CyberShield Command

Document version 4.0
del 25.01.2025

Group 1
Group Name: Horses Probability
Group Members
Nicola Balzano 828101
Alessandro Aldo Boffolo 841260
Narcis Paviliuc 838039

"Defend. Detect. Respond."

1 Index

2 Planning – Game Overview

CyberShield Command is a serious web game set in a realistic corporate environment. Players take on the role of a Security Operations Centre (SOC) analyst. Their job is to defend the company's network from a variety of realistic cyber threats by carrying out technical analysis and incident response tasks.

They use professional SOC tools, such as a SIEM dashboard, terminal, browser, email client and code editor, to identify attacks, analyse logs and implement mitigations. Scenarios include phishing detection, DDoS mitigation, SQL injection and cross-site scripting (XSS) defence, cross-site request forgery (CSRF) protection, cache poisoning, ransomware response, and cryptography vulnerabilities.

Throughout each mission, the player is supported by CyberNav, an intelligent hint system that provides guidance tailored to their progress. If the player gets stuck, the hints become more specific over time, so that beginners can learn without frustration and experienced users can still be challenged.

The game features a health bar that decreases with incorrect actions or time pressure, as well as a star rating system that rewards thorough analysis and the completion of bonus objectives. At the end of each mission, a debriefing screen summarises performance and reinforces learning.

Overall, CyberShield Command offers an engaging and immersive way to develop practical cybersecurity skills, bridging the gap between theory and practice in a safe virtual environment.

2.1 Goal of the Game

The goal of the game is to enhance cybersecurity competencies by placing the player in the role of a SOC Analyst who must respond to active cyber-attacks through hands-on technical analysis. The player will:

- **Analyze and respond to realistic cyber threats:** Identify, investigate, and mitigate attacks based on real-world scenarios including Phishing, DDoS, SQL Injection, XSS (Cross-Site Scripting), CSRF (Cross-Site Request Forgery), Cache Poisoning, Ransomware, and Cryptography Vulnerabilities.
- **Use professional SOC tools:** Operate a simulated workstation with Terminal, SIEM Dashboard, Browser, Email Client, Code Editor, Packet Analyzer, and Decryption Tools to investigate incidents.
- **Monitor and correlate security events:** Analyze SIEM logs in real-time, click on suspicious entries to investigate, and identify attack patterns (e.g., SQL injection patterns, malicious IPs, HTTP flood indicators).
- **Execute mitigations via command-line:** Use terminal commands to block malicious IPs, enable security features (sanitization, CSP, HttpOnly cookies, rate limiting), purge poisoned caches, and restart affected services.
- **Fix vulnerable code:** Edit source code to patch security vulnerabilities (e.g., implementing prepared statements for SQL injection, proper input sanitization for XSS).
- **Make time-critical decisions:** Many levels include a countdown timer and a health bar that decreases with mistakes or time pressure, simulating the urgency of real incident response.
- **Earn performance ratings:** Complete bonus objectives to earn up to 3 stars per level, encouraging thorough analysis and mastery of security concepts.

2.2 Target Audience

2.2.1 Player Profile Characteristics

Item	High School Students	University Student
Age	14-18 years	18-30 years
Educational Level	High School or Vocational Training	Bachelor's Degree or ITS
Motivation	Curiosity about cybersecurity, Required coursework	Career advancement, Practical skills
Prerequisite Knowledge	Basic IT concepts (files, networks, passwords)	Intermediate IT knowledge, Some security awareness
Prerequisite Skills	Basic computer navigation, File management	Network understanding, Command-line basics
Facility with a Computer	Good (daily use)	Excellent (professional use)
Familiarity with the Web	Good (browsing, basic interaction)	Excellent (web applications, online tools)
Typing Ability	Average (40-50 WPM)	Fast (60-80 WPM)
Access to Computers	Home/School computer access	Personal computer + Work access
Access to Web	Reliable broadband connection	Consistent high-speed internet
Time Availability	90/135 minutes to complete all the game	45/90 minutes to complete all the game

Tabella 1 - Player Profile Characteristics

2.2.2 Learning Preferences by Player Type

2.2.2.1 Weaker Learners

- **Game Difficulty**: 3 levels are easy
- **Support Needed**: Extensive tutorials, step-by-step guides, tooltips, hint system
- **Content**: Short sessions, clear objectives
- **Pacing**: Self-paced with optional time limits
- **Focus Areas**: Basic concepts (what is a firewall, what is SIEM)

2.2.2.2 Average Learners

- **Game Difficulty**: 3 levels are intermediate
- **Support Needed**: Context-sensitive help, knowledge base
- **Content**: Balanced mix, progressive challenges
- **Pacing**: Guided progression with some freedom
- **Focus Areas**: Practical skills (when to upgrade, incident response basics)

2.2.2.3 Stronger Learners

- **Game Difficulty**: 3 levels are advanced
- **Support Needed**: Minimal guidance, advanced analytics, challenge modes
- **Content**: Complex scenarios, multiple simultaneous threats
- **Pacing**: Self-directed, competitive leaderboards
- **Focus Areas**: Strategic optimization, advanced threat scenarios

2.2.3 Accessibility Requirements

Requirement	Implementation
Screen Reader Support	ARIA labels, semantic HTML, keyboard navigation
Color Blind Mode	Alternative color palettes, pattern-based indicators
Font Scaling	Responsive typography, zoom support
Keyboard Navigation	Full game playable with keyboard and mouse
Language Support	Italian, English, French, Spanish, German
Mobile Friendly	Desktop-first, mobile support planned for Phase 2

Tabella 2 - Accessibility Requirements

2.2.4 Device Requirements

Device Type	Minimum Specs	Recommended Specs
Desktop/Laptop	4GB RAM, Intel i5/AMD R5, 5Mbps internet	8GB RAM, Intel i7/AMD R7, 10+ Mbps
Browser	Chrome 90+, Firefox 88+, Safari 14+, Edge 90+	Latest versions recommended
Screen Resolution	1280x720 (720p)	1920x1080 (1080p) or higher
Connection	Stable broadband (5+ Mbps)	High-speed (20+ Mbps recommended)

Tabella 3 - Device Requirements

2.3 Game Genre(s)

The game is classified as a **Cybersecurity Simulation Game** with elements of:

- **Strategy Game**: Players must analyze logs, code, and communications to identify specific vulnerabilities
- **Real-time Strategy (RTS)**: Active threat events (like DDoS) require immediate reaction to preserve system integrity

The player simulates the role of a Chief Information Security Officer (CISO) or Security Operations Center Manager in charge of defending a company against a dynamic threat landscape.

2.4 Constraints

The game can be played from any device capable of accessing the web, including Windows and macOS computers, tablets, and smartphones. The interface is designed with a responsive layout that adapts to different screen sizes while maintaining a clear and professional appearance suitable for its SOC workstation simulation.

Progress is automatically saved to a backend server. The game tracks stars earned per level (up to 3 stars for each of the 10 levels), user settings (language, music volume, sound effects volume), and timestamps. When the player completes a level or changes settings, the data is persisted via API calls to the backend.

Currently, the game does not include user authentication. All players share the same save file on the server. In a future release, a login system could be implemented to allow multiple users to maintain separate progress profiles and synchronize their data across devices.

2.4.1 Platform

The game is designed to run on modern web browsers (Chrome, Firefox, Safari, Edge) on desktop platforms. The architecture consists of:

- **Frontend:** React.js 19 + Vite 7 (web-based single-page application) with TailwindCSS for styling and React Router for navigation
- **Backend:** Python Flask REST API with CORS support
- **Data Storage:** JSON file-based persistence (database.json) for game progress and settings

The containerized deployment uses Docker Compose to orchestrate both frontend and backend services.
Mobile platform support may be considered for future releases after the initial launch.

2.4.2 Budget

This project follows a **zero-budget approach** for initial MVP development:

Expert's Role	Experts number	Amount for 3 months	Total (3 months)
Game Designer	2	5.000$	10.000$
Security Expert	2	7.500$	15.000$
Game Developer	2	5.000$	10.000$
Sound Expert	1	5.000$	5.000$
UX Expert	1	7.500$	7.500$
Backend Developer	1	3.500$	3.500$
DevOps	1	3.000$	3.000$
Total x 3 months	54.000$		

Tabella 4 - Budget

2.4.3 Timetable

Phase	Duration	Key Deliverables	Team Focus
Phase 1: Foundation & Setup	2 weeks	• Backend API infrastructure • Frontend project scaffolding (React + Vite) • Development environment setup • Version control and CI/CD pipeline	Game Designer Security Expert Game Developer Sound Expert UX Expert Backend Developer DevOps
Phase 2: Core Game Engine	3 weeks	• Game mechanics implementation • SIEM console UI and functionality • Attack generation engine • Threat catalog • Company management system • Budget and resource system	Backend Developer Game Developer UX Expert
Phase 3: Polish & Education	3 weeks	• UI/UX refinement and testing • Interactive tutorial system • Educational content creation • Employee training module • Knowledge base and glossary • Sound and visual effects	Game Developer Game Designer Content Writer UX Expert
Phase 4: Quality Assurance	2 weeks	• Comprehensive QA testing • Bug identification and fixing • Performance optimization • Security audit and penetration testing • Cross-browser compatibility testing	DevOps Backend Developer Security Expert
Phase 5: Launch & Release	2 weeks	• Deployment to production servers • Final documentation • User acceptance testing (UAT) • Beta launch to limited audience • Community feedback collection • Performance monitoring setup	DevOps Backend Developer Security Expert

Tabella 5 - Timetable

2.5 Look and Feel

Visual Style:

- Color Scheme: Dark theme with cyber-black background (#0a0e17) and neon accent colors (green #00ff41, blue #00f3ff, red #ff3333, yellow #ffbf00)
- Aesthetic: Futuristic cybersecurity operations room, inspired by modern SOC environments
- Tone: Professional yet engaging; blend of technical realism with intuitive UI

Interface Design:

- SIEM console with severity-colored log entries (Green=Low, Yellow=Medium, Orange=High, Red=Critical)
- Health bar representing company integrity and a star rating system (up to 3 stars per level)
- SOC workstation with multiple tools: Terminal, Browser, Email Client, SIEM Dashboard, Code Editor, Packet Analyzer
- Timer display for time-limited missions
- CyberNav hint panel with typewriter text animation
- Clean, uncluttered design with dark surfaces and glowing accents
- Responsive layout (initially desktop-focused)

Typography and Icons:

- System sans-serif fonts with monospace styling for terminal and code elements
- PNG image assets for backgrounds, characters, and UI elements
- Emoji-based icons for quick tool identification (examples: 📧 Email, 💻 Terminal, 🌐 Browser, 📊 SIEM)

Audio:

- Looped background music with cybersecurity thriller atmosphere
- Sound effects for: earning stars, receiving hints, taking damage, healing, mouse clicks, and keyboard typing
- Separate volume controls for music and sound effects
- Audio unlocks after first user interaction (browser autoplay policy)

2.6 Resources

Development Stack:

- Frontend Framework: React 19 with Vite 7 build tool
- UI Styling: TailwindCSS 4
- Routing: React Router DOM 7
- Backend Framework: Flask (Python) with Flask-CORS
- Data Storage: JSON file-based persistence (database.json)
- API Communication: REST API with JSON payloads
- Containerization: Docker Compose for orchestrating frontend and backend services
- Version Control: Git with GitHub

Third-party Libraries:

- Frontend:
  - react / react-dom (UI library)
  - react-router-dom (client-side routing)
  - tailwindcss (utility-first CSS framework)
  - @vitejs/plugin-react (Vite React integration)
  - autoprefixer / postcss (CSS processing)
- Backend:
  - flask (web framework)
  - flask-cors (cross-origin request handling)

Assets and Resources:

- Icons: Emoji-based icons and PNG images
- Images: Custom PNG assets (backgrounds, characters, UI elements)
- Audio: MP3 files for background music and sound effects (keyboard, mouse, hints, damage, heal, star)
- Documentation: Markdown files in repository

Development Environment:

- Code Editor: VSCode
- Containerization: Docker + Docker Compose
- Local Development: npm run dev (frontend on port 5173), flask run (backend on port 5000)

2.7 Game Flow Summary

The player accesses the web application and begins by selecting "New Game" or "Continue" from the home screen. The game presents a level-based structure where each level simulates a specific cyber attack scenario that the player must investigate and mitigate.

Main Flow:

1) **Home Screen:** Player selects "New Game" (resets progress) or "Continue" (resumes from last checkpoint)
2) **Level Map:** Overview of all 10 levels (Tutorial + Levels 1-9), showing earned stars and progression status
3) **Core Gameplay Loop (per level):**
    a) Receive briefing: CyberNav provides context and initial hints about the scenario
    b) Monitor SIEM events: Analyze real-time security logs with severity indicators
    c) Investigate threats: Use Browser, Terminal, Email Client, and other SOC tools
    d) Execute mitigations: Block IPs, enable protections, fix code, purge caches, etc.
    e) Manage time and health: Complete objectives before the timer expires or health depletes
    f) Earn stars: Complete bonus objectives for 1-3 stars per level
    g) Mission Debrief: Summary screen showing success/failure, stars earned, and statistics
4) **Progression**: Completing levels unlocks the next scenario; stars accumulate across all levels
5) **End States** (per level):
    a) Success: All required objectives completed, optional stars earned
    b) Failure: Health depleted or timer expired; player can retry the level

3 Design

3.1 Mechanics

The core of CyberShield Command lies in the realistic simulation of the responsibilities of a SOC (Security Operations Center) Analyst. The game mechanics are designed to translate abstract cybersecurity concepts into hands-on technical actions and time-critical incident response decisions, bridging the gap between theory and practice.

Players use authentic SOC tools, including a SIEM dashboard, terminal, browser, email client, and code editor, to investigate security alerts, analyze attack patterns, and execute mitigations in real-time. Each level presents a specific cyber threat scenario (phishing, DDoS, SQL injection, XSS, ransomware, etc.) that requires the player to apply defensive techniques within time and health constraints.

3.1.1 Space

The game environment is conceived as a discrete two-dimensional space that faithfully reproduces the operational workstation of a corporate SOC analyst. There is no physical world to explore through an avatar; instead, the "battlefield" is the interface itself.

The primary view presents a simulated SOC workstation with a central monitor surrounded by multiple functional panels. The aesthetic blends professional operating systems with a futuristic, high-tech dark theme featuring neon accent colors. The player acts as a SOC analyst, navigating between various security tools, including the SIEM Dashboard, Terminal, Browser, Email Client, and Code Editor. Movement is not physical but informational, shifting focus between different monitoring tools to investigate threats and execute defensive actions.

An InfoPanel (CyberNav) provides contextual hints and guidance, appearing in the corner of the screen to assist players who need direction during each scenario.

3.1.2 Objects

The game world is populated by interface elements and dynamic data objects that the player must analyze and act upon.

Tools and Interface Elements form the player's workspace: the SIEM Dashboard, Terminal, Browser, Email Client, Code Editor, and specialized tools like the Packet Analyzer. These remain consistent across levels but display different content based on the active scenario.

Dynamic Data Objects represent the security events requiring investigation:

- SIEM Log Entries: Real-time alerts with timestamps, source IPs, severity levels (low/medium/high/critical), and threat indicators
- Emails: Incoming messages that may be legitimate communications or phishing attempts
- User Inputs: Comments, form submissions, or code that may contain malicious payloads (XSS, SQL injection)

- Cache Entries: HTTP cached responses that may be poisoned with malicious content
- Network Packets: Captured traffic data for forensic analysis
- Source Code Files: Editable code containing vulnerabilities that must be patched

Quantifiable Resources act as the barometer of success:

- Health Bar: Represents company integrity; decreases with mistakes or time pressure (0 = mission failure)
- Stars: Performance rating (1-3 per level) earned by completing bonus objectives
- Timer: Countdown clock for time-critical scenarios

3.1.3 Actions

The player's interaction with the system is direct, performed via mouse or touch interface to navigate between SOC tools. The fundamental action is Monitoring: the player must scrutinize the SIEM console for anomalies among the system logs, identifying entries marked with critical or high severity.

Once a potential threat is spotted, the gameplay shifts to Investigation: suspicious logs are analyzed by clicking on entries, researching threats in the Browser (consulting OWASP guides and threat intelligence), running diagnostic commands in the Terminal (show-logs, analyze, status), and inspecting email headers for phishing indicators.

If a threat is confirmed, the player must Mitigate the risk through direct technical actions:

- Terminal Commands: Block malicious IPs (block-ip), enable security features (enable-sanitization, enable-csp, enable-httponly), purge caches (purge-cache), enable rate limiting, and restart services
- Code Editing: Fix vulnerable source code by implementing secure coding practices (e.g., prepared statements, proper input validation)
- Email Classification: Flag phishing emails and approve legitimate ones

All actions must be completed within time and health constraints, requiring the player to prioritize effectively and act decisively under pressure.

3.1.4 Rules

The game runs in real-time with a countdown timer that creates urgency. Most levels impose a 5-minute time limit, during which the player must complete objectives while managing a health bar that decreases with mistakes or time pressure.

Win Condition: Each level has a specific primary objective that must be completed before health reaches zero. Examples include:

- Correctly classifying all phishing emails
- Blocking all malicious IP addresses during a DDoS attack
- Patching vulnerable code to prevent SQL injection
- Purging a poisoned cache and securing HTTP headers

Failure Condition: The mission fails if health depletes to zero. Health is lost through:

- Time pressure: In some levels, health decreases continuously as time passes
- Incorrect actions: Blocking legitimate IPs, misclassifying emails, or failing security tests

Reward System: Successful completion earns 1-3 stars based on bonus objectives:

- Using advanced analysis commands
- Inspecting email headers or log details
- Implementing multiple security layers
- Completing the level quickly

Progressive Guidance: If the player struggles, the CyberNav hint system provides increasingly specific guidance every 15 seconds, ensuring accessibility for beginners while maintaining challenge for experienced players.

3.1.5 Skills

To excel in CyberShield Command, the player must develop a set of transferable cybersecurity skills that mirror real-world SOC analyst competencies.

Log Analysis is fundamental: the player must learn to scan SIEM logs efficiently, distinguishing routine events (low severity, internal IPs) from genuine threats (critical severity, external sources, attack patterns).

Pattern Recognition develops over time as the player learns to identify the digital signatures of specific attack types SQL injection patterns (' OR 1=1 --), XSS payloads (<script>, onerror=), suspicious HTTP headers, or abnormal request rates indicating DDoS.

Prioritization Under Pressure is crucial when facing time-limited scenarios. The player must quickly identify which threats pose the greatest risk and address them first, balancing thoroughness with speed.

Technical Execution requires mastering terminal commands, understanding security configurations (CSP, HttpOnly cookies, input sanitization), and knowing how to patch vulnerable code.

Phishing Detection sharpens the player's ability to spot social engineering attempts by examining sender addresses, email headers, suspicious links, and urgency tactics.

These skills directly translate to real-world cybersecurity awareness, making the game an effective training tool for aspiring security professionals.

3.1.6 Chanc (Easter eggs)

CyberShield Command is primarily a skill-based game with deterministic scenarios. Each level presents a fixed set of challenges, allowing players to learn from mistakes and improve on subsequent attempts.

While the core scenarios are scripted, variability comes from player choice: the order in which threats are addressed, which tools are used, and which bonus objectives are pursued all affect the outcome and star rating.

Future Development: Random elements such as varied attack timings, dynamic false positive rates, or procedurally generated scenarios could be added in future versions to increase replayability.

3.2 Story

The narrative provides essential context to give weight to the player's actions, transforming technical operations into meaningful incident response decisions.

Rather than an overarching storyline, CyberShield Command uses scenario-based storytelling: each level presents a specific security incident with contextual briefings delivered through the CyberNav hint system, email alerts, and SIEM logs. The player experiences the urgency of real-world threats phishing campaigns targeting employees, DDoS attacks overwhelming company servers, ransomware encrypting critical files and must respond as a SOC analyst would.

This approach reinforces the educational objective: players learn not just how to respond to threats, but why these responses matter in a corporate security context.

3.2.1 Characters

The protagonist is the Player themselves, stepping into the role of a newly hired SOC Analyst. They are a silent protagonist whose competence is defined by the quality of their defensive actions and analytical decisions.

Guiding them is CyberNav, an intelligent assistant integrated into the game interface. This digital guide provides progressive hints and contextual guidance, appearing in a dedicated panel with a typewriter text effect. CyberNav walks the player through tutorials and offers increasingly specific suggestions if they appear stuck, ensuring accessibility without removing the challenge.

The antagonists are represented abstractly through the attack scenarios themselves: malicious IP addresses, phishing senders, and attack patterns visible in SIEM logs. Attackers are identified by suspicious usernames (e.g., "attacker", "malicious_user") and external IP addresses, while legitimate users appear as internal employees going about normal business creating a realistic environment where threat detection requires distinguishing malicious actors from regular network activity.

This approach keeps the focus on technical skill development rather than narrative drama, mirroring the reality of SOC work where threats are often faceless and identified only through their digital footprints.

3.3 Aesthetics

The game's visual identity defines a "Cyber-Future" aesthetic designed to immerse the player in a high-stakes security environment.

Visual Style:

- Dark Mode Core: The interface is built on a deep cyber-black (#0a0e17) background that reduces eye strain during long sessions and mimics professional dark-mode developer tools.
- Neon Accents: Critical system status is communicated through high-contrast neon colors: cyber-green (#00ff41) for safe systems, cyber-neon-blue (#00f3ff) for information, and cyber-alert-red (#ff3333) for active threats.
- Atmosphere: The design creates a "Hollywood Hacking" vibe clean, uncluttered, and strictly functional, stripping away decorative elements to focus on raw data and immediate threat indicators.

This aesthetic serves a functional purpose: the stark contrast between the dark background and glowing alerts ensures that the player's attention is instantly drawn to anomalies and critical security breaches.

Figure 1 - Main Menu

3.3.1 Mission/challenge Structure

The game is divided into distinct Levels, each representing a specific cyber incident scenario. Every level follows a standardized cycle:

- Briefing: Upon starting, CyberNav or an email alert provides the context for the current threat (e.g., "Suspicious login attempts detected" or "Ransomware outbreak confirmed").
- Investigation Phase: The player explores the simulated environment using SOC tools to identify the nature and source of the threat without a time limit (in early stages) or under initial pressure.

- Action Phase (Response): The core gameplay where the player executes mitigations blocking IPs, patching code, or isolating systems often while a countdown timer adds urgency.
- Debriefing: A final report screen evaluates performance based on:
    - Success/Failure status
    - Star Rating (1-3): Based on completing bonus objectives and thoroughness
    - Time taken: Efficiency of the response
    - Health remaining: Integrity of the company's systems

3.3.2 Levels

Difficulty progression is linear but scalable, introducing new mechanics and tools as the player gains confidence. Each level focuses on a specific type of cyber threat.

Figura 2 - Level Map

3.3.2.1 Training level

The introductory level, titled "Tutorial: SOC Onboarding," serves as an interactive guide. The objective is to familiarize the player with the interface and the core workflow.

Scenario: The player joins the SOC team and learns to use the foundational tools:

- Email Client: Identifying security alerts
- SIEM Dashboard: Interpreting log severity (INFO vs CRITICAL)
- Browser: Researching threat patterns
- Terminal: Executing the first defensive command

Objective: Identify a SQL Injection attempt in the logs and block the malicious IP (203.0.113.42).

Guidance: The CyberNav assistant guides the player step-by-step. While failure is possible (wrong actions damage the health bar to teach consequences), the difficulty is forgiving to ensure a smooth learning curve.

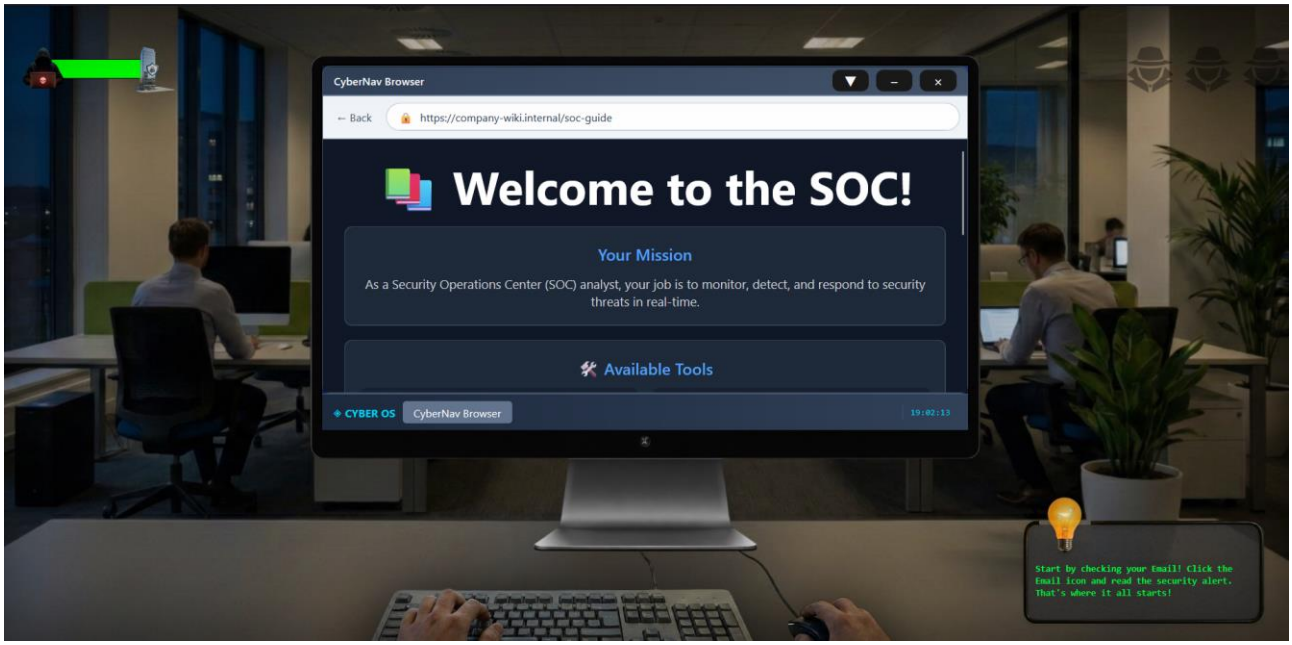

Figure 3 - Tutorial

3.3.2.2 For each level

The campaign takes the player through a series of escalating incidents, each focusing on specific vulnerability classes and defense strategies.

Level 1: The Human Factor (Phishing)

Objective: The player must analyze a batch of 6 incoming emails and correctly classify them as "Legitimate" or "Phishing" before time runs out.

Storyboard: A wave of suspicious emails has hit the company inbox. The player must use the Email Client to inspect sender addresses, subject lines, and message content for social engineering indicators (urgency, mismatched domains, suspicious attachments).

Mechanics:

- Header Inspection: Clicking on the "From" field reveals the true sender address (e.g., ceo@compny.com instead of ceo@company.com).
- Content Analysis: Using the Browser to cross-reference claims (e.g., verifying a PayPal alert against the official anti-phishing guide).
- Decision Making: Marking an email as "Phishing" or "Safe". Incorrect decisions (marking a safe email as phishing or vice versa) result in immediate damage to the Health Bar.

Threats: The level features a mix of broad spam attempts and sophisticated spear-phishing (impersonating the CEO or IT support). Success relies on attention to detail rather than technical tools.

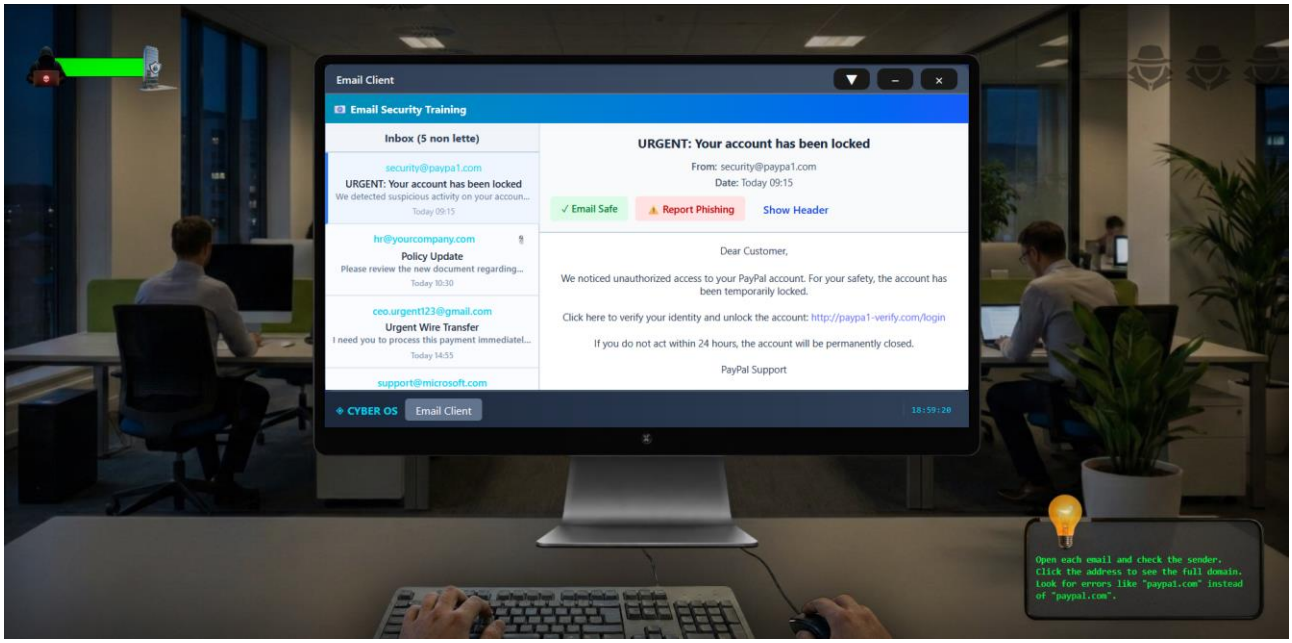

Figure 4 - Level 1

Level 2: The Traffic Flood (DDoS Mitigation)

Objective: Maintain service availability by mitigating a massive Distributed Denial of Service (DDoS) attack while avoiding blocking legitimate users.

Storyboard: The company website has gone offline due to an overwhelming surge in traffic. The SIEM dashboard shows a spike in incoming requests, but it's unclear which are real customers and which are bots.

Mechanics:

- Traffic Analysis: The player uses the Terminal command analyze to see request patterns and list-ips to view the most active IP addresses.
- differentiation: The player must distinguish malicious IPs (high request rate > 400 req/s, suspicious patterns) from legitimate users (normal rate).
- Mitigation:
  - Blocking: Use block <ip> to ban specific malicious sources.
  - Rate Limiting: Execute rate-limit to heavily throttle high-volume traffic.
  - Firewall: Enable the global firewall with enable-firewall.
- Consequence: Blocking a legitimate IP (False Positive) causes immediate health damage. The level ends when traffic returns to normal levels (< 5%).

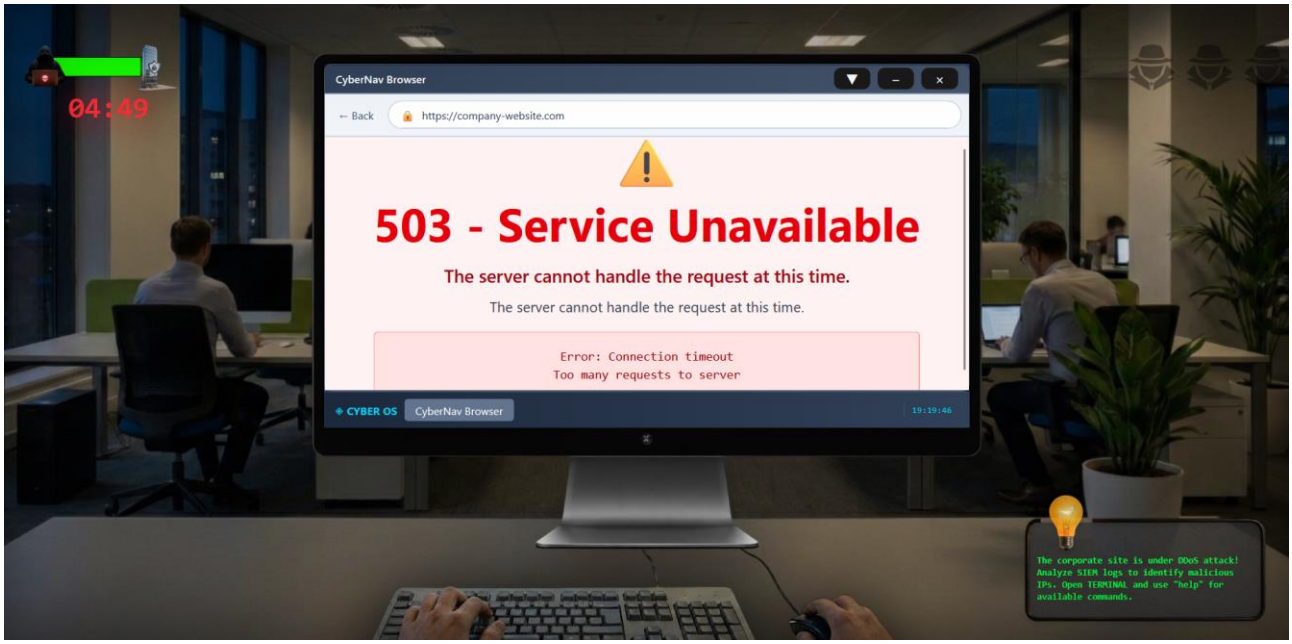

Figure 5 - Level 2

Level 3: The Database Breach (SQL Injection)

Objective: Patch a critical SQL Injection vulnerability in the legacy login portal before customer data is compromised.

Storyboard: The SIEM Dashboard flags multiple failed login attempts with suspicious syntax (' OR 1=1 --). The player investigates the logs and confirms an active SQL Injection attack targeting the login.php endpoint. The attacker is attempting to bypass authentication to access the administrative dashboard.

Mechanics:

- Analysis: The player uses the Terminal command analyze-code to identify the vulnerable function in the legacy PHP codebase.
- Code Patching: The player must open the Code Editor and manually rewrite the vulnerable query used in authenticate_user.
- Solution: The player replaces the insecure string concatenation ("SELECT *... username='" . $u . "'") with secure Prepared Statements (prepare, bind_param).
- Verification: Once the code is saved, the player runs test-login in the Terminal to verify the fix works and the vulnerability is closed.

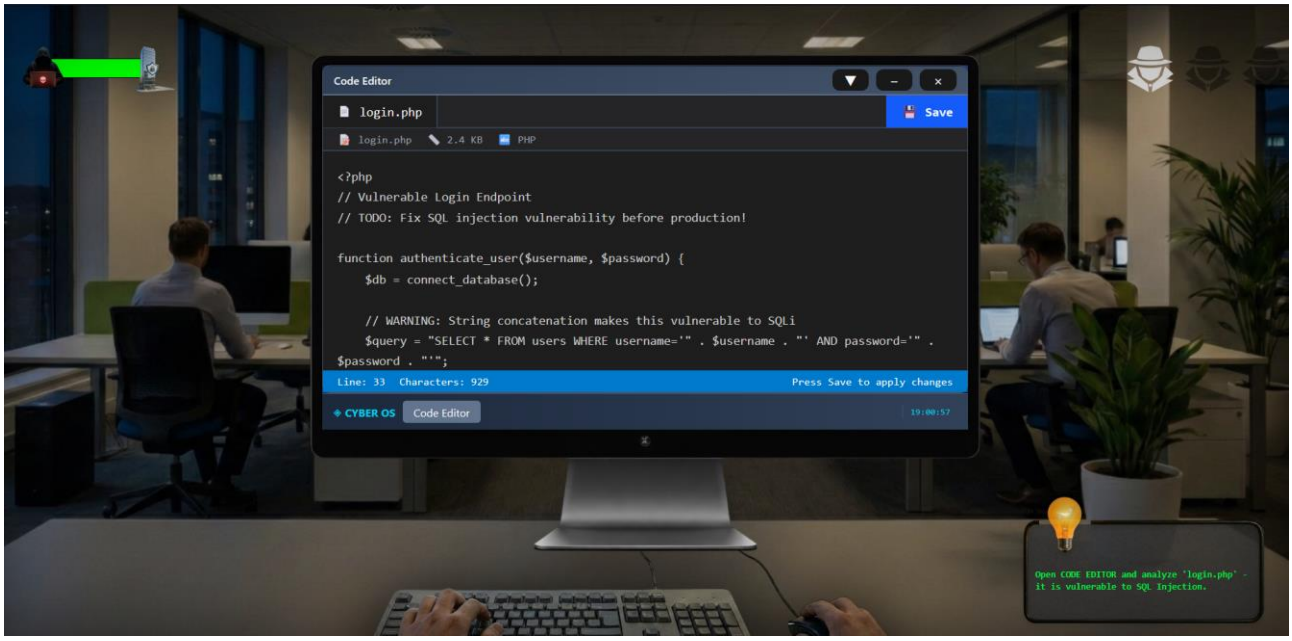

Figure 6 - Level 3

Level 4: The Script Injection (XSS Defense)

Objective: Secure the company's internal portal by mitigating Cross-Site Scripting (XSS) vulnerabilities in the comments section.

Storyboard: The employee announcement portal has been compromised. Malicious scripts injected into the comments section are executing in users' browsers, stealing session cookies. The player enters the SOC dashboard to find the portal defaced with popup alerts and broken layouts. SIEM logs confirm "active exploitation" via Stored XSS.

Mechanics:

- Identification: The player uses the Browser to observe the compromised page, noting malicious payloads like <script>alert('XSS')</script>. The Terminal command analyze-comments lists suspicious entries.
- Mitigation: Using the Terminal, the player must enable defense layers sequentially:
  - enable-sanitization: Strips dangerous HTML tags.
  - enable-csp: Activates Content Security Policy to block inline scripts.
  - enable-httponly: Protects cookies from being accessed by scripts.
- Resolution: Finally, the player must restart-app to apply all changes and verify the portal is secure.

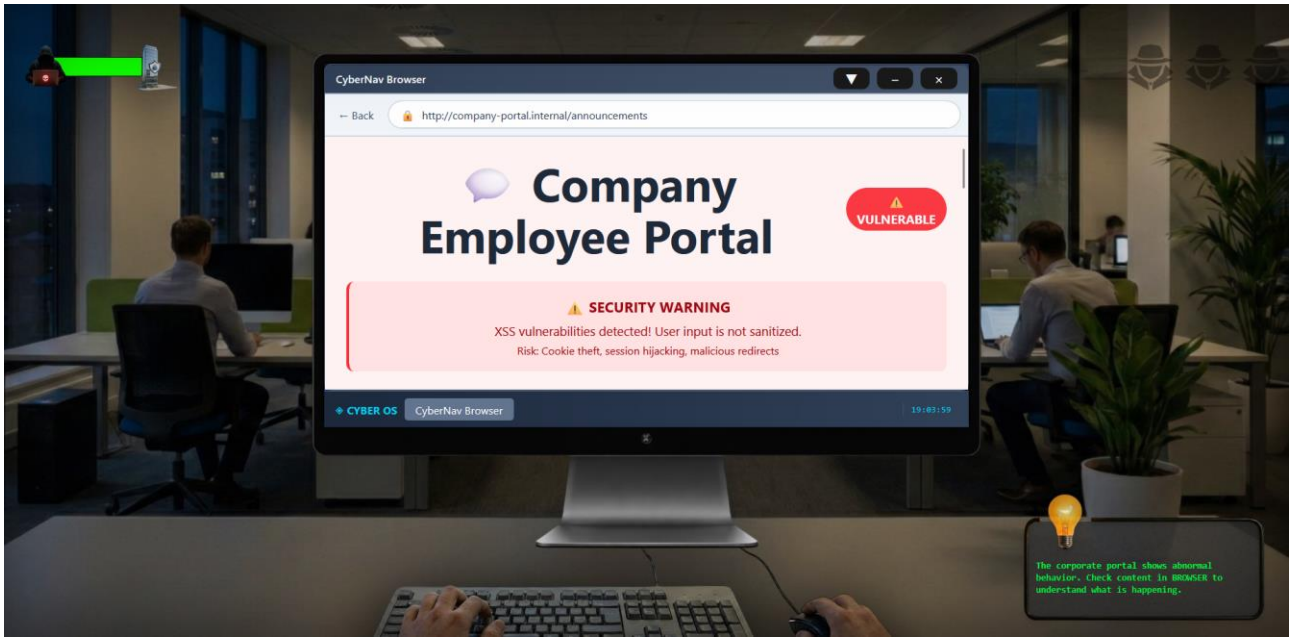

Figure 7 - Level 4

Level 5: The Poisoned Well (Cache Poisoning)

Objective: Purge corrupted cache entries and configure HTTP headers to prevent "cache poisoning" attacks.

Storyboard: Users report that the company homepage is displaying a "You have been hacked" message (<script>alert('Cache Poisoned!')</script>), yet the internal server shows the correct file. The CyberNav assistant explains that attackers have "poisoned" the intermediate cache by manipulating unkeyed inputs.

Mechanics:

- Investigation: The player uses the Browser to inspect the poisoned page and sees the X-Cache: HIT header. The Terminal command show-headers reveals that the Vary header is missing, allowing the cache to serve malicious content to legitimate users.
- Analysis: The player identifies X-Forwarded-Host as the vector used to inject the malicious content.
- Mitigation:
  - purge-cache: Clears the compromised data immediately.
  - enable-vary-header: Keys the cache based on the Host header.
  - set-cache-control no-store: Prevents caching of dynamic content.
  - restart-proxy: Applies the new secure configuration.
- Outcome: The site returns to normal, showing X-Cache: MISS (correctly fetching fresh content) or HIT with secure headers.

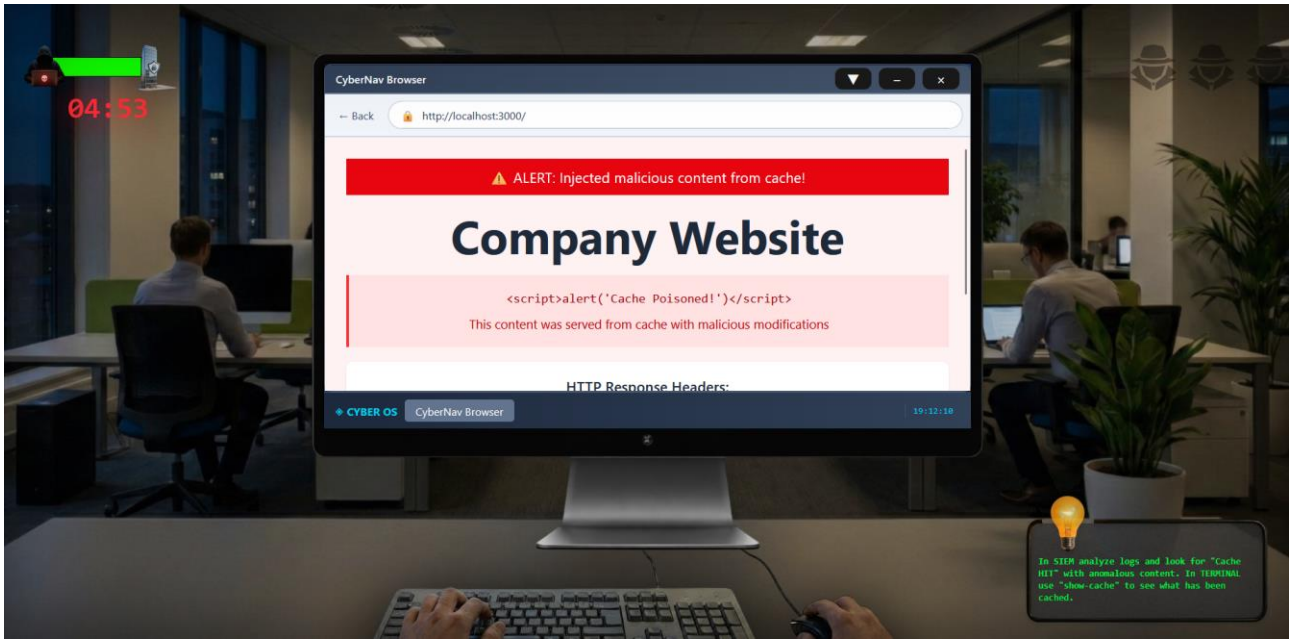

Figure 8 - Level 5

Level 6: The Silent Request (CSRF Defense)

Objective: Stop unauthorized fund transfers by implementing Anti-CSRF Tokens and SameSite cookie policies.

Storyboard: The finance department reports mysterious outbound transfers. The SIEM dashboard shows valid transaction requests originating from employees' specialized workstations, but occurring while they are browsing external sites. The CyberNav assistant suggests a Cross-Site Request Forgery (CSRF) attack, where a malicious site is piggybacking on the employees' active bank sessions.

Mechanics:

- Analysis: The player analyzes the "Referer" header in the SIEM logs to confirm that requests are originating from an external gambling or news site (evil-casino.com) rather than the internal dashboard.
- Mitigation: Using the Terminal, the player executes commands to secure the session:
  - enable-tokens: Injects unique cryptographic tokens into every form submission, invalidating the attacker's forged requests.
  - enable-samesite: Configures cookies to SameSite=Strict, preventing the browser from sending session cookies with cross-site requests.
- Outcome: The attack is neutralized, and the " Unauthorized Transactions" count in the log drops to zero.

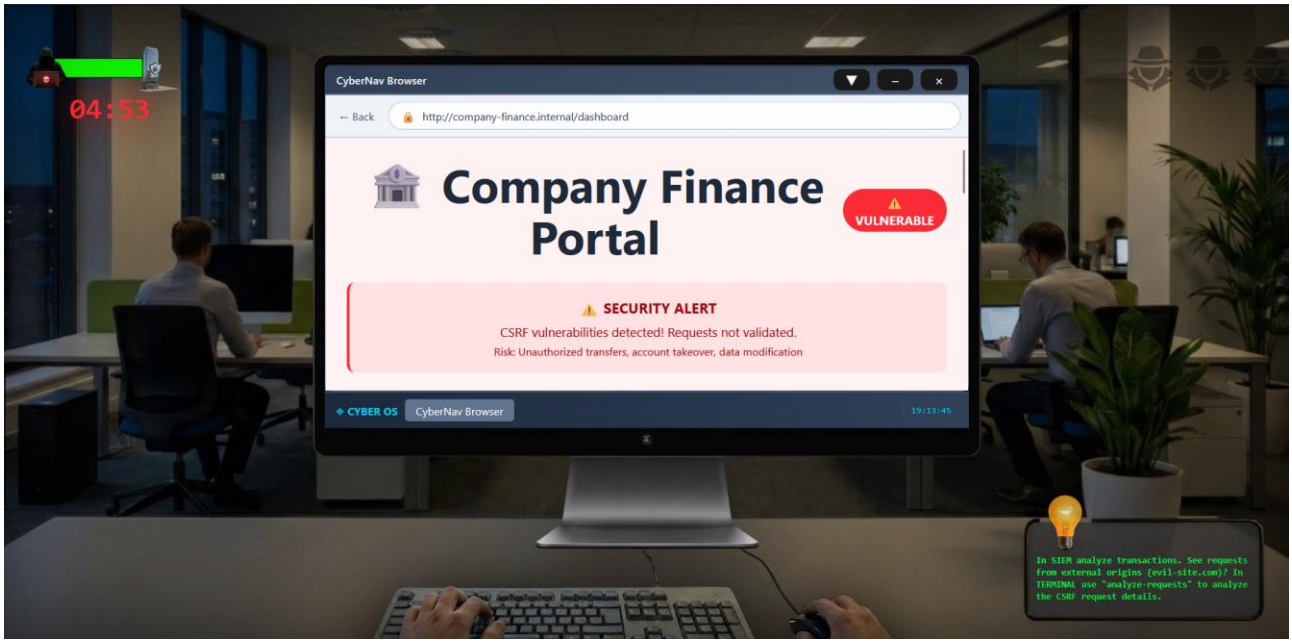

Figure 9 - Level 6

Level 7: The Binary Timebomb (Malware Analysis)

Objective: Analyze a suspicious executable file to identify its functionality and find the "kill switch" password to neutralize it.

Storyboard: A suspicious file named invoice_v2.exe has been quarantined by the antivirus. Sandbox reports indicate it's a "logic bomb" set to encrypt files if a specific condition isn't met. The source code is unavailable, so the player must work at the assembly level.

Mechanics:

- Static Analysis: The player opens the Reverse Engineering Viewer to examine the disassembled code (Assembly instructions like MOV, CMP, JE).
- Code Investigation: The player traces the execution flow to find the specific CMP (compare) instruction that checks for a hardcoded password.
- Solution: Identifying the hidden string (e.g., H4rdC0d3d!) stored in the data segment.
- Disarming: The player runs the neutralize-malware command in the Terminal, providing the extracted password to deactivate the logic bomb before the countdown triggers.

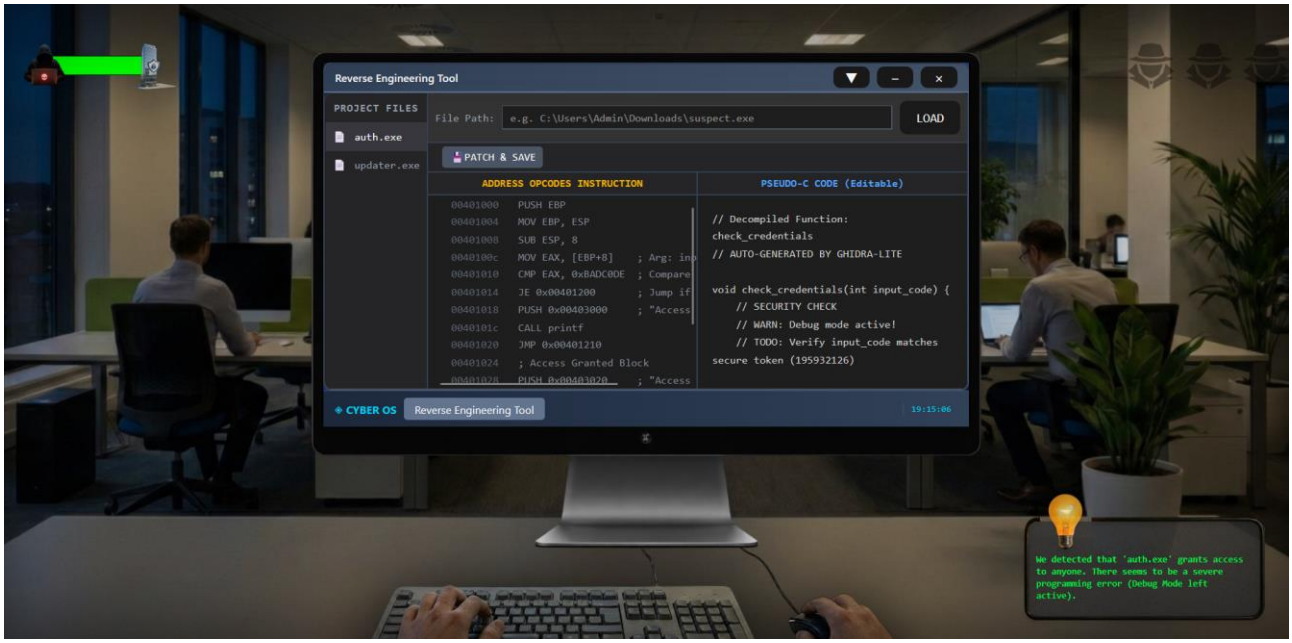

Figure 10 - Level 7

Level 8: The Final Standoff (Ransomware Response)

Objective: Detect the Command & Control (C2) communication, block the attack source, and decrypt critical files before the ransom timer expires.

Storyboard: A red "FILES ENCRYPTED" overlay seizes the workstation. A ransom note demands payment within minutes. The CyberNav advises against payment, urging immediate forensic action. The malware is still communicating with its server to finalize the encryption key exchange.

Mechanics:

- Network Analysis: The player opens the Packet Analyzer tool to sift through captured network traffic. They must identify the "heartbeat" signal a recurring connection to an external IP sending suspicious payloads.
- Containment: Once the C2 IP is identified (e.g., via unusual port usage or payload content), the player uses the Terminal command block-ip to cut the connection, preventing further encryption.
- Recovery: By inspecting the packet payloads, the player discovers the encryption key (often sent in plain text or base64 during the handshake). They enter this key into the Decryption Tool to unlock the company's files and restore operations.

Figure 11 - Level 8

Level 9: The Broken Lock (Cryptography Vulnerability)

Objective: Identify and replace weak encryption algorithms (DES) with secure modern standards (AES) to prevent sensitive data exposure.

Storyboard: An internal audit reveals that critical financial data is being transmitted using deprecated cryptographic standards. The SIEM dashboard flags suspicious decryption attempts, indicating that an attacker is brute-forcing the weak keys.

Mechanics:

- Analysis: The player uses the Terminal command analyze-crypto to scan the codebase and identifies the use of DES (Data Encryption Standard) with short 56-bit keys.
- Code Patching: The player enters the Code Editor to modify the encryption_module.py. The vulnerable DES implementation must be replaced with AES.new(key, AES.MODE_GCM).
- Key Management: The player must also update the key generation logic to use 256-bit keys instead of the weak legacy keys.
- Validation: Running verify-encryption confirms that the new algorithm is active and the "Data Exposure Risk" has dropped to 0%.

Figure 12 - Level 9

4 User test

User testing is a process whereby the interface and functionality of an application are tested by real users, who perform specific tasks under realistic conditions. In our case, this method was used to assess whether the levels of the serious game are appropriate and engaging for the target audience.

4.1 Description of the user test

The user test was conducted on a sample of eight people with different characteristics. Participants included both males and females, aged between 18 and 25, with varying levels of knowledge of cybersecurity and different educational backgrounds.

Each user was asked to complete a series of tasks at different levels of the serious game and, at the end, to fill out the SUS and QUIS questionnaires to assess usability and satisfaction. The specific tasks are detailed in the dedicated section.

4.2 Execution of the user test

Tasks were defined for each level of the serious game. The user test was structured in the following phases:

1. Introduction to the test – Participants were explained the purpose of the usability test, clarifying that the objective was to evaluate the interface and functions of the game and not to measure individual abilities.
2. Initial anonymous questionnaire – Users completed an anonymous questionnaire designed to collect general personal information, such as:
    a. Age
    b. Gender
    c. Educational qualification
    d. Knowledge of cybersecurity
3. Level 0 (preparation phase) – Participants completed the introductory level to familiarise themselves with the main elements of the serious game: the interface, the various tools that can be used, life, and tips.
4. Main level tests – Users then completed levels 1 to 9, performing the specific tasks for each level. The tasks included, for example, email analysis, network security management, vulnerable code review and interventions to mitigate attacks.
5. Evaluation questionnaires – At the end of the test, users completed the QUIS and SUS questionnaires to evaluate the usability, satisfaction and ease of use of the serious game.

Task	Level	Task during the preparation
Complete level 1	1	• Check all emails • Avoid letting your health drop to 0
Get all the stars of level 1	1	1. Check all emails. 2. Inspect at least one email header. 3. Open the anti-phishing guide in your browser or read the instructions
Complete level 2	2	• Block all malicious IP addresses • Ensure you do not exhaust your health
Get all the stars of level 2	2	1. Block all malicious IP addresses 2. Activate the firewall using the enable-firewall command 3. Activate rate limiting using the rate-limit command
Complete level 3	3	• Analyse the SIEM log that reports the SQL injection • Modify the vulnerable code to remove the SQLi vulnerability • Use prepared statements or input sanitisation • Test the login with the test-login command • Prevent health from dropping to 0

Get all the stars of level 3	3	1. Click on the critical SQL Injection log 2. Use the analyse-code command to examine the vulnerable code 3. Modify the login.php file and use test-login successfully only if the code now uses prepared statements
Complete level 4	4	• Enable basic protections via terminal: • Restart the application to apply the protections • Prevent health from reaching 0
Get all the stars of level 4	4	1. Use show-payload <ID> on a malicious comment 2. Use enable-escaping to enable content escaping 3. Use identify-xss to determine the type of attack
Complete level 5	5	• Clear the cache • Configure the cache correctly
Get all the stars of level 5	5	1. Cache emptied and proxy restarted. 2. Cache configured correctly and dynamic content not cached. 3. Advanced analysis: responsible header identified, correct cache key and Vary header enabled.
Complete level 6	6	• Identify the SQL injection attack in the SIEM logs • Analyse and correct the vulnerable code • Save the modified code • Test the mitigation • Prevent health from reaching 0
Get all the stars of level 6	6	1. SIEM log analysis 2. Code analysis 3. Proper mitigation
Complete level 7	7	• Receipt of SIEM alert • Patch of auth.exe • Execution of auth.exe • Patch of updater.exe • Execution of updater.exe • Health management
Get all the stars of level 7	7	1. Corrected auth.exe patch 2. Corrected updater.exe patch 3. Code clean-up and critical comments
Complete level 8	8	• Briefing • Infected system • Analysis and response • Ransomware unlocking • Health management

Get all the stars of level 8	8	1. Level completion 2. Fast time (<150 seconds) 3. No errors
Complete level 9	9	• Suspicious email analysis • SIEM and network analysis • Code review and correction • Server build • Health management
Get all the stars of level 9	9	1. Detect MITM attack by reading email and SIEM logs 2. Fix DES → AES-256 vulnerability in code 3. Complete build and verify status on terminal

Table 6 – Task of Test

4.3 Type of testing users

The game was tested by eight different types of users.

User	Sex	Age	Knowledge of Cybersecurity	Educational Level
User 1	Male	19	Intermediate	High School
User 2	Female	20	Intermediate	High School
User 3	Male	24	Advanced	Master's Degree
User 4	Female	21	Intermediate	Bachelor's Degree
User 5	Female	18	Basic	High School
User 6	Male	25	Advanced	Master's Degree
User 7	Male	22	Advanced	Bachelor's Degree
User 8	Male	22	Basic	Bachelor's Degree

Table 7 - Type of testing users

4.4 Analysis of the user test

In this phase, all user activity results are collected. The table summarizes the outcomes of the task, where "R" indicates the result, Success (S), Partial (P), or Failed (F), and "T" represents the time taken to complete the task.

Task	Users															
	1		2		3		4		5		6		7		8	
	R	T	R	T	R	T	R	T	R	T	R	T	R	T	R	T
1	S	45	S	50	S	25	S	48	S	90	S	22	S	28	S	95
2	S	60	S	65	S	30	S	66	P	120	S	28	S	35	P	130
3	S	55	S	60	S	35	S	58	S	110	S	32	S	38	S	115
4	S	70	S	90	S	40	S	75	F	150	S	38	S	42	F	160
5	S	120	S	130	S	60	S	115	P	240	S	55	S	65	P	250
6	P	150	P	160	S	75	S	140	F	300	S	65	S	70	F	310
7	S	110	S	115	S	55	S	100	P	220	S	50	S	58	P	230
8	S	110	S	115	S	55	S	100	P	220	S	50	S	58	P	230
9	P	180	P	190	S	90	S	170	F	350	S	85	S	95	F	360
10	F	200	F	210	S	110	P	190	F	400	S	100	S	115	F	410
11	S	140	S	150	S	70	S	135	P	280	S	65	S	75	P	290
12	S	160	P	180	S	85	S	155	F	300	S	80	S	90	F	320
13	S	130	S	140	S	65	S	125	S	260	S	60	S	70	S	270
14	P	170	S	165	S	80	P	150	F	320	S	75	S	65	F	260
15	S	100	S	110	S	50	S	105	F	200	S	45	S	55	F	210
16	S	145	P	160	S	60	S	140	F	250	S	55	S	65	R	260
17	P	220	P	230	S	120	S	210	P	400	S	100	S	130	F	420
18	F	250	F	260	S	150	P	240	F	450	S	130	S	160	F	460

Table 8 - Result of the Test

Total of tasks: 144

Total success result: 100

Partial success result: 24

Failed result: 22

Rate of total success task: 77.78%

Analyzing the activities in the tests, it can be seen that the remaining percentage rate is mainly about the problem of **Task 10** and **Task 16**.

The failures in **Task 10** happen because the advanced configuration of the cache (Level 5) requires specific technical knowledge that users with a "Basic" profile did not possess. These problems aren't solvable by intuition alone because the correct setup of the headers is a strict part of the cybersecurity simulation.

The problem of **Task 16** is related to the time constraint. This task required completing the level in less than 150 seconds, but some users failed because they spent too much time analyzing the infected system in the terminal, exceeding the limit allowed to get the stars.

Finally, the partial results in other tasks happen because some users focused only on finishing the level to proceed, neglecting the secondary optional steps required to obtain all the stars.

4.4.1 QUIS

QUIS Questions:

Part 1: Overall Reaction to the Software

1. Overall, how would you rate the experience? (Terrible / Wonderful)
2. How was the difficulty level? (Difficult / Easy)
3. How did you feel while playing? (Frustrating / Satisfying)
4. Was the game engaging? (Dull / Stimulating)
5. Did the system feel rigid or flexible? (Rigid / Flexible)
6. Was the flow of the game cohesive? (Fragmented / Integrated)

Part 2: Screen and Interface (Terminal & GUI)

7. Characters on the computer screen were: (Hard to read / Easy to read)
8. Organization of information on the interface was: (Confusing / Clear)
9. Sequence of screens and windows was: (Confusing / Clear)
10. Use of colors and highlighting (e.g., in code) was: (Poor / Good)

Part 3: Terminology and System Information

11. Use of cybersecurity terminology throughout the game: (Inconsistent / Consistent)
12. Computer terms (IP, SQL, Firewall) were: (Vague / Precise)
13. Position of messages and prompts on screen: (Inconsistent / Consistent)
14. Error messages provided by the terminal: (Unhelpful / Helpful)

Part 4: Learning and Capabilities

15. Learning to operate the game mechanics: (Difficult / Easy)
16. Exploring new features by trial and error: (Discouraging / Encouraging)
17. Remembering names and use of commands: (Difficult / Easy)
18. Feedback on task completion (Success/Fail): (Unclear / Clear)

Questions	**Values of QUIS**							
	User 1	**User 2**	**User 3**	**User 4**	**User 5**	**User 6**	**User 7**	**User 8**
1	7	7	9	7	4	9	8	4
2	6	7	8	6	3	9	8	3
3	7	6	9	7	4	9	9	4
4	7	7	8	8	5	9	8	5
5	6	6	9	6	3	8	8	3

6	7	7	9	7	4	9	9	4
7	8	7	8	7	5	8	8	5
8	7	8	9	7	4	9	9	4
9	7	7	8	8	5	8	8	5
10	6	6	9	6	3	9	8	3
11	7	7	9	7	4	9	9	4
12	6	7	8	6	3	8	8	2
13	7	6	9	7	4	9	9	4
14	7	7	8	8	5	9	8	5
15	6	6	9	6	3	8	8	3
16	7	7	9	7	4	9	9	4
17	7	7	8	7	5	8	8	5
18	6	6	9	6	3	9	8	3

Table 9 - QUIS Results

Analyzing the answers in the QUIS questionnaire, it can be noticed that the users, on average, have found the application in line with their expectations despite the difficulties encountered.

The data reveals a clear distinction in satisfaction based on the specific sections of the questionnaire. The questions regarding the interface and screen layout (first half of the table) received generally positive ratings from all participants, indicating that the visual design is intuitive and legible.

However, the scores in the terminology and learning sections (second half of the table) show a dependency on the user's background. While advanced users rated the system highly (scores of 8-9) for its technical accuracy, users with basic knowledge expressed lower satisfaction (scores of 3-5). This suggests that the perceived difficulty was not due to the application's usability, but rather to the complexity of the cybersecurity concepts required to play.

4.4.2 SUS

SUS Questions:

1. I think that I would like to use this serious game frequently.
2. I found the serious game unnecessarily complex.
3. I thought the serious game was easy to use.
4. I think that I would need the support of a technical person to be able to use this serious game.
5. I found the various functions in this serious game were well integrated.
6. I thought there was too much inconsistency in this serious game.
7. I would imagine that most people would learn to use this serious game very quickly.
8. I found the serious game very cumbersome to use.
9. I felt very confident using the serious game.
10. I needed to learn a lot of things before I could get going with this serious game.

Questions	**Values of SUS**							
	User 1	**User 2**	**User 3**	**User 4**	**User 5**	**User 6**	**User 7**	**User 8**

1	4	4	5	4	2	5	5	2
2	2	2	1	2	4	1	1	4
3	4	4	5	4	2	5	5	2
4	2	2	1	2	4	1	2	4
5	4	3	5	4	3	5	4	3
6	2	2	1	2	4	1	1	4
7	4	4	5	3	2	5	5	2
8	2	2	1	2	4	1	1	4
9	4	4	5	4	3	5	5	2
10	2	2	1	2	5	1	1	4
Result	75	72.5	95	72.5	37.5	95	92.5	37.5

Table 10 - SUS Results

The SUS score for every user is calculated in the following mode:

- For every odd question (1-3-5-7-9) it is calculated as: value of scale – 1
- For every even question (2-4-6-8-10) it is calculated as: 5 – value of scale
- This adds up the score of each question
- The result of the sum is multiplied by 2.5 to obtain the value of the SUS. The result obtained fluctuates between 0 and 100.

The application can be considered usable when the value exceeds 68. A value of more than 80 is desirable to consider the application very usable.

In general, the users define that the serious game is usable. Users with intermediate and advanced knowledge (Users 1, 2, 3, 4, 6, 7) consistently scored the system above the 68 threshold, with peaks above 90 for the experts, indicating an excellent user experience.

The choices of User 5 and User 8, however, resulted in scores below the acceptance limit. This is based on the problems encountered during the execution of the advanced technical tasks (such as the cache configuration), which influenced their perception of the system's complexity.

5 Conclusion and future developments

5.1 Conclusion

CyberShield Command successfully translates complex cybersecurity concepts into an engaging, interactive experience. By placing the player in the role of a SOC Analyst, the game bridges the gap between theoretical knowledge and practical application. The scenario-based approach allows players to face realistic threats, from Phishing and SQL Injection to advanced Ransomware attacks, in a safe, simulated environment.

The current prototype demonstrates the viability of "gamified training," proving that technical skills like log analysis, code patching, and incident response can be taught effectively through gameplay mechanics. The combination of a high-fidelity interface, time-critical decision-making, and immediate feedback loops creates a compelling learning tool suitable for both students and aspiring security professionals.

5.2 Future Development

While the core gameplay loop is complete, the modular architecture allows for significant expansion. Future development phases will focus on deepening the simulation and broadening accessibility.

5.2.1 Advanced Modules

Red Team Mode: A new game mode where players take on the role of the attacker, learning offensive techniques (Penetration Testing) to better understand defense statistics.

Forensics Lab: A dedicated environment for post-incident analysis, allowing players to analyze memory dumps and hard drive images to find evidence of APT (Advanced Persistent Threat) activity.

Multiplayer Co-op: A cooperative mode where one player manages the SIEM (Blue Team) while another handles Incident Response, requiring real-time communication to mitigate coordinated attacks.

5.2.2 Technical Enhancements

Persistent User Profiles: Implementation of a full backend authentication system (MySQL/PostgreSQL) to save progress, unlockable achievements, and performance metrics across devices.

Global Leaderboards: Competitive ranking based on "Time to Mitigation" and "Accuracy Score" for each level, encouraging replayability and mastery.

Mobile Optimization: Adaptation of the interface for tablets and large smartphones, effectively turning mobile devices into portable "incident response terminals."

5.2.3 Content Expansion

Dynamic Scenarios: Replacing scripted levels with procedurally generated attacks (randomized IPs, payloads, and timing) to provide endless replay value.

Certification Alignment: Content updates tailored to cover specific topics from industry certifications like CompTIA Security+ or Certified Ethical Hacker (CEH), making the game a valid study aid.

6 References

Link GitHub: CyberShield Command

Link Canva: Presentation

Link Online Game: https://cybershield.mntcrl.it/