CS458/558
Introduction to Computer Security

Buffer Overflow Attack

Buffer Overflow

- Buffer overflow: the top vulnerability in Linux/Unix
- A buffer can be formally defined as "a contiguous block of computer memory that holds more than one instance of the same data type".
- In C and C++, buffers are usually implemented using arrays.
- An extremely common kind of buffer is simply an array of characters.
- A buffer overflow is the result of stuffing more data into a buffer than it can handle.

Buffer Overflow: Basic Example

- A program has defined two data items which are adjacent in memory: an 8-byte-long string buffer A, and a 2-byte string buffer B.
- Initially, A contains nothing and B contains character `a'.

A	A	A	A	A	A	A	A	B	B
									a

Buffer Overflow: Basic Example (Cont.)

- Now, the program attempts to store the string classroom in the A buffer, followed by a `\0' to mark the end of the string. Assume that characters are one byte wide.

A	A	A	A	A	A	A	A	B	B
									a

Buffer Overflow: Basic Example (Cont.)

- Now, the program attempts to store the character string classroom in the A buffer, followed by a `\0' to mark the end of the string. Assume that characters are one byte wide.
- If we do not check the length of the string, it overwrites the value of B

A	A	A	A	A	A	A	A	B	B
`c'	`l`	`a'	`s'	`s'	`r'	`o'	'o'	'm'	0

Buffer Overflow: Example (buffer1.c)

```
#include <stdio.h>

int foo(){
        unsigned int yy = 0;
        char buffer[5]; char ch; int i = 0;
        while ((ch = getchar()) != '\n')
              buffer[i++] = ch;
        buffer[i] = '\0';
        printf("Input: %s\n", buffer);
        printf("yy =  %d\n", yy);
        return 0;
}

int main() {
        while(1) foo(); }
```

7

Buffer Overflow: Example (buffer1.c)

```
#include <stdio.h>

int foo(){
        unsigned int yy = 0;
        char buffer[5]; char ch; int i = 0;
        while ((ch = getchar()) != '\n')
              buffer[i++] = ch;
        buffer[i] = '\0';
        printf("Input: %s\n", buffer);
        printf("yy =  %d\n", yy);
        return 0;
}
int main() {
        while(1) foo();
}
```

bingsun2% buffer1
abcd

8

Buffer Overflow: Example (buffer1.c)

```
#include <stdio.h>

int foo(){
        unsigned int yy = 0;
        char buffer[5]; char ch; int i = 0;
        while ((ch = getchar()) != '\n')
              buffer[i++] = ch;
        buffer[i] = '\0';
        printf("Input: %s\n", buffer);
        printf("yy =  %d\n", yy);
        return 0;
}

int main() {
        while(1) foo();
```

bingsun2% buffer1
abcd
Input: abcd
yy = 0
abcdefghijklmn

9

Buffer Overflow: Example (buffer1.c)

```
#include <stdio.h>

int foo(){
        unsigned int yy = 0;
        char buffer[5]; char ch; int i = 0;
        while ((ch = getchar()) != '\n')
              buffer[i++] = ch;
        buffer[i] = '\0';
        printf("Input: %s\n", buffer);
        printf("yy =  %d\n", yy);
        return 0;
}
int main() {
        while(1) foo();
}
```

bingsun2% buffer1
abcd
Input: abcd
yy = 0
abcdefghijklmn
Input: abcdefghijklmn
yy = 1835925504

10

Buffer Overflow: Example2 (buffer.c)

```
void f(char *str) {
      char buffer[16];
      strcpy(buffer,str);  }
void main() {
      char large_string[256];
      int i; for (i = 0; i < 255; i++)
              large_string[i] = 'A';
      f (large_string);  }
```

11

Buffer Overflow: Example2 (buffer.c)

```
void f(char *str) {
      char buffer[16];
      strcpy(buffer,str);  }
void main() {
      char large_string[256];
      int i; for (i = 0; i < 255; i++)
              large_string[i] = 'A';
      f (large_string);  }
```

- Output: bus error
  - Access memory that the CPU cannot physically address (the return address is overwritten).

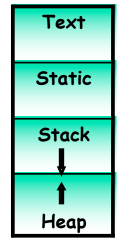
12

What Damage Can Buffer Overflow Cause

- You could run into unpredictable behavior from a program vulnerable to buffer overflow.
- The return address may be overwritten. In most cases, such overwrite of the return address would create a pointer to some invalid location in the memory – segmentation fault.

13

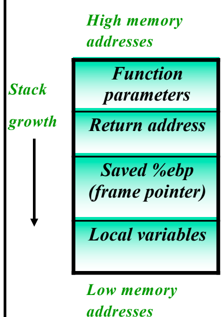
Process Memory Organization



- A process is divided into four regions
  - Text: Program code; read-only. any attempts to write to it will result in segmentation fault.
  - Statically allocated memory (global and static data)
  - Stack: local variables, arguments, return address, etc.
  - Dynamically allocated memory (heap)

14

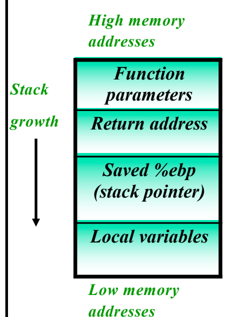
Stack Frames



- Consists of stack frames that are pushed when calling a function and popped when returning.
- Stack pointer (esp) – a register that points to the top of the stack.
- Frame pointer (ebp) – A pointer to the stack frame from which the current stack frame was entered.
- When a function is called, the return address, ebp, and the variables are pushed on the stack.

15

Stack Frames



- The return address has a higher address than the local variables.
- When we overflow the buffer, the return address may be overwritten.

16

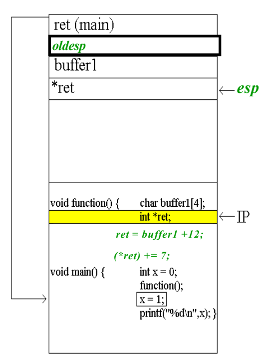
Example:Modify the Execution Flow (attack.c)

```
#include<stdio.h>
void function() {
        char buffer1[4];
        int *ret;
        ret = buffer1 + 12;
        (*ret) += 7;
}

int main() {
        int x = 0;
        function();
        x = 1;
        printf("%d\n",x);
}
```

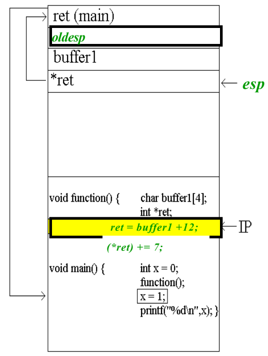
17

Example:Modifying the Execution Flow



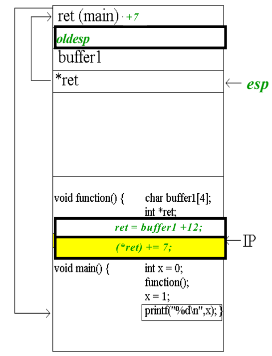
18

Example:Modifying the Execution Flow



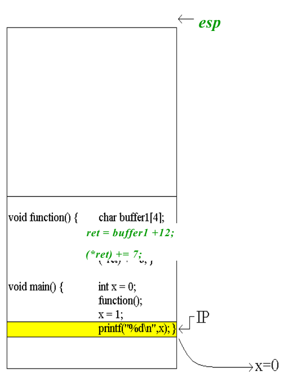
19

Example:Modifying the Execution Flow



20

Example:Modifying the Execution Flow



21

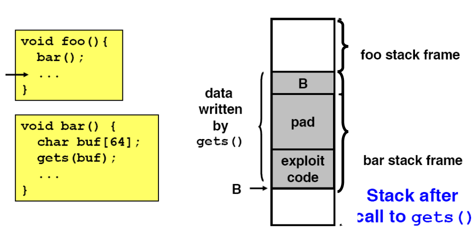
Example:Modifying the Execution Flow

> gdb attack
(gdb) disassemble main
……
0x080483cd : mov %esp,%ebp
……
0x080483d3 : movl $0x0,-0x8(%ebp) //assign 0 to x -- copy 0 to x
0x080483da : call 0x80483a4 <function> //call function f
0x080483df : movl $0x1,-0x8(%ebp) //assign 1 to x
0x080483e6 : mov -0x8(%ebp),%eax //assign the address of x //to eax register
……

22

Buffer Overflow Attack

- Input string contains byte representation of executable code
- Overwrite return address with address of buf
- When bar() executes ret, will jump to exploit code.



23

Real-World Example: Buffer Overflow Attacks

- Attacks exploited buffer overflow vulnerability
  - Code Red Worm
    - Worm: a program that can replicate itself and copies from computer to computer across network connections.
    - Used a long string of the repeated character 'N' to overflow a buffer, allowing the worm to execute arbitrary code and infect the machine.
    - June 18, 2001. Microsoft announces buffer overflow vulnerability in IIS (Internet Information Services) Internet server
    - July 19, 2001. over 250,000 machines running Microsoft's IIS web server are infected by new virus in 9 hours.
    - It waited 20-27 days after it was installed to launch denial of service attacks on several fixed IP addresses.

24

Real-World Example: Buffer Overflow Attacks

- Attacks exploited buffer overflow vulnerability
  - Internet worm: spread by sending code to the finger daemon.
    - Early versions of the finger server used gets() to read the argument sent by the client:
      finger <username>
    - By exploiting the weakness in this system call, the worm was able to load its own code into the finger daemon's read buffer, overwrite the return address in gets()'s stack frame and begin executing itself.

25

Common C and C++ Mistakes that Permit Buffer Overflows

26

Common C and C++ Mistakes that Permit Buffer Overflows

- Fundamentally, any time your program reads or copies data into a buffer, it needs to check that there's enough space before making the copy.
- Sadly, there are a large number of dangerous functions that come with C and C++ that even fail to do this – arrays become the favorite targets of buffer overflow attacks.
  - strcpy, strcat, sprintf, vsprintf, gets, scanf, ……
- Using the format %s in scanf() set of functions is almost always a mistake when reading untrusted input.

27

Counter Buffer Overflows

- Make standard library routines more resistant to attack.
  - Lucent developed libsafe, a wrapper of several standard C functions like strcpy() known to be vulnerable to stack-smashing attacks.
  - The libsafe versions of those functions check to make sure that array overwrites can't exceed the stack frame
  - However, this approach only protects those specific functions, not stack overflow vulnerabilities in general

28

Counter Buffer Overflows

- Canary-based defenses
  - Researcher Crispen Cowan created an interesting approach called StackGuard.
  - Stackguard modifies the C compiler (gcc) so that a canary value is inserted in front of return address.
  - Before any function returns, it checks to make sure that the canary value hasn't changed.
  - If an attacker overwrites the return address, the canary's value will probably change and the system can stop instead.
  - Disable: fno-stack-protector (gcc)

29

Counter Buffer Overflows

- Address Space Layout Randomization (ASLR)
  - ASLR randomly arranges the position of the memory, including the memory position of stack, heap, and libraries.
  - With ASLR, every time when a program executes, the stack will have a different start address.
  - Disable/enable
  - echo 0/1 > /proc/sys/kernel/randomize_vs_space
    - 0 – No randomization. Everything is static.
    - 1 – Shared libraries, stack, mmap(), and heap are randomized.

30

References

- http://www.maths.leeds.ac.uk/~read/bofs.html
- http://www.windowsecurity.com/articles/Analysis_of_Buffer_Overflow_Attacks.html

SQL Injection Attack

http://www.unixwiz.net/techtips/sql-injection.html

What is SQL?

- SQL stands for Structured Query Language
- Is used for many database systems including Microsoft SQL Server, Oracle, MySQL, etc.
- SQL can:
  - Retrieve data from a database *(SELECT)*
  - Insert new records in a database *(INSERT)*
  - Delete records from a database *(DELETE)*
  - Update records in a database *(UPDATE)*
  - Create a new database table *(CREATE TABLE)*
  - Drop a database table *(DROP TABLE)*

SQL Database Tables

- A relational database contains one or more tables identified each by a name
- E.g. a database table "users"

UserID	Name	LastName	Login	Password
1	John	Smith	jsmith	hello
2	Mary	Taylor	mary	qwerty
3	Daniel	Thompson	dthompson	dthompson

- Query:

  SELECT LastName FROM users WHERE UserID = 1;

SQL Database Tables

- A relational database contains one or more tables identified each by a name
- E.g. a database table "users"

UserID	Name	LastName	Login	Password
1	John	Smith	jsmith	hello
2	Mary	Taylor	mary	qwerty
3	Daniel	Thompson	dthompson	dthompson

- Query:

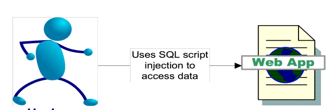
  SELECT LastName FROM users WHERE UserID = 1;

  Result: Smith

What is a SQL Injection Attack?

- Convince the application to run SQL code that was not intended.
- Many web applications take user input from a form.
- Often this user input is used literally in the construction of a SQL query submitted to a database. For example:

  SELECT productdata FROM table
  WHERE productname = 'user input product name'

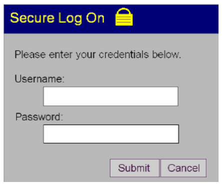


http://www.ja-sig.org/wiki/download/attachments/19378/JASIGWinter2006-Security-Reviews.ppt?version=1

SQL Injection Attack: Example

- SQL Query in Web application code:

SQLQuery = “SELECT * FROM users
WHERE login = ‘” + userName +
“’ and password= ‘” + password + “’;”



SQL Injection Attack: Example

- SQL Query in Web application code:

SQLQuery = “SELECT * FROM users
WHERE login = ‘” + userName +
“’ and password= ‘” + password + “’;”

- Input: **a’ or ‘t’ = ‘t’; --**

SELECT * FROM users
WHERE login = ‘a’ or ‘t’ = ‘t’;
--’; and password=‘’;

’ : Close the user input field.
-- : Comments out the rest of the line.

SQL Injection Attack: Example

- SQL Query in Web application code:

SQLQuery = “SELECT * FROM users
WHERE login = ‘” + userName +
“’ and password= ‘” + password + “’;”

- Input: **a’ or ‘t’ = ‘t’; --**

SELECT * FROM users
WHERE login = ‘a’ or ‘t’ = ‘t’;
--’; and password=‘’;

’ : Close the user input field.
-- : Comments out the rest of the line.

Allow an attacker to bypass the security and retrieve information from the database

SQL Injection Attack: Example

- SQL Query in Web application code:

SQLQuery = “SELECT * FROM users
WHERE login = ‘” + userName +
“’ and password= ‘” + password + “’;”

- **Input:** ***a’ or ‘t’ = ‘t’; DROP TABLE users; —***

SELECT * FROM users
WHERE login = ‘a’ or ‘t’=‘t’;
DROP TABLE users;
--’; and password=‘’;

SQL Injection Attack: Example

- SQL Query in Web application code:

SQLQuery = “SELECT * FROM users
WHERE login = ‘” + userName +
“’ and password= ‘” + password + “’;”

- **Input:** ***a’ or ‘t’ = ‘t’; DROP TABLE users; --***

SELECT * FROM users
WHERE login = ‘a’ or ‘t’=‘t’;
DROP TABLE users;
--’; and password=‘’;

The hacker deletes the users table

Other Injection Possibilities

- Using SQL injections, attackers can:
  - Add new data to the database
    - Could be embarrassing to find yourself selling some strange items on an eCommerce site
  - Modify data currently in the database
    - Could be very costly to have an expensive item suddenly be deeply ‘discounted’

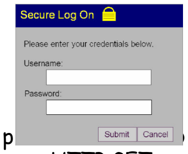
Normal SQL Injection

- The attacker formats his query to match the developer's by using the information contained in the error message that are returned in the response.

43

Example: Normal SQL Injection

- Assume that the site that we will be targeting is http://www.site.com/login.php.



- After we p and the password, hit the button, an HTTP GET request is sent to login.php with the values that we entered (www.site.com/login.php?user=USER&pass=PASS)
- Try to find a SQL Injection flaw.

44

Step 1

- Code:
  http://www.site.com/login.php?user='a&pass='a
- After clicking the submit button, we receive an error page, e.g.

  Microsoft OLE DB Provider for SQL Server error '#######'

  Error on query SELECT id FROM tblUsers WHERE userame = ''a' AND password =''a'
  /login.php, line 31
- SQL server belongs to microsoft → the comment character is --.

45

Step 2

- Now that we know that the site is vulnerable and that we have the original SQL Query we can think of a input that can change the query to do our willing.
- One good query would be:

  SELECT * FROM tblUsers WHERE username = 'admin'--'AND password='a'

46

Step 2

- Now that we know that the site is vulnerable and that we have the original SQL Query we can think of a input that can change the query to do our willing.
- One good query would be:

  SELECT * FROM tblUsers WHERE username = 'admin'--'AND password='a'

  The above query would result into logging in as admin without needing the password.

47

Real World Examples

- It is probably the most common website vulnerability today.
- It is a flaw in web application development, not a DB or web server problem
- On Jan. 13, 2006, hackers broke into a Rhode Island government web site and allegedly stole credit card data from individuals who have done business online with state agencies.
- On June 29, 2007, Hacker defaces Microsoft UK web page using SQL injection
- On Aug. 12, 2007, The United Nations web site was defaced using SQL injection.

48

Defenses

- Check the syntax of input for validity
  - Many classes of input have fixed languages
    - Email addresses, dates, part numbers, etc.
  - Scan query string for undesirable word combinations that indicate SQL statements, e.g. --, select, insert, drop, etc.
- Have length limits on input: many SQL injection attacks depend on entering long strings.

49

Defenses

- Modify Error Reports
  - In error reports, some time full query is shown, pointing to the syntax error involved, and attacker could use it for further attacks.
  - The developer should handle or configure the error reports in such a way that error cannot be shown to outside users.

50

References

- http://www.cert.org.in/knowledgebase/whitepapers/ciwp-2005-06.pdf
- http://www.securitydocs.com/pdf/3348.PDF
- http://fr3dc3rv.blogspot.com/2007/03/sql-injection-part-1.html

51