How to Avoid Information Disclosure when Managing Windows with WMI

Alex Timkov

How to Avoid Information Disclosure when Managing Windows with WMI

GSEC Gold Certification

Author: Alex M. Timkov, atimkov@gmail.com

Adviser: John C. A. Bambenek

Accepted: April 2007

Table of Content

List of Figures

1. Introduction

Windows Management Instrumentation (WMI) is a widespread mechanism of managing Windows infrastructure. WMI is convenient. It gives Windows administrators and developers easy access to a wealth of system objects and object properties, allowing for relatively easy automation of administrative tasks and monitoring of the operating system, and in many cases applications and hardware. The convenience of WMI is a compelling reason for adopting WMI as a way of monitoring and managing Windows infrastructure. Many vendors have implemented their Windows monitoring and management solutions by using WMI. Before you do, it is worth taking a quick look at some of the security aspects of the WMI scripting mechanism. It is all too often that Windows provides the security features that can be used effectively, but fails to supply the default configuration that would benefit from own security features that are available and are waiting to be of service. WMI is no exception.

This paper provides an introduction to accessing Windows via WMI in a secure manner. After introducing the subject of WMI security, we will demonstrate how the default WMI access level leads to unnecessary exposure of rather sensitive information, as management data travels between the management station and the Windows hosts that are being managed via WMI. We will make recommendations on using WMI to manage remote Windows hosts securely, without exposing the sensitive management session

information. We will demonstrate how very simple and effective measures can stop unnecessary information leaks and boost management access security.

2. Brief Introduction to Windows Management Interface Access

2.1. Using the Object Method to Access WMI

WMI is most frequently accessed with VBScript due to the simplicity of VBScript code. Other scripting languages may be used to access WMI, however in most cases more lines of code may be required to achieve the same purpose. Another reason for the widespread use of VBScript for scripting WMI connectivity is the large number of classes and objects immediately accessible for retrieval. Last and not least, VBScript is included with any modern Windows operating system. Provided you do not mind using Notepad to edit text, a simple development environment and interpretation is taken care of, without the requirement for additional software. All of the examples in this paper are written in VBScript.

The goal of this paper is to focus on the security issues associated with WMI, and to suggest workarounds to these issues. This paper is not a WMI scripting guide. However, initial understanding of WMI access techniques is required to follow further sections of the paper. This section provides such basic introduction to accessing WMI with VBScript.

Here is the sample script that was used to perform some of the testing during the

research phase. This script can be used to start a service on a remote system:

```
'---------- Service Start Script - WMI Based ----------
'
' Test the state of the service of user choice on the remote host
' Start the service on the remote host if the service is not running
'
' Written by Alex Timkov to facilitate SANS GSEC research paper
'------------------------------------------------------------

Option Explicit

Dim strComputer, strPassword, strService, strUser
Dim objLocator, objService, objWMIService
Dim colServices
Dim intTimeout

intTimeout = 10000

Do
  strComputer = inputbox _
    ("Please enter host name or IP address", "Input")
Loop until strComputer <> ""

Do
  strUser = inputbox("Please enter username", "Input")
Loop until strUser <> ""

Do
  strPassword = inputbox("Please enter password", "Input")
Loop until strPassword <> ""

strService = inputbox _
  ("Please enter the name of service to start, " _
    & "or hit Enter to exit", "Input")

If strService <> "" Then

  Set objLocator = CreateObject("WbemScripting.SWbemLocator")

  Set objWMIService = objLocator.ConnectServer _
    (strComputer, "root\CIMV2", strUser, strPassword)

  Set colServices = objWMIService.ExecQuery _
    ("Select * from Win32_Service Where Name = '" _
      & strService & "'")

    If Not(colServices Is Nothing) Then
      For Each objService in colServices
        If Not(objService Is Nothing) Then
          If objService.State = "Stopped" Then
            Wscript.Echo vbCrlf & "Starting " & strService _
              & " service..."
```

```
            objService.StartService()
            WSCript.Sleep intTimeout
          Else
            Wscript.Echo vbCrlf & "Service is not stopped."
          End If
        End If
      Next
    End If

  Set objService = Nothing
  Set colServices = Nothing
  Set objWMIService = Nothing
  Set objLocator = Nothing

End If

WScript.Echo vbCrlf & "Script execution completed. Exiting."
WScript.Quit

'---------------------------------------
' End of WMI Based Service Start Script
'---------------------------------------
```

Figure 1: WMI Service Start Script

The WMI connection establishment and object retrieval starts mid-way through the sample script. The script is of 73 lines, and the first WMI specific line is line 37:

```
Set objLocator = CreateObject("WbemScripting.SWbemLocator")
```

Prior to this line, the code is non WMI specific VB Script: declarations and dialogs to accept user input of the remote system name or IP address, the name of the user authorised to carry out the remote operation, the user password, and the name of the service to be tested for current state and started if it is found to be in a stopped state.

The line referenced above does nothing else but accessing the standard object named SWbemLocator. The next line makes use of the method called ConnectServer to connect to the standard WMI namespace root\CIMV2 on the remote host specified by

strComputer, using the user account strUser with the account password strPassword:

```
Set objWMIService = objLocator.ConnectServer(strComputer, "root\CIMV2", strUser, strPassword)
```

objWMIService.Security_ property can then be used to set specific security parameters for the WMI namespace connection. We will cover WMI security parameters in more detail in the following sections. We will stick with the default WMI security values for the moment. It is one of our targets to see what is happening across a WMI management connection when the default WMI security settings are used.

Once the WMI namespace connection has been established, we can retrieve the objects that belong to the namespace. We also get a powerful capability to read and modify object properties that will control the behaviour of the operating system on the remote host. There are several ways to achieve this purpose. We will use the ExecQuery method. The ExecQuery method allows for the selective retrieval of objects, thus minimising the size of data retrieved and transported across the network. Other methods of WMI object retrieval typically extract more information than is necessary to perform the system management or monitoring task of choice. The line below retrieves only the collection of service(s) that match the string specified by strService:

```
Set colServices=objWMIService.ExecQuery("Select * from Win32_Service Where Name='"& strService & "'")
```

The next section of code is iterative. It steps through the returned object collection, performs error tests, checks the state of the requested service, and starts the requested

service if it is in a stopped state. Otherwise, the script reports that the service is not found to be in a stopped state:

```
If Not(colServices Is Nothing) Then
  For Each objService in colServices
    If Not(objService Is Nothing) Then
      If objService.State = "Stopped" Then
        Wscript.Echo vbCrlf & "Starting " & strService _
          & " service..."
        objService.StartService()
        WSCript.Sleep intTimeout
      Else
        Wscript.Echo vbCrlf & "Service is not stopped."
      End If
    End If
  Next
End If
```

There are more service state tests that can be and should be carried out, but for the simplicity sake, we will limit ourselves with a single service state test per iteration.

The last section of the script performs the cleanup:

```
Set objService = Nothing
Set colServices = Nothing
Set objWMIService = Nothing
Set objLocator = Nothing
```

Our brief introduction to WMI scripting by stepping through our sample script should be sufficient for the purpose of basic understanding of how to construct WMI access to a remote system. Hope this section helps appreciation of the simplicity and power of WMI

access, and more importantly, appreciation of the security responsibility associated with the easy and powerful system management mechanism such as WMI.

If you would like a text introducing basic WMI architecture and scripting technique, you could pay a visit to Microsoft's Scripting Clinic and retrieve the set of three articles written by Greg Stemp, Dean Tsaltas, Bob Wells of Microsoft, and Ethan Wilansky of Network Design Group. These texts were written in 2002 - 2003, however this set of introductory articles remains a good point to get familiar with WMI. Here are the links:

WMI Scripting Primer: Part 1: http://msdn2.microsoft.com/en-us/library/ms974579.aspx

WMI Scripting Primer: Part 2: http://msdn2.microsoft.com/en-us/library/ms974592.aspx

WMI Scripting Primer: Part 3: http://msdn2.microsoft.com/en-us/library/ms974547.aspx

After developing the understanding of WMI architecture and technique, use your favourite search engine to find plenty of WMI script resources available on the Internet. One of my favourites is Cwashington WMI script library that contains plenty of simple, useful, and easy to start with scripts. Here is where to find it:

http://cwashington.netreach.net/depo/default.asp?topic=wmifaq

2.2. Alternative WMI Access with Monkier String

Monkier string is another way to script WMI access. This method allows to skip a

line when coding, at the expense of clarity and flexibility that is partially lost if you choose to use the monkier. The monkier string method is used frequently nonetheless. We will cover the basics by comparing with the object access method described in previous section.

Here is the code fragment used to return the collection of WMI objects as an example in previous section:

```
Set objLocator = CreateObject("WbemScripting.SWbemLocator")

Set objWMIService = objLocator.ConnectServer _
 (strComputer, "root\CIMV2", strUser, strPassword)

Set colServices = objWMIService.ExecQuery _
 ("Select * from Win32_Service Where Name = '" _
  & strService & "'")
```

The monkier string equivalent would look as shown below. As promised, it is one line shorter:

```
Set objWMIService = GetObject("winmgmts:\\" _
 & strComputer & "\root\CIMV2")

Set colServices = objWMIService.ExecQuery _
 ("Select * from Win32_Service Where Name = '" _
  & strService & "'")
```

Monkier may be Ok for testing purpose. The downside of the moniker way of scripting a WMI connection is that the monkier way employs standard COM access. WMI namespace connection will use the credentials of the currently logged on user. This is

probably not what our security policy would like us to do most of the time, when administering remote Windows hosts.

Considering performance, moniker may provide a quicker once off access to a collection of WMI objects. On the other side, if we were to access the objects repeatedly as with most real life administrative tasks, the moniker way may be a less efficient way of connecting to WMI. If performance is important, the overall recommendation would be to use the object way of connecting to WMI, as described in previous section.

The object way of WMI scripting not only performs better, but also more importantly provides better authentication. Monkier immediately retrieves an object instance, while the object method first creates an object, and then allows logging on to the object name space with the explicitly specified credentials. The capability to specify user credentials is a significant advantage of the object way of WMI connection scripting. In addition, the object way provides better error reporting, when comparing with the monkier mechanism.

Despite our justified preference for the object way of WMI scripting, we provide a number of examples of using monkier throughout this paper, due to the fact that the monkier string technique is often used in Microsoft scripting examples, as well as in non-vendor specific script archives on the Internet.

At this point, it is time to move on and see what happens when the sample script

presented in section 2.1 is run against a remote Windows host.

3. WMI Default Security on the Wire

The purpose of this section is to demonstrate how system management information is transported along the wire, when the default WMI security settings are in use. We run the sample script from Section 2.1 on the local host (scripting host) against the remote host. Both systems are running Windows XP Professional, Service Pack 2.

Packets are captured on the wire during the running of the sample script, as WMI authentication and name space connection is taking place. Then a WMI query is sent from the local host to the remote hosts. Finally, WMI management information is returned by the remote host to the local host.

Below is the part of the network dialog between the two systems from the moment when the initial management query is sent, and to the moment when the management data of interest is returned. The full network conversation between the two hosts is shown in Appendix 1.

Packet	Source	Destination	Protocol and Information
52	10.252.253.8	10.252.253.44	DCERPC Request: call_id: 5 opnum: 20 ctx_id: 3 IWbemServices V0
53	10.252.253.44	10.252.253.8	DCERPC Response: call_id: 5 ctx_id: 3 IWbemServices V0
54	10.252.253.8	10.252.253.44	IRemUnknown2 RemQueryInterface request IID[1]=IWbemFetchSmartEnum
55	10.252.253.44	10.252.253.8	IRemUnknown2 RemQueryInterface response S_OK[1] -> S_OK
56	10.252.253.8	10.252.253.44	DCERPC Alter_context: call_id: 7 IWbemFetchSmartEnum V0.0
57	10.252.253.44	10.252.253.8	DCERPC Alter_context_resp: call_id: 7 accept max_xmit: 5840 max_recv: 5840
58	10.252.253.8	10.252.253.44	DCERPC Request: call_id: 7 opnum: 3 ctx_id: 4 IWbemFetchSmartEnum V0

59	10.252.253.44	10.252.253.8	DCERPC	Response: call_id: 7 ctx_id: 4 IWbemFetchSmartEnum V0
60	10.252.253.8	10.252.253.44	DCERPC	Alter_context: call_id: 8 IWbemWCOSmartEnum V0.0
61	10.252.253.44	10.252.253.8	DCERPC	Alter_context_resp: call_id: 8 accept max_xmit: 5840 max_recv: 5840
62	10.252.253.8	10.252.253.44	DCERPC	Request: call_id: 8 opnum: 3 ctx_id: 5 IWbemWCOSmartEnum V0
63	10.252.253.44	10.252.253.8	TCP	1032 > 1716 [ACK] Seq=1025 Ack=1299 Win=65535 Len=0
64	10.252.253.8	10.252.253.44	TCP	1715 > 1032 [ACK] Seq=1107 Ack=625 Win=64911 Len=0
65	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
66	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
67	10.252.253.8	10.252.253.44	TCP	1716 > 1032 [ACK] Seq=1299 Ack=3545 Win=65535 Len=0
68	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
69	10.252.253.8	10.252.253.44	TCP	1716 > 1032 [ACK] Seq=1299 Ack=4805 Win=65535 Len=0
70	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
71	10.252.253.44	10.252.253.8	DCERPC	Response: call_id: 8 ctx_id: 5 [DCE/RPC first fragment, reas: #73]
72	10.252.253.8	10.252.253.44	TCP	1716 > 1032 [ACK] Seq=1299 Ack=6865 Win=65535 Len=0
73	10.252.253.44	10.252.253.8	DCERPC	Response: call_id: 8 ctx_id: 5 IWbemWCOSmartEnum V0

Figure 2: WMI Data Exchange on the Wire

Please note packets 52 and 73. Packet 52 contains the WMI query sent from the host with IP address 10.252.253.8 (our scripting host) to the host with IP address 10.252.253.44 (our remote host). Packet 73 contains the WMI management data returned by the remote host 10.252.253.44 to the scripting host 10.252.253.8.

The partial payload of packet 52 is shown below. We are not reproducing the entire packet here. The entire packet and its payload are included in Appendix 1.

```
00 53 00 65 00 6c 00 65 00 63 00 74 00 20 00 2a 00 20
00 66 00 72 00 6f 00 6d 00 20 00 57 00 69 00 6e 00 33
00 32 00 5f 00 53 00 65 00 72 00 76 00 69 00 63 00 65
00 20 00 57 00 68 00 65 00 72 00 65 00 20 00 4e 00 61
00 6d 00 65 00 20 00 3d 00 20 00 27 00 54 00 6c 00 6e
00 74 00 53 00 76 00 72 00 27
```

Conversion from hexadecimal to ASCII is a straightforward matter. I have written a

short VBScript to do such conversions, and to deal with the capture file format. Alternatively, here are tools available on the Internet that would suit the same purpose. After running the selected partial packet payload through the converter script, we get the following data:

```
Select * from Win32_Service Where Name = 'TlntSvr'
```

This corresponds to the ExecQuery, lines 42 - 44 of the sample script, where TlntSvr is the value assigned to the strService string variable. This value was indeed assigned during the interactive part of the script, as the Telnet service was selected on the remote host for the purpose of this experiment.

We observe that the WMI management query is transmitted over the network in clear. It is trivial to find out the exact query information by snooping on the network.

The next interesting packet is packet 73. This packet contains partial WMI data sent from the remote host 10.252.253.44 to the scripting host 10.252.253.8, in response to the previously shown WMI query. Once again, please refer to Appendix 1 to see the full packet payload content. The partial packet payload is shown below:

```
45 3a 5c 57 49 4e 44 4f 57 53 5c 53 79 73 74 65 6d 33
32 5c 74 6c 6e 74 73 76 72 2e 65 78 65 00 00 4c 6f 63
61 6c 53 79 73 74 65 6d 00 00 57 69 6e 33 32 5f 53 65
72 76 69 63 65 00 00 57 69 6e 33 32 5f 43 6f 6d 70 75
74 65 72 53 79 73 74 65 6d 00 00 50 49 47 41 00 00 54
6c 6e 74 53 76 72
```

As with previous payload fragment, we run this through our converter script, and find the data transmitted by the remote host to the scripting host:

```
E:\WINDOWS\System32\tlntsvr.exeLocalSystemWin32_ServiceWin32_ComputerSystemPIGATlntSvr
```

The WMI management data is returned in clear, just as the WMI query was sent. It is easy to see that the information available in clear includes system name, service name, path, access authority, and more. We have referenced only the small fragment of the returned management data. Captured data also included service state, description, and other information. Such system data is more often than not considered confidential information. It should not travel across the network without protection.

In this section, we have shown that it is trivial to snoop on the sensitive WMI management data, as long as the default WMI security settings are in use. We have accessed system service information in this particular example. While system service information is often unique and sensitive, it is easy to imagine the cases where data that is even more sensitive is exchanged across the network via Windows Management Interface. Clear text is not the best way to transmit such data. Next section will focus on the basic security options available for configuring WMI access security. WMI security options must be set explicitly in order to avoid unnecessary exposure of the sensitive management data.

4. WMI Scripting Security Options

"We have all some experience of a feeling, that comes over us occasionally, of what we are saying and doing having been said and done before, in a remote time – of our having been surrounded, dim ages ago, by the same faces, objects, and circumstances – of our knowing perfectly what will be said next, as if we suddenly remember it!"

Dickens, Charles (1991)
The Personal History of David Copperfield

WMI is unquestionably a deja vu experience, as far as Windows reputation of providing rich functionality coupled with not so secure default settings is concerned.

Windows Management Interface does not provide its own security mechanism. Instead, WMI makes use of the DCOM security functionality offered by Windows. WMI supports a number of security levels. The WMI security levels are best documented in the Microsoft TechNet article called WMI Security Settings (Microsoft TechNet, 2007). There is a good supplement article that provides updates to the above-mentioned reference, along with useful examples. This second Microsoft MSDN Library article is called Setting the Default Process Security Level Using VBScript (Microsoft MSDN Library, 2007). The links to both articles are included in the References section.

At this point, we will discuss basic WMI security options. This should start us on the road to securing WMI remote management connections. Basic WMI security settings may

be set according to so-called impersonation and authentication levels.

In simple terms, impersonation setting is what WMI service will use on the remote host to carry out the tasks requested by the WMI script. Authentication setting allows WMI scripts to request the desired level of DCOM authentication for the WMI connection. There is more to the authentication setting. The WMI authentication setting allows specifying the use of encryption across the WMI connection. This is exactly the option that will help us dealing with the clear data transport problem identified in previous section.

First, let us take a look at the impersonation levels available. The table containing the description of the impersonation levels is below. We provide the value of WbemImpersonationLevelEnum for each of the impersonation levels, along with a description, a summary of useability, and a general scripting use recommendation. We reference both numeric and constant values. More on the values and their implementation in WMI scripts in the next section.

Level: Anonymous Level Value: 1 or wbemImpersonationLevelAnonymous Description: Caller credentials are hidden when the Anonymous level of impersonation is chosen. From the security standpoint, this should not be allowed, and in fact, this is not allowed. WMI will automatically upgrade the requested impersonation level to Identify. Such upgrade is of questionable useability. Identify is likely to not work in the WMI script against a remote host, and the WMI script is likely to fail. General recommendation: Do not use Anonymous impersonation level.
Level: Identify Level Value: 2 or wbemImpersonationLevelIdentify Description: Objects will query the caller credentials. As this would fail a script, this is not a recommended impersonation setting for scripting WMI connections. Microsoft specified the Identify level for performing

access control list checks, not so much for WMI scripting.

General recommendation: Do not use Identify impersonation level.

Level: Impersonate Level

Value: 3 or wbemImpersonationLevelImpersonate

Description: Objects will use the caller credentials. In simple terms, such impersonated WMI script will use current user credentials or better yet, explicitly specified user credentials. Remote host should have no problem with such authentication mechanism, and the script should run against the remote host once the authentication is successful.

General recommendation: Do use Impersonate impersonation level, as this is the best impersonation option available for most WMI scripting tasks.

Level: Delegate Level

Value: 4 or wbemImpersonationLevelDelegate

Description: Caller credentials are trusted for delegation to other objects. If you run your WMI script against a remote host specifying Delegate impersonation level, not only remote host object access is granted, but also the remote host is then able to use the same credentials for other hosts. While such scenario will work, it will allow excessive access, not required for the majority of scripting tasks.

There is a possibility of Delegate impersonation level being required in a situation when remotely managed system needs to manage other systems. An example of this may be a central patch distribution system. If this is not your case, the Delegate impersonation level should not be required.

Please note that the user account used for scripting and the computer accounts used in the distributed operation or transaction should be marked as Trusted for Delegation in the Active Directory. Delegate impersonation level was first implemented in Windows 2000. It will only work with Windows 2000 or later.

General recommendation: Do not use Delegate impersonation level, unless it is required. Delegate impersonation level should be considered a security risk.

Figure 3: WMI Impersonation Levels

There are six authentication levels that can be used with WMI:

Level: Default Level

Value: 0 or WbemAuthenticationLevelDefault

Description: This setting allows WMI to negotiate the authentication level with the remote host. The authentication level specified by the remote host will be used. Microsoft recommends using this setting.

General recommendation: Do not use Default authentication level in WMI. Do not rely on the default security as it allows WMI exchange management data in clear. Do use Level 6, PktPrivacy instead.

Please note that our recommendation contradicts the practice suggested by Microsoft. As we have seen in previous section, the default authentication level allows clear data exchange over the network. We recommend that the default authentication is only used if the use of encryption is not allowed. Normally there should not be an issue with encrypting WMI connections.

Level: None Level

Value: 1 or WbemAuthenticationLevelNone

Description: Connection without authentication is attempted. Security settings are ignored.

General recommendation: Needless to say, we recommend against using this authentication level.

Level: Connect Level

Value: 2 or WbemAuthenticationLevelConnect

Description: Perform credentials authentication only when the WMI script attempts to connect to the remote host. No additional authentication is performed after this moment.

General recommendation: Do not use Connect authentication level. While Connect is better than None, there are better options available.

Level: Call Level

Value: 3 or WbemAuthenticationLevelCall

Description: Call authentication level does a little better. Authentication of credentials is performed at the beginning of each call/ request to the remote host. The initial packets have their headers signed and checked. However, data packets that are subsequently exchanges between the scripting host and the remote host are neither signed nor encrypted.

General recommendation: Do not use Call authentication level. While some session authentication is performed, neither data integrity nor confidentiality is implemented with the Call authentication level.

Level: Pkt Level

Value: 4 or WbemAuthenticationLevelPkt

Description: Authentication is performed on all of the data received from the scripting host. In a way, Pkt authentication level is not far away from Call level. All the headers are signed and checked. We may be better assured about the source, but as with the Call level, data packet payload is neither signed nor encrypted.

General recommendation: Do not use Pkt level of WMI authentication. Do use Level 6, PktPrivacy instead.

Level: PktIntegrity Level

Value: 5 or WbemAuthenticationLevelPktIntegrity

Description: Authenticates and verifies that none of the data transferred between the scripting host and the remote host has been modified.

PktIntegrity authentication level provides integrity. Every WMI data packet travelling between the script host and the remote host is signed and checked. We have the assurance that the management data traversing the network has not been modified in transit.

There is no protection from data traffic snooping, as the data packet payload is not encrypted.

General recommendation: PktIntegrity WMI authentication level can be used when data confidentiality is not an issue, or if the use of encryption is not allowed. Generally, it is better to use PktPrivacy authentication level that provides the data encryption service.

Level: PktPrivacy Level Value: 6 or WbemAuthenticationLevelPktPrivacy Description: PktPrivacy authentication signs and encrypts each data packet, in addition to providing authentication service. PktPrivacy WMI authentication provides the confidentiality service for all data packets traversing the network between the scripting host and the remote host, offering protection from network traffic snooping. General recommendation: We recommend the PktPrivacy WMI authentication level as a preferred option, offering better security and management data protection.

Figure 4: WMI Authentication Levels

This brings us to the end of this section. We have introduced the basic WMI security settings: impersonation and authentication. Our recommendation is to never rely on the default security settings, but instead set the WMI impersonation level to 3 (Impersonate), and set the WMI authentication level to 6 (PktPrivacy) for general WMI scripting tasks. We will look at how to implement the basic WMI security options in VBScript based WMI scripts in the next section.

5. Avoiding the Trap with Packet Privacy

We have referred to the Microsoft MSDN Library article called Setting the Default Process Security Level Using VBScript (Microsoft MSDN Library, 2007) in previous section. The Web link to the article is available in the References section. This article contains useful examples of implementing basic WMI security with VBScript.

For the purpose of our discussion, we will return to the sample script used for the illustrations through this paper. The script is found in Section 2.1. We first discussed the

sample script when providing an introduction to WMI scripting. The script implemented a WMI call with the default WMI security settings. As we pointed out, this was done deliberately, for the purpose of our subsequent investigation of whether the default WMI security is sufficient to provide secure management data exchange.

The part of the script that makes use of the object method to retrieve a service collection using the default WMI security levels is reproduced below for your convenience:

```
Set objLocator = CreateObject("WbemScripting.SWbemLocator")

Set objWMIService = objLocator.ConnectServer _
  (strComputer, "root\CIMV2", strUser, strPassword)

Set colServices = objWMIService.ExecQuery _
  ("Select * from Win32_Service Where Name = '" _
    & strService & "'")
```

The same scripted with monkier instead of the object method would look as:

```
Set objWMIService = GetObject("winmgmts:\\" _
  & strComputer & "\root\CIMV2")

Set colServices = objWMIService.ExecQuery _
  ("Select * from Win32_Service Where Name = '" _
    & strService & "'")
```

We have discussed that we generally recommend the object method for the reasons of better security and better performance.

In Section 3, we have shown that using the default WMI security settings is not the

best approach to WMI scripting security. The default WMI connection does not encrypt data payload. Instead, sensitive WMI management information is transmitted over the network in clear text.

Here is how to modify the sample script to set the basic WMI security options explicitly, for the object access method. Please note the two extra lines:

```
Set objLocator = CreateObject("WbemScripting.SWbemLocator")

Set objWMIService = objLocator.ConnectServer _
  (strComputer, "root\CIMV2", strUser, strPassword)

objWMIService.Security_.impersonationlevel = 3
objWMIService.Security_.authenticationlevel = 6

Set colServices = objWMIService.ExecQuery _
  ("Select * from Win32_Service Where Name = '" _
    & strService & "'")
```

Monkier example with the explicitly set security options is also two lines longer:

```
Set objWMIService = GetObject("winmgmts:" _
  & "{impersonationLevel = impersonate," _
    & "authenticationLevel = pktPrivacy}!\\" _
      & strComputer & "\root\CIMV2")

Set colServices = objWMIService.ExecQuery _
  ("Select * from Win32_Service Where Name = '" _
    & strService & "'")
```

The two extra lines used to set the explicit WMI security for the object access are:

```
objWMIService.Security_.impersonationlevel = 3
objWMIService.Security_.authenticationlevel = 6
```

The corresponding extension of the monkier string with the security parameters is:

```
{impersonationLevel = impersonate, authenticationLevel = pktPrivacy}!
```

The newly added lines set the impersonation and authentication levels to the levels that we have generally recommend for optimal security, performance, and interoperability:

WbemImpersonationLevelImpersonate or 3 best used as impersonation level

WbemAuthenticationLevelPktPrivacy or 6 best used as authentication level

There is a unique case of distributed WMI management, whereby the scripting host connects to the remote management host that should in turn perform operations on other hosts. While it may not be ideal from the WMI security standpoint, such scenario may require the use of the Delegate impersonation level. In such case, the two lines setting the explicit WMI security parameters would become, for the object method:

```
objWMIService.Security_.impersonationlevel = 4
objWMIService.Security_.authenticationlevel = 6
```

The same lines for the monkier string are below:

```
  & "{impersonationLevel=delegate," _
   & "authenticationLevel=pktPrivacy}!" _
```

Once again, only use the Delegate impersonation level if your design cannot use the preferred Impersonate impersonation level.

Now let us take a look at the updated service start script. We have added the line explicitly setting WMI authentication level to PktPrivacy. We should now have the WMI connection that encrypts data packet payload protecting the sensitive management data.

```
'---------- Service Start Script - WMI Based ----------
'
' Test the state of service of user choice on the remote host
' Start service on the remote host if the service is not running
'
' Written by Alex Timkov to facilitate SANS GSEC research paper
'------------------------------------------------------------

Option Explicit

Dim strComputer, strPassword, strService, strUser
Dim objLocator, objService, objWMIService
Dim colServices
Dim intTimeout

intTimeout = 10000

Do
  strComputer = inputbox _
    ("Please enter host name or IP address", "Input")
Loop until strComputer <> ""

Do
  strUser = inputbox("Please enter username", "Input")
Loop until strUser <> ""

Do
  strPassword = inputbox("Please enter password", "Input")
Loop until strPassword <> ""

strService = inputbox _
  ("Please enter the name of service to start, " _
    & "or hit Enter to exit", "Input")

If strService <> "" Then

  Set objLocator = CreateObject("WbemScripting.SWbemLocator")

  Set objWMIService = objLocator.ConnectServer _
    (strComputer, "root\CIMV2", strUser, strPassword)

  objWMIService.Security_.impersonationlevel = 3
  objWMIService.Security_.authenticationlevel = 6

  Set colServices = objWMIService.ExecQuery _
```

```
    ("Select * from Win32_Service Where Name = '" _
      & strService & "'")

    If Not(colServices Is Nothing) Then
      For Each objService in colServices
        If Not(objService Is Nothing) Then
          If objService.State = "Stopped" Then
            Wscript.Echo vbCrlf & "Starting " & strService _
              & " service..."
            objService.StartService()
            WSCript.Sleep intTimeout
          Else
            Wscript.Echo vbCrlf & "Service is not stopped."
          End If
        End If
      Next
    End If

  Set objService = Nothing
  Set colServices = Nothing
  Set objWMIService = Nothing
  Set objLocator = Nothing

End If

WScript.Echo vbCrlf & "Script execution completed. Exiting."
WScript.Quit

'---------------------------------------
' End of WMI Based Service Start Script
'---------------------------------------
```

Figure 5: Updated WMI Service Start Script

6. Improved WMI Connection Security on the Wire

We are going to execute the updated service start script against the remote host, and make sure that when the correct WMI security options are set explicitly, we are no longer able to snoop on the WMI management data traversing the network.

We run the updated sample script from Section 5 on the scripting host against the remote host. Both systems are running Windows XP Professional, Service Pack 2.

Packets are captured on the wire during the running of the updated sample script, as WMI authentication and name space connection is taking place, the WMI query is sent from the local host to the remote hosts, and the WMI management information is returned by the remote host to the local host.

Below is the part of the network dialog between the two systems from the moment when the initial management query is sent, and to the moment when the management data of interest is returned. The full network conversation between the two hosts is shown in Appendix 2.

Packet	Source	Destination	Protocol and Information
52	10.252.253.8	10.252.253.44	DCERPC Request: call_id: 5 opnum: 20 ctx_id: 3 IWbemServices V0
53	10.252.253.44	10.252.253.8	TCP 1032 > 1842 [ACK] Seq=185 Ack=667 Win=64869 Len=0
54	10.252.253.44	10.252.253.8	DCERPC Response: call_id: 5 ctx_id: 3 IWbemServices V0
55	10.252.253.8	10.252.253.44	DCERPC Alter_context: call_id: 6 IRemUnknown2 V0.0
56	10.252.253.44	10.252.253.8	DCERPC Alter_context_resp: call_id: 6 accept max_xmit: 5840 max_recv: 5840
57	10.252.253.8	10.252.253.44	IRemUnknown2 RemQueryInterface request[Malformed Packet]
58	10.252.253.44	10.252.253.8	IRemUnknown2 RemQueryInterface response[Malformed Packet]
59	10.252.253.8	10.252.253.44	DCERPC Alter_context: call_id: 7 IWbemFetchSmartEnum V0.0
60	10.252.253.44	10.252.253.8	DCERPC Alter_context_resp: call_id: 7 accept max_xmit: 5840 max_recv: 5840
61	10.252.253.8	10.252.253.44	DCERPC Request: call_id: 7 opnum: 3 ctx_id: 4 IWbemFetchSmartEnum V0
62	10.252.253.8	10.252.253.44	TCP 1838 > epmap [ACK] Seq=97 Ack=193 Win=65343 Len=0
63	10.252.253.8	10.252.253.44	TCP 1839 > epmap [ACK] Seq=1147 Ack=1241 Win=64295 Len=0
64	10.252.253.44	10.252.253.8	DCERPC Response: call_id: 7 ctx_id: 4 IWbemFetchSmartEnum V0
65	10.252.253.8	10.252.253.44	DCERPC Alter_context: call_id: 8 IWbemWCOSmartEnum V0.0
66	10.252.253.44	10.252.253.8	DCERPC Alter_context_resp: call_id: 8 accept max_xmit: 5840 max_recv: 5840
67	10.252.253.8	10.252.253.44	DCERPC Request: call_id: 8 opnum: 3 ctx_id: 5 IWbemWCOSmartEnum V0
68	10.252.253.44	10.252.253.8	TCP 1032 > 1842 [ACK] Seq=929 Ack=1491 Win=65535 Len=0
69	10.252.253.8	10.252.253.44	TCP 1840 > 1032 [ACK] Seq=899 Ack=497 Win=65039 Len=0
70	10.252.253.8	10.252.253.44	TCP 1841 > 1032 [ACK] Seq=507 Ack=409 Win=65127 Len=0
71	10.252.253.44	10.252.253.8	TCP [TCP segment of a reassembled PDU]
72	10.252.253.44	10.252.253.8	TCP [TCP segment of a reassembled PDU]
73	10.252.253.8	10.252.253.44	TCP 1842 > 1032 [ACK] Seq=1491 Ack=3449 Win=65535 Len=0

74	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
75	10.252.253.8	10.252.253.44	TCP	1842 > 1032 [ACK] Seq=1491 Ack=4709 Win=65535 Len=0
76	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
77	10.252.253.44	10.252.253.8	DCERPC	Response: call_id: 8 ctx_id: 5 [DCE/RPC first fragment, reas: #79]
78	10.252.253.8	10.252.253.44	TCP	1842 > 1032 [ACK] Seq=1491 Ack=6769 Win=65535 Len=0
79	10.252.253.44	10.252.253.8	DCERPC	Response: call_id: 8 ctx_id: 5 IWbemWCOSmartEnum V0

Figure 6: Secure WMI Data Exchange on the Wire

When investigating the default WMI security in section 3, we analysed the packet payload containing the WMI query, and the response packet of interest. The interesting packets selected were packet 52 and packet 73. Packet numbers are different this time. This does not come as a surprise, since if encryption is used indeed, the packet sequence and size may vary from our prior experiment.

We notice that the packet with the payload of the query is still packet 52, however the data of interest is returned not in packet 73, but in packet 79.

As previously, we take a look at the partial payload of packet 52, as shown below. The entire packet and its payload are shown in Appendix 2.

```
f8 e6 c0 f9 81 14 b5 eb 6f 14 ad 1a a3 9a a1 1c a1 c7
0f e0 0c fb b5 1d 82 e8 01 c2 68 73 7c 8a ff 57 74 b7
24 87 7f 57 53 55 65 6a 8b 64 7c 31 48 cc ae 05 74 ae
30 b3 75 cc 09 16 9a c0 7d f4 bd 3d 47 6d 50 df 46 f1
8a a5 2a b2 ca 83 04 f7 d0 b0 26 42 59 49 93 46 88 e3
ce 34 4c 04 2b 0e 13 77 27 50
```

Upon a superficial look, this time around the payload seems harder to snoop at, more than simple encoded text. Running our ASCII converter would not do any good to

help reading this payload fragment. Indeed, after performing a more thorough analysis of the packet, the payload appears to be encrypted. Such analysis goes beyond the scope of this paper, and we will leave the detailed analysis out of this text.

Let us take a look at the second packet of interest, packet 79. This packet is the response packet, containing some of the WMI management data. Once again, please refer to Appendix 2 for the full packet payload content. Below is the partial payload:

```
56 9b a8 57 dd af 2c a7 af 86 ab 65 f5 4b 2c bf dd db
d5 d8 37 e4 9b dd 28 b6 f3 52 36 bf 3a 89 3b cc 52 be
c4 56 23 62 ca af ce e0 44 2b ea 50 6f ef f4 dd c5 8b
49 32 2e 5a 66 8c 60 c3 f5 86 3e b4 67 6a 94 4b 43 b4
6f 40 49 3c bb a2 14 44 6a 0a 2e 3c d4 7f c1 ba 79 ba
1a 99 02 8d 02 22
```

As with the query packet payload, the data packet payload does not look like giving much hope of reading its content easily. Same as the WMI query packet, the payload of the packet carrying WMI management data appears to be encrypted. A detailed analysis of the entire packet content confirms this to be the case.

Adding the recommended explicit WMI authentication setting to our sample WMI script did the trick. The WMI management data exchange is now considerably more secure. We have eliminated the data exposure caused by the weak default WMI authentication setting.

7. Advanced WMI Scripting Security Options

It is time to move on from the analysis of the basic WMI security options to discuss advanced WMI security topics, and to offer tips that should aid our WMI management security policy and practice.

7.1. Keep Administrative Passwords out of Scripts

In this section, we would like to step back for a moment, and to bring to your attention an example illustrating the obvious fact that no amount of advanced technology will compensate for the lack of security policy and the lack of complete approach to security.

Not long ago I was checking Microsoft Web pages for the new developments and updates in the field of WMI security. I came across new secure WMI access examples. If I were to re-write the sample script used through this paper, using the new recommendations from the new Microsoft site articles, the WMI object access lines from the sample script would change to look as this:

```
Set objLocator = CreateObject("WbemScripting.SWbemLocator")

Set objWMIService = objLocator.ConnectServer _
  ("computername", "root\CIMV2", "useraccount", "userpassword")

objWMIService.Security_.impersonationlevel = 3
objWMIService.Security_.authenticationlevel = 6
```

```
Set colServices = objWMIService.ExecQuery _
  ("Select * from Win32_Service Where Name = '" _
    & strService & "'")
```

There is a new good intention. Microsoft scripting examples are now showing explicitly set WMI security levels. However, is there a problem with this code? The ConnectServer method shows the user password in clear. If we were to guard our scripts, could we get away with such approach? This does not seem likely. In practice, scripts reside on the servers, CDs, USB disks and the like. Not many administrators keep scripts inside safes.

In other WMI scripting examples that I have seen, the script account was in fact called "administrator", and the administrative password was typed in clear text as the next parameter. Is it worth worrying about the authentication and the encryption of the WMI data exchange, if scripts leave administrative account password in clear?

May this be only an example, and nothing to do with the real life? No one would force administrators to write their real scripts in such a way. The problem is that the examples are often taken from vendor sites, and re-used without due analysis of implications. The other week I was shown a management script used to administer a critical and very sensitive real life system. The script contained administrative account and password, both in clear text. Would this be as bad as it gets? It got worse. The script resided on the server file system with world read access, and there were multiple users on

that system. We may never know where the scripts end up.

One way around exposing passwords through scripts is to use interactive scripts, just as part of the sample we have been using throughout this paper shows:

```
Do
  strComputer = inputbox _
    ("Please enter host name or IP address", "Input")
Loop until strComputer <> ""

Do
  strUser = inputbox("Please enter username", "Input")
Loop until strUser <> ""

Do
  strPassword = inputbox("Please enter password", "Input")
Loop until strPassword <> ""

strService = inputbox _
  ("Please enter the name of service to start, " _
    & "or hit Enter to exit", "Input")

If strService <> "" Then

  Set objLocator = CreateObject("WbemScripting.SWbemLocator")

  Set objWMIService = objLocator.ConnectServer _
    (strComputer, "root\CIMV2", strUser, strPassword)

  objWMIService.Security_.impersonationlevel = 3
  objWMIService.Security_.authenticationlevel = 6

  Set colServices = objWMIService.ExecQuery _
    ("Select * from Win32_Service Where Name = '" _
      & strService & "'")
```

The ConnectServer is using the string variables that are filled with user input during an interactive session.

There may be a situation where we may prefer not to use an interactive session. In such situation, we would recommend the use of the monkier account authentication. This

is not ideal because we lose the flexibility to specify credentials of the user other than the user currently logged on, but it is better still than storing administrative password in clear text. It is most dangerous to lose track of where privileged passwords end up.

Here is the example of the monkier account authentication:

```
Set objWMIService = GetObject("winmgmts:" _
  & "{impersonationLevel = impersonate," _
    & "authenticationLevel = pktPrivacy}!\\" _
      & strComputer & "\root\CIMV2")

Set colServices = objWMIService.ExecQuery _
  ("Select * from Win32_Service Where Name = '" _
    & strService & "'")
```

We have deviated from discussing the pure specifics of WMI security. This was a quick detour, a reminder that the security approach should be complete, and that all effects must be taken into account when security is considered. Now we will return to the specifics of WMI and discuss the advanced subject of using specific permissions in WMI scripts.

7.2. Using Specific Permissions in WMI Scripts

Aside from correctly setting the impersonation and authentication for WMI connections, and along with following sound security practices, there are several advanced security options that can help us secure WMI connections.

WMI allows controls that are more granular and go beyond the general security levels. One of the available options is to override the default privileges. A specific access privilege can be added or revoked as part of a WMI script. Below is the full list of privileges that can be granted and revoked, as specified in the Microsoft TechNet article WMI Security Settings (Microsoft TechNet, 2007):

Privilege	Description
CreateToken	Required to create a primary token.
AssignPrimaryToken	Required to assign the primary token of a process.
LockMemory	Required to lock physical pages in memory.
IncreaseQuota	Required to increase the quota assigned to a process.
MachineAccount	Required to create a computer account.
Tcb	Identifies its holder as part of the trusted computer base. Some trusted, protected subsystems are granted this privilege.
Security	Required to perform a number of security-related functions, such as controlling and viewing audit messages. This privilege identifies its holder as a security operator.
TakeOwnership	Required to take ownership of an object without being granted discretionary access. This privilege allows the owner value to be set only to those values that the holder might legitimately assign as the owner of an object.
LoadDriver	Required to load or unload a device driver.
SystemProfile	Required to gather profiling information for the entire system.
SystemTime	Required to modify the system time.
ProfileSingleProcess	Required to gather profiling information for a single process.
IncreaseBasePriority	Required to increase the base priority of a process.
CreatePagefile	Required to create a paging file.
CreatePermanent	Required to create a permanent object.
Backup	Required to perform backup operations.
Restore	Required to perform restore operations. This privilege lets you set any valid user or group SID as the owner of an object.
Shutdown	Required to shut down a local computer.
Debug	Required to debug a process.
Audit	Required to generate audit-log entries.
SystemEnvironment	Required to modify the non-volatile RAM of systems that use this type of memory to store configuration information.

ChangeNotify	Required to receive notifications of changes to files or directories. This privilege also causes the system to skip all traversal access checks. It is enabled by default for all users.
RemoteShutdown	Required to shut down a computer using a network request.
Undock	Required to remove a computer from its docking station.
SyncAgent	Required to synchronize directory service data.
EnableDelegation	Required to enable computer and user accounts to be trusted for delegation.

Figure 7: Specific Permissions for WMI Scripting

We will modify our monkier example from Section 7.1 to add the explicit privileges to lock memory and debug, and to remove the privilege to remotely shut down the system. Please note the exclamation mark preceding the privilege that is being revoked:

```
Set objWMIService = GetObject("winmgmts:" _
  & "{impersonationLevel = impersonate," _
    & "authenticationLevel = pktPrivacy," _
      & "(LockMemory, Debug, !RemoteShutdown)}!\\" _
        & strComputer & "\root\CIMV2")

Set colServices = objWMIService.ExecQuery _
  ("Select * from Win32_Service Where Name = '" _
    & strService & "'")
```

Please do not assume that WMI will allow the scripting account to gain the privileges that the scripting account does not hold. WMI mechanism is not designed to circumvent Windows user account security. The user executing the script can only enable those privileges for which his account is authorised by the remote host, as part of that user account authorisation on the target system. If the user running the WMI script specifies a privilege for which his account is not authorised by the remote host, access will fail.

7.3. WMI Authority

WMI Authenticating Authority is another security parameter that can be used with the WMI object connection. Examples of this are rather rare. The authority can be set to use either Kerberos or NTLM authentication. Usually the parameter is not used, and the effect is that the authentication mechanism is negotiated between the scripting host and the remote host. Microsoft article Connecting to WMI Objects (Microsoft TechNet, 2007) contains a couple of relevant examples.

We will build on the previous monkier example, and you can port this to the object method as a practical exercise if you wish. We will re-use our most recent example from Section 7.2, adding one line to it:

```
Set objWMIService = GetObject("winmgmts:" _
  & "{impersonationLevel = impersonate," _
    & "authenticationLevel = pktPrivacy," _
      & "authority = ntlmdomain:DomainName," _
        & "(LockMemory ,Debug, !RemoteShutdown)}!\\" _
          & strComputer & "\root\CIMV2")

Set colServices = objWMIService.ExecQuery _
  ("Select * from Win32_Service Where Name = '" _
    & strService & "'")
```

Please note the added line:

```
& "authority = ntlmdomain:DomainName," _
```

This line requests that NTLM authentication is used. If we prefer to use Kerberos

instead of NTLM, we should write instead:

```
Set objWMIService = GetObject("winmgmts:" _
  & "{impersonationLevel = impersonate," _
    & "authenticationLevel = pktPrivacy," _
      & "authority = kerberos:DomainName\ServerName," _
        & "(LockMemory, Debug, !RemoteShutdown)}!\\" _
          & strComputer & "\root\CIMV2")

Set colServices = objWMIService.ExecQuery _
  ("Select * from Win32_Service Where Name = '" _
    & strService & "'")
```

where the added line is:

```
& "authority = kerberos:DomainName\ServerName," _
```

The DomainName is the name of our domain, and the ServerName is the name of the remote host.

A word of caution, authenticating authority parameter can only be used against a remote system. If we attempted a WMI connection to the local system with authenticating authority parameter set, the connection would fail. Omit the authentication authority parameter in the WMI scripts running against the local host.

Another word of caution, if the Delegate impersonation level is used, Kerberos authentication may be required by the remote system.

7.4. Summary of WMI Scripting Security Options

The purpose of this short section is to provide a reference summary of the WMI security options specified in WMI scripts, with a one-liner example for each option. For brevity sake, we use monkier string based examples.

Set Impersonation level to Impersonate:

```
winmgmts:{impersonationLevel = 3}
```

Set Authentication level to PktPrivacy:

```
winmgmts:{authenticationLevel =6 }
```

Set Authenticating Authority instead of leaving it to negotiation between systems:

```
winmgmts:{authority = ntlmdomain:DomainName}
```

for NTLM authentication, or

```
winmgmts:{authority = kerberos:DomainName\ServerName}
```

for Kerberos authentication.

Grant or revoke specific privileges, preceding removals with the exclamation mark:

```
winmgmts:{(Debug, !RemoteShutdown)}
```

To add the Debug privilege and to revoke the Remote Shutdown privilege.

7.5. WMI Default Impersonation and Legacy Platforms

This is another very short section. It contains the notes that may be needed when managing older Windows platforms, prior to Windows 2000.

Starting with Windows 2000, the default impersonation level is set to Impersonate. If we somehow forgot to specify impersonation level explicitly, the impersonation level is set to Impersonate automatically. This is a nice safety net.

We could encounter a problem with this if managing mixed environments. If the environment contained hosts with the operating system prior to Windows 2000, any scripts not explicitly stating Impersonation level would fail against such legacy hosts.

If we needed to change this default setting for some odd reason, the default DCOM impersonation level setting can be managed with the help of a registry entry:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\WBEM\Scripting\Default Impersonation Level
```

Windows 2000 and later set this entry to 3, Impersonate level.

8. Securing the Server for WMI Management

So far, we have looked at the options applicable to the script code and the scripting host in order to secure WMI connectivity. This section will look at the options available at the remote host to make sure incoming WMI connections are as secure as possible.

8.1. Scripting Account and Administrative Rights

As we all know, good security practice always applies the principle of the least privilege. In other terms, if the right is not required to perform the function, do not grant the right. Good security practice suggests against using administrative accounts, unless necessary. However, if you consider the default Windows permissions, the scripting account would need to be a member of the administrative group, in order to access WMI namespace objects, retrieve object properties, and modify object properties.

In order to follow good security practice, we should adopt the view of creating specific accounts and account groups for specific purpose. We recommend creating a user account, and granting that account specific access to specific namespaces, as the remote system management tasks require.

Setting WMI Namespace Permissions

Microsoft MSDN Library article Setting Namespace Security with the WMI Control (Microsoft MSDN Library, 2007) describes how to set security on WMI namespaces. The instructions below closely follow the instructions from the above-mentioned article. Setting up WMI namespace permissions is a very simple three-step process:

1. Run the WMI Control. On the Start menu, click Run and type wmimgmt.msc.
2. In the WMI Control pane, right-click WMI Control, choose Properties, and then select the Security tab.
3. Navigate to the new namespace of choice, click Security, and configure groups and permissions.

What exactly permissions can be granted to user accounts? Open the Security tab of the WMI Control Properties, and click the Security button to find the following available permission options:

Edit Security	Grant read and write access to WMI security information
Enable Account	Grant read access to objects within the namespace
Execute Methods	Allow object methods from the CIM Object Manager to be run
Full Write	Grant full read, write, and delete access to all CIM objects, classes, and instances
Partial Write	Grant write access to static objects in the repository
Provider Write	Grant write access to objects that are provided by the provider
Read Security	Grant read-only access to WMI security information
Remote Enable	Grant remote access with the same rights as if connecting from the local host

There is an advanced option that allows editing the ACL, and to set the inheritance on child objects. To get there click the Advanced button from the Security Property screen, and follow the three simple steps outlined below.

1. Add an account, or select existing account and click Edit.

2. Use the Apply Onto dialog box to set inheritance in one of the following ways:

On the current namespace
On the current namespace and subnamespaces
Only on the subnamespaces

3. Save for the settings to take effect.

Changing the Default WMI Namespace

Sometimes programmers may not state explicitly to which WMI namespace their

script should be connecting. In such scenario, the script will be directed to the default WMI namespace. The default WMI name space is Root\CIMV2.

If we would like to change the default namespace that is hit by the scripts that are not explicit about their namespace access requirement, we could change this setting. The default WMI namespace setting can be accessed through the following registry entry:

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\WBEM\Scripting\Default Namespace

There is another way of changing the default WMI namespace. The default can be changed from the WMI control interface:

1. Run the WMI Control. On the Start menu, click Run and type wmimgmt.msc.
3. In the WMI Control pane, right-click WMI Control, choose Properties, and then select the Advanced tab.
4. Click the Change button and select the namespace and save the setting to make it the default.

8.2. WMI Namespace Security Auditing

In the last section, we focused on implementing the security principle of the least privilege for WMI. The next important security principle is the audit trail. Administrators should implement a way to verify the day to day functioning of the security policy.

WMI namespace security auditing is best described in the Microsoft MSDN Library article named Access to WMI Namespaces (Microsoft MSDN Library, 2007).

Windows versions prior to Vista, including Windows XP and Windows Server 2003, provide only limited logging function. In Windows versions prior to Vista, there is a Logging tab available through the WMI Control Properties page. The Logging tab allows setting the logs location, size, and level. The WMI logs are located in the following Windows subdirectory:

```
System32\WBEM\Logs\
```

There are only three options available for the WMI logging:

```
The default logging level is Error. It will log errors only.
Second available level is Verbose. Set this to enable additional WMI logging.
The third available option is Disabled, that will disable WMI logging.
```

WMI logging is one of the areas where Windows Vista improves WMI security, comparing with previous Windows versions. Windows Vista WMI makes use of the system ACLs that can be set along the WMI namespace tree, providing improved and more flexible auditing of WMI events.

There is no Logging tab in the WMI Control Properties interface in Windows Vista. Instead, we should use previously mentioned Security tab. From the Security tab interface, select Advanced, and then Auditing, to find the option to select the audited accounts of users, groups, computers, and security principals. The audited privileges and the inheritance of the audited privileges can be specified.

By default, auditing is not enabled in Vista. We must configure auditing as described above. Please do not forget that Windows Group Policy for the local computer must be set to allow auditing. Once auditing is enabled, the Windows event log is used to log WMI audit events.

There is one more distinction as far as WMI access is concern that favours Vista over previous versions of Windows. Starting with Windows Vista, the following security groups are granted access to WMI:

Administrators LOCAL SERVICE NETWORK SERVICE Authenticated Users

Previous Windows versions used to grant WMI access to the following security groups:

Administrators LOCAL SERVICE NETWORK SERVICE Everyone

It is good to see Everyone is finally disallowed WMI access by default!

The default access rights are similar for both Windows Vista and previous Windows versions:

Execute Methods

Full Write Enable Account

8.3. Request Encrypted WMI Connections to the Server

With the permissions and the audit trail set correctly, the next important security consideration is securing server communications. We have talked a lot about securing the WMI information exchange, due to the sensitivity of some of the WMI data. We have discussed that the scripts should specify the most secure authentication option available, PktPrivacy. As it happens, some administrators may not be willing to spend the time required to make their scripts secure. Is there a way to force less diligent administrators to write their scripts to request encrypted connections?

Starting with Windows Server 2003 Service Pack 1, the answer is yes. Provider namespace security can be configured to require encryption, prior to sending WMI data back. This is a very good practice, as it allows enforcing encryption at the server end, to make sure the scripts requesting a connection to the server via WMI have to use the PktPrivacy authentication level, and to encrypt WMI data. Should the script attempt to use a lower WMI authentication level, the server will deny access.

The technical detail of setting up namespace encryption enforcement is documented in the Microsoft MSDN article Requiring an Encrypted Connection to a

Namespace (Microsoft MSDN, 2007). If you would like to see how this is done, please refer to Appendix 3.

When we specify the WMI security levels required by the server, and in the event incoming WMI connection requests have a lower (e.g. default) security setting, the setting is negotiated to the highest level required. If the WMI script has specified the explicit security level that is lower than the one required by the server system, access is denied. The WMI script is left no choice but to specify the security level high enough to be accepted by the remote system.

The default WMI security settings are best documented in the article Connecting Between Different Operating Systems (Microsoft MSDN Library, 2007, Win32 and COM Development). Below is the list of the default DCOM impersonation, authentication level, and authentication service settings for WMI access to the server (remote) system, starting with the most recent Windows Vista, and going back to WMI version 1 in Windows NT.

Operating system	Impersonation level scripting string	Authentication Level scripting string	Authentication Service
Windows Vista	Impersonate	Pkt	Kerberos
Windows Server 2003	Impersonate	Pkt	Kerberos
Windows XP Professional	Impersonate	Pkt	Kerberos
Windows 2000 (WMI 1.5)	Impersonate	Connect	Kerberos
Windows NT Server 4.0 SP4 and later (WMI 1.5)	Impersonate	Connect	NTLMDomain
Windows NT Server 4.0 SP4 and later (WMI 1.01)	Identify	Connect	NTLMDomain

8.4. Configure DCOM for Remote WMI Access

DCOM configuration for WMI access is described in the Microsoft article Securing a Remote WMI Connection (Microsoft MSDN Library, 2007, Win32 and COM Development). As we have mentioned previously, WMI security relies on the underlying DCOM security model. Windows allows administrative access to DCOM by default. When administrators perform the hardening task, and add specific scripting accounts for WMI access to specific WMI namespaces, the underlying DCOM access must be allowed.

To set this up, go to Start, and Run. Run Dcomcnfg.exe. Alternatively, navigate to Control Panel, Administrative Tools, and Component Services. This should bring up the Component Services interface. After expanding Component Services, and then Computers, check the Properties of My Computer, and the DCOM Security tab. This is where the user access is set. Select Edit Limits, and add the accounts that require access.

The article extract containing step-by-step instructions is provided in Appendix 5.

9. How Windows Vista Security Mechanisms Affect WMI Security

In this section, we discuss the configuration of the security mechanisms new to Windows Vista that affect WMI management access. The major Windows Vista security features that affect WMI security and operation include Windows Firewall configuration, User Account Control (UAC) filtering, DCOM settings, and the specific WMI security

controls implemented in Vista for the first time. Microsoft article Connecting to WMI Remotely Starting with Vista (Microsoft MSDN Library, 2007) provides an overview of Vista security features, as the new features affect the security of the WMI interface.

Windows Firewall Configuration

When the firewall is enabled on the remote host, a firewall rule (or exception) must be set on the firewall for WMI, in order to allow remote WMI access. The firewall exception rule will also work if the WMI service was configured for a fixes port, using the *winmgmt /standalonehost* command.

Here is the extract from the above referenced Microsoft article containing step-by-step instructions for enabling or disabling WMI across Windows firewall:

> 1. In the Control Panel, click Security then Windows Firewall.
> 2. Click Change Settings and click the Exceptions tab.
> 3. In the Exceptions window, select the check box for Windows Management Instrumentation (WMI) to enable WMI traffic through the firewall. To disable WMI traffic, clear the check box.

A service port lockdown always makes life easier in firewalled environments. Prior to Windows Vista, WMI was not known to be firewall friendly. This is because WMI runs using the service ports assigned through DCOM. Windows Vista allows WMI service to run as a separate host, using a specific fixed port. Clearly, this is an advantage to WMI management, particularly for firewalled environments.

WMI Port Lockdown in Vista

The procedure for configuring a fixed WMI service port is described in the Microsoft article Setting Up a Fixed Port for WMI (Microsoft MSDN Library, 2007). Here are the three simple steps to set up the fixed WMI service port, based on the Microsoft article content:

1. At the command prompt, type winmgmt -standalonehost.
2. Restart the WMI service
3. Establish a new port number for the WMI service:

    netsh firewall add portopening port=24158 name=WMIFixedPort

User Account Control Filter

With the introduction of User Account Control (UAC) in Vista, access token filter controls the operations allowed for the WMI namespaces. Quoting the above mentioned Microsoft article Connecting to WMI Remotely Starting with Vista (Microsoft MSDN Library, 2007),

"Under UAC, all accounts in the local Administrators group, run with a standard user access token, also known as UAC access token filtering. An administrator account can run a script with elevated privilege — 'Run as Administrator'. When you are not connecting to the built-in Administrator account, UAC affects connections to a remote computer differently depending on whether the two computers are in a domain or a workgroup."

DCOM Settings

DCOM settings have not changed in Windows Vista. The situation may look different in the above-mentioned case of connecting to the remote host from the local scripting host while the systems are not domain members, because of the effect of the UAC. The best approach is to grant explicit remote DCOM access, activation rights, and launch rights for your scripting account, as discussed in previous sections.

Specific WMI Security Controls New to Vista

Windows Vista added the flexibility to ease WMI management and improve WMI security by allowing granular access to many new security properties of WMI objects. WMI security properties can be set specifically on printers, services, registry keys, DCOM applications, and WMI namespaces, under the guard of the UAC. Specific permissions can be set on these security properties for the users and user groups designated for WMI management tasks, providing for flexible security architecture.

Please refer to Microsoft MSDN Library article Changing Access Security on Securable Objects (Microsoft MSDN Library, 2007) for more detail. If you are planning to implement access to the specific security descriptors, please look up Appendix 6.

10. References

1. Dickens, Charles (1991). *The Personal History of David Copperfield*. Time Warner Libraries. ISBN 1879329018.

2. Microsoft TechNet (2007), Script Centre. *WMI Security Settings*. Retrieved 29 January 2007, from http://www.microsoft.com/technet/scriptcenter/guide/sas_wmi_vzbp.mspx?mfr=true

3. Microsoft MSDN Library (2007). *Setting the Default Process Security Level Using VBScript*. Retrieved 30 January 2007, from http://msdn2.microsoft.com/en-us/library/aa393618.aspx

4. Microsoft TechNet Speech Server Library (Published 20 June 2005). *Connecting to WMI Objects*. Retrieved 31 January 007, from http://www.microsoft.com/technet/prodtechnol/speech/library/mss/sas_adm_WMI_Connecting.mspx

5. Microsoft MSDN Library (2007), Securing Scripting Clients. *Setting the Authentication Service Using VBScript.* Retrieved 2 February 2007, from http://msdn2.microsoft.com/en-us/library/aa393616.aspx

6. PJ Technologies GOVERLAN (2005). *Configuring WMI Security*. Document ID A29262214, Last Revised On 12/13/2005. Retrieved 25 January 2007, from http://www.pjtec.com/Support/knowledgeManager.cgi?action=DisplayDoc&DOCID=A29262214

7. Microsoft MSDN Library (2007), Win32 and COM Development. *Setting Namespace Security with the WMI Control*. Retrieved 3 February 2007, from http://msdn2.microsoft.com/en-us/library/aa393613.aspx

8. Microsoft United Kingdom Help and Support (2007). *How to Set WMI Namespace Security in Windows XP*. Article ID 295292, Last Review 15 January 2006, Rev. 2.2. Retrieved 1 February 2007, from http://support.microsoft.com/kb/295292. Alternative URL http://support.microsoft.com/default.aspx?scid=kb;en-us;295292

9. Microsoft MSDN Library (2007). *Access to WMI Namespaces*. Retrieved 4 February 2007, from http://msdn2.microsoft.com/en-us/library/aa822575.aspx

10. Microsoft MSDN Library (2007), Win32 and COM Development. *Connecting Between Different Operating Systems*. Retrieved 3 February 2007, from http://msdn2.microsoft.com/en-us/library/aa389284.aspx

11. Microsoft MSDN Library (2007), Win32 and COM Development. *Securing a Remote WMI Connection*. Retrieved 3 February 2007, from http://msdn2.microsoft.com/en-us/library/aa393266.aspx

12. Microsoft MSDN (2007). *Requiring an Encrypted Connection to a Namespace*. Retrieved 18 February 2007, from http://msdn2.microsoft.com/en-us/library/aa393068.aspx

13. Microsoft MSDN Library (2007). *Setting Up a Fixed Port for WMI*. Retrieved 4 February 2007, from http://msdn2.microsoft.com/en-us/library/bb219447.aspx

14. Microsoft MSDN Library (2007). *Connecting to WMI Remotely Starting with Vista*. Retrieved 4 February 2007, from http://msdn2.microsoft.com/en-us/library/aa822854.aspx

15. Microsoft MSDN Library (2007). *Changing Access Security on Securable Objects*. Retrieved 4 February 2007, from http://msdn2.microsoft.com/en-us/library/aa384905.aspx

16. Bragg, Roberta (2002). Redmondmag.com (February 2002). *Securing Remote Management with WMI*. Retrieved 24 November 2006, from http://redmondmag.com/columns/article.asp?EditorialsID=381

11. Appendixes

Appendix 1: Capture of Default WMI Request and Response

WMI query packet with the default WMI security options in use:

Packet	Source	Destination	Protocol and Information
52	10.252.253.8	10.252.253.44	DCERPC Request: call_id: 5 opnum: 20 ctx_id: 3 IWbemServices V0

Frame 52 (342 bytes on wire, 342 bytes captured)
Internet Protocol, Src: 10.252.253.8 (10.252.253.8), Dst: 10.252.253.44 (10.252.253.44)
Transmission Control Protocol, Src Port: 1716 (1716), Dst Port: 1032 (1032), Seq: 579, Ack: 465, Len: 288

DCE RPC Request, Fragment: Single, FragLen: 288, Call: 5 Ctx: 3:

00 80 c7 ea 03 27 00 17 31 2e 1f 0a 08 00 45 00
01 48 2c 6a 40 00 80 06 bd 18 0a fc fd 08 0a fc
fd 2c 06 b4 04 08 9c 14 fe 4a 69 29 90 ab 50 18
fe 2f 40 50 00 00 05 00 00 83 10 00 00 00 20 01
10 00 05 00 00 00 d8 00 00 00 03 00 14 00 1e 20
00 00 f0 03 00 00 af ee 6b 09 24 c2 3d 43 05 00
07 00 00 00 00 00 00 00 00 00 1a 91 cf 93 90 e3

```
a4 45 84 08 ff 17 88 4e db 1f 00 00 00 00 55 73
65 72 03 00 00 00 06 00 00 00 03 00 00 00 57 00
51 00 4c 00 28 0a 55 73 65 72 32 00 00 00 64 00
00 00 32 00 00 00 53 00 65 00 6c 00 65 00 63 00
74 00 20 00 2a 00 20 00 66 00 72 00 6f 00 6d 00
20 00 57 00 69 00 6e 00 33 00 32 00 5f 00 53 00
65 00 72 00 76 00 69 00 63 00 65 00 20 00 57 00
68 00 65 00 72 00 65 00 20 00 4e 00 61 00 6d 00
65 00 20 00 3d 00 20 00 27 00 54 00 6c 00 6e 00
74 00 53 00 76 00 72 00 27 00 10 01 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 0a 04
08 00 30 b9 19 00 01 00 00 00 78 84 5f cb aa c1
ce d2 01 00 00 00
```

Initial WMI response packet with the default WMI security options in use (not including the reassembly):

```
Packet  Source          Destination     Protocol and Information
73      10.252.253.44   10.252.253.8    DCERPC   Response: call_id: 8 ctx_id: 5 IWbemWCOSmartEnum
V0


Frame 73 (854 bytes on wire, 854 bytes captured)
Internet Protocol, Src: 10.252.253.44 (10.252.253.44), Dst: 10.252.253.8 (10.252.253.8)
Transmission Control Protocol, Src Port: 1032 (1032), Dst Port: 1716 (1716), Seq: 6865, Ack: 1299, Len: 800


DCE RPC Response, Fragment: Last, FragLen: 800, Call: 8 Ctx: 5:


00 17 31 2e 1f 0a 00 80 c7 ea 03 27 08 00 45 00
03 48 02 46 40 00 80 06 e5 3c 0a fc fd 2c 0a fc
fd 08 04 08 06 b4 69 29 a9 ab 9c 15 01 1a 50 18
ff ff 0e 9a 00 00 05 00 02 02 10 00 00 00 20 03
10 00 08 00 00 00 ec 02 00 00 05 00 00 00 73 65
72 76 69 63 65 20 69 73 20 73 74 6f 70 70 65 64
```

2c 20 72 65 6d 6f 74 65 20 75 73 65 72 20 61 63
63 65 73 73 20 74 6f 20 70 72 6f 67 72 61 6d 73
20 6d 69 67 68 74 20 62 65 20 75 6e 61 76 61 69
6c 61 62 6c 65 2e 20 49 66 20 74 68 69 73 20 73
65 72 76 69 63 65 20 69 73 20 64 69 73 61 62 6c
65 64 2c 20 61 6e 79 20 73 65 72 76 69 63 65 73
20 74 68 61 74 20 65 78 70 6c 69 63 69 74 6c 79
20 64 65 70 65 6e 64 20 6f 6e 20 69 74 20 77 69
6c 6c 20 66 61 69 6c 20 74 6f 20 73 74 61 72 74
2e 00 00 4f 77 6e 20 50 72 6f 63 65 73 73 00 00
4d 61 6e 75 61 6c 00 00 4e 6f 72 6d 61 6c 00 00
45 3a 5c 57 49 4e 44 4f 57 53 5c 53 79 73 74 65
6d 33 32 5c 74 6c 6e 74 73 76 72 2e 65 78 65 00
00 4c 6f 63 61 6c 53 79 73 74 65 6d 00 00 57 69
6e 33 32 5f 53 65 72 76 69 63 65 00 00 57 69 6e
33 32 5f 43 6f 6d 70 75 74 65 72 53 79 73 74 65
6d 00 00 50 49 47 41 00 00 54 6c 6e 74 53 76 72
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

```
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 0a 04
04 00 30 b9 19 00 01 00 00 00 30 3b c7 2f 4e 59
34 fd 04 00 00 00
```

WMI management session with the default WMI security options in use:

Packet	Source	Destination	Protocol and Information
3	10.252.253.8	10.252.253.44	NBNS Name query NBSTAT
4	10.252.253.44	10.252.253.8	NBNS Name query response NBSTAT
5	10.252.253.8	10.252.253.44	TCP 1713 > epmap [SYN] Seq=0 Len=0 MSS=1460
6	10.252.253.44	10.252.253.8	TCP epmap > 1713 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1260
7	10.252.253.8	10.252.253.44	TCP 1713 > epmap [ACK] Seq=1 Ack=1 Win=65535 Len=0
8	10.252.253.8	10.252.253.44	DCERPC Bind: call_id: 1 IOXIDResolver V0.0
9	10.252.253.44	10.252.253.8	TCP epmap > 1713 [ACK] Seq=1 Ack=73 Win=65463 Len=0
10	10.252.253.44	10.252.253.8	DCERPC Bind_ack: call_id: 1 accept max_xmit: 5840 max_recv: 5840
11	10.252.253.8	10.252.253.44	IOXIDResolver ServerAlive2 request
12	10.252.253.44	10.252.253.8	IOXIDResolver ServerAlive2 response[Long frame (2 bytes)]
13	10.252.253.8	10.252.253.44	TCP 1714 > epmap [SYN] Seq=0 Len=0 MSS=1460
14	10.252.253.44	10.252.253.8	TCP epmap > 1714 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1260
15	10.252.253.8	10.252.253.44	TCP 1714 > epmap [ACK] Seq=1 Ack=1 Win=65535 Len=0
16	10.252.253.8	10.252.253.44	DCERPC Bind: call_id: 2 ISystemActivator V0.0, NTLMSSP_NEGOTIATE
17	10.252.253.44	10.252.253.8	DCERPC Bind_ack: call_id: 2, NTLMSSP_CHALLENGE accept max_xmit: 5840
18	10.252.253.8	10.252.253.44	DCERPC AUTH3: call_id: 2, NTLMSSP_AUTH, User: PIGO\manager
19	10.252.253.8	10.252.253.44	ISystemActivator RemoteCreateInstance request
20	10.252.253.44	10.252.253.8	TCP epmap > 1714 [ACK] Seq=185 Ack=1147 Win=64389 Len=0
21	10.252.253.44	10.252.253.8	TCP epmap > 1714 [ACK] Seq=185 Ack=1147 Win=64389 Len=0
22	10.252.253.8	10.252.253.44	TCP 1713 > epmap [ACK] Seq=97 Ack=193 Win=65343 Len=0
23	10.252.253.44	10.252.253.8	ISystemActivator RemoteCreateInstance response
24	10.252.253.8	10.252.253.44	TCP 1715 > 1032 [SYN] Seq=0 Len=0 MSS=1460
25	10.252.253.44	10.252.253.8	TCP 1032 > 1715 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1260

26	10.252.253.8	10.252.253.44	TCP 1715 > 1032 [ACK] Seq=1 Ack=1 Win=65535 Len=0
27	10.252.253.8	10.252.253.44	DCERPC Bind: call_id: 1 IRemUnknown2 V0.0
28	10.252.253.44	10.252.253.8	TCP 1032 > 1715 [ACK] Seq=1 Ack=121 Win=65415 Len=0
29	10.252.253.8	10.252.253.44	TCP 1714 > epmap [ACK] Seq=1147 Ack=1241 Win=64295 Len=0
30	10.252.253.44	10.252.253.8	DCERPC Bind_ack: call_id: 1 accept max_xmit: 5840 max_recv: 5840
31	10.252.253.8	10.252.253.44	DCERPC AUTH3: call_id: 1
32	10.252.253.8	10.252.253.44	IRemUnknown2 RemQueryInterface request IID[1]=IWbemLoginClientID
33	10.252.253.44	10.252.253.8	TCP 1032 > 1715 [ACK] Seq=185 Ack=523 Win=65013 Len=0
34	10.252.253.44	10.252.253.8	IRemUnknown2 RemQueryInterface response S_OK[1] -> S_OK
35	10.252.253.8	10.252.253.44	DCERPC Alter_context: call_id: 2 IWbemLoginClientID V0.0
36	10.252.253.44	10.252.253.8	DCERPC Alter_context_resp: call_id: 2 accept max_xmit: 5840 max_recv: 5840
37	10.252.253.8	10.252.253.44	DCERPC Request: call_id: 2 opnum: 3 ctx_id: 1 IWbemLoginClientID V0
38	10.252.253.44	10.252.253.8	DCERPC Response: call_id: 2 ctx_id: 1 IWbemLoginClientID V0
39	10.252.253.8	10.252.253.44	TCP 1716 > 1032 [SYN] Seq=0 Len=0 MSS=1460
40	10.252.253.44	10.252.253.8	TCP 1032 > 1716 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1260
41	10.252.253.8	10.252.253.44	TCP 1716 > 1032 [ACK] Seq=1 Ack=1 Win=65535 Len=0
42	10.252.253.8	10.252.253.44	DCERPC Bind: call_id: 3 IWbemLevel1Login V0.0
43	10.252.253.44	10.252.253.8	DCERPC Bind_ack: call_id: 3 accept max_xmit: 5840 max_recv: 5840
44	10.252.253.8	10.252.253.44	DCERPC AUTH3: call_id: 3
45	10.252.253.8	10.252.253.44	DCERPC Request: call_id: 3 opnum: 6 ctx_id: 2 IWbemLevel1Login V0
46	10.252.253.44	10.252.253.8	TCP 1032 > 1716 [ACK] Seq=185 Ack=507 Win=65029 Len=0
47	10.252.253.44	10.252.253.8	DCERPC Response: call_id: 3 ctx_id: 2 IWbemLevel1Login V0
48	10.252.253.8	10.252.253.44	IRemUnknown2 RemRelease request Cnt=2 Refs=5-0,5-0[Long frame (4 bytes)]
49	10.252.253.44	10.252.253.8	IRemUnknown2 RemRelease response -> S_OK
50	10.252.253.8	10.252.253.44	DCERPC Alter_context: call_id: 5 IWbemServices V0.0
51	10.252.253.44	10.252.253.8	DCERPC Alter_context_resp: call_id: 5 accept max_xmit: 5840 max_recv: 5840
52	10.252.253.8	10.252.253.44	DCERPC Request: call_id: 5 opnum: 20 ctx_id: 3 IWbemServices V0
53	10.252.253.44	10.252.253.8	DCERPC Response: call_id: 5 ctx_id: 3 IWbemServices V0
54	10.252.253.8	10.252.253.44	IRemUnknown2 RemQueryInterface request IID[1]=IWbemFetchSmartEnum
55	10.252.253.44	10.252.253.8	IRemUnknown2 RemQueryInterface response S_OK[1] -> S_OK
56	10.252.253.8	10.252.253.44	DCERPC Alter_context: call_id: 7 IWbemFetchSmartEnum V0.0
57	10.252.253.44	10.252.253.8	DCERPC Alter_context_resp: call_id: 7 accept max_xmit: 5840 max_recv: 5840
58	10.252.253.8	10.252.253.44	DCERPC Request: call_id: 7 opnum: 3 ctx_id: 4 IWbemFetchSmartEnum V0
59	10.252.253.44	10.252.253.8	DCERPC Response: call_id: 7 ctx_id: 4 IWbemFetchSmartEnum V0
60	10.252.253.8	10.252.253.44	DCERPC Alter_context: call_id: 8 IWbemWCOSmartEnum V0.0
61	10.252.253.44	10.252.253.8	DCERPC Alter_context_resp: call_id: 8 accept max_xmit: 5840 max_recv: 5840
62	10.252.253.8	10.252.253.44	DCERPC Request: call_id: 8 opnum: 3 ctx_id: 5 IWbemWCOSmartEnum V0
63	10.252.253.44	10.252.253.8	TCP 1032 > 1716 [ACK] Seq=1025 Ack=1299 Win=65535 Len=0
64	10.252.253.8	10.252.253.44	TCP 1715 > 1032 [ACK] Seq=1107 Ack=625 Win=64911 Len=0
65	10.252.253.44	10.252.253.8	TCP [TCP segment of a reassembled PDU]
66	10.252.253.44	10.252.253.8	TCP [TCP segment of a reassembled PDU]

67	10.252.253.8	10.252.253.44	TCP	1716 > 1032 [ACK] Seq=1299 Ack=3545 Win=65535 Len=0
68	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
69	10.252.253.8	10.252.253.44	TCP	1716 > 1032 [ACK] Seq=1299 Ack=4805 Win=65535 Len=0
70	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
71	10.252.253.44	10.252.253.8	DCERPC	Response: call_id: 8 ctx_id: 5 [DCE/RPC first fragment, reas: #73]
72	10.252.253.8	10.252.253.44	TCP	1716 > 1032 [ACK] Seq=1299 Ack=6865 Win=65535 Len=0
73	10.252.253.44	10.252.253.8	DCERPC	Response: call_id: 8 ctx_id: 5 IWbemWCOSmartEnum V0
74	10.252.253.8	10.252.253.44	DCERPC	Request: call_id: 9 opnum: 6 ctx_id: 3 IWbemServices V0
75	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
76	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
77	10.252.253.8	10.252.253.44	TCP	1716 > 1032 [ACK] Seq=1491 Ack=10185 Win=65535 Len=0
78	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
79	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
80	10.252.253.8	10.252.253.44	TCP	1716 > 1032 [ACK] Seq=1491 Ack=12705 Win=65535 Len=0
81	10.252.253.44	10.252.253.8	DCERPC	Response: call_id: 9 ctx_id: 3 [DCE/RPC first fragment, reas: #153]
82	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
83	10.252.253.8	10.252.253.44	TCP	1716 > 1032 [ACK] Seq=1491 Ack=14765 Win=65535 Len=0
84	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
85	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
86	10.252.253.8	10.252.253.44	TCP	1716 > 1032 [ACK] Seq=1491 Ack=17285 Win=65535 Len=0
87	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
88	10.252.253.44	10.252.253.8	DCERPC	Response: call_id: 9 ctx_id: 3 [DCE/RPC middle fragment, reas: #153]
89	10.252.253.8	10.252.253.44	TCP	1716 > 1032 [ACK] Seq=1491 Ack=19345 Win=65535 Len=0
90	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
91	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
92	10.252.253.8	10.252.253.44	TCP	1716 > 1032 [ACK] Seq=1491 Ack=21865 Win=65535 Len=0
93	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
94	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
95	10.252.253.8	10.252.253.44	TCP	1716 > 1032 [ACK] Seq=1491 Ack=24385 Win=65535 Len=0
96	10.252.253.44	10.252.253.8	DCERPC	Response: call_id: 9 ctx_id: 3 [DCE/RPC middle fragment, reas: #153]
97	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
98	10.252.253.8	10.252.253.44	TCP	1716 > 1032 [ACK] Seq=1491 Ack=26445 Win=65535 Len=0
99	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
100	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
101	10.252.253.8	10.252.253.44	TCP	1716 > 1032 [ACK] Seq=1491 Ack=28965 Win=65535 Len=0
102	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
103	10.252.253.44	10.252.253.8	DCERPC	Response: call_id: 9 ctx_id: 3 [DCE/RPC middle fragment, reas: #153]
104	10.252.253.8	10.252.253.44	TCP	1716 > 1032 [ACK] Seq=1491 Ack=31025 Win=65535 Len=0
105	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
106	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
107	10.252.253.8	10.252.253.44	TCP	1716 > 1032 [ACK] Seq=1491 Ack=33545 Win=65535 Len=0

108	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
109	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
110	10.252.253.8	10.252.253.44	TCP	1716 > 1032 [ACK] Seq=1491 Ack=36065 Win=65535 Len=0
111	10.252.253.44	10.252.253.8	DCERPC	Response: call_id: 9 ctx_id: 3 [DCE/RPC middle fragment, reas: #153]
112	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
113	10.252.253.8	10.252.253.44	TCP	1716 > 1032 [ACK] Seq=1491 Ack=38125 Win=65535 Len=0
114	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
115	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
116	10.252.253.8	10.252.253.44	TCP	1716 > 1032 [ACK] Seq=1491 Ack=40645 Win=65535 Len=0
117	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
118	10.252.253.44	10.252.253.8	DCERPC	Response: call_id: 9 ctx_id: 3 [DCE/RPC middle fragment, reas: #153]
119	10.252.253.8	10.252.253.44	TCP	1716 > 1032 [ACK] Seq=1491 Ack=42705 Win=65535 Len=0
120	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
121	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
122	10.252.253.8	10.252.253.44	TCP	1716 > 1032 [ACK] Seq=1491 Ack=45225 Win=65535 Len=0
123	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
124	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
125	10.252.253.8	10.252.253.44	TCP	1716 > 1032 [ACK] Seq=1491 Ack=47745 Win=65535 Len=0
126	10.252.253.44	10.252.253.8	DCERPC	Response: call_id: 9 ctx_id: 3 [DCE/RPC middle fragment, reas: #153]
127	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
128	10.252.253.8	10.252.253.44	TCP	1716 > 1032 [ACK] Seq=1491 Ack=49805 Win=65535 Len=0
129	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
130	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
131	10.252.253.8	10.252.253.44	TCP	1716 > 1032 [ACK] Seq=1491 Ack=52325 Win=65535 Len=0
132	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
133	10.252.253.44	10.252.253.8	DCERPC	Response: call_id: 9 ctx_id: 3 [DCE/RPC middle fragment, reas: #153]
134	10.252.253.8	10.252.253.44	TCP	1716 > 1032 [ACK] Seq=1491 Ack=54385 Win=65535 Len=0
135	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
136	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
137	10.252.253.8	10.252.253.44	TCP	1716 > 1032 [ACK] Seq=1491 Ack=56905 Win=65535 Len=0
138	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
139	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
140	10.252.253.8	10.252.253.44	TCP	1716 > 1032 [ACK] Seq=1491 Ack=59425 Win=65535 Len=0
141	10.252.253.44	10.252.253.8	DCERPC	Response: call_id: 9 ctx_id: 3 [DCE/RPC middle fragment, reas: #153]
142	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
143	10.252.253.8	10.252.253.44	TCP	1716 > 1032 [ACK] Seq=1491 Ack=61485 Win=65535 Len=0
144	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
145	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
146	10.252.253.8	10.252.253.44	TCP	1716 > 1032 [ACK] Seq=1491 Ack=64005 Win=65535 Len=0
147	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
148	10.252.253.44	10.252.253.8	DCERPC	Response: call_id: 9 ctx_id: 3 [DCE/RPC middle fragment, reas: #153]

149	10.252.253.8	10.252.253.44	TCP	1716 > 1032 [ACK] Seq=1491 Ack=66065 Win=65535 Len=0
150	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
151	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
152	10.252.253.8	10.252.253.44	TCP	1716 > 1032 [ACK] Seq=1491 Ack=68585 Win=65535 Len=0
153	10.252.253.44	10.252.253.8	DCERPC	Response: call_id: 9 ctx_id: 3 IWbemServices V0
154	10.252.253.8	10.252.253.44	DCERPC	Request: call_id: 10 opnum: 24 ctx_id: 3 IWbemServices V0
155	10.252.253.44	10.252.253.8	TCP	1032 > 1716 [ACK] Seq=69473 Ack=1763 Win=65071 Len=0
156	10.252.253.44	10.252.253.8	DCERPC	Response: call_id: 10 ctx_id: 3 IWbemServices V0
157	10.252.253.8	10.252.253.44	TCP	1716 > 1032 [ACK] Seq=1763 Ack=69953 Win=65535 Len=0
158	10.252.253.8	10.252.253.44	DCERPC	Request: call_id: 11 opnum: 3 ctx_id: 5 IWbemWCOSmartEnum V0
159	10.252.253.44	10.252.253.8	DCERPC	Response: call_id: 11 ctx_id: 5 IWbemWCOSmartEnum V0
160	10.252.253.8	10.252.253.44	IRemUnknown2 RemRelease request Cnt=2 Refs=5-0,5-0[Long frame (4 bytes)]	
161	10.252.253.44	10.252.253.8	TCP	1032 > 1715 [ACK] Seq=625 Ack=1267 Win=65535 Len=0
162	10.252.253.44	10.252.253.8	IRemUnknown2 RemRelease response -> S_OK	
163	10.252.253.8	10.252.253.44	IRemUnknown2 RemRelease request Cnt=1 Refs=5-0[Long frame (4 bytes)]	
164	10.252.253.44	10.252.253.8	IRemUnknown2 RemRelease response -> S_OK	
165	10.252.253.8	10.252.253.44	IRemUnknown2 RemRelease request Cnt=1 Refs=5-0[Long frame (4 bytes)]	
166	10.252.253.44	10.252.253.8	IRemUnknown2 RemRelease response -> S_OK	
167	10.252.253.8	10.252.253.44	TCP	1715 > 1032 [FIN, ACK] Seq=1555 Ack=817 Win=64719 Len=0
168	10.252.253.8	10.252.253.44	TCP	1716 > 1032 [FIN, ACK] Seq=1955 Ack=70033 Win=65455 Len=0
169	10.252.253.44	10.252.253.8	TCP	1032 > 1715 [ACK] Seq=817 Ack=1556 Win=65247 Len=0
170	10.252.253.44	10.252.253.8	TCP	1032 > 1716 [ACK] Seq=70033 Ack=1956 Win=64879 Len=0
171	10.252.253.44	10.252.253.8	TCP	1032 > 1715 [FIN, ACK] Seq=817 Ack=1556 Win=65247 Len=0
172	10.252.253.8	10.252.253.44	TCP	1715 > 1032 [ACK] Seq=1556 Ack=818 Win=64719 Len=0
173	10.252.253.44	10.252.253.8	TCP	1032 > 1716 [FIN, ACK] Seq=70033 Ack=1956 Win=64879 Len=0
174	10.252.253.8	10.252.253.44	TCP	1716 > 1032 [ACK] Seq=1956 Ack=70034 Win=65455 Len=0
175	10.252.253.8	10.252.253.44	TCP	1714 > epmap [FIN, ACK] Seq=1147 Ack=1241 Win=64295 Len=0
176	10.252.253.8	10.252.253.44	TCP	1713 > epmap [RST, ACK] Seq=97 Ack=193 Win=0 Len=0
177	10.252.253.44	10.252.253.8	TCP	epmap > 1714 [ACK] Seq=1241 Ack=1148 Win=64389 Len=0
178	10.252.253.44	10.252.253.8	TCP	epmap > 1714 [FIN, ACK] Seq=1241 Ack=1148 Win=64389 Len=0
179	10.252.253.8	10.252.253.44	TCP	1714 > epmap [ACK] Seq=1148 Ack=1242 Win=64295 Len=0

Appendix 2: Secure WMI Information Exchange on the Wire

WMI query packet with the recommended base WMI security options set explicitly:

Packet	Source	Destination	Protocol	Info

52 10.252.253.8 10.252.253.44 DCERPC Request: call_id: 5 opnum: 20 ctx_id: 3 IWbemServices V0

Frame 52 (406 bytes on wire, 406 bytes captured)

Internet Protocol, Src: 10.252.253.8 (10.252.253.8), Dst: 10.252.253.44 (10.252.253.44)
Transmission Control Protocol, Src Port: 1842 (1842), Dst Port: 1032 (1032), Seq: 315, Ack: 185, Len: 352

DCE RPC Request, Fragment: Single, FragLen: 352, Call: 5 Ctx: 3:

Frame (406 bytes):

00 80 c7 ea 03 27 00 17 31 2e 1f 0a 08 00 45 00
01 88 34 ad 40 00 80 06 b4 95 0a fc fd 08 0a fc
fd 2c 07 32 04 08 9a ed 4f 00 3f c0 4a 8b 50 18
ff 47 c3 5a 00 00 05 00 00 83 10 00 00 00 60 01
10 00 05 00 00 00 14 01 00 00 03 00 14 00 12 5c
00 00 f0 03 00 00 bf 4f 10 82 9e 42 46 2f b7 84
ac 9d 85 f7 20 62 34 da b2 77 f8 fa da 21 2f 9a
57 16 48 9e 66 db b2 37 8e ae c7 50 b6 9b 6e e8
72 e4 2b 17 09 f1 0e 14 8d f2 a0 a9 7e bd f7 42
a0 e2 a3 b6 28 d3 dd 43 af 06 c7 ce d3 03 91 5e
72 97 03 09 92 1d f8 e6 c0 f9 81 14 b5 eb 6f 14
ad 1a a3 9a a1 1c a1 c7 0f e0 0c fb b5 1d 82 e8
01 c2 68 73 7c 8a ff 57 74 b7 24 87 7f 57 53 55
65 6a 8b 64 7c 31 48 cc ae 05 74 ae 30 b3 75 cc
09 16 9a c0 7d f4 bd 3d 47 6d 50 df 46 f1 8a a5
2a b2 ca 83 04 f7 d0 b0 26 42 59 49 93 46 88 e3
ce 34 4c 04 2b 0e 13 77 27 50 24 58 dc ec 78 31
75 9d 80 7a fd 59 0f 2b b6 8d 82 89 b9 99 7e 80
a5 96 6e db e7 1d 66 01 97 74 24 6c 92 91 13 b3
31 bb 70 95 a9 22 d0 4b cd c5 e7 85 19 da 92 92
e0 e5 6e 34 e7 d6 2f 08 9b fb f9 c7 a5 42 27 1e
6c 08 d6 ec 06 84 e1 de 22 f9 ee d3 41 e0 b3 a9
cd ba 2e 8f d4 20 a5 2f 42 cc 53 57 aa e6 55 17
2c fc 10 c8 f8 6d 73 bb 44 27 8f 38 91 f9 0a 06

```
0c 00 20 55 19 00 01 00 00 00 78 77 ca fc d0 3d
4d 99 00 00 00 00
```

Initial WMI response packet with the recommended base WMI security options set explicitly (not including the reassembly):

```
Packet  Source          Destination     Protocol Info
79      10.252.253.44   10.252.253.8    DCERPC   Response: call_id: 8 ctx_id: 5 IWbemWCOSmartEnum
V0

Frame 79 (854 bytes on wire, 854 bytes captured)


Internet Protocol, Src: 10.252.253.44 (10.252.253.44), Dst: 10.252.253.8 (10.252.253.8)
Transmission Control Protocol, Src Port: 1032 (1032), Dst Port: 1842 (1842), Seq: 6769, Ack: 1491, Len: 800


DCE RPC Response, Fragment: Last, FragLen: 800, Call: 8 Ctx: 5:


Frame (854 bytes):


00 17 31 2e 1f 0a 00 80 c7 ea 03 27 08 00 45 00
03 48 03 0c 40 00 80 06 e4 76 0a fc fd 2c 0a fc
fd 08 04 08 07 32 3f c0 64 43 9a ed 53 98 50 18
ff ff cb 14 00 00 05 00 02 02 10 00 00 00 20 03
10 00 08 00 00 00 ec 02 00 00 05 00 00 00 94 29
54 95 77 f0 bc 9f 7b ab 7d 84 3f 0a af ab d7 98
dd 66 6f 2a a0 9e a0 67 23 65 05 e6 31 ac 63 70
d3 ca 3e 44 15 d7 8a 7a b9 3b 54 77 2f 90 6c 6d
ad b7 80 d1 8d f8 61 57 a1 c9 c9 61 25 ea 7c 84
d8 51 6d 07 58 23 ca 88 c9 6d 5b 54 f7 24 93 b6
bd 41 fa 5f 89 ad 2e 2c 8c 88 a3 d2 d7 b3 d7 eb
49 23 ad fe 48 d8 3a b2 cd 4e 47 73 a1 5f d7 3d
30 c4 e5 73 28 eb a6 dc 85 00 e4 d5 85 6a d3 02
45 9f e4 7e 9c 7f a6 2d d6 c6 00 b3 d5 9c 5f d8
cb a7 61 00 6d 18 6e 1b 70 21 6d 6b b0 cd a5 8d
73 2c 14 9f 3d 26 25 08 b1 6b a7 35 cb 72 09 94
e8 94 68 1f f6 03 c2 70 5e 70 a3 04 2a 69 79 75
```

```
56 9b a8 57 dd af 2c a7 af 86 ab 65 f5 4b 2c bf
dd db d5 d8 37 e4 9b dd 28 b6 f3 52 36 bf 3a 89
3b cc 52 be c4 56 23 62 ca af ce e0 44 2b ea 50
6f ef f4 dd c5 8b 49 32 2e 5a 66 8c 60 c3 f5 86
3e b4 67 6a 94 4b 43 b4 6f 40 49 3c bb a2 14 44
6a 0a 2e 3c d4 7f c1 ba 79 ba 1a 99 02 8d 02 22
bb 41 d3 00 cd 0c ed 70 00 a1 9b 84 6f 65 b4 b9
2c a2 ee 78 71 7f 27 8c 47 fa ed 6b 72 dc 35 fd
7e b8 43 af 02 67 a9 6f 38 1e 61 97 02 99 ed 89
54 90 82 4a 8c 4f 69 9f cf 20 fc 73 20 e3 5e c9
fe 42 95 13 f1 8c 04 59 3f 82 20 28 b5 cc c2 e2
db 5e 63 69 32 4e 77 13 f4 0b a5 58 a6 b0 da b4
73 73 0c 83 22 76 3b 40 53 b1 3d e5 f7 77 de 55
69 34 13 b3 a7 a4 e5 ea 0b 2f ba 7e 88 da da 71
9b 3d e9 65 b6 7f e4 55 1f d3 7b 6f ed a1 60 8a
a0 f6 9c f3 f5 34 fb 8e 19 f8 38 c6 c8 fb ec 9e
39 3f 11 45 4a d0 62 35 ba fa dc 6f bc 66 fd 2c
a2 de ec 52 59 bb 37 6b 3e 0e 93 b7 cd dd 05 e7
f0 9d b5 0d 90 6d d8 2d 42 6a e6 b4 8c 53 b5 11
82 84 78 7e 66 3d cc 5b 59 1b e4 2d 7d e1 85 30
ed 9e 70 1c 2f 86 a0 fe a5 10 c4 d8 81 a2 5e 34
20 66 6b 1d e4 44 f9 ed cd 5c 1c cf 8d 5d 32 5d
7f ab 25 23 7b ff 42 46 ad 3a 8b fb c2 bc f5 2b
3e 1e 45 97 9d 62 77 c5 6e fa 46 be ba 86 f9 a9
df e4 bf 09 16 a0 f2 1d 36 36 0a c0 dc 43 80 52
c7 a5 6a 40 30 68 73 5a 0a bd af 7d e3 1d a5 bc
22 26 1c 6a 56 79 28 6b 2f ea 79 3c 62 3e be c8
7c 55 1b d4 09 02 93 b9 3f 2a 1e 89 88 9b 44 d1
38 05 a7 40 7f 56 bf e6 55 aa a4 fe 41 9f 66 03
f0 e3 46 93 cc d2 e9 2f 26 c1 f4 95 ef 4f 88 70
e7 d7 05 d1 c4 ef d3 1d b7 6c 53 a7 aa 09 8a 23
89 fa f6 57 97 e8 0d c1 eb 62 1d 3b 7e 5c 28 c5
fb 71 22 fe 16 74 18 25 47 a4 12 ee 21 38 63 fa
31 6e 5b 10 98 c8 13 79 42 d9 6e 0f 48 c9 b3 28
fa ef 86 50 a3 09 15 8f bc a8 9a bf 31 00 0a 06
04 00 20 55 19 00 01 00 00 00 df a7 16 67 65 6f
```

6b a3 04 00 00 00

WMI management session with the recommended base WMI security options set explicitly:

Packet	Source	Destination	Protocol and Information
3	10.252.253.8	10.252.253.44	NBNS Name query NBSTAT
4	10.252.253.44	10.252.253.8	NBNS Name query response NBSTAT
5	10.252.253.8	10.252.253.44	TCP 1838 > epmap [SYN] Seq=0 Len=0 MSS=1460
6	10.252.253.44	10.252.253.8	TCP epmap > 1838 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1260
7	10.252.253.8	10.252.253.44	TCP 1838 > epmap [ACK] Seq=1 Ack=1 Win=65535 Len=0
8	10.252.253.8	10.252.253.44	DCERPC Bind: call_id: 1 IOXIDResolver V0.0
9	10.252.253.44	10.252.253.8	DCERPC Bind_ack: call_id: 1 accept max_xmit: 5840 max_recv: 5840
10	10.252.253.8	10.252.253.44	IOXIDResolver ServerAlive2 request
11	10.252.253.44	10.252.253.8	IOXIDResolver ServerAlive2 response[Long frame (2 bytes)]
12	10.252.253.8	10.252.253.44	TCP 1839 > epmap [SYN] Seq=0 Len=0 MSS=1460
13	10.252.253.44	10.252.253.8	TCP epmap > 1839 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1260
14	10.252.253.8	10.252.253.44	TCP 1839 > epmap [ACK] Seq=1 Ack=1 Win=65535 Len=0
15	10.252.253.8	10.252.253.44	DCERPC Bind: call_id: 2 ISystemActivator V0.0, NTLMSSP_NEGOTIATE
16	10.252.253.44	10.252.253.8	DCERPC Bind_ack: call_id: 2, NTLMSSP_CHALLENGE accept max_xmit: 5840
17	10.252.253.8	10.252.253.44	DCERPC AUTH3: call_id: 2, NTLMSSP_AUTH, User: PIGO\manager
18	10.252.253.8	10.252.253.44	ISystemActivator RemoteCreateInstance request
19	10.252.253.44	10.252.253.8	TCP epmap > 1839 [ACK] Seq=185 Ack=1147 Win=64389 Len=0
20	10.252.253.44	10.252.253.8	TCP epmap > 1839 [ACK] Seq=185 Ack=1147 Win=64389 Len=0
21	10.252.253.44	10.252.253.8	ISystemActivator RemoteCreateInstance response
22	10.252.253.8	10.252.253.44	TCP 1840 > 1032 [SYN] Seq=0 Len=0 MSS=1460
23	10.252.253.44	10.252.253.8	TCP 1032 > 1840 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1260
24	10.252.253.8	10.252.253.44	TCP 1840 > 1032 [ACK] Seq=1 Ack=1 Win=65535 Len=0
25	10.252.253.8	10.252.253.44	DCERPC Bind: call_id: 1 IRemUnknown2 V0.0
26	10.252.253.44	10.252.253.8	DCERPC Bind_ack: call_id: 1 accept max_xmit: 5840 max_recv: 5840
27	10.252.253.8	10.252.253.44	DCERPC AUTH3: call_id: 1
28	10.252.253.8	10.252.253.44	IRemUnknown2 RemQueryInterface request IID[1]=IWbemLoginClientID
29	10.252.253.44	10.252.253.8	TCP 1032 > 1840 [ACK] Seq=185 Ack=523 Win=65013 Len=0
30	10.252.253.44	10.252.253.8	IRemUnknown2 RemQueryInterface response S_OK[1] -> S_OK
31	10.252.253.8	10.252.253.44	DCERPC Alter_context: call_id: 2 IWbemLoginClientID V0.0
32	10.252.253.44	10.252.253.8	DCERPC Alter_context_resp: call_id: 2 accept max_xmit: 5840 max_recv: 5840
33	10.252.253.8	10.252.253.44	DCERPC Request: call_id: 2 opnum: 3 ctx_id: 1 IWbemLoginClientID V0
34	10.252.253.44	10.252.253.8	DCERPC Response: call_id: 2 ctx_id: 1 IWbemLoginClientID V0
35	10.252.253.8	10.252.253.44	TCP 1841 > 1032 [SYN] Seq=0 Len=0 MSS=1460
36	10.252.253.44	10.252.253.8	TCP 1032 > 1841 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1260

37	10.252.253.8	10.252.253.44	TCP 1841 > 1032 [ACK] Seq=1 Ack=1 Win=65535 Len=0
38	10.252.253.8	10.252.253.44	DCERPC Bind: call_id: 3 IWbemLevel1Login V0.0
39	10.252.253.44	10.252.253.8	DCERPC Bind_ack: call_id: 3 accept max_xmit: 5840 max_recv: 5840
40	10.252.253.8	10.252.253.44	DCERPC AUTH3: call_id: 3
41	10.252.253.8	10.252.253.44	DCERPC Request: call_id: 3 opnum: 6 ctx_id: 2 IWbemLevel1Login V0
42	10.252.253.44	10.252.253.8	TCP 1032 > 1841 [ACK] Seq=185 Ack=507 Win=65029 Len=0
43	10.252.253.44	10.252.253.8	DCERPC Response: call_id: 3 ctx_id: 2 IWbemLevel1Login V0
44	10.252.253.8	10.252.253.44	IRemUnknown2 RemRelease request Cnt=2 Refs=5-0,5-0[Long frame (4 bytes)]
45	10.252.253.44	10.252.253.8	IRemUnknown2 RemRelease response -> S_OK
46	10.252.253.8	10.252.253.44	TCP 1842 > 1032 [SYN] Seq=0 Len=0 MSS=1460
47	10.252.253.44	10.252.253.8	TCP 1032 > 1842 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1260
48	10.252.253.8	10.252.253.44	TCP 1842 > 1032 [ACK] Seq=1 Ack=1 Win=65535 Len=0
49	10.252.253.8	10.252.253.44	DCERPC Bind: call_id: 5 IWbemServices V0.0, NTLMSSP_NEGOTIATE
50	10.252.253.44	10.252.253.8	DCERPC Bind_ack: call_id: 5, NTLMSSP_CHALLENGE accept max_xmit: 5840
51	10.252.253.8	10.252.253.44	DCERPC AUTH3: call_id: 5, NTLMSSP_AUTH, User: PIGO\manager
52	10.252.253.8	10.252.253.44	DCERPC Request: call_id: 5 opnum: 20 ctx_id: 3 IWbemServices V0
53	10.252.253.44	10.252.253.8	TCP 1032 > 1842 [ACK] Seq=185 Ack=667 Win=64869 Len=0
54	10.252.253.44	10.252.253.8	DCERPC Response: call_id: 5 ctx_id: 3 IWbemServices V0
55	10.252.253.8	10.252.253.44	DCERPC Alter_context: call_id: 6 IRemUnknown2 V0.0
56	10.252.253.44	10.252.253.8	DCERPC Alter_context_resp: call_id: 6 accept max_xmit: 5840 max_recv: 5840
57	10.252.253.8	10.252.253.44	IRemUnknown2 RemQueryInterface request[Malformed Packet]
58	10.252.253.44	10.252.253.8	IRemUnknown2 RemQueryInterface response[Malformed Packet]
59	10.252.253.8	10.252.253.44	DCERPC Alter_context: call_id: 7 IWbemFetchSmartEnum V0.0
60	10.252.253.44	10.252.253.8	DCERPC Alter_context_resp: call_id: 7 accept max_xmit: 5840 max_recv: 5840
61	10.252.253.8	10.252.253.44	DCERPC Request: call_id: 7 opnum: 3 ctx_id: 4 IWbemFetchSmartEnum V0
62	10.252.253.8	10.252.253.44	TCP 1838 > epmap [ACK] Seq=97 Ack=193 Win=65343 Len=0
63	10.252.253.8	10.252.253.44	TCP 1839 > epmap [ACK] Seq=1147 Ack=1241 Win=64295 Len=0
64	10.252.253.44	10.252.253.8	DCERPC Response: call_id: 7 ctx_id: 4 IWbemFetchSmartEnum V0
65	10.252.253.8	10.252.253.44	DCERPC Alter_context: call_id: 8 IWbemWCOSmartEnum V0.0
66	10.252.253.44	10.252.253.8	DCERPC Alter_context_resp: call_id: 8 accept max_xmit: 5840 max_recv: 5840
67	10.252.253.8	10.252.253.44	DCERPC Request: call_id: 8 opnum: 3 ctx_id: 5 IWbemWCOSmartEnum V0
68	10.252.253.44	10.252.253.8	TCP 1032 > 1842 [ACK] Seq=929 Ack=1491 Win=65535 Len=0
69	10.252.253.8	10.252.253.44	TCP 1840 > 1032 [ACK] Seq=899 Ack=497 Win=65039 Len=0
70	10.252.253.8	10.252.253.44	TCP 1841 > 1032 [ACK] Seq=507 Ack=409 Win=65127 Len=0
71	10.252.253.44	10.252.253.8	TCP [TCP segment of a reassembled PDU]
72	10.252.253.44	10.252.253.8	TCP [TCP segment of a reassembled PDU]
73	10.252.253.8	10.252.253.44	TCP 1842 > 1032 [ACK] Seq=1491 Ack=3449 Win=65535 Len=0
74	10.252.253.44	10.252.253.8	TCP [TCP segment of a reassembled PDU]
75	10.252.253.8	10.252.253.44	TCP 1842 > 1032 [ACK] Seq=1491 Ack=4709 Win=65535 Len=0
76	10.252.253.44	10.252.253.8	TCP [TCP segment of a reassembled PDU]
77	10.252.253.44	10.252.253.8	DCERPC Response: call_id: 8 ctx_id: 5 [DCE/RPC first fragment, reas: #79]

78	10.252.253.8	10.252.253.44	TCP	1842 > 1032 [ACK] Seq=1491 Ack=6769 Win=65535 Len=0
79	10.252.253.44	10.252.253.8	DCERPC	Response: call_id: 8 ctx_id: 5 IWbemWCOSmartEnum V0
80	10.252.253.8	10.252.253.44	DCERPC	Request: call_id: 9 opnum: 6 ctx_id: 3 IWbemServices V0
81	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
82	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
83	10.252.253.8	10.252.253.44	TCP	1842 > 1032 [ACK] Seq=1683 Ack=10089 Win=65535 Len=0
84	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
85	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
86	10.252.253.8	10.252.253.44	TCP	1842 > 1032 [ACK] Seq=1683 Ack=12609 Win=65535 Len=0
87	10.252.253.44	10.252.253.8	DCERPC	Response: call_id: 9 ctx_id: 3 [DCE/RPC first fragment, reas: #159]
88	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
89	10.252.253.8	10.252.253.44	TCP	1842 > 1032 [ACK] Seq=1683 Ack=14669 Win=65535 Len=0
90	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
91	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
92	10.252.253.8	10.252.253.44	TCP	1842 > 1032 [ACK] Seq=1683 Ack=17189 Win=65535 Len=0
93	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
94	10.252.253.44	10.252.253.8	DCERPC	Response: call_id: 9 ctx_id: 3 [DCE/RPC middle fragment, reas: #159]
95	10.252.253.8	10.252.253.44	TCP	1842 > 1032 [ACK] Seq=1683 Ack=19249 Win=65535 Len=0
96	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
97	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
98	10.252.253.8	10.252.253.44	TCP	1842 > 1032 [ACK] Seq=1683 Ack=21769 Win=65535 Len=0
99	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
100	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
101	10.252.253.8	10.252.253.44	TCP	1842 > 1032 [ACK] Seq=1683 Ack=24289 Win=65535 Len=0
102	10.252.253.44	10.252.253.8	DCERPC	Response: call_id: 9 ctx_id: 3 [DCE/RPC middle fragment, reas: #159]
103	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
104	10.252.253.8	10.252.253.44	TCP	1842 > 1032 [ACK] Seq=1683 Ack=26349 Win=65535 Len=0
105	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
106	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
107	10.252.253.8	10.252.253.44	TCP	1842 > 1032 [ACK] Seq=1683 Ack=28869 Win=65535 Len=0
108	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
109	10.252.253.44	10.252.253.8	DCERPC	Response: call_id: 9 ctx_id: 3 [DCE/RPC middle fragment, reas: #159]
110	10.252.253.8	10.252.253.44	TCP	1842 > 1032 [ACK] Seq=1683 Ack=30929 Win=65535 Len=0
111	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
112	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
113	10.252.253.8	10.252.253.44	TCP	1842 > 1032 [ACK] Seq=1683 Ack=33449 Win=65535 Len=0
114	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
115	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
116	10.252.253.8	10.252.253.44	TCP	1842 > 1032 [ACK] Seq=1683 Ack=35969 Win=65535 Len=0
117	10.252.253.44	10.252.253.8	DCERPC	Response: call_id: 9 ctx_id: 3 [DCE/RPC middle fragment, reas: #159]
118	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]

119	10.252.253.8	10.252.253.44	TCP	1842 > 1032 [ACK] Seq=1683 Ack=38029 Win=65535 Len=0
120	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
121	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
122	10.252.253.8	10.252.253.44	TCP	1842 > 1032 [ACK] Seq=1683 Ack=40549 Win=65535 Len=0
123	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
124	10.252.253.44	10.252.253.8	DCERPC	Response: call_id: 9 ctx_id: 3 [DCE/RPC middle fragment, reas: #159]
125	10.252.253.8	10.252.253.44	TCP	1842 > 1032 [ACK] Seq=1683 Ack=42609 Win=65535 Len=0
126	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
127	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
128	10.252.253.8	10.252.253.44	TCP	1842 > 1032 [ACK] Seq=1683 Ack=45129 Win=65535 Len=0
129	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
130	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
131	10.252.253.8	10.252.253.44	TCP	1842 > 1032 [ACK] Seq=1683 Ack=47649 Win=65535 Len=0
132	10.252.253.44	10.252.253.8	DCERPC	Response: call_id: 9 ctx_id: 3 [DCE/RPC middle fragment, reas: #159]
133	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
134	10.252.253.8	10.252.253.44	TCP	1842 > 1032 [ACK] Seq=1683 Ack=49709 Win=65535 Len=0
135	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
136	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
137	10.252.253.8	10.252.253.44	TCP	1842 > 1032 [ACK] Seq=1683 Ack=52229 Win=65535 Len=0
138	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
139	10.252.253.44	10.252.253.8	DCERPC	Response: call_id: 9 ctx_id: 3 [DCE/RPC middle fragment, reas: #159]
140	10.252.253.8	10.252.253.44	TCP	1842 > 1032 [ACK] Seq=1683 Ack=54289 Win=65535 Len=0
141	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
142	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
143	10.252.253.8	10.252.253.44	TCP	1842 > 1032 [ACK] Seq=1683 Ack=56809 Win=65535 Len=0
144	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
145	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
146	10.252.253.8	10.252.253.44	TCP	1842 > 1032 [ACK] Seq=1683 Ack=59329 Win=65535 Len=0
147	10.252.253.44	10.252.253.8	DCERPC	Response: call_id: 9 ctx_id: 3 [DCE/RPC middle fragment, reas: #159]
148	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
149	10.252.253.8	10.252.253.44	TCP	1842 > 1032 [ACK] Seq=1683 Ack=61389 Win=65535 Len=0
150	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
151	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
152	10.252.253.8	10.252.253.44	TCP	1842 > 1032 [ACK] Seq=1683 Ack=63909 Win=65535 Len=0
153	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
154	10.252.253.44	10.252.253.8	DCERPC	Response: call_id: 9 ctx_id: 3 [DCE/RPC middle fragment, reas: #159]
155	10.252.253.8	10.252.253.44	TCP	1842 > 1032 [ACK] Seq=1683 Ack=65969 Win=65535 Len=0
156	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
157	10.252.253.44	10.252.253.8	TCP	[TCP segment of a reassembled PDU]
158	10.252.253.8	10.252.253.44	TCP	1842 > 1032 [ACK] Seq=1683 Ack=68489 Win=65535 Len=0
159	10.252.253.44	10.252.253.8	DCERPC	Response: call_id: 9 ctx_id: 3 IWbemServices V0

160	10.252.253.8	10.252.253.44	DCERPC Request: call_id: 10 opnum: 24 ctx_id: 3 IWbemServices V0
161	10.252.253.44	10.252.253.8	TCP 1032 > 1842 [ACK] Seq=69377 Ack=1955 Win=65071 Len=0
162	10.252.253.44	10.252.253.8	DCERPC Response: call_id: 10 ctx_id: 3 IWbemServices V0
163	10.252.253.8	10.252.253.44	TCP 1842 > 1032 [ACK] Seq=1955 Ack=69857 Win=65535 Len=0
164	10.252.253.8	10.252.253.44	DCERPC Request: call_id: 11 opnum: 3 ctx_id: 5 IWbemWCOSmartEnum V0
165	10.252.253.44	10.252.253.8	DCERPC Response: call_id: 11 ctx_id: 5 IWbemWCOSmartEnum V0
166	10.252.253.8	10.252.253.44	IRemUnknown2 RemRelease request[Malformed Packet]
167	10.252.253.44	10.252.253.8	IRemUnknown2 RemRelease response[Malformed Packet]
168	10.252.253.8	10.252.253.44	IRemUnknown2 RemRelease request[Malformed Packet]
169	10.252.253.44	10.252.253.8	IRemUnknown2 RemRelease response[Malformed Packet]
170	10.252.253.8	10.252.253.44	IRemUnknown2 RemRelease request[Malformed Packet]
171	10.252.253.44	10.252.253.8	IRemUnknown2 RemRelease response[Malformed Packet]
172	10.252.253.8	10.252.253.44	TCP 1842 > 1032 [FIN, ACK] Seq=2595 Ack=70129 Win=65263 Len=0
173	10.252.253.8	10.252.253.44	TCP 1840 > 1032 [FIN, ACK] Seq=899 Ack=497 Win=65039 Len=0
174	10.252.253.8	10.252.253.44	TCP 1841 > 1032 [FIN, ACK] Seq=507 Ack=409 Win=65127 Len=0
175	10.252.253.44	10.252.253.8	TCP 1032 > 1842 [ACK] Seq=70129 Ack=2596 Win=64431 Len=0
176	10.252.253.44	10.252.253.8	TCP 1032 > 1840 [ACK] Seq=497 Ack=900 Win=64637 Len=0
177	10.252.253.44	10.252.253.8	TCP 1032 > 1841 [ACK] Seq=409 Ack=508 Win=65029 Len=0
178	10.252.253.44	10.252.253.8	TCP 1032 > 1842 [FIN, ACK] Seq=70129 Ack=2596 Win=64431 Len=0
179	10.252.253.8	10.252.253.44	TCP 1842 > 1032 [ACK] Seq=2596 Ack=70130 Win=65263 Len=0
180	10.252.253.44	10.252.253.8	TCP 1032 > 1840 [FIN, ACK] Seq=497 Ack=900 Win=64637 Len=0
181	10.252.253.8	10.252.253.44	TCP 1840 > 1032 [ACK] Seq=900 Ack=498 Win=65039 Len=0
182	10.252.253.44	10.252.253.8	TCP 1032 > 1841 [FIN, ACK] Seq=409 Ack=508 Win=65029 Len=0
183	10.252.253.8	10.252.253.44	TCP 1841 > 1032 [ACK] Seq=508 Ack=410 Win=65127 Len=0
184	10.252.253.8	10.252.253.44	TCP 1839 > epmap [FIN, ACK] Seq=1147 Ack=1241 Win=64295 Len=0
185	10.252.253.8	10.252.253.44	TCP 1838 > epmap [RST, ACK] Seq=97 Ack=193 Win=0 Len=0
186	10.252.253.44	10.252.253.8	TCP epmap > 1839 [ACK] Seq=1241 Ack=1148 Win=64389 Len=0
187	10.252.253.44	10.252.253.8	TCP epmap > 1839 [FIN, ACK] Seq=1241 Ack=1148 Win=64389 Len=0
188	10.252.253.8	10.252.253.44	TCP 1839 > epmap [ACK] Seq=1148 Ack=1242 Win=64295 Len=0

Appendix 3: Enforcing Encrypted Access at the Server

The instructions are documented in the Microsoft article Requiring an Encrypted Connection to a Namespace (Microsoft MSDN, 2007). The article extract is provided below for your convenience. To set required encryption:

1. Create a Managed Object Format (MOF) file or modify your existing MOF file that defines the namespace. In the following example, the namespace to be modified is root\MyNamespace and the file is named MyNamespace_security.mof. RequiresEncryption has a Boolean data type so it must be set to TRUE or FALSE.

```
#pragma namespace("\\\\.\\Root\\MyNamespace")
 [RequiresEncryption(TRUE)]
 instance of __systemSecurity { };
```

2. Run mofcomp.exe to compile the MOF file.

```
c:\mofcomp MyNamespace_security.mof
```

When returning data on an asynchronous callback connection, WMI returns an access denied message to the requesting computer. WMI also makes a log entry in the NT Event Log of the computer with the encrypted namespace stating that a secure connection cannot be established to the client.

Starting with Windows Vista, the Wbemcore.log file no longer exists. You can check the NT Event Log for entries indicating rejected inbound data requests to namespaces that require encryption.

Windows Server 2003, Windows XP, Windows 2000, and Windows NT 4.0: Entries for rejected inbound data requests can be found in both the NT Event Log and Wbemcore.log.

Appendix 4: Cross Platform WMI Connection Limitations

The article Connecting Between Different Operating Systems (Microsoft MSDN Library, 2007, Win32 and COM Development) documents the limitations that are occurring in mixed Windows environments using WMI management. The article extract is provided in this appendix for your convenience:

Some connections between operating system versions are not supported:

You cannot connect to a computer that is running Windows XP Home Edition.
A computer running Windows NT cannot connect to an operating system later than Windows 2000, such as Windows XP or Windows Server 2003.
Accessing a Windows Server 2003 computer from Windows 98 or Windows 95 is not supported.

Some connections between operating system versions have special requirements:

To connect to a Windows 2000 Server SP4 and later running WMI version 1.01, you must explicitly set the impersonation level to Impersonate.
Windows 2000 computers must have Service Pack 2 installed to be able to connect to Windows XP and later operation systems.
To connect from Windows Server 2003 to computers running Windows 98, Windows Me, and Windows NT Server 4.0 SP3 and later, the credentials must be specified. If you try to use the default current user by not supplying a username and password, then the connection does not work.

Appendix 5: Setting DCOM Remote Access Security

Configuring DCOM for WMI access is described in the Microsoft article Securing a Remote WMI Connection (Microsoft MSDN Library, 2007, Win32 and COM Development). This Appendix provides the article extract for your convenience:

The following procedure describes how to grant DCOM remote launch and activation permissions for certain users and groups. If Computer A is connecting remotely to Computer B, you can set these permissions on Computer B to allow a user or group that is not part of the administrators group on Computer B to execute DCOM launch and activation calls on Computer B. To grant DCOM remote launch and activation permissions for a user or group:

1. Click Start, click Run, type DCOMCNFG, and then click OK.
2. In the Component Services dialog box, expand Component Services, expand Computers, and then right-click My Computer and click Properties.
3. In the My Computer Properties dialog box, click the COM Security tab.
4. Under Launch and Activation Permissions, click Edit Limits.
5. In the Launch Permission dialog box, follow these steps if your name or your group does not appear in the Groups or user names list:
    1. In the Launch Permission dialog box, click Add.
    2. In the Select Users, Computers, or Groups dialog box, add your name and the group in the Enter the object names to select box, and then click OK.
6. In the Launch Permission dialog box, select your user and group in the Group or user names box. In the Allow column under Permissions for User, select Remote Launch and select Remote Activation, and then click OK.

The following procedure describes how to grant DCOM remote access permissions for certain users and groups. If Computer A is connecting remotely to Computer B, you can set these permissions on Computer B to allow a user or group that is not part of the administrators group on Computer B to connect to Computer B. To grant DCOM remote access permissions:

1. Click Start, click Run, type DCOMCNFG, and then click OK.

2. In the Component Services dialog box, expand Component Services, expand Computers, and then right-click My Computer and click Properties.

3. In the My Computer Properties dialog box, click the COM Security tab.

4. Under Access Permissions, click Edit Limits.

5. In the Access Permission dialog box, select ANONYMOUS LOGON name in the Group or user names box. In the Allow column under Permissions for User, select Remote Access, and then click OK.

Appendix 6: Vista Specific Security Descriptor Access

Here is the extract from the Microsoft MSDN library article Changing Access Security on Securable Objects (Microsoft MSDN Library, 2007), detailing Windows Vista specific access to some of the security descriptors.. Please refer to the article for full detail: http://msdn2.microsoft.com/en-us/library/aa384905.aspx.

WMI Namespaces

A provider can establish security that only allows certain groups to have access to the data in a WMI namespace. Namespace security is controlled by methods on the __SystemSecurity class. Starting with Windows Vista, the GetSecurityDescriptor and SetSecurityDescriptor methods return and write __SecurityDescriptor objects. For more information, see Setting Namespace Security Descriptors.

Windows Server 2003, Windows XP, Windows 2000, Windows XP, Windows NT 4.0, and Windows Me/98/95: The GetSD and SetSD methods in the __SystemSecurity

class allow you to change the namespace security. However, these methods only use the binary byte array form of the security descriptor, which is difficult to manipulate. You can call methods in the BinarySDToWin32SD or BinarySDToSDDL methods in the Win32_SecurityDescriptorHelper class to convert the binary descriptor to an instance of Win32_SecurityDescriptor or to Security Descriptor Definition Language (SDDL).

Windows Server 2003, Windows XP, Windows 2000, Windows NT 4.0, and Windows Me/98/95: In a script or Visual Basic, you can use the procedure described in Securing WMI Namespaces to change the security of a namespace.

Registry keys

Starting with Windows Vista, you can secure registry keys so that they cannot be changed by unauthorized users. The StdRegProv class has the GetSecurityDescriptor and SetSecurityDescriptor methods. These methods return and write Win32_SecurityDescriptor objects.

Windows Server 2003, Windows XP, Windows 2000, Windows NT 4.0, and Windows Me/98/95: The GetSecurityDescriptor and SetSecurityDescriptor methods in the StdRegProv class are not available. You can call the CheckAccess method in StdRegProv to determine if a user has access to a registry key.

Printers

Starting with Windows Vista, you can secure access to instances of the Win32_Printer class using the GetSecurityDescriptor and SetSecurityDescriptor methods. These methods return and write Win32_SecurityDescriptor objects.

Windows Server 2003, Windows XP, Windows 2000, Windows NT 4.0, and Windows Me/98/95: The SetSecurityDescriptor and SetSecurityDescriptor methods in the Win32_Printer class are not available.

Services

Starting with Windows Vista, you can secure access to instances of the Win32_Service class using the GetSecurityDescriptor and SetSecurityDescriptor methods. These methods return and write Win32_SecurityDescriptor objects.

Windows Server 2003, Windows XP, Windows 2000, Windows NT 4.0, and Windows Me/98/95: The GetSecurityDescriptor and SetSecurityDescriptor methods in the Win32_Service class are not available.

DCOM applications

DCOM applications instances have several security descriptors. Starting with Windows Vista, you can use methods of the Win32_DCOMApplicationSetting class to get or change the various security descriptors. Security descriptors are returned as instances

of the Win32_SecurityDescriptor class. To get or change the configuration permissions, call the GetConfigurationSecurityDescriptor or SetConfigurationSecurityDescriptor methods. To get or change the access permissions, call the GetAccessSecurityDescriptor or SetAccessSecurityDescriptor methods. To get or change the startup and activation permissions, call the GetLaunchSecurityDescriptor or SetLaunchSecurityDescriptor methods.

Windows Server 2003, Windows XP, Windows 2000, Windows NT 4.0, and Windows Me/98/95: The Win32_DCOMApplicationSetting security descriptor methods are not available.

Files

The GetSecurityDescriptor and SetSecurityDescriptor methods are in the Win32_LogicalFileSecuritySetting class, rather than in the CIM_DataFile class.

Shares

The GetSecurityDescriptor and SetSecurityDescriptor methods are in the Win32_LogicalShareSecuritySetting class, rather than in the Win32_Share class.

Security Issues

It is recommended that changes to security descriptors be done with great caution

so that the security of the object is not compromised. Be aware that the order of ACEs in a DACL can affect access security. For more information, see Order of ACEs in a DACL.

Windows Server 2003, Windows XP, Windows 2000, and Windows NT 4.0: If the SE_DACL_PRESENT bit is set, but a DACL parameter is not set, a null DACL is written to the new security descriptor. A null DACL creates security vulnerability because it grants the Everyone account full access to the object. For more information, see Creating a DACL.

Appendix 7: Sample WMI Service Management Script

You may recall our very short sample script used for illustrations through this paper. Here is a slightly more robust service manipulation script. This is the version that was used to perform most of the testing tasks for this paper. If you would like to re-use parts of this script, please implement your own error checks. Please keep in mind this only a basic test script. All it does is accepting remote system name/IP, user name, user password, and a local file path. The script then retrieves the full service list from the remote system, to both the shell and the log file specified by the local file path. The script then prompts to select the service from the service list for manipulation, performs the service state test, and allows for the service state change:

```
'------------ Service Management Script WMI Based ------------
' Retrieve service information from remote system to file
' Display service information
' Search service list for specific service
```

```
' Manipulate service
' Test service state
' Written by Alex Timkov to facilitate SANS GSEC research paper
'------------------------------------------------------------

'-------------------------------------------------------
' Main Section
' Accept user input
' Retrieve and list service information from remote host
' Search for the service of user choice
' Manage service
' Test service state
'-------------------------------------------------------

Option Explicit

Dim strComputer, strFileName, strManage
Dim strPassword, strService, strUser
Dim intSearchResult

intSearchResult = 0

Do
  strComputer = inputbox _
    ("Please enter host name or IP address", "Input")
Loop until strComputer <> ""

Do
  strUser = inputbox("Please enter username", "Input")
Loop until strUser <> ""

Do
  strPassword = inputbox("Please enter password", "Input")
Loop until strPassword <> ""

Do
  strFileName = inputbox("Please enter file path to store WMI Data", "Input")
Loop until strPassword <> ""

WScript.Echo vbCrlf & "Getting remote service list to a file."

ServiceList strComputer, strFileName, strUser, strPassword

WScript.Echo vbCrlf & _
  "The services running on the remote system:" & vbCrlf

ServiceDisplay strFileName

Wscript.Echo vbCrlf

strService = inputbox _
  ("Please enter the name of service to manage, " _
    & "or hit Enter to exit", "Input")

  If strService <> "" Then
```

```
    ServiceSearch strFileName, strService, intSearchResult

    ' ---- Debug ----
    ' Wscript.Echo vbCrlf & "Search result = " & intSearchResult
    ' -- End Debug --

    If intSearchResult Then
      WScript.Echo strService & " service found."
      ServiceTest strComputer, strService, strUser, strPassword

      strManage = inputbox _
        ("Type Stop or Start to manage service. " _
          & "Hit Enter to exit.", "Input")

        If strManage = "Stop" Then
          ServiceStop strComputer, strService, strUser, strPassword
        End If

        If strManage = "Start" Then
          ServiceStart strComputer, strService, strUser, strPassword
        End If

      ServiceTest strComputer, strService, strUser, strPassword
    Else
      WScript.Echo vbCrlf & strService _
        & " service not installed. Nothing to do."
    End If

  End If

WScript.Echo vbCrlf & "Script execution completed. Exiting."

WScript.Quit

'---------------------------------------------------------------
' List services running on remote host and output to local file
'---------------------------------------------------------------

Sub ServiceList(strComputer, strFileName, strUser, strPassword)

  Dim colServices
  Dim objFSO, objLocator, objService, objTextFile, objWMIService

  Const constWrite = 2

  Set objFSO = CreateObject("Scripting.FileSystemObject")

  Set objTextFile = objFSO.OpenTextFile _
    (strFileName, constWrite, True)

  Set objLocator = CreateObject("WbemScripting.SWbemLocator")

  Set objWMIService = objLocator.ConnectServer _
    (strComputer, "root\CIMV2", strUser, strPassword)
```

```
  objWMIService.Security_.impersonationlevel = 3

  Set colServices = objWMIService.ExecQuery _
    ("Select * from Win32_Service")

    If Not(colServices Is Nothing) Then
      For Each objService in colServices
        If Not(objService Is Nothing) Then
          objTextFile.WriteLine(objService.DisplayName _
            & vbTab & objService.Name & vbTab & objService.State)
        End If
      Next
    End If

  objTextFile.Close
  Set objService = Nothing
  Set colServices = Nothing
  Set objWMIService = Nothing
  Set objLocator = Nothing

End Sub

'---------------------------------------------------------------------
' Read local file to display file content on screen in shell window
'---------------------------------------------------------------------

Sub ServiceDisplay(strFileName)

  Dim objFSO, objTextFile
  Dim strLine

  Const constRead = 1

  Set objFSO = CreateObject("Scripting.FileSystemObject")

  Set objTextFile = objFSO.OpenTextFile _
    (strFileName, constRead, True)

  Do Until objTextFile.AtEndOfStream
    strLine = objTextFile.ReadLine
    Wscript.Echo strLine
  Loop

  objTextFile.Close

End Sub

'-----------------------------------------------------
' Search local file for the service of user choice
'-----------------------------------------------------

Sub ServiceSearch(strFileName, strService, intSearchResult)

  Dim objFSO, objTextFile
```

```
  Dim strFile

  Const constRead = 1

  Set objFSO = CreateObject("Scripting.FileSystemObject")

  Set objTextFile = objFSO.OpenTextFile _
    (strFileName, constRead, True)

  strFile = objTextFile.ReadAll

  objTextFile.close

  If InStr(1, strFile, strService, 1) Then
    intSearchResult = 1
  Else
    intSearchResult = 0
  End If

End Sub

'-----------------------------------------------------
' Stop service of user choice running on remote host
'-----------------------------------------------------

Sub ServiceStop(strComputer, strService, strUser, strPassword)

  Dim colServices
  Dim objLocator, objService, objWMIService
  Dim intTimeout

  intTimeout = 10000

  Set objLocator = CreateObject("WbemScripting.SWbemLocator")

  Set objWMIService = objLocator.ConnectServer _
    (strComputer, "root\CIMV2", strUser, strPassword)

  objWMIService.Security_.impersonationlevel = 3

  Set colServices = objWMIService.ExecQuery _
    ("Select * from Win32_Service Where Name = '" _
      & strService & "'")

    If Not(colServices Is Nothing) Then
      For Each objService in colServices
        If Not(objService Is Nothing) Then
          If objService.State = "Running" Then
            Wscript.Echo vbCrlf & "Stopping " & strService _
              & " service..."
            objService.StopService()
            WSCript.Sleep intTimeout
          End If
        End If
      Next
```

```
    End If

  Set objService = Nothing
  Set colServices = Nothing
  Set objWMIService = Nothing
  Set objLocator = Nothing

End Sub

'-----------------------------------------------------
' Start service of user choice not running on remote host
'-----------------------------------------------------

Sub ServiceStart(strComputer, strService, strUser, strPassword)

  Dim colServices
  Dim objLocator, objService, objWMIService
  Dim intTimeout

  intTimeout = 10000

  Set objLocator = CreateObject("WbemScripting.SWbemLocator")

  Set objWMIService = objLocator.ConnectServer _
    (strComputer, "root\CIMV2", strUser, strPassword )

  objWMIService.Security_.impersonationlevel = 3

  Set colServices = objWMIService.ExecQuery _
    ("Select * from Win32_Service Where Name = '" _
      & strService & "'")

    If Not(colServices Is Nothing) Then
      For Each objService in colServices
        If Not(objService Is Nothing) Then
          If objService.State <> "Running" Then
            Wscript.Echo vbCrlf & "Starting " & strService _
              & " service..."
            objService.StartService()
            WSCript.Sleep intTimeout
          End If
        End If
      Next
    End If

  Set objService = Nothing
  Set colServices = Nothing
  Set objWMIService = Nothing
  Set objLocator = Nothing

End Sub

'--------------------------------------
' Test state of service chosen by user
'--------------------------------------
```

```
Sub ServiceTest(strComputer, strService, strUser, strPassword)

  Dim colServices
  Dim objLocator, objService, objWMIService

  Set objLocator = CreateObject("WbemScripting.SWbemLocator")

  Set objWMIService = objLocator.ConnectServer _
    (strComputer, "root\CIMV2", strUser, strPassword)

  objWMIService.Security_.impersonationlevel = 3

  Set colServices = objWMIService.ExecQuery _
    ("Select * from Win32_Service Where Name = '" _
      & strService & "'")

    If Not(colServices Is Nothing) Then
      For Each objService in colServices
        If Not(objService Is Nothing) Then
          If objService.State = "Stopped" Then
            Wscript.Echo vbCrlf & strService _
              & " service is not running."
          Else If objService.State = "Running" Then
            Wscript.Echo vbCrlf & strService _
              & " service is running."
            Else
              Wscript.Echo vbCrlf & strService _
                & " service state is unknown."
            End If
          End If
        End If
      Next
    End If

  Set objService = Nothing
  Set colServices = Nothing
  Set objWMIService = Nothing
  Set objLocator = Nothing

End Sub

'-----------------------------------------------
' End of WMI Based Service Manipulation Script
'-----------------------------------------------
```