24-Hour Cybersecurity Hackathon Project Plan

Project Overview

Objective: Build an AI-driven security automation framework that integrates with an existing Security Information and Event Management (SIEM) system to enhance threat detection, alert prioritization, and automated response. The solution will use machine learning and heuristic scoring to reduce alert fatigue by filtering false positives, correlate multi-source threat data for high-confidence detections, and automate response workflows through predefined playbooks. Transparency and human oversight are built in via manual approval gates and audit trails.

Problem Statement:

> Develop an AI-driven security automation framework that seamlessly integrates with an existing SIEM infrastructure to enhance threat detection, alert prioritization, and automated response actions. The solution should leverage machine learning to reduce alert fatigue by filtering false positives, correlate multi-source threat data for high-confidence detections and automate response workflows through predefined playbooks — all while maintaining transparency and enabling human oversight.

Open-Source Stack

To satisfy the 24-hour constraint and ensure a fully open-source approach, the following tools comprise the primary stack:

Component	Purpose
Wazuh	SIEM platform (Elastic under the hood). Provides log collection, rule-based alerting and integration points for webhooks and active response.
MITRE ATT&CK	Adversary emulation & technique mapping. Used via **Atomic Red Team** or **CALDERA** for testing and via heatmap visualisation.
MISP (optional)	Threat-intelligence sharing platform. Pulls public feeds and provides indicator hits to the AI scorer.
TheHive + **Cortex**	Case management and enrichment. Cortex analyzers perform indicator lookups; TheHive holds cases and manual approval tasks.
Shuffle (SOAR)	Open-source orchestration engine for workflows. Connects Wazuh, AI triage service, TheHive and active response scripts with manual approval steps.
FastAPI AI Triage Service	Custom microservice written in Python. Scores Wazuh alerts using rule severity, burst counts,

Component	Purpose
	threat intel hits, asset criticality, technique risk and simple heuristics (e.g., suspicious command lines). Produces a score (0–100), human -readable reasons, deduplication key and recommended playbook.
Streamlit (optional)	Lightweight dashboard to visualise alert trends, reasons for suppression, and distribution of recommended playbooks.

Team Roles and Responsibilities

The team consists of four members: two using Windows laptops and two using macOS. Because Wazuh's server/manager runs on Linux, the Windows teammates will host the core infrastructure inside containers (via Docker Desktop and WSL2). Mac teammates will focus on AI development, visualisation and documentation. Use the following team names and assignments:

SplitStack Sentinels (S^3 SOC)

Windows-Only Members (Backend)

Role	Member	Responsibilities
Backend Lead (Win-1)	Windows user 1	Set up Wazuh manager/indexer/dashboard in Docker under WSL2; enrol Windows agent; configure webhooks or polling to send alerts to the AI service; manage infrastructure configuration and bypass paths; maintain active response scripts and firewall rules.
SOAR/Response Lead (Win-2)	Windows user 2	Deploy TheHive + Cortex via Docker Compose; configure Shuffle workflows including enrichment, case creation, manual approval and active response; set up MISP feeds if time allows; maintain role-based access control ensuring only Windows users can trigger containment.

Mac-Only Members (AI and UX)

Role	Member	Responsibilities
AI/Explainability (Mac-1)	macOS user 1	Develop the FastAPI AI triage service; implement scoring logic using severity, threat intel hits,

Role	Member	Responsibilities
		burst counts, asset criticality, technique risk and heuristics; implement deduplication and audit trail (/why/<dedup_key> endpoint); write unit tests; handle MISP lookups (read-only).
UX/Demo & MITRE (Mac-2)	macOS user 2	Design TheHive case templates; build optional Streamlit dashboard; maintain ATT&CK Navigator heatmap JSON; capture metrics for before/after comparisons; assemble slides and demo script; manage documentation.

Rules to enforce Windows vs Mac boundaries

- Only Windows teammates run and modify infrastructure (Wazuh, TheHive, Cortex, Shuffle, MISP). Mac users have read-only access to dashboards and cannot execute containment actions. Secrets and environment variables for infrastructure live only on Windows hosts. The repository can enforce this via a CODEOWNERS file requiring Windows approvals for /infra and /playbooks directories.
- Shuffle roles restrict containment actions (firewall blocks, process kills) to Windows users. Mac users act as analysts with view privileges. All containment tasks require manual approval by a Windows member.

Environment Setup

Because Wazuh's manager is Linux-only, the Windows hosts will leverage **WSL2** and **Docker Desktop** to run the SIEM stack. The following steps summarise the environment preparation:

1. **Install WSL2 and Docker Desktop** on both Windows laptops. Set Docker to use the WSL2 backend and run Linux containers. Adjust the system limit (sudo sysctl -w vm.max_map_count=262144) inside WSL2 to satisfy Elasticsearch requirements.
2. **Deploy Wazuh** using the official single-node Docker Compose. This brings up the manager, indexer and dashboard containers. Exposed ports include 443 (dashboard), 55000 (API), 1514/1515 (TCP for agents), 514/udp and 9200 (Elasticsearch). Enrol the Windows host as a Wazuh agent to generate alerts.
3. **Deploy TheHive and Cortex** using the testing profile from StrangeBee's docker repository. This single compose file launches TheHive, Cortex, Cassandra, Elasticsearch and Nginx. Access TheHive UI to create and view cases and configure Cortex analyzers.
4. **Optional: Deploy MISP** using the official Docker images. Enable at least one public feed; generate an API key to query for indicator hits. This feed can be used by the AI triage service to set a threat intel hit flag.
5. **Optional: Deploy Shuffle** (SOAR). Use the open-source version running on Docker or the hosted community edition. Create connectors for Wazuh, AI triage service, TheHive,

and active response scripts. Build workflows that handle high and low scoring alerts with manual approval.

6. **Set up the AI Triage Service** on Mac-1. Install dependencies (FastAPI, uvicorn, requests). Start the service at http://localhost:8000/score. Optionally expose it to the network so the Windows host can send webhooks directly.

Hour-by-Hour Plan

The following schedule breaks the 24-hour hackathon into hour-long blocks, assigning tasks to each team member. Some tasks may overlap; treat the timeline as a guideline and adjust as necessary.

Hours 0–1: Kickoff & Repository Setup

- **All:** Create a GitHub repository with folders /infra, /ai-triage, /playbooks, /dash and /docs. Add a README summarising the project and roles. Prepare a CODEOWNERS file to enforce approvals: Windows owners for /infra and /playbooks, Mac owners for /ai-triage and /dash.
- **Mac-2:** Draft initial docs/architecture.md and docs/demo-script.md scaffolds with placeholders for diagrams and narrative.

Hour 1–2: Environment Preparation

- **Win-1:** Install WSL2 and Docker Desktop; confirm docker run hello-world works. Set vm.max_map_count inside WSL2. Install Wazuh agent on the Windows host.
- **Win-2:** Install Docker Desktop; verify ability to run containers. Prepare PowerShell environment for active response scripts.
- **Mac-1/2:** Create Python virtual environments; install FastAPI and other dependencies.

Hour 2–3: Deploy Core Services

- **Win-1:** Clone the official wazuh-docker repository and bring up the single-node stack using docker compose up -d. Ensure the dashboard is reachable and the Windows agent is enrolled.
- **Win-2:** Clone StrangeBee's testing profile and bring up TheHive + Cortex via docker compose. Confirm you can create a case and run a Cortex analyzer. Optionally spin up MISP and enable at least one feed.

Hour 3–4: AI Triage Skeleton & Event Wiring

- **Mac-1:** Implement the FastAPI /score endpoint returning a JSON object with score, reasons, dedup_key, suggested_playbook and mitre_techniques. Add a /why/<dedup_key> endpoint to retrieve reasons for audit.
- **Win-1:** Configure a webhook or simple script in Wazuh that posts alert JSON to the AI triage service. If webhooks prove cumbersome, Mac-1 can build a poller that queries Wazuh's API every few seconds.
- **Win-2:** Start building a basic Shuffle workflow: input from the AI service → enrichment via Cortex → create TheHive case → manual approval → active response → notification.

Hour 4–6: Implement Scoring & Deduplication

- **Mac-1:** Define scoring factors and implement them: normalised Wazuh severity, threat-intel hit (MISP search), recent burst count, asset criticality (static YAML or dictionary), technique risk (mapping of MITRE IDs to risk values) and heuristics (e.g., presence of PowerShell or WMI commands). Combine these into a weighted score (0–100) and produce human-readable reasons for each contributing factor.
- **Mac-1:** Implement a deduplication key (e.g., SHA-1 hash of rule ID, source IP, destination IP, process name and technique). Suppress duplicate alerts for a set time window unless the new alert has a higher score.
- **Win-1:** Prepare static asset criticality mapping (e.g., db-prod and dc01 are high criticality). Share with Mac-1 for scoring.
- **Win-2:** Continue refining the Shuffle workflow to include recommended playbooks based on the score (e.g., block_ip for high scores, enrich_only for lower scores).

Hour 6–7: Threat Intelligence & MITRE Mapping

- **Win-2:** Integrate MISP lookups into the AI service: perform a REST search for the source IP or hash and set a ti_hit boolean. Pass this to the scoring function.
- **Mac-2:** Start building an ATT&CK Navigator JSON file highlighting the techniques you plan to emulate (T1059 – command & scripting interpreter, T1047 – WMI, T1021 – remote services). This file will later be used for heatmap visualisation in the demo.

Hour 7–9: SOAR Workflows and Active Response

- **Win-2:** Build two Shuffle workflows:
    1. **High confidence (score ≥ 75):** Enrich via Cortex, create a case in TheHive with all AI metadata, then pause for manual approval. Upon approval, run a Windows active response script to block the IP or kill a process. Send a notification to Slack or another channel.
    2. **Low/medium (score < 75):** Tag and store the alert in Wazuh/Kibana or TheHive for later review. No containment action is taken.
- **Win-1:** Develop the active response scripts. For Windows, provide a PowerShell script that adds a firewall rule (e.g., New-NetFirewallRule -RemoteAddress <IP> -Action Block) and logs the action. Another script can kill a process by name.
- **Mac-1:** Add recommended playbooks into the AI response (block_ip or enrich_only) based on the score.

Hour 9–11: Attack Simulation and End-to-End Test

- **Win-2:** Use Atomic Red Team to run two or three attack simulations against the Windows host (e.g., execute a PowerShell command for T1059, perform a WMI query for T1047, and make a network connection to a known malicious IP). Ensure Wazuh decoders fire and alerts are generated.
- **All:** Observe the end-to-end flow: Wazuh alert → AI triage with score and reasons → Shuffle workflow → TheHive case creation → manual approval by Windows member → active response on the host. Confirm that the case is updated and that the firewall rule or process kill takes effect.

Hour 11–12: Baseline vs Assisted Metrics

- **Win-1:** Temporarily disable the AI triage service (or set all scores to 0) and collect raw alert counts over 15 minutes. Record severity distributions and screenshot relevant dashboards.
- **Mac-2:** Compile baseline metrics into a spreadsheet or chart; note the number of alerts and severity breakdown.
- **Win-1:** Re-enable the AI triage service and run with deduplication on for another 15 minutes. Collect the number of alerts processed, number suppressed, number escalated as high-priority, and the average time from alert generation to manual approval.
- **Mac-2:** Combine baseline and assisted metrics into a single table for the presentation. Highlight percentage reduction in alert volume and improvements in prioritisation.

Hour 12–14: UX Polish and Dashboard

- **Mac-2:** Build out TheHive case template to include fields for AI score, reasons, MITRE techniques, recommended playbook, asset criticality and threat-intel references. Ensure the case view clearly shows why the AI assigned a particular priority and what playbook it recommends.
- **Mac-1:** Implement an audit log file (e.g., triage_audit-YYYYMMDD.jsonl) that records the full scoring context for each alert. Provide an endpoint such as /why/<dedup_key> that returns the reasons and factor values for any deduplicated cluster.
- **Mac-2 (optional):** Create a small Streamlit dashboard in /dash to visualise alert trends (alerts per minute), top suppression reasons, and distribution of recommended playbooks. Use a simple bar or line chart; avoid specifying colours unless necessary.

Hour 14–16: Resilience & Failure Modes

- **Win-1:** Implement a bypass path: if the AI service is unreachable, Wazuh should send alerts directly to TheHive or simply log them without suppression. Document this behaviour in docs.
- **Win-2:** Add a rollback node in Shuffle to remove firewall rules or reverse containment actions. This node should be accessible via manual trigger to demonstrate that actions can be undone.
- **Mac-1:** Write unit tests covering low-score, medium-score and high-score scenarios. Use fixtures to ensure consistent outputs.

Hour 16–18: Demo Script and Slides

- **Mac-2:** Draft a six-slide presentation:
    1. **Problem Statement:** summarise the alert fatigue challenge.
    2. **Architecture Diagram:** show the data flow from Wazuh through the AI service, Shuffle, TheHive and active response.
    3. **Live Demo Plan:** outline the sequence of actions you will demonstrate.
    4. **Results:** present baseline vs assisted metrics and highlight reductions.
    5. **Transparency & Oversight:** show the audit trail and manual approval process.
    6. **Roadmap:** discuss potential future enhancements (e.g., online learning from analyst feedback, improved similarity clustering, Slack integration).

- **All:** Rehearse the live demo. Assign speaking roles: Mac-2 narrates the story, Mac-1 explains the scoring and reasons, Win-2 drives the manual approval click, and Win-1 fields infrastructure questions.

Hour 18–20: Stretch Features (Optional)

If time allows, pick one or two enhancements:

- **Add Isolation Forest:** Implement an Isolation Forest model in the AI service using scikit-learn to detect anomalies based on features like burst count, severity and asset criticality. Blend its output into the final score (e.g., weighting 10–15%). Provide explanations for the anomaly score.
- **Slack/Discord Bot:** Configure Shuffle to send a message to Slack or Discord when a high-priority alert arrives, providing buttons to approve or reject containment directly within the chat client.

Hour 20–22: Packaging & Documentation

- **Win-2:** Export Shuffle workflows to a JSON file and commit it to /playbooks. Commit active response scripts (PowerShell) into /playbooks/ar_windows.
- **Win-1:** Add an infra/README.md detailing how to run the Wazuh and TheHive stacks via Docker Compose. Document environment variables and port configurations.
- **Mac-2:** Store the ATT&CK Navigator heatmap JSON and metrics screenshots in /docs. Finalise the slides and rehearse again.
- **Mac-1:** Create an ai-triage/README.md with instructions to start the service, required environment variables (e.g., MISP API key) and example input/output.

Hour 22–24: Final Rehearsal & Contingency

- **All:** Perform a full end-to-end rehearsal with a fresh attack simulation. Verify that the entire pipeline works reliably: alert ingestion, scoring, case creation, manual approval and response.
- **Win-1:** Prepare a short prerecorded screen recording of a successful run as a contingency in case of connectivity issues during the live demo.
- **Mac-2:** Export the slides to PDF and have a printed copy as backup.
- **Win-2:** Keep manual scripts ready to block an IP or kill a process via PowerShell if Shuffle or network connectivity fails.

Deliverables & Judging Points

To win the hackathon, the project should showcase the following:

1. **Working Pipeline:** Wazuh alerts processed by the AI triage service, ingested into Shuffle, enriched by Cortex, escalated to TheHive with AI metadata, gated by manual approval, and executed via Windows active response. At least three MITRE techniques (e.g., T1059, T1047, T1021) should be demonstrated end-to-end.
2. **Impact Metrics:** Evidence of alert reduction, improved prioritisation and faster response. Provide before/after numbers, e.g., percentage of alerts suppressed, number of high-confidence alerts surfaced and average time to approve.

3. **Transparency and Oversight:** Human-readable reasons for each decision; /why endpoint for audit; manual approval and rollback steps; role-based access control ensuring only Windows members can take destructive actions.
4. **Open-Source Stack:** Use only open-source tools (Wazuh, TheHive, Cortex, MISP, Shuffle, FastAPI, Streamlit). Demonstrate how containers make Linux-only components accessible on Windows via WSL2.
5. **Documentation and Ease of Use:** Provide clear README files and a starter kit that allow judges or other teams to replicate the environment and run the demo. Include diagrams, scripts and instructions.

Conclusion & Next Steps

This plan guides the SplitStack Sentinels through a 24-hour hackathon to build a compelling AI-assisted SOC solution. By dividing responsibilities across Windows and macOS teammates, leveraging containerised open-source tools and focusing on transparency and impact, the team can deliver a polished demo with measurable results and a clear roadmap for future improvements. After the hackathon, future work could include learning from analyst feedback to adjust scoring weights automatically, implementing advanced clustering for deduplication, integrating chat-based approval workflows, and enriching asset criticality with dynamic inventory data.