Software Exploitation

OVERVIEW

The process of software exploitation requires a solid knowledge of reverse engineering. As such, it is a great way to practice one's reverse engineering techniques.

ASSIGNMENT

This project is comprised of multiple parts that will each result in a working exploit. For the first two parts, exploits may initially be coded in whatever language you choose so long as the final exploit is coded in Ruby as a Metasploit plugin. The third part may be coded in any language. Make sure to use the latest development version of Metasploit (download with svn, TortoiseSVN is a good svn client for Windows). Simple Python (version 2.7) programs are provided as examples to show how the server programs operate normally.

Remember, you are exploiting x86. Use a nop sled of all 0x90 when developing your exploits (as it is very easy to identify on the stack).

You should disable ASLR, DEP and stack protections. If you are compiling code for Windows, use the following commands to disable GS stack protections, ASLR, and DEP for your compiled executable:

```
cl.exe /GS- vuln_server.c /link wsock32.lib

editbin.exe /NXCOMPAT:NO /DYNAMICBASE:NO vuln_server.exe
```

If you are compiling your code for Linux, make sure to compile it as follows:

```
gcc -z execstack -fno-stack-protector -o vuln_server.o vuln_server.c
```

This will disable DEP and stack protections. Before running your code in Linux, disable ASLR by the following command (**you must do this even if you decide not to compile your binaries from source!**):

```
sudo echo 0 > /proc/sys/kernel/randomize_va_space
```

Part 1: Arbitrary Remote Code Execution via Simple Stack Overflow (in Windows)

Server source code is provided. Analyze the source code for vulnerabilities. Create a Metasploit exploit module that exploits this vulnerability. Complete the following:

1. What is the address of the return address?
2. What is the address of your shellcode?
3. Describe the layout of your overflow string.

Part 2: Arbitrary Remote Code Execution via Simple Stack Overflow (in Linux)

Server source code is provided. Analyze the source code for vulnerabilities. Create a Metasploit exploit module that exploits this vulnerability. Complete the following:

1. What is the address of the return address?
2. What is the address of your shellcode?
3. Describe the layout of your overflow string.

Part 3: Authentication Bypass via Formatting String Attack

Often, an exploit can use code within the program it is exploiting to achieve its means. Create a formatting string attack to alter program variables so that a login is successful.

Server source code is provided. Analyze the source code for vulnerabilities. Create a custom client program (in whichever programming language you choose) to exploit the server and control the behavior of the server such that it grants access during the authentication procedure. Complete the following:

1. What program variable does your attack target?
2. What is address of that variable?
3. How many dwords (4 bytes) must your formatting string consume to make it to the beginning of the string?
4. How many dwords (4 bytes) must your formatting string consume to make it to the exploit supplied address?
5. Describe your format string.

Part 4: Arbitrary Remote Code Execution via Formatting String Attack

This will be the most difficult exploit (albeit not a very difficult one as far as exploits go). Devise a format string to gain arbitrary remote code execution. Use the integer overflow method discussed in class to generate the correct overwrite value.

Complete the following:

1. What is the address of the return address?
2. What is the address of your shellcode?
3. How many dwords (4 bytes) must your formatting string consume to make it to the beginning of the string?
4. How many dwords (4 bytes) must your formatting string consume to make it to the exploit supplied address?
5. Describe your format string.