

# SOFTWARE RESTRICTION POLICIES IN WINDOWS XP

*John Lambert*

Microsoft Corporation, One Microsoft Way, Redmond WA 98052, USA  
Tel +1 425 705 9689 • Fax +1 425 936 7329 • Email: johnla@microsoft.com

## ABSTRACT

*Software Restriction Policies is a new feature in Windows XP and Windows .NET Server 2003 that prevents unwanted software from running on a system. It can be used to provide increased control over software that runs on desktop systems, delivering improved manageability and lower support costs. This paper provides an overview of the feature and describes the motivation and thinking behind its design. It also highlights how the feature can be used in virus prevention.*

## TERMINOLOGY

This paper refers to the Windows 2000, Windows XP, and Windows .NET Server 2003 operating systems as part of the Windows NT family. Any of the aforementioned systems may be generically referred to as “NT” distinguishing them from the Windows 9x line of operating systems.

## BACKGROUND

Shortly after Windows 2000 shipped, a number of efforts were taken at Microsoft to improve the resilience of the Windows platform to hostile code. Microsoft studied the effects of the Remote Explorer worm and analyzed the problems it presented. Up until that point, few viruses and worms had been specifically written to run on a Windows NT based operating system. The Remote Explorer worm was something of a wake up call to the designers of the security system. Even with the advances NT provided over Win9x, including an account based security model, discretionary access control, separation between kernel and user mode applications, and system file protection, in many cases NT based systems were just as susceptible to viruses and worms as Win9x based systems. This was due to a number of reasons.

- All applications run under full privileges of the user. If the user has access to the server housing source code, then an instant messenger process or an e-mail process run by the user also has access to the source control server.
- Applications can silently perform operations using the user’s privileges
- Many users log on as Administrator, download, and run untrusted code
- It is easy for hostile code to masquerade as any well-known application executable

One solution was to regulate the running of untrusted code. This paper describes this solution by surveying the existing features, technology, and community efforts Microsoft has in the anti-virus space, presenting some findings during the development of the Software Restriction Policies feature, and finally, providing an overview of the feature and some analysis of it in anti-virus scenarios.

## SUMMARY OF EXISTING ANTI-VIRUS EFFORTS

The virus problem was analyzed in three dimensions: detection, recovery, and prevention.

## DETECTION

Windows has a strong anti-virus ISV community with many products that specialize in virus detection. Within that community, Microsoft plays a unique role in providing technical information, creating interfaces into Microsoft products to enable virus detection, and offering testing and training opportunities with the Windows Development team at ISV “plug fests” and other

events. In particular, Microsoft provides a filesystem filter interface which is used by anti-virus products to intercept filesystem activity. This is the interface used to provide “real time” virus scanning. Office and Internet Explorer support an anti-virus API allowing such events as “document open” and “ActiveX download” to be hooked. Exchange provides an anti-virus API, EVSI, that allows a virus scanner to scan messages before a client sees them. In the area of virus detection, we also provide documentation and assistance with file formats and obscure system behavior. We provide education, training, and testing opportunities in the form of “plug fests” for filesystem filter drivers and “anti-virus community briefings” through a forum called MVI. At plug fests, anti-virus ISVs come to the Microsoft campus, meet and collaborate with Microsoft kernel and filesystem developers, and run their filter drivers through a battery of comprehensive tests. The collaboration between the filesystem group at Microsoft and the anti-virus developer community has resulted in dramatic improvements in reliability. A few years ago, a plug fest would turn up hundreds of serious issues in the kernel mode filter drivers. At the latest plug fest in 2002, only a handful of issues were found. As a result, the number of blue screens caused by bugs in the AV filter drivers has dropped to the noise level. The filesystem team has an improved filter model in development that is designed to improve the performance, reliability, and synchronization issues seen today. It reduces the amount of kernel mode code an anti-virus ISV has to write to hook file activity. Microsoft is also collaborating with anti-virus ISVs to improve the performance of their real-time scanners against the NetBench benchmark by identifying bottlenecks and lock contention. In summary, Microsoft acts as an enabler in the detection space. Microsoft also shares crash data with anti-virus ISVs. Crash data is the information sent to Microsoft via the system and program error reporting feature. This helps ISVs uncover bugs in their software.

## **RECOVERY**

In the area of recovery, Microsoft has a number of technologies designed to help systems recovery from a variety of problems, ranging from a bad software install to a hard disk failure. None of these technologies are expressly designed for virus recovery, but they provide a set of capabilities that are available when recovering from a virus. System Restore, added in Windows XP, monitors changes to the system and some application files and automatically creates easily identified restore points. These restore points allow you to revert the system to a previous time. They are created daily and at the time of significant system events (such as when an application or driver is installed). Automated System Recovery (ASR) restores all disk signatures, volumes, and partitions on the disks required to start the computer. ASR then installs a simplified version of Windows and automatically starts a restoration using the backup created by the ASR wizard. In Windows .NET Server 2003, a feature called “shadow copies” uses the Volume Shadow Copy Service (VSS) to take regular shadow copies on a server that can be browsed by a client for manually initiated copy-undelete or rollback.

## **PREVENTION**

In the area of virus prevention, the goal is to prevent new, unknown viruses by limiting or eliminating the ability of the virus to propagate or to damage the system. Previous efforts in this

space included Authenticode, HTML script sandboxing and Internet Explorer zones, Outlook attachment control, the Outlook Object Model Guard, Office macro signing, and Windows scripting host security. Authenticode is described in a following section in this paper.

The Internet Explorer browser sandboxes HTML based scripts and Java applets by enforcing the limits defined in the appropriate IE zone. Internet Explorer divides the Internet into zones, so that Web sites can be added to a zone and assigned a suitable security level. The security capabilities of a zone can be customized as needed.

With Outlook 2000 SR-1 and Outlook XP, Microsoft introduced the following prevention improvements:

- E-mail attachment security prevents users from accessing several file types when sent as e-mail attachments. Affected file types include executables, batch files, and other file types that contain executable code often used by malicious hackers to spread viruses. An administrator can configure certain file types to be either level 1 or level 2.
  - o Level 1 security files are inaccessible and hidden from users.
  - o Level 2 security files must be saved to disk before they can be accessed.
- The Outlook Object Model Guard prompts users with a dialog box when an external program attempts to access their Outlook Address Book or send e-mail on their behalf, which is how worms such as ILOVEYOU spread.
- Heightened Outlook default security settings increase the default Internet security zone setting within Outlook from “Internet” to “restricted sites.” In addition, active scripting within restricted sites is disabled by default. These security features help protect users from many viruses that are spread by means of scripting.

Outlook XP SP-1 introduced a new feature that permits users to configure Outlook to read all non-digitally-signed e-mail or non-encrypted e-mail messages in plain text format. This can be used to reduce the number of scripting vulnerabilities and web bugs found in HTML mail messages.

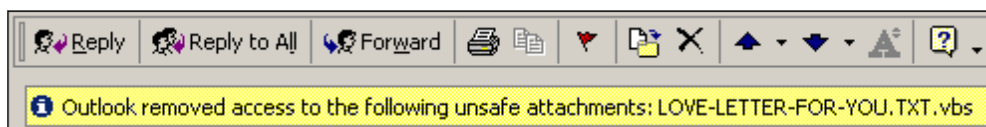


Figure 1 Preventing access to attachments in Outlook

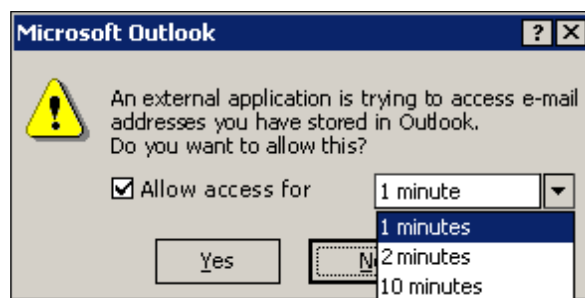
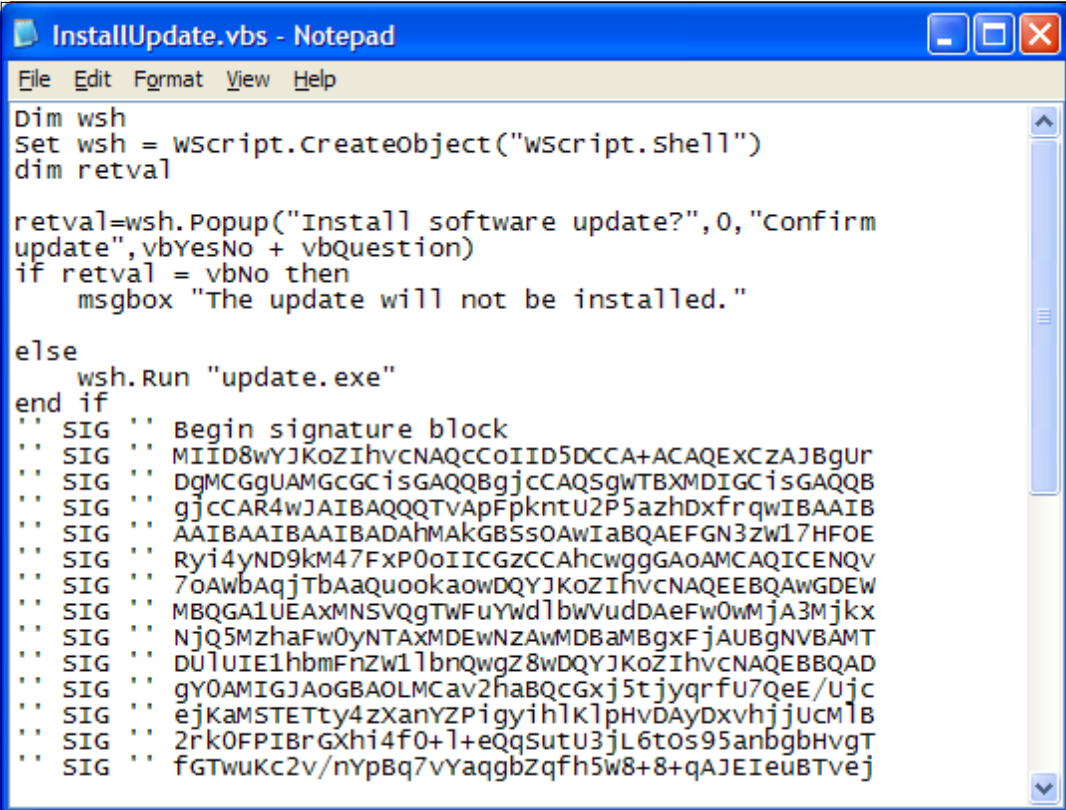


Figure 2 Outlook Object Model Guard

Microsoft Office documents have supported macro signing since 1997. If a document contains unsigned macros, the macro is either disabled or the user is warned about the presence of the unsigned macro, depending on the security level chosen. Recent versions of Microsoft Office default to the highest security level.

The Windows Scripting Host (WSH) serves as the runtime for VB Script (.VBS) and JScript (.JS) code. The highly damaging ILOVEYOU worm was written in the VB Script language. With the latest version of the script runtime, script files can be signed. WSH has a trust policy which can be set to not allow any unsigned scripts or to warn before running an unsigned script.



```

Dim wsh
Set wsh = wscript.CreateObject("wscript.shell")
dim retval

retval=wsh.Popup("Install software update?",0,"Confirm
update",vbYesNo + vbQuestion)
if retval = vbNo then
    msgbox "The update will not be installed."
else
    wsh.Run "update.exe"
end if
'' SIG '' Begin signature block
'' SIG '' MIID8wYJKoZIhvcNAQcCoIID5DCCA+ACAQEXCZAJBgUr
'' SIG '' DgMCGgUAMGcGCisGAQQBgjCCAQSGWTBMDIGCisGAQQB
'' SIG '' gjcCAR4wJAIBAQQQTvApFpkntU2P5azhDxfrqwIBAAIB
'' SIG '' AAIBAAIBAAIBADAHMAKGBSsoAwIaBQAIEFGN3Zw17HFOE
'' SIG '' Ryi4yND9km47Fxp0oIICGZCCAhcwggGAoAMCAQICENQV
'' SIG '' 7oAwbAqjTbAaQuookaowDQYJKoZIhvcNAQEEBQAwGDEW
'' SIG '' MBQGA1UEAxMNSVQgTWFuYWdlbnVudDAeFw0wMjA3Mjkx
'' SIG '' NjQ5MzhaFw0yNTAxMDEwNzAwMDBaMBGxGjAUBGNVBAAMT
'' SIG '' DUlUE1hbmFnZW1lbnQwZ8wDQYJKoZIhvcNAQEBBQAD
'' SIG '' gY0AMIGJAoGBAOLMCav2haBQCgXj5tjyqrfU7QeE/UjC
'' SIG '' ejkaMSTETty4zxanyZPigyih1k1phvDAYDxvhjjucM1B
'' SIG '' 2rk0FPiBrGXhi4f0+l+eQqsutU3jL6tOs95anbgbhvgT
'' SIG '' fGTWuKc2v/nYpBq7vYaqgbzqfh5w8+8+qAJEIEuBTvej

```

Figure 3 A digital signature in a VB Script

## AUTHENTICODE AND CODE SIGNING

Introduced in the mid '90's, Authenticode was a way to tie a software publisher identity to a distribution unit of code with high integrity. The two major problems addressed by Authenticode were secure software delivery and strong publisher identity. Secure software delivery means that client software such as the IE browser can detect if code has been modified since it has been Authenticode signed. To this end, Authenticode used a standard PKCS#7 signature to protect the integrity of the code. Publisher identity was addressed by a partnership between Microsoft and Verisign to issue digital IDs for code signing purposes. Verisign provided the issuance and credential verification behind the certificates. The program has expanded to include more certificate authorities than simply Verisign.

Any time digital certificates are used, care must be taken to verify the legitimacy of the identity

in the certificate. Revocation checking is built-in to the Authenticode verification process. In Windows XP and IE 6, revocation checking for Authenticode is turned on by default. In early 2001, some digital identities were issued to a party fraudulently claiming to be Microsoft employees. These digital identities were revoked and no evidence of their use was ever found. This event should not be construed as a failure in code signing methodologies. Instead, Public Key Infrastructure (PKI) accommodates for mistakes like fraudulent issuance or key compromise by providing a revocation mechanism. In general, the value of code signing is largely dependent on the rigor used in certificate issuance, operational protection of the private keys, and timeliness of the revocation information. For customers that choose to operate their own PKI, the PKI in Windows XP and Windows .NET Server 2003 provides customers with a wide variety of options to increase the integrity of code signing. Users can choose to sign their own code by using certificates issued from a Windows .NET Server 2003 Certificate Authority. They can tie certificate issuance to Kerberos-based domain authentication or require that the certificate request be signed with another strong credential, such as a smart card. Software based keys can be marked as non-exportable, meaning the keys cannot be removed from the machine where the enrollment was performed. They can also be restricted to being issued to a smart card which provides a high degree of assurance because use of the certificates is tied to a physical device and knowledge of the PIN associated with the card.

A number of technologies have incorporated digital signatures as a way for users to have more control and integrity over the code they are distributing. Portable Executable files such as EXEs, DLLs, and OCXs can be signed. Windows Scripting Host file types such as .VBS and .JS added digital signature support natively in Windows XP and this support can also be installed on downlevel systems. The Windows Installer format, MSI, can also be digitally signed. Signing tools are available from the Microsoft MSDN web site. Code signing can provide a strong way to identify code where the level of assurance can be adjusted to suit the needs of the user's environment.

It is important to note what digital signatures do not guarantee. Authenticode signatures do not guarantee the signed content is safe, non-malicious, or virus free. There are some provisions in the Authenticode agreement that software publishers should not knowingly sign malicious or infected content, but the technology provides for no assurance of this. The strength of the Authenticode signature relies on the diligence of the bearer of the certificate to use the private key in a secure way.

## **REGULATING CODE EXECUTION**

Having surveyed the landscape in prevention technologies, a core capability was found to be missing. With Windows 2000, administrators can control the resources a user can access, their software installation settings, their application preferences, and the user rights and password policy. However, there was no mechanism in the operating system to control what software ran on a user's desktop. If a user has write access to someplace on the disk, such as their profile or "My Documents" folder, the user can copy an infected file to the system and unknowingly execute it. To this end, the Software Restriction Policies team designed a system for regulating the execution of code.

The Software Restriction Policies team broke down the area of virus prevention into two approaches: partial trust models and binary trust models. In a partial trust model, an application is given a subset of the user's full capabilities. This is often referred to as a sandbox. There may be multiple sandboxes corresponding to differing levels of trust, each with its own set of permissions. Sandboxing is used in a couple of Microsoft products, such as the Internet Explorer browser which sandboxes HTML script and the Common Language Runtime (CLR) which provides a rich code access security model for managed code.

## **SANDBOXING WIN32**

The sandboxing approach was investigated during the development of Software Restriction Policies. Sandboxing is used successfully in some Microsoft products and technologies. However, these technologies were built with the notion of partial trust from the beginning. The overwhelming majority of software for Windows writes to the Win32 API. Win32 has no notion of a sandbox. Indeed, only the NT line of Windows operating systems has a notion of a security model at all. The large community of Windows developers two to three years ago wrote and tested their application only on Win9x, which has no security model. This set of circumstances had a number of effects.

Retrofitting a sandbox onto Win32 has been attempted a number of times by Microsoft. Each one of these attempts led to less than satisfactory results for various reasons. In one case, the sandboxing work was ahead of its time, as very few customers were using NT or the NTFS filesystem upon which the sandbox depended for access control. In other later cases, it was clear that applications and the operating system itself were simply not designed to run with an arbitrary set of capabilities. The user experience in Windows is largely built and tested around three classes of accounts: limited user, power user, and administrator. Testing showed that sandboxing these accounts further resulted in bizarre or inconsistent behavior. In one case, the File Open dialog produced a series of inexplicable error messages ("Failed to Initialize File Dialogs. Change the Filename and try again") when it failed to access its default folder location, the "My Documents" folder. This kind of erratic error handling results from a lack of test coverage at these custom sandbox levels. The user experience was not tested to perform in a graceful way at arbitrary privilege levels. Application performance under these levels varied from acceptable to failure (access violation). This result is not surprising. Ignoring additional sandbox levels, consider just the standard limited user account built into Windows 2000. The Application Compatibility team at Microsoft specializes in understanding, correcting, and reporting application compatibility issues discovered when migrating applications from one version of Windows to another. The application compatibility labs showed that during testing for Windows XP, 80% of applications experience some kind of failure when run under the limited user account. With the hurdles faced in just making the limited user experience work well, retrofitting arbitrary sandbox levels seemed unlikely to result in more than a haphazard collection of application failures and bizarre user experiences.

Also, the behavior of the sandbox is unclear. For example, running e-mail attachments as an unprivileged guest account resulted in users being unable to print their attachments or save the document to the desktop or their "My Documents" folder. They were not able to print because the guest account was not authorized to access network printers. They were not able to save the

document to their folders because the folders were set only to allow their user account to have access and not the guest account. It was not clear that the sandbox could provide additional safety without radically altering well entrenched user behavior.

From a technological point of view, it was found to be difficult to achieve the isolation desired. The sandbox was built using the control mechanisms of the Windows NT platform. This included technologies such as Access Control Lists (ACLs), restricted tokens, and job objects. Even with these mechanisms, elevation was still possible using Windows messages to remote control a higher privileged process or by attacking higher privileged processes run by the same user or by methods such as out-of-proc COM activation. Additionally, even if the sandbox levels could be built successfully, a seamless sandboxing approach might result in little practical security because of the discipline needed by users to avoid the kind of data sharing inherent in a single desktop model. Hostile code would likely find a way to tamper with shared data and elevate their privileges.

The Software Restriction Policies team concluded that for sandboxing to be successful at the Win32 layer, it would require a heavy level of customization and tweaking in customer environments—severely limiting deployment to the degree that Microsoft believed the feature would have little success.

## **BINARY TRUST**

An alternative to the partial trust model is the binary trust model. Binary trust models determine if the software can run at all. When the software runs, it runs with the full capabilities of the user. The Software Restriction Policies feature utilizes a binary trust model for Win32 software. Essentially, a sandboxing approach is used by systems when software cannot be identified as trusted. Given a means to identify software as trusted, one can make a binary decision about whether it should run. Software Restriction Policies provides ways to identify software as trusted or untrusted.

## **PROBLEMS TO SOLVE**

1. The enforcement must be policy based. Administrators create the policy, defining which applications are trusted and which are not. The policy is enforced at execution time. Users are not consulted with a “do you want to run” style dialog.
2. The policy must apply to more than just EXEs. The definition of what constitutes software is ambiguous. The Software Restriction Policies feature integrates with the Window scripting host to provide control over VB Script and JScript. It also integrates with the Windows Installer feature to provide control over which packages can be installed. The Software Restriction Policies feature also has an API allowing other runtimes to integrate with it.
3. The policy must be scalable. The policy must be manageable in a large enterprise with a wide variety of machine types and application roles. The Software Restriction Policies feature leverages Active Directory and Group Policy for manageability. The policy is stored in a Group



Policy object (GPO). A GPO can be delegated to an appropriate administrator. A GPO can be targeted at a specific set of machines or user accounts. The group policy infrastructure provides a mechanism that updates all client machines if a change is made to the policy in the GPO. The Software Restriction Policies feature also can be used in a standalone non-domain joined environment.

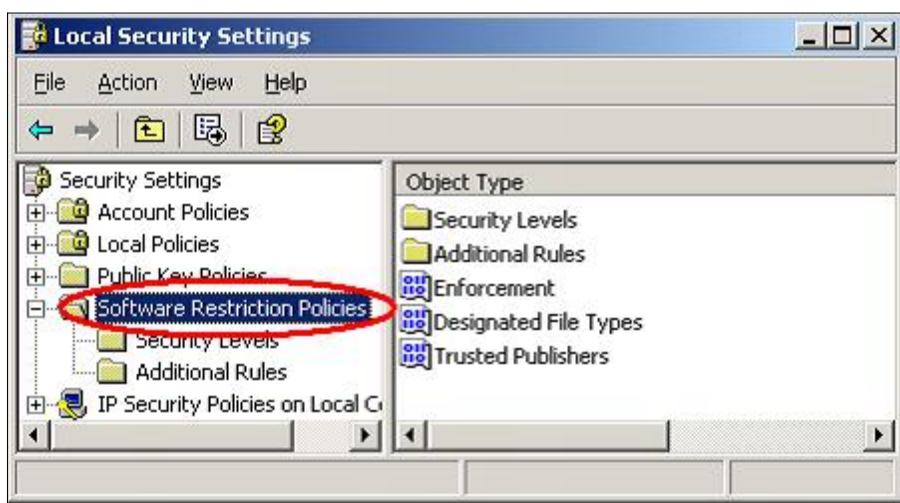
4. The policy must be flexible. Some users want to just prohibit unauthorized scripts. Others want to regulate ActiveX controls or lockdown the machine. The Software Restriction Policies feature works for any of these needs.
5. The policy must enable cryptographically strong ways to identify software. The policy allows for identification of software by a hash or signature. These are discussed later.

## DESCRIPTION OF THE FEATURE

A policy is created using the MMC Group Policy snap-in. A policy consists of a default security level specifying whether programs are allowed to run, and rules that define exceptions to the default security level. The default security level can be set to Unrestricted or Disallowed—essentially run or don't run.

Setting the default security level to Unrestricted allows an administrator to define the set of programs that are not allowed to run. A more secure approach is to set the default security level to Disallowed and specify only the programs that are known and trusted to run.

The policy can be configured on a per machine basis or distributed through Group Policy. By using Group Policy, a policy can be managed at scale. Different machines or users can have different policies according to their needs by leveraging the targeting and filtering capabilities of Group Policy. Group Policy also ensures that the policy configured at deployment time remains consistent through the lifetime of the machine. Should a change to a policy be necessary, this update is made to the policy in the Group Policy Object (GPO) and this change is replicated to all managed machines.



## IDENTIFICATION METHODS

### DEFAULT SECURITY LEVEL

There are two ways to use Software Restriction Policies:

- If an administrator knows all of the software that should run, then a policy can be applied to control execution to a list of trusted applications.
- If all the applications that users might run are not known, then administrators can disallow undesired applications or file types as needed.

### HASH RULE

A hash is a cryptographic fingerprint that uniquely identifies a file regardless of where it is accessed or what it is named. A hash rule is a rule, or exception to the default security level, that identifies software based on its hash. An administrator may not want users to run a particular version of a program. This may be the case if the program has security or privacy bugs, or compromises system stability. With a hash rule, software can be renamed or moved into another location on a disk, but it will still match the hash rule because the rule is based on a cryptographic calculation involving file contents.

### CERTIFICATE RULE

A certificate rule specifies a code-signing, software publisher certificate. For example, a company can require that all scripts and ActiveX controls be signed with a particular set of publisher certificates. Certificates used in a certificate rule can be issued from a commercial certificate authority (CA) such as VeriSign, a Windows 2000/Windows .NET Server 2003 PKI, or a self-signed certificate.

A certificate rule is a strong way to identify software because it uses signed hashes contained in the signature of the signed file to match files, regardless of name or location. If you wish to make exceptions to a certificate rule, you can use a hash rule to identify the exceptions.

### PATH RULE

A path rule can specify a folder or fully qualified path to a program. When a path rule specifies a folder, it matches any program contained in that folder and any programs contained in subfolders. Both local and UNC paths are supported. A path rule can use environment variables. Since path rules are evaluated in the client environment, the ability to use environment variables (for example, %WINDIR%) allows a rule to adapt to a particular user's environment. A path rule can incorporate the '?' and '\*' wildcards, allowing rules such as "\*.vbs" to match all Visual Basic® Script files.

Many applications store paths to their installation folders or application directories in the Windows registry. You can create a path rule that looks up these registry keys. For example, some applications can be installed anywhere on the file system. These locations may not be easily identifiable by using specific folder paths, such as C:\Program Files\Microsoft Platform SDK, or environment variables, such as %ProgramFiles%\Microsoft Platform SDK. If the program stores its application directories in the registry, you can create a path rule that will use the value stored in the registry, such as %HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\PlatformSDK\Directories\Install Dir%. This type of path rule is called a “registry path rule”.

When there are multiple matching path rules, the most specific matching rule takes precedence.

## ZONE RULE

A zone rule can identify software from the Internet Explorer zone from which it is downloaded. Currently this applies to only Windows Installer (\*.MSI) packages. It does not apply to software downloaded in Internet Explorer.

## RULE PRECEDENCE

Rules are evaluated in a specific order. The rules that more specifically match a program override rules that more generally match a program.

1. Hash rule
2. Certificate rule
3. Path rule
4. Internet zone rule
5. Default rule

Table 1 and the following examples illustrate how rules are processed when a program is started.

Table 1. Understanding Rule Precedence

Default Security Level: Unrestricted		
Hash Rules		
Rule 1	Hash of pagefileconfig.vbs	Disallowed
Certificate Rules		
Rule 2	IT Management Certificate	Unrestricted
Path Rules		
Rule 3	C:\Windows\System32\*.VBS	Unrestricted
Rule 4	*.VBS	Disallowed
Rule 5	C:\Windows	Unrestricted

Program being started: C:\WINDOWS\SYSTEM32\EventQuery.vbs

This program matches the following rules:

- Rule 3 because it is a .vbs file in the System32 folder.
- Rule 4 because it has a .vbs extension.
- Rule 5 because it is stored in a subfolder of the Windows directory.

Rule 3 is the most specific match for this program. Because Rule 3 has a security level of Unrestricted, the program is allowed to run.

Program being started: C:\WINDOWS\SYSTEM32\pagefileconfig.vbs

This program matches the following rules:

- Rule 1 because the hash in the rule matches the hash of the file.
- Rule 3 because it is a .vbs file in the System32 folder.
- Rule 4 because it has a .vbs extension.
- Rule 5 because it is stored in a subfolder of the Windows directory.

Rule 1 is the most specific match for this program. Because Rule 1 has a security level of Disallowed, the program is disallowed.

Program being started: \\LOGIN\_SRV\Scripts\CustomerScript1.vbs

This program matches the following rules:

- Rule 2 because it is digitally signed by the certificate belonging to the customer's IT management group.
- Rule 4 because it has a .vbs extension.

Rule 2 is the most specific match for this program. Because Rule 2 has a security level of Unrestricted, the program is allowed to run.

Program being started: C:\Documents and Settings\user1\LOVE-LETTER-FOR-YOU.TXT.VBS

This program matches Rule 4 because it has a .vbs extension.

Rule 4 is the most specific match for this program. Because the Rule 4 has a security level of Disallowed, the program is disallowed.

## **WHAT'S NOT COVERED**

Software Restriction Policies do not apply to the following:

- Drivers or other kernel mode software.
- Any program run by the SYSTEM, Local Service, or Network Service account.
- Macros inside of Microsoft Office 2000 or Office XP documents.

A software restriction policy will not prevent worms that exploit product vulnerabilities, as in the

cases of Code Red and Nimda. Keeping a machine up-to-date with security updates is, as always, critical. However, a software restriction policy may help mitigate the payload of a worm that exploits product vulnerabilities.

## **STRENGTHS AND WEAKNESSES**

This section analyzes the strengths and weaknesses of the rule types. The analysis also includes interesting capabilities when combined with other rule types.

## **ANALYSIS OF HASH RULES**

Hash rules can be both strong and weak. A hash is a cryptographic calculation over the bytes in the binary. A hash has the following properties: fixed length output, small changes in the source material result in large changes in the calculated hash, and low likelihood of two different inputs resulting in the same hash (a collision). What this means for Software Restriction Policies is that disallowing software with a hash rule can be problematic. While it allows a user to identify a program regardless of its location or filename, any change to the program binary results in a different hash. Changing one bit in the file or performing an operation like rebasing, re-linking, or digitally signing the program all result in variances in the hash. Allowing a program with a hash rule is quite a strong way to identify trusted code, but any changes to the file mean the administrator has to change the software restriction policy to allow it to run. This can result in a manageability problem for frequently changing files.

Hash rules can be paired with other rules to make them stronger. For example, it is possible to disallow a specific ActiveX control by hash. If the control is modified to change the hash value, this change breaks the digital signature on the ActiveX control and it will not be loaded by the browser. Hash rules also have the advantage that they don't require a user to operate a PKI or maintain secure code signing keys.

## **ANALYSIS OF PATH RULES**

Path rules are an interesting class of rules, perhaps prone to the most misuse. They do not require use of a PKI or a code signing certificate deployment. However, they require ACLs to be used effectively. If a rule allows anything in C:\windows to run and a user has write access to that folder, the path rule might as well not exist. Path rules can be used to allow network resources to run, as in the case of logon script shares or common software images. It is crucial that the shares housing the logon scripts or install images be protected by a proper ACL. Path rules also allow environment variables, such as %PROGRAMFILES%. However, these should only be used in special circumstances. Environment variables are not secure, and any user who can load a command prompt can temporarily redefine them. Path rules also support "registry path rules". Registry path rules have the advantage of being protected by ACLs. In many cases, path rules using environment variables can usually be replaced with a registry path rule. Beware registry path rules that have embedded environment variables. Path rules can be used to prevent files from running that have double extensions, such as GreatPicture.JPG.EXE. Many viruses and worms use double extensions as a way to trick the user into double-clicking on an unsafe file

type. They take advantage of the fact that some Windows systems hide known extensions by default. One can create a rule such as “\*.???.EXE” which will disallow all EXE programs that have a three letter double extension.

Path rules also support wild cards, such as \*.VBS, allowing powerful combinations when used with certificate or hash rules. This case is considered in the “Block Malicious Scripts” section below.

## **ANALYSIS OF ZONE RULES**

Zone rules in the current version of Software Restriction Policies are only useful in certain circumstances. They are only used by the Windows Installer system when it processes MSI packages. So, they can be useful in certain software distribution scenarios.

## **ANALYSIS OF CERTIFICATE RULES**

Certificate rules are more manageable than hash rules. The advantage of a certificate rule is that an administrator can make one rule and then use the signing certificate to authorize programs by applying a digital signature as often as they need. It serves as a useful way to group code that you want to trust. Certificate rules have some useful combinations with hash rules, as mentioned above. They also have some useful combinations with path rules.

For example, one can disallow an entire class of programs, \*.VBS, for example, but trust a particular certificate. An administrator then just needs to sign all of the VB Script files they use. This simple combination of a path and certificate rules, users are inoculated against an entire class of script based viruses, but one can still use VB Script for systems management.

There are several operational concerns with the trustworthiness of certificate rules. For example, the private key associated with the certificate must be protected. It is inexpensive to use smart cards with PKI today and smart cards are well integrated in the Windows operating system. Secondly, should a compromise of a private key occur, revocation and the timely publication of revocation information is essential to the secure operation of the system. Revocation checking for signed code is turned on by default in Windows XP, a departure from previous operating system releases. Disallowing by certificate rule faces a similar problem as hash rules. It is entirely possible to remove a digital signature from signed code. Obviously, this can circumvent the disallow rule. However, in the case of ActiveX, removing the signature also results in the browser rejecting the code.

## **USE IN PREVENTION**

### **BLOCK MALICIOUS SCRIPTS**

An organization wants to be protected from script-based viruses. The LoveLetter virus, technically called a worm, was estimated to have caused between \$6 and \$10 billion in damage. This worm, which has more than 80 variants, continues to be encountered frequently.

The LoveLetter worm, written in the Visual Basic Script language (VBS), is encountered as LOVE-LETTER-FOR-YOU.TXT.VBS. A software restriction policy blocks this worm simply by disallowing any .vbs file from running.

However, many organizations use VBS files for systems management and logon scripts. Blocking all VBS files from running protects an organization, but a VBS can no longer be used for legitimate purposes. A software restriction policy overcomes this handicap by blocking the undesirable VBS, while allowing legitimate ones to run. This policy also disallows some executables that utilize “double extensions”. The rule “\*.???.exe” prevents programs like foo.mp3.exe and foo.jpg.exe from running. More variations on this rule are possible to broaden the classes of double extensions blocked.

This policy can be created using the rules in Table 2.

Table 2. Rules for Blocking Malicious Scripts

Default Security Level: Unrestricted	
Path Rules	
*.VBS	Disallowed
*.VBE	Disallowed
*.JS	Disallowed
*.JSE	Disallowed
*.WSF	Disallowed
*.WSH	Disallowed
*.???.exe	Disallowed
*.???.scr	Disallowed
Certificate Rules	
IT Department Certificate	Unrestricted

This policy prevents all scripting files associated with the Windows Scripting Host from running, except those that are digitally signed by the IT Department certificate.

## SOFTWARE LOCKDOWN FOR LINE OF BUSINESS PC

In some cases, an administrator may want to manage all of the software that runs on a machine. This is because even when users have insufficient rights to replace system files or files in shared folders such as Program Files, if they have a place on the file system they can write to, then they can also copy a program there and start it up.

Viruses contracted this way can damage the system by modifying operating system settings and files; they can also cause great damage by misusing the user’s privileges. For example, mass-mailer worms can be spread by accessing the user’s address book and sending mail. Even normal users on a system are vulnerable to this kind of attack.

As long as users are not administrators on their local machines, the policy in Table 3 protects them from accidentally running malicious code. Because users cannot modify the contents of the Program Files or Windows folders, they can only run software installed by an administrator.

Table 3. Policy for Managing all Software on a Machine

Default Security Level: Disallowed	
Apply software restriction policies to the following users:	
All users except administrators	
Path Rules	
%HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\SystemRoot%	Unrestricted
%HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\ProgramFilesDir%	Unrestricted

This policy disallows all software on the user's machine, except that installed in the Windows directory, Program Files directory, or their respective subfolders. It does not apply to administrators.

If a user receives a virus attachment in an e-mail, for example WORM.vbs, the mail program will copy it to the profile directory (%USERPROFILE%) and launch it from there. Because the profile directory is not a subfolder of the Windows folder or the Program Files folder, programs launched from there will not run.

If all the programs a user needs are not installed in the Windows directory or Program Files, or there are programs in those folders that the administrator does not want the user running, the administrator can make additional exceptions as shown in Table 4.

Table 4. Exceptions for Managing all Software on a Machine

Path Rules	
%HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\SystemRoot%regedit.exe	Disallowed
%HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\SystemRoot%cmd.exe	Disallowed
\\CORP_DC_??\scripts	Unrestricted
%HKEY_LOCAL_MACHINE\SOFTWARE\ComputerAssociates\InoculateIT\6.0\Path\HOME%	Unrestricted

The effects of these exceptions are:

- Both the command prompt (cmd.exe) and the registry editor (regedit.exe) are disallowed.
- An exception is created to allow logon scripts to run on the user's machine.
- The use of the "?" wildcard allows the rule to match \\CORP\_DC\_01, \\CORP\_DC\_02, and others.
- A registry path rule is added that allows the anti-virus software on the machine to run.

## ONLY TRUSTED PROGRAMS FOR ADMINISTRATORS

While a typical Software Restriction Policy will be designed to limit the code that users run, another use for a Software Restriction Policy is to limit the software that administrators run. The administrator account is the most privileged account on the system. In order to prevent acciden-



tal abuse, one can design a Software Restriction Policy that limits the administrator account to only running trusted administrative applications. Applications such as e-mail, web browsing, instant messaging, and productivity applications such as spreadsheets or database applications can be prevented from running under an administrative account. Instead, administrators must log on with their unprivileged user account to access those functions.

## **SUMMARY**

With the Software Restriction Policies feature in Windows XP and Windows .NET Server 2003, Microsoft has added increased management capabilities to users. The feature can be used to prevent unwanted applications from running or for virus prevention.

## **REFERENCES AND LINKS**

Software Restriction Policies Whitepaper

<http://www.microsoft.com/WindowsXP/pro/techinfo/administration/restrictionpolicies/default.asp>

File System Filter IFS Kit

<http://www.microsoft.com/ddk/>

.NET Framework Developer's Guide to Code Access Security

<http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpguide/html/cpconcodeaccesssecurity.asp>

Outlook Security Features

<http://msdn.microsoft.com/library/default.asp?url=/library/en-us/xpreskit/html/outg03.asp>

New Windows Scripting Host (WSH) Security Features

<http://msdn.microsoft.com/msdnmag/issues/01/04/WSH/WSH.asp>

The Smart Card Deployment Cookbook

<http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/prodtech/smrtcard/smrtcdb/default.asp>