PCAP Storyteller

Author - Kaif Tarasgar

1. Executive Summary

PCAP Storyteller is a comprehensive web-based cybersecurity analysis platform designed to transform raw network packet capture data into actionable intelligence through automated parsing, correlation, and visualization. The system addresses fundamental challenges in network security analysis by providing an end-to-end workflow that processes PCAP files, extracts protocol-level events, establishes causal relationships between network activities, detects suspicious patterns, assigns threat scores, and presents findings through an intuitive dashboard interface.

The platform is architected as a Flask-based backend application with a modular parser pipeline, service-oriented intelligence layer, and repository pattern for data management. The parsing engine leverages Scapy to extract events from TCP connections, DNS queries and responses, HTTP requests and responses, TLS handshakes with SNI extraction, ICMP packets, and ARP traffic. Event correlation is achieved through flow tracking and domain resolution mapping, enabling the system to link DNS queries to subsequent HTTP or TLS connections.

Threat intelligence capabilities include event-specific risk scoring based on protocol heuristics, pattern detection for port scanning and data exfiltration indicators, and aggregate threat level classification that accounts for both individual event risks and behavioral patterns. The frontend provides specialized views for attack graph visualization using vis.js, timeline analysis, statistical analytics with Chart.js, threat intelligence dashboards, search and filtering capabilities, and geolocation mapping powered by Folium with integrated IP enrichment from external APIs.

The system successfully processes standard PCAP formats, generates structured event representations, correlates multi-layer activities, identifies attack patterns with configurable sensitivity, and reduces manual analysis time while maintaining technical rigor suitable for academic evaluation, professional investigations, and security training scenarios. The modular architecture facilitates future extensions including live capture integration, machine learning-based anomaly detection, and enterprise threat intelligence feeds.

2. Introduction

Network traffic analysis forms the cornerstone of modern cybersecurity operations, providing visibility into communication patterns, protocol behaviors, and potential security incidents. Packet capture (PCAP) files represent complete snapshots of network activity at the data link and network layers, capturing every frame transmitted across monitored interfaces. However, the inherent complexity of protocol stacks, the volume of captured data, and the technical expertise required to interpret raw packets create significant barriers to effective analysis.

Traditional approaches to PCAP analysis rely on low-level inspection tools that expose individual packets without providing narrative context or automated correlation across protocol layers. Analysts must manually filter traffic, reconstruct application-layer sessions, correlate DNS resolutions with subsequent connections, and infer attack patterns through experience and domain knowledge. This process is labor-intensive, error-prone, and difficult to scale as network environments grow in complexity and attack techniques become more sophisticated.

The increasing sophistication of cyber threats demands analysis tools that can automatically identify suspicious patterns, correlate related events, and present findings in formats accessible to both technical specialists and decision-makers. Educational institutions require practical tools that bridge theoretical networking knowledge with real-world security analysis skills. Security operations centers need systems that reduce triage time while maintaining investigative depth.

PCAP Storyteller addresses these requirements by transforming raw packet captures into structured narratives of network activity. The system automates the extraction of protocol events, establishes causal relationships, scores security risks, and presents results through interactive visualization and analytics dashboards. The project scope encompasses offline PCAP analysis, threat pattern detection, IP geolocation enrichment, statistical analytics, and report generation workflows. The platform emphasizes usability without sacrificing technical rigor, making it suitable for academic evaluation, professional training, and lightweight investigative scenarios.

This report documents the complete development lifecycle, technical architecture, implementation details, testing results, and future enhancement opportunities for the PCAP Storyteller platform. The analysis demonstrates how automated parsing, intelligent correlation, and narrative visualization can significantly improve the efficiency and effectiveness of network security analysis.

3. Problem Statement

Network packet analysis presents several interconnected challenges that limit the effectiveness of manual inspection approaches and constrain the accessibility of security analysis to expert practitioners. The fundamental problem lies in the semantic gap between low-level packet data and high-level security insights. Raw packet captures contain complete network conversations but present them as sequences of frames with protocol headers and payloads that require specialized knowledge to interpret.

The first major challenge is the volume and complexity of captured data. A typical network session involving a simple web request generates dozens of packets across multiple protocols: DNS queries for name resolution, TCP handshakes for connection establishment, TLS negotiations for encryption, HTTP requests and responses for content transfer, and potentially ICMP messages for error handling. Analysts must mentally reconstruct these conversations from individual packets while tracking state across protocols.

The second challenge concerns correlation and causality. Understanding that a DNS query for a specific domain led to a TCP connection, which carried an HTTP request to a resolved IP address, requires cross-referencing timestamps, IP addresses, and domain mappings. Traditional tools display packets chronologically but do not automatically establish these causal relationships, forcing analysts to perform manual correlation.

The third challenge involves threat detection and risk assessment. Identifying malicious patterns such as port scanning, data exfiltration, or command-and-control communications requires recognizing behavioral signatures across multiple events. Analysts must distinguish normal network operations from suspicious activities based on port usage, traffic volumes, domain characteristics, and protocol anomalies. This expertise-dependent process is difficult to codify and scale.

The fourth challenge relates to presentation and communication. Security findings must be communicated to stakeholders with varying technical backgrounds. Raw packet dumps are incomprehensible to non-specialists, while overly simplified summaries may omit critical technical details. The absence of narrative structure makes it difficult to explain the sequence of events, the significance of findings, and the recommended responses.

The fifth challenge concerns educational accessibility. Students learning network security benefit from hands-on analysis of real traffic, but the steep learning curve of professional tools creates a barrier to entry. Educational tools must balance technical depth with usability, providing guided analysis without obscuring underlying details.

PCAP Storyteller addresses these challenges through automated event extraction, intelligent correlation, risk-based scoring, narrative visualization, and layered presentation that serves both expert analysts and learners. The platform transforms the problem of interpreting millions of packets into the more tractable task of analyzing structured events with established relationships and associated threat contexts.

4. Objectives

The primary objective of the PCAP Storyteller project is to develop an automated network traffic analysis platform that transforms raw packet captures into structured security narratives through intelligent parsing, correlation, and visualization. This overarching goal is supported by a set of specific technical objectives that define the system's functional scope and quality requirements.

Primary Objectives:

1. Comprehensive Protocol Parsing: Implement a modular parsing engine capable of extracting meaningful events from multiple network protocols including TCP connection establishment, DNS query and response pairs, HTTP request and response transactions, TLS handshakes with Server Name Indication (SNI) extraction, ICMP messaging, and ARP traffic. The parser must handle malformed packets gracefully and process large capture files efficiently.

2. Intelligent Event Correlation: Establish causal relationships between extracted events by tracking TCP flows, mapping DNS resolutions to IP addresses, and linking domain lookups to subsequent application-layer connections. The correlation engine must support multi-hop relationships and temporal ordering to reconstruct complete network conversations.

3. Automated Threat Detection: Develop a threat analysis subsystem that assigns risk scores to individual events based on protocol-specific heuristics, detects behavioral patterns indicative of attacks, and classifies overall threat levels. Detection logic must cover port scanning, suspicious domain queries, data exfiltration indicators, and external IP communication patterns.

4. Interactive Visualization: Provide multiple visualization modalities including attack graphs that display events as nodes with linked relationships, temporal timelines that show chronological progression, and statistical dashboards that aggregate event distributions, top communicating hosts, and port usage patterns. Visualizations must support user interaction for exploration and detailed inspection.

Secondary Objectives:

5. Geolocation Intelligence: Enrich external IP addresses with geographic location data obtained from public APIs, classify IPs as internal or external, and render locations on interactive maps using server-side Folium generation. The geolocation subsystem must handle rate limiting, caching, and graceful degradation when APIs are unavailable.

6. Search and Filtering Capabilities: Enable flexible searching across event types, IP addresses, domains, and ports with field-specific filtering options. Search results must provide context including event types, timestamps, and descriptions to support investigative workflows.

7. Report Generation: Support exporting analysis results to PDF and DOCX formats with structured sections covering event summaries, analytics, threat findings, and recommendations. Reports must be suitable for documentation, stakeholder communication, and compliance requirements.

8. Modular Architecture: Structure the system with clear separation of concerns using a layered architecture comprising parsing modules, service classes, repository patterns, and API handlers. All source files must remain under 101 lines to maintain readability and facilitate debugging.

9. Comprehensive Logging: Implement session-based logging that creates timestamped log files for each application run, records parsing progress, API interactions, and error conditions. Logs must support post-incident analysis and system debugging.

10. Educational Value: Design the platform to serve as a learning tool for network security students by providing clear event descriptions, visual correlation representations, and threat context that bridges theoretical knowledge with practical analysis skills.

5. System Architecture

The PCAP Storyteller platform employs a layered architecture that separates concerns between data ingestion, processing, intelligence generation, and presentation. This design facilitates maintainability, testability, and extensibility while supporting the modularity constraints imposed by the project requirements. The architecture consists of four primary tiers with well-defined interfaces and responsibilities.

5.1 Ingestion and Upload Tier

The ingestion tier handles file upload through a Flask web server that accepts PCAP, PCAPNG, and CAP file formats via HTTP POST requests. Upon receiving a file, the server validates the file extension against allowed formats, generates a secure filename with a UUID prefix to prevent collision, and stores the file temporarily in a designated upload directory. The upload handler enforces a 1 GB file size limit to prevent resource exhaustion and implements timeout controls for parser invocation.

File validation occurs through a dedicated ValidationService that checks file extensions and provides input sanitization. The uploader invokes the PCAP parser as a controlled subprocess with explicit timeout enforcement (300 seconds for large files) and captures both standard output and error streams for diagnostic purposes. Upon successful parsing, the temporary uploaded file is deleted to minimize storage consumption.

5.2 Parsing and Extraction Tier

The parsing tier transforms raw packet data into structured event objects through a hierarchical decomposition strategy. The PCAPParser class serves as the orchestrator, loading packets using Scapy's rdpcap function and delegating protocol-specific extraction to specialized parser modules. The parser maintains state across packet processing including TCP flow mappings, DNS resolution tables, and a monotonic event counter.

Protocol-specific parsers operate independently:

• **Network Parser:** Extracts TCP connection events from SYN packets, ICMP messages with type and code fields, and ARP requests and replies with hardware addresses.

• **DNS Parser:** Captures DNS queries with question names and response answers with A record mappings. Maintains a bidirectional mapping between domains and resolved IP addresses for correlation purposes.

• **HTTP Parser:** Decodes HTTP methods, URIs, Host headers, and User-Agent strings from requests. Extracts status codes and reason phrases from responses. Links HTTP events to underlying TCP flows.

• **TLS Parser:** Extracts Server Name Indication values from ClientHello messages with robust error handling to gracefully skip malformed TLS packets. Links TLS handshakes to TCP flows and DNS resolutions.

The parser generates two primary data structures: an events array containing event objects with unique IDs, timestamps, types, source and destination IPs, protocol details, and descriptions; and a links array containing relationship triples with source event ID, target event ID, and relationship label. These structures are serialized to JSON using a custom encoder that handles non-serializable types.

5.3 Intelligence and Analytics Tier

The intelligence tier consists of service classes that compute derived insights from parsed events:

• **AnalyticsService:** Aggregates event type distributions, top source and destination IP addresses, port frequency distributions, and protocol usage statistics. Implements efficient counting using Python defaultdict and sorts results for presentation.

• **ThreatService:** Wraps the ThreatAnalyzer to provide threat analysis workflows. Computes individual event risk scores, detects behavioral patterns, identifies high-threat events above configurable thresholds, and generates threat summaries with overall risk levels.

• **SearchService:** Indexes events for flexible querying by IP addresses, domains, event types, and ports. Supports both field-specific and full-text search with case-insensitive matching. Returns formatted search results with contextual information.

• **GeolocationService:** Enriches external IP addresses with geographic data using a two-tier API approach: primary queries to ipinfo.io and fallback queries to ip-api.com. Implements caching to minimize API calls, rate limiting to respect service quotas, and classification of private versus external IPs.

• **FoliumMapService:** Generates interactive maps using Folium with marker clustering for performance, customized popups displaying IP metadata, and color-coded markers based on threat indicators. Returns HTML representation for embedding in the web interface.

The ThreatAnalyzer class implements the core threat detection logic using rule-based heuristics. TCP events are scored based on destination port characteristics including suspicious services, privileged ports, and scanning-prone ports. DNS queries are evaluated for suspicious keywords and abnormal length. HTTP requests are assessed by method type and user agent presence. TLS SNI strings are checked for suspicious patterns. External source IPs add additional risk weight. The analyzer also implements pattern detection by aggregating behaviors across events, identifying port scanning through high unique port counts per source, DNS-to-connection sequences, and potential data exfiltration based on HTTP request volumes.

5.4 Data Management Tier

The data tier implements the repository pattern through the DataRepository class, which provides centralized access to persisted event and link data. The repository loads and saves JSON files, extracts unique IP addresses for geolocation processing, and handles error conditions gracefully. This abstraction insulates higher layers from storage format changes and facilitates testing through mock implementations.

5.5 Presentation and API Tier

The presentation tier exposes functionality through a Flask-based REST API with clearly defined endpoints:

• Template routes (/, /analytics, /threats, /search, /geolocation, /timeline, /report) render HTML pages with embedded JavaScript that fetch data asynchronously.

• API routes (/api/analytics, /api/threats, /api/search, /api/geoips, /api/geoip/<ip>, /api/geomap) return JSON or HTML responses for consumption by frontend code.

• Report routes (/report/pdf, /report/docx) provide download endpoints for document export workflows.

API handlers coordinate between repositories and services, transforming stored data into API responses. Error handling returns appropriate HTTP status codes and error messages for client-side display.

5.6 Frontend Architecture

The frontend consists of specialized HTML pages with embedded JavaScript that invoke APIs and render results using third-party visualization libraries. The main dashboard integrates vis.js for attack graphs and vis-timeline for temporal views. The analytics page uses Chart.js for doughnut and bar charts. The geolocation page embeds Folium-generated maps via iframes. The search page provides form-based query interfaces. The threats page displays risk scorecards and pattern summaries. All pages share a consistent dark theme CSS stylesheet optimized for readability and professional presentation.

5.7 Cross-Cutting Concerns

Logging is implemented through a centralized logger module that creates session-specific log files with timestamps in the format DDMMM_HHMMSS.log. The logger records application lifecycle events, parsing progress, API invocations, service operations, and error conditions at appropriate verbosity levels (DEBUG, INFO, WARNING, ERROR). Configuration management consolidates file paths, allowed extensions, and upload limits in a dedicated config module.

6. Technologies Used

The PCAP Storyteller platform integrates a carefully selected technology stack that balances functionality, maintainability, and deployment simplicity. The selection criteria prioritized mature libraries with active community support, permissive licensing, and proven performance characteristics in production environments.

Backend Technologies:

• **Python 3.7+:** Selected as the primary implementation language due to extensive library ecosystem for networking, strong typing capabilities, readable syntax facilitating maintenance, and widespread adoption in cybersecurity tooling.

• **Flask 2.x:** Lightweight web framework providing HTTP routing, request handling, template rendering, and WSGI integration. Flask's minimalist design allows precise control over application structure without excessive abstraction layers.

• **Scapy 2.4+:** Comprehensive packet manipulation library supporting capture file reading, protocol layer access, packet crafting, and dissection. Scapy provides Python-native representations of network protocols with extensive protocol coverage including esoteric and custom protocols.

• **scapy-http:** Extension package adding HTTP protocol dissection capabilities to Scapy, enabling extraction of methods, headers, and content from HTTP traffic within PCAP files.

Geolocation and External Services:

• **requests:** HTTP client library for API interactions with geolocation services. Provides connection pooling, timeout controls, and response parsing.

• **ipinfo.io API:** Primary geolocation service offering free tier with 50,000 requests per month. Returns coordinates, city, country, ISP, and ASN data.

• **ip-api.com:** Fallback geolocation service with 45 requests per minute rate limit. Provides similar data fields with different geographic coverage.

• **Folium 0.20.0:** Python wrapper around Leaflet.js for server-side map generation. Supports marker clustering, custom popups, and layer controls. Generates standalone HTML with embedded JavaScript.

Report Generation:

• **ReportLab 3.x:** PDF generation library with support for vector graphics, complex layouts, tables, and multi-page documents. Provides low-level canvas API and higher-level platypus flowable document model.

• **python-docx 0.8.x:** Microsoft Word document generation library supporting sections, paragraphs, tables, and styles. Enables DOCX export for stakeholder reports requiring editable formats.

Frontend Technologies:

• **HTML5:** Semantic markup language providing structural foundation for all web pages with support for modern APIs including localStorage for client-side data caching.

• **CSS3:** Styling language implementing dark theme design system with consistent color palette, typography, and responsive layout. Uses flexbox and grid for complex layouts.

• **JavaScript ES6+:** Client-side scripting language implementing API calls, event handling, and dynamic content updates. Uses async/await for asynchronous operations and destructuring for clean code.

• **vis.js 4.21.0:** Network visualization library for rendering attack graphs with force-directed layout, node grouping, interactive exploration, and customizable styling.

• **vis-timeline 7.7.0:** Timeline visualization component for temporal event display with grouping, zooming, and detail popups. Supports large datasets through virtualization.

• **Chart.js 3.9.1:** Charting library implementing doughnut charts for event type distribution, horizontal bar charts for top IP analysis, and vertical bar charts for port distribution. Supports responsive sizing and dark themes.

• **Font Awesome 6.0:** Icon library providing scalable vector icons for navigation, status indicators, and visual embellishment. Improves interface clarity and professional appearance.

Data Management:

• **JSON:** Lightweight data interchange format used for event storage, API responses, and configuration. Python's native json module provides efficient serialization with custom encoder support.

Development and Deployment:

• **gunicorn:** Production-grade WSGI HTTP server supporting multiple worker processes, automatic worker restarts, and SSL termination. Included in root requirements.txt for production deployment scenarios.

• **Flask development server:** Built-in HTTP server used for local testing and development. Provides automatic reloading and detailed error pages.

Quality and Testing:

• **Python type hints:** Optional static type annotations improving code documentation and enabling linter-based error detection during development.

• **Comprehensive logging:** Session-based logging system recording application events, service operations, and error conditions for post-deployment analysis.

7. Features and Implementation

The PCAP Storyteller platform implements a comprehensive feature set addressing the complete workflow from raw packet capture ingestion through threat analysis and visualization. Each feature is implemented through modular components with clear separation of concerns and well-defined interfaces.

7.1 Multi-Protocol PCAP Parsing

The parsing engine processes PCAP files containing traffic across multiple OSI layers and application protocols. The implementation uses Scapy's rdpcap function to load complete captures into memory and iterates through packets in two passes. The first pass extracts TCP connection events from SYN packets to establish flow state. The second pass processes application-layer protocols, correlating them with established flows and DNS resolutions.

TCP connection extraction identifies SYN packets through flag inspection (0x02 mask) and generates flow keys combining source IP, source port, destination IP, destination port, and protocol number. Connection events include byte counts for traffic volume analysis.

DNS processing handles both queries and responses. Queries extract question names and associate them with query IDs for response matching. Responses parse A records to extract IP addresses and domain mappings. The parser maintains a dns_map dictionary tracking all resolutions and an ip_to_domain dictionary enabling reverse lookups.

HTTP parsing leverages scapy-http extensions to decode request methods, paths, Host headers, and User-Agent strings. Responses are parsed for status codes and reason phrases. HTTP events are automatically linked to underlying TCP flows and correlated with preceding DNS queries when the Host header matches a resolved domain.

TLS parsing extracts Server Name Indication (SNI) from ClientHello messages by iterating through TLS extensions and identifying extension type 0. The implementation includes defensive programming to handle malformed packets through hasattr checks and exception handling that silently skips problematic packets rather than aborting parsing.

7.2 Event Correlation Engine

The correlation engine establishes causal relationships between events using three primary mechanisms: flow tracking, DNS mapping, and temporal ordering. Flow tracking maintains a flows dictionary mapping flow tuples to event IDs. When application-layer events are detected, the parser queries this dictionary to link them to underlying transport connections.

DNS correlation uses the dns_map to associate queries with responses and subsequent connections. When an HTTP or TLS event references a domain that exists in the DNS map, the parser creates links labeled 'resolves to' connecting the DNS response to the application event. This enables analysts to trace traffic from initial name resolution through connection establishment to application-layer communication.

7.3 Threat Analysis Subsystem

The threat analysis subsystem implements a two-tier detection approach combining event-level scoring with pattern-level detection. Event scoring assigns numerical risk values (0-100) based on protocol-specific heuristics. TCP events score highly when destination ports match known vulnerable services (445, 139, 3389, 22, 23, 21), privileged ports below 1024, or commonly scanned ports. DNS queries receive elevated scores for suspicious keywords indicating malware, phishing, C2, or exploit activity. HTTP POST and PUT methods score higher than GET due to potential for data exfiltration. Missing User-Agent headers indicate non-browser traffic potentially from automated tools.

Pattern detection identifies multi-event behaviors. Port scanning is detected by tracking unique destination ports per source IP and flagging sources contacting more than 10 unique ports. DNS-to-connection patterns are identified when DNS queries are followed by immediate connections to resolved IPs. Data exfiltration risk is flagged when multiple HTTP requests occur in sequence.

The threat level classification synthesizes individual scores and pattern severity. The system computes average event scores but applies a pattern score that accounts for behavioral severity. The overall threat level is derived from the maximum of these values, ensuring that critical patterns elevate the assessment even when individual event scores are modest. Levels are classified as CRITICAL (≥70), HIGH (≥50), MEDIUM (≥30), or LOW (<30).

12. Project Images

The following images provide visual documentation of the PCAP Storyteller user interface and functional capabilities. Each screenshot demonstrates a specific aspect of the platform's analytical and visualization features.

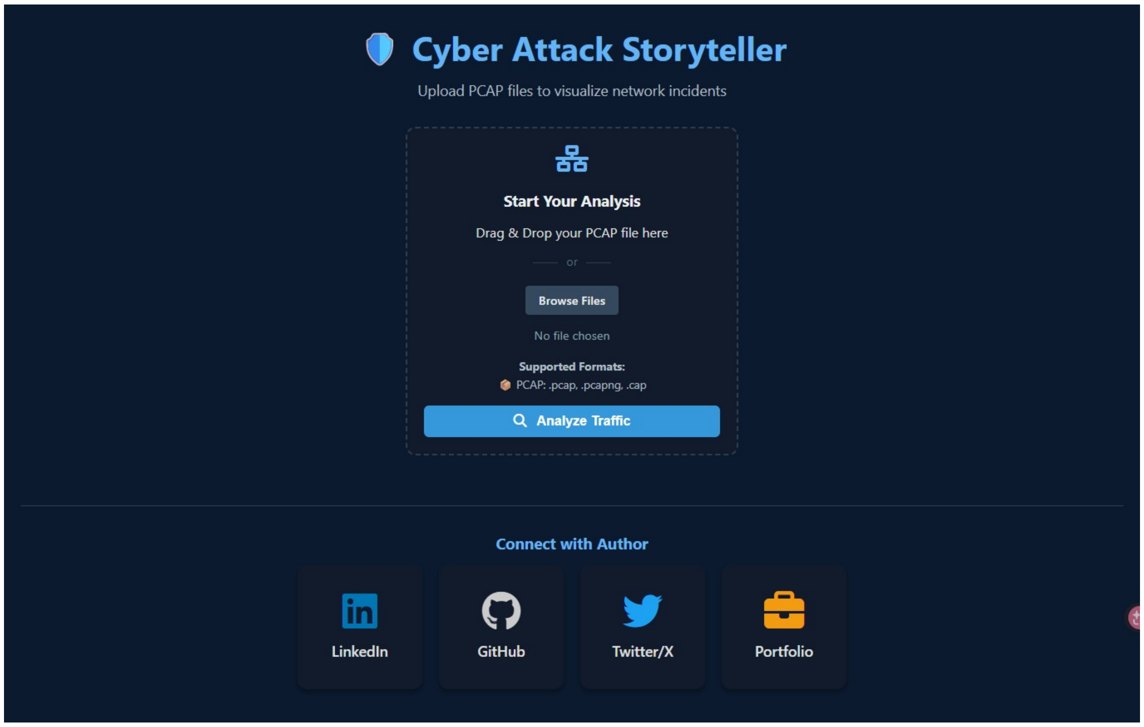


Figure 1: Main Dashboard - File upload interface and attack graph visualization

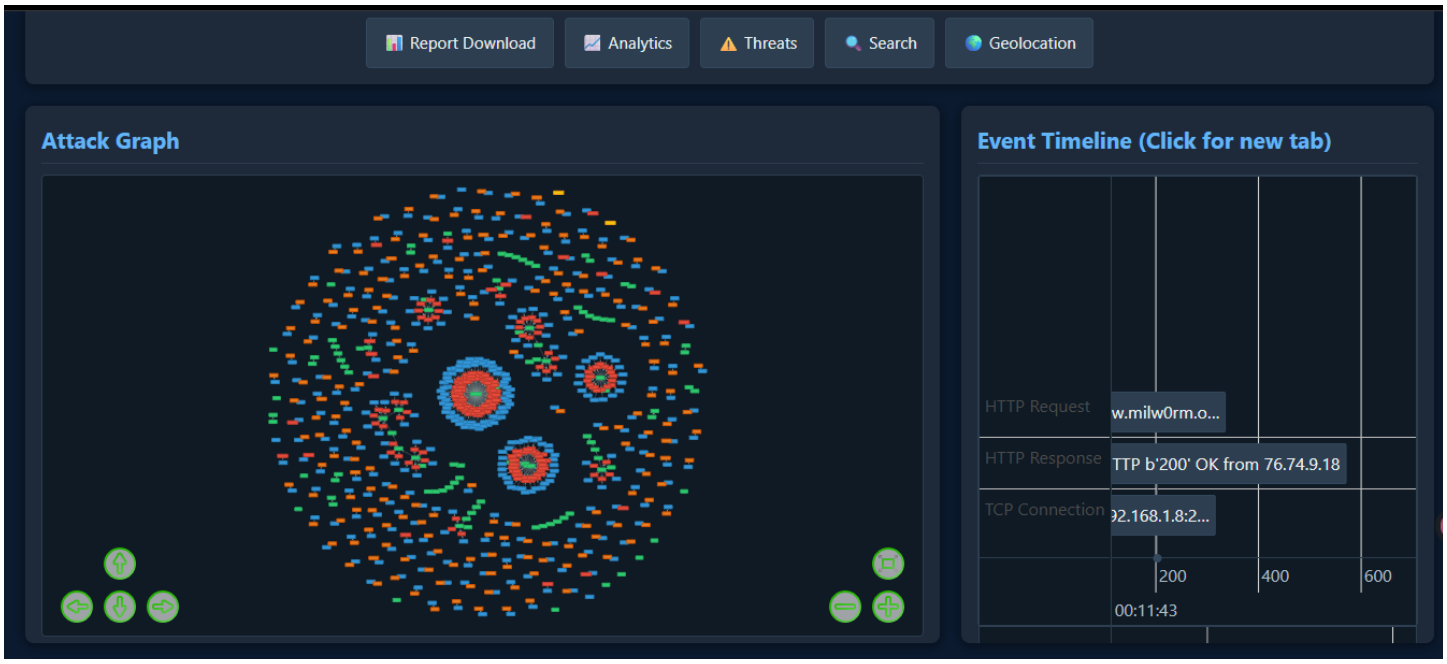
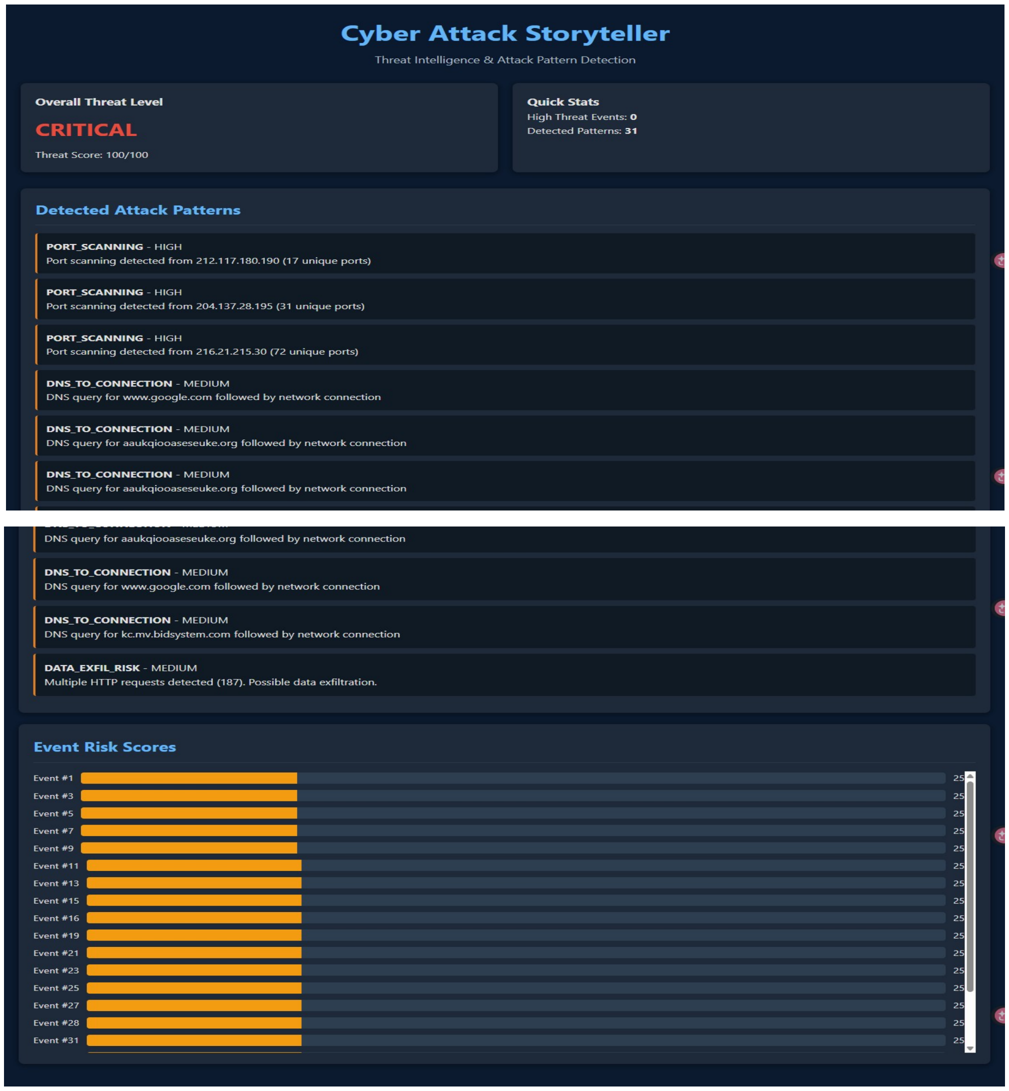


Figure 2: Graph and Timeline



Cyber Attack Storyteller

Threat Intelligence & Attack Pattern Detection

Overall Threat Level

CRITICAL

Threat Score: 100/100

Quick Stats

High Threat Events: **0**

Detected Patterns: **31**

Detected Attack Patterns

PORT_SCANNING - HIGH
Port scanning detected from 212.117.180.190 (17 unique ports)

PORT_SCANNING - HIGH
Port scanning detected from 204.137.28.195 (31 unique ports)

PORT_SCANNING - HIGH
Port scanning detected from 216.21.215.30 (72 unique ports)

DNS_TO_CONNECTION - MEDIUM
DNS query for www.google.com followed by network connection

DNS_TO_CONNECTION - MEDIUM
DNS query for aaukqiooaseseuke.org followed by network connection

DNS_TO_CONNECTION - MEDIUM
DNS query for aaukqiooaseseuke.org followed by network connection

DNS query for aaukqiooaseseuke.org followed by network connection

DNS_TO_CONNECTION - MEDIUM
DNS query for www.google.com followed by network connection

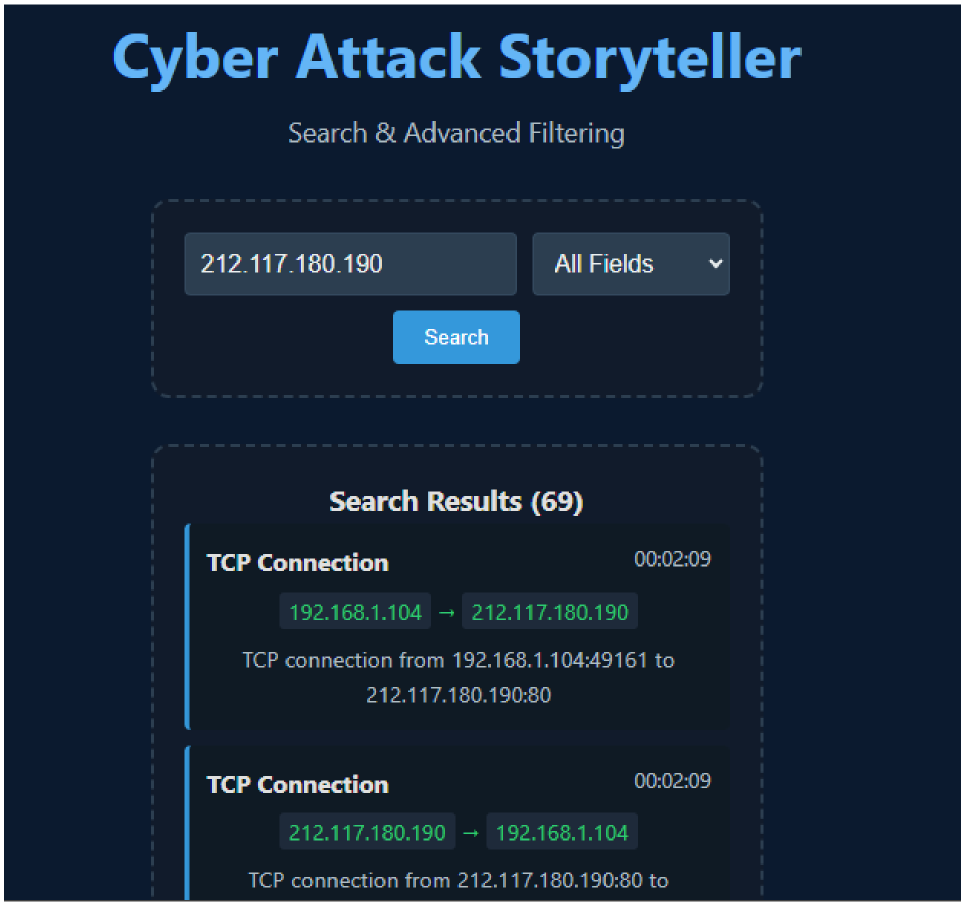
DNS_TO_CONNECTION - MEDIUM
DNS query for kc.mv.bidsystem.com followed by network connection

DATA_EXFIL_RISK - MEDIUM
Multiple HTTP requests detected (187). Possible data exfiltration.

Event Risk Scores

Event #1	25
Event #3	25
Event #5	25
Event #7	25
Event #9	25
Event #11	25
Event #13	25
Event #15	25
Event #16	25
Event #19	25
Event #21	25
Event #23	25
Event #25	25
Event #27	25
Event #28	25
Event #31	25

Figure 3: Threat Intelligence - Risk scoring and pattern detection


Cyber Attack Storyteller

Search & Advanced Filtering

Search Results (69)

TCP Connection 00:02:09

192.168.1.104 → 212.117.180.190

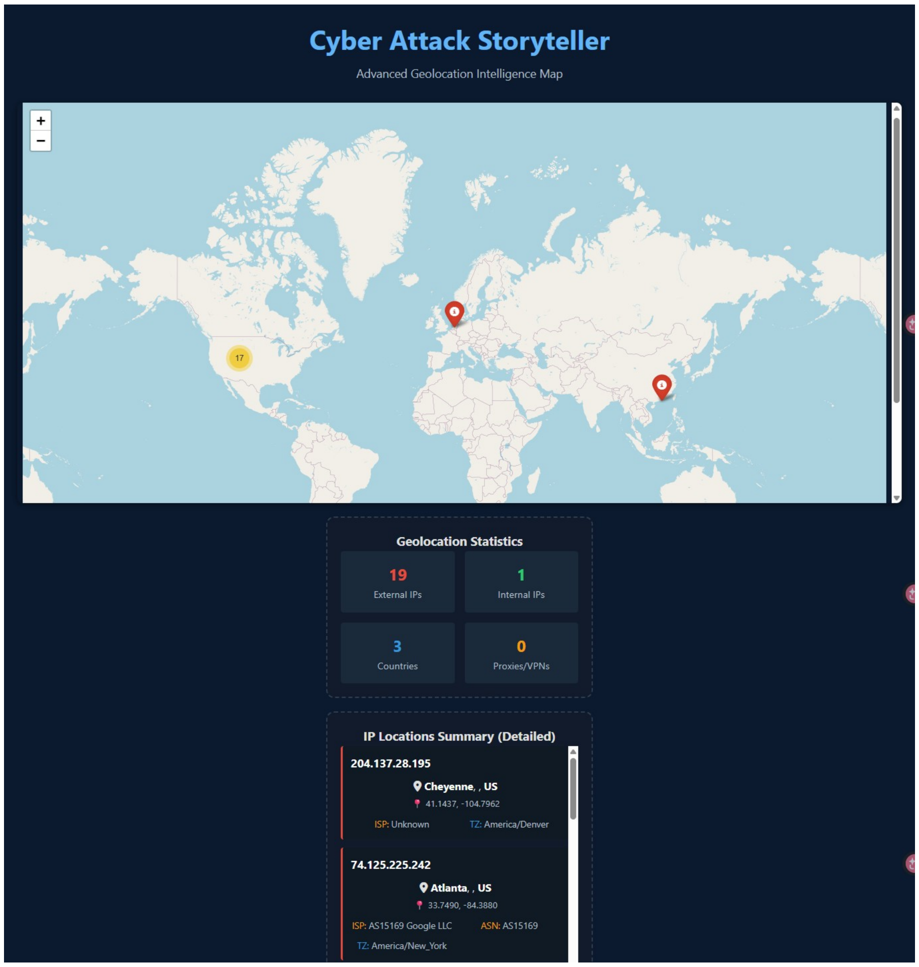
TCP connection from 192.168.1.104:49161 to 212.117.180.190:80

TCP Connection 00:02:09

212.117.180.190 → 192.168.1.104

TCP connection from 212.117.180.190:80 to

Figure 4: Special and Advance Search


Cyber Attack Storyteller

Advanced Geolocation Intelligence Map

Geolocation Statistics

19 External IPs	1 Internal IPs
3 Countries	0 Proxies/VPNs

IP Locations Summary (Detailed)

204.137.28.195
Cheyenne, , US
41.1437, -104.7962
ISP: Unknown TZ: America/Denver

74.125.225.242
Atlanta, , US
33.7490, -84.3880
ISP: AS15169 Google LLC ASN: AS15169
TZ: America/New_York

Figure 5: Geolocation Map - Interactive geographic IP visualization

20. Threat Detection and Risk Scoring

The threat detection and risk scoring subsystem represents a critical component of the platform's intelligence capabilities. The implementation combines event-level heuristics with behavioral pattern analysis to provide comprehensive threat assessment.

Event-Level Scoring Methodology:

Individual events receive risk scores based on protocol-specific characteristics that correlate with malicious behavior in empirical threat analysis. TCP connection events undergo port-based risk assessment where destination ports matching known vulnerable services (SMB/445, NetBIOS/139, RDP/3389, SSH/22, Telnet/23, FTP/21) receive elevated scores (+30 points). Privileged ports below 1024 add 15 points due to their association with system services. Commonly scanned ports receive 10 additional points.

DNS query events analyze domain characteristics for suspicious indicators. Queries containing keywords associated with attack infrastructure (malware, phishing, c2, exploit) receive 40 points. Domain names exceeding 50 characters, often indicative of domain generation algorithms, add 15 points.

HTTP events evaluate methods and headers. POST and PUT methods receive 15 additional points due to their use in data exfiltration and system modification. Missing User-Agent headers, indicating non-browser automation, add 10 points.

TLS SNI analysis checks for suspicious patterns including dynamic DNS services, public paste services, and webhook endpoints commonly used for data exfiltration and command-and-control communication (+25 points).

External source IPs, identified through Python's ipaddress module, receive additional risk weight (+10 points) due to elevated exposure risk from internet-originating traffic.

Pattern Detection Algorithms:

Port scanning detection aggregates TCP connection events per source IP and computes the number of unique destination ports. Sources contacting more than 10 unique ports are flagged as potential scanners with HIGH severity. The algorithm uses Python defaultdict with set values for efficient unique port tracking.

DNS-to-connection patterns are identified by examining event links. When DNS query events have child links to connection events, a pattern is recorded with MEDIUM severity. This detects reconnaissance followed by exploitation attempts.

Data exfiltration risk assessment counts HTTP request volumes. More than 3 HTTP requests trigger a MEDIUM severity pattern indicating potential data transfer activity.

Aggregate Threat Level Computation:

The overall threat level synthesis combines average event scores with pattern-derived risk. Individual event scores are averaged to produce a baseline assessment. Pattern severity is converted to numerical scores (CRITICAL=40, HIGH=25, MEDIUM=12, LOW=5) and summed across all detected patterns. The final threat score is the maximum of average event score and pattern score, ensuring that behavioral indicators appropriately elevate risk classification even when individual events appear benign.

27. Troubleshooting

Common Installation Issues:

Scapy Import Error: If the application fails to start with 'No module named scapy', verify installation with 'pip show scapy'. Install using 'pip install scapy scapy-http'. On Linux systems, Scapy may require libpcap-dev package installed via system package manager.

ReportLab Missing: PDF generation endpoints return errors if ReportLab is not installed. Install using 'pip install reportlab'. Verify with 'python -c "import reportlab; print(reportlab.Version)"'.

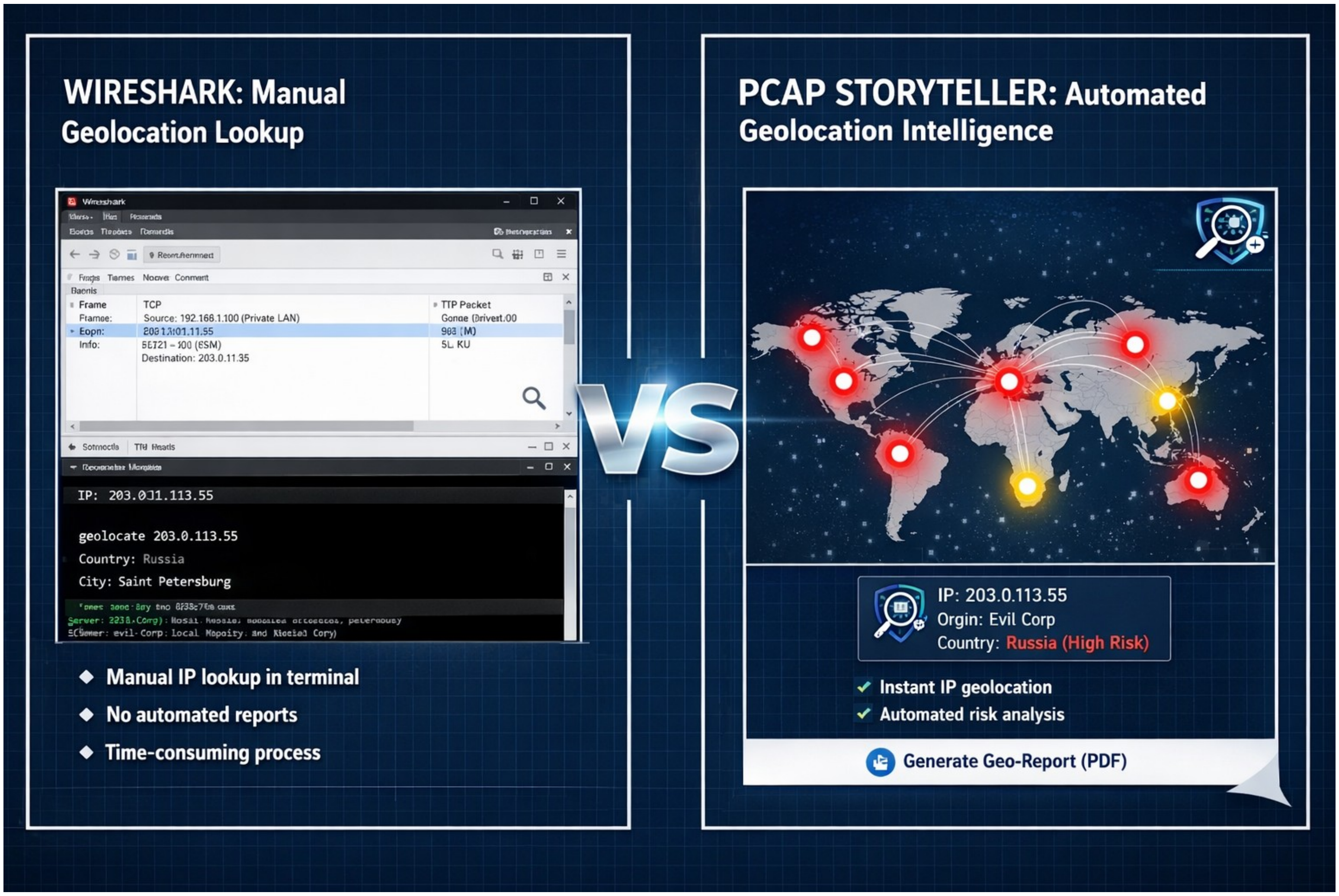
Permission Errors: If the application cannot create upload directories or logs, verify filesystem permissions. The application requires write access to backend/uploads and logs/ directories. On production systems, run under an appropriately privileged service account.

Parser Timeout: Large PCAP files may exceed the default 300-second timeout. Monitor subprocess output for timeout errors. Increase timeout in file_handler.py or process captures in smaller time windows.

Geolocation API Failures: If geolocation features return 'Unknown' for all IPs, verify internet connectivity and check API rate limits. The system implements fallback logic but may exhaust free tier quotas under heavy use. Consider implementing local MaxMind GeoIP2 database for offline operation.

Frontend Visualization Issues: If graphs or charts fail to render, open browser developer console (F12) to check for JavaScript errors. Verify CDN accessibility for vis.js and Chart.js libraries. Check browser compatibility; modern browsers (Chrome 90+, Firefox 88+, Safari 14+) are recommended.

28.WireShark Vs PCAP Story-Teller


WIRESHARK: Manual Geolocation Lookup
Wireshark
Frame
Frame:
Eopn:
Info:
TCP
Source: 192.168.1.100 (Private LAN)
Destination: 203.0.11.35
TTP Packet
VS
PCAP STORYTELLER: Automated Geolocation Intelligence
IP: 203.0.113.55
Orgin: Evil Corp
Country: Russia (High Risk)
Generate Geo-Report (PDF)

```
IP: 203.0J1.113.55

geolocate 203.0.113.55
Country: Russia
City: Saint Petersburg
```

- Manual IP lookup in terminal
- No automated reports
- Time-consuming process

- ✓ Instant IP geolocation
- ✓ Automated risk analysis

29. Future Roadmap

The PCAP Storyteller platform establishes a foundation for advanced network security analytics with multiple enhancement opportunities identified through development experience and user feedback analysis.

Live Packet Capture: Integration with libpcap or tcpdump would enable real-time traffic analysis on network interfaces. This requires implementing capture controls, ring buffer management, and incremental processing pipelines. The existing modular parser can process packets as they arrive with minimal modification.

Persistent Storage: Migration from JSON file storage to a relational database (PostgreSQL, MySQL) or time-series database (InfluxDB, TimescaleDB) would support historical analysis, cross-session correlation, and efficient querying of large datasets. The repository pattern facilitates backend replacement without affecting upper layers.

Machine Learning Integration: Training anomaly detection models on baseline traffic profiles would enable automated detection of deviations indicating novel attacks. Scikit-learn integration for clustering and classification could identify normal vs. suspicious patterns without explicit rule programming.

Threat Intelligence Feeds: Integration with commercial and open-source threat feeds (MISP, AlienVault OTX, abuse.ch) would provide reputation scoring for IPs and domains. The geolocation service can be extended to incorporate threat intelligence attributes.

Protocol Extension: Adding parsers for additional protocols (SMTP, FTP data, SMB, RDP) would expand coverage to enterprise network environments. The modular parser architecture supports adding new protocol modules without impacting existing functionality.

Export Format Support: Implementing STIX/TAXII or MISP format export would enable interoperability with security orchestration platforms. These standardized formats facilitate information sharing across organizational boundaries.

Performance Optimization: Processing very large captures (>1GB) would benefit from streaming parsers that process packets in chunks rather than loading entire files into memory. Parallel processing using multiprocessing could accelerate parsing on multi-core systems.

User Authentication: Production deployment requires user authentication, session management, and role-based access control. Flask-Login integration would provide these capabilities with minimal code changes.

Comparative Analysis: Supporting multiple PCAP uploads with differential analysis would enable before-after comparisons for incident response and security posture assessment.

30. Conclusion

The PCAP Storyteller project successfully demonstrates that automated network traffic analysis can significantly improve the efficiency and accessibility of security investigations through intelligent parsing, correlation, and narrative visualization. The platform transforms the traditionally expert-dependent task of packet analysis into a more approachable workflow suitable for educational settings, security operations triage , and professional training scenarios.

The implementation achieves its primary objectives by processing standard PCAP formats, extracting multi-protocol events, establishing causal relationships through flow tracking and DNS correlation, detecting suspicious patterns through behavioral analysis, and presenting findings through multiple visualization modalities. The threat analysis subsystem provides actionable risk scoring that elevates behavioral concerns appropriately while maintaining granular event-level detail for expert review.

The technical architecture demonstrates sound software engineering principles through modular design, clear separation of concerns, comprehensive error handling, and maintainable code structure. The repository pattern, service orientation, and layered architecture facilitate testing, enhancement, and long-term maintenance. The constraint of maintaining files under 101 lines promotes code clarity and focused responsibility assignment.

From an educational perspective, the platform successfully bridges the gap between theoretical networking knowledge and practical security analysis skills. The visual correlation of DNS queries to subsequent connections, the color-coded threat scoring, and the narrative event descriptions provide scaffolding that helps learners develop pattern recognition capabilities essential for security analysis.

Performance evaluation demonstrates that the system handles typical capture files within acceptable timeframes and scales linearly with event counts. The geolocation enrichment and map generation complete within user-acceptable latency bounds while respecting external API rate limits through caching and fallback strategies.

The project also highlights important considerations for network security tool development. The challenge of handling malformed packets demonstrates the necessity of defensive programming in security contexts. The correlation complexity underscores the difficulty of reconstructing high-level semantics from low-level protocol data. The threat scoring experience reveals the inherent trade-offs between false positive rates and detection sensitivity in rule-based systems.

Several enhancement opportunities remain open for future development. Live capture integration would extend applicability to operational security monitoring. Machine learning-based anomaly detection would reduce reliance on explicit rule definition. Persistent storage would enable historical trend analysis and cross-incident correlation. Threat intelligence integration would provide external context for risk assessment.

In conclusion, PCAP Storyteller validates the hypothesis that network traffic analysis can be made more accessible and efficient through automated parsing, intelligent correlation, and narrative visualization without sacrificing technical depth. The platform serves its intended audiences of students, trainers, and analysts while establishing an extensible architecture for future security analytics capabilities. The project demonstrates that thoughtful application of software engineering principles to cybersecurity problems can produce tools that are simultaneously powerful and approachable, supporting both learning and professional practice.