IDS for cryptographic protocol misuses

Sreenija Kanugonda
Department of Computer Science
Georgia State University
skanugonda1@student.gsu.edu

Shishir Kumar Vallapuneni
Department of Computer Science
Georgia State University
svallapuneni1@student.gsu.edu

***Abstract*—Encryption is essential for safeguarding information, but ironically, it has also become a tool for attackers to conceal their activities — a dilemma we refer to as the encryption paradox. In this work, we propose a privacy-respecting detection framework that identifies misuse within TLS-encrypted traffic without decrypting packet payloads. Our method relies on analyzing metadata alone, combining flow-level characteristics such as session duration, packet counts, and entropy with TLS handshake details like protocol version, certificate validity, cipher suite selection, and handshake timing. We extracted these features from three real-world datasets CIC-IDS2017, CIC-Darknet2020, and USTC-TFC2016 and formulated the problem as a multi-label classification task. To evaluate detection performance, we compared three machine learning models: Random Forest, XGBoost, and a Deep Multi-Layer Perceptron (MLP). Among them, XGBoost achieved a micro-F1 score exceeding 0.95, while the MLP demonstrated an ability to detect more subtle misuse patterns with comparable accuracy. Beyond detection, we outline how this approach can be integrated into enterprise intrusion detection systems (IDS/IPS), cloud monitoring infrastructures, and large-scale ISP-level traffic analysis.**

I. Introduction

Encryption has become a cornerstone of modern digital communication, playing a critical role in securing financial transactions, healthcare records, and interactions on social platforms. Users today heavily depend on encryption protocols to protect the confidentiality and integrity of their data exchanges. Central to this protective shield is Transport Layer Security (TLS), which ensures that information remains private, authenticated, and free from tampering while in transit.

Yet, this heavy reliance on encryption introduces an ironic challenge: while encryption shields legitimate communication from prying eyes, it also blinds traditional security mechanisms. Intrusion Detection Systems (IDS), firewalls, and packet inspection tools have historically depended on reading packet contents to detect threats. In today's world, dominated by HTTPS, the contents are no longer easily accessible, giving rise to what is known as the *encryption paradox*.

Adversaries have quickly adapted to this shift. Instead of attacking encryption algorithms directly, they exploit weaknesses in how encryption is deployed. Configuration errors, mishandled certificates, and support for outdated protocols create opportunities for attacks like SSL stripping [5], TLS downgrade exploits [6], man-in-the-middle (MITM) interceptions, and the use of invalid or self-signed certificates. Malware such as Dridex has also been known to leverage encrypted tunnels for command-and-control (C2) activities, evading detection [7].

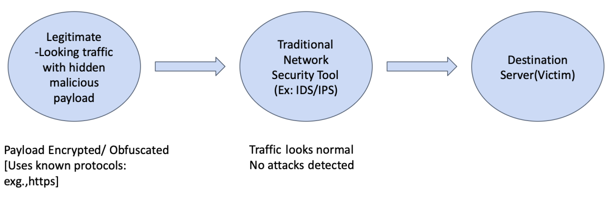
Legitimate -Looking traffic with hidden malicious payload
Traditional Network Security Tool (Ex: IDS/IPS)
Destination Server(Victim)
Payload Encrypted/ Obfuscated [Uses known protocols: exg.,https]
Traffic looks normal No attacks detected

Fig. 1. Encrypted malicious traffic bypassing traditional network security tools — the encryption paradox.

Overcoming this paradox demands a new approach: rather than decrypting traffic, detection efforts must focus on analyzing *traffic metadata*—details such as handshake patterns, certificate attributes, negotiated TLS versions, flow entropy, and packet statistics. These indirect indicators can reveal misuse without breaching the confidentiality encryption aims to uphold.

Our primary Steps are summarized as follows:

- Development of a privacy-respecting metadata extraction pipeline tailored for encrypted traffic analysis.
- Creation of a balanced, labeled dataset covering SSL stripping, TLS downgrades, MITM, and weak certificate scenarios.
- Comparative evaluation of ensemble learning techniques and deep learning methods for detecting

cryptographic misuse.
- Insights into deployment strategies and future avenues for improving encrypted traffic threat detection.

II. Background and Threat Landscape

A. TLS Fundamentals

Transport Layer Security (TLS) serves as the foundation for private and authenticated communication across the Internet. Built on top of TCP, TLS secures billions of transactions every day, from financial exchanges to personal messaging. A critical step in this process is the TLS handshake, where the client and server agree on encryption settings, verify each other's identities using certificates, and establish shared session keys.

The handshake process generally involves the following steps:

- **ClientHello:** The client proposes supported TLS versions, cipher suites, and key exchange methods.
- **ServerHello:** The server selects the encryption parameters and presents its certificate.
- **Key exchange:** A shared secret is established between client and server.
- **Session establishment:** Secure, encrypted communication begins.

TLS 1.3 introduced major improvements, including the elimination of outdated algorithms and a faster handshake process. Despite this, many systems still support older versions like SSL 3.0 and TLS 1.0/1.1, leaving them vulnerable to known exploits.

While the actual data transmitted is encrypted following the handshake, several handshake parameters — such as the TLS version, cipher suite choice, and certificate attributes — remain visible. Monitoring this metadata provides a valuable, non-intrusive way to assess the security posture of encrypted sessions.

B. Cryptographic Protocol Misuses

Rather than trying to break encryption algorithms directly, modern attackers often exploit flaws in how TLS is implemented or managed. Several common misuse patterns have been observed:

1) SSL Stripping: In SSL stripping attacks, adversaries downgrade an HTTPS connection to HTTP by interfering with the initial request. This tactic leaves users exposed to credential theft and session hijacking. The SSLStrip tool, introduced by Moxie Marlinspike in 2009, brought wide attention to this vulnerability and led to the adoption of security measures like HSTS [5].

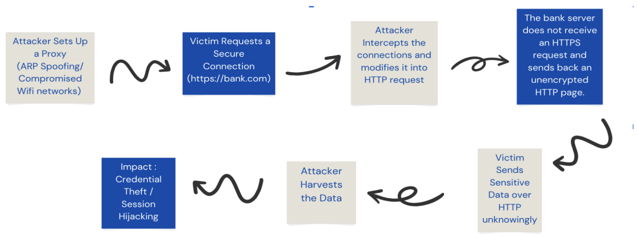
Attacker Sets Up a Proxy (ARP Spoofing/ Compromised Wifi networks)
Victim Requests a Secure Connection (https://bank.com)
Attacker Intercepts the connections and modifies it into HTTP request
The bank server does not receive an HTTPS request and sends back an unencrypted HTTP page.
Victim Sends Sensitive Data over HTTP unknowingly
Attacker Harvests the Data
Impact : Credential Theft / Session Hijacking

Fig. 2. SSL Stripping Attack Flow Overview.

2) TLS Downgrade (POODLE Attack): A downgrade attack tricks a connection into falling back to older, insecure protocols such as SSL 3.0. The POODLE attack specifically exploited weaknesses in SSL 3.0's padding implementation, enabling attackers to decrypt sensitive communications [6].

3) Man-in-the-Middle (MITM): Man-in-the-middle attacks occur when a malicious actor intercepts and potentially manipulates communication between two parties. This can involve presenting forged certificates, compromising certificate authorities, or tampering with DNS responses to redirect traffic.

4) Weak or Self-Signed Certificates: Some servers mistakenly use expired, self-signed, or improperly issued certificates. Attackers take advantage of these weaknesses to impersonate legitimate sites, deceiving users into trusting fraudulent connections.

III. Materials and Methods

A. Datasets Used

This study relies on three publicly available datasets selected to cover a wide range of encrypted traffic behaviors, attack types, and real-world misuse patterns:

- **CIC-IDS2017:** Compiled by the Canadian Institute for Cybersecurity [1], this dataset includes normal network traffic as well as various attack scenarios, such as SSL stripping, man-in-the-middle attacks, and certificate misuse. It provides both PCAP files and extensive flow-level metadata, enabling detailed analysis at the network and TLS layers.
- **CIC-Darknet2020:** This collection focuses on encrypted command-and-control (C2) traffic frequently used by malware [2]. It captures traffic originating from the dark web, offering realistic examples of how attackers leverage encryption.
- **USTC-TFC2016:** Produced by the University of Science and Technology of China [3], this dataset contains TLS-encrypted flows from both benign and malicious sources, making it valuable for studying

handshake timing irregularities and flow-level encryption characteristics.

The datasets were intentionally chosen to ensure diversity in traffic types, improving the generalizability of our trained models across different misuse cases.

B. Feature Engineering and Extraction

To maintain privacy and reduce computational overhead, we focused on extracting observable *metadata* rather than decrypting packet contents. Feature engineering was organized into two key categories:

1) TLS Metadata Extraction: We utilized PyShark, a Python interface for the TShark network protocol analyzer [4], to extract critical handshake-level details from the PCAP files:

- **TLS Version:** Highlights the protocol version negotiated, identifying outdated versions vulnerable to downgrade attacks.
- **Cipher Suite:** Specifies the encryption and hashing algorithms agreed upon.
- **Handshake Duration:** Extended handshake times can signal interception or MITM attempts.
- **Certificate Fields:** Captures information about certificate issuers, subjects, and validity periods to flag issues like expired or self-signed certificates.

Analyzing these fields allowed us to uncover potential anomalies without decrypting the actual data.

2) Flow-Level Feature Extraction: In addition to TLS-specific details, we computed statistical features derived from flow-level metadata:

- **Flow Duration:** Measures the overall lifetime of a session.
- **Packet and Byte Counts:** Quantifies the volume of communication in both directions.
- **Packet Size Entropy:** Estimates randomness in packet sizes, typically elevated in encrypted or obfuscated traffic.
- **Byte and Packet Rates:** Calculates the throughput and density of network flows.
- **Inter-Arrival Times (IAT):** Tracks variability in packet arrival intervals, which may reveal suspicious timing patterns.

Altogether, more than 50 features were extracted per flow. Continuous variables were normalized to zero mean and unit variance, while categorical fields such as cipher suites were numerically encoded.

C. Label Generation and Dataset Preparation

Since a single flow could exhibit multiple misuse traits simultaneously, we approached the detection problem as a **multi-label classification task**.

Each flow was assigned one or more binary labels corresponding to the following categories:

- SSL Stripping
- TLS Downgrade (e.g., POODLE vulnerability) [6]
- Man-in-the-Middle (MITM)
- Weak or Self-Signed Certificates

Labeling combined information from dataset ground truths with heuristics based on extracted features. For example, sessions negotiating SSL 3.0 were flagged as downgrade attempts, while those using self-signed certificates were labeled under weak certificate misuse.

To build a balanced dataset suitable for multi-label learning, we performed under-sampling of the dominant benign classes. The resulting dataset included around 5,000 flows, ensuring adequate representation across all four misuse types.

D. Machine Learning Models

Three distinct machine learning models were developed and evaluated, each representing a different learning approach:

1) Random Forest: A Random Forest consisting of 25 decision trees [8], with a maximum depth of 7, was trained. Random Forests are well-suited for structured datasets and naturally highlight feature importance while reducing overfitting by aggregating multiple trees.

2) XGBoost: Extreme Gradient Boosting (XGBoost), a high-performance boosting algorithm [9], was configured with 13 estimators, a maximum depth of 6, and a learning rate of 0.05. Its built-in handling of missing data and class imbalance made it a strong contender for the multi-label classification task.The boosting structure of general XGBoost is illustrated in Figure **??**, where each decision tree contributes to the final prediction in a sequential manner.

3) Deep Multi-Layer Perceptron (MLP): We implemented a deep feed-forward neural network with three hidden layers comprising 512, 256, and 128 neurons, respectively. LeakyReLU activations promoted stable learning, while dropout regularization (ranging from 20% to 30%) helped prevent overfitting. The network was trained using the Adam optimizer with a binary cross-entropy loss, appropriate for independent multi-label outputs. Early stopping was applied to terminate training if validation loss ceased improving.

All models were trained using an 80/20 stratified train-test split to preserve label distributions. Hyperparameter tuning was conducted through randomized grid search, and model selection prioritized maximizing the micro-averaged F1 score.

Figure 3 shows the layered architecture of Deep MLP model, highlighting its dense connectivity and capacity for learning non-linear relationships.

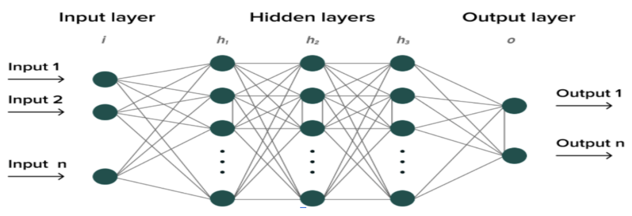

Fig. 3. Structure of a Deep MLP.

E. Evaluation Metrics

Given the multi-label nature of the classification task, multiple metrics were used to capture different aspects of model performance. These include metrics that consider both overall correctness and class-level balance.

- **Hamming Loss:** Measures the fraction of incorrect labels to the total number of predicted labels.
- **Micro-Averaged Precision, Recall, and F1 Score:** Aggregates contributions of all classes to compute overall performance.
- **Macro-Averaged F1 Score:** Computes the metric independently for each label, then averages the results—treating all classes equally.
- **Subset Accuracy:** The most stringent metric, which considers a prediction correct only if all labels match exactly.

The following equations define these metrics:

1. Hamming Loss:

$$\text{Hamming Loss} = \frac{1}{N \cdot L} \sum_{i=1}^{N} \sum_{j=1}^{L} \mathbf{1}[y_{ij} \neq \hat{y}_{ij}] \qquad (1)$$

Where:

- N is the number of samples
- L is the number of labels
- y_{ij} is the true value (0 or 1) for label j of sample i
- $\hat{y}_{ij}$ is the predicted value
- $\mathbf{1}[\cdot]$ is the indicator function

4. Micro-Averaged F1 Score:

$$\text{F1}_{micro} = \frac{2 \cdot \text{Precision}_{micro} \cdot \text{Recall}_{micro}}{\text{Precision}_{micro} + \text{Recall}_{micro}} \qquad (2)$$

5. Macro-Averaged F1 Score:

$$\text{F1}_{macro} = \frac{1}{L} \sum_{l=1}^{L} \frac{2 \cdot Precision_l \cdot Recall_l}{Precision_l + Recall_l} \qquad (3)$$

Where:

- TP_l, FP_l, and FN_l refer to true positives, false positives, and false negatives for label l

6. Subset Accuracy:

$$\text{Subset Accuracy} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}[Y_i = \hat{Y}_i] \qquad (4)$$

Where:

- Y_i and $\hat{Y}_i$ are the sets of true and predicted labels for sample i

IV. Results and Discussion

A. Training Dynamics: Deep Multi-Layer Perceptron (MLP)

The Deep MLP model was trained over 25 epochs using the extracted feature set. The corresponding training and validation loss trends are depicted in Figure 4. In the early stages of training, both loss values decreased steadily, indicating that the model was effectively capturing important patterns from the metadata.

Training was halted after approximately 18 epochs due to early stopping, once the validation loss ceased to improve, thereby helping to prevent overfitting. This suggests that the MLP generalized well to unseen data without merely memorizing the training samples.

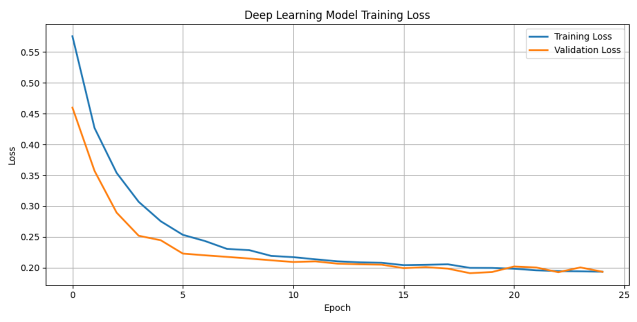

Fig. 4. Training and Validation Loss Curves for Deep MLP.

The use of dropout layers and LeakyReLU activation functions contributed to stable convergence. Nevertheless, as anticipated, the Deep MLP showed higher sensitivity to hyperparameter choices compared to the tree-based models.

B. Model Comparison

We evaluated Random Forest, XGBoost, and Deep MLP models using several multi-label classification metrics: Hamming Loss, Micro-Averaged F1 Score, Macro-Averaged F1 Score, and Subset Accuracy. A summary of the results is presented in Table I and illustrated in Figure 5.

TABLE I
MODEL PERFORMANCE COMPARISON

Model	Hamming Loss	Micro-F1	Macro-F1	Subset Accuracy
Random Forest	0.02	0.97	0.96	0.96
XGBoost	0.02	0.97	0.95	0.96
Deep MLP	0.07	0.87	0.86	0.77

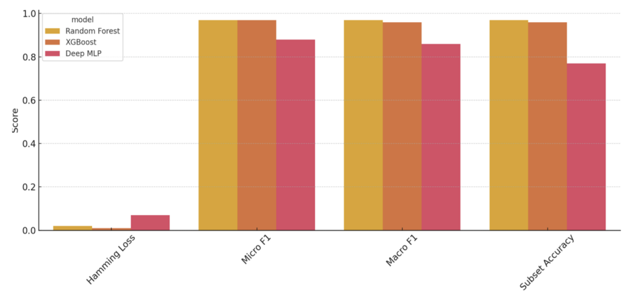

Fig. 5. Comparison of Hamming Loss, Micro-F1, Macro-F1, and Subset Accuracy Across Models.

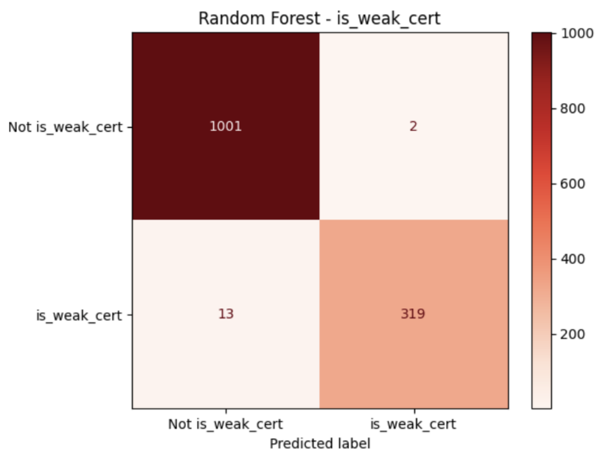

Fig. 6. Random Forest confusion matrix for `is_weak_cert` classification.

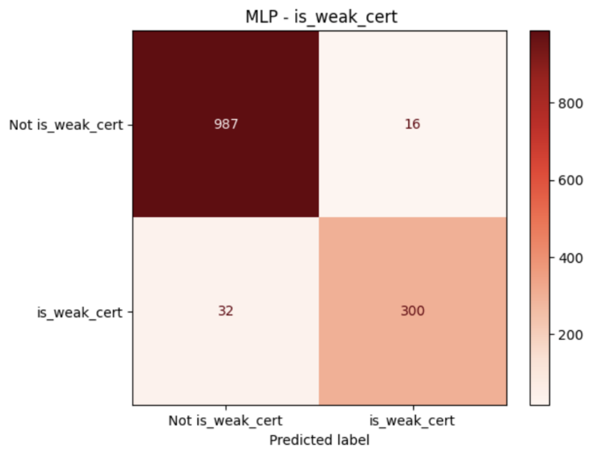

Fig. 7. Deep MLP confusion matrix for `is_weak_cert` classification.

C. Confusion Matrices

To gain a more detailed understanding of how each classifier performed on specific misuse types, we examined the confusion matrices for all the classes. Here we can see performances of models on the `is_weak_cert` class — which corresponds to the detection of flows containing expired, self-signed, or otherwise invalid certificates.

From the confusion matrices, several insights emerge regarding the model behavior:

Random Forest correctly identified 319 out of 332 malicious flows involving weak certificates, with only 13 false negatives and 2 false positives. This indicates a strong ability to distinguish benign from misuse flows with minimal confusion. Its high true positive rate, combined with a low false positive count, makes it a dependable choice for certificate-based misuse detection.

XGBoost achieved nearly identical performance, correctly classifying 318 weak certificate instances while producing only 1 false positive and 14 false negatives. The slight variations compared to Random Forest suggest that XGBoost may be slightly more conservative in misclassification, particularly in ambiguous edge cases. Nonetheless, its confusion matrix reflects excellent precision and recall for this misuse category.

The Deep MLP model, on the other hand, showed a notably higher error rate. It classified only 300 out of 332 positive instances correctly and produced 32 false negatives — nearly two and a half times more than the tree-based models. Moreover, it yielded 16 false positives, misidentifying benign flows as misuse. These results suggest that while neural networks are capable of learning complex patterns, they may be more sensitive to class imbalance and noise in the feature space, especially when metadata alone is used without raw packet-level input.

These results reinforce the findings from the overall metrics: while the MLP can capture non-linear patterns in encrypted traffic, it lacks the consistency and precision of Random Forest and XGBoost when dealing with specific misuse types. In practical deployment scenarios — especially in environments where false positives may have operational costs — ensemble models provide a more robust and trustworthy option.

D. Discussion of Results

Both Random Forest and XGBoost achieved outstanding performance, with micro-F1 scores exceeding 0.95 and very low Hamming Loss rates of 2%. These results demonstrate that the models accurately predicted most labels across multiple classes.

Between the two ensemble methods, Random Forest showed a slight edge in macro-F1 score, indicating better balanced performance across all misuse types. XGBoost, however, exhibited greater robustness in handling less frequent misuse categories, such as encrypted malware traffic.

The Deep MLP model also performed reasonably well but fell behind the ensemble methods. Although it achieved a respectable micro-F1 score of 0.87, the model's lower subset accuracy (77%) reflects occasional partial mispredictions in the multi-label setting. These results suggest that while neural networks can learn rich patterns from traffic metadata, they require more data and tuning to match the robustness of tree-based approaches.

Overall, the findings affirm that encrypted traffic metadata contains rich, discriminative signals that can be used to identify cryptographic protocol misuse without the need to decrypt the underlying data.

V. Conclusion and Future Work

This study tackled the encryption paradox—the challenge where encryption, while securing legitimate communication, also allows malicious activities to remain hidden. Rather than decrypting traffic and violating user privacy, we demonstrated that analyzing TLS handshake metadata and flow statistics can effectively expose cryptographic protocol misuse.

By extracting features such as handshake duration, negotiated TLS versions, certificate properties, and flow entropy, we trained models capable of categorizing encrypted flows into misuse types including SSL stripping, TLS downgrades, MITM attacks, and weak certificate scenarios.

Testing on three public datasets — CIC-IDS2017, CIC-Darknet2020, and USTC-TFC2016 — revealed that Random Forest and XGBoost models consistently achieved high detection performance, with micro-F1 scores surpassing 0.95. Deep MLP models also performed well, highlighting their ability to capture subtle traffic patterns, albeit with slightly more sensitivity to data quality and hyperparameter tuning.

Overall, the results validate that valuable threat intelligence can be extracted purely from encrypted traffic metadata, offering a privacy-respecting approach to securing encrypted communications.

A. Future Work

While this work lays a strong foundation, several avenues exist for future exploration:

- **Real-time Deployment:** Integrating the detection framework with real-time processing pipelines (e.g., Kafka streams) to allow live monitoring of encrypted traffic.
- **Learning from Raw Packet Sequences:** Investigating deep learning models like 1D convolutional neural networks (CNNs) or transformer architectures trained directly on packet arrival patterns and handshake sequences.
- **Zero-Day Misuse Detection:** Enhancing the system's ability to detect unknown or evolving misuse behaviors using semi-supervised learning and anomaly detection techniques.
- **Dataset Expansion:** Broadening the training corpus to include IoT traffic, emerging encryption standards like QUIC, and other real-world encrypted traffic sources.

By extending the framework in these directions, it may be possible to strengthen encrypted traffic monitoring further without compromising the fundamental principles of user privacy.

Code Availability

The implementation code used in this work, including feature extraction, model training, and evaluation, is available at:

https://github.com/Shishirr11/
IDS-for-cryptographic-protocol-misuses

References

[1] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, “Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization,” in *Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP)*, 2018.

[2] Canadian Institute for Cybersecurity, “CIC Darknet 2020 Dataset,” [Online]. Available: https://www.unb.ca/cic/datasets/darknet2020.html.

[3] L. Zhao, J. Li, Z. Sun, and Y. Jin, “USTC-TFC2016: A Traffic Classification DataSet with Encrypted Traffic Flows,” University of Science and Technology of China.

[4] K. Karasarinis, “PyShark: Pythonic Wireshark Packet Parsing,” [Online]. Available: https://github.com/KimiNewt/pyshark.

[5] M. Marlinspike, “New Tricks for Defeating SSL in Practice,” presented at Black Hat USA, 2009.

[6] B. Moeller, T. Duong, and K. Kotowicz, “This POODLE Bites: Exploiting the SSL 3.0 Fallback,” Google Online Security Blog, 2014. [Online]. Available: https://googleonlinesecurity.blogspot.com/2014/10/this-poodle-bites-exploiting-ssl-30.html.

[7] F. Paget, “The Evolution of Dridex Malware,” McAfee Labs Threat Report, 2019.

[8] L. Breiman, “Random Forests,” *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[9] T. Chen and C. Guestrin, “XGBoost: A Scalable Tree Boosting System,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 785–794, 2016.