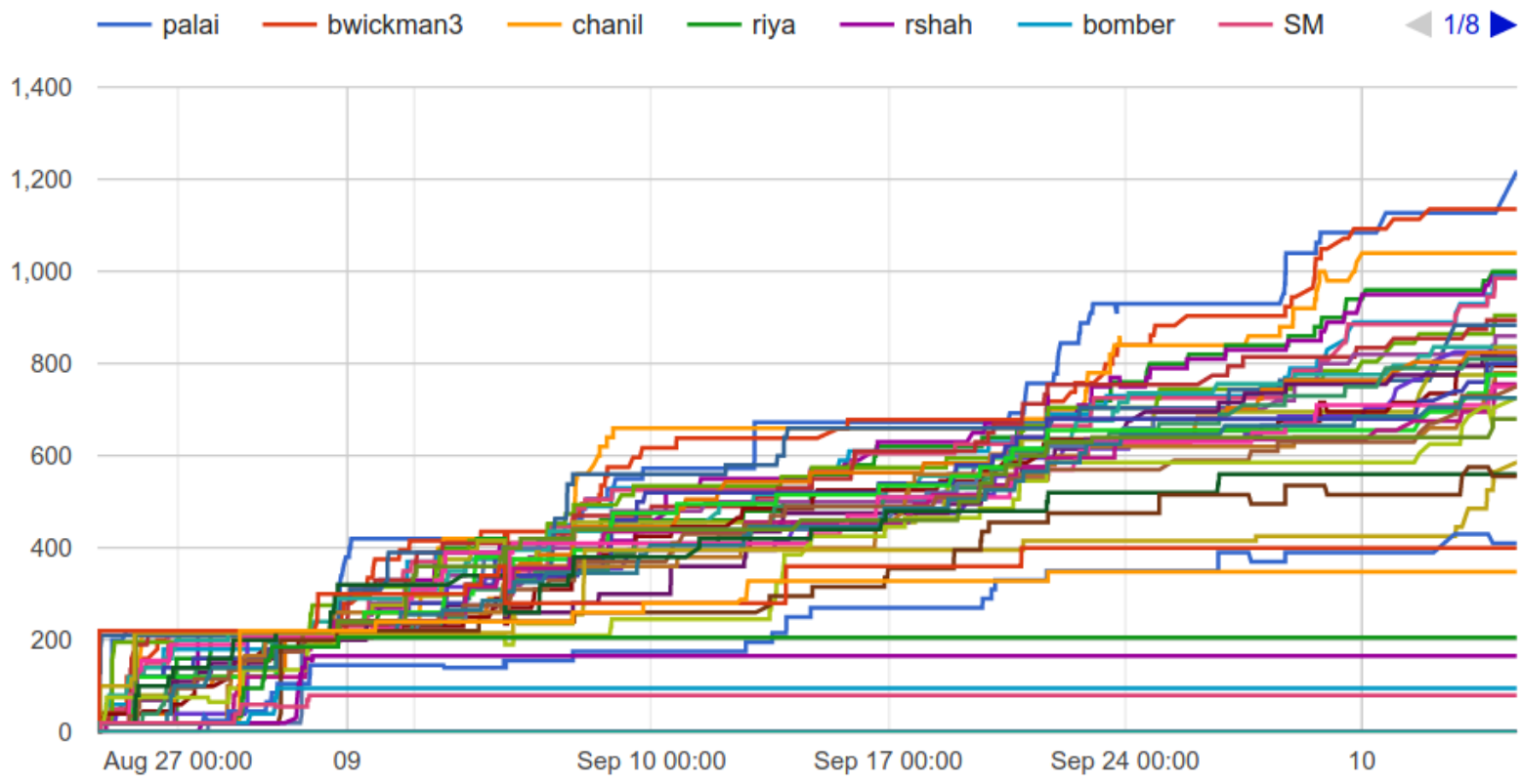
Lec07: Return-oriented programming

Taesoo Kim

Scoreboard


palai
bwickman3
chanil
riya
rshah
bomber
SM
1/8
1,400
1,200
1,000
800
600
400
200
0
Aug 27 00:00
09
Sep 10 00:00
Sep 17 00:00
Sep 24 00:00
10

Administrivia

- Please submit both ‘working exploit’ and write-up on time!
- Due: Lab04 is due on Oct 11
- Due: Lab05 is out and its due on Oct 18 (two weeks)!
- NSA Codebreaker Challenge → Due: Nov 29

Best Write-ups for Lab04

xor	**gkamuzora3, burak**
stackshield	**gkamuzora3, nhicks6**
weak-random	**palai, stong**
gs-random	**stong, riya**
terminator	**seulbae, stong**
assassination	**jwalsh45, nhicks6**
mini-heartbleed	**stong, riya**
pltgot	**nhicks6, stong**
ssp	**palai, nhicks6**
fd	**palai, fsang**

Discussion: Lab04

- What's the most "annoying" bug or challenge?
- What's the most "interesting" bug or challenge?
- So, should we use canary or not?
- So, which one would you like to use?

Take-outs from Stack Canary?

- Stack Canary indirectly protects the “integrity” of RA, funcptr, etc
  - (e.g., exploitation mitigation → NX, canary)
- We better prevent buffer overflows at the first place
  - (e.g., code analysis, better APIs)

Subtle Design Choices for the Stack Canary

- Where to put? (e.g., right above ra? fp? local vars?)
- Which value should I use? (e.g., secrete? random? per exec? per func?)
- How to check its integrity? (e.g., xor? cmp?)
- What to do after you find corrupted? (e.g., crash? report?)

Subtle Design Choices for the Stack Canary

- Where to put? (e.g., right above ra? fp? local vars?)
  - gs-random, terminator
- Which value should I use? (e.g., secrete? random? per exec? per func?)
  - xor, weak-random, gs-random, terminator
- How to check its integrity? (e.g., xor? cmp?)
  - xor
- What to do after you find corrupted? (e.g., crash? report?)
  - ssp, stackshield
- Fundaemtnal limitations → stackshield, assassination, gs-random

Discussion: xor

- How xor canary works?
- What happens if RA is overwritten (or leaked)?
  - RA ^ canary
  - what happens if RA is overwritten?
  - what if we make it random?

Discussion: xor

```
@prologue
pop     %eax
xor     $0x63736265,%eax
push    %eax
```

Discussion: stackshield (safestack)

- How stackshield works? (can you overwrite ra/fp?)
- Compared to xor, what's better?
- Then, could you control its control flow?

Discussion: weak-random

- How weak-random is implemented?
- How did you exploit?
- What if we use a perfect random value (e.g., /dev/random)?

Discussion: gs-random

- Near perfect (Microsoft CL):
  - strong randomness: /dev/random
  - protect fp/ra

Discussion: gs-random

```
void echo(char *msg) {
  char buf[80];

  strcpy(buf, msg);
  capitalize(buf);
  strcpy(msg, buf);
  ...
}
```

Discussion: gs-random (arbitrary overwrite)

Discussion: gs-random

Discussion: terminator

- Why is the terminator canary special?
  - 0x0d000aff: NULL(0x00), CR (0x0d), LF (0x0a) and EOF (0xff)

Discussion: terminator

- What's the vulnerability?

Discussion: terminator (off-by-one)

Discussion: terminator

- How to prevent this vulnerability?

Discussion: assassination

- Near perfect (GCC)
  - random canary
  - protect fp, ra
- What's the bug?
- How to prevent?

Discussion: mini-heartbleed

Discussion: ssp

- What happens if you cause a crash?

Discussion: ssp

Discussion: ssp

Discussion: ssp

Discussion: pltgot

- What was the vulnerability?
- Where to overwrite?
- How to prevent?

Discussion: fd

- Overwriting ‘struct FILE’

```
@libio.h
struct _IO_FILE {
  int _flags;
  ...
  struct _IO_wide_data {
    ...
    const struct _IO_jump_t *_wide_vtable;
  }
}
```

Discussion: fd

- Why need vtable?

Discussion: fd

```
_IO_wfile_jumps (default)
_IO_wfile_jumps_mmap
...

fclose(fp)?
  - _IO_file_close(): close()
  - _IO_file_close_mmap(): munmap() & close()
```

Discussion: fd

- How to prevent this vulnerability?

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

```
$ checksec ./target
 [*] '/home/lab/tut-rop/target'
  Arch:     i386-32-little
  RELRO:    Partial RELRO
  Stack:    No canary found
  NX:       NX enabled
  PIE:      No PIE (0x8048000)
```

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

```
open("/proc/flag", O_RDONLY)
read(3, tmp, 1024)
write(1, tmp, 1024)
```

In-class Tutorial

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

```
$ ssh lab06@computron.gtisc.gatech.edu -p 9006
$ ssh lab06@cyclonus.gtisc.gatech.edu -p 9006
Password: lab06

$ cd tut-rop
$ cat README
```

References

- ROP