Detecting and Investigating Windows Management Instrumentation (WMI) Based Remote Attacks in Windows Operating Systems

A dissertation submitted for the Degree of Master of Science in Information Security

K.K. Kulasekera

University of Colombo School of Computing

2017

Declaration

The thesis is my original work and has not been submitted previously for a degree at this or any other university/institute.

To the best of my knowledge it does not contain any material published or written by another person, except as acknowledge in the text.

Student Name: Kanishka Kulasekera

-------------------------------------	------------------------------
Signature	Date

This is to certify that this thesis is based on the work of Mr. Kanishka Kulasekera Under my supervision. The thesis has been prepared according to the format stipulated and is of acceptable standard.

Certified by

Supervisor Name: Dr. Kasun De Zoysa

-------------------------------------	------------------------------
Signature	Date

Abstract

The ability to watch the network traffic generated by client – server applications can greatly assist in both understanding how client – server applications works as well as identifying the issues related to the technology.

The research focuses the concept for Detecting and Investigating Windows Management Instrumentation (WMI) Based Remote Attacks in Windows Operating Environment. Windows Management Instrumentation (WMI) is the main source of management data and functionality on local and remote computers that run Microsoft Windows Operating Systems. As existing technologies for packet sniffing and intrusion detection has proven to be inadequate to detect WMI based remote attacks on Windows Operating Environments, an urge has arrived to find a mechanism to detect remote attacks which uses WMI.

In this context DCERPC (Distributed Computing Environment / Remote Procedure Calls) stub-data has used to detect the mentioned attacks. The resulting system is required to capture packets with DCERPC payloads, decode the captured payload to get DCERPC stub data and explore the hidden Microsoft COM (Component Object Model) Request.

Acknowledgment

Here and now, I would like to express my sincere thanks to all who have helped me make this thesis possible and better. Firstly, I am deeply grateful to my honorable supervisor, Dr. Kasun De Zoysa, who has checked through my thesis with patience and has given me instructive suggestions.

Then special thanks also goes to the staff University of Colombo School of Computing (UCSC), especially, Dr. Manju Wickramasinghe & Mr. Charith Madhushanka, who are very kind and generous in offering advice, as well as to the teachers and professors who have taught me over the past two years of study UCSC.

Finally, I am very grateful to my lovely wife Dilshani, My Family and friends who have offered selfless support to me.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

CLI	- Command Line Interface
IDS	- Intrusion Detection System
VM	- Virtual Machine
WMI	- Windows Management Instrumentation
WQL	- WMI Query Language
PSEXEC	- Sysinternals' PS Tools suite
DCOM	- Distributed Component Object Model
WBEM	- Web-based Enterprise Management
DMTF	- Desktop Management Task Force
SMS	- System Management Server
COM	- Component Object Model
WINRM	- Windows Remote Management

Chapter 1 - Introduction

Over the past three decades the computer has become the most significant innovation of man due to rapid growth of Information and Communication Technology. As the world has entered the era of information, computer has become a need for day to day life of people. Information and Data has become much more valuable that misuse of it may cause an instant world war or loss of billions of US Dollars. So, protecting computerized systems from breaching has become a commodity for any Governmental or Private institutes.

During past 3 decades Microsoft has become a leading giant in the computer industry by producing Windows, the most favorable operating system environment for computerized systems. Since the first ever Windows Operating System in 1985 Microsoft has introduced many more features to Windows and improved the usability and user-experience of computing environment satisfying the users from any level of computing knowledge to accomplish their day to day tasks using computers. As, over 86% of computer users prefer Windows, it has become the most selective Operating System Environment for computer systems and hence it has become the most attacked operating system environment in world. [1]

While knowing the growing favor of using Windows based operating environment around the world, the research aims on developing a tool to detect and investigate one of the most significant attacks on Windows Operating Environments. The attack is based on Windows Management Instrumentation (WMI) and will be moving forward to investigate the significance of WMI, the motivation to study on this issue and exploring the scope and limitations of the research.

1.1 Windows Management Instrumentation (WMI)

Windows Management Instrumentation or WMI is a middle Layer Technology that enables standardize management of Windows based computer systems. It collects computer management data from wide variety of sources and makes it accessible by using standard interfaces. [2]

WMI is the Microsoft implementation of the Web-based Enterprise Management (WBEM) standard, which was developed by the Desktop Management Task Force (DMTF). WBEM build on series of industry-wide initiatives to standardize computer management. Those initiatives include Simple Network Management Protocol (SNMP) and Desktop Management Interface Standards. Standardizing Computer Management allows to use multiple tools to address various computer management needs without requiring a separate infrastructure for each tool. For example, System Management Server (SMS) and Microsoft Operations Manager (MOM) both use WMI. SMS use it for centralized inventory collection of all computers whereas MOM to alert critical changes in servers, but the server does not require two different types of client agents to collect configuration information. WMI will write scripts to change computer configurations and then distribute those scripts to the SMS client.

WMI is comprised with powerful set of tools used to manage Windows systems both locally and remotely. But it does not consolidate the management data in a central location - that is one function of Systems Management Server (SMS). WMI can be used to set configuration details of a computer and to detect and respond to changes in the computer configurations using WMI events.

Since WMI user's providers (relatively simple component that the developer of the system makes available) to work with various hardware types, operating system components and subsystems and application systems, it can collect and set configuration details for every hardware variance, operating system component or application system. Any system which has a WMI provider can be managed with WMI and can therefore be managed by any computer management application (such as SMS or MOM) which uses WMI. Some well know WMI providers are Active Directory Provider, Biz Talk Provider etc.

WMI is available for all Microsoft Windows 95 or later operating systems. It is installed with Microsoft Windows Millennium Edition, Windows 2000, Windows XP, Microsoft Windows Server 2003 family and available for download for the other supported versions of Windows.

1.2 Motivation

The power that WMI possesses has made it a tool that attackers use in many phases in Attack-Lifestyle. There is a wealth of WMI objects, methods and events that can be extremely powerful for performing anything from reconnaissance, AV/VM detection, code execution, lateral movement, covert data storage, to persistence. It is even possible to build a pure WMI back-door that does not introduce a single file on disk.

This kind of malicious use of WMI can cause disastrous results as lot of organizations around the world uses Windows based computer environments and most of the Windows distributions has WMI by default.

Currently there are no monitoring tools for WMI and it is a well-known threat that attackers could use WMI to perform serious attacks on Windows based computing environments. The research is aimed on providing a tool to detect and investigate remote attacks on WMI. [3]

1.3 Thesis Contribution

1.3.1 Goals and Objectives

The goal of this project is to develop a detecting and investigating tool for WMI based remote attacks. This tool will allow users to detect and investigate remote attacks based on WMI in Windows computing environments.

1.3.2 Research Question

The research considers answering two question which arises with the above-mentioned goal.

1. How to accurately detect WMI packet across the network?
2. How to define parameters to detect such malicious activities?

1.3.3 Scope and Limitations

The research will only consider about the LAN and it will detect the attacks from Internet to the local intranet. The research will develop a tool to analyze and detect attacks on windows environment real time.

Chapter 2 - Literature Review

This section will cover the related literature for the research.

2.1 Component Object Model (COM)

Microsoft COM (Component Object Model) technology in the Microsoft Windows-family of Operating Systems enables software components to communicate. COM is used by developers to create re-usable software components, link components together to build applications, and take advantage of Windows services. COM objects can be created with a variety of programming languages. Object-oriented languages, such as C++, provide programming mechanisms that simplify the implementation of COM objects. The family of COM technologies includes COM+, Distributed COM (DCOM) and ActiveX® Controls. Microsoft provides COM interfaces for many Windows application programming interfaces such as Direct Show, Media Foundation, Packaging API, Windows Animation Manager, Windows Portable Devices, and Microsoft Active Directory (AD). COM is used in applications such as the Microsoft Office Family of products. For example, COM OLE technology allows Word documents to dynamically link to data in Excel spreadsheets and COM Automation allows users to build scripts in their applications to perform repetitive tasks or control one application from another. [4]

2.2 Distributed Computing Environment Remote Procedure Calls (DCE/RPC)

DCE/RPC, short for "Distributed Computing Environment / Remote Procedure Calls", is the remote procedure call system developed for the Distributed Computing Environment (DCE). This system allows programmers to write distributed software as if it were all working on the same computer, without having to worry about the underlying network code. DCE/RPC is an implementation of the Remote Procedure Call technology developed by the Open Group as part of the Distributed Computing Environment. DCE/RPC is most commonly used to interact with Windows network services.

2.3 TCP

TCP (Transmission Control Protocol) is a standard that defines how to establish and maintain a network conversation via which application programs can exchange data. TCP works with the Internet Protocol (IP), which defines how computers send packets of data to each other. Together, TCP and IP are the basic rules defining the Internet. TCP is defined by the Internet Engineering Task Force (IETF) in the Request for Comment (RFC) standards document number 793. TCP is a connection-oriented protocol, which means a connection is established and maintained until the application programs at each end have finished exchanging messages. It determines how to break application data into packets that networks can deliver, sends packets to and accepts packets from the network layer, manages flow control, and because it is meant to provide error-free data transmission handles retransmission of dropped or garbled packets as well as acknowledgement of all packets that arrive. In the Open Systems Interconnection (OSI) communication model, TCP covers parts of Layer 4, the Transport Layer, and parts of Layer 5, the Session Layer. [5]

2.4 UDP

UDP (User Datagram Protocol) is an alternative communications protocol to Transmission Control Protocol (TCP) used primarily for establishing low-latency and loss tolerating connections between applications on the Internet. Both UDP and TCP run on top of the Internet Protocol (IP) and are sometimes referred to as UDP/IP or TCP/IP. Both protocols send short packets of data, called datagrams. UDP provides two services not provided by the IP layer. It provides port numbers to help distinguish different user requests and, optionally, a checksum capability to verify that the data arrived intact. TCP has emerged as the dominant protocol used for the bulk of Internet connectivity owing to services for breaking large data sets into individual packets, checking for and resending lost packets and reassembling packets into the correct sequence. But these additional services come at a cost in terms of additional data overhead, and delays called latency. In contrast, UDP just sends the packets, which means that it has much lower bandwidth overhead and latency. But packets can be lost or received out of order as a result, owing to the different paths individual packets traverse between sender and receiver.

2.5 Marshalling / Unmarshalling

Marshalling is the process of gathering data and transforming it into a standard format before it is transmitted over a network so that the data can transcend network boundaries. In order for an object to be moved around a network, it must be converted into a data stream that corresponds with the packet structure of the network transfer protocol. This conversion is known as data marshalling. Data pieces are collected in a message buffer before they are marshaled. When the data is transmitted, the receiving computer converts the marshaled data back into an object. Data marshalling is required when passing the output parameters of a program written in one language as input to a program written in another language. [6]

Unmarshalling in the other hand is the opposite process of Marshalling. In Unmarshalling the data stream sent through network is reassemble to the original object.

2.6 Interacting with WMI

There is a whole heap of tools which have been provided by Microsoft and other 3rd party software developers that enables the capabilities for you to work with WMI. The following are some of the utilities that can interact with WMI.

2.6.1 Powershell

PowerShell is a very powerful scripting language that has a lot of functionality for working with WMI. With PowerShell version 3, the following commands are available which can be used with WMI. (These will be another detailed chapter on powershell and its usefulness's discussed later in this thesis)

```
Get-WmiObject
Get-CimAssociatedInstance
Get-CimClass
Get-CimInstance Get-CimSession Set-WmiInstance Set-CimInstance Invoke-WmiMethod
Invoke-CimMethod New-CimInstance New-CimSession
New-CimSessionOption Register-CimIndicationEvent Register-WmiEvent
Remove-CimInstance Remove-WmiObject Remove-CimSession
```

The WMI and CIM commands offer almost equivalent functionality, however the CIM commands were inbuilt to PowerShell version 3 and offer some additional capabilities over the WMI commands. The biggest advantage of using the CIM commands is that they work over both WinRM and DCOM protocols. The WMI commands only work over DCOM. Not all systems will have PowerShell v3+ installed, even though PowerShell v2 is installed by default on Windows 7 so it is targeted as the least common version that is looked at by attackers.

From an attacker's angle, commands required to the creation, modification, and deletion of WMI/CIM classes are surprisingly absent. Considering there are no majorly likely reason to have such commands however, it is understood that they do not exist. Nevertheless, WMI classes still can be easily created using WMI.

The of examples in this thesis will be using PowerShell due to its capabilities efficiencies and increasingly used by attackers.

2.6.2 Wmic.exe

wmic.exe is a very useful command line utility for working with WMI. It has a very high amount of useful default usefulness for WMI objects but you can also perform more advanced queries. wmic.exe can also run WMI methods and is utilized commonly by attackers to perform by calling the Win32_Process Create method. One of the drawbacks of wmic.exe is that you cannot call methods that accept embedded WMI objects. If PowerShell is not percent though, it is good enough for performing reconnaissance and basic method invocation. wmic.exe is still used frequently by pentesters and attackers. [7]

2.6.3 Winrm.exe

winrm.exe can be utilized to enumerate WMI object instances, enable methods, and create and remove object instances on local and remote devices running the WinRM service. winrm.exe can also be used to perform configuration of WinRM settings. The commonly used method of interacting with WMI over WinRM is PowerShell using the CIM command but this is an alternative method for doing so that defenders should be aware of.

Depicted below are some example uses of winrm.exe:

```
winrm invoke Create wmicimv2/Win32_Process
@{CommandLine="notepad.exe";CurrentDirectory="C:\"}
winrm enumerate
http://schemas.microsoft.com/wbem/wsman/l/wmi/root/cimv2/Win32_Process
winrm get
http://schemas.microsoft.com/wbem/wsman/l/wmi/root/cimv2/Win32_OperatingSystem
```

2.6.4 Remote WMI

While one can work with WMI locally, the strength of WMI is confirmed when it is used over the network. There are two protocols used for everything from querying objects, registering events, and executing WMI class methods. DCOM and WinRM. [8], [9]

Both these protocols may be viewed as beneficial to an attacker since most companies and security practitioners generally does not check the content of this traffic for patterns of malicious activity. All an attacker needs to do is remote WMI are valid, privileged user credentials. In the case of the Linux wmis-pth utility, all that is needed is the hash of the victim user.

2.7 Remote WMI protocols – DCOM

2.7.1 Distributed Component Object Model (DCOM)

DCOM is the default protocol that is being used by WMI since its inception. DCOM establishes an initial connection over TCP port 135. Eventually data is then exchanged over a randomly selected TCP port. The port range can be configured either via dcomcnfg.exe which finally modifies the following registry key. [10]

```
HKEY_LOCAL_MACHINE\Software\Microsoft\Rpc\Internet – Ports (REG_MULTI_SZ)
```

All the built-in WMI cmdlets in PowerShell communicate using DCOM.

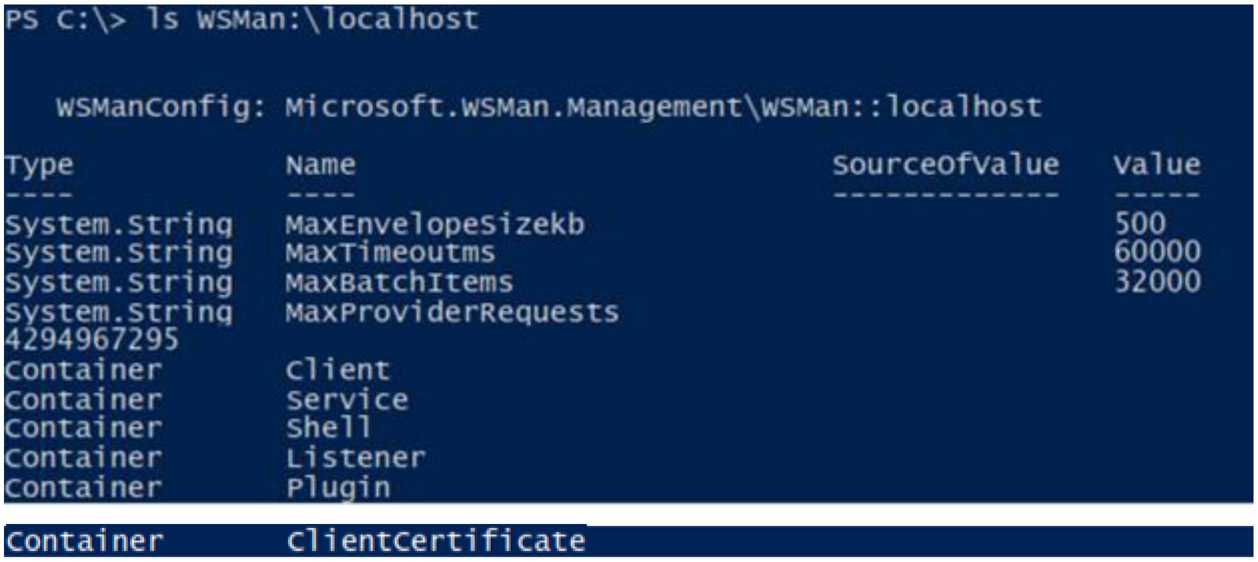
```
PS C:\> Get-WmiObject -Class Win32_Process -ComputerName
192.168.72.134 -Credential 'WIN-B85AAA7ST4U\Administrator'
```

2.7.2 Windows Remote Management (WinRM)

In the recent past WinRM has superseded DCOM as the recommended remote management protocol for Windows. WinRM is developed upon the Web Services-Management (WSMan) specification – a SOAP-based device management protocol. Additionally, PowerShell Remoting is built upon the WinRM specification and enables for extremely powerful remote management of a Windows enterprise for using PowerShell. WinRM was also built to support WMI or more generically, CIM operations over the network.

By default, the WinRM service operates on TCP port 5985 (HTTP) and is encrypted by default. Certificates may also be configured enabling HTTPS support over TCP port 5986.

WinRM settings are easily configurable using GPO winrm.exe and the PowerShell WSMan "drive".



```
PS C:\> ls WSMan:\localhost


   WSManConfig: Microsoft.WSMan.Management\WSMan::localhost

Type            Name                           SourceOfValue   Value
----            ----                           -------------   -----
System.String   MaxEnvelopeSizekb                              500
System.String   MaxTimeoutms                                   60000
System.String   MaxBatchItems                                  32000
System.String   MaxProviderRequests
4294967295
Container       Client
Container       Service
Container       Shell
Container       Listener
Container       Plugin

Container       ClientCertificate
```

PowerShell enables a convenient command for verifying the WinRM service is listening – Test-WSMan. If Test-WSMan returns an output, it indicates that the WinRM service is listening on that system. This command does not need authentication. It is simply a command wrapper for the WSMan Identify command and Microsoft developed the recommendation of the WSMan specification to not require authentication for this action.

For connecting with WMI on systems running the WinRM service, the only built-in tools that supports remote WMI operations is winrm.exe and the PowerShell CIM commands.

2.8 WMI Attacks

WMI is a highly powerful tool for attackers across various phases of the attack lifecycle. There is a whole heap of WMI objects, methods, and events that can be highly powerful for performing anything from reconnaissance, AV/VM detection, code execution, lateral movement, covert data storage, to persistence. It is even capable enough to build a pure WMI backdoor that doesn't introduce a single file to disk.

There are many advantages of using WMI to an attacker:

- It is installed and running by default on all Windows operating systems going back to Windows 98.
- For code execution, it offers a stealthier alternative to running psexec.
- Permanent WMI event subscriptions run as SYSTEM.
- Defenders are generally unaware of WMI as a multi-purpose attack vector.
- Nearly every operating system action can trigger a WMI event.
- Other than storage in the WMI repository, no payloads touch disk.

2.9 Reconnaissance

One of the initial steps taken by most malware and penetration testers will be reconnaissance. WMI has many classes that can assist an attacker to get an experience for the environment they're targeting. These are few of the commonly used reconnaissance tasks carried out by attackers and the respective WMI objects that can be queried:

- Host/OS information: Win32_OperatingSystem, Win32_ComputerSystem
- File/directory listing: CIM_DataFile
- Disk volume listing: Win32_Volume
- Registry operations: StdRegProv
- Running processes: Win32_Process
- Service listing: Win32_Service
- Event log: Win32_NtLogEvent
- Logged on accounts: Win32_LoggedOnUser
- Mounted shares: Win32_Share
- Installed patches: Win32_QuickFixEngineering

2.10 Anti-Virus Detection

Installed Anti-Virus products will naturally register themselves in WMI via the AntiVirusProduct class included within either the root\SecurityCenter or root\SecurityCenter2 namespaces depending upon the OS version.

Example of Antivirus Product Class:

```
PS C:\> Get-WmiObject -Namespace root\SecurityCenter2 -Class AntivirusProduct
GENUS                   : 2
CLASS                   : AntivirusProduct
SUPERCLASS              :
DYNASTY                 : AntivirusProduct
RELPATH                 : AntivirusProduct.instanceGuid="{B7ECF8CD-0188-
6703-DBA4-AA65C6ACFB0A}"
PROPERTY_COUNT          : 5
DERIvATION              : {}
SERvER                  : WIN-B85AAA7ST4U
NAMESPACE               : ROOT\SecurityCenter2
PATH                                                                    \WIN-
B85AAA7ST4U\ROOT\SecurityCenter2:AntivirusProduct.instanceGuid="{B7ECF8CD-
0188-6703-DB
A4-AA65C6ACFB0A}"
displayName             : Microsoft Security Essentials
instanceGuid                        :{B7ECF8CD-0188-6703-DBA4-AA65C6ACFB0A}
pathToSignedProductExe  : C:\Program Files\Microsoft Security Client\msseces.exe
pathToSignedReportingExe : C:\Program Files\Microsoft Security Client\MsMpEng.exe
productState            : 397328
PSComputerName          : WIN-B85AAA7ST4U
PS C:\> Get-WmiObject -Namespace root\SecurityCenter2 -Class AntivirusProduct
GENUS                   : 2
CLASS                   : AntivirusProduct
SUPERCLASS              :
DYNASTY                 : AntivirusProduct
RELPATH                 : AntivirusProduct.instanceGuid="{B7ECF8CD-0188-
6703-DBA4-AA65C6ACFB0A}"
```

Example of Antivirus Product Class Cont:

```
PROPERTY_COUNT       : 5
DERIvATION           : {}
SERVER               : WIN-B85AAA7ST4U
NAMESPACE            : ROOT\SecurityCenter2
PATH                                                                        :\\WIN-
B85AAA7ST4U\ROOT\SecurityCenter2:AntivirusProduct.instanceGuid="{B7ECF8CD-
0188-6703-DBA4-AA65C6ACFB0A}"
displayName          : Microsoft Security Essentials
instanceGuid                   :        {B7ECF8CD-0188-6703-DBA4-AA65C6ACFB0A}
pathToSignedProductExe  : C:\Program Files\Microsoft Security Client\msseces.exe
pathToSignedReportingExe : C:\Program Files\Microsoft Security
Client\MsMpEng.exe
productState         : 397328
PSComputerName       : WIN-B85AAA7ST4U
```

Sample WQL Query:

```
SELECT * FROM AntiVirusProduct
```

2.11 VM Detection

Generic detection of VM and sandboxing environments can be performed. For example, if physical memory is less than 2GB or if there is only a single processor core it is likely you're running in a VM.

Sample WQL Queries:

```
SELECT * FROM Win32_ComputerSystem WHERE TotalPhysicalMemory < 2147483648
SELECT * FROM Win32_ComputerSystem WHERE NumberOfLogicalProcessors < 2
```

Example:

```
$VMDetected = $False
$Arguments = @{
class = 'win32_computerSystem'
Filter = 'NumberofLogicalprocessors < 2 AND TotalphysicalMemory < 2147483648'
}
if (Get-wmiobject @Arguments) { $VMDetected = $True }
```

2.12 VMware Detection

Sample WQL Queries:

```
SELECT    *              FROM    Win32_NetworkAdapter   WHERE   Manufacturer   LIKE
"%VMware%"
SELECT  *          FROM   Win32_BIOS WHERE SerialNumber LIKE "%VMware%"
SELECT  *          FROM   Win32_Process WHERE Name="vmtoolsd.exe"
SELECT  *          FROM   Win32_NetworkAdapter WHERE Name LIKE "%VMware%"
```

Example:

```
$VMwareDetected = $False
$VMAdapter   =   Get-wmiobject   win32_NetworkAdapter   -Filter   'Manufacturer   LIKE
n%VMware%n oR Name LIKE n%VMware%n'
$VMBios = Get-wmiobject win32_BIoS -Filter 'SerialNumber LIKE n%VMware%n'
$VMToolsRunning = Get-wmiobject win32_process -Filter 'Name=nvmtoolsd.exen'
if ($VMAdapter -or $VMBios -or $VMToolsRunning) { $VMwareDetected = $True }
```

2.13 Code Execution and Lateral Movement

There are two methods of achieving remote arbitrary code execution in WMI: Win32_Process Create Method. The Win32_Process class contains a static method - Create that can spawn a process locally or remotely. This is effectively the WMI equivalent of running psexec (Sysinternals' PsTools suite). The following example demonstrates executing a process on a remote machine. Attackers will most probably choose to run a malicious, encoded PowerShell command with the Win32_Process Create method.

2.14 Event Consumers

Another method of gaining arbitrary remote code execution is by creating a permanent WMI event subscription. Normally, a permanent WMI event subscription is developed to persist and respond to certain events. If an attacker requires to execute a single payload however, the respective event consumer would just need to delete its corresponding event filter, consumer, and filter to consumer binding. The advantage of this technique is that the payload runs as SYSTEM, and it avoids having a payload be displayed in plaintext in the presence of command line auditing. For example, if a VBScript ActiveScriptEventConsumer payload was selected, the only process created would WMI script host process:

```
%SystemRoot%\system32\wbem\scrcons.exe -Embedding
```

As an attacker, the challenge for pursuing this class of attack vector would be selecting an intelligent event filter. If they just wanted to trigger the payload after a few seconds, an IntervalTimerInstruction class could be used. An attacker might choose to execute the payload upon a user locking their screen.

In that case, an extrinsic Win32_ProcessStartTrace event could be used to trigger upon the LogonUI.exe process being created. An attacker can get creative in their choice of an appropriate event filter.

Chapter 3 - Analysis

3.1 WMI for Detection and Response

As technology is introduced and subsequently deprecated over time in the Windows operating system, one powerful technology that has remained consistent since Windows NT 4.01 and Windows 95 is Windows Management Instrumentation (WMI). Present on all Windows operating systems, WMI is comprised of a powerful set of tools used to manage Windows systems both locally and remotely.

While it has been well known and utilized heavily by system administrators since its inception, WMI was likely introduced to the mainstream security community when it was discovered that it was used maliciously as one component in the suite of exploits and implants used by Stuxnet3. Since then, WMI has been gaining popularity amongst attackers for its ability to perform system reconnaissance, AV and VM detection, code execution, lateral movement, persistence, and data theft.

As attackers increasingly utilize WMI, it is important for defenders, incident responders, and forensic analysts to have knowledge of WMI and to know how they can wield it to their advantage. This whitepaper will introduce the reader to WMI, actual and proof-of-concept attacks using WMI, how WMI can be used as a rudimentary intrusion detection system (IDS), and how to perform forensics on the WMI repository file format. [11]

3.2 High-Level WMI Architecture

WMI represents most data which is related to operating system information and functions in the form of objects. An object is a member of a class, a class is a member of a namespace, and all namespaces derive from the “Root” namespace. This thesis will later show and explain examples of how to list, find, and use namespaces, classes, and objects various multiple tools and methods such as PowerShell, WQL (WMI Query Language), WMI Code Creator and others.

WMI classes can be found on the Microsoft MSDN site

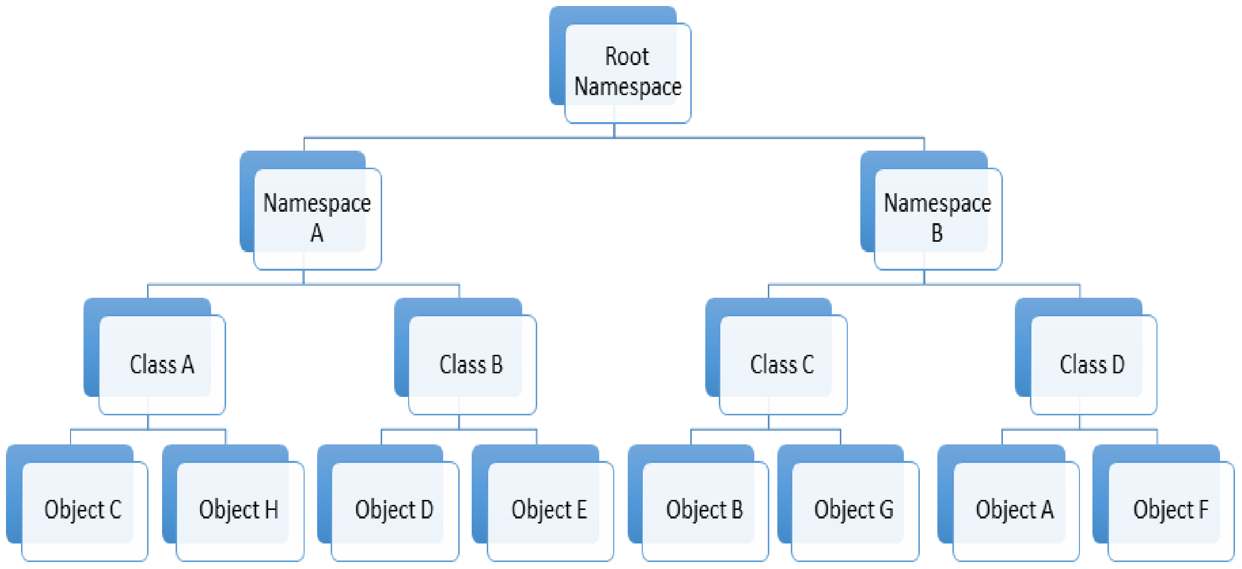


Figure 3. 1 : WMI Architecture Diagram.

3.3 WMI Architecture

WMI is the Microsoft implementation of the Web-Based Enterprise Management (WBEM)4 and Common Information Model (CIM)5 standards published by the Distributed Management Task Force (DMTF)6. These two standards aim to provide an industry-agnostic means of collecting and transmitting information related to any managed component in an enterprise. For an example of a managed component in WMI would be a running process, registry key, installed service, file information, etc. These standards communicate the means by which implementers should query, populate, structure, transmit, perform actions on, and consume data.

Following is a high level, Microsoft's implementation of these standards which can be summarized as follows:

3.3.1 Managed Components

Managed components are represented as WMI objects – class instances representing highly structured operating system data. Microsoft provides a wealth of WMI objects that communicate information related to the operating system. E.g. Win32_Process, Win32_Service, AntiVirusProduct, Win32_StartupCommand, etc.

3.3.2 Consuming Data

Microsoft provides several means for consuming WMI data and executing WMI methods. For example, PowerShell provides a very simple means for interacting with PowerShell.

3.3.3 Querying Data

All WMI objects are queried using a SQL like language called WMI Query Language (WQL). WQL enables fine grained control over which WMI objects are returned to a user.

3.3.4 Populating Data

When a user requests specific WMI objects, the WMI service (Winmgmt) needs to know the means by which to populate the requested WMI objects. This is accomplished with WMI providers. A WMI provider is a COM-based DLL that contains an associated GUID that is registered in the registry. WMI providers do the heavy lifting in populating data – e.g. querying all running processes, enumerating registry keys, etc. When the WMI service populates WMI objects, there are two types of class instances: dynamic and persistent objects. Dynamic objects are generated on the fly when a specific query is performed. For example, Win32_Process objects are generated on the fly. Persistent objects are stored in the CIM repository located by default in %SystemRoot%\System32\wbem\Repository\OBJECTS.DATA.

3.3.5 Structuring Data

The structure/schema of the vast majority of WMI object is described in Managed Object Format (MOF) files. MOF files use a C++ like syntax and provide the schema for a WMI object. So while WMI providers generate raw data, MOF files provide the schema in which the generated data is formatted. From a defender's perspective, it is worth noting that WMI object definitions can be created without a MOF file. Rather, they can be inserted directly into the CIM repository using some basic .NET code.

3.3.6 Transmitting Data

Microsoft provides two protocols for transmitting WMI data remotely: DCOM and Windows Remote Management (WinRM).

3.4 Performing Actions

Some WMI objects include methods that can be executed. For example, a common method executed by attackers for performing lateral movement is the static Create method in the

Win32_Process class. WMI also provides an eventing system whereby users can register event handlers upon the creation, modification, or deletion of any WMI object instance.

Figure 3.2 provides a high-level overview of the Microsoft implementation of WMI and the relationship between its implemented components and the standards they implement.

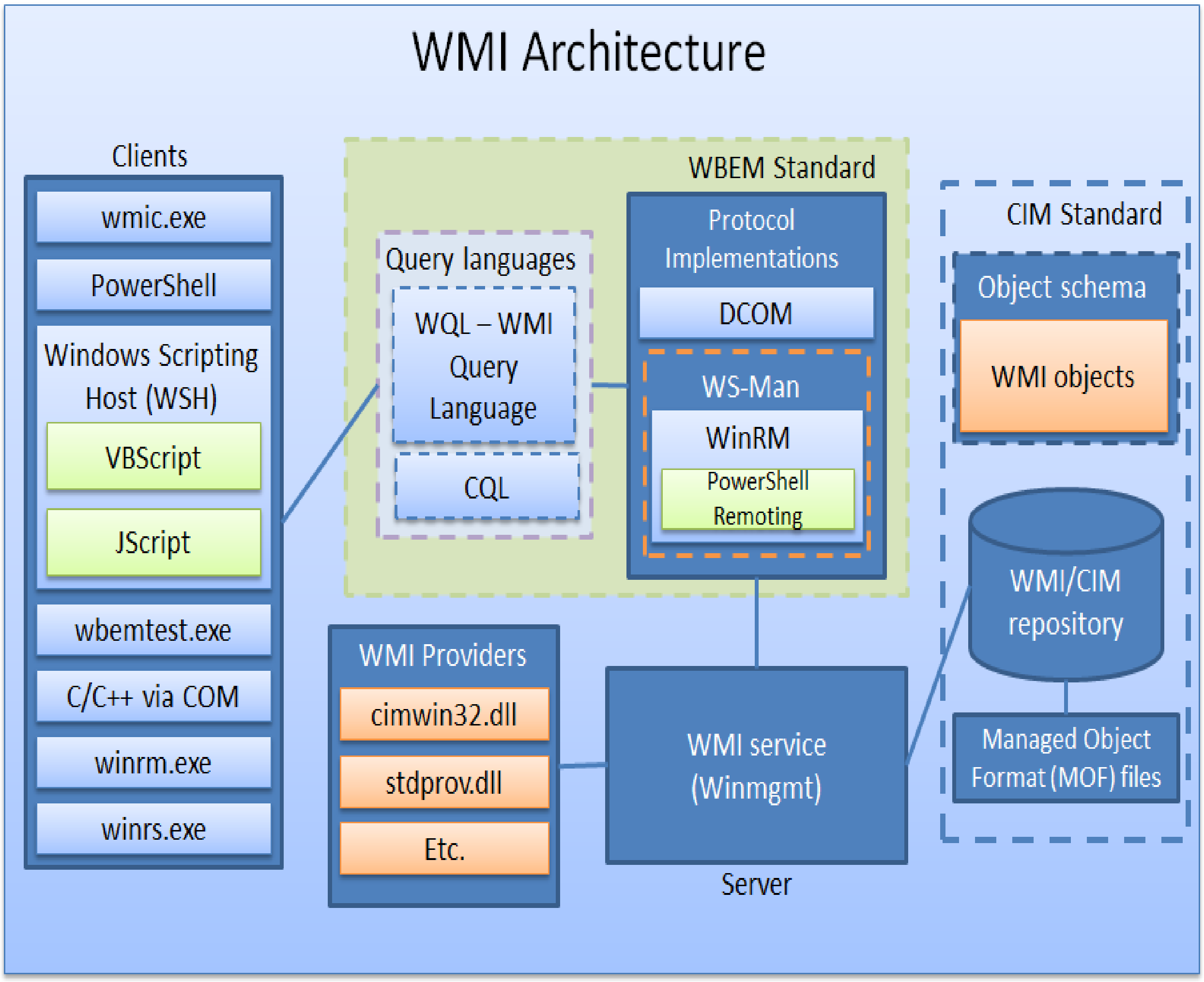


Figure 3. 2 : A high-level overview of the WMI architecture [11]

Remote WMI

The real threat and power of WMI is realized when used over the network. There are two protocols which enables remote object queries, event registration, WMI class method execution, and class creation:

• DCOM TCP Port 135

• WinRM TCP Ports 5985 (HTTP) and 5986 (HTTPS).

These protocols are viewed as useful to an attacker because most companies and vendors generally does not check the content of this traffic for signs of malicious activity. All that an attacker requires to leverage remote WMI is valid, privileged user's credentials. In the case of the Linux wmis-pth utility, all that is needed is the hash of the victim user.

Chapter 4 - Design

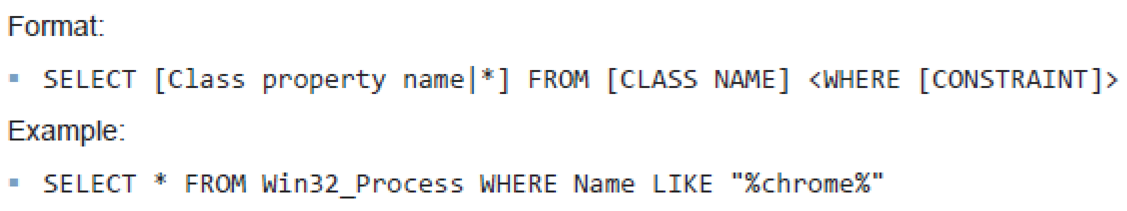
4.1 Querying WMI

WMI provides a straightforward syntax for querying WMI object instances, classes, and namespaces Windows Query Language (WQL). From a defensive perspective, it is very important to understand and can adequately use queries. Queries are used often for malicious purposes and should be used for defensive purposes.

These queries can do everything from attacker reconnaissance to intrusion detection.
The three categories of WQL queries are as follows:

4.1.1 Instance queries

a) These queries are used to query WMI class instances.
b) Primarily are used by attackers for conducting reconnaissance and gathering information about a targeted system.



Format:

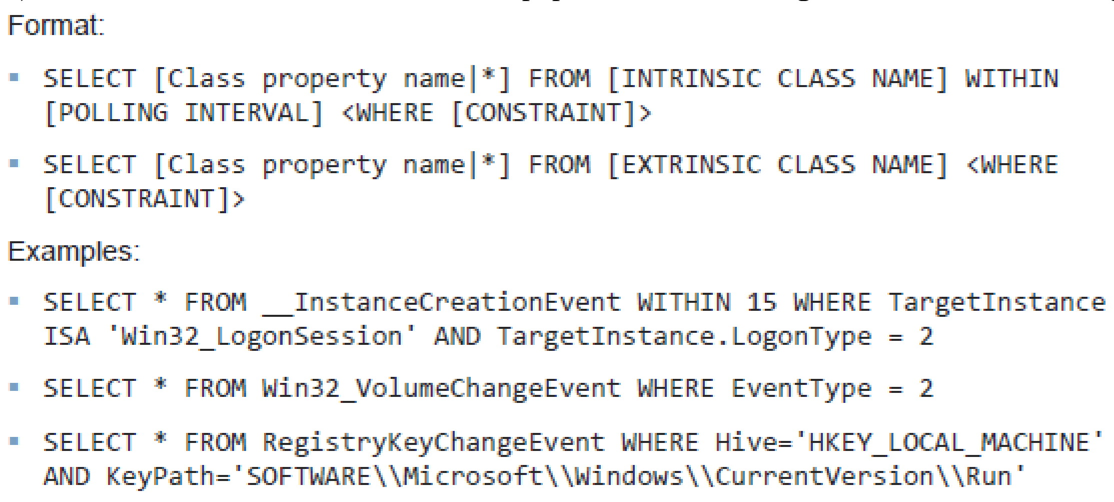
- `SELECT [Class property name|*] FROM [CLASS NAME] <WHERE [CONSTRAINT]>`

Example:

- `SELECT * FROM Win32_Process WHERE Name LIKE "%chrome%"`

4.1.2 Event queries

a). Are used as a WMI event registration mechanism, e.g., WMI object creation, deletion, or modification.
b). Will be one area of focus later in the paper when discussing WMI defense and mitigation.



Format:

- `SELECT [Class property name|*] FROM [INTRINSIC CLASS NAME] WITHIN [POLLING INTERVAL] <WHERE [CONSTRAINT]>`
- `SELECT [Class property name|*] FROM [EXTRINSIC CLASS NAME] <WHERE [CONSTRAINT]>`

Examples:

- `SELECT * FROM __InstanceCreationEvent WITHIN 15 WHERE TargetInstance ISA 'Win32_LogonSession' AND TargetInstance.LogonType = 2`
- `SELECT * FROM Win32_VolumeChangeEvent WHERE EventType = 2`
- `SELECT * FROM RegistryKeyChangeEvent WHERE Hive='HKEY_LOCAL_MACHINE' AND KeyPath='SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run'`

4.1.3 Meta queries

a). Are used to query WMI class schemas.

Format:

```
SELECT [Class property name|*] FROM [Meta_Class|SYSTEM CLASS NAME]
<WHERE [CONSTRAINT]>
```

The following query lists all WMI classes that start with the string “Win32.”
Example:

```
SELECT * FROM Meta_Class WHERE __Class LIKE “Win32%”
```

NOTE: When performing any WMI query, the default namespace ROOT\CIMV2 is implied unless explicitly provided.

4.2 Attacker WMI Detection

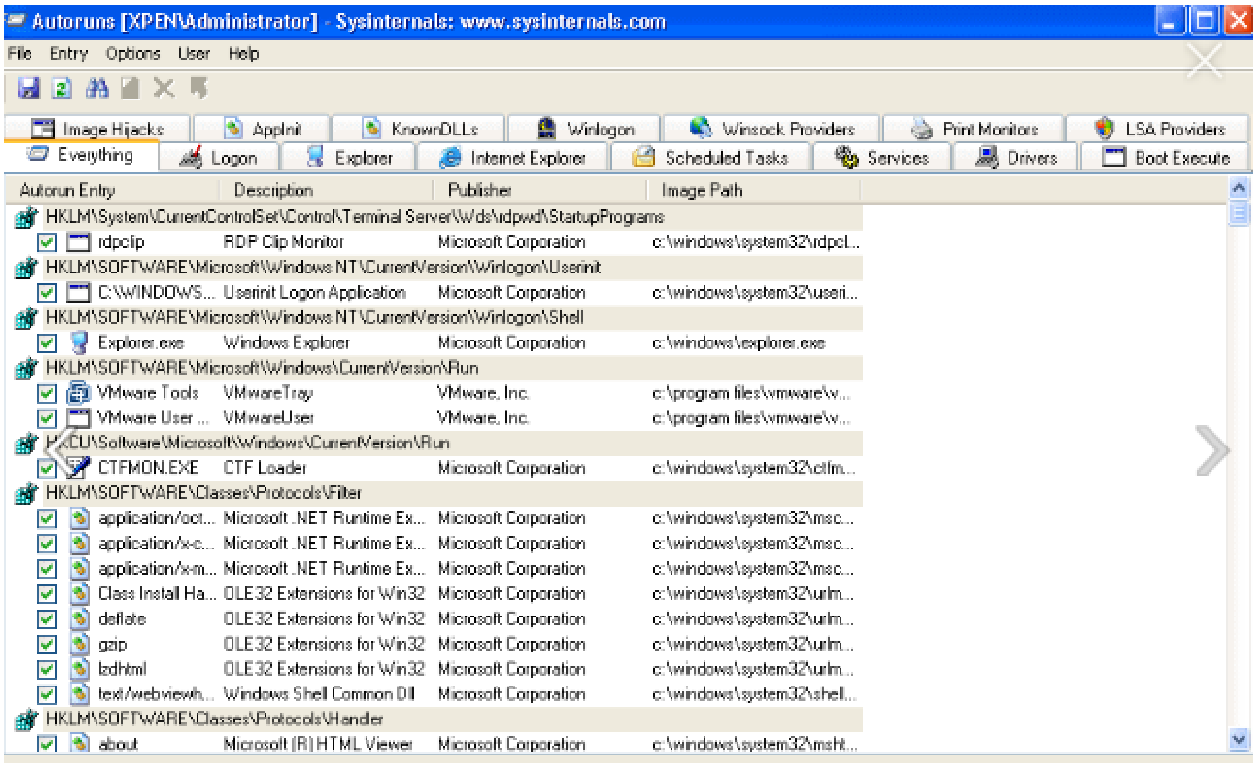
4.2.1 Existing Detection Utilities

In addition to using WMI events to alert users to possible attacks, detection utilities are also available.

4.2.2 Sysinternals Autoruns

Autoruns is a free utility that unveils every startup item on a Windows-based PC. All images are stored in the startup folders, the Registry, and other areas. Autoruns shows the name and location of each image. For files, it displays the directory path; for Registry entries, it provides the exact key. Autoruns also supplies the name of the publisher and a brief description based on the item’s version data. Double-clicking on an entry guides the user to its directory or Registry key; right clicking Opens a popup menu with more options, including a Properties command that displays the Standard File Properties window with full version information.

Users has the ability to check on the digital signature of an entry through the Verify command, which queries web sites with certificate revocation lists (CRLs) to determine if an image is digitally signed and whether the signature is valid. Another option to “Hide Signed Microsoft Entries” excludes entries already signed by Microsoft, allowing the user to focus on third-party images.



Autorun Entry	Description	Publisher	Image Path
HKLM\System\CurrentControlSet\Control\Terminal Server\Wds\rdpwd\StartupPrograms			
✓ rdpclip	RDP Clip Monitor	Microsoft Corporation	c:\windows\system32\rdpcl...
HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Userinit			
✓ C:\WINDOWS...	Userinit Logon Application	Microsoft Corporation	c:\windows\system32\useri...
HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Shell			
✓ Explorer.exe	Windows Explorer	Microsoft Corporation	c:\windows\explorer.exe
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run			
✓ VMware Tools	VMwareTray	VMware, Inc.	c:\program files\vmware\v...
✓ VMware User ...	VMwareUser	VMware, Inc.	c:\program files\vmware\v...
HKCU\Software\Microsoft\Windows\CurrentVersion\Run			
✓ CTFMON.EXE	CTF Loader	Microsoft Corporation	c:\windows\system32\ctfm...
HKLM\SOFTWARE\Classes\Protocols\Filter			
✓ application/oct...	Microsoft .NET Runtime Ex...	Microsoft Corporation	c:\windows\system32\msc...
✓ application/x-c...	Microsoft .NET Runtime Ex...	Microsoft Corporation	c:\windows\system32\msc...
✓ application/x-m...	Microsoft .NET Runtime Ex...	Microsoft Corporation	c:\windows\system32\msc...
✓ Class Install Ha...	OLE32 Extensions for Win32	Microsoft Corporation	c:\windows\system32\urlm...
✓ deflate	OLE32 Extensions for Win32	Microsoft Corporation	c:\windows\system32\urlm...
✓ gzip	OLE32 Extensions for Win32	Microsoft Corporation	c:\windows\system32\urlm...
✓ lzdhtml	OLE32 Extensions for Win32	Microsoft Corporation	c:\windows\system32\urlm...
✓ text/webviewh...	Windows Shell Common Dll	Microsoft Corporation	c:\windows\system32\shell...
HKLM\SOFTWARE\Classes\Protocols\Handler			
✓ about	Microsoft (R) HTML Viewer	Microsoft Corporation	c:\windows\system32\msht...

Figure 4. 1: Sysinternals Autoruns screenshot with startup items.

4.3 Kansa

Kansa is modular. It features a core script, dozens of collector modules, and analysis scripts to help make sense of the data collected. Kansa takes advantage of Windows Remote Management and PowerShell remoting by using PowerShell’s default non-delegated Kerberos network logons, not CredSSP and, therefore, does not expose credentials to harvesting. Kansa is a great tool with many benefits but particularly useful are its Get-Autorunsc, Get-WMIEvtConsumer, Get-WMIEvtFilter, and Get-WMIFltConBind PowerShell Scripts. [12]

1. Get-Autorunsc - A great piece of a tool for gathering data from many known ASEP (Auto Start Extension Point) locations, including the path to the executable or script, command line arguments, and cryptographic hashes such as MD5.

2. Get-WMIEvt(Consumer/Filter)/Get-WMIFltConBind - Collects data about WMI Event Consumers/Filters/Consumer – Filter Binding. Kansa provides modules that can query and return information that an admin would need to detect WMI persistence.

The disadvantages side to these tools is that they only detect WMI persistence artifacts at a certain snapshot in time. This means that tools like Sysinternals Autoruns and Kansa won’t detect persistence from clever attackers who clean up their code once they’ve performed their actions. The solution to this problem is to use WMI eventing.

4.4 Defensive WMI Event

The eventing subsystem present in WMI could be thought of as the free host-based IDS from Microsoft. Because almost all operating system actions fire a WMI event, such as Instance, Class, Namespace, and Registry Creation and Modification events, WMI is well positioned to catch and alert admins of attacker actions as they occur. Administrators can decide how to receive alerts on events they have created. One popular method is to have a user send an email or an alert popup to notify the admin when an event fires. One of the most powerful features of WMI from an attacker's and defender's perspective is the WMI event, which can be utilized to respond to nearly any operating system event.

For example, a WMI event may be used to trigger an event upon process creation. This could then be used as a means to perform command-line auditing on any Windows OS.

There are Two Classes of WMI Events

1. Events that runs locally in the context of a single process. - Local events last for the lifetime of the host process.
2. Permanent WMI event subscriptions. - Permanent WMI events are stored in the WMI repository, run as SYSTEM, and persist across reboots.

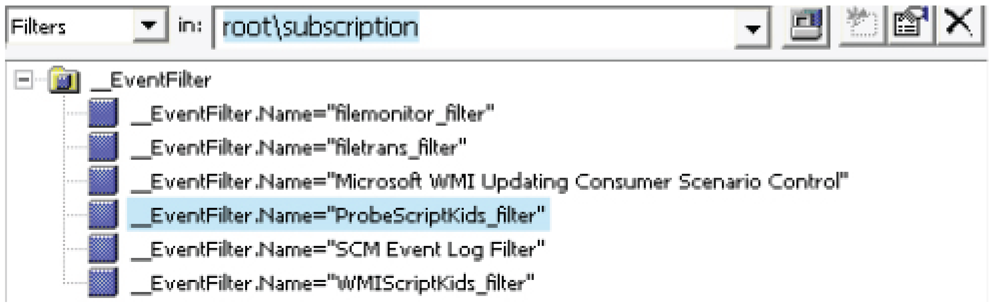
Three Important Parts of a WMI Event to install a permanent WMI event subscription, and they are as follows:

1. An event filter the event of interest,
2. An event consumer an action to perform upon triggering an event, and
3. A filter to consumer binding the registration mechanism that binds a filter to a Consumer.

4.5 Event Filters

Once a filter has been configured, it can be used to receive alerts when new events are created. Event filters are stored in an instance of ROOT\Subscription: __EventFilter object as an example, event filters might be used to describe some of the following events:

• Creation of a process with a certain name;
• Loading of a DLL into a process,
• Creation of an event log with a specific ID;
• Insertion of removable media;
• User logoff; and
• Creation, modification, or deletion of any file or directory.



Filters in: root\subscription

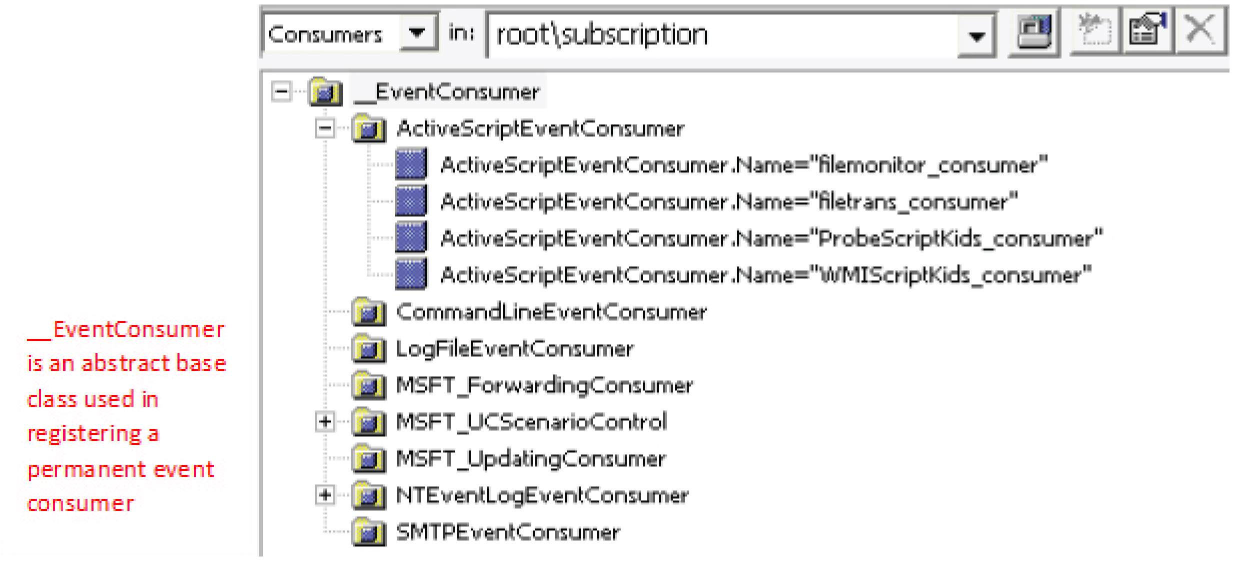
```
__EventFilter
    __EventFilter.Name="filemonitor_filter"
    __EventFilter.Name="filetrans_filter"
    __EventFilter.Name="Microsoft WMI Updating Consumer Scenario Control"
    __EventFilter.Name="ProbeScriptKids_filter"
    __EventFilter.Name="SCM Event Log Filter"
    __EventFilter.Name="WMIScriptKids_filter"
```

Figure 5. 1 : Event Filters shown with wmimgmt.msc

4.6 Event Consumers

An event consumer is a class that is derived from the __EventConsumer system class that represents the action to take on firing an event.

The following useful standard event consumer classes:
• LogFileEventConsumerWrites event data to a specified log file.
• ActiveScriptEventConsumerExecutes an embedded VBScript or JScript script payload.
• NTEventLogEventConsumerCreates an event log entry containing the event data.
• SMTPEventConsumerSends an email containing the event data.
• CommandLineEventConsumerExecutes a command-line program.



Consumers in: root\subscription

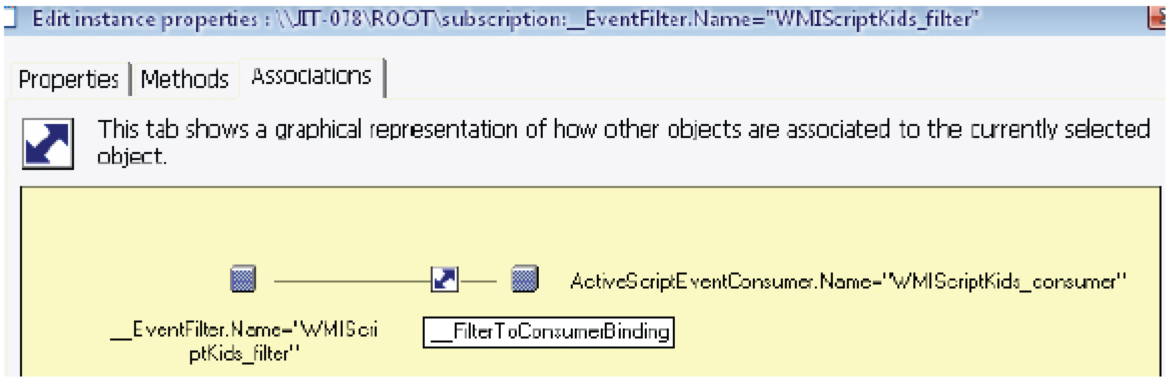
```
__EventConsumer
    ActiveScriptEventConsumer
        ActiveScriptEventConsumer.Name="filemonitor_consumer"
        ActiveScriptEventConsumer.Name="filetrans_consumer"
        ActiveScriptEventConsumer.Name="ProbeScriptKids_consumer"
        ActiveScriptEventConsumer.Name="WMIScriptKids_consumer"
    CommandLineEventConsumer
    LogFileEventConsumer
    MSFT_ForwardingConsumer
    MSFT_UCScenarioControl
    MSFT_UpdatingConsumer
    NTEventLogEventConsumer
    SMTPEventConsumer
```

__EventConsumer is an abstract base class used in registering a permanent event consumer

Figure 5. 2: Event consumer Listing.

Attackers heavily utilizes the ActiveScriptEventConsumer and CommandLineEventConsumer classes when responding to their events. Both these event consumers offer a tremendous amount of flexibility for attackers to execute any payload they want all without needing to drop a single malicious executable or script to disk. [11]

Filter to Consumer Binding Once an event filter and an event consumer have been created, the only thing left to do is to bind them together so the consumer knows off of which filter to base itself. This class instance associates the __EventFilter instance with the __EventConsumer instance. It completes the cycle by relating the class instances with each other. It answers the question, "With what Windows event (__EventFilter) will I execute my script program (__EventConsumer)?" __FilterToConsumerBinding is used in registering permanent event consumers to relate the __EventConsumer instance to the __EventFilter instance.


Edit instance properties : \\JIT-078\ROOT\subscription:__EventFilter.Name="WMIScriptKids_filter"

This tab shows a graphical representation of how other objects are associated to the currently selected object.

Properties
Methods
Associations
ActiveScriptEventConsumer.Name="WMIScriptKids_consumer"
__EventFilter.Name="WMIScriptKids_filter"
__FilterToConsumerBinding

Figure 5. 3: Showing how Filters and Consumers interact with one another by using a FilterToConsumerBinding to tie them together.

4.7 Event Types

4.7.1 Intrinsic Events

Intrinsic events are events that use polling and fire upon the creation, modification, and deletion of any WMI class, object, or namespace. They can also be used to alert to the firing of timers or the execution of WMI methods. The following intrinsic events take the form of system classes (those that start with two underscores) and are present in every WMI namespace:

• __NamespaceOperationEvent,
• __NamespaceModificationEvent,
• __NamespaceDeletionEvent,
• __NamespaceCreationEvent,
• __ClassOperationEvent
• __ClassDeletionEvent,
• __ClassModificationEvent,
• __ClassCreationEvent,
• __InstanceOperationEvent,
• __InstanceCreationEvent,
• __MethodInvocationEvent,

• __InstanceModificationEvent,
• __InstanceDeletionEvent, and
• __TimerEvent

These events are very highly powerful, because they can be used as triggers for nearly any conceivable event in the operating system.

Because of the rate at which intrinsic events can fire, a polling interval must be specified in queries specified with the WQL WITHIN clause.

Because of the polling interval, it is possible on occasion to miss events. For example, if an event query is formed targeting the creation of a WMI class instance and if that instance is created and destroyed within the polling interval, that event would be missed.

The following query is translated to firing on the creation of an instance of a Win32_LogonSession class with a logon type of 2 (Interactive).

```
SELECT * FROM __InstanceCreationEvent WITHIN 15 WHERE TargetInstance
ISA 'Win32_LogonSession' AND TargetInstance.LogonType = 2
```

4.7.2 Extrinsic Events

Extrinsic events solve the potential polling issues related to intrinsic events because they fire immediately on an event occurring.

• The downside is not many extrinsic events are present in WMI.
• The events that do exist are highly powerful but the following extrinsic events may also be of value to an attacker or defender:
- ROOT\CIMV2:Win32_ComputerShutdownEvent
- ROOT\CIMV2:Win32_IP4RouteTableEvent
- ROOT\CIMV2:Win32_ProcessStartTrace
- ROOT\CIMV2:Win32_ModuleLoadTrace
- ROOT\CIMV2:Win32_ThreadStartTrace
- ROOT\CIMV2:Win32_VolumeChangeEvent
- ROOT\CIMV2: Msft_WmiProvider*
- ROOT\DEFAULT:RegistryKeyChangeEvent
- ROOT\DEFAULT:RegistryValueChangeEvent.

This query would capture all executable modules loaded into every process:

```
SELECT * FROM Win32_ModuleLoadTrace
```

4.8 Using WMI CLI Commands for Detection And Removal Of Malicious WMI

4.8.1 Manual Detection: WMI Command Line Tool

To manually detect instances of the threat in a system, the following commands can be used with the Command line tool:

• wmic/namespace:\\root\subscription PATH __EventConsumer get/format:list
• wmic/namespace:\\root\subscription PATH __EventFilter get/format:list
• wmic/namespace:\\root\subscription PATH __FilterToConsumerBinding get/ format:list
• wmic/namespace:\\root\subscription PATH __TimerInstruction get/format:list.

4.8.2 Manual Removal: WMI Command Line Tool

To manually remove instances of the threat in a system, the following commands can be used with the Command line tool:
• wmic/namespace:\\root\subscription PATH__EventConsumer delete
• wmic/namespace:\\root\subscription PATH__EventFilter delete
• wmic/namespace:\\root\subscription PATH__FilterToConsumerBinding delete
• wmic/namespace:\\root\subscription PATH__TimerInstruction delete

4.9 WMI Intrusion Detection Using PowerShell Code Ex

The six examples following show how PowerShell syntax can be used for WMI Detection:

```
New-AlertTrigger -EventConsumer <String> [-TriggerType <String>] [-TriggerName
<String>] [-PollingInterval <Int32>]

New-AlertTrigger -StartupCommand [-TriggerType <String>] [-TriggerName
<String>] [-PollingInterval <Int32>]

New-AlertTrigger -RegistryKey <String> [-TriggerName <String>] [-
PollingInterval <Int32>]

New-AlertAction -Trigger <Hashtable> -Uri <Uri> [-ActionName <String>]

New-AlertAction -Trigger <Hashtable> -EventLogEntry [-ActionName <String>]

Register-Alert [-Binding] <Hashtable> [[-ComputerName] <String[]>]
```

Below are three examples of how one could use PowerShell to alert on either EventConsumers, RegistryKey's or StartupCommands.

```
New-AlertTrigger -EventConsumer ActiveScriptEventConsumer
-TriggerType Creation | New-AlertAction -Uri
'http://127.0.0.1' | Register-Alert -ComputerName
'VigilentHost1'
```

```
New-AlertTrigger -RegistryKey
HKLM:\SYSTEM\CurrentControlSet\Control\Lsa | New-
AlertAction -EventLogEntry | Register-Alert -ComputerName
'192.168.1.24'
```

```
New-AlertTrigger -StartupCommand | New-AlertAction -Uri
'http://www.awesomeSIEM.com' | Register-Alert
```

4.10 Event Logs

WMI, DCom and WinRM events are the components discussed in the following event logs:

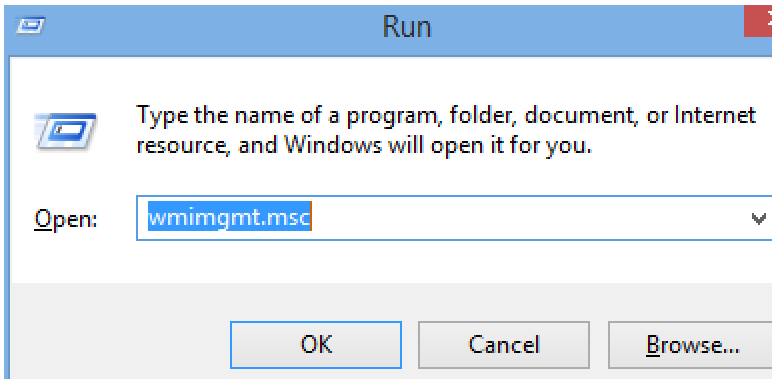
• Microsoft-Windows-WinRM/Operational - Shows failed WinRM connection attempts including the originating IP address
• Microsoft-Windows-WMIActivity/Operational - Contains failed WMI queries and method invocations that may contain evidence of attacker activity
• Microsoft-Windows-DistributedCOM. - Shows failed DCOM connection attempts including the originating IP address.

Many Network level IDS and IPS's can incorporate logs from WMI events stored in the above locations and be configured to perform an action such as marking action for review on dashboard, emailing administrator, preventing an action from performing (ex: file being executed), and more depending on the system.

4.11 Additional WMI Mitigation

4.11.1 WMI Backup & Restore

Having a WMI baseline for the computer systems within a company is a great first step in understanding and being able to identify malicious WMI activity and mitigate it. Wmimgmt.msc as well as the tools listed previously in this thesis can all be used to explore the namespaces and associated classes and objects with ease. However, with dozens of namespaces and thousands of classes, it may not be practical to manually gain a close familiarity with WMI and a baseline thereof.

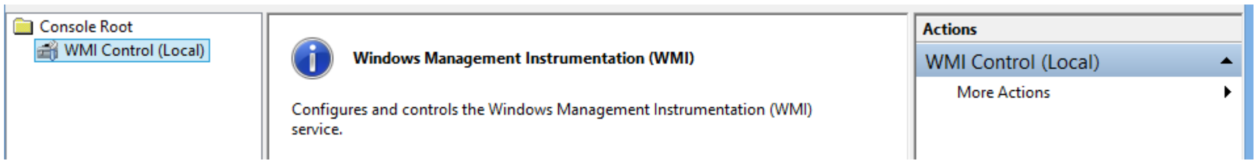

Run

Type the name of a program, folder, document, or Internet resource, and Windows will open it for you.

Open: wmimgmt.msc

Figure 5. 4: WMI Management Startup.

NOTE: Click more actions or right click on WMI Control (Local) -> actions -> properties to access the WMI Control (Local) Properties window.

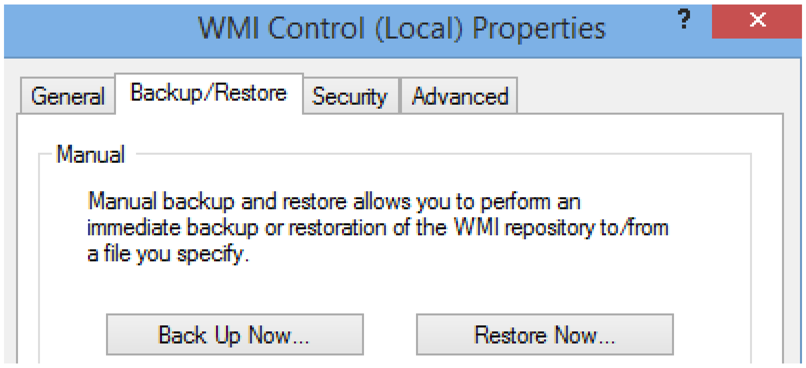


Windows Management Instrumentation (WMI)

Configures and controls the Windows Management Instrumentation (WMI) service.

Figure 5. 5: wmimgmt.msc main menu.

An alternative to familiarization and baselining WMI is to back up the WMI repository once an company has configured the system. Having a backup of the WMI repository allows users the option to restore from the backup file when users discover evidence of malicious WMI activity or suspect potential malicious WMI activity from attackers (see Figure 8).

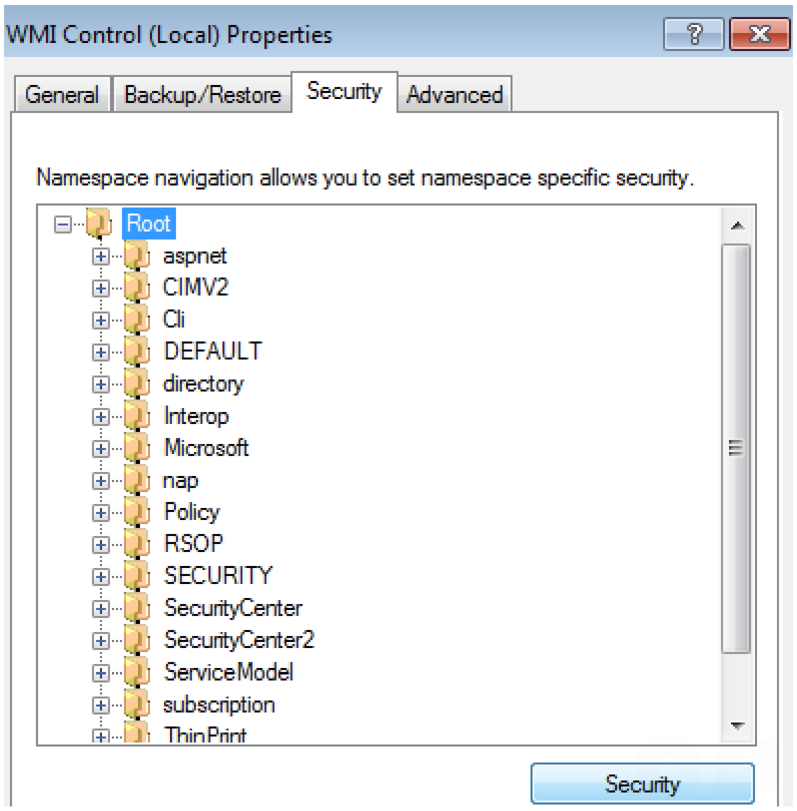

WMI Control (Local) Properties

Manual backup and restore allows you to perform an immediate backup or restoration of the WMI repository to/from a file you specify.

Figure 5. 6 : wmimgmt backup/restore window.

4.12 WMI Access Control

The security tab is located in the same properties window as the Backup/Restore tab. The security tab allows the user to configure user permissions for WMI Repository interaction.

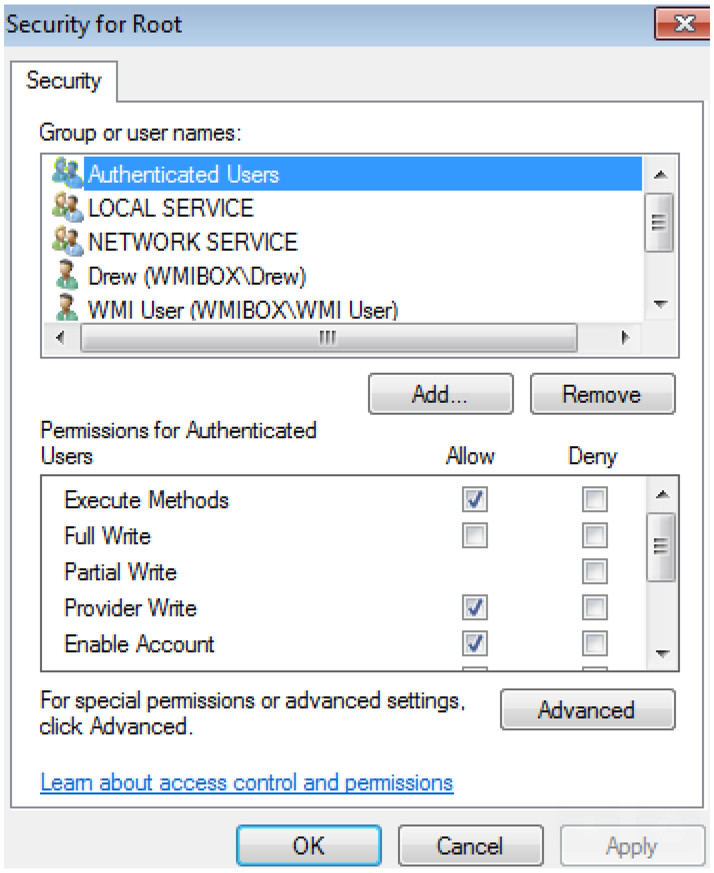


Namespace navigation allows you to set namespace specific security.

```
Root
  aspnet
  CIMV2
  Cli
  DEFAULT
  directory
  Interop
  Microsoft
  nap
  Policy
  RSOP
  SECURITY
  SecurityCenter
  SecurityCenter2
  ServiceModel
  subscription
  ThinPrint
```

Figure 5. 7: Accessing Root directory to apply access control (security) settings.

Organizations can set the controls to match their preferences. For example, permissions can be set for the entire root directory or for specific namespaces like CIMV2. Another way is to have a separate account specifically for WMI management with a unique set of credentials. While some would argue that simple techniques, such as hash dumping, render this defensive method irrelevant from a defense-in depth Perspective, granular access control is applicable to WMI management.

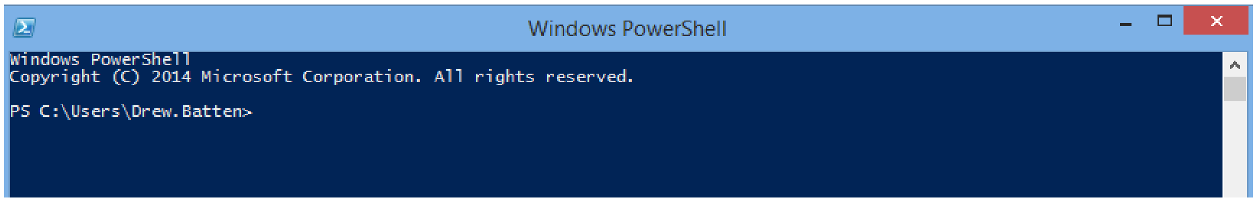


Permissions for Authenticated Users	Allow	Deny
Execute Methods	✓	☐
Full Write	☐	☐
Partial Write		☐
Provider Write	✓	☐
Enable Account	✓	☐

Figure 5. 8: Screenshot of Security for Root.

4.13 PowerShell

PowerShell is becoming more popular and more widely used for a variety of reasons. One reason in particular PowerShell has become a popular method of interacting with WMI is the existence of WMI cmdlets (see Figure).

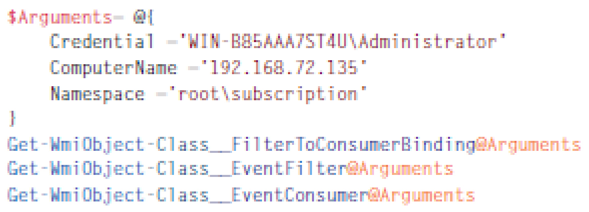


```
Windows PowerShell
Copyright (C) 2014 Microsoft Corporation. All rights reserved.

PS C:\Users\Drew.Batten>
```

• Get-WmiObject
• Get-CimAssociatedInstance
• Get-CimClass
• Get-CimInstance
• Get-CimSession
• Set-WmiInstance
• Set-CimInstance
• Invoke-WmiMethod

• Invoke-CimMethod
• New-CimInstance
• New-CimSession
• New-CimSessionOption
• Register-CimIndicationEvent
• Register-WmiEvent
• Remove-CimInstance
• Remove-WmiObject
• Remove-CimSession

Following is an example of PowerShell code that detects WMI persistence on the specified remote system. It shows just how easy it is to use PowerShell to interact with WMI.

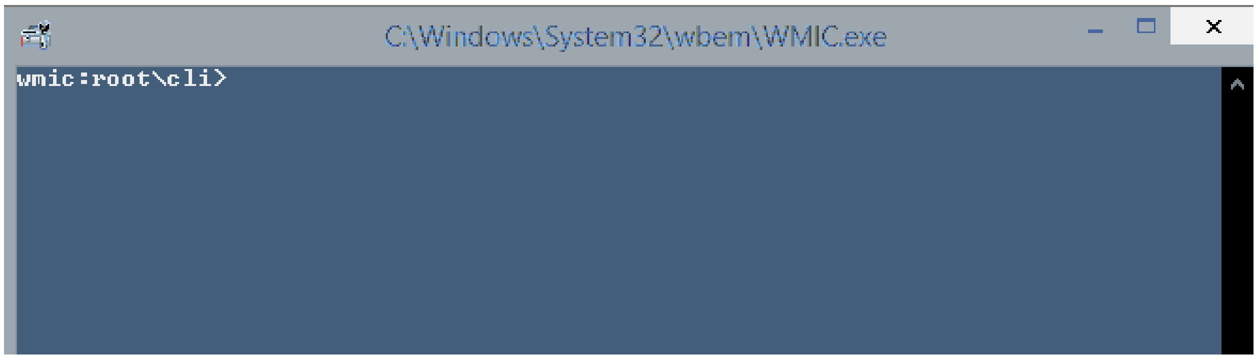


```
$Arguments= @{
    Credential ='WIN-B85AAA7ST4U\Administrator'
    ComputerName ='192.168.72.135'
    Namespace ='root\subscription'
}
Get-WmiObject-Class__FilterToConsumerBinding@Arguments
Get-WmiObject-Class__EventFilter@Arguments
Get-WmiObject-Class__EventConsumer@Arguments
```

Figure 5. 9: Powershell prompt and list of popular WMI cmdlets.

4.14 Wmic.exe

Wmic.exe is a powerful command line utility for interacting with WMI. It has a large amount of default aliases for WMI objects, and users can perform more complicated queries. Wmic.exe can execute WMI methods, and attackers can use it to perform lateral movement by using the Win32_ProcessCreate method.



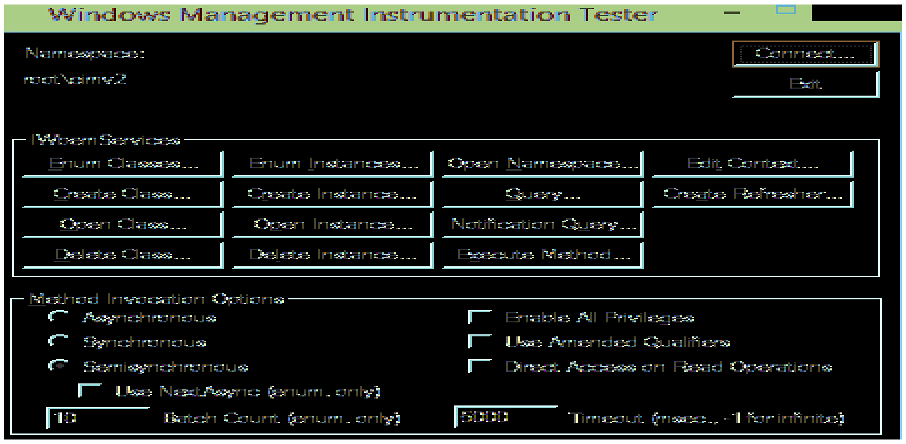
```
wmic:root\cli>
```

Figure 5. 10: Administrators and malicious attackers can use the Wmic.exe terminal to execute WMI methods.

In circumstances where PowerShell is not available, Wmic.exe is a sufficient alternative for performing reconnaissance and basic method invocation.

4.15 Wbemtest.exe

Wbemtest.exe is a powerful GUI WMI diagnostic tool. It isn’t attractive, but it sure is useful. It allows you to explore deep into the WMI repository to discover what an administrator might be able to utilize in PowerShell scripts. It can enumerate object instances, class names, get properties and methods, get property datatypes, perform queries, register events, modify WMI objects and classes, and invoke methods both locally and remotely.



Namespace:
root\cimv2

Semisynchronous [✓]

10 Batch Count (enum. only)

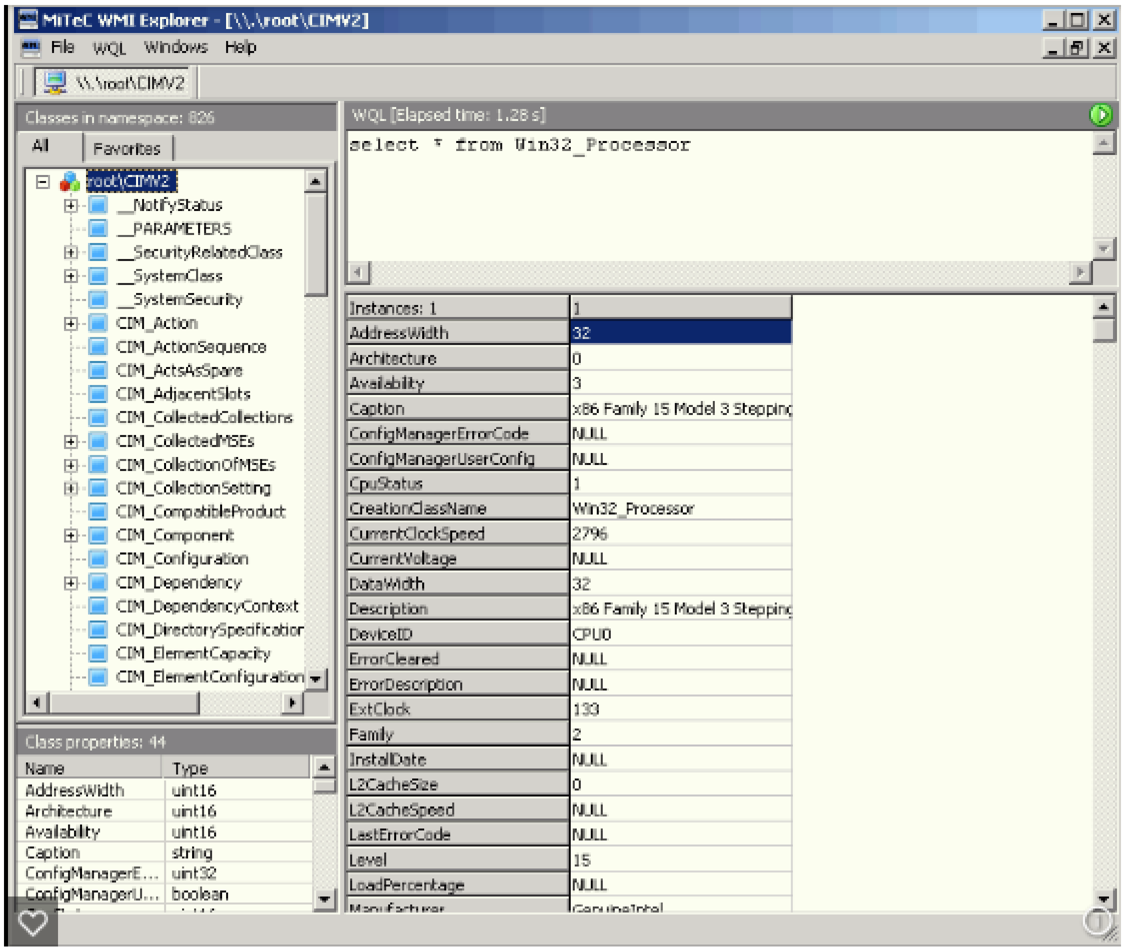
5000 Timeout (msec., -1 for infinite)

Namespace Exploration, Query options and Method

Figure 5. 11: Wbemtest.exe dashboard once connected.

4.16 WMI Explorer

WMI Explorer is a fabulous WMI class discovery tool. It has a polished GUI that allows the user to explore the WMI repository in a hierarchical fashion. It is also able to connect to remote WMI repositories and perform queries.


MiTeC WMI Explorer - [\\.\root\CIMV2]
File
WQL
Windows
Help
\\.\root\CIMV2
Classes in namespace: 826
All
Favorites
WQL [Elapsed time: 1.28 s]
Class properties: 44

root\CIMV2
- __NotifyStatus
- __PARAMETERS
- __SecurityRelatedClass
- __SystemClass
- __SystemSecurity
- CIM_Action
- CIM_ActionSequence
- CIM_ActsAsSpare
- CIM_AdjacentSlots
- CIM_CollectedCollections
- CIM_CollectedMSEs
- CIM_CollectionOfMSEs
- CIM_CollectionSetting
- CIM_CompatibleProduct
- CIM_Component
- CIM_Configuration
- CIM_Dependency
- CIM_DependencyContext
- CIM_DirectorySpecificatior
- CIM_ElementCapacity
- CIM_ElementConfiguration

Name	Type
AddressWidth	uint16
Architecture	uint16
Availability	uint16
Caption	string
ConfigManagerE...	uint32
ConfigManagerU...	boolean

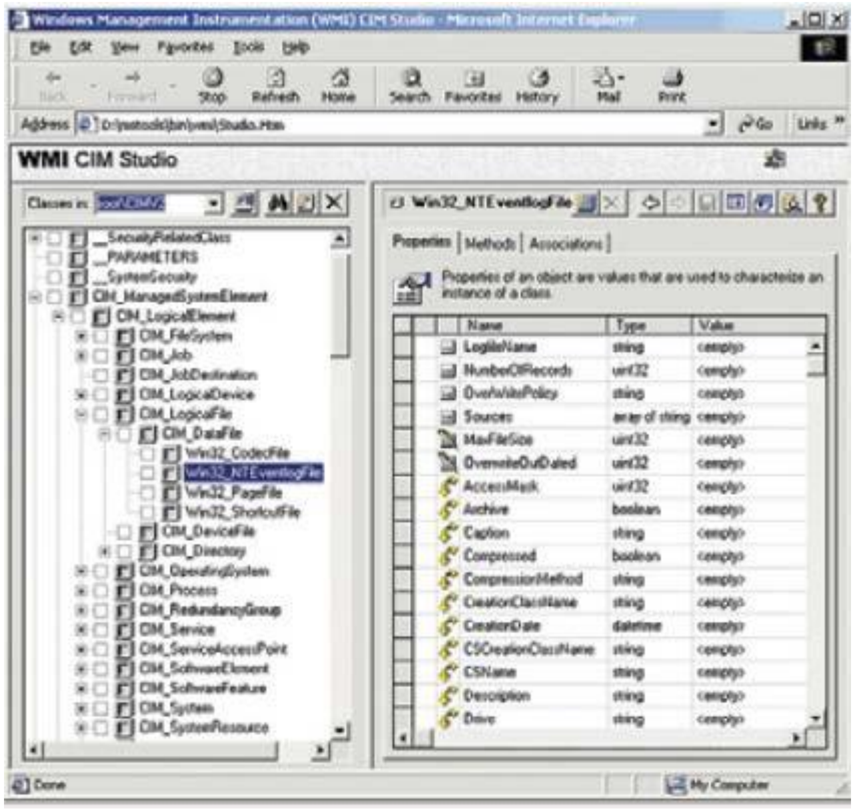
```
select * from Win32_Processor
```

Instances: 1	1
AddressWidth	32
Architecture	0
Availability	3
Caption	x86 Family 15 Model 3 Steppin
ConfigManagerErrorCode	NULL
ConfigManagerUserConfig	NULL
CpuStatus	1
CreationClassName	Win32_Processor
CurrentClockSpeed	2796
CurrentVoltage	NULL
DataWidth	32
Description	x86 Family 15 Model 3 Steppin
DeviceID	CPU0
ErrorCleared	NULL
ErrorDescription	NULL
ExtClock	133
Family	2
InstallDate	NULL
L2CacheSize	0
L2CacheSpeed	NULL
LastErrorCode	NULL
Level	15
LoadPercentage	NULL
Manufacturer	GenuineIntel

Figure 6. 1: Contents of root\CIMV2 directory and WMI Explorer generated WQL language to select all contents from Win32_Processor object.

4.17 CIM Studio

CIM Studio is a free, legacy tool from Microsoft that allows the user to easily browse the WMI repository. Like WMI Explorer, this tool is good for WMI class discovery.



Properties of an object are values that are used to characterize an instance of a class.

Name	Type	Value
LogfileName	string	<empty>
NumberOfRecords	uint32	<empty>
OverWritePolicy	string	<empty>
Sources	array of string	<empty>
MaxFileSize	uint32	<empty>
OverwriteOutDated	uint32	<empty>
AccessMask	uint32	<empty>
Archive	boolean	<empty>
Caption	string	<empty>
Compressed	boolean	<empty>
CompressionMethod	string	<empty>
CreationClassName	string	<empty>
CreationDate	datetime	<empty>
CSCreationClassName	string	<empty>
CSName	string	<empty>
Description	string	<empty>
Drive	string	<empty>

Figure 6. 2: Exploring 5 levels into the root\CIMV2 directory and displaying the contents of selected object.

Chapter 5 - Implementation

Add WINPCAP into the introduction section [13]

5.1 Problem

Why WMI?

Attackers use of WMI to carry out objectives such as system reconnaissance, remote code execution, persistence, lateral movement, covert data storage, and VM detection.

5.2 Steps

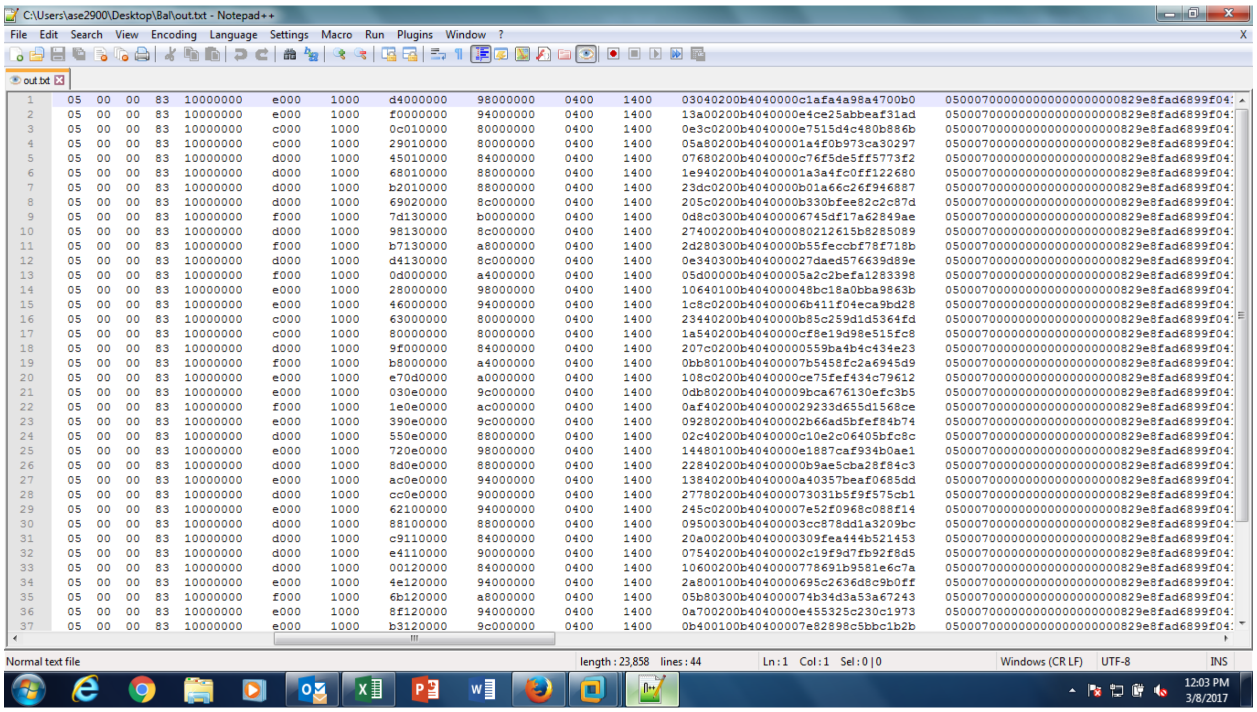
Collect series of packets transmit during wmi query execution. (using tcpdump or wireshark)

Analyze the collected data set (packets or pcap file)

Select and grab the packets contains wmi query. (Wireshark has a functionality to save packet by packet). Full Raw Cap file attached on Appendix III

```
4500 108 09a5 40 0 80 6 5ae5 c0a80a0b c0a80a0a c05a c003 b2814fc2 e10459df 5018 3fa8 46aa 0 5 0 0 83 10000000 e000 1000
d4000000 98000000 400 1400 03040200b4040000c1afa4a98a4700b0
05000700000000000000000829e8fad6899f0418e3cb81d0b0ece1200000000557365720300000006000000030000005700510
04c000000557365722300000046000000230000000730065006c006500630074002002a002000660072006f006d002000570069
006e00330032005f004f00700065007200610074006900 6e0067005300790073007400650 06d000000100000000000000000000
00000000000 0a 4 8 0 0 0100000095b4f6a3b16ec98601000000 4500 108 09d9 40 0 80 6 5ab1 c0a80a0b c0a80a0a c05a c003
b2815d6c e1049255 5018 3f61 f3ad 0 5 0 0 83 10000000 e000 1000 f0000000 94000000 400 1400
13a00200b4040000e4ce25abbeaf31ad
05000700000000000000000829e8fad6899f0418e3cb81d0b0ece1200000000557365720300000006000000030000005700510
04c000000557365722200000044000000220000000730065006c006500630074002002a002000660072006f006d002000570069
006e00330032005f0043006f006d0070007500740065007200530079007300740065006d001000000000000000000000000000
00000000000 0a 4 0c 0 0 01000000ad2065acde68391913000000 4500 0.00E+00 0a07 40 0 80 6 5aa3 c0a80a0b c0a80a0a c05a
c003 b2816d80 e104ca41 5018 3f0e bc51 0 5 0 0 83 10000000 c000 1000 0c010000 80000000 400 1400
0e3c0200b4040000e7515d4c480b886b
```

Structure the hex/binary formatted packets. (Structure: separate the fields (src_ip, dst_ip,..etc)) Doing an analysis on those data. (manually / using neural networks /machine learning techniques) Above analysis can be used to figure out the relationship between the selected packets (common pattern).



```
05  00  00  83  10000000  e000  1000  d4000000  98000000  0400  1400  03040200b4040000c1afa4a98a4700b0  050007000000000000000000000829e8fad6899f04
05  00  00  83  10000000  e000  1000  f0000000  94000000  0400  1400  13a00200b4040000e4ce25abbeaf31ad  050007000000000000000000000829e8fad6899f04
05  00  00  83  10000000  c000  1000  0c010000  80000000  0400  1400  0e3c0200b4040000e7515d4c480b886b  050007000000000000000000000829e8fad6899f04
05  00  00  83  10000000  c000  1000  29010000  80000000  0400  1400  05a80200b40400001a4f0b973ca30297  050007000000000000000000000829e8fad6899f04
05  00  00  83  10000000  d000  1000  45010000  84000000  0400  1400  07680200b4040000c76f5de5ff5773f2  050007000000000000000000000829e8fad6899f04
05  00  00  83  10000000  d000  1000  68010000  88000000  0400  1400  1e940200b40400001a3a4fc0ff122680  050007000000000000000000000829e8fad6899f04
05  00  00  83  10000000  d000  1000  b2010000  88000000  0400  1400  23dc0200b4040000b01a66c26f946887  050007000000000000000000000829e8fad6899f04
05  00  00  83  10000000  d000  1000  69020000  8c000000  0400  1400  205c0200b4040000b330bfee82c2c87d  050007000000000000000000000829e8fad6899f04
05  00  00  83  10000000  f000  1000  7d130000  b0000000  0400  1400  0d8c0300b40400006745df17a62849ae  050007000000000000000000000829e8fad6899f04
05  00  00  83  10000000  d000  1000  98130000  8c000000  0400  1400  27400200b404000080212615b8285089  050007000000000000000000000829e8fad6899f04
05  00  00  83  10000000  f000  1000  b7130000  a8000000  0400  1400  2d280300b4040000b55feccbf78f718b  050007000000000000000000000829e8fad6899f04
05  00  00  83  10000000  d000  1000  d4130000  8c000000  0400  1400  0e340300b404000027daed576639d89e  050007000000000000000000000829e8fad6899f04
05  00  00  83  10000000  f000  1000  0d000000  a4000000  0400  1400  05d00000b40400005a2c2befa1283398  050007000000000000000000000829e8fad6899f04
05  00  00  83  10000000  e000  1000  28000000  98000000  0400  1400  10640100b404000048bc18a0bba9863b  050007000000000000000000000829e8fad6899f04
05  00  00  83  10000000  e000  1000  46000000  94000000  0400  1400  1c8c0200b40400006b411f04eca9bd28  050007000000000000000000000829e8fad6899f04
05  00  00  83  10000000  c000  1000  63000000  80000000  0400  1400  23440200b4040000b85c259d1d5364fd  050007000000000000000000000829e8fad6899f04
05  00  00  83  10000000  c000  1000  80000000  80000000  0400  1400  1a540200b4040000cf8e19d98e515fc8  050007000000000000000000000829e8fad6899f04
05  00  00  83  10000000  d000  1000  9f000000  84000000  0400  1400  207c0200b40400000559ba4b4c434e23  050007000000000000000000000829e8fad6899f04
05  00  00  83  10000000  f000  1000  b8000000  a4000000  0400  1400  0bb80100b40400007b5458fc2a6945d9  050007000000000000000000000829e8fad6899f04
05  00  00  83  10000000  e000  1000  e70d0000  a0000000  0400  1400  108c0200b4040000ce75fef434c79612  050007000000000000000000000829e8fad6899f04
05  00  00  83  10000000  e000  1000  030e0000  9c000000  0400  1400  0db80200b40400009bca676130efc3b5  050007000000000000000000000829e8fad6899f04
05  00  00  83  10000000  f000  1000  1e0e0000  ac000000  0400  1400  0af40200b404000029233d655d1568ce  050007000000000000000000000829e8fad6899f04
05  00  00  83  10000000  e000  1000  390e0000  9c000000  0400  1400  09280200b40400002b66ad5bfef84b74  050007000000000000000000000829e8fad6899f04
05  00  00  83  10000000  d000  1000  550e0000  88000000  0400  1400  02c40200b4040000c10e2c06405bfc8c  050007000000000000000000000829e8fad6899f04
05  00  00  83  10000000  e000  1000  720e0000  98000000  0400  1400  14480100b4040000e1887caf934b0ae1  050007000000000000000000000829e8fad6899f04
05  00  00  83  10000000  d000  1000  8d0e0000  88000000  0400  1400  22840200b40400000b9ae5cba28f84c3  050007000000000000000000000829e8fad6899f04
05  00  00  83  10000000  e000  1000  ac0e0000  94000000  0400  1400  13840200b4040000a40357beaf0685dd  050007000000000000000000000829e8fad6899f04
05  00  00  83  10000000  d000  1000  cc0e0000  90000000  0400  1400  27780200b404000073031b5f9f575cb1  050007000000000000000000000829e8fad6899f04
05  00  00  83  10000000  e000  1000  62100000  94000000  0400  1400  245c0200b40400007e52f0968c088f14  050007000000000000000000000829e8fad6899f04
05  00  00  83  10000000  d000  1000  88100000  88000000  0400  1400  09500300b40400003cc878dd1a3209bc  050007000000000000000000000829e8fad6899f04
05  00  00  83  10000000  d000  1000  c9110000  84000000  0400  1400  20a00200b4040000309fea444b521453  050007000000000000000000000829e8fad6899f04
05  00  00  83  10000000  d000  1000  e4110000  90000000  0400  1400  07540200b40400002c19f9d7fb92f8d5  050007000000000000000000000829e8fad6899f04
05  00  00  83  10000000  d000  1000  00120000  84000000  0400  1400  10600200b404000778691b9581e6c7a  050007000000000000000000000829e8fad6899f04
05  00  00  83  10000000  e000  1000  4e120000  94000000  0400  1400  2a800100b4040000695c2636d8c9b0ff  050007000000000000000000000829e8fad6899f04
05  00  00  83  10000000  f000  1000  6b120000  a8000000  0400  1400  05b80300b404000074b34d3a53a67243  050007000000000000000000000829e8fad6899f04
05  00  00  83  10000000  e000  1000  8f120000  94000000  0400  1400  0a700200b4040000e455325c230c1973  050007000000000000000000000829e8fad6899f04
05  00  00  83  10000000  e000  1000  b3120000  9c000000  0400  1400  0b400100b40400007e82898c5bbc1b2b  050007000000000000000000000829e8fad6899f04
```

C:\Users\ase2900\Desktop\Bal\out.txt - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
out.txt
Normal text file
length : 23,858 lines : 44
Ln : 1 Col : 1 Sel : 0 | 0
Windows (CR LF)
UTF-8
INS
12:03 PM
3/8/2017

Figure 5. 12 – PCAP Output file

according to the analysis,

version == “0500”

op_num == “1400” | “1500” | .. etc

Describe the Packet Structure of the DCERPC **(use tcp, udp, smb)** use diagrams

Implementation

Minimum Size of the DCERPC packet (eth(0) + ipheader (20) + tcpheder (20) + .. + stub_data(0) + .. + .. = 118)

Explanations of Code parts

```
while ((res = pcap_next_ex(fp, &header, &pkt_data)) >= 0)
        {
                if (res == 0)
                /* Timeout elapsed */
                        continue;
                process(pkt_data, header->caplen);
}
```

above code snippet is used to capture each and every frame using winpcap library**.**

5.3 Methods in RPC Opnum Order

The IWbemServices interface exposes methods that MUST provide management services to client processes. The implementation MUST implement all methods and return errors if the semantics of the operation cannot be completed. IWbemServices defines the execution scope for all methods implemented on the interface. The initial scope MUST be established by the IWbemLevel1Login::NTLMLogin call, which returns the interface pointer. [11]

Table 5. 1WMI Opnum Table

Method	Description
OpenNamespace	Provides the client with an IWbemServices interface pointer that is scoped to the requested namespace. Opnum: 3
CancelAsyncCall	Cancels a currently pending asynchronous method call identified by the IWbemObjectSink pointer passed to the initial asynchronous method. Opnum: 4
QueryObjectSink	Obtains a notification handler that allows the client to send events directly to the server. Opnum: 5
GetObject	Retrieves a CIM class or a CIM instance. Opnum: 6
GetObjectAsync	Asynchronous version of the IWbemServices::GetObject method. Opnum: 7
PutClass	Creates a new class or updates an existing class in the namespace associated with the current IWbemServices interface. Opnum: 8

Method	Description
PutClassAsync	Asynchronous version of the IWbemServices::PutClass method. Opnum: 9
DeleteClass	Deletes a specified class from the namespace associated with the current IWbemServices interface Opnum: 10
DeleteClassAsync	Asynchronous version of the IWbemServices::DeleteClass method. Opnum: 11
CreateClassEnum	Creates a class enumeration. Opnum: 12
CreateClassEnumAsync	Asynchronous version of the IWbemServices::CreateClassEnum method. Opnum: 13
PutInstance	Creates or updates an instance of an existing class. Opnum: 14
PutInstanceAsync	Asynchronous version of the PutInstance method. Opnum: 15
DeleteInstance	Deletes an instance of an existing class from the namespace that is pointed to by the IWbemServices interface object that is used to call the method. Opnum: 16

DeleteInstanceAsync	Asynchronous version of the IWbemServices::DeleteInstance method. Opnum: 17
CreateInstanceEnum	Creates an instance enumeration of all class instances that satisfy the selection criteria. Opnum: 18
CreateInstanceEnumAsync	Asynchronous version of the IWbemServices::CreateInstanceEnum method. Opnum: 19
ExecQuery	Returns an enumerable collection of IWbemClassObject interface objects based on a query. Opnum: 20
ExecQueryAsync	Asynchronous version of the IWbemServices::ExecQuery method. Opnum: 21
ExecNotificationQuery	Server runs a query to receive events when called by a client to request subscription to the events. Opnum: 22
ExecNotificationQueryAsync	Asynchronous version of the IWbemServices::ExecNotificationQuery method. Opnum: 23
ExecMethod	Executes a CIM method implemented by a CIM class or a CIM instance retrieved from the IWbemServices interface. Opnum: 24
ExecMethodAsync	Asynchronous version of the IWbemServices::ExecMethod method. Opnum: 25

Case 1: section can use to detect BruteForce attacks.

filter the packet by length(minimum length should be 118), Full code attached on Appendix II

```
        if (iph->tot_len > 118){
```

filter the packet by DCERPC version. bit or binary **x** to **y** should be 0500

```
if (dce_h->dcerpc_h.version == 5){
```

filter the packet by pack_type. …

```
if (dce_h->dcerpc_h.pack_type == 16){
```

filter the packet by auth type. …

```
if (dce_auth->auth.auth_type == 10){
```

if auth_level == 2 that is an attempting to connect.

auth_level == 4 that is an authorization.

below code snippet is check for the auth_level and categorize the packet according to the relevant section.

Case 2: section can use to detect Authorizations **(in this point attacker is able to crack the password of the host machine).**

```
switch (dce_auth->auth.auth_level){
                                        case 2:
                                                memset(id, 0, 4);
                                                alert(buf, "Attempting to connect");
                                                process_auth(buf,       nbytes,       sizeof(struct
dcerpc_pack_auth), nbytes);
                                                log_auth(buf);
                                                break;
                                        case 3:
                                                memset(id, 0, 4);
                                                _snprintf(id,       4,       "%d",       dce_auth-
>auth.auth_context_id);
```

```
                    strncpy((alrt + 37), id, 4);
                    alert(buf, alrt);
                    process_auth(buf,        nbytes,        sizeof(struct
dcerpc_pack_auth), nbytes);
                    log_auth(buf);
                    break;
                }
```

if op_num == 20 | 21 | … (this is the decimal format of 14, 15, 16, ..), this packet carries the wmi query. by decoding the stub data field can get the execution type actively.

```
else{
        switch (dce_h->dcerpc_h.op_num){
        case 20:
        case 21:
        case 22:
        case 23:
        case 24:
        case 25:
                alert_ex(buf, "Executing  :  ");
                process_data(buf, nbytes, 118, (nbytes - 24));
                log_ex(buf);
                break;
}
}
```

Decode data by removing non ASCII characters.

```
void process_data(const u_char *buf, size_t nbytes, int spos, int epos){
        memset(data, 0, sizeof(BUF_SIZE));
        unsigned char   ch;
        int j = 0;
        for (int i = spos; i < epos; i++){
```

```
        ch = buf[i];
        if ((ch < 0x20) || (ch > 0x7E)){
            continue;
        }
```

Code Descriptions

```
data[j++] = ch;
    }
    data[j] = '\0';
    std::string text(data);
    std::size_t found = -1;
    if ((found = text.find("object:Win32_")) == std::string::npos){
        if ((found = text.find("Win32_")) == std::string::npos){
            if ((found = text.find("cim_datafile")) == std::string::npos){
                printf("%s\n", data);
            }
        }
    }

    if (found != std::string::npos){
        memset(command, 0, 50);
        int j = 0;
        for (int i = found;; i++){
            if (data[i] > 0x7A || data[i] < 0x30)
                break;
            command[j++] = data[i];
        }
```

Code Descriptions

```
command[j] = '\0';
                printf("%s", command);
        }

        printf("\n");
}
```

Logging the results.

```
void log_ex(const u_char *buf){
        struct dcerpc_pack *dce_h = (struct dcerpc_pack *)buf;
        fprintf(log_fd, "%s,", time_buf);                       // timestamp
        fprintf(log_fd, "%s,", sour);                           // source ip address
        fprintf(log_fd, "%s,", dest);                           // destination ip address

        fprintf(log_fd, "%u,", ntohs(dce_h->tcp.source));       // tcp source address
        fprintf(log_fd, "%u,", ntohs(dce_h->tcp.dest));                 // tcp destination address
        fprintf(log_fd, ",");
        fprintf(log_fd, ",");
        fprintf(log_fd, ",");
        fprintf(log_fd, "%d,", dce_h->dcerpc_h.op_num);                 // dcerpc packet op num
        fprintf(log_fd,     "%s,",     opnum_methods[dce_h->dcerpc_h.op_num]);//     method
        fprintf(log_fd, "%s,", command);                        // executed command
```

Code Descriptions

```
fprintf(log_fd, "%s", data);                            // raw data
        fprintf(log_fd, "\n");
        fflush(stdout);
        fflush(log_fd);
}
```

Just discuss below functions are use to alert the attacks actively.

```
void alert_ex(const u_char *buf, char *msg){

void alert(const u_char *buf, char *msg){
```

The output log file format and the details are attached below details on log goes here.

Initially this scenario was tried with Snort where some limitations were discovered since the software was not performing consistently in handling WMI packets.

The captures accuracy on capturing WMI queries was less than 75% and it was evident that snort was not the ideal implementation to detect WMI queries.

Demonstration process of the implementation is described in the below steps with the relevant functionalities explained in each and every part of the solution.

The environment consists of four workstations which comprises of Windows 7 (attacker) Windows Server 2008 8 (Victim) Windows 8.1 (Admin) and a Samba Workstation (Arcsight Logger) In order to carry out this process the attacker will need to carry out some ground work in order to successfully accomplish the tasks which includes activities in relation to bruteforce or an attack of that nature to get the authentication credentials and also some means of social engineering to get the authentication credentials and vulnerability existence details. Below Figure displays all 4 workstations which is involved in the scenario for the solution.

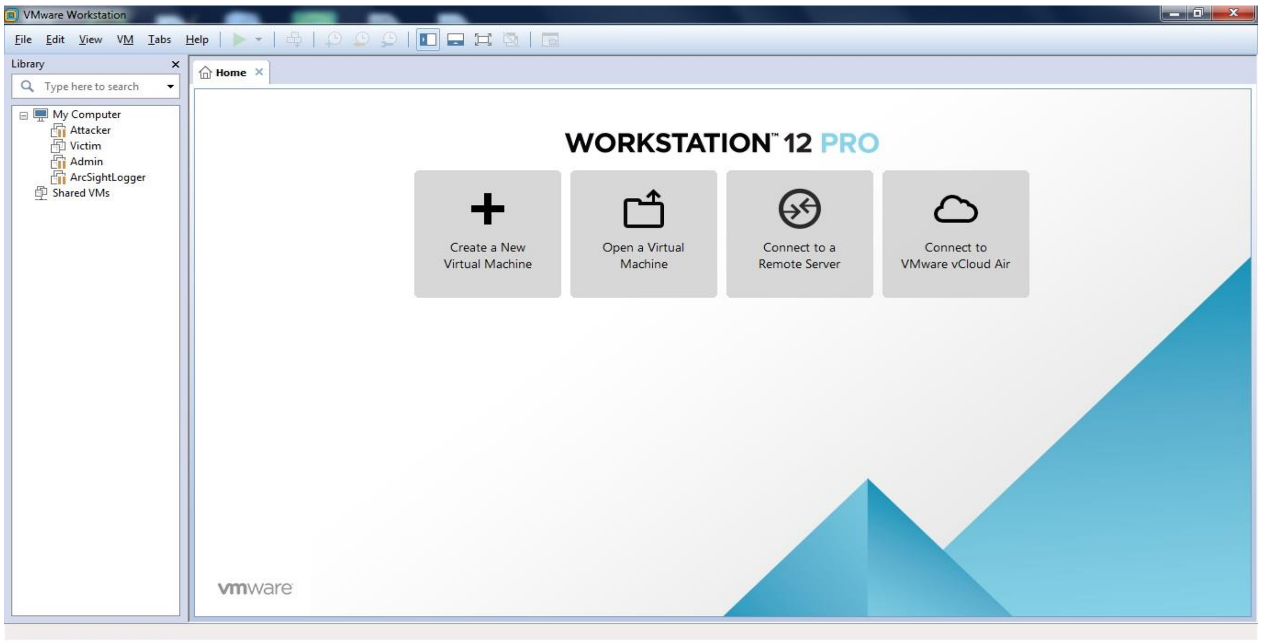
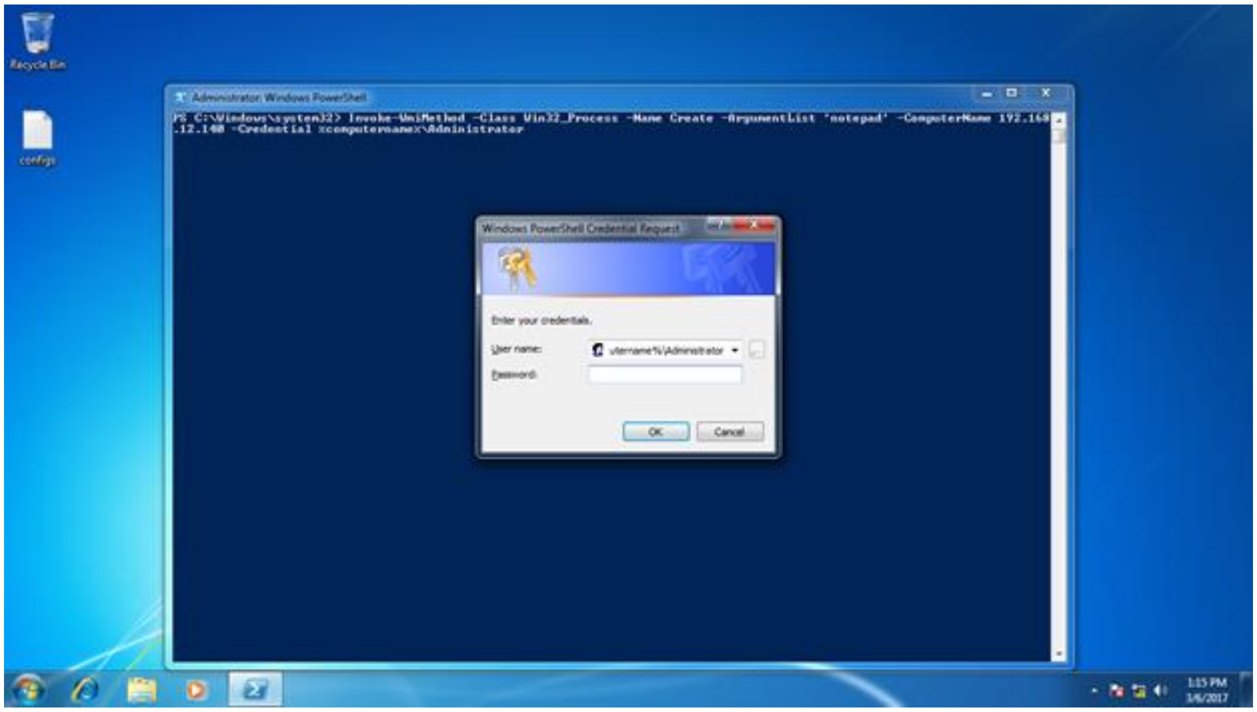


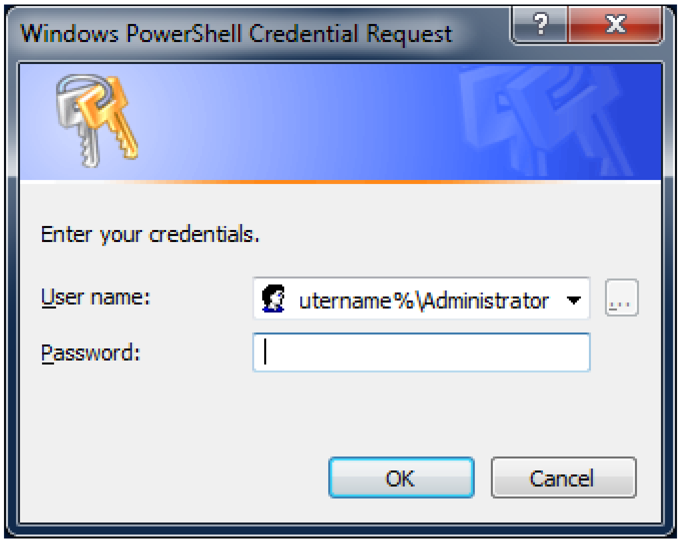
Figure 5. 13 – Workstations involved in the scenario

5.4 Attacker Machine

Attacker machine contains powershell which is used to execute the WMI Authentication/Service commands to initiate the process Once one of the above commands are executed it will prompt from the victim's authentication credentials and on successful authentication the required task will be performed.



```
PS C:\Windows\system32> Invoke-WmiMethod -Class Win32_Process -Name Create -ArgumentList 'notepad' -ComputerName 192.168
.12.140 -Credential %computername%\Administrator
```



Windows PowerShell Credential Request

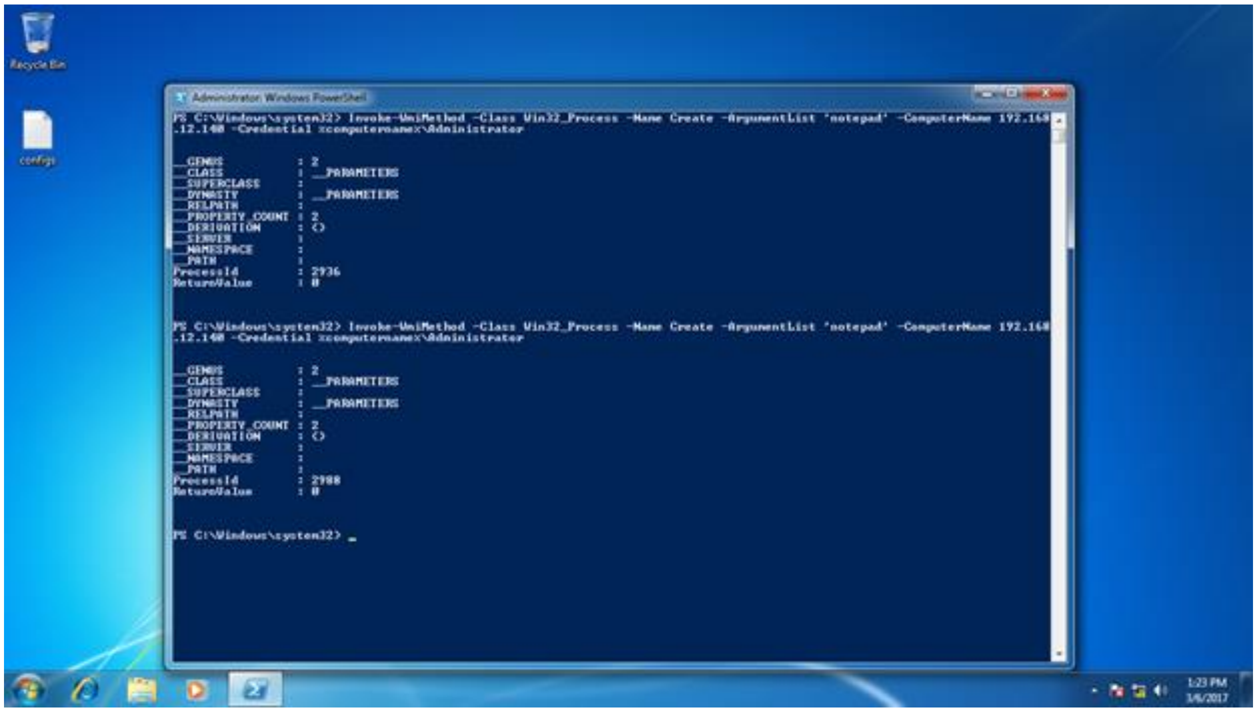
Enter your credentials.

User name: utername%\Administrator

Password:

Figure 5. 14 - Attacker Machine

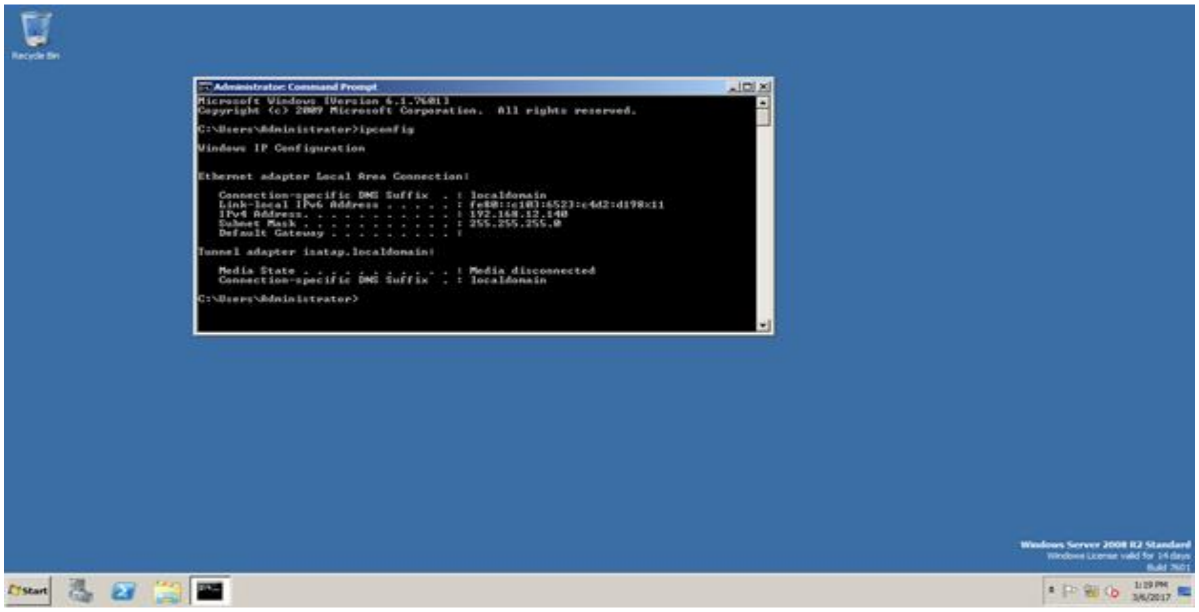
The status of the executed commands is displayed on the PowerShell window



```
PS C:\Windows\system32> Invoke-WmiMethod -Class Win32_Process -Name Create -ArgumentList 'notepad' -ComputerName 192.168
.12.140 -Credential xcomputernamex\Administrator


__GENUS          : 2
__CLASS          : __PARAMETERS
__SUPERCLASS     :
__DYNASTY        : __PARAMETERS
__RELPATH        :
__PROPERTY_COUNT : 2
__DERIVATION     : {}
__SERVER         :
__NAMESPACE      :
__PATH           :
ProcessId        : 2936
ReturnValue      : 0


PS C:\Windows\system32> Invoke-WmiMethod -Class Win32_Process -Name Create -ArgumentList 'notepad' -ComputerName 192.168
.12.140 -Credential xcomputernamex\Administrator


__GENUS          : 2
__CLASS          : __PARAMETERS
__SUPERCLASS     :
__DYNASTY        : __PARAMETERS
__RELPATH        :
__PROPERTY_COUNT : 2
__DERIVATION     : {}
__SERVER         :
__NAMESPACE      :
__PATH           :
ProcessId        : 2988
ReturnValue      : 0


PS C:\Windows\system32> _
```

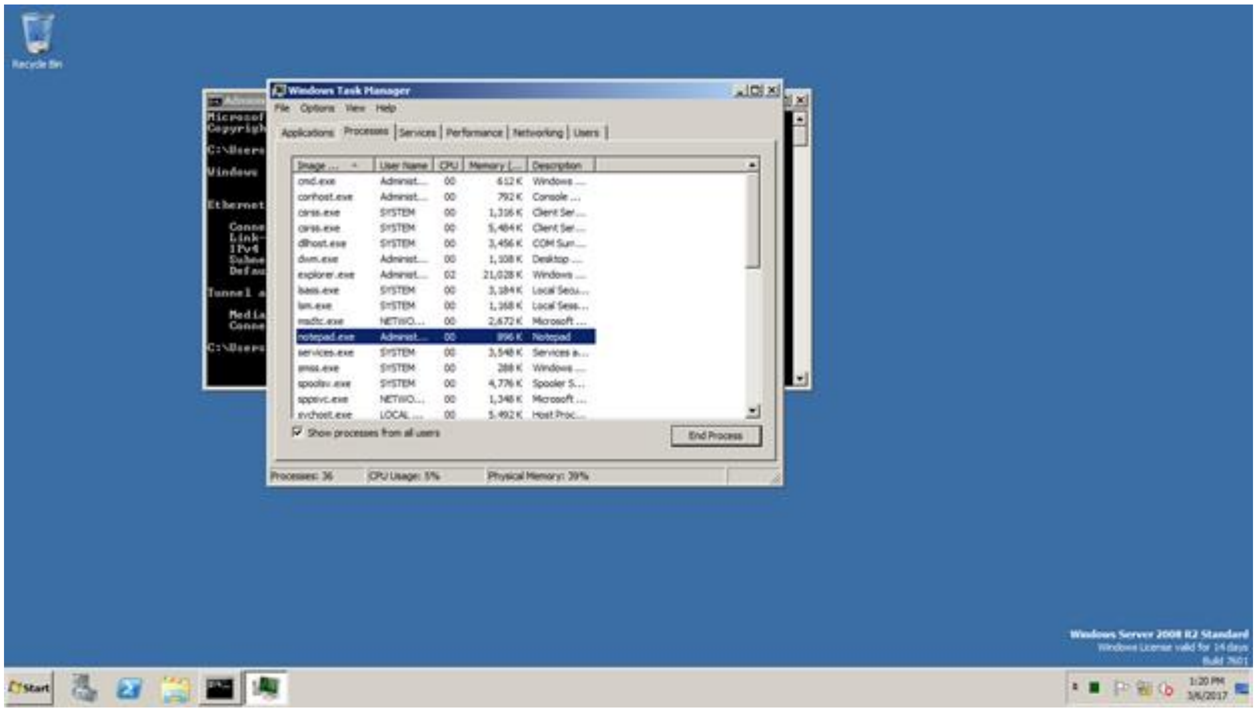
5.5 Victim machine

Once an authentication is succeeded on the victims machine a desired service can be executed on the victim's machines without the knowledge of the victim and also on this case victim does not see the program running on his or her machine This mechanism can be used to exploit and run a known vulnerability for Windows



```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\Administrator>ipconfig

Windows IP Configuration


Ethernet adapter Local Area Connection:

   Connection-specific DNS Suffix  . : localdomain
   Link-local IPv6 Address . . . . . : fe80::c103:6523:c4d2:d198%11
   IPv4 Address. . . . . . . . . . . : 192.168.12.140
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . :

Tunnel adapter isatap.localdomain:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . : localdomain

C:\Users\Administrator>
```

Below shows the notepad.exe is being executed by running the required PowerShell commands with the relevant authentications

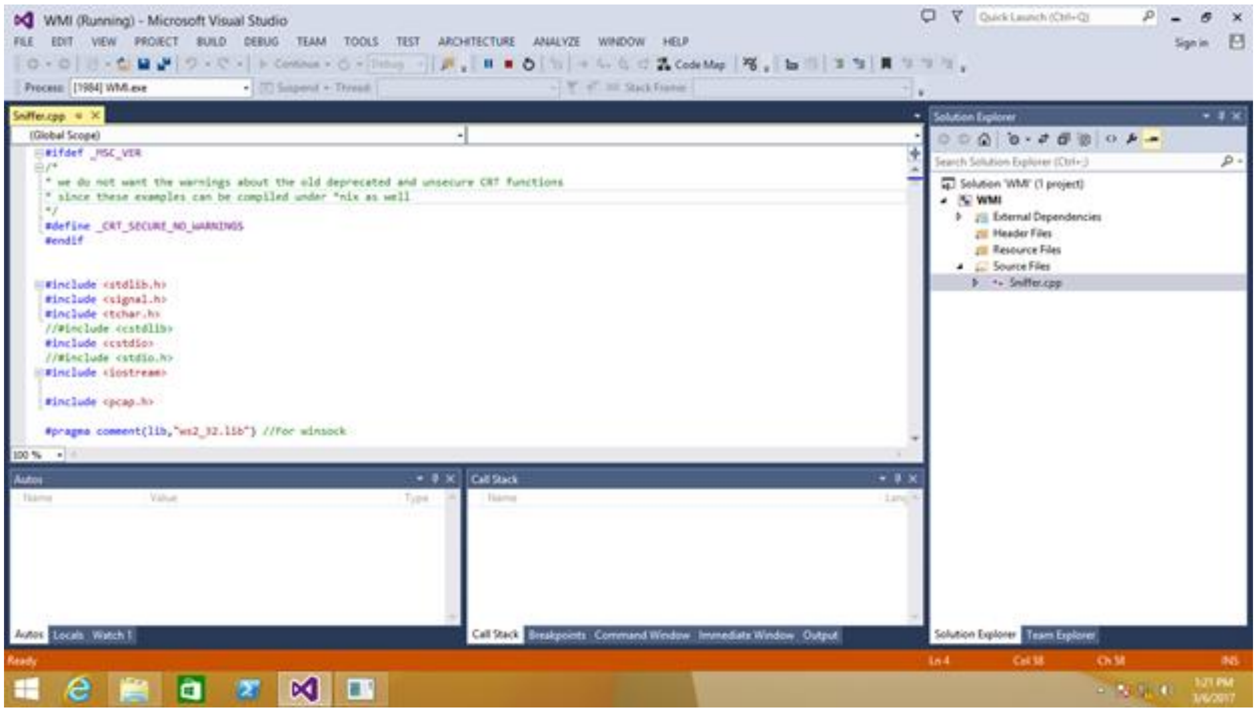


```
Microsof
Copyrigh

C:\Users

Windows

Ethernet

    Conne
    Link-
    IPv4
    Subne
    Defau

Tunnel a

    Media
    Conne

C:\Users
```

```
Windows Task Manager
Image ...       User Name    CPU  Memory (...  Description
cmd.exe         Administ...  00       612 K    Windows ...
conhost.exe     Administ...  00       792 K    Console ...
csrss.exe       SYSTEM       00     1,316 K    Client Ser...
csrss.exe       SYSTEM       00     5,484 K    Client Ser...
dllhost.exe     SYSTEM       00     3,456 K    COM Surr...
dwm.exe         Administ...  00     1,108 K    Desktop ...
explorer.exe    Administ...  02    21,028 K    Windows ...
lsass.exe       SYSTEM       00     3,184 K    Local Secu...
lsm.exe         SYSTEM       00     1,168 K    Local Sess...
msdtc.exe       NETWO...     00     2,672 K    Microsoft ...
notepad.exe     Administ...  00       896 K    Notepad
services.exe    SYSTEM       00     3,548 K    Services a...
smss.exe        SYSTEM       00       288 K    Windows ...
spoolsv.exe     SYSTEM       00     4,776 K    Spooler S...
sppsvc.exe      NETWO...     00     1,348 K    Microsoft ...
svchost.exe     LOCAL ...    00     5,492 K    Host Proc...

Processes: 36    CPU Usage: 5%    Physical Memory: 39%
```

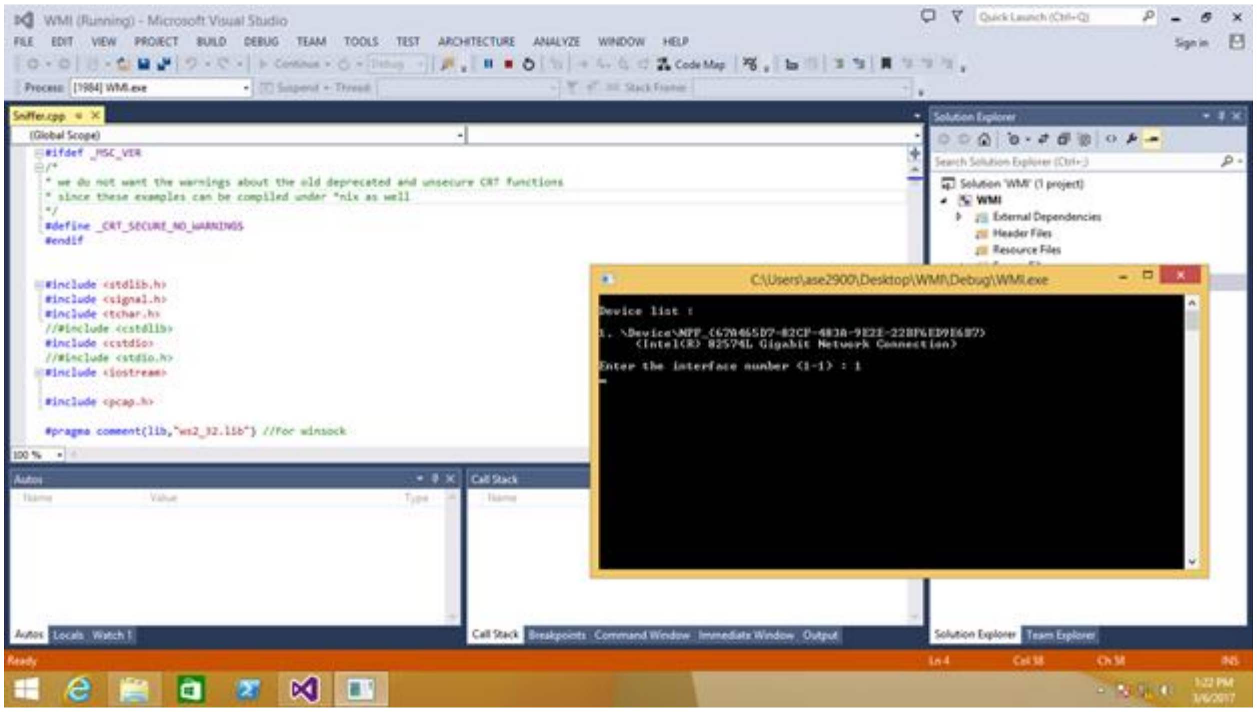
5.6 Administrator

The relevant visual C program is executed and the network interface is selected where it will monitor all the activities performed by the attacker’s network interface and monitored and the output activities are logged in to a txt file. In return the txt file is integrated to the Arcsight log management system which is installed on the centos machine and log correlation activities are performed accordingly.



```
#ifdef _MSC_VER
/*
 * we do not want the warnings about the old deprecated and unsecure CRT functions
 * since these examples can be compiled under *nix as well
 */
#define _CRT_SECURE_NO_WARNINGS
#endif


#include <stdlib.h>
#include <signal.h>
#include <tchar.h>
//#include <cstdlib>
#include <cstdio>
//#include <stdio.h>
#include <iostream>

#include <pcap.h>

#pragma comment(lib,"ws2_32.lib") //For winsock
```

Figure 5. 15 – Visual C+ cording of the tool

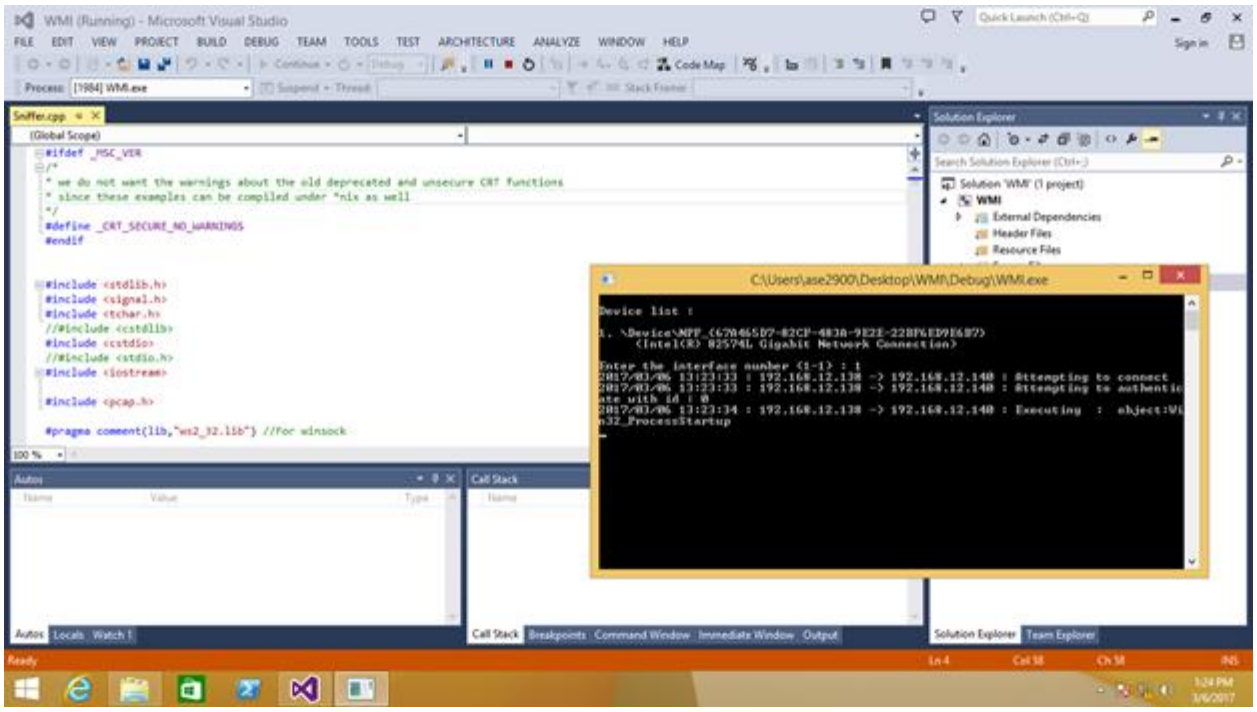


```
#ifdef _MSC_VER
/*
 * we do not want the warnings about the old deprecated and unsecure CRT functions
 * since these examples can be compiled under *nix as well
 */
#define _CRT_SECURE_NO_WARNINGS
#endif


#include <stdlib.h>
#include <signal.h>
#include <tchar.h>
//#include <cstdlib>
#include <cstdio>
//#include <stdio.h>
#include <iostream>

#include <pcap.h>

#pragma comment(lib,"ws2_32.lib") //For winsock
```

C:\Users\ase2900\Desktop\WMI\Debug\WMI.exe

```
Device list :

1. \Device\NPF_{67A465D7-82CF-483A-9E2E-228F6ED9E687}
      (Intel(R) 82574L Gigabit Network Connection)

Enter the interface number (1-1) : 1
_
```

Figure 5. 16 – Prompt selection of Network Interface



```
#ifdef _MSC_VER
/*
 * we do not want the warnings about the old deprecated and unsecure CRT functions
 * since these examples can be compiled under *nix as well
 */
#define _CRT_SECURE_NO_WARNINGS
#endif


#include <stdlib.h>
#include <signal.h>
#include <tchar.h>
//#include <cstdlib>
#include <cstdio>
//#include <stdio.h>
#include <iostream>

#include <pcap.h>

#pragma comment(lib,"ws2_32.lib") //For winsock
```

C:\Users\ase2900\Desktop\WMI\Debug\WMI.exe

```
Device list :

1. \Device\NPF_{67A465D7-82CF-483A-9E2E-228F6ED9E687}
      (Intel(R) 82574L Gigabit Network Connection)

Enter the interface number (1-1) : 1
2017/03/06 13:23:33 : 192.168.12.138 -> 192.168.12.140 : Attempting to connect
2017/03/06 13:23:33 : 192.168.12.138 -> 192.168.12.140 : Attempting to authentic
ate with id : 0
2017/03/06 13:23:34 : 192.168.12.138 -> 192.168.12.140 : Executing  :  object:Wi
n32_ProcessStartup
_
```

Figure 5. 17 – Output of available network interfaces

Chapter 6 - Testing & Evaluation

6.1 Log management

HPE Security Arcsight provides a range of device-specific Smart Connectors with which to gather security event information. The connectors send normalized security events to the specified destination for storage and further processing. Flex Connectors are custom connectors you define to gather security events from log files, databases, and other software and devices. Flex Connectors let you create custom connectors that can read and parse information from third-party devices and map that information to Arcsight's event schema.

ArcSight FlexConnector Types

- ArcSight FlexConnector File
- ArcSight FlexConnector ID-Based DB
- ArcSight FlexConnector Multiple DB
- ArcSight FlexConnector Regex File
- ArcSight FlexConnector Regex Folder File
- ArcSight FlexConnector Simple Network Management Protocol (SNMP Unified)
- ArcSight FlexConnector Time-Based DB
- ArcSight FlexConnector XML File

Regex File: Choose this type if the source log files have one event per line, but the format of each line varies based on the type of event information. In this case, each line shares a common section (for example, the date and hostname), but the number and content of the other fields on the line varies. The regular expression-based FlexConnectors require a familiarity with Java-compatible regular expressions. Sample log file attached on Appendix I.

6.2 Windows Firewall Configuration

//add firewall (group) rule and enable it

netsh advfirewall firewall set rule group="Windows Management Instrumentation (WMI)" new enable=yes

// add input firewall rule to "svchost.exe" on TCP port 135

netsh advfirewall firewall add rule dir=in name="DCOM" program=%systemroot%\system32\svchost.exe service=rpcss action=allow protocol=TCP localport=135

//add input firewall rule to "svchost" on TCP ANY windows management

```
netsh advfirewall firewall add rule dir=in name ="WMI"
program=%systemroot%\system32\svchost.exe service=winmgmt action = allow protocol=TCP
localport=any
```

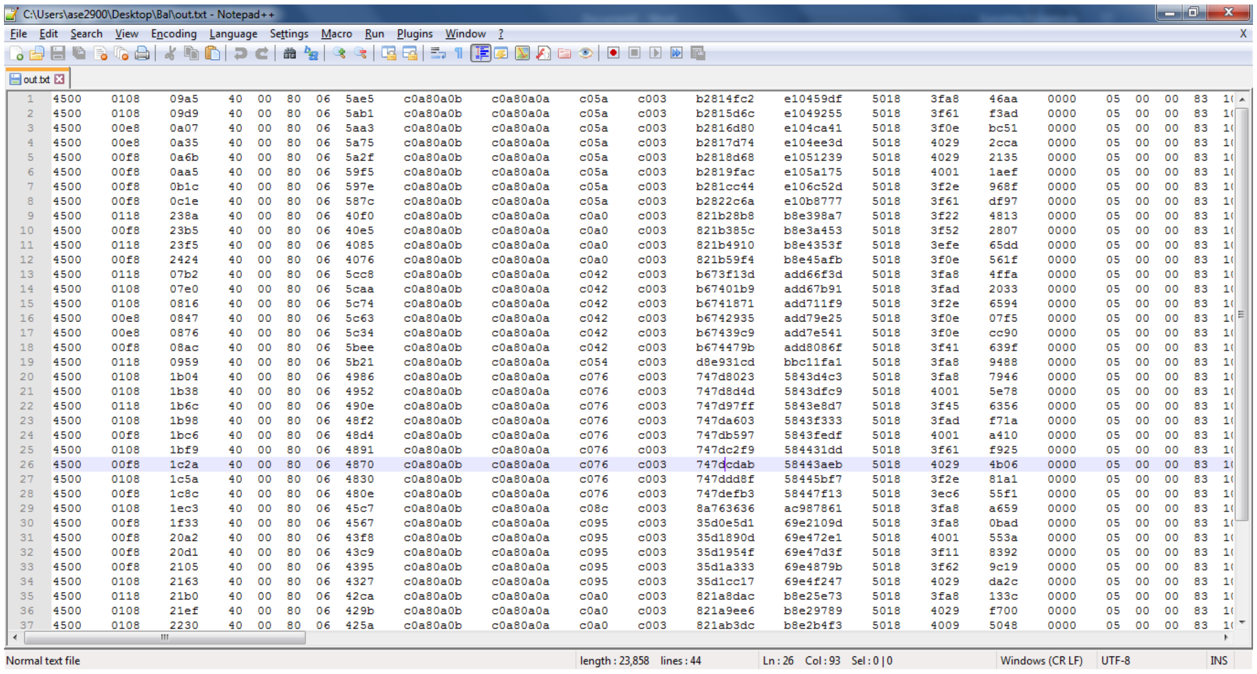
Victim (Windows 2008) - 192.168.10.10

Attacker (Windows 7) - 192.168.10.12

6.3 Execution of Commands from attacker

```
Invoke-WmiMethod -Class Win32_Process -Name Create -ArgumentList 'notepad.exe' -ComputerName 192.168.12.129 -Credential %computername%\Administrator

Get-WmiObject Win32_LoggedOnUser -ComputerName 192.168.12.129 -Credential %computername%\administrator

Get-WmiObject Win32_share -ComputerName 192.168.12.129 -Credential %computername%\administrator

Get-WmiObject cim_datafile -ComputerName 192.168.12.129 -Credential %computername%\administrator
```

6.4 Sorted PCAP log

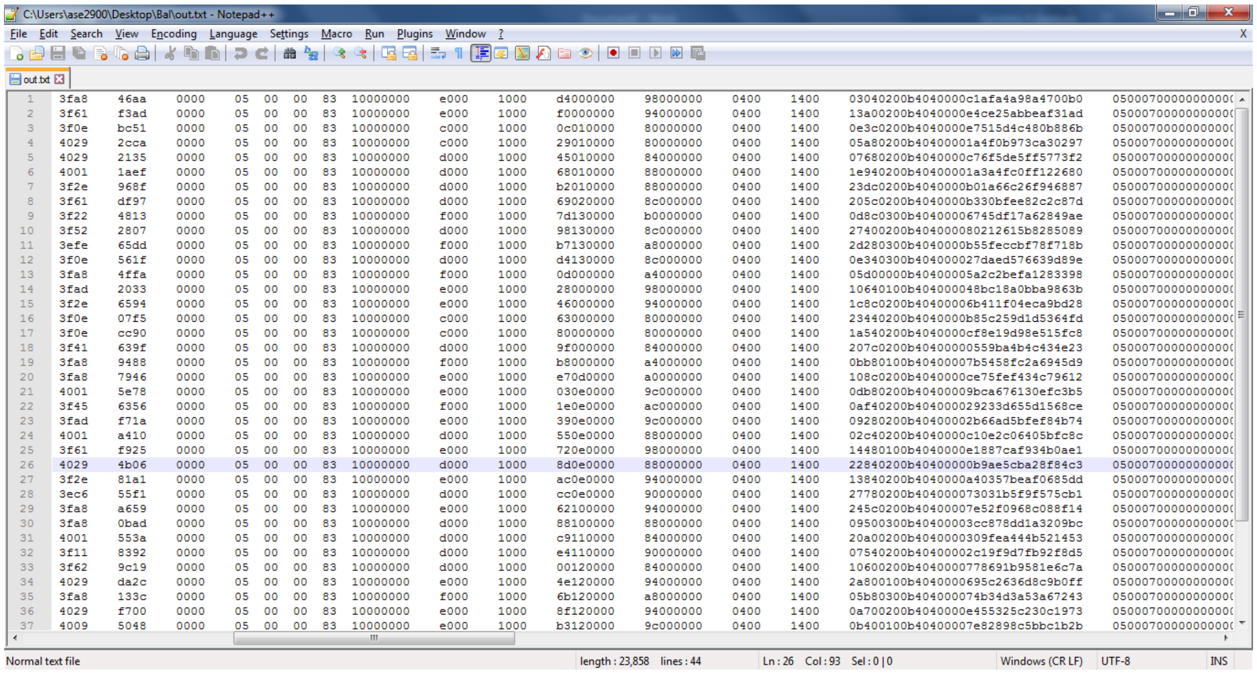


C:\Users\ase2900\Desktop\Bal\out.txt - Notepad++

```
4500  0108  09a5  40  00  80  06  5ae5  c0a80a0b  c0a80a0a  c05a  c003  b2814fc2  e10459df  5018  3fa8  46aa  0000  05  00  00  83  1
4500  0108  09d9  40  00  80  06  5ab1  c0a80a0b  c0a80a0a  c05a  c003  b2815d6c  e1049255  5018  3f61  f3ad  0000  05  00  00  83  1
4500  00e8  0a07  40  00  80  06  5aa3  c0a80a0b  c0a80a0a  c05a  c003  b2816d80  e104ca41  5018  3f0e  bc51  0000  05  00  00  83  1
4500  00e8  0a35  40  00  80  06  5a75  c0a80a0b  c0a80a0a  c05a  c003  b2817d74  e104ee3d  5018  4029  2cca  0000  05  00  00  83  1
4500  00f8  0a6b  40  00  80  06  5a2f  c0a80a0b  c0a80a0a  c05a  c003  b2818d68  e1051239  5018  4029  2135  0000  05  00  00  83  1
4500  00f8  0aa5  40  00  80  06  59f5  c0a80a0b  c0a80a0a  c05a  c003  b2819fac  e105a175  5018  4001  1aef  0000  05  00  00  83  1
4500  00f8  0b1c  40  00  80  06  597e  c0a80a0b  c0a80a0a  c05a  c003  b281cc44  e106c52d  5018  3f2e  968f  0000  05  00  00  83  1
4500  00f8  0c1e  40  00  80  06  587c  c0a80a0b  c0a80a0a  c05a  c003  b2822c6a  e10b8777  5018  3f61  df97  0000  05  00  00  83  1
4500  0118  238a  40  00  80  06  40f0  c0a80a0b  c0a80a0a  c0a0  c003  821b28b8  b8e398a7  5018  3f22  4813  0000  05  00  00  83  1
4500  00f8  23b5  40  00  80  06  40e5  c0a80a0b  c0a80a0a  c0a0  c003  821b385c  b8e3a453  5018  3f52  2807  0000  05  00  00  83  1
4500  0118  23f5  40  00  80  06  4085  c0a80a0b  c0a80a0a  c0a0  c003  821b4910  b8e4353f  5018  3efe  65dd  0000  05  00  00  83  1
4500  00f8  2424  40  00  80  06  4076  c0a80a0b  c0a80a0a  c0a0  c003  821b59f4  b8e45afb  5018  3f0e  561f  0000  05  00  00  83  1
4500  0118  07b2  40  00  80  06  5cc8  c0a80a0b  c0a80a0a  c042  c003  b673f13d  add66f3d  5018  3fa8  4ffa  0000  05  00  00  83  1
4500  0108  07e0  40  00  80  06  5caa  c0a80a0b  c0a80a0a  c042  c003  b67401b9  add67b91  5018  3fad  2033  0000  05  00  00  83  1
4500  0108  0816  40  00  80  06  5c74  c0a80a0b  c0a80a0a  c042  c003  b6741871  add711f9  5018  3f2e  6594  0000  05  00  00  83  1
4500  00e8  0847  40  00  80  06  5c63  c0a80a0b  c0a80a0a  c042  c003  b6742935  add79e25  5018  3f0e  07f5  0000  05  00  00  83  1
4500  00e8  0876  40  00  80  06  5c34  c0a80a0b  c0a80a0a  c042  c003  b67439c9  add7e541  5018  3f0e  cc90  0000  05  00  00  83  1
4500  00f8  08ac  40  00  80  06  5bee  c0a80a0b  c0a80a0a  c042  c003  b674479b  add8086f  5018  3f41  639f  0000  05  00  00  83  1
4500  0118  0959  40  00  80  06  5b21  c0a80a0b  c0a80a0a  c054  c003  d8e931cd  bbc11fa1  5018  3fa8  9488  0000  05  00  00  83  1
4500  0108  1b04  40  00  80  06  4986  c0a80a0b  c0a80a0a  c076  c003  747d8023  5843d4c3  5018  3fa8  7946  0000  05  00  00  83  1
4500  0108  1b38  40  00  80  06  4952  c0a80a0b  c0a80a0a  c076  c003  747d8d4d  5843dfc9  5018  4001  5e78  0000  05  00  00  83  1
4500  0118  1b6c  40  00  80  06  490e  c0a80a0b  c0a80a0a  c076  c003  747d97ff  5843e8d7  5018  3f45  6356  0000  05  00  00  83  1
4500  0108  1b98  40  00  80  06  48f2  c0a80a0b  c0a80a0a  c076  c003  747da603  5843f333  5018  3fad  f71a  0000  05  00  00  83  1
4500  00f8  1bc6  40  00  80  06  48d4  c0a80a0b  c0a80a0a  c076  c003  747db597  5843fedf  5018  4001  a410  0000  05  00  00  83  1
4500  0108  1bf9  40  00  80  06  4891  c0a80a0b  c0a80a0a  c076  c003  747dc2f9  584431dd  5018  3f61  f925  0000  05  00  00  83  1
4500  00f8  1c2a  40  00  80  06  4870  c0a80a0b  c0a80a0a  c076  c003  747dcdab  58443aeb  5018  4029  4b06  0000  05  00  00  83  1
4500  0108  1c5a  40  00  80  06  4830  c0a80a0b  c0a80a0a  c076  c003  747ddd8f  58445bf7  5018  3f2e  81a1  0000  05  00  00  83  1
4500  00f8  1c8c  40  00  80  06  480e  c0a80a0b  c0a80a0a  c076  c003  747defb3  58447f13  5018  3ec6  55f1  0000  05  00  00  83  1
4500  0108  1ec3  40  00  80  06  45c7  c0a80a0b  c0a80a0a  c08c  c003  8a763636  ac987861  5018  3fa8  a659  0000  05  00  00  83  1
4500  00f8  1f33  40  00  80  06  4567  c0a80a0b  c0a80a0a  c095  c003  35d0e5d1  69e2109d  5018  3fa8  0bad  0000  05  00  00  83  1
4500  00f8  20a2  40  00  80  06  43f8  c0a80a0b  c0a80a0a  c095  c003  35d1890d  69e472e1  5018  4001  553a  0000  05  00  00  83  1
4500  00f8  20d1  40  00  80  06  43c9  c0a80a0b  c0a80a0a  c095  c003  35d1954f  69e47d3f  5018  3f11  8392  0000  05  00  00  83  1
4500  00f8  2105  40  00  80  06  4395  c0a80a0b  c0a80a0a  c095  c003  35d1a333  69e4879b  5018  3f62  9c19  0000  05  00  00  83  1
4500  0108  2163  40  00  80  06  4327  c0a80a0b  c0a80a0a  c095  c003  35d1cc17  69e4f247  5018  4029  da2c  0000  05  00  00  83  1
4500  0118  21b0  40  00  80  06  42ca  c0a80a0b  c0a80a0a  c0a0  c003  821a8dac  b8e25e73  5018  3fa8  133c  0000  05  00  00  83  1
4500  0108  21ef  40  00  80  06  429b  c0a80a0b  c0a80a0a  c0a0  c003  821a9ee6  b8e29789  5018  4029  f700  0000  05  00  00  83  1
4500  0108  2230  40  00  80  06  425a  c0a80a0b  c0a80a0a  c0a0  c003  821ab3dc  b8e2b4f3  5018  4009  5048  0000  05  00  00  83  1
```

Normal text file   length : 23,858   lines : 44   Ln : 26   Col : 93   Sel : 0 | 0   Windows (CR LF)   UTF-8   INS

Figure 6. 3 – PCAP Log output



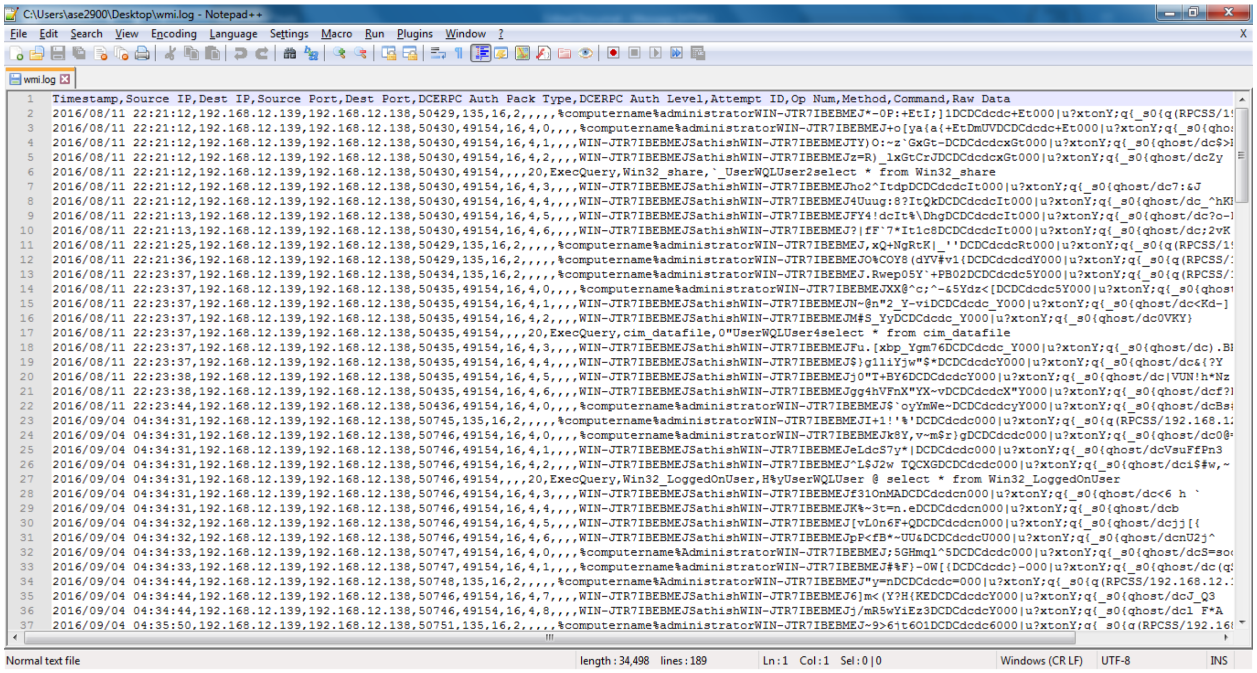
C:\Users\ase2900\Desktop\Bal\out.txt - Notepad++

```
3fa8  46aa  0000  05  00  00  83  10000000  e000  1000  d4000000  98000000  0400  1400  03040200b4040000c1afa4a98a4700b0  0500070000000000
3f61  f3ad  0000  05  00  00  83  10000000  e000  1000  f0000000  94000000  0400  1400  13a00200b4040000e4ce25abbeaf31ad  0500070000000000
3f0e  bc51  0000  05  00  00  83  10000000  c000  1000  0c010000  80000000  0400  1400  0e3c0200b4040000e7515d4c480b886b  0500070000000000
4029  2cca  0000  05  00  00  83  10000000  c000  1000  29010000  80000000  0400  1400  05a80200b40400001a4f0b973ca30297  0500070000000000
4029  2135  0000  05  00  00  83  10000000  d000  1000  45010000  84000000  0400  1400  07680200b4040000c76f5de5ff5773f2  0500070000000000
4001  1aef  0000  05  00  00  83  10000000  d000  1000  68010000  88000000  0400  1400  1e940200b40400001a3a4fc0ff122680  0500070000000000
3f2e  968f  0000  05  00  00  83  10000000  d000  1000  b2010000  88000000  0400  1400  23dc0200b4040000b01a66c26f946887  0500070000000000
3f61  df97  0000  05  00  00  83  10000000  d000  1000  69020000  8c000000  0400  1400  205c0200b4040000b330bfee82c2c87d  0500070000000000
3f22  4813  0000  05  00  00  83  10000000  f000  1000  7d130000  b0000000  0400  1400  0d8c0300b40400006745df17a62849ae  0500070000000000
3f52  2807  0000  05  00  00  83  10000000  d000  1000  98130000  8c000000  0400  1400  27400200b404000080212615b8285089  0500070000000000
3efe  65dd  0000  05  00  00  83  10000000  f000  1000  b7130000  a8000000  0400  1400  2d280300b4040000b55feccbf78f718b  0500070000000000
3f0e  561f  0000  05  00  00  83  10000000  d000  1000  d4130000  8c000000  0400  1400  0e340300b404000027daed576639d89e  0500070000000000
3fa8  4ffa  0000  05  00  00  83  10000000  f000  1000  0d000000  a4000000  0400  1400  05d00000b40400005a2c2befa1283398  0500070000000000
3fad  2033  0000  05  00  00  83  10000000  e000  1000  28000000  98000000  0400  1400  10640100b404000048bc18a0bba9863b  0500070000000000
3f2e  6594  0000  05  00  00  83  10000000  e000  1000  46000000  94000000  0400  1400  1c8c0200b40400006b411f04eca9bd28  0500070000000000
3f0e  07f5  0000  05  00  00  83  10000000  c000  1000  63000000  80000000  0400  1400  23440200b4040000b85c259d1d5364fd  0500070000000000
3f0e  cc90  0000  05  00  00  83  10000000  c000  1000  80000000  80000000  0400  1400  1a540200b4040000cf8e19d98e515fc8  0500070000000000
3f41  639f  0000  05  00  00  83  10000000  d000  1000  9f000000  84000000  0400  1400  207c0200b40400000559ba4b4c434e23  0500070000000000
3fa8  9488  0000  05  00  00  83  10000000  f000  1000  b8000000  a4000000  0400  1400  0bb80100b40400007b5458fc2a6945d9  0500070000000000
3fa8  7946  0000  05  00  00  83  10000000  e000  1000  e70d0000  a0000000  0400  1400  108c0200b4040000ce75fef434c79612  0500070000000000
4001  5e78  0000  05  00  00  83  10000000  e000  1000  030e0000  9c000000  0400  1400  0db80200b40400009bca676130efc3b5  0500070000000000
3f45  6356  0000  05  00  00  83  10000000  f000  1000  1e0e0000  ac000000  0400  1400  0af40200b404000029233d655d1568ce  0500070000000000
3fad  f71a  0000  05  00  00  83  10000000  e000  1000  390e0000  9c000000  0400  1400  09280200b40400002b66ad5bfef84b74  0500070000000000
4001  a410  0000  05  00  00  83  10000000  d000  1000  550e0000  88000000  0400  1400  02c40200b4040000c10e2c06405bfc8c  0500070000000000
3f61  f925  0000  05  00  00  83  10000000  e000  1000  720e0000  98000000  0400  1400  14480100b4040000e1887caf934b0ae1  0500070000000000
4029  4b06  0000  05  00  00  83  10000000  d000  1000  8d0e0000  88000000  0400  1400  22840200b40400000b9ae5cba28f84c3  0500070000000000
3f2e  81a1  0000  05  00  00  83  10000000  e000  1000  ac0e0000  94000000  0400  1400  13840200b4040000a40357beaf0685dd  0500070000000000
3ec6  55f1  0000  05  00  00  83  10000000  d000  1000  cc0e0000  90000000  0400  1400  27780200b404000073031b5f9f575cb1  0500070000000000
3fa8  a659  0000  05  00  00  83  10000000  e000  1000  62100000  94000000  0400  1400  245c0200b40400007e52f0968c088f14  0500070000000000
3fa8  0bad  0000  05  00  00  83  10000000  d000  1000  88100000  88000000  0400  1400  09500300b40400003cc878dd1a3209bc  0500070000000000
4001  553a  0000  05  00  00  83  10000000  d000  1000  c9110000  84000000  0400  1400  20a00200b404000309fea444b521453  0500070000000000
3f11  8392  0000  05  00  00  83  10000000  d000  1000  e4110000  90000000  0400  1400  07540200b40400002c19f9d7fb92f8d5  0500070000000000
3f62  9c19  0000  05  00  00  83  10000000  d000  1000  00120000  84000000  0400  1400  10600200b404000778691b9581e6c7a  0500070000000000
4029  da2c  0000  05  00  00  83  10000000  e000  1000  4e120000  94000000  0400  1400  2a800100b404000695c2636d8c9b0ff  0500070000000000
3fa8  133c  0000  05  00  00  83  10000000  f000  1000  6b120000  a8000000  0400  1400  05b80300b404000074b34d3a53a67243  0500070000000000
4029  f700  0000  05  00  00  83  10000000  e000  1000  8f120000  94000000  0400  1400  0a700200b4040000e455325c230c1973  0500070000000000
4009  5048  0000  05  00  00  83  10000000  e000  1000  b3120000  9c000000  0400  1400  0b400100b40400007e82898c5bbc1b2b  0500070000000000
```

Normal text file   length : 23,858   lines : 44   Ln : 26   Col : 93   Sel : 0 | 0   Windows (CR LF)   UTF-8   INS

Figure 6. 4 – PCAP Log output

6.5 WMI activity log output from the tool

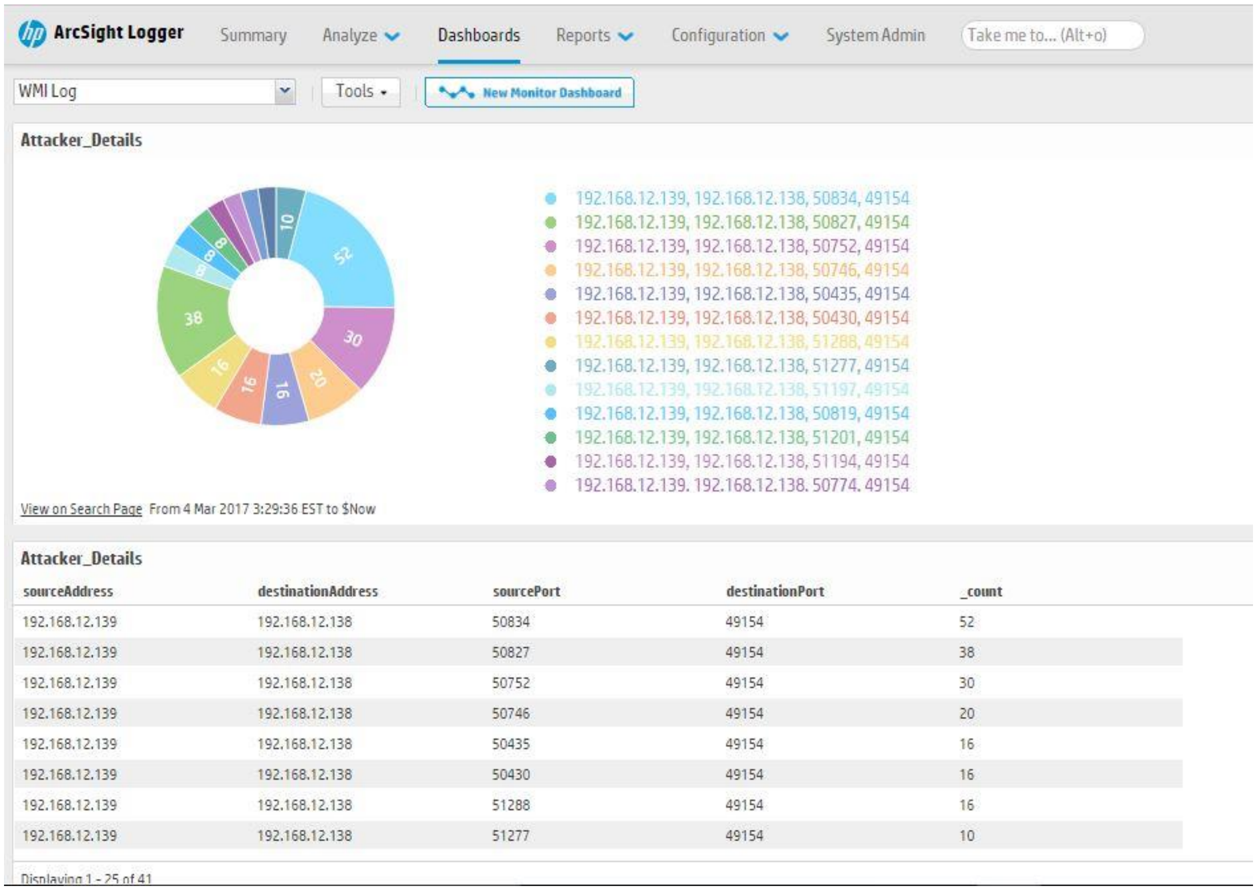

C:\Users\ase2900\Desktop\wmi.log - Notepad++

```
Timestamp,Source IP,Dest IP,Source Port,Dest Port,DCERPC Auth Pack Type,DCERPC Auth Level,Attempt ID,Op Num,Method,Command,Raw Data
2016/08/11 22:21:12,192.168.12.139,192.168.12.138,50429,135,16,2,,,,,%computername%administratorWIN-JTR7IBEBMEJ*-0P:+EtI;]1DCDCdcdc+Et000|u?xtonY;q{_s0{q(RPCSS/1
2016/08/11 22:21:12,192.168.12.139,192.168.12.138,50430,49154,16,4,0,,,,%computername%administratorWIN-JTR7IBEBMEJ+o[ya{a{+EtDmUVDCDCdcdc+Et000|u?xtonY;q{_s0{qho
2016/08/11 22:21:12,192.168.12.139,192.168.12.138,50430,49154,16,4,1,,,,WIN-JTR7IBEBMEJSathishWIN-JTR7IBEBMEJTY)O:~z`GxGt-DCDCdcdcxGt000|u?xtonY;q{_s0{qhost/dc$>
2016/08/11 22:21:12,192.168.12.139,192.168.12.138,50430,49154,16,4,2,,,,WIN-JTR7IBEBMEJSathishWIN-JTR7IBEBMEJz=R)_lxGtCrJDCDCdcdcxGt000|u?xtonY;q{_s0{qhost/dcZy
2016/08/11 22:21:12,192.168.12.139,192.168.12.138,50430,49154,,,,20,ExecQuery,Win32_share,`_UserWQLUser2select * from Win32_share
2016/08/11 22:21:12,192.168.12.139,192.168.12.138,50430,49154,16,4,3,,,,WIN-JTR7IBEBMEJSathishWIN-JTR7IBEBMEJho2^ItdpDCDCdcdcIt000|u?xtonY;q{_s0{qhost/dc7:&J
2016/08/11 22:21:12,192.168.12.139,192.168.12.138,50430,49154,16,4,4,,,,WIN-JTR7IBEBMEJSathishWIN-JTR7IBEBMEJ4Uuug:8?ItQkDCDCdcdcIt000|u?xtonY;q{_s0{qhost/dc_^hK
2016/08/11 22:21:13,192.168.12.139,192.168.12.138,50430,49154,16,4,5,,,,WIN-JTR7IBEBMEJSathishWIN-JTR7IBEBMEJFY4!dcIt%\DhgDCDCdcdcIt000|u?xtonY;q{_s0{qhost/dc?o-
2016/08/11 22:21:13,192.168.12.139,192.168.12.138,50430,49154,16,4,6,,,,WIN-JTR7IBEBMEJSathishWIN-JTR7IBEBMEJ?|fF`7*It1c8DCDCdcdcIt000|u?xtonY;q{_s0{qhost/dc;2vK
2016/08/11 22:21:25,192.168.12.139,192.168.12.138,50429,135,16,2,,,,,%computername%administratorWIN-JTR7IBEBMEJ,xQ+NgRtK|_''DCDCdcdcRt000|u?xtonY;q{_s0{q(RPCSS/1
2016/08/11 22:21:36,192.168.12.139,192.168.12.138,50429,135,16,2,,,,,%computername%administratorWIN-JTR7IBEBMEJO%COY8(dYV#v1{DCDCdcdcdY000|u?xtonY;q{_s0{q(RPCSS/
2016/08/11 22:23:37,192.168.12.139,192.168.12.138,50434,135,16,2,,,,,%computername%administratorWIN-JTR7IBEBMEJ.Rwep05Y`+PB02DCDCdcdc5Y000|u?xtonY;q{_s0{q(RPCSS/
2016/08/11 22:23:37,192.168.12.139,192.168.12.138,50435,49154,16,4,0,,,,%computername%administratorWIN-JTR7IBEBMEJXX@^c;^-&5Ydz<[DCDCdcdc5Y000|u?xtonY;q{_s0{qhos
2016/08/11 22:23:37,192.168.12.139,192.168.12.138,50435,49154,16,4,1,,,,WIN-JTR7IBEBMEJSathishWIN-JTR7IBEBMEJN~@n"2_Y-viDCDCdcdc_Y000|u?xtonY;q{_s0{qhost/dc<Kd-]
2016/08/11 22:23:37,192.168.12.139,192.168.12.138,50435,49154,16,4,2,,,,WIN-JTR7IBEBMEJSathishWIN-JTR7IBEBMEJM#S_YyDCDCdcdc_Y000|u?xtonY;q{_s0{qhost/dc0VKY}
2016/08/11 22:23:37,192.168.12.139,192.168.12.138,50435,49154,,,,20,ExecQuery,cim_datafile,0"UserWQLUser4select * from cim_datafile
2016/08/11 22:23:37,192.168.12.139,192.168.12.138,50435,49154,16,4,3,,,,WIN-JTR7IBEBMEJSathishWIN-JTR7IBEBMEJFu.[xbp_Ygm76DCDCdcdc_Y000|u?xtonY;q{_s0{qhost/dc).B
2016/08/11 22:23:37,192.168.12.139,192.168.12.138,50435,49154,16,4,4,,,,WIN-JTR7IBEBMEJSathishWIN-JTR7IBEBMEJ$}g1liYjw"$*DCDCdcdcY000|u?xtonY;q{_s0{qhost/dc&{?Y
2016/08/11 22:23:38,192.168.12.139,192.168.12.138,50435,49154,16,4,5,,,,WIN-JTR7IBEBMEJSathishWIN-JTR7IBEBMEJj0"T+BY6DCDCdcdcY000|u?xtonY;q{_s0{qhost/dc|VUN!h*Nz
2016/08/11 22:23:38,192.168.12.139,192.168.12.138,50435,49154,16,4,6,,,,WIN-JTR7IBEBMEJSathishWIN-JTR7IBEBMEJgg4hVFnX"YX~vDCDCdcdcX"Y000|u?xtonY;q{_s0{qhost/dcf?
2016/08/11 22:23:44,192.168.12.139,192.168.12.138,50436,49154,16,4,0,,,,%computername%administratorWIN-JTR7IBEBMEJ$`oyYmWe~DCDCdcdcyY000|u?xtonY;q{_s0{qhost/dcBs
2016/09/04 04:34:31,192.168.12.139,192.168.12.138,50745,135,16,2,,,,,%computername%administratorWIN-JTR7IBEBMEJI+1!'%'DCDCdcdc000|u?xtonY;q{_s0{q(RPCSS/192.168.1
2016/09/04 04:34:31,192.168.12.139,192.168.12.138,50746,49154,16,4,0,,,,%computername%administratorWIN-JTR7IBEBMEJk8Y,v~m$r}gDCDCdcdc000|u?xtonY;q{_s0{qhost/dc0@
2016/09/04 04:34:31,192.168.12.139,192.168.12.138,50746,49154,16,4,1,,,,WIN-JTR7IBEBMEJSathishWIN-JTR7IBEBMEJeLdcS7y*|DCDCdcdc000|u?xtonY;q{_s0{qhost/dcVsuFfPn3
2016/09/04 04:34:31,192.168.12.139,192.168.12.138,50746,49154,16,4,2,,,,WIN-JTR7IBEBMEJSathishWIN-JTR7IBEBMEJ^L$J2w TQCXGDCDCdcdc000|u?xtonY;q{_s0{qhost/dci$#w,~
2016/09/04 04:34:31,192.168.12.139,192.168.12.138,50746,49154,,,,20,ExecQuery,Win32_LoggedOnUser,H%yUserWQLUser @ select * from Win32_LoggedOnUser
2016/09/04 04:34:31,192.168.12.139,192.168.12.138,50746,49154,16,4,3,,,,WIN-JTR7IBEBMEJSathishWIN-JTR7IBEBMEJf31OnMADCDCdcdcn000|u?xtonY;q{_s0{qhost/dc<6 h `
2016/09/04 04:34:31,192.168.12.139,192.168.12.138,50746,49154,16,4,4,,,,WIN-JTR7IBEBMEJSathishWIN-JTR7IBEBMEJK%~3t=n.eDCDCdcdcn000|u?xtonY;q{_s0{qhost/dcb
2016/09/04 04:34:32,192.168.12.139,192.168.12.138,50746,49154,16,4,5,,,,WIN-JTR7IBEBMEJSathishWIN-JTR7IBEBMEJ[vL0n6F+QDCDCdcdcn000|u?xtonY;q{_s0{qhost/dcjj[{
2016/09/04 04:34:32,192.168.12.139,192.168.12.138,50746,49154,16,4,6,,,,WIN-JTR7IBEBMEJSathishWIN-JTR7IBEBMEJpP<fB*~UU&DCDCdcdcU000|u?xtonY;q{_s0{qhost/dcnU2j^
2016/09/04 04:34:33,192.168.12.139,192.168.12.138,50747,49154,16,4,0,,,,%computername%AdministratorWIN-JTR7IBEBMEJ;5GHmql^5DCDCdcdc000|u?xtonY;q{_s0{qhost/dcS=so
2016/09/04 04:34:33,192.168.12.139,192.168.12.138,50747,49154,16,4,1,,,,%computername%administratorWIN-JTR7IBEBMEJ#%F}-0W[{DCDCdcdc}-000|u?xtonY;q{_s0{qhost/dc(q
2016/09/04 04:34:44,192.168.12.139,192.168.12.138,50748,135,16,2,,,,,%computername%AdministratorWIN-JTR7IBEBMEJ"y=nDCDCdcdc=000|u?xtonY;q{_s0{q(RPCSS/192.168.12.
2016/09/04 04:34:44,192.168.12.139,192.168.12.138,50746,49154,16,4,7,,,,WIN-JTR7IBEBMEJSathishWIN-JTR7IBEBMEJ6]m<(Y?H{KEDCDCdcdcY000|u?xtonY;q{_s0{qhost/dcJ_Q3
2016/09/04 04:34:44,192.168.12.139,192.168.12.138,50746,49154,16,4,8,,,,WIN-JTR7IBEBMEJSathishWIN-JTR7IBEBMEJj/mR5wYiEz3DCDCdcdcY000|u?xtonY;q{_s0{qhost/dcl F*A
2016/09/04 04:35:50,192.168.12.139,192.168.12.138,50751,135,16,2,,,,,%computername%administratorWIN-JTR7IBEBMEJ~9>6jt6O1DCDCdcdc6000|u?xtonY;q{_s0{q(RPCSS/192.16
```

Normal text file   length : 34,498   lines : 189   Ln : 1   Col : 1   Sel : 0 | 0   Windows (CR LF)   UTF-8   INS

Figure 6. 5 – WMI Activity Log

6.6 Dashboard from the Arcsight logger on WMI activities



Attacker_Details

● 192.168.12.139, 192.168.12.138, 50834, 49154
● 192.168.12.139, 192.168.12.138, 50827, 49154
● 192.168.12.139, 192.168.12.138, 50752, 49154
● 192.168.12.139, 192.168.12.138, 50746, 49154
● 192.168.12.139, 192.168.12.138, 50435, 49154
● 192.168.12.139, 192.168.12.138, 50430, 49154
● 192.168.12.139, 192.168.12.138, 51288, 49154
● 192.168.12.139, 192.168.12.138, 51277, 49154
● 192.168.12.139, 192.168.12.138, 51197, 49154
● 192.168.12.139, 192.168.12.138, 50819, 49154
● 192.168.12.139, 192.168.12.138, 51201, 49154
● 192.168.12.139, 192.168.12.138, 51194, 49154
● 192.168.12.139, 192.168.12.138, 50774, 49154

View on Search Page From 4 Mar 2017 3:29:36 EST to $Now

Attacker_Details

sourceAddress	destinationAddress	sourcePort	destinationPort	_count
192.168.12.139	192.168.12.138	50834	49154	52
192.168.12.139	192.168.12.138	50827	49154	38
192.168.12.139	192.168.12.138	50752	49154	30
192.168.12.139	192.168.12.138	50746	49154	20
192.168.12.139	192.168.12.138	50435	49154	16
192.168.12.139	192.168.12.138	50430	49154	16
192.168.12.139	192.168.12.138	51288	49154	16
192.168.12.139	192.168.12.138	51277	49154	10

Displaying 1 - 25 of 41

Figure 6. 6 – Arcsight Logger Dashboard

Chapter 7 – Conclusion

Main goal of this thesis was to design and implement a framework capable of capturing WMI data packets and queries and analyzing them using a tool that has been created.

We tried to implement this with two scenarios which the first one is with an already available software which is Snort and in this evaluation, it was noticed that the capturing of WMI packets has a noticeable drop and analyzing the queries also contained limitations.

The second option is to use a custom developed tool which analyses the packets for the type of query and in this tool, it identifies the success or the failure of the authentication to the victim client It must be mentioned that in order for this sort of authentication to be successful it has to consider using other attacking techniques such as brute force attacks and social engineering techniques etc. This customized method can be easily modified and used in WMI packet analysis and the accuracy rate of evaluation is at the maximum rate.

The captured results on the log file can be integrated on a log management software solution and appropriate reports can be generated to specified customizations.

As far as the statistics and the evaluations techniques are concerned it is quite evident that the customized developed tool is much efficient and accurate in determining the subjected matter rather than already available tools in the market with its configurable tools.

7.1 Further Work

Although the solution proposed in the thesis proven to be efficient, some further work can be done to make it more persuasive and precise as listed below

- The tool can be better developed in order to analyze other activities which can be performed through the utilization of WMI where it can be even further detected for the purpose or the function the queries are utilized.

- Also, this solution can be better integrated with a fully functional log correlation system where real-time logging and alerting activities can be performed in a much more professional manner.

- The tool can also be integrated as a part of a fully Operating system management utility to perform automated tasks and make the administration functionalities more efficient and reliable.

7.2 Drawbacks / Limitation

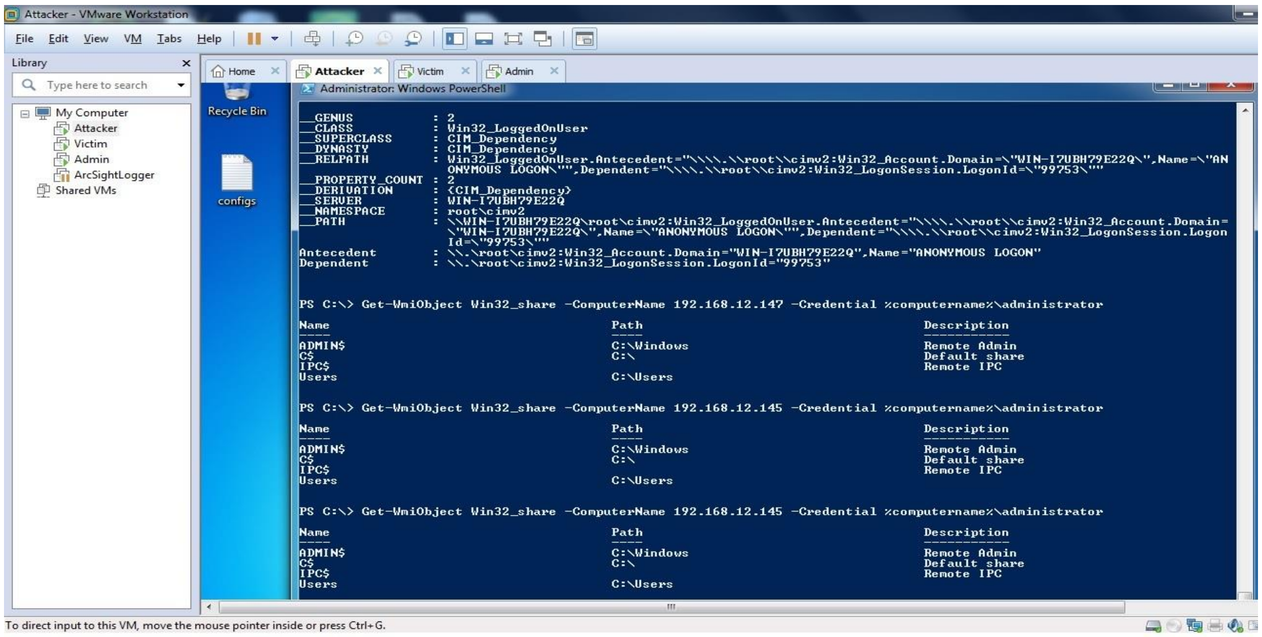
The biggest challenging reason being this project was the limitation of information related to the subject matter even though this has existed since the inception of windows.

However, the subject was only taken for a noticeable amount of magnitude only in 2010 but subsequently had faded away within the same period.

Even though it was taken up for discussion in 2015 at some of the security forums currently only there is a noticeable amount of discussion related to the subject and it was a tedious process to gather knowledge and arrive at solid conclusions relating to theories. [14]

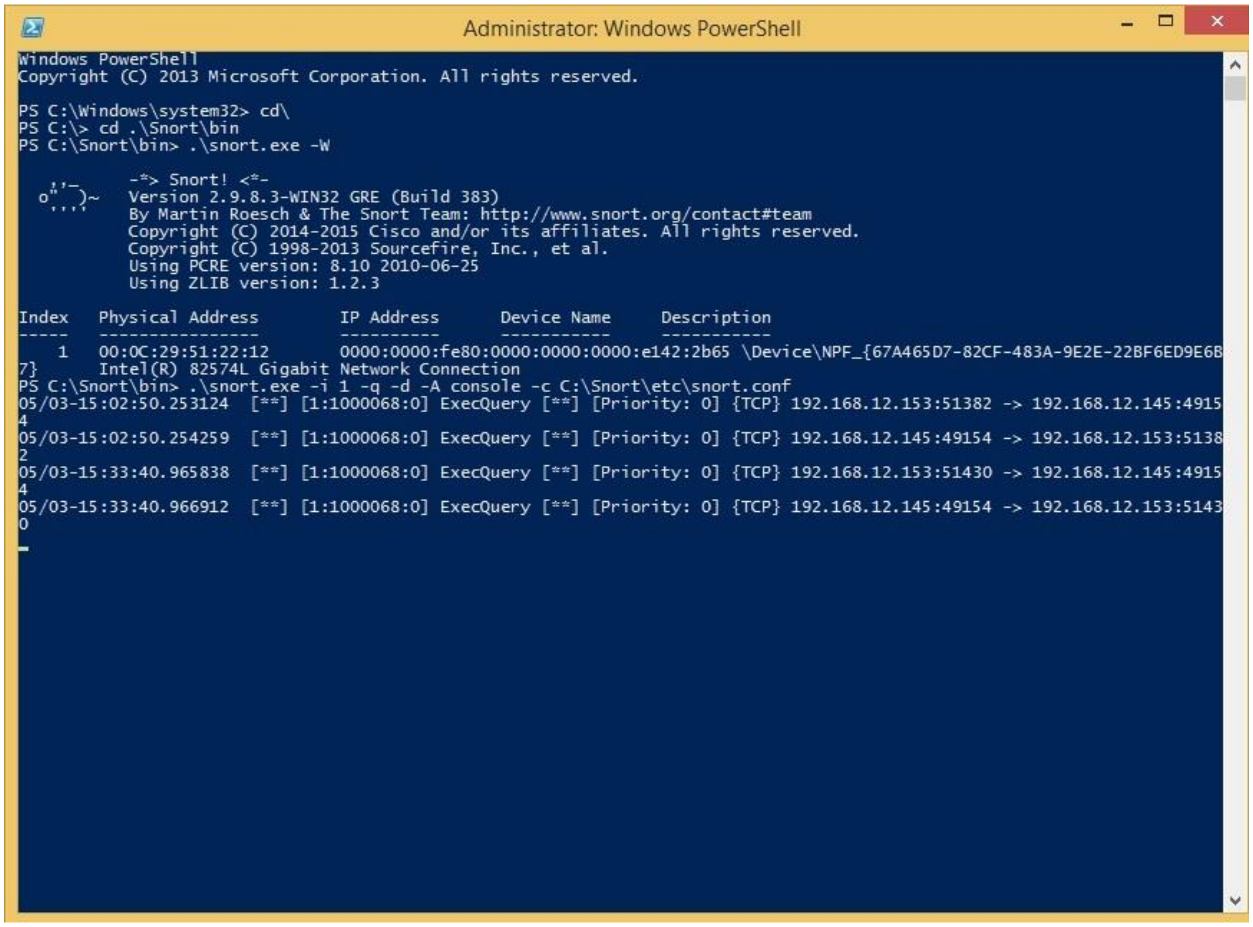
It was also a hampering factor of limitation the currently available tools inability to perform consistently with the WMI packets and the analysis factor of the project become a more challenging factor than expected.

The question of Why this cannot be performed with the already available tools was clarified through evaluated by carrying out POC's through well know products in order to arrive at the development of a new tool to fulfil the requirement (SNORT).


Attacker - VMware Workstation
File Edit View VM Tabs Help
Library
Type here to search
My Computer
Attacker
Victim
Admin
ArcSightLogger
Shared VMs
Home
Attacker
Victim
Admin
Recycle Bin
configs
Administrator: Windows PowerShell
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

```
__GENUS            : 2
__CLASS            : Win32_LoggedOnUser
__SUPERCLASS       : CIM_Dependency
__DYNASTY          : CIM_Dependency
__RELPATH          : Win32_LoggedOnUser.Antecedent="\\\\.\\root\\cimv2:Win32_Account.Domain=\"WIN-I7UBH79E22Q\",Name=\"AN
                     ONYMOUS LOGON\"",Dependent="\\\\.\\root\\cimv2:Win32_LogonSession.LogonId=\"99753\""
__PROPERTY_COUNT   : 2
__DERIVATION       : {CIM_Dependency}
__SERVER           : WIN-I7UBH79E22Q
__NAMESPACE        : root\cimv2
__PATH             : \\WIN-I7UBH79E22Q\root\cimv2:Win32_LoggedOnUser.Antecedent="\\\\.\\root\\cimv2:Win32_Account.Domain=
                     \"WIN-I7UBH79E22Q\",Name=\"ANONYMOUS LOGON\"",Dependent="\\\\.\\root\\cimv2:Win32_LogonSession.Logon
                     Id=\"99753\""
Antecedent         : \\.\root\cimv2:Win32_Account.Domain="WIN-I7UBH79E22Q",Name="ANONYMOUS LOGON"
Dependent          : \\.\root\cimv2:Win32_LogonSession.LogonId="99753"


PS C:\> Get-WmiObject Win32_share -ComputerName 192.168.12.147 -Credential %computername%\administrator

Name                                    Path                                    Description
----                                    ----                                    -----------
ADMIN$                                  C:\Windows                              Remote Admin
C$                                      C:\                                     Default share
IPC$                                                                            Remote IPC
Users                                   C:\Users

PS C:\> Get-WmiObject Win32_share -ComputerName 192.168.12.145 -Credential %computername%\administrator

Name                                    Path                                    Description
----                                    ----                                    -----------
ADMIN$                                  C:\Windows                              Remote Admin
C$                                      C:\                                     Default share
IPC$                                                                            Remote IPC
Users                                   C:\Users

PS C:\> Get-WmiObject Win32_share -ComputerName 192.168.12.145 -Credential %computername%\administrator

Name                                    Path                                    Description
----                                    ----                                    -----------
ADMIN$                                  C:\Windows                              Remote Admin
C$                                      C:\                                     Default share
IPC$                                                                            Remote IPC
Users                                   C:\Users
```

Figure 5. 18 - Output of the SNORT attack



```
Windows PowerShell
Copyright (C) 2013 Microsoft Corporation. All rights reserved.

PS C:\Windows\system32> cd\
PS C:\> cd .\Snort\bin
PS C:\Snort\bin> .\snort.exe -W

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.8.3-WIN32 GRE (Build 383)
   ''''    By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
           Copyright (C) 2014-2015 Cisco and/or its affiliates. All rights reserved.
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using PCRE version: 8.10 2010-06-25
           Using ZLIB version: 1.2.3

Index   Physical Address        IP Address      Device Name     Description
-----   ----------------        ----------      -----------     -----------
    1   00:0C:29:51:22:12       0000:0000:fe80:0000:0000:0000:e142:2b65 \Device\NPF_{67A465D7-82CF-483A-9E2E-22BF6ED9E6B
7}      Intel(R) 82574L Gigabit Network Connection
PS C:\Snort\bin> .\snort.exe -i 1 -q -d -A console -c C:\Snort\etc\snort.conf
05/03-15:02:50.253124  [**] [1:1000068:0] ExecQuery [**] [Priority: 0] {TCP} 192.168.12.153:51382 -> 192.168.12.145:4915
4
05/03-15:02:50.254259  [**] [1:1000068:0] ExecQuery [**] [Priority: 0] {TCP} 192.168.12.145:49154 -> 192.168.12.153:5138
2
05/03-15:33:40.965838  [**] [1:1000068:0] ExecQuery [**] [Priority: 0] {TCP} 192.168.12.153:51430 -> 192.168.12.145:4915
4
05/03-15:33:40.966912  [**] [1:1000068:0] ExecQuery [**] [Priority: 0] {TCP} 192.168.12.145:49154 -> 192.168.12.153:5143
0
```

Figure 5. 19 -SNORT packet capture output

Appendix I

Timestamp,Source IP,Dest IP,Source Port,Dest Port,DCERPC Auth Pack Type,DCERPC Auth Level,Attempt ID,Op Num,Method,Command,Raw Data

```
Timestamp,Source IP,Dest IP,Source Port,Dest Port,DCERPC Auth Pack Type,DCERPC Auth
Level,Attempt ID,Op Num,Method,Command,Raw Data

2016/08/11
22:21:12,192.168.12.139,192.168.12.138,50429,135,16,2,,,,,%computername%administratorWI
N-JTR7IBEBMEJ*-
0P:+EtI;]1DCDCdcdc+Et000|u?xtonY;q{_s0{q(RPCSS/192.168.12.138|h8qu

2016/08/11
22:21:12,192.168.12.139,192.168.12.138,50430,49154,16,4,0,,,,%computername%administrator
WIN-JTR7IBEBMEJ+o[ya{a{+EtDmUVDCDCdcdc+Et000|u?xtonY;q{_s0{qhost/dc=)n|OR,f\

2016/08/11 22:21:12,192.168.12.139,192.168.12.138,50430,49154,16,4,1,,,,WIN-
JTR7IBEBMEJSathishWIN-JTR7IBEBMEJTY)O:~z`GxGt-
DCDCdcdcxGt000|u?xtonY;q{_s0{qhost/dc$>K

2016/08/11 22:21:12,192.168.12.139,192.168.12.138,50430,49154,16,4,2,,,,WIN-
JTR7IBEBMEJSathishWIN-
JTR7IBEBMEJz=R)_lxGtCrJDCDCdcdcxGt000|u?xtonY;q{_s0{qhost/dcZy

2016/08/11
22:21:12,192.168.12.139,192.168.12.138,50430,49154,,,,20,ExecQuery,Win32_share,`_UserWQ
LUser2select * from Win32_share

2016/08/11 22:21:12,192.168.12.139,192.168.12.138,50430,49154,16,4,3,,,,WIN-
JTR7IBEBMEJSathishWIN-
JTR7IBEBMEJho2^ItdpDCDCdcdcIt000|u?xtonY;q{_s0{qhost/dc7:&J

2016/08/11 22:21:12,192.168.12.139,192.168.12.138,50430,49154,16,4,4,,,,WIN-
JTR7IBEBMEJSathishWIN-
JTR7IBEBMEJ4Uuug:8?ItQkDCDCdcdcIt000|u?xtonY;q{_s0{qhost/dc_^hKBk9

2016/08/11 22:21:13,192.168.12.139,192.168.12.138,50430,49154,16,4,5,,,,WIN-
JTR7IBEBMEJSathishWIN-
JTR7IBEBMEJFY4!dcIt%\DhgDCDCdcdcIt000|u?xtonY;q{_s0{qhost/dc?o-k"

2016/08/11 22:21:13,192.168.12.139,192.168.12.138,50430,49154,16,4,6,,,,WIN-
JTR7IBEBMEJSathishWIN-
JTR7IBEBMEJ?|fF`7*It1c8DCDCdcdcIt000|u?xtonY;q{_s0{qhost/dc;2vK

2016/08/11
22:21:25,192.168.12.139,192.168.12.138,50429,135,16,2,,,,,%computername%administratorWI
N-
JTR7IBEBMEJ,xQ+NgRtK|_"DCDCdcdcRt000|u?xtonY;q{_s0{q(RPCSS/192.168.12.138\QLT
PH]Cv
```

```
2016/08/11
22:21:36,192.168.12.139,192.168.12.138,50429,135,16,2,,,,,%computername%administratorWI
N-
JTR7IBEBMEJO%COY8(dYV#v1{DCDCdcdcdY000|u?xtonY;q{_s0{q(RPCSS/192.168.12.13
8|Sn0

2016/08/11
22:23:37,192.168.12.139,192.168.12.138,50434,135,16,2,,,,,%computername%administratorWI
N-
JTR7IBEBMEJ.Rwep05Y`+PB02DCDCdcdc5Y000|u?xtonY;q{_s0{q(RPCSS/192.168.12.138P,
N?]C

2016/08/11
22:23:37,192.168.12.139,192.168.12.138,50435,49154,16,4,0,,,,%computername%administrator
WIN-JTR7IBEBMEJXX@^c;^-&5Ydz<[DCDCdcdc5Y000|u?xtonY;q{_s0{qhost/dcspB,

2016/08/11                    22:23:37,192.168.12.139,192.168.12.138,50435,49154,16,4,1,,,,WIN-
JTR7IBEBMEJSathishWIN-JTR7IBEBMEJN~@n"2_Y-
viDCDCdcdc_Y000|u?xtonY;q{_s0{qhost/dc<Kd-]

2016/08/11                    22:23:37,192.168.12.139,192.168.12.138,50435,49154,16,4,2,,,,WIN-
JTR7IBEBMEJSathishWIN-
JTR7IBEBMEJM#S_YyDCDCdcdc_Y000|u?xtonY;q{_s0{qhost/dc0VKY}

2016/08/11
22:23:37,192.168.12.139,192.168.12.138,50435,49154,,,,20,ExecQuery,cim_datafile,0"UserWQ
LUser4select * from cim_datafile

2016/08/11                    22:23:37,192.168.12.139,192.168.12.138,50435,49154,16,4,3,,,,WIN-
JTR7IBEBMEJSathishWIN-
JTR7IBEBMEJFu.[xbp_Ygm76DCDCdcdc_Y000|u?xtonY;q{_s0{qhost/dc).BR!}

2016/08/11                    22:23:37,192.168.12.139,192.168.12.138,50435,49154,16,4,4,,,,WIN-
JTR7IBEBMEJSathishWIN-
JTR7IBEBMEJ$}g1liYjw"$*DCDCdcdcY000|u?xtonY;q{_s0{qhost/dc&{?Y

2016/08/11                    22:23:38,192.168.12.139,192.168.12.138,50435,49154,16,4,5,,,,WIN-
JTR7IBEBMEJSathishWIN-
JTR7IBEBMEJj0"T+BY6DCDCdcdcY000|u?xtonY;q{_s0{qhost/dc|VUN!h*Nz

2016/08/11                    22:23:38,192.168.12.139,192.168.12.138,50435,49154,16,4,6,,,,WIN-
JTR7IBEBMEJSathishWIN-
JTR7IBEBMEJgg4hVFnX"YX~vDCDCdcdcX"Y000|u?xtonY;q{_s0{qhost/dcf?P|W

2016/08/11
22:23:44,192.168.12.139,192.168.12.138,50436,49154,16,4,0,,,,%computername%administrator
WIN-JTR7IBEBMEJ$`oyYmWe~DCDCdcdcyY000|u?xtonY;q{_s0{qhost/dcBs#0B,
```

Appendix II

```
#ifdef _MSC_VER
/*
* we do not want the warnings about the old deprecated and unsecure CRT functions
* since these examples can be compiled under *nix as well
*/
#define _CRT_SECURE_NO_WARNINGS
#endif


#include <stdlib.h>
#include <signal.h>
#include <tchar.h>
//#include <cstdlib>
#include <cstdio>
//#include <stdio.h>
#include <iostream>

#include <pcap.h>

#pragma comment(lib,"ws2_32.lib") //For winsock

#define LINE_LEN 16
#define ETH_ALEN 6                 /* Octets in one ethernet addr   */
#define BUF_SIZE 65536
#define TIME_BUF_SIZE 20

char alrt[42];                     // holds attempt message
char id[4];                 // holds attempt id
char command[50];           // holds excuting commad

char time_buf[TIME_BUF_SIZE];      // holds timestamp

char *sour, *dest;          // source and destination ip addresses as strings

struct sockaddr_in addrs;   // hold soruce and destination ip address

time_t curtime;                    // time variables
struct tm *loctime;         // time variables

FILE *log_fd;                   // log file descriptor

// for decrypted data
char *data;

const char *opnum_methods[] = { "", "", "", \
"OpenNamespace", \
"CancelAsyncCall", \
"QueryObjectSink", \
"GetObject", \
"GetObjectAsync", \
"PutClass", \
"PutClassAsync", \
"DeleteClass", \
"DeleteClassAsync", \
"CreateClassEnum", \
"CreateClassEnumAsync", \
"PutInstance", \
"PutInstanceAsync", \
"DeleteInstance", \
"DeleteInstanceAsync", \
"CreateInstanceEnum", \
"CreateInstanceEnumAsync", \
"ExecQuery", \
"ExecQueryAsync", \
"ExecNotificationQuery", \
"ExecNotificationQueryAsync", \
"ExecMethod", \
"ExecMethodAsync" };

typedef struct iphdr
{
#if __BYTE_ORDER == __LITTLE_ENDIAN
       unsigned char ihl : 4;
       unsigned char version : 4;
#elif __BYTE_ORDER == __BIG_ENDIAN
       unsigned int version : 4;
       unsigned int ihl : 4;
#else
# error "Please fix <bits/endian.h>"
#endif
       u_int8_t tos;
       u_int16_t tot_len;
       u_int16_t id;
       u_int16_t frag_off;
       u_int8_t ttl;
       u_int8_t protocol;
       u_int16_t check;
       u_int32_t saddr;
       u_int32_t daddr;
       /*The options start here. */
};

typedef u_int32_t tcp_seq;
typedef struct tcphdr
{
       union
       {
              struct
              {
                     u_int16_t th_sport;          /* source port */
                     u_int16_t th_dport;          /* destination port */
                     tcp_seq th_seq;                          /* sequence number */
                     tcp_seq th_ack;                          /* acknowledgement
number */
# if __BYTE_ORDER == __LITTLE_ENDIAN
                     u_int8_t th_x2 : 4;                  /* (unused) */
                     u_int8_t th_off : 4;         /* data offset */
# elif __BYTE_ORDER == __BIG_ENDIAN
                     u_int8_t th_off : 4;         /* data offset */
                     u_int8_t th_x2 : 4;          /* (unused) */
# endif
                     u_int8_t th_flags;

# define TH_FIN 0x01
# define TH_SYN 0x02
# define TH_RST 0x04
# define TH_PUSH        0x08
# define TH_ACK 0x10
# define TH_URG 0x20
                     u_int16_t th_win;                /* window */
                     u_int16_t th_sum;                /* checksum */
                     u_int16_t th_urp;                /* urgent pointer */
              };
              struct
              {
                     u_int16_t source;
                     u_int16_t dest;
                     u_int32_t seq;
                     u_int32_t ack_seq;
# if __BYTE_ORDER == __LITTLE_ENDIAN
                     u_int16_t res1 : 4;
                     u_int16_t doff : 4;
                     u_int16_t fin : 1;
                     u_int16_t syn : 1;
                     u_int16_t rst : 1;
                     u_int16_t psh : 1;
                     u_int16_t ack : 1;
                     u_int16_t urg : 1;
                     u_int16_t res2 : 2;
# elif __BYTE_ORDER == __BIG_ENDIAN
                     u_int16_t doff : 4;
                     u_int16_t res1 : 4;
                     u_int16_t res2 : 2;
                     u_int16_t urg : 1;
                     u_int16_t ack : 1;
                     u_int16_t psh : 1;
                     u_int16_t rst : 1;
                     u_int16_t syn : 1;
                     u_int16_t fin : 1;
# else
#  error "Adjust your <bits/endian.h> defines"
# endif
                     u_int16_t window;
                     u_int16_t check;
                     u_int16_t urg_ptr;
              };
       };
};

typedef struct ethhdr {
       unsigned char   h_dest[ETH_ALEN];       /* destination eth addr */
       unsigned char   h_source[ETH_ALEN];     /* source ether addr    */
       unsigned short h_proto;                              /* packet type ID field */
};

struct dcerpc_hdr{
       u_int8_t version;
       u_int8_t version_minor;
       u_int8_t pack_type;
       u_int8_t pack_flag;

       u_int32_t data_rep;

       u_int16_t frag_len;
       u_int16_t auth_len;
       u_int32_t call_id;
```

```
        u_int32_t alloc_hint;
        u_int16_t context_id;
        u_int16_t op_num;

        unsigned char obj_uuid[16];
};

struct dcerpc_trl{
        u_int8_t auth_type;
        u_int8_t auth_level;
        u_int8_t auth_pad_len;
        u_int8_t auth_rsvrd;
        u_int32_t auth_context_id;

        u_int32_t ntlmssp_ver_num;
        unsigned char ntlmssp_ver_body[12];
};

struct dcerpc_auth{
        u_int8_t version;
        u_int8_t version_minor;
        u_int8_t pack_type;
        u_int8_t pack_flag;

        u_int32_t data_rep;

        u_int16_t frag_len;
        u_int16_t auth_len;
        u_int32_t call_id;

        u_int32_t unknown;

        u_int8_t auth_type;
        u_int8_t auth_level;
        u_int8_t auth_pad_len;
        u_int8_t auth_rsvrd;
        u_int32_t auth_context_id;

};

struct ntlmssp{
        u_int64_t identifier;
        u_int32_t message_type;

        u_int16_t lan_man_res_length;
        u_int16_t lan_man_res_maxlen;
        u_int32_t lan_man_res_offset;

        u_int16_t ntlm_res_length;
        u_int16_t ntlm_res_maxlen;
        u_int32_t ntlm_res_offset;


        u_int16_t domain_name_length;
        u_int16_t domain_name_maxlen;
        u_int32_t domain_name_offset;

        u_int16_t user_name_length;
        u_int16_t user_name_maxlen;
        u_int32_t user_name_offset;
        u_int16_t host_name_length;
        u_int16_t host_name_maxlen;
        u_int32_t host_name_offset;

        u_int16_t session_key_length;
        u_int16_t session_key_maxlen;
        u_int32_t session_key_offset;

        u_int32_t negotiate_flags;

        union
        {
                u_int64_t verson;

                struct {
                        u_int8_t version_major_version;
                        u_int8_t version_minor_version;
                        u_int16_t version_build_number;

                        unsigned char version_pad[3];
                        u_int8_t  version_ntlm_cur_rev;
                };
        };

        unsigned char MIC[16];

};

#pragma pack(2)
struct dcerpc_pack{
        struct ethhdr eth;
        struct iphdr ip;
        struct tcphdr tcp;
        struct dcerpc_hdr dcerpc_h;
};

#pragma pack(2)
struct dcerpc_pack_auth{
        struct ethhdr eth;
        struct iphdr ip;
        struct tcphdr tcp;
        struct dcerpc_auth auth;
        struct ntlmssp ntlmssp_h;
};

void process(const u_char *, size_t);
void process_data(const u_char *, size_t, int, int);
void process_auth(const u_char *, size_t, int, int);
void alert(const u_char *, char *);
void alert_ex(const u_char *, char *);
void log_ex(const u_char *);
void log_auth(const u_char *);
void eth_print(const u_char *);
void SignalHandler(int);

int main(int argc, char **argv)
{
        pcap_if_t *alldevs, *d;
        pcap_t *fp;
```

```
	u_int inum, i = 0;
	char errbuf[PCAP_ERRBUF_SIZE];
	int res;
	struct pcap_pkthdr *header;
	const u_char *pkt_data;

	// create log file or point the log file descriptor to log file
	if (!(log_fd = fopen("wmi.log", "a+"))){
		fprintf(stderr, "Error : Openning Log File : %s\n", errbuf);
		exit(1);
	}

	// checks if log file is empty
	fseek(log_fd, 0, SEEK_END);
	// if log file is empty write the login header
	if (ftell(log_fd) == 0){
		fprintf(log_fd, "Timestamp,Source IP,Dest IP,Source Port,Dest Port,DCERPC
Auth Pack Type,DCERPC Auth Level,Attempt ID,Op Num,Method,Command,Raw Data\n");
	}

	printf("\nDevice list :\n\n");
	/* The user didn't provide a packet source: Retrieve the local device list */
	if (pcap_findalldevs(&alldevs, errbuf) == -1)
	{
		fprintf(stderr, "Error in pcap_findalldevs_ex: %s\n", errbuf);
		exit(1);
	}

	/* Print the list */
	for (d = alldevs; d; d = d->next)
	{
		printf("%d. %s\n    ", ++i, d->name);

		if (d->description)
			printf(" (%s)\n", d->description);
		else
			printf(" (No description available)\n");
	}

	if (i == 0)
	{
		printf("\nNo interfaces found! Make sure WinPcap is installed.\n");
		return -1;
	}
	printf("\n");
	printf("Enter the interface number (1-%d) : ", i);
	scanf("%d", &inum);

	if (inum < 1 || inum > i)
	{
		printf("\nInterface number out of range.\n");

		/* Free the device list */
		pcap_freealldevs(alldevs);
		return -1;
	}

	/* Jump to the selected adapter */
	for (d = alldevs, i = 0; i < inum - 1; d = d->next, i++);
```

```
        /* Open the adapter */
        if ((fp = pcap_open_live(d->name,  // name of the device
                65536,                                          // portion of the packet to
capture.
                // 65536 grants that the whole packet will be captured on all the MACs.
                1,                                                      // promiscuous mode
(nonzero means promiscuous)
                1000,                                           // read timeout
                errbuf                                          // error buffer
                )) == NULL)
        {
                fprintf(stderr, "\nError opening adapter\n");
                return -1;
        }

        data = (char *)malloc(BUF_SIZE);

        if (!data){
                fprintf(stderr, "Error : Allocating Memory for data buffer : %s\n", errbuf);
                exit(1);
        }

        strncpy(alrt, "Attempting to authenticate with id : ", 37);

        typedef void(*SignalHandlerPointer)(int);
        SignalHandlerPointer previousHandler;
        previousHandler = signal(SIGINT, SignalHandler);

        /* Read the packets */
        while ((res = pcap_next_ex(fp, &header, &pkt_data)) >= 0)
        {

                if (res == 0)
                        /* Timeout elapsed */
                        continue;

                process(pkt_data, header->caplen);
        }

        if (res == -1)
        {
                printf("Error reading the packets: %s\n", pcap_geterr(fp));
                return -1;
        }
        pcap_close(fp);
        return 0;
}

void process(const u_char *buf, size_t nbytes){

        struct iphdr *iph = (struct iphdr *)(buf + 14);

        if (iph->tot_len > 118){

                struct dcerpc_pack *dce_h = (struct dcerpc_pack *)buf;

                if (dce_h->dcerpc_h.version == 5){
                        if (dce_h->dcerpc_h.pack_type == 16){
                                struct dcerpc_pack_auth *dce_auth = (struct dcerpc_pack_auth
*)buf;
```

```
                                        if (dce_auth->auth.auth_type == 10){
                                                switch (dce_auth->auth.auth_level){
                                                case 2:
                                                        memset(id, 0, 4);
                                                        alert(buf, "Attempting to connect");
                                                        process_auth(buf, nbytes, sizeof(struct
dcerpc_pack_auth), nbytes);
                                                        log_auth(buf);
                                                        break;
                                                case 4:
                                                        memset(id, 0, 4);
                                                        _snprintf(id, 4, "%d", dce_auth-
>auth.auth_context_id);
                                                        strncpy((alrt + 37), id, 4);
                                                        alert(buf, alrt);
                                                        process_auth(buf, nbytes, sizeof(struct
dcerpc_pack_auth), nbytes);
                                                        log_auth(buf);
                                                        break;
                                                }
                                        }
                                }
                                else{
                                        switch (dce_h->dcerpc_h.op_num){
                                        case 20:
                                        case 21:
                                        case 22:
                                        case 23:
                                        case 24:
                                        case 25:
                                                alert_ex(buf, "Executing  :  ");
                                                process_data(buf, nbytes, 118, (nbytes - 24));
                                                log_ex(buf);
                                                break;
                                        }
                                }
                        }
                }
        }

void process_auth(const u_char *buf, size_t nbytes, int spos, int epos){
        memset(data, 0, sizeof(BUF_SIZE));
        unsigned char   ch;
        int j = 0;
        for (int i = spos; i < epos; i++){
                ch = buf[i];
                if ((ch < 0x20) || (ch > 0x7E)){
                        continue;
                }
                data[j++] = ch;
        }
        data[j] = '\0';
}

void process_data(const u_char *buf, size_t nbytes, int spos, int epos){
        memset(data, 0, sizeof(BUF_SIZE));
        unsigned char   ch;
        int j = 0;
        for (int i = spos; i < epos; i++){
                ch = buf[i];
                if ((ch < 0x20) || (ch > 0x7E)){
                        continue;
                }
                data[j++] = ch;
        }
        data[j] = '\0';


        std::string text(data);
        std::size_t found = -1;
        if ((found = text.find("object:Win32_")) == std::string::npos){
                if ((found = text.find("Win32_")) == std::string::npos){
                        if ((found = text.find("cim_datafile")) == std::string::npos){
                                printf("%s\n", data);
                        }
                }
        }

        if (found != std::string::npos){
                memset(command, 0, 50);
                int j = 0;
                for (int i = found;; i++){
                        if (data[i] > 0x7A || data[i] < 0x30)
                                break;
                        command[j++] = data[i];
                }
                command[j] = '\0';
                printf("%s", command);
        }
        printf("\n");
}

void eth_print(const u_char *buf){
        for (int i = 0; i<ETH_ALEN; i++){
                if (i>0)
                        printf(":");
                if (buf[i] != 255)
                        printf("%02x", (unsigned char)buf[i]);
        }
        printf("\n");
}

// log execution details
void log_ex(const u_char *buf){
        struct dcerpc_pack *dce_h = (struct dcerpc_pack *)buf;
        fprintf(log_fd, "%s,", time_buf);               // timestamp
        fprintf(log_fd, "%s,", sour);                           // source ip address
        fprintf(log_fd, "%s,", dest);                           // destination ip address

        fprintf(log_fd, "%u,", ntohs(dce_h->tcp.source));       // tcp source address
        fprintf(log_fd, "%u,", ntohs(dce_h->tcp.dest));         // tcp destination address

        fprintf(log_fd, ",");
        fprintf(log_fd, ",");
        fprintf(log_fd, ",");

        fprintf(log_fd, "%d,", dce_h->dcerpc_h.op_num);         // dcerpc packet op num
        fprintf(log_fd, "%s,", opnum_methods[dce_h->dcerpc_h.op_num]);// method relevant to
op num
        fprintf(log_fd, "%s,", command);                // executed command
        fprintf(log_fd, "%s", data);                              // raw data
        fprintf(log_fd, "\n");

        fflush(stdout);
        fflush(log_fd);

}

// log authentication attempt details
void log_auth(const u_char *buf){
        struct dcerpc_pack_auth *dce_auth = (struct dcerpc_pack_auth *)buf;
        fprintf(log_fd, "%s,", time_buf);                  // timestamp
        fprintf(log_fd, "%s,", sour);                             // source ip address
        fprintf(log_fd, "%s,", dest);                             // destination ip address

        fprintf(log_fd, "%u,", ntohs(dce_auth->tcp.source));    // tcp source port
        fprintf(log_fd, "%u,", ntohs(dce_auth->tcp.dest));      // tcp destination port

        fprintf(log_fd, "%d,", dce_auth->auth.pack_type);       // dcerpc auth packet type
        fprintf(log_fd, "%d,", dce_auth->auth.auth_level);      // dcerpc auth packet
authentication level
        fprintf(log_fd, "%s,", id);

        fprintf(log_fd, ",");
        fprintf(log_fd, ",");
        fprintf(log_fd, ",");

        fprintf(log_fd, "%s", data);                              // raw data
        fprintf(log_fd, "\n");

        fflush(stdout);
        fflush(log_fd);
}

void alert(const u_char *buf, char *msg){
        memset(time_buf, 0, TIME_BUF_SIZE);
        curtime = time(NULL);
        loctime = localtime(&curtime);

        strftime(time_buf, TIME_BUF_SIZE, "%Y/%m/%d %H:%M:%S", loctime);

        struct iphdr *iph = (struct iphdr *)(buf + 14);
        memset(&addrs, 0, sizeof(struct sockaddr_in));
        addrs.sin_addr.s_addr = iph->saddr;
        sour = _strdup(inet_ntoa(addrs.sin_addr));
        addrs.sin_addr.s_addr = iph->daddr;
        dest = _strdup(inet_ntoa(addrs.sin_addr));

        printf("%s : %s -> %s : %s\n", time_buf, sour, dest, msg);
}

void alert_ex(const u_char *buf, char *msg){
        memset(time_buf, 0, TIME_BUF_SIZE);
        curtime = time(NULL);
        loctime = localtime(&curtime);

        strftime(time_buf, TIME_BUF_SIZE, "%Y/%m/%d %H:%M:%S", loctime);

        struct iphdr *iph = (struct iphdr *)(buf + 14);
        memset(&addrs, 0, sizeof(struct sockaddr_in));
        addrs.sin_addr.s_addr = iph->saddr;
        sour = _strdup(inet_ntoa(addrs.sin_addr));
        addrs.sin_addr.s_addr = iph->daddr;
        dest = _strdup(inet_ntoa(addrs.sin_addr));
        printf("%s : %s -> %s : %s", time_buf, sour, dest, msg);
}

void SignalHandler(int signal)
{
        if (signal == SIGINT) {
                free(data);
                fclose(log_fd);
        }
}
```

Appendix III

Raw PCAP output file

```
05000700000000000000000000829e8fad6899f0418e3cb81d0b0ece1200000000055736572030000000600000003000000005700510
04c00000055736572180000003000000018000000730065006c006500630074002000 2a002000660072006f006d002000570069
006e00330032005f00420049004f00530010000000000000000 0a 4 0 0 0 01000000e72043722428978d25000000 4500 0.00E+00
0a35 40 0 80 6 5a75 c0a80a0b c0a80a0a c05a c003 b2817d74 e104ee3d 5018 4029 2cca 0 5 0 0 83 10000000 c000 1000 29010000
80000000                400                1400                05a80200b40400001a4f0b973ca30297
05000700000000000000000000829e8fad6899f0418e3cb81d0b0ece1200000000055736572030000000600000003000000005700510
04c00000055736572180000003000000018000000730065006c006500630074002000 2a002000660072006f006d002000570069
006e00330032005f00420049004f00530010000000000000000 0a 4 0 0 0 0100000073e44468f6e81b1837000000 4500 00f8 0a6b
40 0 80 6 5a2f c0a80a0b c0a80a0a c05a c003 b2818d68 e1051239 5018 4029 2135 0 5 0 0 83 10000000 d000 1000 45010000
84000000                400                1400                07680200b4040000c76f5de5ff5773f2
05000700000000000000000000829e8fad6899f0418e3cb81d0b0ece1200000000055736572030000000600000003000000005700510
04c0000005573657 21a000000340000001a000000730065006c006500630074002000 2a002000660072006f006d002000570069
006e00330032005f0056006f006c0075006d006500100000000000000000000000000000000000000     0a     4     0c     0     0
01000000d077773e2c79577249000000 4500 00f8 0aa5 40 0 80 6 59f5 c0a80a0b c0a80a0a c05a c003 b2819fac e105a175 5018
4001  1aef  0  5  0  0  83  10000000  d000  1000  68010000  88000000  400  1400  1e940200b40400001a3a4fc0ff122680
05000700000000000000000000829e8fad6899f0418e3cb81d0b0ece1200000000055736572030000000600000003000000005700510
04c0000005573657 21b000000360000001b000000730065006c006500630074002000 2a002000660072006f006d002000570069
006e00330032005f00500072006f006300650073007300000010000000000000000000000000000000     0a     4     8     0     0
01000000629aec9acd0962355f000000 4500 00f8 0b1c 40 0 80 6 597e c0a80a0b c0a80a0a c05a c003 b281cc44 e106c52d 5018
3f2e  968f  0  5  0  0  83  10000000  d000  1000  b2010000  88000000  400  1400  23dc0200b4040000b01a66c26f946887
05000700000000000000000000829e8fad6899f0418e3cb81d0b0ece1200000000055736572030000000600000003000000005700510
04c0000005573657 21b000000360000001b000000730065006c006500630074002000 2a002000660072006f006d002000570069
006e00330032005f00530065007200760069006300650000000100000000000000000000000000000000     0a     4     8     0     0
0100000175864bf9a3bbe4e9d000000 4500 00f8 0c1e 40 0 80 6 587c c0a80a0b c0a80a0a c05a c003 b2822c6a e10b8777 5018
3f61  df97  0  5  0  0  83  10000000  d000  1000  69020000  8c000000  400  1400  205c0200b4040000b330bfee82c2c87d
05000700000000000000000000829e8fad6899f0418e3cb81d0b0ece1200000000055736572030000000600000003000000005700510
04c0000005573657 21e0000003c0000001e000000730065006c006500630074002000 2a002000660072006f006d002000570069
006e00330032005f004e0074004c006f00670045007600650 06e00740010000000000000000000000     0a     4     4     0     0
010000005e80917beee53b2648010000 4500 118 238a 40 0 80 6 40f0 c0a80a0b c0a80a0a c0a0 c003 821b28b8 b8e398a7 5018
3f22  4813  0  5  0  0  83  10000000  f000  1000  7d130000  b0000000  400  1400  0d8c0300b40400006745df17a62849ae
05000700000000000000000000829e8fad6899f0418e3cb81d0b0ece1200000000055736572030000000600000003000000005700510
04c0000005573657 22f0000005e0000002f000000730065006c006500630074002000 2a002000660072006f006d0020005700690
06e00330032005f004c006f00670069006300610 06c0053006800610072006500530065006300750072006900740079005300650
07400740069006e006700000001000000000000000 0a 4 0 0 0 01000000bf426ab2125e5a07b2000000 4500 00f8 23b5 40 0 80 6
4.00E+06 c0a80a0b c0a80a0a c0a0 c003 821b385c b8e3a453 5018 3f52 2807 0 5 0 0 83 10000000 d000 1000 98130000 8c000000
400                1400                27400200b404000080212615b8285089
05000700000000000000000000829e8fad6899f0418e3cb81d0b0ece1200000000055736572030000000600000003000000005700510
04c0000005573657 21e0000003c0000001e000000730065006c006500630074002000 2a002000660072006f006d002000570069
006e00330032005f0057004d00490053006500740074006900 6e006700100000000000000000000000     0a     4     4     0     0
01000000ef3b4f4164f4d733c3000000 4500 118 23f5 40 0 80 6 4085 c0a80a0b c0a80a0a c0a0 c003 821b4910 b8e4353f 5018
3efe  65dd  0  5  0  0  83  10000000  f000  1000  b7130000  a8000000  400  1400  2d280300b4040000b55feccbf78f718b
05000700000000000000000000829e8fad6899f0418e3cb81d0b0ece1200000000055736572030000000600000003000000005700510
04c0000005573657 22b000000560000002b000000730065006c006500630074002000 2a002000660072006f006d002000570069
006e00330032005f004f0053005200650063006f007600650072007900430 06f006e0066006900670075007200610074006900 6f
006e00000010000000000000000000000000000000000 0a 4 8 0 0 01000000bed16a3c8fff5b08d6000000 4500 00f8 2424 40 0 80 6
4076 c0a80a0b c0a80a0a c0a0 c003 821b59f4 b8e45afb 5018 3f0e 561f 0 5 0 0 83 10000000 d000 1000 d4130000 8c000000 400
1400                0e340300b404000027daed576639d89e
05000700000000000000000000829e8fad6899f0418e3cb81d0b0ece1200000000055736572030000000600000003000000005700510
04c0000005573657 21e0000003c0000001e000000730065006c006500630074002000 2a002000660072006f006d002000570069
006e00330032005f0043004f004d00530065007400740069006e00670010000000000000000000000
```

References

[1] "Desktop Operating System Market Share Worldwide," [Online]. Available: http://gs.statcounter.com/os-market-share/desktop/worldwide.

[2] "Introduction to WMI," [Online]. Available: https://technet.microsoft.com/en-us/library/cc181125.aspx.

[3] "WMI: Monitoring for Malicious Activity," [Online]. Available: https://www.fireeye.com/blog/threat-research/2016/08/wmi_vs_wmi_monitor.html.

[4] L. Galang, 2010. [Online]. Available: http://blog.trendmicro.com/windows-wmi-abused-formalware-operations.

[5] P. Craig, 2010. [Online]. Available: http://ha.cked.net.

[6] "Shadowserver Foundation," 2010. [Online]. Available: http://www.nartv.org/mirror/shadows-in-the-cloud.pdf.

[7] "Trand Micro," 2010. [Online]. Available: http://threatinfo.trendmicro.com/vinfo/virusencyclo/default5..

[8] "Microsoft MSDN LibrRY," [Online]. Available: https://msdn.microsoft.com/en-us/library/aa389276(v=vs.85).aspx.

[9] "Passing the Hash," [Online]. Available: http://passing-the-hash.blogspot.com/2013/04/missing-pth-tools-writeup-wmic-wmis-curl.html.

[10] 2014. [Online]. Available: http://www.codeproject.com.

[11] "Abusing Windows Management," [Online]. Available: https://www.blackhat.com/docs/us-15/materials/us-15-Graeber-Abusing-Windows-Management-Instrumentation-WMI-To-Build-A-Persistent%20Asynchronous-And-Fileless-Backdoor-wp.pdf.

[12] "Kansa," [Online]. Available: http://www.powershellmagazine.com/2014/07/18/kansa-a-powershell-based-incident-response-framework/.

[13] "Win Cap," [Online]. Available: https://www.winpcap.org.