Living Off the Land: Powershell

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 Cyber Threat Actors (CTAs) and commonly used in Living-off-the-Land (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.

Attacks Using PowerShell

- CTAs can evade 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)
- CTAs can easily access PowerShell modules that are widely available on many open source platforms.
- CTAs are able to escalate privileges and execute PowerShell scripts via the command-line or in memory, evading detection from traditional anti-malware applications.
- 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 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.

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).

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.

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 AppLocker.

Enterprises may also control access using role-based access control (RBAC) to restrict PowerShell for only specific roles in the enterprise. 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.

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.

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.

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.

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.

Using an application that is behavior-based (e.g., uses heuristic analysis) will allow an enterprise to detect more advanced PowerShell attacks.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.

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.

Email and web browsers are two main avenues that a CTA can use to abuse PowerShell. 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. 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).

CIS BENCHMARK RECOMMENDATIONS:

RECOMMENDATION TITLE	POWERSHELL DEFENSE
(L2) Ensure 'Allow Remote Shell Access' is set to 'Disabled'	Manage your PowerShell environment
(L1) Ensure 'Allow Basic authentication' is set to 'Disabled'	Secure Configurations for PowerShell
(L1) Ensure 'Allow unencrypted traffic' is set to 'Disabled'	Secure Configurations for PowerShell
(L1) Ensure 'Disallow Digest authentication' is set to 'Enabled'	Secure Configurations for PowerShell
(L1) Ensure 'Allow Basic authentication' is set to 'Disabled'	Secure Configurations for PowerShell
(L2) Ensure 'Allow remote server management through WinRM' is set to 'Disabled'	Secure Configurations for PowerShell
(L1) Ensure 'Allow unencrypted traffic' is set to 'Disabled'	Secure Configurations for PowerShell
(L1) Ensure 'Disallow WinRM from storing RunAs credentials' is set to 'Enabled'	Secure Configurations for PowerShell
(L2) Ensure 'Windows Remote Management (WS-Management) (WinRM)' is set to 'Disabled'	Secure Configurations for PowerShell
(L1) Ensure 'Configure Attack Surface Reduction rules' is set to 'Enabled'	Malware Defenses for PowerShell Email and Browser Protections Against Malicious PowerShell Activity
(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
(L1) Ensure 'Turn on PowerShell Script Block Logging' is set to 'Enabled'	PowerShell Logging
(L1) Ensure 'Include command line in process creation events' is set to 'Enabled'	PowerShell Logging
(L1) Ensure 'No auto-restart with logged on users for scheduled automatic updates installations' is set to 'Disabled'	Continuous Vulnerability Management for PowerShell
(L1) Ensure 'Configure Automatic Updates' is set to 'Enabled'	Continuous Vulnerability Management for PowerShell
(L1) Ensure 'Configure Automatic Updates: Scheduled install day' is set to '0 - Every day'	Continuous Vulnerability Management for PowerShell
(L1) Ensure 'Remove access to "Pause updates" feature' is set to 'Enabled'	Continuous Vulnerability Management for PowerShell
(L1) Ensure 'Manage preview builds' is set to 'Disabled'	Continuous Vulnerability Management for PowerShell
(L1) Ensure 'Select when Preview Builds and Feature Updates are received' is set to 'Enabled: 180 or more days'	Continuous Vulnerability Management for PowerShell
(L1) Ensure 'Select when Quality Updates are received' is set to 'Enabled: 0 days'	Continuous Vulnerability Management for PowerShell
(L1) Ensure 'Turn on e-mail scanning' is set to 'Enabled'	Email and Browser Protections Against Malicious PowerShell Activity