR: Server returned less data than it was supposed to (one or more fields are missing); aborting [14]

|_samba-vuln-cve-2012-1182: Could not negotiate a connection:SMB: ERROR: Server returned less data than it was supposed to (one or more fields are missing); aborting [14]

| smb-vuln-regsvc-dos:

| VULNERABLE:

| Service regsvc in Microsoft Windows systems vulnerable to denial of service

| State: VULNERABLE

| The service regsvc in Microsoft Windows 2000 systems is vulnerable to denial of service caused by a null deference

| pointer. This script will crash the service if it is vulnerable. This vulnerability was discovered by Ron Bowes

| while working on smb-enum-sessions.

|_

|_smb-vuln-ms10-054: false

Nmap done: 1 IP address (1 host up) scanned in 136.69 seconds

Step 5: Web Vulnerability Scanning with nikto

Exercise 4:

Perform a vulnerability scan on your OWASP VM and record your findings:

• Vulnerabilities Found:

- Nikto v2.5.0

+ Target IP: 192.168.29.128

+ Target Hostname: 192.168.29.128

+ Target Port: 80

+ Start Time: 2024-11-02 14:10:21 (GMT-4)

+ Server: Apache/2.2.14 (Ubuntu) mod_mono/2.4.3 PHP/5.3.2-1ubuntu4.30 with Suhosin-Patch proxy_html/3.0.1 mod_python/3.3.1 Python/2.6.5 mod_ssl/2.2.14 OpenSSL/0.9.8k Phusion_Passenger/4.0.38 mod_perl/2.0.4 Perl/v5.10.1

+ /: Server may leak inodes via ETags, header found with file /, inode: 286483, size: 28067, mtime: Thu Jul 30 22:55:52 2015. See: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2003-1418

+ /: The anti-clickjacking X-Frame-Options header is not present. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options

+ /: The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type. See: https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/missing-content-type-header/

+ /cgi-bin/: Directory indexing found.

+ /crossdomain.xml contains a full wildcard entry. See: http://jeremiahgrossman.blogspot.com/2008/05/crossdomainxml-invites-cross-site.html

+ /images: IP address found in the 'location' header. The IP is "127.0.1.1". See: https://portswigger.net/kb/issues/00600300_private-ip-addresses-disclosed

+ /images: The web server may reveal its internal or real IP in the Location header via a request to with HTTP/1.0. The value is "127.0.1.1". See: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2000-0649

+ /index: Uncommon header 'tcn' found, with contents: list.

+ /index: Apache mod_negotiation is enabled with MultiViews, which allows attackers to easily brute force file names. The following alternatives for 'index' were found: index.css, index.html. See: http://www.wisec.it/sectou.php?id=4698ebdc59d15,https://exchange.xforce.ibmcloud.com/vulnerabilities/8275

+ OpenSSL/0.9.8k appears to be outdated (current is at least 3.0.7). OpenSSL 1.1.1s is current for the 1.x branch and will be supported until Nov 11 2023.

+ PHP/5.3.2-1ubuntu4.30 appears to be outdated (current is at least 8.1.5), PHP 7.4.28 for the 7.4 branch.

+ Phusion_Passenger/4.0.38 appears to be outdated (current is at least 6.0.7).

+ mod_python/3.3.1 appears to be outdated (current is at least 3.5.0).

+ Python/2.6.5 appears to be outdated (current is at least 3.9.6).

+ proxy_html/3.0.1 appears to be outdated (current is at least 3.1.2).

+ Perl/v5.10.1 appears to be outdated (current is at least v5.32.1).

+ Apache/2.2.14 appears to be outdated (current is at least Apache/2.4.54). Apache 2.2.34 is the EOL for the 2.x branch.

+ mod_perl/2.0.4 appears to be outdated (current is at least 2.0.11).

+ mod_mono/2.4.3 appears to be outdated (current is at least 3.12).

+ mod_ssl/2.2.14 appears to be outdated (current is at least 2.9.6) (may depend on server version).

+ /favicon.ico: identifies this app/server as: owasp.org. See: https://en.wikipedia.org/wiki/Favicon

+ OPTIONS: Allowed HTTP Methods: GET, HEAD, POST, OPTIONS, TRACE .

+ /: HTTP TRACE method is active which suggests the host is vulnerable to XST. See: https://owasp.org/www-community/attacks/Cross_Site_Tracing

+ mod_ssl/2.2.14 OpenSSL/0.9.8k Phusion_Passenger/4.0.38 mod_perl/2.0.4 Perl/v5.10.1 - mod_ssl 2.8.7 and lower are vulnerable to a remote buffer overflow which may allow a remote shell.

+ PHP/5.3 - PHP 3/4/5 and 7.0 are End of Life products without support.

+ /phpBB2/search.php?search_id=1\\: Retrieved x-powered-by header: PHP/5.3.2-1ubuntu4.30.

+ /phpBB2/search.php?search_id=1\\: Cookie phpbb2owaspbwa_data created without the httponly flag. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies

+ /phpBB2/search.php?search_id=1\\: Cookie phpbb2owaspbwa_sid created without the httponly flag. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies

+ /phpmyadmin/changelog.php: phpMyAdmin is for managing MySQL databases, and should be protected or limited to authorized hosts.

+ /test/: Directory indexing found.

+ /test/: This might be interesting.

+ /icons/: Directory indexing found.

+ /images/: Directory indexing found.

+ /icons/README: Apache default file found. See: https://www.vntweb.co.uk/apache-restricting-access-to-iconsreadme/

+ /wordpress/readme.html: This WordPress file reveals the installed version.

+ /wordpress/wp-login/: Admin login page/section found.

+ /wordpress/: A Wordpress installation was found.

+ /phpmyadmin/: phpMyAdmin directory found.

+ /phpmyadmin/Documentation.html: phpMyAdmin is for managing MySQL databases, and should be protected or limited to authorized hosts.

+ /#wp-config.php#: #wp-config.php# file found. This file contains the credentials.

+ /wordpress/#wp-config.php#: #wp-config.php# file found. This file contains the credentials.

+ 9063 requests: 3 error(s) and 41 item(s) reported on remote host

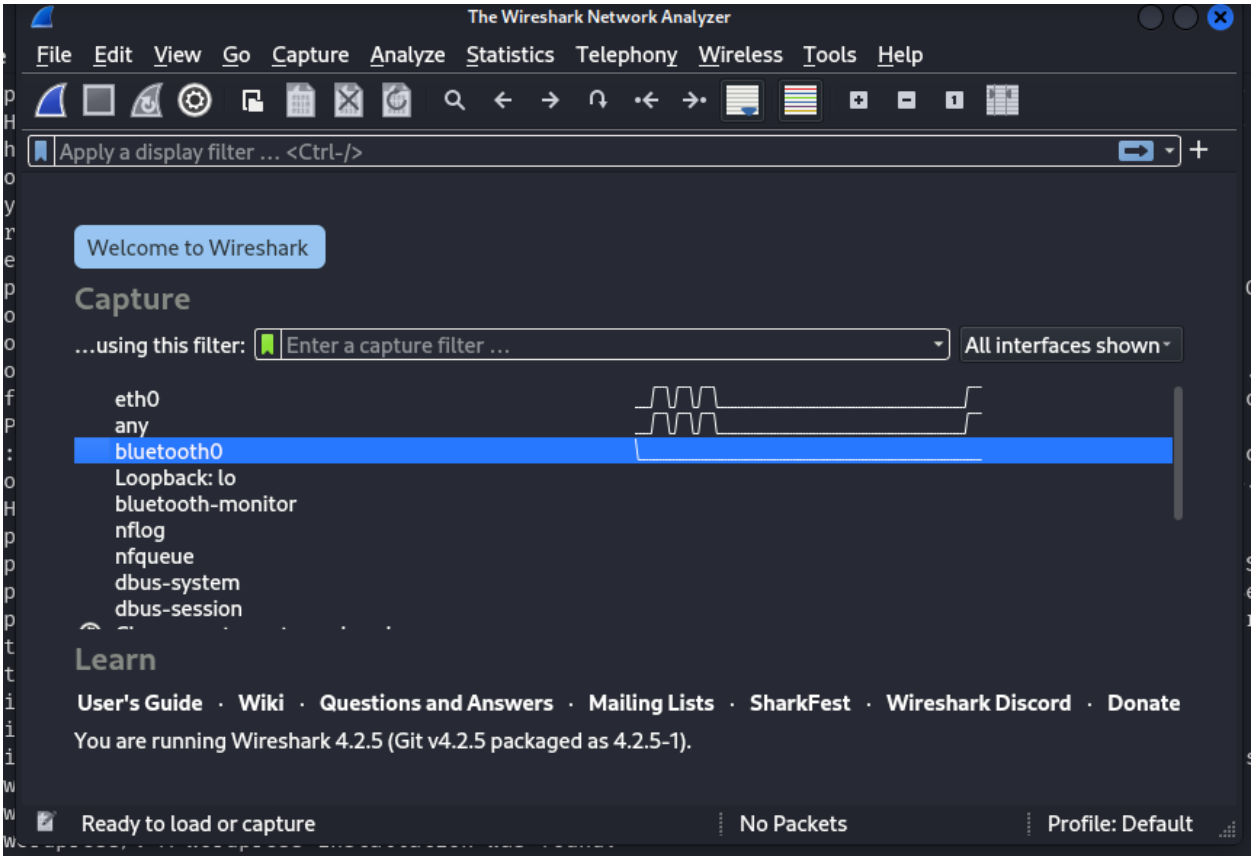
+ End Time: 2024-11-02 14:12:14 (GMT-4) (113 seconds)

+ 1 host(s) tested

Lab 5: Wireshark

Exercise 1:

• Explore the Wireshark GUI. Identify and list the main components you see, including where to find the Statistics menu



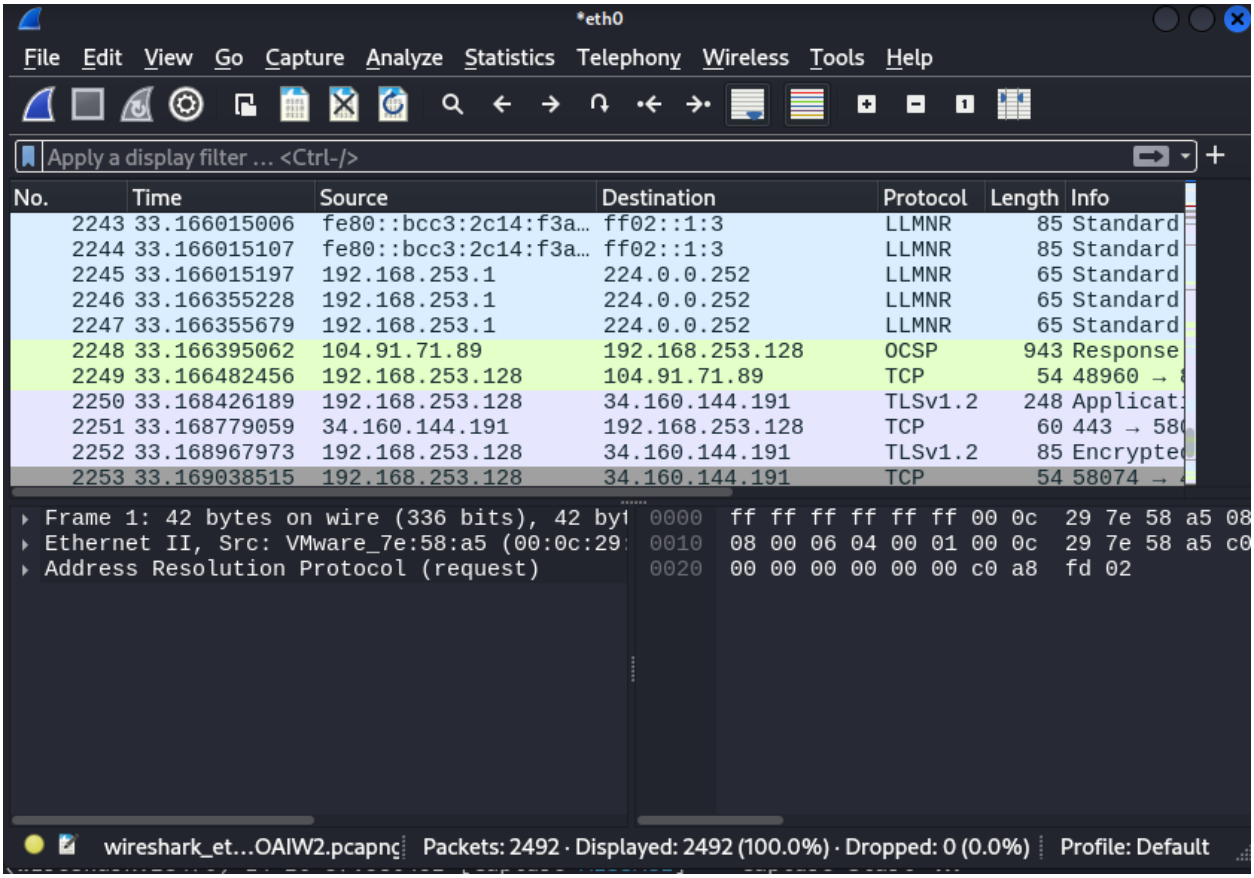
Step 2: Capturing Network Traffic

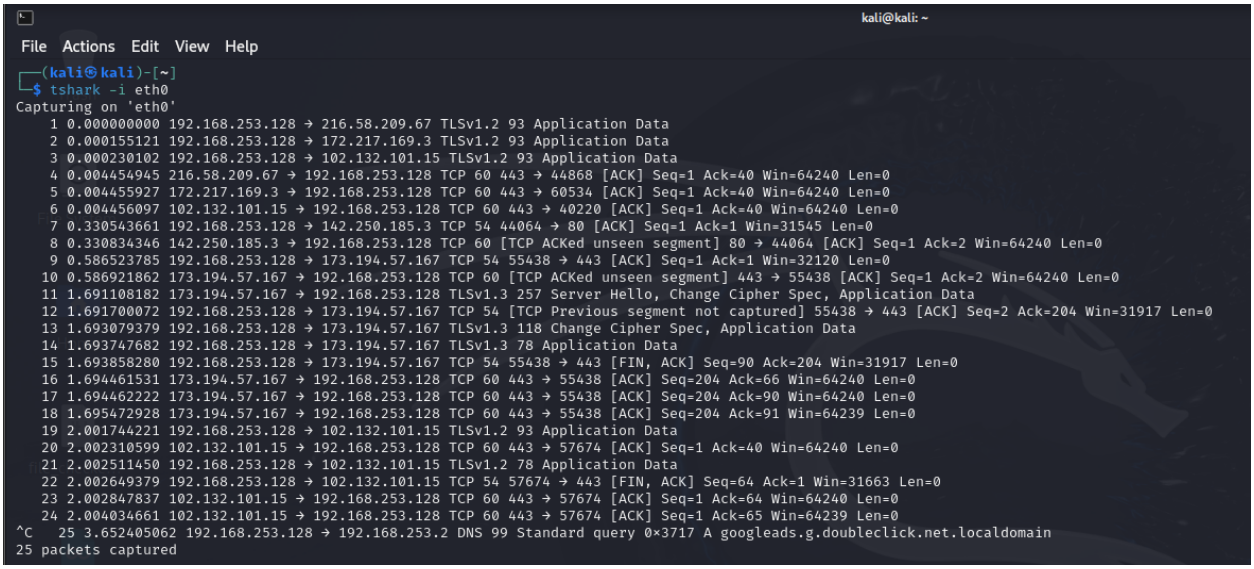
Exercise 2:

• Capture network traffic using both Wireshark and tshark.

Compare the two methods and note any differences in the user experience.

```
┌──(kali㉿kali)-[~]
└─$ wireshark
** (wireshark:28470) 14:26:37.580452 [Capture MESSAGE] -- Capture Start ...
** (wireshark:28470) 14:26:37.648498 [Capture MESSAGE] -- Capture started
** (wireshark:28470) 14:26:37.648598 [Capture MESSAGE] -- File: "/tmp/wireshark_eth0PONZW2.pcapng"
** (wireshark:28470) 14:27:26.905001 [Capture MESSAGE] -- Capture Stop ...
** (wireshark:28470) 14:27:26.941962 [Capture MESSAGE] -- Capture stopped.
** (wireshark:28470) 14:27:26.942170 [Capture WARNING] ./ui/capture.c:722 -- capture_input_closed():
** (wireshark:28470) 14:29:29.284837 [Capture MESSAGE] -- Capture Start ...
** (wireshark:28470) 14:29:29.371649 [Capture MESSAGE] -- Capture started
** (wireshark:28470) 14:29:29.371784 [Capture MESSAGE] -- File: "/tmp/wireshark_eth0TOAIW2.pcapng"
** (wireshark:28470) 14:30:15.516406 [Capture MESSAGE] -- Capture Stop ...
** (wireshark:28470) 14:30:15.561312 [Capture MESSAGE] -- Capture stopped.
** (wireshark:28470) 14:30:15.561451 [Capture WARNING] ./ui/capture.c:722 -- capture_input_closed():
```





Step 3: Analyzing Captured Packets

Exercise 3:

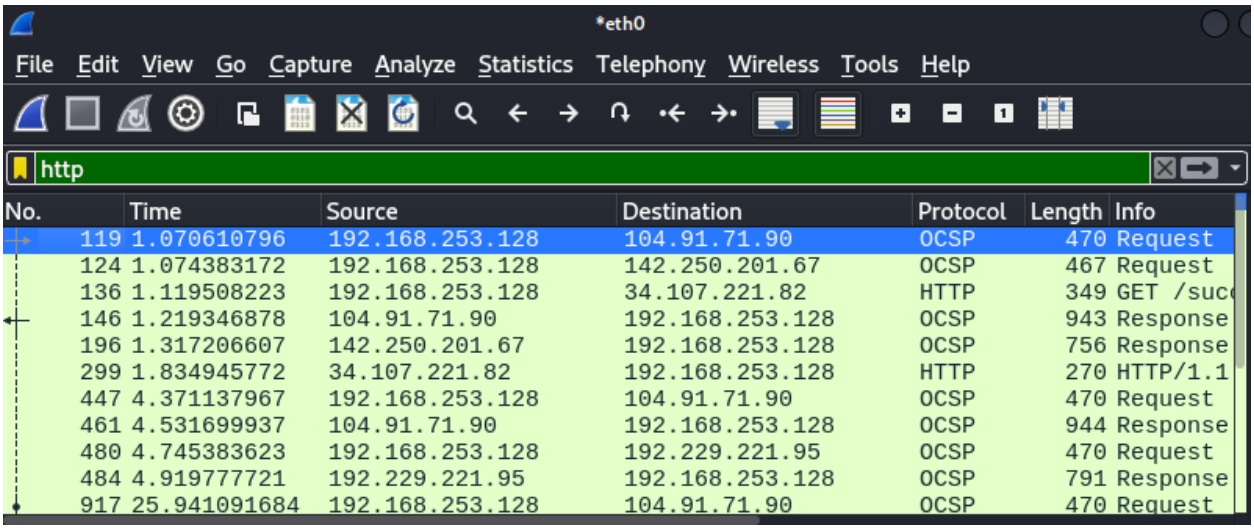
• Use filters to analyze different types of traffic.

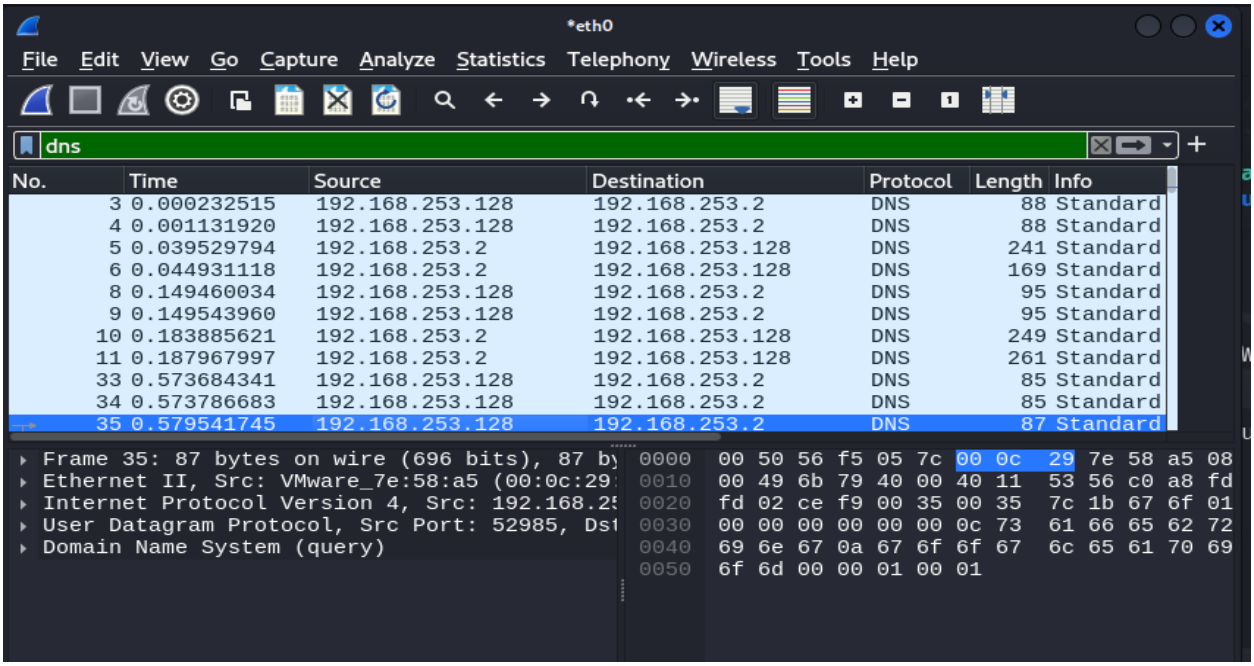
Record the following:

o Number of HTTP packets captured: 2

o Number of DNS packets captured: 128

o Specific IP addresses you identified in the traffic: 192.168.253.128





Step 4: Understanding Packet Details

Exercise 4:

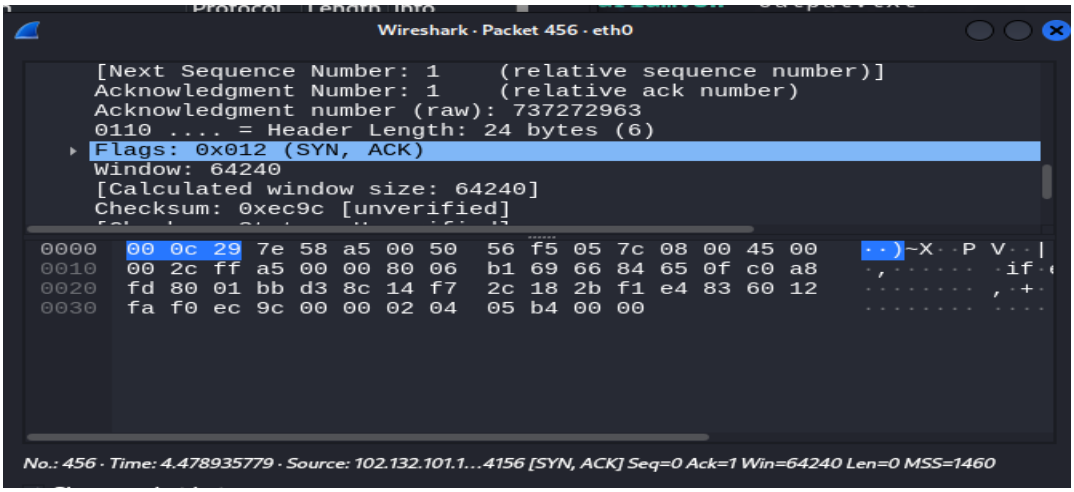
• Select a packet and list the following information:

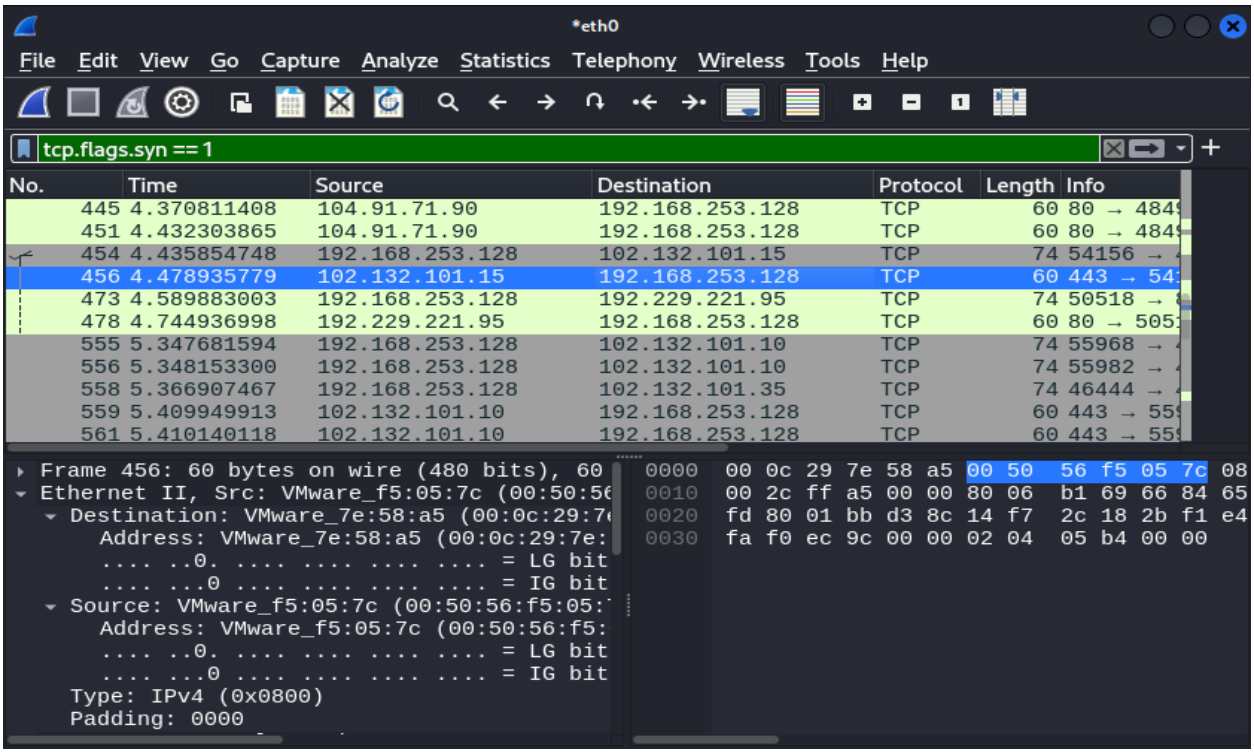
o Source IP: 102.132.101.15

o Destination IP: 192.168.253.128

o Protocol: tcp

o Any TCP Flags observed: yes

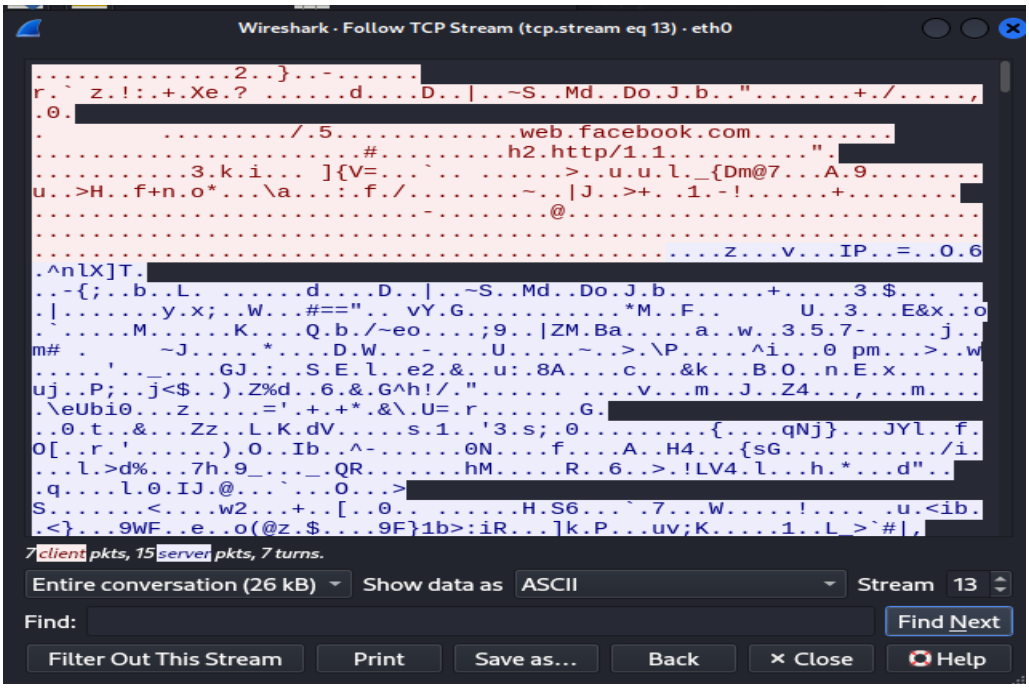




Step 5: Advanced Packet Analysis Techniques

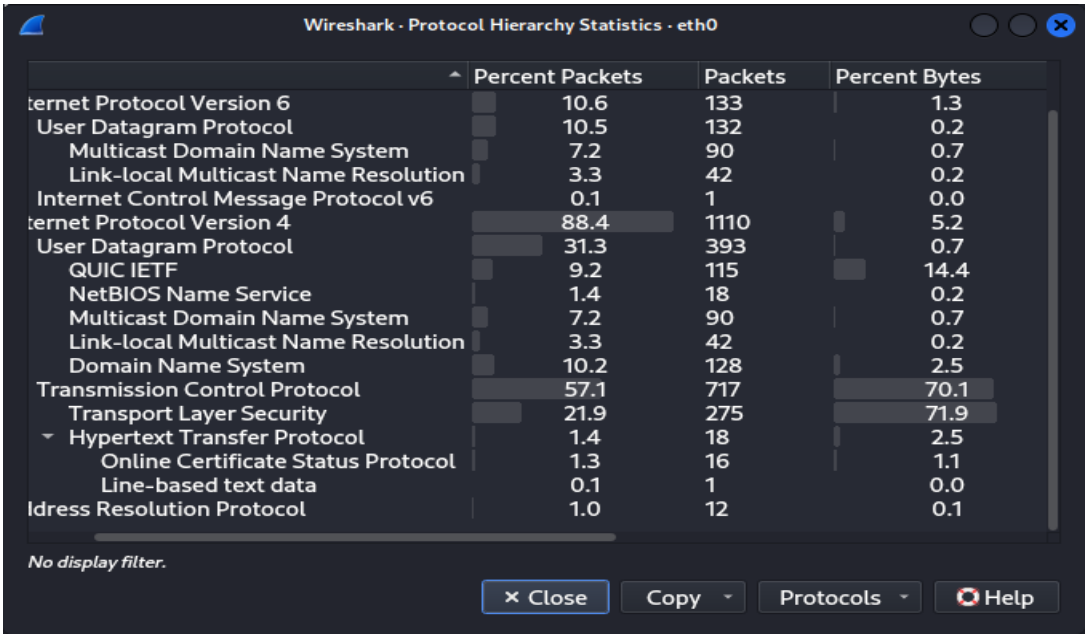
Exercise 5:

• Follow a TCP stream for a specific session and summarize the data exchanged between the client and server.

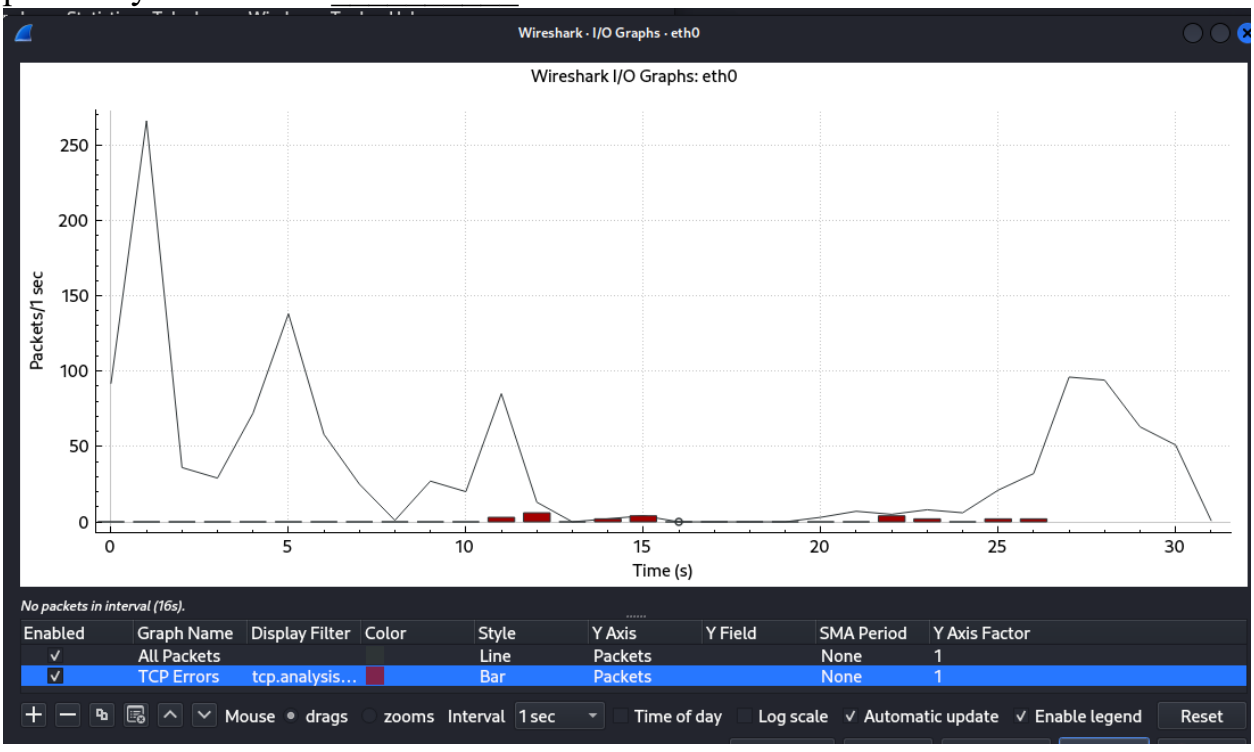


Exercise 6:

• Take a screenshot of the Protocol Hierarchy and analyze the data. Which protocol is most prevalent in your capture? Tls



Exercise 7: • Create an IO Graph showing TCP traffic. Describe any noticeable patterns you observe: _____



Step 6: Exporting Captured Data

Exercise 8:

• Save your capture file and describe a scenario where you would need to review this data later. What specific findings do you hope to extract?

 Suspicious IPs and Connections

 TCP Flags for Malicious Behavior

 Unusual Data Transfers

 Protocol Violations

Step 7: Practical Applications of Wireshark

Exercise 9:

• Describe a real-world scenario where you would use Wireshark to troubleshoot a network issue. What specific symptoms would you investigate? Packet Loss, TCP Retransmissions and Duplicates, Server-Side Errors or Misconfigurations, DNS Resolution Delays, Potential Security Threats

Exercise 10:

• Identify at least two potential security threats in your captured traffic. What indicators led you to suspect these activities? The red colour indicator led me to suspect these activities because it often highlights packets with **TCP retransmissions, duplicate acknowledgments, or issues related to delivery**. These can indicate network congestion, packet loss, or issues with reliable delivery, which might impact performance but aren't always security threats.

Lab 6: Advanced Packet Analysis Techniques

Exercise 1:

• Describe the purpose of the SYN and ACK flags in the TCP handshake. How do these flags indicate the status of a connection?

Purpose of SYN and ACK Flags

1. SYN Flag:
    - The SYN flag is used to initiate a TCP connection. When a client wants to connect to a server, it sends a TCP segment with the SYN

flag set to indicate the request for a connection. This segment also includes an initial sequence number, which is essential for synchronizing the sequence numbers between the client and server.

2. ACK Flag:
    - The ACK flag is used to acknowledge the receipt of packets. After receiving a SYN segment, the server responds with a segment that has both the SYN and ACK flags set. This response acknowledges the client's SYN request and includes the server's own initial sequence number.

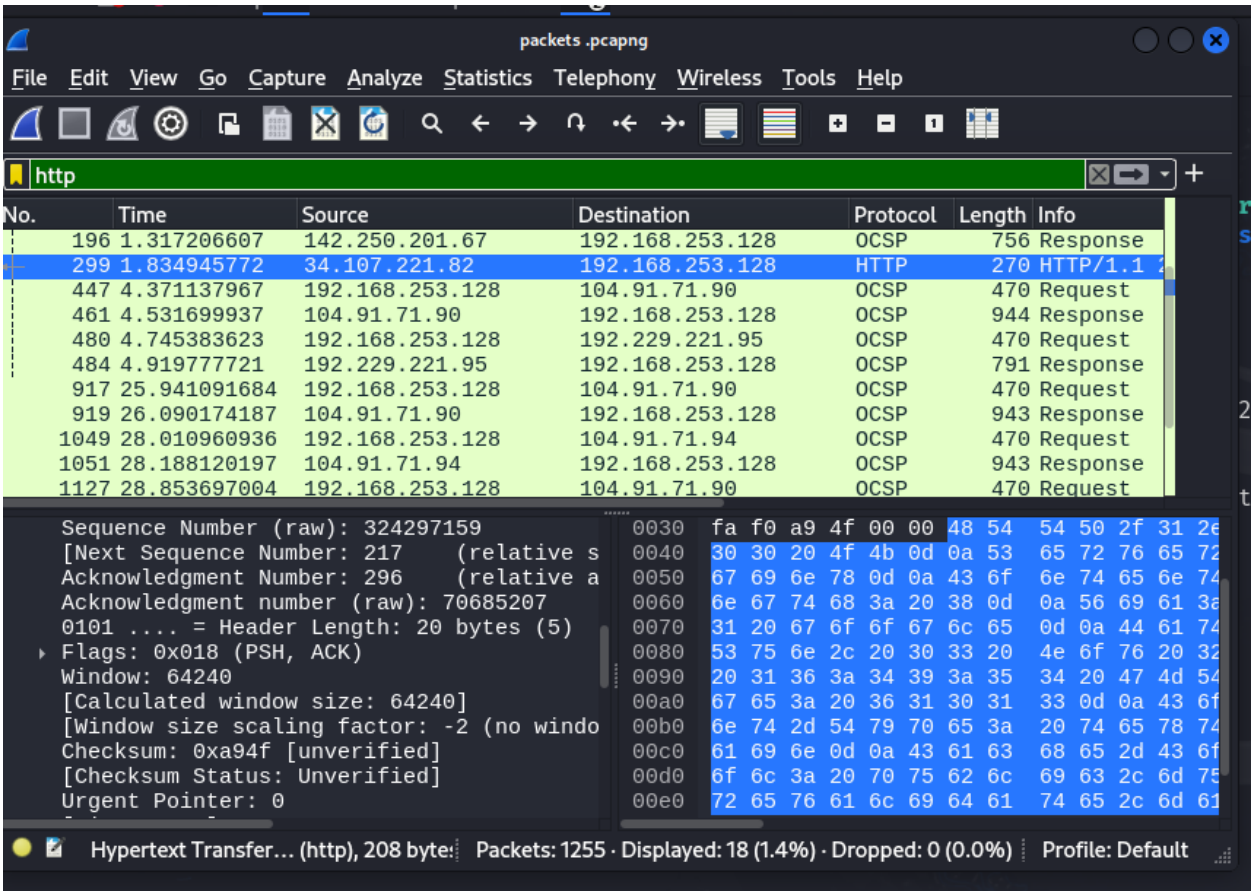
TCP Handshake Process

The TCP handshake involves three main steps:

1. SYN: The client sends a SYN packet to the server to initiate the connection.
2. SYN-ACK: The server responds with a SYN-ACK packet, acknowledging the client's request while simultaneously sending its own SYN to the client.
3. ACK: The client sends an ACK packet back to the server, completing the handshake.

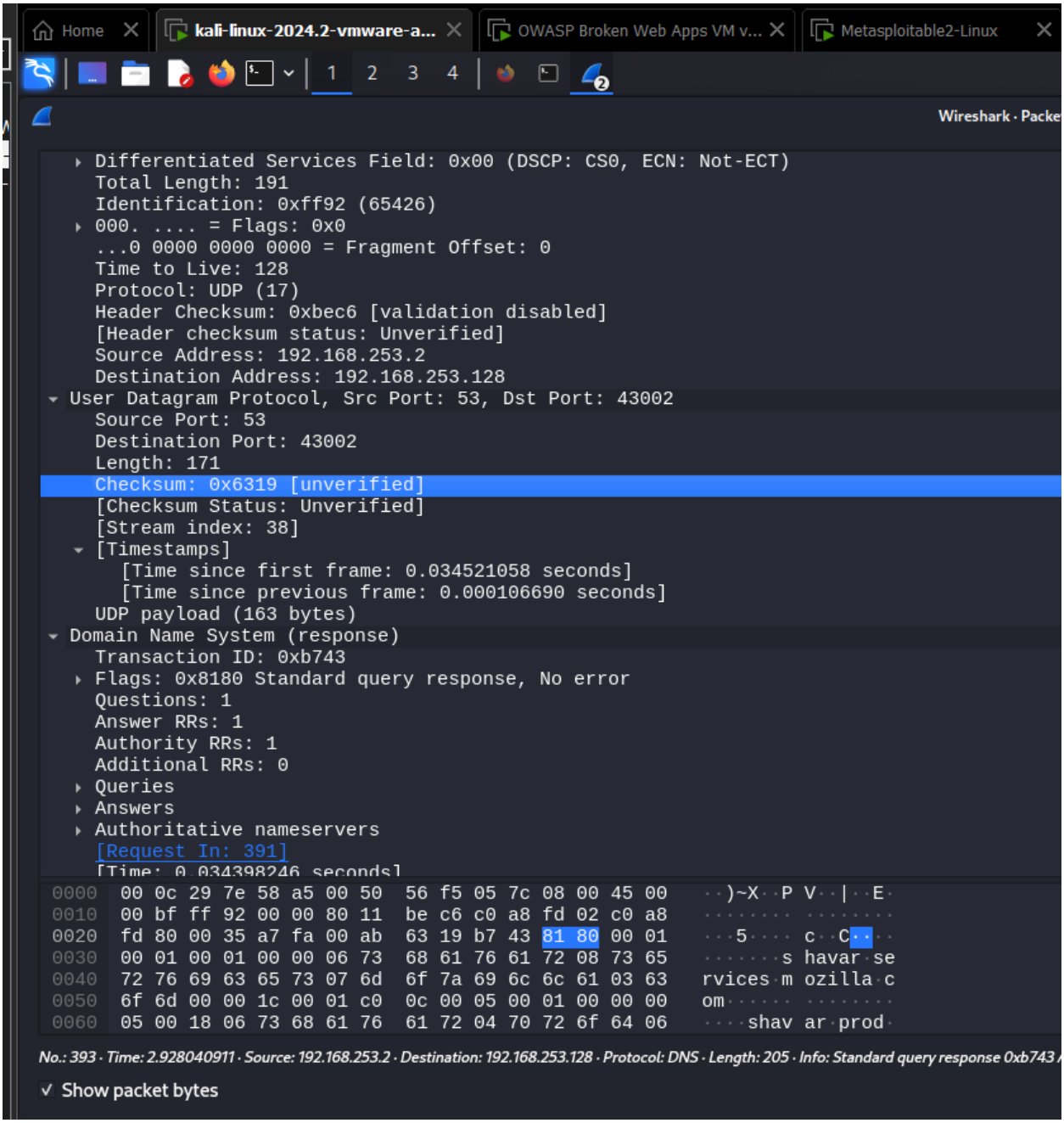
Exercise 2:

• Choose an HTTP packet and summarize its request method, status code, and any notable headers. What can you infer about the transaction?



Exercise 3:

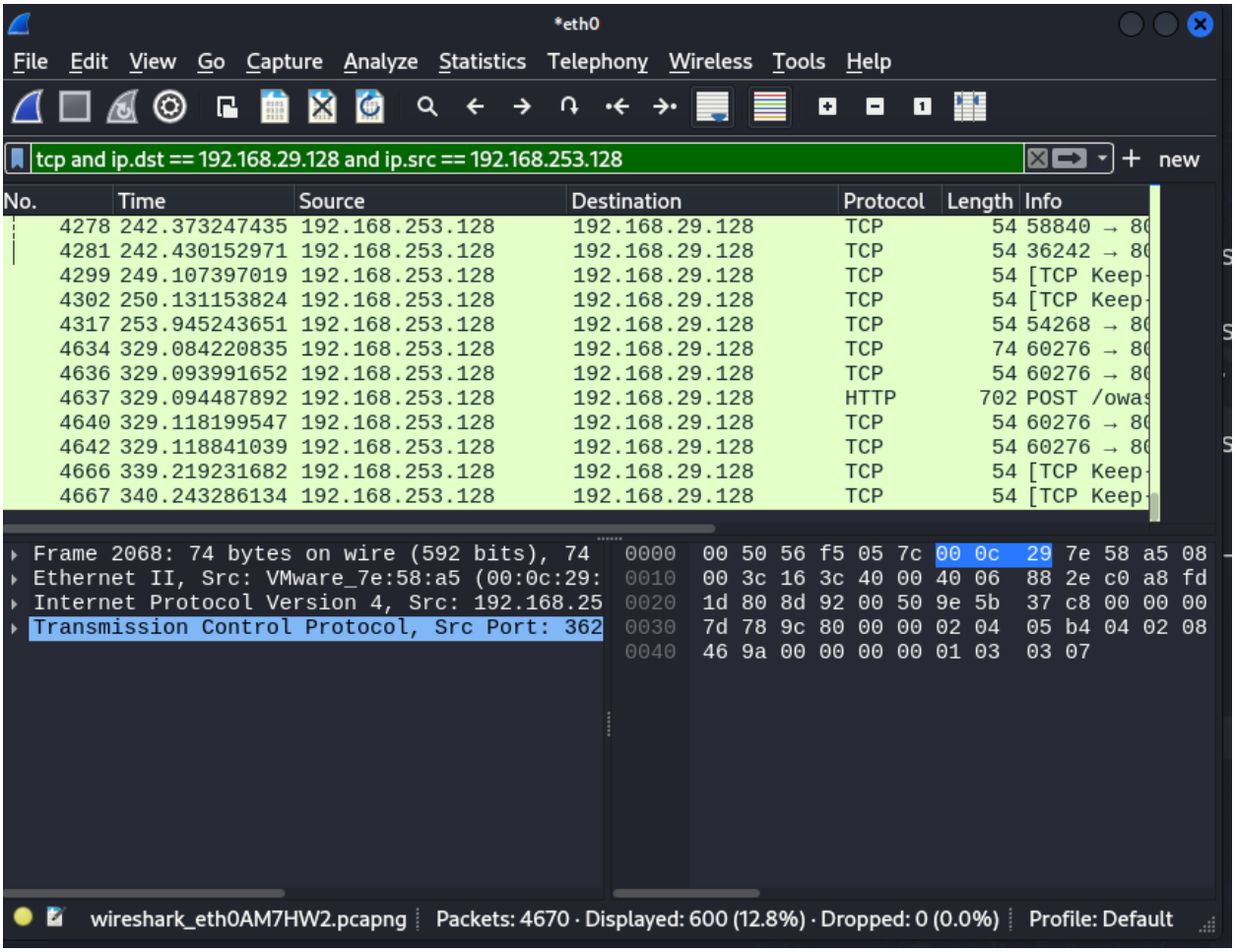
• Identify a DNS query and its corresponding response. What information does the response provide, and how is it structured?



Step 2: Creating Custom Filters

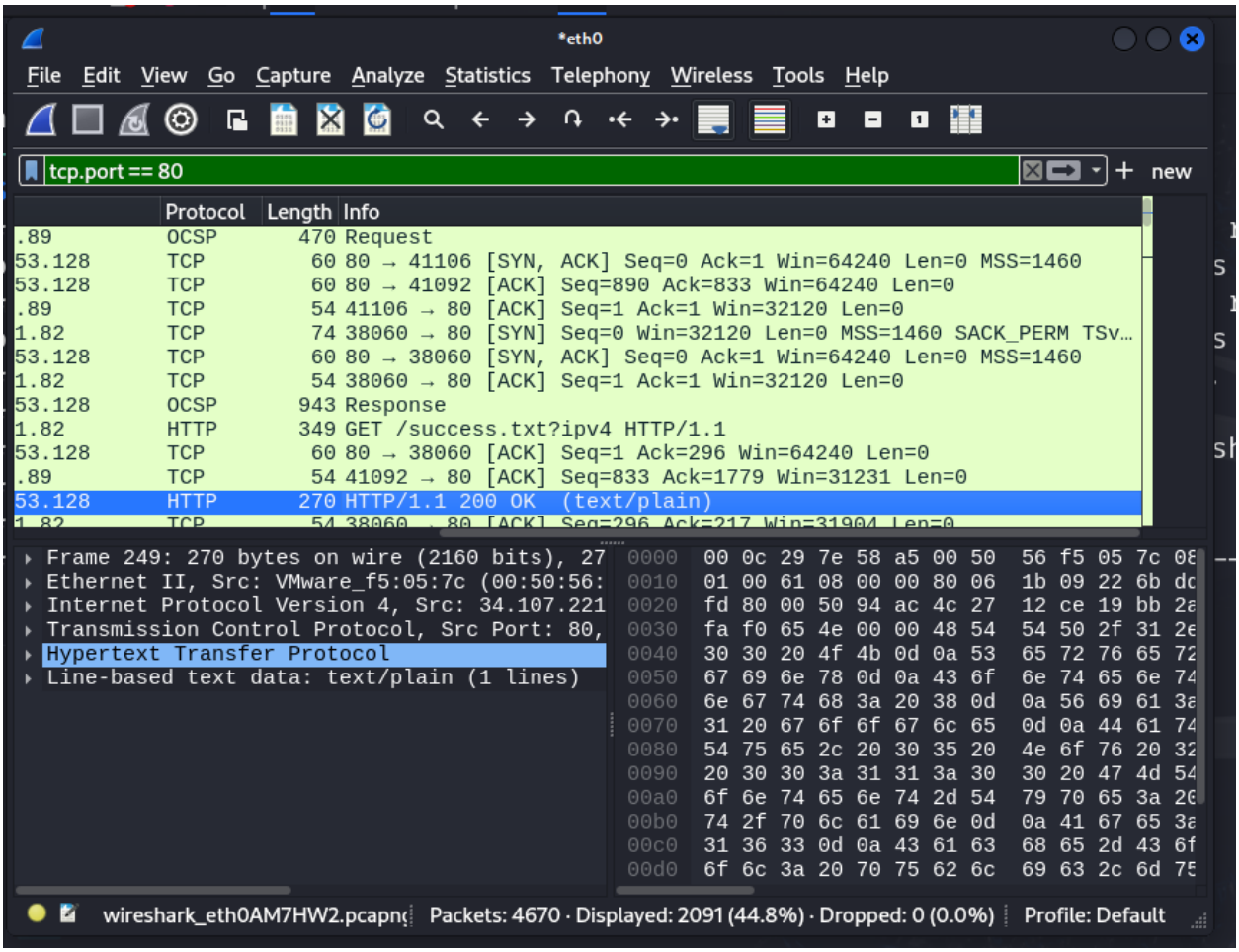
Exercise 4:

• Create a custom filter that captures only TCP traffic from your machine to a specific target IP. Document the filter syntax and the packets captured.



Exercise 5:

• Write a filter that captures traffic on a specific port (e.g., HTTP port 80) and analyze the results. What packets were captured? http port 80



Step 3: Identifying Vulnerabilities

Exercise 6:

• Analyze your capture for any anomalies or indicators of potential vulnerabilities. Document your findings and suggest possible remediation steps.

1. Look for Unusual Patterns in TCP Flags

- Indicators: High volumes of SYN packets without corresponding SYN-ACKs could indicate a SYN flood attack (a DoS attempt).
- Remediation: Implement SYN flood protections on the server, such as using SYN cookies or configuring a firewall to limit connection attempts.

2. Examine HTTP Status Codes

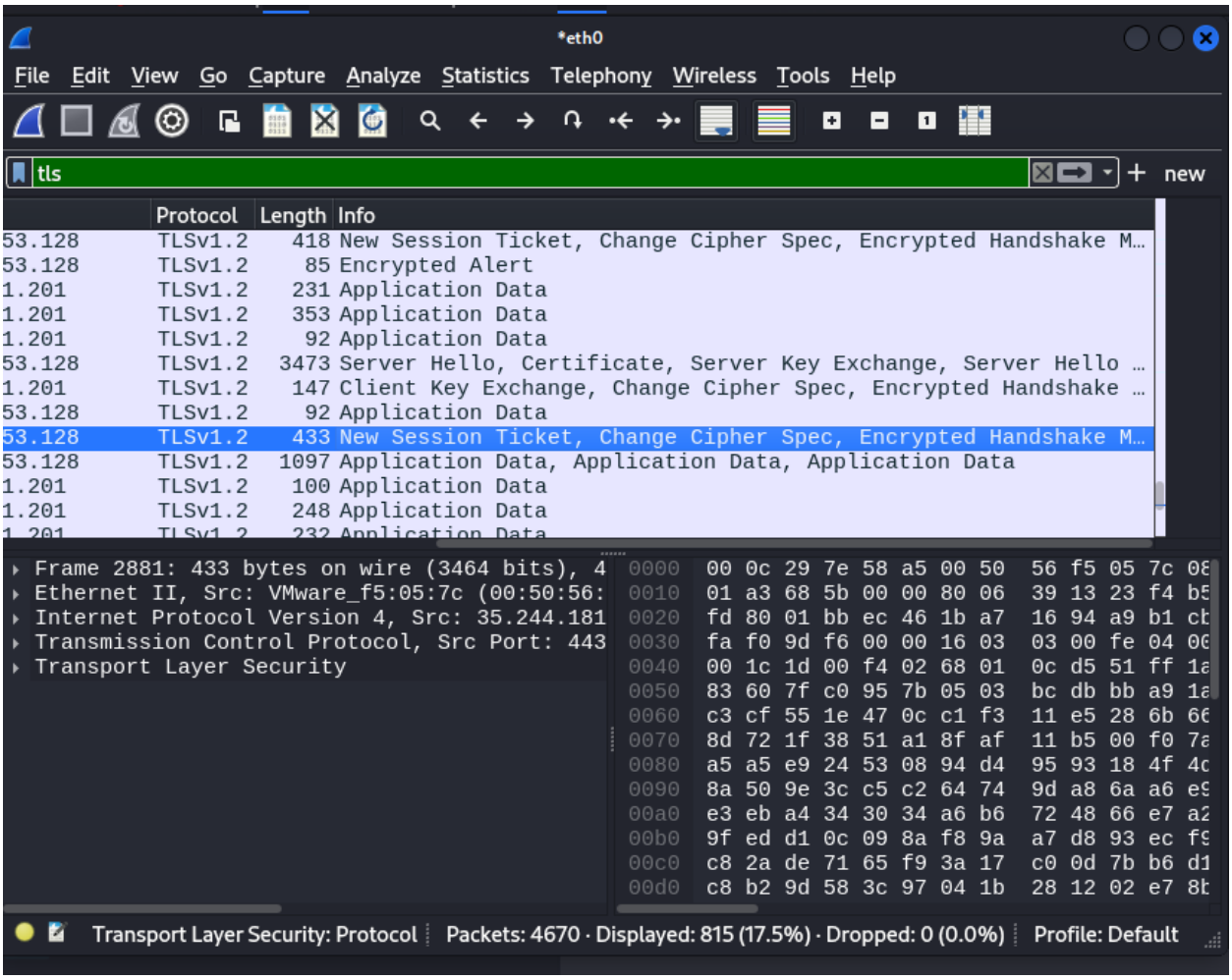
- Indicators: Frequent 404 Not Found or 403 Forbidden responses can suggest directory or file brute-forcing attempts. Numerous 500 Internal Server Error responses might indicate application errors or unhandled exceptions.
- Remediation: Enable rate limiting to prevent automated requests, restrict sensitive directory access, and sanitize input to avoid triggering errors that could expose server information.

3. Identify Sensitive Data Exposed in HTTP Traffic

- Indicators: Any HTTP traffic carrying unencrypted credentials (e.g., basic authentication headers) or other sensitive data, as HTTP traffic is not encrypted by default.
- Remediation: Migrate HTTP traffic to HTTPS to encrypt data in transit, reducing the risk of interception.

Exercise 7:

• Capture HTTPS traffic and identify the initial handshake packets. What information is exchanged during this handshake, and how does it contribute to security?



Capturing HTTPS handshake packets provides insight into how TLS establishes a secure communication channel but does not reveal the actual data exchanged due to encryption. This layered security protects sensitive information from eavesdropping and tampering, making HTTPS a critical protocol for secure web interactions.

Step 4: Practical Applications and Reporting

Exercise 8:

• Prepare a brief report summarizing your findings during the assessment. Include potential risks and recommended actions.

Overview

This report summarizes the findings from a network security assessment conducted through packet capture and traffic analysis. During the assessment, we identified

key vulnerabilities and potential security risks within the network, along with recommended mitigation actions.

1. Unsecured HTTP Traffic

- **Findings**: HTTP traffic was observed, exposing sensitive data such as usernames, session IDs, and browsing activity.
- **Potential Risks**: HTTP traffic is unencrypted, making it vulnerable to interception and man-in-the-middle attacks. Attackers could exploit this vulnerability to gain unauthorized access to user credentials or sensitive information.
- **Recommendation**: Migrate all HTTP traffic to HTTPS. Ensure SSL/TLS certificates are correctly configured on servers to enforce encrypted connections.

2. High Volume of SYN Packets

- **Findings**: A significant number of SYN packets without corresponding SYN-ACK responses were detected, suggesting potential SYN flood attacks.
- **Potential Risks**: SYN flood attacks can exhaust server resources, leading to denial of service (DoS) and network downtime.
- **Recommendation**: Implement rate limiting for connection attempts and enable SYN flood protection (e.g., SYN cookies) on network devices. Consider configuring a firewall to monitor and block excessive SYN requests.

3. Exposure of Server Version Information in Headers

- **Findings**: Server headers disclosed detailed software and version information (e.g., “Apache/2.4.49”).
- **Potential Risks**: Revealing software versions may help attackers identify known vulnerabilities, increasing the risk of targeted attacks and exploits.
- **Recommendation**: Configure servers to minimize header information, concealing version details to reduce exposure to exploits.

4. Excessive 404/403 HTTP Status Codes

- **Findings**: A high number of 404 (Not Found) and 403 (Forbidden) responses suggest potential directory brute-forcing attempts or automated scanning.
- **Potential Risks**: Brute-forcing can expose sensitive directories and files, potentially leading to unauthorized data access or server exploitation.
- **Recommendation**: Implement a web application firewall (WAF) and enforce rate limiting to block automated or suspicious requests. Additionally, limit the visibility of sensitive directories and files.

5. Weak Cookie Security Flags

- **Findings**: Some cookies lacked the Secure and HttpOnly flags.
- **Potential Risks**: Cookies without the Secure flag may be transmitted over unencrypted connections, exposing them to interception. Without the HttpOnly flag, cookies are vulnerable to theft via cross-site scripting (XSS).
- **Recommendation**: Ensure all session cookies are configured with the Secure and HttpOnly flags. This protects cookies from interception and access by malicious scripts.

Summary and Next Steps

The assessment identified vulnerabilities related to unencrypted traffic, improper configuration, and exposure to potential DoS attacks. Implementing the recommended actions will reduce these risks, improving the network's resilience against common threats. Periodic security assessments and continuous monitoring are advised to maintain and enhance security.

Exercise 9:

• Create a capture report that includes your objectives, methods, key findings, and any recommendations for improving network security.

Objective

The primary objective of this assessment was to analyze network traffic to identify potential vulnerabilities, security threats, and misconfigurations within the network infrastructure. Specific focus areas included identifying unencrypted data transmission, detecting unusual traffic patterns, and evaluating security controls for protocols in use.

Methods

1. **Packet Capture**: Used Wireshark to capture network traffic and inspect packet data, focusing on TCP and HTTP/HTTPS protocols.
2. **Filtering**: Applied custom filters to capture only traffic from specific IP addresses, ports, and protocols to isolate relevant traffic.
3. **Traffic Analysis**: Analyzed captured packets for anomalies, unusual patterns, and indicators of potential vulnerabilities, such as unencrypted data, SYN flood attempts, and exposed server headers.
4. **Security Flag Assessment**: Inspected HTTP headers and cookies to evaluate secure flag implementations and detect possible data exposure.

Key Findings

1. **Unencrypted HTTP Traffic**
    - **Issue**: Significant HTTP traffic was observed, indicating that some data, including potentially sensitive information, was transmitted unencrypted.
    - **Risk**: Unencrypted traffic exposes data to interception, potentially allowing unauthorized access to credentials and session data.
2. **High SYN Packet Activity**
    - **Issue**: High volumes of SYN packets were detected without corresponding SYN-ACK responses.
    - **Risk**: This behavior is symptomatic of a SYN flood attack, which could exhaust network resources, leading to potential denial-of-service (DoS) scenarios.
3. **Detailed Server Information in Headers**
    - **Issue**: Server headers disclosed specific software and version information (e.g., “Apache/2.4.49”).
    - **Risk**: Exposing server software and version details can help attackers exploit known vulnerabilities, increasing the likelihood of targeted attacks.
4. **Weak Cookie Security Flags**

- **Issue**: Several cookies lacked the Secure and HttpOnly flags.
- **Risk**: Cookies without these flags are vulnerable to theft via interception and cross-site scripting (XSS), posing a risk to user data and session integrity.

5. **404/403 Status Codes and Potential Scanning**
    - **Issue**: A high frequency of 404 and 403 status codes was observed, indicating potential brute-forcing or directory traversal attempts.
    - **Risk**: This activity suggests that attackers may be probing the server for accessible directories, potentially exposing sensitive files or application components.

Recommendations for Network Security Improvement

1. **Migrate HTTP to HTTPS**
    - **Action**: Implement HTTPS across the network to ensure all data transmission is encrypted. This will mitigate the risk of data interception and protect user credentials.
2. **Implement SYN Flood Protections**
    - **Action**: Configure SYN cookies and rate-limiting mechanisms to prevent SYN flood attacks. Consider using a firewall or load balancer with DoS protection features.
3. **Mask Server Details in Headers**
    - **Action**: Configure servers to remove or obscure server version details from HTTP headers, reducing the risk of targeted attacks on known vulnerabilities.
4. **Enforce Secure and HttpOnly Cookie Flags**
    - **Action**: Ensure all session-related cookies are set with the Secure and HttpOnly flags. This will prevent cookies from being accessed by malicious scripts or exposed over unencrypted connections.
5. **Monitor and Limit Directory Access**

- **Action**: Implement a web application firewall (WAF) to block excessive 404/403 requests. Rate limiting and CAPTCHA checks can further prevent unauthorized scanning or brute-force attempts.

Conclusion

The assessment has identified several vulnerabilities and areas for improvement in network security. Implementing the recommended actions will enhance security by reducing exposure to data interception, DoS attacks, and unauthorized access. Regular monitoring, periodic assessments, and proactive configuration updates are essential for maintaining a secure network environment.