

Malicious IP Intelligence System

Mrs. A. Devi¹, G. Ganesh², B. Madhukar³, B. Sanjeev⁴, A. Akhiraanandhan⁵

¹Guide, ^{2,3,4,5}Students, Department of Computer Science and Engineering, Telangana, India.
Telangana, India.

ABSTRACT-Internet-facing systems continuously communicate with external IP addresses that may be associated with scanning, brute-force attempts, spam, automated abuse and other suspicious activity. Security analysts can obtain reputation information from external services, but a practical investigation also requires validation, context, history and a consistent way to process indicators received from documents. This paper presents the Malicious IP Intelligence System, a web-based threat intelligence application developed using Python Flask, SQLite, HTML, CSS and JavaScript and integrated with the AbuseIPDB API. The system supports direct IPv4 and IPv6 analysis and batch analysis of PDF, DOCX, TXT and CSV files. Candidate addresses are extracted, validated and deduplicated before public addresses are analyzed. The application stores abuse confidence score, report count, country, ISP, usage type, domain, whitelist information, Tor indication and last reported date. A separate project-defined Risk Intelligence Index combines the external score with limited contextual factors such as report volume, Tor status, report recency and whitelist information. Reputation results are presented as Safe, Suspicious or Malicious, while project risk is labeled Low, Medium or High. Manual and file-derived investigations use a common SQLite history, allowing dashboard review, comparison, filtering, CSV export and PDF reporting. The system is intended as an explainable analyst workflow and integration layer around external IP reputation intelligence rather than as a replacement for the external service.

Keywords - Malicious IP, Threat Intelligence, AbuseIPDB, IP Reputation, Flask, SQLite, Risk Intelligence, Batch Analysis, Cybersecurity.

1. INTRODUCTION

Internet-facing applications and networks continuously communicate with external IP addresses. Some addresses may be associated with scanning, brute-force attempts, spam, automated abuse, bot activity or other suspicious behavior. IP reputation services provide useful context during investigation, but the raw result is only one part of a complete analyst workflow. A practical system should validate the indicator, present relevant context, preserve the observation and allow the analyst to review the result later.

The proposed Malicious IP Intelligence System uses AbuseIPDB as its external reputation source and places an application layer around that service. The Flask backend validates IPv4 and IPv6 input, communicates with the AbuseIPDB check endpoint and normalizes returned fields. The interface presents reputation status together with project risk, report count, ISP, country, usage type, domain and contextual indicators. The same workflow can be used for indicators extracted from incident reports and other supported documents.

Security indicators are also frequently received inside incident reports, PDF files, Word documents, text files and CSV exports. To support this workflow, the system extracts IPv4 addresses from supported documents, validates them, removes duplicates and separates private or local addresses from public addresses before querying the external intelligence service. The resulting observations are stored in the same history used by manual searches, allowing analysts to review both input paths through one investigation record.

2. PROBLEM STATEMENT

The main problem addressed by this project is the gap between obtaining an IP reputation result and maintaining a consistent investigation workflow. Manual lookup of many indicators can require repeated copying, validation and recording. Information contained in documents can also be overlooked when analysts process indicators one by one. Without a common history, manually searched addresses and addresses obtained from reports may be tracked separately, making investigation review less consistent.

The project therefore focuses on four connected requirements: reliable IP validation and extraction, consistent retrieval of external reputation intelligence, transparent project-defined risk interpretation and persistent investigation history. It also provides analyst-oriented functions such as comparison, status filtering, deletion of selected records, CSV export and PDF report generation. The system is designed to support investigation decisions while avoiding the claim that a reputation score alone proves malicious activity.

3. OBJECTIVES

1. Provide a web interface for analyzing individual IPv4 and IPv6 addresses using external IP reputation intelligence.
2. Retrieve and normalize AbuseIPDB indicators such as abuse confidence score, report count, ISP, country, usage type and domain.
3. Extract IP addresses from PDF, DOCX, TXT and CSV files and automatically remove duplicate indicators.
4. Separate private, loopback and link-local addresses from public IP addresses before external analysis.
5. Calculate an explainable project-defined Risk Intelligence Index using reputation and contextual indicators.
6. Maintain common investigation history and provide dashboard, comparison, filtering, export and reporting functions.

4. PROPOSED METHODOLOGY

After validation, public addresses are queried through the AbuseIPDB check service. The response is converted into a normalized application object so that the same data structure can be displayed, stored and exported. The project then calculates its own bounded risk value. Finally, the scan is saved with the authenticated user and timestamp, allowing the dashboard and IP-specific history page to retrieve the investigation later.

The workflow is divided into input handling, intelligence acquisition, risk interpretation and persistence. For direct searches, the application first checks whether the supplied value is a valid IP address. For a document, the extraction component identifies candidate IPv4 and IPv6 strings and passes them through the same validation stage. This common validation path ensures that manual and batch investigations use consistent rules.

The proposed system follows a web-based threat-intelligence workflow. A user either enters an IP address directly or uploads a supported document containing indicators. Direct input is validated before the external query. During batch processing, candidate IPv4 addresses are extracted from PDF, DOCX, TXT or CSV content, validated and deduplicated. Private, loopback and link-local addresses are separated, while public addresses are analyzed through the AbuseIPDB API. The normalized intelligence is passed to the project risk engine and then stored in SQLite for later review.

User Input → IP Validation / Document Extraction → AbuseIPDB Query → Risk Calculation → SQLite History → Dashboard / Report

Malicious IP Intelligence System - Architecture

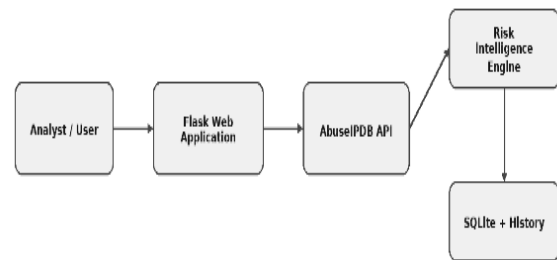


Figure 1: Architecture of the proposed Malicious IP Intelligence System

Figure 1 represents the application architecture. The user interface provides authentication, single-IP analysis, document upload, comparison and investigation views. The Flask application performs validation, document extraction, external API communication and risk calculation. AbuseIPDB supplies reputation indicators, while SQLite stores users, current scan records and scan-history observations. The reporting layer provides CSV export and PDF investigation reports. The architecture keeps the external reputation score separate from the project-defined Risk Intelligence Index.

Core Analysis Logic

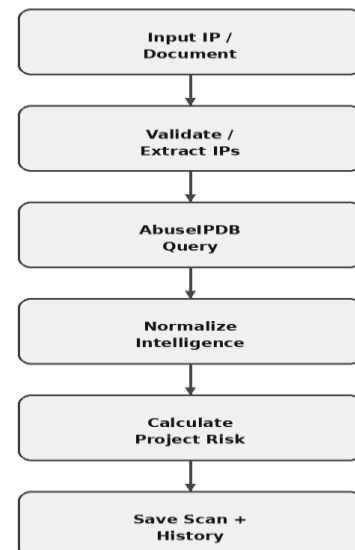


Figure 2: Core analysis logic of the proposed Malicious IP Intelligence System

The core processing logic first validates the supplied address or extracts candidates from a document. For each eligible public address, the application requests AbuseIPDB intelligence, normalizes the response, calculates the project risk score and saves the observation. An authenticated session is used to

associate scan records with the user. Passwords are stored using Werkzeug password hashing, and form submissions use a session-based CSRF token.

The acquisition stage also performs response normalization. Fields returned by the external service are converted into the same internal representation used by the dashboard, comparison view and reporting functions. This avoids maintaining separate display logic for individual investigation paths and makes the stored record consistent with the information shown to the analyst.

The project-defined classification is intentionally separated from the external reputation value. AbuseIPDB provides the source intelligence, while the application applies its own documented thresholds and contextual risk formula. This distinction is important because the local Risk Intelligence Index is intended for project-level prioritization and is not presented as an official vendor classification.

The normalized response is also used to keep the investigation interface consistent. The same fields are available to the dashboard, detailed IP page, comparison view, CSV export and PDF report. This reduces repeated processing and ensures that an analyst sees the same reputation indicators regardless of whether the address was entered manually or obtained from a document.

The application also records the time of each investigation. A current scan record represents the latest known state for the user's saved address, while the history record preserves individual observations. This makes it possible to review repeated checks and observe changes in the external reputation data and project-defined risk value over time.

4.1 IP Intelligence Acquisition and Risk Calculation

Risk Factor	Project Treatment	Purpose
Abuse Confidence Score	Primary factor (75%)	Main reputation signal
Report Volume	Up to +15 points	Context from total reports
Recent Activity	Up to +5 points	Recent reporting context
Tor / Whitelist	Contextual adjustment	+5 Tor; up to -10 whitelist

4.2 Document-Based Batch Analysis

The extraction stage is deliberately separated from the API stage. This prevents private or malformed values from being treated as public threat indicators and makes the batch result easier to interpret. The result page also reports the number of extracted addresses, private/local addresses and analyzed public addresses.

1. The user selects a PDF, DOCX, TXT or CSV file from the batch-analysis interface.
2. The system extracts IPv4 candidates from the selected document and validates each candidate as a valid IP address.
3. Duplicate addresses are removed while preserving the order of the extracted indicators.
4. Private, loopback and link-local addresses are separated so that public reputation queries are not sent for local indicators.
5. Public addresses are analyzed through AbuseIPDB and the project Risk Intelligence Index is calculated for each result.
6. Each analyzed address is saved through the same scan and history mechanism used by manual searches, and the results are displayed in the batch table.

Workflow of Malicious IP Analysis and Notification



Figure 3: Batch IP analysis workflow of the proposed Malicious IP Intelligence System

Figure 3 shows the batch workflow from document upload to persistent investigation history. The current implementation accepts PDF, DOCX, TXT and CSV files and limits one uploaded file to 25 public IP addresses for controlled API use. Errors for individual addresses are reported in the batch result table rather than stopping the complete workflow. Private addresses are displayed separately so that the analyst can distinguish local indicators from publicly queried addresses.

5. IMPLEMENTATION ENVIRONMENT

The Malicious IP Intelligence System is implemented as a lightweight Python web application. Flask manages routing, sessions, authentication and application logic, while HTML, CSS and JavaScript provide the browser interface. SQLite provides local persistence for user accounts, scan records and scan history. The requests library communicates with the AbuseIPDB API, while

pypdf, python-docx and pandas support document processing and ReportLab is used for PDF reporting.

COMPONENT	TECHNOLOGY / PLATFORM	PURPOSE / DESCRIPTION
Web Framework	Python Flask	Routing, sessions and application logic
Frontend	HTML, CSS, JavaScript	Search, upload and dashboard interface
Threat Intelligence	AbuseIPDB API v2	IP reputation indicators
Database	SQLite	Users, scans and investigation history
PDF Processing	pypdf	PDF text extraction
DOCX / CSV	python-docx / pandas	Word and CSV indicator processing
Reporting	ReportLab	PDF investigation reports

The implementation contains separate modules for authentication, single-IP analysis, dashboard presentation, filtering, comparison, batch processing, history retrieval, export and report generation. Input validation uses Python's ipaddress module. The project classifies AbuseIPDB scores into Safe, Suspicious and Malicious categories using project-defined thresholds.

The project Risk Intelligence Index starts with 75 percent of the Abuse Confidence Score, adds a limited contribution for total reports, adds points for a Tor indicator and recent reporting activity, and subtracts points when the address is whitelisted. The resulting score is bounded between 0 and 100. A score below 30 is labeled Low, 30–69 Medium and 70 or above High.

For database persistence, the scans table maintains the current score, status, project risk score, user association and scan time. The scan_history table stores individual observations including abuse score, risk score, status, report count, Tor status, last reported date and scan time. This design allows repeated observations of the same address to remain available for timeline analysis.

5. SYSTEM MODELING

5.1 Overview

The system model represents the Malicious IP Intelligence System as an analyst-facing threat-intelligence pipeline. The authenticated user interacts with the web interface to submit an IP address, upload a

document, compare indicators, inspect history or generate reports. The Flask application coordinates validation, external intelligence retrieval, risk calculation and persistence. System modeling therefore describes how the interface, analysis functions, external intelligence source and database communicate during an investigation.

5.2 Use Case Model

The use case model contains the authenticated analyst as the primary actor. The user can register and log in, analyze an IP address, upload a batch file, view detailed intelligence, compare two addresses, filter saved scans by Safe, Suspicious or Malicious status, delete selected records, export CSV data and generate an IP-specific PDF report. The system also exposes a history API used by the reputation timeline shown on an IP investigation page.

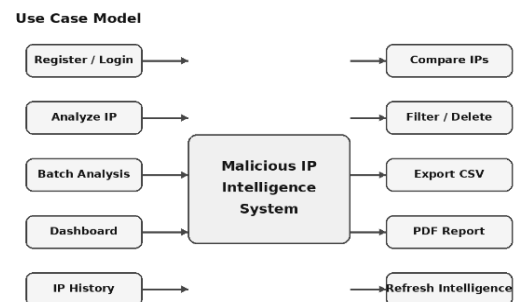


Figure 4: Use Case Model of the proposed Malicious IP Intelligence System

Figure 4 summarizes the principal interactions between the analyst and the application. Single-IP analysis and batch analysis both lead to the same persistence layer. Comparison and dashboard functions reuse the stored scan information, while the report and export functions transform investigation data into analyst-readable outputs.

5.3 System Components and Data Flow

The proposed system contains five major components: the web interface, Flask application layer, AbuseIPDB intelligence service, project risk engine and SQLite persistence layer. The web interface collects the indicator or document. Flask validates and processes the input and communicates with AbuseIPDB. The risk engine calculates the project-defined score, after which the current scan and the observation history are saved. The dashboard, comparison page, filtering page and reporting functions read these stored records to support investigation review.

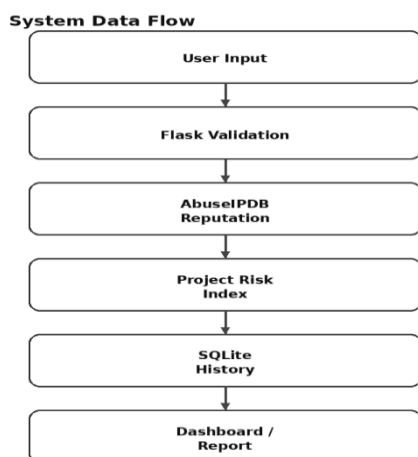


Figure 5: Data flow of the proposed Malicious IP Intelligence System

6. RESULTS AND DISCUSSION

The batch interface provides a second form of validation. It shows extracted indicators, identifies private or local addresses and lists the public addresses that were analyzed. This makes the extraction process visible instead of silently submitting every detected string to the external service. The common history design means that an address found in a document can later be opened and reviewed in the same way as a manually searched IP.

The home interface provides a direct search field and a batch-upload option, allowing the analyst to begin an investigation without navigating to a separate workflow. The detailed investigation page presents the Abuse Confidence Score, project risk score, total reports and contextual intelligence. The comparison page places two addresses side by side so that the analyst can inspect differences in status, score, report count, Tor indication, country and ISP.

The implemented prototype combines direct IP analysis, document-based batch analysis, explainable risk scoring and persistent investigation history in one application. A key functional result is that an IP extracted from a PDF, DOCX, TXT or CSV file is saved through the same scan-saving path as a manually entered IP. This provides one consistent history for different investigation inputs.

The project also provides two clearly separated interpretation layers. AbuseIPDB supplies the external Abuse Confidence Score and contextual fields, while the application calculates its own Risk Intelligence Index. The risk assessment page explains the factors contributing to the project score, provides a recommended monitoring or investigation action and

includes a research disclaimer stating that the heuristic is not an official AbuseIPDB score.

The screenshots in the implementation demonstrate the working web interface, authenticated access, direct IP analysis, comparison of two addresses, document-based batch analysis and detailed investigation output. Because controlled experiments have not been supplied with the project files, this paper does not invent accuracy, latency, throughput or detection-performance numbers. Such measurements should be added after testing on a defined dataset.

7. IMPLEMENTATION

The implementation is organized around reusable processing functions so that the single-IP and batch routes do not need separate intelligence logic. The API response is normalized before storage, and the scan-saving function records both the current state and the historical observation. This keeps dashboard counts, comparison results, IP history and generated reports consistent with the same stored investigation data.

Stage	Activity
Requirement analysis	Define IP investigation, batch processing, history and reporting requirements.
Architecture design	Define the Flask, AbuseIPDB, risk engine, SQLite and presentation layers.
API integration	Connect the AbuseIPDB check endpoint and normalize returned intelligence.
Batch integration	Implement PDF, DOCX, TXT and CSV extraction, validation and deduplication.
Risk integration	Calculate the project Risk Intelligence Index and labels.
Testing	Verify extraction, history persistence, workflow correctness and report generation.

8. FUTURE SCOPE

Future versions can add caching, scheduled rechecks, additional reputation providers and richer IOC extraction from security reports. The reporting layer can also be extended with analyst notes, evidence references and time-based trend summaries. A larger evaluation

can compare extraction accuracy and processing time across different document types and document sizes.

Future work can extend the system with additional threat-intelligence providers, caching and rate-limit handling, scheduled rechecks, richer IOC extraction, support for JSON and log files, analyst notes and evidence attachments. Additional indicators such as domains, URLs and hashes could be integrated into a broader investigation view. The history layer could also support trend analysis, alerting and configurable retention policies. A future research version could evaluate machine-learning methods for prioritization, but such work should use a defined dataset, documented features, suitable baselines and measured evaluation metrics before claiming an ML contribution.

The Malicious IP Intelligence System provides a practical web-based workflow for investigating IPv4 and IPv6 addresses using external IP reputation intelligence. Its contribution is the integration of reputation retrieval with validation, document-based batch analysis, explainable project-defined risk calculation, persistent history, dashboard visualization, comparison and reporting. The system supports the common analyst task of moving from an indicator to a structured investigation record.

9. CONCLUSION

The project demonstrates that external reputation intelligence can be integrated with local application logic without hiding the difference between a vendor-provided indicator and a project-defined interpretation. This separation is important for transparency because the Risk Intelligence Index is an application heuristic and not an official AbuseIPDB score.

The application does not attempt to replace AbuseIPDB or treat an abuse score as definitive proof of malicious activity. Instead, it separates external reputation information from a transparent project-defined risk index and provides contextual factors that can be reviewed before response actions. Manual and file-derived investigations use the same persistence path, allowing repeated observations to be examined through the investigation history and reputation timeline.

9.1 COMPARISON OF RESULTS

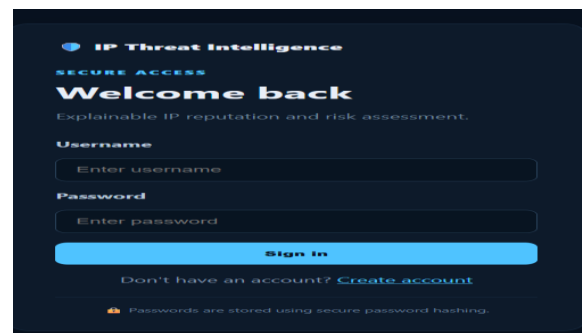


Figure 6: Secure login interface of the Malicious IP Intelligence System

The implemented system combines functions that are often handled as separate steps in a manual workflow. Single-IP lookup is integrated with validation and persistence; document analysis adds extraction and deduplication; the project risk index provides a transparent second interpretation layer; and the dashboard, comparison, export and reporting functions organize the resulting observations. The following project screenshots illustrate the implemented interfaces and analysis workflow.

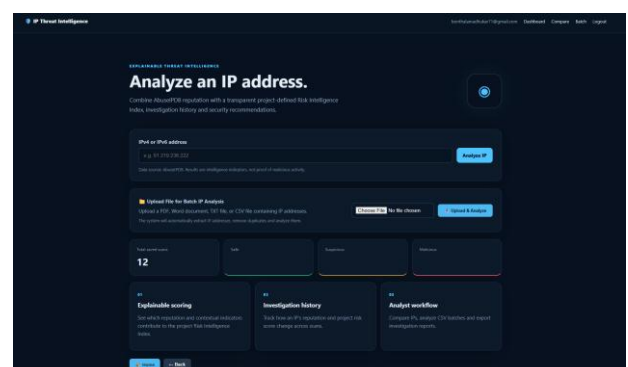


Figure 7: Home and single-IP analysis interface of the Malicious IP Intelligence System

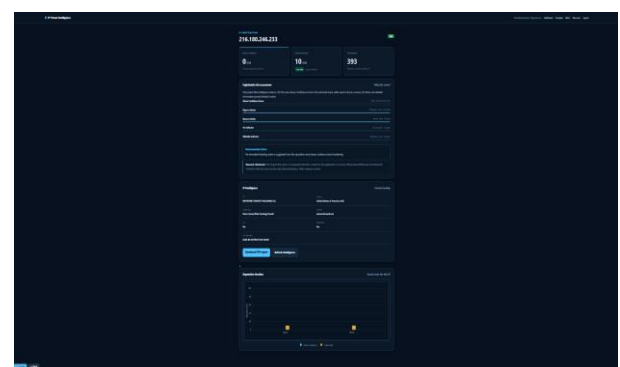


Figure 8: IP investigation and explainable risk assessment interface

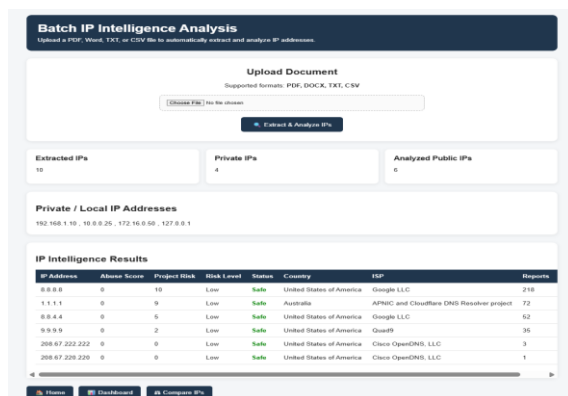


Figure 9: Batch IP Intelligence Analysis interface

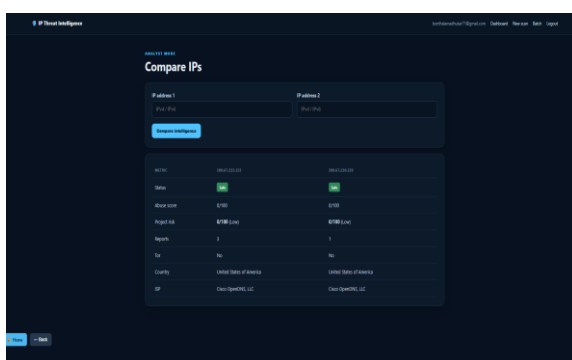


Figure 10: Compare IPs interface of the Malicious IP Intelligence System

REFERENCE

- [1] AbuseIPDB, "AbuseIPDB APIv2 Documentation," Check Endpoint and IP Reputation API Reference. Available at: <https://docs.abuseipdb.com/>.
- [2] Pallets Projects, "Flask Documentation," Flask Web Application Framework. Available at: <https://flask.palletsprojects.com/>.
- [3] SQLite, "SQLite Documentation," SQLite Database Engine Documentation. Available at: <https://www.sqlite.org/docs.html>.
- [4] Python Software Foundation, "ipaddress — IPv4/IPv6 manipulation library," Python Standard Library Documentation. Available at: <https://docs.python.org/3/library/ipaddress.html>.
- [5] OWASP Foundation, "Input Validation Cheat Sheet," OWASP Cheat Sheet Series. Available at: <https://cheatsheetseries.owasp.org/>.
- [6] pypdf Documentation, "pypdf Python PDF Library." Available at: <https://pypdf.readthedocs.io/>.
- [7] python-docx Documentation, "python-docx Documentation." Available at: <https://python-docx.readthedocs.io/>.

[8] pandas Documentation, "pandas Documentation and User Guide." Available at: <https://pandas.pydata.org/docs/>.