Combining URL and User-Agent Features for Malicious Web Request Identification

Michael Evans
Data Science
Anderson College
Regis University, Denver, CO, USA
canastotacoach@gmail.com

Abstract

Web-based cyberattacks, such as phishing and bot-driven intrusions, often exploit HTTP requests through malicious URLs and spoofed User-Agent strings. Traditional detection systems typically rely on single-feature analysis, which limits their ability to identify sophisticated threats. This practicum project proposes a large language model with machine learning approach that combines lexical features extracted from URLs with behavioral indicators derived from User-Agent strings to improve malicious request detection. Using publicly available datasets, including CICIDS 2017 and phishing URL repositories, the project implements feature engineering techniques for entropy, tokenization, and anomaly detection, followed by model training using Random Forest, XGBoost, and neural networks. Comparative evaluations between URL-only, User-Agent-only, and hybrid models demonstrate that feature fusion significantly enhances detection accuracy and robustness. The outcomes of this research provide actionable insights for web security analytics and lay the foundation for real-time deployment in intrusion detection systems.

I. Introduction/Background

Cybersecurity threats targeting web applications have grown exponentially, with attackers exploiting HTTP requests through phishing URLs and spoofed User-Agent strings. Traditional detection systems often rely on single-feature analysis, limiting their ability to identify sophisticated attacks. This practicum proposes a hybrid approach that combines lexical features from URLs with behavioral indicators from User-Agent strings to improve malicious request detection. Using machine learning techniques, this project aims to improve detection accuracy and provide actionable insights for web security.

II. Problem Statement

Cyberattacks targeting web applications often exploit HTTP requests through phishing URLs and spoofed User-Agent strings. Traditional detection systems rely on single-feature analysis, limiting their ability to identify sophisticated attacks. This practicum proposes a hybrid approach that combines lexical features from URLs with behavioral indicators from User-Agent strings to improve malicious request detection using machine learning.

III. Related Work

The goal of Information Security models is to reduce the number of FPs by increasing the Precision. In Information Security there is always a signification number of TNs, in the billions, due to the log file input. This high number of FPs will cause a lower Accuracy number so it really can't be used well in Information Security. My approach for finding fewer FPs is to send the log events (User-Agent String and URL to an LLM for analysis to determine a degree of anomalous score.

Others have used approaches Support Vector Method (SVM) for Novelty Detection [1] introduces the One-Class SVM algorithm, directly addressing scenarios is similar to my combining URL and User-Agent features. The difference with SVM is where only "normal" or positive examples (e.g., blocked user agents

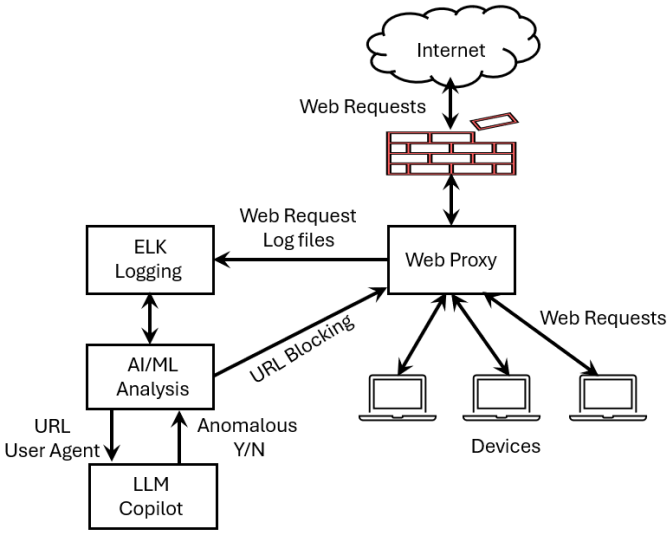

Fig. 1. Web Request Work Flow

treated as a single class with noise from FP) are available for training. SVM tackles the similar problem of novelty detection in imbalanced data, where anomalies (e.g., evolving malicious user agents) must be identified without labeled negatives (TN), but it is only one dimensional.

In One-Class SVM (SVDD) [2] the focus is on enclosing normal data modeling without TN by treating blocked user agents as a ”normal” distribution with outliers (e.g., extreme TP or FN). It addresses similar problems in unsupervised outlier detection for sparse datasets and uses comparable methods like kernel tricks for non-linear data, providing insights into robustness against noise—relevant for distinguishing FP (benign-but-blocked) from true threats in user agent logs. A key limitation with SVDD is its assumption that anomalies are uniformly distant from the normal cluster, while these may fail in my combining features case when malicious user agents mimic benign ones closely (e.g., slight variations in strings), leading to higher FP rates. I’m trying to limit FP rates.

[3]

IV. Methodology/Approach

My project employs an observational research design with elements of mixed methods, focusing on retrospective analysis of existing network security data rather than controlled experiments. The observational component involves examining historical blocked traffic logs to identify patterns in user agent strings, classifying them into degrees of true positives (TP: correctly blocked malicious activity), false positives (FP: incorrectly blocked benign activity), and true negatives. FN, malicious activity that was initially allowed but later identified, are extremely hard to find in Information Security. In Information Security, new data is not generated experimentally. Instead, historic real-world, imbalanced datasets data are used with unsupervised machine learning to derive insights by mirroring operational environments.

The primary data source is network traffic logs from a Web Application Firewalls (WAF), proxy servers, or intrusion detection systems (IDS). These requests contain user agent strings and URLs from inside and outside the organizaiton. These logs are observational in nature, capturing HTTP/HTTPS headers, and application requests from production environments. 1. Data format is structured logs contain fields such as timestamp, IP address, user agent string, request URL, and block reason. These logs are sent to Elastic Logstash and Kibana (ELK) for storage and LLM analysis for anomalous behavior.

The LLM chosen was Microsoft CoPilot. The data input feed was manual for this project, but would be automated using an API in the future. Also, in the future 3 LLMs would be use to score the log events for anomalous scores. Post-analysis labels (TP/FP/TN assigned via manual or semi-automated review) from the LLM and stored in the ELK database.

V. Data Description

The primary data source is network traffic logs from a Web Application Firewalls (WAF), proxy servers, or intrusion detection systems (IDS). These requests contain user agent strings and URLs from inside and outside the organization. These logs are observational in nature, capturing HTTP/HTTPS headers, and application requests from production environments. 1. Data format is structured logs contain fields such as timestamp, IP address, user agent string, request URL, and block reason. These logs are sent to Elastic Logstash and Kibana (ELK) for storage and LLM analysis for anomalous behavior.

TABLE I
ModSecurity Audit Summary — August 1, 2025

Metric	Count / Value
Total ModSecurity audit events (Aug 1 only)	6,063
Unique URLs (host + path)	4,056
Unique User-Agents	831
Requests missing a User-Agent header	223
Unique client IPs	690

The dataset obtained was August 1, 2025 from A Thirty-Day Dataset of Malicious HTTP Requests Blocked by OWASP ModSecurity on a Production Web Server [4]. The data input fields are timestamp, txid, unique_id, client_ip, method, host, uri, full_url, user_agent, status_code, rule_ids, severity, and message_excerpt. The breakdown of the data is Table I with a total number of events analyzed as 6,063. The breakdown of http/s responses Table II including Not Found, 404, is 3,947. Not found results are with missing or broken links, or human threat actors trying to find configuration and other blocked files–thus anomalous activity.

TABLE II
Response Outcomes — August 1, 2025
(Most requests blocked/redirected or 404 — desirable for security)

Status Code	Count
404 Not Found	3,947
301 Redirect	1,113
403 Forbidden	807
304 Not Modified	124
500 Internal Server Error	40
302 Found	16
503 Service Unavailable	7
400 Bad Request	4
405 Method Not Allowed	3
Others	various smaller counts

The most common User Agents Table III in the dataset. Since this is blocked traffic, these Top User Agents can show us common Human Threat Actors (HTAs), anomalous activity, since most HTAs don't update their attacking machines. We also see the BOTs attacks in the User_Agent string as DotBot, BlexBot, YandexBot, and python-httpx. These attacks will have more TPs than TNs or FPs. While these BOTs can be valid search engines, most of the time they are anomalous. We will find that when coupled with Status Code or URLs, the anomalous score will increase the TP, TN accordingly.

We also see in this table the out-of-date browsers (Mozilla/5.0 ... Chrome/91.0.4472.124). These tend to be HTAs since most valid users update their systems because their machines are set to auto update. The presence of python-httpx / Go-http-client strongly indicates automation.

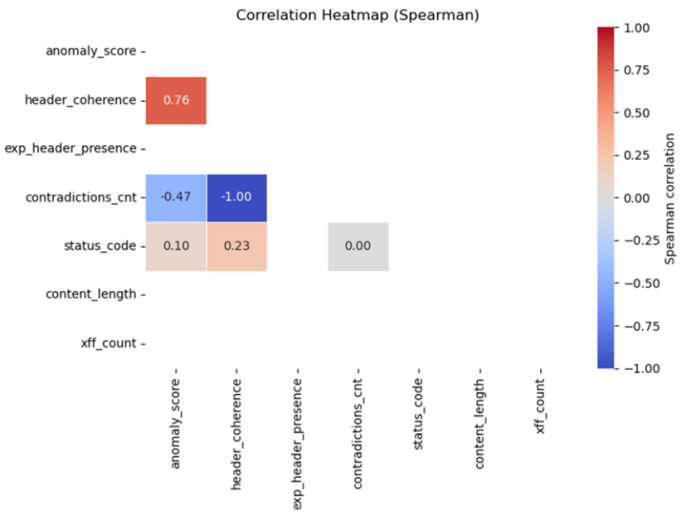

Fig. 2. Correlation Heatmap

TABLE III: Top User-Agents — August 1, 2025

User-Agent	Count
Mozilla/5.0 ... Chrome/91.0.4472.124 ...	2,569
Mozilla/5.0 (compatible; Cse/1.0; +https://cse)	428
DotBot/1.2 ... [regis365-m...epoint.com]	251
iPhone Safari (iOS 13.2.3)	233
BLEXBot/1.0	233
(missing)	223
YandexBot/3.0 [regis365-m...epoint.com]	192
trendictionbot0.5.0 [regis365-m...epoint.com]	154
python-httpx/0.28.1	128
Go-http-client/1.1	100

VI. Expected Outcomes

The expected outcomes for combining URL and User-Agent features for malicious request identification are a curated list of malicious URL patterns, a list of suspicious User-Agent strings flagged as anomalies for the Cyber Threat Operations Center team to review. After being fed the raw data asking them to find how anomalous the events are and provide anomalous_score and categorization of the event.

The Correlation Heatmap in Table 2 shows header_coherence compared with anomaly_score has a correlation score of 76 percent. Second, the status_code correlated with the header_coherence has a 23 percent correlation. The others don't show a correlation. This appears to be appropriate since header_coherence with the anomaly score would show bad URLs HTAs are going to and the status_code would show anomalous behavior when HTAs are looking to find configuration files that are accessible. The reason status_code with header_coherence is lower would be caused by webpages that are renamed and happen to be broken links, non-anomalous, versus anomalous.

In Table III, the huge Chrome/91 block is suspicious because it often appears in scripted traffic (spoofed UA) rather than in real browsers. The presence of pythonhttpx / Go-http-client strongly indicates automation. In Table IV, we see the anomalous activity broken down by category/trigger. These events are highly anomalous TP, where when we compare the breakdowns of Table 3 path_traversal, policy_header, rce, restricted_file, and php_header are all highly anomalous and the percentage of TP. When we look at numeric_host in Table 3, most of the percentage of events are non-anomalous.

TABLE IV
TOP MODSECURITY RULE CATEGORIES / TRIGGERS — AUGUST 1, 2025

Category / Trigger	Count
restricted_file (e.g., /.env, /.git/config, wp-config.php) [regis365-m...epoint.com]	2,927
policy_header (restricted header policy, e.g., Accept-Charset) [regis365-m...epoint.com]	1,127
bad_bot (blocked crawlers / bot UAs) [regis365-m...epoint.com]	844
other (misc WAF alerts not in the above buckets)	781
protocol_violation (conflicting Connection, illegal content-type patterns, etc.) [regis365-m...epoint.com]	237
php_injection (gzdecode filename/function triggers; CRS 933150) [regis365-m...epoint.com]	153
rce (command execution patterns; CRS 932150 etc.) [regis365-m...epoint.com]	74
numeric_host (Host header is numeric IP) [regis365-m...epoint.com]	59
path_traversal (directory traversal style requests)	20

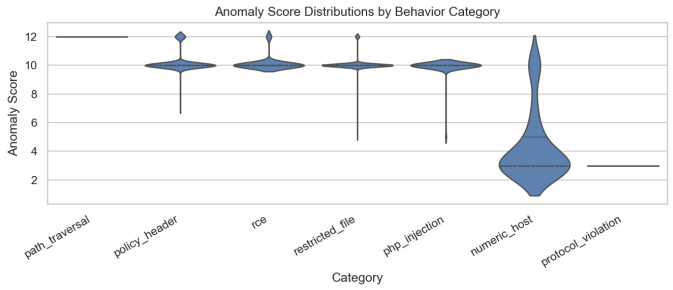

Fig. 3. Anomaly Score Distributions by Behavior Category

The Composite Behavior Score (CBS) by Category, Table 4, the average z-scores or risk_lift, and anomaly mean show the same anomalous activity of protocol_version, policy_header, rce, and php_injection as anomalous where restricted_file and numeric_host as non-anomalous.

VII. TIMELINE

The key phases/milestones for combining URL and User-Agent features for malicious request identification are listed below.

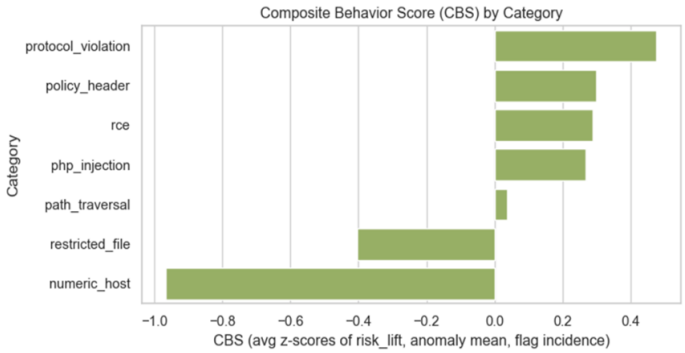

Fig. 4. Composite Behavior Score (CBS) by Category

Week	Task
1-2	Data Collection and Cleaning
3-4	Feature engineering for URL and User-Agent
5-6	Model Training and baseline comparisons
7-8	Evaluation, Visualization, and Final Report

Data collection and cleaning are scheduled for weeks 1-2. Once completed, weeks 3-4 are feature engineering for the URL and User Agent strings. Weeks 5-6 the model training and base lining will be completed, and weeks 7-8 are reserved for evaluation, visualization building and producing the final report.

VIII. Future

In the future, I will be automating the running this script via API to 2-3 LLMs and comparing which one provides the best answer, or should we us a composite score of them. Next, we store the User_Agent strings with their respective URLs along with the anomaly scores/category in an ELK index (dataset). This would allow us to reduce the costs of sending millions of pairs of input data when scores are already known. Thus, we would only be sending unknown input pairs to the LLM on a cadence, say every 20 minutes. This would allow the Information Security team to respond in an appropriate amount of time. All of these would be automated, so the Information Security Team wouldn't be burdened with manual tasks.

IX. Conclusion

Provide a synthesizing overview of your practicum proposal that reinforces its significance and coherence. Briefly recapitulate the central problem your project addresses and the key approaches you will employ. Emphasize the **unique contribution** your work aims to make to both data science as a field and the specific domain or organization involved. Reflect on the educational value of the project, highlighting how it represents the culmination of your data science studies and preparation for professional practice. This section should leave readers with a clear understanding of why your project matters and why you are well-positioned to execute it successfully, serving as a final, compelling argument for the merit and promise of your proposed practicum.

Note: The final practicum report must cover the following components:

1) Project Title
2) Abstract
3) Introduction/Background of the Project
4) Problem Statement
5) Literature Review/Related Work
6) Methodology/Proposed Approach
7) Data Analysis
8) Expected Outcomes
9) Timeline
10) Conclusion
11) Reference

References

[1] B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, and J. C. Platt, "Support vector method for novelty detection," in *Advances in Neural Information Processing Systems 12, [NIPS 1999, Denver, CO, USA, 1999]*, 1999, pp. 582–588. [Online]. Available: https://papers.nips.cc/paper/1723-support-vector-method-for-novelty-detection

[2] D. M. J. Tax and R. P. W. Duin, "Support vector data description," *Machine Learning*, vol. 54, no. 1, pp. 45–66, Jan. 2004. [Online]. Available: https://link.springer.com/article/10.1023/B:MACH.0000008084.60811.49

[3] A. Name and A. Name, “Article title,” *International Journal of Network Management*, vol. XX, no. XX, pp. XXX–XXX, YEAR. [Online]. Available: https://doi.org/10.1002/nem.1900

[4] G. Lucz, “A thirty-day dataset of malicious http requests blocked by owasp modsecurity on a production web server,” Sep. 2025. [Online]. Available: https://doi.org/10.5281/zenodo.17178461