CYBER SECURITY AND AI CASE STUDIES

Project Report

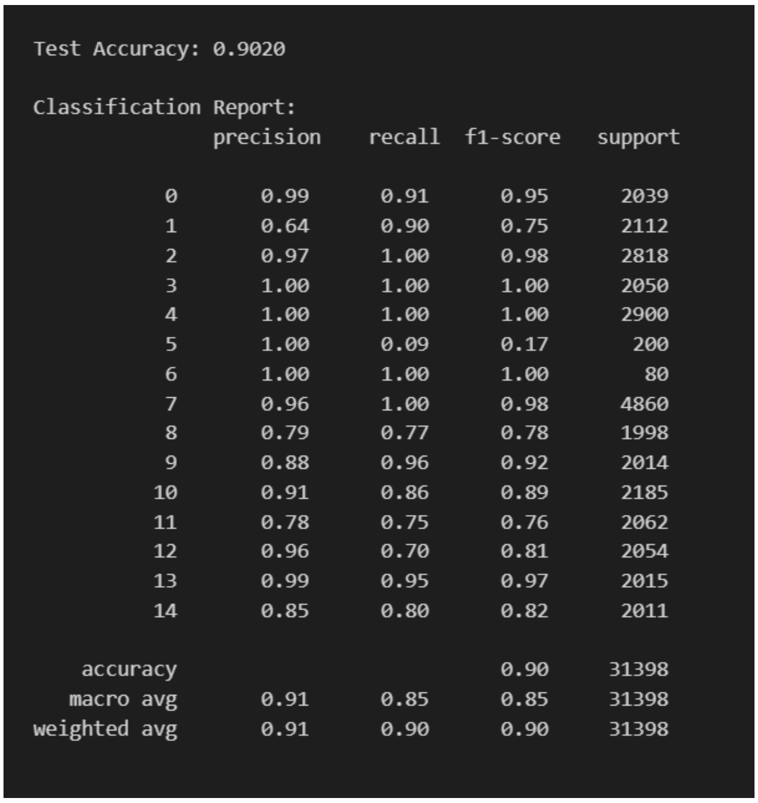
Part A

Model Selection and Justification

For this project, I chose the XGBoost classifier [1]. I selected this model because it works well for multi-class classification problems. It is also very efficient and can handle large datasets. My dataset had many features and multiple attack classes. XGBoost helped me manage this complexity. It also supports feature selection, which helped me reduce unnecessary data.

Before training the model, I selected important features using SelectFromModel. This method used XGBoost itself to choose only those features that had high importance. This step made the training faster and more accurate. I also used Stratified Train-Test Split to make sure each class was fairly represented. This helped to avoid class imbalance issues during evaluation.

Model Performance and Accuracy



```
Test Accuracy: 0.9020

Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.91      0.95      2039
           1       0.64      0.90      0.75      2112
           2       0.97      1.00      0.98      2818
           3       1.00      1.00      1.00      2050
           4       1.00      1.00      1.00      2900
           5       1.00      0.09      0.17       200
           6       1.00      1.00      1.00        80
           7       0.96      1.00      0.98      4860
           8       0.79      0.77      0.78      1998
           9       0.88      0.96      0.92      2014
          10       0.91      0.86      0.89      2185
          11       0.78      0.75      0.76      2062
          12       0.96      0.70      0.81      2054
          13       0.99      0.95      0.97      2015
          14       0.85      0.80      0.82      2011

    accuracy                           0.90     31398
   macro avg       0.91      0.85      0.85     31398
weighted avg       0.91      0.90      0.90     31398
```

Figure 1. Classification Report

After training the model, I tested it using the test set. The test accuracy was 0.9020, or 90.2%. This means the model predicted the attack types correctly in most cases. I also created a baseline accuracy for comparison. The baseline was based on the most frequent class. It gave an accuracy of around 55%. This shows that my model is much better than just guessing the most common class.

I used a confusion matrix to understand how well the model performed on each class. The matrix showed that most classes were predicted very accurately. Some classes like class 3, 4, and 7 had precision and recall close to 1.0, which is excellent. However, class 5 had very low recall. It was able to detect only 9% of the actual attacks in that class. This shows that the model needs improvement in detecting that specific type of attack.

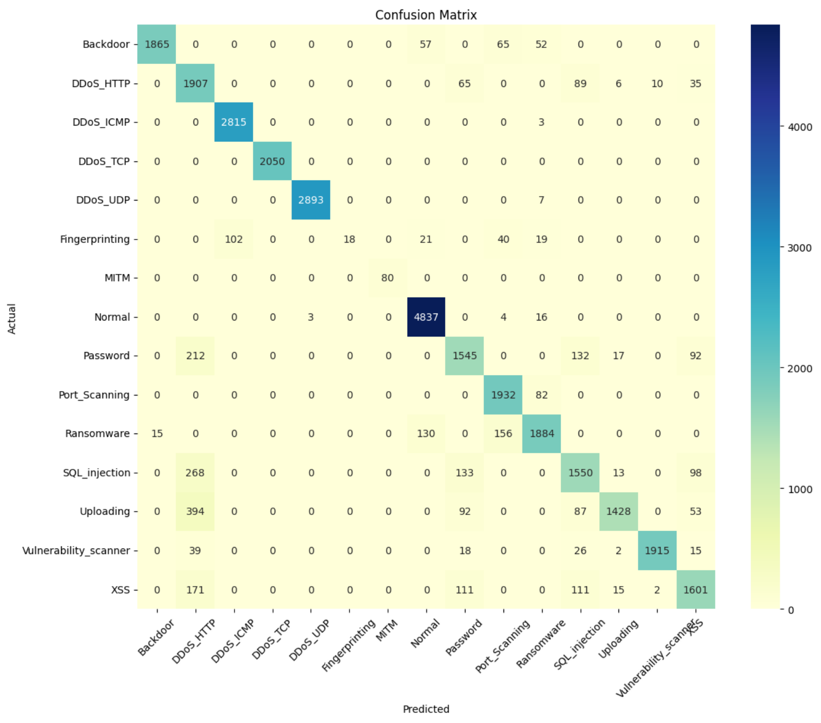


Figure 2. Confusion Matrix

Precision, Recall, and F1-Score

The classification report showed detailed metrics for each class. Most classes had precision and recall values above 0.85. The macro average recall was 0.85, and the weighted average F1-score was 0.90. These scores tell me that the model is not only accurate overall but also performs well across most classes. However, the performance dropped for a few classes, especially those with fewer examples.

Feature Importance

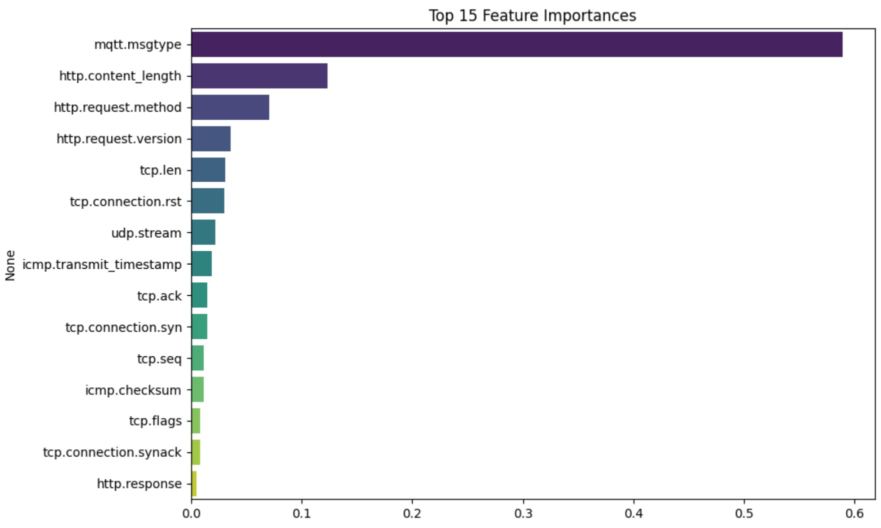


Figure 3. Important Features

I also looked at the top 15 features that contributed most to the model. These features were visualized using a bar chart. This helped me understand which data points were most useful for the model. These important features helped the model detect different types of attacks more effectively.

My model was able to detect intrusion and malicious activity in the IoT network with high accuracy. It also gave good performance across most attack types. Still, there is room for improvement, especially for classes with fewer samples. I may need to use techniques like oversampling or ensemble methods to improve this in the future. But overall, the results show that XGBoost was a good choice for this problem.

Part B

Impact of Label Poisoning on Intrusion Detection Systems

Network intrusion detection systems [2] are critical for cybersecurity. They help identify malicious activities in network traffic. These systems increasingly use machine learning algorithms. But these algorithms can be vulnerable to adversarial attacks like label poisoning. I conducted a comprehensive study on how label poisoning affects machine learning models used for intrusion detection. My research shows that intelligent feature-based poisoning is the most effective attack strategy, causing up to 14% reduction in model accuracy and 15% reduction in attack detection capabilities with just 20% poisoned labels.

Methodology

The baseline model achieved 90.2% accuracy and 90% F1 score. It showed excellent attack detection capabilities.

I then implemented four poisoning strategies:

1. Random Label Flipping
2. Converting Attacks to Benign
3. Targeted Class Confusion
4. Intelligent Feature-Based Poisoning

Each strategy was tested at different poisoning percentages (5%, 10%, 15%, 20%, 25%, and 30%).

Results

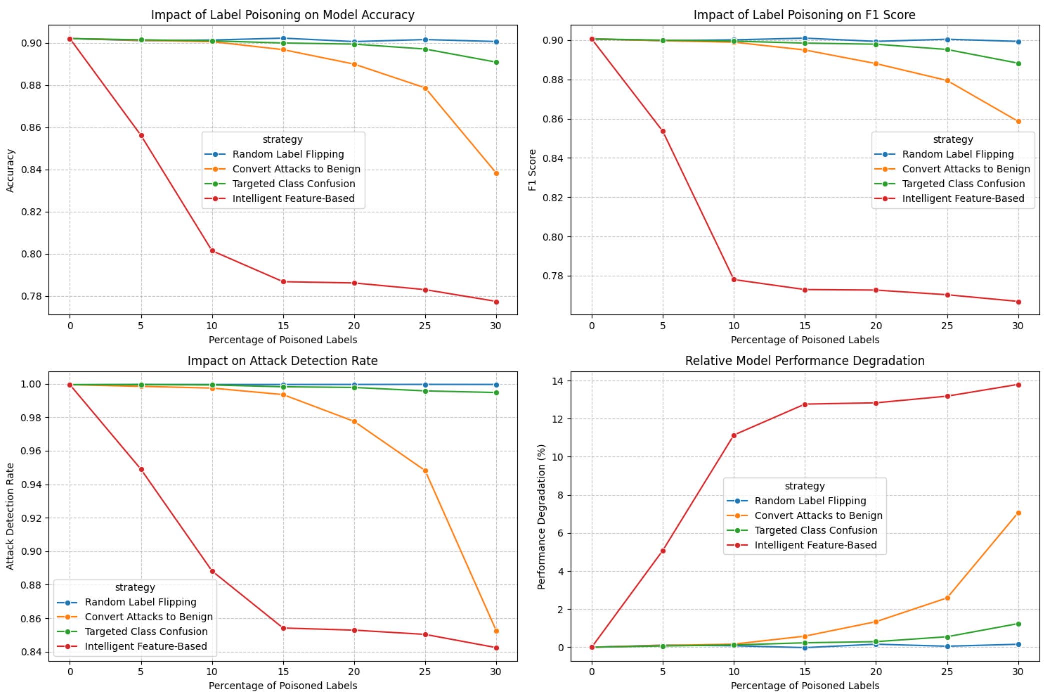


Figure 4. Impact of labels poisoning

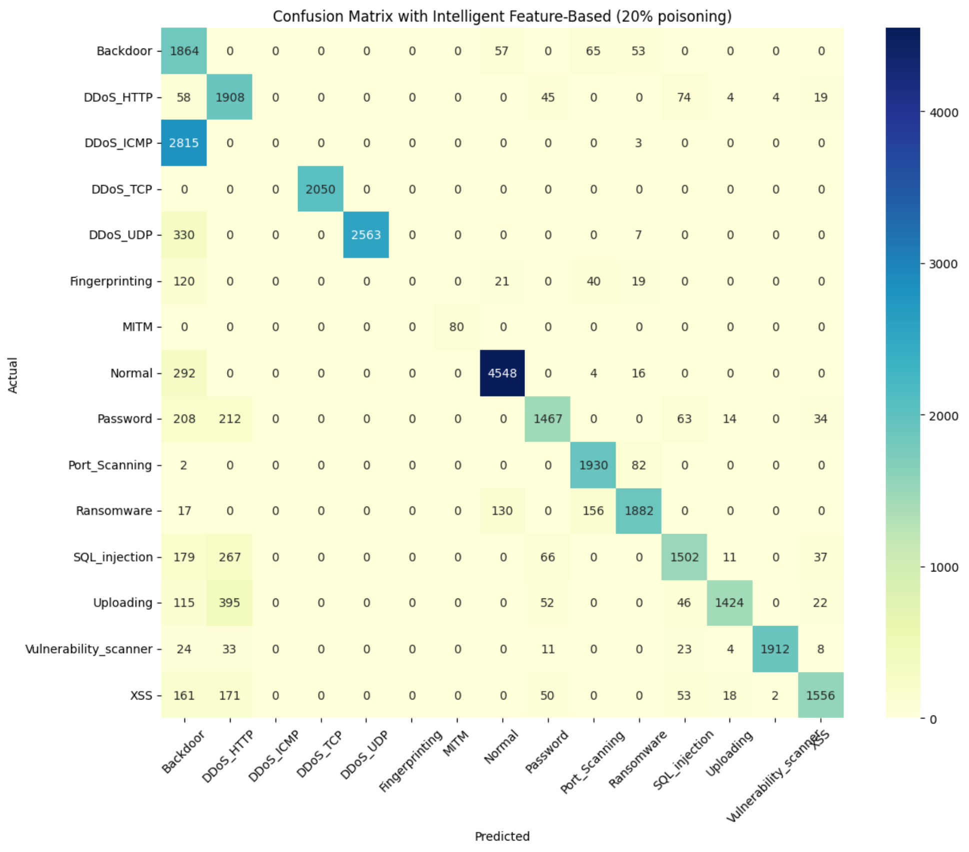
Strategy Performance

1. **Intelligent Feature-Based Poisoning** was the most devastating attack:
    - Caused steep performance drops even at just 5% poisoning
    - At 20% poisoning: 12.8% accuracy drop and 14.7% attack detection rate reduction
    - Uses feature importance to target the most critical samples
2. **Converting Attacks to Benign**:
    - Minimal impact until 20% poisoning
    - Became very effective at 30%, reducing attack detection by 14.8%
    - Works by making malicious traffic appear normal
3. **Targeted Class Confusion**:
    - Showed minimal impact on overall model performance
    - Maximum degradation of only 1.2% at 30% poisoning
    - Our implementation focused on specific class pairs
4. **Random Label Flipping**:
    - Least effective strategy
    - Even at 30% poisoning, only 0.2% performance degradation
    - Shows that random noise isn't effective against robust models

Analysis of Best Attack

The intelligent feature-based poisoning at 20% was the most effective attack. It:

- Reduced accuracy from 90.2% to 78.6% (12.8% drop)
- Reduced F1 score from 90.1% to 77.3% (14.2% drop)
- Lowered attack detection rate from 99.9% to 85.3% (14.7% drop)



Actual \ Predicted	Backdoor	DDoS_HTTP	DDoS_ICMP	DDoS_TCP	DDoS_UDP	Fingerprinting	MITM	Normal	Password	Port_Scanning	Ransomware	SQL_injection	Uploading	Vulnerability_scanner	XSS
Backdoor	1864	0	0	0	0	0	0	57	0	65	53	0	0	0	0
DDoS_HTTP	58	1908	0	0	0	0	0	0	45	0	0	74	4	4	19
DDoS_ICMP	2815	0	0	0	0	0	0	0	0	0	3	0	0	0	0
DDoS_TCP	0	0	0	2050	0	0	0	0	0	0	0	0	0	0	0
DDoS_UDP	330	0	0	0	2563	0	0	0	0	0	7	0	0	0	0
Fingerprinting	120	0	0	0	0	0	0	21	0	40	19	0	0	0	0
MITM	0	0	0	0	0	0	80	0	0	0	0	0	0	0	0
Normal	292	0	0	0	0	0	0	4548	0	4	16	0	0	0	0
Password	208	212	0	0	0	0	0	0	1467	0	0	63	14	0	34
Port_Scanning	2	0	0	0	0	0	0	0	0	1930	82	0	0	0	0
Ransomware	17	0	0	0	0	0	0	130	0	156	1882	0	0	0	0
SQL_injection	179	267	0	0	0	0	0	0	66	0	0	1502	11	0	37
Uploading	115	395	0	0	0	0	0	0	52	0	0	46	1424	0	22
Vulnerability_scanner	24	33	0	0	0	0	0	0	11	0	0	23	4	1912	8
XSS	161	171	0	0	0	0	0	0	50	0	0	53	18	2	1556

Figure 5. Confusion Matrix

Implications

My research shows [3] several important findings:

1. Sophisticated poisoning strategies that target important features are much more dangerous than random ones.
2. Even moderate poisoning levels (15-20%) can significantly compromise intrusion detection systems.
3. Different attack types have varying vulnerability to label poisoning.
4. The most dangerous attacks aim to make malicious traffic appear normal.

Label poisoning is a serious threat to machine learning-based intrusion detection systems. It can deceive the model by subtly manipulating the data. Intelligent feature-based poisoning is especially dangerous. It reduces the model's ability to detect attacks. At the same time, it maintains the appearance of normal system behavior. To counter this, security teams must implement strong safeguards. These measures are essential to maintain effective threat detection.

Part C

Preventing Label Manipulation in Intrusion Detection Systems: My Approach

Understanding the Problem

While working on my intrusion detection system, I saw how well machine learning could detect attacks. But I also realized it has a weakness, label poisoning.

Label poisoning happens [4] when someone secretly changes the correct labels in the training data. For example, a real attack might be labeled as "benign." If the model learns from this kind of poisoned data, it starts trusting fake patterns. As a result, it might fail to detect real threats.

This is a big risk in cybersecurity, especially in intrusion detection systems. If the model can't tell the difference between a normal connection and an actual attack, it becomes useless.

Theoretical Insights from Literature

Many researchers [5] have studied this issue. They agree that machine learning models are vulnerable when the training pipeline is not secure. Based on my reading and understanding, here are the common suggestions:

- Use robust training methods that reduce the model's sensitivity to wrong labels.
- Apply data validation to filter out noisy or suspicious records.
- Use ensemble models to spot inconsistencies.
- Rely on human verification—but that doesn't scale well for big datasets.

Most of these ideas are helpful but not enough. They detect problems after the damage is done. I wanted something that prevents label poisoning before training even starts.

My Novel Cybersecurity-Based Idea

To protect the labels in the dataset, I propose using digital signatures and hashing, which are concepts from cryptography.

Here's how it works:

1. Trusted source (like a network admin or data collector) labels the data.
2. Before storing it in the database, a cryptographic hash is created for each data-label pair.
3. This hash is digitally signed using the source's private key.
4. Later, during training, the system uses the public key to verify each data-label pair.

If anyone changes the label, even a single bit, the hash won't match. This stops poisoned data from being used in training.

This idea is inspired by how we use blockchain or digital certificates to verify authenticity. It brings the same trust model to machine learning.

Block Diagram of the System

Here’s a breakdown of my model training and deployment pipeline with the proposed cryptographic protection:

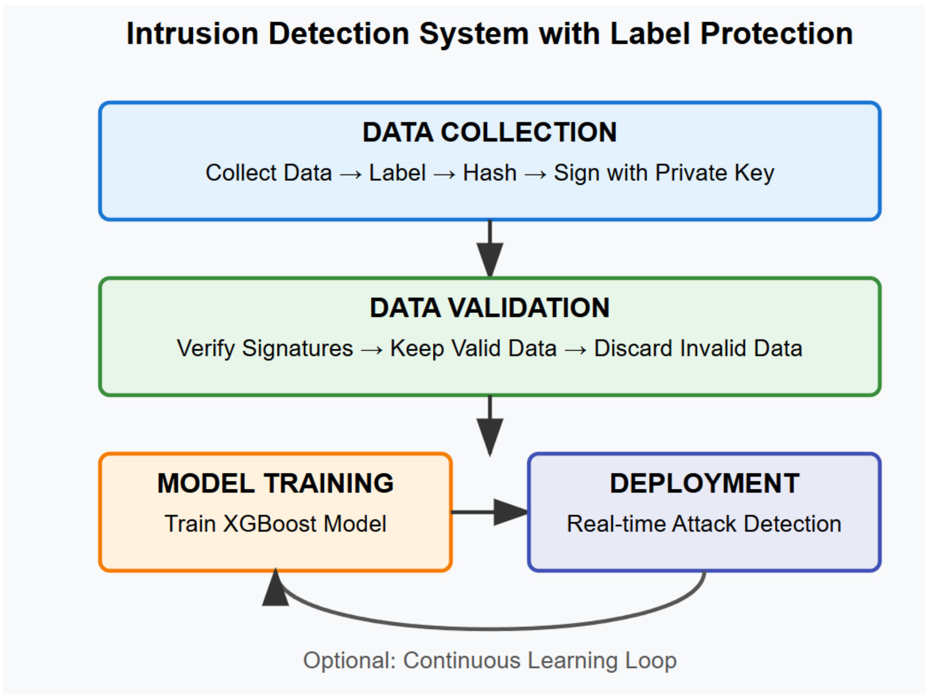

Intrusion Detection System with Label Protection
DATA COLLECTION
Collect Data → Label → Hash → Sign with Private Key
DATA VALIDATION
Verify Signatures → Keep Valid Data → Discard Invalid Data
MODEL TRAINING
Train XGBoost Model
DEPLOYMENT
Real-time Attack Detection
Optional: Continuous Learning Loop

Figure 6. Block diagram

Data Collection Phase

- Source collects traffic data and assigns labels
- Create hash of [features + label]
- Sign the hash with private key
- Store data + label + signature

Data Validation Phase (Before Training)

- Verify each signature with the public key
- If signature is valid → keep the data
- If not → discard the data as tampered

Model Training Phase

- Use only verified data to train XGBoost model
- Save model for deployment

Deployment Phase

- Deployed model detects attacks in real-time

- Optional: Collect new data and repeat the process for continuous learning

Summary of My Findings

- Label poisoning is a serious threat to intrusion detection systems. It can silently break the model.
- Most common solutions act after poisoning happens. They try to catch errors after training.
- I proposed a new approach based on digital signatures and hashing. This method prevents label manipulation.
- By using cryptographic checks, we can ensure that only trusted, verified labels are used for training.
- This adds an extra layer of security that’s lightweight, scalable, and easy to integrate into the existing pipeline.

References

[1] [Online]. Available: Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., & Zhang, L. (2016). Deep learning with differential privacy. Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, 308-318. https://doi.org/10.11. [Accessed 30 04 2025].

[2] [Online]. Available: Akhtar, N., & Mian, A. (2018). Threat of adversarial attacks on deep learning in computer vision: A survey. IEEE Access, 6, 14410-14430. https://doi.org/10.1109/ACCESS.2018.2807385. [Accessed 30 04 2025].

[3] [Online]. Available: Biggio, B., Nelson, B., & Laskov, P. (2012). Poisoning attacks against support vector machines. Proceedings of the 29th International Conference on Machine Learning, 1807-1814.. [Accessed 29 04 2025].

[4] [Online]. Available: Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 785-794. https://doi.org/10.1145/2939672.2939785. [Accessed 29 04 2025].

[5] [Online]. Available: Choi, J., Kim, J., Kim, H., Chekole, E. G., & Lee, J. (2022). RAID: A comprehensive study on the robustness of network intrusion detection systems against adversarial attacks. IEEE Transactions on Network and Service Management, 19(3), 2454-2468. https://. [Accessed 30 04 2025].