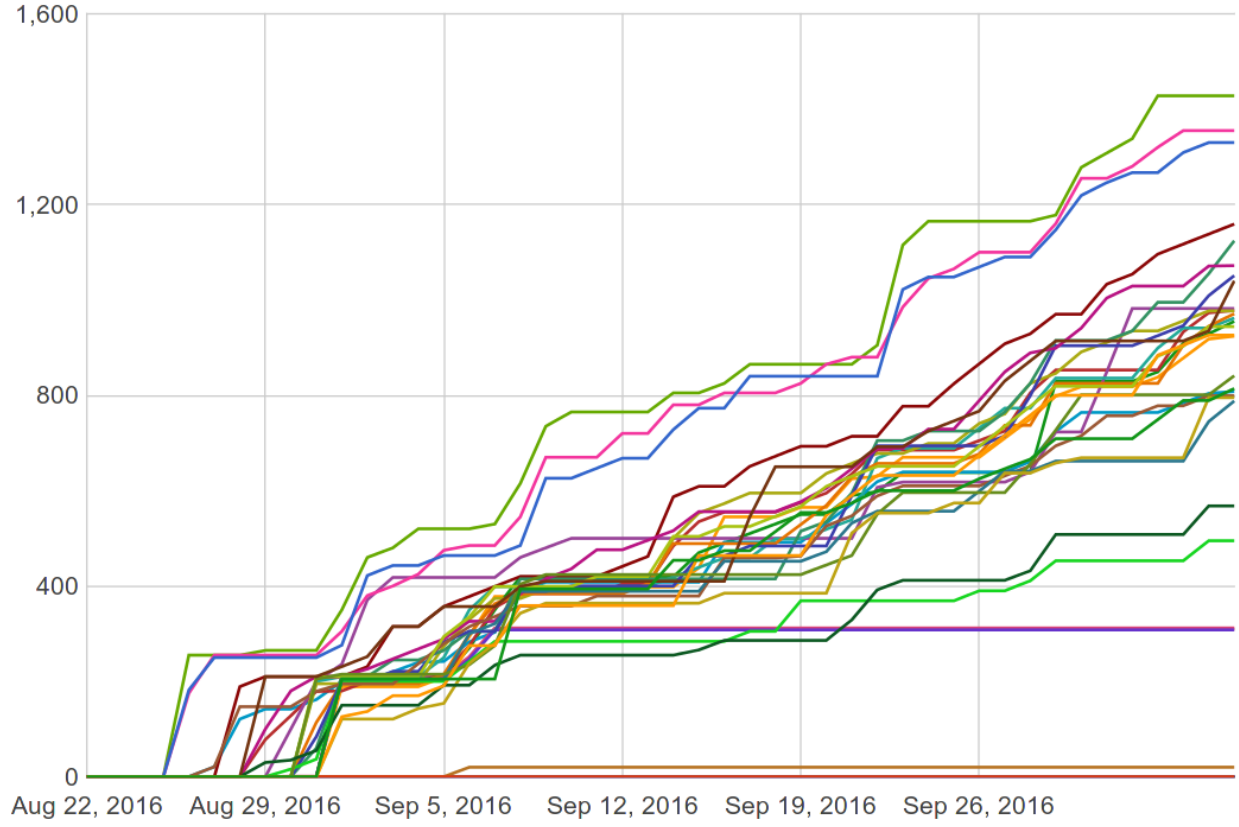
Lec07: Return-oriented programming

Taesoo Kim

Scoreboard


1,600
1,200
800
400
0
Aug 22, 2016
Aug 29, 2016
Sep 5, 2016
Sep 12, 2016
Sep 19, 2016
Sep 26, 2016

NSA Codebreaker Challenges

University	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6
Georgia Institute of Technology	49	40	36	27	10	3
Carnegie Mellon University	26	24	14	10	5	2
Dakota State University	56	40	26	20	8	0
Naval Postgraduate School	6	6	5	5	5	0
University of Colorado at Colorado Springs	14	11	8	8	3	0
Purdue University	11	9	6	6	2	0
Davenport University	7	5	5	5	2	0
Rensselaer Polytechnic Institute	5	4	4	3	2	0
University of Tulsa	5	5	3	2	2	0
University of Maryland, Baltimore County	26	21	12	10	1	0

Administrivia

- Now, it's realistic? :)
- Due: Lab07 is out and its due on Oct 20 at midnight
- NSA Codebreaker Challenge → Due: Dec 1
- Oct 14 : Web Penetration Testing

Lab06: DEP and ASLR

Name	Points	Release	Deadline	Solved
libbase	20	09-30-2016 00:00:00	10-07-2016 00:00:00	22
moving-target	20	09-30-2016 00:00:00	10-07-2016 00:00:00	23
fmtstr-read	20	09-30-2016 00:00:00	10-07-2016 00:00:00	25
fmtstr-write	20	09-30-2016 00:00:00	10-07-2016 00:00:00	25
fmtstr-digging	20	09-30-2016 00:00:00	10-07-2016 00:00:00	23
brainfxxk	20	09-30-2016 00:00:00	10-07-2016 00:00:00	10
fd-const	20	09-30-2016 00:00:00	10-07-2016 00:00:00	7
fmtstr-heap	20	09-30-2016 00:00:00	10-07-2016 00:00:00	6
profile	20	09-30-2016 00:00:00	10-07-2016 00:00:00	5
mini-sudo	20	09-30-2016 00:00:00	10-07-2016 00:00:00	17

Discussion: Lab06

- What's the most "annoying" bug or challenge?
- What's the most "interesting" bug or challenge?
- So, DEP and ASLR are useless?

Take-outs from DEP/ASLR?

- Do you think DEP/ASLR make your life more difficult?
- Is exploitation still possible?
- Although we can't place shellcode into stack/heap, we can still hijack the control flow of a program in many interesting ways

Discussion: Modern Exploit on ASLR (PIE)

- Leak (or infer) code pointers (so map into library or code)
- Construct ROP (today's topic)
- (although there are a few proposals, such as CFI, to mitigate ROPs)

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

```
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

```
printf("Password OK:)")
system("/bin/sh")
exit(0)
```

In-class Tutorial

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

```
$ git git@clone tc.gtisc.gatech.edu:seclab-pub cs6265
or
$ git pull
$ cd cs6265/lab07
$ ./init.sh

$ cd tut
$ cat README
```

References

- ROP