



Center
for Threat
Informed
Defense

Summitting the Pyramid

v1.0.0

CONTENTS

Overview

Introduction

Definitions

Model Mapping Pages

Combining Observables

Example Mappings

How to Score an Analytic

Analytics Repository

Suspicious ADFind

Scheduled Task/Job

Service Registry Permissions Weakness Check

Original Analytic Scoring

Improved Analytic Scoring

Potential Access Token Abuse

Capability Abstraction

Future Work

Acknowledgements

Changelog

— / [Analytics Repository](#)

/ Service Registry Permissions Weakness Check

- https://github.com/SigmaHQ/sigma/blob/master/rules/windows/powershell/powershell_script/posh_ps_get_acl_service.yml

```
title: Service Registry Permissions Weakness Check
id: 95afc12e-3cbb-40c3-9340-84a032e596a3
status: test
description: |
    Adversaries may execute their own malicious payloads
    Adversaries may use flaws in the permissions for regi
    Windows stores local service configuration informatio
references:
    - https://github.com/redcanaryco/atomic-red-team/blob
    - https://docs.microsoft.com/en-us/powershell/module/
author: frack113
date: 2021/12/30
tags:
    - attack.persistence
    - attack.t1574.011
logsource:
    product: windows
    category: ps_script
    definition: 'Requirements: Script Block Logging must
detection:
    selection:
        ScriptBlockText|contains|all:
            - 'get-acl'
            - 'REGISTRY::HKLM\SYSTEM\CurrentControlSet\Se
    condition: selection
falsepositives:
    - Legitimate administrative script
level: medium
```

III

MITRE

ENGENUITY.

Center for Threat Informed Defense

Summiting the Pyramid

v1.0.0

CONTENTS

Overview

Introduction

Definitions

Model Mapping Pages

Combining Observables

Example Mappings

How to Score an Analytic

Analytics Repository

Suspicious ADFind

Scheduled Task/Job

Service Registry Permissions Weakness Check

Original Analytic Scoring

Improved Analytic Scoring

Potential Access Token Abuse

Capability Abstraction

Future Work


Acknowledgements

Changelog

	Application (A)	User-mode (U)	Kernel-mode (K)
Core to (Sub-) Technique (5)			
Core to Part of (Sub-) Technique (4)			
Core to Pre-Existing Tool (3)			
Core to Adversary-brought Tool (2)	EventID: 4104 ScriptBlockText contains all: - 'get-acl' - 'REGISTRY::HKLM\SYSTEM\CurrentControlSet\		
Ephemeral (1)			

	Application (A)	User-mode (U)	Kernel-mode (K)
Core to (Sub-) Technique (5)			EventID: 4663 TargetObject: "*SYSTEM\Current
Core to Part of			

Page 2 of 5



Center
for Threat
Informed
Defense

Summitting the Pyramid

v1.0.0

CONTENTS

- Overview
- Introduction
- Definitions
- Model Mapping Pages
- Combining Observables
- Example Mappings
- How to Score an Analytic

Analytics Repository

- Suspicious ADFind
- Scheduled Task/Job
- Service Registry Permissions Weakness Check**
 - Original Analytic Scoring
 - Improved Analytic Scoring
- Potential Access Token Abuse

- Capability Abstraction
- Future Work
- Acknowledgements
- Changelog

(Sub-) Technique (4)			
Core to Pre-Existing Tool (3)			
Core to Adversary-brought Tool (2)			
Ephemeral (1)			

This analytic uses the Windows PowerShell logging Event ID 4104 and detects on specific values in the ScriptBlockText field¹. While the specified registry key is core to the subtechnique², the actual observable is the string representation of that registry key inside the script text. It is relatively easy for an attacker to obfuscate keywords or values in a PowerShell script. For example, the cmdlet `get-ac1` is defined and included in the Microsoft.PowerShell.Security module, but equivalent functionality can be accomplished with a renamed or custom cmdlet that doesn't require `get-ac1` exist in the script text. The registry key string can be obfuscated in other ways³, several of which are shown below. Since the adversary can modify their tools and associated scripts before deployment to evade this analytic, it is **2A**.

```
# Let's start with a simple example:
function Invoke-Malware {
    Write-Host 'Malware!';
}

# Simple signature: if script contains "Write-Host 'Malwa
# Simple bypass:
function Invoke-Malware {
    Write-Host "Malware!";
```



Center
for Threat
Informed
Defense

Summitting the Pyramid

v1.0.0

CONTENTS

- Overview
- Introduction
- Definitions
- Model Mapping Pages
- Combining Observables
- Example Mappings
- How to Score an Analytic

Analytics Repository

- Suspicious ADFind
- Scheduled Task/Job

Service Registry Permissions Weakness Check

- Original Analytic Scoring
- Improved Analytic Scoring

Potential Access Token Abuse

- Capability Abstraction
- Future Work
- Acknowledgements
- Changelog

```
}

# Simple signature: if re.findall("Write-Host .Malware.",
# Simple bypass:
function Invoke-Malware {
    Write-Host ("Mal" + "ware!");
}

# Let's start being a little more sophisticated (just a b
function Invoke-NotMalware {
    $malware_base64 = "V3JpdGUtSG9zdCAiTWFs2FyZSEi";
    $malware = [System.Text.Encoding]::ASCII.GetString([Sys
    IEX ($malware);
}

# Simple signature:
# if script contains "V3JpdGUtSG9zdCAiTWFs2FyZSEi" → Mal
# Simple bypass:
function Invoke-NotMalware {
    $malware_base64 = "VwByAGkAdABlAC0ASABvAHMAdAAgACIATQBh
    $malware = [System.Text.Encoding]::UNICODE.GetString([S
    IEX ($malware);
}

# Security solutions are able to emulate base64 decoding
# So malware authors move to algorithm based obfuscation
$key = 0x64
$encodedMalware = "M2QWZA1kEGQBZE1kLGQLZBdkEGREZEZkKWQFZA
$bytes = [Convert]::FromBase64String($encodedMalware)
$decodedBytes = foreach ($byte in $bytes) {$byte -bxor $k
$decodedMalware = [System.Text.Encoding]::Unicode.GetStri
IEX ($decodedMalware)
```

A more robust way of detecting the original behavior involves setting a Security Access Control List (SACL) on the registry key. Setting a SACL on the registry key enables using a kernel-mode data source to detect the `get-acl` behavior of a script without looking at the contents of the script itself. Once the SACL is set and configured, an EventID 4663 will be generated whenever an attempt is made to access the registry key.

Note

SACLs have configuration options which can change the precision of an analytic. One configuration option is to



Center
for Threat
Informed
Defense

Summiting the Pyramid

v1.0.0

CONTENTS

- Overview
- Introduction
- Definitions
- Model Mapping Pages
- Combining Observables
- Example Mappings
- How to Score an Analytic

Analytics Repository

- Suspicious ADFind
- Scheduled Task/Job

Service Registry Permissions Weakness Check

- Original Analytic Scoring
- Improved Analytic Scoring

Potential Access Token Abuse

- Capability Abstraction
- Future Work
- Acknowledgements
- Changelog

log the “Full Control” set of activity and get a complete view of registry key activity, and then query those results for when the registry key is read (when the `AccessMask` field has the corresponding value `READ_CONTROL`⁴). However, this approach could generate a large amount of benign noise. As an alternative, the SACL can be configured to generate an event only when the key is read.

References

- [1] https://github.com/OTRF/OSSEM-DD/blob/5e16ccfe548c8c0249430247a99e213636b2a5a5/windows/etw-providers/Microsoft-Windows-PowerShell/events/event-4104_v1.yml#L22
- [2] <https://attack.mitre.org/techniques/T1574/011/>
- [3] <https://i.blackhat.com/briefings/asia/2018/asia-18-Tal-Liberman-Documenting-the-Undocumented-The-Rise-and-Fall-of-AMSI.pdf>
- [4] https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-lsad/5ee8db78-5f0e-47b2-aba7-8447ff454e3b

© Copyright 2023, Center for Threat-Informed Defense.