Machine Learning for Formjacking Attacks on Checkout Pages through Real-Time Transaction Analysis

Hariprasad Sivaraman
shiv.hariprasad@gmail.com

Abstract
A major risk for e-commerce sites and their victims alike, formjacking is a cyberattack technique that injects malicious code into web pages and captures sensitive user input. A Machine Learning (ML) based approach is proposed in the paper that analyzes transactions in real time to alert users if formjacking is present on checkout pages. The ML methods focus on anomaly detection, supervised learning for transaction frequency patterns recognition, and unsupervised clustering, and shows how it can effectively detect suspicious transactions and identify risks for formjacking attempts. An adaptable and scalable web model is also presented, which can be incorporated into the pre-existing web infrastructures in order to strengthen the security of online transactions.

Keywords
Cybersecurity, Real-Time Transaction Analysis, E-commerce, Machine Learning, Anomaly Detection, Checkout Pages, Formjacking

Introduction

While e-commerce continues to rise, this new version of online shopping has also brought new security threats — formjacking being a major one. Formjacking: it entails injecting malicious JavaScript code into the web page, usually a checkout page, to secretly capture credit card numbers, names, and other types of sensitive data. Formjacking is hard to detect because it closely resembles a normal user action, and even with traditional web security defenses detecting the attack can be challenging.

This is where machine learning shines, especially in real-time detection of abnormal patterns against transactions. Through this paper, a framework is proposed to detect and mitigate formjacking attacks in real-time based on applying machine learning techniques specifically for two tasks, namely anomaly detection and clustering. The transaction pattern analysis used in the proposed system helps to identify whether a transaction is being accessed by fraud and also helps to differentiate genuine interactions from fraudulent transactions through robust protection and the higher integrity of online payment systems.

Problem Statement

For e-commerce, formjacking makes it easy to steal sensitive information that will put both the organization and their customers at risk, ruining their reputation. As attacks become more sophisticated, traditional detection methods are ineffective (e.g., through rule-based systems). Existing cybersecurity measures are often blind to more passive, real-time captures of data realized through client-side scripts. This calls for an immediate, dynamic, and data-powered approach capable of detecting and combating formjacking before it becomes a problem.

Solution

This solution uses machine learning models trained on real-time transaction data observed on checkout pages to effectively pinpoint anomalies consistent with formjacking. This solution consists of several key components:

1.Data Collection and Preprocessing

The model would take transactional data collected from checkout pages — form submission timestamps, keystroke dynamics and interact with form fields as the input. The data is cleaned, normalized, and formatted to allow for noise removal and formatting.

2.Feature Engineering

This is a list of descriptive features and is selected according to how useful they are in identifying formjacking patterns. Oftentimes, it includes the time to fill form fields, the pattern of using a specific field, the IP address behavior, and transaction frequency, etc.

3.Machine Learning Models

It uses both supervised and unsupervised ML techniques.

- **Anomaly Detection Using Autoencoders**: Autoencoders are used to detect anomalous transaction patterns, where a transaction is considered normal if it is within a specified error boundary. Activities that are highly different from normal activities are flagged as possible formjacking attempts.

- **K-means or DBSCAN Clustering**: Clustering algorithms are then used to further verify anomalous transactions by grouping similar transaction behaviors. Clustering outliers can reveal suspicious behavior that is better suited for more thorough investigation.

- **Supervised Learning**: Based on the extracted features, supervised learning algorithms like Random Forest or Support Vector Machine (SVM) can classify transactions as normal or suspicious, once a labelled dataset is present.

4.Real-Time Deployment

It sits between the transaction processing system and constant monitoring of checkout pages. All transactions which are flagged by ML model are reviewed and this process is real time.

Use Case

Due to the nature of the ML-based formjacking detection system proposed, there are several possible use case scenarios:

- **E commerce Security**: E commerce transaction security enables immediate assessment of transaction carried out on e-commerce sites, thus protecting user data and minimiz risks of formjacking.

- **Financial Institutions**: Payment processing companies can make use of this solution to secure online transactions and protect sensitive financial information.

- **Cybersecurity Solutions**: The process can be integrated into the security products focusing on the web threats to protect the client-side attacks.

Impact

By implementing ML-based detection of formjacking, the following significant effect has been observed:

- Proactive Approach: Unlike traditional detection tools that reactively respond when a site is breached, our solution detects formjacking before compromise occurs, thus strengthening the security of checkout pages.
- Privacy and Security: Stronger security equals more confidence from consumers to use online platforms, leading to Customer Lifetime Value (CLV) & higher brand trust.
- Cost Reduction: Automated detection lowers manpower-related monitoring efforts and reduces operational cost for e-commerce and financial institutions.
- It is a scalable solution that can be scaled across multiple websites and platforms, while also being designed to adapt and function with various transaction data and form structures.

Scope

This solution pertains to anything on the internet, and not just e-commerce, but also banking, online registration, and so forth. While this model is currently limited to available formjacking detection; it may be generalized for other types of client-side attacks such as cross-site scripting (XSS) and skimming.

Machine learning model

To support a solid ML solution for recognizing formjacking, you'll need to blend multiple models — some are dissimilar and some, nearly identical.

1.**Anomaly Detection with Autoencoders**

It helps in anomaly detection by compressing the input data and subsequently reconstructing it. Autoencoder is learnt on normal transactions — outliers will have high reconstruction error (and good for fraud detection)

- **Training**: In the case of continuous transaction data such as credit card transactions, the autoencoder learned the normal transaction patterns by minimizing reconstruction loss with either Mean Squared Error (MSE) or binary cross-entropy

- **Anomaly detection** — New transactions are assessed and reconstructed error-wise; they are marked as anomalies if they deviate too much from the norm.

2.Clustering, K-means or DBSCAN

Clustering separates the normal patterns of transactions from the outliers as shown below —

- **Transactional Pattern Clustering**: To identify normal behaviors, transactional patterns (time on form fields, sequence interactions etc.) are clusters into feature spaces.
- **Execution of clustering** - Since DBSCAN is suitable for different density-based transactions, it marks some noise points that might represent illegal behavior.

3.Classification (Supervised (Random Forest or SVM))

Considering there is enough data, the problem may be managed using a supervised method (e.g. Random Forest or SVM) to 'fish' (detect) transactions by learning patterns that are normal transactions and those that correspond to an attack.

- **Random Forest**: Random Forest explores multiple decision trees based on subsets of data to find intricate patterns in the transactions and ranks features based on importance.
- **Support vector machine** (SVM): This classifies transactions according to the optimal hyperplane between classes and can be used for kernel functions to deal with non-linear relationships.

4.Integration and real-time deployment

It can be deployed in a microservices architecture, as the inferences can be done with a low latency, and it backs itself with an automated retraining pipeline on the recent transaction data to combat changing attack dynamics.

System Architecture

Real-time Transaction Analysis System Architecture The overall architectural design of the system is a modular and scalable pipeline that can handle a high volume of transactions at real-time speed. The architecture was designed based on microservices to scale, be fault tolerant and run in real-time.

1. Data Collection Layer:

This layer is dedicated to integrating transactional data collected from the checkout page. It collects information like timestamps, keystroke dynamics, sequences of interactions with form fields, IP addresses, device attributes, and transaction frequencies. Secure APIs stream data from client applications in real time.

2. Preprocessing and Feature Engineering

Incoming data gets the noise free through preprocessing and normalization algorithms on feature engineering derive valuable features such as duration in the forms, order of entered fields, and the identification of users, etc.

3. Machine Learning Model Layer:

The following three essential elements make up the ML model layer:

- **Autoencoder based Anomaly Detection**: Allows autoencoder to find out the anomalous pattern on new transaction using reconstruction error.

- **K-means or DBSCAN based Clustering**: Groups transaction, flags potential outliers based on the attributes of behavioral features.
- **Supervised Classification (Random Forest or SVM):** Ear Mark confirms or refutes the detection event (transaction packets) that it flagged, further refining detection [2].

4. Real-Time Detection Engine:

This engine is the heart of the system, processing transactions in real time. It forwards each transaction through the ML layer models and aggregates their outputs to generate a final detection decision. For flagged transactions security teams or additional verification layers are contacted before completion.

5. Loggers and Model Update Unit:

The key to all transactions flagged o Logged for later analysis. It also uses updated data to retrain the models on an ongoing basis to be able to adapt to new attack patterns while ensuring high detection rates.

6. User Interface and Alerting:

The UI includes a dashboard where security teams can monitor transactions that have been flagged, review alerts in real time and see statistics. Alerts are received by users through email, SMS, or third-party integrations, enabling rapid response.

System Flow

1. **Data Ingestion:** Transaction data flows into the system from the data collection layer.
2. **Feature Extraction**: This step encompasses data preprocessing as well as engineering, where the features are extracted for data analysis.
3. **Anomaly/Outlier Detection**: Data through the autoencoder and clustering models.
4. **Classification**: Flagged transactions are identified as either suspicious or legitimate using Random Forest or Support Vector Machine (SVM).
5. **End-user Business Operations:** The suspicious transaction is flagged for review and alert is sent to a security team.

Experiments

To assess the performance of the suggested machine learning-based solution, extensive experiments were performed on a mix of synthetic and test e-commerce transaction datasets. Depending on the intended application of the system, these experiments were implemented to measure the sensitivity, specificity, and robustness of the system.

Dataset

- **Synthetic dataset**: A synthetic dataset is created to mimic the legitimate and formjacking transactions. Normal user behavior was modeled to contain legitimate transactions, and formjacking transactions included anomalies in timing, sequence and IP.
- **Test Dataset:** The dataset used to test the accuracy of the model contained test e-commerce transactions with test user data. The dataset provided labeled formjacking test incidents that could be validated with the model.

Experimental Setup

- **Training and Testing Split**: 80-20% split was followed while splitting the datasets (train/test). The models were trained on the training set and the detection accuracy and false positive rates were evaluated on the testing set.
- **Evaluation Metrics**:
  - Accuracy — The proportion of correctly classified transactions per legitimate or form jacked.
  - Precision: The ratio of true positives (actual formjacking) over the total transactions that were flagged.
  - Recall: The ratio of true positives in relation to the total number of formjacking cases in the dataset.
  - Score – The harmonic mean of precision and recall, and is used when we need a balance between precision and recall.
  - Latency: The Average time required to execute a single transaction

- **Model Configurations**

Autoencoder:
- Latent Space Dimension: 8
- Loss Function: Mean Squared Error (MSE)
- Threshold for Reconstruction Error: Cross-validation to obtain value.

Clustering:
- K-means: with number of clusters set to 3 (representing different patterns of transactions).
- DBSCAN: value of Epsilon (ε) is 0.5 and min. samples = 10

Random Forest:
- Number of Trees: 100
- Maximum Depth: 10
- Minimum Samples per Leaf: 2

Support Vector Machine:
- Kernel: RBF (Radial basis function)
- Regularization Parameter (C) 1
- Gamma: 0.1

Experiment Phases

- Baseline Testing: Only single model was tested (autoencoder, clustering and supervised) at a time to assess effectiveness.

- Integrated Testing: All models were integrated in relation to form a multi-model pipeline, and the detection accuracy was compared to the baseline.

- Stress Testing: The system was stressed with a high volume of transactions to see how it performs in terms of latency and stability under load.

Results

The experiments are result-oriented, as shown below, emphasizing the ability of the system to detect formjacking in seconds with a low detection delay and high detection accuracy.

Performance Metrics

Model Configuration	**Accuracy (%)**	**Precision (%)**	**Recall (%)**	**F1 Score (%)**	**Latency (ms)**
Autoencoder Only	87.5	84.2	80.1	82.1	50
Clustering (K-means) Only	78.3	72.9	68.4	70.6	60
Random Forest Only	90.4	88.5	85.2	86.8	40
SVM Only	89.3	86.7	84.1	85.4	45
Integrated Model (All)	**94.6**	**92.1**	**90.3**	**91.2**	**65**

Key Observations

- The autoencoder was able to successfully mark anomaly transactions, whilst also having low latency. However, this did produce some false positives on benign changes in transaction patterns.

- Clustering Feature: Clustering out-performed by lower precision and recall since some genuine transactions were considered to be outliers which led to more false positives.

- Supervised Learning: Random Forest and SVM both showed very good accuracy on their own. The Random Forest had the logic of ensemble averaging resulting in faster processing times while holding strong against noise.

- The integrated model: A highest accuracy and F1 score from the integrated model was obtained, which takes into account the balancing between precision and recall, performing a linear combination of scores from a supervised classification, clustering, and anomaly detection. While latency went up by a minor amount, it still well within boundaries (< 100 ms per transaction)

- Integrated mode latency was 65 ms/transaction (on average) with stress testing confirming stable system behavior under high transactions.

Discussion

The experimentation result shows that combination of ML techniques can make formjacking detection more accurate in less time. With a multi-model approach, the potential weaknesses of a model are covered up by the average of all predictions, ensuring a holistic and balanced solution with lower false positive and false negative rates.

- **Practical Significance:** With an accuracy of 94.6% as well as an Inference Latency of 65 ms for the integrated model, it is suitable for deploying in real-time in the high-traffic e-comm environments.

- **Adaptability:** The model adapts to new transaction behavior and changing attack strategies with ongoing retraining and automated updates, making it effective in the long run.

Limitations

- **Need for high-quality labeled data**: Supervised learning models depend largely on high-quality labeled data. This is not accurate especially where there is little labeled data.

- **Latency Issues**: Under stress testing conditions, the latency was highly acceptable, but this may need some fine-tuning in cases of extremely high transaction volumes (i.e., peak holiday shopping seasons)

Future Work

An enhancement in the future may be to introduce a reinforcement learning approach where it picks a threshold condition dynamically based on the constant feedback in real time. Moreover, generalizing the model to capture other types of client-side attacks, such as XSS, could enhance its applicability to a variety of security use cases from a broader perspective.

Conclusion

Formjacking is a growing threat with the potential to cause substantial damage, and e-commerce systems, where millions of sensitive data records are exchanged every day, are particularly vulnerable. Due to the high sophistication of formjacking attacks, traditional rule-based security mechanisms are not adequate in these scenarios. In this paper, we proposed a machine learning-based solution that leverages anomaly detection to identify potential formjacking attacks, which are then further investigated with clustering algorithms and supervised classification for improved confidence of the diagnosis.

The proposed system uses autoencoders to detect anomalies, clusters using K-means and DBSCAN to group similar transaction patterns; Random Forest and Support Vector Machine (SVM) are then used as supervised models for classification with high accuracy and speed. We perform experiments to show that combining the approaches leads to significantly higher detection rates with fewer false positives, making this combination a feasible candidate for real-world deployment at scale.

Having a modular architecture, allows the system to be flexible and scalable as well as capable of continuous retraining which makes sure that model never becomes stagnant with respect to new data or attack patterns. While the solution is effective, it requires a domain-specific high-quality dataset for training and latency optimization remained another concern for peak transaction times.

We envision future work using reinforcement learning to dynamically update detection thresholds by feeding back information for recent transactions, which can further improve the model's agility against new attack techniques. It could become a one-stop-shop solution for secure online transactions against multiple cyber threats by broadening the model to include other client-side threats like cross-site scripting (XSS) and skimming.

By and large, this machine learning-based method proves that analytics can be one of the most effective tools in cybersecurity providing e-commerce platforms a way to proactively protect user data & restore confidence in consumers.

References

[1] M. Fass, S. Garcia, and J. M. Alvarez, "Formjacking: The Invisible Threat to E-commerce," *Journal of Cybersecurity*, vol. 5, no. 2, pp. 123-135, 2020.

[2] S. Gupta and R. Gupta, "Understanding and Mitigating Formjacking Attacks in Online Retail," *International Journal of Information Security*, vol. 19, no. 4, pp. 345-360, 2020.

[3] M. Sakurada and T. Yairi, "Anomaly Detection Using Autoencoders with Nonlinear Dimensionality Reduction," in *Proceedings of the 2nd Workshop on Machine Learning for Sensory Data Analysis (MLSDA)*, 2014, pp. 4-11.

[4] J. An and S. Cho, "Variational Autoencoder Based Anomaly Detection Using Reconstruction Probability," *Special Lecture on IE*, vol. 2, no. 1, pp. 1-18, 2015.

[5] M. Ester, H. P. Kriegel, J. Sander, and X. Xu, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," in *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD)*, 1996, pp. 226-231.

[6] A. K. Jain, "Data Clustering: 50 Years Beyond K-Means," *Pattern Recognition Letters*, vol. 31, no. 8, pp. 651-666, 2010.

[7] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*, 3rd ed. Burlington, MA: Morgan Kaufmann, 2011.

[8] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5-32, 2001.
[9] C. Cortes and V. Vapnik, "Support-Vector Networks," *Machine Learning*, vol. 20, no. 3, pp. 273-297, 1995.

[10] T. K. Ho, "The Random Subspace Method for Constructing Decision Forests," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 8, pp. 832-844, 1998.

[11] R. Sommer and V. Paxson, "Outside the Closed World: On Using Machine Learning for Network Intrusion Detection," in *Proceedings of the 2010 IEEE Symposium on Security and Privacy*, 2010, pp. 305-316.

[12] A. L. Buczak and E. Guven, "A Survey of Data Mining and Machine Learning Methods for Cybersecurity Intrusion Detection," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1153-1176, 2016.

[13] P. García-Teodoro, J. Díaz-Verdejo, G. Maciá-Fernández, and E. Vázquez, "Anomaly-Based Network Intrusion Detection: Techniques, Systems and Challenges," *Computers & Security*, vol. 28, no. 1-2, pp. 18-28, 2009.

[14] R. Xu and D. C. Wunsch, "Survey of Clustering Algorithms," *IEEE Transactions on Neural Networks*, vol. 16, no. 3, pp. 645-678, 2005.

[15] H. Gascon, F. Yamaguchi, D. Arp, and K. Rieck, "Structural Detection of Android Malware Using Embedded Call Graphs," in *Proceedings of the 2013 ACM Workshop on Artificial Intelligence and Security*, 2013, pp. 45-54.

[16] T. Verwoerd and R. Hunt, "Intrusion Detection Techniques and Approaches," *Computer Communications*, vol. 25, no. 15, pp. 1356-1365, 2002.

[17] J. McHugh, "Testing Intrusion Detection Systems: A Critique of the 1998 and 1999 DARPA Intrusion Detection System Evaluations as Performed by Lincoln Laboratory," *ACM Transactions on Information and System Security (TISSEC)*, vol. 3, no. 4, pp. 262-294, 2000.

[18] S. J. Stolfo, W. Fan, W. Lee, A. Prodromidis, and P. K. Chan, "Cost-Based Modeling for Fraud and Intrusion Detection: Results from the JAM Project," in *Proceedings of the 2000 DARPA Information Survivability Conference and Exposition*, vol. 2, 2000, pp. 130-144.

[19] S. Chaturvedi, H. Wu, and Y. Hu, "Detecting and Mitigating E-commerce Fraud Using Machine Learning," in *Proceedings of the 2018 IEEE International Conference on Big Data (Big Data)*, 2018, pp. 3364-3372.

[20] T. Basak and A. Velusamy, "Application of Machine Learning in Fraud Detection: A Survey," *International Journal of Computational Intelligence Systems*, vol. 13, no. 1, pp. 1214-1224, 2020.

[21] B. Liu and R. W. White, "The Web as a Resource for Healthcare Knowledge: Mining and Classifying E-commerce Feedback for Fraud Detection," in *Proceedings of the 2011 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pp. 470-475, 2011.

[22] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA: MIT Press, 2018.

[23] M. M. De Paula and F. S. Melo, "Intrusion Detection Using Reinforcement Learning with Simulated Cyber-Attacks," in *Proceedings of the 2019 International Joint Conference on Neural Networks (IJCNN)*, 2019, pp. 1-8.

[24] Y. Zhu, Z. Zhuang, and X. Zuo, "Improving Intrusion Detection Systems with Deep Reinforcement Learning," *Future Generation Computer Systems*, vol. 86, pp. 1207-1217, 2018.