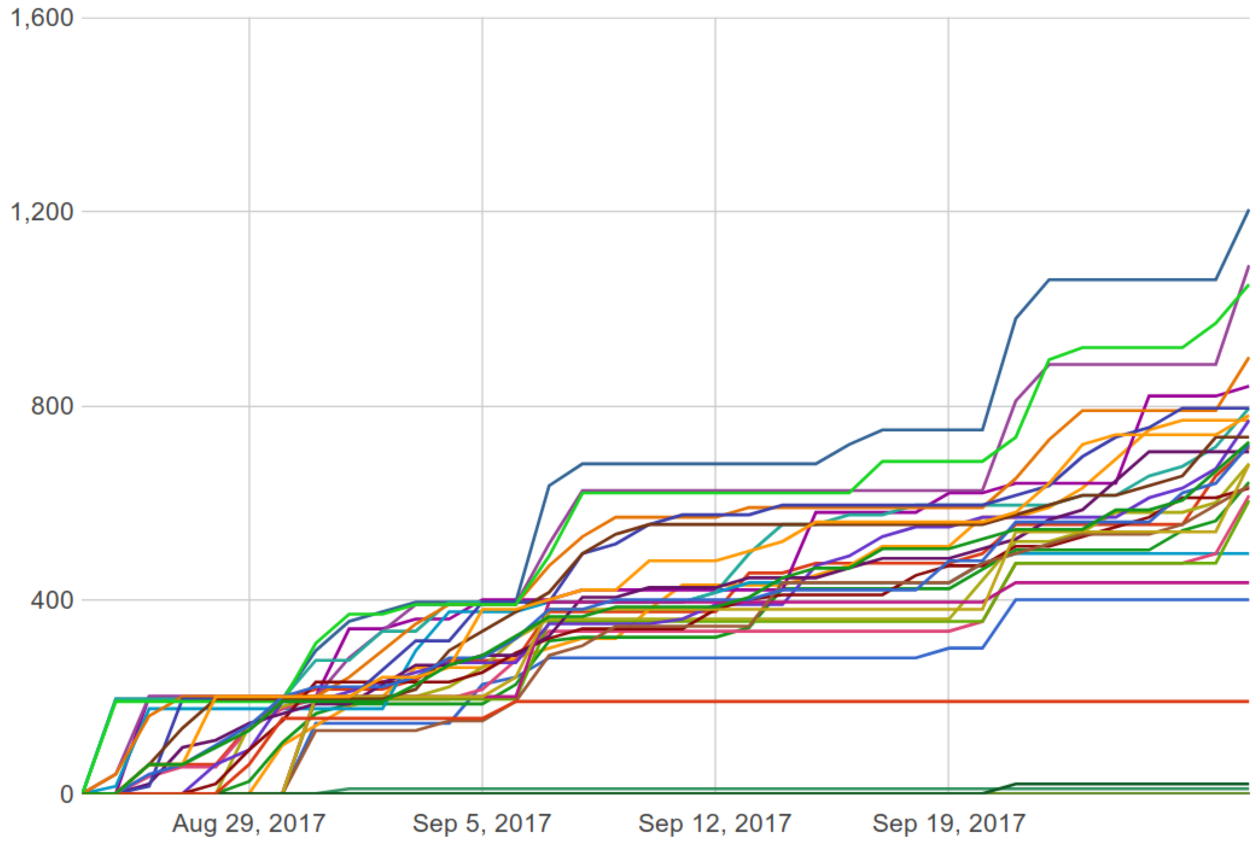
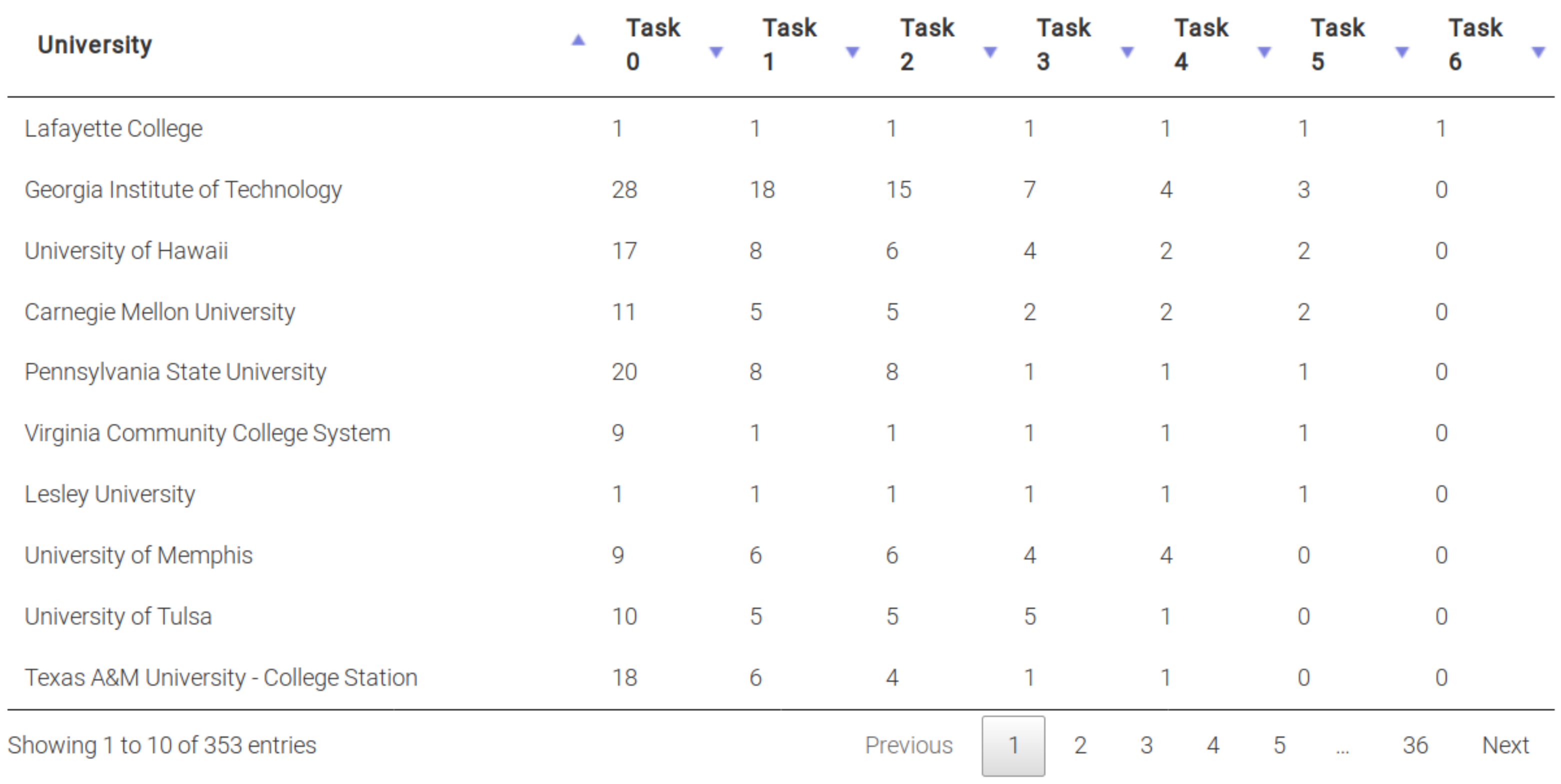
Lec06: DEP and ASLR

Taesoo Kim

Scoreboard


1,600
1,200
800
400
0
Aug 29, 2017
Sep 5, 2017
Sep 12, 2017
Sep 19, 2017

NSA Codebreaker Challenges



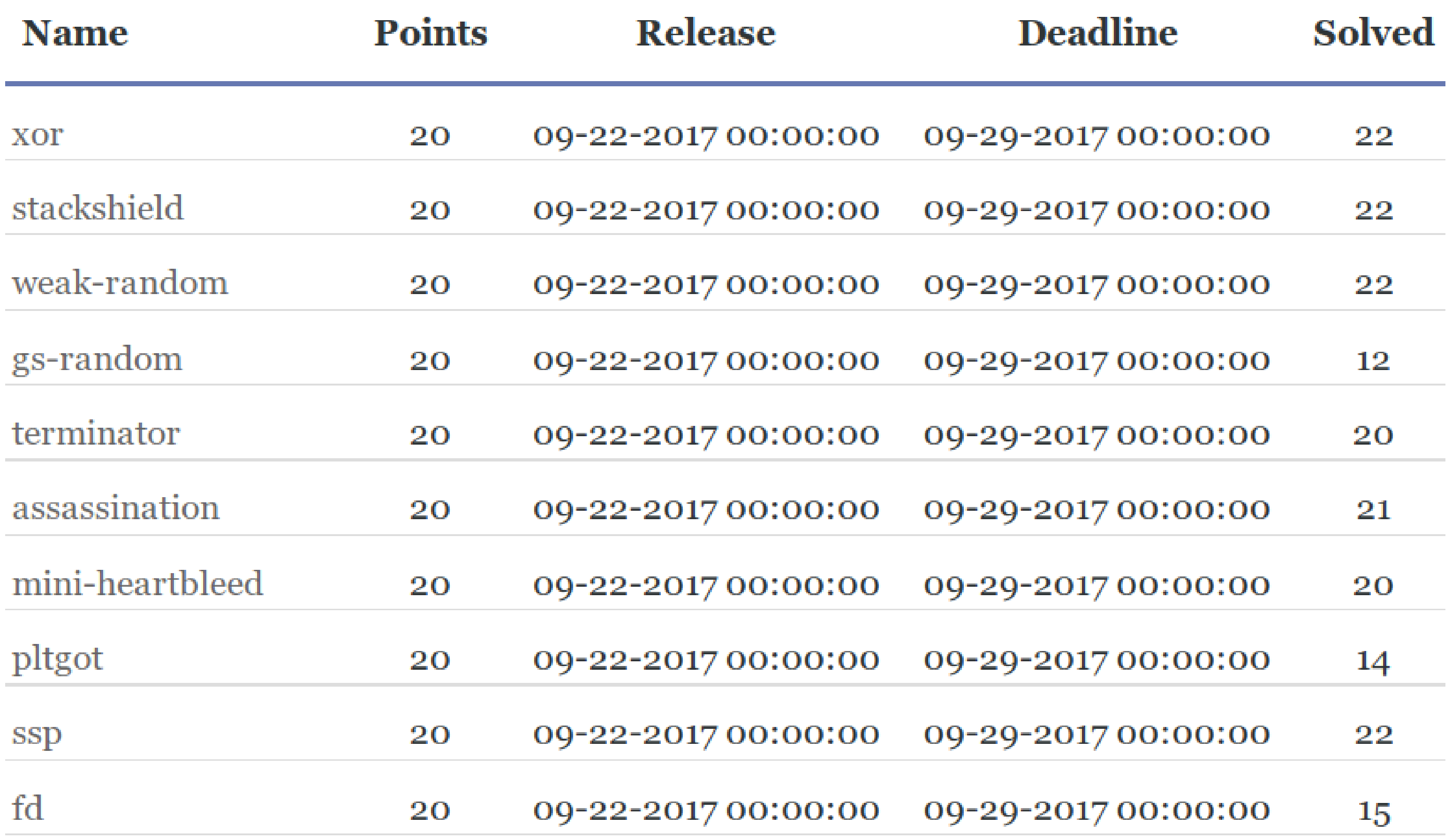
University	Task 0	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6
Lafayette College	1	1	1	1	1	1	1
Georgia Institute of Technology	28	18	15	7	4	3	0
University of Hawaii	17	8	6	4	2	2	0
Carnegie Mellon University	11	5	5	2	2	2	0
Pennsylvania State University	20	8	8	1	1	1	0
Virginia Community College System	9	1	1	1	1	1	0
Lesley University	1	1	1	1	1	1	0
University of Memphis	9	6	6	4	4	0	0
University of Tulsa	10	5	5	5	1	0	0
Texas A&M University - College Station	18	6	4	1	1	0	0

Showing 1 to 10 of 353 entries

Administrivia

- Congrats!! We've completed the half of labs!
- Due: Lab06 is out and its due on Oct 5 at midnight
- NSA Codebreaker Challenge → Due: Nov 30
- We'll release new lab every Thursday at 8pm
- If you are working on Thursday, please connect to "-p 2024"
- If you haven't read yet, please check some time saving tips on Piazza.

Lab05: Stack Protection



Name	Points	Release	Deadline	Solved
xor	20	09-22-2017 00:00:00	09-29-2017 00:00:00	22
stackshield	20	09-22-2017 00:00:00	09-29-2017 00:00:00	22
weak-random	20	09-22-2017 00:00:00	09-29-2017 00:00:00	22
gs-random	20	09-22-2017 00:00:00	09-29-2017 00:00:00	12
terminator	20	09-22-2017 00:00:00	09-29-2017 00:00:00	20
assassination	20	09-22-2017 00:00:00	09-29-2017 00:00:00	21
mini-heartbleed	20	09-22-2017 00:00:00	09-29-2017 00:00:00	20
pltgot	20	09-22-2017 00:00:00	09-29-2017 00:00:00	14
ssp	20	09-22-2017 00:00:00	09-29-2017 00:00:00	22
fd	20	09-22-2017 00:00:00	09-29-2017 00:00:00	15

Best Write-ups for Lab05

- xor: shudak3, carterchen
- stackshield: spark720, shudak3
- weak-random: markwis, spark720
- gs-random: carterchen, shudak3
- terminator: spark720, brian_edmonds
- assassination: carterchen, dhaval
- mini-heartbleed: rpgiri, brian_edmonds
- pltgot: carterchen, N/A
- ssp: shudak3, carterchen
- fd: luoyinfeng, spark720

Discussion: Lab05

- 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?)

Discussion: xor

- How xor canary works?
- What happens if RA is overwritten (or leaked)?

Discussion: xor

Discussion: stackshield

- 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)

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

   /* buf = [val] ... [addr] */
   /* *addr = val */

   strcpy(buf, msg);  /* overwrite msg (addr) */
   capitalize(buf);
   strcpy(msg, buf);  /* overwrite addr with buf */
   ...
}
```

Discussion: gs-random

Discussion: terminator

- How is the terminator canary implemented?

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: pltgot

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

Discussion: fd

Discussion: fd

- Why need vtable?

Discussion: fd

Discussion: fd

- How to prevent this vulnerability?

Discussion: How to make exploitation difficult?

Discussion: How to make exploitation difficult?

- What if the stack address (or code/heap) is random?
    - How could you exploit any challenge in the last week?
- What if the stack/heap memory is not executable?
    - Then, where to put your shellcode?

Today's Tutorial

- In-class tutorial:
    - About: format string vulnerability
    - Format string to arbitrary read
    - Format string to arbitrary write
    - (optional) Format string to arbitrary execution

Format string: *printf

```
1) printf("hello: %d", 10);
2) printf("hello: %d/%d", 10, 20);
3) printf("hello: %d/%d/%d", 10, 20);
```

Format string: *printf

```
printf("%d/%d/%d", a1, a2 ...)

     +----(n)----+
     |           v
[ra][fmt][a1][a2][a3][..]
         (1) (2) (3) ....
```

Format string specifiers

```
printf(fmt);

%p: pointer
%s: string
%d: int
%x: hex

%[nth]$p
 (e.g., %1$p = first argument)
```

Arbitrary Read

```
printf("\xaa\xbb\xcc\xdd%3$s")

     +---(3rd)---+
     |           v
[ra][fmt][a1][a2][\xaa\xbb\xcc\xdd%3$s]
         (1) (2) (3) ....

-> "\xaa\xbb\xcc\xdd[value]"
```

More Format Specifiers

```
printf("1234%n", &len) -> len=4

%n: write #bytes
%hn (short), %hhn (byte)

NOTE. %10d: print an int on 10-space word (e.g., "        10")
```

Write (sth) to an Arbitrary Location

```
printf("\xaa\xbb\xcc\xdd%3$n")

     +---(3rd)---+
     |           v
[ra][fmt][a1][a2][\xaa\xbb\xcc\xdd%3$n]
         (1) (2) (3) ....

-> "\xaa\xbb\xcc\xdd" = 4
```

Arbitrary Write

```
printf("\xaa\xbb\xcc\xdd%6c%3$n")

     +---(3rd)---+
     |           v
[ra][fmt][a1][a2][\xaa\xbb\xcc\xdd%6c%3$n]
         (1) (2) (3) ....

-> *(int *)(0xddccbbaa) = strlen("\xaa\xbb\xcc\xdd......") = 10
```

In-class Tutorial

- Step1: Format string to arbitrary read
- Step2: Format string to arbitrary write
- Step3: (optional) Format string to arbitrary execution

```
$ 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/lab06
$ cat README
```

References

- Bypassing ASLR
- Advanced return-into-lib(c) exploits
- Format string vulnerability