ROP (Return Oriented Programming)

Advanced Stack Overflows

Stuart Nevans Locke

Overview

- Tools
  - Pwntools
- Review of Stack Overflows
  - DEP
- ROP
  - Demo
- Ret2libc
  - Demo
- Stack Pivoting
- Tools
  - one_gadget

Pwntools

Python package for helping with binary exploitation

```
pip install pwntools
```

```
from pwn import *

pe = process(‘./binary’) #run the binary

pe.sendline(‘A line’)       #send ‘A line’ to the binary

pe.sendline(p64(0xFFFFFF)) #Sends a 64 bit pointer in string format

address = u64(pe.recvuntil(‘is the address’)) #Read a string address as an integer
```

Pwntools (Cont .)

pe = process(“/path/to/binary”)

pe.clean() #Essentially receives all messages and cleans that buffer

gdb.attach(pe) #Attach gdb to process

elf = ELF(‘./path/to/file)

print elf.symbols[callme’] #Gets the offset of callme

print elf.search("/bin/sh").next() #Prints the offset of the string /bin/sh in the file

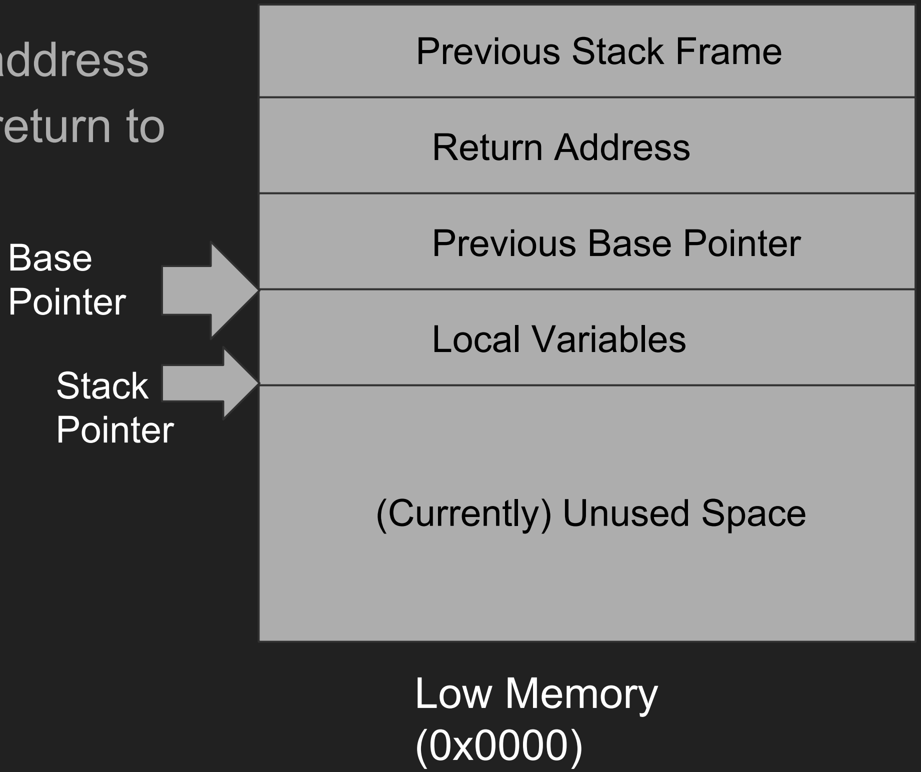
PwnTools Demo

stnevans.me/binex/2/pwntools/demo

Follow the instructions given by the binary

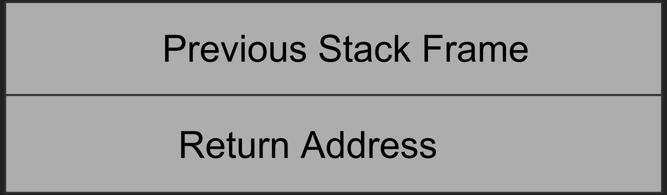
Stack Overflows (Review)

- We read too much onto the stack
- Exploited by overwriting the return address
- We put shellcode on the stack and return to it
- Mitigations
  - DEP
    - Data Execution Prevention
    - Let's get around this



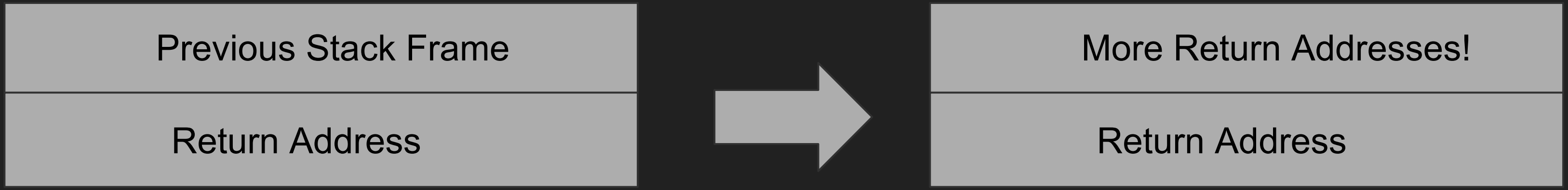
ROP (Return Oriented Programming)

Right before returning, the stack looks like



- What exactly does ret do?
  - ret
    - Basically pops into the instruction pointer
    - pop rip
- What if we overwrite the return address to return to a ret instruction?
  - ret
    - Our first ret, we go to another ret instruction
  - ret
    - This returns to whatever was above the previous return address on the stack

ROP (Cont.)



Example:

0x1234: ret

0x1257: xor rax, rax

0x1258: ret

If we return to 0x1257, we can set rax to zero, and continue returning to more addresses we control

ROP (Cont.)

- Using pieces of the victim binary against itself
- These pieces of code we return to are called “gadgets”
- Returning to multiple gadgets makes up a “ROP chain”
- Our goal: make rax==1,rbx==0x2127

```
0x123: xor rax, rax        0x303: pop rbx

0x124: ret                 0x304: ret

0x291: add rax, 1

0x292: ret
```

Our ROP chain:

0x123	xor rax,rax
0x291	rax+=1
0x303	pop rbx
0x2127	data for pop rbx

Initial Return Pointer

Tool - ROPgadget

Used to find ROP gadgets

```
ROPgadget --binary /path/to/binary
```

Outputs the address of gadgets in the binary

EX:

```
0x000000000040060b : pop rdi ; ret

0x0000000000400609 : pop rsi ; pop r15 ; ret

0x0000000000400448 : call rax
```

Demo

stnevans.me/binex/2/easy

ret2libc (ROP)

- Finding ROP gadgets and chaining them together is really annoying
- Especially if you want to do anything interesting
- Much easier, call system from your ROP

```
0x122: push rax

0x123: pop rdi

0x124: ret

0x302: mov rax,0x402

0x303: ret
```

```
0x402: data( ‘/bin/sh )

0x501 <system>: push rbp
```

Our ROP chain:

0x302	rax=0x402 (address of /bin/sh)
0x122	rdi=rax
0x501	call system(rdi)

Misc info:
System is located in libc
To find your libc, run ldd /binary
To get the offset, look at pwntools ELF demo from earlier

Demo

stnevans.me/binex/2/medium

Stack Pivoting

- In all prior examples, we don't worry about how much space we have to ROP
- What if we can only overflow 8 bytes?
  - We can only call one thing with our ROP.
  - Assuming nothing magically gives us a shell in the binary, we're stuck
- Solution:
  - Make rsp point into some bigger buffer we can control
  - Let's assume rax points to some string we can control
  - We want to pivot our stack to point to the buffer.

```
0x123: xchg rax, rsp

0x124: ret
```

If we return to 0x123, we can then put the rest of our rop in the larger buffer we can control.

Tool - one_gadget

- In libc, there are actually multiple ROP gadgets that call system(/bin/sh)
- They do require some prerequisites
- one_gadget prints them and their prerequisites
- usage: one_gadget /path/to/libc

These special gadgets are often called magic gadgets.

Note: Don’t become reliant on this, it acts as a crutch. This is not always a possibility, and sometimes flat out doesn’t work (especially if shell isn’t bash)

Questions?

Next Presentation: ELF Structure/Defeating ASLR