Lec07: Return-oriented programming

Taesoo Kim

Scoreboard

NSA Codebreaker Challenges

University	Task 0	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6
Carnegie Mellon University	11	5	5	2	2	2	1
Lafayette College	3	2	2	1	1	1	1
Georgia Institute of Technology	28	18	15	7	5	3	0
University of Hawaii	18	8	7	4	2	2	0
University of Tulsa	10	5	5	5	2	1	0
Pennsylvania State University	37	11	9	1	1	1	0
Virginia Community College System	11	1	1	1	1	1	0
Lesley University	1	1	1	1	1	1	0
University of Memphis	11	7	6	4	4	0	0
Texas A&M University - College Station	21	10	6	3	1	0	0

Showing 1 to 10 of 368 entries

Previous 1 2 3 4 5 ... 37 Next

Administrivia

- Please submit both 'working exploit' and write-up on time; otherwise, no score.
- Due: Lab07 is out and its due on Oct 19 (two weeks)!
- NSA Codebreaker Challenge → Due: Nov 30
- Oct 13 : A special talk from NSA

Lab06: DEP and ASLR

Name	Points	Release	Deadline	Solved
libbase	20	09-29-2017 00:00:00	10-06-2017 00:00:00	21
moving-target	20	09-29-2017 00:00:00	10-06-2017 00:00:00	21
fmtstr-read	20	09-29-2017 00:00:00	10-06-2017 00:00:00	21
fmtstr-write	20	09-29-2017 00:00:00	10-06-2017 00:00:00	21
fmtstr-digging	20	09-29-2017 00:00:00	10-06-2017 00:00:00	21
brainfxxk	20	09-29-2017 00:00:00	10-06-2017 00:00:00	19
fd-const	20	09-29-2017 00:00:00	10-06-2017 00:00:00	18
fmtstr-heap	20	09-29-2017 00:00:00	10-06-2017 00:00:00	10
profile	20	09-29-2017 00:00:00	10-06-2017 00:00:00	10
mini-sudo	20	09-29-2017 00:00:00	10-06-2017 00:00:00	17

Best Write-ups for Lab06

- libbase: carterchen, shudak3
- moving-target: carterchen, markwis
- fmtstr-read: brian_edmonds, shudak3
- fmtstr-write: poning, carterchen
- fmtstr-digging: N/A, N/A
- brainfxxk: jli850, rohandvora
- fd-const: prengasamy6, nagendra
- fmtstr-heap: carterchen, N/A
- profile: myao42, carterchen
- mini-sudo: markwis, carterchen

Discussion: Lab06

- What's the most "annoying" bug or challenge?
- What's the most "interesting" bug or challenge?
- So, DEP and ASLR are not so effective?

Discussion: libbase

- What do you learn from ./check?

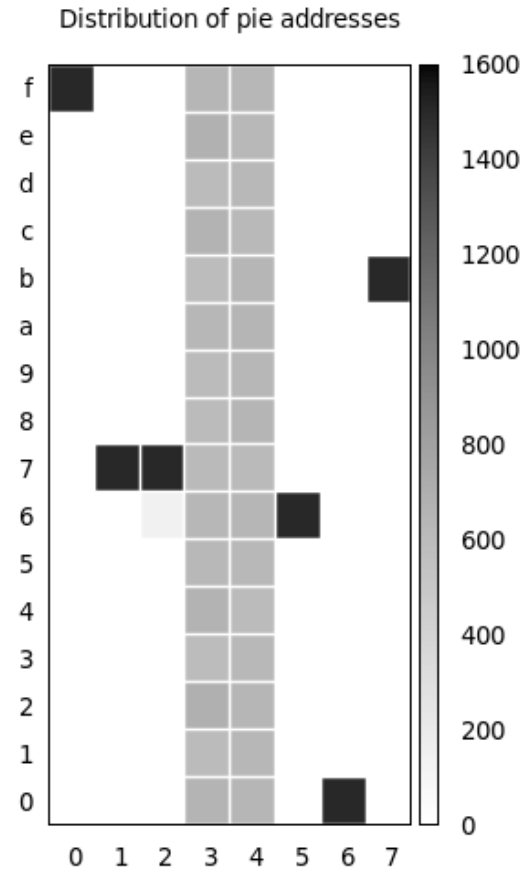
```
$ ./check
stack   : 0xff930aa0
system(): 0xf7521c50
printf(): 0xf7536670

$ ./check
stack   : 0xff930250
system(): 0xf755dc50
printf(): 0xf7572670
```

Discussion: libbase

Discussion: moving-target

- What's "check-aslr.sh" and pie.c?
- How many times should we try to exploit?

Discussion: moving-target



Discussion: fmtstr-read

Discussion: fmtstr-write

How to Prevent fmtstr-*?

How to Prevent fmtstr-*?

- set a max on width (e.g., "%.512x" in XP, "%.622496x" in 2000)
- no direct argument access (i.e., "%N$")
- static (ro) format string
- proposal: push N (#argument) in varargs?
- check all Ns (not skip)

```
$ ./fortify "%2\$d"
*** invalid %N$ use detected ***
```

Discussion: brainfxxk

Discussion: brainfxxk

Discussion: fd-const

- What's the bug?

Discussion: profile

- What's program about?
- What's the bug?

Discussion: profile

```
void edit_all() {
  struct profile p;
  printf("[*] Edit all attributes\n");
  p.name = get_name();
  p.birthday = get_birthday();
  get_phone_number(p.phone_number);
  p.censored = get_censored();
  if (!p.censored)
    p.print = print_phone_number;
  memcpy(&my, &p, sizeof(p));
}
```

Discussion: profile

```
bool get_censored() {
  char buf[SIZE];
  printf("[*] Censored? (y/n)\n");

  while (true) {
    stripped_read(buf, sizeof(buf));
    if (buf[0] == 'y')
      return true;
    else if (buf[0] == 'n')
      return false;
    else
      printf("y/n\n");
  }
}
```

Discussion: profile

```
void print_profile() {
  printf("========== My profile ==========\n");
  printf("Name : %s\n", my.name);
  printf("Birthday : %d-%d-%d\n", my.birthday.year,
                                  my.birthday.month,
                                  my.birthday.day);

  if (my.censored)
    printf("Phone number : CENSORED\n");
  else
    my.print(my.phone_number);
  printf("================================\n");
}
```

Discussion: mini-sudo (CVE-2012-0809)

- What is '-D9' for?

Discussion: mini-sudo (CVE-2012-0809)

```
void sudo_debug(int level, const char *fmt, ...) {
  va_list ap;
  char *fmt2;

  if (level > debug_level) return;

  /* Backet fmt with program name and a newline to make it
     a single write */
  easprintf(&fmt2, "%s: %s\n", getprogname(), fmt);
  va_start(ap, fmt);
  vfprintf(stderr, fmt2, ap);
  va_end(ap);
  efree(fmt2);
}
```

Take-outs from DEP/ASLR?

- Do you think DEP/ASLR make your life more difficult?
- Is still possible to exploit? why?
- Although we can't place shellcode into stack/heap, we can still hijack the control flow of a program in many interesting ways

Discussion: Modern Exploit on ASLR (PIE)

- Leak (or infer) code pointers (so map into library or code)
- Construct ROP (today's topic)
- (although there are a few proposals, such as CFI, to mitigate ROPs)

Today's Tutorial

- In-class tutorial:
  - Ret-to-libc
  - Code pointer leakage / gadget finding
  - First ROP!

Reminder: crackme0x00

```
void start() {
  printf("IOLI Crackme Level 0x00\n");
  printf("Password:");

  char buf[32];
  memset(buf, 0, sizeof(buf));
  read(0, buf, 256);

  if (!strcmp(buf, "250382"))
    printf("Password OK :)\n");
  else
    printf("Invalid Password!\n");
}
```

Reminder: crackme0x00

```
int main(int argc, char *argv[])
{
  setvbuf(stdout, NULL, _IONBF, 0);
  setvbuf(stdin, NULL, _IONBF, 0);

  void *self = dlopen(NULL, RTLD_NOW);
  printf("stack   : %p\n", &argc);
  printf("system(): %p\n", dlsym(self, "system"));
  printf("printf(): %p\n", dlsym(self, "printf"));

  start();

  return 0;
}
```

Ret-to-libc: printf

```
[buf  ]
[.....]
[ra   ] -> printf
[dummy]
[arg1 ] -> "Password OK :)"
```

Ret-to-libc: system

```
[buf  ]
[.....]
[ra   ] -> system
[dummy]
[arg1 ] -> "/bin/sh"
```

Chaining Two Function Calls

```
printf("Password OK:)")
system("/bin/sh")
```

Chaining Two Function Calls

```
[buf      ]
[.....    ]
[old-ra   ] -> 1) printf
[ra       ] ------------------> 2) system
[old-arg1 ] -> 1) "Password OK :)"
[arg1     ] -> "/bin/sh"
```

Chaining N Function Calls

```
[buf      ]
[.....    ]
[old-ra   ] -> 1) printf
[ra       ] ------------------> pop/ret gadget
[old-arg1 ] -> 1) "Password OK :)"
[ra       ] -> 2) system
[ra       ] ------------------> pop/ret gadget
[arg1     ] -> "/bin/sh"
[ra       ] ...
```

Tutorial Goal: Chaining Three Calls

```
printf("Password OK:)")
system("/bin/sh")
exit(0)
```

In-class Tutorial

- Step1: Ret-to-libc
- Step2: Understanding module base
- Step3: First ROP

```
$ ssh YOURID@cyclonus.gtisc.gatech.edu -p 2023
$ ssh YOURID@cyclonus.gtisc.gatech.edu -p 2022
$ ssh YOURID@computron.gtisc.gatech.edu -p 2023
$ ssh YOURID@computron.gtisc.gatech.edu -p 2022

$ cd tut/lab07
$ cat README
```

References

- ROP