EDR 與 Windows .NET 惡意程式的技術戰爭

CYCRAFT
奧義智慧科技

講者簡介

- 現任奧義智慧資安軟體工程師
- 畢業於國立陽明交通大學
- 研究專長
  - Endpoint Security
  - AD Security
- 曾發表在全國資訊安全會議
  - CISC 2020
  - CISC 2021

AGENDA

- .NET Framework
- Telemetry
  - ETW
- Analysis
- 案例分析：TA410
- 總結

案例回顧

2020

Analysis: Abuse of .NET features for compiling malicious programs

March 12, 2020 Cyber Security Review

The .NET framework, a software development framework created by Microsoft and is now a built-in component of Windows, includes components that enable developers to compile and execute C# source code during runtime. This allows programs to update or load modules without having to restart.

While the .NET framework is originally intended to help software engineers, cybercriminals have found a way to abuse its features to compile and execute malware on the fly. Recently, we discovered several kinds of malware, such as LokiBot (detected by Trend micro as Trojan.Win32.LOKI), utilizing this technique. This particular LokiBot variant disguises itself as a fake game launcher to trick users into downloading the malware into their machines and drops a compiled C# code into the system.

Microsoft Exchange servers increasingly hacked with IIS backdoors

By Sergiu Gatlan July 26, 2022 02:01 PM 0

Microsoft says attackers increasingly use malicious Internet Information Services (IIS) web server extensions to backdoor unpatched Exchange servers as they have lower detection rates compared to web shells.

Because they're hidden deep inside the compromised servers and often very hard to detect being installed in the exact location and using the same structure as legitimate modules, they provide attackers' with a perfect and durable persistence mechanism.

案例回顧

'IceApple' Post-Exploitation Framework Created for Long-Running Operations

By Ionut Arghire on May 13, 2022

Share Tweet 推薦 10 RSS

CrowdStrike has detailed a new post-exploitation framework that could be the work of a state-sponsored threat actor, one likely linked to China.

Dubbed IceApple and targeting Microsoft Exchange servers, the framework is an in-memory-only tool designed to evade detection and provide long-term access to the compromised environments. The framework can also run on Internet Information Services (IIS) web server software.

CrowdStrike's researchers have been tracking IceApple since late 2021, with the observed attacks spanning across the technology, academic and government sectors in multiple geographies. The observed activity, they say, aligns with China's information gathering interests.

IceApple, the researchers note, is a highly sophisticated IIS post-exploitation framework focused on increasing an adversary's visibility of the target environment, without offering exploitation or lateral movement capabilities.

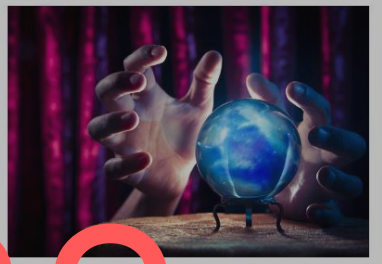
[READ: Super-Stealthy 'Daxin' Backdoor Linked to Chinese Threat Actor]

To date, the researchers have identified 18 different IceApple modules that offer various types of functionality, and also observed that the framework is under active development and that the modules are constantly updated.

2022

Microsoft Describes 'MagicWeb' Attacks Using Active Directory Federation Services

By Kurt Mackie | 08/25/2022



Microsoft on Wednesday described "MagicWeb" attacks by an advanced persistent threat group called "Nobelium," advising organizations using Active Directory Federation Services (ADFS) to take hardening steps.

Nobelium is Microsoft's name for attackers thought to be associated with Russia. About a year ago, Microsoft had called this group "Solorigate," with the name arising from a supply-chain compromise of SolarWinds' Orion software. That compromise led to widespread espionage tampering on Exchange Online e-mail around the globe. At that time, ADFS was one of the technologies getting targeted by this attack group to gain access to Exchange Online e-mails.

ADFS is a Windows Server role used by organizations for connecting with apps and services using single sign-on access. It enables federation trusts, where the identity aspects get managed locally in organizations, per Microsoft's documentation description.

The MagicWeb attack approach is a newly discovered attack method that leverages ADFS, but it isn't associated with a supply-chain compromise of software. Instead, MagicWeb is a "post-compromise capability" that's just available to attackers after they've obtained "highly privileged" credentialed access, explained the announcement by the Microsoft Threat Intelligence Center, the Microsoft Detection and Response Team, and the Microsoft 365 Defender Research Team.

2022

為何駭客選擇 .NET Malware?

.NET Framework 吸引駭客的特色

適用情境廣

- 大部分 Windows OS 內建 [1]
  - Win 8 後不能被 uninstalled

功能豐富

- 駭客可靠系統原生工具達成目標
  - LotL (Living-off-the-land)

In-memory Execution

- 檔案不落地， 繞過靜態分析
- .NET framework 4.8 加入 AMSI [2]
  - 在那之前，可透過 Assembly.Load() 輕易繞過 AMSI

[1] https://docs.microsoft.com/en-us/dotnet/framework/migration-guide/versions-and-dependencies
[2] https://web.archive.org/web/20210402102452/https://docs.microsoft.com/en-us/dotnet/framework/whats-new/#common-language-runtime

.NET Malware

- 利用 .NET framework features，減少攻擊留下的足跡
- 攻擊者經常透過 heavily obfuscated 繞過靜態偵測
  - 需要動態偵測輔助觀察
- 除了感染 host 以外，也會攻擊 Windows 環境中常見的服務 [1]
  - AD
  - ADFS
  - ADCS
  - IIS

[1] https://github.com/Flangvik/SharpCollection

Telemetry

死亡筆記本主角如何規避攝影機?
將筆記藏在攝影機看不到的地方

如何避免攻擊者找到死角？

Telemetry

盤點資料來源

> 攝影機的布置狀況

整理攻擊手法

> 攻擊者尋找死角

檢視觀測狀況

> 根據狀況增設鏡頭

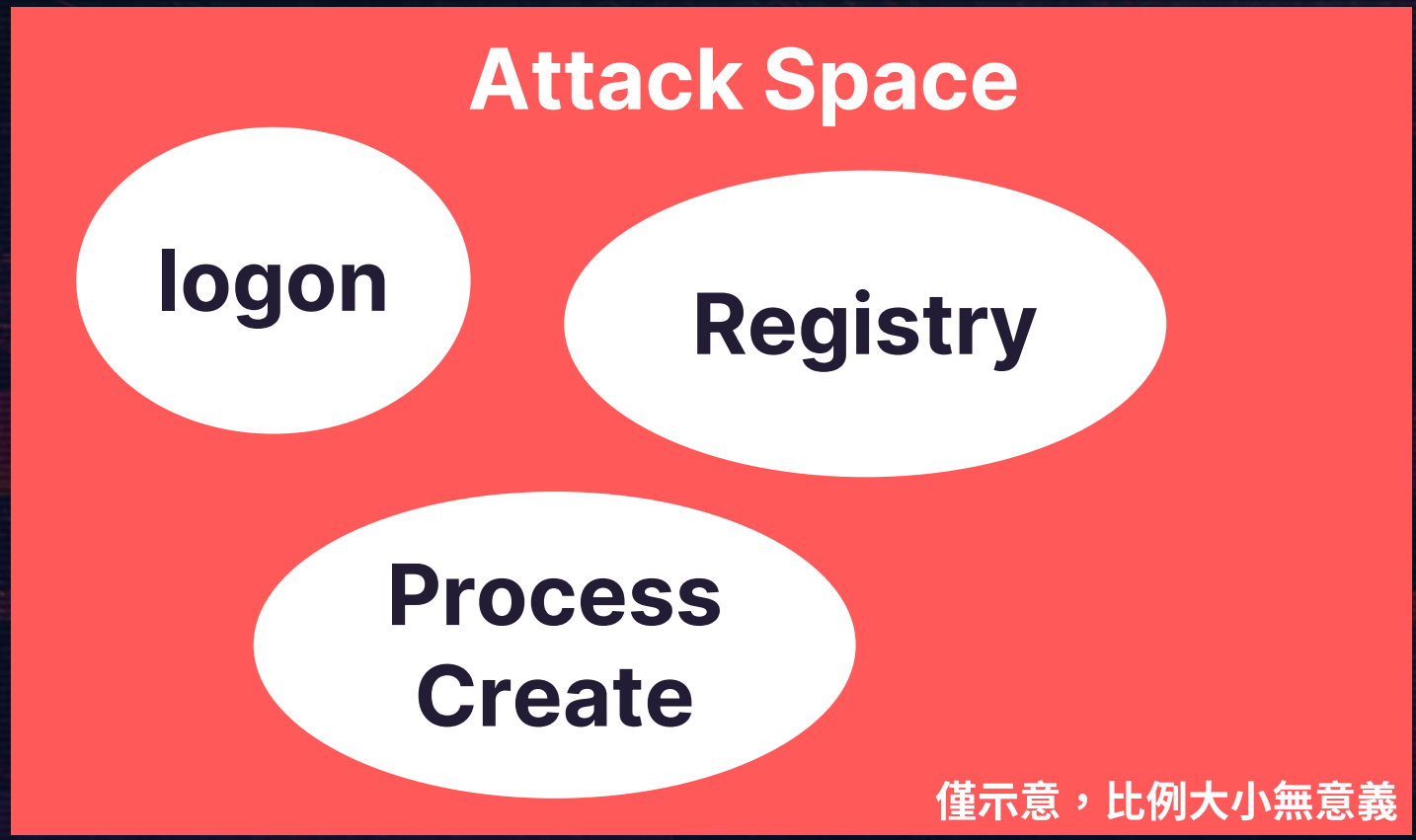
Telemetry

盤點資料來源 → 整理攻擊手法 → 檢視觀測狀況

- 攝影機的布置狀況
- 攻擊者尋找死角
- 根據狀況增設鏡頭

盤點資料來源

> 檢視 data source 可觀察的攻擊手法



檢視 Data Source 可觀察到的攻擊手法 [1]

[1] https://mitre-attack.github.io/attack-navigator/

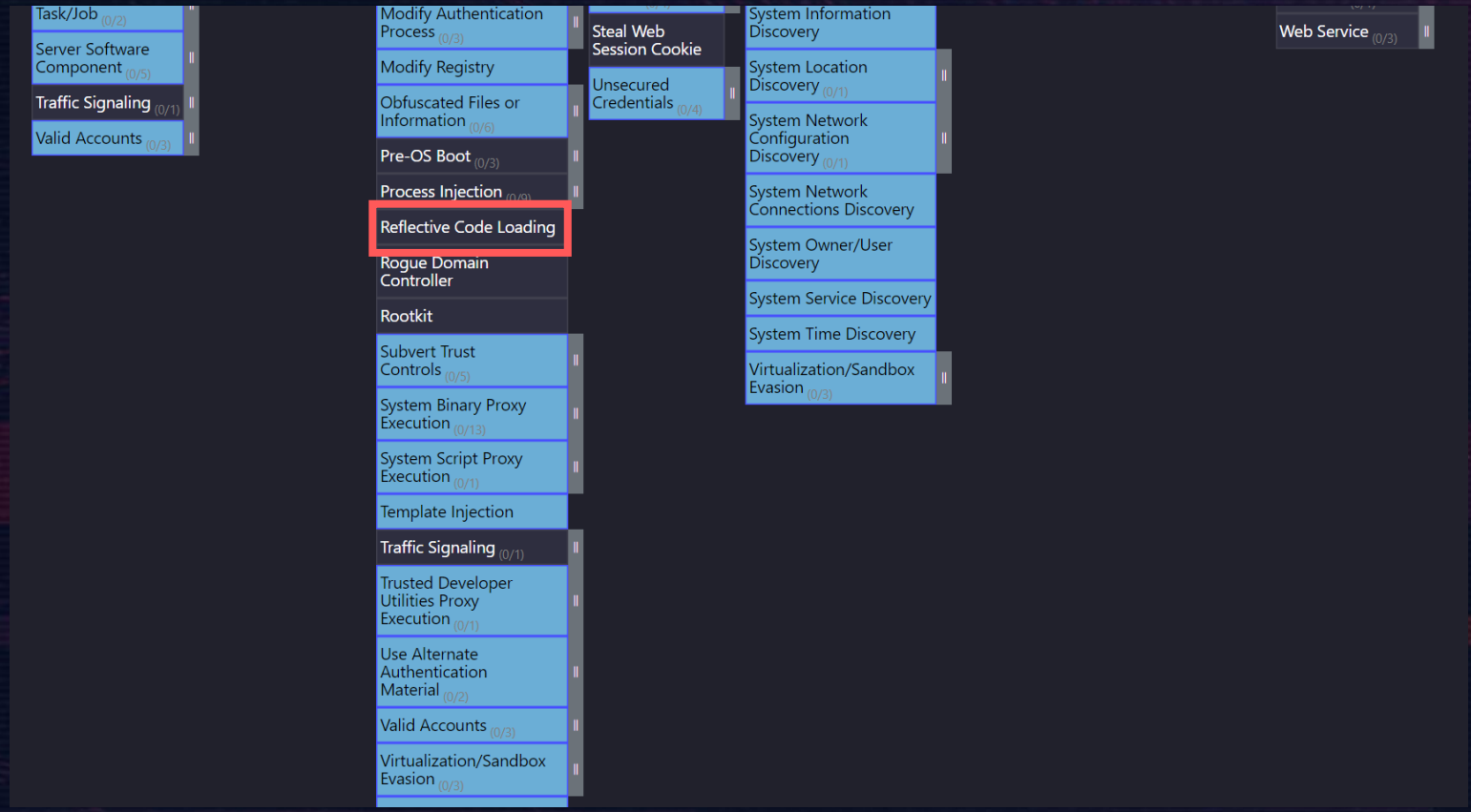
Telemetry

盤點資料來源 → 整理攻擊手法 → 檢視觀測狀況

- 攝影機的布置狀況
- 攻擊者尋找死角
- 根據狀況增設鏡頭

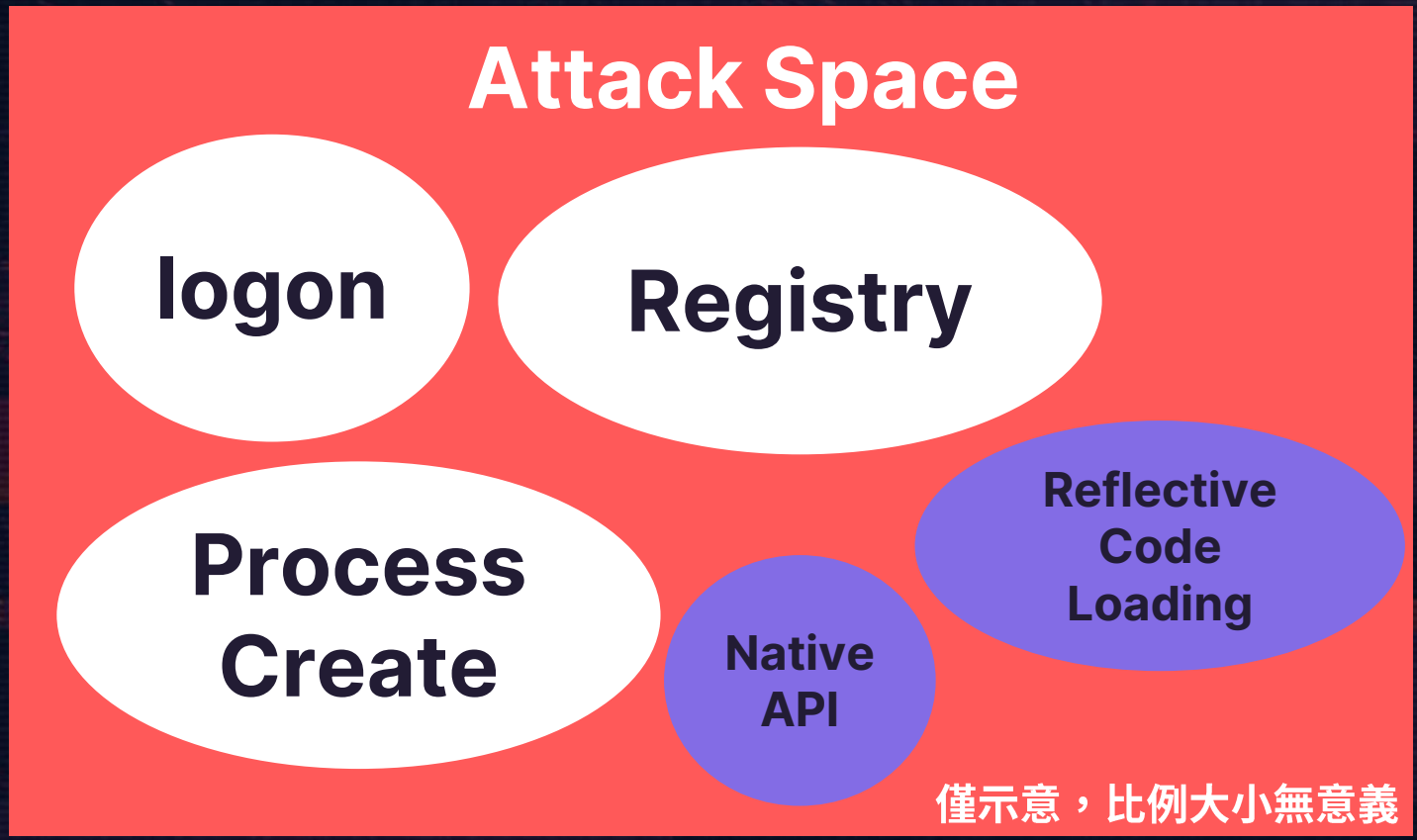
現階段的死角

> 盤點灰色區塊是否有常見的威脅、或者最近攻擊趨勢



.NET Malware 常見攻擊手法

> 列出 .NET 攻擊手法，並檢視現有的 data source 是否能觀測



整理攻擊手法

- 盤點灰色區塊是否有常見的威脅、或者最近攻擊趨勢

Reflective Code Loading

Adversaries may reflectively load code into a process in order to conceal the execution of malicious payloads. Reflective loading involves allocating then executing payloads directly within the memory of the process, vice creating a thread or process backed by a file path on disk. Reflectively loaded payloads may be compiled binaries, anonymous files (only present in RAM), or just snubs of fileless executable code (ex: position-independent shellcode).[1][2][3][4][5]

Reflective code injection is very similar to Process Injection except that the "injection" loads code into the processes' own memory instead of that of a separate process. Reflective loading may evade process-based detections since the execution of the arbitrary code may be masked within a legitimate or otherwise benign process. Reflectively loading payloads directly into memory may also avoid creating files or other artifacts on disk, while also enabling malware to keep these payloads encrypted (or otherwise obfuscated) until execution.[3][4][6][7]

Detection

ID	Data Source	Data Component	Detects
DS0011	Module	Module Load	Monitor for artifacts of abnormal process execution. For example, a common signature related to reflective code loading on Windows is mechanisms related to the .NET Common Language Runtime (CLR) -- such as mscor.dll, mscoree.dll, and clr.dll -- loading into abnormal processes (such as notepad.exe)
DS0009	Process	OS API Execution	Monitor for code artifacts associated with reflectively loading code, such as the abuse of .NET functions such as `Assembly.Load()` and Native API functions such as `CreateThread()`, `memfd_create()`, `execve()`, and/or `execveat()`.[4][7]
DS0012	Script	Script Execution	Similarly, AMSI / ETW traces can be used to identify signs of arbitrary code execution from within the memory of potentially compromised processes.[19][1]

整理攻擊手法

> 根據攻擊手法描述，找到對應的資料來源

Reflective Code Loading

Adversaries may reflectively load code into a process in order to conceal the execution of malicious payloads. Reflective loading involves allocating then executing payloads directly within the memory of the process, vice creating a thread or process backed by a file path on disk. Reflectively loaded payloads may be compiled binaries, anonymous files (only present in RAM), or just snubs of fileless executable code (ex: position-independent shellcode).[1][2][3][4][5]

Reflective code injection is very similar to Process Injection except that the "injection" loads code into the processes' own memory instead of that of a separate process. Reflective loading may evade process-based detections since the execution of the arbitrary code may be masked within a legitimate or otherwise benign process. Reflectively loading payloads directly into memory may also avoid creating files or other artifacts on disk, while also enabling malware to keep these payloads encrypted (or otherwise obfuscated) until execution.[3]

偵測手法

- Module Load
- OS API Execution
- Script Execution && ETW

檢視 Data Source 可觀察到的攻擊手法 [1]

[1] https://mitre-attack.github.io/attack-navigator/

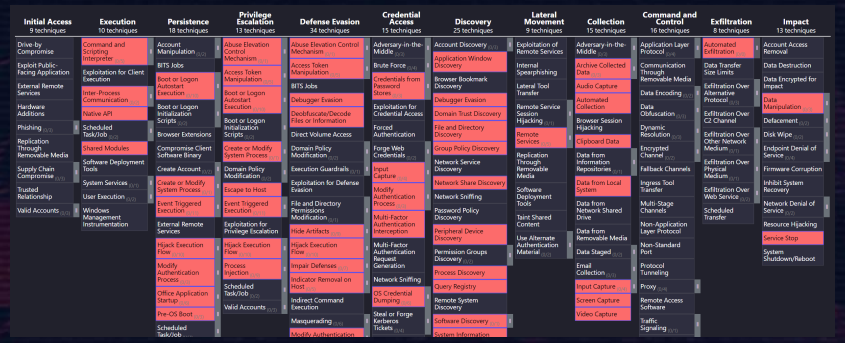
Telemetry

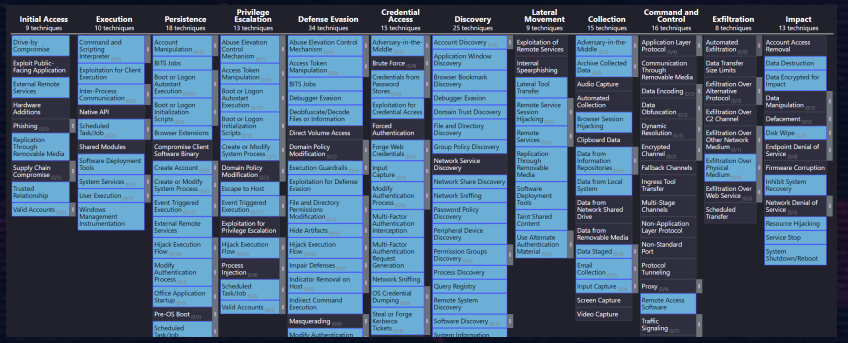
盤點資料來源 → 整理攻擊手法 → 檢視觀測狀況

- 攝影機的布置狀況
- 攻擊者尋找死角
- 根據狀況增設鏡頭

檢測觀測狀況

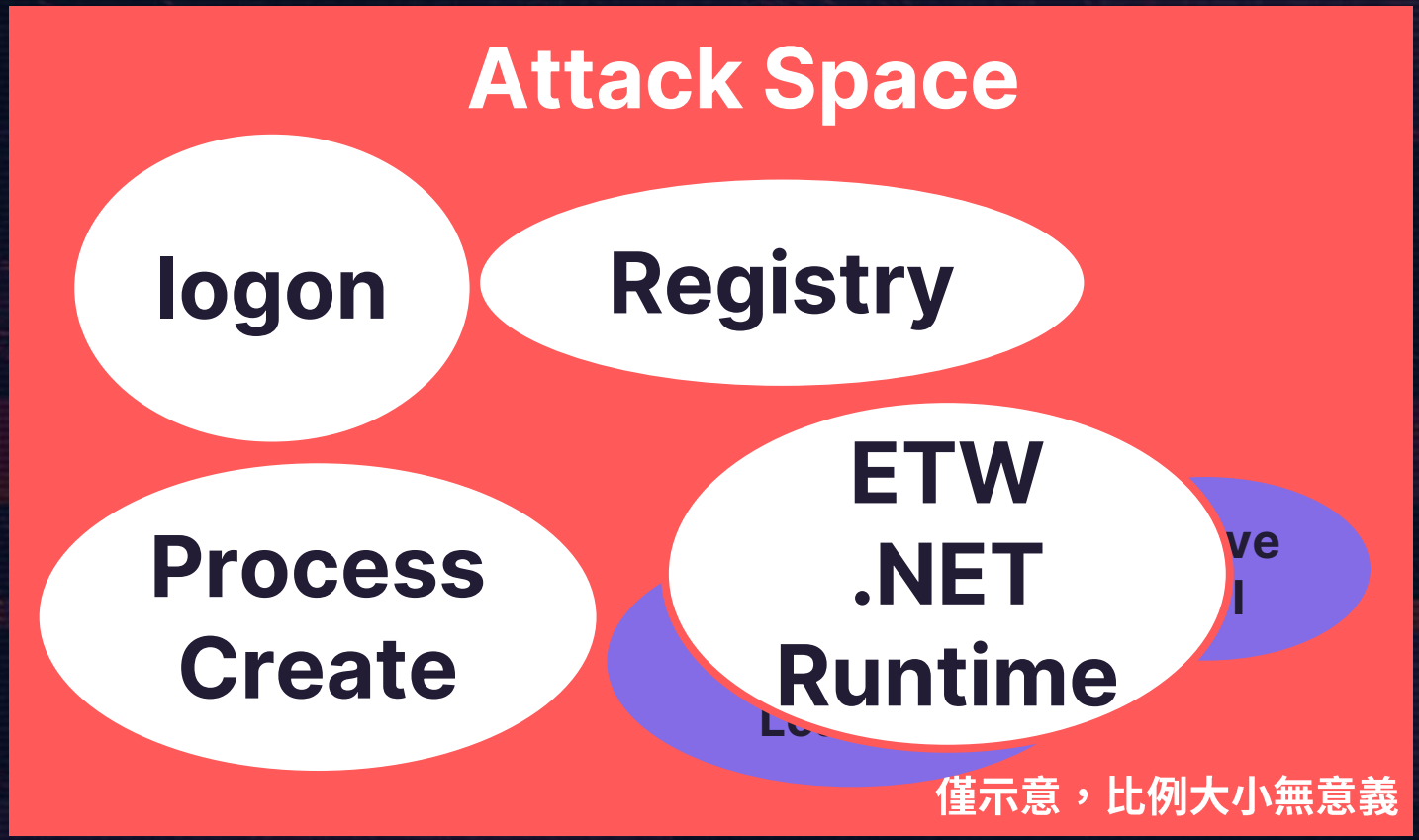
- 根據死角做延伸，如果追加鏡頭，可以觀察到那些相似類型的攻擊手法





Telemetry 檢視

- 新增 data source 擴大對 attack space 的涵蓋



還有什麼方法躲避攝影機追蹤?

關掉 Camara

繞過 Telemetry

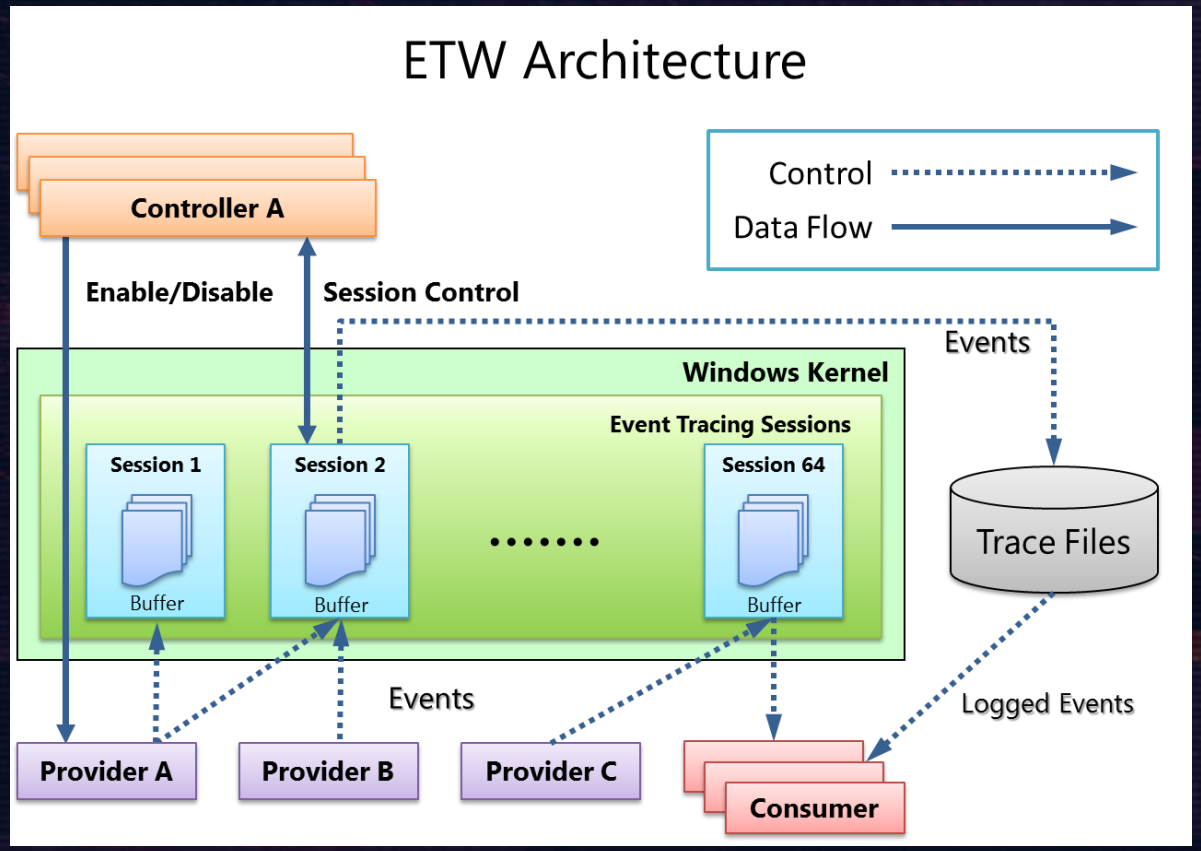
- Windows 常見的 telemetry 都有方法可以繞過
  - Windows event log
  - Sysmon
  - ETW
- 攻擊者可關閉 event 觀測，避免特定高風險動作被發現
  - 犯案時間點前後，攝影機總是神奇地黑畫面
- 過度仰賴單一 data source 容易被攻擊者繞過
  - 每多一個 data source，攻擊者就需要付出更多成本

ETW

Event Tracing for Windows

ETW 簡介 [1]

- ETW 在 Windows 2000 引入，是 Windows 中一種高效追蹤系統事件的方式



[1] https://docs.microsoft.com/zh-tw/windows-hardware/test/weg/instrumenting-your-code-with-etw

ETW 簡介

- ETW 資訊非常詳細且可動態啟用
  - 成為 EDR 重要的 log 來源
- ETW event 在各個 OS 版本有些微差異 [1]
  - 可透過指令查詢當前電腦的事件 provider

Microsoft-Windows-DotNETRuntime
Microsoft-Windows-WinRM
Microsoft-Windows-WMI
Microsoft-Windows-OfflineFiles
Microsoft-Windows-Kernel-Registry

[1] https://github.com/jdu2600/Windows10EtwEvents/tree/master/manifest

ETW 簡介

- ETW 資訊非常詳細且可動態啟用
  - 成為 EDR 重要的 log 來源
- ETW event 在各個 OS 版本有些微差異 [1]
  - 可透過指令查詢當前電腦的事件 provider

Microsoft-Windows-DotNETRuntime
Microsoft-Windows-WinRM
Microsoft-Windows-WMI
Microsoft-Windows-OfflineFiles
Microsoft-Windows-Kernel-Registry

[1] https://github.com/jdu2600/Windows10EtwEvents/tree/master/manifest

ETW .NET Runtime Event

- 觀察 process load 哪些 assembly
  - 正常 process 是否 load 可疑的 assembly
- 觀察 function call 有哪些行為

Load Assembly

The public key token is a unique 16-character key that is given to the assembly when it is built and signed in Microsoft Visual Studio.

- 部分服務會載入特定 assembly [1]
- 觀察是否有異常的 public key token [2]
  - 竄改過的 malicious DLL 會與原本的值不同

Name	Value
	3tm7nr3i63qarneu
Microsoft.ActiveDirectory.Management	3tm7nr3j8kfiws6i4
	NULL

[1] https://docs.microsoft.com/en-us/dotnet/api/
[2] https://docs.microsoft.com/en-us/dynamics-nav/how-to--determine-the-public-key-token-of-the-windows-client-control-add-in-and-.net-framework-assembly

Function Call

- Process 使用的 API 是否對應到某一種攻擊手法
- Process 做哪些 behavior
  - File R/W
  - Registry
  - WMI
  - Network connection
- Process function 是否有 obfuscated

API Map to Technique

- 觀察 process 使用的 API 是否對應到某一種攻擊手法

Input Capture: Keylogging

Other sub-techniques of Input Capture (4)

ID: T1056.001

Sub-technique of: T1056

ⓘ Tactics: Collection, Credential Access

ⓘ Platforms: Linux, Network, Windows, macOS

ⓘ Permissions Required: Administrator,

Adversaries may log user keystrokes to intercept credentials as the user types them. Keylogging is likely to be used to acquire credentials for new access opportunities when OS Credential Dumping efforts are not effective, and may require an adversary to intercept keystrokes on a system for a substantial period of time before credentials can be successfully captured.

Detection

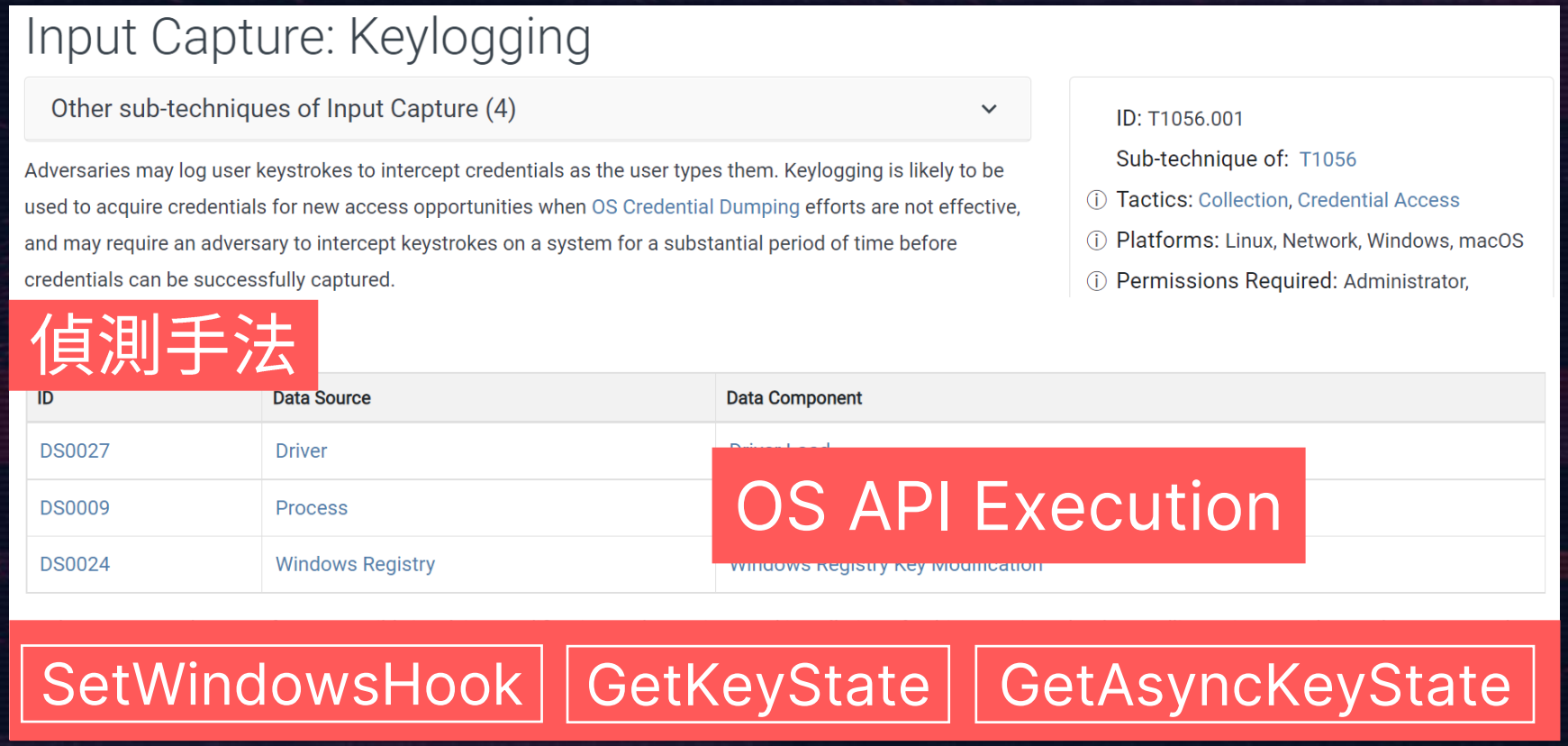
ID	Data Source	Data Component
DS0027	Driver	Driver Load
DS0009	Process	OS API Execution
DS0024	Windows Registry	Windows Registry Key Modification

Keyloggers may take many forms, possibly involving modification to the Registry and installation of a driver, setting a hook, or polling to intercept keystrokes. Commonly used API calls include `SetWindowsHook`, `GetKeyState`, and `GetAsyncKeyState`.[1] Monitor the Registry and file system for such changes, monitor driver installs, and look for common keylogging API calls. API calls alone are not an indicator of keylogging, but may provide behavioral data that is useful when combined with other information such as new files written to disk and unusual processes.

[1] MITRE, “Keylogging”, https://attack.mitre.org/techniques/T1056/001/

API Map to Technique

- 觀察 process 使用的 API 是否對應到某一種攻擊手法



[1] MITRE, “Keylogging”, https://attack.mitre.org/techniques/T1056/001/

小總結

- 觀察是否載入異常的 assembly
- 觀察 process 執行的 function
  - 是否可以對應到攻擊手法
  - 是否可以對應到行為
  - 是否有 obfuscated

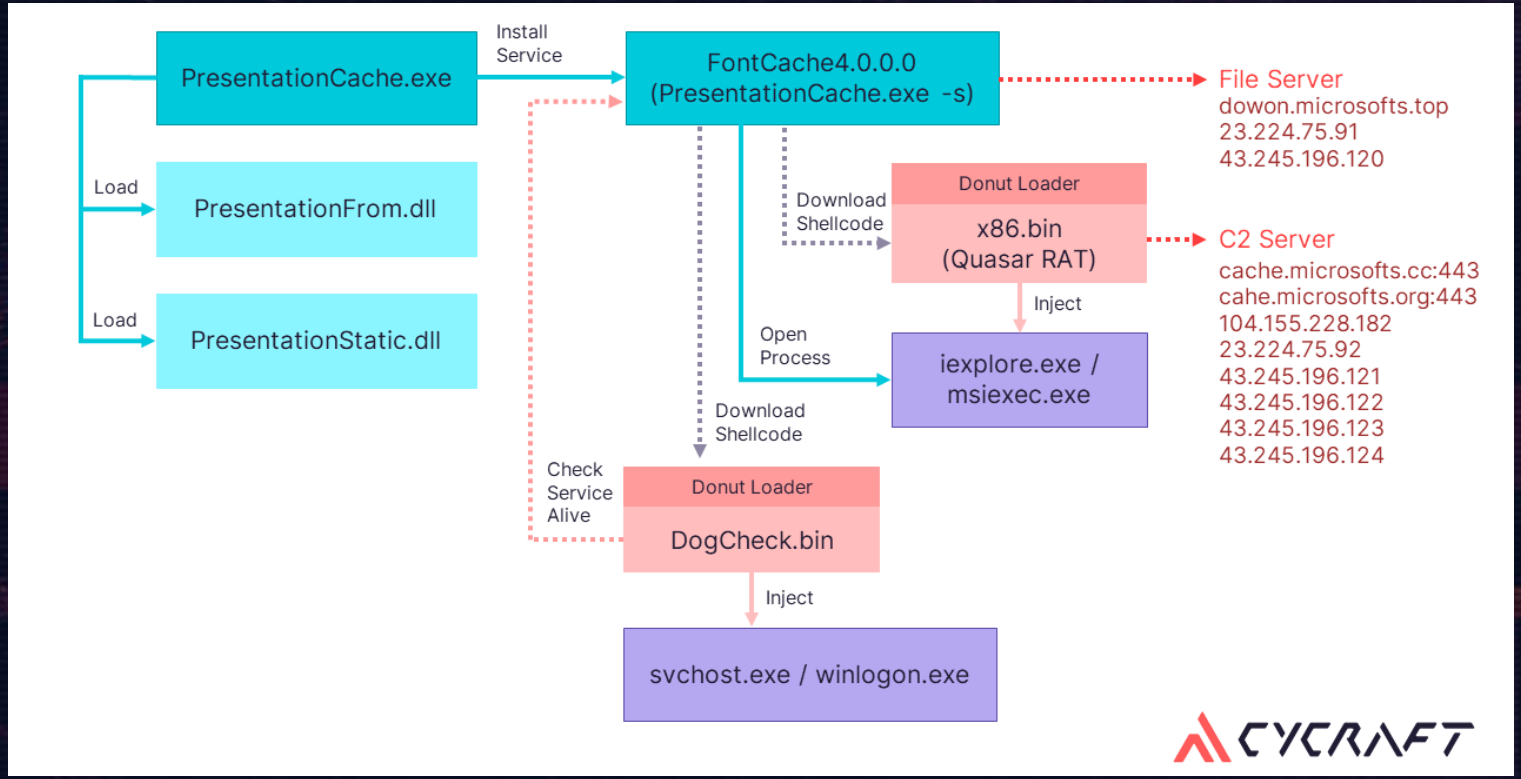
Case Study: TA410

CYCRAFT
奧義智慧科技

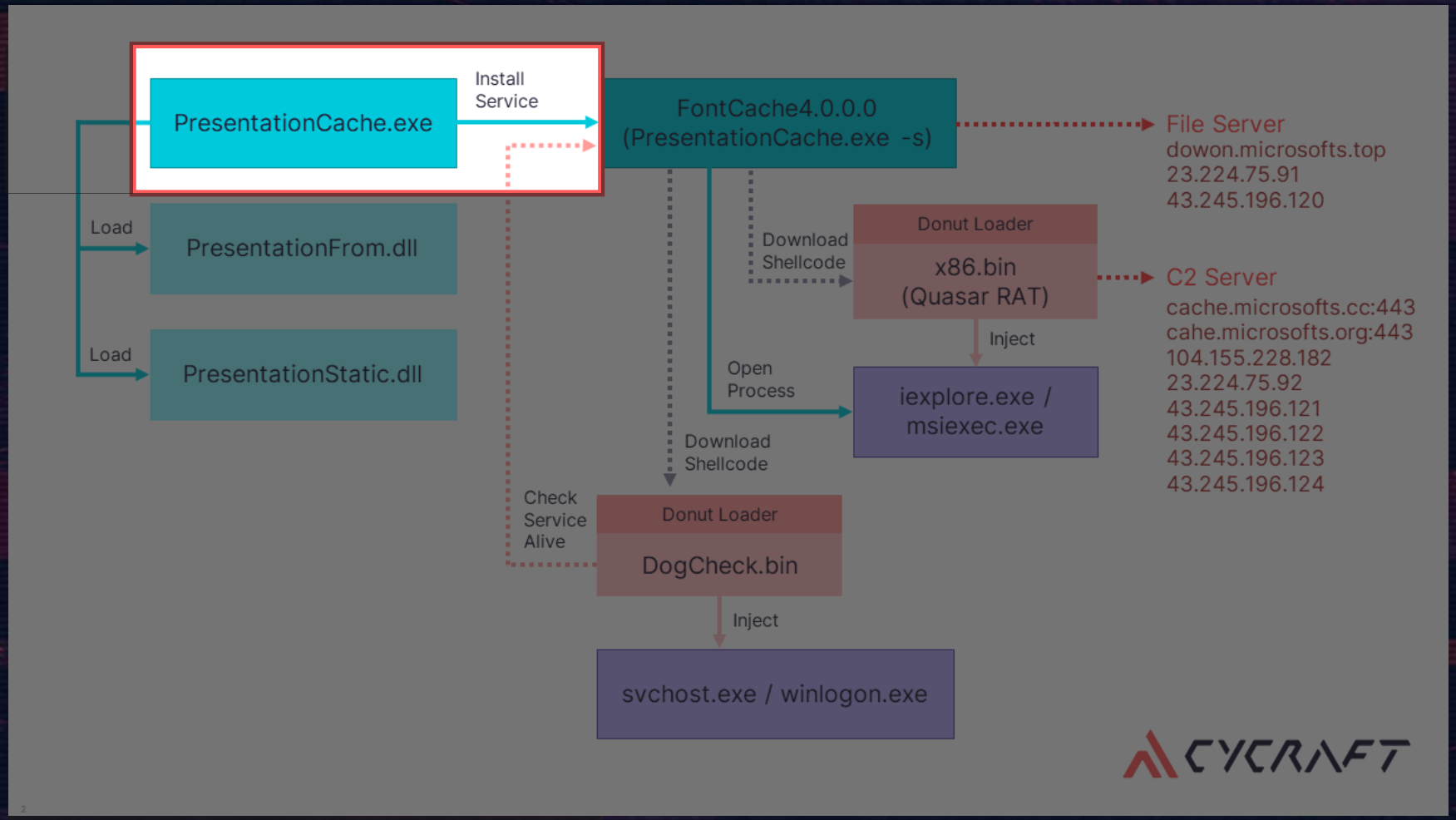
深度剖析針對臺灣金融業的 Operation Cache Panda 組織型供應鏈攻擊

TA410

- 有發現大量的 obfuscated functions



TA410



Presentation Cache

> 取名與 .NET API 相似

Load Assembly

- PresentationCache, PublicKeyToken=null
- System.Activities.Presentation

Function Call

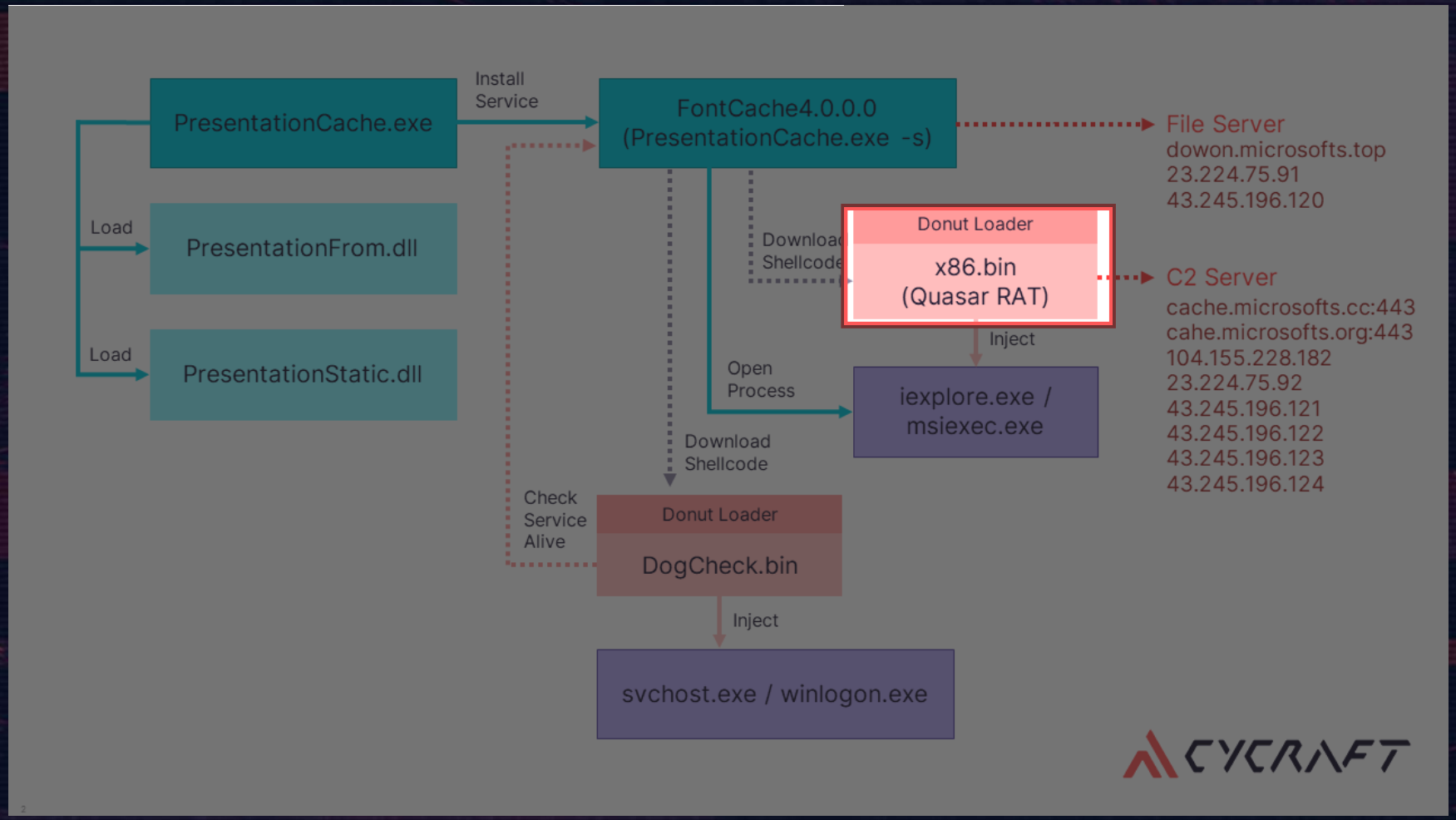
- OpenSCManager
  OpenService
- ServiceManager

Presentation Cache

> 有 anti sandbox 機制

功能	名稱
Sleep	check_sleep_acceleration
檢查 debugger	CheckRemoteDebuggerPresent

TA410

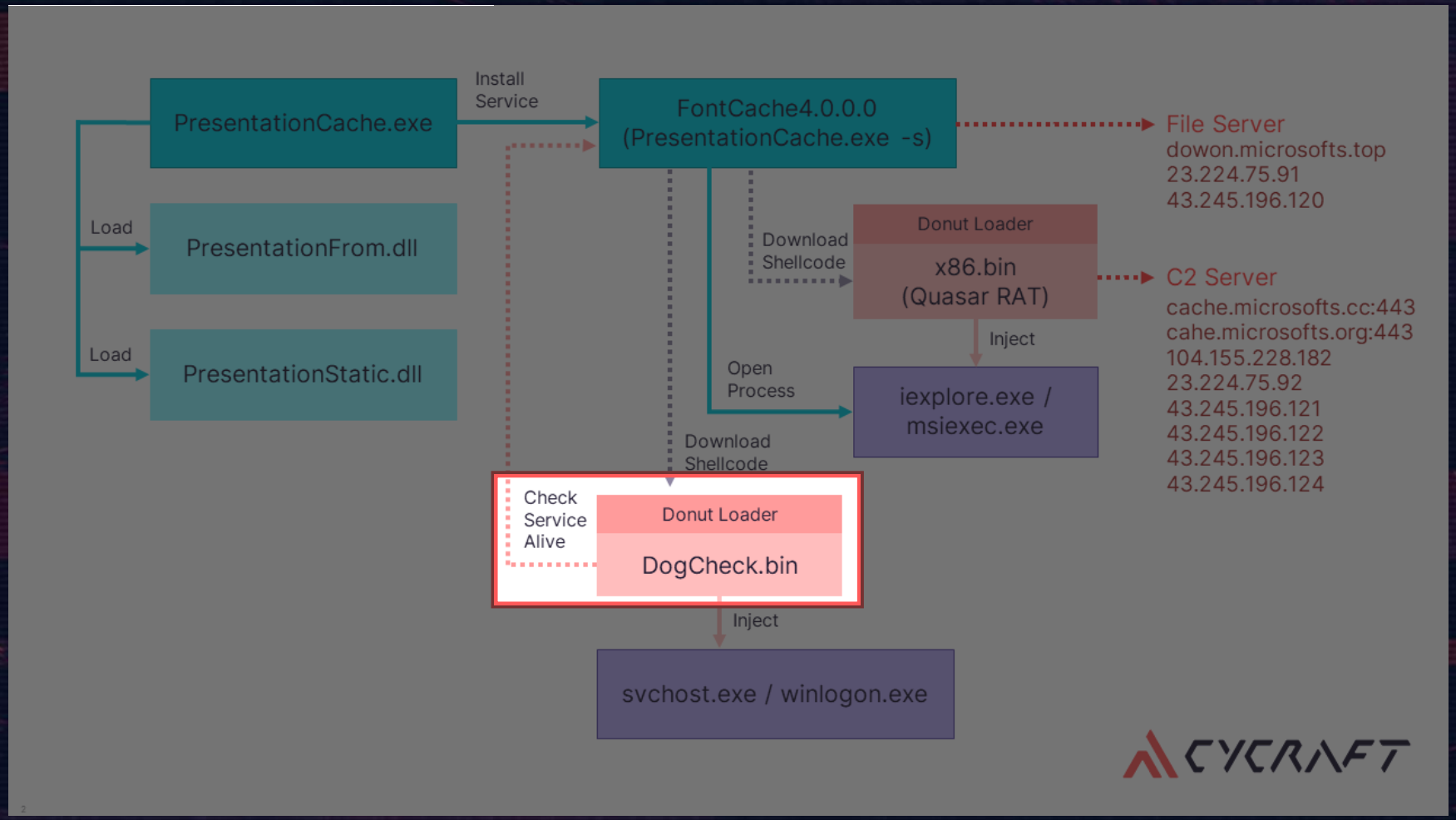


x86

- Massive ProtoBuf related function calls
  - 在 Quasar RAT source code 中也可以發現 using Protobuf;

Load Assembly	Defender, PublicKeyToken=null LoadPlug, PublicKeyToken=null
Function Call	ExecQueryWmi

TA410



DogCheck

> 從 function 大概推測是檢查 service 連線狀況

Load Assembly	DoCheck, PublicKeyToken=null
Function Call	GetExtendedTcpTable ISWindowsServiceInstalled timer1_Tick get_RemotePort GetTable

將 Function 對應到 ATT&CK

Binary	ATT&CK ID	Techniques
Presentation Cache	T1622	Debugger Evasion
	T1497.003	Virtualization/Sandbox Evasion
	T1543	Create or Modify System Process
	T1027	Obfuscated Files or Information
	T1569.002	Service Execution
x86	T1047	Windows Management Instrumentation
DogCheck	T1082	System Information Discovery

接下來，行動！(管理)

- 短期
  - 了解場域中資安產品的 data source
- 中期
  - 盤點場域中的 data source，並檢視在 attack space 中的 coverage
- 長期
  - 檢視 EDR 對 .NET malware 抵禦能力

接下來，行動！(鑑識)

- 短期
  - 玩玩看 ETW
- 中期
  - 了解常見的 .NET 攻擊工具
- 長期
  - 透過工具重現攻擊手法並觀察 log