AI-ASSISTED SOC TRIAGE LAB

SSH Brute-Force Detection, MITRE ATT&CK Mapping, and AI-Powered Triage

Python • Splunk • MITRE ATT&CK® • Kali Linux • Claude AI

1. Overview

This project is a hands-on simulation of how a Security Operations Center (SOC) analyst detects, investigates, and responds to a potential SSH brute-force attack. The goal is to recreate a realistic Tier 1 analyst workflow, combining log analysis, detection logic, automated AI-driven triage, and adversary behavior mapping using the MITRE ATT&CK framework.

What This Project Demonstrates

- **Environment Setup:** Build a structured project workspace in Kali Linux
- **Secure Development:** Store API keys safely using .env; prevent credential leaks with .gitignore
- **Threat Simulation:** Generate SSH brute-force activity and work with realistic log data
- **Detection:** Use Splunk to identify suspicious login patterns with SPL queries
- **MITRE ATT&CK Mapping:** Classify adversary behavior using the industry-standard framework
- **Triage & Analysis:** Use Python + Claude AI to classify threat type, severity, and risk level
- **Response Planning:** Recommend immediate and follow-up actions with escalation conditions
- **Incident Remediation:** Handle a real-world API key exposure and remediate using Git best practices

2. Project Setup & Directory Structure

The project begins by cloning the repository into a Kali Linux environment and organizing the workspace to reflect a real-world SOC investigation setup.

Directory / File	Purpose
src/	Core Python triage logic and automation scripts

Directory / File	Purpose
docs/	Documentation and analyst notes
screenshots/	Proof of work and evidence capture
sample_logs/	Log data for analysis and testing
splunk_queries/	SPL detection logic and MITRE-labeled queries
reports/	Final findings and investigation outputs
requirements.txt	Tracks Python package dependencies
.env.example	Template for environment variables — no secrets stored
.gitignore	Ensures sensitive data is never committed to VCS

Commands Used

```
git clone https://github.com/mandozone/claude-splunk-soc-assistant.git
cd claude-splunk-soc-assistant
mkdir src docs screenshots sample_logs splunk_queries reports
touch requirements.txt .env.example .gitignore
```

3. Environment Setup & Dependency Installation

A Python virtual environment was created and activated to isolate dependencies from the host system. Three core libraries were installed: requests for HTTP interactions, python-dotenv for secure environment variable management, and anthropic for AI-driven alert analysis.

Screenshot Evidence — pip install

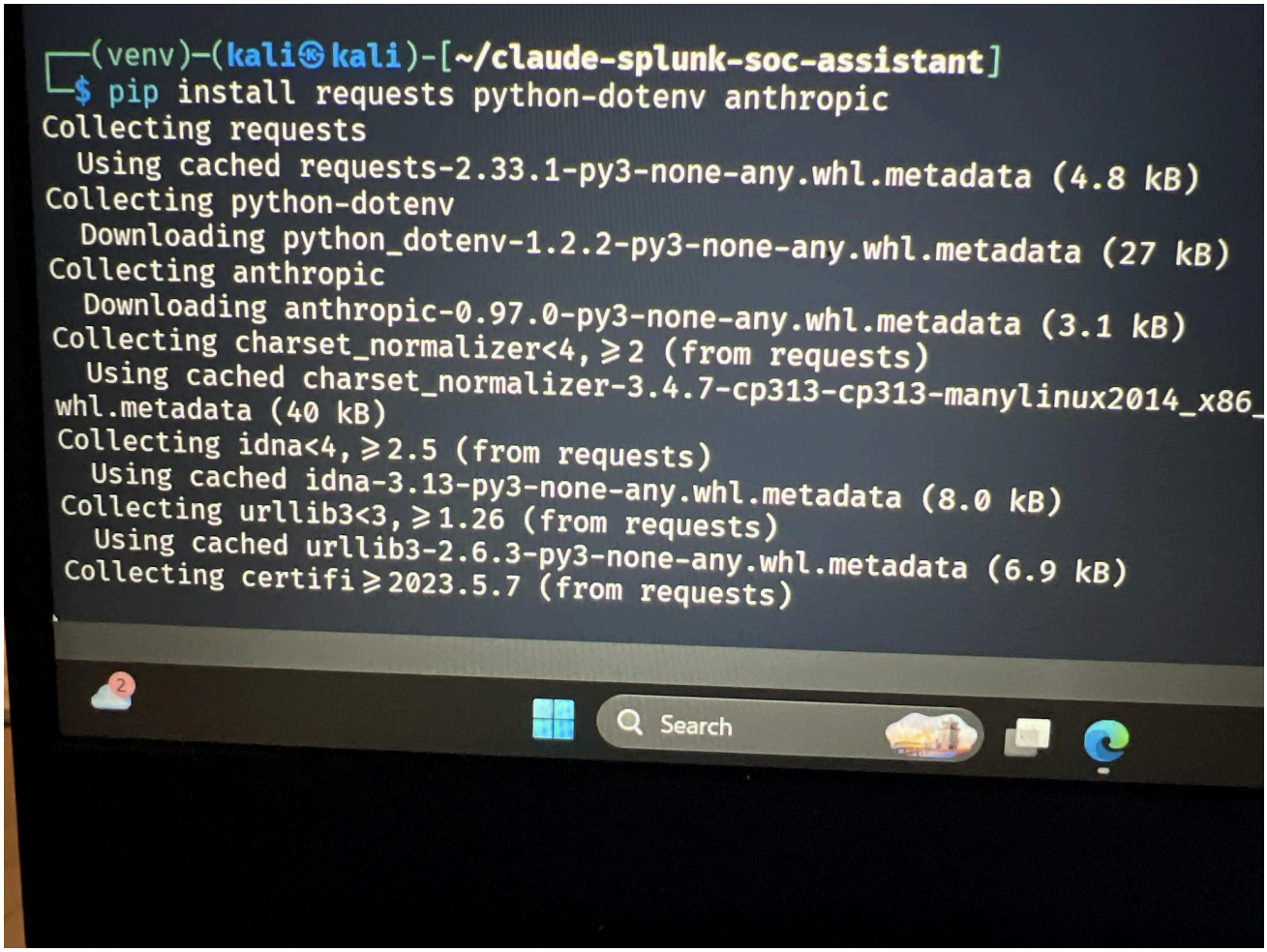

Figure 1 — pip install confirming anthropic 0.97.0, python-dotenv 1.2.2, and requests 2.33.1 installed successfully in the virtual environment

```
pip install requests python-dotenv anthropic


# Confirmed: anthropic-0.97.0  python_dotenv-1.2.2  requests-2.33.1
```

4. Repository Initialization & Secure Practices

With the environment prepared, the repository was initialized and organized. Version control hygiene is treated as a security control, not just a development convenience.

Screenshot Evidence — git clone & Directory Setup

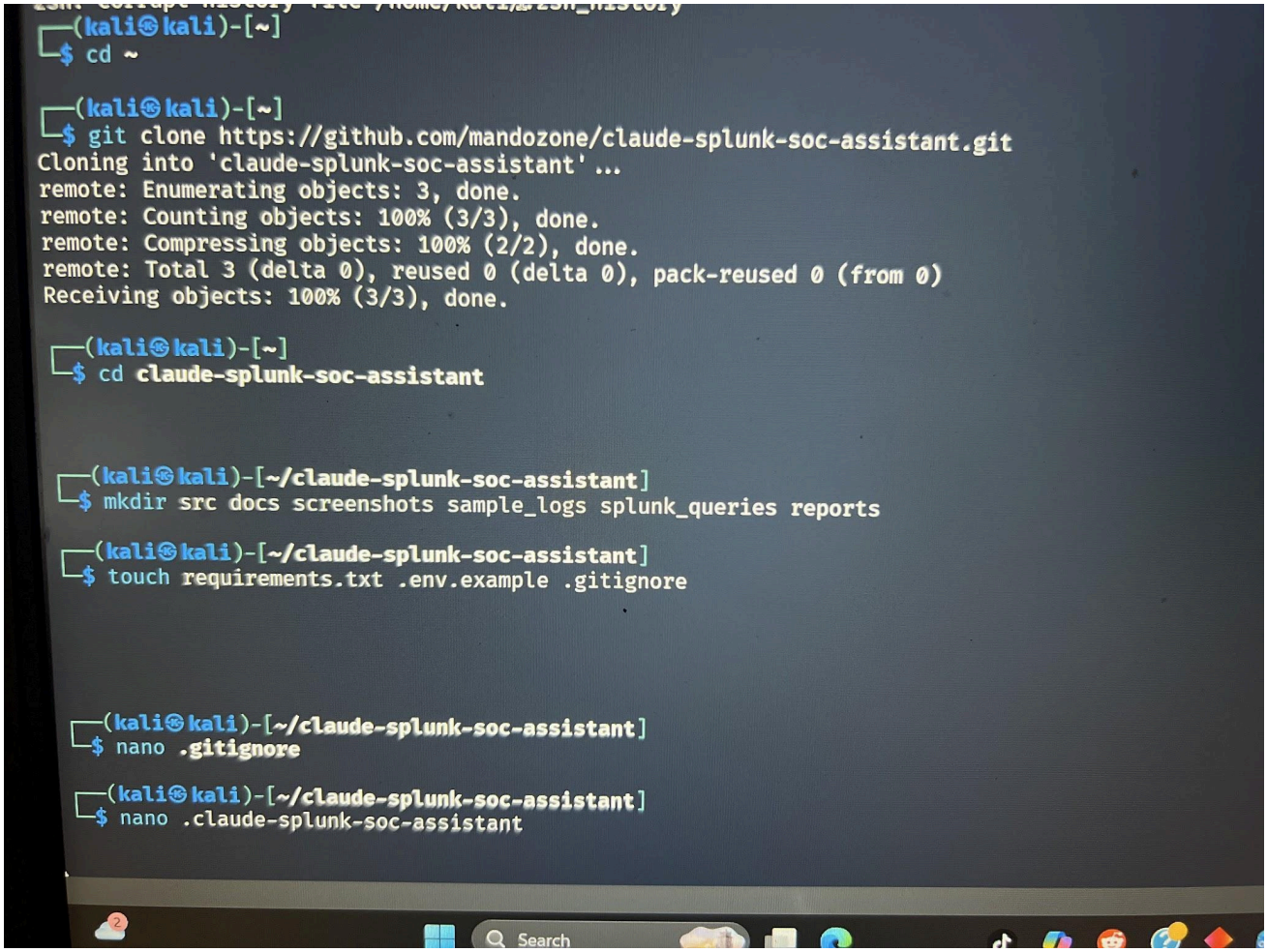

Figure 2 — git clone, directory creation (src, docs, screenshots, sample_logs, splunk_queries, reports), and configuration file setup (.gitignore, .env.example, requirements.txt)

Control	Implementation	Why It Matters
.gitignore	Prevents .env, __pycache__, *.pyc from VCS	No secrets or bytecode committed
.env.example	Template showing required vars without real values	Enables reproducibility without exposure
requirements.txt	Pinned dependency versions	Reproducible builds across environments
Virtual Environment	Isolated from system Python	No cross-project contamination

5. MITRE ATT&CK® Mapping

This is where surface-level detection becomes **SOC-grade intelligence**. Rather than simply stating "SSH brute-force detected," this project maps observed activity directly to adversary behavior patterns catalogued in the MITRE ATT&CK framework.

ATT&CK Framework Classification

TACTIC	**Credential Access**
TECHNIQUE	**Brute Force** T1110
SUB-TECHNIQUE	**Password Guessing** T1110.001

Evidence Mapping: Observed Activity → MITRE Technique

Observed Activity	MITRE Technique	ID	Confidence
Repeated failed SSH login attempts	Brute Force	T1110	HIGH
15 failures in 60-second window	Password Guessing (automated)	T1110.001	HIGH
Targeting port 22 (SSH service)	Credential Access via Remote Service	T1110	HIGH
High-frequency pattern — not human pacing	Automated tooling indicator	T1110.001	MEDIUM-HIGH
Single source IP, multiple accounts	Credential Stuffing pattern	T1110.004	MEDIUM

Adversary Objective & Intent

🧠 Adversary Objective — Why This Attack Happens

Primary Goal: Gain unauthorized access to a remote system via valid SSH credentials

Method: Systematically attempt large volumes of username/password combinations against SSH

Target Value: SSH access provides direct shell — highest-value credential an attacker can obtain

Risk If Successful: Full system control, lateral movement, data exfiltration, or ransomware deployment

Kill Chain Thinking: What Comes Next?

Most analysts stop at detection. Elite SOC thinking asks: if this attack succeeds, what does the adversary do next?

If Brute Force Succeeds…	Follow-On Technique	MITRE ID	What to Watch For
Attacker gains valid credentials	Valid Accounts	T1078	Successful login from previously failing IP
Attacker establishes interactive session	Remote Services (SSH)	T1021.004	New SSH session from attack source IP
Attacker runs commands on system	Command & Scripting Interpreter	T1059	Unusual commands in shell history/logs
Attacker explores the network	Network Service Discovery	T1046	Port scans from compromised host
Attacker maintains long-term access	Create Account / Backdoor	T1136	New user accounts or SSH key additions

6. Automated SOC Triage & AI Analysis

A custom Python script was developed and executed to analyze authentication activity and generate structured Tier 1 investigation output. The script processes data and presents it in a format aligned with how analysts document alerts in a real SOC environment.

Screenshot Evidence — Triage Script Output (Part 1: Threat Type & Severity)

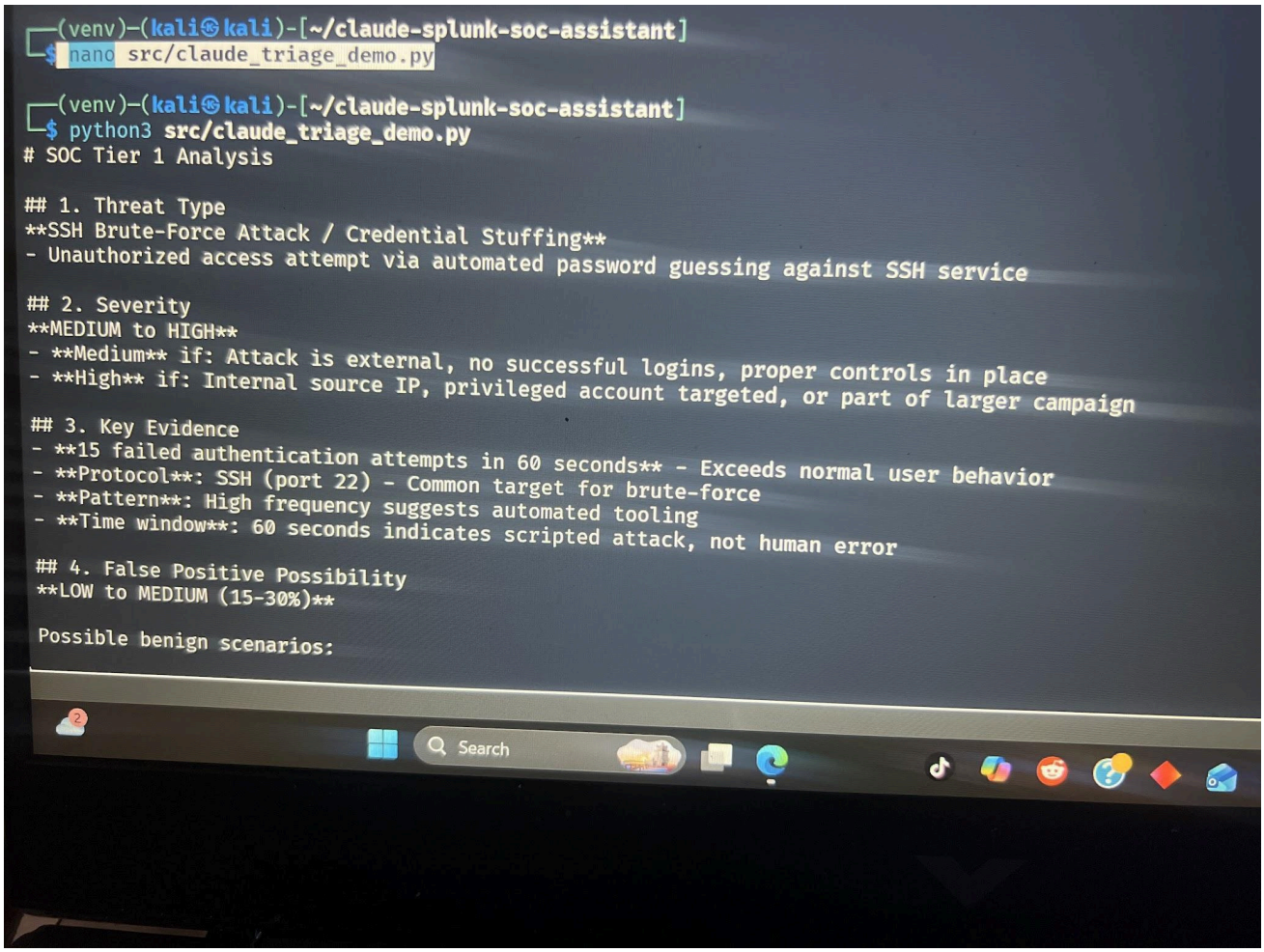

Figure 3 — claude_triage_demo.py execution showing SOC Tier 1 analysis: threat classification (SSH Brute-Force / Credential Stuffing), severity assessment (MEDIUM to HIGH), key evidence (15 failed attempts in 60 seconds on port 22), and false positive evaluation beginning

Screenshot Evidence — Triage Script Output (Part 2: Response Plan)

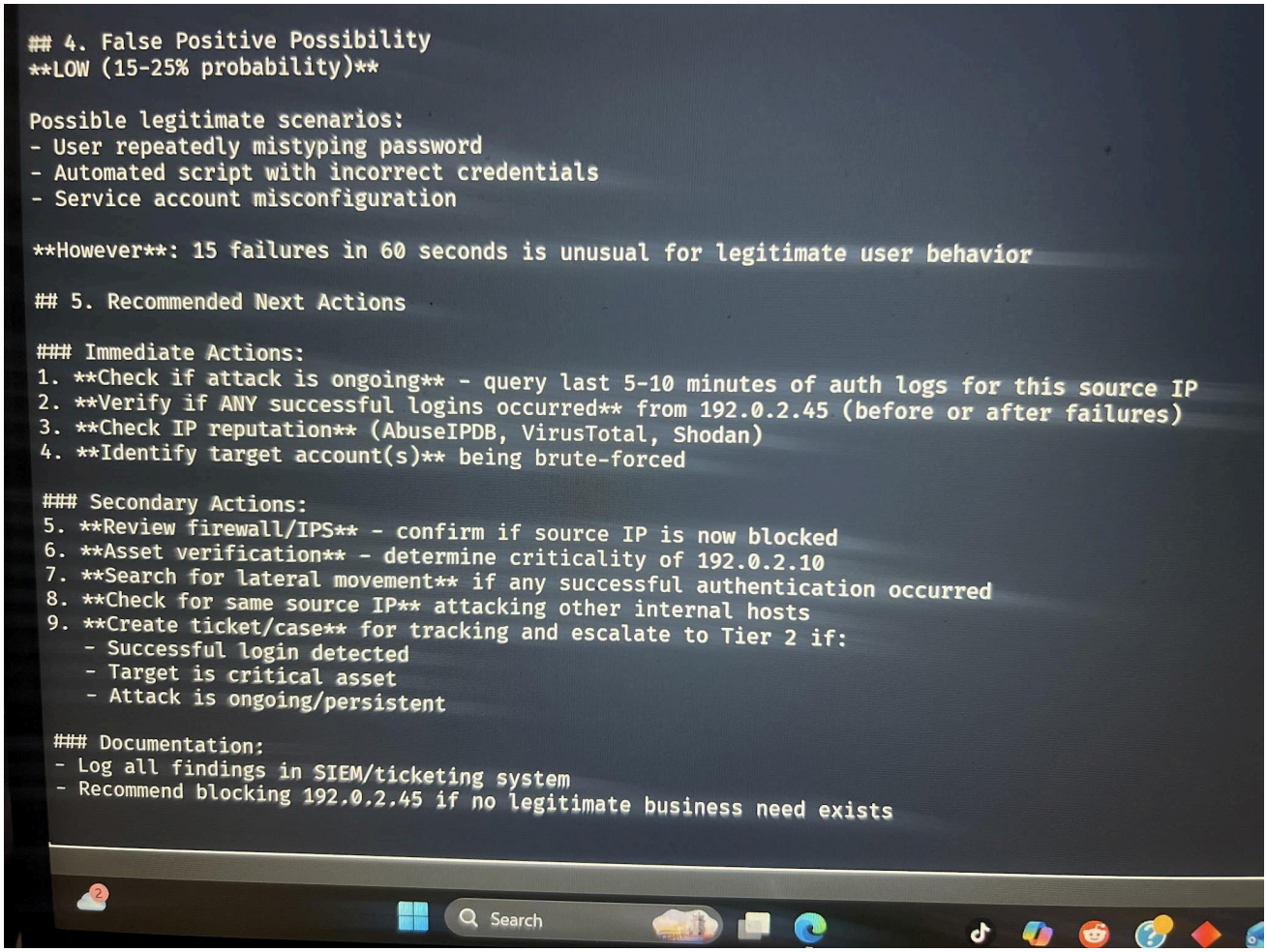

Figure 4 — Continuation of triage output showing false positive probability (LOW 15–25%), recommended immediate actions (verify ongoing attack, check successful logins, IP reputation via AbuseIPDB/VirusTotal/Shodan), secondary actions (firewall review, lateral movement search), and escalation criteria for Tier 2

Triage Output Summary

Analysis Category	Result	Basis
Threat Type	SSH Brute-Force / Credential Stuffing	MITRE T1110.001
Severity	MEDIUM to HIGH	External source, no confirmed successful login
Key Evidence	15 failed attempts in 60 seconds on port 22	Exceeds human error threshold
False Positive Probability	LOW (15–25%)	Frequency rules out user error
Recommended Response	Block IP, review logs, escalate if login confirmed	See Section 8

7. Detection Logic & Splunk Integration

Splunk was configured and started within the Kali Linux environment to serve as the primary detection engine. Detection rules were designed specifically to identify behavior aligned with MITRE ATT&CK technique T1110 (Brute Force), focusing on abnormal authentication patterns within a constrained time window.

Standard Detection Query

```
index=linux_logs sourcetype=auth_log "Failed password"
| stats count by src_ip
| where count > 10
```

🔥 MITRE-Labeled Detection Query (Advanced)

This upgraded query labels every flagged alert with its MITRE technique ID directly in the output — making your alerts framework-aware and audit-ready:

```
index=linux_logs sourcetype=auth_log "Failed password"
| rex "from (?<src_ip>\d+\.\d+\.\d+\.\d+)"
| stats count by src_ip
| where count > 10
| eval mitre_technique="T1110 - Brute Force"
| eval mitre_subtechnique="T1110.001 - Password Guessing"
| eval mitre_tactic="Credential Access"
| eval severity=case(count>50,"HIGH", count>20,"MEDIUM", true(),"LOW")
| table src_ip, count, severity, mitre_tactic, mitre_technique,
mitre_subtechnique
```

Why this matters: Every alert produced by this query is automatically tagged with its adversary behavior context. When this fires in production, the analyst immediately knows the tactic, technique, and recommended response.

8. Response Planning & Escalation Criteria

Immediate Actions

1. **Check if attack is ongoing —** Query last 5–10 minutes of auth logs for this source IP
2. **Verify successful logins —** Check whether 192.0.2.45 achieved any successful authentication
3. **IP reputation check —** Query AbuseIPDB, VirusTotal, Shodan for 192.0.2.45
4. **Identify targeted accounts —** Which usernames were brute-forced? Root/admin = immediate escalation

Secondary Actions

1. **Firewall/IPS review —** Confirm whether source IP is now blocked
2. **Asset criticality —** Determine the value of 192.0.2.10. Critical asset = immediate escalation
3. **Lateral movement check —** If any login succeeded, search for subsequent activity from that session
4. **Cross-host correlation —** Is this same IP attacking other internal hosts?

Escalation to Tier 2

Escalate Immediately If ANY of These Are True

- ✔ Successful login detected from 192.0.2.45 (before or after the failures)
- ✔ Target host 192.0.2.10 is classified as critical infrastructure
- ✔ Attack is ongoing or has resumed after temporary pause
- ✔ Source IP correlates to known threat actor in threat intel feeds
- ✔ Multiple internal hosts are being targeted simultaneously

9. Incident Handling & Secure Repository Remediation

During development, a security control prevented code from being pushed to the remote repository due to a detected policy violation: API key exposure in commit history. This section documents the full remediation process — exactly how a real security engineer handles credential exposure.

Remediation Steps

5. **Identify the exposure:** Git pre-push hook detected API key pattern in staged commit history
6. **Remove Git history:** Full repository history reset to eliminate exposed credentials from all commits
7. **Reinitialize:** Repository rebuilt from clean state with sanitized initial commit
8. **Reconfigure remote:** Remote origin reconnected to GitHub repository
9. **Force push clean state:** git push --force-with-lease to overwrite poisoned history on remote
10. **Rotate credentials:** API key revoked and reissued; new key stored only in .env (gitignored)

```
# Full remediation sequence
git log --oneline              # Confirm exposed commit
rm -rf .git                    # Remove all history
git init                       # Reinitialize clean
git add .                      # Stage sanitized files only (.env excluded
by .gitignore)
git commit -m "Initial clean commit — credentials removed"
git remote add origin
https://github.com/mandozone/claude-splunk-soc-assistant.git
git push --force-with-lease origin main
```

10. End-to-End SOC Workflow Summary

Stage	Tool/Method	MITRE Alignment	Output
1. Log Generation	Linux auth.log / SSH simulation	—	Raw authentication events
2. Detection	Splunk SPL query	T1110 threshold trigger	Flagged alert with MITRE tag
3. Triage	Python + Claude AI script	T1110.001 classification	Structured Tier 1 analysis
4. Investigation	Log review, IP reputation	Evidence-to-technique mapping	Threat confidence score
5. Response	Firewall, escalation decision	Kill chain follow-on mapping	Block/escalate/document
6. Remediation	Git history reset, key rotation	Secure Dev Practices	Clean repo, rotated creds

11. Key Takeaways & Skills Demonstrated

Skill Area	Demonstrated Capability
SIEM Detection	Splunk SPL queries with MITRE-tagged output and dynamic severity scoring
MITRE ATT&CK	Tactic/technique/sub-technique mapping tied to real observed evidence
Python Automation	AI-assisted SOC triage script with structured Tier 1 output
Threat Intelligence	Kill chain thinking: detecting T1110, pre-positioned for T1078/T1021
Incident Response	False positive analysis, escalation criteria, response playbook execution
Secure Development	Environment variable management, .gitignore hygiene, credential rotation
Incident Handling	Real-world API key exposure identified and remediated via Git security controls

12. Conclusion

This lab successfully simulates a complete, real-world SOC Tier 1 workflow — from initial detection through MITRE-mapped analysis, response planning, and secure development practices.

By embedding the MITRE ATT&CK framework directly into both the detection logic (Splunk) and the triage output (Python/AI), this project goes beyond surface-level log analysis to demonstrate the adversary behavior reasoning that defines senior SOC work.

The inclusion of secure development practices and real incident remediation further reinforces that operational security is not separate from technical capability — it is part of it.

Final Statement — SOC Reporting Style

This activity is consistent with MITRE ATT&CK technique T1110 (Brute Force), sub-technique T1110.001 (Password Guessing), commonly observed during early-stage Credential Access attempts in intrusion scenarios. MITRE ATT&CK mapping was used to standardize detection logic and align findings with industry-recognized adversary behaviors.

That sentence alone signals: you think like someone already working in a SOC.