Stack Overflow

COMP620

Outline

• Threat
• Problem definition
• Vulnerabilities
• Solutions

Buffer Overflows Are A Major Threat

• Buffer overflows are a major security vulnerability.
• When a security alert contains the phrase “The most severe of these vulnerabilities allows a remote attacker to execute arbitrary code.”, the underlying problem is probably a buffer overflow.
• The Morris worm (the first Internet worm) spread in part by exploiting a stack buffer overflow in the Unix finger server.
• Many students do not know what they are.

National Cyber Alert System

Technical Cyber Security Alert TA09-161A
Adobe Acrobat and Reader Vulnerabilities

Original release date: June 10, 2009
Last revised: --
Source: US-CERT

Systems Affected

* Adobe Reader versions 9.1.1 and earlier
* Adobe Acrobat versions 9.1.1 and earlier

Overview

Adobe has released Security Bulletin APSB09-07, which describes several buffer overflow vulnerabilities that could allow a remote attacker to execute arbitrary code.
Buffer Overflows as Percent of Total Vulnerabilities

source: DHS National Cyber Security Division/US-CERT National Vulnerability Database

What will this program do?

```c
void examplefunc() {
    int stuff = 0;
    char info[4];
    int i;
    for (i = 0; i < 7; i++) {
        info[i] = stuff++;
    }
}
```

1. Compiler Error
2. Run time buffer overflow error
3. Corrupt data
4. Data execute exception

Basic Buffer Overflow

```c
boolean rootPriv = false;
char name[8];
cin >> name;
```

• When the program reads the name “Smith”

<table>
<thead>
<tr>
<th>S</th>
<th>m</th>
<th>i</th>
<th>t</th>
<th>h</th>
<th></th>
<th>false</th>
</tr>
</thead>
<tbody>
<tr>
<td>char name[8]</td>
<td>rootPriv</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

• When the program reads the name “Armstrong”

<table>
<thead>
<tr>
<th>A</th>
<th>r</th>
<th>m</th>
<th>s</th>
<th>t</th>
<th>r</th>
<th>o</th>
<th>n</th>
<th>g</th>
</tr>
</thead>
<tbody>
<tr>
<td>char name[8]</td>
<td>rootPriv</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Stack Overflow

- A stack overflow exploit occurs when a user enters data that exceeds the memory reserved for the input. The input can change adjacent data or the return address on the stack.

```
char myStuff[4];
```

Program Stack

The Other Stack Overflow

- The phrase “Stack Overflow” also applies to programs whose stack grows to exceed the available memory.
- These stack overflows are often generated by runaway recursion or allocated large data structures on the stack.
- This form of stack overflow is not usually the result of a malicious attack.

  This is a problem for another day.

Program Memory Organization

Stack Review

- Consider the function

  ```
  void thefunc( float &dog, int cat ){
    char cow[4];
  }
  ```
  - that is called by the main program

  ```
  int oak = 5;
  float pine = 7.0;
  float *birch = &pine;
  thefunc( birch, oak );
  ```
Stack for Call

• push oak

5 (value of oak)

thefunc(birch, oak);

Stack for Call

• push oak
• push birch

5 (value of oak)
address of pine

thefunc(birch, oak);

Stack for Call

• push oak
• push birch
• push return address

5 (value of oak)
address of pine
return address

thefunc(birch, oak);

Stack for Call

• push oak
• push birch
• push return address
• push frame pointer

5 (value of oak)
address of pine
return address
Addr of last frame

thefunc(birch, oak);
Stack for Call

- push oak
- push birch
- push return address
- push frame pointer
- Allocate space for local variable, cow[4]

```
thefunc( birch, oak );
```

Overflowing Local Variables

- On an Intel processor (and many others) the stack is extended to lower addresses
- If you address beyond a local variable, you will overwrite the return address.

```
Frame ptr = 'wxyz'
cow[4]=‘abcd’
```

Hacking the Stack

- If a program does not properly check array bounds, it may be possible to give the program specially crafted input that overwrites the return address with a binary value.
- cow[8] to cow[11] are the return address

```
5 (value of oak)
address of pine
return address
Addr of last frame
cow[4]
```

Hacking the Stack

- If a program does not properly check array bounds, it may be possible to give the program specially crafted input that overwrites the return address with a binary value.
- The return address can be changed to the address of a function in the program.
- Function parameters can also be put on the stack

```
5 (value of oak)
address of pine
return address
Addr of last frame
cow[4]
```
Loading Malicious Code

- A long input to a short buffer might also contain binary machine language.
- The return address can be overwritten to cause the program to jump to the newly loaded machine language when it returns.

What is the most evil thing a stack overflow exploit can do?

1. Change the return address to call some function
2. Load binary machine language
3. Crash the program
4. Alter data

Preventing Stack Overflow Exploits

- Better software engineering
- Avoid dangerous functions
- Language choice
- Compiler tools (Stack Guard)
- Analysis tools
- Execution Prevention

Program Errors

- Buffer overflows are primarily caused by programs which fail to properly check for invalid input, particularly longer input than expected.
- Many older library functions (such as `strcpy` or `gets`) did not check the length of the target buffer.
Good Software Engineering Practices

• Because vulnerabilities are primarily caused by unsafe programming, good software engineering principles can significantly improve a program’s safety.
  – Code reviews
  – Testing with long input or too much input
  – Looking for vulnerabilities

Frequent Vulnerabilities

• Careless use of buffers without bounds checking
• Off-by-one errors
• Unsafe library function calls
• Old code used for new purposes (like UNICODE international characters)
• All sorts of other far-fetched but deadly-serious things you should think about

Off-by-one errors

• Off-by-one errors occur when a programmer takes the proper precautions in terms of bounds checking, but forgets that the last index is one less than the size.
• In C strings are terminated by a null character. Programmers often forget to reserve space for the null terminator.
• char myString[10] can only hold 9 printable characters, indexed from 0 to 8.

Unsafe Libraries

• The original C libraries contain several functions that do not consider the length of a target buffer.
• Many of these libraries have been superseded by safer versions that limit the target length.
• Several C++ methods still do not check the size of the target buffer.
Unsafe I/O Functions

• Avoid gets(). It has no way to limit input length
• Use precision specifiers with the scanf() family of functions (scanf(), fscanf(), sscanf(), etc.). Otherwise they will not do any bounds checking for you.
• cin >> char[] will read more characters than the length of the string.
• The cin.get and cin.getline functions allow you to specify a maximum input length.

Unsafe String Functions

• Avoid functions like strcpy() and strcat(). Use strncpy() and strncat() instead.
• Functions like fgets(), strncpy(), and memcpy() are safe if you make sure your buffer is the size you say it is. Be careful of off-by-one errors.
• When using streadd() or strecpy(), make sure the destination buffer is four times the size of the source buffer.

What is the proper call to read characters into a 16 byte array?

1. scanf("%s", myString);
2. scanf("%14s", myString);
3. scanf("%15s", myString);
4. scanf("%16s", myString);
5. scanf("%17s", myString);

Data Expansion

• The size in bytes of the input might not be what causes the buffer overflow, it might be the input itself.
  – For example, if you’re converting a large integer to a string (maybe in binary) make sure the buffer is long enough to hold all possible outputs
  – When converting special characters for web pages (i.e. “>” to “&gt;”) the output can become much larger
  – Unicode is twice the size of ASCII
Safer Languages

- Several modern languages have built-in protection against stack overflow.
- Java and C# check every array reference to ensure that it is within bounds.
- Java does not allow stack violations.

Stack Canaries

- A stack canary is a random number placed on the stack between the user data and the return address.
- Overflowing the local variable and changing the return address will also change the stack canary.
- Before returning, the program checks the canary value.

Data Execution Prevention

- Most newer processors have a bit in the page table that inhibits instruction fetches from that page.
- Newer operating systems can set data execution prevention for stacks.
- This prevents the program from executing machine language loaded on the stack by an exploit.
- This does not prevent programs from overwriting the return address.

Stack Overflow Threats

- Most students:
  - Can't recognize a buffer overflow vulnerability when they see it, so they don't even think of it when coding
  - Are not attuned into the dangers of buffer overflows
  - Do not inspect or test their code as well as you would like
  - Are often not made aware of buffer overflows by instructors or textbooks
Overflow Lab

- Run all of the simulations at
  [http://williams.comp.ncat.edu/overflow/Labs.html](http://williams.comp.ncat.edu/overflow/Labs.html)