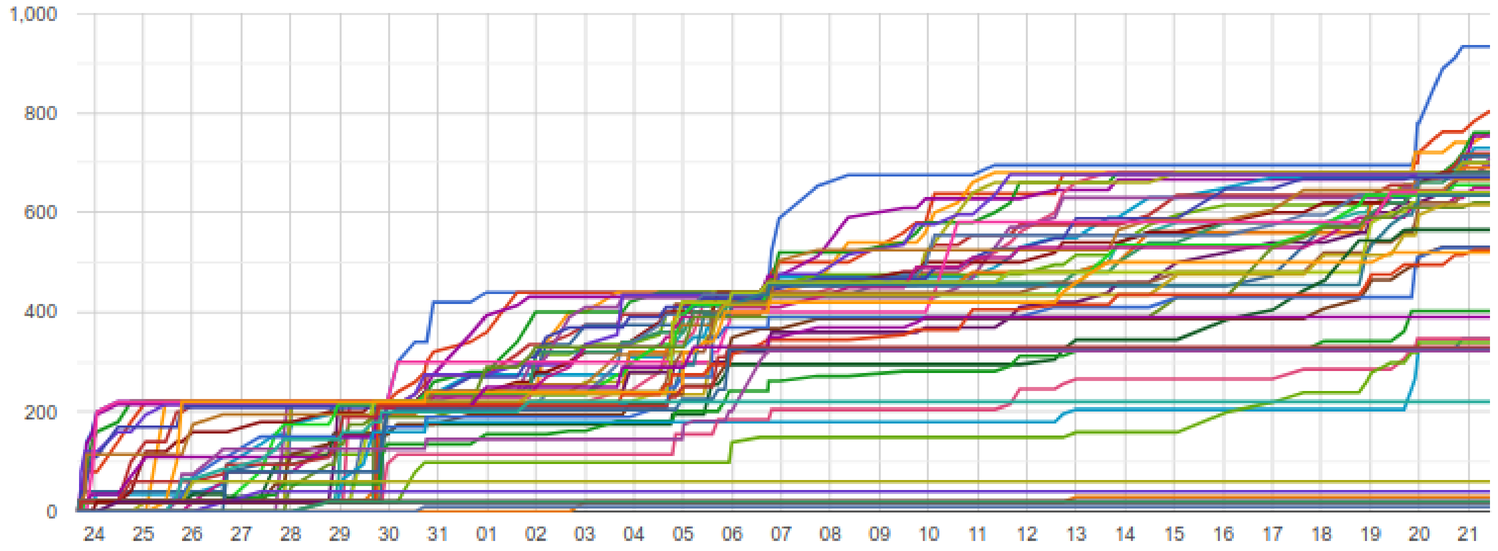
Lec05: Stack Protections

Taesoo Kim

Scoreboard


1,000
800
600
400
200
0
24
25
26
27
28
29
30
31
01
02
03
04
05
06
07
08
09
10
11
12
13
14
15
16
17
18
19
20
21

Administrivia

- Please submit your write-ups on time!
- Please write down your collaborators' names on the write-ups
- Due: Lab04 is out, and its due on Sept 27 at midnight

Best Write-ups for Lab03

simple-bof	mliu366, Megan_Huber
jmp-to-stack	mliu366, leitning
jmp-to-env	leitning, mliu366
frobnicated	viyer43, leitning
argc0	leitning, mliu366
lack-of-four	yiqincai, leitning
jmp-to-where	Joseph_Rice, mliu366
unusual-main	achang66, yiqincai
man-strncpy	yiqincai, meduka
upside-down	yiqincai, meduka

Lab03: Stack Overflow

Discussion: Lab03

- What's the most "annoying" challenge?
- What's the most "interesting" challenge?
- What did you learn in general?

Discussion: Not Yet Motivated?

Discussion: Not Yet Motivated?

Discussion: jmp-to-where

- What's the bug?
- What's special about this challenge?

Discussion: jmp-to-where

- What's your lesson?

Discussion: unusal-main

- What's the bug?
- What's special about this challenge?

Discussion: man-strncpy

- What's the bug?
- What's special about this challenge?

Discussion: man-strncpy

- What's your lesson?
- How to prevent this?

Discussion: man-strncpy (safe usage)

```
char buf[BUFSIZ];
strncpy(buf, input, sizeof(buf) - 1);
buf[sizeof(buf) - 1] = '\0';
```

Discussion: alternative strlcpy()

```
strlcpy(buf, s, sizeof(buf));
```

Discussion: upside-down

```
0x00000000           0x00000000
^                    | arg1
| buf     ||         | ret
| [ ]     ||         | fp
| [ ]     ||         | buf       || ??
| fp      ||         | [ ]       ||
| ret     vv         | [ ]       ||
| arg1               |           vv
| ..                 v
0xFFFFFFFF           0xFFFFFFFF
```

Discussion: upside-down

- More secure? less? in terms of security?
- What if we are not using stack at all? (e.g., stackless python)

Discussion: How to Prevent Stack Overflow?

- Two approaches:
  - Bug prevention
  - Exploitation mitigation
- Protect "integrity" of ra, funcptr, etc (code pointers)
  - (e.g., exploitation mitigation → NX, canary)
- Prevent the buffer overflow at the first place
  - (e.g., code analysis, better APIs)

Today's Tutorial

- In-class tutorial
  - Let's understand the implementation of the stack protector.
  - Let's exploit the (insecurely) protected crackme0x00 to get a flag!

Reminder: crackme0x00

```
$ objdump-intel -d crackme0x00
...
8048448:      lea    eax,[ebp-0x18]
804844b:      mov    DWORD PTR [esp+0x4],eax
804844f:      mov    DWORD PTR [esp],0x804858c
8048456:      call   8048330 <scanf@plt>

           |<=- 0x18-=>|+--- ebp
top                     v
[          [~~~~>  ]   ][fp][ra]
|<=---   0x28  ------=>|
```

Reminder: Exploiting crackme0x00

```
               |<=- 0x18-=>|+--- ebp
top                         v
[              [~~~~>   ]   ][fp][ra]
|<=---    0x28  ------=>|
               AAAABBBB.....GGGGHHHH
```

crackme0x00 in C

```
int main(int argc, char *argv[])
{
  char buf[16];
  printf("IOLI Crackme Level 0x00\n");
  printf("Password:");

  scanf("%s", buf);

  if (!strcmp(buf, "250382"))
    printf("Password OK :)\n");
  else
    printf("Invalid Password!\n");
  return 0;
}
```

By the way, how to fix crackme0x00's bug?

```
scanf("%15s", buf);  // NOTE. 15 not 16
or
scanf("%as", &buf);  // NOTE. char *buf, require a manual fre
```

DEMO: GCC's Stack Protector

- makefile
- compilation options
- diff.sh

Core Idea of Stack Protector

- Use a "canary" value as an indicator of the integrity of fp/ra

```
           |<=- 0x14 ------------=>|+--- ebp
top                                 v
[          [               ][canary][fp][ra][      ....]
|<=---   0x30  -------------------=>|
                            XOXOXO XXXX
                            (corrupted?)
```

Why is it called "Canary"?

Why is it called "Canary"?

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?)

In-class Tutorial

- Step 1: Understanding GCC's Stack Protector
- Step 2: Let's exploit 0xdeadbeef canary!

```
$ ssh lab04@3.223.237.92
Password: <password>

$ cd tut05-ssp
$ cat README
```

References

- Bypassing StackShield