Arming SMB's Against Ransomware Attacks

GIAC GCIA Gold Certification

Author: Tim Ashford, tcashford@gmail.com
Advisor: Stephen Northcutt

Abstract

Ransomware has become one of the most serious cyber threats to small and medium businesses today. A recent variant permanently deletes files within one hour of infection. The situation grows increasingly dire: the FBI even encourages victims to make payment, though there is still no guarantee that owners will recover their data (ICIT Fellows, 2016). Despite such threats, small and medium enterprises can follow recommended best practices to mitigate this risk. Businesses with tighter budgets and fewer security team members can adopt many of the protections available to the largest enterprises. The most important recommendation is the use of application whitelisting. In Windows environments, this can be accomplished through free tools within Active Directory. Other options will also be discussed, as well as a brief discussion of the future of ransomware.

1. Introduction

Ransomware is probably the most serious cyber threat for small and medium businesses today. For threat actors, ransomware is a logical choice. Vulnerabilities abound, and plenty of victims are willing to pay up. Risks of prosecution are low, while financial rewards are easy, and can even be automated. Worst of all, the ransomware landscape has become a “competitive” black market. Many attackers contribute to the improvement of this evolving threat, constantly tweaking platforms to yield more consistent returns on investment. In only a few short years, ransomware has evolved from threatening to lock a victim’s files, to encrypting them, and most recently either deleting or publishing them. Where do small and medium businesses (SMB’s) stand, in terms of the dangers from such a landscape? It turns out that they are very much in the crosshairs. There has been a steady increase over the last five years in cyber-attacks on businesses with fewer than 250 employees, and 43 percent of all attacks in 2015 targeted small businesses (Symantec, 2016, p. 44). One of the most closely watched cyber reports each year states that ransomware is the second most prevalent variety of malware (Verizon, 2016, p. 46) Together, these statistics mean that SMB’s will experience an increasing and targeted threat from ransomware. The purpose of this paper is to help them prepare.

Is ransomware uniquely threatening? Technologically speaking, no. Ransomware uses the same vectors of attack and the same vehicles of infection as other malware. SMB's will find this good news since they can thwart these attacks with a thoughtful implementation of several key cyber disciplines. This paper will recommend the highest priority prevention techniques, which promise to go a long way toward mitigating the risk.

Here are some tasks for tackling ransomware mitigation, together with lessons learned and future direction.

1. Utilize AppLocker, a built-in application whitelisting tool included with our recommended Windows software. To help SMB's, a step by step guide will walk through a sampling of AppLocker settings that can thwart recent variants of ransomware.

2. Standardize on both hardware and software. For Windows environments, utilize the benefits of Windows Server 2012 R2, and Windows 10 Enterprise for desktops. The combination of the latest Microsoft Windows server and desktop software can be used to block many types of infection.
3. Understand AppLocker's limitations. Learn how to monitor AppLocker through logging, and investigate ways to review logs intelligently.
4. Gain familiarity with PowerShell. Understand how PowerShell relates to security. Explore the possible link between ransomware and PowerShell. Become educated about ways to restrict PowerShell's use, and stay informed about the evolving landscape.

These steps take into consideration the need to contain cost and complexity, which is crucial for SMB's.

2. Focus on Preparation

This discussion focuses on AppLocker preparations to prevent ransomware. We will assume that SMB's have already tackled several other best practiceshave already been tackled. In my previous Gold paper, the basics of Group Policy for SMB's were covered, including directions on how to implement least privilege by removing administrative rights at the desktop. A tested backup strategy should be in place as well. Policies should already govern the use of web browsing and email, the two most common vectors of ransomware infection (rcole, jdelio, 2016). Regarding email, a report published in June of 2016 stated that 93% of all phishing emails are now infected with ransomware (Korolov, 2016). Small companies with policies in place governing email will already be blocking many file attachments, scripts, and Office macros, as well as ads in the browser. Users should be receiving periodic training to spot phishing attacks.

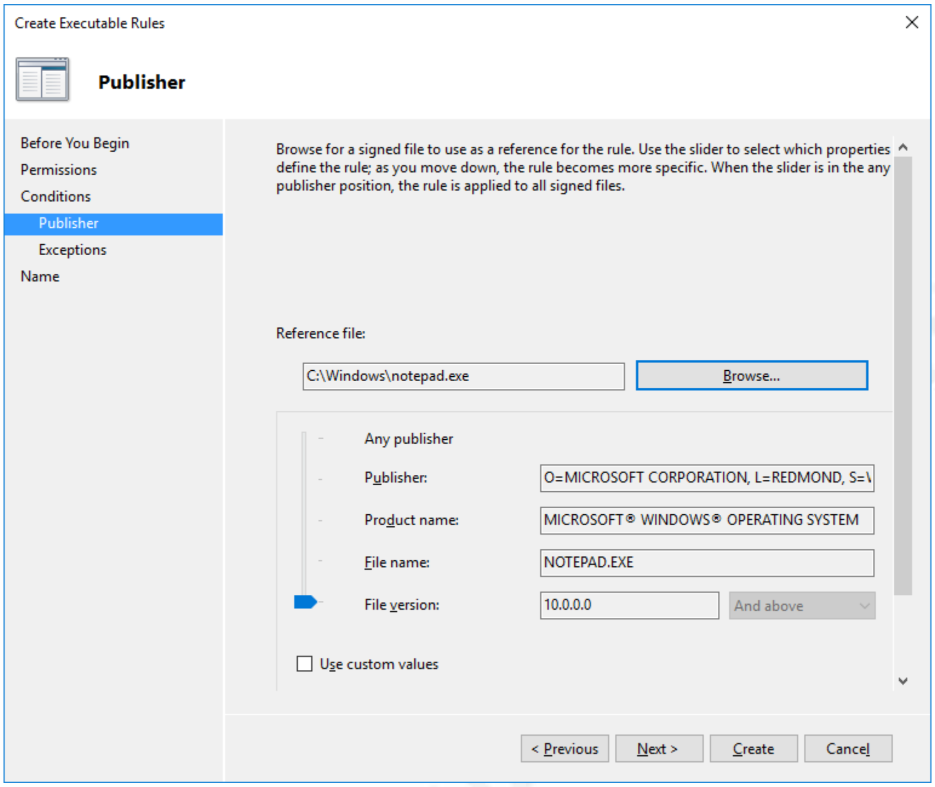
3. Dive into AppLocker

The Windows Server environment is where we will begin our preparations against ransomware-style threats. Specifically, we will look at AppLocker, an essential native Windows security tool. AppLocker whitelists applications so that only those apps with the IT department's approval are allowed to run. Small and medium businesses can easily manage an inventory of applications in current use. AppLocker has this capability built in. But before we walk through the creation of a sample AppLocker policy and use it to inventory existing software, we need to know the system requirements of AppLocker. At the server level, we need a minimum of Windows Server 2008 R2, though Windows Server 2012 R2 is preferred (more on that later). At the desktop level, the enterprise variants of Windows 7, Windows 8, or preferably Windows 10, in the Enterprise versions, are required to enforce and manage AppLocker rules.

The following is a sample AppLocker policy. We will use it to help us audit existing software, and thus provide an inventory of what is already running. From the Group Policy Management console, create a new Group Policy Object under Computer Configuration→Policies→Windows Settings→Security Settings→Application Control Policies→AppLocker. Under this menu, right click on Executable Rules, and choose "Create New Rule…" The first configuration page will be Permissions. From here, start by selecting the defaults of "Allow" and the "Everyone" group. On the next configuration page, called Conditions, we have three main ways of identifying applications: By publisher, by path, or by file hash.

3.1 Executables Identified by Publisher

Executables identified by publisher are the first and most obvious way of filtering. For example, we may be interested in allowing any software published by Microsoft. This can be accomplished using an executable rule that filters on Microsoft as the publisher. On the next screen, titled Publisher, browse to a single executable on the local drive, known to be published by Microsoft. This can serve as our reference file for all Microsoft-published files. On the following screen, a slider bar appears, with four main field options. See Figure 1 below for an example, using Notepad (the built-in text editor signed by Microsoft):



Browse for a signed file to use as a reference for the rule. Use the slider to select which properties define the rule; as you move down, the rule becomes more specific. When the slider is in the any publisher position, the rule is applied to all signed files.

Reference file: C:\Windows\notepad.exe

Any publisher

Publisher: O=MICROSOFT CORPORATION, L=REDMOND, S=\

Product name: MICROSOFT® WINDOWS® OPERATING SYSTEM

File name: NOTEPAD.EXE

File version: 10.0.0.0 And above

☐ Use custom values

Figure 1

With the slider bar at the bottom, our options are the most granular; in other words, we can allow only the selected file version of the particular program (in this case, Notepad.exe, version 10.0). If we move the slider up one more notch, we now allow any Microsoft program named Notepad. Sliding all the way to the top fulfills our original goal of allowing all Microsoft-published program files. This will populate the bottom three fields with wildcard values (*). After choosing Next, our final option is to identify any exceptions. The last screen allows us to name this rule. Best practice is always to label our Group Policy rules as intuitively as possible.

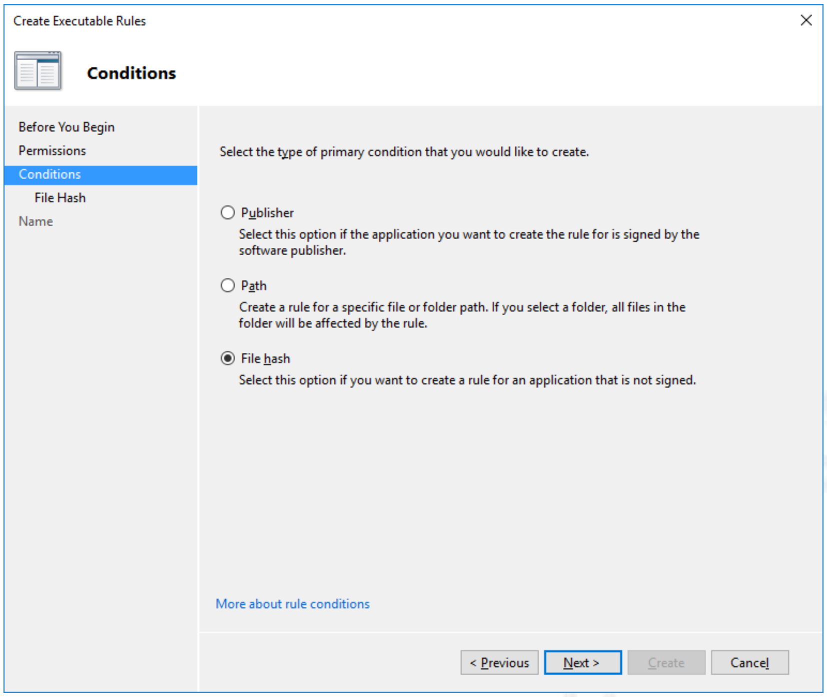
3.2 Executables Identified by Path

In some cases, it is better to allow files to run based on their drive location, rather than their publisher. In this case, on the Conditions screen, we choose Path. The next few screens are self-explanatory and offer the option to whitelist by either files or folders. Anticipating the need for system files and other mission critical programs to run, Microsoft implemented some default rules. To activate these, right click on "Executable

Rules" below AppLocker and choose "Create Default Rules." Three rules will then populate: First, all programs within the Windows folder will be allowed to run; second, those within the Program Files – or Program Files (x86) – directories are allowed to run; and finally, any member of the local Administrators group can run any executable file. AppLocker defines an executable file as anything with a .exe or .com extension (Microsoft, 2012). Incidentally, all three default rules are based on path or file location. If this final default rule is allowed, anyone who gains administrative privileges to the local machine can run anything – including malicious programs aimed at pivoting to other domain resources. Such privileged credentials will reside in memory until the system is rebooted, making the system potentially vulnerable to certain credential harvesting attacks (Mimikatz). It may be wise to consider creating a more limited administrative account at the local level.

3.3 Executables Identified by File Hash

Our last way to whitelist by executable is to use a cryptographic hash generated by the host system. Our previously mentioned publisher rules were also dependent on hashes, or digital signatures, but in that case, the hash was created by a trusted software publisher. In situations where trusted file hashes are not available, we can generate them, with one caveat: Each time the referenced executable gets patched or updated, the hash rule must be updated manually. To create a hash rule is straightforward: After creating a new rule, on the Conditions page, choose the third option (File hash). See Figure 2.



Select the type of primary condition that you would like to create.

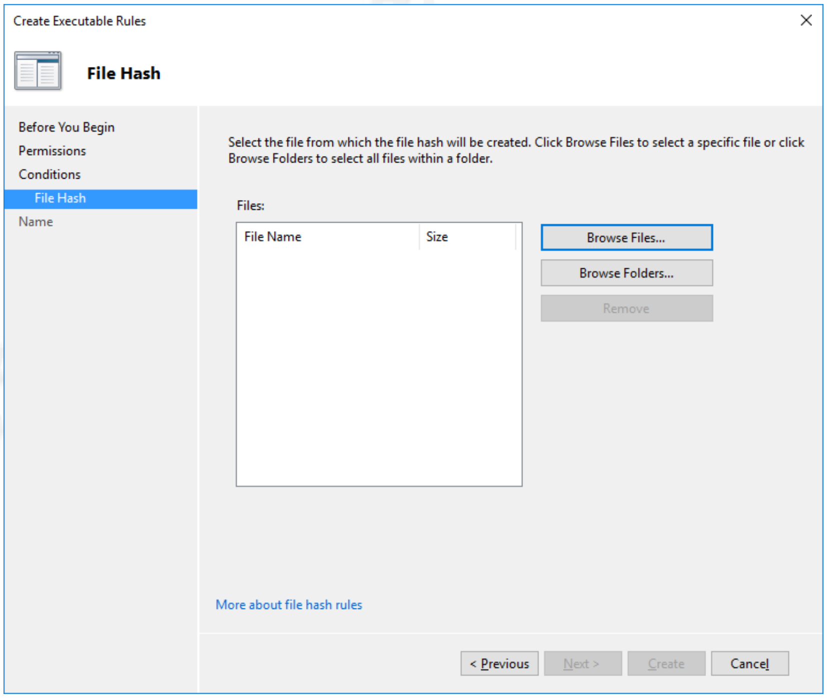
○ Publisher
Select this option if the application you want to create the rule for is signed by the software publisher.

○ Path
Create a rule for a specific file or folder path. If you select a folder, all files in the folder will be affected by the rule.

● File hash
Select this option if you want to create a rule for an application that is not signed.

Figure 2

On the next screen, click Browse Files, as in Figure 3.



Select the file from which the file hash will be created. Click Browse Files to select a specific file or click Browse Folders to select all files within a folder.

Files:

File Name	Size

Figure 3

Locate the file you desire to hash, and follow the prompts. Remember to document these actions.

3.4 Managing Windows Installer Rules

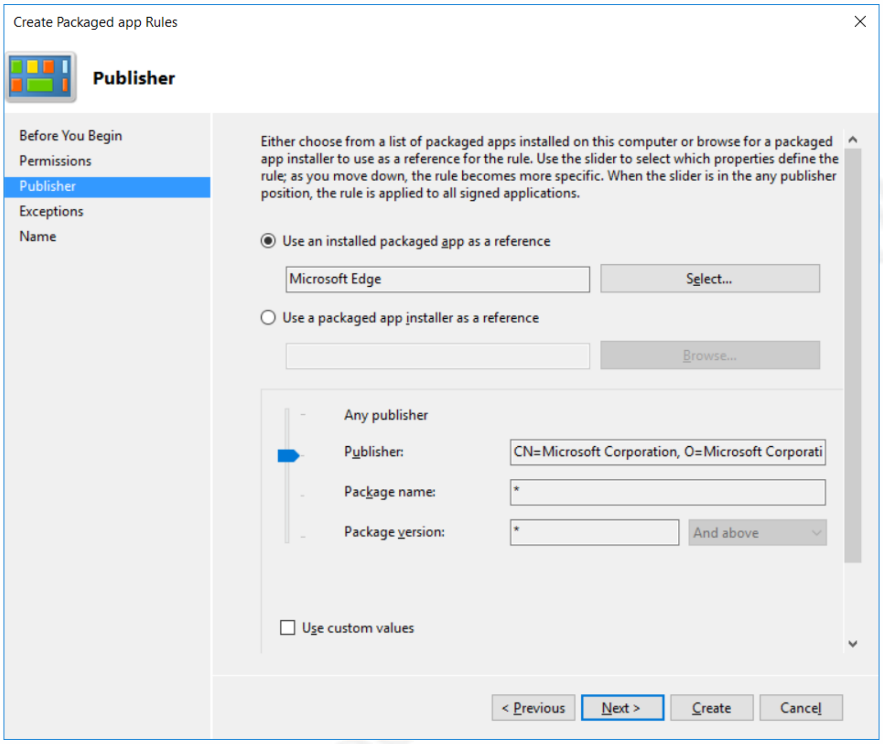
The second of four main sections of AppLocker rules pertains to Windows installer files. These MSI files, as they are also known, not only differ by file extension from executables, but also function differently, in that they can be deployed directly by Group Policy (whereas an exe file cannot). Delving into this subsection of AppLocker reveals the same process as whitelisting executables: We have settings for publisher, path and file hash, with the same configuration steps and options. Not surprisingly, there are also default rules that allow all MSI files assigned to the Program Files and Windows directories, and allow local Administrators to run any MSI's.

3.5 Managing Script Rules

Our third major set of options in AppLocker concerns scripts. These follow the same conventions as executables and installer files. Interestingly, the installation of default rules does not whitelist batch files or scripts that Active Directory has authorized in the NETLOGON folder. Add these manually. We will develop a deeper understanding of AppLocker's settings and limitations concerning scripts later.

3.6 Managing Packaged App Rules

Packaged apps are the fourth and final category of AppLocker settings. With the introduction of Windows 8 and the Windows Server 2012 operating systems, packaged apps have been Microsoft's effort to make app deployment simpler to manage. Specifically, packaged apps, unlike classic Windows applications, share a common set of components. These include installer as well as executable files, among others, and therefore, they need only a single AppLocker rule for the purpose of whitelisting, which need only be a publisher rule (Lich, Packaged Apps and Packaged App Installer Rules in AppLocker, 2016). Since all packaged apps must be signed, these apps will never require either custom hashes or manual updating. For our recommended Windows 10 deployment, it is imperative to right click and Create Default Rules, since packaged apps form such an integral part of the utility of the operating system. In addition, a default Microsoft rule should be added, by referencing a common app such as Microsoft Edge, as seen in Figure 2:


Create Packaged app Rules

Publisher

Either choose from a list of packaged apps installed on this computer or browse for a packaged app installer to use as a reference for the rule. Use the slider to select which properties define the rule; as you move down, the rule becomes more specific. When the slider is in the any publisher position, the rule is applied to all signed applications.

● Use an installed packaged app as a reference

Microsoft Edge

○ Use a packaged app installer as a reference

Any publisher

Publisher: CN=Microsoft Corporation, O=Microsoft Corporati

Package name: *

Package version: * And above

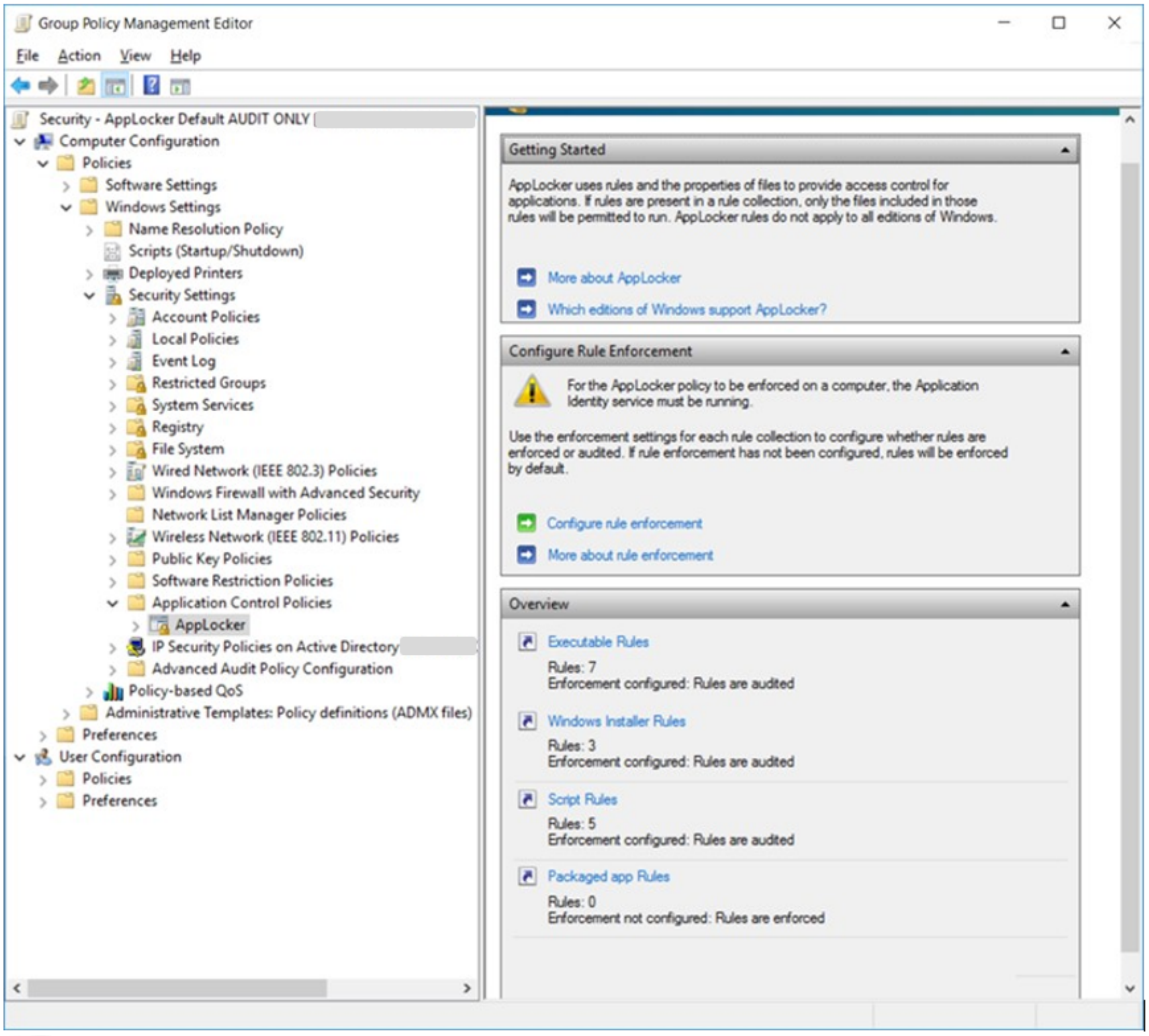
□ Use custom values

Figure 4

Packaged apps require little configuration, so the interface offers only publisher-related options.

3.7 Auditing with AppLocker

With some default rules configured, we need to take advantage of AppLocker's built in auditing feature. This way, without enforcing any rules, we can find out which legitimate apps around the network will require whitelisting. We may even discover some unauthorized installations. Because we are deploying AppLocker through Group Policy, our endpoint Windows machines will have the Application Identity Service turned on by default, which is required for AppLocker to function. Back at the main AppLocker configuration page (see Figure 3), each section has the option for either enforcement or auditing. After clicking "Configure rule enforcement," each sub-heading offers the option to either enforce or audit. Once auditing is turned on, and with the Application Identity service running, the local system Even Viewer will indicate which apps, installers or scripts would be prevented by our current AppLocker policy when enforced.


Group Policy Management Editor
File
Action
View
Help
Security - AppLocker Default AUDIT ONLY [redacted]
Computer Configuration
Policies
Software Settings
Windows Settings
Name Resolution Policy
Scripts (Startup/Shutdown)
Deployed Printers
Security Settings
Account Policies
Local Policies
Event Log
Restricted Groups
System Services
Registry
File System
Wired Network (IEEE 802.3) Policies
Windows Firewall with Advanced Security
Network List Manager Policies
Wireless Network (IEEE 802.11) Policies
Public Key Policies
Software Restriction Policies
Application Control Policies
AppLocker
IP Security Policies on Active Directory [redacted]
Advanced Audit Policy Configuration
Policy-based QoS
Administrative Templates: Policy definitions (ADMX files)
Preferences
User Configuration
Policies
Preferences

Getting Started

AppLocker uses rules and the properties of files to provide access control for applications. If rules are present in a rule collection, only the files included in those rules will be permitted to run. AppLocker rules do not apply to all editions of Windows.

More about AppLocker

Which editions of Windows support AppLocker?

Configure Rule Enforcement

For the AppLocker policy to be enforced on a computer, the Application Identity service must be running.

Use the enforcement settings for each rule collection to configure whether rules are enforced or audited. If rule enforcement has not been configured, rules will be enforced by default.

Configure rule enforcement

More about rule enforcement

Overview

Executable Rules
Rules: 7
Enforcement configured: Rules are audited

Windows Installer Rules
Rules: 3
Enforcement configured: Rules are audited

Script Rules
Rules: 5
Enforcement configured: Rules are audited

Packaged app Rules
Rules: 0
Enforcement not configured: Rules are enforced

Figure 5

In a small to medium business, the process of auditing apps in an environment should only take a short time. If we have done a recommended inventory of all software and hardware already (20 Critical Controls, 2016), we have a good idea of what is already authorized to run. By sampling the Event Logs of a few selected workstations, we can verify which specific rules to add to our existing defaults. To find the appropriate entries in the Event Viewer, navigate to Applications and Services

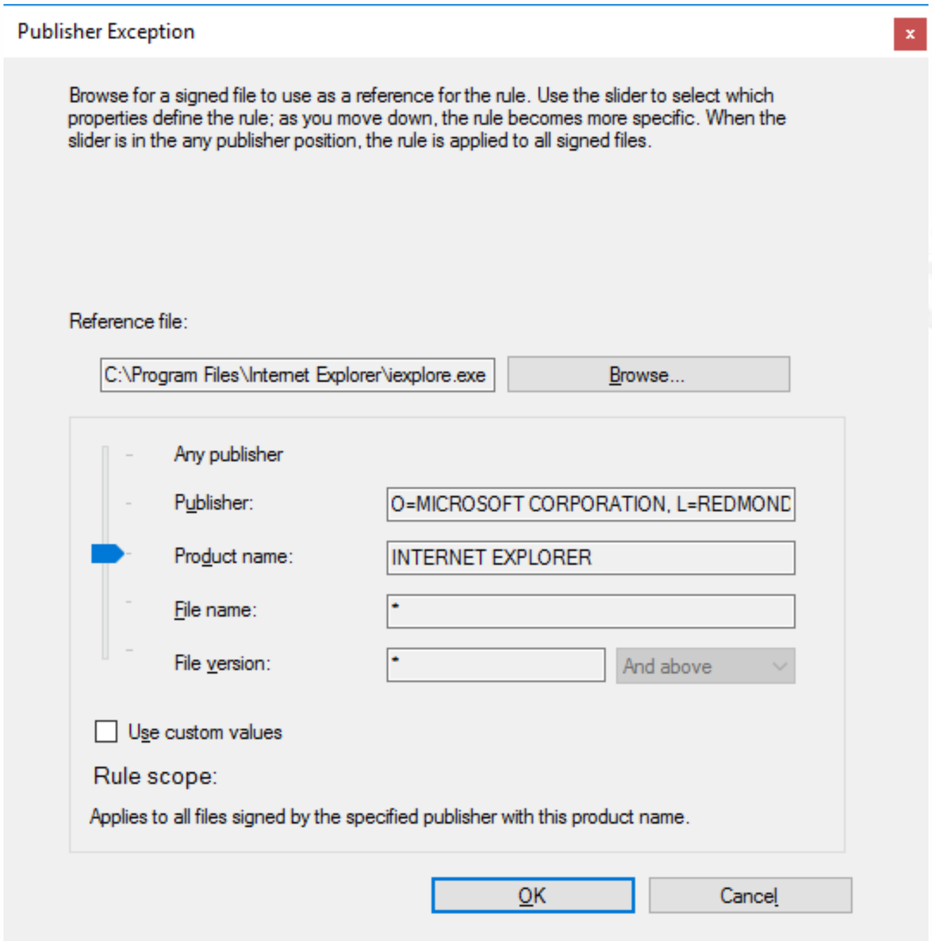
Author: Tim Ashford, tcashford@gmail.com

Logs→Windows→AppLocker, and see the various options. In most cases, our apps will have been published with digital signatures, or will reside in an expected location like Program Files. For those few apps that run from the %UserProfile% directory, exceptions will need to be made. But many legitimate vendors install apps here with digital signatures, and an increasing number like Dropbox are creating packaged apps for Windows 10, further simplifying application lockdown.

3.8 When Auditing Reveals the Need for Adjustment

Often the audit process will surface an app that should be allowed to run. How do we whitelist our discovery? Usually, we need to ask ourselves a few questions: Do all users need access to this app or only some? We need to apply the policy of least privilege, assigning the right to run it to only a group of users, rather than to everyone (see the next paragraph regarding Scope). Is the app digitally signed? We would likely then follow our steps in utilizing a Publisher rule. Does the app run from somewhere in the local user's directory? We need to consider the implications of whitelisting this path since other more dangerous apps may seek to run from the same location. In any case, whitelisting is a straightforward process for well-known apps.

Speaking of making adjustments, what if we are concerned about restricting otherwise legitimate apps from running? For example, some built-in processes in Windows may broaden the attack surface while offering no added utility in the current environment. Internet Explorer is an example: Many known vulnerabilities in IE have existed over the years, and recently, Microsoft has focused its security efforts on the newer Edge browser for Windows 10. Because of the way that whitelisting works in AppLocker, we can make an exception to deny this one application, while whitelisting all other Microsoft apps (Lich, Administer AppLocker, 2016). To apply this specific example, edit the existing AppLocker policy that allows signed Microsoft apps. Right click on Executable Rules, and click through until reaching the Exceptions page. Click Add, and browse to the Internet Explorer application under Program Files, or Program Files (x86). On the next screen, move the slider bar up to Product Name, so that any version of IE will be selected. See Figure 4. Click through the next screens to finish editing the policy object.



Publisher Exception

Browse for a signed file to use as a reference for the rule. Use the slider to select which properties define the rule; as you move down, the rule becomes more specific. When the slider is in the any publisher position, the rule is applied to all signed files.

Reference file:

C:\Program Files\Internet Explorer\iexplore.exe

Any publisher

Publisher: O=MICROSOFT CORPORATION, L=REDMOND

Product name: INTERNET EXPLORER

File name: *

File version: * And above

□ Use custom values

Rule scope:

Applies to all files signed by the specified publisher with this product name.

Figure 6

What if we have a select number of users that require IE, but we want to block the app for everyone else? Some older applications utilize IE for reporting, as an example. AppLocker applies the deny action before the allow action (Microsoft, 2012), so we need to create a separate rule for denying just this app, and under Security Filtering (see below), restrict this app for all users except those that need to use it.

3.9 Enforcing AppLocker Rules

AppLocker rules can now be enforced. This assumes that users do not have local Administrator rights on their machines, and User Account Control settings are in place. Both of these options can be enforced through Group Policy. As with all Group Policy Objects, our AppLocker object must be linked to the correct Organizational Units. To roll out AppLocker enforcement, return to the main AppLocker configuration page, shown in Figure 3. This time, choose "Enforce rules" for each of the four sections. If there is concern that mission-critical systems may encounter unexpected behavior, create a new Global Security Group in Active Directory, named AppLocker, and add workstations one at a time to this group. Then, on the main Group Policy Management page, with the appropriate AppLocker Group Policy Object selected, choose the Scope tab. Next, in the bottom pane, under Security Filtering, click on the Add button. Here the Security Group can be selected, which insures that only the systems or users that have been added to that group will be affected by the AppLocker policy enforcement. Consequently, if a system encounters issues after enforcement is invoked, it can easily be removed from the Security Group, without affecting enforcement on other systems where the policy has rolled out successfully.

4. Invest in Windows Server 2012 and Windows 10 or later

Utilize the benefits of the most current Windows versions. At the time of this writing, they are Windows Server 2012 R2 and Windows 10. We have already seen that some server and desktop security settings are only available in Windows Server 2012 R2 and Windows 10. For example, packaged apps cannot be administered by Group Policy in Windows Server 2008 R2. Although Windows 8.1 allows for packaged apps, Windows 10 is a far better option from a security standpoint. While beyond the scope of this paperfully, Windows 10 offers several differentiating features, including sophisticated biometric authentication with Windows Hello and a more secure default browser in Microsoft Edge. In terms of protecting from ransomware, the Edge browser does not allow the use of Browser Helper Objects or ActiveX controls. Microsoft reports that Edge utilizes a form of sandboxing that limits damage from security holes in Adobe Flash or Java. More specifically, ransomware threats from these vulnerable extensions should be separated from the browser itself (Brian Lich et. al., 2016). It may be a challenge up front, but SMB's are in a unique position to standardize on this best practice deployment at the server and desktop level. They typically do not need to support a high number of disparate systems.

5. Limitations of AppLocker Whitelisting

Planning and implementing AppLocker as described here will help SMB's go a long way toward blocking most ransomware threats. Even after following these best practices, however, AppLocker has its limitations. For example, any ransomware that finds a whitelisted installer path open to a Standard User could potentially run. This means that if a particular path is whitelisted, the current user could install programs there, even without Admin rights. Any systems with lingering local Administrator rights can potentially override all other AppLocker settings since we allow local Administrators to execute anything by default. Perhaps most importantly, AppLocker policies that whitelist Microsoft applications by publisher allow systems to be weaponized by using built-in, legitimate programs. Take the recent discovery of the regsvr32 vulnerability. Regsvr32 is a significant built-in Windows utility that registers and unregisters DLL's (Regsvr32). A security researcher discovered that a single line of code entered into this signed Windows executable can deliver encrypted outbound traffic capable of pulling malicious scripts back to the host environment (Smith, 2016). There are no published ransomware attacks that utilize this yet. Even with AppLocker configured correctly, this vulnerability could be weaponized. It is worth noting that an attacker would already need to be present on a system in order to execute this outbound attack (Windows AppLocker Bypassed to Execute Remote Scripts, 2016). Once present, "unregister" the code block using regsvr32, even as a Standard User, and the malicious traffic is activated (Smith, 2016). The reason this works so effectively with AppLocker in place is that regsvr32 is a trusted part of the Windows operating system. The researcher who discovered this hole added, "There's really no patch for this; it's not an exploit. It's just using the tool in an unorthodox manner. It's a bypass, an evasion tactic." (manishs, 2016). Once activated, it could readily pull ransomware files back to the host.

Another serious liability is PowerShell. PowerShell is an entire framework built into Windows for allowing administrators to automate an almost unlimited amount of tasks from a command shell. By default, it is turned on with every Windows system shipped since Windows 7. Many of the more advanced attacks seen in the last six years have been known to utilize PowerShell. The potential for harm goes beyond simply being signed by Microsoft: like the previous regsvr32 vulnerability, it can run powerful code without the need for specific files or scripts. To learn more about ways that PowerShell has been weaponized, see sites like harmj0y's PowerUp Guide, Using the Veil Framework from the Varis Group, and the PowerShell Empire tutorials on YouTube. For SMB's, PowerShell is likely not needed at the typical user's workstation, and should simply be restricted. Unfortunately, there is no way to turn off PowerShell completely, as we will discuss in a moment.

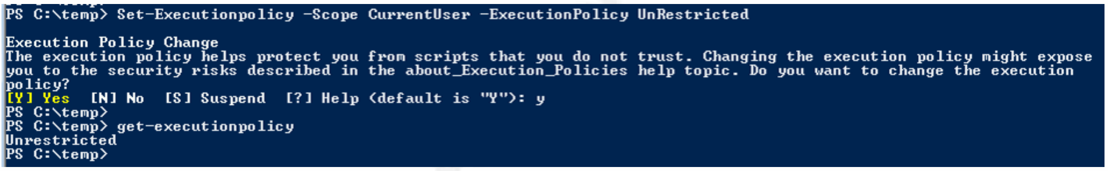
First, we should ask, has PowerShell ever been used in ransomware attacks? In fact, a recent report details an attack called PowerWare, which invokes PowerShell to run a script that encrypts local drives (Rico Valdez, Mike Sconzo, 2016). The attack could be thwarted by some controls already discussed: in particular, this attack requires a user to click on a macro-enabled Word document. If macros were disabled – and Word macros serve little purpose in most current environments – the attack would be thwarted. But if the attack leveraged some other available attack mechanism, and PowerShell were unrestricted at the endpoint, it would still evade our current AppLocker policy efforts. More needs to be done to control PowerShell.

First, let's detail more fully the challenges of having a powerful built-in scripting framework like PowerShell residing on every Windows host. First of all, a compromised system running PowerShell can execute any command, making it even more powerful than the Windows GUI. For example, with PowerShell turned on, the language can interact with Active Directory, or run other programs; it can access other built-in Windows tools, like the .Net framework, system DLL's, and other Windows API functions (Graeber, Use PowerShell to Interact with the Windows API: Part 1, 2013); and finally, it can perform these types of actions without ever writing to disk, all by running in memory (Graeber, Exploiting Powershell's Features (Not Flaws), 2012).

Second, PowerShell itself is not a single executable: This means that PowerShell can be invoked several ways, making it harder to secure. "PowerShell can be executed through a custom coded executable (such as MyPowershell.exe). In fact, there are several current methods of running PowerShell code without Powershell.exe being executed." (Metcalf, Detecting Offensive PowerShell Attack Tools, 2016) To clarify, let's say we add an exception to the AppLocker rule that trusts Microsoft as a publisher. In that exception, we can explicitly deny either the file Powershell.exe or its path

(C:\Windows\System32\WindowsPowerShell\v1.0). These changes can easily be bypassed by renaming the executable file or moving it to a different location. In fact, being so closely integrated into the Windows operating system means that PowerShell can be invoked without the executable at all.

PowerShell's restriction against running scripts (called the Execution Policy) can also be bypassed easily. By default, .PS1 or .PSM1 files are disabled. But this can easily be changed. As one researcher explains, this was never intended as a security measure. End users can easily turn off this restriction. "Bypassing the PowerShell Execution Policy is as easy as asking." In the following screenshot in Figure 7, he shows us how easy it is to query Windows to turn off this restriction, allowing scripts to run (Metcalf, Detecting Offensive PowerShell Attack Tools, 2016).

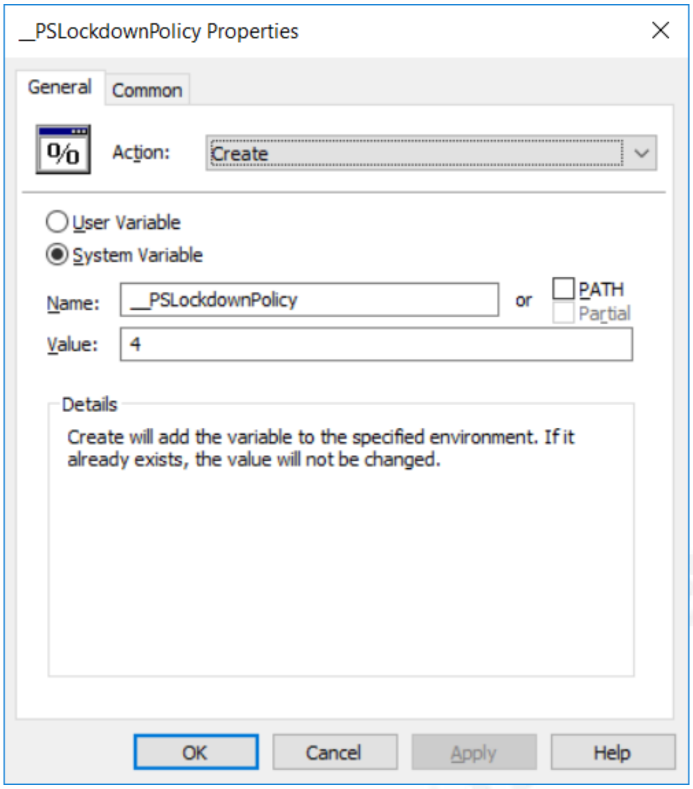


```
PS C:\temp> Set-Executionpolicy -Scope CurrentUser -ExecutionPolicy UnRestricted

Execution Policy Change
The execution policy helps protect you from scripts that you do not trust. Changing the execution policy might expose
you to the security risks described in the about_Execution_Policies help topic. Do you want to change the execution
policy?
[Y] Yes  [N] No  [S] Suspend  [?] Help (default is "Y"): y
PS C:\temp>
PS C:\temp> get-executionpolicy
Unrestricted
PS C:\temp>
```

Figure 7

Thankfully, with PowerShell version 5 implemented together with AppLocker on Windows 10, we can begin to set default limits on this range of functionality. More specifically, we can limit PowerShell's ability to have COM access, to use .Net libraries, and to make Win32 API calls. (Metcalf, Detecting Offensive PowerShell Attack Tools, 2016) One more setting needs to be added to Group Policy. In an AppLocker Group Policy Object, browse to Computer Configuration→Preferences→Windows Settings→Environment. Right click and choose New→Environment Variable. Follow the settings in Figure 8; note that the variable name includes two underscores (Metcalf, AD Security, 2016). This enforces a setting called Constrained Language. The reason that Constrained Language adds a layer of security to PowerShell is that it prevents PowerShell from leveraging some of its more dangerous capabilities. For example, with access to .Net libraries, PowerShell can communicate with Active Directory, grabbing information about users and groups (Perez, 2016).



__PSLockdownPolicy Properties

Action: Create

○ User Variable
● System Variable

Name: __PSLockdownPolicy or □ PATH □ Partial

Value: 4

Details

Create will add the variable to the specified environment. If it already exists, the value will not be changed.

Figure 8

PowerShell cannot be completely turned off. Even with available mitigations, the environment can eventually be exploited. Going forward, SMB’s need to explore the ability to log. Larger enterprises collect and collate logs (for example, from the Windows Event Viewer) into a central repository. This is currently a challenge for SMB’s with more limited resources, and especially manpower. They typically don’t have staff dedicated or skilled enough to comb through Event Logs for signs of shell activity. Will managed security service providers (MSSP’s) fill in the gaps, and offer an appropriate level of log collection and reporting capabilities? This could prove extremely timely for SMB’s, for two reasons. One, PowerShell has gradually evolved to provide sophisticated logging detail, including visibility into actual cmdlets and code. With PowerShell version 5, two new features in particular provide excellent visibility into the use of the framework. The first, called script block logging, reveals actual code that was run in the shell, even if an attempt was made to obfuscate it (Mackie, 2016). Conversely two, SMB's are often satisfied by knowing if PowerShell was invoked at all. For many smaller businesses, any use of PowerShell, especially from a local user, is considered

Author: Tim Ashford, tcashford@gmail.com anomalous behavior. This need for logging could be satisfied by MSSP's if handled intelligently.

5. Conclusion

Ransomware is on the rise. Many researchers predict that 2016 will see the Ransomware threat increase dramatically, with ransomware-as-a-service commoditizing easy money for shadowy threat actors around the world. We have only begun to see the use of some variants: RSA predicted the rise of “extortionware” this year, which exfiltrates company data, and then threatens to dump it online – small businesses, they noted, are particularly vulnerable (Arsene, 2016). Another report predicts a more latent data encryption process, locking files silently over time, until several generations of backups are affected, greatly reducing their efficacy (McAfee Labs 2016 Threats Predictions, 2016). In either case, the motive is simple: Get victims to pay. As we have seen, small and medium businesses can defend themselves with many of the same tools available to their larger counterparts. Mitigation begins with addressing the highest risks that have the greatest likelihood. By following widely accepted disciplines and best practices, we can position SMB’s to thwart the most common attacks. With a thoughtful AppLocker rollout, we can prevent the great majority of file-based infections.

We expect that future ransomware attacks may increasingly leverage frameworks like PowerShell as the PowerWare variant did. We looked at the liabilities inherent in PowerShell and are hopeful that the near future will offer SMB’s options to gain visibility into PowerShell activity, through centralized logging. With all reasonable preventions in place, SMB’s must remain vigilant, studying the landscape for both evolving threats and improving mitigations.

6. References

20 Critical Controls. (2016). Retrieved from Center for Internet Security: https://www.cisecurity.org/critical-controls.cfm

Arsene, L. (2016, March 15). *Ransomware Goes Corporate.* Retrieved from RSA Conference: http://www.rsaconference.com/blogs/ransomware-goes-corporate-in-2016

Brian Lich et. al. (2016, May 31). *Windows 10 Security Overview.* Retrieved from TechNet: https://technet.microsoft.com/en-us/itpro/windows/keep-secure/windows-10-security-guide

Graeber, M. (2012, October 13). *Exploiting Powershell's Features (Not Flaws).* Retrieved from Exploit Monday: http://www.exploit-monday.com/2011/10/exploiting-powershells-features-not.html

Graeber, M. (2013, June 25). *Use PowerShell to Interact with the Windows API: Part 1.* Retrieved from Hey, Scripting Guy! Blog: https://blogs.technet.microsoft.com/heyscriptingguy/2013/06/25/use-powershell-to-interact-with-the-windows-api-part-1/

ICIT Fellows. (2016). *The ICIT Ransomware Report.* Retrieved from icitech.org: http://icitech.org/wp-content/uploads/2016/03/ICIT-Brief-The-Ransomware-Report.pdf

J. Gomez, Genwei Jiang. (2015, September 22). *Malware with Your News? Forbes Website Victim of Malvertising Attack.* Retrieved from FireEye Blog: https://www.fireeye.com/blog/threat-research/2015/09/malvertising_attack.html

Korolov, M. (2016, 6 1). *93% of phishing emails are now ransomware.* Retrieved from CSO Online: http://www.csoonline.com/article/3077434/security/93-of-phishing-emails-are-now-ransomware.html

Lich, B. (2016, May 31). *Administer AppLocker.* Retrieved from TechNet: https://technet.microsoft.com/en-us/itpro/windows/keep-secure/administer-applocker

Lich, B. (2016, May 31). *Packaged Apps and Packaged App Installer Rules in AppLocker.* Retrieved from TechNet: https://technet.microsoft.com/en-us/itpro/windows/keep-secure/packaged-apps-and-packaged-app-installer-rules-in-applocker

Mackie, K. (2016, February 24). *Windows PowerShell 5.0 Expected To Add Improved Security.* Retrieved from Redmond Magazine: https://redmondmag.com/articles/2016/02/24/powershell-improved-security.aspx

manishs. (2016, April 22). *Core Windows Utility Can Be Used To Bypass Whitelisting.* Retrieved from Slashdot: https://it.slashdot.org/story/16/04/22/1513231/core-windows-utility-can-be-used-to-bypass-whitelisting

McAfee Labs 2016 Threats Predictions. (2016). Retrieved from McAfee Labs: http://www.mcafee.com/us/resources/reports/rp-threats-predictions-2016.pdf

Metcalf, S. (2016, May). *AD Security.* Retrieved from PowerShell Security: Defending the Enterprise from the Latest Attack Platform: https://adsecurity.org/wp-content/uploads/2016/05/BSidesCharm-2016-PowerShellSecurity-Defending-the-Enterprise-from-the-Latest-Attack-Platform-FINAL.pdf

Metcalf, S. (2016, April). *Detecting Offensive PowerShell Attack Tools.* Retrieved from Active Directory Security: https://adsecurity.org/?p=2604

Microsoft. (2012, June 21). *Executable Rules in AppLocker.* Retrieved from TechNet: https://technet.microsoft.com/en-us/library/ee460956(v=ws.11).aspx

Microsoft. (2012, June 21). *Understanding AppLocker Allow and Deny Actions on Rules.* Retrieved from TechNet: https://technet.microsoft.com/en-us/library/ee460955(v=ws.11).aspx

Mimikatz. (n.d.). Retrieved from Offensive Security: https://www.offensive-security.com/metasploit-unleashed/mimikatz/

Perez, C. (2016, March 21). *Writing a[n] Active Directory Audit Module - Getting Forest Info.* Retrieved from Dark Operator: http://www.darkoperator.com/blog/2016/3/15/4mbsgqwkek4doqahf4nwxb1k5kiok8

rcole, jdelio. (2016, March 7). *Best Practices for Ransomware Prevention.* Retrieved from Palo Alto Networks Live Community: https://live.paloaltonetworks.com/t5/Threat-Articles/Best-practices-for-ransomware-prevention/ta-p/74148

Regsvr32. (n.d.). Retrieved from TechNet: https://technet.microsoft.com/en-us/library/bb490985.aspx

Rico Valdez, Mike Sconzo. (2016, March 25). *PowerWare.* Retrieved from Carbon Black: https://www.carbonblack.com/2016/03/25/threat-alert-powerware-new-ransomware-written-in-powershell-targets-organizations-via-microsoft-word/

Smith, C. (2016, April 19). *Bypass Application Whitelisting Script Protections - Regsvr32.exe & COM Scriptlets (.sct files).* Retrieved from subTee: http://subt0x10.blogspot.ro/2016/04/bypass-application-whitelisting-script.html

Symantec. (2016). *Symantec Internet Security Threat Report 2016.* Retrieved from https://www.symantec.com/content/dam/symantec/docs/reports/istr-21-2016-en.pdf

Verizon. (2016, April). *Verizon Data Breach Investigations Report 2016.* Retrieved from Verizon Enterprise: http://www.verizonenterprise.com/verizon-insights-lab/dbir/2016/

Windows AppLocker Bypassed to Execute Remote Scripts. (2016, April 22). Retrieved from Security Week: http://www.securityweek.com/windows-applocker-bypassed-execute-remote-scripts