Final Report

Penetration Testing and Log Anomaly Detection for OWASP Juice Shop

Submitted by:

Mohamed Ehab Zaghloul	120220090
Amr Khaled Mohamed	120220236
Abdelrahman Ashraf	120220292
Mohamed Khaled Yahya	120220081
Omar Khalifa Abdelmonem	120220201

Faculty of Engineering
Computer Science and Engineering Department
Egypt-Japan University of Science and Technology (E-JUST)

June 8, 2025

Contents

1 Problem Statement

Modern web applications are increasingly exposed to sophisticated cybersecurity threats due to the growing complexity of their architectures and user interactions. Vulnerabilities in these systems can lead to severe consequences including data breaches, unauthorized access, privilege escalation, and service disruption. Detecting and mitigating such threats remains a critical challenge, especially in real-time environments.

To explore and address these challenges, OWASP Juice Shop—a deliberately insecure web application—was used as a testing platform. This platform is widely adopted in cybersecurity training and ethical hacking exercises due to its comprehensive range of vulnerabilities mimicking real-world attack vectors.

The core problem investigated in this project is twofold:

1. **Manual Exploitation and Documentation:**
   Real-world security vulnerabilities must be manually identified, exploited, and thoroughly documented. This includes discovering flaws such as SQL injection, broken access control, authentication bypass, and sensitive data exposure. The goal is to emulate the behavior of an attacker and understand the risk impact of each issue.

2. **Automated Log-Based Anomaly Detection:**
   Once exploitation occurs, the system generates logs capturing these abnormal behaviors. Detecting such anomalies manually from logs is impractical at scale. Therefore, AI-based models need to be developed and evaluated to automatically recognize suspicious or malicious patterns in system logs. These models must operate effectively without relying on labeled attack data and adapt to dynamic log formats.

This dual approach aims to bridge manual penetration testing with automated machine learning detection, offering a full-spectrum response to modern web application security.

2 Our Approach

To address the challenge of securing modern web applications from both offensive and defensive angles, our team adopted a dual-track strategy—dividing into two specialized subgroups that collaborated under a unified vision. This approach mirrored real-world cybersecurity workflows, aiming to bridge practical penetration testing with intelligent anomaly detection.

2.1 Team Division and Responsibilities

Exploitation Team (Penetration Testing and Vulnerability Discovery):

This team simulated real-world attacks on OWASP Juice Shop—a deliberately vulnerable web application. They conducted extensive penetration testing, exploiting a total of 27 unique vulnerabilities using tools such as Burp Suite, SQLMap, and Gobuster. These exploits covered a wide range of critical security issues:

- Authentication bypass
- SQL injection and NoSQL injection

- XSS (Cross-Site Scripting)
- Broken access control
- Password reset manipulation
- Sensitive data exposure
- Privilege escalation through role tampering

Each vulnerability was documented with detailed write-ups, request headers, payloads, impact analysis, and practical steps for reproduction. This manual process provided the ground-truth dataset for later machine learning validation.

Modeling Team (Log-Based Anomaly Detection with Deep Learning):

The second team took on the responsibility of detecting anomalies in server logs that correspond to the above exploits, simulating the role of a Blue Team using AI tools.

The modeling path started with the goal of implementing VoBERT, a state-of-the-art vocabulary-free Transformer model designed to detect anomalies in dynamic log formats. However, due to complex dependencies and computational challenges, the team was unable to successfully train VoBERT.

To compensate and deepen their understanding, the team produced a comprehensive survey paper comparing VoBERT, LogBERT, and other models (LogAnomaly, DeepLog, LogRobust). This study:

- Traced the evolution of log anomaly detection models from statistical to Transformer-based approaches.
- Compared architectural principles such as masked language modeling and hypersphere loss.
- Analyzed trade-offs in precision, interpretability, robustness to noise, and real-time deployment.

Based on insights from the survey, the team shifted focus and successfully implemented LogBERT, a self-supervised Transformer-based model. The entire pipeline was executed:

- Log preprocessing using the Drain parser
- Sequence encoding of log keys
- Training LogBERT on normal data from the HDFS dataset
- Evaluating detection performance using metrics like ROC-AUC, Precision, Recall, and F1-score
- Benchmarking against DeepLog and LogAnomaly models

LogBERT was found to be the most effective, achieving the highest recall and F1-score while maintaining low false positives, confirming its superior ability to detect unseen attacks.

2.2 Unified Methodology and Path

Although working on different domains—manual exploitation vs AI detection—both teams followed a mirrored process:

- **Behavioral Analysis:** Both teams started by understanding user interactions and system behavior within Juice Shop.
- **Evidence Collection:** The Exploitation Team produced real attacks, while the Modeling Team trained LogBERT to detect such patterns based on logs.
- **Cross-Reference:** The ultimate aim was to connect specific exploits (e.g., SQL injection) with their corresponding log traces and have the LogBERT model recognize them without prior labeling.
- **Planned Integration:** In the next phase, the logs captured during penetration testing will be parsed and input into the trained LogBERT model. This will demonstrate whether the AI system can successfully identify and classify the attacks executed by the Exploitation Team—completing the loop between offensive actions and automated defensive recognition.

2.3 Strategic Value

This unified methodology allowed the team to cover:

- Realistic attack simulation
- Ground-truth creation for anomalies
- Full-stack implementation of an unsupervised detection pipeline

The project not only reflects industry practices in Red/Blue teaming but also contributes to academic research by applying and evaluating cutting-edge AI models like LogBERT in realistic attack scenarios.

3 Penetration Testing and Exploitation Results

The Exploitation Team conducted a full-scale security analysis on OWASP Juice Shop, a purposely vulnerable application widely used in cybersecurity education. A total of 27 exploits were successfully identified, reproduced, and documented. These were classified into six major vulnerability categories.

3.1 Authentication Bypass

This category focuses on exploiting flaws in how the application authenticates users. Attackers were able to gain access without valid credentials.

Representative Exploits:

- **Exploit 1: SQL Injection Login Bypass**
  - Intercepted login request using Burp Suite

  - Modified email field with `' OR 1=1 --`
  - Received valid JWT token and accessed admin dashboard

- **Exploit 2: JWT Tampering**
  - Decoded the JWT token
  - Changed `"role": "admin"` and re-encoded without verifying signature
  - Bypassed access control mechanisms

Additional Exploits in This Category:

- Exploit 11: Reset Jim's Password
- Exploit 12: Reset Bender's Password
- Exploit 17: Simulate Login for Non-existent User

Log Traces Left: Suspicious login patterns, malformed inputs, JWT role mismatches.

3.2 Broken Access Control

Attackers were able to access or manipulate resources that should have been protected.

Representative Exploits:

- **Exploit 4: Add Item to Another User's Basket**
  - Found and used another user's `basketId`
  - Modified POST request to insert unauthorized item
- **Exploit 25: Product Tampering via PUT API**
  - Discovered insecure PUT endpoint
  - Modified product link to inject unauthorized content

Additional Exploits:

- Exploit 5: Review Forgery
- Exploit 14: Role Escalation
- Exploit 26: Reset Password via Security Question Brute Force

Log Traces Left: Unusual object ID access patterns, cross-user actions.

3.3 Sensitive Data Exposure

Data that should be private was accessed through weak endpoints, backup files, or insecure directories.

Representative Exploits:

- **Exploit 6: Exposed FTP Directory**
  - Accessed `/ftp/` and found confidential file list
- **Exploit 27: Forgotten Developer Backup**
  - Used %00 null byte injection to bypass extension filtering
  - Accessed `.bak` file with config and credentials

Additional Exploits:

- Exploit 7: Accessing Sensitive .bak Files
- Exploit 8: Order Tracking Email Leak
- Exploit 22: Access Logs Contain PII

Log Traces Left: Access to rare file types, null byte encoding, strange URL paths.

3.4 Cross-Site Scripting (XSS)

Client-side scripts were injected and executed via DOM vulnerabilities.

Representative Exploit:

- **Exploit 9: DOM-Based XSS in Search Field**
  - Inserted `<iframe src="javascript:alert('xss')">`
  - Triggered alert popup in victim's browser

Log Traces Left: Log parameters containing `<`, `>`, `script`, `iframe`.

3.5 Insecure Business Logic

These exploits took advantage of flaws in how logic was implemented rather than technical misconfigurations.

Representative Exploits:

- **Exploit 10: Bypass CAPTCHA to Spam Feedback**
  - Reused CAPTCHA token to submit feedback 18 times via Burp Intruder
- **Exploit 24: Expired Coupon Abuse**
  - Manipulated JavaScript time locally to apply expired discount

Additional Exploits:

- Exploit 23: Brute-Force Login for Amy
- Exploit 21: NoSQL Injection Attempt (Partial)

Log Traces Left: Repeated feedback attempts, manipulated timestamps, rapid-fire requests.

3.6 Database Schema & Injection Discovery

These exploits were used to gain insight into the application's database and internal structure.

Representative Exploits:

- **Exploit 13: Database Schema Enumeration**
  - Used UNION SELECT payloads in search field
  - Identified `sqlite_master` and extracted schema info
- **Exploit 16: SQL Injection in Search Bar**
  - Used input like `banana'` to trigger SQL error
  - Identified vulnerable backend structure

Log Traces Left: SQL error messages in logs, unexpected SELECT/UNION patterns.

3.7 Summary

Category	Number of Exploits
Authentication Bypass	5
Broken Access Control	6
Sensitive Data Exposure	5
Cross-Site Scripting	1
Insecure Logic Abuse	4
Database Exploration	6
Total	**27**

Table 1: Exploit Distribution Across Categories

All 27 exploits were confirmed using real tools, and each left log artifacts that can be used by anomaly detection models (like LogBERT) to identify future attacks. For more detailed information on the exploitation procedures, payloads, and step-by-step reproduction, please refer to the dedicated **Exploitation Report**.

4 From Exploits to Log-Based Detection

This section demonstrates how the manually executed exploits on OWASP Juice Shop reflect in server logs and how these traces can be used to detect abnormal behavior. This analysis is entirely based on the exploits conducted and documented by the Exploitation Team, using the log evidence resulting from those activities.

4.1 How Exploits Appear in Logs

Each vulnerability exploited during penetration testing results in one or more abnormal log traces. These logs contain metadata about HTTP requests, including endpoints accessed, payloads used, session behaviors, and server responses. The structure and frequency of these entries help in identifying deviations from expected system behavior.

The anomalies that signal exploitation often include:

- Access to admin or hidden endpoints without proper authentication
- Repeated requests to the same endpoint with varying payloads (fuzzing)
- Suspicious query parameters containing encoded characters or SQL expressions
- Abnormal status code sequences (e.g., multiple 403s followed by a 200)
- Role escalation or data tampering actions inconsistent with user permissions

Such patterns represent the kinds of log sequences that anomaly detection models like LogBERT aim to learn. The structure and frequency of these patterns are derived directly from the log files captured during the penetration testing phase.

4.2 Examples from the Exploited Vulnerabilities

The following examples illustrate how specific exploits executed against Juice Shop map directly to detectable log patterns:

1. **SQL Injection – Login Bypass**
   - **Exploit:** Used `admin'--` in the login field to bypass authentication.
   - **Log Pattern:** A POST request to `/rest/user/login` with a suspicious payload followed by a 200 OK response without valid credentials.
2. **Broken Access Control – Basket Manipulation**
   - **Exploit:** Modified the basket ID to access another user's cart.
   - **Log Pattern:** PUT request to `/api/BasketItems/{id}` with unauthorized access, no prior session escalation in the logs.
3. **Sensitive Data Exposure – Backup File Access**
   - **Exploit:** Accessed `/ftp/coupons_2013.md.bak` via direct URL.
   - **Log Pattern:** GET request to `/ftp/` path including `.bak` extensions, returned with status 200, indicating exposed sensitive content.
4. **CAPTCHA Logic Flaw**
   - **Exploit:** Reused the same CAPTCHA token across multiple requests.
   - **Log Pattern:** Multiple consecutive POST requests using identical token values, showing illogical behavior bypassing expected validation flow.
5. **XSS – Search Injection**

- **Exploit:** Injected `<script>alert(1)</script>` in the search bar.
- **Log Pattern:** GET request to `/rest/products/search?q=...` with HTML/JS content embedded in query string.

These examples demonstrate that even without real-time detection, the presence of structured, timestamped server logs makes it feasible to identify and trace abnormal activity post-exploitation. These traces form the ground truth used later for automated anomaly detection training and validation.

5 Log-Based Anomaly Detection with Machine Learning

This section documents the Modeling Team's workflow, which aimed to build AI models capable of detecting abnormal behavior from server logs. The goal was to evaluate and compare multiple deep learning approaches using a real-world dataset.

5.1 Initial Attempt with VoBERT

The original plan was to implement VoBERT, a vocabulary-free transformer model designed for log anomaly detection. VoBERT introduces element-level tokenization without relying on a predefined vocabulary, making it ideal for dynamic log formats.

The team attempted to clone the VoBERT repository and run the training code on provided datasets. However, several issues prevented successful deployment:

- **Tokenization Complexity:** The model required a custom tokenizer that did not support the HDFS format out-of-the-box.
- **Environment Conflicts:** The VoBERT codebase depended on legacy versions of PyTorch and external libraries, leading to runtime errors.
- **Excessive Memory Usage:** VoBERT required high GPU memory for self-attention layers, which exceeded the available capacity during training.

Due to these constraints, the team decided to pivot toward implementing LogBERT, a more established and better-documented alternative.

5.2 Final Implementation with LogBERT, LogAnomaly, and DeepLog

The team implemented and compared three state-of-the-art log anomaly detection models:

- **LogBERT**: A Transformer-based model trained via masked log key prediction and hypersphere loss. It learns contextual log sequences in a self-supervised manner.
- **LogAnomaly**: A Bi-LSTM model with semantic vectorization of log keys using Template2Vec and numerical distance measurement.
- **DeepLog**: A traditional LSTM sequence model that predicts the next log event and flags deviations as anomalies.

Dataset and Training

The HDFS dataset from the LogHub repository was used for evaluation. It contains over 11 million system log entries with labeled anomalies. The following steps were applied:

1. Preprocessing using the Drain parser to extract structured log templates.
2. Sequence generation with a sliding window of size 10–20.
3. Model-specific tokenization and input formatting.
4. Training each model on normal logs and evaluating on test logs containing both normal and anomalous sequences.

Results and Evaluation

The following figures illustrate the model comparison:

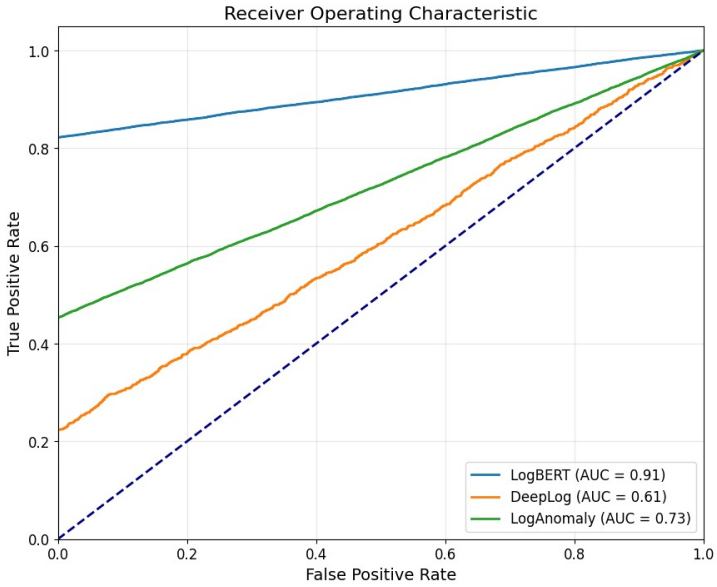

Figure 1: ROC Curve Comparison — LogBERT achieved the highest AUC score (0.91), followed by LogAnomaly (0.73), and DeepLog (0.61).

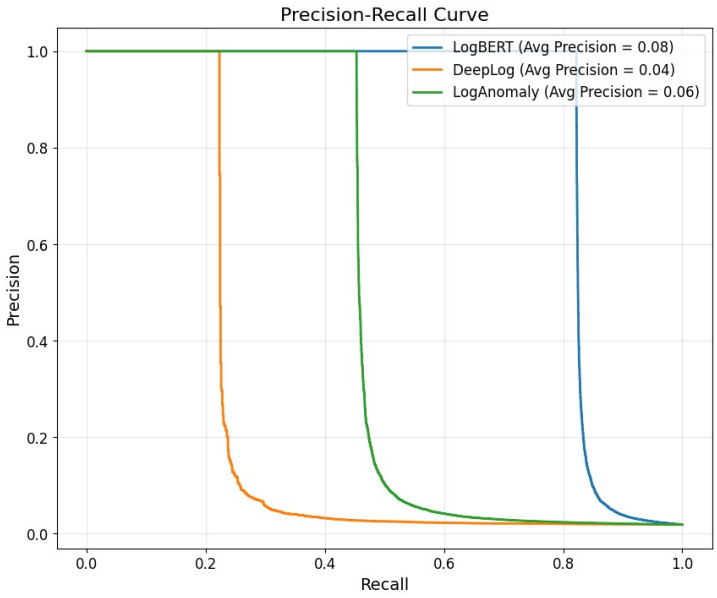

Figure 2: Precision-Recall Curve — LogBERT showed stronger recall and precision, indicating its effectiveness at detecting true anomalies.

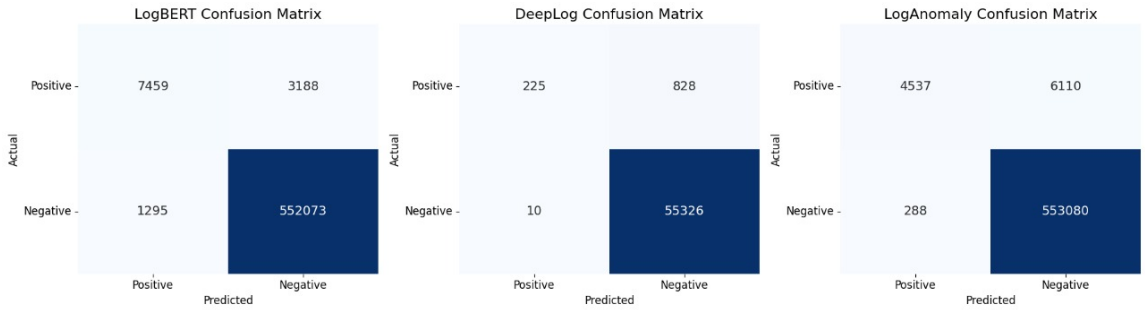

Figure 3: Confusion Matrices — LogBERT had the highest true positive count and lowest false negatives among all models.

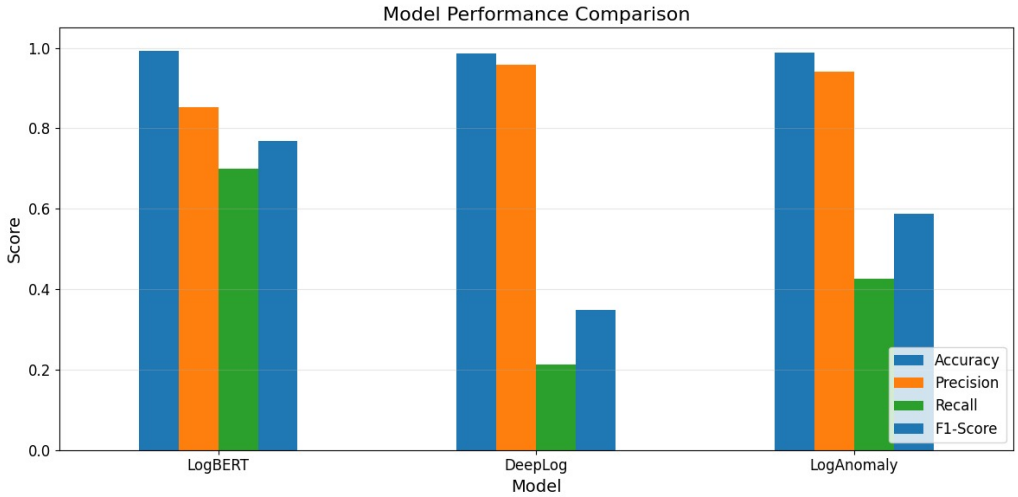

Figure 4: Overall Metric Comparison — LogBERT outperformed both DeepLog and LogAnomaly in F1-score, Recall, and Precision.

Key Observations

- **LogBERT** provided the best overall performance with high AUC and F1-score, making it the most reliable for real-world detection.
- **DeepLog** had the highest accuracy, but it suffered from extremely low recall, meaning it missed most actual anomalies.
- **LogAnomaly** showed balanced results but underperformed compared to LogBERT in both precision and recall.

For more technical implementation details, evaluation configurations, and architecture explanations of the models, please refer to the accompanying conference paper.

6 Future Work

Building upon the current evaluation of log anomaly detection models, future work will focus on extending their applicability to dynamic, adversarial environments. The key direction is to integrate machine learning-based anomaly detection with real-world penetration testing workflows, forming a complete, automated threat detection pipeline.

Integrating Penetration Testing and Detection Models

The next milestone involves aligning the machine learning pipeline with security testing efforts to validate anomaly detection in realistic web application scenarios. Specifically, we plan to:

- **Capture Web Exploit Logs:** Leverage `BurpSuite` to record HTTP traffic and log traces generated during the execution of deliberate exploits against OWASP Juice Shop, a widely used vulnerable application for cybersecurity training.
- **Apply Trained LogBERT:** Use the existing LogBERT model, trained on structured logs, to infer anomalies in the captured Juice Shop logs, aiming to detect exploit-induced deviations in behavior or sequence flow.
- **Correlate Model Alerts with Ground Truth:** Compare LogBERT's anomaly detections with known attack signatures and exploitation timestamps from the penetration testing session. This will assess the model's sensitivity to application-layer attacks.

This plan serves to connect supervised penetration testing with unsupervised anomaly detection, creating a hybrid evaluation framework that mimics real-world detection systems.

Extending Detection Scope and Adaptability

Beyond integration, further improvements are planned to enhance the effectiveness and adaptability of the detection pipeline:

- **Fine-tune LogBERT on Juice Shop Logs:** Adjust model weights on application-specific logs to improve its sensitivity to web request anomalies and reduce false positives.
- **Improve Detection Granularity:** Refine the output layer of LogBERT to not only flag anomalies but also classify them by exploit category (e.g., injection, broken authentication, XSS).
- **Deploy in Real-Time Scenarios:** Integrate LogBERT with live Juice Shop instances, capturing logs in real time via streaming APIs or log aggregation tools (e.g., Fluentd, Logstash).
- **Explore Hybrid Approaches:** Investigate combined techniques that integrate learned representations from LogBERT with deterministic, rule-based systems (e.g., signature matching) for enhanced robustness.

These efforts will help bridge the gap between academic anomaly detection models and production-grade intrusion detection systems, particularly for modern web applications with dynamic and diverse attack surfaces.