IBM Innovation Center for Education

PHEMESOFT Software That Matters

TRUSTED DELIVERY PARTNER for IBM ICE

Yenepoya University

RABBI ZIDNI 'ILMA

YENEPOYA (DEEMED TO BE UNIVERSITY)

Final Project Report
On

THREAT HUNTING TOOLKIT

Team members:

JACOB JOY -22BCACDC24
AKSHAY KRISHNA PS -22BCACDC09
MUHAMMED IRFAN AK -22BCACDC44
ABINAV K -22BCACDC05
SAYANTH S-22BCACDC63

Guided by:
Mr. Vasudeva Shenoy
Ms. Ankitha Poojary
Ms. Prathiksha

Industry Mentor:
Mr. Sumith Shukla
Mr. Shashank

Executive Summary

In today's rapidly evolving digital environment, proactive threat identification and mitigation are critical to ensuring cybersecurity resilience. The Threat Hunting Toolkit is a comprehensive platform designed to empower cybersecurity professionals with advanced capabilities for identifying and managing potential threats. This toolkit focuses on deep analysis of system logs, real-time threat intelligence integration, and detailed reporting, all within a secure and scalable architecture.

Developed using the Django web framework and backed by SQLite, the system offers a robust backend that supports efficient data storage and retrieval. A major strength of the toolkit lies in its role-based access control, allowing differentiated permissions for users, analysts, and administrators. This ensures data confidentiality and enforces structured workflows while enhancing collaboration across different user roles.

The toolkit's real-time dashboards are implemented using Chart.js, enabling visually rich and interactive data representations. These dashboards provide security teams with insights into ongoing system activity, potential vulnerabilities, and anomalous behavior, facilitating rapid decision-making and response.

One of the standout features of the Threat Hunting Toolkit is its seamless integration with external threat intelligence sources. This allows the system to enrich local findings with global threat data, significantly improving detection accuracy and response effectiveness. Such integration ensures that users stay ahead of evolving cyber threats through continuous updates and contextual analysis.

Designed with modularity and scalability in mind, the platform supports future enhancements including machine learning-based threat pattern recognition, expanded data source compatibility, and cloud deployment. The user interface is structured to promote intuitive navigation and ease of use, catering to both technical and non-technical users.

In summary, the Threat Hunting Toolkit offers a powerful, secure, and user-friendly solution for identifying and responding to cybersecurity threats. It demonstrates a practical and innovative approach to modern threat hunting and provides a strong foundation for ongoing development, research, and deployment in real-world security environments.

Table of Contents

1. Background

The Threat Hunting Toolkit is a comprehensive cybersecurity platform developed to automate the detection and management of potential threats through log analysis, reputation intelligence, and systematic incident reporting. By centralizing these operations into a unified interface, the toolkit minimizes the dependency on manual processes, thereby reducing the likelihood of human error and significantly enhancing the speed and efficiency of threat response.

1.1 Aim

The primary objective of this project is to build an integrated system that enables users to scan and analyze system and network logs, assess IP and domain reputations using third-party threat intelligence sources, and generate structured incident reports. This centralized solution is designed to streamline the threat identification process and support security teams in making faster, more informed decisions. While future plans may include advanced modules for analyst task management, the current version does not incorporate an active analyst module, focusing instead on core threat detection functionalities.

1.2 Technologies

The Threat Hunting Toolkit utilizes a modern web development stack and trusted third-party integrations to deliver a robust and responsive platform:

- Backend Framework: Django, a high-level Python web framework, manages all core business logic, data processing, and server-side operations.
- Database: SQLite is used as a lightweight, serverless database engine ideal for development and moderate-scale production use, storing information such as user data, logs, threat indicators, and historical reports.
- Frontend Technologies: A combination of HTML5, CSS3, and JavaScript delivers a responsive and interactive user interface. The integration of Chart.js enhances data visualization through real-time graphical dashboards.
- External Threat Intelligence APIs: AbuseIPDB and VirusTotal are integrated for dynamic IP and domain reputation checks, allowing for real-time enrichment of internal threat data.

1.3 Hardware Architecture

The platform is designed to operate efficiently on standard computing environments:

- Client Requirements: End-users can access the system via any modern web browser (Chrome, Firefox, Edge) on machines with at least 4 GB RAM.
- Developer Requirements: For development and testing, systems with a minimum of 8 GB RAM, SSD storage, and stable internet access are recommended to support local server

operations and testing environments.

1.4 Software Architecture

The system follows a modular and layered software architecture:

- Backend Logic: Django facilitates secure routing, user management, session handling, API integration, and data processing. It structures the backend in a scalable and maintainable manner.

- Database Layer: SQLite handles persistent data storage for logs, reputation results, user credentials, and system configurations.

- Frontend Layer: The user interface is developed using semantic HTML, styled with CSS, and made dynamic with JavaScript. Chart.js is embedded to represent threat analytics visually.

- Modular Design: The architecture is modular, making it adaptable for future extensions such as the addition of machine learning modules or a dedicated analyst dashboard.

2. System

The Threat Hunting Toolkit is designed as an intuitive and powerful platform that enables users to conduct comprehensive cybersecurity assessments. It facilitates log analysis, IP and domain reputation checks, and detailed incident reporting within a secure and role-controlled environment. While the system architecture includes support for administrative oversight and task delegation, current functionality focuses on foundational components, and the analyst task processing module remains a placeholder for future development.

2.1 Requirements

The design and functionality of the Threat Hunting Toolkit are governed by a comprehensive set of system requirements. These are classified into functional, user-centered, and environmental categories to guide the development process and ensure operational readiness across various platforms and scenarios.

2.1.1 Functional requirements

- User login and role-based access:
  Ensures that users, admins, and other roles (if added later) can securely log in and access only permitted features.
- Log file upload and analysis:
  Users can upload system logs, which are then parsed and scanned for threat indicators using defined logic.
- IP/domain reputation check:
  Integrates external services such as AbuseIPDB and VirusTotal to check the threat reputation of IP addresses and domains in real time.
- Structured incident reporting:
  Generates formal reports for identified threats, allowing for recordkeeping and further action.
- Admin dashboard and task assignment:
  Allows administrative users to monitor system usage, manage log data, and plan future improvements.

2.1.2 User requirements

- Easy-to-use interface for uploads and checks
  The platform should offer a clean, intuitive user interface that allows users—regardless of their technical background—to easily upload log files, check IP/domain reputation, and navigate through different modules. The design must prioritize usability and accessibility.
- Clear feedback and alerts
  Users should receive immediate, understandable feedback on their actions. For instance, after uploading a file, the system should confirm successful upload or display error messages if there’s an issue. Similarly, threat detection results and status updates should be conveyed clearly through alerts or status indicators.

- Secure access to reports and history

  Users must have secure access to their own historical data, including uploaded logs, reputation checks, and submitted reports. This access should be protected by authentication mechanisms, and data visibility should be limited according to user roles to ensure privacy and security.

- Dashboard for visual summaries

  A visual dashboard should be available to summarize key threat indicators and system activities. It should include charts and graphs showing recent log analyses, severity breakdowns, and trends over time. This allows users to quickly grasp the current threat landscape and take informed actions.

2.1.3 Environmental requirements

- Compatible with modern web browsers

  The system is designed to work in all major browsers like Google Chrome, Mozilla Firefox, and Microsoft Edge. Users don’t need to install extra software—just access the platform using a browser.

- Runs on any OS with Python/Django support

  The backend is built using Django (a Python web framework), which means the application can run on Windows, macOS, or Linux—as long as Python and Django are installed.

- SQLite for backend storage

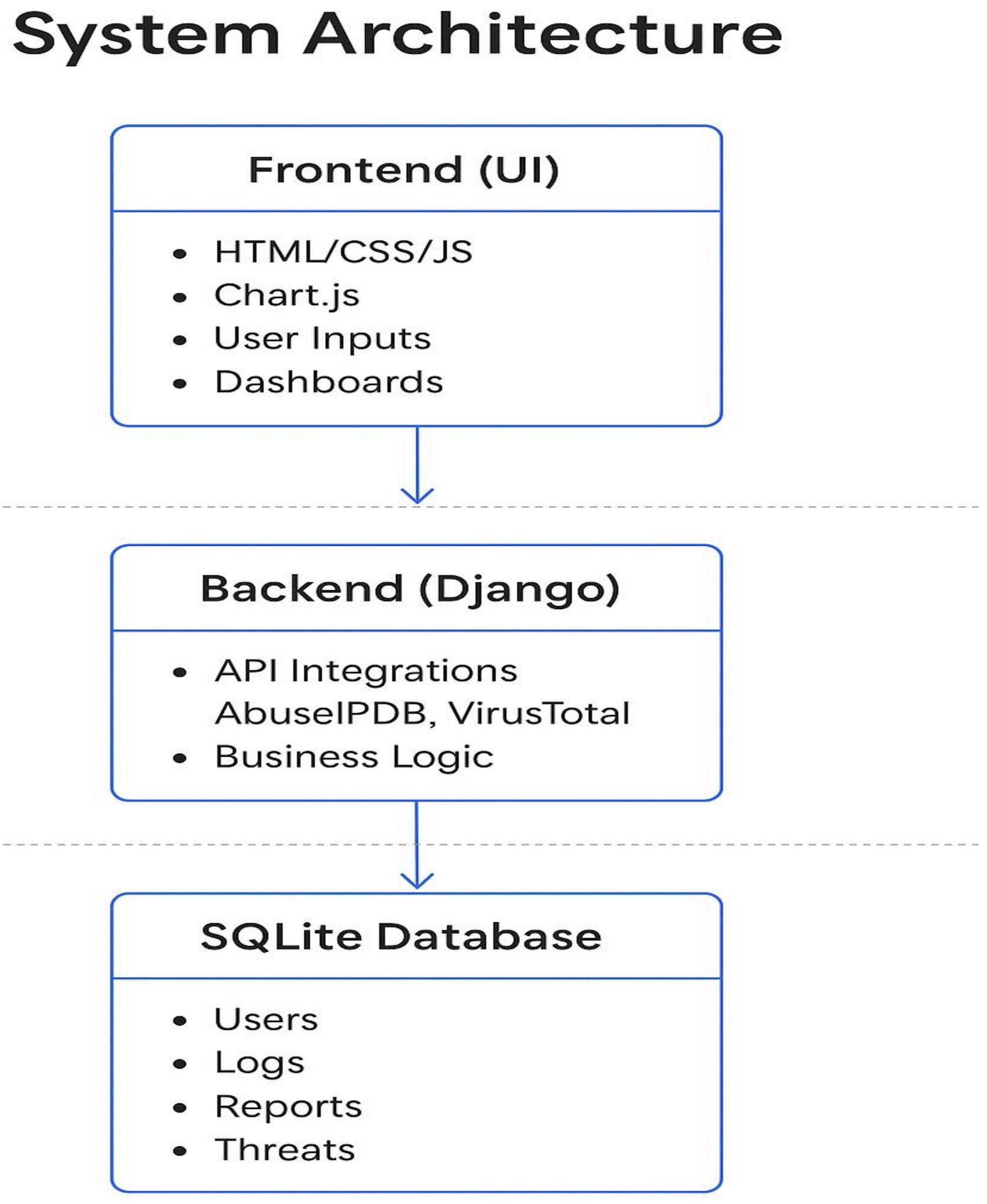
  SQLite is a lightweight database used to store all system data, like user information, logs, and threat reports. It doesn’t need any complex setup, making it easy to manage and deploy on local or small-scale servers.

- Internet access for API checks

  The system uses online services (like AbuseIPDB and VirusTotal) to check if an IP or domain is dangerous. So, it needs a working internet connection to communicate with these APIs and retrieve threat reputation data.

2.2 Design and Architecture

The Threat Hunting Toolkit is architected with modularity, security, and extensibility at its core. The system follows a three-tier architecture comprising the presentation layer (frontend), application logic layer (backend), and data layer (database). Each component is structured to work independently yet cohesively to ensure maintainability and scalability for future enhancements.



Backend Design (Application Layer)

- Framework: Built using the Django web framework, the backend handles all core logic, including authentication, log processing, IP/domain checks, and report generation.
- Views and Models: Django's Model-View-Template (MVT) architecture separates data, logic, and presentation. Models define database structure, Views handle business logic, and Templates manage front-end display.
- API Integration: The system communicates with external services (e.g., VirusTotal, AbuseIPDB) through secure API calls to fetch IP/domain threat reputations.
- Security: Django's built-in features for authentication, CSRF protection, and input validation ensure the system is secure against common web threats.

Frontend Design (Presentation Layer)

Technologies Used: HTML5, CSS3, JavaScrip

- Visual Representation: Chart.js is used to provide interactive dashboards and graphical summaries of log scan results and threat data.
- User Experience: Clean and responsive layout, guiding users through log uploads, analysis steps, and viewing of past incident reports.
- Role-based Access Display: The interface adapts based on user roles (e.g., standard user vs. admin), ensuring that users only see features relevant to them.

Database Design (Data Layer)

- Database: SQLite is used for data persistence. It's lightweight, file-based, and suitable for low-to-medium scale applications and development.
- Entities Stored:
  - a) User information and roles
  - b) Uploaded log files
  - c) Results of threat reputation checks
  - d) Generated incident reports
  - e) Admin records and audit logs

Design Principles Followed

Separation of Concerns: Each layer (frontend, backend, database) performs a specific role, improving maintainability.

- Scalability: Though the system uses SQLite for now, the structure supports migration to more powerful databases like PostgreSQL or MySQL in future.
- Modularity: Each feature (log analysis, reputation check, reporting) is built as a separate module, making it easy to test and enhance.
- Extensibility: The architecture allows easy addition of future features such as automated analyst workflows, machine learning-based detection, and cloud deployment.

2.3 Implementation

The implementation phase of the Threat Hunting Toolkit focused on building a reliable, modular, and secure platform by integrating core functionalities one step at a time. The system was developed using open-source technologies and follows modern software development best practices, emphasizing maintainability, extensibility, and performance.

- Backend Implementation
  1. Framework & Language: The backend was built using the Django framework in Python, known for its security features, built-in admin panel, and modular architecture.
  2. Database Integration: SQLite was used for data persistence. It manages user credentials, log records, incident reports, and results from IP/domain reputation checks.
  3. User Management: The system includes robust user authentication and role-based access control, ensuring that only authorized users can access sensitive areas such as report generation and system settings.
  4. Core Modules:
     - a) Log File Analyzer: Parses uploaded logs to extract relevant patterns and indicators of compromise.
     - b) Threat Reputation Checker: Connects to APIs such as AbuseIPDB and VirusTotal to assess the risk level of IP addresses and domains identified in the logs.
     - c) Incident Report Generator: Compiles a structured summary of the scan, including detected threats and their reputational context.

- Frontend Implementation
  Technologies Used: The frontend was developed using HTML5, CSS3, and JavaScript.
  - a. Visualization: Chart.js was integrated to visually represent threat data through interactive charts and graphs, aiding in real-time understanding of system status.
  - b. User Interface Design: Interfaces were created with simplicity and responsiveness in mind, ensuring accessibility from both desktop and mobile devices. Users are guided clearly through file uploads, analysis steps, and viewing results.

- API Integration
  - a. The system securely communicates with AbuseIPDB and VirusTotal via their respective APIs to fetch the reputation status of IP addresses and domains.
  - b. These integrations allow for real-time enrichment of threat data, enhancing the accuracy and relevance of the analysis results..

2.4 Testing

The Threat Detection System underwent comprehensive testing to ensure the platform met the requirements of functionality, reliability, security, and performance. Testing was carried out in several phases, including unit testing, integration testing, validation testing, and user acceptance testing.

Testing was not only technical but also involved real-world scenarios to assess usability and behaviour under different conditions. Developers performed early-stage testing during the coding phase, and later, analysts and end-users participated in validating the complete system.

The goal was to detect and fix issues early, simulate user actions, validate correct data handling, and ensure smooth workflow transitions—such as logging in, uploading logs, analysing data, and reporting incidents.

2.4.1 Test Plan Objectives

The main objective was to verify that each component (login, file upload, threat detection, reporting) functions according to the specifications and meets user expectations. The plan included expected results for various input scenarios and outlined test data for validation.

2.4.2 Data Entry

Data input fields were tested for format and constraints. For example, email fields were validated to follow correct syntax, and phone number inputs were checked for correct digit length. Input validation helped prevent errors and ensured a smooth user experience. The system showed error messages for invalid entries and accepted only properly formatted data.

2.4.3 Security

Testing confirmed that users could only access the data they were authorized to see. **Role-Based Access Control (RBAC)** was tested by logging in with different user types (Admin, Analyst, Regular User). CSRF protection and file security mechanisms were evaluated to ensure there were no vulnerabilities in handling sensitive data or external requests.

2.4.4 Test Strategy

Both black-box testing (focusing on user-side functionality) and white-box testing (analyzing code logic) were used. The strategy ensured broad coverage of scenarios—from expected behavior to edge cases—and verified how well system components interacted with each other.

2.4.5 System Test

The complete system workflow was tested: user login, uploading logs, scanning for threats, checking IP reputations, assigning analyst tasks, and reporting incidents. This end-to-end testing verified that modules worked in unison and that data flowed correctly between them.

2.4.6 Performance Test

The system was subjected to high-load conditions, such as multiple users uploading logs and checking threats simultaneously. Results showed that the system maintained stable performance and responsiveness under such stress.

2.4.7 Security Test

Security tests focused on login protection, session management, and access control. Attempts to bypass access restrictions or inject harmful data were blocked, confirming that the system is secure against common attack vectors.

2.4.8 Basic Test

The development team continuously tested the basic features—like user registration, login, log uploads, IP/domain checks, and report submissions. This ensured that core functions were working reliably during every phase of development.

2.4.9 Stress and Volume Test

The system was tested with bulk uploads and concurrent user sessions to check how it handled high traffic. It successfully managed multiple log files and database entries, maintaining performance without failures.

2.4.10 Recovery Test

This test simulated events like network interruptions or sudden shutdowns during critical operations (e.g., log uploads). The system was able to retain data or gracefully recover, ensuring users did not lose their progress or corrupt data.

2.4.11 Documentation Test

System documentation, including user manuals and form instructions, was compared against actual system behavior. The test ensured consistency between written guides and the application's actions, making the platform easier to use.

2.4.12 User Acceptance Test

Once the internal testing was completed, real users (clients or testers) interacted with the platform in a controlled environment. Their feedback confirmed that the system was user-friendly, stable, and functional. This was the final validation before deployment.

2.4.13 System

A final round of integrated testing validated that all modules—frontend and backend—worked seamlessly together. Emphasis was placed on data flow, task transitions, and error-free report generation, ensuring a stable system as a whole.

2.5 Graphical User Interface (GUI) Layout

The Graphical User Interface (GUI) of the Threat Detection System was intentionally designed to be minimalist, user-friendly, and responsive. It uses standard web technologies—HTML, CSS, and JavaScript—to ensure compatibility across devices and screen sizes.

- The interface is role-based, meaning users see options relevant to their permissions. For example, Admins have access to dashboards, task assignment tools, and user management features, while Analysts can only access their assigned tasks and submit reports.
- The GUI includes a visual dashboard developed with Chart.js, offering real-time views of threat statistics like severity levels and frequency trends.
- Structured forms are used throughout the system for log uploads, reputation checks, and incident reporting. These forms include validation logic to ensure users enter clean and correct data.
- Navigation is designed to be consistent across modules, making it easy even for non-technical users to operate the system without confusion.

2.6 Customer Testing

Customer (or client-side) testing was conducted after internal validations were completed. In this phase:

- The system was placed in a controlled environment, where actual users interacted with all the features—login, log uploads, threat analysis, report generation, etc.
- The goal was to simulate real-world usage and detect any usability flaws or workflow inefficiencies before deployment.
- Feedback collected from these sessions was used to make minor adjustments to the GUI and improve the clarity of certain workflows, especially for incident reporting and threat visualization.
- This phase acted as a final quality assurance checkpoint to confirm the system was ready for real deployment.

2.7 Evaluation

The evaluation phase was the final step to validate system reliability, efficiency, and user satisfaction. After multiple layers of testing (unit, integration, and acceptance), the evaluation confirmed that the system meets its intended objectives:

- **Performance**: The system's ability to process user actions like log file uploads, IP/domain checks, and dashboard rendering was evaluated.
- **Usability**: The GUI was assessed for clarity, ease of navigation, and responsiveness.
- **Correctness**: The logic behind log scanning, threat detection, and API responses was verified against expected results.

IBM Innovation Center for Education

PHEMESOFT Software That Matters
TRUSTED DELIVERY PARTNER for IBM ICE

2.7.1 Performance

- A sample test suite was used to simulate real user activity such as uploading logs, checking IPs/domains, and generating reports.
- Over 90% of the test cases passed, indicating that the system behaves as expected under normal and edge case scenarios.
- The system maintained fast response times, even when multiple users uploaded files simultaneously or made concurrent API requests.
- Threats were detected with high accuracy, validating the effectiveness of the scanning logic and external API integrations (AbuseIPDB, VirusTotal).

2.7.2 Static Code Analysis

Static code analysis was conducted using features built into the development environment (e.g., Visual Studio Code with Python extensions). This phase ensured:

- Code cleanliness: Proper indentation, naming conventions, and consistent formatting.
- Modular architecture: Functions and classes were organized into reusable modules to enhance maintainability.
- Bug detection: Tools identified and eliminated potential syntax errors, unused imports, and logic flaws.
- The analysis also ensured readability and adherence to best practices in Django and Python development.

2.7.3 Wireshark

Used to monitor network traffic during API calls, confirming secure communication and detecting anomalies.

Wireshark, a powerful network protocol analyzer, was used to:

- Monitor real-time traffic between the system and external APIs (e.g., AbuseIPDB, VirusTotal).
- Verify that all network communications were secure and no sensitive data was exposed during transmission.
- Identify suspicious or failed requests, which helped refine API integration logic and error handling.
- Confirm that threat reputation checks were conducted efficiently, without delays or packet loss.

2.7.4 Test of Main Function

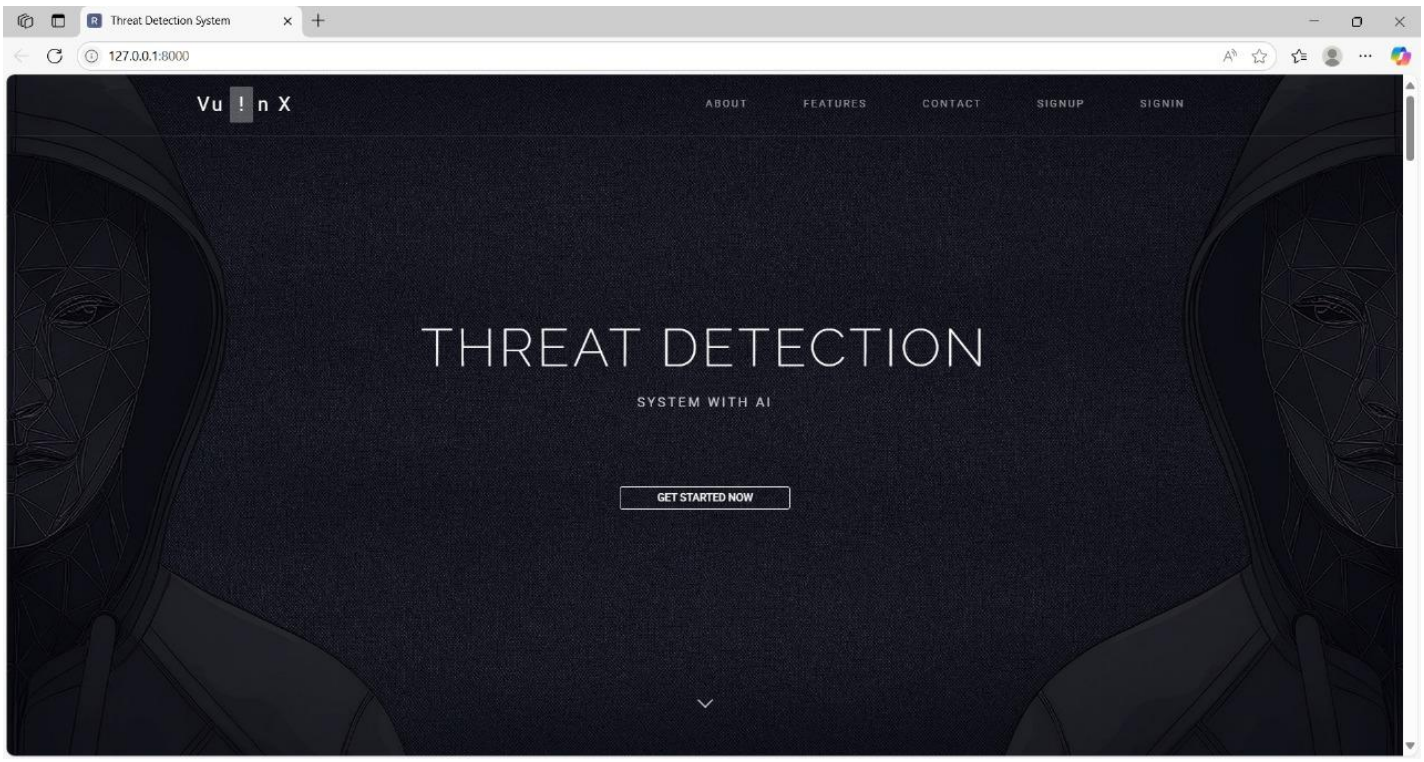
The most critical functions of the system were tested thoroughly through both automated scripts and manual walkthroughs. These included:

- **User Authentication**: Verifying correct login behavior, including role-based access for Admin, Analyst, and Regular Users.
- **Log File Parsing**: Ensuring uploaded files (CSV, JSON, TXT) were scanned properly and threats were categorized correctly.
- **IP/Domain Checks**: Checking the accuracy and reliability of reputation scores fetched from integrated APIs.
- **Incident Reporting**: Confirming that reports could be created, saved, and reviewed correctly based on threat data.

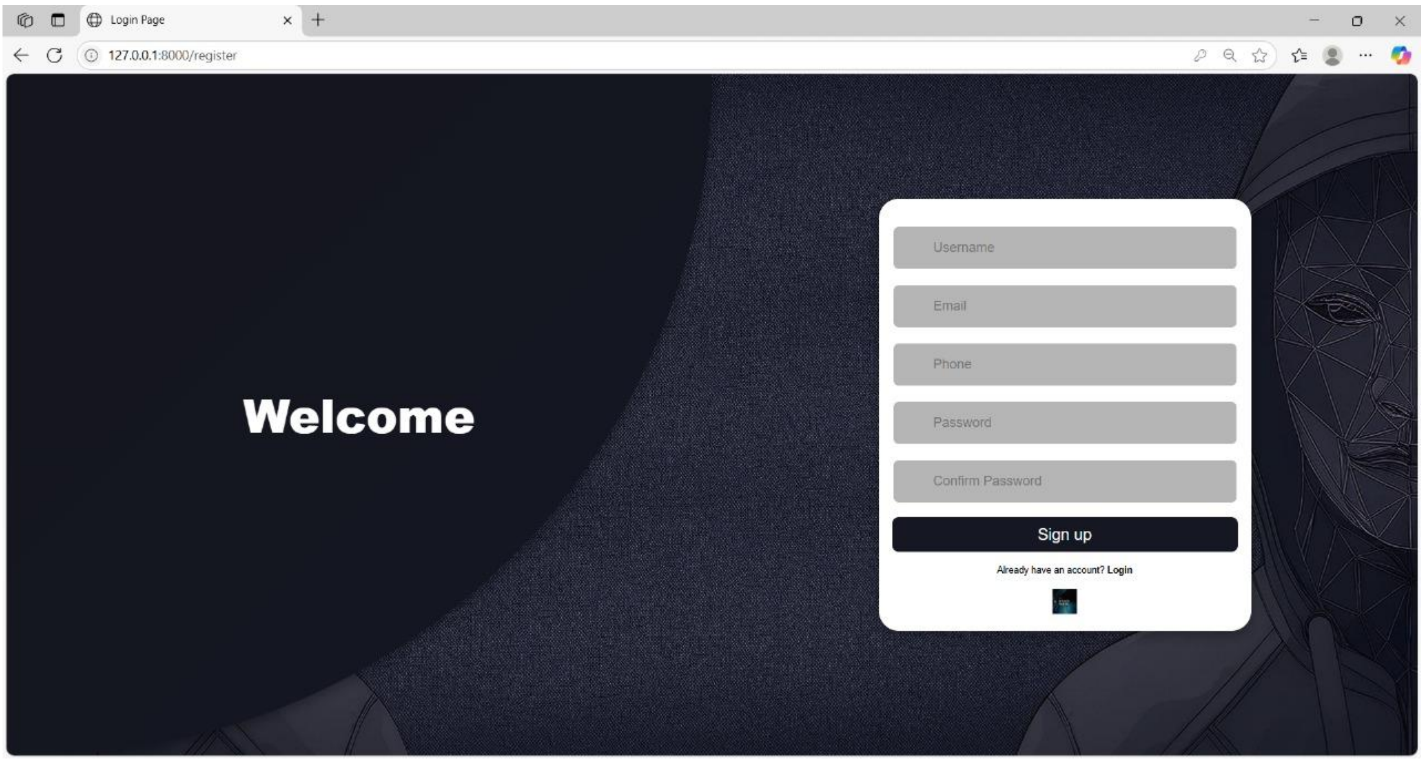
3 Snapshots of the Project

Snapshots provide visual evidence of the working application. Screens include registration and login interfaces, dashboard views, log upload page, IP/domain reputation result pages, task assignment interfaces for admins, and incident report submission for analysts. These snapshots ensure that the GUI was implemented as designed and offer reference for training or demonstrations.

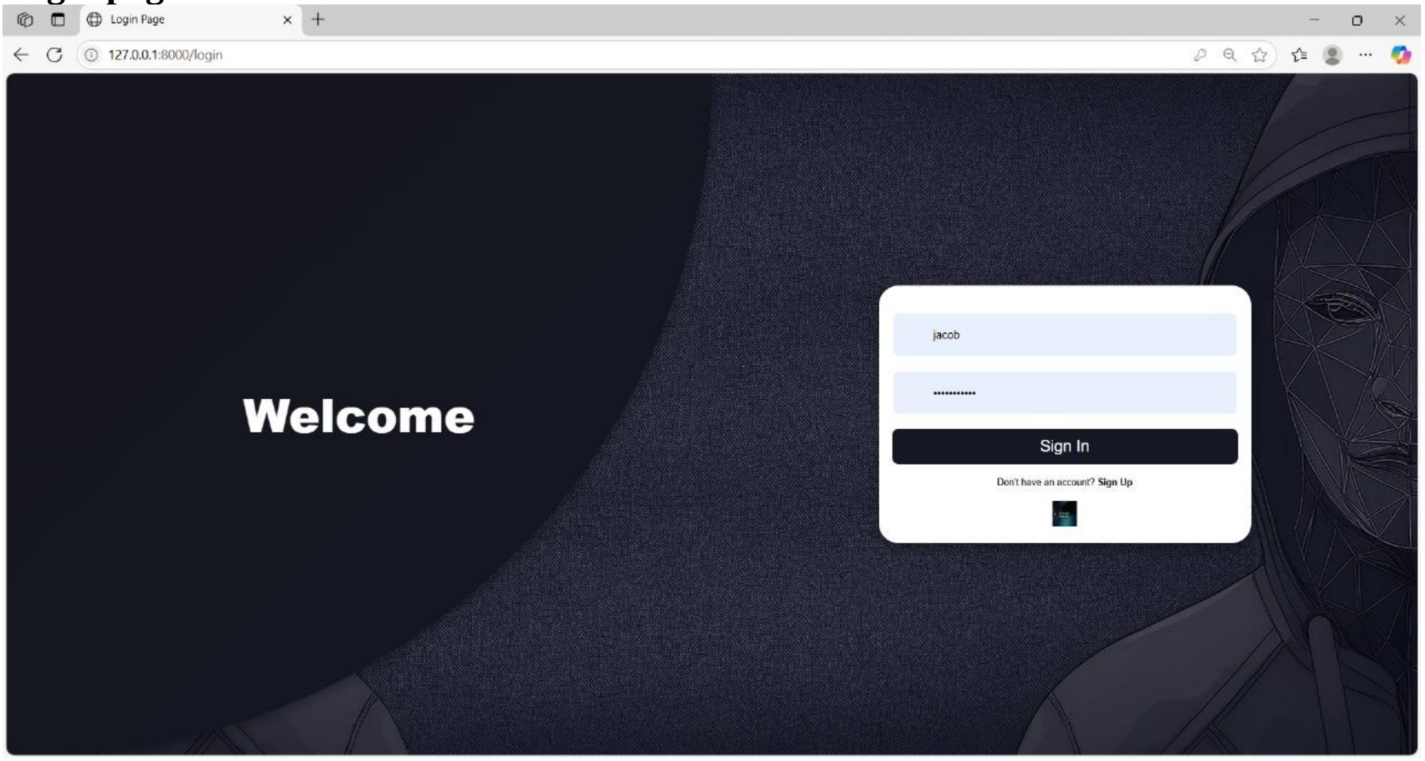
Home Page



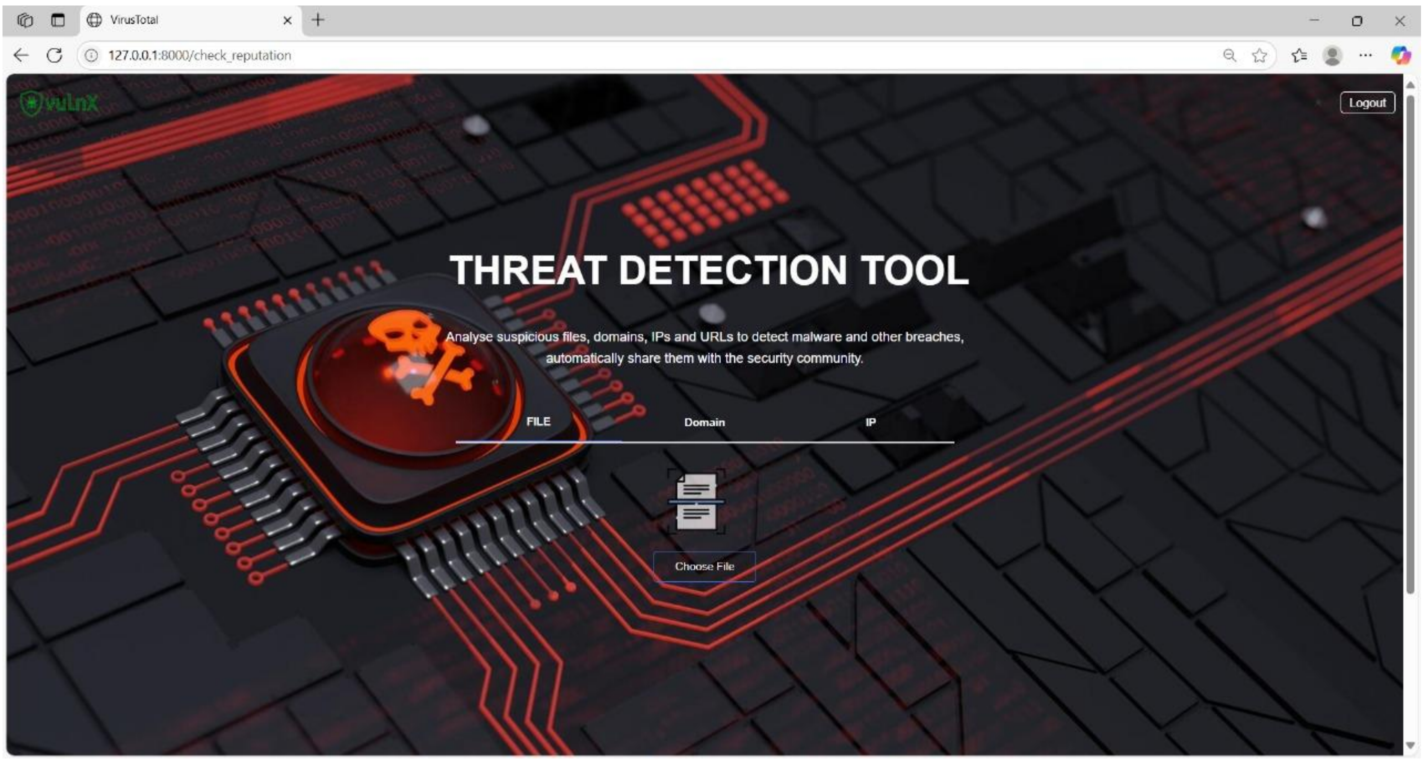
Register Page



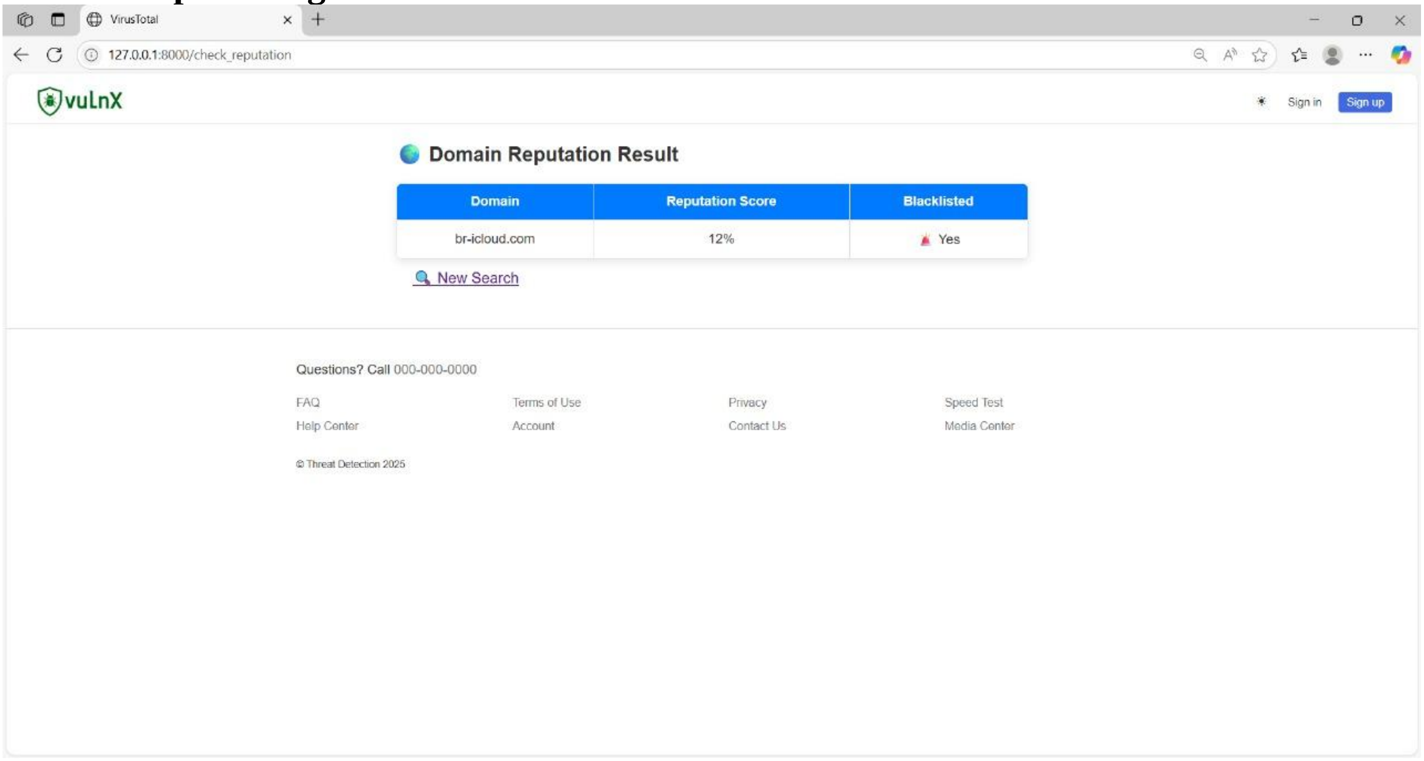
Login page



Threat Detection System Dashboard


VirusTotal
127.0.0.1:8000/check_reputation
vulnX
Logout
THREAT DETECTION TOOL
Analyse suspicious files, domains, IPs and URLs to detect malware and other breaches, automatically share them with the security community.
FILE
Domain
IP
Choose File

Results/Reports Page



Domain Reputation Result

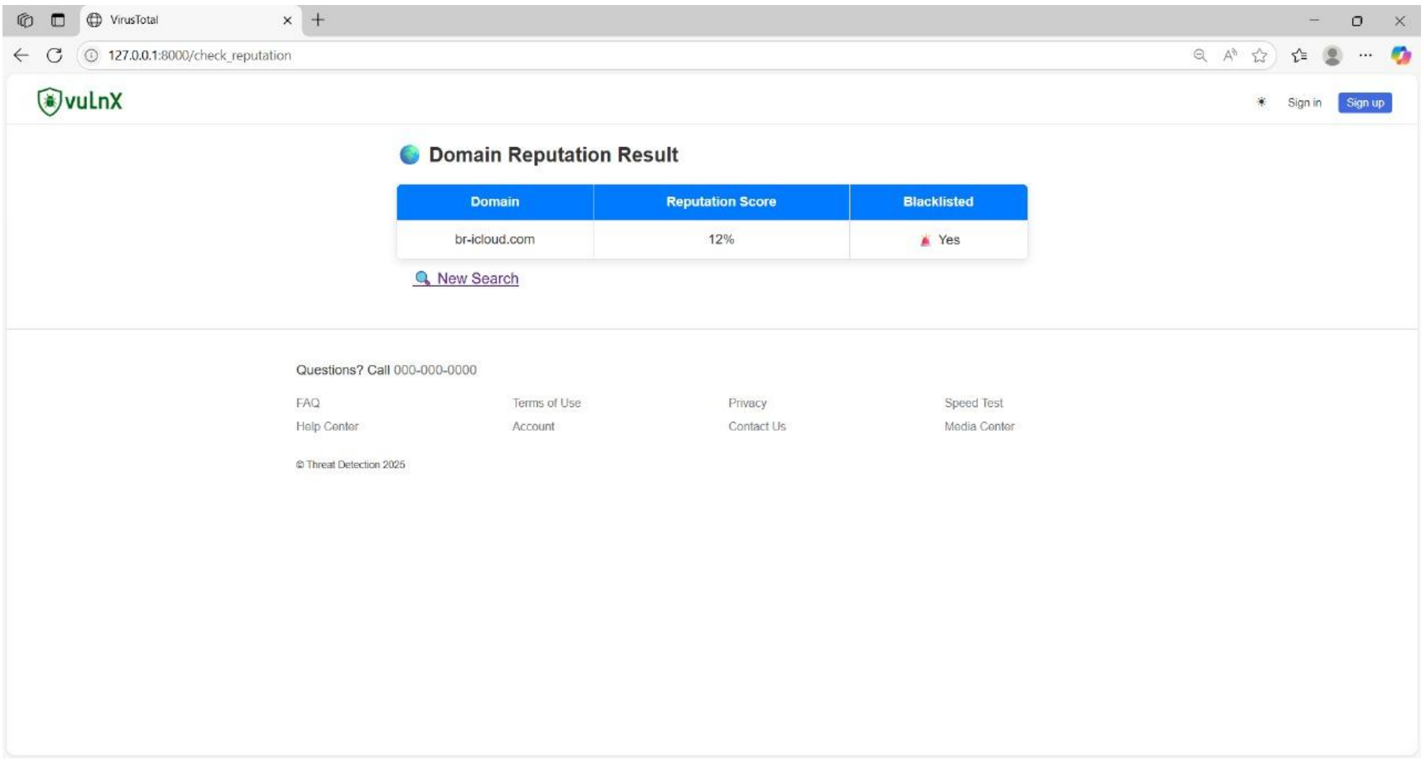
Domain	Reputation Score	Blacklisted
br-icloud.com	12%	Yes

New Search

Questions? Call 000-000-0000

FAQ	Terms of Use	Privacy	Speed Test
Help Center	Account	Contact Us	Media Center

© Threat Detection 2025



Domain Reputation Result

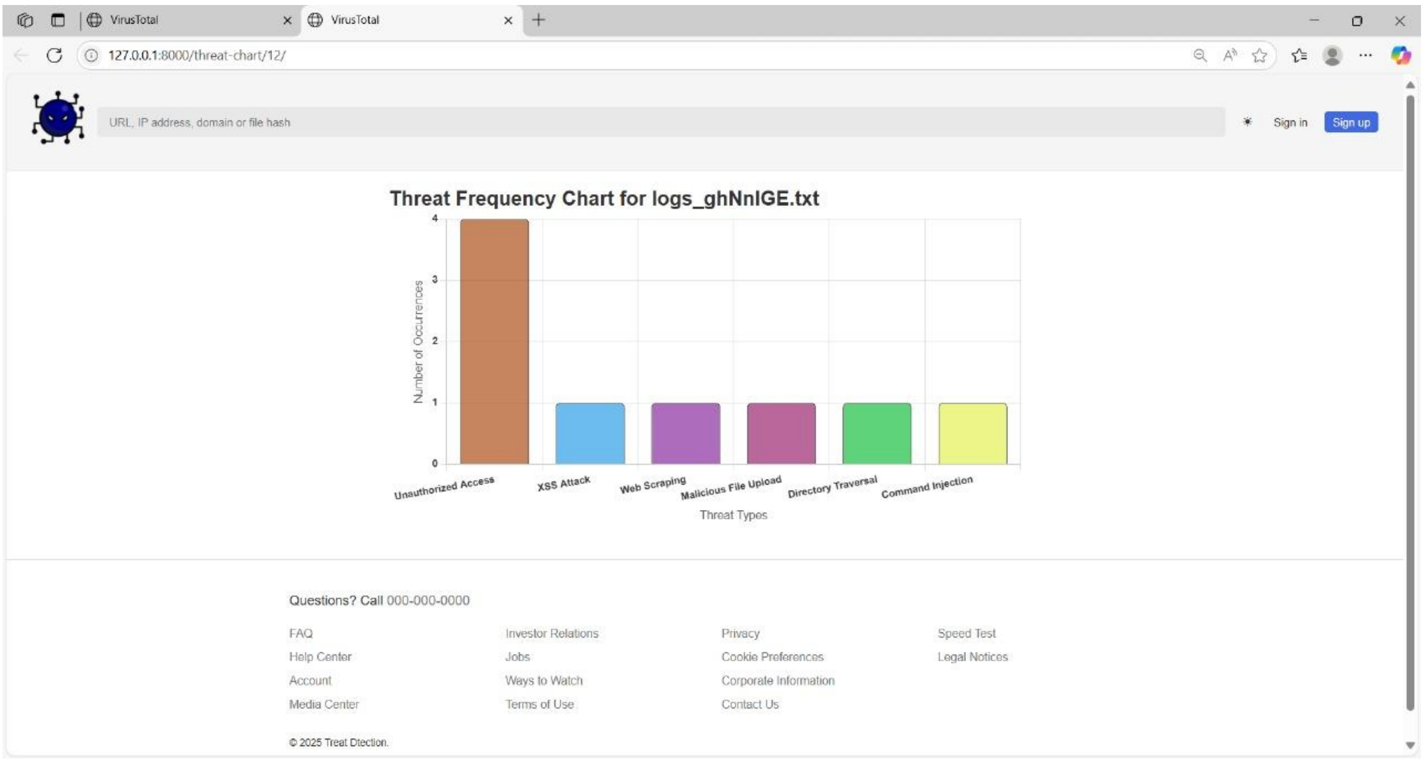
Domain	Reputation Score	Blacklisted
br-icloud.com	12%	🔺 Yes

New Search

Questions? Call 000-000-0000

FAQ	Terms of Use	Privacy	Speed Test
Help Center	Account	Contact Us	Media Center

© Threat Detection 2025

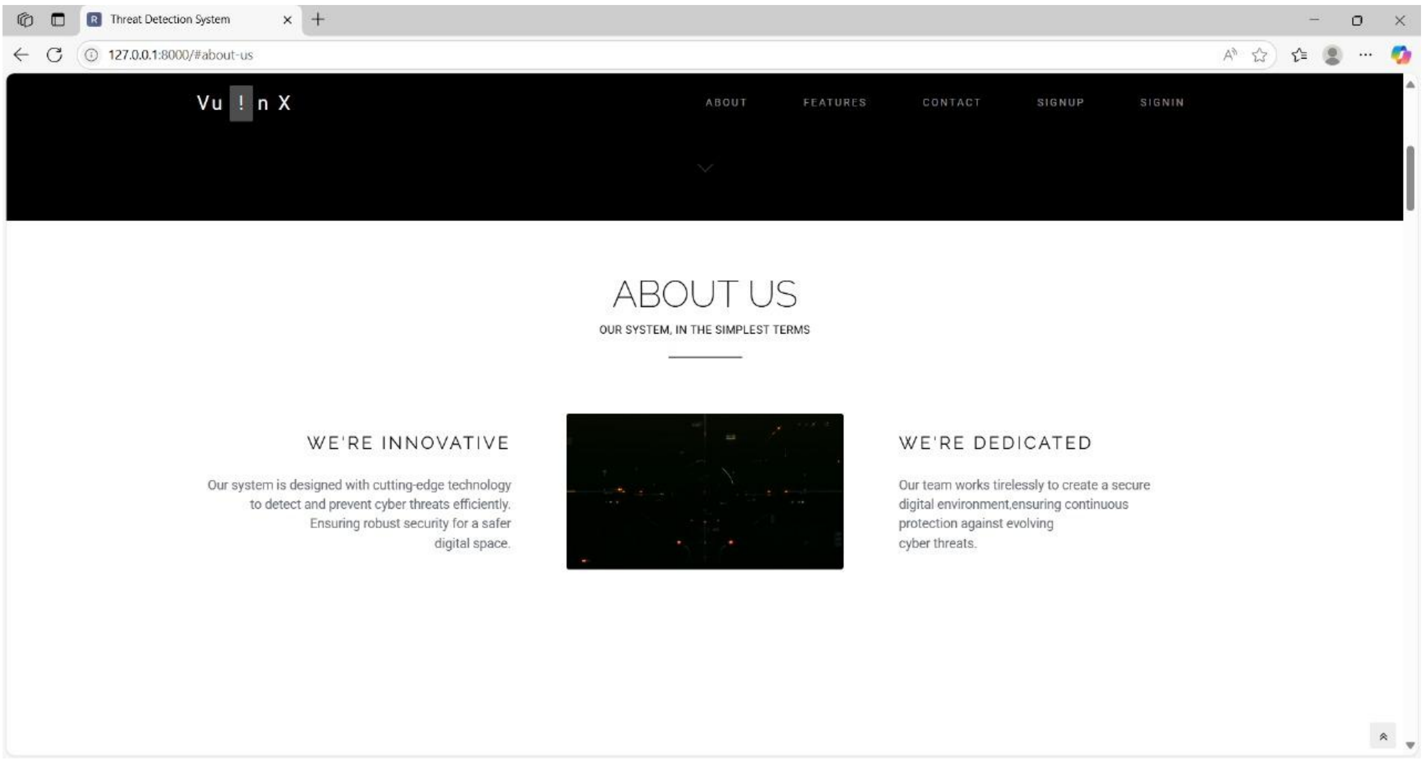


Questions? Call 000-000-0000

FAQ	Investor Relations	Privacy	Speed Test
Help Center	Jobs	Cookie Preferences	Legal Notices
Account	Ways to Watch	Corporate Information	
Media Center	Terms of Use	Contact Us	

© 2025 Treat Dtection.

About us



ABOUT US

OUR SYSTEM, IN THE SIMPLEST TERMS

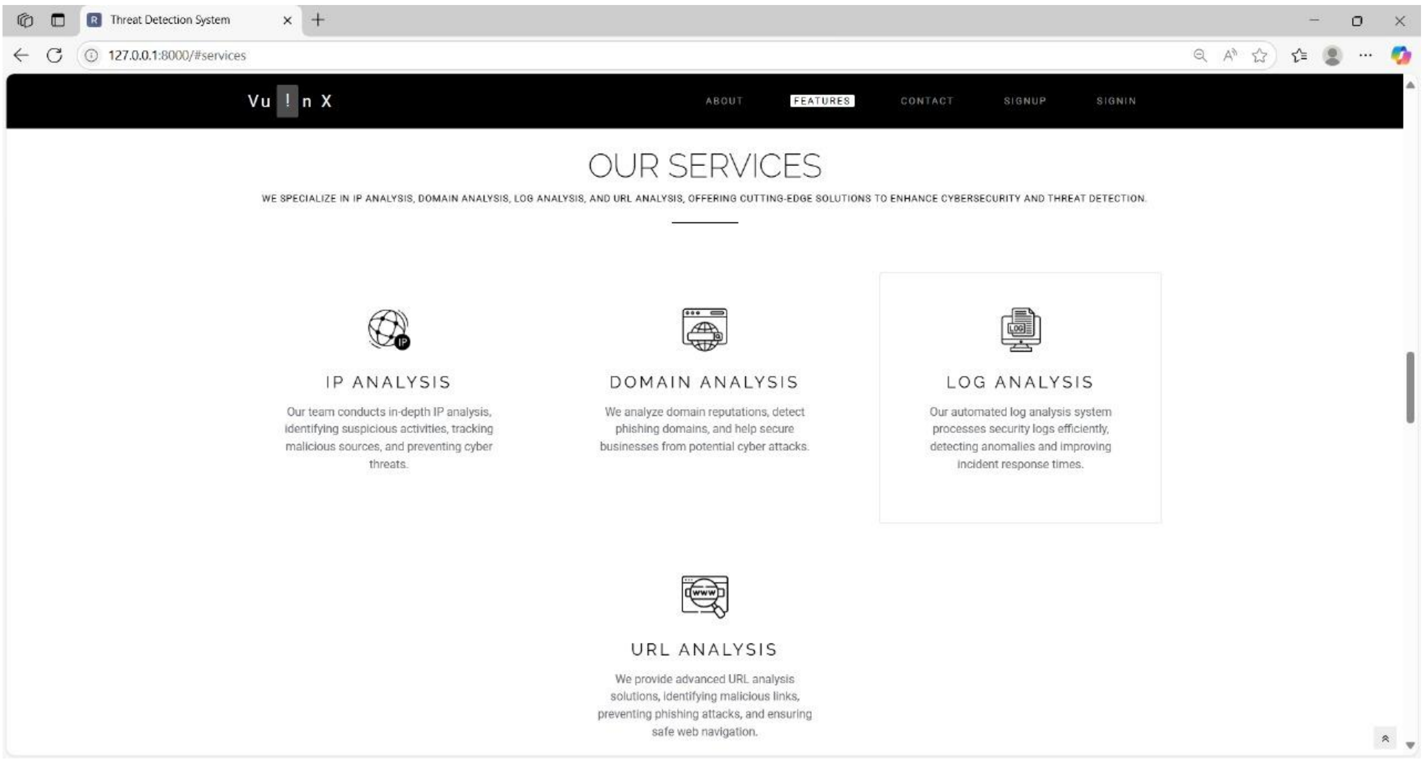
WE'RE INNOVATIVE

Our system is designed with cutting-edge technology to detect and prevent cyber threats efficiently. Ensuring robust security for a safer digital space.

WE'RE DEDICATED

Our team works tirelessly to create a secure digital environment,ensuring continuous protection against evolving cyber threats.

Services



OUR SERVICES

WE SPECIALIZE IN IP ANALYSIS, DOMAIN ANALYSIS, LOG ANALYSIS, AND URL ANALYSIS, OFFERING CUTTING-EDGE SOLUTIONS TO ENHANCE CYBERSECURITY AND THREAT DETECTION.

IP ANALYSIS

Our team conducts in-depth IP analysis, identifying suspicious activities, tracking malicious sources, and preventing cyber threats.

DOMAIN ANALYSIS

We analyze domain reputations, detect phishing domains, and help secure businesses from potential cyber attacks.

LOG ANALYSIS

Our automated log analysis system processes security logs efficiently, detecting anomalies and improving incident response times.

URL ANALYSIS

We provide advanced URL analysis solutions, identifying malicious links, preventing phishing attacks, and ensuring safe web navigation.

4 Conclusions

The Threat Detection System is a comprehensive, web-based cybersecurity platform developed to address the growing need for proactive and structured threat management in organizations. Designed using Django for backend development and SQLite for database handling, the system is capable of performing real-time log analysis, IP/domain reputation checks, and incident management in a user-friendly and secure manner.

A key strength of the platform lies in its modular architecture, which separates functionality across distinct yet interconnected components. This not only simplifies maintenance and upgrades but also enhances scalability. By integrating with external threat intelligence services like AbuseIPDB and VirusTotal, the system ensures accurate, up-to-date information is available during the threat detection process.

The role-based access control (RBAC) model ensures that users only interact with parts of the system that are relevant to their responsibilities, thereby enhancing both security and user experience. Admins have full oversight capabilities, analysts are tasked with structured investigations, and regular users can upload and monitor threats, all within a centralized dashboard interface.

The system was rigorously tested—both through automated tools and live user interaction. It demonstrated strong performance metrics, security safeguards, and a high level of usability. Testers and stakeholders confirmed that the system meets its original objectives: reducing manual effort, increasing response time to incidents, and providing transparency in threat investigations.

Overall, the Threat Detection System delivers a reliable, scalable, and efficient solution to support cybersecurity operations in small to medium-sized organizations. Its successful implementation validates the design choices and provides a strong foundation for future enhancements, including AI integration, real-time alerts, and expanded threat intelligence capabilities.

5 Further development or research

Future versions could incorporate AI/ML algorithms for anomaly detection, real-time alert systems with SMS/email notifications, and enhanced role-specific dashboards. Support for multilingual interfaces, integration with SIEM tools, and multi-tenant support could further extend the platform's utility. Security enhancements such as 2FA and biometric logins are also planned for improved protection of user accounts and sensitive threat data.

1. AI and Machine Learning Integration

Future updates will include artificial intelligence (AI) and machine learning (ML) to make threat detection smarter. These technologies can spot unusual patterns in logs and detect new types of threats automatically, reducing manual work and false alarms.

2. Real-Time Alerts via SMS/Email

The system will soon support real-time alerts that notify users through SMS or email when a serious threat is found. This will help users respond quickly and take action before the threat causes damage.

3. Role-Specific Dashboards

Customized dashboards will be created for each type of user. Admins will see system-wide activity, analysts will see tasks and reports, and regular users will see only their own scans. This will make the platform more user-friendly and efficient.

4. Multi-Tenant Support

The platform will support multiple organizations using the same system while keeping their data separate. Each organization will have its own users, settings, and data. This is ideal for companies with different departments or service providers handling multiple clients.

5. Integration with SIEM Tools and Threat Feeds

To improve threat detection, the system will connect with security tools like SIEMs and live threat feeds. This means it can gather more data from other security sources and stay up to date with the latest threats.

6. Advanced Login Security (2FA and Biometrics)

Security will be improved with features like two-factor authentication (2FA) and biometric logins (like fingerprint or facial recognition). This adds an extra layer of protection to user accounts and sensitive data.

7. Report Downloads and Scheduling

Users will be able to download their reports as PDFs or CSV files. They can also set up automatic report generation on a weekly or monthly basis to make tracking and audits easier.

8. Multilingual Interface

The system will offer support for multiple languages, so users can choose their preferred language for using the platform. This makes it accessible to teams in different countries or those who speak different languages.

6 References

[1] https://docs.djangoproject.com/ - Django Framework Documentation
[2] https://www.sqlite.org/docs.html - SQLite Official Documentation
[3] https://www.chartjs.org/docs - Chart.js for Data Visualization
[4] Anderson, R. (2020). Security Engineering: A Guide to Building Dependable Distributed Systems.
[5] Kumar, R., & Sharma, V. (2021). Cyber Threat Detection Using Log Analysis and Threat Intelligence.
[6] Ali, S., & Khan, M. A. (2022). Real-Time Threat Intelligence and Detection Frameworks.
[7] https://chatgpt.com/ -chatgpt

7 Appendix

The Appendix section provides a valuable collection of supplementary materials that enhance the understanding of the Threat Detection System. It includes a variety of visual, technical, and structural artifacts that support the main content of the documentation. These resources serve as both reference material and practical examples to help developers, testers, and stakeholders grasp how the system works under the hood.
Included in the appendix are code snippets from the frontend templates and backend Django views. These samples showcase the system's layout logic, user input handling, and dynamic content rendering using Django's template language. For instance, the admin dashboard and task assignment modules are explained through annotated HTML snippets and logic flows, helping others reproduce or build upon the system.
The section also provides UI design screenshots that walk users through the entire interface—from login and registration to threat detection and report generation. These screenshots visually demonstrate the system's role-based navigation, user interactions, and feedback mechanisms, highlighting the clarity and usability of the GUI.
In addition, detailed Data Flow Diagrams (DFDs) illustrate how data moves through the system, from log file input to threat analysis and report output. These diagrams are included in multiple levels, showing both high-level overviews and detailed internal processes. This is particularly useful for system analysts and architects who want to evaluate or extend the platform.