Memory Wars:
對記憶體攻擊手法與防禦技術的探討

Leo Liaw
數位資安系統股份有限公司

iSecurity
The Hub of Security Innovations

Agenda

- 記憶體漏洞
- How Windows execute an executable?
- What is PE?
- 微軟防護記憶體攻擊的工具 (EMET)
  - EMET 版本沿革
  - EMET 的限制
- EMET/ASLR 以外的工具
- We need to do better
- Before and After

記憶體漏洞

記憶體漏洞

記憶體漏洞都是來自於載入可能錯誤的內容到程式中。

- Buffer overflow (heap 或 stack)
- Type confusion
- Use-after-free (UAF)
- Integer overflow
- stack corruptions
- Inter-chunk heap overflow or corruption
- Intra-chunk heap overflow or relative write
- Shader Parameter heap corruption
- ……

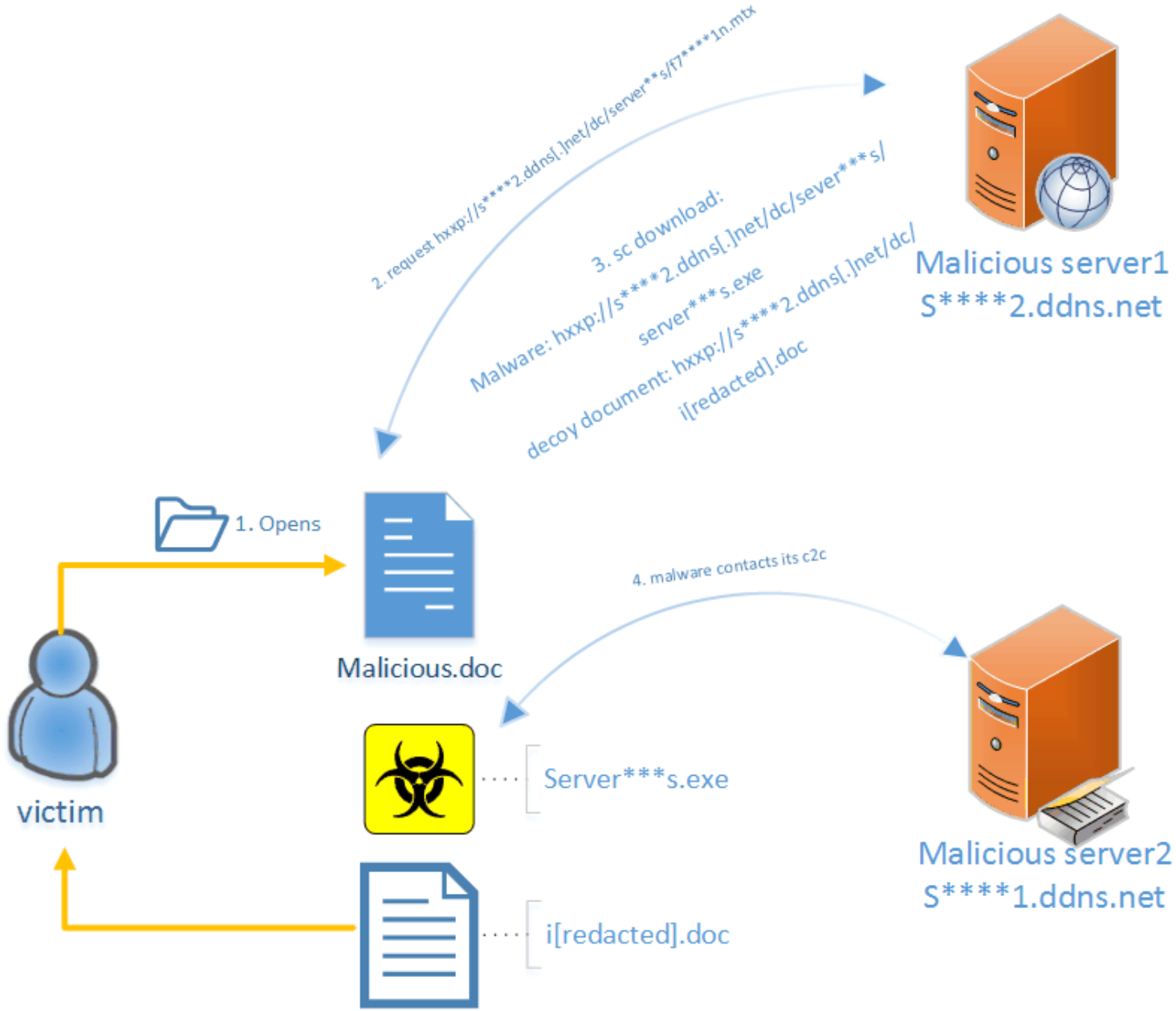
你可以用記憶體漏洞做甚麼事

- 執行惡意程式
- 操控電腦
- 攻擊網路
- 窺視其他網路主機
- 竊取資料
- 破壞數據
- 讓程式或系統當機
- 拒絕服務
- Everything bad all because of you

記憶體攻擊範例:
CVE-2016-4117

記憶體攻擊類型: Type confusion

1. 受害者開啟惡意的 Office 文件
    1. Office 文件開始執行內嵌的 Flash 檔案
        1. 如果 Flash Player 版本太舊，攻擊就會終止
        2. 否則，攻擊就會執行內嵌的 Flash Exploit (Type Confusion/All Memory Attack 都是在這裡發生, Devils are here)
2. Exploit 執行內嵌的原生 Shellcode
    1. Shellcode 會從攻擊者的伺服器，下載並執行第二個 Shellcode
3. 第二個 Shellcode
    1. 下載並執行 Malware
    2. 下載並顯示 Decoy 文件
4. Malware 連線到第二個 Command and Control (C2) 伺服器，等待進一步的指示



Source: https://tirateunping.wordpress.com/2016/05/17/cve-2016-4117-fireeye-revealed-the-exploit-chain-of-recent-attacks/

長久以來，微軟也體認到記憶體漏洞的嚴重威脅，所以陸續推出相關防護的技術。

How Windows execute an executable?

Process Creation Overview

1. **.EXE is loaded from disk**
   - Parse PE headers
   - Map sections into memory
   - Parse Import Table
2. **Load DLL dependencies**
   - Resolve import API functions
3. **Transfer execution to the .EXE Entry Point**
   - AddressOfEntryPoint

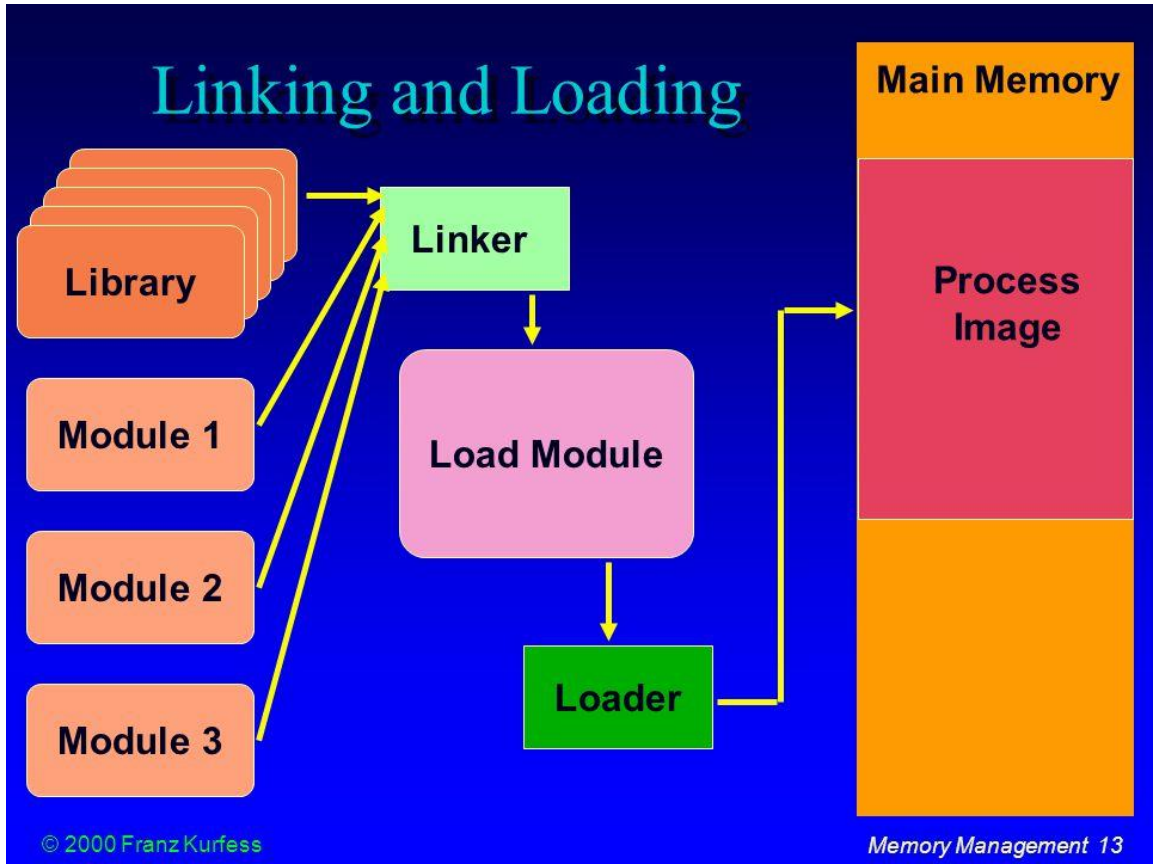


Figure Source: Memory Management 1 © 2000 Franz Kurfess Course Overview Principles of Operating Systems Introduction Computer System Structures Operating System.

Process Memory

- **Windows implements a virtual memory model**
  - Virtual memory is independent of physical memory
- **Every process had its own private virtual address space**
  - Process are isolated and cannot inadvertently modify each others memory
  - Physical-to-virtual memory translations are managed in the Windows kernel
  - Completely transparent to user process

EXE file launching

- **Each time an executable (.EXE) is launched, Windows will**
  - Validate the PE image and parameters
  - Create a virtual address space where the program will be loaded and executed
- **The virtual address space contains:**
  - All of the executable's code and data (the .EXE itself)
  - All of the DLL dependencies code and data (required .DLLs)

Process Memory Layout

- **PE files define a preferred base memory address in the Optional Header:**
  - EXE default: 0x400000
  - DLL default: 0x10000000
- **DLLs can be relocated by the Windows loader**
  - Occurs when preferred address already in use
  - Relocatable Dlls have a special .reloc section
- **ASLR (Address Space Layout Randomization) ensures DLLs will be moved**

What is PE and Why it is so important?

PE File format - Overview

Portable Executable (PE) file is the standard binary file format for Windows executables

- **.EXE**
  - Executable program. The Windows OS creates a virtual address space for program to run.
- **.DLL**
  - Dynamic Link Library. Windows concept of shard library. Also referred to as a module
- **.SYS**
  - Kernel driver. Executes in kernel-mode alongside core OS components

PE File Format – Header and Sections

- **The PE file format is a structured organization of Headers and Sections**
- **Header tell the OS how to interpret the PE file**
  - Type of the PE file (EXE/DLL/SYS)?
  - What memory location does execution begin? (Entry Point)
  - How should the sections be arranged in memory? (Section headers)
  - What DLL dependencies does the EXE need? (Imports)
  - What functionality does the PE file expose to other applications? (Exports)
- **Sections store the PE file content. This includes:**
  - Executable code
  - Program data
  - Binary resources

PE File Format – PE Optional Header

- **The PE Optional Header is used to store NT-specific attributes.**

  On NT systems the "optional" header is required.

  - ImageBase
    - Tells the OS its preferred base memory address
  - AddressOfEntryPoint
    - Tells the OS where to start executing
  - Other metadata:
    - Subsystem: (e.g., GUI, Console, Native, WinCE, etc.)
    - DllCharacteristics: Security-related linker options (e.g., ASLR, NX, DEP, SAFESEH)
    - Minimum supported NT version

Figure Source: http://www.cnblogs.com/shangdawei/p/4785494.html

微軟防護記憶體攻擊的工具

Enhanced Mitigation Experience Toolkit, EMET

- 一種公用程式，整合多種防止緩衝區漏洞功能，預防軟體中的弱點被利用。
- 使用安全防護技術。這些技術可以提高入侵的障礙，必須通過才能利用軟體弱點。
- 不保證弱點不被利用。然而可讓弱點更難以遭到入侵。
  - DEP (Data Execution Prevention，資料防止執行)
  - SEHOP (Structured Exception Handling Overwrite Protection，防止結構異常處理覆寫)
  - ASLR (Address Space Layout Randomization，記憶體位置編排隨機化)

Source: https://support.microsoft.com/zh-tw/kb/2909257

EMET防護模組

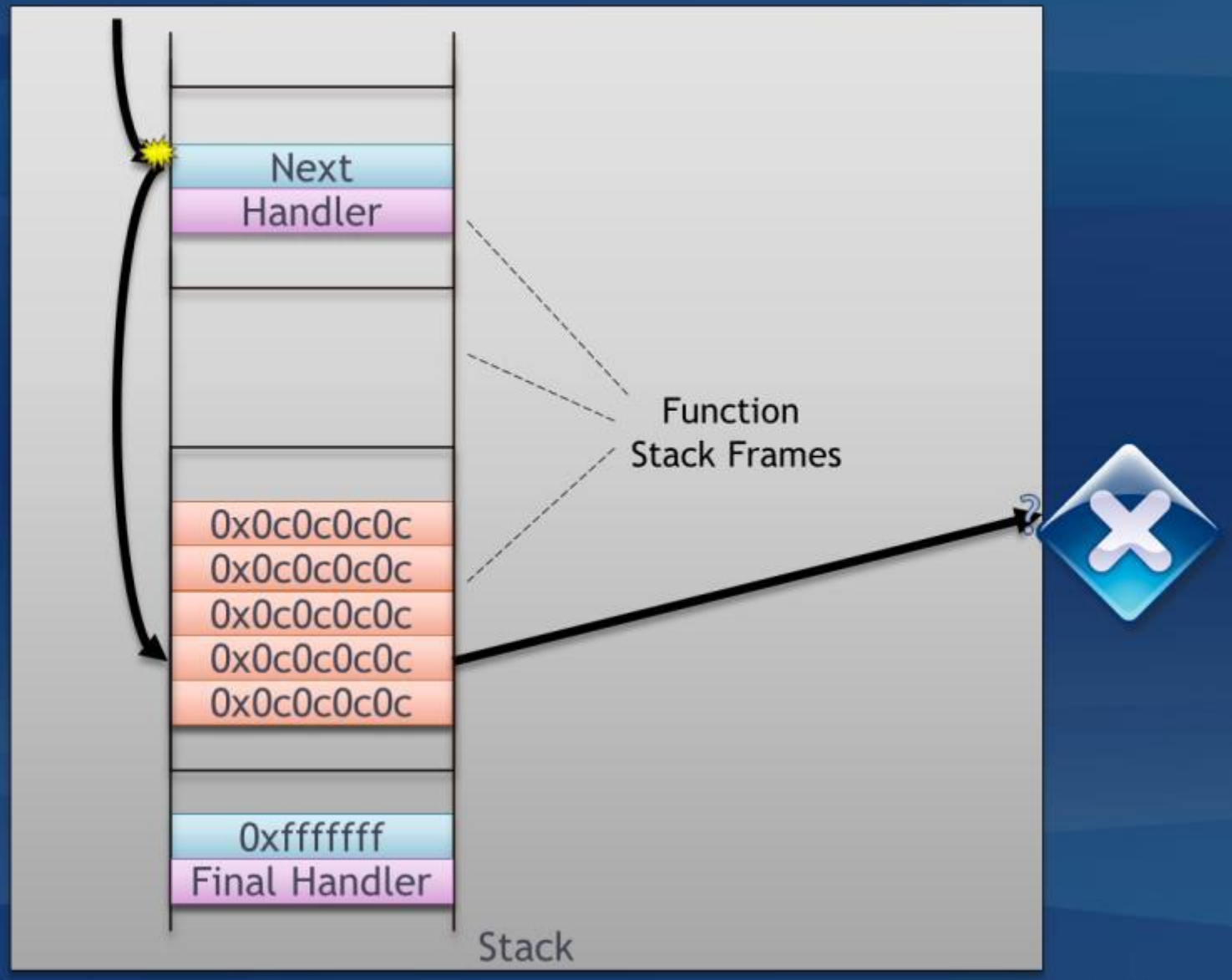
EMET Security Mitigations	Included
Attack Surface Reduction (ASR) Mitigation	✓
Export Address Table Filtering (EAF+) Security Mitigation	✓
Data Execution Prevention (DEP) Security Mitigation	✓
Structured Execution Handling Overwrite Protection (SEHOP) Security Mitigation	✓
NullPage Security Mitigation	✓
Heapspray Allocation Security Mitigation	✓
Export Address Table Filtering (EAF) Security Mitigation	✓
Mandatory Address Space Layout Randomization (ASLR) Security Mitigation	✓
Bottom Up ASLR Security Mitigation	✓
Load Library Check – Return Oriented Programming (ROP) Security Mitigation	✓
Memory Protection Check – Return Oriented Programming (ROP) Security Mitigation	✓
Caller Checks – Return Oriented Programming (ROP) Security Mitigation*	✓
Simulate Execution Flow – Return Oriented Programming (ROP) Security Mitigation*	✓
Stack Pivot – Return Oriented Programming (ROP) Security Mitigation	✓
Windows 10 untrusted fonts***	✓

** Available and applicable only to 32-bit processes*
**** Available on EMET 5.5 Beta, and available only for Windows 10*

*** EMET 5.2 and 5.5 Beta support Windows Vista Service Pack 2, Windows 7 Service Pack 1, Windows 8, Windows 8.1, Windows Server 2008 Service Pack 2, Windows Server 2008 R2 Service Pack 1, Windows Server 2012, Windows Server 2012 R2. EMET 5.5 Beta also supports Windows 10.*

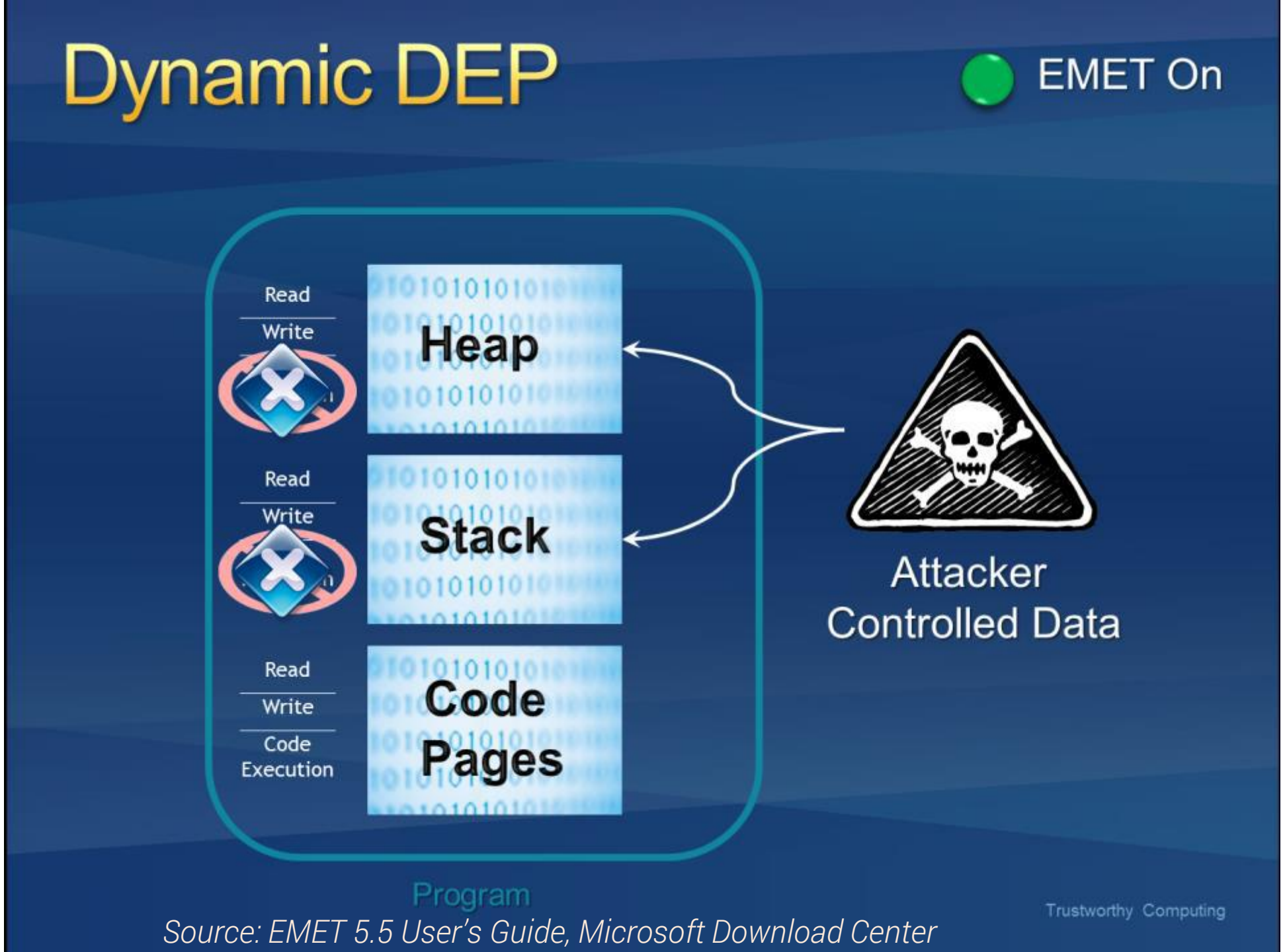
SEHOP

EMET On

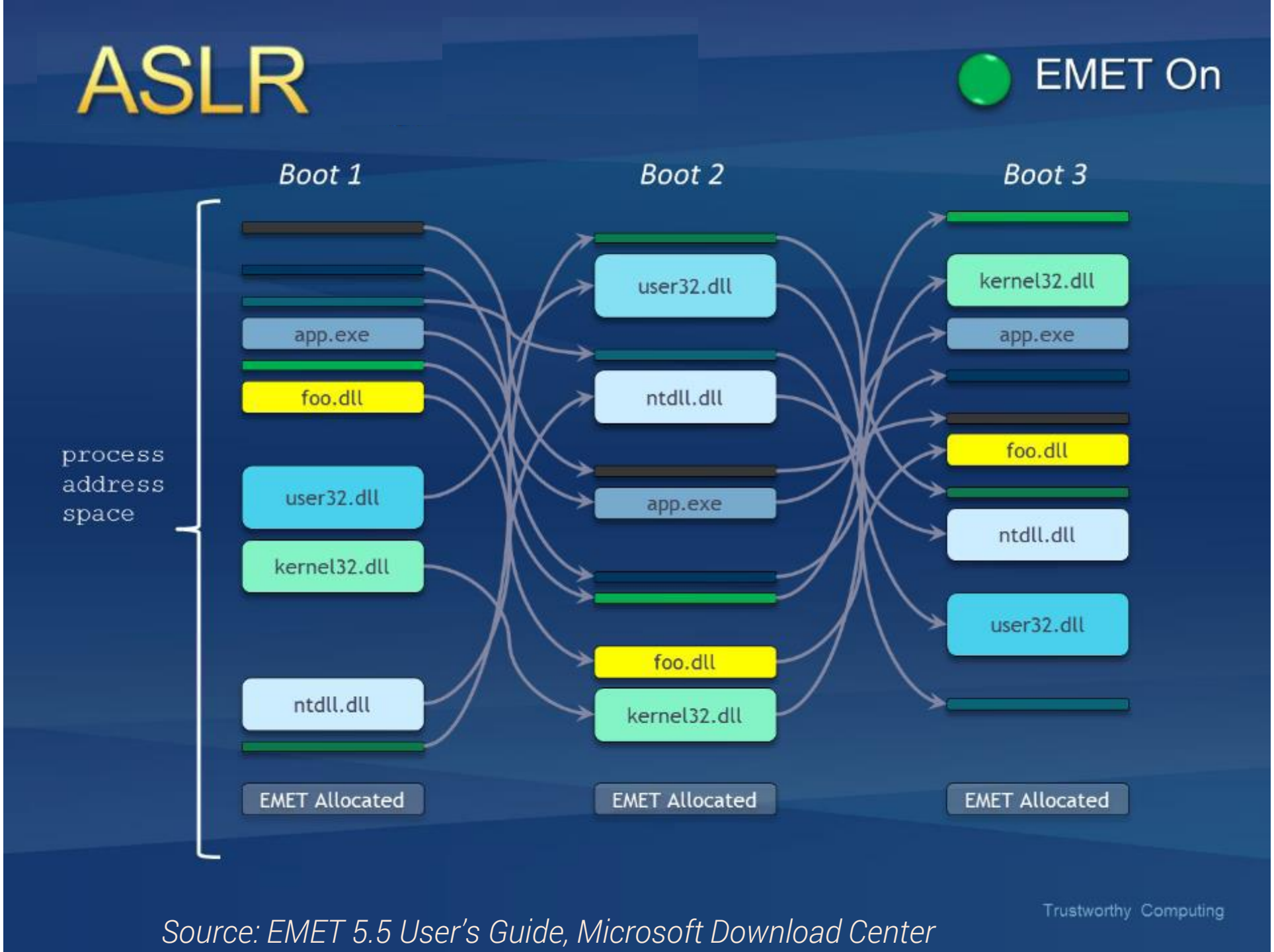


Trustworthy Computing

Source: EMET 5.5 User's Guide, Microsoft Download Center



Source: EMET 5.5 User's Guide, Microsoft Download Center



Source: EMET 5.5 User's Guide, Microsoft Download Center

Randomization Examples

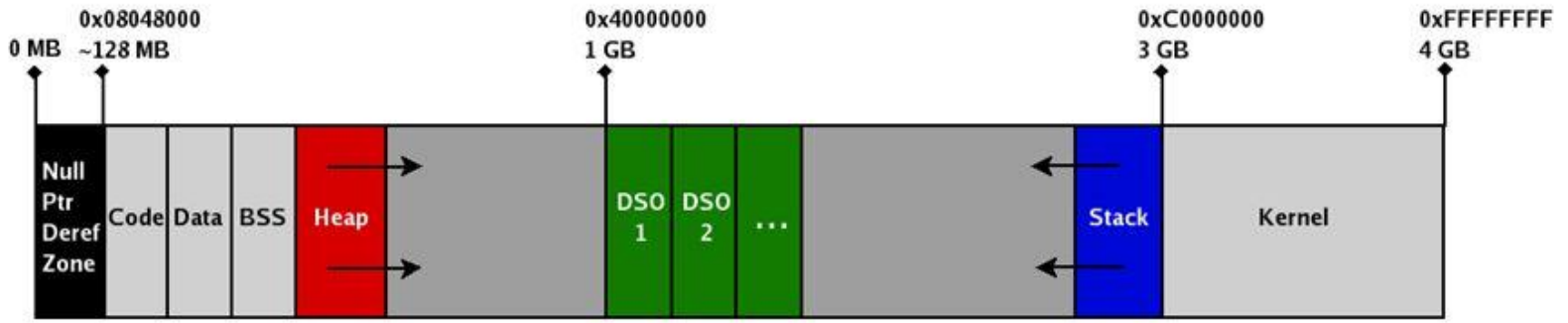


Fig 1. Normal Process Memory Layout

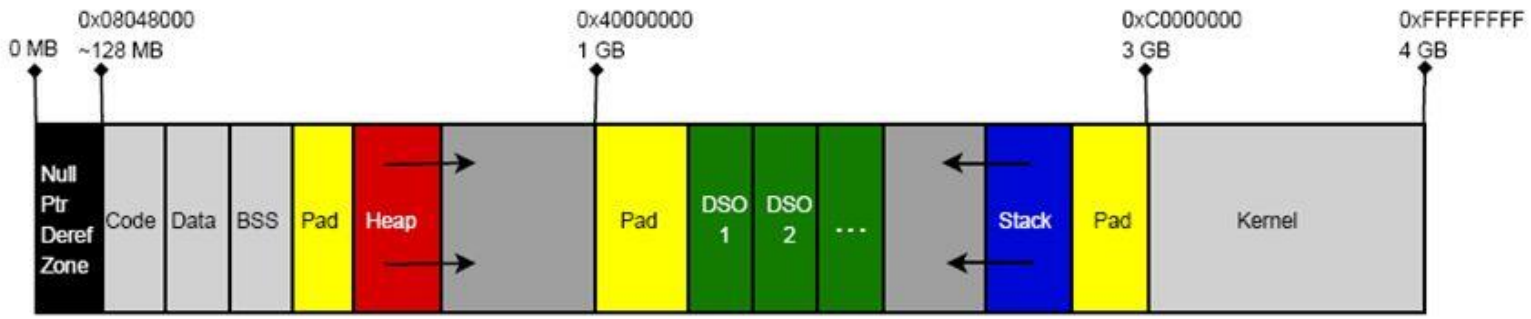


Fig 2. PaX ASLR Process Memory Layout

Figure Source: Computer Science 10/06/051 Address Space Layout Permutation Chongkyung Kil Systems Research Seminar

EMET的版本沿革

Cat and Mouse

- **Defense: Make stack/heap nonexecutable to prevent injection of code**
  - Attack response: Jump/return to libc
- **Defense: Using ASLR to hide the address of desired libc code or return address**
  - Attack response: Brute force search (32-bit systems) or information leak (format string vulnerability)
- **Defense: Avoid using libc code entirely and use code in the program text instead**
  - Attack response: Construct needed functionality using Return Oriented Programming (ROP)

*Source: https://www.youtube.com/watch?v=XZa0Yu6i_ew, **Return Oriented Programming – ROP**, Maryland Cyber Security Center*

Figure Source: http://www.slideshare.net/saumilshah/exploit-delivery/13-netsquareGSSafeSEHDEPASLRPermanent_DEPASLR_and_DEPSEH_overwritesnonSEH

EMET 1.x, released in October 27, 2009

- Structured Exception Handling Overwrite Protection (SEHOP): Provides protection against exception handler overwriting.
- Dynamic Data Execution Prevention (DEP): Enforces DEP so data sections such as stack or heap are not executable.
- NULL page allocation: Prevents exploitation of null dereferences.
- Heap spray allocation: Prevents heap spraying..

Source: https://www.fireeye.com/blog/threat-research/2016/02/using_emet_to_disabl.html

EMET 2.x, released in September 02, 2010

- **Mandatory Address Space Layout Randomization (ASLR):** Enforces modules base address randomization; even for legacy modules, which are not compiled with ASLR flag.
- **Export Address Table Access Filtering (EAF)**: EMET uses hardware breakpoints stored in debugging registers (e.g. DR0) to stop any thread which tries to access the export table of kernel32.dll, ntdll.dll and kernelbase.dll, and lets the EMET thread verify whether it is a legitimate access.

Source: https://www.fireeye.com/blog/threat-research/2016/02/using_emet_to_disabl.html

EMET 3.x, released in May 25, 2012

- Imported mitigations from **ROPGuard** to protect against Return Oriented Programming (ROP).
- **Load Library Checks**: Prevents loading DLL files through Universal Naming Convention (UNC) paths.
- **ROP Mitigation - Memory protection checks**: Protects critical Windows APIs like VirtualProtect, which might be used to mark the stack as executable.
- **ROP Mitigation - Caller check**: Prevents critical Windows APIs from being called with jump or return instructions.
- **ROP Mitigation - Stack Pivot**: Detects if the stack has been pivoted.
- **ROP Mitigation - Simulate Execution Flow**: Detects ROP gadgets after a call to a critical Windows API, by manipulating and tracking the stack register.
- **Bottom-up ASLR**: Adds entropy of randomized 8-bits to the base address of the bottom-up allocations (including heaps, stacks, and other memory allocations).

Source: https://www.fireeye.com/blog/threat-research/2016/02/using_emet_to_disabl.html

EMET 4.x, released in April 18, 2013

- **Deep Hooks**: With this feature enabled, EMET is no longer limited to hooking what it may consider as critical Windows APIs, instead it hooks even the lowest level of Windows APIs, which are usually used by higher level Windows APIs.
- **Anti-detours**: Because EMET places a jump instruction at the prologue of the detoured (hooked) Windows API functions, attackers can craft a ROP that returns to the instruction that comes after the detour jump instruction. This protection tries to stop these bypasses.
- **Banned functions**: By default it disallows calling ntdll!LdrHotpatchRoutine to prevent DEP/ASLR bypassing. Additional functions can be configured as well.
- **Certificate Trust (configurable certificate pinning):** Provides more checking and verification in the certificate chain trust validation process.

Source: https://www.fireeye.com/blog/threat-research/2016/02/using_emet_to_disabl.html

EMET 5.x, released in July 31, 2014

- Introduced **Attack Surface Reduction (ASR):** Allows configuring list of modules to be blocked from being loaded in certain applications.
- **EAF+:** Similar to EAF, it provides additional functionality in protecting the export table of kernel32.dll, ntdll.dll and kernelbase.dll.

Source: https://www.fireeye.com/blog/threat-research/2016/02/using_emet_to_disabl.html

EMET的限制

EMET 的注意事項

- 某些主機型入侵預防系統 (HIPS) 應用程式可能會提供類似於 EMET 的保護。在系統上同時安裝這些應用程式和 EMET 時，可能需要進行其他設定，以便讓這兩種產品共存。

  此外，EMET 目的是與桌面應用程式 (User Application) 搭配使用，因此您只應保護會接收或處理不受信任資料的應用程式。系統和網路服務不屬於 EMET 的範圍。雖然技術上可能可以使用 EMET 保護這些服務，但我們不建議您執行這項操作。

Source: https://support.microsoft.com/zh-tw/kb/2909257

不建議使用 EMET 保護的軟體

- EMET 安全防護功能在作業系統的極低層級 (Kernel) 運作
- 某些在類似低層級運作的軟體類型在設為使用 EMET 保護時，可能會發生相容性問題
- 下列為不建議使用 EMET 保護的軟體類型清單：

  - 反惡意程式碼和入侵預防或偵測軟體
  - 偵錯工具
  - 處理數位版權管理 (DRM) 技術 (也就是電玩) 的軟體
  - 使用防偵錯、模糊化或攔截技術的軟體

Source: https://support.microsoft.com/zh-tw/kb/2909257

EMET 不是 Windows 的內建工具

但 EMET 沒有成為 Windows 系統的預設程式

因為有許多工具程式為了保護其程式不被反組譯或有其他特定目的，在 EMET 的保護架構下沒辦法正常運行

不相容的安全防護功能

產品	EMET 4.1 Update 1	EMET 5.2	EMET 5.5
7-Zip 主控台/GUI/檔案管理員	EAF	EAF	EAF
AMD 62xx 處理器	EAF	EAF	EAF
Beyond Trust Power Broker	不適用	EAF、EAF+、Stack Pivot	EAF、EAF+、Stack Pivot
某些 AMD/ATI 視訊驅動程式	System ASLR=AlwaysOn	System ASLR=AlwaysOn	System ASLR=AlwaysOn
DropBox	EAF	EAF	EAF
Excel Power Query、Power View、Power Map 和 PowerPivot	EAF	EAF	EAF
Google Chrome	SEHOP*	SEHOP*	SEHOP*、EAF+
Google Talk	DEP、SEHOP*	DEP、SEHOP*	DEP、SEHOP*
Immidio Flex+	不適用	EAF	EAF
McAfee HDLP	EAF	EAF	EAF
Microsoft Office Web Components (OWC)	System DEP=AlwaysOn	System DEP=AlwaysOn	System DEP=AlwaysOn
Microsoft Word	Heapspray	不適用	不適用
Oracle Java‡	Heapspray	Heapspray	Heapspray
Pitney Bowes Print Audit 6	SimExecFlow	SimExecFlow	SimExecFlow
8.1.1.9 版 Siebel CRM	SEHOP	SEHOP	SEHOP
Skype	EAF	EAF	EAF
SolarWinds Syslogd Manager	EAF	EAF	EAF
VLC Player 2.1.3+	SimExecFlow	不適用	不適用

Source: https://support.microsoft.com/zh-tw/kb/2909257

EMET 的缺點

1. Rule必須先定義
2. 程式必須先設定
3. EMET在 Kernel 層運作有相容性問題
4. 需要大量的記憶體
5. 重新開機設定變更才會生效
6. 不提供攻擊詳細資訊
7. 在 Win 7, 8 與 10 可以被跳過

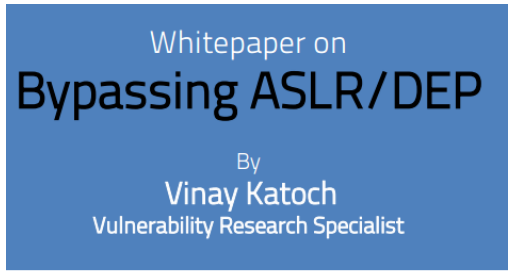
記憶體位置編排隨機化缺點
Address Space Layout Randomization (ASLR)

1. 只有每次重開機才會再次隨機排列
2. 針對編譯階段並未啟用ASLR的模組不支援，或是效果不好(Win8以上)
3. 在 32 位元系統上會有低不可預測性 (Low entropy) 的問題，造成容易暴力破解攻擊
4. Only get exception, crash or stuck
5. 發生攻擊不會警示
6. 不提供攻擊鑑識資訊
7. 容易被攻擊工具破解

閃過 **ASLR** 其實很簡單

- Information that will evade the ASLR. There are mainly two ways:

1. Any anti ASLR modules gets loaded into the target application. You have the base address of any module at fixed location always even after the system restart.
2. You get a pointer leak from a memory leak/buffer overflow/any zero day. You can adjust the offsets to grab the base address of the module whose pointer gets leaked.
   - When you have a pointer, so you can either make your shellcode from **ROP**, ROP is a little advanced return to LibC attack and is return oriented programming.



Source: Whitepaper on Bypassing ASLR/DEP, Vinay Katoch

除了EMET/ASLR還有甚麼工具

Current Solutions

- Patch
- Traditional White listing
- Machine learning
- Anti-exploitation tools

永遠都在更新 Patch

PROS:

- 大多數非針對型攻擊會繼續使用相同的漏洞，因為可以快速散播，所以更新程式可以有效防堵

CONS:

- 企業很難承受更新週期所造成的作業中斷
- 每次更新都要重新測試與所有開發軟體的相容性
- 在更新周期當中又冒出新的漏洞
- 某些更新程式又出現要命的相容性問題

傳統白名單或 Group Policy 控管

PROS:

- 有助於對抗某些類型的攻擊套件 (例如大多數的勒索軟體都會下載檔案到磁碟)

CONS:

- 管理非常麻煩 (大多數的企業討厭傳統白名單解決方案)
- 對於非下載檔案類型的攻擊，傳統白名單不能阻止 (例如 Registry 或是記憶體攻擊)
- 如果沒有 Windows 認證，傳統白名單產品會在 Kernel 層產生很多相容性問題

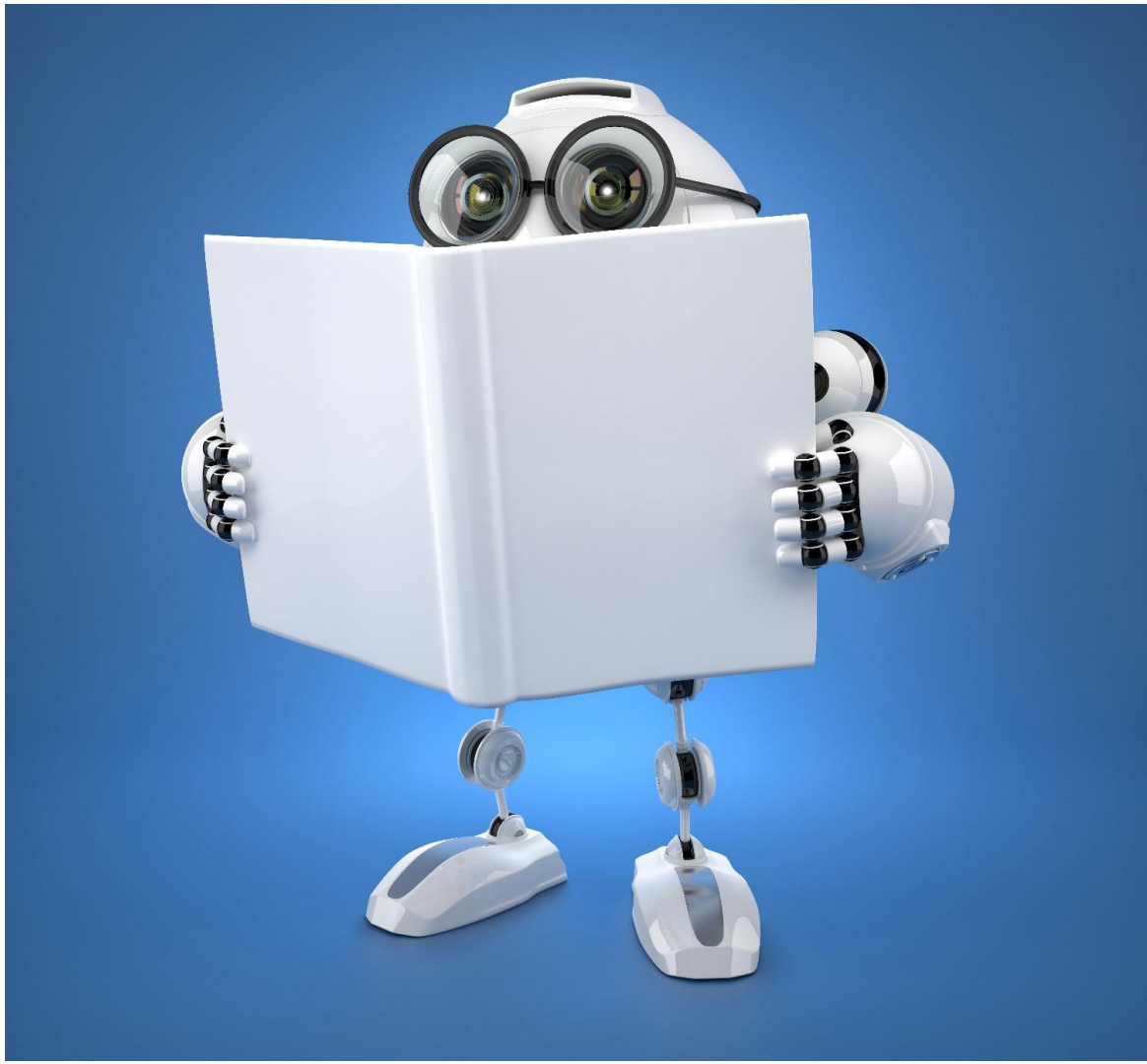
機器學習

PROS:

- 可以不需要特徵值，掃描環境中惡意程式的存在 (在被執行前)

CONS:

- 誤報很多，需要很多微調來適應企業的環境
- 難以防範結合多個惡意程式所發動的聯合攻擊
- 對真正的高階針對型攻擊無效



Anti-exploitation 工具

PROS:

- 可以阻止大多數的一般攻擊套件(例如 Angler, RIG, Nuclear)

CONS:

- 這類工具大多數都是 Rule based – 也就是摸熟的話就可以 Bypass
- 消耗大量處理器資源
- 管理麻煩

現有防禦機制的共同問題

- 除了白名單機制之外，以上三種防禦機制都必須對攻擊行為有所了解
- 但是企業對於白名單的接受度不高
- 有沒有更有效的新思維、新觀念呢?

EMET/ASLR is not enough
We need to do better

資訊安全的根本改變

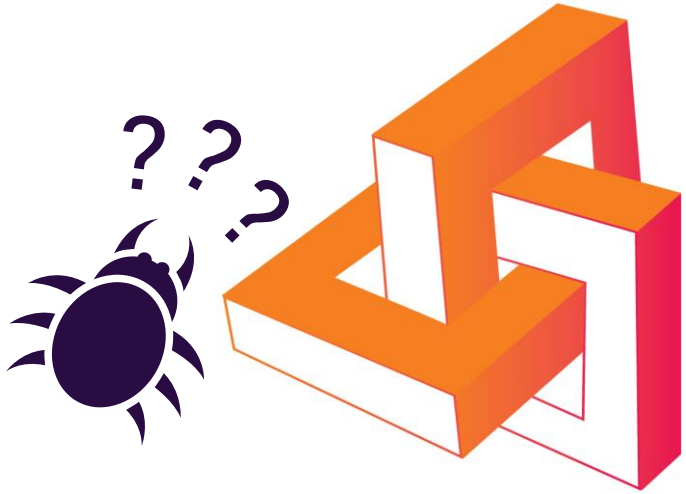
防守方想盡辦法找出難以捉摸的攻擊與威脅

傳統與號稱次世代的解決方案

偵測 **Detection**

- 特徵 Signatures
- 行為 Behavioral
- 啟發式演算法 Heuristics

讓攻擊方想盡辦法找尋難以捉摸的目標



移動目標防護 **Moving Target Defense**

防護 **Prevention**

- 多樣變形 Polymorphism
- 隱藏目標 Hiding the target
- 確定性 Deterministic

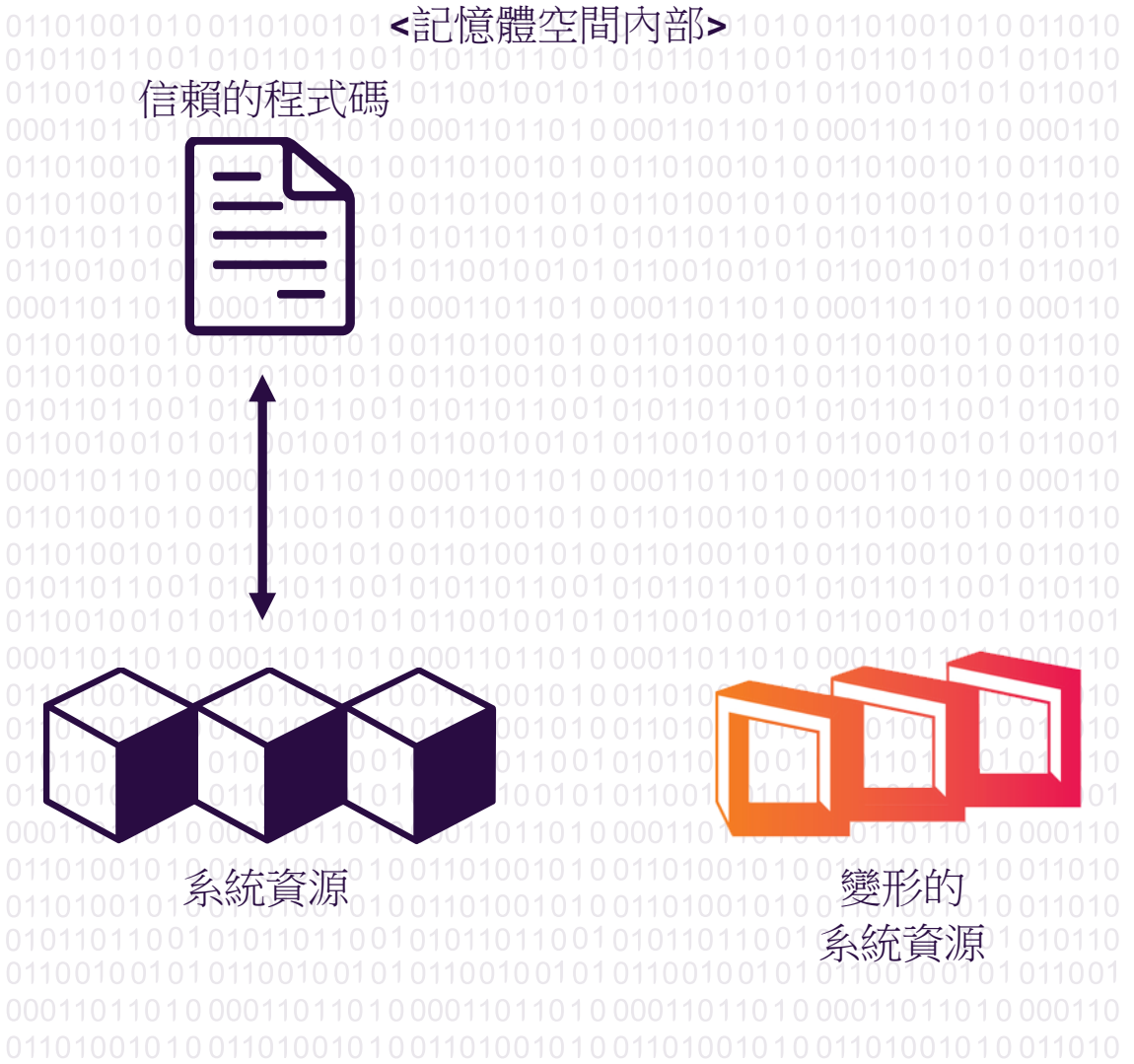
移動目標防護 Moving Target Defense (MTD)

使用者啟動應用程式，隨後載入記憶體空間中。

步驟 01

將記憶體結構變形，讓記憶體無法被猜測以進行攻擊。

- 每次載入程式時就即時變形
- 單向的隨機重組，沒有還原金鑰

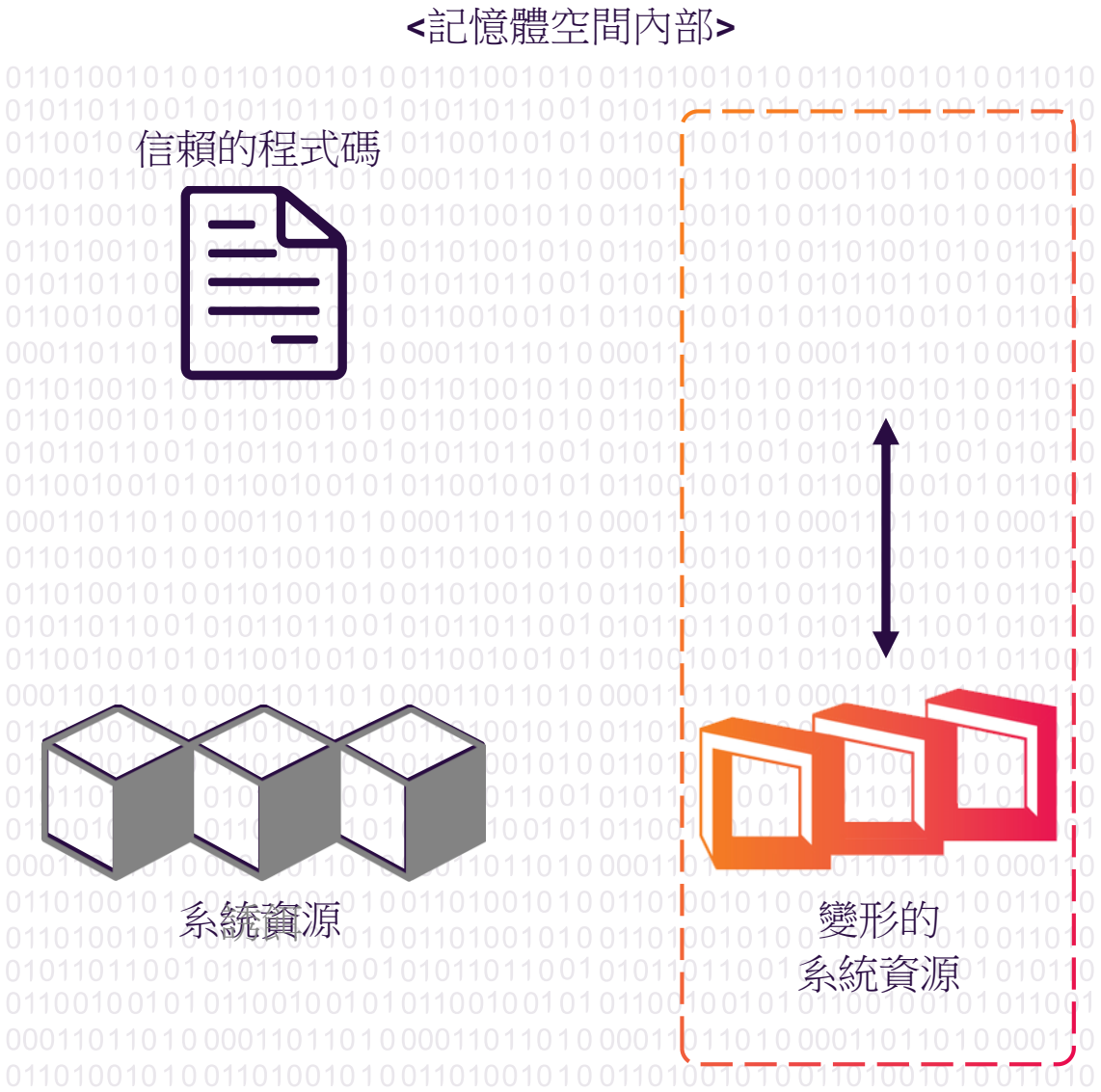


移動目標防護 Moving Target Defense (MTD)

步驟 02

讓處理程序知道有一個合法存在的新變形記憶體結構。

- 保留原本結構的虛擬副本
- 應用程式照正常運作



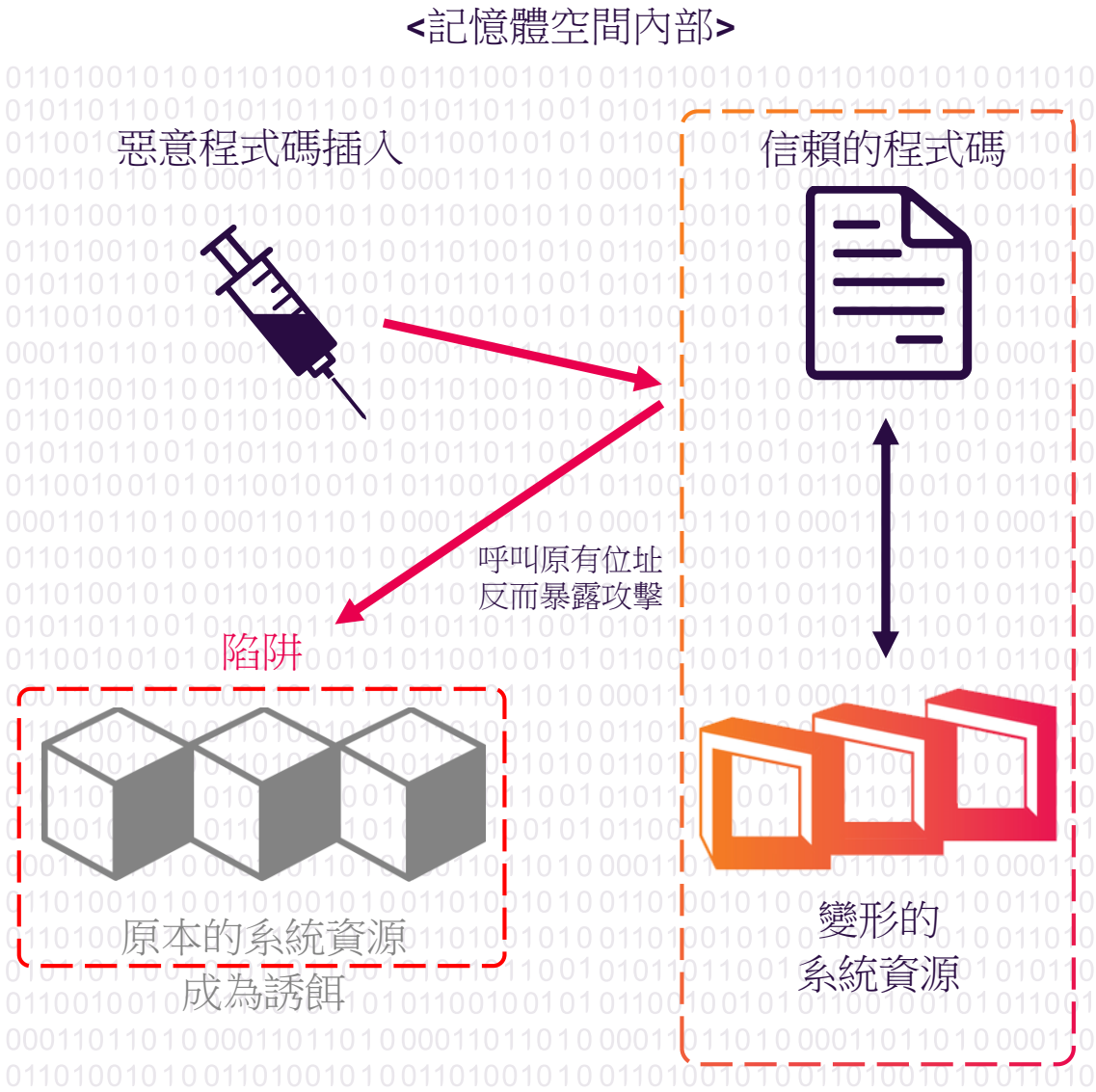
移動目標防護 Moving Target Defense (MTD)

步驟 03

任何嘗試存取原本記憶體結構的程式碼，並不會知道位址變化，就會被認定是惡意程式!

步驟 04

在初期刺探時，攻擊就會立刻掉入陷阱，並予以儲存以準備進一步調查。

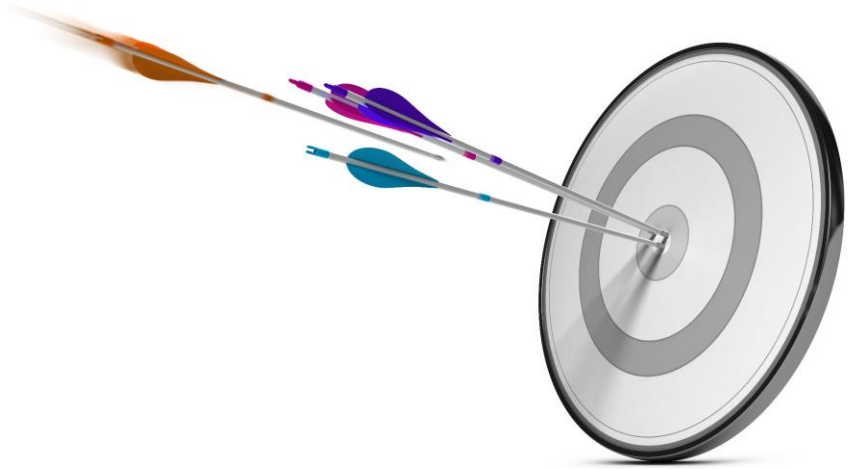


記憶體攻擊手法展示
Before and after MTD

APT/Ransomware

- 正所謂

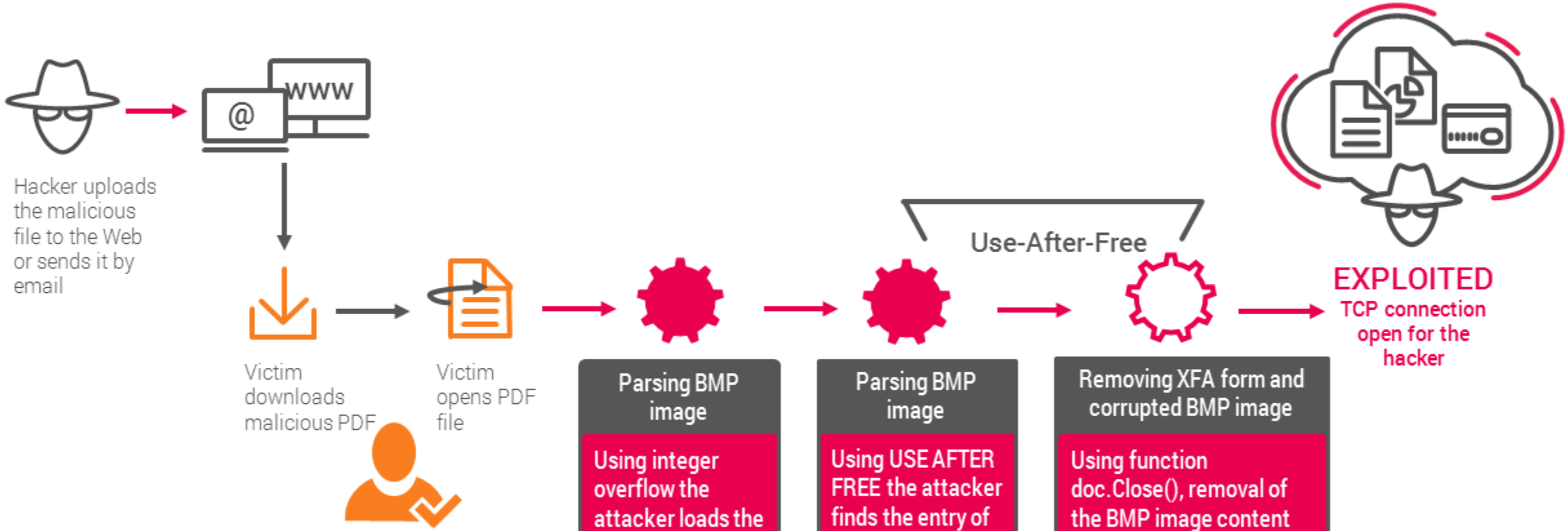
烈女怕纏郎(針對性)



靚女怕色狼(大規模)

Adobe Reader: BMP/RLE

Exploited vulnerability: BMP File Parsing Integer Overflow



CVE 2015-0336

Exploited vulnerability:
Type Confusion

- Hacker creates BitmapData object which points to a valid ARGB pixel
- The NetConnection connect function will try to access the netconnection field in the BitmapData object, which point to an IP location

Hacker uploads flash html and swf (flash) file to server

Malicious website / server

HTTP Request receives html and flash/media resources

Internet Explorer

Plays swf file

Flash Player before 16.0.0.305

Object points to valid ARGB pixel

BitmapData with "fake" netconnection pointer, passes the typecheck successfully

Native function trigger

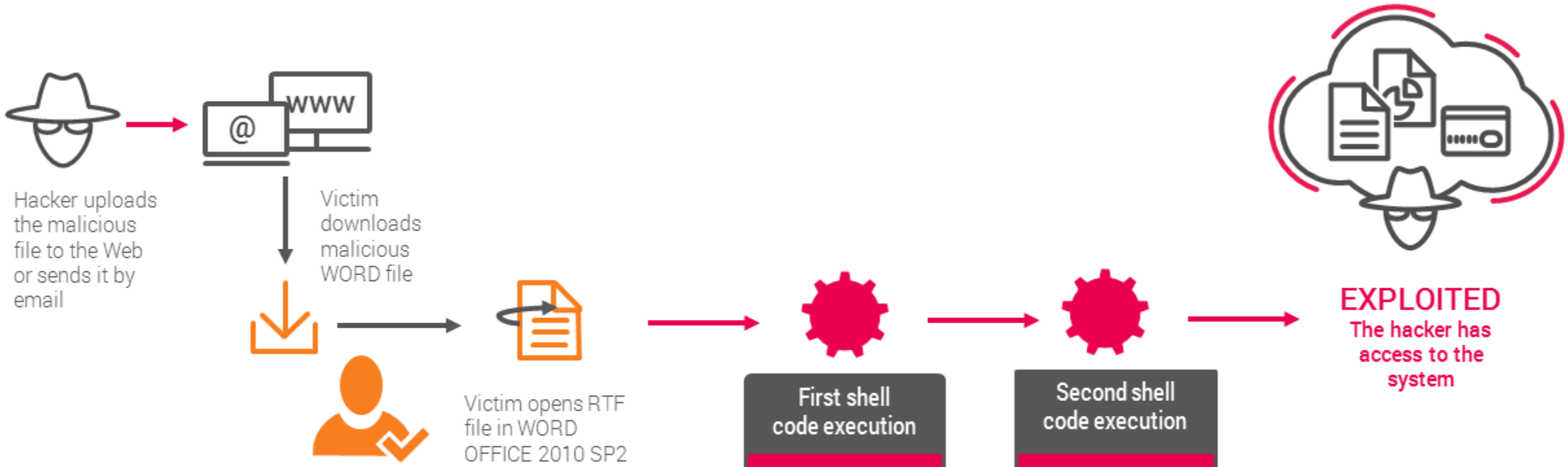
Attempt to access faked vtable in the BitmapData object, which was set to a specific IP

Triggers the vulnerability

EXPLOITED

TCP connection open for the hacker

Microsoft Word RTF Corrupted File



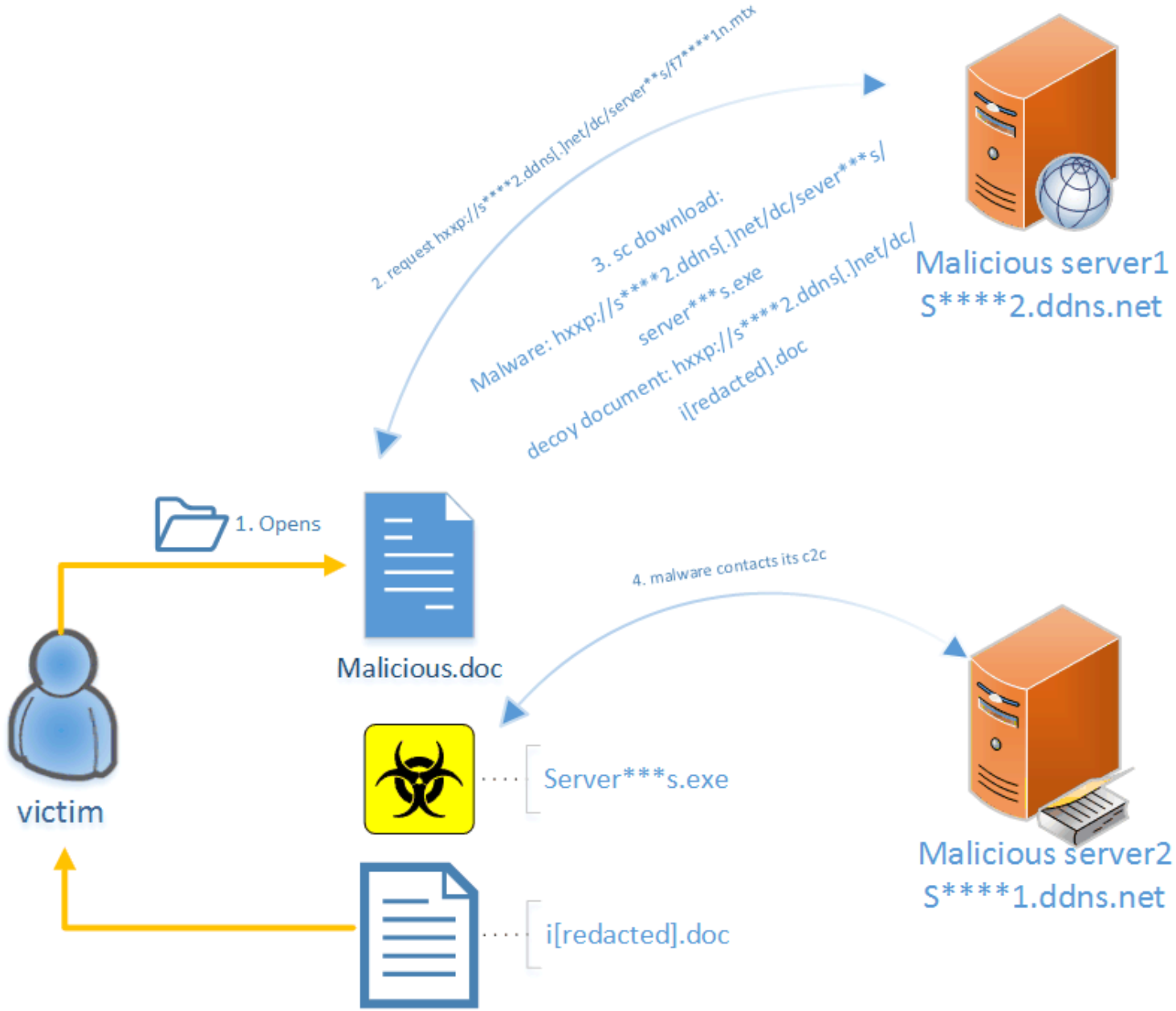
Exploited vulnerability:
Buffer overflow

Buffer overflow of a structure in RTF format

記憶體攻擊範例: CVE-2016-4117

記憶體攻擊類型: Type confusion

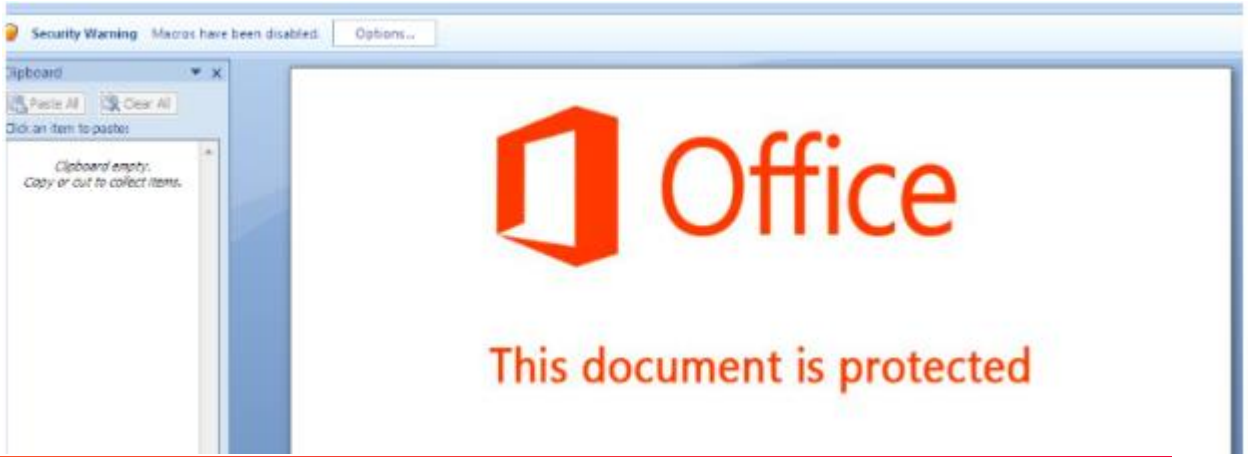
1. 受害者開啟惡意的 Office 文件
    1. Office 文件開始執行內嵌的 Flash 檔案
        1. 如果 Flash Player 版本太舊，攻擊就會終止
        2. 否則，攻擊就會執行內嵌的 Flash Exploit (Type Confusion/All Memory Attack 都是在這裡發生, Devils are here)
2. Exploit 執行內嵌的原生 Shellcode
    1. Shellcode 會從攻擊者的伺服器，下載並執行第二個 Shellcode
3. 第二個 Shellcode
    1. 下載並執行 Malware
    2. 下載並顯示 Decoy 文件
4. Malware 連線到第二個 Command and Control (C2) 伺服器，等待進一步的指示



Source: https://tirateunping.wordpress.com/2016/05/17/cve-2016-4117-fireeye-revealed-the-exploit-chain-of-recent-attacks/

A Brief History of Hancitor

- Hancitor (aka Chanitor and TorDal) is a downloader-type malware and usually a part of a larger targeted campaign.
- New evasive technique(s) that allow it to elude most existing endpoint security solutions.
- Using an embedded calls to launch and grab additional payloads.
- Injecting a DLL or EXE downloaded from a URL and executing it without writing it to the disk.



移動目標防護新觀念

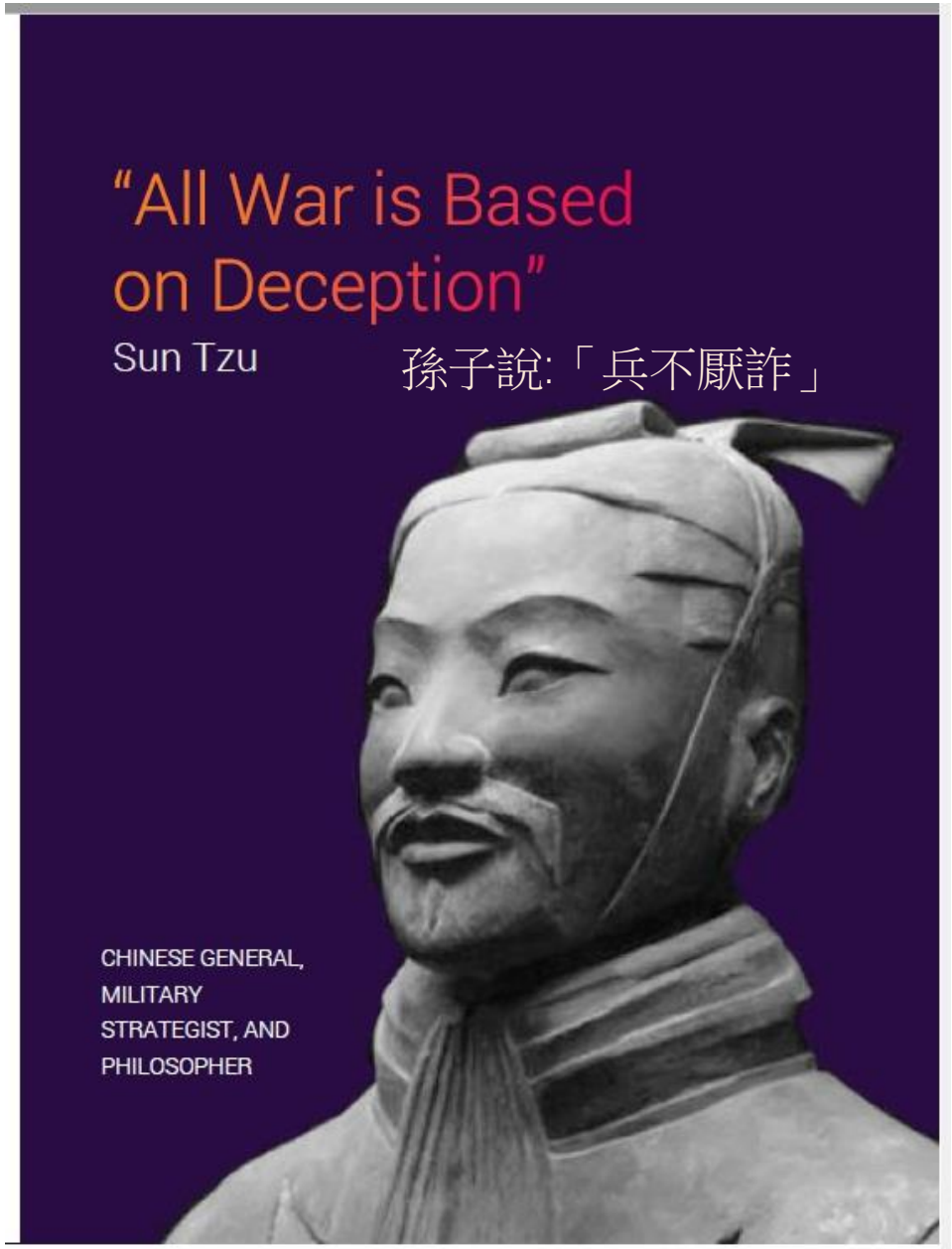
1. 不須設定 **Rule**
2. 不須學習攻擊知識
3. 加入誘捕設計
4. 適用全部程式類型
5. 相容現有資安產品
6. 沒有執行階段元件，不影響效能
7. 提供鑑識資訊，可供**SIEM**使用
8. 提供組織內部資安狀態資訊

Key Take Away

- 透過 **MTD** 技術可以防護所有的 **In-memory attacks**

Browser, Office, Adobe PDF/Flash, Java, Backdoor 五大類型

- **Zero-day, one-day, exploit based malware, PowerShell/Java Script**
  - 來自 Web 的攻擊，如勒索軟體
  - 所有的惡意檔案，如 Adobe 與 Office 文件
  - 窩藏 Flash, Silverlight 以及 JavaScript 攻擊的惡意或合法網站
  - 利用 shellcode 來執行 Payload 的 Java 攻擊 (大多數最近的攻擊類型)
  - File-less 或 Non-persistent 惡意程式的漏洞攻擊


"All War is Based on Deception"
Sun Tzu
孫子說:「兵不厭詐」
CHINESE GENERAL, MILITARY STRATEGIST, AND PHILOSOPHER

打不到，打不到，就是讓你打不到

Special Thanks to

01 | Matt Chen

02 | Jason Lai

03 | Julian Su

04 | Myself

Gartner

Cool Vendor 2016

“無論大小企業，凡是尋找最佳防護方案以對抗進階持續威脅、勒索軟體與入侵探刺等惡意攻擊的組織，都應該考慮 Morphisec。”

LeoLiaw@iSecurity.com.tw

iSecurity

The Hub of Security Innovations

MORPHISEC

Moving Target Defense