Threat Explorer: Technical and Design Study Report

BGN: 1782P

December 12, 2025

Word Count: 1483

1 USE-CASE AND GOALS

Threat Explorer is a chatbot for analyzing cybersecurity threats in a database using natural language, inspired by Stellar Cyber's AI Investigator [10]. The system lets security experts analyze data quickly, without deep knowledge of the schema or query language, in collaboration with AI. The database uses Inscribo's dataset from Kaggle [4], with 40,000 records and 25 columns (e.g., Timestamp, Source IP Address, Attack Type, Anomaly Score, IDS/IPS Alerts). Threat Explorer can answer queries like "show the last 10 attacks with an anomaly score over 75" or "show the number of high severity attacks by type." The system uses three agents: a custom LLM chain, a ReAct agent [13], and a multi-agent architecture. This paper explores the system in two main dimensions:

1. **Technical Robustness:** To evaluate the retrieval performance, speed, cost efficiency, and perceived usefulness of different agent architectures for response generation in dialogues.
2. **Design Effectiveness:** To experiment with different structured output strategies (plain text vs. charts) in terms of usability, helpfulness, and cognitive load in a user study.

2 BASELINE SYSTEM DESIGN

Threat Explorer is a **Retrieval-Augmented Generation (RAG)** [6] system. The system is powered by OpenAI's **GPT-4o mini** [7] LLM for its cost-effectiveness. The workflow involves: 1) receiving a natural language prompt, 2) an agent constructing an SQL query, 3) retrieving relevant records from the database by executing the query, and 4) an agent reporting the results. The baseline agent is a predefined LLM chain that has tools for executing SQL queries and checking the database schema.

3 TECHNICAL EXPERIMENTAL STUDY

3.1 Research Question

To what extent do different agentic architectures affect the accuracy, speed, cost, and perceived utility of the RAG-based dialogue system?

3.2 Setup and Evaluation Metrics

I annotated a test set of 10 dialogues with 30 system turns each, each containing a rubric and ground-truth SQL queries of varying complexity and use cases, such as attack analysis, protocol investigation, severity analysis, and temporal analysis. An example of the rubric structure is illustrated in Figure 1.

```
# ==================== DIALOGUE 9: IDS/IPS ALERTS ANALYSIS ====================
{
    "id": "d9",
    "title": "IDS/IPS Alert Investigation",
    "category": "alert_analysis",
    "description": "User investigates IDS/IPS alert patterns",
    "turns": [
        {
            "turn_id": 1,
            "user_message": "Show me attacks that triggered IDS/IPS alerts",
            "expected_query_pattern": r'SELECT.*"IDS/IPS Alerts".*LIMIT',
            "expected_visualization": "db-table",
            "rubric": {
                "query_validity": "Must filter for non-empty IDS/IPS alerts",
                "correctness": "Must return attacks with IDS/IPS alerts",
                "visualization": "Should use table for detailed records"
            }
        },
        {
            "turn_id": 2,
            "user_message": "What attack types generated these alerts?",
            "expected_query_pattern": r'SELECT.*"Attack Type".*COUNT',
            "expected_visualization": "db-chart",
            "rubric": {
                "query_validity": "Must count attack types",
                "context_awareness": "Should maintain IDS/IPS alert filter from turn 1",
                "visualization": "Should use chart for attack type distribution"
            }
        },
        {
            "turn_id": 3,
            "user_message": "Show me the network segments where these alerts occurred",
            "expected_query_pattern": r'SELECT.*"Network Segment".*COUNT',
            "expected_visualization": "db-chart",
            "rubric": {
                "query_validity": "Must count by network segment",
                "context_awareness": "Must maintain IDS/IPS alert context",
                "visualization": "Should use chart for segment distribution"
            }
        }
    ]
},
```

Figure 1: Example of a test rubric

Each test is evaluated on the following:

- **Retrieval Performance:** *Query Validity* (percentage of executable queries) and *Pattern Match Accuracy* (calculating the correctness of the query results).
- **Cost:** Token consumption and total price.
- **Speed:** Total time in seconds to respond.
- **Generation Quality (LLM-as-a-Judge):** Scores (1–5) on factuality, helpfulness, and overall quality.

3.3 Results

3.4 Discussion

The **ReAct agent** achieved 100% query validity, meaning that all generated SQL queries, produced through its iterative refinement and reasoning, were executable on the database. It also had the highest query pattern accuracy together with the LLM chain, achieving 86.7%. The **LLM Chain** had the fastest response times and the lowest token consumption and cost.

Table 1: Retrieval Performance Across Agents ($N = 30$ turns)

Metric	LLM Chain	ReAct	Multi-Agent
Query Validity	96.7%	**100.0%**	70.0%
Pattern Match	**86.7%**	**86.7%**	43.3%

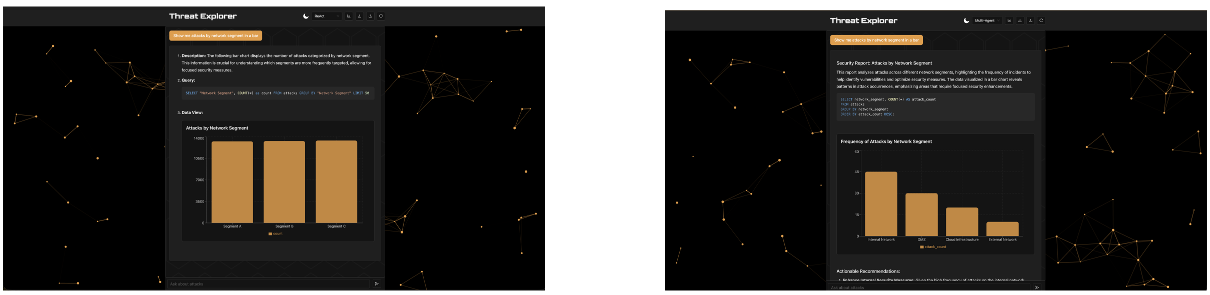
Table 2: Efficiency and Token Distribution Across Agents ($N = 30$ turns)

Metric	LLM Chain	ReAct	Multi-Agent
Total Time (s)	**306.9**	475.8	2741.4
Input Tokens	**84,265**	190,517	2,047,357
Output Tokens	**13,275**	23,259	444,435
Total Tokens	**97,540**	213,776	2,491,792
Total Cost ($)	**$0.0206**	$0.0425	$0.5738

Table 3: Generation Quality (LLM-as-a-Judge Score, 1–5)

Metric	LLM Chain	ReACT	Multi-Agent Value
Factuality	**4.67**	4.53	3.70
Helpfulness	4.40	**4.57**	4.17
Overall Quality	4.37	**4.47**	3.67

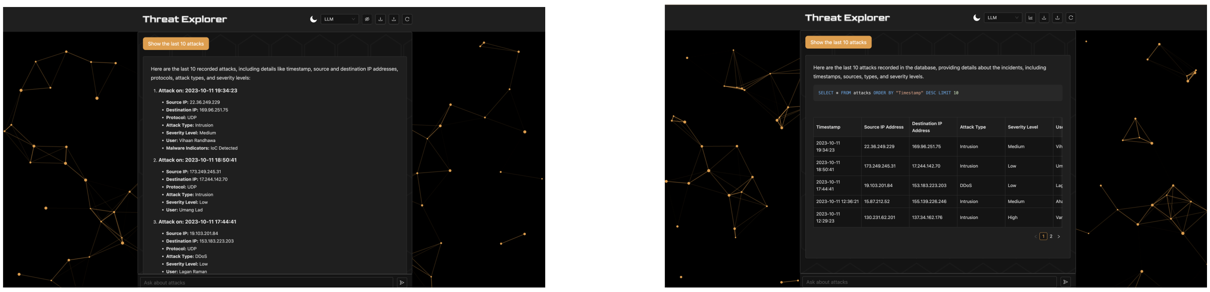
The multi-agent architecture performed the worst across all metrics due to poor orchestration. However, it was not a priority to improve it since the simpler agents already achieved high accuracy while being cheaper and faster. The LLM judge [3] scored the LLM chain highest for factuality but favored ReAct for helpfulness and overall quality. The trade-offs between agents motivate a system design that supports switching based on preference, requirements, and budget. Figure 2 shows an example where the multi-agent hallucinated the output convincingly, whereas the ReAct agent provided accurate results given identical prompting. The multi-agent orchestration consists of an SQL Query Analyst, a Cybersecurity Threat Analyst, and a Report Formatter specialized agents. The system prompts of all agents specify the tools, schema, reasoning, role, and expected markup format and can be found in the repository.



(a) ReAct Correct Result

(b) Multi-Agent Hallucination

Figure 2: Comparison of ReAct vs. Multi-Agent outputs for the same query



(a) Text-Only Response (b) Chart Response

Figure 3: Comparison of Text-Only vs. Chart-Based Responses in the Design Study

4 DESIGN-FOCUSED EXPERIMENTAL STUDY

4.1 Research Question

Does different structured output, such as using LLM-generated visualizations versus text-only, affect the usability, helpfulness, clarity, confidence, and cognitive load of a cybersecurity data analytics chatbot?

4.2 Setup

The experiment included a header button to toggle visualizations, which update the agents' structured output. Twelve postgraduate Cambridge students were given the chatbot with a **text** (plain-text summary) and a **chart** (visual data presentation) for 5 minutes each. Half of the students had undergraduate degrees in computer science, and none had formal cybersecurity experience. The post-interaction Google Forms survey included 5 questions, each on a 5-point Likert scale, comparing the usability, helpfulness, clarity, trust, and efficiency of the two systems, and a final question asking which configuration was preferred.

Table 4: Post-Interaction Survey Questions

Dimension	Survey Question	Measurement Type
A. Usability	The responses were presented in a well-organized way.	Likert 1-5
B. Helpfulness	The system gave me the right level of detail.	Likert 1-5
C. Clarity	I could identify the key evidence supporting the conclusion.	Likert 1-5
D. Trust	I trust the chatbot's output for the tasks I performed.	Likert 1-5
E. Efficiency	This version helped me understand the output quickly.	Likert 1-5
F. Preference	Which version do you prefer?	Chart vs. Text

4.3 Results and Statistical Analysis

Analysis of the Likert-scale responses using a one-sided Wilcoxon Signed-Rank Test ($\alpha = 0.05$) given a priori of an expected improvement and small sample size showed the chart version had higher scores across all dimensions and statistical significance ($p < 0.05$) after Holm-Bonferroni adjustment in usability, clarity, and efficiency.

4.4 Discussion

The results show that **structured visual output** provides user experience value in Threat Explorer, supporting the decision to make visualizations the default output setting.

Table 5: Design Experiment Results with Holm-Bonferroni Correction (1–5 Likert Scale)

Dimension	Mean T	Mean C	SD T	SD C	p	p_{adj}
A. Usability	3.42	**4.67**	0.79	0.49	0.002	0.008*
B. Helpfulness	3.50	4.42	1.09	0.67	0.045	0.078
C. Clarity	3.17	**4.50**	0.83	0.67	0.002	0.008*
D. Trust	3.83	4.58	0.94	0.67	0.039	0.078
E. Efficiency	2.58	**4.75**	1.31	0.45	0.001	0.005*

Note: p = raw p-value; p_{adj} = Holm-Bonferroni adjusted p-value. *Significant at $\alpha = 0.05$.

4.4.1 Socio-Technical Reflection and Trust

Trust and **Helpfulness** in the chart version require critical reflection, as users may perceive data visualizations as more trustworthy than plain text, which can be misleading in an LLM-powered system that generates inaccurate or incomplete queries, hallucinates explanations, or misinterprets data. This may be especially true for users without domain expertise, who may over-rely on the visually compelling output, leading to unquestioned acceptance of the information.

4.4.2 Mitigation and Transparency

To improve transparency, agents always include SQL queries used in the output and support syntax highlighting so the queries are inspectable, as seen in the dialogues in Figure 4. Furthermore, the agents provide an explanation of their thinking. In future designs, the query should be editable so users can explore the data even if the agents are unable to fulfill the requirements. Interactive features will change the user's role from a passive recipient to an active validator, promoting human-in-the-loop ML [12] and human-AI collaboration [11].

5 FINAL SYSTEM DESIGN

The final system can switch between agents. It supports logging (Figure 5), uploading, and downloading dialogues in JSON format. The backend is a FastAPI [2] server using LangChain [5] and CrewAI [1] for orchestration. The interface is made with React [9]. The database is SQLite3 [8]. The repository includes documentation for running and scripts for evaluations and report generation. Figure 4 shows example dialogues.

6 CONCLUSION

This work explored technical and design aspects of Threat Explorer, a conversational AI cybersecurity analysis tool. The technical study showed that while the ReAct agent is most reliable, the LLM Chain is most efficient, leading to the design decision to support switching between agents. The design experiment yielded statistically significant results showing that visualizations enhance usability, helpfulness, clarity, trust, and efficiency, making them a core feature of Threat Explorer. While helpful, these visualizations pose a risk of misleading and inaccurate LLM-generated content. Threat Explorer aims to improve transparency and mitigate over-reliance by providing its thought process and SQL queries. Future work should focus on stronger guardrails against unrelated or dangerous prompts, editable inline SQL queries and charts, support for more chart types, faster response times by sending completed segments over WebSockets, adaptive agent routing, improved prompting and orchestration, short and long-term memory, and observability.

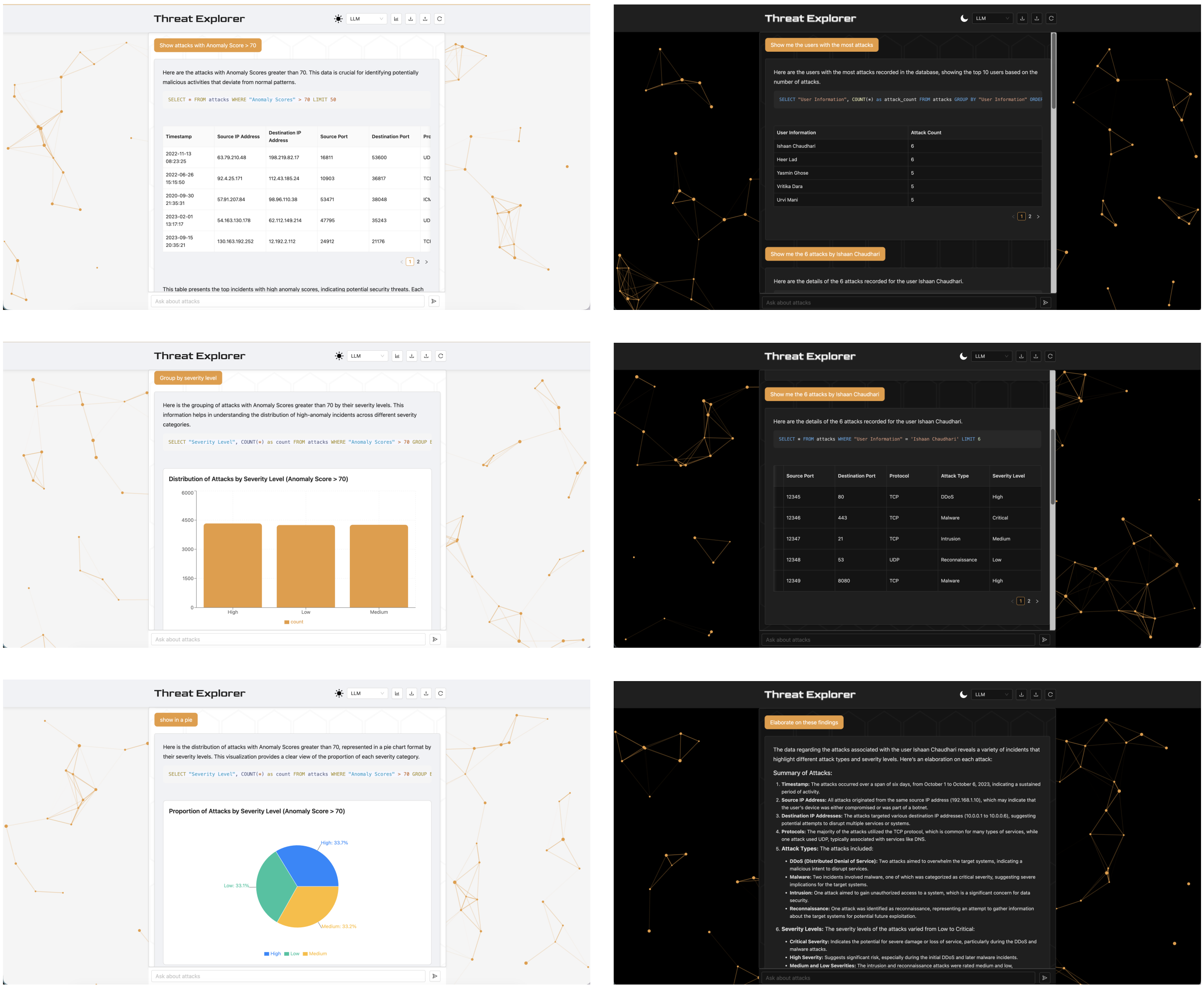


Figure 4: Dialogue Examples

Figure 5: Logs Output from Threat Explorer

References

[1] CrewAI. "CrewAI - Platform for Multi AI Agents Systems." `https://www.crewai.com/`.

[2] FastAPI. "FastAPI Documentation." 2023. `https://fastapi.tiangolo.com/`.

[3] Gu, Jiawei, et al. "A Survey on LLM-As-a-Judge." ArXiv, 2024. `https://arxiv.org/abs/2411.15594`.

[4] Incribo. "Cyber Security Attacks." Kaggle.com, 2024. `https://www.kaggle.com/datasets/teamincribo/cyber-security-attacks`.

[5] LangChain. Chase, Harrison. "LangChain." GitHub, 1 Oct. 2022. `https://github.com/langchain-ai/langchain`.

[6] Lewis, Patrick, et al. "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks." ArXiv, 2021. `https://arxiv.org/abs/2005.11401`.

[7] OpenAI. "GPT-4o Mini: Advancing Cost-Efficient Intelligence." 18 July 2024. `https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/`.

[8] Python Software Foundation. "sqlite3 — DB-API 2.0 Interface for SQLite Databases." Python 3 Documentation, 2024. `https://docs.python.org/3/library/sqlite3.html`.

[9] React. Meta Open Source. "React." 2025. `https://react.dev/`.

[10] Stellar Cyber. "AI Investigator - Natural Language Threat Hunting with XDR." 4 Dec. 2025. `https://stellarcyber.ai/ai-investigator-natural-language-threat-hunting/`. Accessed 10 Dec. 2025.

[11] Vats, Vanshika, et al. "A Survey on Human-AI Teaming with Large Pre-Trained Models." ArXiv, 2024. `https://arxiv.org/abs/2403.04931`.

[12] Wu, Xingjiao, et al. "A Survey of Human-In-The-Loop for Machine Learning." *Future Generation Computer Systems*, vol. 135, Oct. 2022, pp. 364–381. `https://doi.org/10.1016/j.future.2022.05.014`.

[13] Yao, Shunyu, et al. "ReAct: Synergizing Reasoning and Acting in Language Models." 2022.