PriveonLabs Research

Cisco Security Agent Protection Series:

The MS06-040 Vulnerability

Buffer Overrun in Server Service

Chad Sullivan
Senior Security Consultant

Overview

On August 8, 2006 Microsoft announced a buffer overrun in the Server Service on various platforms. This is documented in MS Security Bulletin MS06-040 (KB921883). The Server Service provides RPC services for File, Print and Named Pipe Network Sharing. This vulnerability is a result of an unchecked buffer in unpatched versions of various Microsoft operating systems.

CVE: *CVE-2006-3439*
IMPACT: *Remote Code Execution*
AFFECTED SOFTWARE:
Windows 2000 SP4
Windows XP SP1-2
Windows XP x64
Windows Server 2003 SP0-1
Windows Server 2003-Itanium SP0-1
Windows Server 2003 x64

This vulnerability can allow a remote attacker to gain full control of the remote system. The remote attacker can, after exploitation, have total control including permissions alteration, software installation, and file manipulation. Because of the potential impact of this vulnerability it is recommended that you update your systems immediately.

The purpose of this document is to explain and expand upon the PoC exploits available as well as how the Cisco Security Agent (CSA) product can protect systems from this day-zero vulnerability.

Exploit PoC Code Exploit Process Overview

CVE-2006-3439 is a remotely exploitable attack with various malware variants currently propagating in the wild. This attack can occur as part of a directed attack over private networks or remotely executed over the public Internet. It can also be exploited as part of a multistage worm attack or other variants.

Within 48 hours of Microsoft posting the public version of the MS Security Bulletin, various websites published Proof of Concept (PoC) exploit code. The most prominent source of this PoC code is what is currently available via the Metasploit Project (www.metasploit.org).

PoC Exploit Testing Overview

Testing Environment

The Priveon test lab consisted of several Operating Systems and application combinations including various patch revision levels running on virtual hosts. The purpose of the testing process is to confirm prevention of payload delivery through the Cisco Security Agent. The testing involved concept code obtained from Metasploit.com using various shellcode payloads such as win32_bind, win32_reverse, and win32_adduser which can be obtained via the Metasploit project.

Cisco Security Agent Version Tested:

- CSA 5.0.0.187 (Default CSA Policies as shown in Figure 1)
- CSA 5.1.0.69

Figure 1: Protected Host CSA Policies from v5.0.0.187

Group Name	Version	Description	Policies
<All Windows>		Auto-enrollment group for Windows hosts	2 policies

Policy Name	Version	Description	Rule Modules
Application Classification	5.0 r187	Base policy for behavioral classification of applications.	3 modules
Operating System - Base Permissions - Windows	5.0 r187	Basic permissions for Windows OS	2 modules

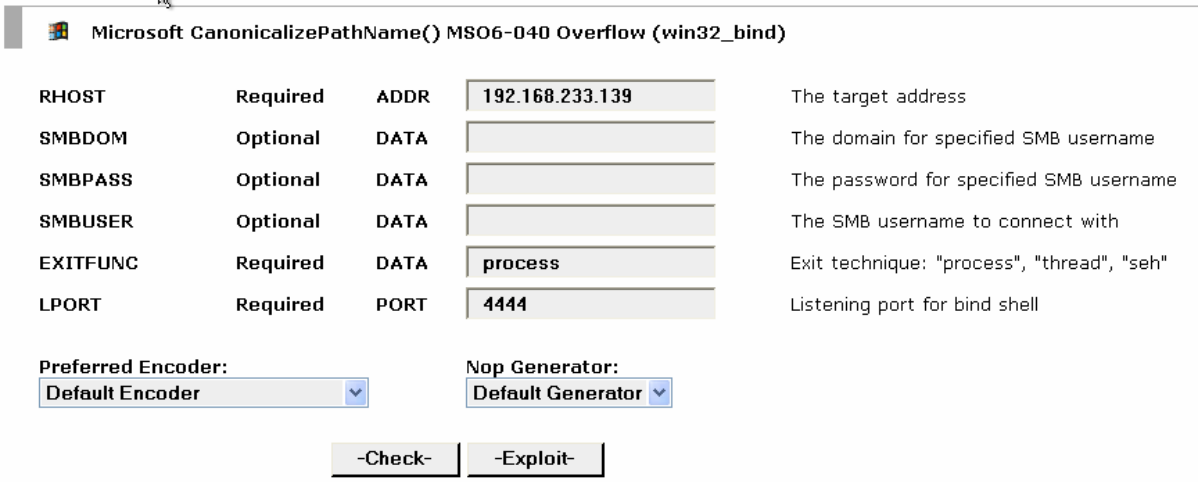
Group Name	Version	Description	Policies
Desktops - All types	5.0 r187	Default group for systems that install the Desktop agent kit	9 policies

Policy Name	Version	Description	Rule Modules
Agent UI control	5.0 r187	Policy which governs Agent User Interface	1 module
Document Security - Windows	5.0 r187	Policy to protect user documents	1 module
Email Client - Basic Security - Windows	5.0 r187	Basic application enforcement policy for email client software.	3 modules
General application - Basic Security - Windows	5.0 r187	Basic, Application independent security policy for Windows	3 modules
Installation Applications - Windows	5.0 r187	Software Installers for Windows	4 modules
IP Stack - Internal Network Security	5.0 r187	Policy for protecing the IP Stack on internal systems	1 module
Network Personal Firewall	5.0 r187	Control network access and provide some end user access controls.	1 module
Operating System - Base Protection - Windows	5.0 r187	Basic protection for Windows OS	6 modules
Virus Scanner - Windows	5.0 r187	Application enforcement policy for virus scanner software.	1 module

Exploit PoC Testing Results – Unprotected System

The initial testing of the MS06-040 vulnerability was via Metasploit's win32_bind payload and Microsoft CanonicalizedPathName() Overflow Exploit. The test utilized standard Metasploit configuration with the only test specific configuration related to the remote target IP address of 192.168.233.139 as shown in figure 2. The win32_bind payload, when successful, results in a remotely accessible command shell listening on the predetermined port (in this case TCP/4444).

Figure 2: Metasploit win32_bind shell configuration



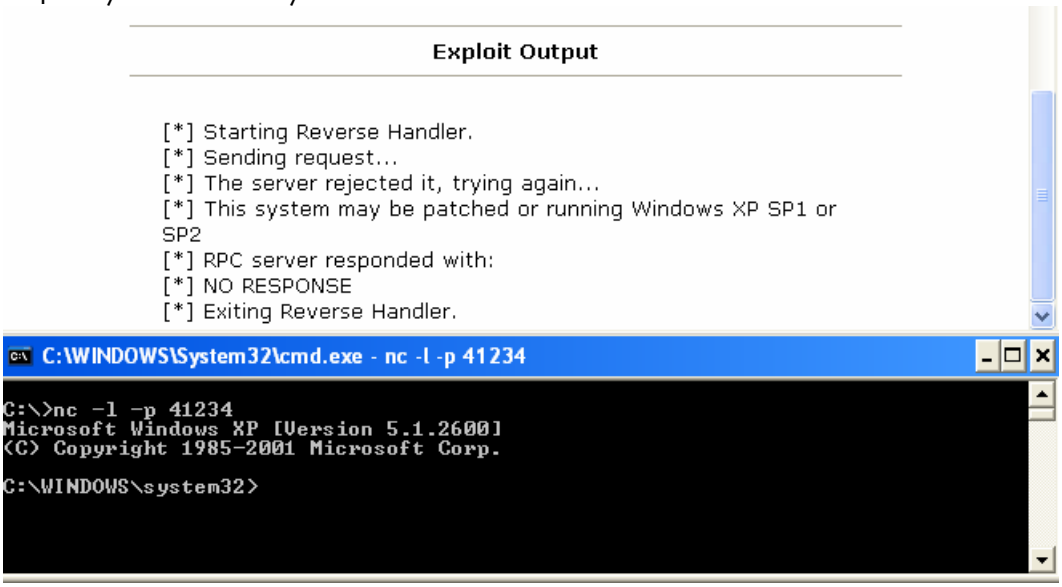
Once configured, our test proceeded by simply clicking on the –Exploit- button on the Metasploit configuration webpage. This resulted in the remotely connected and fully usable command shell as displayed in Figure 3. At this point the remote attacker whether on the local subnet or halfway around the world could proceed by adding system accounts, modifying the target system files and/or installing various software packages such as Trojans, spyware, and other malicious services.

Figure 3: Payload delivery results in listening command shell on victim host

```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\WINDOWS\system32>
>> dir *.exe

dir *.exe
 Volume in drive C has no label.
 Volume Serial Number is F8D8-6E3D

 Directory of C:\WINDOWS\system32

07/07/2003  08:00 AM           179,200 accwiz.exe
07/07/2003  08:00 AM             4,096 actmovie.exe
07/07/2003  08:00 AM            91,648 ahui.exe
07/07/2003  08:00 AM            41,984 alg.exe
07/07/2003  08:00 AM            12,498 append.exe
07/07/2003  08:00 AM            19,456 arp.exe
```

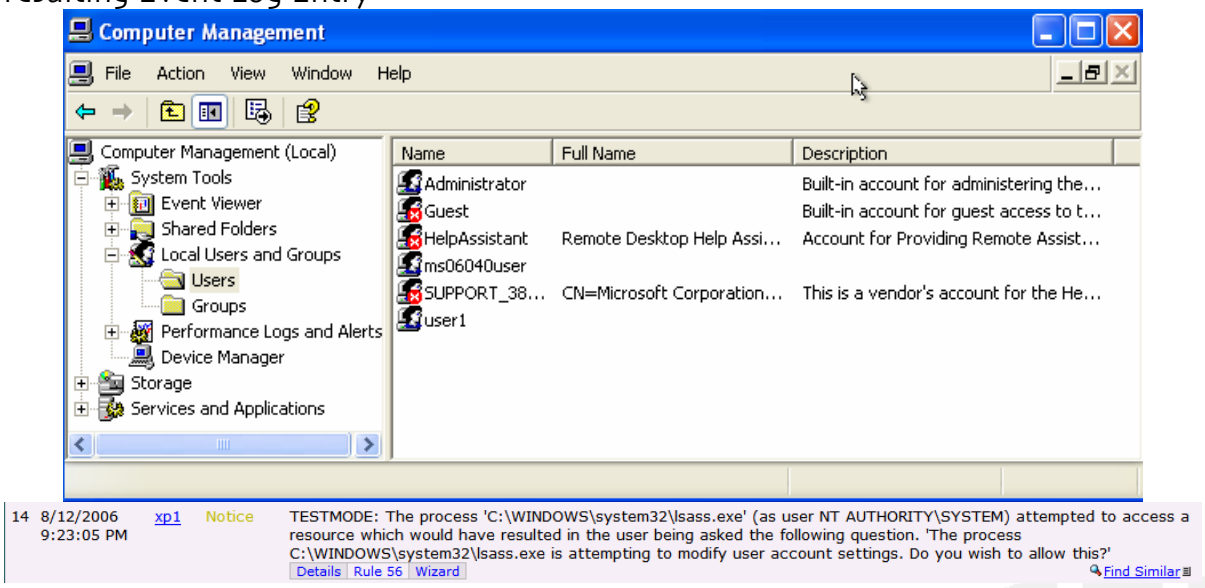
The results for the other tested payloads were also successful in exploiting the target system remotely. The Win32_reverse payload, when implemented, forces the remotely exploited system to connect back to our own listener. In this case we utilized NetCat (nc) for this purpose listening on TCP/41234 and did receive the connect-back shell which was once again fully usable. The win32_reverse shell achieved in our testing is displayed in figure 4.

Figure 4: Payload delivery results a reverse connection back to the attacker



The final payload test attempted was the Metasploit win32_adduser payload. This payload attempts to add an administrative user to the remote system that can then be utilized for remote access to the exploited target during subsequent connections. During our testing we configured the win32_adduser payload to create a remote account called ms06040user which was successful and can be seen in figure 5.

Figure 4: Successful win32_adduser payload delivery to vulnerable system and resulting Event Log Entry

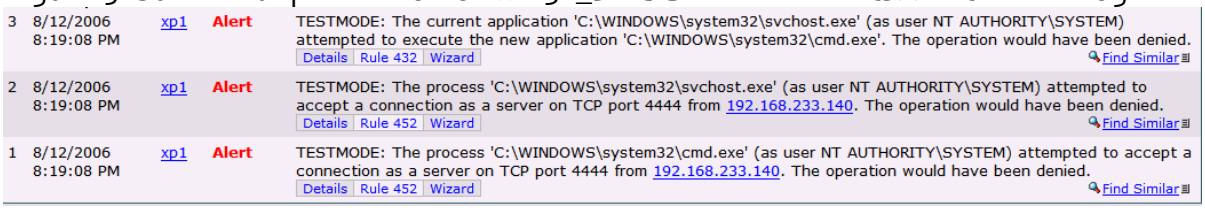


CSA Protected Exploit PoC Testing Results

After proving various attack vectors were successful and that our systems were vulnerable to the exploits available on the Internet, we proceeded by installing the Cisco Security Agent Product on the target hosts to illustrate the successful prevention of the exploit on unpatched hosts.

Our first test was to attempt the win32_bind payload exploitation of the remote host. This test resulted in the prevention of the TCP/4444 attempted connection via the default CSA policies. By default, CSA, with standard policies, prevents inbound access to the system for all processes attempting to act as servers. This allows the CSA administrator to define what services should be allowed to service remote connections as per the corporate security standard policy for personal firewalls. In addition to preventing this connection, if the attack would have been allowed to proceed, such as would be the case if the CSA administrator had tuned this remote access as acceptable, CSA would still have prevented the process acting as a network server from starting a command shell (cmd.exe). This is also once again standard default CSA policy. Figure 5 illustrates these denied actions. You should also notice in Figure 5 that the test system was in TESTMODE to illustrate each action that would have been prevented throughout the lifecycle of the attack even if certain default CSA policies had been disabled.

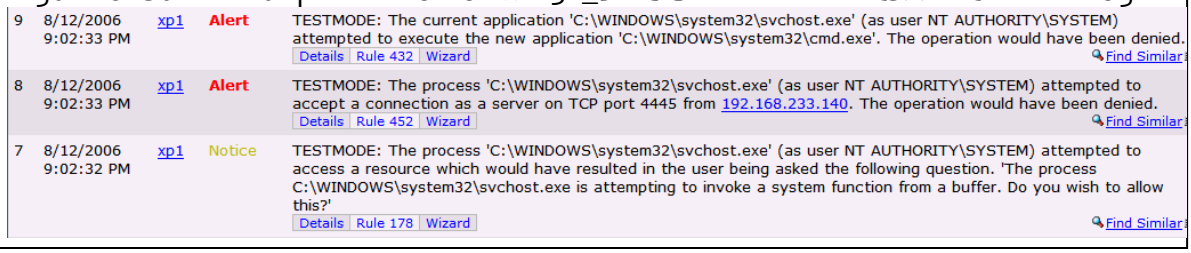
Figure 5: Successful prevention of win32_bind as seen in the CSA MC Event Log



3	8/12/2006 8:19:08 PM	xp1	Alert	TESTMODE: The current application 'C:\WINDOWS\system32\svchost.exe' (as user NT AUTHORITY\SYSTEM) attempted to execute the new application 'C:\WINDOWS\system32\cmd.exe'. The operation would have been denied. Details Rule 432 Wizard Find Similar
2	8/12/2006 8:19:08 PM	xp1	Alert	TESTMODE: The process 'C:\WINDOWS\system32\svchost.exe' (as user NT AUTHORITY\SYSTEM) attempted to accept a connection as a server on TCP port 4444 from 192.168.233.140. The operation would have been denied. Details Rule 452 Wizard Find Similar
1	8/12/2006 8:19:08 PM	xp1	Alert	TESTMODE: The process 'C:\WINDOWS\system32\cmd.exe' (as user NT AUTHORITY\SYSTEM) attempted to accept a connection as a server on TCP port 4444 from 192.168.233.140. The operation would have been denied. Details Rule 452 Wizard Find Similar

NOTE: You may notice that the CSA default policies did prevent the network server from allowing inbound connections and also prevented the spawning of the command shell. It did not however catch the buffer overflow as part of the default policy. By default, CSA does not deny all buffer overflows. It will prevent buffer overflows in any process that attempts to communicate on the network. Since this attack targets the MS Server Service, if this system had been remotely connected to (or even attempted to be connected to) via another host, the Server Service would have been tagged by the default policy and the actual attack would have been prevented at that point. In our lab environment, we did not have any other adjacent network connection attempts occurring so the initial buffer overflow was allowed.
To illustrate this point, we added a share to our remotely exploitable target host and prior to reattempting the win32_bind payload, initiated a simple connection to the remote share which immediately tagged the server service as a network application and therefore applied buffer overflow protection to it as seen in figure 6.

Figure 6: Successful prevention of win32_bind as seen in the CSA MC Event Log



9	8/12/2006 9:02:33 PM	xp1	Alert	TESTMODE: The current application 'C:\WINDOWS\system32\svchost.exe' (as user NT AUTHORITY\SYSTEM) attempted to execute the new application 'C:\WINDOWS\system32\cmd.exe'. The operation would have been denied. Details Rule 432 Wizard Find Similar
8	8/12/2006 9:02:33 PM	xp1	Alert	TESTMODE: The process 'C:\WINDOWS\system32\svchost.exe' (as user NT AUTHORITY\SYSTEM) attempted to accept a connection as a server on TCP port 4445 from 192.168.233.140. The operation would have been denied. Details Rule 452 Wizard Find Similar
7	8/12/2006 9:02:32 PM	xp1	Notice	TESTMODE: The process 'C:\WINDOWS\system32\svchost.exe' (as user NT AUTHORITY\SYSTEM) attempted to access a resource which would have resulted in the user being asked the following question. 'The process C:\WINDOWS\system32\svchost.exe is attempting to invoke a system function from a buffer. Do you wish to allow this?' Details Rule 178 Wizard Find Similar

The details of the prevented buffer overflow as part of the initial attack once the server service was tagged as a network application is seen in figure 7 as supporting data.

Figure 7: Buffer Overflow Details as illustrated in the CSA MC Event Log

PString3	The process C:\WINDOWS\system32\svchost.exe is attempting to invoke a system function from a buffer. Do you wish to allow this?
args(6)	LoadLibraryA
time	1793.5 (seconds since boot)
type	APICALL
ProcessId	1824
ApiOperation	BufferOverflowDetected
Credentials	os=win32,T=NT AUTHORITY\SYSTEM,t=0101000000000000512000000,G=NT AUTHORITY\SYSTEM,g=0101000000000000512000000
ApiPInt1	9828105
ApiPString1	ffd66653 66683332 68777332 5f54ffd0 68cbedfc 3b50ffd6 5f89e566 81ed0802
ApiPString2	LoadLibraryA
ApiPInt2	9824488
ApiPString3	49706c73 09f79500 f4e89500 7773325f 33320000 0000e677 8e4e0eec ebffffff 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
args(4)	ws2_32

Summary of Results

CSA prevented exploitation of systems vulnerable to the MS06-040 vulnerability using default CSA policies in all testing scenarios utilizing various exploitation payloads. The success was proven for the default PoC code which is readily available via Metasploit.com.

It is important that anyone managing a CSA deployment thoroughly understand the policy they have chosen to deploy on their protected systems. By default, only certain applications and programs that communicate over the network are protected by buffer overflow mechanisms (System API Rules). If your systems server service had not yet attempted to communicate on the network, the default CSA policies would not have prevented the initial buffer overflow but the policy would have prevented the attempted resulting shell connections and/or user creation. This is a result of CSA default policies demonstrating the true defense-in-depth nature of the Cisco Security Agent.

References

- http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2006-3439
- http://metasploit.com
- http://netcat.sourceforge.net/

Where to Go for More Information:

Custom Research Documents
Exploit Reverse Engineering/Forensics
Security Implementation
Real-World Training
Managed Services

Available @ www.Priveon.com