Lec06: DEP and ASLR

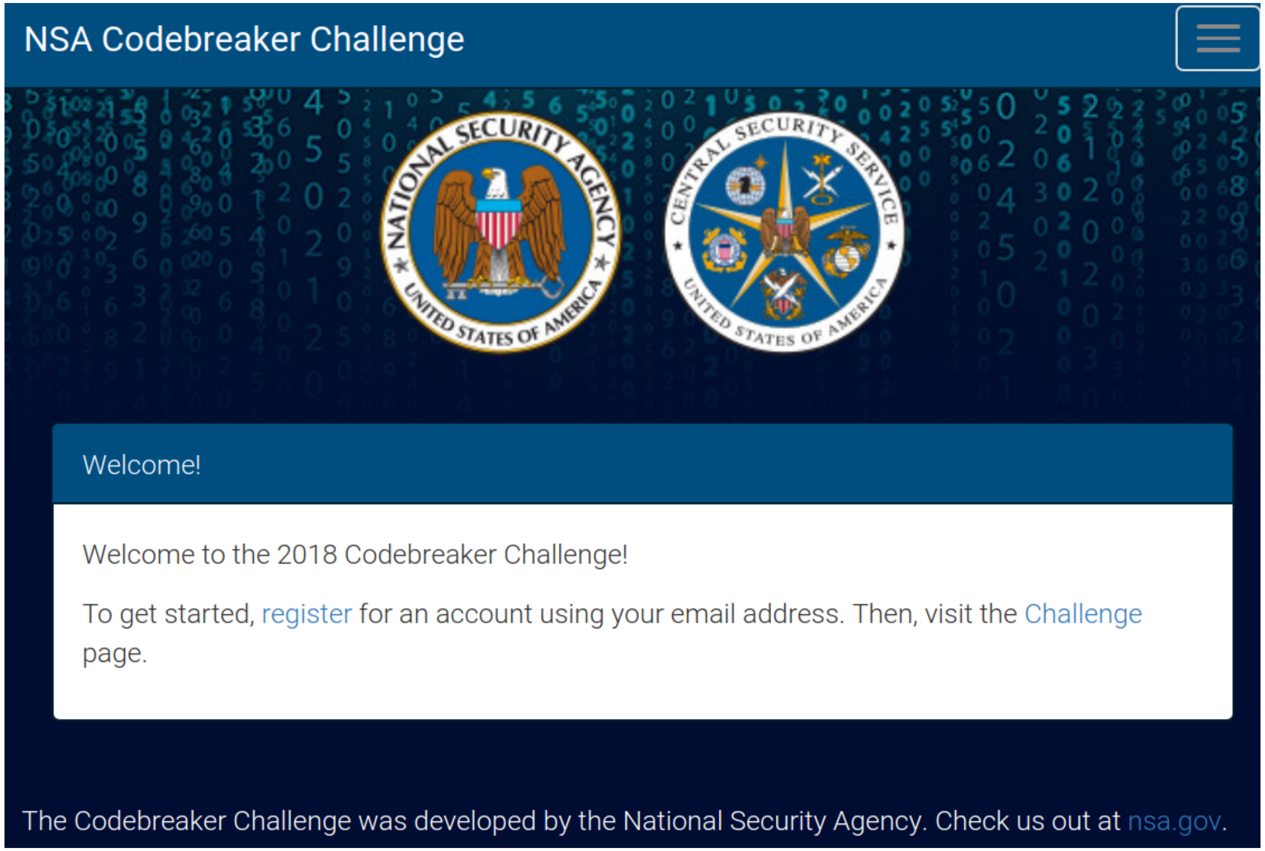
Taesoo Kim

Scoreboard

Administrivia

- Due: Lab04 is extended for one week!
- Due: Lab05 is out and its due on Oct 5 at midnight
- Lab10: NSA Codebreaker Challenge → Due: Nov 29
- Offline CTF (Nov 16/17): Please find your team mates (3-4 people)!

NSA Codebreaker Challenges



Welcome!

Welcome to the 2018 Codebreaker Challenge!

To get started, register for an account using your email address. Then, visit the Challenge page.

The Codebreaker Challenge was developed by the National Security Agency. Check us out at nsa.gov.

NSA Codebreaker Challenges

“ *A new strain of ransomware has managed to penetrate several critical government networks and NSA has been called upon to assist in remediating the infection to prevent massive data losses. For each infected machine, an encrypted copy of the key needed to decrypt the ransomed files has been stored in a smart contract on the Ethereum blockchain and is set to only be unlocked upon receipt of the ransom payment. Your mission is to ultimately* ***(1) find a way to unlock the ransomware without giving in to the attacker's demands*** *and (2)* ***figure out a way to recover all of the funds already paid by other victims****.*

About Offline CTF

- Starts in Nov 16th at 3pm (in class)
  - Options: 1-day, 3-day, one week, or two weeks?
- Each team (3-4 people) prepares one challenge problem + N (from TAs)
  - Oct 26: Releasing a sample challenge, structure, and scoring rules
  - Nov 13: Deadline for your team’s challenge (think ahead!)

Today's Tutorial

- Learning about a format string bug
- A format string bug → an arbitrary read
- A format string bug → an arbitrary write
- A format string bug → an arbitrary execution

Format String: e.g., printf()

- How does printf() know of #arguments passed?
- How do we access the arguments in the function?

```
1) printf("hello: %d", 10);
2) printf("hello: %d/%d", 10, 20);
3) printf("hello: %d/%d", 10, 20, 30);
```

Format String: e.g., printf()

- What does it happen if we miss one argument?

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

Format String: e.g., printf()

- What does printf() print out? guess?

```
printf("%d/%d/%d", 10, 20)

     +----(n)----+
     |           v
[ra][fmt][10][20][??][..]
         (1) (2) (3) ....
```

About a “Variadic” Function

```
int sum_up(int count,...) {
  va_list ap;
  int i, sum = 0;

  va_start (ap, count);
  for (i = 0; i < count; i++)
    sum += va_arg (ap, int);

  va_end (ap);
  return sum;
}
```

About a “Variadic” Function

```
va_start (ap, count);
  lea    eax,[ebp+0xc]                  // Q1. 0xc?
  mov    DWORD PTR [ebp-0x18],eax

for (i = 0; i < count; i++)
  sum += va_arg (ap, int);

  mov    eax,DWORD PTR [ebp-0x18]
  lea    edx,[eax+0x4]                  // Q2. +4?
  mov    DWORD PTR [ebp-0x18],edx
  mov    eax,DWORD PTR [eax]
  add    DWORD PTR [ebp-0x10],eax
  ...
```

Format String Specifiers

```
printf(fmt);

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

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

Arbitrary Read

- If fmtbuf locats on the stack (perhaps, one of caller's),
- Then, we can essentially control its argument!

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

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

                    (1)(2)(3)
-> printf("...%3$s", _, _, 0xddccbbaa)
```

More Format Specifiers

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

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

Tip 2.
    %10d: print an int on 10-space word
    (e.g., "        10")
```

Write (sth) to an Arbitrary Location

- Similar to the arbitrary read, we can control the arguments!

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

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

                    (1)(2)(3)
-> printf("...%3$n", _, _, 0xddccbbaa)
   *0xddccbbaa = 4 (#chars printed so far)
```

Arbitrary Write

- In fact, we can control what to write (see more in the tutorial)!

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

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

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

In-class Tutorial

- Step1: A format string bug → an arbitrary read
- Step2: A format string bug → an arbitrary write
- Step3: A format string bug → an arbitrary execution

```
$ ssh lab05@computron.gtisc.gatech.edu -p 9005
$ ssh lab05@cyclonus.gtisc.gatech.edu -p 9005
Password: lab05

$ cd tut-fmtstr
$ cat README
```

References

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