DETECTING APT USING ANOMALOUS WINDOWS REMOTE MANAGEMENT METHODS AND DYNAMIC RPC ENDPOINT MAPPING

KEY POINTS

- Adversaries can leverage methods that are often used for legitimate use cases in order to commit malicious actions without the use of advanced malware
- It is often difficult to distinguish between genuine activity and misuse
- RSA Security Analytics and RSA ECAT can be leveraged to detect and combat these threats
- Related parsers have been delieved to customers via RSA Live and posted on the RSA Security Analytics Community (rsa.im/SAcommunity)

Much has been said regarding Advanced Persistent Threats (APTs) in the last few years. To many IT managers and departments, the acronym alone conjures thoughts of nation-states conducting highly intricate and advanced attacks, targeting government and critical infrastructure and causing catastrophic intrusions. However, these actors will only be as advanced as they need to be to accomplish their objective, and in many instances they are able to utilize existing technologies and policies to do this.

There are a number of inherent methods within Windows environments that allow adversaries to conduct actions that achieve their objectives without the use of advanced malware. The issue with these native methods is that they all have legitimate use cases. As such, determining the use of these utilities on the network as being malicious or benign is heavily dependent on an organization's knowledge of their environment, and having well defined intercommunication policies. This issue only increases with the size of the organization. In many cases, a lack of insight or visibility into an organization's own technologies, practices, and policies allow the adversary to leverage them against an organization. This allows threat actors to hide in plain sight, minimizing exposure of their more advanced tools and indicators while maximizing their access. This report will discuss some of the methods that adversaries use against organizations, why they are effective, and methods to leverage RSA Security Analytics and RSA ECAT (Enterprise Compromise Assessment Tool) to detect and combat these threats.

OVERVIEW

One of the primary methods in which APT actors utilize existing architecture is by using Windows interconnectivity to increase their foothold into the network. Most commonly, allowed interaction between systems allows attackers to conduct lateral movement. This allows attackers to navigate from the initial compromised hosts to the more high-value targets in the environment. Some of the methods used to compromise additional hosts and move throughout the environment are as follows:

- Manually Scheduling Tasks Remotely
    - At.exe command
    - Schtasks.exe command
- Creating/Modifing Remote Services
    - Service Control (sc.exe command)
- Remote execution / Reconnaisance
    - Windows Management Instrumentation

The difficulty with combatting these methods is that any of them may have legitimate use cases within an environment. This serves to increase the utility of these methods to attackers, as the signal-to-noise ratio of malicious use to legitimate use is often very high. Also, these tools are legitimate binaries and methods in Windows, so the detection of malicious use by endpoint protection mechanisms is consistently very low. Given this exploitation of legitimate communication between systems, it is much more important to understand the environment in which this activity is seen if response personnel are to combat these threats. This report will look at the known methods within the protocols used by these utilities to determine their use on the network, so that the activity can be compared to the legitimate use cases of an environment to determine possible malicious activity.

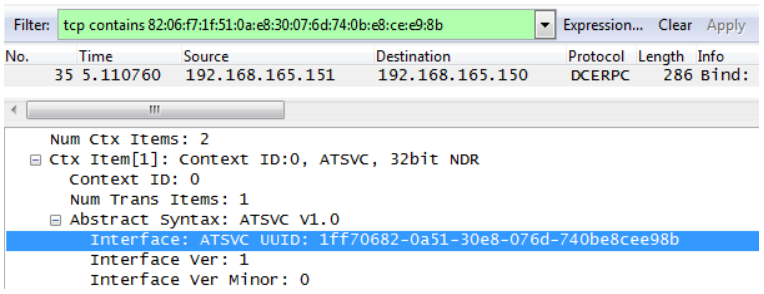
AT.EXE

The *at* command was implemented in Windows 2000 as a method of manually scheduling tasks from the command line.[1] It was designed to allow administrators to schedule tasks on host or remote systems. However, the command still exists, and is operable, up to Windows 7 and Server 2008 systems. Attackers can use this command to create immediate or longer-term tasks that allow them to execute commands on remote systems. The *at* command works by:

1. Connecting to the IPC$ share on the remote system over TCP/445
2. Creating a named pipe named "atsvc"
3. Sending a JobAdd request with the command to be run

The JobAdd request is useful in detecting this type of traffic. Since the *at* command will submit a known UUID in the bind request to port TCP/445, this value can be detected, and parsed through the stream to the JobAdd request.[2]

[1] "How To Use the At command to Schedule Tasks", http://support.microsoft.com/kb/313565
[2] "[MS-TSCH]: Task Scheduler Service Remoting Protocol", http://download.microsoft.com/download/9/5/E/95EF66AF-9026-4BB0-A41D-A4F81802D92C/[MS-TSCH].pdf

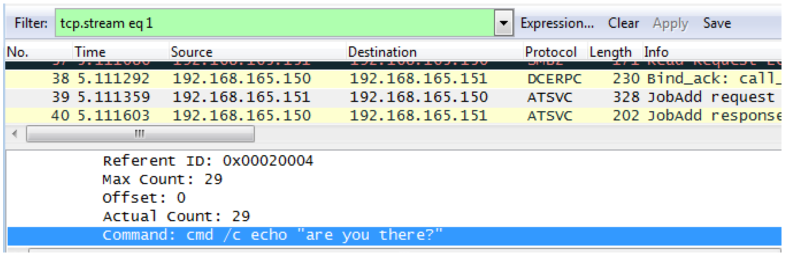


```
Filter: tcp contains 82:06:f7:1f:51:0a:e8:30:07:6d:74:0b:e8:ce:e9:8b    Expression...  Clear  Apply
No.    Time        Source            Destination       Protocol  Length  Info
   35 5.110760    192.168.165.151   192.168.165.150   DCERPC       286  Bind:

    Num Ctx Items: 2
  ⊟ Ctx Item[1]: Context ID:0, ATSVC, 32bit NDR
      Context ID: 0
      Num Trans Items: 1
    ⊟ Abstract Syntax: ATSVC V1.0
        Interface: ATSVC UUID: 1ff70682-0a51-30e8-076d-740be8cee98b
        Interface Ver: 1
        Interface Ver Minor: 0
```

Figure 1: *ATSVC* UUID Being Passed During RPC Bind Request

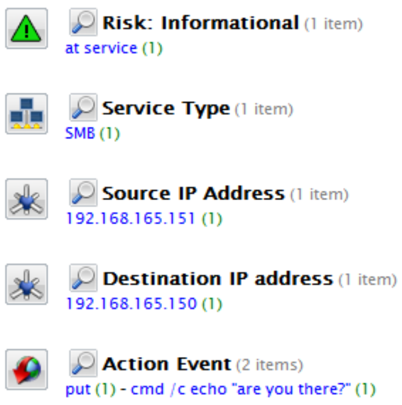
As the JobAdd request is well defined in the "Task Scheduler Remoting Protocol" documentation, the command location and length fields can be parsed to retrieve and register the command being sent into meta.



```
Filter: tcp.stream eq 1    Expression...  Clear  Apply  Save
No.    Time        Source            Destination       Protocol  Length  Info
   38 5.111292    192.168.165.150   192.168.165.151   DCERPC       230  Bind_ack: call_
   39 5.111359    192.168.165.151   192.168.165.150   ATSVC        328  JobAdd request
   40 5.111603    192.168.165.150   192.168.165.151   ATSVC        202  JobAdd response

        Referent ID: 0x00020004
        Max Count: 29
        Offset: 0
        Actual Count: 29
        Command: cmd /c echo "are you there?"
```

Figure 2: Length Value and Command String in JobAdd Request



Risk: Informational (1 item)
at service (1)

Service Type (1 item)
SMB (1)

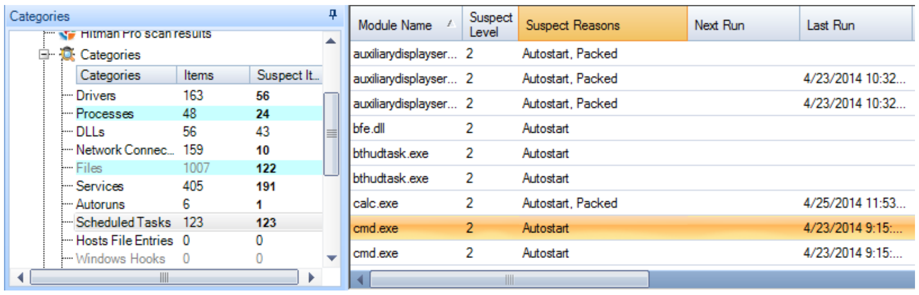
Source IP Address (1 item)
192.168.165.151 (1)

Destination IP address (1 item)
192.168.165.150 (1)

Action Event (2 items)
put (1) - cmd /c echo "are you there?" (1)

Figure 3: Meta Created from JobAdd Request

Once the remote system receives the request, it creates a .job file under c:\WINDOWS\System32\Tasks with a file name of At<*job number*>.job. The ECAT agent will see this, and show the command to be run under the "Scheduled Tasks" Category.

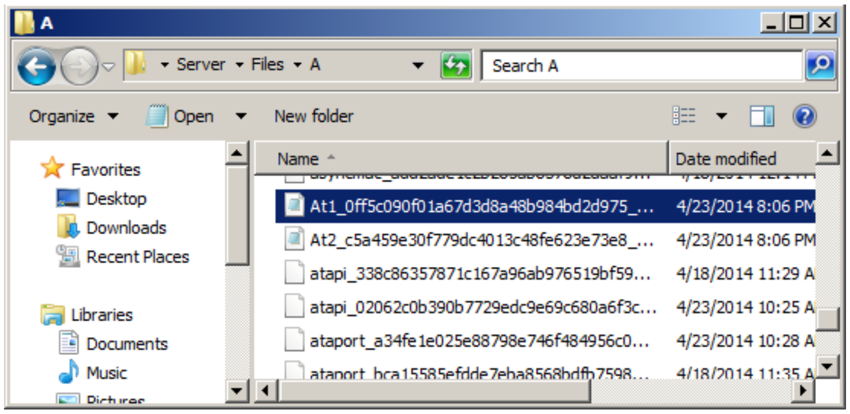


Categories	Items	Suspect It...
Drivers	163	56
Processes	48	24
DLLs	56	43
Network Connec...	159	10
Files	1007	122
Services	405	191
Autoruns	6	1
Scheduled Tasks	123	123
Hosts File Entries	0	0
Windows Hooks	0	0

Module Name	Suspect Level	Suspect Reasons	Next Run	Last Run
auxiliarydisplayser...	2	Autostart, Packed		
auxiliarydisplayser...	2	Autostart, Packed		4/23/2014 10:32...
auxiliarydisplayser...	2	Autostart, Packed		4/23/2014 10:32...
bfe.dll	2	Autostart		
bthudtask.exe	2	Autostart		
bthudtask.exe	2	Autostart		
calc.exe	2	Autostart, Packed		4/25/2014 11:53...
cmd.exe	2	Autostart		4/23/2014 9:15:...
cmd.exe	2	Autostart		4/23/2014 9:15:...

Figure 4: *cmd.exe* Sent via *at* Command in Scheduled Tasks

Upon a full scan, ECAT will download the .job files from the client. The .job files can be manually reviewed by downloading the files from the Files directory under Server in the ECAT directory.



Name	Date modified
At1_0ff5c090f01a67d3d8a48b984bd2d975_...	4/23/2014 8:06 PM
At2_c5a459e30f779dc4013c48fe623e73e8_...	4/23/2014 8:06 PM
atapi_338c86357871c167a96ab976519bf59...	4/18/2014 11:29 A
atapi_02062c0b390b7729edc9e69c680a6f3c...	4/23/2014 10:25 A
ataport_a34fe1e025e88798e746f484956c0...	4/23/2014 10:28 A
ataport_bca15585efdde7eba8568bdfb7598...	4/18/2014 11:35 A

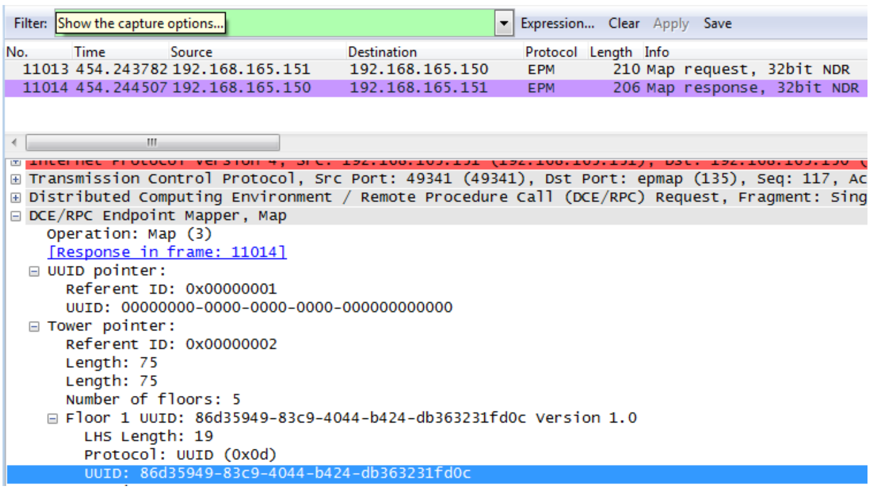
Figure 5: At.job files in *<ECAT Directory>\Server\Files\A*

SCHEDULED TASKS (SCHTASKS.EXE)

The *schtasks* command was introduced with Windows XP/Server 2003 as a replacement method for the *at* command to allow administrators to schedule jobs from the command line.[3] As such, adversaries can utilize this functionality to exploit the same attack vector as with the *at* command.

While the *at* command uses a well-known endpoint in the form of the named pipe “atsvc”, the *schtasks* command uses dynamic RPC endpoint mapping to determine its communication port. As such, it submits the *ITaskSchedulerService* UUID to the RPC endpoint mapping port (TCP/135) and receives a port assignment with which to begin communication.

[3] “Schtasks.exe”, http://msdn.microsoft.com/en-us/library/windows/desktop/bb736357(v=vs.85).aspx

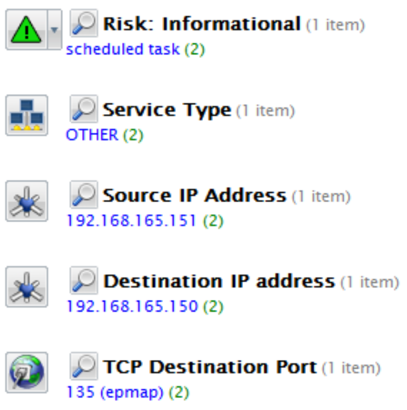


```
Filter: Show the capture options...                Expression...  Clear  Apply  Save
No.    Time       Source           Destination      Protocol Length Info
 11013 454.243782 192.168.165.151  192.168.165.150  EPM         210 Map request, 32bit NDR
 11014 454.244507 192.168.165.150  192.168.165.151  EPM         206 Map response, 32bit NDR

⊞ Transmission Control Protocol, Src Port: 49341 (49341), Dst Port: epmap (135), Seq: 117, Ac
⊞ Distributed Computing Environment / Remote Procedure Call (DCE/RPC) Request, Fragment: Sing
⊟ DCE/RPC Endpoint Mapper, Map
    Operation: Map (3)
    [Response in frame: 11014]
  ⊟ UUID pointer:
      Referent ID: 0x00000001
      UUID: 00000000-0000-0000-0000-000000000000
  ⊟ Tower pointer:
      Referent ID: 0x00000002
      Length: 75
      Length: 75
      Number of floors: 5
    ⊟ Floor 1 UUID: 86d35949-83c9-4044-b424-db363231fd0c Version 1.0
        LHS Length: 19
        Protocol: UUID (0x0d)
        UUID: 86d35949-83c9-4044-b424-db363231fd0c
```

Figure 6: *ITaskSchedulerService* UUID Passed to EPMAP

As the UUID for this service is known, Security Analytics can create meta when this UUID is seen passed with the appropriate protocol payload.[4]



Risk: Informational (1 item)
scheduled task (2)

Service Type (1 item)
OTHER (2)

Source IP Address (1 item)
192.168.165.151 (2)

Destination IP address (1 item)
192.168.165.150 (2)

TCP Destination Port (1 item)
135 (epmap) (2)

Figure 7: Meta Created from *ITaskSchedulerService* EPMAP Request

Once the job is scheduled with the Task Scheduler, the ECAT agent will recognize it as the command being executed by the job. The binary that is scheduled to be run will appear in the "Scheduled Tasks" category in the ECAT server UI in the same manner as the *at* command result in

Figure 4.

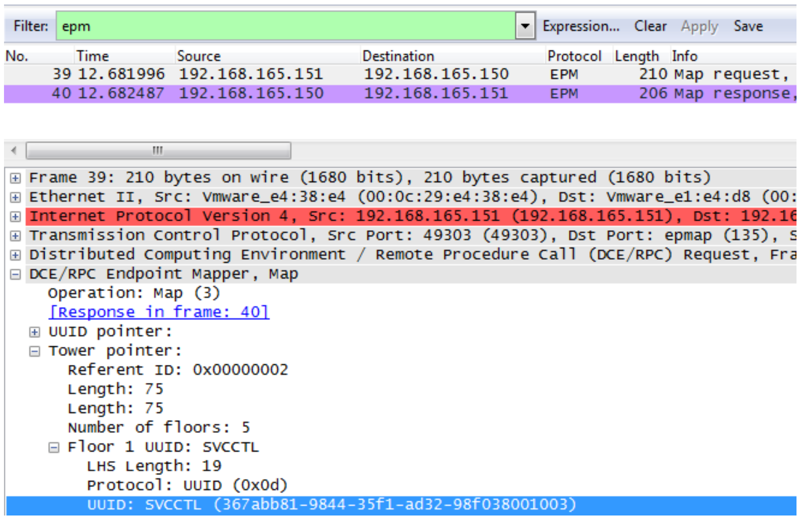
SERVICE CONTROL (SC.EXE)

Another method in which attackers will interact with other systems over the network is by using the *sc* command. The *sc* command was introduced with Windows NT 3.51 and included on endpoints as of Windows XP. The *sc* command uses the Service Control Manager Remote Protocol to interact with the Service Control Manger on the local, or remote, system. This command was designed to allow administrator to create, control, or remove services from the command line. It is commonly used by attackers to create

[4] "[MS-TSCH]: Task Scheduler Service Remoting Protocol", http://download.microsoft.com/download/9/5/E/95EF66AF-9026-4BB0-A41D-A4F81802D92C/[MS-TSCH].pdf

new malicious services, disable services that may prove problematic, or remove services as necessary. While there are other command line binaries that allow for control of services remotely (namely *netsvc* and *instsvc*), these do not allow for the creation of new services remotely and as such, the *sc* command is more commonly preferred by attackers.[5]

Microsoft has allocated both an RPC interface UUID to present to the RPC endpoint mapping port, as well as a named pipe named "svcctl".

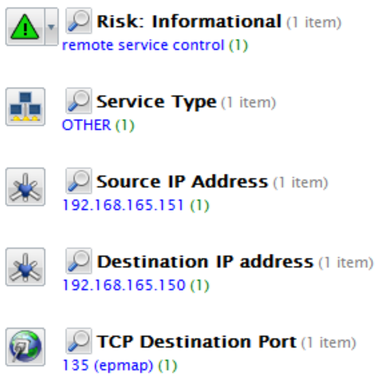


```
Filter: epm                                   Expression... Clear Apply Save
No.    Time        Source           Destination      Protocol Length Info
    39 12.681996  192.168.165.151  192.168.165.150  EPM         210 Map request,
    40 12.682487  192.168.165.150  192.168.165.151  EPM         206 Map response,

⊞ Frame 39: 210 bytes on wire (1680 bits), 210 bytes captured (1680 bits)
⊞ Ethernet II, Src: Vmware_e4:38:e4 (00:0c:29:e4:38:e4), Dst: Vmware_e1:e4:d8 (00:
⊞ Internet Protocol Version 4, Src: 192.168.165.151 (192.168.165.151), Dst: 192.16
⊞ Transmission Control Protocol, Src Port: 49303 (49303), Dst Port: epmap (135), S
⊞ Distributed Computing Environment / Remote Procedure Call (DCE/RPC) Request, Fra
⊟ DCE/RPC Endpoint Mapper, Map
    Operation: Map (3)
    [Response in frame: 40]
  ⊞ UUID pointer:
  ⊟ Tower pointer:
      Referent ID: 0x00000002
      Length: 75
      Length: 75
      Number of floors: 5
    ⊟ Floor 1 UUID: SVCCTL
        LHS Length: 19
        Protocol: UUID (0x0d)
        UUID: SVCCTL (367abb81-9844-35f1-ad32-98f038001003)
```

Figure 8: *SVCCTL* UUID Passed to EPMAP

As the UUID for this service is known, Security Analytics can create meta when this UUID is seen passed with the appropriate protocol payload.[6]



Risk: Informational (1 item)
remote service control (1)

Service Type (1 item)
OTHER (1)

Source IP Address (1 item)
192.168.165.151 (1)

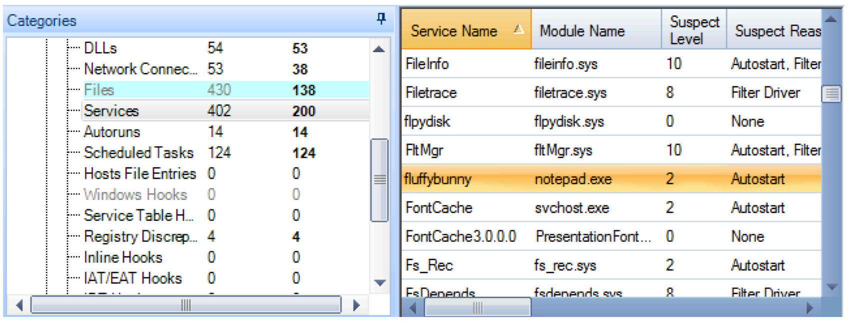
Destination IP address (1 item)
192.168.165.150 (1)

TCP Destination Port (1 item)
135 (epmap) (1)

Figure 9: Meta Created from Service Control Manager Remote Protocol EPMAP Request

As the purpose of this command is to interact with the database of installed services, ECAT will scan this database to allow the review of host artifacts resultant from the command's execution. The artifacts will be shown in the "Services" category in the Console UI on the ECAT server.

[5] "How to create a Windows service by using Sc.exe", http://support.microsoft.com/kb/251192/en-us
[6] "[MS-SCMR]: Service Control Manager Remote Protocol", http://download.microsoft.com/download/9/5/E/95EF66AF-9026-4BB0-A41D-A4F81802D92C/[MS-SCMR].pdf



Categories		
DLLs	54	53
Network Connec...	53	38
Files	430	138
Services	402	200
Autoruns	14	14
Scheduled Tasks	124	124
Hosts File Entries	0	0
Windows Hooks	0	0
Service Table H...	0	0
Registry Discrep...	4	4
Inline Hooks	0	0
IAT/EAT Hooks	0	0

Service Name	Module Name	Suspect Level	Suspect Reas
FileInfo	fileinfo.sys	10	Autostart, Filter
Filetrace	filetrace.sys	8	Filter Driver
flpydisk	flpydisk.sys	0	None
FltMgr	fltMgr.sys	10	Autostart, Filter
fluffybunny	notepad.exe	2	Autostart
FontCache	svchost.exe	2	Autostart
FontCache3.0.0.0	PresentationFont...	0	None
Fs_Rec	fs_rec.sys	2	Autostart
FsDepends	fsdepends.sys	8	Filter Driver

Figure 10: Service Entry in *Services* Category within the ECAT Console User Interface

WMI

Windows Management Instrumentation (WMI) is described by Microsoft as "the infrastructure for management data and operations on Windows-based operating systems."[7] It is the latest method provided by Microsoft to conduct administrative tasks on host and remote systems via C++, Visual Basic, or the *wmic* command-line utility.

In order to connect to remote systems, one of two primary methods is used. The first is to specify the connection information in an SWbemServices object using a moniker string.

```
Set objWMIService = GetObject("winmgmts:" & _
    "{impersonationLevel=Impersonate}!\\" & _
    Computer_B & "\root\cimv2")
```

Figure 11:SWbemServices Object Set with Moniker String[8]

This allows the script to communicate with a remote system using the user's current credentials. However, this approach is limited as alternate credentials cannot be supplied and this method is unable to connect to systems across domains.

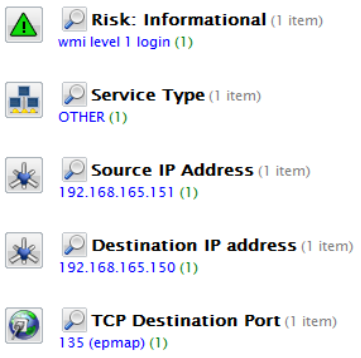
The second method is more flexible, as it allows for specification of target computer, domain, username, and password.[8] As the first method doesn't allow for on-the-fly use of stolen credentials, this second method is more attractive to adversaries. This method involves using the ConnectServer method from SWbemLocator (IWbemLocator for C++ applications) to specify and conduct connection operations.

```
strComputer = "atl-dc-01"
Set objSWbemLocator = CreateObject("WbemScripting.SWbemLocator")
Set objSWbemServices = objSWbemLocator.ConnectServer _
    (strComputer, "root\cimv2", "fabrikam\administrator", "password")
objSWbemServices.Security_.ImpersonationLevel = 3
```

Figure 12: ConnectServer Method in Visual Basic[8]

Both of these methods use the IWbemLevel1Login interface to connect to the management services interface within the requested namespace. This is useful in detecting this traffic as the interface must use the pre-designated UUID from Microsoft. As the UUID for this service is known, Security Analytics can create meta when this UUID is seen passed with the appropriate protocol payload.[9]

[7] "Windows Management Instrumentation", http://msdn.microsoft.com/en-us/library/aa394582(v=vs.85).aspx
[8] "Connecting to WMI on a Remote Computer", http://msdn.microsoft.com/en-us/library/aa389290(v=vs.85).aspx
[9] "[MS-WMI]: Windows Management Instrumentation Remote Protocol", http://download.microsoft.com/download/9/5/E/95EF66AF-9026-4BB0-A41D-A4F81802D92C/[MS-WMI].pdf



Risk: Informational (1 item)
wmi level 1 login (1)

Service Type (1 item)
OTHER (1)

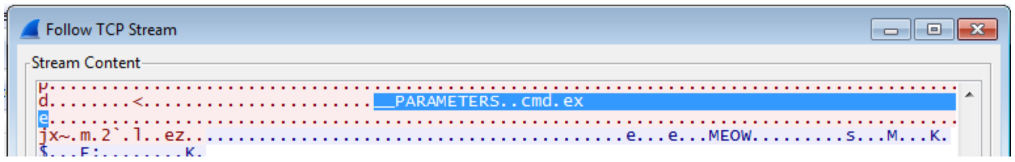
Source IP Address (1 item)
192.168.165.151 (1)

Destination IP address (1 item)
192.168.165.150 (1)

TCP Destination Port (1 item)
135 (epmap) (1)

Figure 13: Meta Created From *IWbemLevel1Login* UUID Passing to EPMAP and WMI Communication

In addition to passing the UUID to the RPC endpoint mapper, the UUID is also sent in remote requests along with parameters containing queries and commands to be executed on the remote system. In the case that *pktPrivacy* is not set, these parameters are sent to the remote system in the clear with a leading value of "__PARAMETERS".

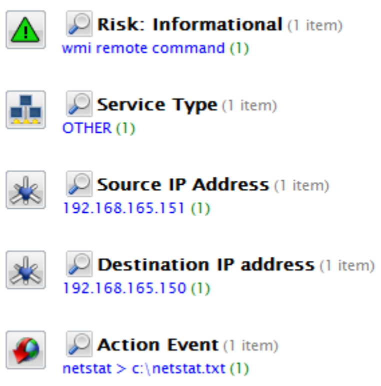


```
p.........................................................................................
d........<.......................__PARAMETERS..cmd.ex
e.........................................................................................
jx~.m.2`.l..ez.........................................e...e...MEOW.........s...M...K.
$...F:........K.
```

Figure 14: *cmd.exe* Displayed as "__PARAMETERS" Value

This value is resultant from the use of the *__PARAMETERS* class to set input parameters for the aforementioned WMI methods. By detecting the "__PARAMETERS" values and removing the "abstract" values (*__PARAMETERS* is an abstract class), SA is able to retrieve and register the parameters given to the WMI remote command, including queries and commands to be executed on the remote system.[10]

[10] "__PARAMETERS class", http://msdn.microsoft.com/en-us/library/aa394667(v=vs.85).aspx



Risk: Informational (1 item)
wmi remote command (1)

Service Type (1 item)
OTHER (1)

Source IP Address (1 item)
192.168.165.151 (1)

Destination IP address (1 item)
192.168.165.150 (1)

Action Event (1 item)
netstat > c:\netstat.txt (1)

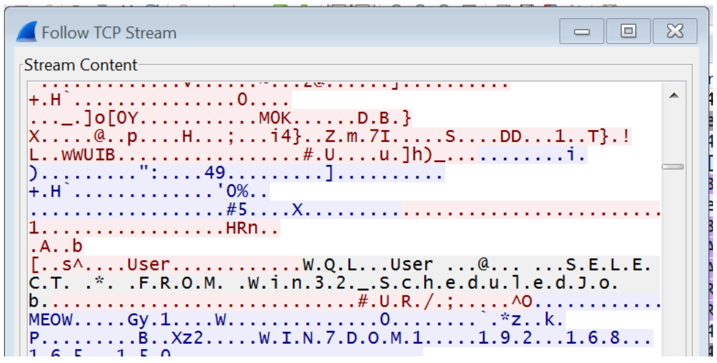
Figure 15: WMI Remote Commands Registered as Action Meta

Additionally, the use of the ExecQuery method is also interesting, as it allows adversaries to conduct actions and reconnaissance on remote systems by conducting WQL (WMI Query Language) queries against the remote system. The ExecQuery method is defined in Figure 16.

```
objWbemObjectSet = .ExecQuery( _
  ByVal strQuery, _
  [ ByVal strQueryLanguage ], _
  [ ByVal iFlags ], _
  [ ByVal objWbemNamedValueSet ] _
)
```

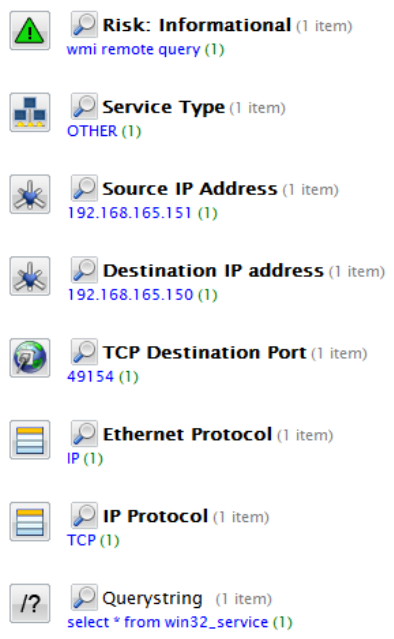
Figure 16: ExecQuery Visual Basic Method Definition

The *strQueryLanguage* parameter defines the query language to be used. Fortunately, this parameter must be defined with the value "WQL". The *strQuery* parameter is required, as it contains the value of the query to be executed. Without *pktPrivacy*, this value is sent to the remote system in the clear. Knowing that the value of *strQueryLanguage* must be "WQL", and that the following parameter is the query to be executed, the query itself can be parsed and registered as meta for any session that contains the *IWbemLevel1Login* UUID. The parser attached to this report will be necessary to create the meta shown in Figure 18.



```
+.H`...............O....
..._.]o[OY..........MOK......D.B.}
X.....@..p....H...;...i4}..Z.m.7I.....S....DD...1..T}.!
L..wWUIB.................#.U....u.]h)_............i.
).........":....49.........]..........
+.H`.............'O%..
..................#5....X...............................
1.................HRn..
.A..b
[..s^....User............W.Q.L...User ...@... ...S.E.L.E.
C.T. .*. .F.R.O.M. .W.i.n.3.2._.S.c.h.e.d.u.l.e.d.J.o.
b..............................#.U.R./.;.....^O............
MEOW.....Gy.1....W..............O........`.*z..k.
P.........B..Xz2.....W.I.N.7.D.O.M.1.....1.9.2...1.6.8...
```

Figure 17: *strQueryLanguage* and *strQuery* Values in Session



Risk: Informational (1 item)
wmi remote query (1)

Service Type (1 item)
OTHER (1)

Source IP Address (1 item)
192.168.165.151 (1)

Destination IP address (1 item)
192.168.165.150 (1)

TCP Destination Port (1 item)
49154 (1)

Ethernet Protocol (1 item)
IP (1)

IP Protocol (1 item)
TCP (1)

Querystring (1 item)
select * from win32_service (1)

Figure 18: Meta Created from *ExecQuery*

CONCLUSION

As this report has shown, there are several methods that already exist in Windows environments that allow an adversary to conduct lateral actions on systems within an environment. These methods are also difficult to distinguish between legitimate use cases and misuse. While this report has shown ways in which Security Analytics and RSA's Enterprise Compromise Assessment Tool (ECAT) can assist security personnel and incident responders in detecting and reacting to these attacks, it is critical that organizations maintain in-depth knowledge of their environments to maintain the ability to combat these more advanced threats.

The Security Analytics parser functionality that creates the meta values referenced in this report is now included in the updated 'SMB_lua' and 'DCERPC' parsers available from LIVE. A separate parser that provides only the referenced functionality has also been posted on the RSA Security Analytics Community (rsa.im/SAcommunity) to demonstrate the parser methodology and custom content functionality outlined in this report. This parser is provided as a sample and should not be placed into production if the updated 'SMB Lua' and 'DCERPC' are already in use or without previous testing in your environment.

CONTACT US

To learn more about how EMC products, services, and solutions can help solve your business and IT challenges, contact your local representative or authorized reseller—or visit us at www.emc.com/rsa.

Part Number H13846