SWAVLAMBAN 2025 HACKATHON CHALLENGE – 3

DEVELOPMENT OF A MACHINE LEARNING ENABLED NETWORK ANOMALY DETECTION MODULE FOR FUTURE INTEGRATION WITH WEB APPLICATION FIREWALL (WAF)

1. **Challenge Overview**. Modern organisations rely heavily on web applications and APIs, making them targets for sophisticated attacks such as zero-day exploits, bot-driven intrusions, API abuse, and multi-stage threats. WAFs form the first layer of defence in such a use case scenario. Traditional WAFs rely heavily on static, signature-based rules that are labor-intensive to maintain and ineffective against evolving threats.

2. Encrypted traffic, microservices architectures, and unpredictable behaviour patterns have increased difficulty in distinguishing legitimate anomalies from malicious activity. Security teams face severe operational challenges such as rule fatigue, high false positives, and reactive patching cycles. The challenge is to augment Machine Learning capabilities in an existing Web Application Firewall (WAF) which will combine traditional rule-based filtering with intelligent ML-driven analysis to detect anomalies, discover new attacks, and autonomously recommend new security policies.

3. **Core Technical Objectives**. Participants must develop a Machine Learning module capable of Network baselining and anomaly detection which can be integrated with an open source WAF for future use by means of API calls or any other means faesible.

3.1 **ML-Module**. The Machine Learning module should be capable of inspecting all inbound/ outbound traffic including HTTP(S) content, baselining network traffic and perform behavioral analysis along with anomaly detection. The module should incorporate a GUI for administrators to view and analyse the reports/ recommendations generated.

3.2 **Adaptive Anomaly Detection**. Develop ML model capable of learning normal traffic baselines and identifying deviations that indicate malicious behaviour. Support supervised, unsupervised, and semi-supervised approaches, with explainable output.

3.3 **Automated Security Rule Recommendation**. Convert ML insights into human-readable security rules that administrators can approve and deploy. The system must integrate seamlessly with existing rule logic.

3.4 **High-Performance, Low-Latency Operation**. Ensure real-time inspection of encrypted and unencrypted traffic, with minimal latency overhead, suitable for high-throughput production environments.

3.5 **Continuous Learning Framework**. Include mechanisms for periodic retraining, administrator feedback loops, and log-driven learning pipelines to improve accuracy and reduce false positives over time.

4. **Simulation and Scenarios**. Participants will be evaluated through controlled scenarios representing real-world deployments:-

4.1 **Baseline Traffic Scenarios.** Mixed legitimate traffic with periodic anomalies to test ML baseline accuracy.

4.2 **Encrypted Traffic Handling Scenarios**. Routing HTTPS traffic through decryption layers or TLS termination to test analysis capabilities.

4.3 **Zero-Day Attack Scenarios**. Describe as to how the developed solution would be resilient to Zero day attacks.

4.4 **API Abuse and Bot Traffic Scenarios**. Stealthy, behavioural attacks targeting APIs or simulating automated bot patterns.

5. **Evaluation and Scoring**

5.1 **Primary Score Components**

- **Detection Accuracy:** Ability to detect known and unknown threats.

- **False-Positive Rate:** Stability under high volumes of legitimate irregular traffic.

- **Performance:** Throughput, latency, and scalability under stress and ease of use.

- **Explainability:** Clarity and usefulness of ML-generated insights.

- **Rule Recommendation Quality:** Accuracy and relevance of generated policy suggestions.

5.2 **Pass/Fail Gates**

- Real-time detection of anomalies.
- User Friendly and informative dashboard.
- Integration of ML outputs into rules.
- Stable performance at scale.
- Meaningful explainability for all ML generated alerts.

6. Final Deliverables

6.1 **Fully Functional ML Module**. Complete ML module with dashboard and interfaces provided for integration with open source WAF. To be submitted as part of the private repository on GitHub classroom.

6.2 **Source Code**. Complete source code with clear directory structure, comments and coding standards. Build scripts and a README explaining how to build and run the solution. To be submitted as part of the private repository on GitHub classroom.

6.3 **Demonstration Video (5 minutes)**. Showcasing anomaly detection, live traffic analysis, rule recommendations, and dashboards. *The first round of evaluation would be completely based on the video submitted.*

6.4 **Technical Documentation (2–3 pages)**. Covering architecture, ML models, data pipeline design, rule-integration logic, and performance considerations. To be submitted as part of the Product Description Document.

6.5 **Logs, Metrics & Reports**. Including anomaly timelines, ML decisions, accuracy measurements, false-positive statistics, and rule recommendation outputs. To be submitted as part of the private repository on GitHub classroom.

6.6 **Presentation (8-10 slides)**. Short presentation containing problem statement, solution approach, architecture, Key features, demonstration summary, challenges faced and future enhancements.

Note – Participants may submit their solutions that may incorporate all/ some of the abovementioned features. Reasons for noncompliance may be indicated.

The Code may be submitted in the mentioned link-
https://classroom.github.com/a/bH96jorA
You need to submit your GitHub user name and link to the private repository provided by the organizer via the GitHub Classroom.