The fourth challenge: network layers and wireshark

a) Learn about the OSI model

OSI Model: Open System Interconnection Model is a networking framework to implement protocols in seven layers. A conceptual framework so we can understand the interactions.

Layer 1 – Physical
Layer 2 - Data Link
Layer 3 – Network
Layer 4 – Transport
Layer 5 – Session
Layer 6 – Presentation
Layer 7 – Application

Physical ***-Layer 1 Physical examples include Ethernet, FDDI, B8ZS, V.35, V.24, RJ45.***	Conveys the bit stream - electrical impulse, light or radio signal — through the network at the electrical and mechanical level. Provides the hardware means of sending and receiving data on a carrier, including defining cables, cards and physical aspects.
Data Link ***-Layer 2 Data Link examples include PPP, FDDI, ATM, IEEE 802.5/ 802.2, IEEE 802.3/802.2, HDLC, Frame Relay.***	Data packets are encoded and decoded into bits. It furnishes transmission protocol knowledge and management and handles errors in the physical layer, flow control and frame synchronisation. The data link layer is divided into two sub layers: The Media Access Control (MAC) layer and the Logical Link Control (LLC) layer. The MAC sub layer controls how a computer on the network gains access to the data and permission to transmit it. The LLC layer controls frame synchronisation, flow control and error checking.
Network ***-Layer 3 Network examples include AppleTalk DDP, IP, IPX.***	Provides switching and routing technologies, creating logical paths, known as virtual circuits, for transmitting data from node to node. Routing and forwarding are functions of this layer, as well as addressing, inter networking, error handling, congestion control and packet sequencing.
Transport ***-Layer 4 Transport examples include SPX, TCP, UDP.***	Provides transparent transfer of data between end systems, or hosts, and is responsible for end-to-end error recovery and flow control. It ensures complete data transfer.
Session ***-Layer 5 Session examples include NFS, NetBios names, RPC, SQL.***	Establishes, manages and terminates connections between applications. The session layer sets up, coordinates, and terminates conversations, exchanges, and dialogues between the applications at each end. It deals

	with session and connection coordination.
Presentation ***-Layer 6 Presentation examples include encryption, ASCII, EBCDIC, TIFF, GIF, PICT, JPEG, MPEG, MIDI.***	Provides independence from differences in data representation (e.g., encryption) by translating from application to network format, and vice versa. The presentation layer works to transform data into the form that the application layer can accept. This layer formats and encrypts data to be sent across a network, providing freedom from compatibility problems. It is sometimes called the syntax layer.
Application ***-Layer 7 Application examples include WWW browsers, NFS, SNMP, Telnet, HTTP, FTP***	Supports application and end-user processes. Communication partners are identified, quality of service is identified, user authentication and privacy are considered, and any constraints on data syntax are identified. Everything at this layer is application-specific. This layer provides application services for file transfers, e-mail, and other network software services

b) Open a PCAP in wireshark
c) Identify DNS requests
d) Identify HTTP traffic
e) Note examples of different kinds of traffic in different layers of the OSI model

--
The fifth challenge: find some vulnerabilities

a) Learn what Nikto is and how it works
What is Nikto: Nikto is an Open Source (GPL) web server scanner which performs comprehensive tests against web servers for multiple items, including over 6700 potentially dangerous files/programs, checks for outdated versions of over 1250 servers, and version specific problems on over 270 servers. It also checks for server configuration items such as the presence of multiple index files, HTTP server options, and will attempt to identify installed web servers and software.
https://cirt.net/nikto2-docs/

```
git clone https://github.com/sullo/nikto.git
cd nikto
docker build -t sullo/nikto .
# Call it without arguments to display the full help
docker run --rm sullo/nikto
# Basic usage
docker run --rm sullo/nikto -h http://www.example.com
# To save the report in a specific format, mount /tmp as a volume:
docker run --rm -v $(pwd):/tmp sullo/nikto -h http://www.example.com -o /tmp/out.json
```

b) Use Nikto to scan the OWASP machine

c) Read up on what the alerts it gives mean

```
adumbration@Adumbration:~/Downloads/nikto$ nikto -h http://192.168.56.101
- Nikto v2.1.5
---------------------------------------------------------------------------
+ Target IP:          192.168.56.101
+ Target Hostname:    192.168.56.101
+ Target Port:        80
+ Start Time:         2019-11-25 07:48:21 (GMT11)
---------------------------------------------------------------------------
+ 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 leaks inodes via ETags, header found with file /, inode: 286483, size: 28067, mtime: 0x51c22f5365e00
+ The anti-clickjacking X-Frame-Options header is not present.
+ OSVDB-3268: /cgi-bin/: Directory indexing found.
+ /crossdomain.xml contains a full wildcard entry. See http://jeremiahgrossman.blogspot.com/2008/05/crossdomainxml-invites-cross-site.html
+ /crossdomain.xml contains 0 line which should be manually viewed for improper domains or wildcards.
+ IP address found in the 'location' header. The IP is "127.0.1.1".
+ OSVDB-630: IIS may reveal its internal or real IP in the Location header via a request to the /images directory. The value is "http://127.0.1.1/images/".
+ mod_ssl/2.2.14 appears to be outdated (current is at least 2.8.31) (may depend on server version)
+ mod_perl/2.0.4 appears to be outdated (current is at least 2.0.7)
+ Perl/v5.10.1 appears to be outdated (current is at least v5.14.2)
+ proxy_html/3.0.1 appears to be outdated (current is at least 3.1.2)
+ Apache/2.2.14 appears to be outdated (current is at least Apache/2.2.22). Apache 1.3.42 (final release) and 2.0.64 are also current.
+ mod_mono/2.4.3 appears to be outdated (current is at least 2.8)
+ Python/2.6.5 appears to be outdated (current is at least 2.7.3)
+ PHP/5.3.2-1ubuntu4.30 appears to be outdated (current is at least 5.4.4)
+ OpenSSL/0.9.8k appears to be outdated (current is at least 1.0.1c). OpenSSL 0.9.8r is also current.
+ Allowed HTTP Methods: GET, HEAD, POST, OPTIONS, TRACE
+ OSVDB-877: HTTP TRACE method is active, suggesting the host is vulnerable to XST
+ 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 (difficult to exploit). CVE-2002-0082, OSVDB-756.
+ Retrieved x-powered-by header: PHP/5.3.2-1ubuntu4.30
+ Cookie phpbb2owaspbwa_data created without the httponly flag
+ Cookie phpbb2owaspbwa_sid created without the httponly flag
+ OSVDB-3092: /phpmyadmin/changelog.php: phpMyAdmin is for managing MySQL databases, and should be protected or limited to authorized hosts.
+ OSVDB-3268: /test/: Directory indexing found.
+ OSVDB-3092: /test/: This might be interesting...
+ OSVDB-3092: /cgi-bin/: This might be interesting... possibly a system shell found.
```

+ OSVDB-3093: /.bash_history: A user's home directory may be set to the web root, the shell history was retrieved. This should not be accessible via the web.
+ OSVDB-3268: /icons/: Directory indexing found.
+ OSVDB-3268: /images/: Directory indexing found.
+ OSVDB-3268: /images/?pattern=/etc/&sort=name: Directory indexing found.*
+ Cookie phpMyAdmin created without the httponly flag
+ OSVDB-3233: /icons/README: Apache default file found.
+ Uncommon header 'x-pingback' found, with contents: http://192.168.56.101/wordpress/xmlrpc.php
+ /wordpress/: A Wordpress installation was found.
+ /phpmyadmin/: phpMyAdmin directory found
+ 6544 items checked: 1 error(s) and 35 item(s) reported on remote host
+ End Time: 2019-11-25 07:48:51 (GMT11) (30 seconds)

+ 1 host(s) tested

Extra: install OpenVAS and compare its scan results to Nikto

--

The sixth challenge: scan a malware sample with YARA

a) Learn what YARA is and how it works

YARA: A tool used by malware analysts to identify and classify malware samples. YARA can create malware families based on textual and binary patterns. Each description, a.k.a rule, consists of a set of strings and a boolean expression which determine its logic.

```
rule silent_banker : banker
{
    meta:
        description = "This is just an example"
        threat_level = 3
        in_the_wild = true
    strings:
        $a = {6A 40 68 00 30 00 00 6A 14 8D 91}
        $b = {8D 4D B0 2B C1 83 C0 27 99 6A 4E 59 F7 F9}
        $c = "UVODFRYSIHLNWPEJXQZAKCBGMT"
    condition:
        $a or $b or $c
}
```

b) Install Yara on Parrot OS

c) Obtain a malware sample (http://tracker.virusshare.com:6969/)

d) Obtain a set of yara files **[Rules Master]**

e) Scan samples with yara files and find interesting results **[yara -w ~/*Downloads*/rules-master/filename .]**

IMPORTANT: Only download and use these samples in a snapshotted, host-only networked virtual Linux machine!

--
The seventh challenge: learn how to exploit a network

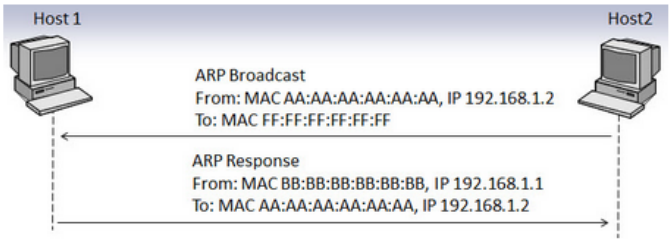
In this challenge we are going to learn about the ARP cache, and how to use it to our advantage.

Preparation: install a Windows 7 VM, a Kali Linux VM and a Parrot Security VM. Place them in a Host-only network.

https://www.cellstream.com/reference-reading/tipsandtricks/219-what-is-the-arp-command-and-how-can-i-use-it
First some questions:
a) What is ARP? What is it used for?
ARP stands for the Address Resolution Protocol. It is a network layer protocol used to convert an IP address into a physical address (MAC Address).
EG: A host wishing to obtain a physical address sends an ARP request to the TCP/IP Network. Host on the network that has IP Address in request replies with physical hardware address.

- **One part determines a physical address when sending a packet**
- **Other part answers requests from other machines**

The operation of the ARP protocol looks like this:



1. Process begins with caches being empty
2. Host 2 knows that it wants to send a packet to Host 1 (eg Default GW)
3. Host 2 has to send a broadcast ARP message (destination FF:FF:FF:FF:FF:FF) requesting an answer for 192.168.1.1.
4. Host 1 responds with its MAC address
5. Host 1 and 2 both insert this received information into their ARP caches for future use

https://www.cellstream.com/reference-reading/tipsandtricks/219-what-is-the-arp-command-and-how-can-i-use-it

b) How to read ARP in Linux and Windows?

We can read ARP in Linux and windows through the command arp-a

arp [-a [InetAddr] [-N IfaceAddr]] [-g [InetAddr] [-N IfaceAddr]] [-d InetAddr [IfaceAddr]] [-s InetAddr EtherAddr [IfaceAddr]]

Here are the switch definitions:

-a [InetAddr] [-NIfaceAddr] : Displays current ARP cache tables for all interfaces. To display the ARP cache entry for a specific IP address, use arp -a with the InetAddr parameter, where InetAddr is an IP address. To display the ARP cache table for a specific interface, use the -N IfaceAddr parameter where IfaceAddr is the IP address assigned to the interface. The -N parameter is case-sensitive.https://www.cellstream.com/reference-reading/tipsandtricks/219-what-is-the-arp-command-and-how-can-i-use-it

-g [InetAddr] [-NIfaceAddr] : Identical to -a.

/?: Displays help at the command prompt.

c) How to add entries to ARP in Windows and Linux?

-sInetAddr EtherAddr [IfaceAddr] : Adds a static entry to the ARP cache that resolves the IP address InetAddr to the physical address EtherAddr. To add a static ARP cache entry to the table for a specific interface, use the IfaceAddr parameter where IfaceAddr is an IP address assigned to the interface.

d) How to delete entries to ARP in Windows and Linux?

-dInetAddr [IfaceAddr] : Deletes an entry with a specific IP address, where InetAddr is the IP address. To delete an entry in a table for a specific interface, use the IfaceAddr parameter where IfaceAddr is the IP address assigned to the interface. To delete all entries, use the asterisk (*) wildcard character in place of InetAddr. So "arp -d *" will flush your ARP cache.

e) What is a MITM attack?

An MITM Attack (Man In the Middle Attacks) – An attack against various cryptographic protocols. An attacker sits in the middle and negotiates/manipulates the protocol within a client and server

f) What is ARP spoofing?

ARP Spoofing – Also known as ARP poising routing (APR) or ARP cache poising, this technique tries to steal data for a victim and possibly without them knowing. Attacks an

Ethernet LAN by updating a target computer's ARP cache with a forged ARP request and reply packets to change the Layer 2 Ethernet MAC Address to one an attacker can monitor. As ARP replies are forged, target computer sends frames meant for original destination to attacker's computer first so frames can be read.

g) What is Ettercap?

Ettercap is a comprehensive suite for man in the middle attacks. It features sniffing of live connections, content filtering on the fly and many other interesting tricks. It supports active and passive dissection of many protocols and includes many features for network and host analysis.

Some excersices:

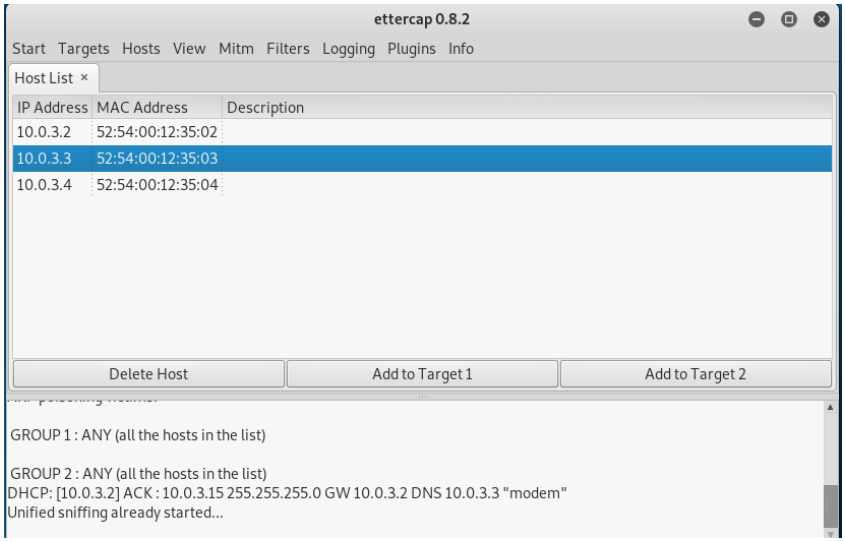
h) Boot up all VM's with Host-Only networking only

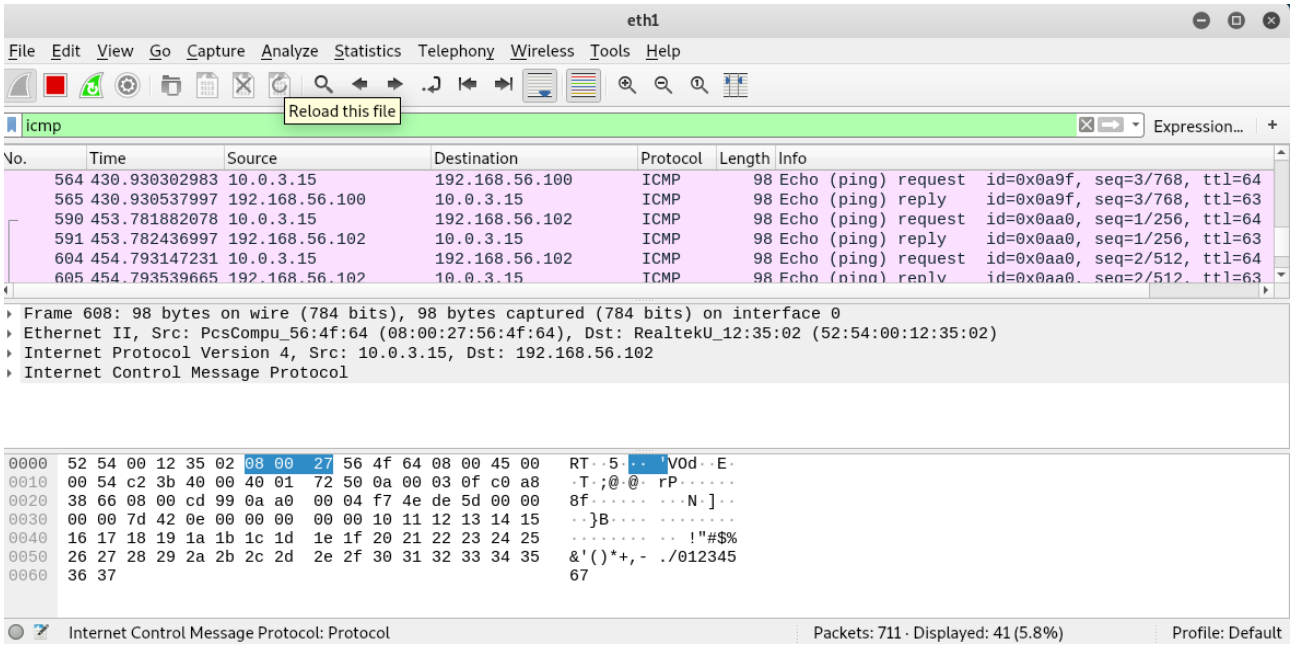
i) Sniff the network using Ettercap

j) Do an ARP poisoning attack using Ettercap

k) Generate traffic between the VM's

l) What do you see?**Ping the VM IP's**




eth1
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Reload this file
icmp
Expression...
No. Time Source Destination Protocol Length Info
564 430.930302983 10.0.3.15 192.168.56.100 ICMP 98 Echo (ping) request id=0x0a9f, seq=3/768, ttl=64
565 430.930537997 192.168.56.100 10.0.3.15 ICMP 98 Echo (ping) reply id=0x0a9f, seq=3/768, ttl=63
590 453.781882078 10.0.3.15 192.168.56.102 ICMP 98 Echo (ping) request id=0x0aa0, seq=1/256, ttl=64
591 453.782436997 192.168.56.102 10.0.3.15 ICMP 98 Echo (ping) reply id=0x0aa0, seq=1/256, ttl=63
604 454.793147231 10.0.3.15 192.168.56.102 ICMP 98 Echo (ping) request id=0x0aa0, seq=2/512, ttl=64
605 454.793539665 192.168.56.102 10.0.3.15 ICMP 98 Echo (ping) reply id=0x0aa0, seq=2/512, ttl=63
Frame 608: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
Ethernet II, Src: PcsCompu_56:4f:64 (08:00:27:56:4f:64), Dst: RealtekU_12:35:02 (52:54:00:12:35:02)
Internet Protocol Version 4, Src: 10.0.3.15, Dst: 192.168.56.102
Internet Control Message Protocol
0000 52 54 00 12 35 02 08 00 27 56 4f 64 08 00 45 00
0010 00 54 c2 3b 40 00 40 01 72 50 0a 00 03 0f c0 a8
0020 38 66 08 00 cd 99 0a a0 00 04 f7 4e de 5d 00 00
0030 00 00 7d 42 0e 00 00 00 00 00 10 11 12 13 14 15
0040 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 24 25
0050 26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33 34 35
0060 36 37
Internet Control Message Protocol: Protocol
Packets: 711 · Displayed: 41 (5.8%)
Profile: Default