Purple Perspective: Execution Methods

Tim Schulz
@teschulz

SCYTHE

Bio

Adversary Emulation Lead

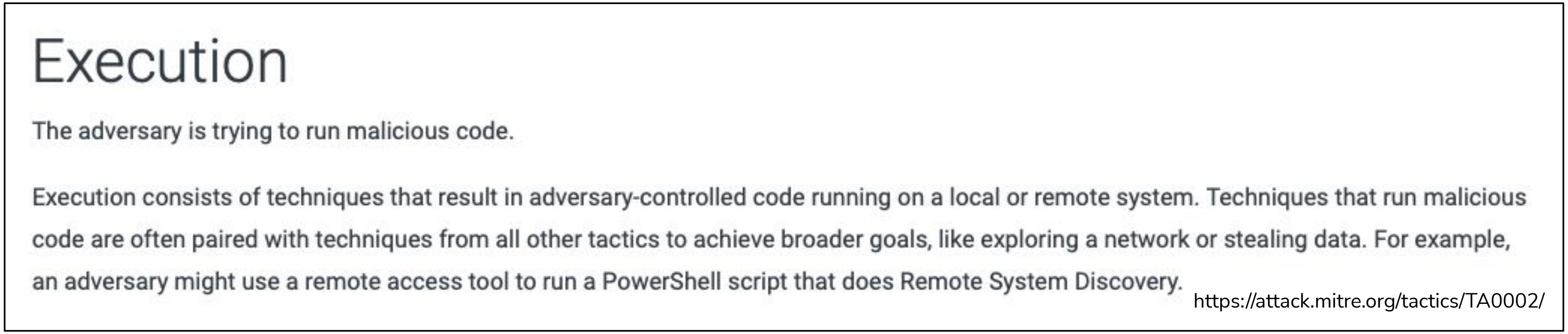
- Security Research
- Red Teaming
- Purple Teaming
- ICS/OT

- Adversary Emulation
- Purple Teaming
- Red Teaming

3 Things I hope you take away from this talk

- Why execution methods make security challenging
- Mindset for testing execution methods
- Lots of resources for how to test different types of execution methods

What are execution methods?

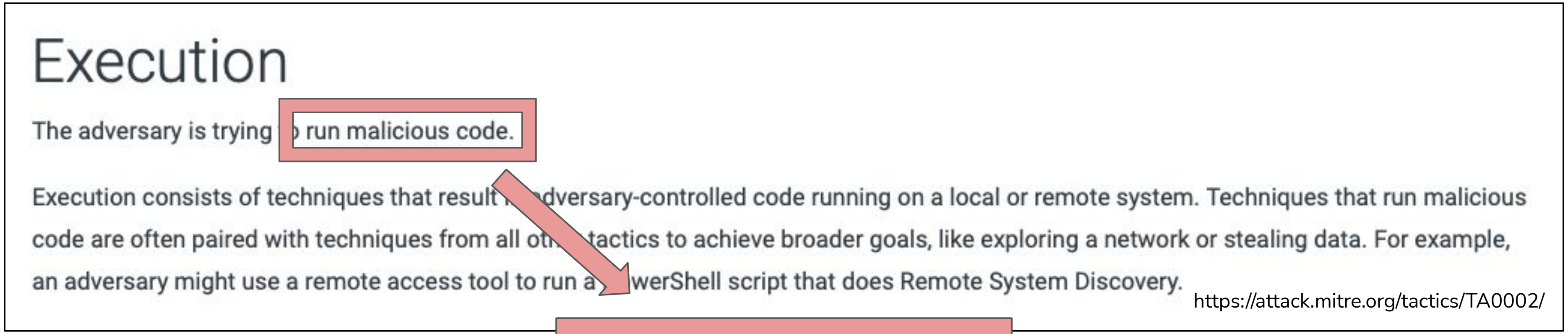

Execution

The adversary is trying to run malicious code.

Execution consists of techniques that result in adversary-controlled code running on a local or remote system. Techniques that run malicious code are often paired with techniques from all other tactics to achieve broader goals, like exploring a network or stealing data. For example, an adversary might use a remote access tool to run a PowerShell script that does Remote System Discovery.

https://attack.mitre.org/tactics/TA0002/

What are execution methods?



Execution

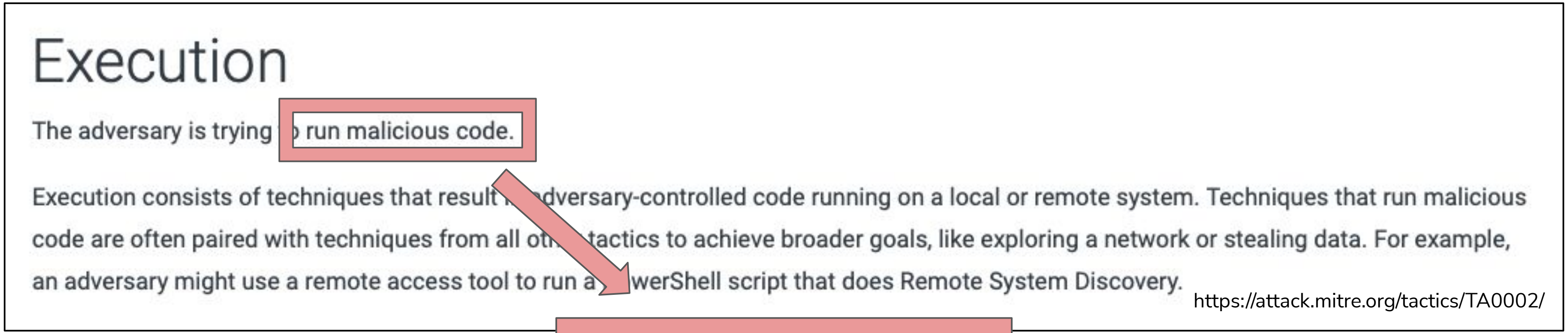
The adversary is trying to run malicious code.

Execution consists of techniques that result in adversary-controlled code running on a local or remote system. Techniques that run malicious code are often paired with techniques from all other tactics to achieve broader goals, like exploring a network or stealing data. For example, an adversary might use a remote access tool to run a PowerShell script that does Remote System Discovery.

https://attack.mitre.org/tactics/TA0002/

Running code

What are execution methods?



Execution

The adversary is trying to run malicious code.

Execution consists of techniques that result in adversary-controlled code running on a local or remote system. Techniques that run malicious code are often paired with techniques from all other tactics to achieve broader goals, like exploring a network or stealing data. For example, an adversary might use a remote access tool to run a PowerShell script that does Remote System Discovery.

https://attack.mitre.org/tactics/TA0002/

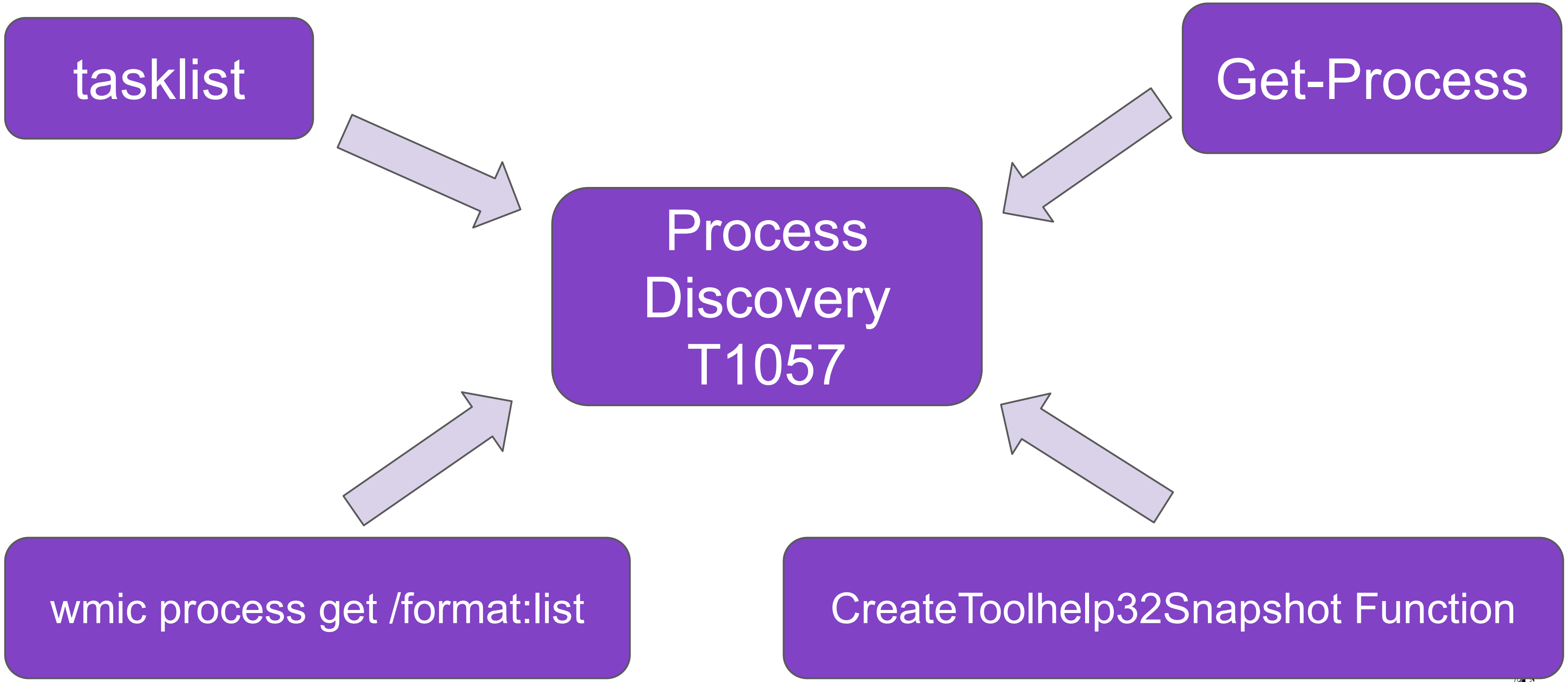
Running code

Windows focus here, but challenges and test approaches are widely applicable

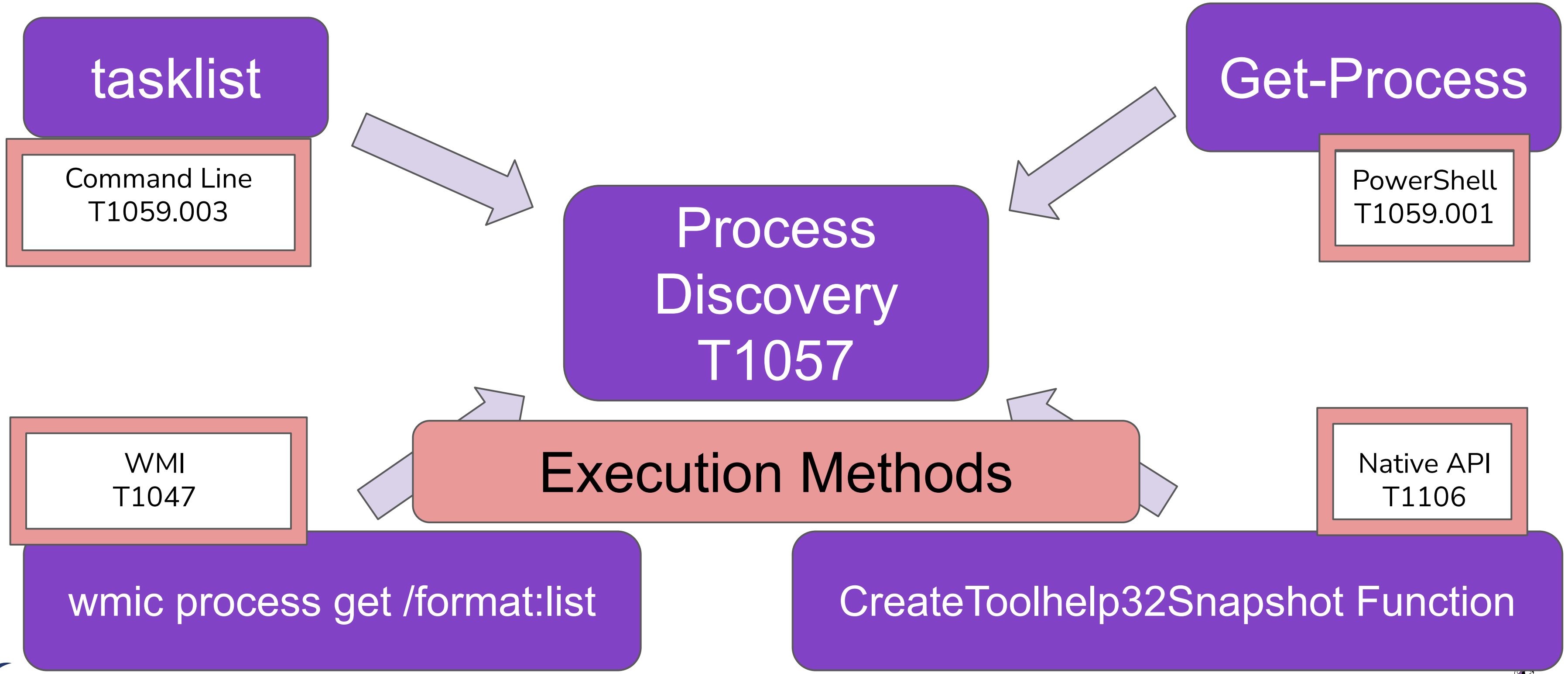
Execution Methods: Process Discovery (T1057)

Process
Discovery
T1057

Execution Methods: Process Discovery (T1057)


tasklist
Get-Process
Process Discovery T1057
wmic process get /format:list
CreateToolhelp32Snapshot Function

Execution Methods: Process Discovery (T1057)

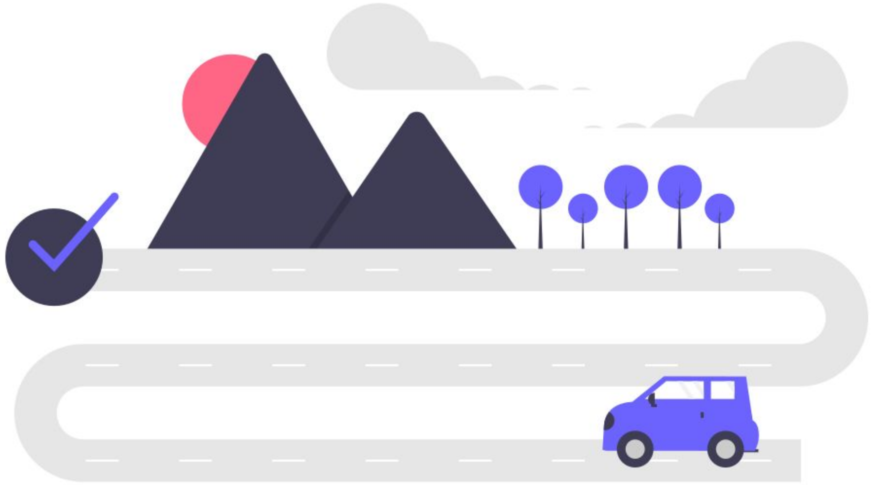


Why do execution methods matter?

- ✔ Flexibility in expertise
- ✔ Telemetry inconsistencies
- ✔ Require determining intent
- ✔ Operating systems offer A LOT of ways to execute code

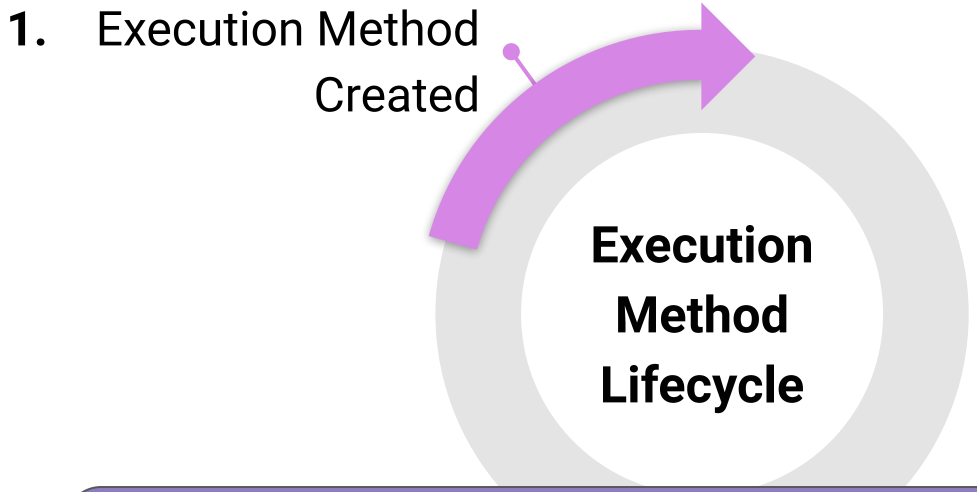
Roadmap: What are we covering today?

- Cycle of Execution Methods
- Testing Environments and Tools
- Windows PowerShell
  - Execution Method Variance
- LOLBAS Project
  - MSBuild



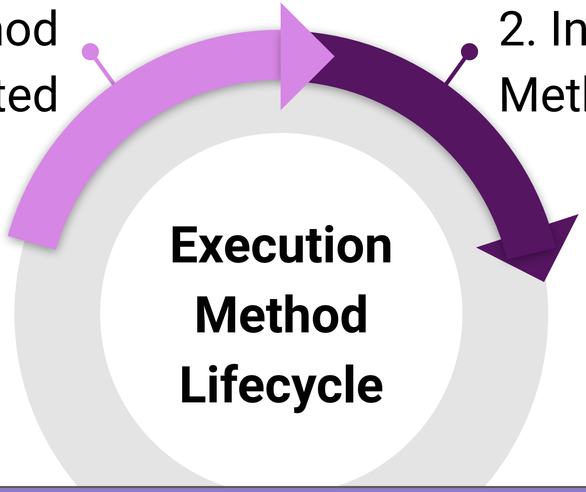
Cycle of Execution Methods

Execution Method Lifecycle: PowerShell



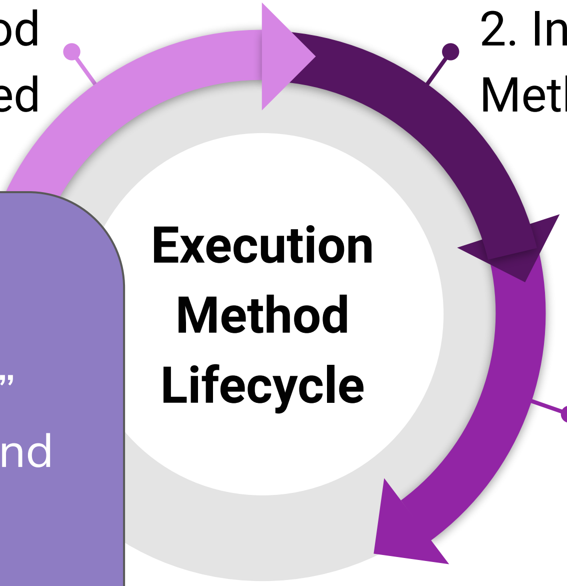
- PowerShell 1.0 Released (2006)

Execution Method Lifecycle: PowerShell


1. Execution Method Created
2. Intended Use of Execution Method
Execution Method Lifecycle

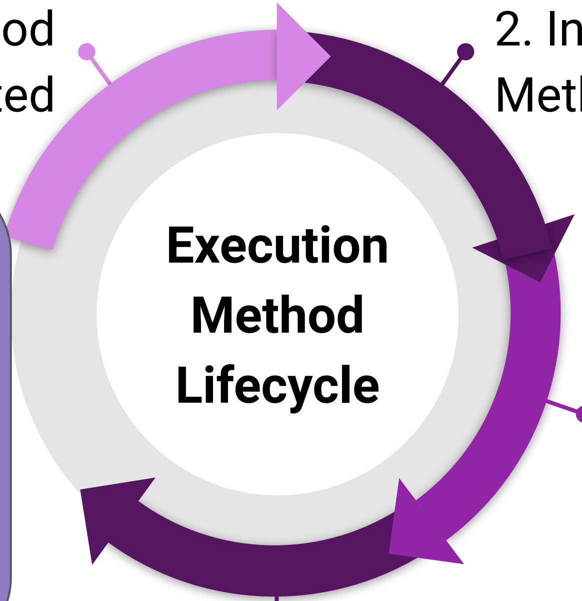
- PowerShell 2.0 Released (2009)
  - Included in Windows 7 by default

Execution Method Lifecycle: PowerShell


1. Execution Method Created
2. Intended Use of Execution Method
3. Unintended Use of Execution Method
Execution Method Lifecycle

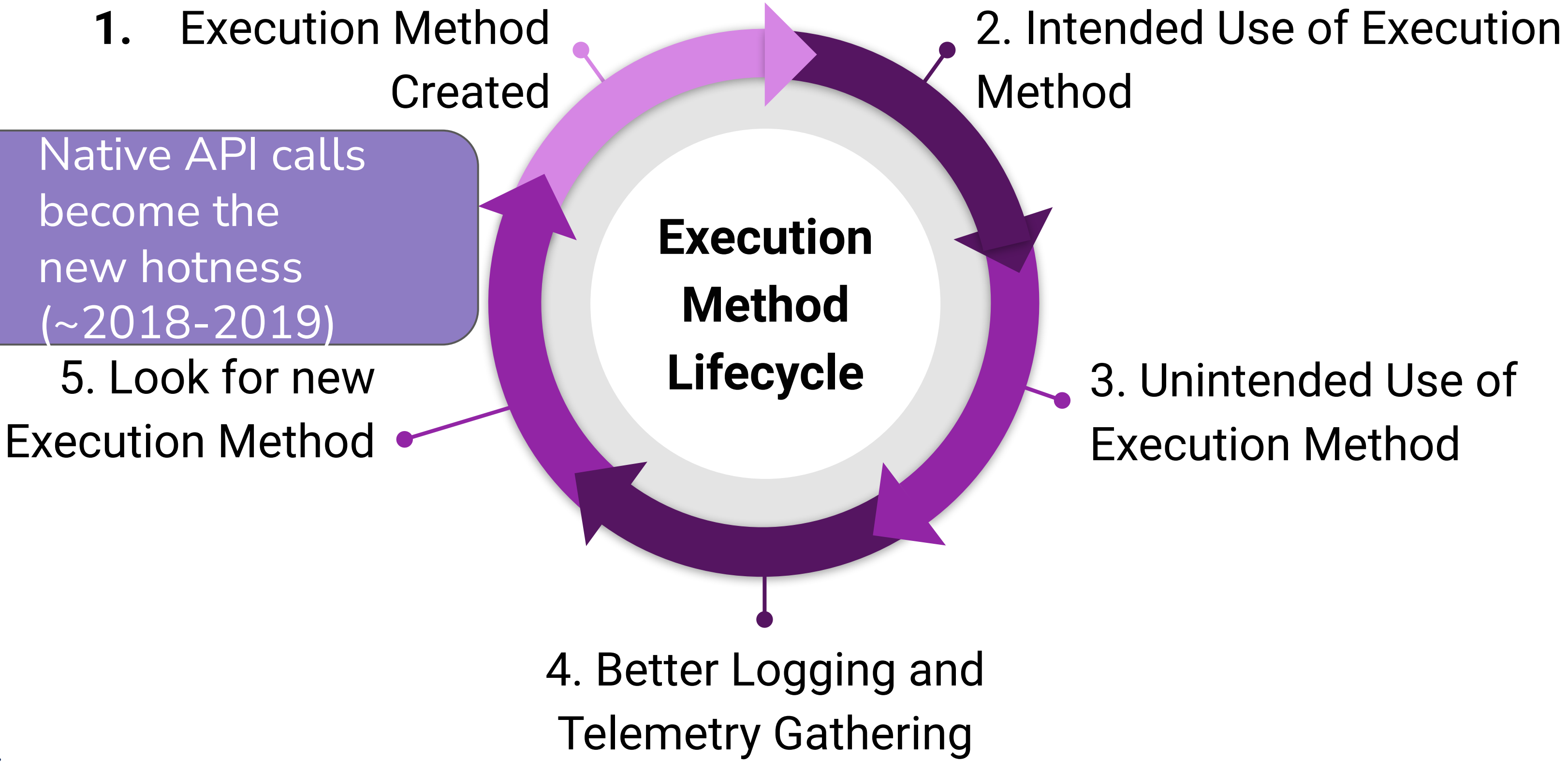
- DerbyCon Talk: “PowerShell…omfg” by Dave Kennedy and Josh Kelly (2013)
- PowerShell Empire Created (2015)

Execution Method Lifecycle: PowerShell


1. Execution Method Created
2. Intended Use of Execution Method
3. Unintended Use of Execution Method
4. Better Logging and Telemetry Gathering
Execution Method Lifecycle

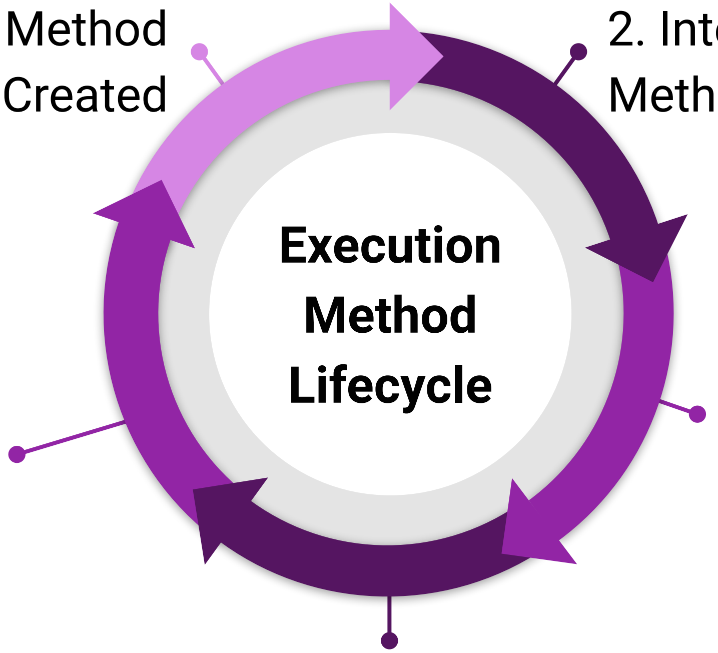
- Enhanced Logging in PSv4 & 5 through Windows Management Framework (2015+)
    - CLM, Module, Script Block Logging

Execution Method Lifecycle: PowerShell


1. Execution Method Created
2. Intended Use of Execution Method
3. Unintended Use of Execution Method
4. Better Logging and Telemetry Gathering
5. Look for new Execution Method
Execution Method Lifecycle

- Native API calls become the new hotness (~2018-2019)

Lifecycle of an Execution Method



1. Execution Method Created

2. Intended Use of Execution Method

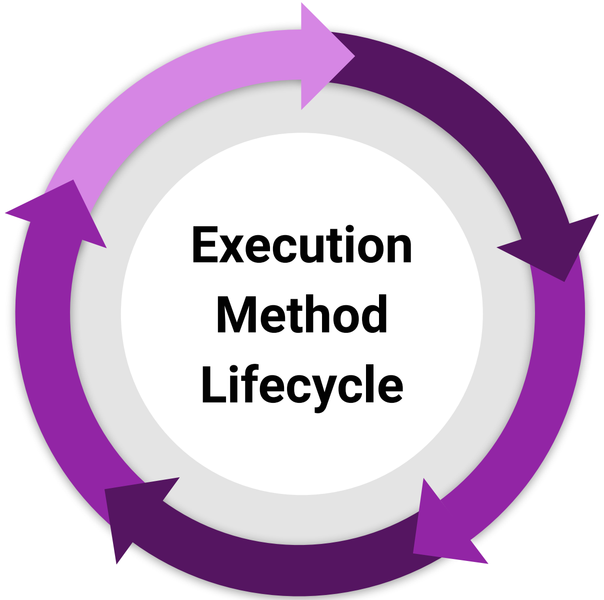
3. Unintended Use of Execution Method

4. Better Logging and Telemetry Gathering

5. Look for new Execution Method

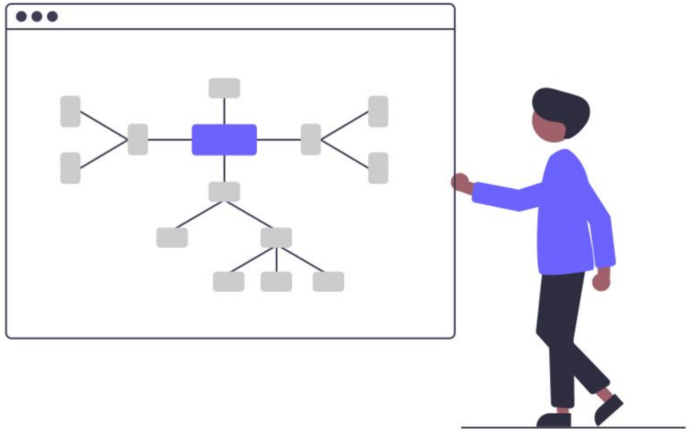
A compounding problem..

- Covering all previous execution methods
- While tackling new methods
- With potentially changing telemetry/data



Purple Perspective

- Execution methods provide a known path for adversary capability
- Previous execution methods provide a maturity map for defenders



Solution?

Testing Methods and Tools

Execution Testing Advice

- **Focus on one question at a time**
  - Break big questions into smaller ones
- Pick a technique and test it thoroughly
  - Discovery techniques are a good starting point

Execution Testing Advice

- **Focus on one question at a time**
  - Break big questions into smaller ones
- Pick a technique and test it thoroughly
  - Discovery techniques are a good starting point

Will Process Discovery (T1057) generate an alert?

Our big question

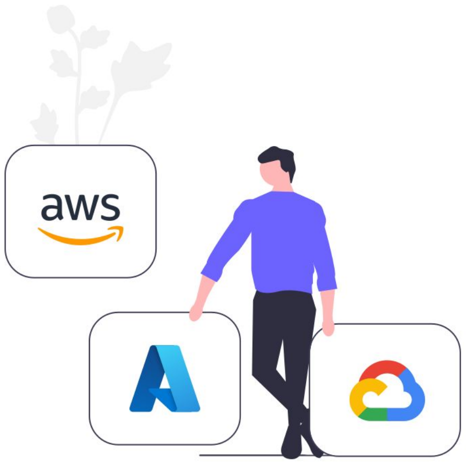
Breaking a big question into smaller ones

Will Process Discovery (T1057) generate an alert?

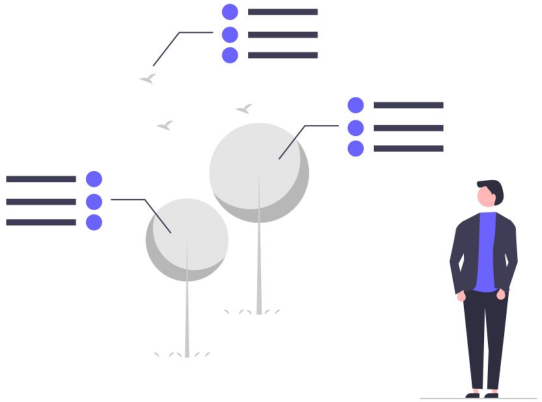
- How to execute the technique?
- What artifacts are generated?
- What changes can be made?

- Data source for execution?
- Data correlation for context?
- Data analysis for generating alert?

Execution Testing Setup



Environment



Data Collection

Data Generation

Test Environments

- Ideally your production environment...
- Virtual Machines, Online Cyber Ranges, Cloud Providers, etc..
- Attack Range by Splunk
  - https://github.com/splunk/attack_range
- Game of Active Directory by Orange Cyber Defense
  - https://github.com/Orange-Cyberdefense/GOAD
- DetectionLab by Chris Long
  - https://github.com/clong/DetectionLab
- Active Directory Ranges by Immersive Labs/SnapLabs
  - https://www.snaplabs.io
- Building Virtual Machine Labs: A Hands On Guide by Tony Robinson
  - https://leanpub.com/avatar2

Data Generation Questions

- ✔ How to execute the technique?
- ✔ What artifacts are generated?
- ✔ What changes can be made?

Data Generators

- Aggregation of a lot of projects: The C2 Matrix
  - https://www.thec2matrix.com
- SANS Slingshot VM: The C2 Matrix Edition
  - https://www.sans.org/tools/slingshot/
- Atomic Red Team by Red Canary
  - https://github.com/redcanaryco/atomic-red-team
- CALDERA by MITRE
  - https://github.com/mitre/caldera
- PurpleSharp by Mauricio Velazco
  - https://github.com/mvelazc0/PurpleSharp
- Blackhat Python or Go books from No Starch Press
  - https://nostarch.com/search/black%20hat

Pick one or two and try them out!

Data Collection Questions

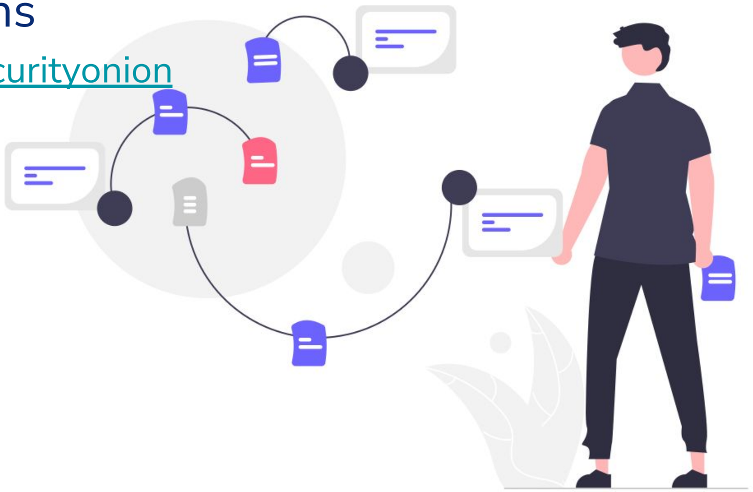
- ✔ Data source for execution?
- ✔ Data correlation for context?
- ✔ Data analysis for generating alert?

Data Collectors

- EDRs and other production tooling collect or can collect a lot of data already
  - May need to turn on a feature instead of deploying something new
- For lab environments, leveraging DetectionLab or Attack Range will automatically instrument the environment with sensors
- Sysmon is a great data collector
  - https://github.com/SwiftOnSecurity/sysmon-config
  - https://github.com/olafhartong/sysmon-modular
- Execution methods tend to happen on the host, however network data can provide crucial information
- Zeek by The Zeek Project
  - https://github.com/zeek/zeek
- Winlogbeats and other beats by Elastic
  - https://www.elastic.co/beats/

Maturing Test Setup: Analysis

- Key question: **What data can you group together that improves analysis?**
- Correlating and enriching data with context is crucial
  - Sysmon/EDRs do some already, but don't solve every problem
  - Process trees are your friend!
- Security Onion by Security Onion Solutions
  - https://github.com/Security-Onion-Solutions/securityonion
- ELK by Elastic
  - https://www.elastic.co/what-is/elk-stack
  - RedELK by Outflank: https://github.com/outflanknl/RedELK
  - SOF-ELK by SANS: https://github.com/philhagen/sof-elk



SCYTHE

Alert Generation

- EDRs have varying degrees out of the box
- Sigma by Florian Roth
  - https://github.com/SigmaHQ/sigma
  - Port SIGMA rules to other formats: https://uncoder.io
  - Aurora Lite free detections: https://www.nextron-systems.com/aurora/
- Detection-rules by Elastic
  - https://github.com/elastic/detection-rules

PowerShell

PowerShell (T1059.001)

- PowerShell is still a security challenge in 2022
- PowerShell Execution of Process Discovery
  - Get-Process via Atomic Red Team Test #3 of T1057[1]
- Where are the logs?
  - Script Block Logging* - Event ID 4104
  - Sysmon Event ID 1
- Is there missing information?
  - Process trees?
    - Check out Process Explorer from Windows Sysinternals
    - Explorer vs not_a_beacon.exe
  - Differences between Atomic Red Team vs through C2 framework?
- Should we disable PowerShell?
  - Probably not, see Keeping PowerShell: Security Measures to Use and Embrace[2]

[1]https://github.com/redcanaryco/atomic-red-team/blob/master/atomics/T1057/T1057.md
[2]https://media.defense.gov/2022/Jun/22/2003021689/-1/-1/1/CSI_KEEPING_POWERSHELL_SECURITY_MEASURES_TO_USE_AND_EMBRACE_20220622.PDF

Variation for execution methods

- Exploring the features and capabilities of the execution method
- Sometimes documented, other times it takes research and discovery
- What assumptions are being made about how it works?

Can we run something without being logged?

Can we obfuscate what we run?

Can we copy/move?

Can we rename?

Can we bring our own?

Variation for execution methods: PowerShell

Can we run something without being logged?

- PowerShell v2, before all the security features!

```
powershell -v 2 Get-Process
```

- Truncated logging? Logging only the first 500 characters to reduce log size

```
$500spaces = (“ ” * 500) + ‘Get-Process’
Invoke-Expression $500spaces
```

Does our telemetry change with any of these variations?

Variation for execution methods: PowerShell

Can we obfuscate what we run?

- Encoded Command (-encodedCommand, -enc, -ec)

```
$encodedcommand = [Convert]::ToBase64String([Text.Encoding]::Unicode.GetBytes('Get-Process'))
powershell.exe -encodedCommand RwBlAHQALQBQAHIAbwBjAGUAcwBzAA==
```

- Invoke-Obfuscation: https://github.com/danielbohannon/Invoke-Obfuscation

```
powershell g`Et`-pROcesS
```

Does our telemetry change with any of these variations?

Variation for execution methods: PowerShell

- All tests up to this point have assumed that powershell.exe from the System path is what is being used to execute commands

Can we copy/move?

```
Copy C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe $env:UserProfile\powershell.exe
./$env:UserProfile\powershell.exe Get-Process
```

Can we rename?

```
Copy C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe $env:UserProfile\ps.exe
ps.exe Get-Process
```

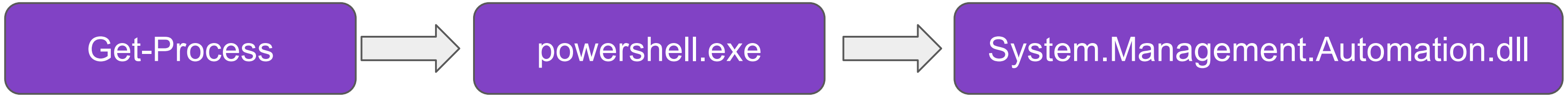
Does our telemetry change with any of these variations?

Variation for execution methods: PowerShell

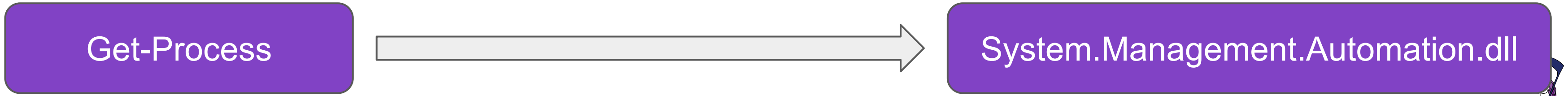
Unmanaged PowerShell

- Lee Christianson created the project in 2014
  - https://github.com/leechristensen/UnmanagedPowerShell
- “PowerPick” in Cobalt Strike in 2016

Normal PowerShell



Unmanaged PowerShell



Variation for execution methods: PowerShell

Can we bring our own?

- Download System.Management.Automation.dll and reference it

Can we obfuscate what we bring?

- Change bytes to change the hash while also maintaining the Microsoft Signed Binary status
- https://github.com/Mr-Un1k0d3r/Windows-SignedBinary

PowerShell (T1059.001) Additional Resources

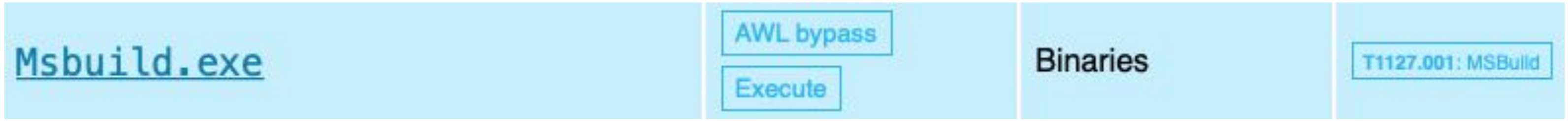
- BC Security's fork of Empire
  - Base Project: https://github.com/BC-SECURITY/Empire
  - Starkiller GUI: https://github.com/BC-SECURITY/Starkiller
- **Free** Adversary Tactics: PowerShell course by SpecterOps
  - https://github.com/specterops/at-ps

LOLBAS Project

LOLBAS

- Started off as Living off the Land Binaries (LOLBINS)
  - Initially coined by Matt Graeber
  - Lots of early public research done by Casey Smith
- Now is Living off the Land Binaries and Scripts (LOLBAS)
  - https://lolbas-project.github.io
- From the Github:
  - A LOLBin/Lib/Script must:
    - Be a Microsoft-signed file, either native to the OS or downloaded from Microsoft.
    - Have extra "unexpected" functionality. It is not interesting to document intended use cases.
    - Exceptions are application whitelisting bypasses
    - Have functionality that would be useful to an APT or red team

MSBuild (T1127.01)



Msbuild.exe	AWL bypass Execute	Binaries	T1127.001: MSBuild

- While not on Windows “out of the box”, can be installed by Windows based applications

```
C:\Windows\Microsoft.NET\Framework\v4.0.30319\MSBuild.exe .\Get-Process-CSharp-32.xml
```

- There are 32 bit and 64 bit versions, are you checking both?

```
C:\Windows\Microsoft.NET\Framework64\v4.0.30319\MSBuild.exe .\Get-Process-CSharp-64.xml
```

https://lolbas-project.github.io/lolbas/Binaries/Msbuild/

MSBuild Testing

✔ How to execute the technique?

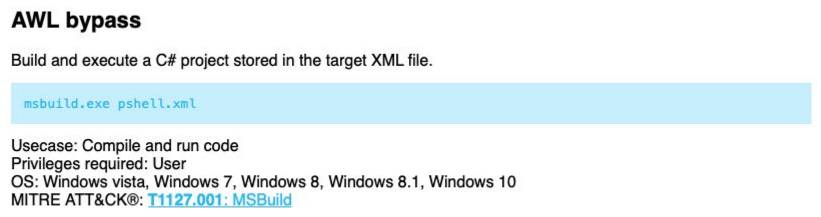
https://lolbas-project.github.io/lolbas/Binaries/Msbuild/

- Provides examples and dependencies
- Need .rsp file, or C sharp, or DLL payloads

Find blog posts/other tools to help out:

- https://www.ired.team/offensive-security/code-execution/using-msbuild-to-execute-shellcode-in-c
- Metasploit

May take some time to figure out what you are doing

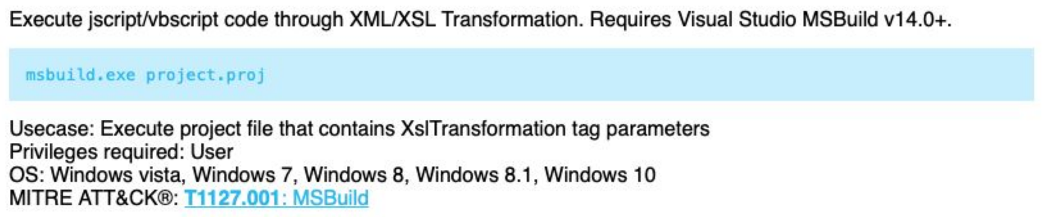


AWL bypass

Build and execute a C# project stored in the target XML file.

```
msbuild.exe pshell.xml
```

Usecase: Compile and run code
Privileges required: User
OS: Windows vista, Windows 7, Windows 8, Windows 8.1, Windows 10
MITRE ATT&CK®: T1127.001: MSBuild



Execute jscript/vbscript code through XML/XSL Transformation. Requires Visual Studio MSBuild v14.0+.

```
msbuild.exe project.proj
```

Usecase: Execute project file that contains XslTransformation tag parameters
Privileges required: User
OS: Windows vista, Windows 7, Windows 8, Windows 8.1, Windows 10
MITRE ATT&CK®: T1127.001: MSBuild

MSBuild (T1127.01) Variation

Can we run something without being logged?

Can we obfuscate what we run?

Can we copy/move?

Can we rename?

Can we bring our own?

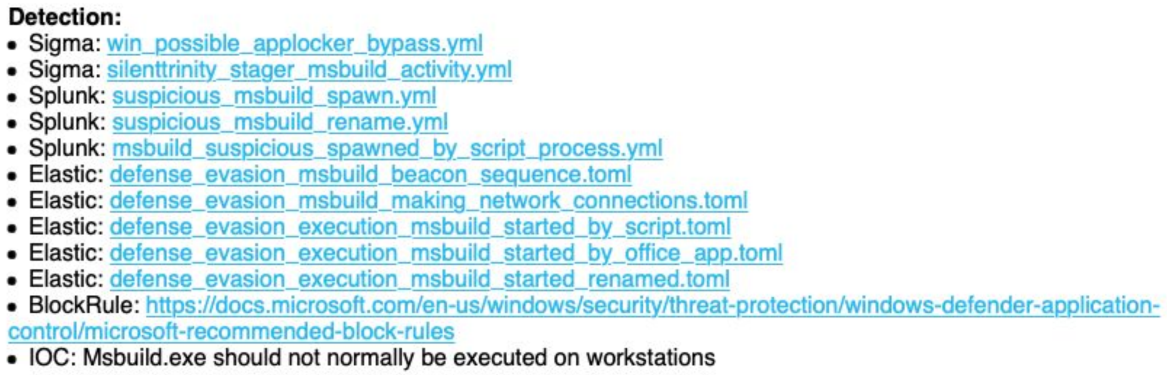
MSBuild (T1127.01) Detections

There is a whole list on the LOLBAS Project Page

However, like a lot of execution methods detections they need to be tuned

LOLBAS Techniques are a prime example of where data correlation is needed

- How many LOLBAS binaries should be creating network connections?



Detection:

- Sigma: win_possible_applocker_bypass.yml
- Sigma: silenttrinity_stager_msbuild_activity.yml
- Splunk: suspicious_msbuild_spawn.yml
- Splunk: suspicious_msbuild_rename.yml
- Splunk: msbuild_suspicious_spawned_by_script_process.yml
- Elastic: defense_evasion_msbuild_beacon_sequence.toml
- Elastic: defense_evasion_msbuild_making_network_connections.toml
- Elastic: defense_evasion_execution_msbuild_started_by_script.toml
- Elastic: defense_evasion_execution_msbuild_started_by_office_app.toml
- Elastic: defense_evasion_execution_msbuild_started_renamed.toml
- BlockRule: https://docs.microsoft.com/en-us/windows/security/threat-protection/windows-defender-application-control/microsoft-recommended-block-rules
- IOC: Msbuild.exe should not normally be executed on workstations

Want more execution methods?

There is always more!

- WMI
- Native API
- ISO
- LNK
- C Sharp
- Nim
- BOFs
- LOLBAS
- Chained execution methods
- And yet to be discovered...

3 Things I hope you took away from this talk

- Why execution methods make security challenging
- Mindset for testing execution methods
- Lots of resources for how to test different types of execution methods

Thank you!

@teschulz