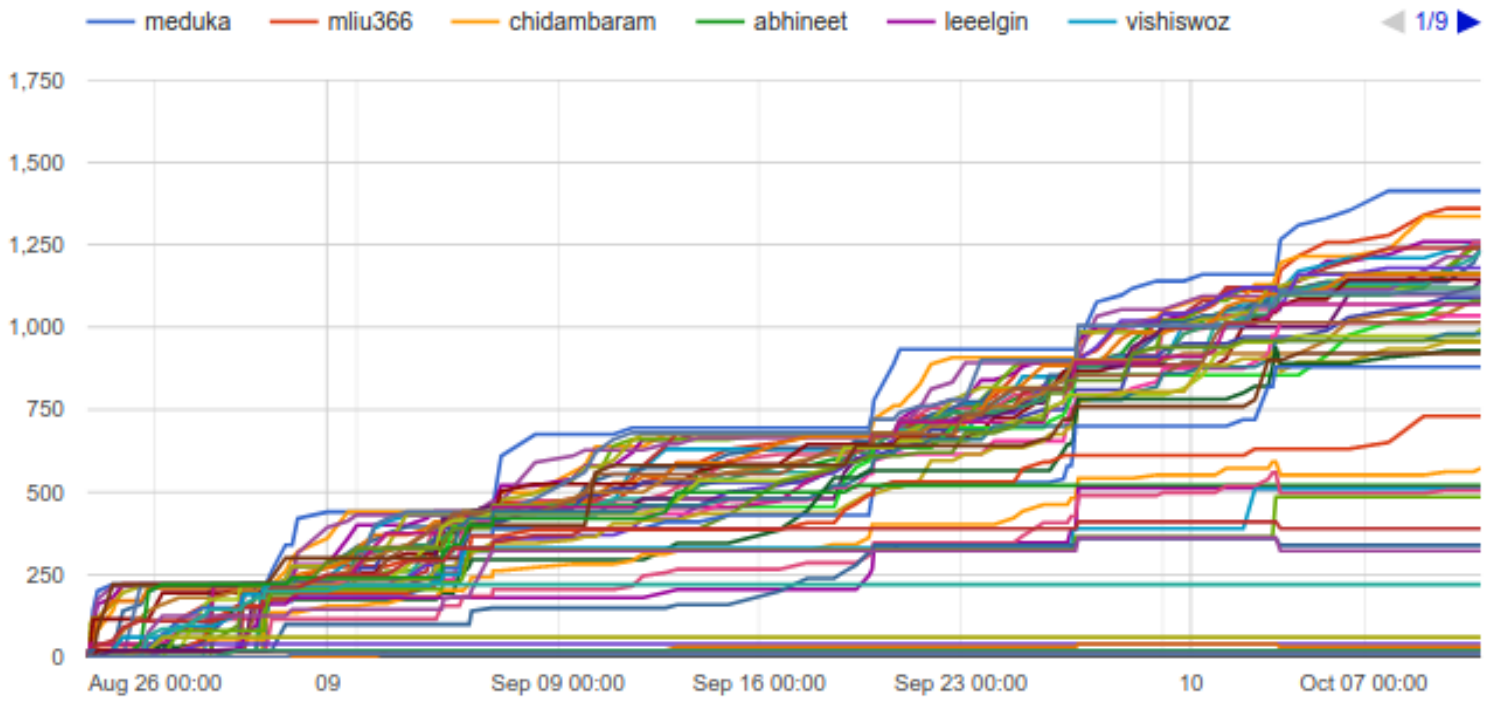
# Lec07: Advanced ROP

*Taesoo Kim*

# Scoreboard


meduka
mliu366
chidambaram
abhineet
leeelgin
vishiswoz
1/9
1,750
1,500
1,250
1,000
750
500
250
0
Aug 26 00:00
09
Sep 09 00:00
Sep 16 00:00
Sep 23 00:00
10
Oct 07 00:00

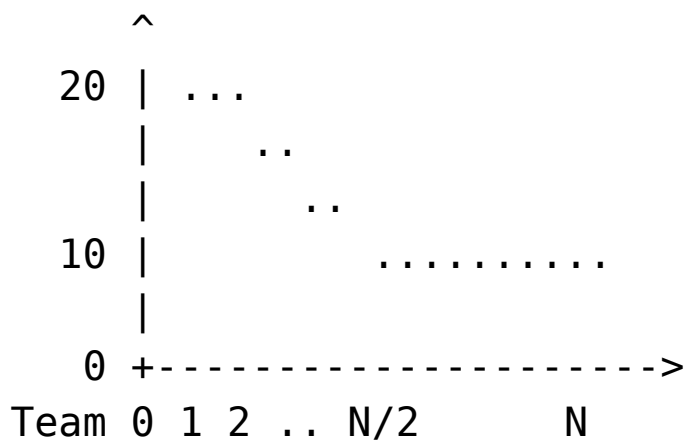
# Administrivia

- In-class CTF: https://ctf.gts3.org/ (Open to public!)
  - Registration: http://bit.ly/tkctf_register (#2-3 persons per team)
  - Rules: https://tc.gts3.org/cs6265/2019/ctf.html
  - Submit your team's challenge by Nov 14
- NSA Codebreaker Challenge → Due: Dec 06

# Scoring: Attack + Defense

Attack (per challenge): 20pt x N challenges

- 10pt for the first blood
- 5pt for the second blood
- 3pt for the third blood



# Scoring: Attack + Defense

Defense: 20pt (per team, about your challenge)

```
- 0    solved:  0 pt (too difficult)
- 1..N solved: 20 pt (okay!)
- N    solved: 10 pt (too easy)
```

# Overview: CTF Template

```
$ wget https://tc.gts3.org/cs6265/2019/_static/ctf-template.zip
$ unzip ctf-template.zip

$ cd ctf-template
$ make help
dist  : build the target and distribute to docker/release
build : build the docker image
run   : run the docker container
test  : test the exploit
submit: zip for submission
```

# Today's Tutorial

- In-class tutorial:
  - ROP with no explicit leaks!
  - ROP on x86_64!

# 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

```
$ cat /proc/sys/kernel/randomize_va_space
2

$ checksec ./target
 [*] '/home/lab/tut06-advrop/target'
  Arch:     amd64-64-little
  RELRO:    Partial RELRO
  Stack:    No canary found
  NX:       NX enabled
  PIE:      No PIE (0x400000)
```

# Reminder: crackme0x00

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

/* __NO LEAK__!
  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;
}
```

# Controlling Arguments in x86_64

Not via stack but via registers!

```
[buf  ]
[.....]
[ra   ] -> pop rdi; ret
[arg1 ]
[ra   ] -> puts()
[ra   ]
```

# Leaking libc's Code Pointer

Leaking the libc function stored at `puts@GOT`

```
[buf  ]
[.....]
[ra   ] -> pop rdi; ret
[arg1 ] -> puts@got
[ra   ] -> puts@plt
[ra   ] (crashing)
```

# Preparing Second Payload

```
[buf  ]
[.....]
[ra   ] -> pop rdi; ret
[arg1 ] -> puts@got
[ra   ] -> puts@plt

[ra   ] -> start
```

# Tutorial Goal: Chaining

```
open("/proc/flag", O_RDONLY)
read(3, tmp, 1040)
write(1, tmp, 1040)
```

# In-class Tutorial

- Step1: Controlling arguments in x86_64
- Step2: Leaking libc's code pointer
- Step3: Preparing Second Payload
- Step4: Chaining multiple functions!

```
$ ssh lab06@3.95.14.86
Password: <password>

$ cd tut06-advrop
$ cat README
```

# References

- ROP