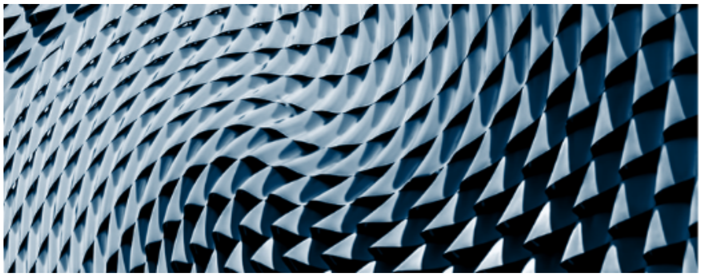
MITRE ATT&CK T1055 Process Injection

picussecurity.com/blog/picus-10-critical-mitre-attck-techniques-t1055-process-injection



Keep up to date with latest blog posts

> In 2019, Picus Labs analyzed 48813 malware to determine tactics, techniques, and procedures (TTPs) used by adversaries in these malicious files. Picus Labs categorized each observed TTP by utilizing the MITRE ATT&CK® framework. As a result of the present research, 445018 TTPs observed in the last year were mapped to ATT&CK to identify the top 10 most common techniques used by attackers.

Our research has found that Process Injection was the most prevalent MITRE ATT&CK technique used by adversaries in their malware. Adversaries emphasize an increased level of stealth, persistence, and privilege in their advanced cyber attacks. As a mechanism that can provide these features, it is not surprising that Process Injection is the most frequently used technique.

The purpose of this blog post is to review:

- the fundamentals of the process injection technique,
- the most used target processes for injection,
- its use cases by threat actors, and
- red, blue, and purple teaming exercises for this technique.

Explore our full research on the 10 Critical MITRE ATT&CK Techniques.

Introduction

It is easy to detect malware processes by listing the running processes and filtering out legitimate ones that are part of the operating system or installed software. If the malware can encapsulate its malicious code within a legitimate process, it will hide on the infected system. Process injection is in fact an “old but gold” technique consisting in running arbitrary code within the address space of another process. As a result, this technique enables access to the target process’s memory, system, and network resources.

On this account, the technique provides three significant benefits for adversaries:

- Executing code under a legitimate process may evade security controls. The legitimate process camouflages the malicious code to evade detection since it is whitelisted.
- Since the malicious code executed inside the legitimate process’s memory space, it may also evade disk forensics.
- If the target process has elevated privileges, this technique will enable privilege escalation. For example, if the target process has access to network resources, the malicious code can communicate legitimately over the Internet and with other computers on the same network.

Processes Targeted by Adversaries for Process Injection

Security controls may quickly detect custom processes. Therefore, threat actors use common Windows processes such as:

- Built-in native Windows processes including explorer.exe, svchost.exe, regsvr32.exe, dllhost.exe, services.exe, cvtres.exe,msbuild.exe, RegAsm.exe, RegSvcs.exe, rundll32.exe, arp.exe, PowerShell.exe, vbc.exe, csc.exe, AppLaunch.exe and cmd.exe
- Processes of common software including iexplore.exe, ieuser.exe, opera.exe, chrome.exe, firefox.exe, outlook.exe, and msinm.exe

Target Process Selection Methods

Adversaries use the following methods when picking their target process for malicious code injection:

- A specific target process is called out in the code. In this case, explorer.exe and svchost.exe are the most commonly used ones.
- A list of target processes is defined in the code. For example, the Turla cyber espionage group’s Carbon backdoor includes a configuration file consisting of a list of target processes for injection[1]. A typical list includes native Windows and browser processes.

- In some attack scenarios, the target process is not previously defined, and a suitable host process is located at runtime in this type of attack. For example, the CopyKittens group used Windows API functions to extract a list of currently active processes and to get a handle to each target process in its campaign[2].

Use Cases by Malware and Threat Actors

Malware	Threat Actor	Target Industries	Target Geographies	Target Process
Backdoor.Oldrea [3]	Dragonfly	Energy	US, Europe	explorer.exe
BlackEnergy [4]	-	Energy, Government	Ukraine	svchost.exe
Cardinal RAT [5]	-	All	All	RegAsm.exe, RegSvcs.exe, vbc.exe, AppLaunch.exe, cvtres.exe
Denis backdoor [6]	APT32	Government, Media	East Asia	rundll32.exe, svchost.exe, arp.exe, PowerShell.exe
Downdelph downloader [7]	APT28	Government	US, Europe	explorer.exe
Dropper (unnamed) [8]	Putter Panda	Government, Telecommunication, Defense, Research, Technology, Aerospace	US, Europe	msinm.exe, outlook.exe, iexplore.exe, firefox.exe
Emotet banking malware [9]	-	All	All	explorer.exe
Kazuar backdoor malware [10]	Turla	Government, Military, Defense	US, Europe, Middle East	explorer.exe
RAT (unnamed) [11]	Emissary Panda	Energy, Government, Technology, Manufacturing	Middle East, Central Asia	svchost.exe
Rokrat RAT [12]	APT37	Government, Finance	Middle East, East Asia	cmd.exe
TClient backdoor malware [13]	Tropic Trooper	Government, Healthcare, Transportation, High-Tech	East Asia	dllhost.exe
Tidyelf dropper malware [14]	APT41	Healthcare, Technology, Telecommunications, Media, Education, Retail	Europe, East Asia, Middle East, US	iexplore.exe
Trickbot banking malware [15]	-	All	All	svchost.exe
Trojan (unnamed) [16]	Gorgon Group	Government	US, Europe	cvtres.exe, MSBuild.exe
Trojan (unnamed)[17]	Kimsuky	Government, Defense, Logistics	South Korea	explorer.exe
ZxShell RAT [18]	Group 72	Manufacturing, Aerospace, Defense, Media	US, East Asia	svchost.exe

Example Process Injection Method: Reflective DLL Injection

Reflective DLL injection (loading) is one of the most used process injection methods employed by adversaries. This method allows injecting and executing a DLL inside another process by creating a DLL that maps itself into memory when executed, instead of relying on Window's API's loader calls. This technique avoids storing the DLL on disk and calling the Windows API's LoadLibrary that might be detected by security tools.

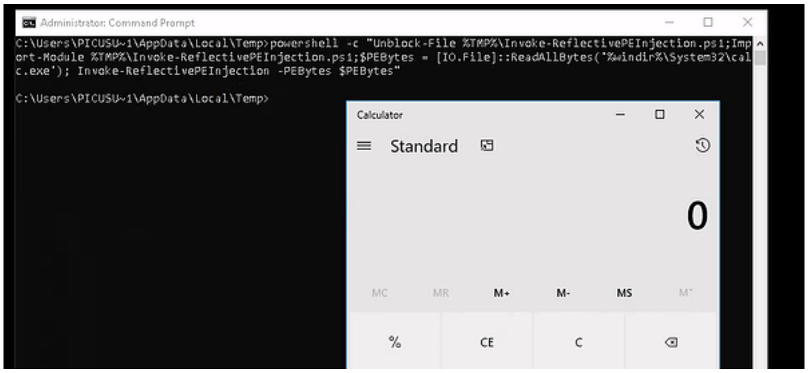
Red Teaming - How to simulate?

Powersploit's Invoke-ReflectivePEInjection [19] module can be used to simulate the reflective DLL injection technique. In addition to loading a DLL or EXE into the PowerShell, It can reflectively load a DLL into a remote process. Because of its capabilities, adversaries are also using this module for injection, such as the Turla APT Group [20].

The below command is a simulation of reflective DLL injection using the Invoke-ReflectivePEInjection module. With this command, contents of the calc.exe file are read into the $PEByte byte array using the ReadAllBytes [21] method. Then the byte array containing the calc.exe is loaded and executed locally using the -PEBytes parameter.

```
powershell -c "Unblock-File %TMP%\Invoke-ReflectivePEInjection.ps1;

Import-Module %TMP%\Invoke-ReflectivePEInjection.ps1;
$PEBytes = [IO.File]::ReadAllBytes('%windir%\System32\calc.exe'); Invoke-ReflectivePEInjection -PEBytes $PEBytes"
```



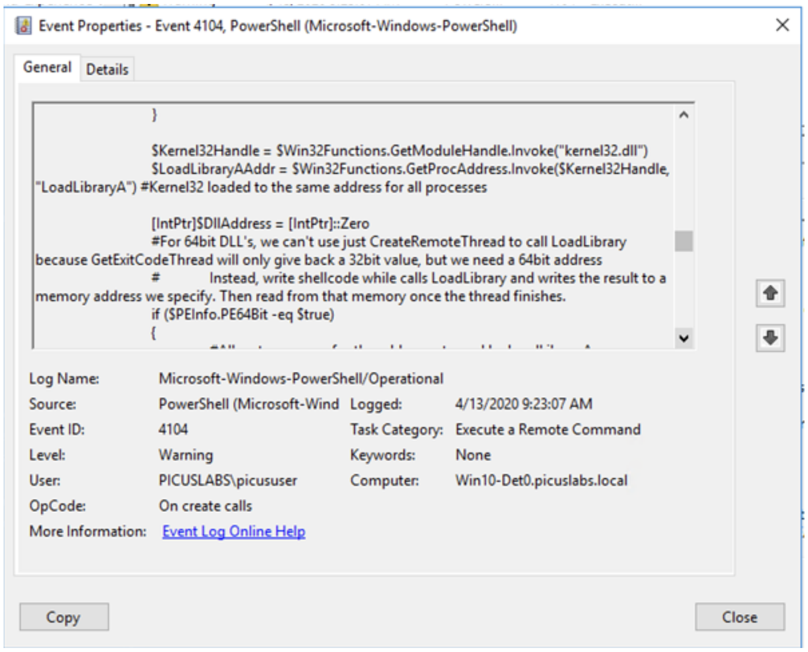
```
C:\Users\PICUSU~1\AppData\Local\Temp>powershell -c "Unblock-File %TMP%\Invoke-ReflectivePEInjection.ps1;Imp
ort-Module %TMP%\Invoke-ReflectivePEInjection.ps1;$PEBytes = [IO.File]::ReadAllBytes('%windir%\System32\cal
c.exe'); Invoke-ReflectivePEInjection -PEBytes $PEBytes"

C:\Users\PICUSU~1\AppData\Local\Temp>
```

Blue Teaming - How to detect?

Sigma Rule

To detect the reflective DLL injection technique, we need logs that include PowerShell activities. Event log entries in the `Microsoft-Windows-PowerShell/Operational` log includes such activities. The `Event ID 4104` (script block logging) records accurate blocks of code as they are executed by the PowerShell engine. Script block logging captures the de-obfuscated full contents of the code as it is executed, including scripts and commands, as shown in the following figure.



Event Properties - Event 4104, PowerShell (Microsoft-Windows-PowerShell)

```
                        }

                        $Kernel32Handle = $Win32Functions.GetModuleHandle.Invoke("kernel32.dll")
                        $LoadLibraryAAddr = $Win32Functions.GetProcAddress.Invoke($Kernel32Handle,
"LoadLibraryA") #Kernel32 loaded to the same address for all processes

                        [IntPtr]$DllAddress = [IntPtr]::Zero
                        #For 64bit DLL's, we can't use just CreateRemoteThread to call LoadLibrary
because GetExitCodeThread will only give back a 32bit value, but we need a 64bit address
                        #        Instead, write shellcode while calls LoadLibrary and writes the result to a
memory address we specify. Then read from that memory once the thread finishes.
                        if ($PEInfo.PE64Bit -eq $true)
                        {
```

```
Log Name:         Microsoft-Windows-PowerShell/Operational
Source:           PowerShell (Microsoft-Wind  Logged:         4/13/2020 9:23:07 AM
Event ID:         4104                        Task Category:  Execute a Remote Command
Level:            Warning                     Keywords:       None
User:             PICUSLABS\picususer         Computer:       Win10-Det0.picuslabs.local
OpCode:           On create calls
More Information: Event Log Online Help
```

When the DLL is injected into the target process, the malware has to map the DLL's raw binary into virtual memory. It uses `kernel32.dll` and `VirtualAlloc` , `GetProcAddress` , and `LoadLibraryA` functions to get the correct address of the injected export function. Picus Labs' Blue team developed the following Sigma rule by taking advantage of this finding mechanism and utilizing the `Microsoft-Windows-PowerShell/Operational` log with the `Event ID 4104` .

```
title: Reflective Portable Executable Injection via PowerShell

status: stable
description: Detects the attempt of reflective portable executable (DLL/EXE) injection by PowerShell that uses API calls. This method is used by adversaries to evade detection from security products since the execution is masked under a legitimate process.
author: Picus Security
references:
    - https://attack.mitre.org/techniques/T1055/
  - https://attack.mitre.org/tactics/TA0004/
    - https://attack.mitre.org/tactics/TA0005/
logsource:
    product: windows
    service: powershell/operational
    definition1: 'Requirements: Group Policy : Computer Configuration\Administrative Templates\Windows Components\Windows PowerShell\Turn On Module Logging'
    definition2: 'Requirements: Group Policy : Computer Configuration\Administrative Templates\Windows Components\Windows PowerShell\Turn On PowerShell Script Block Logging'
detection:
    selection:
        EventID: 4104
    keyword1:
        - '*kernel32.dll*'
    keyword2:
        - '*LoadLibraryA*'
    keyword3:
        - '*GetProcAddress*'
    keyword4:
        - '*VirtualAlloc*'
    condition: All of them
falsepositives:
    - Unlikely, legitimate use in red teaming activities
level: high
tags:
    - attack.defense_evasion
    - attack.privilege_escalation
    - attack.t1055
    - attack.ta0004
    - attack.ta0005
```

Splunk SPL Query

```
(source="WinEventLog:Microsoft-Windows-PowerShell/Operational" EventCode="4104" "*kernel32.dll*" "*LoadLibraryA*"
"*GetProcAddress*" "*VirtualAlloc*")
```

IBM QRadar AQL Query

```
(LOGSOURCETYPENAME(devicetype)='Microsoft Windows Security Event Log' and EventID='4104' and UTF8(payload) ilike
'%kernel32.dll%' and UTF8(payload) ilike '%LoadLibraryA%' and UTF8(payload) ilike '%GetProcAddress%' UTF8(payload)
ilike '%VirtualAlloc%')
```

YARA Rule

The following YARA rule can be used to detect PowerShell scripts used for reflective DLL injection. This rule detects both Powersploit's `Invoke-ReflectivePEInjection` module and Mimikatz's `PE Reflective Injection method` [22].

```
rule power_pe_injection
{
meta:
description = "PowerShell with PE Reflective Injection"
author = "Benjamin DELPY (gentilkiwi)"

strings:
$str_loadlib = "0x53, 0x48, 0x89, 0xe3, 0x48, 0x83, 0xec, 0x20, 0x66, 0x83, 0xe4, 0xc0, 0x48, 0xb9"

condition:
$str_loadlib
}
```

Appendixes

Appendix A - Aliases of Threat Groups

Threat Group	Aliases
APT28	Sednit, Sofacy, Fancy Bear
APT32	OceanLotus
APT37	Group 123, Reaper
Dragonfly	Energetic Bear
Emissary Panda	TG-3390, APT 27, Bronze Union
Group 72	Axiom
Putter Panda	APT2
Tropic Trooper	KeyBoy

Appendix B - Aliases of Malware Families

Malware	Aliases
Backdoor.Oldrea	Havex
ZxShell RAT	Sensocode

References

[1] ESET Research, “Carbon Paper: Peering into Turla’s second stage backdoor | WeLiveSecurity,” *WeLiveSecurity*, 30-Mar-2017. [Online]. Available: https://www.welivesecurity.com/2017/03/30/carbon-paper-peering-turlas-second-stage-backdoor/. [Accessed: 13-Apr-2020].

[2] Minerva Labs LTD, ClearSky Cyber Security, “CopyKittens Attack Group.” [Online]. Available: https://s3-eu-west-1.amazonaws.com/minervaresearchpublic/CopyKittens/CopyKittens.pdf. [Accessed: 13-Apr-2020].

[3] Symantec Security Response, “Dragonfly: Cyberespionage Attacks Against Energy Suppliers.” [Online]. Available: https://www.sbs.ox.ac.uk/cybersecurity-capacity/system/files/Symantec%20-%20Dragonfly_Threat_Against_Western_Energy_Suppliers.pdf. [Accessed: 13-Apr-2020].

[4] F-Secure, “BlackEnergy & Quedagh The convergence of crimeware and APT attacks.” [Online]. Available: https://www.f-secure.com/documents/996508/1030745/blackenergy_whitepaper.pdf. [Accessed: 13-Apr-2020].

[5] J. Grunzweig, “Cardinal RAT Active for Over Two Years,” *Unit42*, 20-Apr-2017. [Online]. Available: https://unit42.paloaltonetworks.com/unit42-cardinal-rat-active-two-years/. [Accessed: 13-Apr-2020].

[6] Cybereason, “Operation Cobalt Kitty.” [Online]. Available: https://cdn2.hubspot.net/hubfs/3354902/Cybereason%20Labs%20Analysis%20Operation%20Cobalt%20Kitty.pdf. [Accessed: 13-Apr-2020].

[7] “En Route with Sednit Part 3: A Mysterious Downloader,” *Eset*. [Online]. Available: https://www.welivesecurity.com/wp-content/uploads/2016/10/eset-sednit-part3.pdf. [Accessed: 13-Apr-2020].

[8] CrowdStrike, “CrowdStrike Intelligence Report PUTTER PANDA.” [Online]. Available: https://cdn0.vox-cdn.com/assets/4589853/crowdstrike-intelligence-report-putter-panda.original.pdf. [Accessed: 13-Apr-2020].

[9] “Emotet Malware | CISA.” [Online]. Available: https://www.us-cert.gov/ncas/alerts/TA18-201A. [Accessed: 13-Apr-2020].

[10] B. Levene, R. Falcone, and T. Halfpop, “Kazuar: Multiplatform Espionage Backdoor with API Access,” *Unit42*, 03-May-2017. [Online]. Available: https://unit42.paloaltonetworks.com/unit42-kazuar-multiplatform-espionage-backdoor-api-access/. [Accessed: 13-Apr-2020].

[11] nccgroup, “Emissary Panda – A potential new malicious tool.” [Online]. Available: https://www.nccgroup.trust/uk/about-us/newsroom-and-events/blogs/2018/may/emissary-panda-a-potential-new-malicious-tool/. [Accessed: 13-Apr-2020].

[12] “Threat Analysis: ROKR
Carbon Black,” *VMware Car*
[Online]. Available:
https://www.carbonblack.com
analysis-rokrat-malware/. [Ac

[13] T. Micro, “Tropic Trooper
TrendLabs Security Intelligen
[Online]. Available:
https://blog.trendmicro.com/t
intelligence/tropic-trooper-ne
13-Apr-2020].

[14] FireEye, “Double Drago
espionage and cyber crime o
Available:
https://www.fireeye.com/cont
apt41-2019.pdf. [Accessed: 1

[15] S2 Grupo, “Evolution of
Available: https://www.securit
content/uploads/2017/07/Tric
Grupo.pdf. [Accessed: 13-Ap

[16] R. Falcone, D. Fuertes,
Wilhoit, “The Gorgon Group:
Nation State and Cybercrime
[Online]. Available:
https://unit42.paloaltonetwor
group-slithering-nation-state-
13-Apr-2020].

[17] D. Tarakanov, “The ‘Kim
Korean APT?” [Online]. Avail
https://securelist.com/the-kim
korean-apt/57915/. [Accesse

[18] Talos Group, “Threat Sp
Opening the ZxShell - Cisco
Oct-2014. [Online]. Available
https://blogs.cisco.com/secur
[Accessed: 13-Apr-2020].

[19] PowerShellMafia,
“PowerShellMafia/PowerSpl
Available:
https://github.com/PowerShe
[Accessed: 13-Apr-2020].

[20] ESET, “A dive into Turla
[Online]. Available:
https://www.welivesecurity.co
powershell-usage/. [Accesse

[21] dotnet-bot, “File.ReadAl
(System.IO).” [Online]. Availa
https://docs.microsoft.com/e
us/dotnet/api/system.io.file.re
13-Apr-2020].

[22] gentilkiwi, “gentilkiwi/mi
Available: https://github.com/
[Accessed: 13-Apr-2020].