E.T.S. de Ingeniería Industrial,
Informática y de Telecomunicación

Título del Trabajo Fin de Grado

Grado en Ingeniería Informática

Trabajo Fin de Grado

Iban Ruiz de Galarreta Cadenas

Santiago García Jiménez

Pamplona, fecha de defensa

upna
Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

Contenido

1. INRODUCCIÓN

Este proyecto consiste en un programa que genera maquinas con servicios web vulnerables. De forma aleatoria cambia la apariencia, el tipo de vulnerabilidades a explotar y otras características de la máquina, como por ejemplo la base de datos. Constará de dos partes: una en la que a través de distintos fallos web se obtiene ejecución remota de la maquina y otra en la que se conseguirá elevar los privilegios para obtener privilegios de administrador. Aunque, el proyecto se centrará más en la primera parte.

Las maquinas vulnerables se crean como contenedores de Docker. Un script en C se encarga de establecer la configuración adecuada y aleatoria de cada máquina.

2. OBJETIVOS

El objetivo principal es que sea una herramienta útil para los estudiantes de ciberseguridad, que puedan generar maquinas que les suponga un reto para practicar. Este proyecto debe ser bastante modular, de forma que se le puedan añadir nuevas vulnerabilidades fácilmente y de esta forma ser ampliable.

El programa contará con una lista de vulnerabilidades web con las que se podrá obtener acceso remoto al servidor mediante distintos pasos y otra lista de vulnerabilidades para escalar privilegios. Aun se pretende que la parte importante sea la primera.

3. FUNCIONAMIENTO

Esta herramienta se puede usar de dos formas distintas, una está directamente dedicada al alumno, ejecutando el programa en una consola generara una maquina (*Imagen 1*) en el puerto 80 (si no se especifica). Una vez que se exploten las vulnerabilidades de esa máquina se obtendrán dos "flags" una cuando se obtiene acceso remoto a la maquina y otra cuando se consigue acceso privilegiado. Estas flags se pueden introducir mediante "-f" para completar la maquina (*Imagen 2*). La otra forma de utilizarla tiene más funciones, mediante "-m" se pueden crear un número ilimitado de máquinas, las cuales irán a la carpeta "dockers" del proyecto (*Imagen 3*). Esas máquinas se pueden utilizar para, por ejemplo, subirlas a una herramienta de CTFs como hackthebox, tryhackmy…

```
❯ ./ctf-generator
Creadndo la nueva maquina...
La maquina esta corriendo en el puerto 80.
```

Imagen 1

```
❯ ./ctf-generator -f 646zjia8ua4td2q12w9z6ykm7z5bjbhv
Enhorabuena!! Has conseguido la flag de User.

Objetivos:
Flag de Usuario Conseguida: Sí
Flag de Root Conseguida: No
```

Imagen 2

```
❯ ./ctf-generator -m 3
Maquina 'mvuln1' creada con éxito.
Maquina 'mvuln2' creada con éxito.
Maquina 'mvuln3' creada con éxito.
```

Imagen 3

4. IMPLEMENTACIÓN

Este proyecto consta de varios archivos necesarios para la creación de la maquina a parte del propio programa escrito en C. En este apartado se explicará la distribución del proyecto y los distintos módulos en C. La carpeta principal consta del programa compilado, un "Makefile" para facilitar la compilación de este y el "README". También encontramos la carpeta "assets" la cual más adelante se explicará su función. la carpeta "doc" que contiene la documentación necesaria del programa. La carpeta "dockers" donde Irán las maquinas una vez creadas. Y por último las carpetas "src", "include" y "obj", donde irán las ".c", ".h" y ".o" respectivamente, más adelante se explicarán los distintos módulos del programa.

4.1 Carpeta "assets"

En esta carpeta se encuentran los archivos necesarios para la creación de las maquinas, las carpetas más importantes son "**bddsConf**", "**listas**" y "**ejemploDocker**".

En la primera hay una carpeta distinta por cada base de datos que haya implementada en el proyecto "bdd1", "bdd2", "bdd3" … Dentro de estas habrá, mínimo, dos archivos, uno que inicia los servicios de apache y la base de datos que toque llamado "inicio.sh". Y otro llamado "install.sh" que contendrá los comandos necesarios que la maquina tiene que ejecutar para instalar esa base de datos. Este archivo esta sin acabar, ya que el script se encarga de terminar el archivo con datos aleatorios.

En "**listas**" se ubica una serie de listas en la que cada línea tiene un dato, para que el script elija una línea aleatoria y la implemente. Por ejemplo, podemos encontrar las listas "contraseñas.txt" y "usuarios.txt" que contienen distintas contraseñas y distintos usuarios para que sean elegidos aleatoriamente, o "sudo.txt" que contiene la ruta de todos los archivos vulnerables por sudo que contiene la máquina.

"**ejemploDocker**" es un ejemplo del archivo de una máquina, pero sin acabar. Esta carpeta se copiará y pegará en la carpeta "dockers" y a partir de ahí se ira modificando con la configuración. Consta de dos archivos "start.sh" y "stop.sh" que inician y paran la máquina, en estos archivos el script establece un puerto y un nombre distinto para cada máquina. También se puede especificar el puerto pasándoselo a "start.sh" como argumento. A parte estará el archivo "Dockerfile" que contiene la configuración necesaria para crear la maquina y la carpeta "src" con los archivos web.

4.2 Modulo “main”

Este es el módulo principal, tiene una serie de Condiciones para leer los distintos argumentos y llamar a las funciones correspondientes. Al principio comprueba que solo se le pasen 2 argumentos como mucho, ya que cada argumento es independiente. Al final comprueba que los requisitos sean correctos (como por ejemplo tener Docker instalado) y que no haya una maquina ya corriendo y en ese caso la crea.

4.3 Modulo “argumentos”

Este módulo recoge las funciones necesarias para el correcto funcionamiento de los argumentos pasados al programa.

Tenemos la función “**mostrarAyuda**” que saca por pantalla la información sobre los argumentos. “**cancelarMaquina**”, “**pararMaquina**” y “**ejecutarMaquina**” que eliminan la máquina, la pausan y la reanudan respectivamente (“-c”,”-s”,”-r”). Las funciones “**requisitosCorrectos**” y “**maquinaCorriendo**” que comprueban que Docker este instalado y funcionando y si hay ya una maquina corriendo con el nombre de “altair”. También dispone de las funciones “**introduccirFlag**” y “**mostrarObjetivos**” para administrar las flags. Y, por último, “**crearVariasMaquinas**” que llama a la función encargada de crear una maquina el número de veces introducido.

4.4 Modulo “herramientas”

En este módulo se ha ido recopilando funciones necesarias para la ejecución del programa pero que tienen varios usos distintos a lo largo del proyecto. Entre ellas las funciones principales son ejecutar comandos, modificar archivos de texto, generar las flags, obtener valores aleatorios…

Este módulo es usado por todos los otros en algún momento.

4.5 Modulo “maquina”

Este es probablemente el módulo más importante del programa, se encarga de crear la máquina. Mas concretamente, la función “**crearNuevaMaquina**” copia el contenido del ejemplo Docker y llama a las distintas funciones que se van a encargar de modificarlo con los valores necesarios. Las funciones que cambian estos valores son “**establecerPuerto**”, “**establecerNombre**” y “**cambiarEstilo**” que cambian el puerto que especifiquemos (si lo especificamos) le dan un nombre (“altair” en caso de ejecutar el programa sin “-m”) y establecer un estilo para los CSS. También está “**anadirFlags**” que las introduce en la maquina y fuera de ella. Y “**crearUsuarios**” que modifica el archivo de la base de datos para añadirle una cantidad de usuarios aleatoria con una cantidad de atributos también aleatoria para que no sea fácil de

predecir. Por último, tenemos la función "**elegirBdd**" y una función por cada posible motor de base de datos que tengamos programado, en un primer momento disponemos de "MYSQL", "PGSQL" y "SQLite", estas funciones elegirán un motor aleatorio y lo configurarán en la máquina.

En este módulo también se encuentra la función "**iniciarNuevaMaquina**" que la pondrá en marcha en caso de haber ejecutado el programa sin el argumento "-m".

4.6 Modulo "vulnerabilidades"

Este módulo tiene 3 funciones principales, "**crearVulnerabilidadLogin**", "**crearVulnerabilidadElevacion**" y "**crearVulnerabilidadEjecucion**" cada una va a elegir aleatoriamente un tipo de vulnerabilidad y llamara a la función correspondiente que configure esa vulnerabilidad. esta es la parte más ampliable del proyecto, aquí se pueden ir sumando funciones con vulnerabilidades distintas y luego añadirla a una de estas tres funciones mencionadas. No se va a explicar cada una de estas funciones ya que se explicarán las vulnerabilidades en el apartado de "Write-up".

5. WRITE-UP

Las diferentes vulnerabilidades que se pueden ejecutar en esta máquina se dividen en tres categorías. Pueden ser vulnerabilidades en la ventana de login, vulnerabilidades de ejecución de código dentro del cuerpo de la web y las vulnerabilidades de la escalada de privilegios.

5.1 Vulnerabilidades login

Usuario y contraseña relajados

Esta vulnerabilidad consiste en que el administrador a añadido a la base de datos usuarios por defecto que fácilmente se pueden encontrar en un diccionario. En concreto he utilizado las credenciales de las listas "mysql-betterdefaultpasslist.txt" y "postgres-betterdefaultpasslist.txt" de "**Seclist**" en la ubicación "SecLists/Passwords/Default-Credentials". También se puede encontrar en el proyecto en "assets/listas/usuarioyPass.txt".

Con un simple ataque de diccionario con alguna herramienta como "Burpsuite" y estas listas se consigue acceder fácilmente.

SQL Injection

Este, es un SQL injection sencillo, la consulta de SQL del login está programada sin "prepared statment" y el formulario no impide la introducción de caracteres extraños, por lo que podemos introducir un texto para que la respuesta sea siempre cierta. Por ejemplo, podemos introducir "**a' or 1=1 --**" (*Imagen $*) esto cierra el nombre de usuario después de la "a" y añade que el 1 sea igual a 1, siempre se cumplirá. Por último, añade "--" para que no de error con el resto de los caracteres. De contraseña se puede introducir cualquier carácter.

Habremos iniciado sesión con un usuario inventado, pero tendremos acceso al contenido de la web (*Imagen $*).

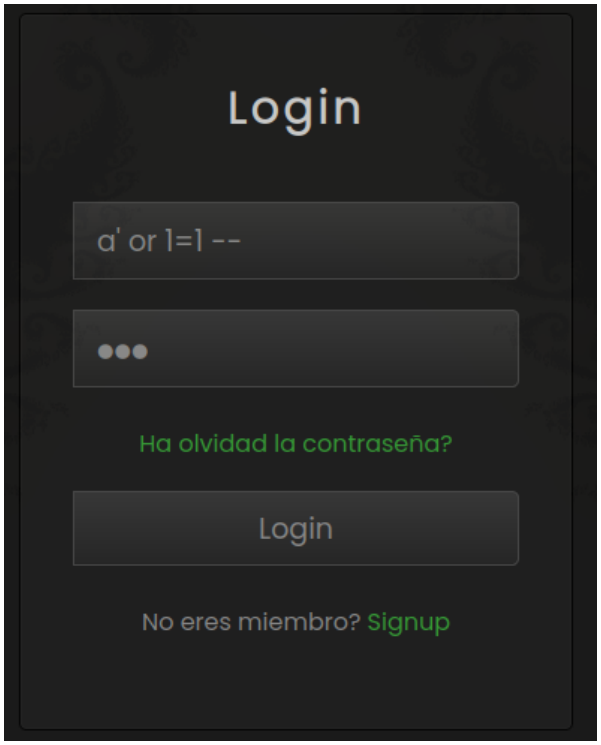

Imagen $

¡Bienvenido, a' or 1=1 --!

Imagen $

5.2 Vulnerabilidades ejecución de comandos

5.3 Vulnerabilidades de escalada de privilegios

Archivo con sudo

Para descubrir si estamos ante esta vulnerabilidad basta con ejecutar el comando "**sudo -l**". Si es así nos aparecerá un archivo con permisos de sudo "**ALL ALL=(ALL) NOPASSWD: #archivo#**" siendo "#archivo#" el archivo con permisos de sudo (*Imagen $*).

El archivo será uno recogido en la lista de "GTFOBins" (https://gtfobins.github.io/) en esa página se detalla como explotar la vulnerabilidad para cada caso.

```
www-data@e0acee17a886:/var/www/html$ sudo -l
sudo -l
Matching Defaults entries for www-data on e0acee17a886:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin,
    use_pty

User www-data may run the following commands on e0acee17a886:
    (ALL) NOPASSWD: /usr/bin/taskset
www-data@e0acee17a886:/var/www/html$
```

Imagen $

Archivo con SUID

Esta vulnerabilidad sirve si tenemos configurado un archivo del sistema con SUID, para encontrar este archivo basta con ejecutar el comando "**find / -perm /6000 2>/dev/null**" (*Imagen $*). No todos los archivos que aparecen aquí son vulnerables, habrá que buscar cual está en la lista de "GTFOBins" (https://gtfobins.github.io/).

Para explotarlo habrá que seguir los datos de la página de "GTFOBins" para cada caso.

```
www-data@1ae404b370c2:/var/www/html$ find / -perm /6000 2>/dev/null
/run/postgresql
/usr/bin/expiry
/usr/bin/chage
/usr/bin/su
/usr/bin/mount
/usr/bin/passwd
/usr/bin/umount
/usr/bin/newgrp
/usr/bin/wall
/usr/bin/chfn
/usr/bin/chsh
/usr/bin/gpasswd
/usr/bin/flock
/usr/bin/dotlockfile
/usr/bin/crontab
/usr/bin/sudo
/usr/sbin/unix_chkpwd
/usr/sbin/exim4
/var/log/exim4
/var/mail
/var/local
www-data@1ae404b370c2:/var/www/html$
```

Imagen $

6. CONCLUSIONES

7. REFERENCIAS

GitHub del proyecto: https://github.com/ibantxu12/SSHKey_openldap

https://gtfobins.github.io/