Understanding WMI Malware

Trend Micro, Incorporated

Julius Dizon, Lennard Galang, and Marvin Cruz

A Trend Micro Research Paper | July 2010

CONTENTS

TREND MICRO

INTRODUCTION

This research paper will discuss how cybercriminals used Windows Management Instrumentation (WMI) as a venue to conveniently perform malicious activities on affected users' systems. The findings in this paper were based on a client-submitted case that TrendLabs engineers handled this March.

In the said attack, a WMI script detected by Trend Micro as TROJ_WMIGHOST.A arrived on a system bundled with a DLL malware detected as BKDR_HTTBOT.EA. The said malicious script opened two Internet browser windows. The first window allowed BKDR_HTTBOT.EA to execute via an *ActiveX* content while the second allowed it to post *Office* files (e.g., *Word, PowerPoint,* or *Excel*) to a remote site and to execute other malicious scripts from the GhostNet IP. These backdoor routines put users at risk of losing pertinent data.

It should, however, be noted that this was not the first time WMI was used for malicious purposes. In fact, in "Kiwicon 2008," a security consultant introduced "The Moth," a proof-of-concept (POC) Trojan that implements WMI __EventConsumer instances as a unique method of malicious code deployment. It is not a serious piece of malicious code but a demonstration of a new method of hiding code inside a native *Windows* functionality. It uses the WMI service to deploy a malicious code.

This paper aims to arm do-it-yourself (DIY) and small and medium-sized business (SMB) network administrators against threats that utilize Trojans leveraging WMI for their malicious purposes. It provides a brief overview as to what WMI is, how the service can be used for malicious purposes, solutions to rid affected systems of the malware, and best practices that network administrators should keep in mind to prevent system infection.

WHAT IS WMI?

WMI is the Microsoft implementation of Web-Based Enterprise Management (WBEM), which is an industry initiative to develop a standard technology for accessing management information in an enterprise environment.

WMI is the Microsoft implementation of Web-Based Enterprise Management (WBEM), which is an industry initiative to develop a standard technology for accessing management information in an enterprise environment. WMI uses the Common Information Model (CIM) industry standard to represent systems, applications, networks, devices, and other managed components. CIM is developed and maintained by the Distributed Management Task Force (DMTF).

WMI is a default service installed on *Windows XP* and *Server 2003* OSs, thus one can readily write WMI scripts or applications to automate administrative tasks on *Windows*-based systems.

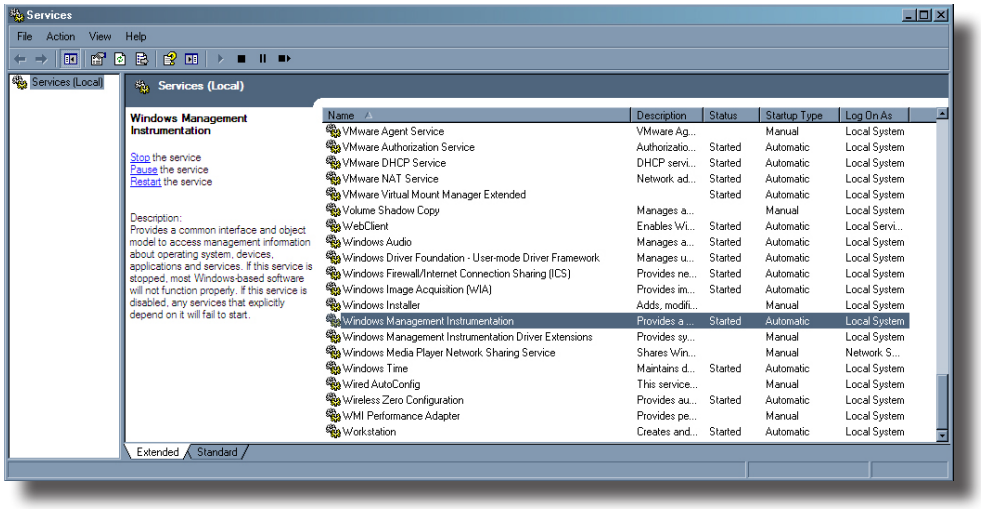


Figure 1. *The WMI service*

WMI acts as a means to acquire information on how an OS operates. It gives administrators a means to extract information about all aspects of an OS. As such, one can consider WMI as:

- A database that contains information about a system's disk, services, processor, and objects
- A means to automate the collection of hardware and software data
- A pipe that connects the inner secrets of the Microsoft OS to one another
- A distinctive dialect of *Visual Basic* script (VBS) with its own WMI Query Language (WQL)
- A tool used to determine an OS's properties

Unfortunately, however, each of the above-mentioned capabilities of WMI can be used for a malicious pragma in the following ways:

- As a database, malware can leverage the information found in WMI for malicious purposes, primarily information stealing.
- Because WMI is a means to automate hardware and software data collection, it can be used to automate malicious activities, too.
- As a pipe that connects the OS's inner secrets to one another, WMI can provide escalated privileges for malware to work on.
- Because WMI supports scripting, it can allow malicious scripts to be embedded in and carried out by the normal service.
- As a tool used to determine an OS's properties, WMI can be a means to spy on and probe a system, which is vital to Trojan spies.

WMI NAMESPACE: root\subscription

WMI classes stored in namespace: subscription allow permanent and general access to WMI services.

WMI classes stored in *namespace: subscription* allow permanent and general access to WMI services. The classes under *namespace* allow access to WMI data and discretely allow Win32 events, in particular, to be acted upon or processed.

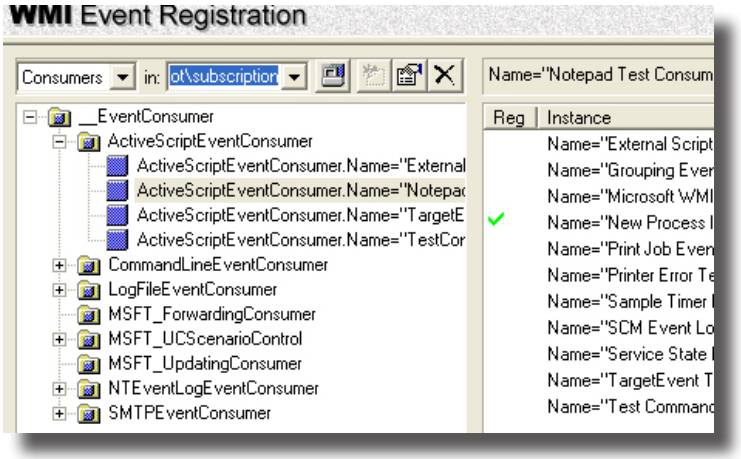


Figure 2. *WMI classes stored in* namespace: subscription

WMI System Classes

Objects from system classes such as event and provider registration, security, and event notification support WMI activities. In this paper, however, we will only highlight the system classes that TROJ_WMIGHOST.A modified using WMI.

__EventConsumer

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

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

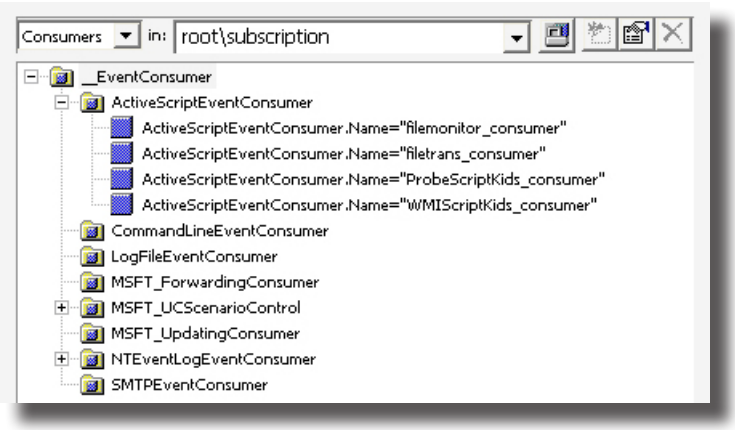
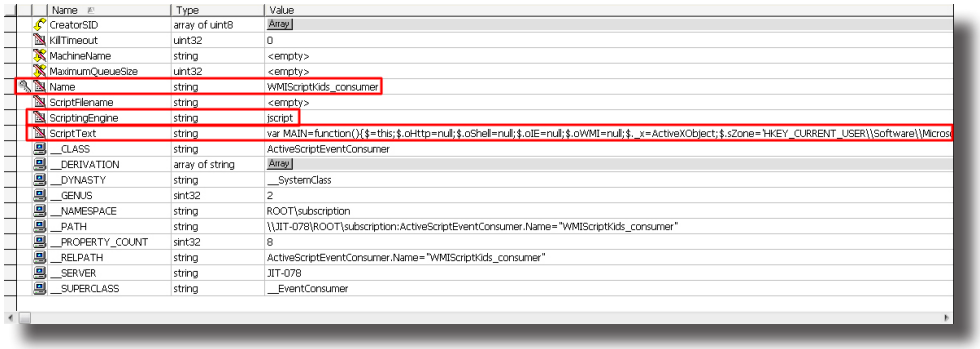


Figure 3. *__EventConsumer class*

The ActiveScriptEventConsumer class is one of the standard event consumer classes. It allows a user to run an *ActiveX* script code whenever an event is delivered to it. Scripts can be inserted to it as well. This class is unique in that it can embed scripts using the specified script language. The following properties define its script-enabled capabilities:

- **Name:** Gives a unique name to an instance of ActiveScriptEventConsumer.
- **ScriptingEngine:** The name of the scripting engine that will be used. Although the documentation states that this can be any arbitrary scripting engine, the usual ones used are VBS and JavaScript (JS).
- **ScriptText:** A string property that contains a VBS or JS code that would be executed when an event is delivered to the ActiveScriptEventConsumer instance.
- **ScriptFileName:** This property holds the full path to the VBS or JS file that would be executed upon event arrival. ScriptText and ScriptFileName properties are mutually exclusive.

WMIScriptKids_consumer is an example of the active script event consumer instance that TROJ_WMIGHOST.A creates on an affected system.



Name	Type	Value
CreatorSID	array of uint8	Array
KillTimeout	uint32	0
MachineName	string	<empty>
MaximumQueueSize	uint32	<empty>
Name	string	WMIScriptKids_consumer
ScriptFilename	string	<empty>
ScriptingEngine	string	jscript
ScriptText	string	var MAIN=function(){$=this;$.oHttp=null;$.oShell=null;$.oIE=null;$.oWMI=null;$._x=ActiveXObject;$.sZone='HKEY_CURRENT_USER\\Software\\Micros
__CLASS	string	ActiveScriptEventConsumer
__DERIVATION	array of string	Array
__DYNASTY	string	__SystemClass
__GENUS	sint32	2
__NAMESPACE	string	ROOT\subscription
__PATH	string	\\JIT-078\ROOT\subscription:ActiveScriptEventConsumer.Name="WMIScriptKids_consumer"
__PROPERTY_COUNT	sint32	8
__RELPATH	string	ActiveScriptEventConsumer.Name="WMIScriptKids_consumer"
__SERVER	string	JIT-078
__SUPERCLASS	string	__EventConsumer

Figure 4. *Sample script TROJ_WMIGHOST.A creates on an affected system*

The script that has been inserted in this example uses the JS engine whose corresponding text is specified in ScriptText. Based on our analysis of using JS, the application *wscript.exe* is responsible for executing the malicious code. However, in the case of WMI implementation, such a script is executed by the *WMI Standard Event Consumer - scripting* application, which can be found in the *WMI* folder in *%system32%/wbem/scrcons.exe.* This makes the script hard to detect since it uses a not-so-common WMI application—*scrcons.exe*—rather than the traditional JS application—*wscript.exe.*

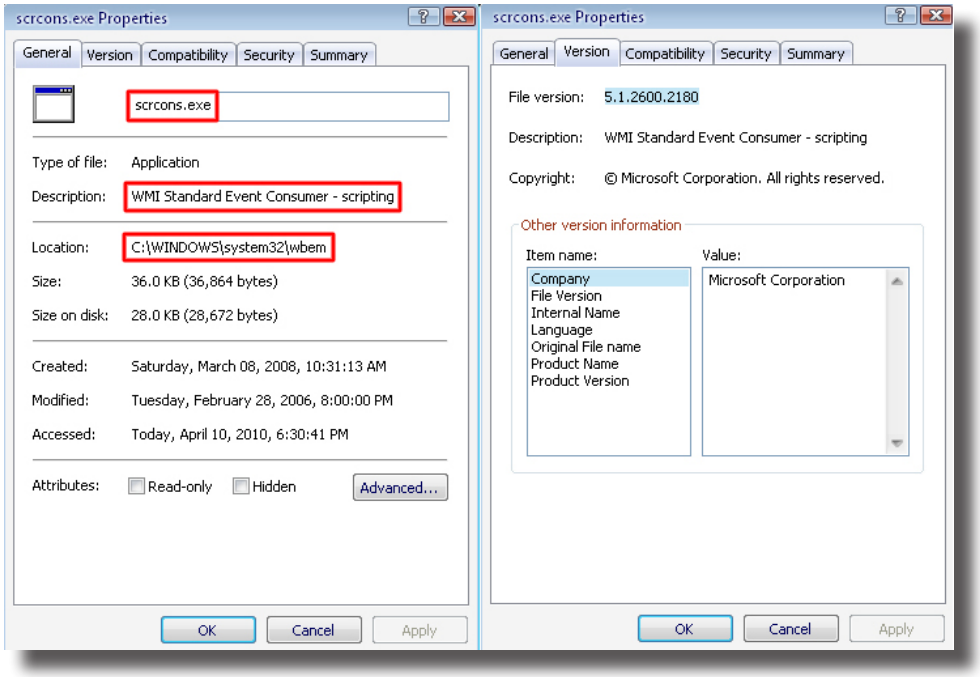


Figure 5. Scrcons.exe's *properties*

TREND MICRO

__EventFilter

An instance of an __EventFilter system class is required to register a permanent event consumer.

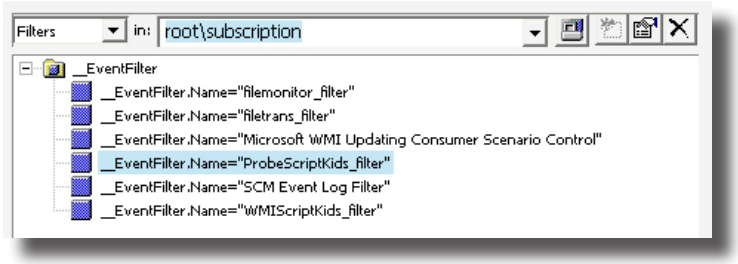


Figure 6. __EventFilter system class

__EventFilter is a mandatory class entry creation process to activate event consumer class instances.

As defined, this is a mandatory class entry creation process to activate event consumer class instances. Event filters are triggers or autostart methods to execute event consumer entries. Within this class instance, a user can monitor *Windows* system events. This can be likened to a commonly used malware method—hooking.

The example below shows that the name *WMIScriptKids_filter* has been inserted.

Name	Type	Value
CreatorSID	array of uint8	Array
EventAccess	string	<empty>
EventNamespace	string	<empty>
Name	string	WMIScriptKids_filter
Query	string	select * from __timerevent where timerid="WMIScriptKids_WMITimer"
QueryLanguage	string	wql
__CLASS	string	__EventFilter
__DERIVATION	array of string	Array
__DYNASTY	string	__SystemClass
__GENUS	sint32	2
__NAMESPACE	string	ROOT\subscription
__PATH	string	\\JIT-078\ROOT\subscription:__EventFilter.Name="WMIScriptKids_filter"
__PROPERTY_COUNT	sint32	6
__RELPATH	string	__EventFilter.Name="WMIScriptKids_filter"
__SERVER	string	JIT-078
__SUPERCLASS	string	__IndicationRelated

Figure 7. WMIScriptKids_filter inserted into a sample __EventFilter system class

Working with event filters allows one to query information from the WMI database using the WQL specified in the Query properties of the class. Once the query satisfies a TRUE condition, it activates a specified event consumer class instance specified in __FilterToConsumerBinding.

__FilterToConsumerBinding

__FilterToConsumer-Binding is used in registering permanent event consumers to relate an __Event-Consumer instance to an __EventFilter instance.

__FilterToConsumerBinding is used in registering permanent event consumers to relate an __EventConsumer instance to an __EventFilter instance.

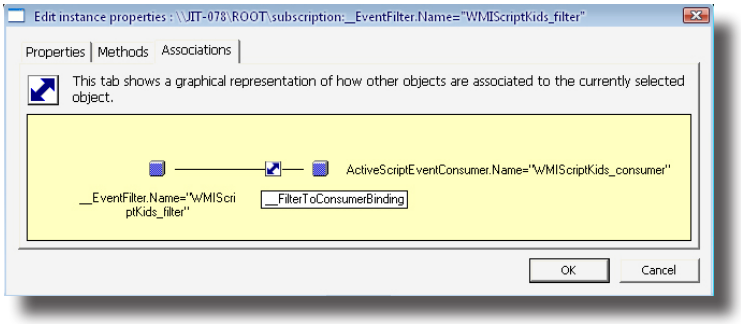


Figure 8. __FilterConsumerBinding class

This class instance associates an __EventFilter instance with an __EventConsumer instance. It completes the cycle by relating the class instances with each other. It answers the question, "What *Windows* event (__EventFilter) will I execute my script program (__EventConsumer) with?"

__TimerInstruction

__TimerInstruction specifies instructions on how timer events should be generated for consumers.

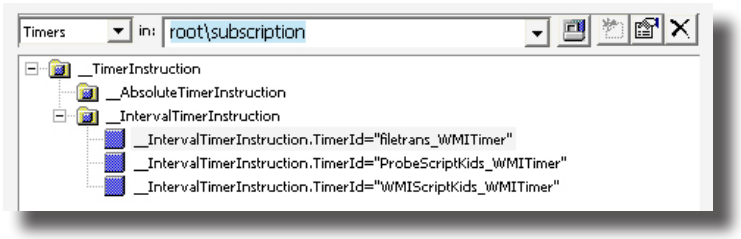


Figure 9. __TimerInstruction class

Timer instruction classes are timer events that one can use within the context of the consumer.

Timer instruction classes are timer events that one can use within the context of the consumer. It can primarily be considered a *Windows* event that an __EventFilter instance can generally query.

Name	Type	Value
IntervalBetweenEvents	uint32	60000
SkipIfPassed	boolean	false
TimerId	string	WMIScriptKids_WMITimer
__CLASS	string	__IntervalTimerInstruction
__DERIVATION	array of string	Array
__DYNASTY	string	__SystemClass
__GENUS	sint32	2
__NAMESPACE	string	ROOT\subscription
__PATH	string	\\JIT-078\ROOT\subscription:__IntervalTimerInstruction.TimerId="WMIScriptKids_WMITimer"
__PROPERTY_COUNT	sint32	3
__RELPATH	string	__IntervalTimerInstruction.TimerId="WMIScriptKids_WMITimer"
__SERVER	string	JIT-078
__SUPERCLASS	string	__TimerInstruction

Figure 10. *Sample __TimerInstruction class instance*

In the example above, the interval is set using the IntervalBetweenEvents property with the return specified in or controlled by the SkipIfPassed property. An IntervalBetweenEvents instance tells one how many milliseconds a __TimerInstruction instance can be triggered as a *Windows* event. The SkipIfPassed property, on the other hand, tells one what is returned if a query has to be done on this class instance. It is similar to asking if 60,000 milliseconds have already passed? The return will either be:

- If yes, return TRUE.
- If no, return FALSE.

This class instance may be disparate from other classes. Using this class, however, allows an __EventFilter instance to be triggered at intervals without solely relying on *Windows* events. This is thus a good means of implementing a repetitive script program specified in an __EventConsumer instance.

PUTTING THE PUZZLE PIECES TOGETHER: MALWARE ROUTINES

TROJ_WMIGHOST.A WMI Class Instances and Correlation

The table below will help users understand TROJ_WMIGHOST.A's routines in a typical malware infection.

Malware Precept	Autorun Entry/ Binding	Executable Code	Condition/Trigger
Class	__EventFilter	__EventConsumer	__TimerInstruction
Instance	filemonitor_filter filetrans_filter ProbeScriptKids_ filter WMIScriptKids_ filter	filemonitor_ consumer filetrans_filter ProbeScriptKids_ filter WMIScriptKids_ filter	(no __TimerInstruction but monitors the *%document%/ recent* folder) filetrans_WMITimer ProbeScriptKids_ WMITimer WMIScriptKids_ WMITimer
Purpose	Serves as a link to system events and the execution of malware code	Contains the malicious code that is executed on a monitored system event	Lists the monitored system events for malicious consumer script execution conditions

Table 1. *Typical TROJ_WMIGHOST.A infection routines*

filemonitor_consumer

The filemonitor_consumer script runs every time a file operation occurs on the folder that monitors the files created or modified. The files are logged on *C:\Documents and Settings\Administrator\Recent*. If the file extensions are *.TXT, .RTF, .PDF, .DOC, .DOCX, .XLS, .XLSX, .PPT,* and/or *.PPTX,* they will be copied onto *%windows%\temp\syslog\p*.

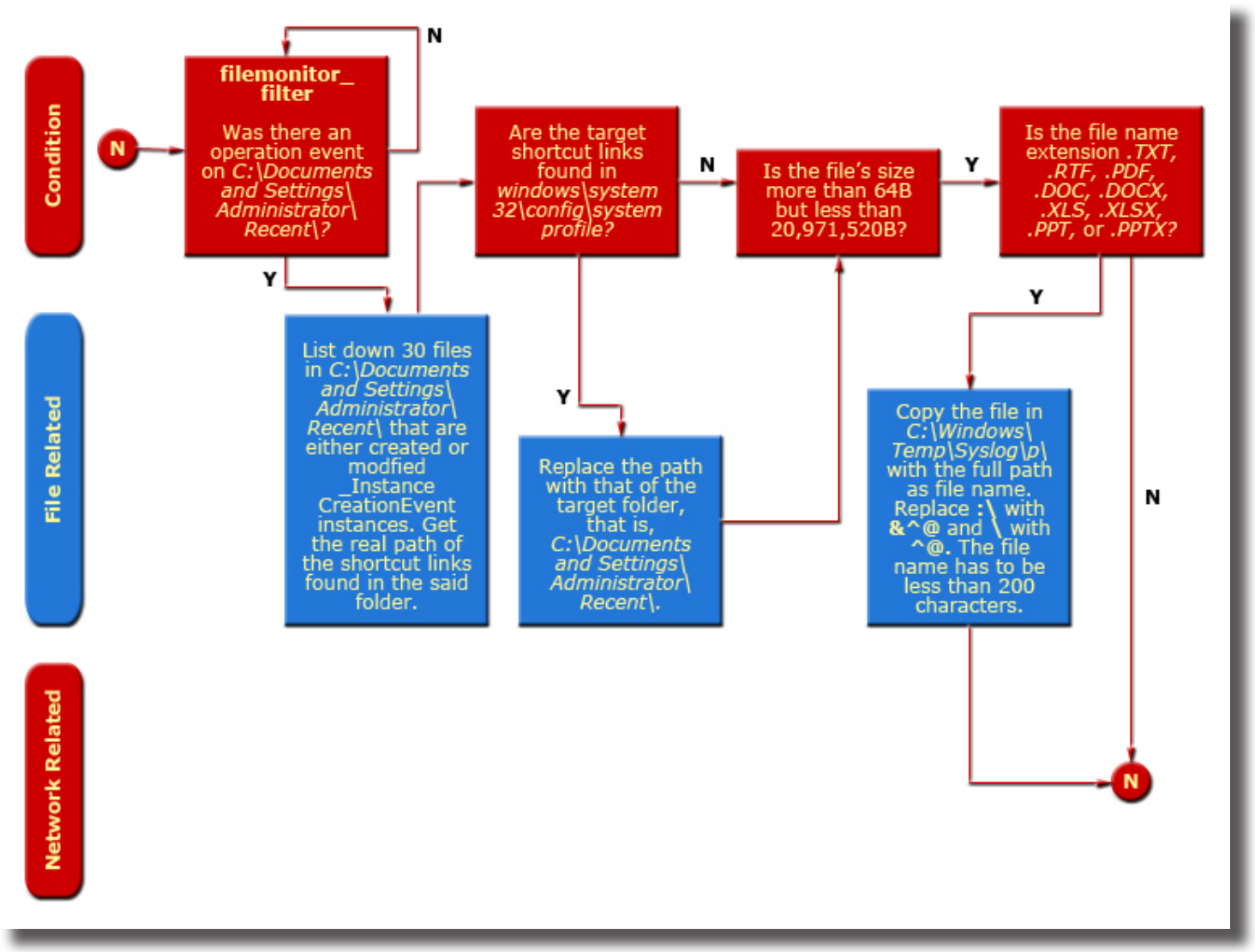


Figure 11. *How TROJ_WMIGHOST.A uses the filemonitor_consumer script*

filetrans_consumer

The filetrans_consumer script runs every 360,000 seconds as specified by the filemonitor_filter and filemonitor_timer scripts, which collect the files in *%windows%\temp\syslog\p* that are more than 360,000 seconds from when they were last modified. It checks when the folder was last modified so as not to disrupt filemonitor_consumer operations.

Once verified, TROJ_WMIGHOST.A prepares the file for compression using .CAB format. If the file is more than 102,400 bytes, it will be split accordingly into separate .CAB files that are 102,400 bytes in size and stored in *%windows%/temp/syslog/s/<hostname><Os createtime><fileLastWriteTime>1.6<curdate><curtime>@@X-Y@@.cab.*

It then posts the files to *http://abhisheksingh.blog.com/feed/* as XML content with the bin.base64 data type.

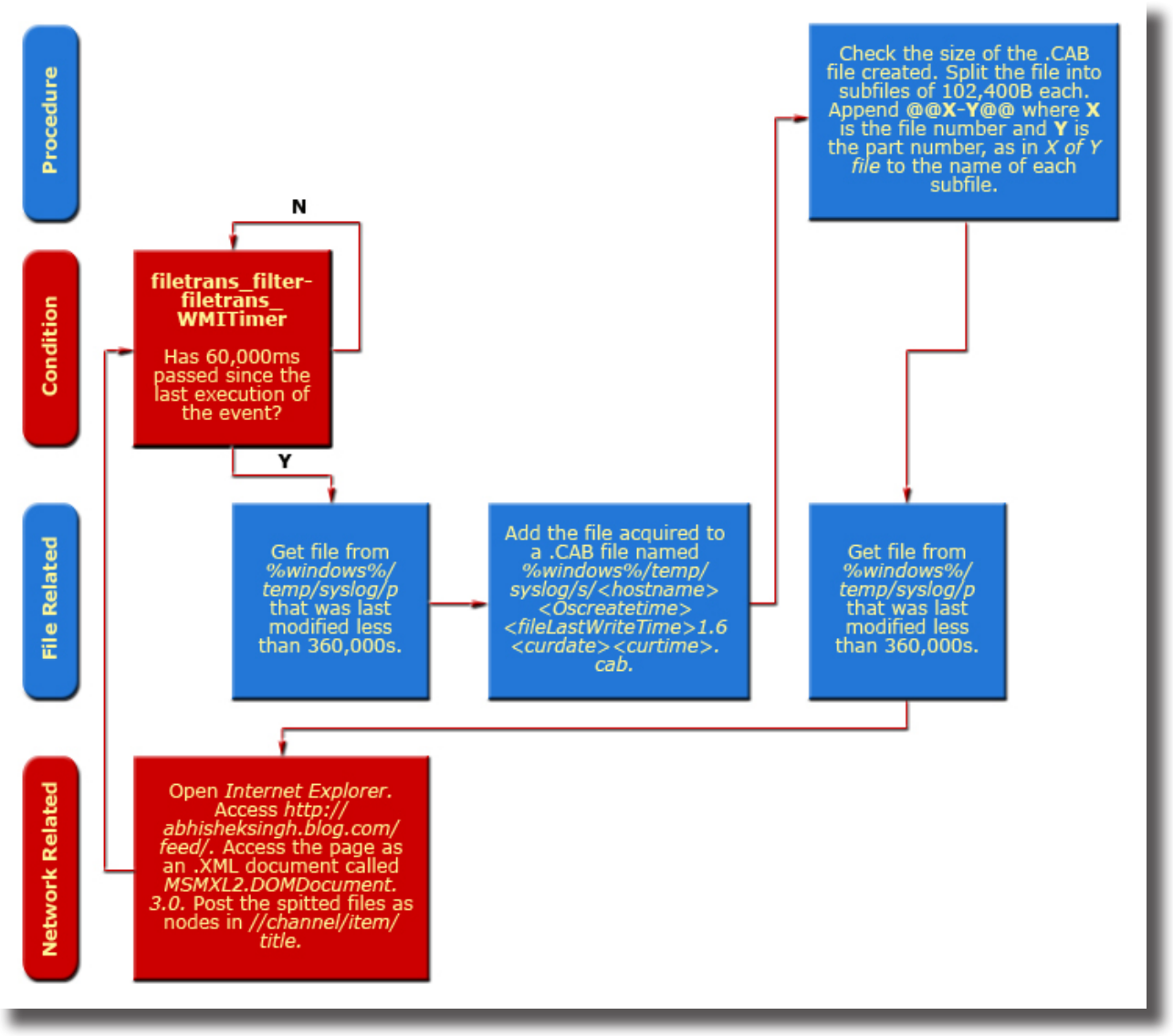


Figure 12. *How TROJ_WMIGHOST.A uses the filetrans_consumer script*

WMIScriptKids_consumer

The WMIScriptKids_consumer script runs every 360,000 seconds, as specified by the WMIScriptKids_filter and WMIScriptKids_timer scripts. Its main purpose is to open an Internet browser while loading the *ActiveX* object of a malicious component detected by Trend Micro as BKDR_HTTBOT.EA.

The WMIScriptKids_consumer script creates a function to activate an *ActiveX* object named *Mycom.myMain.1.*

To check whether BKDR_HTTBOT.EA is already running, it will check the FILE-LOCKING MUTEX used by the WMI script and BKDR_HTTBOT.EA. It will then attempt to delete the file *%temp%/mywmimutex.dat.* It will only open another instance of the malicious script if the above-mentioned file does not exist on the system.

The ATL component (BKDR_HTTBOT.EA) that the script loaded allows the binary component to send and receive data using the spawned Internet browser.

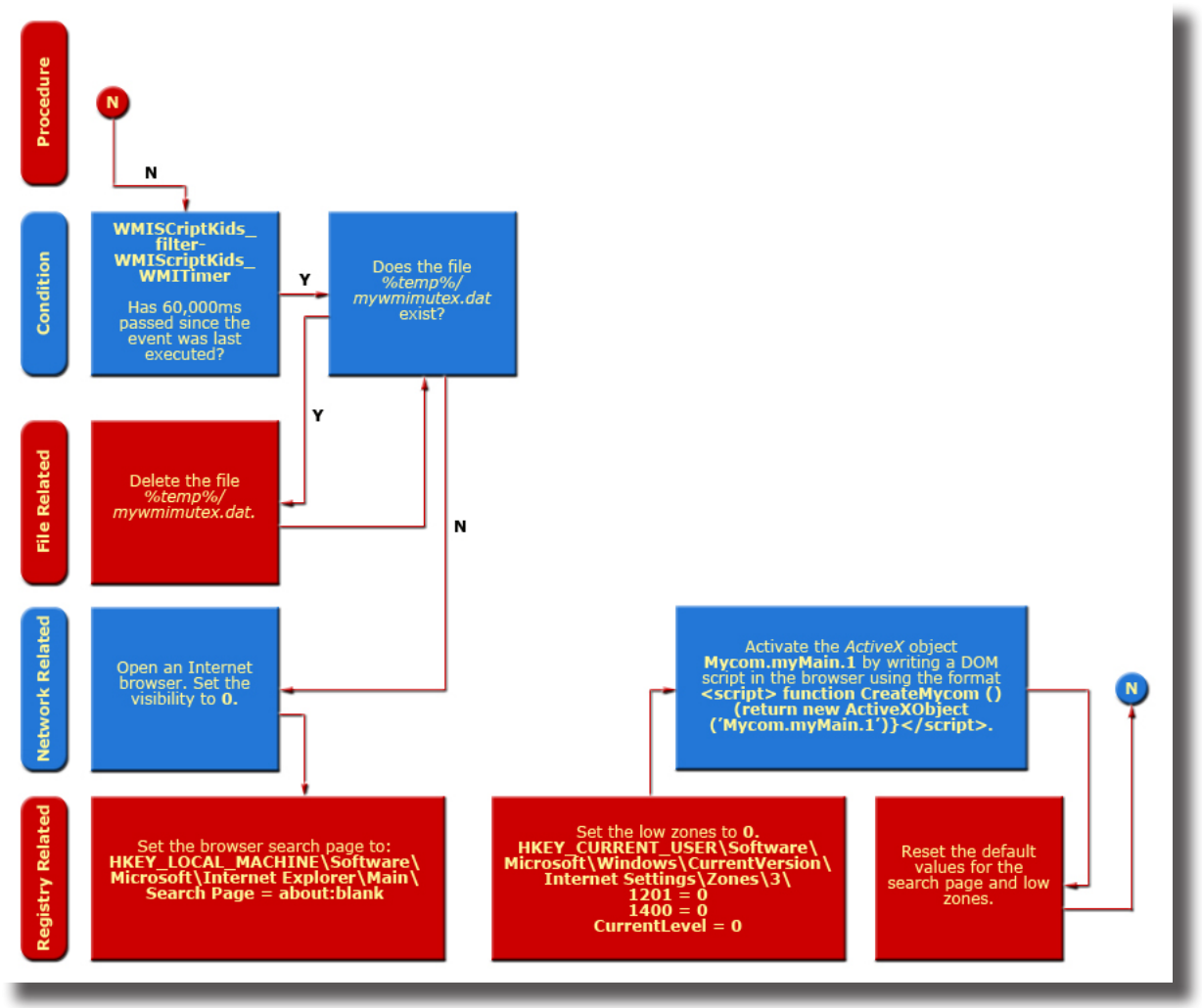


Figure 13. *How TROJ_WMIGHOST.A uses the WMIScriptKids_consumer script*

ProbeScriptKids_consumer

The ProbeScriptKids_consumer script runs every 360,000 seconds, as specified by the ProbeScriptKids_filter and ProbeScriptKids_timer scripts. It acts as an HTTP command-and-control (C&C) bot. It connects to *http://hiok125.blog.com/feed/* to parse the response as XML data and to acquire all the URL addresses on the bot list.

The script will pick only one URL from the said list and will begin to access it using the parameter *cstype=server&authname=servername&authpass=serverpass&hostname=<computername>&ostype=<OSType>&macaddr=<MACaddr>8&owner=bobotest09&version=0.5.2&t=<CurrMin+CurrSecs>.*

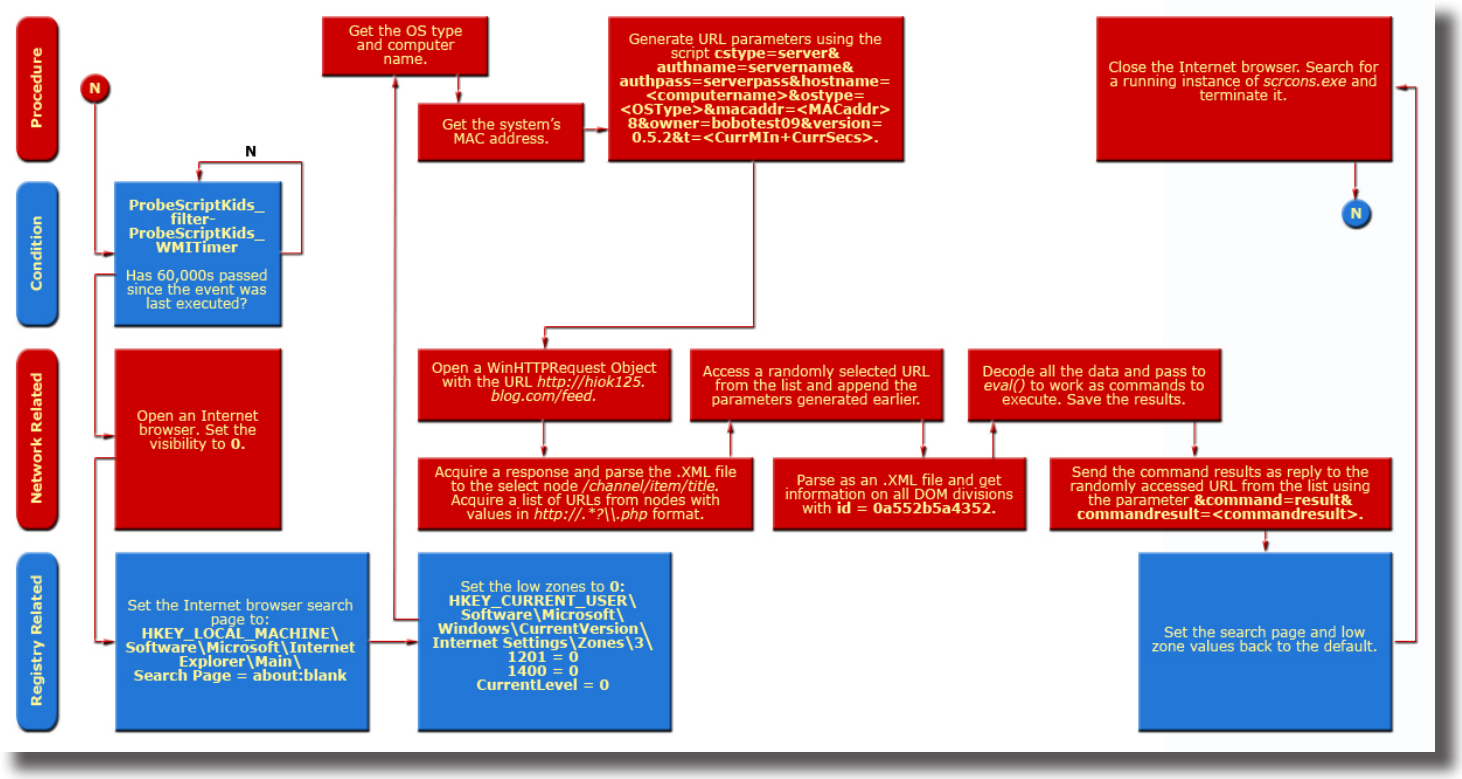


Figure 14. *How TROJ_WMIGHOST.A uses the ProbeScriptKids_consumer script*

The script will again acquire a response in the form of XML data. It will then read out and decode the assumed JS codes, which will be executed one by one as commands from the C&C.

Once executed, it will send feedback to the same URL for command results using the parameter *&command=result&commandresult=<commandresult>.*

TREND MICRO

MANUAL DETECTION

There are several ways to detect threats like TROJ_WMIGHOST.A. The key lies in understanding how to list instances of the WMI class.

Command Line: WMI CommandLine Tool

To manually detect instances of the threat in a system, type the following in 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

GUI: WMI Tools

The graphical user interface (GUI) tool can be downloaded from *http://www.microsoft.com/downloads/details.aspx?FamilyID=6430f853-1120-48db-8cc5-f2abdc3ed314&displaylang=en*.

The Event Viewer in this tool allows one to see all running instances of the malware. It will primarily allow one to see consumer, filter, and timer instances running on his/her system.

MANUAL REMOVAL

We can use the same detection tools to rid a system of malicious WMI class instances.

Command Line: WMI CommandLine Tool

To manually remove instances of the malware from a system, type the following on 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

Note: Using the command line will delete all instances of the specified classes. Ensure that there are no normal WMI class instances installed on the system before going ahead and issuing the above-mentioned commands. Otherwise, use the GUI tool instead.

GUI: WMI Tools

The GUI tool can be downloaded from *http://www.microsoft.com/downloads/details.aspx?FamilyID=6430f853-1120-48db-8cc5-f2abdc3ed314&displaylang=en*.

Accessing each class instance will give one the option to delete them by right clicking each one and selecting “Delete.”

PREVENTION

For the most part, WMI implementation requires administrative permission and rights to be installed on a system. Securing WMI means restricting access to it. For more details on securing WMI namespaces, go to this *MSDN* page.

Typically, the systems in a uniform network setup such as one wherein Active Directory is enabled are not prone to this type of threat. This particular type of attack targets local groups and individual workstations for which cybercriminals gain administrative access.

CONCLUSION

WMI is a useful tool for system administration and computer management. However, the features one adds to his/her system are also potential tools for threat distribution. It would thus be useful for one to question whether these tools are more useful or harmful before actually enabling them. WMI, for instance, has several advantages and disadvantages.

First, as a database that contains information about a system's disk, services, processor, and objects, malware can leverage the information found in WMI for malicious purposes, primarily information stealing. Second, because WMI is a means to automate hardware and software data collection, it can be used to automate malicious activities, too. Third, as a pipe that connects the OS's inner secrets to one another, WMI can provide escalated privileges for malware to work on. Fourth, because WMI supports scripting, it can allow malicious scripts to be embedded in and carried out by the normal service. Finally, as a tool used to determine an OS's properties, WMI can be a means to spy on and probe a system, which is vital to Trojan spies.

WMI feature manipulation is structured in *Windows* system classes, which can be easily modified by any inherent *Windows* programming language. Understanding *Windows* system classes and their implementation is thus a must to understand detailed features of the WMI service's structure.

WMI-related threats are a wake-up call to computing individuals. The need to inadvertently look at certain features of a system for potential damage rather than to better use it is critical in determining what new threat vectors cybercriminals may leverage. There is a thin line between restriction, security, and versatility that one always has to consider when serving one's computing needs.

GLOSSARY

- **Active Directory:** A hierarchical collection of network resources that can contain users, computers, printers, and other Active Directories. Active Directory Services (ADS) allow administrators to handle and maintain all network resources from a single location.

- **ActiveScriptEventConsumer:** A WMI Standard Consumer Class that executes a predefined script in an arbitrary scripting language when an event is delivered to it. This consumer is available on *Windows XP* and *Windows 2000*.

- **ATL component:** An "Active Template Library" component, which is a set of template-based C++ classes developed by Microsoft, intended to simplify the programming of Component Object Model (COM) objects.

- **__EventConsumer:** A WMI system class. The __EventConsumer system class is an abstract base class that is used in registering a permanent event consumer.

- **__EventFilter:** A WMI system class. The registration of a permanent event consumer requires an instance of the __EventFilter system class. This specifies what "event" a user wants to receive or act on.

- **File-locking mutex:** "Mutex" stands for "mutual exclusion," which is the most basic form of synchronization between processes. Locking a file using a mutex means that a computer resource, in this case, a file, can only be made available to one user at a time.

- **__FilterToConsumerBinding:** A WMI system class used in registering permanent event consumers to relate an instance of the __EventConsumer to an instance of __EventFilter. __FilterToConsumerBinding is an association class. It binds the "action" to an "event." The "action" is defined in ActiveScriptEventConsumer.

- **GhostNet:** The name researchers gave to a large-scale cyberspying operation discovered in March 2009 during the Information Warfare Monitor.

- **JS engine:** Refers to a "JavaScript engine," which is a specialized software program that processes JavaScript, especially for Web browsers.

- **WMI:** Stands for "Windows Management Instrumentation," which was designed for enterprise data collection and management that is both flexible and extensible to manage local and remote systems comprising arbitrary components. It is the infrastructure for data management and operation on *Windows*-based OSs. It is Microsoft's implementation of the WBEM and CIM standards from the DMTF.

- **WMI namespace:** An abstract container or environment created to hold a logical grouping of unique identifiers or symbols (i.e., names).

- **WMI system classes:** A collection of predefined classes based on the CIM. Unlike classes supplied by providers, the system classes are not declared in a Managed Object Format (MOF) file. WMI creates a set of these classes whenever a new WMI namespace is created.

- **WQL:** Stands for "WMI Query Language," which is a subset of the American National Standards Institute Structured Query Language (ANSI SQL)—with minor semantic changes. A basic WQL query remains fairly understandable for people with basic SQL knowledge.

REFERENCES

- Lennard Galang. (May 26, 2010). *TrendLabs Malware Blog.* "WMI Abused for Malware Operations." http://blog.trendmicro.com/windows-wmi-abused-for-malware-operations/ (Retrieved June 2010).
- Microsoft Corporation. (May 4, 2010). *MSDN.* "Securing WMI Namespaces." http://msdn.microsoft.com/en-us/library/aa826354%28VS.85%29.aspx (Retrieved July 2010).
- Microsoft Corporation. (May 4, 2010). *MSDN.* "Standard Consumer Classes." http://msdn.microsoft.com/en-us/library/aa393649%28v=VS.85%29.aspx (Retrieved July 2010).
- Microsoft Corporation. (May 4, 2010). *MSDN.* "Windows Management Instrumentation." http://msdn.microsoft.com/en-us/library/aa394582%28VS.85%29.aspx (Retrieved June 2010).
- Microsoft Corporation. (May 4, 2010). *MSDN.* "WMI System Classes." http://msdn.microsoft.com/en-us/library/aa394583%28VS.85%29.aspx (Retrieved June 2010).
- Microsoft Corporation. (May 4, 2010). *MSDN.* "WQL (SQl for WMI)." http://msdn.microsoft.com/en-us/library/aa394606%28VS.85%29.aspx (Retrieved July 2010).
- Paul Craig. (2010). *Ha.cked: A Lone Kiwi Hacker on a Mission of Exploitation.* "New Project: The Moth Trojan." http://ha.cked.net/ (Retrieved June 2010).
- Shadowserver Foundation. (April 6, 2010). "Shadows in the Cloud: Investigating Cyber Espionage 2.0." http://www.nartv.org/mirror/shadows-in-the-cloud.pdf (Retrieved July 2010).
- Sybase Inc. (2004). *Sybase.* "Chapter 4: Process Management." http://infocenter.sybase.com/help/index.jsp?topic=/com.sybase.dc36335_0420/html/ecgunix/ecgunix112.htm (Retrieved July 2010).
- The Trustees of Indiana University. (March 31, 2010). *Indiana University: University Information Technology Services.* "What Is Active Directory?" http://kb.iu.edu/data/ahtd.html (Retrieved July 2010).
- Trend Micro Incorporated. (March 26, 2010). *Threat Encyclopedia.* "BKDR_HTTBOT.EA." http://threatinfo.trendmicro.com/vinfo/virusencyclo/default5.asp?VName=BKDR_HTTBOT.EA (Retrieved June 2010).
- Trend Micro Incorporated. (May 18, 2010). *Threat Encyclopedia.* "TROJ_WMIGHOST.A." http://threatinfo.trendmicro.com/vinfo/virusencyclo/default5.asp?VName=TROJ_WMIGHOST.A (Retrieved June 2010).
- Uros Calakovic. (July 29, 2008). *The Code Project: Your Development Resource.* "Creating WMI Permanent Event Subscriptions Using MOF." http://www.codeproject.com/KB/system/PermEvtSubscriptionMOF.aspx (Retrieved June 2010).

TREND MICRO

- Wikimedia Foundation Inc. (April 13, 2010). *Wikipedia.* "Pragma." http://en.wikipedia.org/wiki/Pragma (Retrieved June 2010).
- Wikimedia Foundation Inc. (April 21, 2010). *Wikipedia.* "Hooking." http://en.wikipedia.org/wiki/Hooking (Retrieved June 2010).
- Wikimedia Foundation Inc. (June 10, 2010). *Wikipedia.* "Active Template Library." http://en.wikipedia.org/wiki/Active_Template_Library (Retrieved July 2010).
- Wikimedia Foundation Inc. (June 30, 2010). *Wikipedia.* "Namespace (Computer Science)." http://en.wikipedia.org/wiki/Namespace_%28computer_science%29 (Retrieved July 2010).
- Wikimedia Foundation Inc. (July 7, 2010). *Wikipedia.* "WQL." http://en.wikipedia.org/wiki/WQL (Retrieved July 2010).
- Wikimedia Foundation Inc. (July 14, 2010). *Wikipedia.* "JavaScript Engine." http://en.wikipedia.org/wiki/JavaScript_engine (Retrieved July 2010).

TREND MICRO INC.

10101 N. De Anza Blvd.
Cupertino, CA 95014

US toll free: 1 +800.228.5651
Phone: 1 +408.257.1500
Fax: 1 +408.257.2003

www.trendmicro.com