CIS Center for Internet Security®

CIS Controls

Living off the Land Attacks

PowerShell

July 2022

Acknowledgments

The Center for Internet Security® (CIS) would like to thank the many security experts who volunteer their time and talent to support the CIS Critical Security Controls® (CIS Controls®) and other CIS work. CIS products represent the effort of a veritable army of volunteers from across the industry, generously giving their time and talent in the name of a more secure online experience for everyone.

Editors	Ginger Anderson, CIS Valecia Stocchetti, CIS
Contributor	Jennifer Jarose, CIS

Contents

Introduction

The Center for Internet Security® (CIS) is the home of the CIS Critical Security Controls® (CIS Controls®), well-regarded and widely-used best practice recommendations that help enterprises focus their resources on the most critical actions to defend against the most prevalent real-life attacks. The community of volunteer experts who develop the CIS Controls come from a wide range of sectors, including defense, academia, government, healthcare, manufacturing, retail, and transportation.

In keeping with the CIS Controls' mission to provide prioritized, simplified, and relevant defensive guidance against real and current threats, CIS developed the CIS Community Defense Model (CDM). The CIS CDM takes a structured approach and looks at threat assessments published from multiple reliable sources to provide defenders prioritized best practices that defend against the top five most common attacks observed across the community. While Cyber Threat Actors (CTAs) use proprietary or commodity malware and other attack vectors to compromise a system, they also use and abuse tools that are native to the operating system, commonly referred to as Living off the Land (LotL) attacks. CIS recognizes the use of specific attack vectors, such as exploits of commonly used protocols, which is why CIS has also published several guides that provide more specific guidance for defenders to address LotL attacks.

Over the past few years, the community observed an increase in tactics, techniques, and procedures (TTPs) used by CTAs to evade detection and maximize the impact of their attacks. These TTPs include the use of post-exploitation tools (e.g., Mimikatz, CobaltStrike, or Metasploit®), the use of legitimate and allowlisted network administration tools (e.g., PowerShell® and PsExec), and the use of native protocols, including (Remote Desktop Protocol (RDP), Server Message Block (SMB), and Windows® Management Instrumentation (WMI)). The use of legitimate administrative tools, post-exploitation tools, and native protocols to target and exploit a network poses unique challenges to an enterprise looking to increase their defenses against cyber attacks.

CIS, as previously mentioned, has published several guides addressing some of the TTPs outlined in the previous paragraph. The guides provide prioritized best practice guidance to address some of the most commonly used vectors and exploited protocols to conduct attacks, such as RDP, SMB, and WMI. This guide, *"Living off the Land: PowerShell,"* is next in the series of LotL guides. It will cover the use of this legitimate network administration tool that is often used in cyber attacks, as well as provide guidance to defenders on how they can protect against a PowerShell-based attack.

PowerShell Overview

What is PowerShell?

PowerShell is a powerful tool used for task automation and configuration management that is built on the .NET framework. It consists of a command shell and scripting language and is supported on multiple platforms including Windows®, Linux®, and macOS®. PowerShell can also be used over port 5985 (Hypertext Transfer Protocol (HTTP)) and 5986 (Hypertext Transfer Protocol Secure (HTTPS)) as PowerShell remoting using Windows Remote Management (WinRM). PowerShell remoting allows a user to run commands remotely from another system, and is often used by administrators for Information Technology (IT) management.

The flexibility and wide-scale usage of PowerShell makes it particularly challenging to secure since it is a legitimate administrative tool that is often abused by CTAs and commonly used in LotL attacks. LotL attacks involve the use of existing tools and tactics on the targeted system or network to carry out an attack, rather than exploit a specific system or control weakness, rendering the attack difficult to detect and defend. Unfortunately, simply blocking the PowerShell executable is not a viable solution, nor is it effective, since PowerShell can be invoked in a number of ways without using the actual executable—and is often used this way. Additionally, a number of legitimate applications use PowerShell to perform everyday business functions. Therefore, a more strategic and multi-faceted approach is necessary to secure against an attack using PowerShell. It is important to note that throughout this *guide,* none of the recommendations are a single "silver bullet" to preventing a successful attack using PowerShell. Instead, taking a defense-in-depth approach will help to secure PowerShell.

Benefits of Using PowerShell

PowerShell is a robust tool that helps IT professionals automate a range of tedious and time-consuming administrative tasks as well as find, filter, and export information about a system on a network through combining commands, called cmdlets, and creating scripts.

The text-based command-line interface allows administrators the ability to achieve more granular control over system management. With PowerShell, a user can improve access to WMI and Control Object Model (COM) to fine-tune administrative management. For those tasked with managing Active Directory (AD), PowerShell automation is extremely helpful for executing management tasks such as adding or deleting accounts, editing groups, and creating listings to view specific types of groups or users. For those administrators that need to repeatedly run command sequences for system configurations, PowerShell offers the Integrated Scripting Environment (ISE) which allows users to develop scripts as command collections with the ability to include logic necessary for execution. Additionally, the use of cmdlets, PowerShell's basic single-function command, is a powerful capability that allows an administrator to combine cmdlets, use them within scripts, and create more robust modules. Cmdlets also create the capability to use PowerShell as a programming language due to its underlying .NET Framework. This means that cmdlets can be combined by a user to work together and facilitate configuration of data and systems.

In general, PowerShell provides scalability, the ability to simplify management tasks, and generate insights into devices on a network quickly. The most commonly seen uses often transform workflows through:

- **Automation.** Administrators are able to automate tasks through the use of cmdlets as opposed to manual configuration for more tedious tasks.

- **Shortcuts.** Administrators are able to use PowerShell to work around software constraints. An example is an IT administrator who pushes out configuration settings (e.g., password policy) for an application to all systems across the network.
- **Scalability.** IT administrators are constantly in need of obtaining information from systems to ensure that they are kept up-to-date and tracked in inventory accordingly. Doing this manually from system to system, especially in a larger enterprise, is near impossible. With PowerShell, administrators are able to automate the collection of data from enterprise assets in a short amount of time, saving resources that can be used for other tasks. Additionally, with the use of PowerShell remoting, administrators are able to create scripts designed for scale, reaching a large group of systems on the network to conduct activities such as installing updates, establishing configuration settings, and more.
- **Visibility.** Most of the data on an operating system is not easily viewable or discoverable; for example, the Windows Registry and digital signature certificates. PowerShell provides visibility into these files and more through the use of the command-line, making it easier for collection and analysis.

Attacks Using PowerShell

Unfortunately, CTAs realize the versatility and usefulness of PowerShell as a tool and have weaponized it to conduct cyber attacks. Even as far back as 2016, Carbon Black reported at least 38%[1] of observed incidents by Carbon Black® and partners included PowerShell use as part of the attack. The majority, approximately 87%[2], used PowerShell in commodity malware such as click fraud, fake anti-virus, and opportunistic malware. Only 13%[3] of the attacks using PowerShell seemed to be targeted. In more recent years, approximately 49%[4] of threats analyzed used PowerShell in the attack chain, according to Red Canary's 2021 Threat Detection Report.

PowerShell has been used in several cyber attacks, including Trojans, backdoors, and ransomware, to name a few. This is because CTAs can use PowerShell for a variety of objectives. For example:

- **CTAs can evade defenses.** Due to PowerShell being a native Windows tool and functional on other platforms such as Linux and macOS, CTAs are able to use it without raising red flags since it is also used by IT professionals – evading traditional network defenses. CTAs can also use PowerShell to disable critical threat detection tools, such as anti-malware applications.
- **CTAs can use PowerShell to automate activities.** Through the use of the Windows Application Programming Interface (API), PowerShell is able to be easily used by CTAs, allowing them to automate tasks and evade detection.
- **CTAs can easily access PowerShell modules that are widely available on many open source platforms.** This also means that CTAs are able to modify PowerShell and use it for malicious purposes.
- **CTAs are able to escalate privileges and execute PowerShell scripts via the command-line or in memory, evading detection from traditional anti-malware applications.** PowerShell can also be used to perform remote code execution from another machine on the network with the potential to move laterally elsewhere on the network. CTAs may also leave backdoors on the system to carry out additional malicious activity in the future.

1 https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/docs/vmwcb-report-powershell-deep-dive.pdf
2 https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/docs/vmwcb-report-powershell-deep-dive.pdf
3 https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/docs/vmwcb-report-powershell-deep-dive.pdf
4 https://resource.redcanary.com/rs/003-YRU-314/images/2021-Threat-Detection-Report.pdf

- **CTAs can write malware in PowerShell.** Examples include: Netwalker, RogueRobin, PowerWare, and POWELIK.
- **CTAs can use post-exploitation frameworks** (e.g., CobaltStrike, PowerSploit, PowerShell Empire, Mimikatz), which leverage PowerShell components, to compromise a network and steal credentials.
- **CTAs can use social engineering to send a macro-enabled document,** subsequently launching PowerShell.
- **CTAs can encode payloads using the command-line** and load PowerShell into other processes.
- **CTAs can use scripts, such as WScript and CScript,** to bypass script-based constraints on operating systems.
- **CTAs can increase their attack surface by using other applications,** such as the WMI command-line, to run scripts.

Why Implement Defenses Against PowerShell?

As noted, PowerShell is an extremely powerful and widely available tool for both legitimate use and malicious activity. It is often unrealistic to expect an enterprise to completely disable, block, or avoid use of PowerShell within its environment. However, unmanaged use of PowerShell can lead to unnecessary risk and avoidable cyber incidents. It is important to implement defensive processes and measures, such as understanding and managing your PowerShell environment, securely configuring PowerShell, malware defenses, logging, continuous vulnerability management, email and browser protections, and security and awareness training. By taking these measures, an enterprise can effectively take advantage of the benefits of PowerShell while minimizing their risk of exploitation.

Purpose

This PowerShell guide is intended to be part of a series of LotL guides that help enterprises prioritize common exploits and attack vectors used by CTAs to conduct cyber attacks. These LotL TTPs are prioritized due to the CTA's ability to effectively use them to obfuscate activity and bypass common security measures, requiring a more robust defense strategy. This *guide* is designed to simplify and prioritize actionable defenses for security professionals in an overwhelming world of information. It is simple enough to be used by novice security professionals, yet flexible and detailed enough to meet experienced professionals' needs. CIS focuses on supporting the small and medium enterprises (SMEs) in need of clear and simplified guidance.

What This Guide Covers

This *guide* covers the defensive measures recommended by CIS to protect an enterprise against the use of the native tool, PowerShell, in a cyber attack. Those defensive measures include managing the PowerShell environment, secure configuration of PowerShell, malware defenses for PowerShell, PowerShell logging, continuous vulnerability management of PowerShell, email and browser protection against PowerShell, and security and awareness training.

Who Should Use This Guide

This *guide* is designed to be user friendly for all IT security professionals ranging from novice to experienced. It is specifically designed to target SMEs who may be under-resourced, but is flexible enough to be adopted by larger enterprises.

How To Use This Guide

This *guide* is divided into seven sections outlining major defensive processes and recommendations to protect against PowerShell-based attacks. Each section is broken down into four major subsections: an explanation of the relevance of the defensive measure, a description of how to best implement it, CIS Controls and CIS Benchmarks™ that are applicable to the implementation of the defensive measure, and finally, any relevant MITRE Adversarial Tactics, Techniques and Common Knowledge (ATT&CK®) (Sub-)Techniques that may be used in an environment without the defensive measures. The seven sections are linked and intended to be cross-referenced to ensure the most effective implementation of the *guide*. It is important that an enterprise adapts the guidance to best fit their individual environment.

PowerShell Defenses

Manage Your PowerShell Environment

Why is This Important?

In order to defend your network, you must first know what is on your network. This involves activities such as inventory and control of enterprise assets and software (CIS Controls 1 and 2), data protection (CIS Control 3), and account/access management (CIS Controls 5 and 6). To defend against an attack abusing PowerShell, an enterprise must first know which assets have PowerShell enabled, which version those assets are operating on, which accounts have access, and the level of access granted to those users (e.g., user vs. administrator). Since PowerShell is a commonly used built-in administrative tool, it is important to identify and manage access since it can also be abused by CTAs.

Implementation

As previously mentioned, keeping track of inventory for enterprise assets and software is the first step. This can be done by using a simple spreadsheet, such as CIS's Asset Tracking Spreadsheet, or by using tooling to automate the process. Without taking this critical first step, the remaining Safeguards become difficult to implement, as many are dependent on this singular Safeguard (1.1).

Using up-to-date software is also important. Currently, PowerShell v7.2[5] is the most recent and supported version available. If your enterprise uses PowerShell, it is recommended to use PowerShell v7.2 and above, as this version has enhanced security and logging features. Additionally, removing older versions of PowerShell is also recommended, since they are generally less secure, not updated, and open to exploitation. For example, PowerShell v2 has since been deprecated and, if installed, should be disabled or uninstalled. If not removed, CTAs can use this outdated version of PowerShell to bypass security features or impair defenses that would otherwise be detected if an updated version of PowerShell was used. This is referred to as a downgrade attack.

In addition to enterprise asset and software management, maintaining an inventory of accounts is recommended to properly manage access to PowerShell. This includes managing access during onboarding and offboarding. As mentioned previously, most users outside of IT generally do not need access to PowerShell, especially PowerShell remoting, as this is primarily used by administrators for remote management. Therefore, it may seem like a logical next step to restrict PowerShell across the board. However, in doing so, there may be implications that are not as obvious. An example is software that uses PowerShell code and/or scripts for installation. There are several ways to limit or restrict the use of PowerShell. One common method is to use software for application control to perform allowlisting or blocklisting that will limit the use of PowerShell on a system, including the ability to block commands that are generally used to bypass security. This can be done using technologies, such as Windows Defender Application Control (WDAC), a security feature in Microsoft® Security Response Center (MSRC), or AppLocker®. Other commercial offerings may also be used to control PowerShell.

5 PowerShell v7.2 is a Long Term Servicing (LTS) release (built on .NET 6.0). It will be supported by Microsoft until November 30, 2024.

Enterprises may also control access using role-based access control (RBAC) to restrict PowerShell for only specific roles in the enterprise. Beyond RBAC, restrictions for PowerShell access can even be more granular, allowing administrators the ability to only perform specific actions on a system, rather than granting a blanket administrator account that allows all privileges. This addresses the principle of least privilege, where users only have access to what is necessary in order to complete their tasks. Starting with PowerShell v5 and above, Just Enough Administration (JEA) is a feature that can assist with the least privilege principle and is used to delegate administrative functions for anything managed by PowerShell. JEA is available in Windows 10 and above and is used through PowerShell remoting. Overall, JEA helps lower the number of administrator accounts on an enterprise's network and reduces the damage that a CTA could do if they gain access to a privileged account.

Related CIS Critical Security Controls

SAFEGUARD	TITLE	IG1	IG2	IG3
1.1	Establish and Maintain Detailed Enterprise Asset Inventory	✓	✓	✓
2.1	Establish and Maintain a Software Inventory	✓	✓	✓
2.2	Ensure Authorized Software is Currently Supported	✓	✓	✓
2.5	Allowlist Authorized Software		✓	✓
2.6	Allowlist Authorized Libraries		✓	✓
2.7	Allowlist Authorized Scripts			✓
5.1	Establish and Maintain an Inventory of Accounts	✓	✓	✓
5.4	Restrict Administrator Privileges to Dedicated Administrator Accounts	✓	✓	✓
6.1	Establish an Access Granting Process	✓	✓	✓
6.2	Establish an Access Revoking Process	✓	✓	✓
6.8	Define and Maintain Role-Based Access Control			✓

Related MITRE ATT&CK (Sub-)Techniques

ATT&CK (SUB-)TECHNIQUE ID	ATT&CK (SUB-)TECHNIQUE TITLE
T1059.001	Command and Scripting Interpreter: PowerShell
T1562.010	Impair Defenses: Downgrade Attack

Related CIS Microsoft Windows 10 Enterprise Benchmark v1.12.0

CIS BENCHMARK SECTION #	RECOMMENDATION #	TITLE
18.9.103	18.9.103.1	(L2) Ensure 'Allow Remote Shell Access' is set to 'Disabled'

Secure Configurations for PowerShell

Why is This Important?

Equally important to managing an enterprise's assets and software, is ensuring they are securely configured. This also means not just accepting the default configurations as "secure," as many are designed with ease-of-use in mind as opposed to security. With PowerShell, there are several areas to consider when establishing secure configurations.

Implementation

There is not one "silver bullet" solution to securing PowerShell. The same goes for configuring PowerShell. Due to its wide-array of usage, securely configuring PowerShell requires careful testing and quality assurance to determine whether or not a configuration will negatively impact day-to-day activities that are required to run the enterprise. The following guidance is not a one-size-fits-all approach and may need to be tailored to fit the enterprise's needs.

Some general best practice recommendations for securing PowerShell include establishing a baseline of PowerShell activity. This step is important, especially for monitoring devices, so the application can learn what is normal vs. abnormal. Additionally, disable any components that are not needed with PowerShell. If there is not a business need, then remove the access. Certain applications may also be able to block PowerShell scripts so they only run from a specific location or path. This is helpful to limit the attack surface where PowerShell can be abused. PowerShell can also be restricted or blocked from performing activities that are used by CTAs, such as invoking commands on remote systems or downloading content via the internet. How these configurations are established will vary depending on the application being used to implement the controls as well as the needs of the enterprise (e.g., block at the user-level, application-level, command-level, etc.). Additionally, script signing and refraining from storing sensitive information (e.g., credentials) in a PowerShell script itself are best practices to follow. If credentials are required for the performance of a script, controls surrounding the protection of those credentials should be deployed.

There are also some key Windows-specific configurations that can be set to help secure PowerShell. For example, enterprises can use the '*Turn on Script Execution*' Group Policy setting in Windows to control which types of scripts are allowed to be executed on a system. There are seven PowerShell execution policies: *AllSigned, Bypass, Default, RemoteSigned, Restricted, Undefined,* and *Unrestricted*. For Windows servers, *RemoteSigned* is the default policy, meaning that scripts can run if they have a digital signature for scripts and files that are downloaded from the internet. For added security, enterprises can change this to *AllSigned*, which will require all scripts, including those downloaded from the internet and those that are written in-house, to be signed by a trusted publisher. On Windows clients, *Restricted* is the default policy, meaning that the user can still run commands, but not scripts (including '.ps1xml,' '.psm1,' and '.ps1' files). It is important to note that the Execution Policy configuration reduces the risk of unintentional violations involving PowerShell, however, users may still be able to bypass this policy by typing scripts into the command-line directly depending on environmental configurations. This configuration is not intended to stop the bad, but rather determine the bad from good. For example, an attack that reaches out to download an unsigned PowerShell script online may be stopped by implementing the *RemoteSigned* or *AllSigned* policies. However, it would not stop a CTA who had "hands on keyboard" access from invoking PowerShell via the command-line. In the event that a CTA does bypass the PowerShell Execution Policy, a review of the Windows Registry Key, 'HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\PowerShell\1\ShellIds\Microsoft.PowerShell\ Value: ExecutionPolicy' would be able to provide the last modified timestamp, which may be helpful during a forensic investigation. This review can be done by exporting the key to a text file in order to show the Last Write Time.

Administrators may also use PowerShell profiles, which are scripts that execute when PowerShell starts, for the customization and automation of certain activities. For example, they may configure a profile (e.g., script) that gets triggered upon a user logging in. Different profiles also exist for different applications, such as profiles for the Windows PowerShell Integrated Scripting Environment (ISE) or Windows PowerShell console. These PowerShell profiles, while helpful, can also be modified by a CTA to carry out malicious actions, such as establishing persistence mechanisms. Securing these profiles by making them unable to be changed and/or limited to modification by a restricted group of administrators is recommended for reducing this risk. One area that profiles can be controlled is within the PowerShell Execution Policy, where scripts and configuration files (to include profiles) can be prevented from running under the *Restricted* setting. Careful consideration should be taken when setting these policies, as certain environments may warrant more or less access to PowerShell depending on the business need.

In addition to setting the PowerShell Execution Policy, enterprises may also use Configuration Manager, now a part of Microsoft Endpoint Manager, to review and approve scripts. For example, if an administrator would like to have their script approved, they would submit it for review. Once the administrator that has the *script approver* role reviews it, they can choose to approve or deny it. By implementing this best practice, it also enforces separation of duties by preventing the author of the script from also being the approver. As a warning, some PowerShell scripts may be obfuscated, making it difficult to detect malicious behavior. Administrators should deploy the use of script-checking tools, such as *PSScriptAnalyzer*, to help perform script reviews. However, it is ultimately up to the script approver to determine whether or not a script has malicious intent and approve or deny.

Another area commonly abused by CTAs is PowerShell parameters, which are components of a script. An enterprise can securely use PowerShell parameters by only allowing pre-defined ones and using the regular expression feature to define those parameters so that permitted characters are established up front. A PowerShell module, called *InjectionHunter*, can also be used to detect malicious code that could lead to an injection attack. Additionally, Microsoft Visual Studio offers tools to check PowerShell syntax.

PowerShell has four main language modes that determine which elements are and are not allowed in a PowerShell session: *FullLanguage*, *ConstrainedLanguage*, *RestrictedLanguage*, and *NoLanguage*. One way to reduce the risk of a PowerShell attack is to enable *ConstrainedLanguage* Mode. This mode allows all cmdlets and language elements, but is used to restrict data types in PowerShell to only a set list of allowed types. When used on its own through setting the *LanguageMode*, *ConstrainedLanguage* Mode is not a security feature, as it can be easily bypassed by opening another PowerShell session. However, when used in conjunction with an application control solution, such as Device Guard or AppLocker, PowerShell can be set to run in *ConstrainedLanguage* Mode, disabling access to features such as .COM objects and system APIs. This cannot be overridden easily, making it a stronger defense in the event of a PowerShell attack. It is important to note that the more restrictions placed, the more likely it is to impact day-to-day business functions. For example, an administrator that has an approved script may find that it no longer will execute. In this instance, the scripts can be added to the Device Guard policy and will then be able to run in *FullLanguage* Mode.

PowerShell can also be used remotely, called PowerShell remoting, through the Windows Remote Management (WinRM) protocol, as mentioned previously. WinRM listens on ports 5985 (HTTP) and 5986 (HTTPS) and uses the WinRM service to connect remotely and run commands. If not needed, it is recommended to restrict and/or disable, if possible, access to WinRM. Additionally, other configuration guidelines for WinRM from the CIS Microsoft Windows 10 Enterprise Benchmark can be seen in the table below.

Related CIS Critical Security Controls

SAFEGUARD	TITLE	IG1	IG2	IG3
4.1	Establish and Maintain a Secure Configuration Process	✓	✓	✓
4.2	Establish and Maintain a Secure Configuration Process for Network Infrastructure	✓	✓	✓
4.4	Implement and Manage a Firewall on Servers	✓	✓	✓
4.5	Implement and Manage a Firewall on End-User Devices	✓	✓	✓
4.8	Uninstall or Disable Unnecessary Services on Enterprise Assets and Software		✓	✓

Related MITRE ATT&CK (Sub-)Techniques

ATT&CK (SUB-)TECHNIQUE ID	ATT&CK (SUB-)TECHNIQUE TITLE
T1059.001	Command and Scripting Interpreter: PowerShell
T1546.013	Event Triggered Execution: PowerShell Profile

Related CIS Microsoft Windows 10 Enterprise Benchmark v1.12.0

CIS BENCHMARK SECTION #	RECOMMENDATION #	TITLE
18.9.102.1	18.9.102.1.1	(L1) Ensure 'Allow Basic authentication' is set to 'Disabled'
18.9.102.1	18.9.102.1.2	(L1) Ensure 'Allow unencrypted traffic' is set to 'Disabled'
18.9.102.1	18.9.102.1.3	(L1) Ensure 'Disallow Digest authentication' is set to 'Enabled'
18.9.102.2	18.9.102.2.1	(L1) Ensure 'Allow Basic authentication' is set to 'Disabled'
18.9.102.2	18.9.9102.2.2	(L2) Ensure 'Allow remote server management through WinRM' is set to 'Disabled'
18.9.102.2	18.9.9102.2.3	(L1) Ensure 'Allow unencrypted traffic' is set to 'Disabled'
18.9.9102.2	18.9.9102.2.4	(L1) Ensure 'Disallow WinRM from storing RunAs credentials' is set to 'Enabled'
5	5.40	(L2) Ensure 'Windows Remote Management (WS-Management) (WinRM)' is set to 'Disabled'

Malware Defenses for PowerShell

Why is This Important?

Traditional anti-malware applications use signature-based detections to identify malware. However, with the increase in fileless malware attacks and LotL attacks over the past few years, it has forced the industry to re-think how malware is detected, prompting the need for heuristic-based algorithms. PowerShell, among other scripting languages, can be invoked directly in memory and execute commands to download a payload remotely, bypassing security controls and leaving little to no artifacts on disk for defenders to identify and analyze.

Malware that uses PowerShell can allow for activities such as privilege escalation, obfuscation, and remote code execution. In recent years, there are several malware variants that have weaponized PowerShell, including Netwalker, RogueRobin, PowerWare, and POWELIK. In addition, it is used by many legitimate post-exploitation frameworks (e.g., Metasploit, PowerSploit, and Mimikatz) that can also double as a weapon used by CTAs to exploit systems and steal credentials. Another popular TTP among CTAs is using macro-enabled Office documents, allowing them to launch PowerShell scripts and take control of a system remotely. The bottom line is that attacks using PowerShell are not going away anytime soon, which is why it is important to enhance defenses while still utilizing PowerShell for day-to-day work.

Implementation

With the growing trend toward fileless malware, especially malware using PowerShell, it is important to implement technologies that have the ability to detect and defend against these complex attacks. It is best practice to implement anti-malware applications on enterprise assets, as well as update anti-malware signature updates. However, if an enterprise has the resources and budget, using an application that is behavior-based (e.g., uses heuristic analysis), will allow an enterprise to detect more advanced PowerShell attacks. Several anti-malware applications and other technologies (e.g., Endpoint Detection and Response (EDR) solutions) employ the use of heuristic-based analysis. For example, if using an EDR solution, it can detect the Dynamic Link Library (DLL) 'System.Management.Automation.DLL' – a common library used by CTAs to execute PowerShell commands.

Another technology available for detecting malicious PowerShell usage is with Microsoft's Antimalware Scan Interface (AMSI). Starting with Windows 10 in 2015, AMSI was released – a vendor-agnostic interface that can be integrated into other anti-malware products on the Windows platform. Its main purpose is to scan the contents of a script prior to execution to determine if it is safe. AMSI is integrated into several areas of Windows 10 including User Account Control (UAC), Windows Script Host, JavaScript, Visual Basic Script (VBS), Visual Basic for Applications (VBA) Macros, and PowerShell. It can be used by developers and vendors in slightly different ways. For a developer, AMSI can be extremely helpful to review code blocks and ensure that they do not contain malicious code, especially with the increase of using code from third-parties. For a vendor, AMSI can be integrated into their product through the AMSI API interface. Unfortunately, AMSI has been attacked and bypassed successfully by CTAs, as we see with many legitimate tools. With the continued use of obfuscation combined with evasive methods to bypass detections, this is another tool that can be used, but not to be relied upon for complete mitigation.

Related CIS Critical Security Controls

SAFEGUARD	TITLE	IG1	IG2	IG3
10.1	Deploy and Maintain Anti-Malware Software	✓	✓	✓
10.2	Configure Automatic Anti-Malware Signature Updates	✓	✓	✓
10.5	Enable Anti-Exploitation Features		✓	✓
10.6	Centrally Manage Anti-Malware Software		✓	✓
10.7	Use Behavior-Based Anti-Malware Software		✓	✓
13.7	Deploy a Host-Based Intrusion Prevention Solution			✓
13.8	Deploy a Network Intrusion Prevention Solution			✓

Related MITRE ATT&CK (Sub-)Techniques

ATT&CK (SUB-)TECHNIQUE ID	ATT&CK (SUB-)TECHNIQUE TITLE
T1059.001	Command and Scripting Interpreter: PowerShell

Related CIS Microsoft Windows 10 Enterprise Benchmark v1.12.0

CIS BENCHMARK SECTION #	RECOMMENDATION #	TITLE
18.9.47.5.1	**18.9.47.5.1.1**	(L1) Ensure 'Configure Attack Surface Reduction rules' is set to 'Enabled'
18.9.47.5.1	**18.9.47.5.1.2**	(L1) Ensure 'Configure Attack Surface Reduction rules: Set the state for each ASR rule' is configured

PowerShell Logging

Why is This Important?

Log collection and analysis is critical for an enterprise to detect a potential cyber attack. Unfortunately, CTAs can use PowerShell to carry out malicious activities, including clearing event logs, which is why it is important to not only collect logs, but also protect them. If all other defenses fail, logging and monitoring may end up being your main line of defense.

Logs can be extremely helpful in providing information. For example, the time of execution for an application and the associated user account it was executed under. With PowerShell, there are several PowerShell-specific logs as well as other audit logs that can help with detection. During an incident, logs can help the enterprise understand the extent of an attack, including when an attack occurred, what systems were impacted, and what data may have been accessed/exfiltrated. The retention period of logging is also important. Logs that do not go back far enough may impact the investigation of an attack, especially if the attack has gone undetected for a long period of time. Bottom line, if you want to detect a PowerShell attack, it starts with logging.

Implementation

While logging from anti-malware applications and EDR platforms are extremely beneficial for enhancing visibility and detecting a wide variety of attacks, some of the most detailed information about PowerShell activity on a system can be captured by enabling simple configuration settings in the Windows environment. There are four main types of PowerShell logging that can be enabled. Transcription logging is one type of logging for PowerShell that can be enabled to log input (commands) and output from a PowerShell session to a specified location in the form of a text file. When enabled, it can tell you several pieces of information, including the username, the date and time of when the session was started, the machine name, the host application, and process ID. Similar to bash history on a Linux system, transcription logging creates a running file of every PowerShell command entered and result generated. Additionally, an option to include invocation headers can be selected to record the timestamps for executed commands. These details can be particularly useful for incident response purposes to determine what specific commands may have been used by a CTA.

Another popular logging type that can be enabled is Script Block logging, which is available in PowerShell v5 and above (in Windows 10). Script Block logging captures entire PowerShell script blocks just before it is delivered to the PowerShell engine to execute. An additional setting can be enabled, called invocation logging, that will record the start and stop times of an invoked command. This type of logging can be particularly useful because it also captures de-obfuscated code, since it captures it before execution. This can be extremely helpful in the event of an attack to determine what was actually done on the system(s). Since the default configuration only logs PowerShell script blocks the first time they are used, enterprises may choose to select *'Log script block invocation start / stop events'* to capture the start and stop times when commands and scripts are invoked. This can also be helpful information during incident response, however, caution should be taken since this setting may generate a high volume of events.

When enabling Script Block logging, enterprises also have the option to enable Protected event logging. Previously, IT administrators acknowledged the risk of revealing sensitive information in PowerShell logs, such as credentials used within a script. With Protected event logging, local logs are encrypted to prevent a CTA from obtaining this sensitive information. This allows the enterprise to later decrypt the logs in a more secure space; for example, a Security Information and Event Management (SIEM) or other centralized logging platform. The logs are encrypted with the Cryptographic Message Syntax (CMS) standard and use public key (asymmetric) cryptography. It is important to note that in order to use Protected event logging, Script Block logging must also be enabled. While these are two separate configurations, they work in tandem.

Module logging captures the execution of modules in PowerShell, including de-obfuscated code and outputs. Through the use of Group Policy Objects (GPOs), an enterprise can choose to configure Module logging for specific modules (e.g., Microsoft.PowerShell.*, Microsoft.WSMan.Management) or they may choose to capture all modules using the wildcard symbol (*). The use of the wildcard may be particularly helpful during monitoring and incident response to detect any custom modules that are used by a CTA.

Once PowerShell logging is enabled, an enterprise can review the following logs for PowerShell-related events:

- Windows PowerShell.evtx
- Microsoft-Windows-PowerShell/Operational.evtx
- Microsoft-Windows-PowerShell/Analytic.etl (disabled by default)
- Microsoft-Windows-WinRM/Operational.evtx
- Microsoft-Windows-WinRM/Analytic.etl (disabled by default)
- PowerShellCore/Operational
- PowerShellCore/Analytical

While not PowerShell-specific, enabling command-line audit logs can also be beneficial for capturing details of the command-line arguments used. These details are captured in the Windows Security Event Log and can be set through Group Policy by setting the 'Include command line in process creation events' GPO to enabled. This policy setting determines what information is logged in security audit events when a new process has been created. A configuration will also need to be set under 'Audit Process Creation' setting, where options for enabling Success and/or Failure events can be set. At a minimum, successful events should be logged.

As with any type of logging, it is important to ensure that some logs may generate a high volume of events. Tuning these features appropriately is important to ensure that critical details are being captured without causing additional issues (e.g., log alert fatigue, reduced storage capacity). Additionally, as increased verbosity of logging can also divulge sensitive information, it is good practice to store logs in a centralized location off of the local system, if possible. If log centralization is not possible, local user access can be denied from accessing event logs as an alternative mitigation.

Related CIS Critical Security Controls

SAFEGUARD	TITLE	IG1	IG2	IG3
8.1	Establish and Maintain an Audit Log Management Process	✓	✓	✓
8.2	Collect Audit Logs	✓	✓	✓
8.3	Ensure Adequate Audit Log Storage	✓	✓	✓
8.4	Standardize Time Synchronization		✓	✓
8.5	Collect Detailed Audit Logs		✓	✓
8.6	Collect DNS Query Audit Logs		✓	✓
8.7	Collect URL Request Audit Logs		✓	✓
8.8	Collect Command-Line Audit Logs		✓	✓
8.9	Centralize Audit Logs		✓	✓
8.10	Retain Audit Logs		✓	✓
8.11	Conduct Audit Log Reviews		✓	✓

Related CIS Microsoft Windows 10 Enterprise Benchmark v1.12.0

CIS BENCHMARK SECTION #	RECOMMENDATION #	TITLE
18.9.100	18.9.100.1	(L1) Ensure 'Turn on PowerShell Script Block Logging' is set to 'Enabled'
18.8.3	18.8.3.1	(L1) Ensure 'Include command line in process creation events' is set to 'Enabled'

Related MITRE ATT&CK (Sub-)Techniques

ATT&CK (SUB-)TECHNIQUE ID	ATT&CK (SUB-)TECHNIQUE TITLE
T1070.001	Indicator Removal on Host: Clear Windows Event Logs

Continuous Vulnerability Management for PowerShell

Why is This Important?

As with any technology, it is important to ensure that it is receiving the latest updates to mitigate the potential for exploitation of a vulnerability. CTAs are constantly looking for ways to exploit a network and defenders are equally facing challenges at keeping up with the pace of patching and managing those vulnerabilities. Unfortunately, it only takes one unpatched vulnerability for a CTA to be successful in exploiting a system, which is why it is important to establish a process for continuous vulnerability management. Additionally, where a patch or fix for a vulnerability is not available or able to be implemented, mitigating controls should be put in place to ensure that the risk is reduced to an allowable level.

Implementation

In November of 2021, PowerShell v7.2 was officially released, including many features, like the ability to integrate with Microsoft Update, which allows for automatic updates to be enabled. Upgrading from PowerShell v7 to v7.2 can be accomplished by using the MSI package, which will provide the automatic update functionality. It is important to note that if downloaded from the Microsoft Store, PowerShell will update, but the MSI package is still needed to enable the automatic update capability through Microsoft Updates. As mentioned previously, there are many older versions of PowerShell (e.g., v2) that are considered insecure and should not be used. If these insecure versions of PowerShell must be used for the purposes of a legacy system, ensure that they are not externally-exposed and have appropriate mitigating controls in place.

Another area that can be controlled with vulnerability management is with PowerShell remoting through WinRM. Open ports, such as 5985 and 5986 for WinRM, may be identified during vulnerability scans on enterprise assets. Once identified, appropriate controls should be applied based on the needs of the enterprise (e.g., Internet Protocol (IP) allowlists for administrators using WinRM). This is especially important for assets that are externally exposed.

Related CIS Critical Security Controls

SAFEGUARD	TITLE	IG1	IG2	IG3
7.1	Establish and Maintain a Vulnerability Management Process	✓	✓	✓
7.2	Establish and Maintain a Remediation Process	✓	✓	✓
7.3	Perform Automated Operating System Patch Management	✓	✓	✓
7.4	Perform Automated Application Patch Management	✓	✓	✓
7.5	Perform Automated Vulnerability Scans of Internal Enterprise Assets		✓	✓
7.6	Perform Automated Vulnerability Scans of Externally-Exposed Enterprise Assets		✓	✓
7.7	Remediate Detected Vulnerabilities		✓	✓

Related CIS Microsoft Windows 10 Enterprise Benchmark v1.12.0

CIS BENCHMARK SECTION #	RECOMMENDATION #	TITLE
18.9.108.1	18.9.108.1.1	(L1) Ensure 'No auto-restart with logged on users for scheduled automatic updates installations' is set to 'Disabled'
18.9.108.2	18.9.108.2.1	(L1) Ensure 'Configure Automatic Updates' is set to 'Enabled'
18.9.108.2	18.9.108.2.2	(L1) Ensure 'Configure Automatic Updates: Scheduled install day' is set to '0 - Every day'
18.9.108.2	18.9.108.2.3	(L1) Ensure 'Remove access to "Pause updates" feature' is set to 'Enabled'
18.9.108.4	18.9.108.4.1	(L1) Ensure 'Manage preview builds' is set to 'Disabled'
18.9.108.4	18.9.108.4.2	(L1) Ensure 'Select when Preview Builds and Feature Updates are received' is set to 'Enabled: 180 or more days'
18.9.108.4	18.9.108.4.3	(L1) Ensure 'Select when Quality Updates are received' is set to 'Enabled: 0 days'

Email and Browser Protections Against Malicious PowerShell Activity

Why is This Important?

Email and web browsers are two main avenues that a CTA can use to abuse PowerShell. For example, an email could contain a malicious link that downloads a document using a VBA macro to launch PowerShell via WMI. From there, it can execute a PowerShell script and download additional payload(s) to the system. This is just one important example of how PowerShell can be used in the early stages of an attack. Additionally, PowerShell can be used for lateral movement and to establish persistence mechanisms for maintaining a foothold on the network.

Implementation

Defenses in this category include using trusted Domain Name System (DNS) services to block access to known malicious domains (e.g., DNS queries), including ones that reach out via PowerShell to download additional malicious content. Examples of such services include Quad9 or the Malicious Domain Blocking and Reporting (MDBR) service from the Multi-State Information Sharing and Analysis Center® (MS-ISAC®) and Elections Infrastructure Information Sharing and Analysis Center® (EI-ISAC®). Additionally, network-based Uniform Resource Locator (URL) filters are an added layer of security that can be used to help to limit a system from connecting to malicious or unapproved websites (e.g., URLs). As phishing emails are a large concern when it comes to the first stage of a PowerShell attack, using anti-malware email protections, like sandboxing or attachment scanning, are helpful in thwarting a potential incident. An enterprise can also block certain file types in their Secure Email Gateway (SEG) that will help to reduce the amount of incoming email attachments that are potentially malicious.

Related CIS Critical Security Controls

SAFEGUARD	TITLE	IG1	IG2	IG3
9.2	Use DNS Filtering Services	✓	✓	✓
9.3	Maintain and Enforce Network-Based URL Filters		✓	✓
9.6	Block Unnecessary File Types		✓	✓
9.7	Deploy and Maintain Email Server Anti-Malware Protections			✓

Related CIS Microsoft Windows 10 Enterprise Benchmark v1.12.0

CIS BENCHMARK SECTION #	RECOMMENDATION #	TITLE
18.9.47.5.1	18.9.47.5.1.1	(L1) Ensure 'Configure Attack Surface Reduction rules' is set to 'Enabled'
18.9.47.5.1	18.9.47.5.1.2	(L1) Ensure 'Configure Attack Surface Reduction rules: Set the state for each ASR rule' is configured (Automated)
18.9.47.12	18.9.47.12.2	(L1) Ensure 'Turn on e-mail scanning' is set to 'Enabled'

Security Awareness and Training

Why is This Important?

An important part of any security program is training the workforce to identify a potential attack. CTAs may use various forms of initial access, including phishing emails laced with malicious links or macro-enabled files. Humans, while sometimes are our weakest link, can also be our greatest defense if trained properly. According to the recent 2022 Verizon Data Breach Investigations Report (DBIR)[6], 82% of breaches involved a human-element.

Implementation

Defenses, in the form of tools and other technologies, for avoiding these attacks are important. However, training is also an integral part of defending against such attacks. This includes activities, including education on what a social engineering attack could look like, what to do/not to do if they receive a suspicious email, and best practices for secure browsing on the internet. Additionally, having a process in place for reporting incidents (e.g., who to contact, how to contact) is critical to not only ensure that the user is contacting the right team, but also that they have a line of communication open to report the issue in a timely manner.

Related CIS Critical Security Controls

SAFEGUARD	TITLE	IG1	IG2	IG3
14.1	Establish and Maintain a Security Awareness Program	✓	✓	✓
14.2	Train Workforce Members to Recognize Social Engineering Attacks	✓	✓	✓
14.6	Train Workforce Members on Recognizing and Reporting Security Incidents	✓	✓	✓
14.7	Train Workforce on How to Identify and Report if Their Enterprise Assets are Missing Security Updates	✓	✓	✓

6 https://www.verizon.com/business/resources/reports/dbir/

Conclusion

Most cyber attacks occur due to a lack of good cyber hygiene. Whether that be through the use and abuse of tools and techniques found on the impacted system (e.g., LotL) or through an exploited vulnerability, many attacks can be defended by implementing Safeguards found within the CIS Controls. This *guide* provides guidance on how an enterprise could apply the Safeguards specifically in terms of defending against or detecting a PowerShell attack. By implementing the security best practices recommended in this *guide*, enterprises can apply a defense-in-depth strategy to strengthen their cybersecurity posture and help better defend against a PowerShell attack.

APPENDIX A

Implementation Groups

The Implementation Group methodology was developed as a new way to prioritize the CIS Controls. These IGs provide a simple and accessible way to help enterprises of different classes focus their scarce security resources, while still leveraging the value of the CIS Controls program, community, and complementary tools and working aids.

The number of Safeguards an enterprise is expected to implement increases based on which group the enterprise falls into.

153 TOTAL SAFEGUARDS

IG3 IG3 assists enterprises with IT security experts to secure sensitive and confidential data. IG3 aims to prevent and/or lessen the impact of sophisticated attacks. **23** SAFEGUARDS

IG2 IG2 assists enterprises managing IT infrastructure of multiple departments with differing risk profiles. IG2 aims to help enterprises cope with increased operational complexity. **74** SAFEGUARDS

IG1 IG1 is the definition of **essential cyber hygiene** and represents a minimum standard of information security for all enterprises. IG1 assists enterprises with limited cybersecurity expertise thwart general, non-targeted attacks. **56** SAFEGUARDS

IG1

An IG1 enterprise is small to medium-sized with limited IT and cybersecurity expertise to dedicate toward protecting IT assets and personnel. The principal concern of these enterprises is to keep the business operational, as they have a limited tolerance for downtime. The sensitivity of the data that they are trying to protect is low and principally surrounds employee and financial information. Safeguards selected for IG1 should be implementable with limited cybersecurity expertise and aimed to thwart general, non-targeted attacks. These Safeguards will also typically be designed to work in conjunction with small or home office commercial off-the-shelf (COTS) hardware and software.

IG2 (Includes IG1)

An IG2 enterprise employs individuals responsible for managing and protecting IT infrastructure. These enterprises support multiple departments with differing risk profiles based on job function and mission. Small enterprise units may have regulatory compliance burdens. IG2 enterprises often store and process sensitive client or enterprise information and can withstand short interruptions of service. A major concern is loss of public confidence if a breach occurs. Safeguards selected for IG2 help security teams cope with increased operational complexity. Some Safeguards will depend on enterprise-grade technology and specialized expertise to properly install and configure.

IG3 (Includes IG1 and IG2)

An IG3 enterprise employs security experts that specialize in the different facets of cybersecurity (e.g., risk management, penetration testing, application security). IG3 assets and data contain sensitive information or functions that are subject to regulatory and compliance oversight. An IG3 enterprise must address availability of services and the confidentiality and integrity of sensitive data. Successful attacks can cause significant harm to the public welfare. Safeguards selected for IG3 must abate targeted attacks from a sophisticated adversary and reduce the impact of zero-day attacks.

If you would like to know more about the Implementation Groups and how they pertain to enterprises of all sizes, there are many resources that explore the Implementation Groups and the CIS Controls in general on our website at https://www.cisecurity.org/controls/cis-controls-list/.

APPENDIX B

Related CIS Safeguards

SAFEGUARD	TITLE	DEFENSE	IG1	IG2	IG3
1.1	Establish and Maintain Detailed Enterprise Asset Inventory	Manage Your PowerShell Environment	✓	✓	✓
2.1	Establish and Maintain a Software Inventory	Manage Your PowerShell Environment	✓	✓	✓
2.2	Ensure Authorized Software is Currently Supported	Manage Your PowerShell Environment	✓	✓	✓
2.5	Allowlist Authorized Software	Manage Your PowerShell Environment		✓	✓
2.6	Allowlist Authorized Libraries	Manage Your PowerShell Environment		✓	✓
2.7	Allowlist Authorized Scripts	Manage Your PowerShell Environment			✓
5.1	Establish and Maintain an Inventory of Accounts	Manage Your PowerShell Environment	✓	✓	✓
5.4	Restrict Administrator Privileges to Dedicated Administrator Accounts	Manage Your PowerShell Environment	✓	✓	✓
6.1	Establish an Access Granting Process	Manage Your PowerShell Environment	✓	✓	✓
6.2	Establish an Access Revoking Process	Manage Your PowerShell Environment	✓	✓	✓
6.8	Define and Maintain Role-Based Access Control	Manage Your PowerShell Environment			✓
4.1	Establish and Maintain a Secure Configuration Process	Secure Configurations for PowerShell	✓	✓	✓
4.2	Establish and Maintain a Secure Configuration Process for Network Infrastructure	Secure Configurations for PowerShell	✓	✓	✓
4.4	Implement and Manage a Firewall on Servers	Secure Configurations for PowerShell	✓	✓	✓
4.5	Implement and Manage a Firewall on End-User Devices	Secure Configurations for PowerShell	✓	✓	✓
4.8	Uninstall or Disable Unnecessary Services on Enterprise Assets and Software	Secure Configurations for PowerShell		✓	✓
10.1	Deploy and Maintain Anti-Malware Software	Malware Defenses for PowerShell	✓	✓	✓
10.2	Configure Automatic Anti-Malware Signature Updates	Malware Defenses for PowerShell	✓	✓	✓
10.5	Enable Anti-Exploitation Features	Malware Defenses for PowerShell		✓	✓
10.6	Centrally Manage Anti-Malware Software	Malware Defenses for PowerShell		✓	✓
10.7	Use Behavior-Based Anti-Malware Software	Malware Defenses for PowerShell		✓	✓
13.7	Deploy a Host-Based Intrusion Prevention Solution	Malware Defenses for PowerShell			✓
13.8	Deploy a Network Intrusion Prevention Solution	Malware Defenses for PowerShell			✓
8.1	Establish and Maintain an Audit Log Management Process	PowerShell Logging	✓	✓	✓
8.2	Collect Audit Logs	PowerShell Logging	✓	✓	✓
8.3	Ensure Adequate Audit Log Storage	PowerShell Logging	✓	✓	✓
8.4	Standardize Time Synchronization	PowerShell Logging		✓	✓
8.5	Collect Detailed Audit Logs	PowerShell Logging		✓	✓
8.6	Collect DNS Query Audit Logs	PowerShell Logging		✓	✓
8.7	Collect URL Request Audit Logs	PowerShell Logging		✓	✓
8.8	Collect Command-Line Audit Logs	PowerShell Logging		✓	✓
8.9	Centralize Audit Logs	PowerShell Logging		✓	✓
8.10	Retain Audit Logs	PowerShell Logging		✓	✓
8.11	Conduct Audit Log Reviews	PowerShell Logging		✓	✓
7.1	Establish and Maintain a Vulnerability Management Process	Continuous Vulnerability Management for PowerShell	✓	✓	✓
7.2	Establish and Maintain a Remediation Process	Continuous Vulnerability Management for PowerShell	✓	✓	✓
7.3	Perform Automated Operating System Patch Management	Continuous Vulnerability Management for PowerShell	✓	✓	✓
7.4	Perform Automated Application Patch Management	Continuous Vulnerability Management for PowerShell	✓	✓	✓
7.5	Perform Automated Vulnerability Scans of Internal Enterprise Assets	Continuous Vulnerability Management for PowerShell		✓	✓

SAFEGUARD	TITLE	DEFENSE	IG1	IG2	IG3
7.6	Perform Automated Vulnerability Scans of Externally-Exposed Enterprise Assets	Continuous Vulnerability Management for PowerShell		✓	✓
7.7	Remediate Detected Vulnerabilities	Continuous Vulnerability Management for PowerShell		✓	✓
9.2	Use DNS Filtering Services	Email and Browser Protections Against Malicious PowerShell Activity	✓	✓	✓
9.3	Maintain and Enforce Network-Based URL Filters	Email and Browser Protections Against Malicious PowerShell Activity		✓	✓
9.6	Block Unnecessary File Types	Email and Browser Protections Against Malicious PowerShell Activity		✓	✓
9.7	Deploy and Maintain Email Server Anti-Malware Protections	Email and Browser Protections Against Malicious PowerShell Activity			✓
14.1	Establish and Maintain a Security Awareness Program	Security Awareness and Training	✓	✓	✓
14.2	Train Workforce Members to Recognize Social Engineering Attacks	Security Awareness and Training	✓	✓	✓
14.6	Train Workforce Members on Recognizing and Reporting Security Incidents	Security Awareness and Training	✓	✓	✓
14.7	Train Workforce on How to Identify and Report if Their Enterprise Assets are Missing Security Updates	Security Awareness and Training	✓	✓	✓

APPENDIX C

ATT&CK (Sub-)Techniques

ATT&CK (SUB-)TECHNIQUE ID	ATT&CK (SUB-)TECHNIQUE TITLE	POWERSHELL DEFENSE
T1059.001	Command and Scripting Interpreter: PowerShell	Manage Your PowerShell Environment Secure Configurations for PowerShell Malware Defenses for PowerShell
T1562.010	Impair Defenses: Downgrade Attack	Manage Your PowerShell Environment
T1546.013	Event Triggered Execution: PowerShell Profile	Secure Configurations for PowerShell
T1070.001	Indicator Removal on Host: Clear Windows Event Logs	PowerShell Logging

APPENDIX D

CIS Benchmark Recommendations[7]

CIS BENCHMARK SECTION #	RECOMMENDATION #	RECOMMENDATION TITLE	POWERSHELL DEFENSE
18.9.103	**18.9.103.1**	(L2) Ensure 'Allow Remote Shell Access' is set to 'Disabled'	Manage your PowerShell environment
18.9.102.1	**18.9.102.1.1**	(L1) Ensure 'Allow Basic authentication' is set to 'Disabled'	Secure Configurations for PowerShell
18.9.102.1	**18.9.102.1.2**	(L1) Ensure 'Allow unencrypted traffic' is set to 'Disabled'	Secure Configurations for PowerShell
18.9.102.1	**18.9.102.1.3**	(L1) Ensure 'Disallow Digest authentication' is set to 'Enabled'	Secure Configurations for PowerShell
18.9.102.2	**18.9.102.2.1**	(L1) Ensure 'Allow Basic authentication' is set to 'Disabled'	Secure Configurations for PowerShell
18.9.102.2	**18.9.9102.2.2**	(L2) Ensure 'Allow remote server management through WinRM' is set to 'Disabled'	Secure Configurations for PowerShell
18.9.102.2	**18.9.9102.2.3**	(L1) Ensure 'Allow unencrypted traffic' is set to 'Disabled'	Secure Configurations for PowerShell
18.9.9102.2	**18.9.9102.2.4**	(L1) Ensure 'Disallow WinRM from storing RunAs credentials' is set to 'Enabled'	Secure Configurations for PowerShell
5	**5.40**	(L2) Ensure 'Windows Remote Management (WS-Management) (WinRM)' is set to 'Disabled'	Secure Configurations for PowerShell
18.9.47.5.1	**18.9.47.5.1.1**	(L1) Ensure 'Configure Attack Surface Reduction rules' is set to 'Enabled'	Malware Defenses for PowerShell Email and Browser Protections Against Malicious PowerShell Activity
18.9.47.5.1	**18.9.47.5.1.2**	(L1) Ensure 'Configure Attack Surface Reduction rules: Set the state for each ASR rule' is configured	Malware Defenses for PowerShell Email and Browser Protections Against Malicious PowerShell Activity
18.9.100	**18.9.100.1**	(L1) Ensure 'Turn on PowerShell Script Block Logging' is set to 'Enabled'	PowerShell Logging
18.8.3	**18.8.3.1**	(L1) Ensure 'Include command line in process creation events' is set to 'Enabled'	PowerShell Logging
18.9.108.1	**18.9.108.1.1**	(L1) Ensure 'No auto-restart with logged on users for scheduled automatic updates installations' is set to 'Disabled'	Continuous Vulnerability Management for PowerShell
18.9.108.2	**18.9.108.2.1**	(L1) Ensure 'Configure Automatic Updates' is set to 'Enabled'	Continuous Vulnerability Management for PowerShell
18.9.108.2	**18.9.108.2.2**	(L1) Ensure 'Configure Automatic Updates: Scheduled install day' is set to '0 - Every day'	Continuous Vulnerability Management for PowerShell
18.9.108.2	**18.9.108.2.3**	(L1) Ensure 'Remove access to "Pause updates" feature' is set to 'Enabled'	Continuous Vulnerability Management for PowerShell
18.9.108.4	**18.9.108.4.1**	(L1) Ensure 'Manage preview builds' is set to 'Disabled'	Continuous Vulnerability Management for PowerShell
18.9.108.4	**18.9.108.4.2**	(L1) Ensure 'Select when Preview Builds and Feature Updates are received' is set to 'Enabled: 180 or more days'	Continuous Vulnerability Management for PowerShell
18.9.108.4	**18.9.108.4.3**	(L1) Ensure 'Select when Quality Updates are received' is set to 'Enabled: 0 days'	Continuous Vulnerability Management for PowerShell
18.9.47.12	**18.9.47.12.2**	(L1) Ensure 'Turn on e-mail scanning' is set to 'Enabled'	Email and Browser Protections Against Malicious PowerShell Activity

7 From the CIS Microsoft Windows 10 Enterprise Benchmark v1.12.0

APPENDIX E

Acronyms and Abbreviations

ACRONYM	ABBREVIATION
AD	Active Directory
AMSI	Antimalware Scan Interface
API	Application Programming Interface
CDM	Community Defense Model
CIS	Center for Internet Security
CMS	Cryptographic Message Syntax
COM	Control Object Model
COTS	Commercial Off-the-Shelf
CTA	Cyber Threat Actor
DLL	Dynamic Link Library
DNS	Domain Name System
EDR	Endpoint Detection and Response
EI-ISAC	Elections Infrastructure Information Sharing and Analysis Center
GPO	Group Policy Object
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IG	Implementation Group
IP	Internet Protocol
ISE	Integrated Scripting Environment
IT	Information Technology
JEA	Just Enough Administration
LotL	Living off the Land
MDBR	Malicious Domain Blocking and Reporting
MITRE ATT&CK	MITRE Adversarial Tactics, Techniques, and Common Knowledge
MS-ISAC	Multi-State Information Sharing and Analysis Center
MSI	Microsoft Windows Installer
MSRC	Microsoft Security Response Center
OS	Operating System
PowerShell ISE	PowerShell Integrated Scripting Environment
RBAC	Role-Based Access Control
RDP	Remote Desktop Protocol
SEG	Secure Email Gateway
SIEM	Security Information and Event Management
SMB	Server Message Block
SME	Small- and Medium-sized Enterprises
TTP	Tactics, Techniques, and Procedures
UAC	User Account Control
URL	Uniform Resource Locator
VBA	Visual Basic for Applications
VBS	Visual Basic Script
Verizon DBIR	Verizon Data Breach Investigations Report
WDAC	Windows Defender Application Control
WinRM	Windows Remote Management
WMI	Windows Management Instrumentation

APPENDIX F

References and Resources

REFERENCE/RESOURCE	URL
CERT-EU: PowerShell—Cybersecurity Perspective	https://media.cert.europa.eu/static/WhitePapers/CERT-EU-SWP2019-001.pdf
CIS Benchmarks	https://www.cisecurity.org/cis-benchmarks/
CIS Community Defense Model 2.0	https://www.cisecurity.org/white-papers/cis-community-defense-model-2-0/
CIS Controls	https://www.cisecurity.org/controls/
CIS Controls v7.1 Exploited Protocols: Remote Desktop Protocol (RDP)	https://www.cisecurity.org/white-papers/exploited-protocols-remote-desktop-protocol-rdp/
CIS Controls v8 Exploited Protocols: Server Message Block (SMB)	https://www.cisecurity.org/white-papers/cis-controls-v8-exploited-protocols-server-message-block-smb/
CIS Controls v8 Commonly Exploited Protocols: Windows Management Instrumentation (WMI)	https://www.cisecurity.org/insights/white-papers/cis-controls-commonly-exploited-protocols-windows-management-instrumentation
Globe News Wire	https://www.globenewswire.com/news-release/2016/04/12/828009/26208/en/Carbon-Black-United-Threat-Research-Report-Reveals-How-Cyber-Attackers-Exploit-Microsoft-PowerShell-to-Launch-Attacks.html
PowerShell on GitHub	https://github.com/PowerShell/PowerShell
Microsoft PowerShell Documentation	https://docs.microsoft.com/en-us/powershell/
Microsoft PowerShell Team	https://devblogs.microsoft.com/powershell/
Microsoft Antimalware Scan Interface (AMSI)	https://docs.microsoft.com/en-us/windows/win32/amsi/antimalware-scan-interface-portal
MITRE ATT&CK® Framework v8.2	https://attack.mitre.org/versions/v8/matrices/enterprise/

CIS Center for Internet Security®

CIS Controls

The Center for Internet Security, Inc. (CIS®) makes the connected world a safer place for people, businesses, and governments through our core competencies of collaboration and innovation. We are a community-driven nonprofit, responsible for the CIS Critical Security Controls® and CIS Benchmarks™, globally recognized best practices for securing IT systems and data. We lead a global community of IT professionals to continuously evolve these standards and provide products and services to proactively safeguard against emerging threats. Our CIS Hardened Images® provide secure, on-demand, scalable computing environments in the cloud.

CIS is home to the Multi-State Information Sharing and Analysis Center® (MS-ISAC®), the trusted resource for cyber threat prevention, protection, response, and recovery for U.S. State, Local, Tribal, and Territorial government entities, and the Elections Infrastructure Information Sharing and Analysis Center® (EI-ISAC®), which supports the rapidly changing cybersecurity needs of U.S. election offices. To learn more, visit CISecurity.org or follow us on Twitter: @CISecurity.

cisecurity.org
info@cisecurity.org
518-266-3460
Center for Internet Security
@CISecurity
TheCISecurity
cisecurity