Off-by-one Vulnerability

Wei Wang
Stack Buffer Overflow Vulnerabilities May NOT Touch Return Addresses

• So far we have been studying stack buffer overflow vulnerabilities that allows direct writing of return addresses

• However, in real life, many stack buffer overflow vulnerabilities do not allow directly access to return addresses

• To exploit these vulnerabilities, the attackers have to take an in-direct approach
Off-by-one Error/Vulnerability

- One class of such vulnerabilities is call Off-by-one Error
- An example:

```c
void bar(){
    char buf[256];
    int i;

    for(i = 0; i <= 256; i++)
        buf[i] = getchar();

    /* some other statements */
    .... .... ...
}
```

- Due to the complexity of pointer manipulation, this kind of bugs are very common
Exploit the Off-by-one Vulnerability

- To exploit the previous example, assuming *bar* is invoked by function *foo*
- The stack before the for loop is executed
  - What happens after *bar* returns?

bar saves Foo's EBP; i.e., bar's saved EBP records the beginning address of foo's stack frame
Exploit the Off-by-one Vulnerability cont'd

- What happens if the extra one char is written to the stack?

![Diagram showing stack frames for foo and bar]

- foo's stack frame
  - return address for foo
  - saved EBP
  - foo's local variables
- bar's stack frame
  - return address for bar
  - saved EBP
  - buf
  - other local vars of bar
Exploit the Off-by-one Vulnerability cont'd

- What happens if the extra one char is written to the stack?
  - the saved EBP is changed
- What happens if the saved EBP is changed?
Exploit the Off-by-one Vulnerability cont'd

- What happens if the extra one char is written to the stack?
  - the saved EBP is changed

- What happens if the saved EBP is changed?
  - When bar returns, foo cannot find its original stack frame, i.e., local variables and return address

```
return address for foo
 saves EBP

foo's local variables

return address for bar
 saves EBP

buf

other local vars of bar
```

foo's stack frame

bar's stack frame
Exploit the Off-by-one Vulnerability cont'd

- If the last byte of the saved EBP is changed to a smaller number, where does the saved EBP pointers to?

```

<table>
<thead>
<tr>
<th>return address for foo</th>
</tr>
</thead>
<tbody>
<tr>
<td>saved EBP</td>
</tr>
<tr>
<td>foo's local variables</td>
</tr>
<tr>
<td>return address for bar</td>
</tr>
<tr>
<td>saved EBP</td>
</tr>
<tr>
<td>xx</td>
</tr>
<tr>
<td>buf</td>
</tr>
<tr>
<td>other local vars of bar</td>
</tr>
</tbody>
</table>

foo's stack frame

bar's stack frame

```
Exploit the Off-by-one Vulnerability cont'd

If the last byte of the saved EBP is changed to a smaller number, where does the saved EBP pointers to?

- It will points to somewhere lower than foo's original stack frame

```
<table>
<thead>
<tr>
<th>return address for foo</th>
</tr>
</thead>
<tbody>
<tr>
<td>saved EBP</td>
</tr>
<tr>
<td>foo's local variables</td>
</tr>
<tr>
<td>return address for bar</td>
</tr>
<tr>
<td>saved EBP</td>
</tr>
<tr>
<td>buf</td>
</tr>
<tr>
<td>other local vars of bar</td>
</tr>
</tbody>
</table>
```

foo's stack frame

bar's stack frame

Some address below foo's original frame
Exploit the Off-by-one Vulnerability cont'd

- An attacker can then write a bogus stack frame into the buffer to pretend to be foo's stack frame.
- Controlling foo's local variables basically controls foo.

```
  return address for foo
  saved EBP
  foo's local variables

  return address for bar
  saved EBP  xx
  foo's local variables

  other local vars of bar
```
Exploit the Off-by-one Vulnerability cont'd

- What's even better is that the attacker has an return address in hand now
- What is this return address controls?

```
return address for foo
saved EBP
foo's local variables

return address for bar
saved EBP  xx
return address for foo
saved EBP
foo's local variables

other local vars of bar
```
Exploit the Off-by-one Vulnerability cont'd

- What's even better is that the attacker has an return address in hand now.
- What is this return address controls?
  - Code to execute after foo's return

![Diagram showing stack frames and return addresses for foo and bar.]
Exploit the Off-by-one Vulnerability cont'd

- Attackers can now inject code, and change foo's return address to point to the injected code.
- Just one byte, gives us full control as in a normal buffer overflow vulnerability.

![Diagram of stack frames and return addresses]
Off-by-one Vulnerability Summary

- A logic error involving boundary condition
  - Programmer mistakes “<” for “<=” in for loop
- If the buffer is close to saved EBP, off-by-one vulnerability allows attack to overwrite saved EBP
- Attackers can then change the saved EBP to point to the attacker-supplied bogus stack frame
- With the bogus stack frame, the attacker controls return address, and in turns control the flow of execution
- Moral of the story: with several steps, the attacker can indirectly acquire full system control, by exploiting some “not-so-serious” vulnerabilities
More General Pointer Vulnerabilities

- Due to the complexity of pointer manipulation, pointer arithmetic bugs are very common.
- There are also cases where pointers are incorrectly accessed – e.g., dangling pointer vulnerability.
- Some of these bugs allowing direct or in-directly overwriting of important data structures, including return addresses.
- Attacks to these vulnerabilities are very hard to defend against.