Buffer Overflow

实验主题：Buffer Overflow

助教：宋宇杰，吕臣臣

主要内容

一、网络安全实验介绍

二、实验注意内容

三、实验提交要求

一、网络安全实验介绍（1/6）

1. 实验目的： buffer overflow 漏洞利用实践
2. 实验内容： 编写exploits攻击漏洞程序
3. 实验结果： 获取具有root权限的shell
4. 实验环境： Ubuntu

已有虚拟机的同学可以使用自己原有的，没有的同学可以下载下面的系统镜像。

Ubuntu16.04镜像： http://mirrors.aliyun.com/ubuntu-releases/16.04/

尽量使用自己的实验环境，如果虚拟机是从其他同学拷贝而来也请新建自己的用户。

一、网络安全实验介绍（2/6）

sudo apt-get install prelink

Vulnerables 文件夹编译、安装说明
/home/user/proj1/vulnerables（漏洞程序）
1. make
2. sudo make install

exploits 文件夹编译说明
/home/user/proj1/exploits（攻击程序）
1.make

一、网络安全实验介绍（3/6）

Vulnerables文件夹编译，安装结果

```
user@network_security:~/proj1/vulnerables$ make
gcc -ggdb -m32 -g -std=c99 -D_GNU_SOURCE -fno-stack-protector -mpreferred-stack-boundary=2 -Wno-form
at-security  -m32  vul1.c   -o vul1
gcc -ggdb -m32 -g -std=c99 -D_GNU_SOURCE -fno-stack-protector -mpreferred-stack-boundary=2 -Wno-form
at-security  -m32  vul2.c   -o vul2
gcc -ggdb -m32 -g -std=c99 -D_GNU_SOURCE -fno-stack-protector -mpreferred-stack-boundary=2 -Wno-form
at-security  -m32  vul3.c   -o vul3
gcc vul4.c -c -o vul4.o -ggdb -m32 -g -std=c99 -D_GNU_SOURCE -fno-stack-protector -mpreferred-stack-
boundary=2 -Wno-format-security
gcc tmalloc.c -c -o tmalloc.o -ggdb -m32 -g -std=c99 -D_GNU_SOURCE -fno-stack-protector -mpreferred-
stack-boundary=2 -Wno-format-security
gcc -m32  vul4.o tmalloc.o   -o vul4
gcc -ggdb -m32 -g -std=c99 -D_GNU_SOURCE -fno-stack-protector -mpreferred-stack-boundary=2 -Wno-form
at-security  -m32  vul5.c   -o vul5
gcc -ggdb -m32 -g -std=c99 -D_GNU_SOURCE -fno-stack-protector -mpreferred-stack-boundary=2 -Wno-form
at-security  -m32  vul6.c   -o vul6
gcc extra-credit.c -c -o extra-credit.o -fstack-protector-all -ggdb -m32 -g -std=c99 -D_GNU_SOURCE
gcc -m32  extra-credit.o   -o extra-credit
execstack -s vul1 vul2 vul3 vul4 vul5 vul6 extra-credit
user@network_security:~/proj1/vulnerables$ ls
extra-credit    Makefile   tmalloc.o  vul2    vul3.c  vul4.o  vul6
extra-credit.c  tmalloc.c  vul1       vul2.c  vul4    vul5    vul6.c
extra-credit.o  tmalloc.h  vul1.c     vul3    vul4.c  vul5.c
user@network_security:~/proj1/vulnerables$ _
```

```
user@network_security:~/proj1/vulnerables$ sudo make install
sudo: unable to resolve host network_security: Connection refused
[sudo] password for user:
gcc -m32  vul4.o tmalloc.o   -o vul4
gcc -m32  extra-credit.o   -o extra-credit
execstack -s vul1 vul2 vul3 vul4 vul5 vul6 extra-credit
install -o root -t /tmp vul1 vul2 vul3 vul4 vul5 vul6 extra-credit
chmod 4755 /tmp/vul*
user@network_security:~/proj1/vulnerables$ ls /tmp/
extra-credit                                                                    vul1  vul4
systemd-private-194c144ce2f446539bde5907e1d50e2d-systemd-timesyncd.service-qAAp2P  vul2  vul5
vmware-root                                                                     vul3  vul6
user@network_security:~/proj1/vulnerables$
```

一、网络安全实验介绍（4/6）

exploits文件夹编译结果

```
user@network_security:~/proj1/exploits$ ls
exploit1.c  exploit3.c  exploit5.c  extra-credit.py  run-shellcode.c  shellcode.S
exploit2.c  exploit4.c  exploit6.c  Makefile         shellcode.h
user@network_security:~/proj1/exploits$ make
gcc -ggdb -m32   -c -o exploit1.o exploit1.c
gcc -m32  exploit1.o   -o exploit1
gcc -ggdb -m32   -c -o exploit2.o exploit2.c
gcc -m32  exploit2.o   -o exploit2
gcc -ggdb -m32   -c -o exploit3.o exploit3.c
gcc -m32  exploit3.o   -o exploit3
gcc -ggdb -m32   -c -o exploit4.o exploit4.c
gcc -m32  exploit4.o   -o exploit4
gcc -ggdb -m32   -c -o exploit5.o exploit5.c
gcc -m32  exploit5.o   -o exploit5
gcc -ggdb -m32   -c -o exploit6.o exploit6.c
gcc -m32  exploit6.o   -o exploit6
gcc -ggdb -m32   -c -o run-shellcode.o run-shellcode.c
gcc -m32  run-shellcode.o   -o run-shellcode
gcc -m32 -c -o shellcode.o shellcode.S
objcopy -S -O binary -j .text shellcode.o shellcode.bin
rm shellcode.o
user@network_security:~/proj1/exploits$ ls
exploit1    exploit2.c  exploit3.o  exploit5    exploit6.c      run-shellcode    shellcode.h
exploit1.c  exploit2.o  exploit4    exploit5.c  exploit6.o      run-shellcode.c  shellcode.S
exploit1.o  exploit3    exploit4.c  exploit5.o  extra-credit.py  run-shellcode.o
exploit2    exploit3.c  exploit4.o  exploit6    Makefile        shellcode.bin
user@network_security:~/proj1/exploits$ _
```

一、网络安全实验介绍（5/6）

exploits编写说明

```c
#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include "shellcode.h"

#define TARGET "/tmp/vul1"

int main(void)
{
  char *args[] = { TARGET, "hi there", NULL };
  char *env[] = { NULL };

  execve(TARGET, args, env);
  fprintf(stderr, "execve failed.\n");

  return 0;
}
~
~
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>

int bar(char *arg, char *out)
{
  strcpy(out, arg);
  return 0;
}

void foo(char *argv[])
{
  char buf[256];
  bar(argv[1], buf);
}

int main(int argc, char *argv[])
{
  if (argc != 2)
    {
      fprintf(stderr, "target1: argc != 2\n");
      exit(EXIT_FAILURE);
    }
  setuid(0);
  foo(argv);
  return 0;
}
~
```

```c
#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include "shellcode.h"

#define TARGET "/tmp/vul1"

int main(void)
{
  char payload[]="";
  char *args[] = { TARGET, payload, NULL };
  char *env[] = { NULL };

  execve(TARGET, args, env);
  fprintf(stderr, "execve failed.\n");

  return 0;
}
~
```

网络安全实验介绍（6/6）

实验结果：
获得一个具有
root权限的shell

```
bubbles@a22be5d79962:/share/proj1/exploits$ ./exploit1
# whoami
root
#
```

二、实验注意内容（1/4)

1. shellcode模板
可参考：http://shell-storm.org/shellcode/
2. 机器码中存在/x00字节（即NULL（0）字符），所有输入函数只要检测到NULL字符就会返回，所以避免出现/x00字节，NULL只能够出现在shellcode的结尾处。
尝试使用xor命令(xor eax,eax)

```
jmp xxx
pop xxx
xxxxxxxx
call pop address
.string
```

实验注意内容（2/4)

3. 编译命令，编译器关闭数据溢出保护NX(DEP)选项。

gcc -z execstack -o test test.c

4. 关闭堆栈溢出保护，编译时禁用SSP机制（ Stack Smashing Protector ）

-fno-stack-protector

5.关闭ASLR(地址随机化)

echo 0 > /proc/sys/kernel/randomize_va_space

根据环境中gcc版本的不同PIE保护的默认设置也不同，可使用-no-pie自行关闭

实验注意内容（3/4)

gdb 常用命令：

1. **set disassembly-flavor intel**
2. **disassemble**
3. **info register**
4. **break *0x80484e3/strcpy**
5. **x/8wx 0x80484e3**
6. **catch exec**
7. **bt**
8. **gdb ./exploit1**
9. **i r $ebp/$esp...**

实验注意内容（4/4）

建议安装gdb-peda插件

```
git clone https://github.com/longld/peda.git ~/peda
echo "source ~/peda/peda.py" >> ~/.gdbinit
```

```
gdb –e exploit1 –s /tmp/vul1
```

注意转换字节序

find 0xd231c931

实验提交要求（1/3）

提交时间

- **2021-5-02 24:00 前**

提交形式

- **源代码+说明文档**
- **总文件夹命名方式：学号-姓名-project1**
- **电子版：xxx@qq.com**

实验提交要求 （2/3）

源代码提交要求

源代码文件夹命名要求：exploits

提交内容

1. exploit[1-6].c
2. Makefile

实验提交要求（3/3）

说明文档提交要求

命名要求：学号-姓名-project1.doc

包含内容：1. vul[1-6]的简单描述
2. shellcode（构造过程）
3. exploit[1-6]攻击方式描述
（攻击原理，payload构造方式，攻击过程描述）