MOAUB

Abysssec Research

1) Advisory information

Title	: HP OpenView NNM webappmon.exe execvp_nc Remote Code Execution
Version	: OpenView Network Node Manager 7.53
Analysis	: http://www.abysssec.com
Vendor	: http://www.hp.com
Impact	: Critical
Contact	: shahin [at] abysssec.com , info [at] abysssec.com
Twitter	: @abysssec
CVE	: CVE-2010-2703

2) Vulnerable version

HP OpenView Network Node Manager 7.53
HP OpenView Network Node Manager 7.51

3) Vulnerability information

Class

1- Buffer overflow

Impact

An attacker can exploit this issue to execute arbitrary code with SYSTEM-level privileges. Successful exploits will completely compromise affected computers.

Remotely Exploitable

Yes

Locally Exploitable

Yes

4) Vulnerabilities detail

In this section according to the file name and vulnerable function patched and unpatched sections of the execvp_nc function are compared:

```
UnPatch
.text:5A0227D9          push    offset asc_5A04395C ; Dest
.text:5A0227DE          lea     edx, [ebp+CommandLine]
.text:5A0227E4          push    edx             ; Str
.text:5A0227E5          call    strcat_new
.text:5A0227EA          add     esp, 8
.text:5A0227ED          mov     eax, [ebp+var_8004]
.text:5A0227F3          mov     ecx, [ebp+arg_4]
.text:5A0227F6          mov     edx, [ecx+eax*4]
.text:5A0227F9          push    edx             ; Source
.text:5A0227FA          lea     eax, [ebp+CommandLine]
.text:5A022800          push    eax             ; Dest
.text:5A022801          call    strcat_new
.text:5A022806          add     esp, 8
.text:5A022809          cmp     [ebp+var_8004], 1
.text:5A022810          jle     short loc_5A022826
.text:5A022812          push    offset asc_5A043960 ; " "
.text:5A022817          lea     ecx, [ebp+Parameters]
.text:5A02281D          push    ecx             ; Dest
.text:5A02281E          call    strcat_new
.text:5A022823          add     esp, 8
.text:5A022826
.text:5A022826 loc_5A022826:                    ; CODE XREF: execvp_nc+A0j
```

```
.text:5A022826      mov   edx, [ebp+var_8004]
.text:5A02282C      mov   eax, [ebp+arg_4]
.text:5A02282F      mov   ecx, [eax+edx*4]
.text:5A022832      push  ecx          ; Source
.text:5A022833      lea   edx, [ebp+Parameters]
.text:5A022839      push  edx          ; Dest
.text:5A02283A      call  strcat_new
.text:5A02283F      add   esp, 8
.text:5A022842      jmp   short loc_5A02288B
```

```
Patch
.text:5A02283D      lea   edx, [ebp+CommandLine]
.text:5A022843      push  edx          ; Str
.text:5A022844      call  strlen_new
.text:5A022849      add   esp, 4
.text:5A02284C      mov   ecx, 3FFFh
.text:5A022851      sub   ecx, eax
.text:5A022853      push  ecx          ; Count
.text:5A022854      push  offset Source   ; " "
.text:5A022859      lea   edx, [ebp+CommandLine]
.text:5A02285F      push  edx          ; Dest
.text:5A022860      call  ds:strncat
.text:5A022866      add   esp, 0Ch
.text:5A022869      mov   [ebp+var_4001], 0
.text:5A022870      lea   eax, [ebp+CommandLine]
.text:5A022876      push  eax          ; Str
.text:5A022877      call  strlen_new
.text:5A02287C      add   esp, 4
.text:5A02287F      mov   ecx, 3FFFh
.text:5A022884      sub   ecx, eax
.text:5A022886      push  ecx          ; Count
.text:5A022887      mov   edx, [ebp+var_8004]
.text:5A02288D      mov   eax, [ebp+arg_4]
.text:5A022890      mov   ecx, [eax+edx*4]
.text:5A022893      push  ecx          ; Source
.text:5A022894      lea   edx, [ebp+CommandLine]
.text:5A02289A      push  edx          ; Dest
.text:5A02289B      call  ds:strncat
.text:5A0228A1      add   esp, 0Ch
.text:5A0228A4      mov   [ebp+var_4001], 0
.text:5A0228AB      cmp   [ebp+var_8004], 1
.text:5A0228B2      jle   short loc_5A0228E4
.text:5A0228B4      lea   eax, [ebp+Parameters]
.text:5A0228BA      push  eax          ; Str
.text:5A0228BB      call  strlen_new
.text:5A0228C0      add   esp, 4
.text:5A0228C3      mov   ecx, 3FFFh
.text:5A0228C8      sub   ecx, eax
.text:5A0228CA      push  ecx          ; Count
.text:5A0228CB      push  offset asc_5A043998 ; " "
.text:5A0228D0      lea   edx, [ebp+Parameters]
```

```
.text:5A0228D6              push    edx           ; Dest
.text:5A0228D7              call    ds:strncat
.text:5A0228DD              add     esp, 0Ch
.text:5A0228E0              mov     [ebp+var_1], 0
.text:5A0228E4
.text:5A0228E4 loc_5A0228E4:                       ; CODE XREF: execvp_nc+E2j
.text:5A0228E4              lea     eax, [ebp+Parameters]
.text:5A0228EA              push    eax           ; Str
.text:5A0228EB              call    strlen_new
.text:5A0228F0              add     esp, 4
.text:5A0228F3              mov     ecx, 3FFFh
.text:5A0228F8              sub     ecx, eax
.text:5A0228FA              push    ecx           ; Count
.text:5A0228FB              mov     edx, [ebp+var_8004]
.text:5A022901              mov     eax, [ebp+arg_4]
.text:5A022904              mov     ecx, [eax+edx*4]
.text:5A022907              push    ecx           ; Source
.text:5A022908              lea     edx, [ebp+Parameters]
.text:5A02290E              push    edx           ; Dest
.text:5A02290F              call    ds:strncat
.text:5A022915              add     esp, 0Ch
.text:5A022918              mov     [ebp+var_1], 0
.text:5A02291C              jmp     short loc_5A022965
```

As demonstrated above in the unpatched version by calling the strcat_new at address 0x5A0227E5, it adds one of input values for the function at offset asc_5A04395C to a fixed length array that address of the array is in the edx register. In this operation there is no check on the copied value to the fixed length array.

In the patched version by calling strlen_new first the length of the input will be stored in eax and then this value will be substitute from 3FFFh and the result of this operation will be pushed on the stack as the number of copies to the calling of strncat function at address 0x5A022860. And with this value length of the copied string in the fixed length array is checked.

In the unpatched version by calling the strcat_new at address 0x5A022801, two string are concatenated with each other without any check.

In the patched version by calling strlen before strncat at address 0x 5A022877 length of the string is checked.Similar checking conditions are performed in address 0x 5A0228BB , 0x 5A0228EB.

In the following section we have a python script that send a long request to the cgi webappmon.exe. After running the script a stack overflow occurs in the program and the error will be displayed on the screen. This script sends a request based on the ping command to the webappmon.exe. We have used the POST operation in http protocol because of long data.

The proof of concept is attached as poc.py

Here is the result after running the script:

```
fatal error - scanner input buffer overflow
```