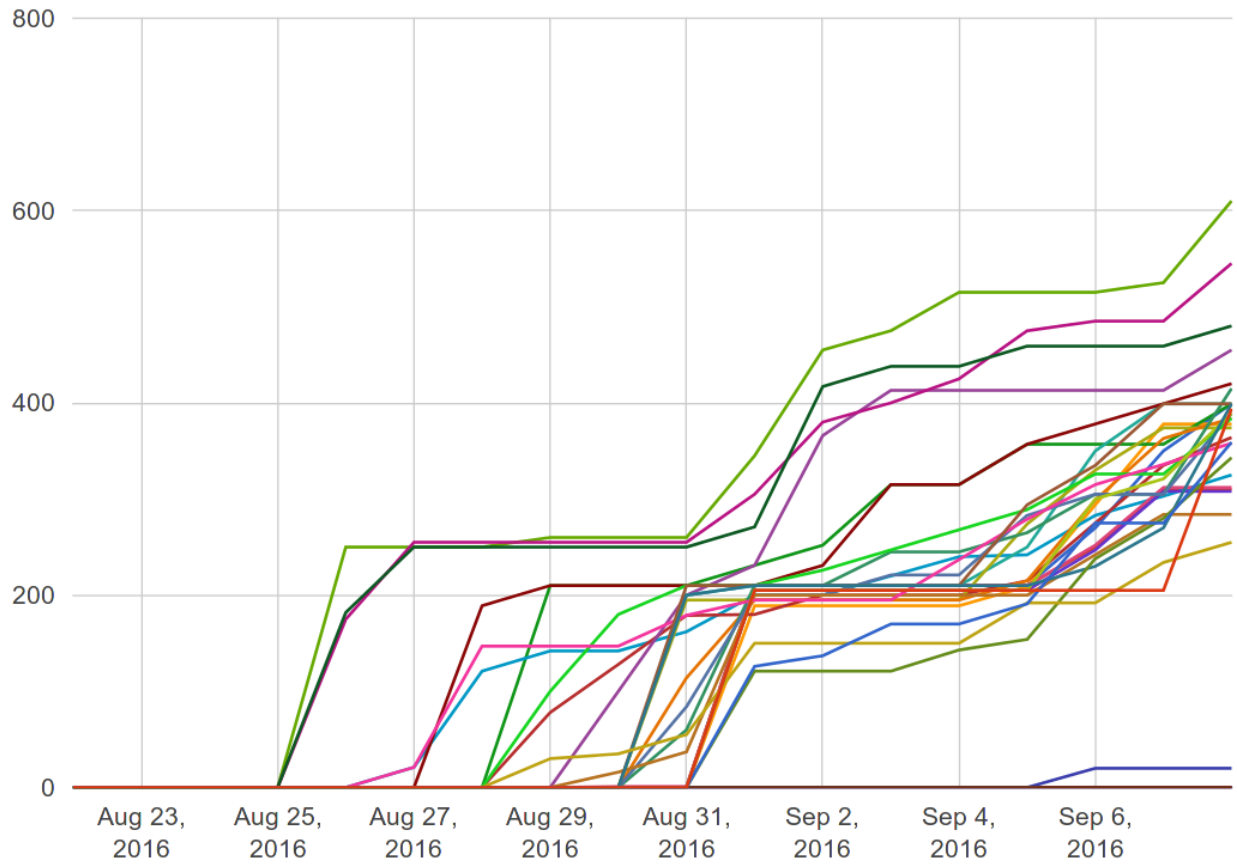
Lec03: Writing Exploits

Taesoo Kim

Scoreboard


800
600
400
200
0
Aug 23, 2016
Aug 25, 2016
Aug 27, 2016
Aug 29, 2016
Aug 31, 2016
Sep 2, 2016
Sep 4, 2016
Sep 6, 2016

Administrivia

- Survey: how many hours did you spend? (<3h, 6h, 10h, >20h)
- Join Piazza!
- An optional recitation on every Wed
  - 5:00-6:00pm (in Klaus 1447)
  - 6:00-6:30pm (in Klaus 3126)
- Lab03: NSA Codebreaker Challenge
- Lab04: stack overflow is released!
- Due : Both, Sept 22 at midnight

NSA Codebreaker Challenges

NSA Codebreaker Challenges

> *The NSA Codebreaker Challenge provides students with a hands-on opportunity to develop their reverse-engineering / low-level code analysis skills while working on a realistic problem set centered around the NSA's mission .*

NSA Codebreaker Mission

> *Terrorists have recently developed a new type of remotely controlled Improvised Explosive Device (IED), making it harder for the U.S. Armed Forces to detect and ultimately prevent roadside bomb attacks against troops deployed overseas. The National Security Agency (NSA), in accordance with its support to military operations mission, has been asked to develop capabilities for use against this new threat. This will consist of six tasks of increasing difficulty, with the ultimate goals of being able to disarm the IEDs remotely and permanently render them inoperable without the risk of civilian casualties.*

NSA Codebreaker Challenges Tasks

- Task 1: Compute a hash and identify IED network ports
- Task 2: Refine IED network traffic signature
- Task 3: Decrypt IED key file
- Task 4: Disarm an IED with the key
- Task 5: Disarm any IED without a key
- Task 6: Permanently disable any IED

Submit your links as flags provided after the completion of each task.

Best Write-ups for Lab02

- aepifano@gatech.edu
- jinho.jung@gatech.edu

Bomb Stats

- Bombs exploded ?? times in total?
- in ?? phases?
- ?? people exploded at least once?

Bomb Stats

- Bombs exploded 29 times in total (29 x -5 = -145 pts)
- in ALL phases!
- 13 people exploded at least once!
  - Each lab: 12/2/2/3 people
  - Each lab: 21/3/2/3 times

Min shellcode

- 2000 bytes? 1000 bytes? 500 bytes? 100 bytes?

Min shellcode

- 6-byte: hirva1093@gatech.edu
- 6-byte: kcarpenter33@gatech.edu

Discussion 0

1. How different is the bomb binary this time?

Discussion 1

1. How did you start exploring the "bomb" (no symbol)?

DEMO

- How to break at the entry point? (idea?)
- PEDA: find "Boom"
- Address translation to find xreferences

Discussion 2 (phase 1)

1. What's going on the first phase?

DEMO

- Check: /proc/self/status
- Binary patching: callq → nop (0x90)
- Attach to the gdb (-p)

Discussion 3 (phase 2)

1. What's going on the second phase?
    - Did you find the main() function (i.e., dispatcher?)

Discussion 3 (obfuscation)

```
$ x/10i 0x555555555952
   0x555555555952:      lea    rsp,[rsp-0x1028]
   0x55555555595a:      or     QWORD PTR [rsp],0x0
   0x55555555595f:      lea    rsp,[rsp+0x1020]
   0x555555555967:      jmp    0x55555555596a
   0x555555555969:      jmp    0x555555549b56
   0x55555555596e:      dec    DWORD PTR [rax-0x7d]
   0x555555555971:      (bad)
   0x555555555972:      or     bl,al
```

Discussion 3 (when tracing)

```
   0x555555555952:      lea    rsp,[rsp-0x1028]
   0x55555555595a:      or     QWORD PTR [rsp],0x0
   0x55555555595f:      lea    rsp,[rsp+0x1020]
-> 0x555555555967:      jmp    0x55555555596a
 |  0x555555555969:      jmp    0x555555549b56
 |  0x55555555596e:      dec    DWORD PTR [rax-0x7d]
 |  0x555555555971:      (bad)
 |  0x555555555972:      or     bl,al
 +->   0x55555555596a:  call   0x5555555558b0
       0x55555555596f:  add    rsp,0x8
       0x555555555973:  ret
       0x555555555974:  push   rbp
```

Discussion 4 (phase 3)

1. What's going on the third phase?

Discussion 4 (phase 3)

```
int count = 0;
void progress_bar(int signo) {
  if (count != 0)
    printf("\b\b\b\b");
  printf("| %02d%%", count);
  count += 2;
}

phase() {
  signal(SIGTRAP, progress_bar);
  for (int i = 0; i < 50; i ++) {
    ...
    __asm__ volatile("int3");
  }
}
```

Discussion 5 (phase 4)

1. What's going on the last phase? (nothing special!)

32/64 Shellcode

1. int $80 vs. syscall

```
$ man syscall
```

What's about poly shellcode?

1. What's your general idea?

Dispatching routine

```
          +-------------+
          |             v
[dispatcher][x86      ][x86_64   ]

e.g., 0x40 0x90
 - x86        inc eax
 - x86_64     REX + nop

x86   : [ * ][goto x86 shellcode]
x86-64: [nop][ * ][goto x86_64 shellcode]
arm   : [nop][nop][ * ][goto arm shellcode]
MIPS  : [nop][nop][nop][ * ][goto MIPS shellcode]
```

Discussion 6 (shellcode ascii/min)

1. Wow, what are your tricks?
2. NOTE. can be as small as zero byte..

Lab04: Stack overflow (due in two weeks)

- Finally! time to write real exploits (i.e., control hijacking)
- TONS of interesting challenges!
  - e.g., lack-of-four, frobnicated, upside-down ..

Today's Tutorial

- Example: hijacking crackme0x00!
- A template exploit code
- In-class tutorial
  - IDA (yeah!)
  - Extending the exploit template (python)

DEMO: IDA/crackme0x00

- IDA w/ crackme0x00
- exploit writing

crackme0x00

```
$ objdump -d crackme0x00
...
8048414:       55                          push   %ebp
8048415:       89 e5                       mov    %esp,%ebp
8048417:       83 ec 28                    sub    $0x28,%esp

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

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>

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

In-class Tutorial

- Step 1: Install IDA (feel free to use from now)
- Step 2: Play with your first exploit!

```
$ git git@clone tc.gtisc.gatech.edu:seclab-pub cs6265
or
$ git pull
$ cd cs6265/lab03
$ ./init.sh

$ cd tut
$ cat README
```

References

- IDA Demo
- Phrack #49-14