PSv5 Scriptblock Logging

Scriptblock Logging - Introduction

- Introduced in PowerShell v5
- Creates a 4104 event in the Microsoft-Windows-PowerShell/Operational event log whenever a scriptblock is invoked.
- Effective at evading wrapped obfuscation.
- Can be applied to the system and user context.
- Logs all scriptblock invocation vs. auto-logging which capture scriptblocks of “suspicious” commands.
  - Obfuscation can often circumvent scriptblock auto-logging.
  - Auto-logging might also miss important attack context.

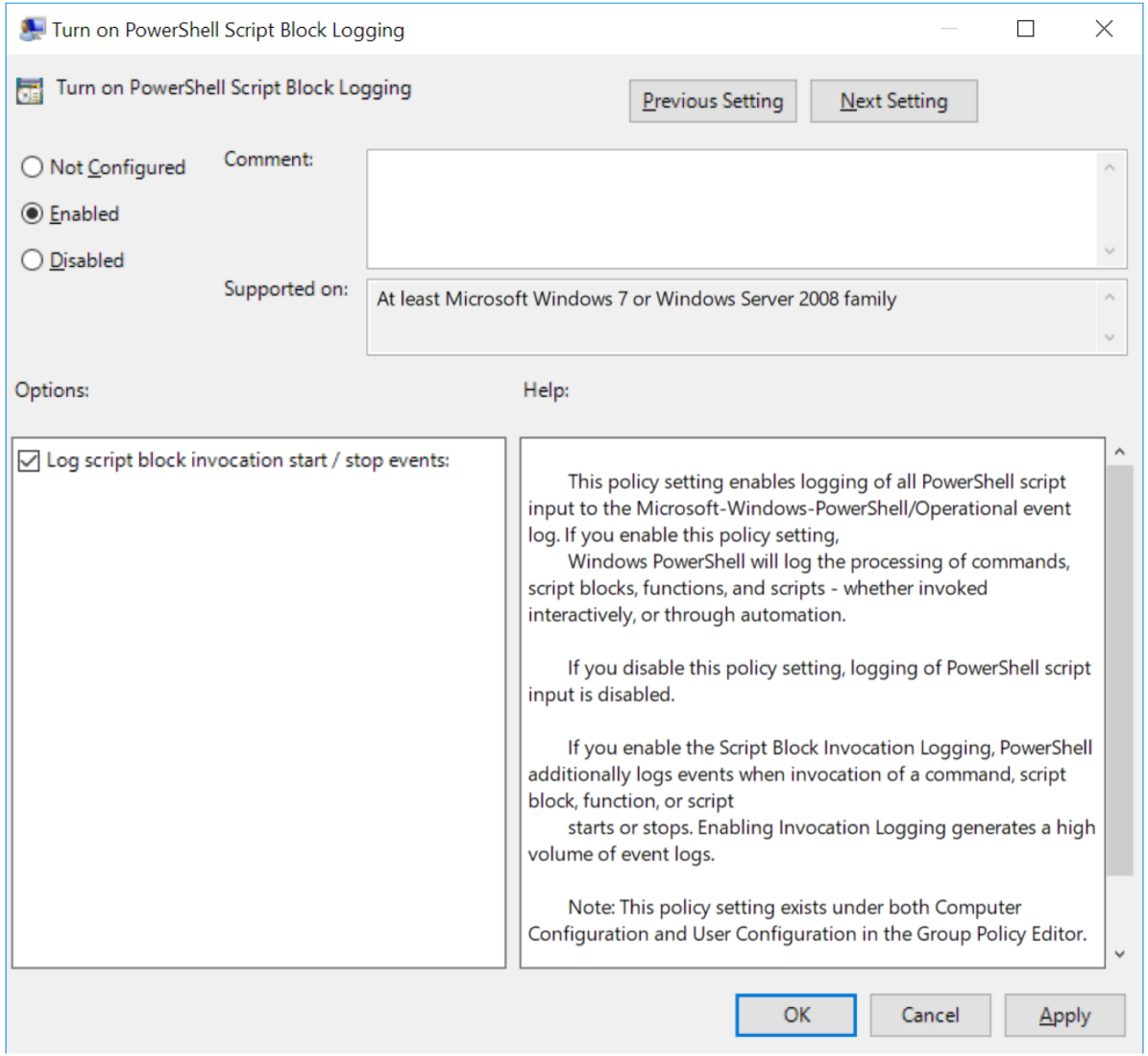
Scriptblock Logging - Configuration

- Can be enabled via GPO or the registry directly.
- Administrative Templates ->

  Windows Components ->

  Windows PowerShell -

  Turn on PowerShell Script Block Logging
- HKLM:\Software\Policies\Microsoft\Windows\PowerShell\ScriptBlockLogging - EnableScriptBlockLogging


Turn on PowerShell Script Block Logging
Turn on PowerShell Script Block Logging
Previous Setting
Next Setting
Not Configured
Enabled
Disabled
Comment:
Supported on:
At least Microsoft Windows 7 or Windows Server 2008 family
Options:
Help:
Log script block invocation start / stop events:
This policy setting enables logging of all PowerShell script input to the Microsoft-Windows-PowerShell/Operational event log. If you enable this policy setting,
Windows PowerShell will log the processing of commands, script blocks, functions, and scripts - whether invoked interactively, or through automation.
If you disable this policy setting, logging of PowerShell script input is disabled.
If you enable the Script Block Invocation Logging, PowerShell additionally logs events when invocation of a command, script block, function, or script
starts or stops. Enabling Invocation Logging generates a high volume of event logs.
Note: This policy setting exists under both Computer Configuration and User Configuration in the Group Policy Editor.
OK
Cancel
Apply

Scriptblock Logging - Auditing

Windows PowerShell

```
PS C:\> Get-Item -Path HKLM:\Software\Policies\Microsoft\Windows\PowerShell\ScriptBlockLogging


    Hive: HKEY_LOCAL_MACHINE\Software\Policies\Microsoft\Windows\PowerShell


Name                           Property
----                           --------
ScriptBlockLogging             EnableScriptBlockLogging           : 1
                               EnableScriptBlockInvocationLogging : 1


PS C:\>
```

Scriptblock Logging - Auditing

Windows PowerShell

```
PS C:\> [IntPtr]::Add(0,1)
1
PS C:\> Get-WinEvent -LogName Microsoft-Windows-PowerShell/Operational -FilterXPath '*[System[EventID=4
104]]' | Select-Object -First 1 | Format-List


TimeCreated  : 10/23/2017 3:26:47 PM
ProviderName : Microsoft-Windows-PowerShell
Id           : 4104
Message      : Creating Scriptblock text (1 of 1):
               [IntPtr]::Add(0,1)

               ScriptBlock ID: 932edb44-9217-437b-906f-35c497ac88c3
               Path:
```

Scriptblock Logging - Implementation

```
internal static void LogScriptBlockStart(ScriptBlock scriptBlock, Guid runspaceId)
{
    bool force = false;
    if (scriptBlock._scriptBlockData.HasSuspiciousContent)
    {
        force = true;
    }
    ScriptBlock.LogScriptBlockCreation(scriptBlock, force);
    if (ScriptBlock.ShouldLogScriptBlockActivity("EnableScriptBlockInvocationLogging"))
    {
        PSEtwLog.LogOperationalVerbose(PSEventId.ScriptBlock_Invoke_Start_Detail, PSOpc
        {
            scriptBlock.Id.ToString(),
            runspaceId.ToString()
        });
    }
}
```

Scriptblock Logging - Implementation

- Scriptblock logging settings are cached in a cachedGroupPolicySettings object presumably for performance reasons.

- What if you could somehow overwrite the cached settings to indicate that scriptblock logging is not enabled?
- Ryan Cobb again has you covered…
  - https://cobbr.io/ScriptBlock-Logging-Bypass.html

Scriptblock Logging - Bypass Methodology

- Observe all code paths that check for scriptblock logging being enabled.
- Identify the conditions where logging does occur and where logging might not occur.
- Can an attacker somehow influence the code paths using reflection or some other technique?

Scriptblock Logging - Bypass Methodology Examples

- At this point, we are going to assess the attack surface together by looking at some bypass weaponization.
- Example bypasses:
  1. cachedGroupPolicySettings
  2. scriptBlock.HasLogged
  3. scriptBlock.ScriptBlockData.IsProductCode

Scriptblock Logging - Bypass Mitigations

- With scriptblock logging enabled, at a minimum, the bypasses will be logged.
- Most PowerShell-specific bypasses will likely require reflection which is mitigated with constrained language mode enforcement.
  - There is no other PowerShell-specific prevention technique.
- An elevated attack can obvious set the policy registry key/values.
  - Registry SACLs can detect these changes.
- None of these bypasses should be fixed as there are no actual logic flaws. An attacker is taking advantage of the fact that PowerShell grants arbitrary code execution when not running constrained language mode.