UNIVERSITY OF TARTU
Institute of Computer Science

„CVE-2017-7269 vulnerability“

Paper in Computer Security
Elisaveta Saveljev

Supervisor: Meelis Roos

Tartu 2017
INTRODUCTION

CVE-2017-7269 is a Microsoft Windows Server 2003 vulnerability - it means that because of Buffer overflow in the ScStoragePathFromUrl function in the WebDAV service in Internet Information Services (IIS) 6.0 remote attackers can execute arbitrary code via a long header beginning with "If: <http://" in a PROPFIND request, as exploited in the wild in July or August 2016. Microsoft ended support for Windows Server 2003 R2 on July 14, 2015 and is not expected to provide a patch for this vulnerability. [1]

PROPFIND is an HTTP method supported by the Web Distributed Authoring and Versioning (WebDAV) protocol, which is an extension of the HTTP protocol that provides a framework for managing documents on web servers.

Shodan shows that there are a little over 600,000 publicly accessible IIS 6.0 servers on the Internet, and most of them are probably running on Windows Server 2003. Of these, 10 percent has WebDAV enabled to allow for remote web authoring, meaning that there are possibly millions of websites out there exposed to this exploit. [2]

EXPLOITING THE VULNERABILITY

1. **exploit.py** provided by Peng and Wu was simple and effective, and it reliably launched **calc.exe** on a local 32-bit Windows Server 2003. [3]
2. Now we can crash the process at its vulnerable location to launching the calculator. So it is necessary to "dump down" the exploit by replacing all of its shell code with A's (0x41 bytes). Also you can use WinDbg's Global Flags to enable Page Heap on the vulnerable w3wp.exe process to see when the buffer is overflowed.

3. After attaching to w3wp.exe and launching simplified poc.py\(^4\), an access violation was caught due to writing beyond an allocated memory block. The offending instruction was at location httpext!ScStoragePathFromUrl+0x360:

\[
\text{rep movs dword ptr es:[edi],dword ptr [esi]}
\]

4. Now find where the too-small destination buffer pointed to by edi was allocated. For this you can use nifty WinDbg extension !heap -p -a edi, which gave:

```
7c83d6d4 ntdll!RtlAllocateHeap+0x00000e9f
5b7e1a40 staxmem!MpHeapAlloc+0x000000f3
5b7e1308 staxmem!ExchMHeapAlloc+0x00000015
67125df9 httpext!CHeap::Alloc+0x00000017
67125ee1 httpext!ExAlloc+0x00000008
67125462 httpext!HrCheckIfHeader+0x00000013c
6712561e httpext!HrCheckStateHeaders+0x00000010
6711f659 httpext!ICPropFindRequest::Execute+0x000000f0
6711f7c5 httpext!DAVPropFind+0x00000047
671296d2 httpext!CDAVExt::DwMain+0x0000012e
67117bc6 httpext!DwDavFSExtensionProc+0x0000003f
\...
```

5. With this code block, we can see what this resize was all about. It turned out that the local variable [ebp-434h] holds the length (in characters) of the input string, and the above code wants to resize an existing buffer to be able to hold this entire string.

```
push [ebp-434h]
lea ecx, [ebp-430h]
call ?resize@?$CStackBuffer@G$0BAE@@@QAEPAGI@Z
test eax, eax
jz loc_155A2
```

6. Instead of specifying the number of bytes to allocate, the above resize call gets the number of characters - making the resulting buffer much too small.

**SOLUTION**

Instead of pushing [ebp-434h] to stack as an argument to the resize function, we should push that same value multiplied by two and further increased by 2, to accommodate for the string-terminating zero. Replace the push [ebp-434h] instruction with the following:

```
push [ebp-434h]
lea ecx, [ebp-430h]
call ?resize@?$CStackBuffer@G$0BAE@@@QAEPAGI@Z
lea ecx, [ebp-430h]
calc ?resize@?$CStackBuffer@G$0BAE@@@QAEPAGI@Z
```
These changes are not enough because access violation still is present. Now the wrong buffer distribution happens in other place in the code – in httpext!CParseLockTokenHeader::HrGetLockIdForPath.

This flaw is logically identical to the first one, so patch this with effectively the same patch code using an additional patchlet.

Now write a .0pp file for this and built a 0patch using our 0patch Builder, then relaunch poc.py to see if this micropatch fixed the bug.

IIS returned "HTTP/1.1 207 Multi-Status" instead of reporting internal server error, and it is the same when the original exploit.py was launched. The bug was fully patched.

The source code of this micropatch:

```plaintext
MODULE_PATH "C:\WINDOWS\system32\inetsrv\httpext.dll"
PATCH_ID 269
PATCH_FORMAT_VER 2
VULN_ID 2287
PLATFORM win32

patchlet_start
PATCHLET_ID 1
PATCHLET_TYPE 2
PATCHLET_OFFSET 0x00015451
JUMPOVERBYTES 6 ; eliminate the original "push dword ptr [ebp-434h]"

code_start
    mov   eax, dword [ebp-434h]
    lea   eax, [eax+eax+2]
    push  eax

code_end
patchlet_end

patchlet_start
PATCHLET_ID 2
PATCHLET_TYPE 2
PATCHLET_OFFSET 0x0001574b
JUMPOVERBYTES 6 ; eliminate the original "push dword ptr [ebp-22Ch]"

code_start
    mov   eax, dword [ebp-22Ch]
    lea   eax, [eax+eax+2]
    push  eax

code_end
patchlet_end
```
REFERENCES AND SOURCES OF INFORMATION

[1] https://www.youtube.com/watch?v=X5w2HIIYn0


[3] https://github.com/edwardz246003/IIS_exploit


https://github.com/rapid7/metasploit-framework/pull/8162