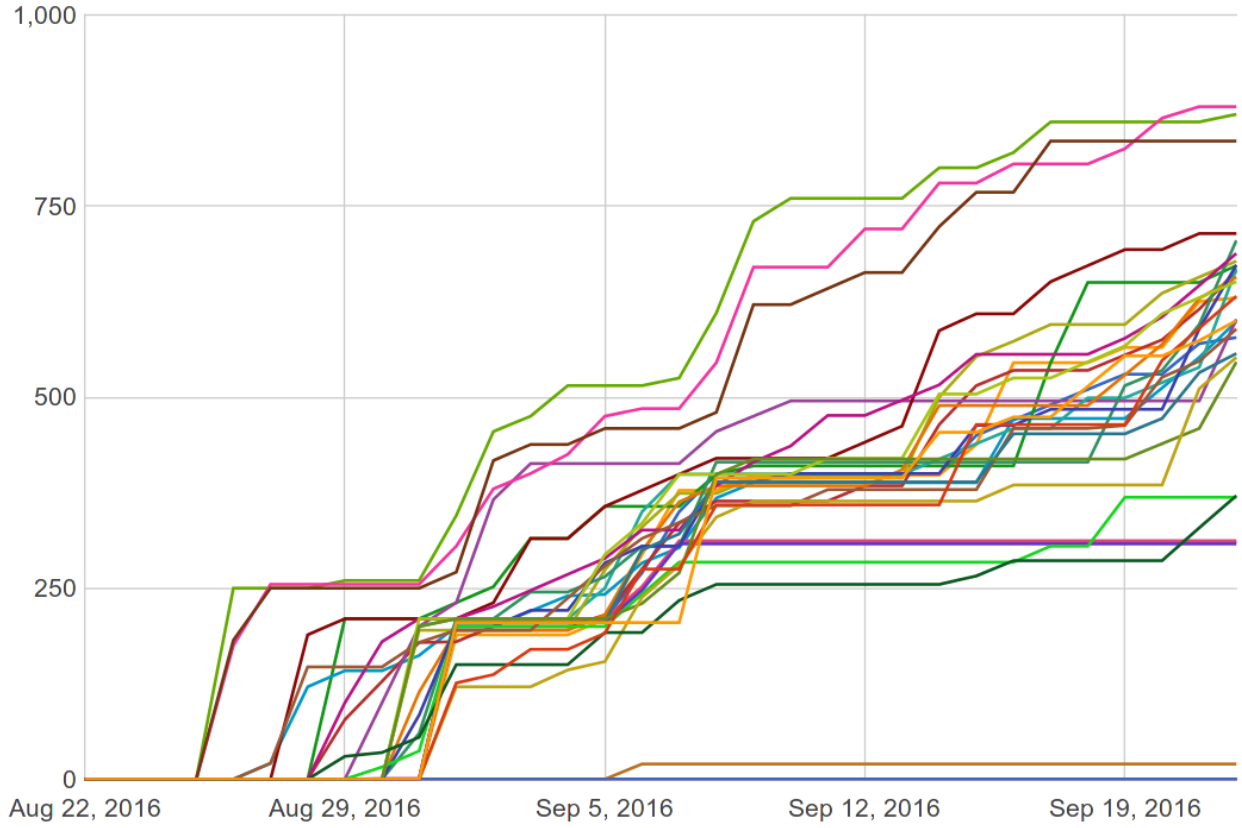
Lec05: Stack Protections

Taesoo Kim

Scoreboard


1,000
750
500
250
0
Aug 22, 2016
Aug 29, 2016
Sep 5, 2016
Sep 12, 2016
Sep 19, 2016

Administrivia

- In total, 10 labs + NSA Challenge (we already completed four!)
- Due: Lab05 is out and its due on Sept 29 at midnight
- NSA Codebreaker Challenge → New due: Dec 1

New! Course Grading (Expectation for A/B)

1. Absolute scoring/grading
2. Expectation:
    - 7.5 on average → A
    - 5.5 on average → B
    - If it's still too high for your goal, please talk to me!
3. Survey: first/second bloods
4. NSA Challenge:
    - Task1: 20, Task2: 20, Task3: 20, Task4: 60 (B), Task5: 60 (A), Task6: 20
    - If you solve Task6, we will bump up your grade!

NSA Codebreaker Challenges

University	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6
Georgia Institute of Technology	43	35	33	26	9	3
Carnegie Mellon University	22	19	11	7	5	2
Dakota State University	53	37	23	17	8	0
Naval Postgraduate School	5	5	4	4	4	0
University of Colorado at Colorado Springs	9	7	5	4	3	0
Rensselaer Polytechnic Institute	5	4	4	3	2	0
University of Tulsa	5	5	3	2	2	0
University of Maryland, Baltimore County	25	21	9	8	1	0
Purdue University	10	9	6	5	1	0
University of Maryland, College Park	5	4	3	2	1	0

Showing 1 to 10 of 303 entries

Previous 1 2 3 4 5 ... 31 Next

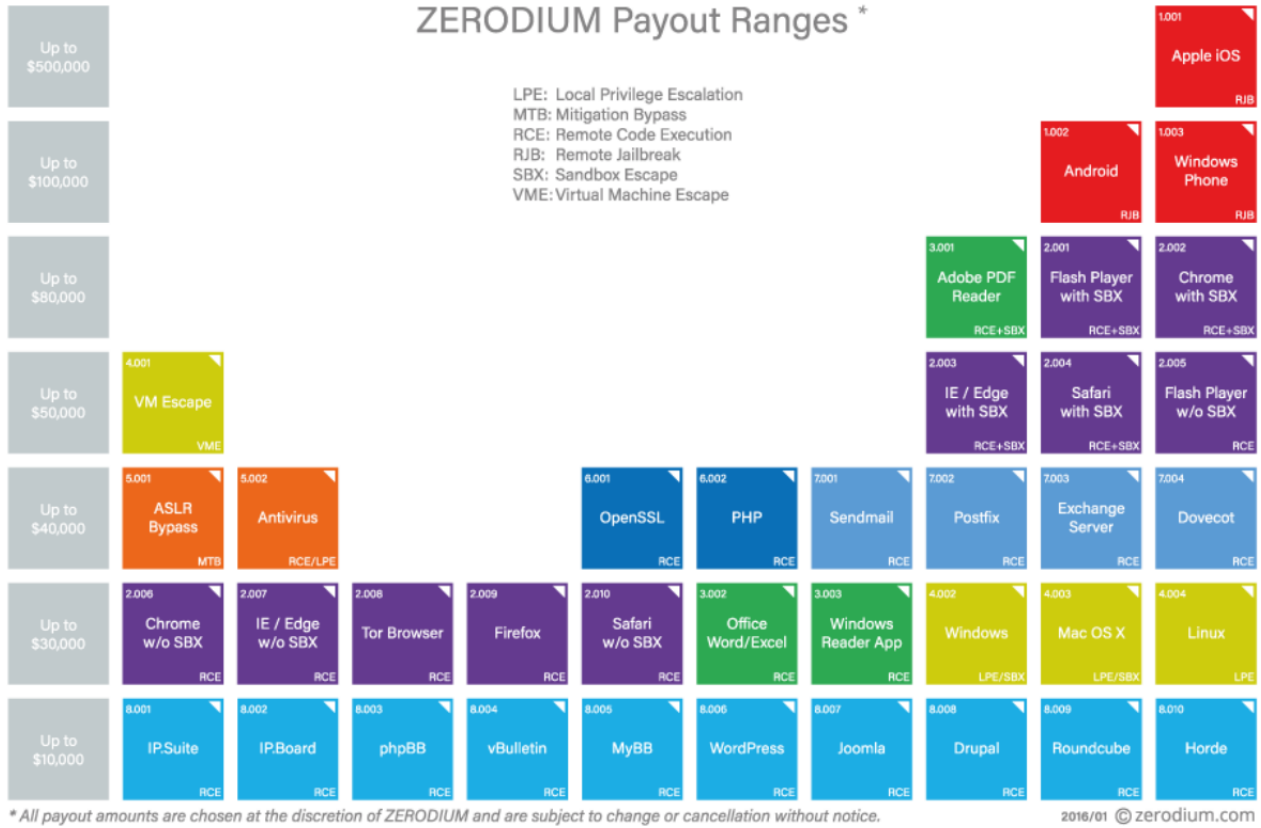
Lab03: Stack Overflow

Name	Points	Release	Deadline	Solved
simple-bof	20	09-09-2016 00:00:00	09-23-2016 00:00:00	25
jmp-to-stack	20	09-09-2016 00:00:00	09-23-2016 00:00:00	25
jmp-to-env	20	09-09-2016 00:00:00	09-23-2016 00:00:00	25
frobnicated	20	09-09-2016 00:00:00	09-23-2016 00:00:00	25
argco	20	09-09-2016 00:00:00	09-23-2016 00:00:00	23
lack-of-four	20	09-09-2016 00:00:00	09-23-2016 00:00:00	23
jmp-to-where	20	09-09-2016 00:00:00	09-23-2016 00:00:00	15
unusal-main	20	09-09-2016 00:00:00	09-23-2016 00:00:00	17
man-strncpy	20	09-09-2016 00:00:00	09-23-2016 00:00:00	15
upside-down	20	09-09-2016 00:00:00	09-23-2016 00:00:00	8

Discussion: Lab03

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

Discussion: Not Yet Motivated?



Discussion: How to Prevent Stack Overflow?

Discussion: How to Prevent Stack Overflow?

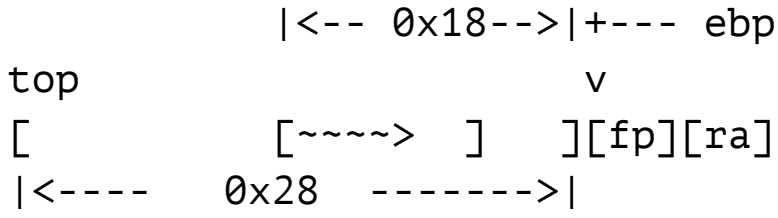
- Exploitation mitigation (today's topic)
- Bug prevention

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 -d crackme0x00
...
8048448:       8d 45 e8                lea    -0x18(%ebp),%eax
804844b:       89 44 24 04             mov    %eax,0x4(%esp)
804844f:       c7 04 24 8c 85 04 08    movl   $0x804858c,(%esp)
8048456:       e8 d5 fe ff ff          call   8048330 <scanf@plt>
...
```



Reminder: Exploiting crackme0x00

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

crackme0x00 in C

```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?

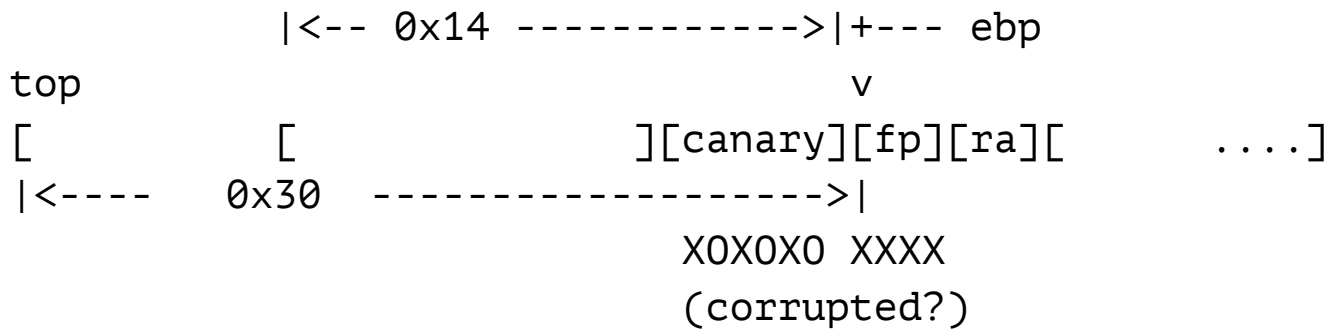
```
scanf("%s", buf)
scanf("%15s", buf);
```

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



Subtle Design Choices

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

Lab05: Exploiting Weakness of Canary

In-class Tutorial

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

```
$ git git@clone tc.gtisc.gatech.edu:seclab-pub cs6265
or
$ git pull
$ cd cs6265/lab05
$ ./init.sh

$ cd tut
$ cat README
```

References

- Bypassing StackShield