Lec07: Return-oriented programming

Taesoo Kim
Scoreboard
## NSA Codebreaker Challenges

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Points</th>
<th>Release</th>
<th>Deadline</th>
<th>Solved</th>
</tr>
</thead>
<tbody>
<tr>
<td>libbase</td>
<td>20</td>
<td>09-30-2016 00:00:00</td>
<td>10-07-2016 00:00:00</td>
<td>22</td>
</tr>
<tr>
<td>moving-target</td>
<td>20</td>
<td>09-30-2016 00:00:00</td>
<td>10-07-2016 00:00:00</td>
<td>23</td>
</tr>
<tr>
<td>fmtstr-read</td>
<td>20</td>
<td>09-30-2016 00:00:00</td>
<td>10-07-2016 00:00:00</td>
<td>25</td>
</tr>
<tr>
<td>fmtstr-write</td>
<td>20</td>
<td>09-30-2016 00:00:00</td>
<td>10-07-2016 00:00:00</td>
<td>25</td>
</tr>
<tr>
<td>fmtstr-digging</td>
<td>20</td>
<td>09-30-2016 00:00:00</td>
<td>10-07-2016 00:00:00</td>
<td>23</td>
</tr>
<tr>
<td>brainfxxk</td>
<td>20</td>
<td>09-30-2016 00:00:00</td>
<td>10-07-2016 00:00:00</td>
<td>10</td>
</tr>
<tr>
<td>fd-const</td>
<td>20</td>
<td>09-30-2016 00:00:00</td>
<td>10-07-2016 00:00:00</td>
<td>7</td>
</tr>
<tr>
<td>fmtstr-heap</td>
<td>20</td>
<td>09-30-2016 00:00:00</td>
<td>10-07-2016 00:00:00</td>
<td>6</td>
</tr>
<tr>
<td>profile</td>
<td>20</td>
<td>09-30-2016 00:00:00</td>
<td>10-07-2016 00:00:00</td>
<td>5</td>
</tr>
<tr>
<td>mini-sudo</td>
<td>20</td>
<td>09-30-2016 00:00:00</td>
<td>10-07-2016 00:00:00</td>
<td>17</td>
</tr>
</tbody>
</table>
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!
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");
}
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 :)
[arg1] -> " /bin/sh"
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