

**Vulnerability Assessment & Reverse Engineering**

**Assignment 03**

**Submitted by:** Samiullah Butt, Ashfaq Ahmed

**Roll number:** 22i-1663, 22i-1616

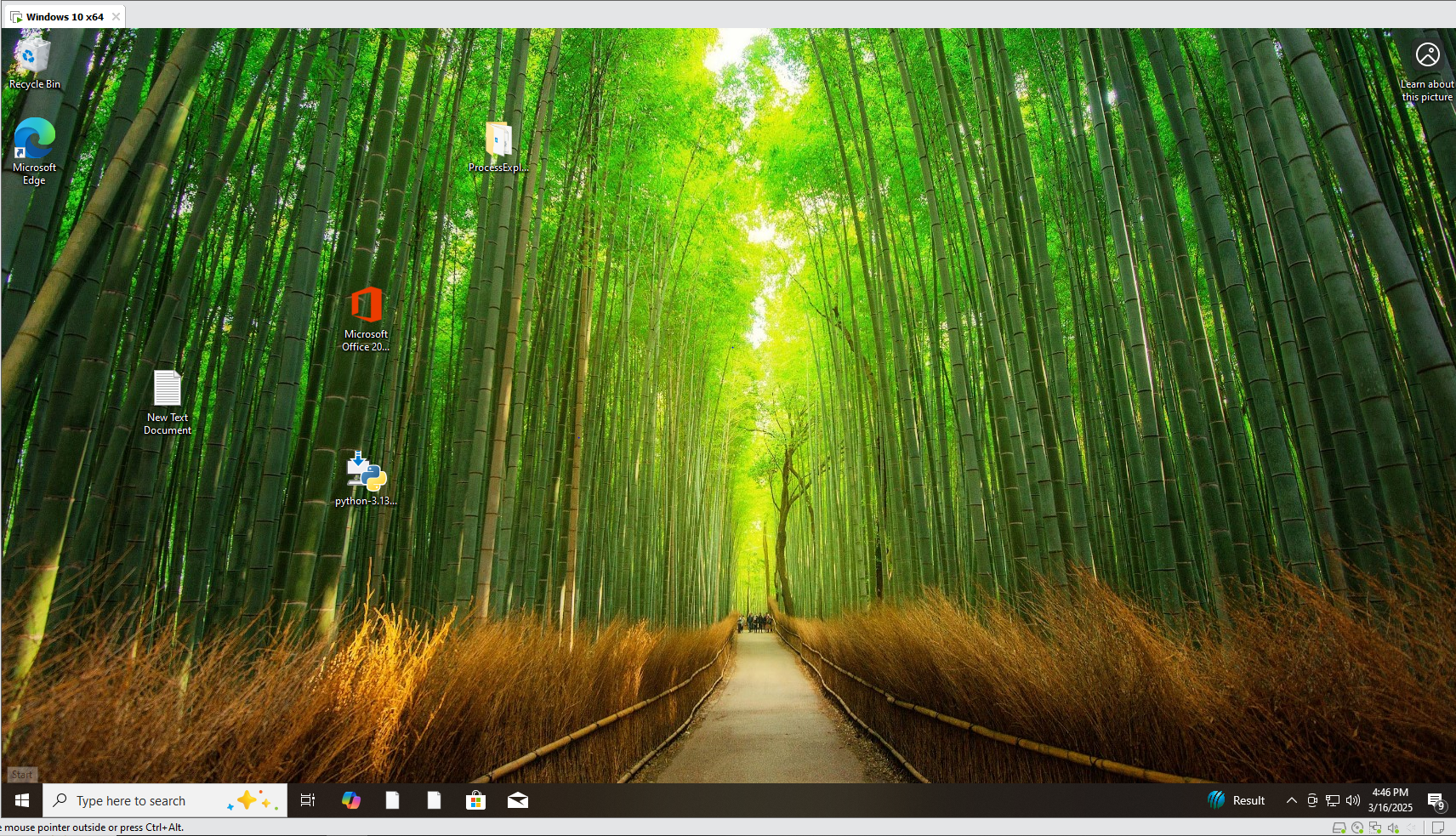
**Date:** 16-03-2024

**Q1: Understanding & Exploiting Vulnerabilities in Enterprise Environments**

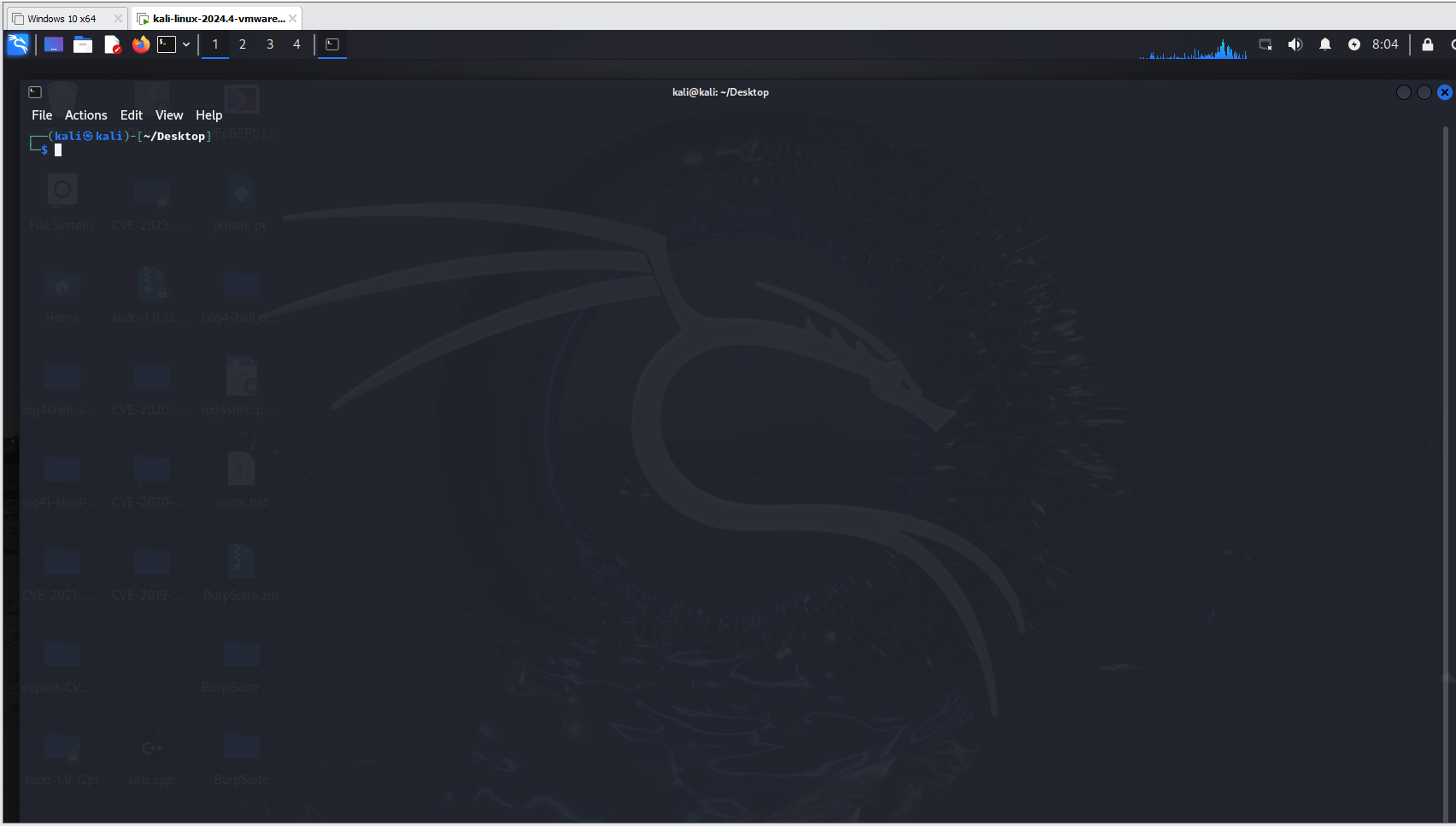
1. **Environment Setup**

To simulate a real-world scenario, the following environment has been set up:

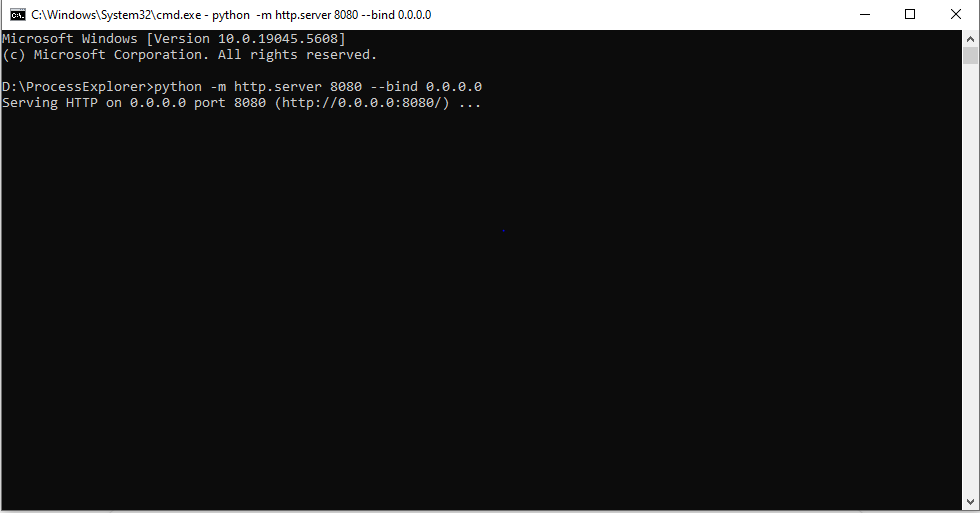
* **Windows 10 (Target Machine)**
  + Running on VMware Workstation.
  + Acts as the victim system where the Proof of concept will be executed to simulate the attacks.
  + Includes Sysinternals, 7zip tools to serve as the vulnerable application.



* **Kali Linux (Attacker Machine)**
  + Running on VMware Workstation.
  + Used to generate the DLL payload for injection.
  + Equipped with necessary tools such as msfvenom for payload generation.



* **Python HTTP Server (File Hosting Machine)**
  + Hosted on the main Windows machine.
  + Serves as a file-sharing server to deliver the malicious DLL.
  + Uses Python’s built-in HTTP server: python -m http.server 8080 --bind 0.0.0.0.



1. **Exploring Proof-of-Concept (PoC) Exploits**

**Microsoft Sysinternals Tools DLL Injection Vulnerability:**

Microsoft Sysinternals tools are widely used for system diagnostics and troubleshooting. However, a zero-day vulnerability in these tools allows attackers to perform DLL injection attacks, leading to unauthorized code execution.

**Vulnerability Overview**

The discovered vulnerability stems from how Sysinternals tools load DLL files. Specifically, many of these applications prioritize untrusted paths—such as the current working directory (CWD) or network paths over secure system directories when loading DLLs.

This oversight allows attackers to replace legitimate DLLs with malicious ones, enabling the execution of arbitrary code.

##### **Attack Mechanics:**

1. The attacker crafts a malicious DLL, such as cryptbase.dll or TextShaping.dll, embedding harmful payloads.
2. The malicious DLL is placed in the same directory as the legitimate Sysinternals executable (e.g., procexp64.exe.exe).
3. When the user executes the application from this directory, the malicious DLL is loaded instead of the trusted system DLL.
4. The attacker’s code executes under the user’s privileges, potentially leading to full system compromise.

* **Attack Type**: DLL Injection
* **Severity**: High (Privilege Escalation & Remote Code Execution possible)
* **CVE ID**: Not yet assigned (Zero-Day)

**Sources: Cyber Security News** – [Sysinternals Tools DLL Injection 0-Day](https://cybersecuritynews.com/0-day-vulnerabilities-in-microsoft-sysinternals-tools/) , **CyberConvoy** – [Sysinternals 0-Day Flaws](https://cyberconvoy.com/blog/microsoft-sysinternals-0-day-flaws-enable-dll-injection-attacks-on-windows/) , **Reddit (BlueTeamSec)** – [DLL Hijacking 0-Day in Sysinternals](https://www.reddit.com/r/blueteamsec/comments/1igqb8n/dll_hijacking_zeroday_vulnerability_in_microsoft/)

**Impact of the Vulnerability**

* **Execution of Arbitrary Code**: Attackers can execute malicious DLLs within the context of a trusted Sysinternals tool.
* **Privilege Escalation**: If the targeted tool is executed with administrative privileges, injected code gains higher-level access.
* **Potential for Remote Exploitation**: Attackers can deliver malicious DLLs through phishing or remote file-sharing services.
* **Denial of Service (DOS)**: Attackers can cause denial of service by exploiting this vulnerability

**Affected Systems**

The vulnerability impacts the following systems:

1. **Operating Systems**:
   * Windows 10
   * Windows 11
   * Windows Server (various versions)
2. **Microsoft Sysinternals Tools (Potentially Affected)**:
   * **Process Explorer (procexp64.exe)**
   * **Process Monitor (procmon.exe)**
   * **Autoruns (autoruns.exe)**
   * **Sysmon (sysmon.exe)**
   * **PsExec (psexec.exe)**
   * **Other Sysinternals utilities that load DLLs insecurely**
3. **Usage Scenarios Prone to Exploitation**:
   * Systems where Sysinternals tools are executed with **administrator privileges**.
   * Enterprise environments relying on **Sysinternals for system diagnostics and malware analysis**.
   * Incident response teams and forensic analysts using **Procmon, Process Explorer, and Autoruns**.

### ****7-Zip Mark-of-the-Web (MotW) Bypass Vulnerability (CVE-2025-0411)****

7-Zip, a widely used open-source file archiver, has been found vulnerable to a **Mark-of-the-Web (MotW) bypass attack** due to improper handling of extracted files. **Tracked as CVE-2025-0411**, this flaw allows attackers to evade Windows security warnings, potentially leading to **arbitrary code execution**.

### ****Vulnerability Overview****

The vulnerability arises from 7-Zip's failure to properly retain the **MotW flag** on files extracted from specially crafted archives. **MotW** is a Windows security feature that marks files downloaded from untrusted sources (such as the internet), triggering security warnings before execution. When this protection is bypassed, users may unknowingly execute **malicious files** without security alerts.

#### **Attack Mechanics:**

1. The attacker creates a malicious **nested archive** that contains payloads carrying the MotW flag.
2. The victim extracts the archive using a vulnerable version of **7-Zip (versions prior to 24.09)**.
3. The extracted files **lose the MotW designation**, meaning Windows treats them as trusted files.
4. The attacker delivers the archive via **phishing emails, compromised websites, or file-sharing platforms**.
5. When the victim executes the extracted file, it runs **without triggering Windows security warnings**, leading to potential **arbitrary code execution**.

* **Attack Type:** Mark-of-the-Web (MotW) Bypass & Arbitrary Code Execution
* **Severity:** High (**CVSS Score: 7.0**)
* **CVE ID:** CVE-2025-0411

Sources: [**GitHub – 7-Zip CVE-2025-0411 PoC**](https://github.com/dhmosfunk/7-Zip-CVE-2025-0411-POC) , [**SOC Prime – CVE-2025-0411 Exploitation Targeting Ukraine**](https://socprime.com/blog/cve-2025-0411-exploitation-to-target-ukraine/) , [**Cyber Security News – PoC Exploit Released for 7-Zip Vulnerability**](https://cybersecuritynews.com/poc-exploit-released-for-7-zip-vulnerability/)

### ****Impact of the Vulnerability****

* **Windows Security Protections Bypassed**: Users do not receive the usual security prompts for untrusted files.
* **Increased Risk of Malware Execution**: Attackers can trick users into running malicious executables without any warning.
* **Potential for Arbitrary Code Execution**: If the extracted files contain malware, they can execute with the user’s privileges.
* **Enhanced Phishing Attacks**: Cybercriminals can deliver payloads via seemingly legitimate archive files.

**Affected Systems**

1. **Operating Systems:**
   * Windows 10
   * Windows 11
   * Windows Server (various versions)
2. **Vulnerable Software:**
   * **7-Zip (versions prior to 24.09)**.
3. **Usage Scenarios Prone to Exploitation:**
   * Users extracting untrusted archives from the internet.
   * Organizations using **7-Zip for automated file processing**.
   * Systems where **security policies rely on MotW warnings** to prevent malware execution.

### ****Critical Windows OLE Zero-Click Vulnerability (CVE-2025-21298)****

A critical vulnerability in **Windows Object Linking and Embedding (OLE)** has been identified, allowing attackers to execute **arbitrary code** without user interaction. **Tracked as CVE-2025-21298**, this **zero-click** vulnerability poses a significant risk, particularly in **phishing and malware delivery campaigns**.

### ****Vulnerability Overview****

The vulnerability is classified as a “Use After Free” issue (CWE-416), where memory that has already been freed is improperly accessed. This occurs during the conversion of data from an “OlePres” stream to a “CONTENTS” stream within OLE storage.

If an attacker crafts the RTF file to exploit this flaw, it can corrupt heap memory management structures, leading to arbitrary code execution.

#### **Attack Mechanics:**

1. The attacker crafts a **malicious document (e.g., Word, RTF, or Excel file)** embedded with an **OLE object**.
2. The victim **receives the document via email or messaging services**.
3. **No user interaction is required**—merely opening or previewing the document **triggers the exploit**.
4. The malicious OLE object executes arbitrary code, leading to **malware infection, data theft, or system takeover**.

* **Attack Type:** Zero-Click Arbitrary Code Execution
* **Severity:** Critical (**CVSS Score: 9.8**)
* **CVE ID:** CVE-2025-21298

**Source:** [**Cyber Security News**](https://cybersecuritynews.com/critical-windows-ole-zero-click-vulnerability/) , [**Hackers Arise – No-Click Vulnerability in Windows (CVE-2025-21298)**](https://www.hackers-arise.com/post/new-no-click-critical-vulnerability-in-microsoft-windows-cve-2025-21298) , [**GitHub – CVE-2025-21298 PoC**](https://github.com/ynwarcs/CVE-2025-21298)

### ****Impact of the Vulnerability****

* **Zero-Click Exploitation**: No user interaction is required, making it **highly dangerous** for **spear-phishing attacks**.
* **Remote Code Execution (RCE)**: Attackers can execute malicious payloads remotely, potentially leading to **full system compromise**.
* **Bypasses Security Protections**: Traditional **email security filters** may fail to detect malicious OLE objects.
* **Potential for Mass Exploitation**: Can be leveraged in **targeted APT campaigns** and **ransomware distribution**.

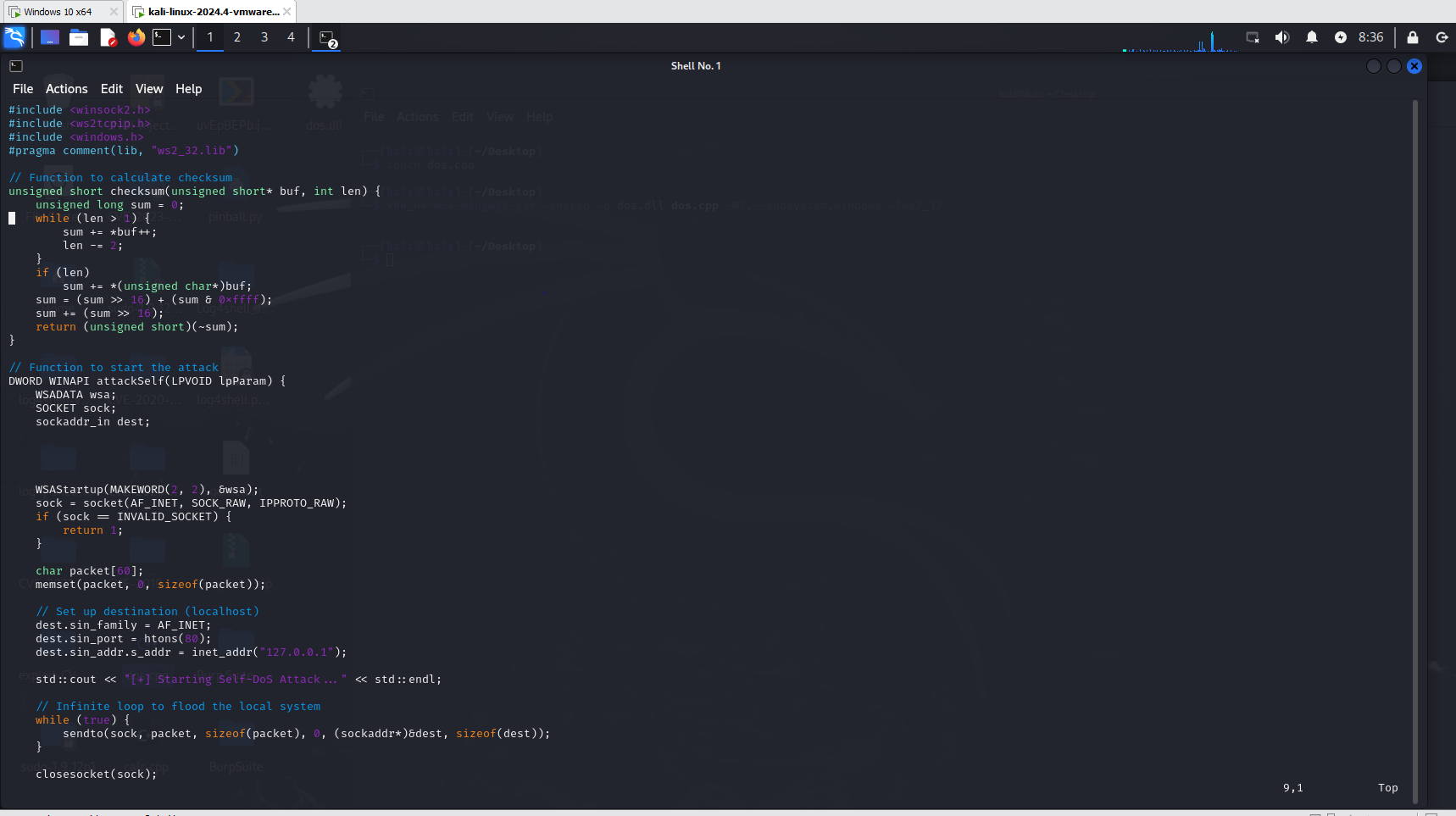
### ****Affected Systems****

1. **Operating Systems:**
   * Windows 10
   * Windows 11
   * Windows Server (various versions)
2. **Applications & Services at Risk:**
   * Microsoft Office Suite (**Word, Excel, PowerPoint**)
   * Rich Text Format (RTF) document processing
   * Outlook and email clients that **auto-preview** OLE-embedded content
3. **Usage Scenarios Prone to Exploitation:**
   * **Enterprise environments** where OLE is widely used for document sharing.
   * **Email users and organizations** relying on Office-based workflows.
   * **Spear-phishing targets** in financial, government, and healthcare sectors.

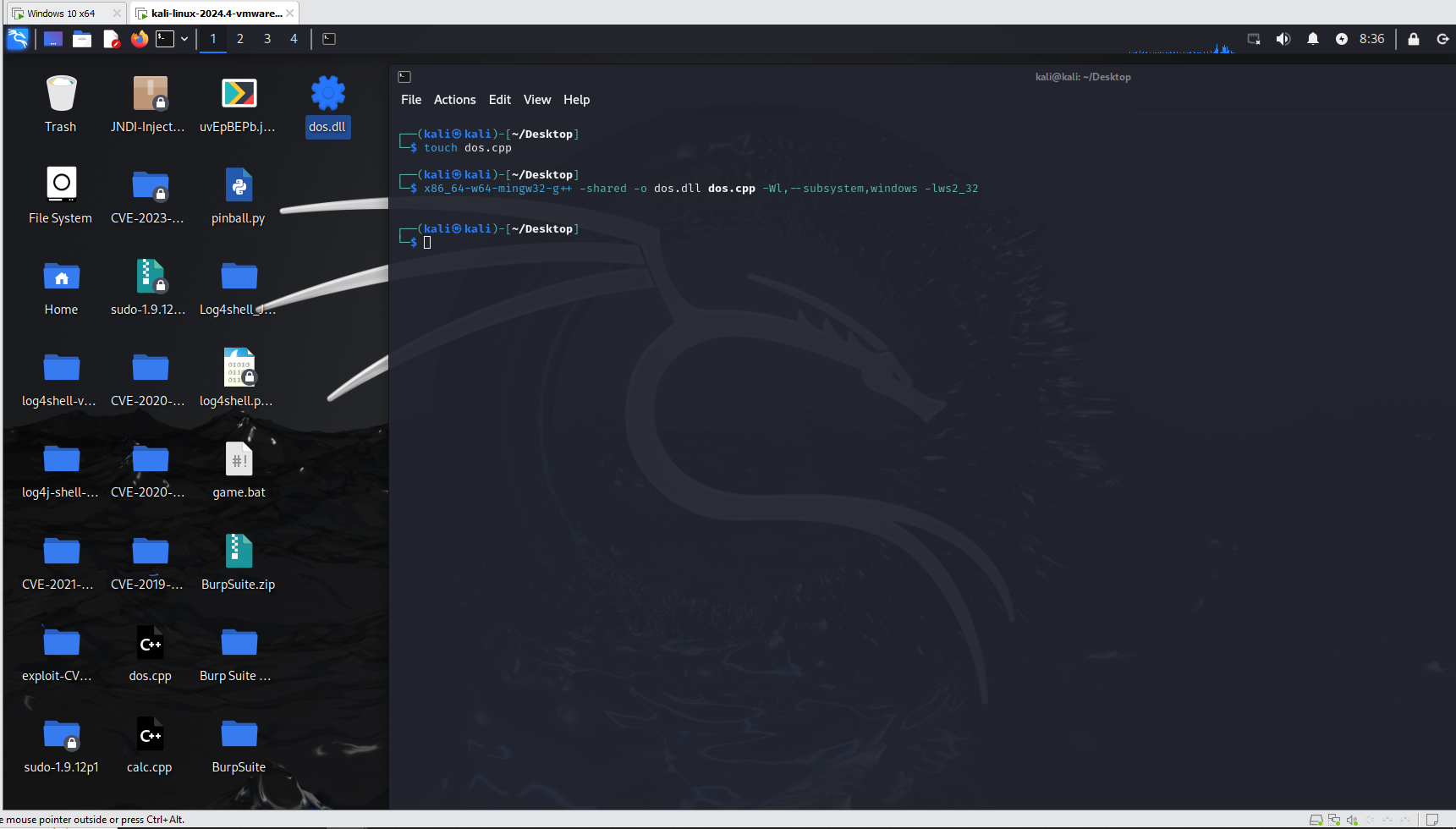
**Exploitation & Attack Techniques**

**Microsoft Sysinternals Tools DLL Injection Vulnerability:**

First, a **malicious DLL** is crafted to exploit the vulnerability. In this instance, the DLL is specifically designed to trigger a **Denial-of-Service (DoS) attack**, disrupting the target system's normal operation. The code continuously transmits **TCP SYN packets** to the **loopback address**, leading to a **Denial-of-Service (DoS) attack** by overwhelming the system with excessive connection requests.

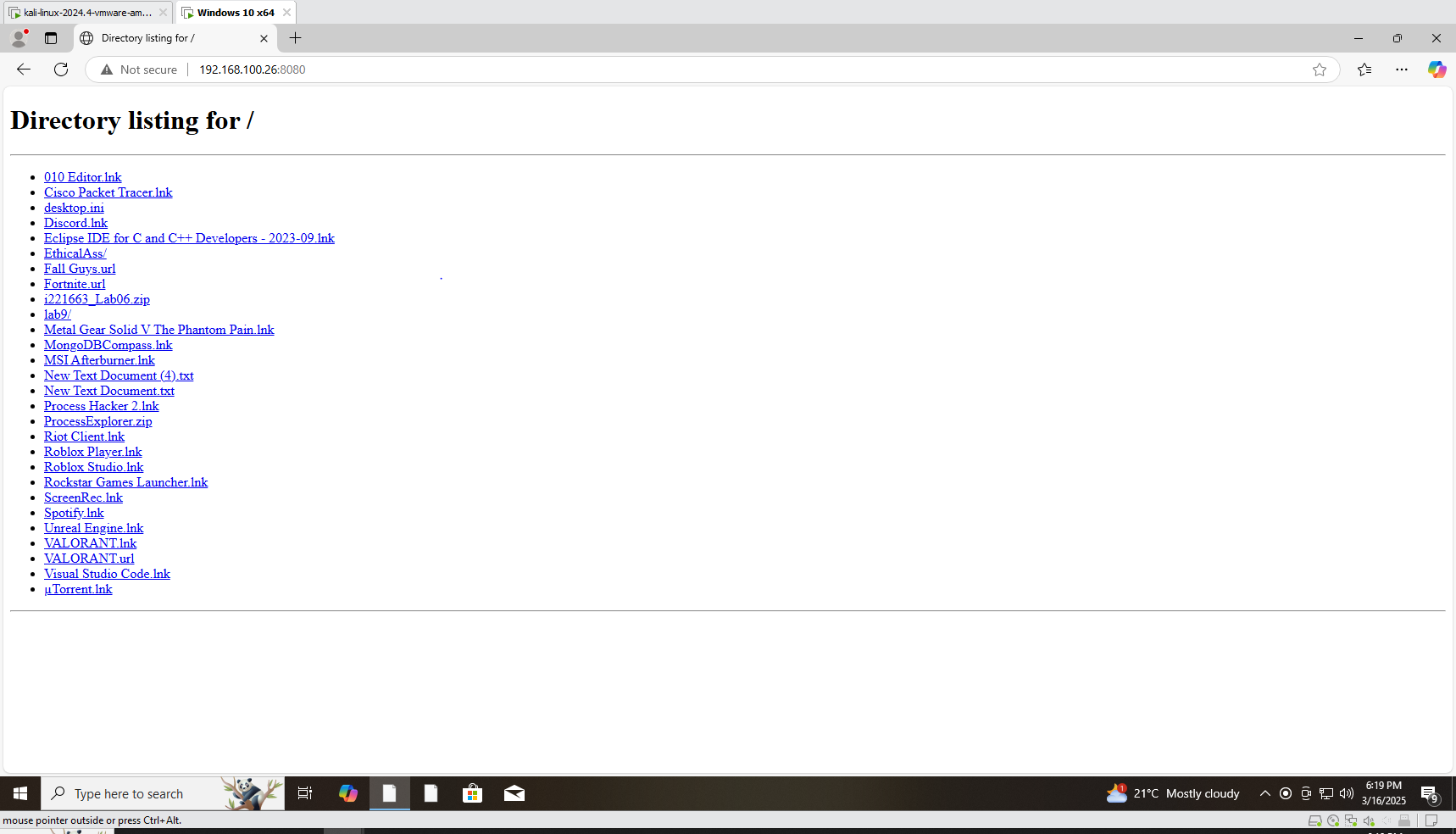


The C++ code is compiled into a **DLL file** using **MinGW**, ensuring compatibility with the **Windows operating system** for successful execution.

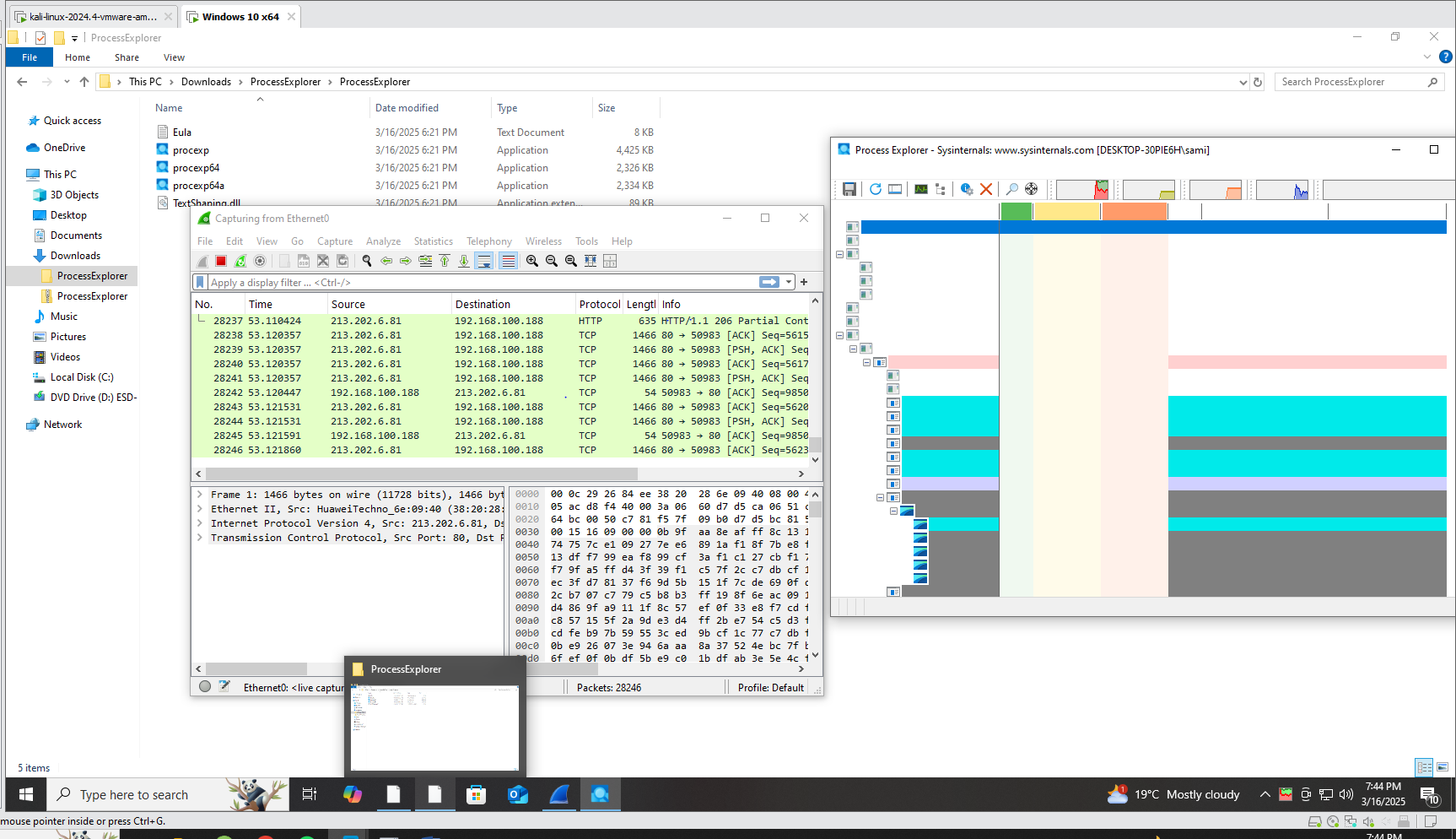


The **DLL file** is renamed to **"TextShaping.dll"** and transferred to the **host Windows machine**. Subsequently, both **TextShaping.dll** and the **Process Explorer tool** are compressed into a **ZIP archive**. A **Python HTTP server** is then launched in the directory containing the compressed file to facilitate its delivery.

The **HTTP server** is accessed from the **Windows 10 victim machine** and the victims **downloads the ZIP file** containing the malicious DLL and Process Explorer tool.



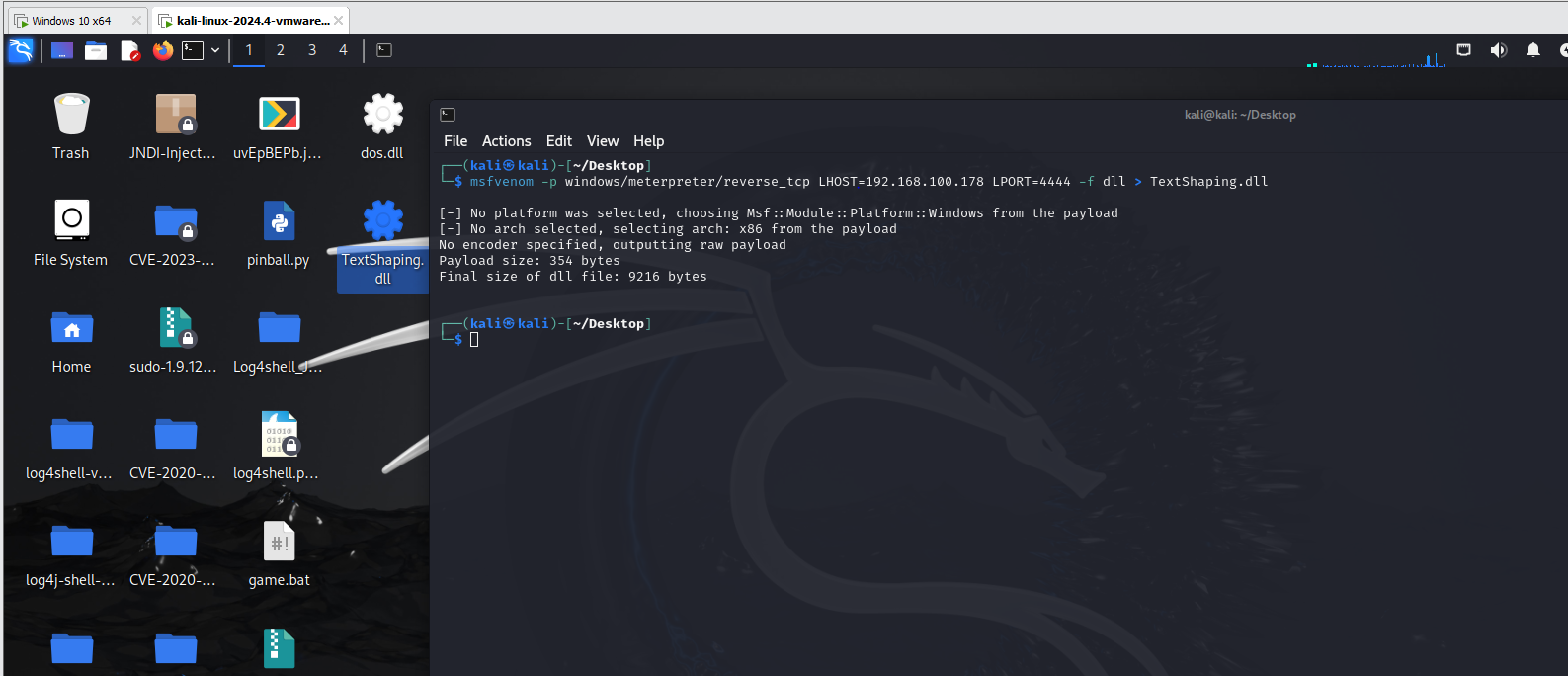
When the victim executes **procexp64.exe**, the application inadvertently loads the **malicious TextShaping.dll**, triggering a **Denial-of-Service (DoS) attack** on the system. This activity is visibly reflected in **Wireshark**, where the captured packets indicate abnormal network behavior consistent with the attack.



**Alternative exploitation method:**

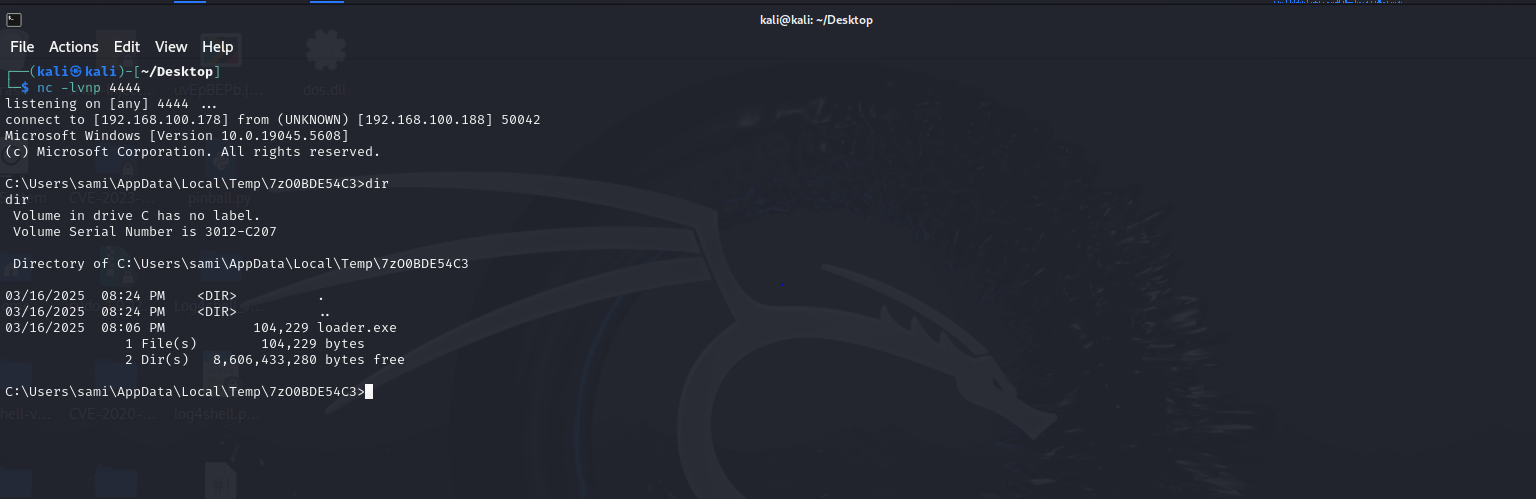
The same **DLL hijacking vulnerability** can be leveraged to achieve **full system compromise** using a **Meterpreter reverse shell**.

A **malicious DLL** is generated using **msfvenom**, configured to establish a connection from the **victim machine** back to the **attacker's IP** on **port 4444**.



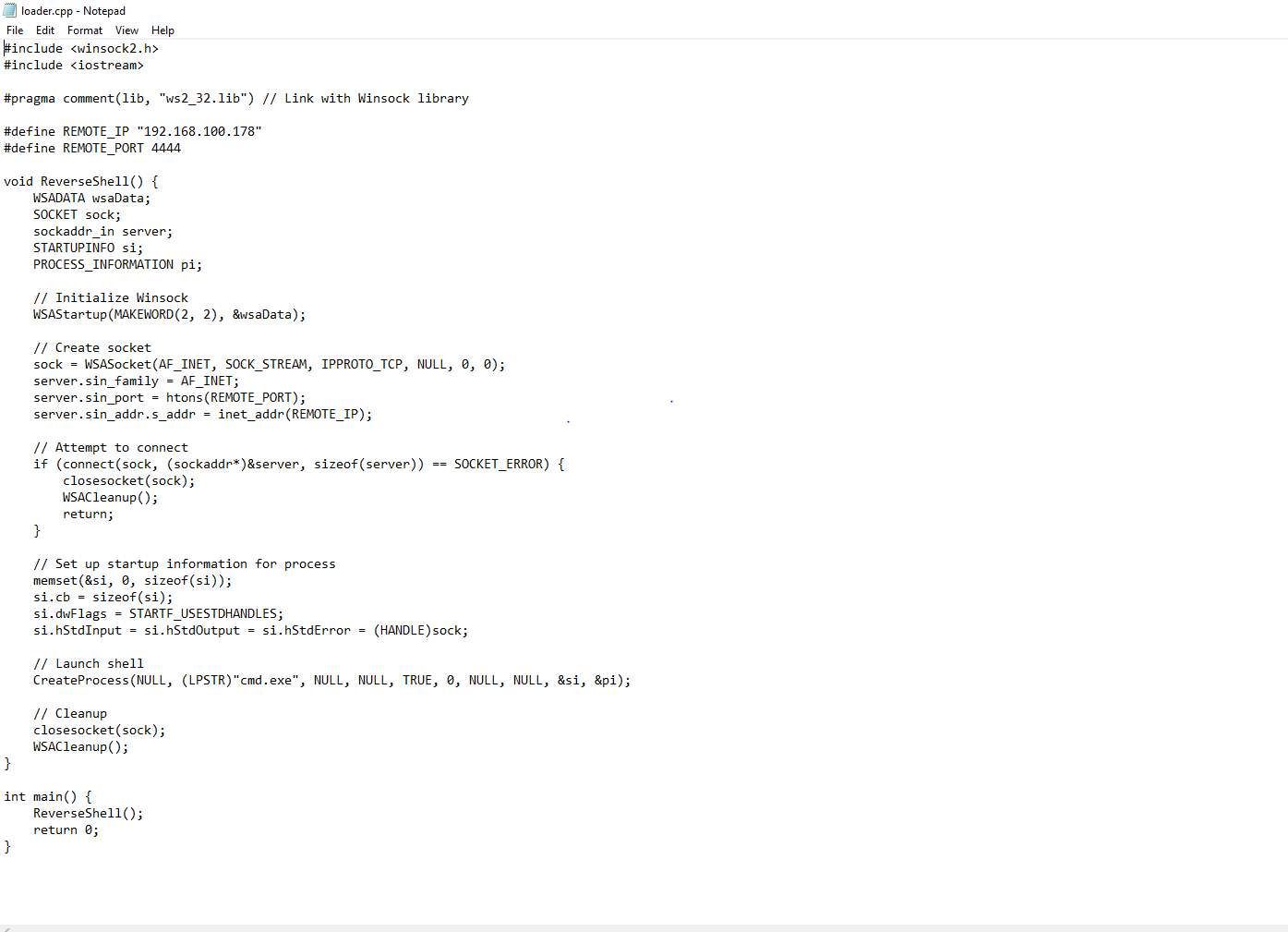
The **malicious DLL** is bundled with **Process Explorer**, compressed into a **ZIP file**, and hosted on a **Python HTTP server**, similar to the previous attack scenario.

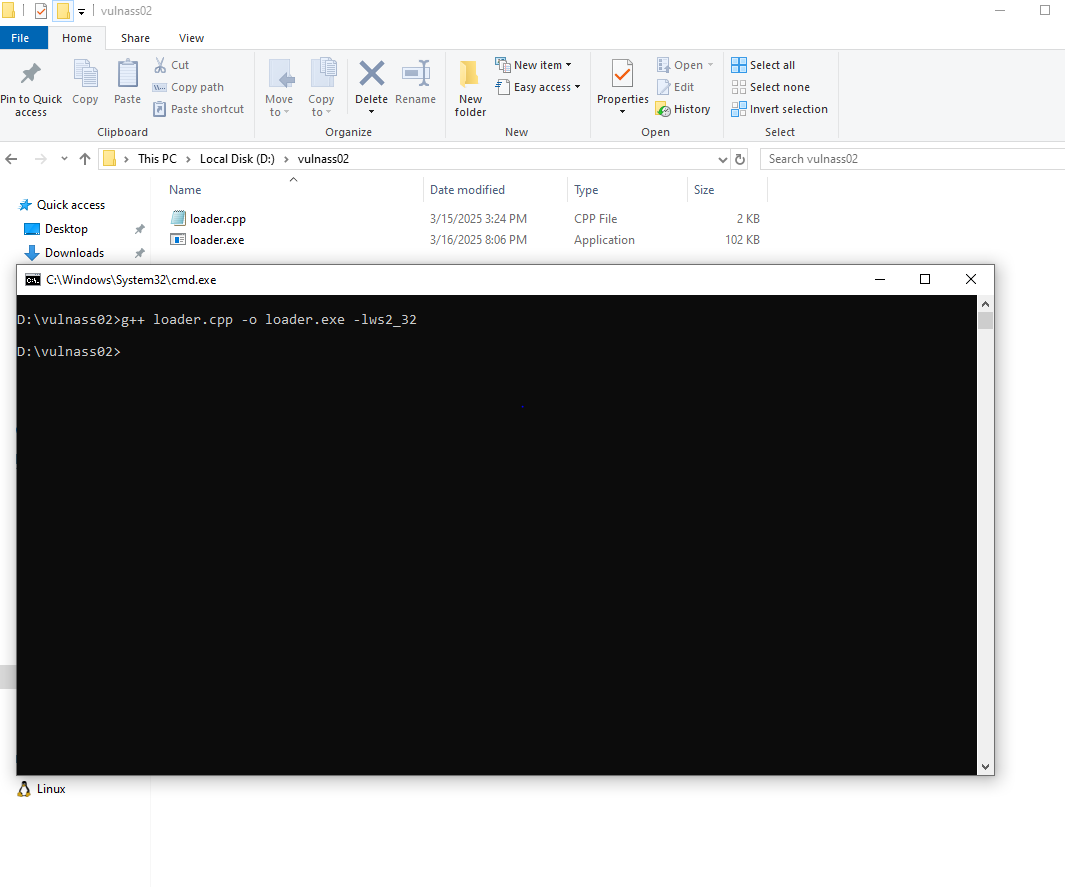
The **victim downloads** the ZIP file and **executes procexp64.exe**, which inadvertently loads the **malicious DLL**, triggering the **reverse shell**. Once executed, the attacker **gains full remote access** to the victim machine, allowing complete system control.



### ****7-Zip Mark-of-the-Web (MotW) Bypass Vulnerability (CVE-2025-0411)****

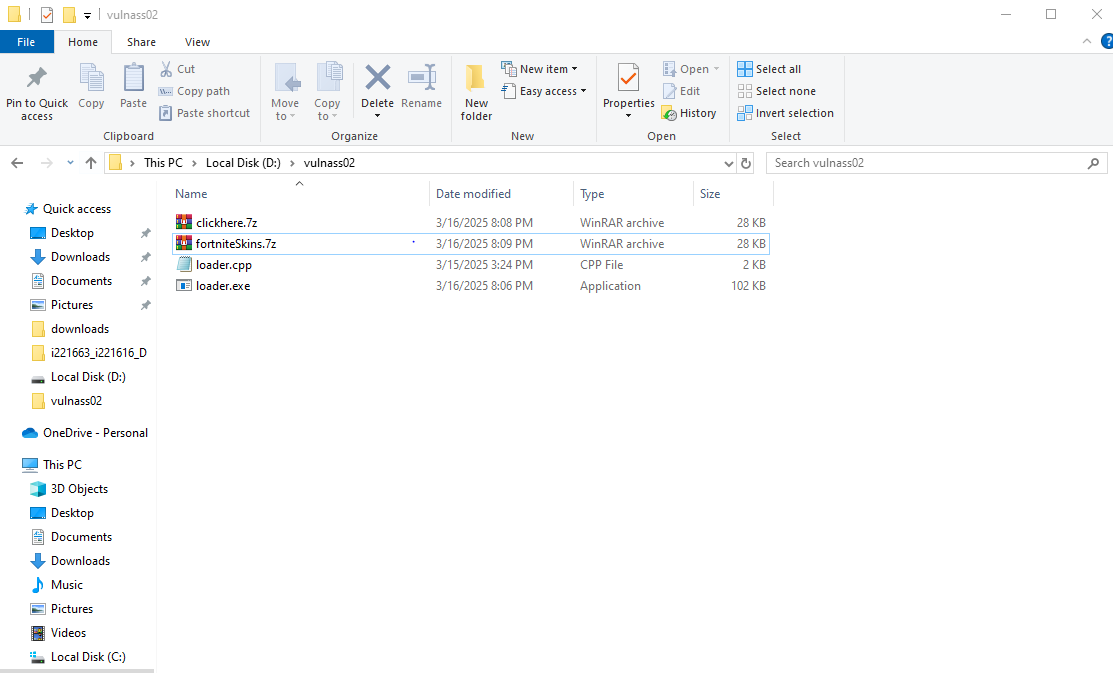
### The attacker **develops a reverse shell** in **C++** and compiles it into an **executable (.exe) file**, enabling remote access to the target system upon execution.





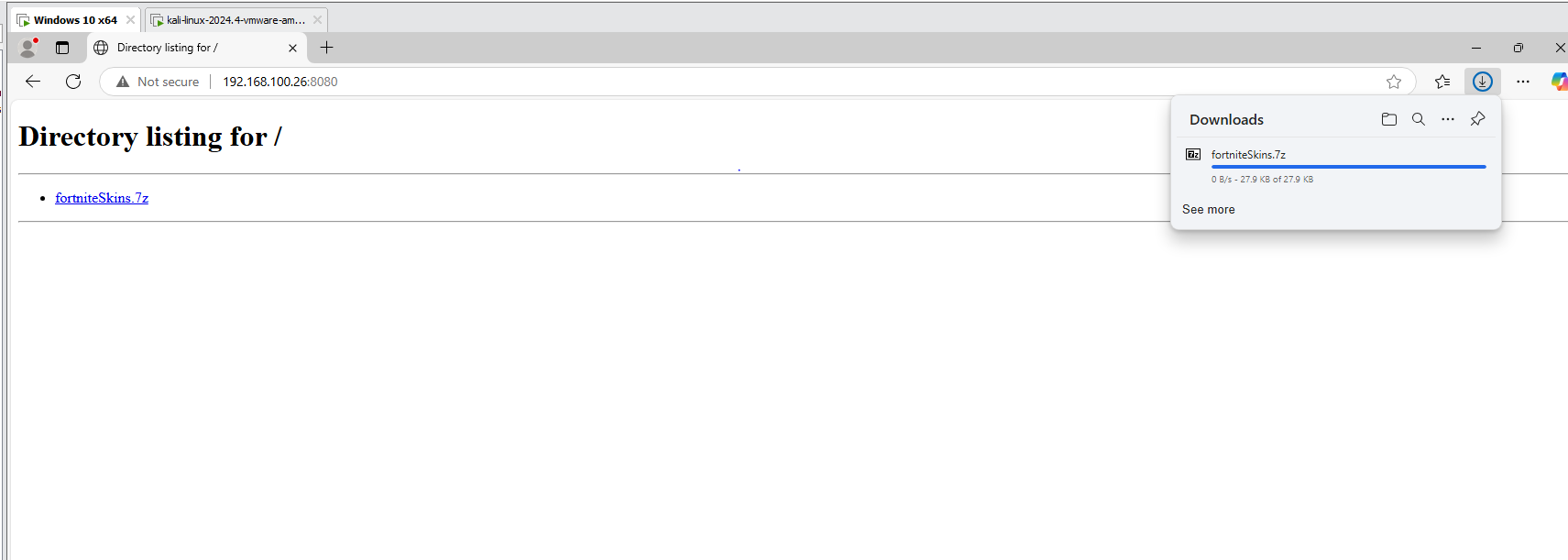
The **loader.exe** file is first compressed using the **vulnerable version 24.08 of 7-Zip**, creating an archive named **clickhere.7z**. This file is then **recompressed into another 7z archive**, with the final file named **fortniteskins.7z**.

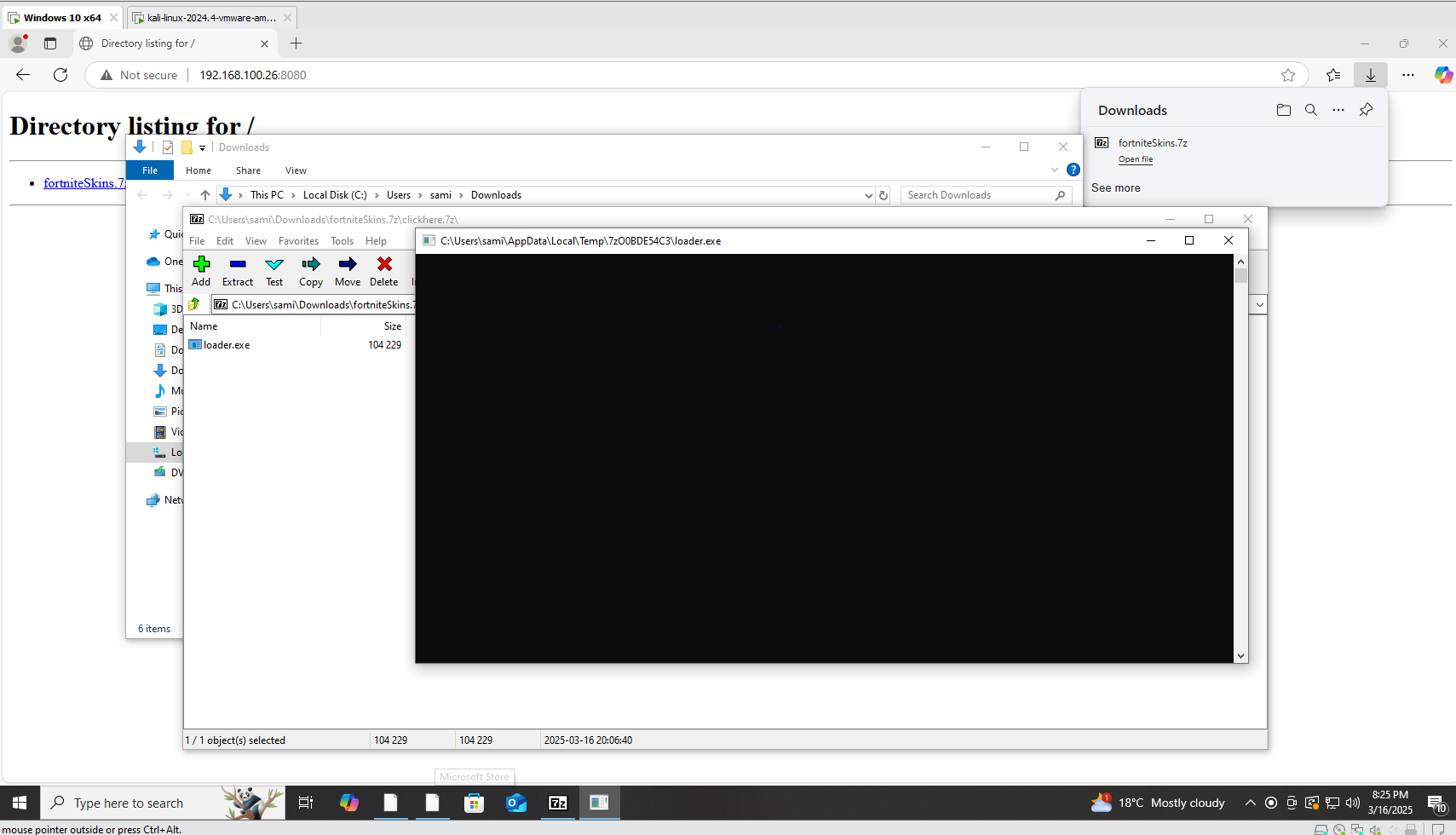
Only the **fortniteskins.7z** file is intended for exploitation, as it allows the attacker to bypass security mechanism mark of the web and deploy the **loader.exe** payload upon extraction.

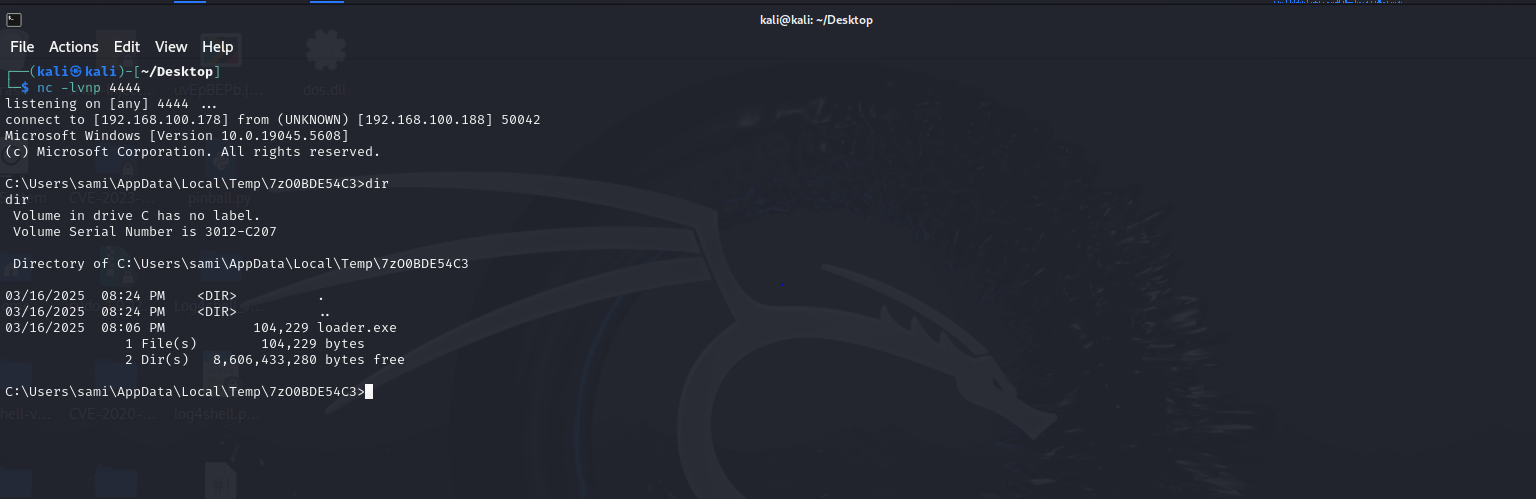


The **fortniteskins.7z** file is **hosted on a Python HTTP server**, allowing the victim to **download it.** Upon **double-clicking the .7z file**, the **loader.exe** executes, establishing a **reverse shell** and granting the attacker **complete control over the victim’s system**.

Notably, the **Mark-of-the-Web (MotW) security mechanism is bypassed**, meaning the browser does not flag the **.7z** file as potentially malicious. Additionally, **Windows Defender does not detect or block the attack**, allowing the exploitation to proceed undetected.



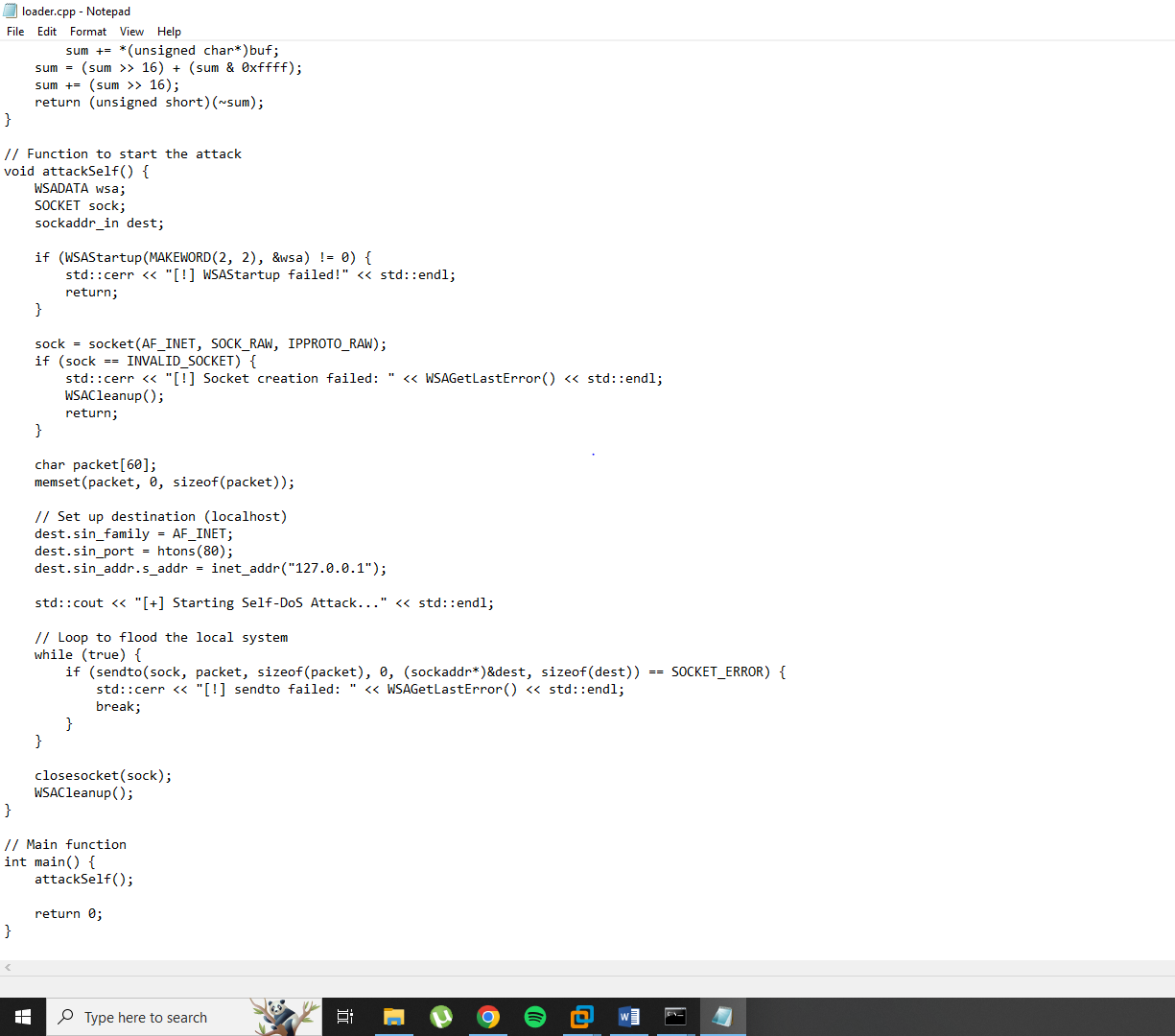




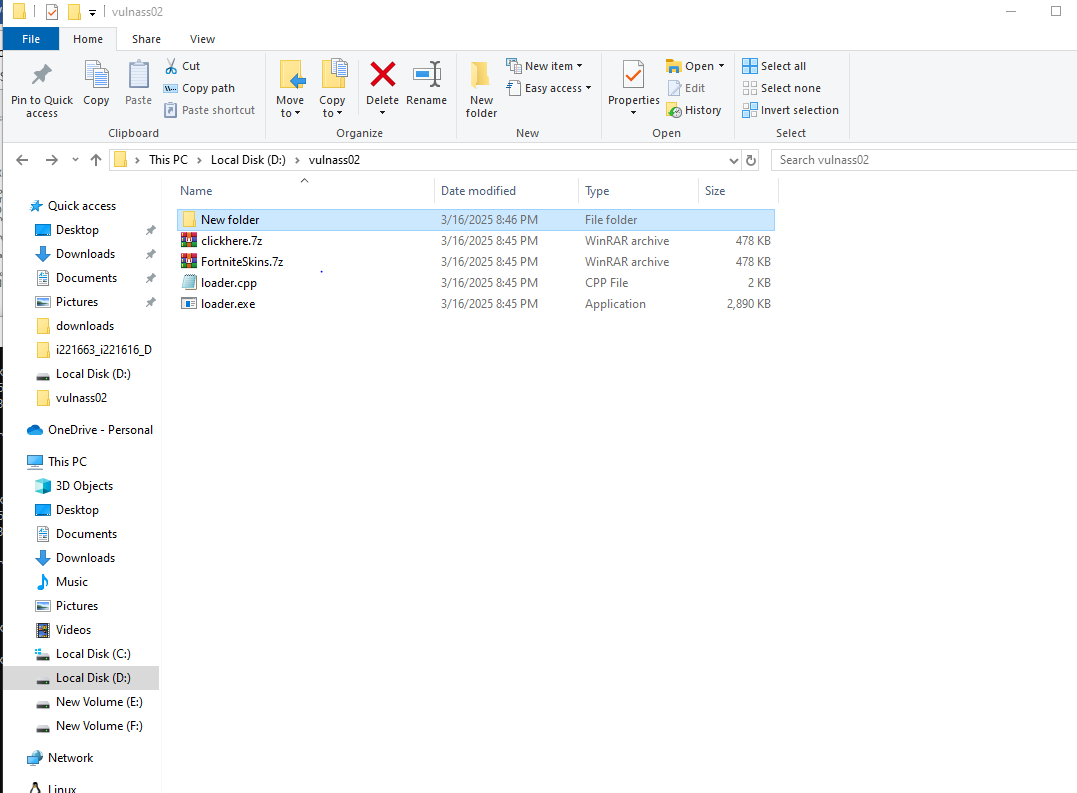
**Alternative exploitation method:**

Instead of achieving a reverse shell, the same **vulnerability** can be exploited to cause a **Denial of Service (DoS) attack** on the victim machine.

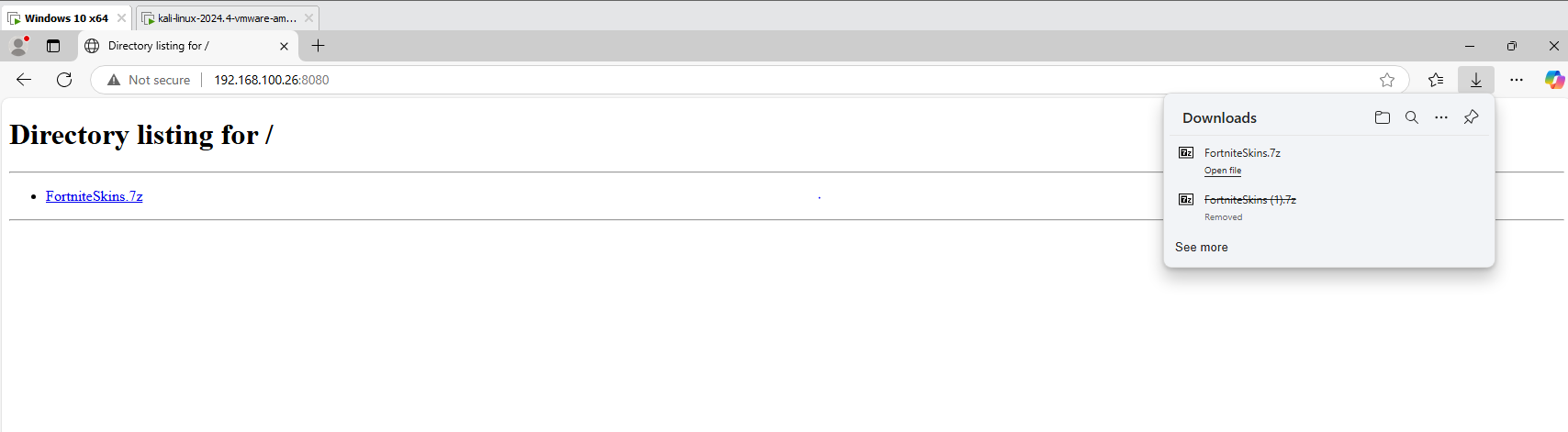
A **malicious executable (loader.exe)** is crafted in **C++** to continuously send **TCP SYN packets** to the **loopback address**, effectively overwhelming system resources and causing a **DoS condition**.

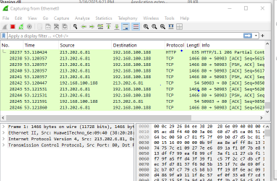


The **loader.exe** is compressed using the vulnerable **7-Zip version 24.08** and named **clickhere.7z**. To **bypass security mechanisms**, **clickhere.7z** is compressed again into a final archive named **fortniteskins.7z**.



The **fortniteskins.7z** file is **hosted on a Python HTTP server**, making it easily accessible to the victim. The victim downloads and extracts the **.7z archive**, revealing the **loader.exe file**. Upon **execution of loader.exe**, the system starts flooding itself with TCP SYN packets, **rendering it unresponsive** and effectively triggering a **Denial of Service (DoS) attack**.



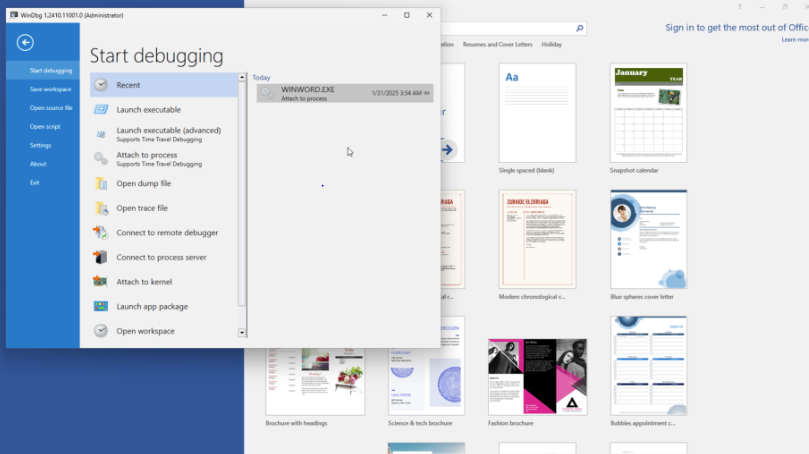


As with the **reverse shell exploitation**, **Mark-of-the-Web (MotW) is bypassed**, and **Windows Defender does not detect or block the attack**.

### ****Critical Windows OLE Zero-Click Vulnerability (CVE-2025-21298)****

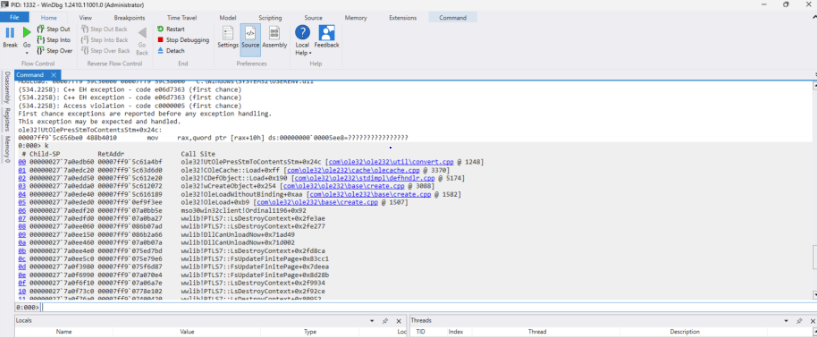
This proof of concept (PoC) demonstrates how an **RTF (Rich Text Format) file** can be used to exploit the vulnerability, causing **Microsoft Word to crash** due to **memory corruption**.

The windbg is attached to the running WINWORD.EXE process

.

The proof of concept (PoC) for **CVE-2025-21298** has been sourced from [GitHub – CVE-2025-21298 PoC](https://github.com/ynwarcs/CVE-2025-21298).

As soon as the **malicious RTF file** is opened in Microsoft Word, it triggers the **memory corruption vulnerability**, causing Word to **immediately crash**. This crash demonstrates the exploitability of the vulnerability and serves as a potential **attack vector for remote code execution (RCE) or denial of service (DoS)**.



**Alternative exploitation method:**

The same **CVE-2025-21298** vulnerability can be exploited to gain full remote access using a **Meterpreter reverse shell**.

The attacker creates a malicious **RTF document** embedded with a **reverse shell payload**. The payload is designed to execute upon opening or previewing the document in **Microsoft Word**.

The attacker sends the **malicious RTF file** via **phishing email** or hosts it via python http server. The victim downloads or previews the file in Microsoft Word, **unknowingly triggering the exploit** and A connection is established from the victim’s machine to the attacker’s **listening Metasploit handler**.

**Security Analysis & Mitigation Strategies**

**Microsoft Sysinternals Tools DLL Injection Vulnerability**

#### **1. Real-World Exploitation of the Vulnerability**

Attackers can exploit vulnerabilities in **Microsoft Sysinternals Tools** to execute **DLL injection attacks**, allowing them to run malicious code with elevated privileges. The key attack vectors include:

* **Phishing Attacks** – Attackers can deliver malicious DLLs alongside legitimate Sysinternals tools, tricking users into running compromised executables.
* **Supply Chain Attacks** – If an organization downloads Sysinternals tools from untrusted sources, attackers can replace them with trojanized versions.
* **Living-Off-The-Land (LOTL) Techniques** – Since Sysinternals tools are widely used for system diagnostics, attackers can exploit them without triggering security alerts.

#### **2. Security Defenses & Mitigation Strategies**

##### **A. System Hardening**

* **Implement Application Control Policies** – Use **Windows Defender Application Control (WDAC)** or **AppLocker** to restrict the execution of unauthorized DLLs.
* **Enforce DLL Signature Validation** – Ensure that only signed and verified DLLs are loaded into Sysinternals tools.

##### **B. Monitoring & Intrusion Detection**

* **Enable Event Logging** – Monitor **Windows Event Logs** for suspicious DLL loading events (Event ID 4688 for process creation).
* **Deploy Endpoint Detection & Response (EDR)** – Use tools like **Microsoft Defender for Endpoint** to detect anomalous DLL injection attempts.
* **Monitor Sysinternals Tool Usage** – Track execution of Process Explorer, Autoruns, and other tools for unusual activity.

##### **C. Patching & Secure Configuration**

* **Keep Sysinternals Tools Updated** – Regularly download verified updates from the **official Microsoft website**.
* **Apply Patch Management Best Practices** – Ensure Windows systems are updated to mitigate any zero-day vulnerabilities.

##### **D. Zero-Trust Architecture Implementation**

* **Limit Privilege Escalation** – Enforce **least privilege access control (LPA)** to prevent execution of malicious DLLs.
* **Network Segmentation** – Restrict execution of administrative tools to secure environments to prevent lateral movement.

#### **3. Practical Recommendations for Prevention**

1. **Use Application Sandboxing** – Run Sysinternals tools in a controlled environment to prevent unauthorized DLL execution.
2. **Disable Unnecessary Sysinternals Tools** – If not needed, restrict access to tools that may be exploited for DLL injection.
3. **Avoid Running Tools from Network Locations**: Always copy Sysinternals executables to local paths before execution.
4. **Verify DLL Integrity**: Employ security solutions to load only trusted DLLs.
5. **Audit Your Environment**: Use the provided test sheet to identify tools vulnerable to DLL injection and take the necessary safeguards.

### ****7-Zip Mark-of-the-Web (MotW) Bypass Vulnerability (CVE-2025-0411)****

#### **1. Real-World Exploitation of the Vulnerability**

The **7-Zip Mark-of-the-Web (MotW) Bypass vulnerability** allows attackers to distribute malicious files that evade Windows security warnings and protections. Attackers can exploit this flaw using:

* **Phishing & Social Engineering** – Malicious **7z archives** are sent via phishing emails, tricking users into extracting and executing harmful payloads.
* **Compromised Websites & Drive-By Downloads** – Attackers host **nested 7z files** on compromised websites, leading to silent malware execution upon extraction.
* **Supply Chain Attacks** – Trojanized software distributed via **trusted but compromised** file-sharing platforms can deliver unflagged malicious files.

Once extracted, these files **lose the MotW flag**, meaning they **bypass SmartScreen, Defender, and Attack Surface Reduction (ASR) rules**, enabling **execution of arbitrary code** without security prompts.

#### **2. Security Defenses & Mitigation Strategies**

##### **A. System Hardening**

* **Restrict 7-Zip Usage for Untrusted Files** – Configure **Windows Group Policies** to prevent execution of unknown **7z archives**.
* **Enforce Alternative Extraction Methods** – Use Windows-native tools like **PowerShell’s Expand-Archive**, which preserves the MotW flag.

##### **B. Monitoring & Intrusion Detection**

* **Enable Security Logging** – Monitor **file extractions and executions** from untrusted sources using **Windows Defender ASR rules**.
* **Deploy Endpoint Protection (EDR)** – Advanced **EDR solutions** can detect anomalous activity related to MotW bypass attempts.
* **Monitor Suspicious PowerShell & CMD Activity** – Attackers often execute payloads using powershell.exe or cmd.exe after extraction.

##### **C. Patching & Secure Configuration**

* **Keep 7-Zip Updated** – Ensure all systems are running **patched versions of 7-Zip** that fix MotW handling flaws.
* **Enforce Windows Defender SmartScreen** – Configure **Defender policies** to prevent execution of files from untrusted origins.

##### **D. Zero-Trust Architecture Implementation**

* **Implement Application Control** – Use **AppLocker** or **Microsoft Defender Application Control (MDAC)** to **restrict execution of unknown executables**.
* **Isolate Untrusted Downloads** – Run downloaded archives in a **sandboxed environment** or **Windows Defender Application Guard (WDAG)** before extraction.

#### **3. Practical Recommendations for Prevention**

1. Ensure all systems are running 7-Zip version 24.09 or later, which addresses this vulnerability.
2. Deploy robust email filtering solutions to block spear-phishing attempts.
3. Educate employees on recognizing [phishing emails](https://cybersecuritynews.com/phishing-attack/) and homoglyph attacks.
4. Disable automatic execution of files from untrusted sources and enforce verification prompts.
5. Use endpoint protection tools capable of identifying and blocking malicious file activity.
6. Look for unusual patterns indicative of malware communication with C2 servers.

### ****Critical Windows OLE Zero-Click Vulnerability (CVE-2025-21298)****

#### **1. Real-World Exploitation of the Vulnerability**

The **Critical Windows OLE Zero-Click Vulnerability (CVE-2025-21298)** allows attackers to execute arbitrary code remotely without user interaction. This exploit leverages **maliciously crafted RTF (Rich Text Format) documents** that abuse **Windows OLE (Object Linking and Embedding)** functionality.

##### **Attack Scenarios**:

* **Email-Based Exploitation (Zero-Click)**
  + A malicious RTF document is **sent via email**. If the email client **previews** the file, the exploit triggers automatically without requiring the victim to open the document.
* **Weaponized Documents in Phishing Campaigns**
  + Attackers distribute RTF documents embedded with **malicious OLE objects**. When the victim **opens the file in Microsoft Word**, it automatically executes the payload.
* **Web-Based Delivery (Drive-By Download)**
  + Malicious RTF files are **hosted on compromised websites**. If a victim **downloads and previews** the file, the exploit triggers.
* **Supply Chain Attacks**
  + Attackers **trojanize business documents** used in financial or legal communications, leading to widespread infection within enterprises.

Once triggered, the vulnerability allows **remote code execution (RCE)**, granting attackers full control over the compromised system.

#### **2. Security Defenses & Mitigation Strategies**

##### **A. System Hardening**

* **Disable RTF File Previews in Email Clients**
  + Configure **Microsoft Outlook and other email clients** to block **RTF file previews**.
* **Restrict OLE Objects in Microsoft Word**
  + Modify **Microsoft Office Group Policies** to prevent **automatic execution of OLE objects** in documents.
* **Use Office Protected View & File Block Settings**
  + Enforce **Microsoft Office Protected View** for all downloaded and email attachments.
  + Configure **File Block Settings** to **disable processing of RTF documents**.

##### **B. Monitoring & Intrusion Detection**

* **Deploy Endpoint Security Solutions**
  + Use **EDR solutions** that can detect **suspicious process execution from Microsoft Word (winword.exe)**.
* **Monitor for Abnormal OLE Activity**
  + Enable **Windows Event Logging** to detect **suspicious OLE object execution** in Word and other Office applications.
* **Detect Malicious Network Connections**
  + Monitor for **unusual outbound connections** from Office applications, which may indicate **command-and-control (C2) activity**.

##### **C. Patching & Secure Configuration**

* **Apply Microsoft Security Updates Immediately**
  + Ensure all systems are updated with the latest **Microsoft security patches** that address **CVE-2025-21298**.
* **Enforce Macro and Script Restrictions**
  + Disable **macros and ActiveX controls** for all **untrusted Office documents**.

##### **D. Zero-Trust Architecture Implementation**

* **Implement Application Control**
  + Use **AppLocker or Microsoft Defender Application Control (MDAC)** to **prevent unauthorized execution of RTF files**.
* **Segment Network Access**
  + Restrict Office applications from **accessing external networks** unless explicitly required.

#### **3. Practical Recommendations for Prevention**

**Apply Security Updates**

* Microsoft addressed this vulnerability in its **January 2025 Patch Tuesday** updates.
* Users and organizations are urged to **apply these patches immediately** to prevent exploitation.

**Configure Outlook for Plain Text Emails** *(Temporary Workaround)*

* If immediate patching is not possible, **configure Outlook to read all emails in plain text format**.
* This reduces the risk of **automatic execution of malicious OLE objects**.
* **Note:** This setting may affect email readability by **removing rich content**, such as images and animations.

**General Security Best Practices**

* **Avoid opening RTF files from untrusted sources** to reduce the risk of infection.
* **Implement least privilege principles** to minimize potential damage from successful exploits.
* **Use Sigma rules or other detection mechanisms** to monitor suspicious interactions with **.rtf files or OLE-related processes**.

**Detection & Monitoring Strategies**

* **Leverage security tools like Microsoft Defender** and third-party solutions to detect exploitation attempts.
* **Use Sigma rules** to identify systems interacting with high-risk file types, such as **.rtf or .dll files**.
* **Utilize debugging tools like WinDbg** to trace memory operations during file processing and confirm exploitation attempts.

**Critical Reflection & Additional Research**

**Challenges Faced & Solutions**

* **Ensuring Reliable Exploitation:**  
  One of the primary challenges was ensuring consistent exploitation across different system configurations. Windows security updates, varying system policies, and third-party security solutions often interfered with payload execution.
  + Solution: Multiple test environments were created using different Windows versions and security settings to identify the most effective exploitation methods.
* **Bypassing Modern Security Mechanisms:**  
  Security measures like Microsoft Defender, AMSI (Antimalware Scan Interface), Windows SmartScreen, and UAC (User Account Control) made it difficult to execute malicious payloads without detection.
  + Solution: Research into alternative execution techniques, such as DLL hijacking, Mark-of-the-Web (MotW) bypass, and RTF file manipulation, allowed for stealthier attack chains. Additionally, obfuscation techniques were tested to reduce detection rates.
* **Crafting a Reliable RTF-Based Exploit:**  
  The Critical Windows OLE Zero-Click Vulnerability (CVE-2025-21298) required crafting a malicious RTF file that could execute arbitrary code upon opening or previewing in Microsoft Word.
  + Solution: Studying past OLE exploits and analyzing Microsoft’s RTF parsing behavior helped in designing a working proof of concept that triggers execution with minimal user interaction.
* **Understanding Sysinternals 0-Day Vulnerability for DLL Injection:**  
  The 0-day vulnerability in Microsoft Sysinternals Tools required an in-depth understanding of how Windows loads DLLs and how attackers can hijack legitimate processes to execute malicious code.
  + Solution: Researching past DLL injection attacks and testing different DLL payloads helped refine the attack methodology for real-world exploitation.

**Additional Research & Sources**

* **Security Research & Blogs:**
  + **Censys Research on CVE-2025-21298** *(Censys, 2025)*
    - [CVE-2025-21298 Research](https://censys.com/cve-2025-21298/)
    - This research provided insights into how attackers exploit the OLE zero-click vulnerability and highlighted the risks associated with Microsoft Word automatically processing malicious RTF files.
  + **CyberConvoy Blog on Microsoft Sysinternals 0-Day DLL Injection** *(CyberConvoy, 2025)*
    - [Microsoft Sysinternals 0-Day Flaws](https://cyberconvoy.com/blog/microsoft-sysinternals-0-day-flaws-enable-dll-injection-attacks-on-windows/)
    - This blog detailed real-world scenarios of DLL injection attacks, explaining how attackers leverage vulnerable Sysinternals tools to execute arbitrary code.
  + **Trend Micro Analysis of CVE-2025-0411 Targeting Ukrainian Organizations** *(Trend Micro, 2025)*
    - [CVE-2025-0411 Targeted Attacks](https://www.trendmicro.com/en_us/research/25/a/cve-2025-0411-ukrainian-organizations-targeted.html)
    - This article provided an in-depth analysis of how attackers exploit the Mark-of-the-Web (MotW) bypass in 7-Zip to deliver malware through phishing emails and malicious downloads.

**Impact of Research on Exploitation & Mitigation**

* **Exploitation Strategy Refinement:**
  + The Censys research helped in modifying the RTF exploit to ensure that it could be triggered reliably in a real-world scenario.
  + The CyberConvoy blog clarified DLL search order hijacking, improving the Sysinternals DLL injection method.
  + The Trend Micro report emphasized the importance of MotW bypass, reinforcing why attackers favor nested archives to avoid detection.
* **Better Mitigation Approaches:**
  + Understanding 7-Zip’s handling of MotW reinforced the need for endpoint detection rules that analyze extracted files for missing security flags.
  + Research on RTF parsing vulnerabilities highlighted why disabling OLE-based execution in Microsoft Office is crucial.
  + Implementing Sysinternals tool integrity monitoring was identified as a key strategy to detect unauthorized DLL injections.