CyberCell Report

Topic: AI driven and Rule based Defence tactics in Web Attacks Group: **ZeroDay BHOS**
Students: Mahammad Babayev, Emil Gallajov, Emil Savzixanov Date: 19.10.2025

1 Overview

In this lab we analyze processes that occur during web attacks and present solutions capable of detecting and responding to them. The core goal of our work is twofold: first, to define detection rules over logs in a rule-based manner, and second, to validate and refine those rule-based detections at a high level using AI. To achieve this we employed two effective approaches which complement each other and improve overall detection accuracy and robustness.

2 Dataset Review

Our datasets consist primarily of two types:

- **Labeled Web Log Dataset:** This dataset is composed of labeled logs collected from the simulated web application. It is particularly effective at identifying NoSQL injection and open redirection attacks when rule-based signatures are applied.

- **Flow Logs Dataset:** A well-known, large flow-based dataset with many features. As shown in the reference materials, this dataset enables precise attack classification — in our experiments it allowed correct identification of attack type in approximately 95

3 Web and Attack Simulations

To generate realistic data we deployed a web server (the `Shopeasyapp` project) which is available on GitHub for convenience and review. In the application we intentionally injected several vulnerabilities to be used for testing:

- Open Redirection

- NoSQL Injection

- Rate-limit bypass / abuse

- IDOR (Insecure Direct Object References)

- Additional miscellaneous weaknesses

We executed attack scenarios against these intentionally vulnerable endpoints and collected the resulting logs for model testing. The specific locations of the vulnerabilities and the code used to create them are documented in the GitHub repository — you can browse the repository to inspect the code and reproduce the simulations manually.

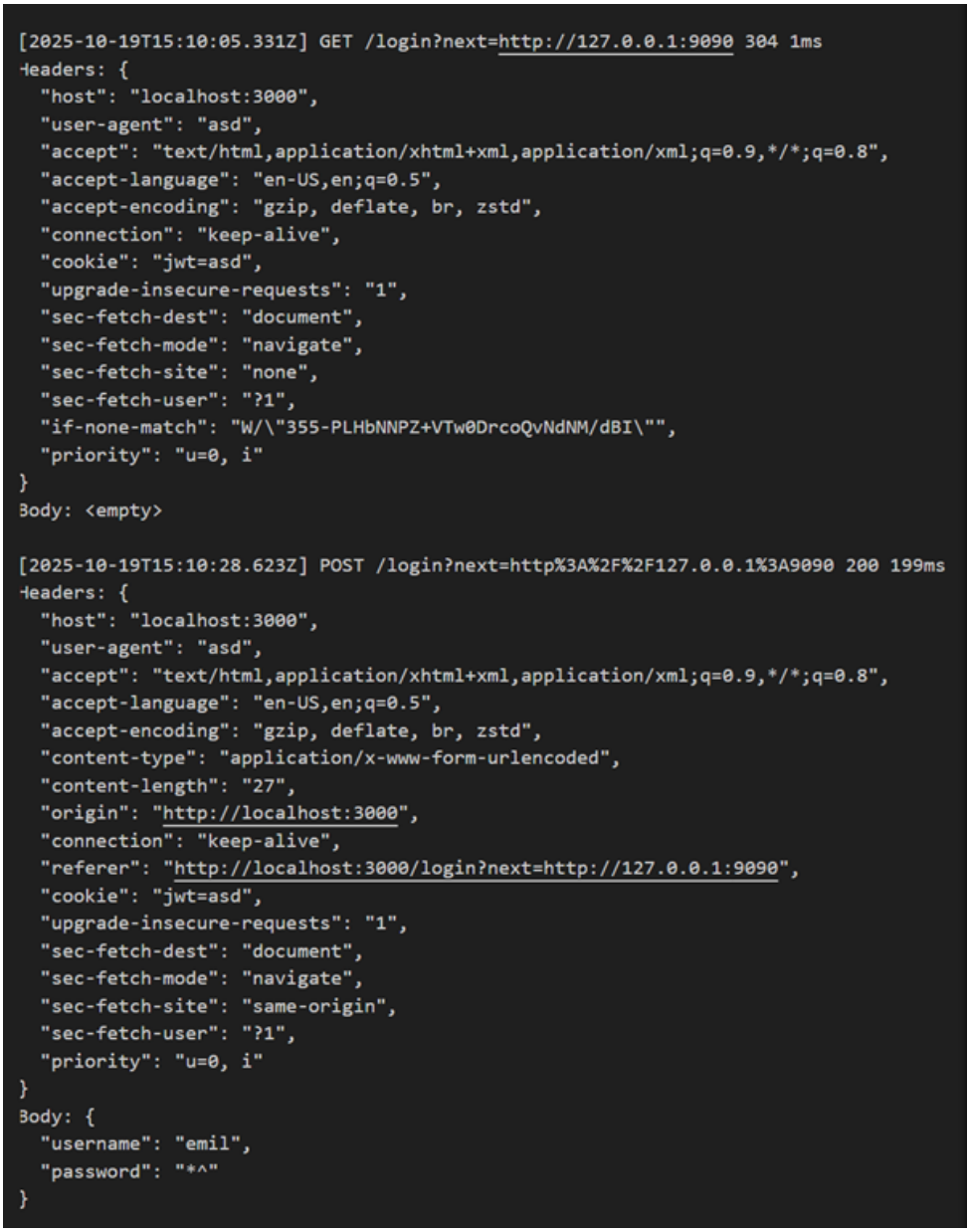

Figure 1: Example of collected logs from vulnerable endpoints (placeholder).

4 AI-driven and Rule-based Solutions

We developed a hybrid detection pipeline that works on two data modalities:

1. Raw web logs (HTTP request logs, application logs)

2. Network flow logs (processed NetFlow/Zeek flow records)

This architecture allows us to leverage lightweight, interpretable rule-based checks on raw logs and more feature-rich ML models on flow data.

4.1 Models on Raw Logs

On raw logs we focused primarily on detecting NoSQL injection and open redirection attempts. From observed attack patterns we derived a set of manual rules. These rules can immediately flag suspicious requests and often indicate the likely attack type.

```python
def detect_attack(log_line, top_k_faiss=50, similarity_threshold=0.4):
    rule_label = None
    if any(k in log_line.lower() for k in ["$regex", "$ne", "$gt", "$where", "$in"]):
        rule_label = "nosql_injection"
    elif any(k in log_line.lower() for k in ["redirect", "next=", "url=", "http", "https"]):
        rule_label = "open_redirect"
```

Figure 2: Example rule-based signatures for NoSQL and open-redirect detection (placeholder).

However, a pure rule-based approach produced false positives and overlapping matches in some scenarios. To reduce misclassification we introduced a final decision layer that combines: Operationally the decision flow works as follows:

1. Evaluate rule-based detection first.
2. Run the ML classifier on the same sample.
3. Compute a similarity score using an encoder-based model that measures closeness to known attack examples.
4. If the AI model confidence is high, prefer the AI decision; if AI confidence is low but the rule indicates an attack, keep the rule decision; otherwise use a combined judgement that balances rule, ML, and similarity outputs.

Technically, the similarity model uses an encoder to produce compact embeddings of log entries (or request payloads). This enables reliable nearest-neighbor style checks against a labeled example set. In our tests the combined system produced a very low error rate: the ML component reached up to (98) accuracy on test sets, and the ensemble decision significantly reduced false positives compared to rules alone.

	log	rule_based	similarity_based	similarity_score	final_decision
0	GET /index.html	None	safe	-5.101076	safe
1	{"username":{"$ne":""}, "password":{"$ne":""}}	nosql_injection	NoSQL	7.764023	nosql_injection
2	GET /login?next=https://evil.com	open_redirect	safe	-7.454884	open_redirect
3	/api/search?query=$where: 'return true'	nosql_injection	NoSQL	5.961707	nosql_injection
4	Normal user visiting home page	None	safe	-10.642330	safe

Figure 3: High-level decision pipeline combining rules, ML, and similarity models (placeholder).

```
 Log: {"username":{"$ne":""}, "password":{"$ne":""}}
→ Class: NoSQL  (63.73% ehtimal ilə)

 Log: GET /login?next=https://evil.com
→ Class: open_redirect  (72.53% ehtimal ilə)

 Log: /api/search?query=$where: 'return true'
→ Class: NoSQL  (84.78% ehtimal ilə)

 Log: Normal user visiting home page
→ Class: 0  (49.44% ehtimal ilə)
```

Figure 4: Detailed flow: confidence thresholds and tie-breaking logic (placeholder).

4.2 Prediction on Flow Logs

Flow-based analysis produced the best overall classification performance but required significantly more preprocessing. To use flow data, raw logs were converted to flow records using `Zeek` and similar tooling. After conversion, the dataset underwent extensive cleaning:

- Removal of irrelevant or redundant features,
- Feature engineering (aggregation, time-window statistics),
- Encoding of categorical attributes,
- Careful train/validation/test splitting to avoid leakage.

After this processing the Random Forest classifier was selected as the best-performing algorithm. In our experiments the Random Forest achieved approximately (98) on combined test/validation metrics for some attack classes). The abundance of high-quality features and careful cleaning were key contributors to this performance.

The primary attack types detected in flow data included brute force, DDoS, DoS, web attack, and others

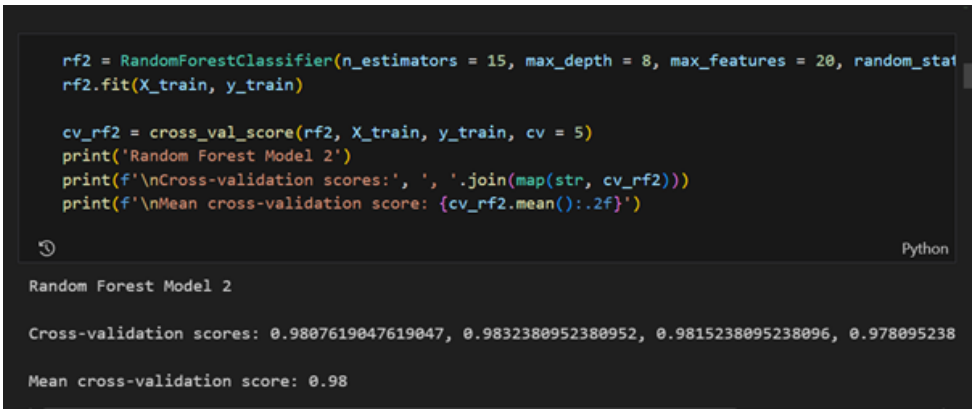

Figure 5: Flow-based model performance and feature importance (placeholder).

5 Results and Discussion

- Rule-based detection provides fast, interpretable signals but suffers from false positives and overlapping rules.
- Combining rule-based signals with ML and similarity models reduces false positives and improves overall decision quality.
- Flow-based models (after Zeek conversion and heavy preprocessing) produced the highest accuracy; Random Forest performed best in our experiments.
- The hybrid approach (raw-log rules + AI + flow-based ML) offers a practical balance between speed, interpretability, and accuracy for real-world web-attack detection.

6 Conclusion

This project demonstrates an effective hybrid defense strategy against web attacks by combining rule-based signatures with AI-driven verification and flow-based machine learning. The rule engine offers immediate, explainable alerts; AI models (classifier + similarity encoder) validate and refine those alerts to lower false positives; and flow-based classifiers provide the highest-accuracy classification after comprehensive preprocessing. For deployment we recommend:

- Operating the rule engine as a low-latency first filter,
- Running ML/similarity checks asynchronously (or in parallel) to confirm and enrich alerts,

- Maintaining a regularly updated labeled example store for the similarity model,
- Continuously monitoring feature drift in flow data and retraining models periodically.

Together, these measures form a resilient detection stack capable of handling both known patterns (rules) and evolving attack behaviors (AI).

References

- Canadian Institute for Cybersecurity (CIC). *CICIDS2017 — Intrusion Detection Evaluation Dataset.* Available: `https://www.unb.ca/cic/datasets/ids-2017.html`
- Project repository / simulation code (Shopeasyapp):`<https://github.com/EmilGallajov/Vulnerable-ShopEasy-`

If you have any questions, please let us know.