Trabajo Práctico
Desbordamiento de buffers

R-222 Arquitectura del Computador

Introducción

El objetivo de este trabajo es introducir las vulnerabilidades de desbordamiento de buffers en el contexto de un servidor web sencillo. El alumno buscará vulnerabilidades en su código, escribirá exploits para utilizarse contra este servidor y finalmente modificará el código fuente para evitar estos y otros problemas de seguridad en el servidor web. El material de este trabajo es derivado del primer laboratorio de la materia "Computer Systems Security" dictada en el MIT durante 2009, disponible aquí.

Requisitos

Antes de comenzar, el alumno deberá descargar y probar la máquina virtual (VM) donde se realizará La máquina virtual posee un Ubuntu 9.04 de 32 bits, con el código fuente necesario para este trabajo práctica y puede descargarse aquí:

Dos alternativas para utilizar esta VM fueron testeadas: Qemu y VMWare. Recomendamos la primera, por ser software libre y funcionar razonablemente rápido aún sin módulos de kernel.

Una vez booteada la VM, se deberá acceder utilizando algún usuario. La información de acceso para cada usuario se resume en la siguiente tabla:

Usuario	Password
root	6893
user	6893
httpd	6893

Para compilar el programa que se utilizará, se acceder utilizando el usuario httpd y deben seguir los siguientes pasos:

```
httpd@vm-6893:~$ cd /home/httpd/lab1
httpd@vm-6893:~/lab1$ make
gcc httpd.c -c -o httpd.o -m32 -g -std=c99 -fno-stack-protector -Wall -Werror -D_GNU_SOURCE=1
gcc httpd.o -o httpd -m32
cp httpd httpd-exstack
execstack -s httpd-exstack
cp httpd httpd-nxstack
execstack -c httpd-nxstack
gcc shellcode.S -c -o shellcode.o -m32 -g -std=c99 -fno-stack-protector -Wall -Werror -D_GNU_SOURCE=1
```

```
shellcode.S: Assembler messages:
shellcode.S:18: Warning: using `%al' instead of `%eax' due to `b' suffix
objcopy -S -O binary -j .text shellcode.o shellcode.bin
httpd@vm-6893:~/lab1$
```

Una vez compilado el programa, se debe ejecutarse utilizando el script clean-env.sh. Este script permite realizar pruebas sobre la memoria de manera predecible, ya que los valores del stack y el código ejecutable pueden cambiar entre distintas ejecuciones. Para ejecutar el servidor HTTP que se explotará, utilice el siguiente comando:

```
httpd@vm-6893:~/lab1$ ./clean-env.sh ./httpd-exstack 8080 . &
```

Todos los resultados producidos por el alumno deben funcionar utilizando *clean-env.sh*.

Para el desarrollo de exploits o testeos en general, se provee un script de Python llamado *exploit-template.py*. Dicho script facilita los testeos enviando información al servidor de manera sistemática. La utilización de este script es la siguiente:

```
httpd@vm-6893:~/lab1$ ./exploit-template.py localhost 8080
HTTP request:
GET / HTTP/1.0

...
httpd@vm-6893:~/lab1$
```

El uso de este script para las pruebas y los exploits es opcional. El alumno podrá construir sus propios programas lograrlos si lo desea.

Antes de empezar con este trabajo práctico, lea en detalle el artículo de Aleph One, "Smashing the Stack for Fun and Profit", disponible en la sección de bibliografía. Dicho artículo contiene un completo tutorial para la creación de exploits de desbordamiento de buffers.

Metodología

Para realizar el trabajo el alumno debe resolver los siguientes puntos:

Parte 1: Encontrando desbordamientos de buffer

- Estudie el código fuente del servidor web en *httpd.c* y encuentre tantas instancias de código vulnerable a corrupción de memoria como pueda. Escriba cada una de dichas instancias describiendo la estructura de la entrada (datos por HTTP) y el fallo que se generaría. Una copia del código fuente se encuentra aquí, para su visualización sin utilizar la máquina virtual.
- Escriba un exploit que utilize cada vulnerabilidad descripta en el punto anterior. No es necesario ejecutar código ni hacer nada específico por ahora, sólo generar una corrupción de memoria (puede verificar esto con gdb o simplemente observando qué el servidor web termina anormalmente).
  Puede utilizar *exploit-template.py* si lo desea. Por cada vulnerabilidad, incluya el programa que corrompe la memoria. Si no puede encontrar una forma de corromper la memoria para una vulnerabilidad, explique porqué.

Parte 2: Inyectando código

- Modifique sus exploits para cada vulnerabilidad de desbordamiento del ejercicio anterior para tomar el control del servidor web y borrar el archivo */home/httpd/grades.txt*
  Si cree que es imposible modificar el flujo del programa en alguna vulnerabilidad encontrada, explique porqué.

  Sugerencias:

  - Obtenga el control del program counter. Para esto diagrame la disposición del stack que usted espera que el programa tenga en el momento de desbordar un buffer concreto. No olvide incluir este y otros diagramas que haya utilizado durante este trabajo en el informe del mismo.
  - Modifique el shellcode de ejemplo (disponible en */home/httpd/lab1/shellcode.S* junto con su *Makefile*) para generar un shellcode que borre el archivo del servidor.
  - Luego de haber encontrado una forma confiable de controlar el flujo del programa, encuentre una dirección adecuada para colocar el código. Puede utilizar gdb para verificar que el código se encuentra donde debe estar.

Parte 3: Solucionando las vulnerabilidades

- Para cada vulnerabilidad distinta encontrada en los puntos anteriores, modifique el código fuente del servidor HTTP para evitar los desbordamientos. Si por alguna razón, no puede encontrar la forma de evitarlo, detalle las razones por las cúales piensa que es así.

Características adicionales

El alumno puede extender el trabajo (opcionalmente) con las siguientes mejoras:

- Si luego de utilizar una vulnerabilidad, el programa vulnerable termina anormalmente (violación de segmento, desreferencia de puntero nulo, etc), el administrador del sistema puede notar actividad sospechosa: Modifique el exploit para hacer qué el programa continue funcionando con normalidad o finalice sin errores.

- En los sistemas operativos modernos, la memoria de stack no puede utilizarse para ejecutar código. Diseñe exploits que funcionen para el binario httpd-nxstack de manera que no utilizar el stack para ejecutar el código, sino con un sucio truco llamado return-to-libc.

- Otra protección implementada por varios sistemas operativos para evitar estos ataques es la denominada ASLR por la cuál los punteros en el stack resultan distinto cada vez que se ejecuta el programa. Describa esta protección así como las técnicas se pueden utilizar para seguir ejecutando exploits de manera confiable.

Entrega del Trabajo

El trabajo será evaluado por la cátedra mediante una presentación en computadora. El alumno debe entregar un informe de al menos dos páginas incluyendo datos académicos (integrantes del grupo, legajos, fechas) y reportando problemas y soluciones encontradas durante la realización del trabajo y posibles extensiones al mismo.

Material y Referencias

Smashing the Stack for Fun and Profit
Artículo sobre return-to-libc
Artículo sobre ASLR