Fall 2021

Section 1: TTh 3:30pm - 4:45pm - JFSB B092 (changed from MARB 130)

Project 9: Buffer Overflow

Objectives

- Conduct simple buffer overflow attacks against a vulnerable program from the command line

Files

You will need to download sample3.tar for this lab. You can untar this file with:

```
tar -xvf sample3.tar
```

VM

The CS department has prepared a VM for each of you to be able to use on the lab machines or your own machine. You need to be connected to the CS network to download the VM. Alternatively, you can setup your own Linux VM.

1. download VM, under the CS Linux Virtual Machine section, getting the smaller Ubuntu 16.04 VM. Run the VM using VMWare or VirtualBox. Log in using the Username: cs-student Password: H@ckTheWorld!
2. In order to compile your program run the following commands in the terminal:
   - sudo apt-get update
   - sudo apt-get install libx32gcc-4.8-dev
   - sudo apt-get install libc6-dev-i386 (this resolves the -m32 compile error)
   - sudo apt-get install build-essential
   - sudo sysctl -w kernel.randomize_va_space=0 (this turns off address space randomization)
   - sudo apt-get install execstack (this allows the stack to be executable)

IMPORTANT: Run the file `checkstack x`, which will print out a stack address and fail. Run it several times and verify that the stack address is the same each time you run it. If it changes, the stack randomization is turned on and if will be harder to complete the attack. Turn off stack randomization before continuing.

Compiler Flags

It is recommended you use the supplied binaries but if you need/want to recompile, use the following compiler flags.

Compiler options must be used to turn off buffer overflow defenses.

Use the following form of compiler options to compile example C programs:

```
gcc -g -m32 -fno-pie -fno-stack-protector -z execstack -o [executable file] [source file]
```

Here is how you compile the `auth_overflow3.c` program:

```
gcc -g -m32 -fno-pie -fno-stack-protector -z execstack -o auth_overflow3 auth_overflow3.c
```

Section A

Goal: Gain access without a valid password.

1. Study the `auth_overflow1.c` program and see the valid passwords.
2. Try different passwords to see if you can cause unexpected behavior (e.g. vary the length - to at least 35 characters).

   - gain access without a valid password
   - gain access without a valid password, then program crashes
   - program crashes without gaining access
3. Use the debugger and examine the stack to understand why the errors occur. Be able to explain how the stack works, including `ebp`, `esp`, `return address`, and `local variables`.

Note, changing the code as shown in `auth_overflow2.c` doesn't fix the problem. This compiler places local variables on the stack in the same order.

For this section, turn in a paragraph explaining what passwords worked for each case and why. Include a printout of the stack with labels showing where the saved return address and local variables are located. You can use `x/16xw $sp` to display the stack in gdb.

Section B

Goal: Use the debugger to obtain access by overwriting the return address.

1. Study `auth_overflow3.c`, run `authoverflow3` to test it.
2. This program no longer provides access for an invalid password.
3. Use the debugger to force the program to grant access without a valid password.
   - overwrite the return address on the stack
   - use `disassemble main` to determine a new return address granting access

For this section, turn in a paragraph explaining how you wrote a new return address on the stack. Include a screenshot of you doing this and then getting access.

Section C

Goal: Gain access using only the command line.

1. Now cause the program to grant access without using the debugger.
2. Input data on the command line so that access is granted.
   - Overwrite return address on the stack
   - Program will crash after access is granted. That is ok. Damage is done.

For this section, turn in a screenshot showing you doing this.

Section D

Goal: Inject shellcode on the stack and execute it.

1. Compile and run `shellcode5.c` to verify it works on your platform. You can run the `shellcode5` binary to see if it invokes a new shell. Your command prompt should change if it runs successfully.
   - The binary shellcode is also included in shellcode5.bin.
   - This is the binary data you must place on the stack.
   - You can add `cat shellcode5.bin` on the command line to insert this in the pw parameter.

   ```
   ./auth_overflow3 `perl -e 'print "cougars" . "\x90" x 200'``cat shellcode5.bin`
   ```
2. Now input the shellcode and cause it to be executed.
   - You may need to use a NOP sled. See an example in shellcode5nop.bin
   - Use the debugger to estimate the address used to overwrite the return address.
   - You may need to use Perl to help you enter the binary data as a string.

For this section, turn in a short paragraph explaining what you did and why it worked. Include a printout of the stack with labels showing how your shellcode and NOP sled are placed. Include a screenshot showing this working.

(clarification - use the debugger to help you estimate where the return address goes, then without the debugger at all, successfully exploit from the command line, and once you have that working, bring it back up in the debugger to generate a printout to show the stack with labels, colors etc...)

How to input shellcode on the command line using perl

Use Perl to generate strings (with repeated patterns).

```
perl -e 'print "AAAAAAAA"'
perl -e 'print "AAAA" . "BBBB"'
perl -e 'print "AAAA" x 4'
perl -e 'print "\xf0\xd0\xff\xff"'   (to enter address xffffd0f0, notice address backwards - little endian)
perl -e 'print "\xf0\xd0\xff\xff"x20'    (repeat address 20 times)
perl -e 'print "\x90" x 200'                   (NOP repeated 200 times)
```

Do not include `\x00` in an address since it will be treated as a string terminator

Example of how to enter a password argument on the command line using Perl -- surround with left tick to execute the Perl and provide the argument:

```
./auth_overflow3 `perl -e 'print "this is the place"'`
```

Submission

Submit a PDF of your report on Learning Suite, including all the things asked for in each section.