MS08-067

Sourcefire Vulnerability Research Team Report

The Sourcefire VRT provides information on Microsoft's out-of-band release of MS08-067, including a look at the current known exploits and a review of the Snort and ClamAV protection available.

Matthew Olney, Lurene Grenier and Alain Zidouemba

10/28/2008

MS08-067

Sourcefire Vulnerability Research Team Report

Introduction

On October 23rd, 2008, Microsoft released MS Security Bulletin MS08-067 to cover a vulnerability in the Server service. At the time of the release, Microsoft noted that there had been "limited, targeted attacks attempting to exploit the vulnerability". This document is intended to provide additional information on this vulnerability, including a look at what is assumed to be a 0-day malware sample, a review of H D Moore's released Metasploit module targeting this vulnerability and a look at how Snort provided 0-day coverage against this attack.

Malware Analysis

The Gimmiv.A virus is widely believed to have used a 0-day attack on the MS08-067 vulnerability as a means of distribution. Alain Zidouemba, a VRT analyst with a strong background in malware analysis, and reverse engineering specialist and analyst team lead Lurene Grenier took the lead on reverse engineering the numerous samples received through the ClamAV submission interface. Alain and Lurene's goal was to build ClamAV signatures for the malware, ensure that the Snort coverage was valid for the attack used by the malware and to understand what the malware samples were doing once the infection was successful.

It has been found that a number of files by the name of n[x].exe (where [x] denotes an integer number) have been observed to be a payload of early exploits to the MS08-067 vulnerability. The trojans are usually 388KB in size. Once executed, n[x] starts off by trying to ping the server 202.108.22.44. A whois check of this IP address reveals that the machine is in an IP block allocated to China.

n[x].exe then drops the file sysmgr.dll to %SYSTEMROOT%\system32\wbem\sysmgr.dll and registers the service "System Maintenance Service" that points to sysmgr.dll.

The following registry keys are also created:

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\sysmgr\Parameters
Servicedll = %SystemDir%\wbem\sysmgr.dll
Servicemain = servicemainfunc

Figure 1 shows the "System Mainenance Service" (sysmgr.dll) properties once the service is registered.

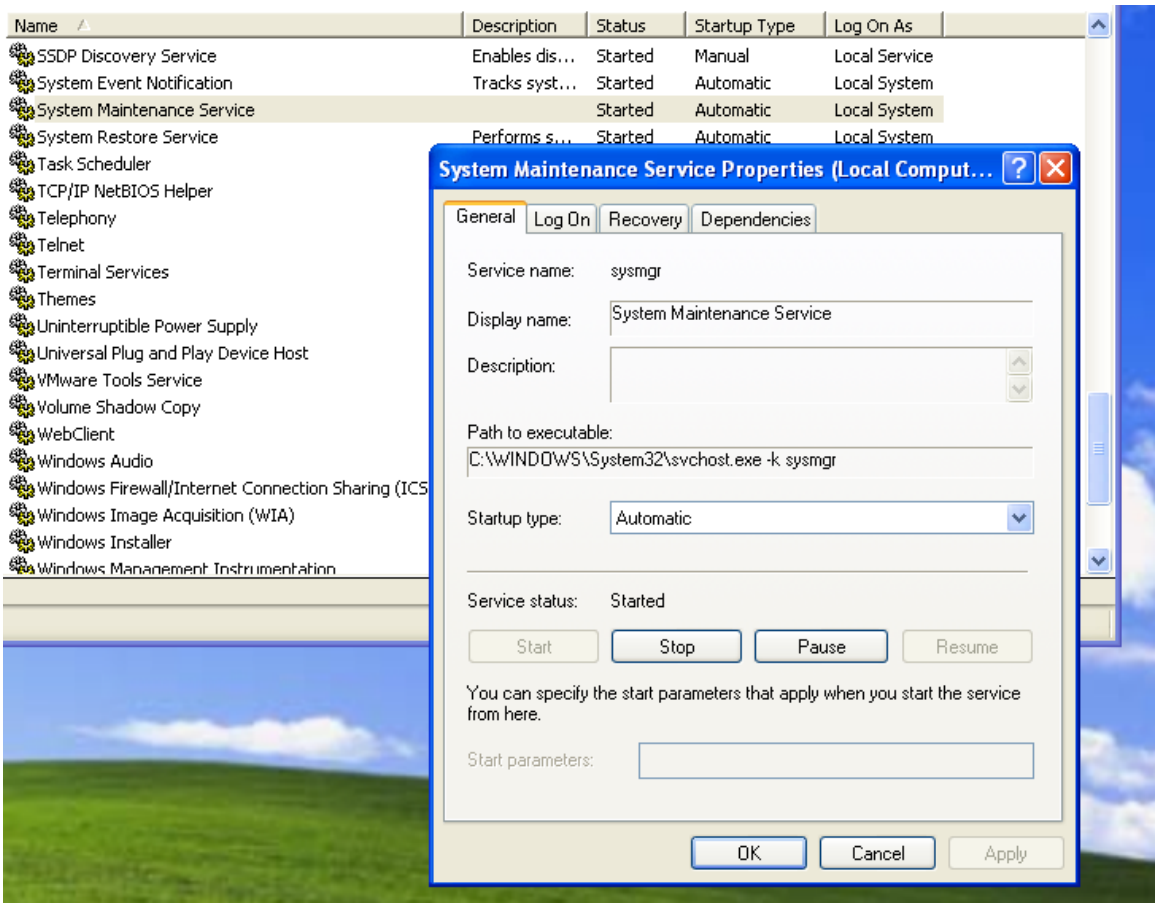

Figure 1: Properties of the registered sysmgr.dll

The file attempts to ping 64.233.189.147 which is an IP assigned to Google. The trojan very likely pings the Google IP to verify network connectivity. The assembly code for sysmgr.dll reveals that the Trojan then checks the registry for the presence of the following registry keys:

HKLM\Software\Jiangmin
HKLM\Software\KasperskyLab
HKLM\Software\Kingsoft
HKLM\Software\Symantec\PatchInst\NIS
HKLM\Software\Microsoft\OneCare Protection
HKLM\Software\rising
HKLM\Software\TrendMicro

All these registry keys are related to prevalent desktop security solutions.

Upon receiving a reply to its PING packet from Google, the trojan has the confirmation that it has access to the Internet and proceeds with an attempt to download additional malware from 59.106.145.58, a server identified as being located in Japan. The following URI is provided while attempting to retrieve the additional software:

test2.php?abc=<***value1***>?def=<***value2***>

The *value1* field is a function of the antivirus software installed on the host and *value2* is a function of the operating system.

The trojan also sends a cookie named *ac* to the server along with the GET request. The cookie contains AES encrypted data about the user. Here is an example of the type of data it attempts to capture:

Outlook Express credentials
Protected Storage credentials
MSN Passport.Net credentials

As a result of the GET request, %SystemDir%\inetproc02x.cab is downloaded to the infected machine. This .cab file contains the following files:

winbase.dll
install.bat
winbaseInst.exe
syicon.dll

Upon the files being extracted from the .cab file, the batch file install.bat copies all the files that were contained in the .cab to the folder %SystemDir%\wbem.

WinbaseInst.exe is then run, which creates another service called "Windows NT Baseline" that points to basesvc.dll. The following registry keys are created in conjunction with the service:

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\BaseSvc\Parameters
Servicedll = %SystemDir%\wbem\ basesvc.dll
Servicemain = servicemainfunc

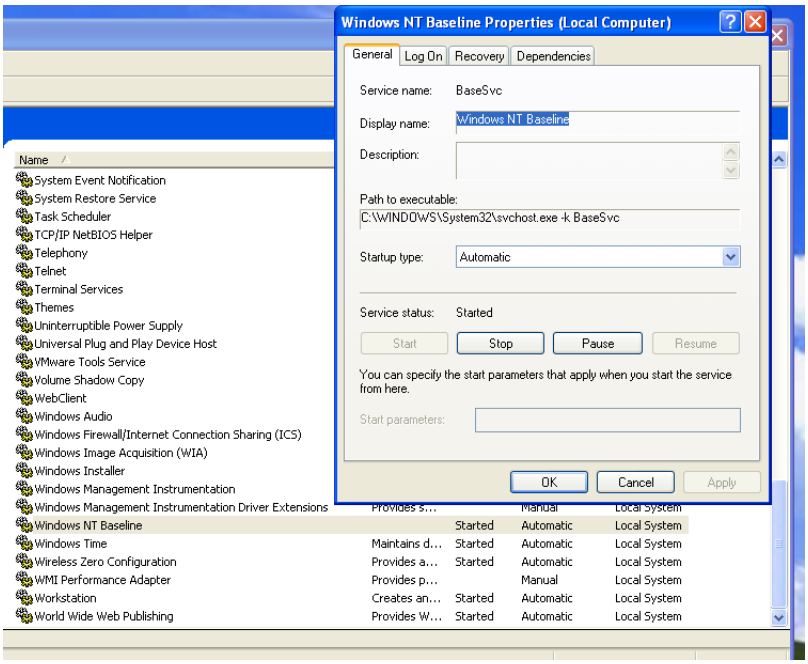

Figure 2: Propereties of the registered basesvc.dll

WinbaseInst.exe is thereafter deleted by install.bat.

We took a look at basesvc.dll in a disassembler (figure 3).

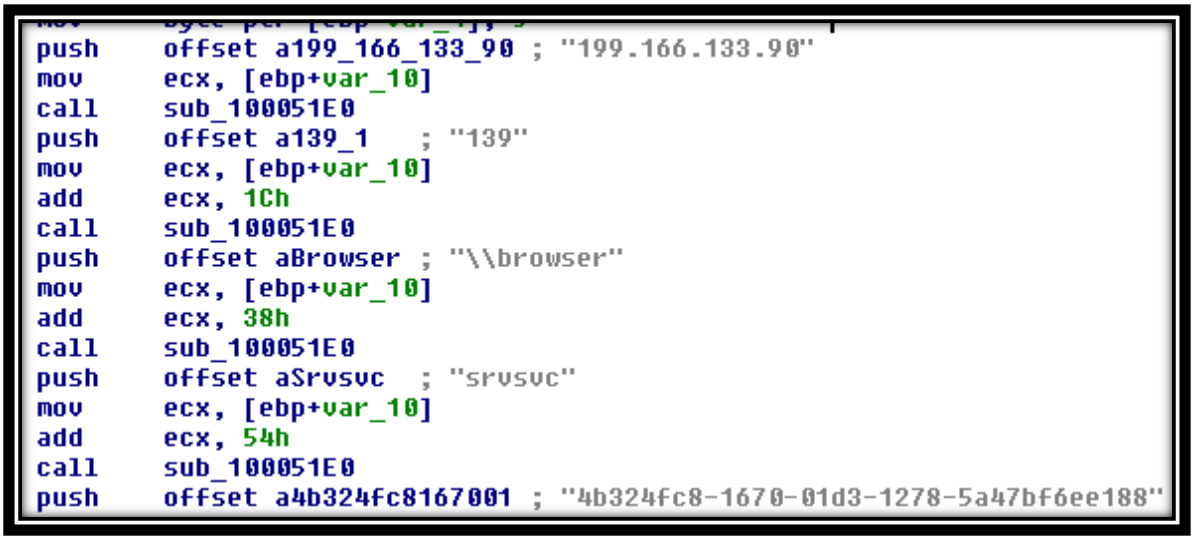

Figure 3: Disassembly of basesvc.dll

We recognized in the ".text" section of this PE file the following strings:

Our research on the vulnerability MS08-067 enables us to easily recognize this strings:

- 4b324fc8-1670-01d3-1278-5a47bf6ee188 is the Universally Unique Identifier, or UUID, that the vulnerable server service registers
- 199.166.133.90 is the IP address assigned to the company "TransCanada Pipeline".
- 139 is the TCP port for Netbios Session Service. This port is commonly used to provide file shares

What we have here is an attempt by this piece of malware to exploit the computer at IP address 199.166.133.90 by sending it malformed DCE-RPC requests and relying on the vulnerable server service that is registered with UUID 4b324fc8-1670-01d3-1278-5a47bf6ee188. However, up to this point, we still not seen traffic on the wire associated with this part of the assembly. Additional work was required to trigger the attack in a timely manner.

Looking at the export table, we found the following functions exported:

DllStartFunc
ServiceMainFunc
DllEntryPoint

Towards the end of the function *ServiceMainFunc*, a subroutine is invoked. This subroutine pauses the execution of the program for a long period of time through the use of a sleep function. This is usually done to make reverse engineering difficult but also so that the trojan will not attract attention immediately upon being executed.

Armed with these three pieces of information (the target IP address, the exported functions and the sleep function), we edited this .dll file with a hex editor to change the sleep time to 1 ms and to change the target of this attack to an internal IP address (we do not want to attack the IP associated with TransCanada Pipeline). Finally, in order to call the desired function in basesvc.dll, we wrote a small custom DLL loader. We ran the *ServiceMainFunc* in our slightly modified .dll and the function began searching for vulnerable Windows machines on our test network. Upon finding a vulnerable Windows machine, the shellcode in the attack used by the trojan is executed on the target machine. This shellcode instructs the newly compromised machine to contact IP address 59.106.145.58 in order to download unspecified software.

ClamAV detects these malicious files under the Trojan.Gimmiv (Trojan.Gimmiv-{1...7}) family.

While covered more fully in the Snort Rules section, we are pleased to report that Snort successfully detects the attack in the 0-day malware, not only with our update rules released on the day of the patch, but also with the MS06-040 ruleset. This provided our customers with 0 day protection against the known attacks using this vulnerability.

Exploit Analysis – H D Moore's Metasploit Module

A number of exploits and proof-of-concepts have been released (and many more have been developed and not released), but H D Moore's implementation of the attack in the Metasploit framework is worthy of a special look because of the implementation in a widely available attack of DEP bypass.

The source code, at the time of this writing, can be found at:

http://metasploit.com/svn/framework3/trunk/modules/exploits/windows/smb/ms08_067_netapi.rb

This module is written for the 3.0 version of Metasploit, available under the 3-clause BSD license at http://www.metasploit.com. The description section lays out both the nature of the attack, and the additional technical hurdle that is addressed:

```
"This module exploits a parsing flaw in the path canonicalization code
of NetAPI32.dll through the Server Service. This module is capable of
bypassing DEP on some operating systems and service packs."
```

Those familiar with the ms06-040 attack in Metasploit will notice that, even though there are many similarities in the attack, there are some changes in the implementation in this module. In particular, in the the 'Targets' section, we not only see the OS specific memory

address of a desired return-to-lib location, we now also have 'DisableNX' for Windows XP SP2 and SP3.

A quick discussion on DEP is probably in order here. DEP stands for "Data Execution Prevention" and is a combination of hardware and software that marks certain memory pages to disable execution of the data there. This means that if the process reaches a state where the next instruction to execute is in, for example, the stack, then an exception will be raised, and if it is not handled, the process will be terminated.

Now, this is a really good idea, but it isn't perfect. It is widely understood that without ASLR (address space layout randomization), DEP is easily defeated. ASLR defeats return-to-lib attacks by randomizing the location and order in which dlls are loaded. Without this randomization, it is relatively straightforward to return to a dll, which can disable DEP or mark pages to permit execution of that data.

Microsoft is keenly aware of this problem, as a blog post from the SVRD (Security Vulnerability Research & Defense) group in Microsoft indicates. From the October, 23rd entry:

"*On Vista and Windows Server 2008, the combination of Address Space Layout Randomization and Data Execution Protection will make the exploitation of this vulnerability more difficult. ASLR will randomize the base address of modules, heaps, stacks, PEB, TEBs, etc. making difficult the return into known locations. Known DEP bypass techniques will not be applicable on these platforms because of the presence of ASLR.*"

Further, a graph in that same blog entry notes that Windows XP SP2 (and SP3) and Windows Server 2003 both implement DEP, but lack ASLR support. Both of these operating systems are marked as remote code execution (RCE) as far as this exploit is concerned, with a note that "DEP without ASLR is bypassable".

It is here that one of Windows Vista and Windows Server 2008 show some of their security strength. In order for an attacker to successfully attack a Vista or 2008 box, they must overcome three obstacles:

1) DEP
2) ASLR
3) Default password protection on vulnerable ports (i.e. Attacker must be authenticated)

H D is well aware of this hurdle, noting in the module that the effort to overcome these obstacles "should be fun". However, even the attacks that are present and overcome the DEP obstacle are interesting to review. In looking at the "def exploit" section, we can follow the module as it builds the attack.

The exploit starts with two commands to establish the network and transport layers necessary to handle the exploit:

connect()
smb_login()

The connect() command is responsible for establishing the TCP connection (by default, on port 445). The smb_login() command establishes the underlying SMB connections necessary to deliver the exploit. This includes the initial negotiation with the SMB_COM_NEGOTIATE command, the session setup via SMB_COM_NEGOTIATE and the initial connection to the IPC$ share with the SMB_COM_TREE_CONNECT_ANDX commands.

The module now begins to build the actual attack string. An important practice that Metasploit module writers tend to use is the randomization of any non-static portion of the attack. This technique ensures that static content signatures cannot be used to detect the attack. In looking at the module, you will see numerous references to Rex::Text.rand_text_alpha and standard ruby rand() calls. These lines are building the randomized content (and in some cases, as in the server name, random length) strings that will be built into the attack.

Before we look at the actual exploitation string, let's look at the methodology of the attack. Because the module first deactivates the DEP with a function in memory, the module actually uses the return-to-lib technique twice. The first it is used to shut off the DEP, and second to hand control of the execution to the address in ESI. In order to do this, the overflow must build a contrived stack structure to handle the return at the end of the DEP deactivation function.

The exploit itself is contained in the path variable. In looking at the lines that build out the path variable, several statements and code comments stand out. Obviously, the payload.encoded statement places the selected payload into the string. Then, the following block of code comes up:

```
# Relative path to trigger the bug
Rex::Text.to_unicode(\ \..\ \..\ \) +
```

According to the code commentary, this is the triggering condition to enter the vulnerable function. From here, we see a randomized pad, and move to the first memory address from the targets variable. "Scratch" points to a writable portion of memory, and will eventually become the stack pointer location. If the payload requires any in-place modification or expansion, the ability to write to the scratch area becomes important.

Next, one of two addresses is written. If the target does not support DEP, then the address of the memory location with "JMP ESI" is written. In this case only a single return to lib is used. However, if DEP has to be disabled first, the address of the disable DEP function is written here instead. This location is the first code execution redirection point.

Then the jump block is written out. The jump block consists of a "\xeb\x62" block, which is a 98 byte jump. Without loading this into a debugger it isn't immediately obvious why this is necessary, but most likely is the part of the final exploitation, during the "JMP ESI". It is also possible that this isn't necessary in all OS versions. Also in the jumper is the final RET location, written early in the jumper block:

```
jumper[ 4,4] = [target.ret].pack("V")
```

This is the memory location that contains the "JMP ESI" instruction, and is the final code execution redirection to occur prior to the execution of the shellcode. Finally there is a terminating \x00 written to the path.

Once the attack string (path variable) is built, the process of executing the attack begins. Handle establishment code block takes care of attaching to the named pipe (Create AndX Request to \BROWSER), and then attaching to the appropriate CLSID, in this case:

```
handle = dcerpc_handle(
        '4b324fc8-1670-01d3-1278-5a47bf6ee188', '3.0',
        'ncacn_np', ["\ \#{datastore['SMBPIPE']}"]
)
```

Once this final piece of connectivity is established, the module is ready to initiate the attack. It builds the stub request so that it can be delivered in a format compatible with the WRITE_ANDX request. The module then sends the attack over the SMB connection, note that the attack will most likely be delivered across multiple packets in the form of SMB fragments. Once the attack is delivered, the module then gracefully handles the disconnection of the handle and the TCP connection.

The Metasploit MS08-067 attack is an excellent example of a mature attack, and is indicative of how quickly a disclosure in binary patch form can go from patch to exploit. By leveraging a standardized methodology (an established SMB/DCERPC library, randomization at available points and the Metasploit framework itself), and an excellent understanding of Windows internals, the exploit was both created quickly (delivered Oct 28th, four days after the vulnerability was patched) and handles multiple versions and protection mechanisms.

Snort Rules

As part of our ongoing coverage analysis, we have recently finished work on the actual pre-patch attack malware and other, post-patch proof of concept and remote code execution attacks released publically. We're pretty pleased with the outcome of our testing. After doing some reverse engineering and writing a quick DLL loader, Alain Zidouemba and Lurene Grenier caused the original, 0-day malware to deliver the ms08-067 attack in a controlled environment. We were able to get a pcap of the attack, and here are the test results from an all-rules load of Snort against that pcap:

```
[0-Day attack]
Alerts:
122:3:0 (portscan) TCP Portsweep
Alerts: 1
1:7218:8 NETBIOS SMB srvsvc
NetrPathCanonicalize WriteAndX little endian
overflow attempt     Alerts: 3
3:14809:1 NETBIOS SMB srvsvc
NetrpPathCononicalize little endian path
cononicalization stack overflow attempt
Alerts: 3
1:7238:8 NETBIOS SMB srvsvc
NetrPathCanonicalize little endian overflow
attempt     Alerts: 3
1:1394:8 SHELLCODE x86 NOOP
Alerts: 42
122:1:0 (portscan) TCP Portscan
Alerts: 5
3:14783:1 NETBIOS DCERPC NCACN-IP-TCP srvsvc
NetrpPathCononicalize little endian path
cononicalization stack overflow attempt
Alerts: 3
```

Because the attackers chose to use the same string that provided the overflow to also deliver the payload, they tripped the overly long string check in our MS06-040 detection. So customers who had protection enabled from either SID 1394 or SID 7238 and SID 7218 had 0-day protection from this attack.

We're happy to see that our newly published rules, based on the Microsoft guidance fired: SID 14809 and SID 14783. But there are a couple of more important things to notice here. We see plenty of evidence of "bad", with 42 alerts from the x86 NOOP rule from SHELLCODE. This would certainly give a skilled analyst a shot of detecting the 0-day attack. But the most important alerts are SID 7218 and SID 7238. These are from our detection set for MS06-040, a vulnerability from the same function as MS08-067. Because the attackers chose to use the same string that provided the overflow to also deliver the payload, they tripped the overly long string check in our MS06-040 detection. So customers who had protection enabled from either SID 1394 or SID 7238 and SID 7218 had 0-day protection from this attack.

After the release of the binary patch, it can take a while to make a reliable exploit, and it wasn't until the weekend following the release of the patch that public exploits began to appear. However, the first proof of concept actually appeared on the same day as the patch release. This proof of concept, found on milw0rm, was fairly interesting. From a quick look at the source code, and in looking at how it behaves against our sensors, we actually believe that this POC is triggering the ms06-040 vulnerability, as opposed to the MS08-067 one. Here is the output from our tests on the PoC pcap:

```
[Milw0rm PoC]

Alerts:
3:14817:1 NETBIOS SMB srvsvc NetrpPathCononicalize unicode little
endian path cononicalization stack overflow attempt     Alerts: 1
1:1923:9  RPC portmap proxy attempt UDP   Alerts: 2
1:7224:8  NETBIOS SMB-DS srvsvc NetrPathCanonicalize unicode little
endian overflow attempt     Alerts: 1
3:14783:1 NETBIOS DCERPC NCACN-IP-TCP srvsvc NetrpPathCononicalize
little endian path cononicalization stack overflow attempt     Alerts:
1
```

The first Milw0rm remote entry didn't show up until the October 26th, here is the output from testing on that pcap:

```
 [Milw0rm Remote]

Alerts:
3:14817:1 NETBIOS SMB srvsvc NetrpPathCononicalize unicode little
endian path cononicalization stack overflow attempt     Alerts: 1
1:7209:8  NETBIOS SMB srvsvc NetrPathCanonicalize unicode little endian
overflow attempt     Alerts: 1
3:14783:1 NETBIOS DCERPC NCACN-IP-TCP srvsvc NetrpPathCononicalize
little endian path cononicalization stack overflow attempt     Alerts:
1
```

Again, we detect both on the MS06-040 rules and our newly provided MS08-067 rules. Because the attackers chose to put payload in the same string as the attack, they trigger the overflow check in our prior MS06-040 rules. This won't always be the case, as payload can be placed outside the stub, but we would still alert on the "path cononicalization stack overflow attempt" rules that target the ms08-067 vulnerability.

We've also verified coverage against Core Security Technologies' Core Impact module that exploits this vulnerability. Not surprisingly, this was a reliable attack mechanism against machines that are shown by Microsoft's documentation to be vulnerable to this attack. Again, it triggers prior coverage on our MS06-040 rules:

```
[Core Impact]

Alerts:
1:1390:6  SHELLCODE x86 inc ebx NOOP   Alerts: 2
3:14809:1 NETBIOS SMB srvsvc NetrpPathCononicalize little endian path
cononicalization stack overflow attempt   Alerts: 2
1:7235:8  NETBIOS SMB-DS srvsvc NetrPathCanonicalize little endian
overflow attempt     Alerts: 2
1:1923:9  RPC portmap proxy attempt UDP Alerts: 10
1:648:9   SHELLCODE x86 NOOP  Alerts: 2
3:14783:1 NETBIOS DCERPC NCACN-IP-TCP srvsvc NetrpPathCononicalize
little endian path cononicalization stack overflow attempt     Alerts:
2
```

Since this is a well-formed attack, we see a couple of additional things in this detection. The RPC portmap alert is triggered as Core Impact determines the attack vectors available to it. We also see indications of NOP sleds through SID 1390 and SID 648. Depending on how the attacker chooses to lay out the attack, NOP sleds can be an important component of attack detection, particularly in 0-day cases. I would strongly recommend that you leave active as many SHELLCODE rules as you can, to give your analysts a chance in 0-day cases. In this case though, we have solid detection, both in the form of SID 7235, our MS06-040 detection, and our MS08-67 specific set of detection.

Finally, we have the Metasploit attack. We've covered it in the Exploit Analysis section. Here are the detection results from the pcap:

```
[Metasploit]

Alerts:
3:14793:1 NETBIOS SMB srvsvc NetrpPathCononicalize WriteAndX little
endian path cononicalization stack overflow attempt Alerts: 1
1:7250:8 NETBIOS SMB-DS srvsvc NetrPathCanonicalize WriteAndX little
endian overflow attempt Alerts: 2
3:14783:1 NETBIOS DCERPC NCACN-IP-TCP srvsvc NetrpPathCononicalize
little endian path cononicalization stack overflow attempt Alerts: 1
```

Once again, we have coverage via the ms06-040 ruleset. Also, note the absence of other identifying traffic (shellcode, scanning, etc...) without targeted coverage, you would miss this attack.

As a final take away, notice that across these attacks, we have several different alerts. Because of the time Brian Caswell and the VRT team have put towards handling a variety of NetBIOS attack vectors; Snort is able to provide broad, comprehensive detection for attacks. This leads to a larger number of rules to provide this coverage, so I strongly encourage you to

upgrade to at least Snort 2.8.2, when we introduced the binary tree structure to the rule parsing process. This provides a significant performance increase to the large, but well formed for a binary tree, NetBIOS ruleset.

Conclusion

By looking at the timeline and releases associated with ms08-067 a small glimpse of the overall threat landscape can be seen. On the defensive side, Microsoft was able to react quickly to a 0-day threat and worked with their partners in the MAPP program to provide quality IPS/IDS coverage on the same day as the patch release. The vulnerability development teams and individual researchers were quick to react to the patch. Researchers reported control of EIP within hours of the patch release, and Core Impact had a commercially viable release available for download two days after the patch release. The Metasploit project delivered a quality, open-source attack against multiple platforms five days after the patch.

Looking broadly at this information, it becomes apparent that speed and agility is a necessary part of security. The window between patch release and vulnerability availability has shrunk dramatically, to the point that administrators are balancing a conservative approach to maintain the integrity of their systems and the aggressive deployment of patches to prevent compromise of those same systems. One option available to administrators is to deploy a Snort or Sourcefire inline IPS, loaded with VRT subscribed rules to provide protection across the network to buy time for required testing of patches. Administrators must generate a plan for handling the emergent, critical threats to their infrastructure.

References

A detailed description of the Data Execution Prevention (DEP) feature in Windows XP Service Pack 2, Windows XP Tablet PC Edition 2005, and Windows Server 2003. (2006, September 26). Retrieved October 28, 2008, from support.microsoft.com: http://support.microsoft.com/kb/875352

More detail about MS08-067, the out-of-band netapi32.dll security update. (2008, October 23). Retrieved October 27, 2008, from blogs.technet.com: http://blogs.technet.com/swi/archive/2008/10/23/More-detail-about-MS08-067.aspx

Microsoft Server Service Relative Path Stack Corruption.(2008, October 27). Retrieved October 28,2008, from Metasploit.com: https://metasploit.com/ms08_067_netapi.rb

Microsoft Security Bulletin MS08-067 – Critical. (2008, October 23). Retrieved October 23, 2008, from www.microsoft.com: http://www.microsoft.com/technet/security/Bulletin/MS08-067.mspx

[MS-SMB]: Server Message Block (SMB) Protocol Specification. (2007, September 28) Retrieved October 29, 2008 from download.microsoft.com: http://download.microsoft.com/download/9/5/E/95EF66AF-9026-4BB0-A41D-A4F81802D92C/%5BMS-SMB%5D.pdf

CIFS Protocol. (2004, June 10). Retrieved October 29, 2008 from www.microsoft.com: http://www.microsoft.com/downloads/details.aspx?FamilyID=c4adb584-7ff0-4acf-bd91-5f7708adb23c&displaylang=en