Stopping "PowerShell without PowerShell" Attacks

By Stav SettyandAviad Meyer

Feb 09, 2021 5 minutes ... views

Must-Read Articles | Attack | Cortex XDR | Defend | Detect | Powershell

Executive Summary

The Cortex XDR Security Research Team recently observed "PowerShell without PowerShell" activity involving PowerShell commands a scripts that do not directly invoke the powershell.exe binary.

PowerShell commands and scripts can be executed by loading the underlying System.Management.Automation namespace, exposed through the .NET framework and Windows Common Language Interface (CLI). As a result, this eliminates the need to spawn powershell.exe.

These attacks can compromise endpoints even if PowerShell is disabled. Palo Alto Networks Cortex XDR protects customers from these attacks with behavioral detection.

Why PowerShell without PowerShell?

PowerShell is a favored attack tool for multiple reasons, but most notably, attackers often encounter environments where powershell.exe execution isn't possible. In order to overcome this, they can use "PowerShell without PowerShell" tools to bypass application whitelisting and environmental restrictions. This provides the ability to execute any PowerShell script or command in an environment that does not allow for PowerShell execution.

Ultimately, blocking powershell.exe does not stop attackers from executing PowerShell. Furthermore, evasion is another major benefit. Some "PowerShell without PowerShell" tools will enable attackers to execute PowerShell without the security features.

Techniques

The "PowerShell without PowerShell" tools employ a variety of techniques. Some tools enable running PowerShell with DLLs. For instance, we have seen "PowerShdll" and "NoPowerShell" in the wild.

These tools rely on LOLBINs (living-off-the-land binaries) like rundll32.exe, installutil.exe, regsvcs.exe, regasm.exe, and regsvr32.exe to invoke the DLL. These LOLBINs are signed by Microsoft and often whitelisted. However, they are often known for proxy execution of malicious code.

Figure 1. PowerShell Execution

Figure 2. NoPowerShell Execution

As we can see in Figure 1 and Figure 2, after invoking the DLL with rundll32, a new window appears with a PowerShell console, and the powershell.exe binary is not invoked.

Other "PowerShell without PowerShell" tools are binary executables. For example, with "NotPowerShell (nps.exe)", we can run single and multiple encoded and non-encoded commands (Figure 3).

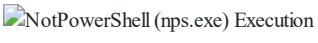

Figure 3. NotPowerShell (nps.exe) Execution

The last section of "PowerShell without PowerShell" tools we have encountered rely on the Microsoft Build Engine (MSBuild) to execute PowerShell scripts and commands. They do this by converting PowerShell scripts and commands to csproj files. "PowerLessShell" is a popular example that we've seen in the wild that features process masquerading with MSBuild disguised as a random executable.

Figure 4. PowerLessShell Execution

Behavioral Activity Observed

Using the Cortex XDR platform, we observed the behavioral activity of these "PowerShell without PowerShell" tools.

DLL Attack Behavior

When diving into the DLL tools, we recognized some unique behavioral traits.

1. The tools we examined call rundll32 with 'main' as EntryPoint
   a. *rundll32 PowerShdll.dll, main*
   b. *rundll32 NoPowerShell.dll,main*
2. LOLBINs create PowerShell script files

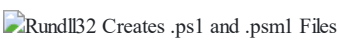

Figure 5. Rundll32 Creates .ps1 and .psm1 Files

3. rundll32.exe spawns conhost.exe. This is an unusual parent-child process relationship and may indicate that an attacker has abused rundll32.exe to run a console-based application.

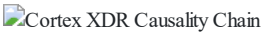

Figure 6. Cortex XDR Causality Chain

4. Unusual module load of *amsi.dll by a LOLBIN (rundll32.exe in this case)*

Figure 7. Cortex XDR Module Loading

MSBuild Attack Behavior

After investigating the MSBuild attacks with Cortex XDR, we noticed the following unusual activity:

1. Process masquerading. MSBuild disguised as a random executable
2. Creation of an executable in the .NET directory

Figure 8. Module Loading PowerShell DLLs with Cortex XDR

3. Executable spawns the C# compiler (csc.exe).

Figure 9. Process Execution Chain with Cortex XDR

4. Executable loads MSBuild DLLs

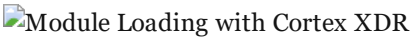

Figure 10. Module Loading with Cortex XDR

Additionally, in almost all of the aforementioned attacks, we noticed the loading of the *System.Management.Automation.dll*. The idea behind this is that powershell.exe is just a process that hosts the *System.Management.Automation.dll*.

Figure 11. Module Loading PowerShell DLLs with Cortex XDR

Cortex XDR Alerts

Following our look into behavioral activity with Cortex XDR, we created the following Behavioral Indicators of Compromise (BIOCs) to detect "PowerShell without PowerShell" tools. Cortex XDR BIOCs can also be configured as prevention rules for greater protection. In addition, the Cortex XDR Agent features Behavioral Threat Protection modules leveraging the Anti-Malware Scan Interface (AMSI) to block PowerShell scripts.

Source	Description
XDR BIOC	Unsigned process loads a known PowerShell DLL
XDR BIOC	Non-PowerShell process accessed the PowerShell history file
XDR BIOC	LOLBIN created a PowerShell script file
XDR BIOC	Rundll32.exe with 'main' as entry point
XDR BIOC	Suspicious .NET process spawns csc.exe
XDR BIOC	Suspicious .NET process loads an MSBuild DLL
XDR BIOC	Suspicious executable created in .NET directory
XDR BIOC	Rundll32.exe spawns conhost.exe
XDR BIOC	Office process loads a known PowerShell DLL
XDR BIOC	Suspicious AMSI DLL load
Cortex XDR Agent	Behavioral Threat Detected
Cortex XDR Agent	WildFire Malware

Table 1. List of PowerShell without PowerShell Alerts

Figure 12. Alerts in the Cortex XDR UI

Conclusion

Overall, disabling and monitoring powershell.exe is not enough to mitigate PowerShell threats. PowerShell is more than just powershell.exe and these "PowerShell without PowerShell" tools are widely used and hard to detect. Cortex XDR™ can overcome this by leveraging behavioral activity to detect and block this attack at several stages of the attack chain.

ATT&CK

The following tactics and techniques are relevant to the threat discussed. Further information can be found in the MITRE ATT&CK framework.

ID	Description	Tactic
T1059.001	**Command and Scripting Interpreter: PowerShell** • Adversaries may abuse PowerShell commands and scripts for execution	Execution
T1218.011	**Signed Binary Proxy Execution: Rundll32** • Adversaries may abuse rundll32.exe to proxy execution of malicious code	Defense Evasion
T1127.001	**Trusted Developer Utilities Proxy Execution: MSBuild** • Adversaries may use MSBuild to proxy execution of code through a trusted Windows utility.	Defense Evasion
T1036	**Masquerading**	Defense Evasion

	• Renaming abusable system utilities to evade security monitoring	

Table 2. Relevant ATT&CK Techniques

RELATED BLOGS

MUST-READ ARTICLES, PRODUCT FEATURES, USE-CASES

Boosting Identity Security with Cortex XDR/XSIAM Honey Users

MUST-READ ARTICLES, PRODUCT FEATURES

Threat Hunting with Mark of The Web Using Cortex XDR

MUST-READ ARTICLES, UNCATEGORIZED

Exploring the Art and Science of Threat Hunting with Oded Awaskar

ANNOUNCEMENT, MUST-READ ARTICLES, NEWS AND EVENTS, PRODUCT FEATURES

What's Next in Cortex - New Wave of Innovations in Cortex (June 2024 Release)

ANNOUNCEMENT, MUST-READ ARTICLES, NEWS AND EVENTS, PRODUCT FEATURES, PRODUCTS AND SERVICES

Forrester Names Palo Alto Networks a Leader in XDR

COMPANY & CULTURE, CUSTOMER SPOTLIGHT, MUST-READ ARTICLES, PRODUCT FEATURES, PRODUCTS AND SERVICES, USE-CASES

AI Powers Sabre's Enhanced Threat Detection & Response

Products and Services