Explainable Anomaly Detection for Zero-Day Threats: An Interpretable Machine Learning Approach for IoT Network Intrusion Detection

Khondokar Sajid (2211954042), Kazi Safin Arafat (2211778642),
Moushumi Akter Mow (2021983642), Rakibul Hasan Ridoy (1731339042)
Department of Electrical and Computer Engineering
North South University, Dhaka, Bangladesh
Faculty Advisor: Md. Mohammad Shifat-E-Rabbi
CSE499A: Senior Design I, Fall 2025

***Abstract*—Zero-day attacks are used most often by attackers to exploit software vulnerabilities that people have not yet learned how to recognize, thus striking first. Such attacks wreak havoc before there is a patch available because most intrusion detection systems merely scan for patterns they have encountered before. A machine learning system in our project can both locate and comprehend what they located. It not only recognizes that someone is behaving abnormally; it recognizes it too. Autoencoders, Isolation Forests, and One-Class SVMs are just a few of the models we employ to seek out abnormal behavior that may be the indication of a new attack. We won't stop there, however. We demonstrate the "why" of each alert by demonstrating how significant a feature is at a feature level using SHAP values and how an anomaly varies from what can be anticipated of traffic with prototype-based explanations. We will benchmark our pipeline against benchmarking datasets such as NSL-KDD, CIC-IDS2017, and UNSW-NB15. We will conduct testing with attack types never encountered before to make tests closer to real zero-day cases. We have an uncommon approach since we don't only care about accuracy. We care about trust, ease, and speed as well. And we're packaging our platform to make it small enough to deploy on routers, sensors, smartphones, and even Raspberry Pis. We're building a cybersecurity tool that is cutting, quick, and easy to understand—one that can be relied on by analysts and used in the real world—by integrating explainability, zero-day detection, and IoT optimization.**

***Index Terms*—anomaly detection, explainable AI, zero-day threats, intrusion detection, SHAP, prototype-based explainability, lightweight AI**

I. Introduction

A. Motivation and Background

Zero-day attacks are a huge pain for cybersecurity—they hit secret software weaknesses before anyone can fix them. Think of the SolarWinds hack in 2020 or Log4Shell in 2021; they caused massive damage, hitting companies and governments because no one was ready. Regular Intrusion Detection Systems (IDS) that rely on known attack patterns just can't keep up. Plus, security teams get flooded with thousands of alerts every day, and most are false alarms, which is super frustrating and risky.

We're pumped to use machine learning (ML) to catch these threats by spotting weird patterns. Models like Autoencoders (checking if data rebuilds right), Isolation Forests (finding oddballs in data), and One-Class SVMs (drawing lines around normal stuff) are great for this. But the problem is, these models don't explain themselves—they're like black boxes spitting out alerts with no "why." That's why we're building a system that not only catches attacks but explains them clearly so analysts can trust it.

Also, IoT devices—like smart lights, sensors, or phones—are everywhere, but they're super easy targets. Attacks on IoT have jumped over 300% lately, and these devices don't have the power for heavy ML models. So, we're focusing on lightweight solutions that run on stuff like an ESP32 or Android phone without draining batteries or needing the cloud all the time. It's a big challenge, but we're excited to tackle it!

B. Literature Review and Gaps

Lots of researchers have worked on anomaly detection with explainable AI (XAI). Pande and Khamparia [1] used explainable neural networks to make intrusion detection clearer. Krishnan et al. [2] built an attention-based model for IoT zero-day attacks, showing interpretability matters on small devices. Fatema et al. [3] mixed federated learning with SHAP for privacy-safe detection, while Almolhis [4] used random forests with attention and XAI visuals. Muhammad et al. [5] made a LIME-based IDS for cluster systems.

Recent work adds more ideas. Peisker et al. [6] compared LIME and SHAP for deep learning IDS, Wali and Alam [7] tackled encrypted traffic, Singh [8] focused on vehicular networks, and Akter et al. [9] reviewed XAI for insider threats.

But there are still gaps we want to fix:

- **Weak Zero-Day Testing:** Most studies don't test real zero-day scenarios by leaving out whole attack types.
- **No Human Testing:** Few check if explanations help people make faster or better decisions.
- **Lack of Explanation Comparisons:** Not enough side-by-side looks at SHAP, LIME, or prototypes for clarity and speed.

- **Ignoring IoT Devices:** Models are too heavy for low-power devices like IoT or phones.
- **Missing Overhead Data:** Nobody measures how much time or power explanations take for real-time use.
- **No Hybrid Edge Solutions:** Combining multiple models with XAI for IoT is rare.
- **Not Future-Ready:** Few think about 5G or quantum-safe methods for IoT security.

As students, we're not trying to invent something wild. We're taking what's out there, mixing it smartly, and tweaking it for IoT with clear explanations and lightweight designs.

C. Research Objectives

Here's what we're aiming for:

- Build a pipeline with Autoencoders, Isolation Forests, and One-Class SVMs, plus SHAP and prototype explanations.
- Test zero-day detection on NSL-KDD, CIC-IDS2017, and UNSW-NB15, focusing on new attack types.
- Compare explanation methods for clarity, consistency, and speed, especially on IoT devices.
- Run human studies to see if explanations help analysts (especially beginners like us) decide faster and trust the system.
- Optimize the pipeline for IoT and phones using small datasets and compressed models.
- Check real-time performance and scalability in IoT network setups.

II. Problem Statement

Zero-day attacks are a big deal because they exploit vulnerabilities in software that no one knows about yet. Think of a hacker using an unknown flaw to steal data, crash computers, or take over devices like smart cameras or hospital sensors—just like the Mirai botnet in 2016, when thousands of IoT devices got hijacked to attack websites.

Normal security software, like signature-based IDS, can't stop them because they only detect known attacks. When a zero-day hits, it can steal sensitive info or shut down critical systems, and it might take weeks or months to fix.

Machine learning (ML) can help by spotting unusual patterns in network traffic, like strange data packets or connections. But here's the catch: ML models are like mysteries with no clues. They'll yell, "Hey, something's wrong!" without saying why, so security analysts—especially newbies like us—get confused and don't trust the alerts. Plus, IoT devices, like smart sensors or phones, have tiny processors and batteries, so running big ML models on them is really tough. Most research assumes you'll send data to powerful cloud servers, but that's slow, uses lots of bandwidth, and risks privacy.

We're trying to fix this with a system that's both clear and lightweight. We want our ML models to explain why they flag something (like, "This alert is because of weird port activity") so analysts can act fast and trust the system. We're also making it run on IoT devices like an ESP32 or Android phone without needing the cloud, which saves time and keeps things secure. By doing this, we're tackling two big problems:

- **Making ML Clear:** Helping analysts understand alerts to cut down on false alarms (which can be 90% of alerts!) and catch real threats faster.
- **Working on Weak Devices:** Designing models that run smoothly on low-power IoT devices for real-time detection without draining batteries.

This is super important because clear, efficient systems can stop zero-day attacks before they cause chaos, especially in the growing world of IoT where devices are everywhere but really vulnerable.

With over 50 billion IoT devices expected by 2030, the attack surface is massive, and zero-day exploits are a growing threat. For example, attacks like Reaper in 2017 targeted IoT devices to build botnets, disrupting entire networks. Our system aims to catch these sneaky attacks by focusing on real-time detection directly on devices, reducing latency to under 1 second, which is critical for IoT networks like smart homes or healthcare systems.

Explainability isn't just about trust—it's also a legal need, like GDPR's "right to explanation," which requires systems to justify automated decisions. By making our alerts clear, we're helping analysts, especially students like us, learn attack patterns and improve future defenses.

Our novel approach combines lightweight models with explainable AI, a rare mix for IoT, ensuring both security and usability in resource-constrained environments.

III. Proposed Methodology

A. Data and Preprocessing

We'll use NSL-KDD (cleaned-up KDD Cup data), CIC-IDS2017 (real-world traffic), and UNSW-NB15 (modern network features). We'll preprocess by encoding categories (like protocols), normalizing numbers (like packet sizes), and balancing data with undersampling. To mimic zero-days, we'll leave out attack types (like DoS or Probe) during training and test on them. For IoT, we'll shrink datasets to 10–20% of their size to simulate edge training and might add synthetic IoT traffic using tools like Scapy.

B. Anomaly Detection Models

Our pipeline uses three models for robust detection:

- **Autoencoder:** Rebuilds data; big errors mean anomalies. We'll make it light with 3–5 layers and 8-bit quantization.
- **Isolation Forest:** Uses trees to spot outliers; fast and tunable for contamination.
- **One-Class SVM:** Draws a boundary around normal data; flexible with RBF kernels.

We'll combine them with majority voting and prune them to fit IoT devices (aiming for $<$ 10MB models).

C. Model Architecture

Our pipeline integrates Autoencoders, Isolation Forests, and One-Class SVMs into a hybrid system for zero-day detection. The Autoencoder uses a 3-layer encoder-decoder with ReLU activation, optimized to 8-bit precision via quantization-aware

training. The Isolation Forest leverages Scikit-learn's implementation, tuned for low-depth trees to reduce memory use. The One-Class SVM uses an RBF kernel with a nu parameter of 0.1 for flexibility. Outputs are combined via majority voting, weighted by model confidence. SHAP generates feature importance scores (e.g., packet size, port number) for each alert, while prototypes use k-NN to compare anomalies to normal traffic, visualized as distance plots. For IoT, we'll deploy via TensorFlow Lite, using pruning to cut 30–40% of weights and batch inference to minimize latency below 500ms on devices like ESP32.

D. Optimization for IoT Devices

To run on IoT or phones (like ESP32 or Android), we'll compress models with pruning (cutting weak weights), quantization (lowering to 8-bit precision), and knowledge distillation (shrinking big models into small ones). We'll test on-device performance with TensorFlow Lite, measuring:

- **Latency:** Average inference time per sample (ms).
- **Energy Usage:** Measured via power profiler tools.
- **Model Size:** Size before and after pruning/quantization.
- **Trade-offs:** Accuracy vs. computational efficiency curves.

We'll batch inputs and tweak thresholds to save power while keeping detection sharp.

E. Explainability Techniques

We'll use:

- **SHAP:** Shows which features (like packet size) matter for each alert; we'll use fast kernel SHAP for speed.
- **Prototypes:** Compares anomalies to normal samples, visualized with distance metrics (e.g., Euclidean) for easy understanding.

To save power, we'll only explain flagged anomalies.

F. Explainability Evaluation Metrics

We'll evaluate explanations with clear metrics:

- **Fidelity:** Correlation between SHAP-predicted feature importance and model score variation, using Pearson correlation coefficient.
- **Consistency:** Cosine similarity between explanations of similar samples to check stability.
- **Comprehensibility:** Mean Likert rating (1–5 scale) from users on clarity.

These will show if our explanations are clear and reliable.

G. False Positive Reduction via Explainability

We'll test if filtering alerts with low SHAP confidence (e.g., weak feature contributions) or inconsistent prototypes (e.g., high variance in similarity) can cut false positives without missing real threats. This could reduce the 90%+ false positive rate, making our system more practical.

H. Evaluation Metrics

For detection: Precision, Recall, F1-Score, False Positive Rate, ROC-AUC, and zero-day accuracy. For explanations: Fidelity, Consistency, Comprehensibility, and Overhead (time/memory). For IoT: Energy consumption and inference speed on devices like ESP32 or Android.

I. Baseline Comparison

We'll compare our explainable pipeline against a non-explainable Random Forest IDS to measure the value of explainability in accuracy, false positive reduction, and analyst trust.

J. Human-in-the-Loop Study

We'll recruit 8–12 cybersecurity students or pros (focusing on beginners) to review 50–100 alerts with or without explanations. We'll measure:

- Decision accuracy (true/false positive identification).
- Decision time per alert.
- Confidence (1–5 Likert scale).

We'll also do interviews to hear their thoughts on explanation clarity.

IV. Project Timeline

To keep on track, we've planned deliverables tied to our class schedule:

- **Class 3 (Week 3):** Submit proposal report and slides with related work and system architecture sketch.
- **Class 5 (Week 5):** Present literature review, highlighting gaps and our approach, with references.
- **Class 6 (Week 6):** Deliver solution report with methodology, datasets, and pseudocode.
- **Class 8 (Week 8):** Provide progress report with early model tests, small dataset results, and challenges.
- **Class 9 (Week 9):** Share code snapshot, demo video of pipeline on IoT setup (e.g., ESP32), and explanation outputs.
- **Class 11 (Week 11):** Submit final report, presentation, complete code with docs, and human study results.

V. Expected Deliverables and Outcomes

- A working detection system with explainability, optimized for IoT devices like ESP32 or Android, deployable in real-world scenarios.
- A user-friendly visualization dashboard showing SHAP feature importance and prototype comparisons, making alerts easy to understand for analysts.
- Tables and graphs comparing model performance (e.g., F1-Score, ROC-AUC), explanation quality (e.g., fidelity, comprehensibility), and IoT efficiency (e.g., latency, energy).
- Zero-day detection results outperforming the Random Forest baseline, with at least 10% higher accuracy on unseen attack types.
- Explanation evaluations for fidelity, consistency, comprehensibility, and overhead, proving our system's reliability.

- Human study data on decision accuracy, speed, and trust, plus qualitative feedback from interviews with cybersecurity beginners.
- A GitHub repo with README, requirements.txt, Jupyter notebooks, and a detailed IoT deployment guide for devices like ESP32 and Raspberry Pi.
- A mobile app prototype (if feasible) showcasing real-time alerts on Android, plus a proposal for a student conference paper to share our unique approach.
- An open-source contribution plan to make our pipeline accessible to the cybersecurity community, fostering further research and real-world adoption.

Our deliverables emphasize novelty by combining explainable AI with lightweight IoT deployment, a rare approach that could set a new standard for zero-day detection in resource-constrained environments.

VI. Technology Stack

We'll use Python 3.x as our core language, leveraging TensorFlow Lite and PyTorch Mobile for lightweight ML models optimized for IoT. Scikit-learn's Isolation Forest and One-Class SVM implementations will handle anomaly detection, while SHAP's kernel explainer will generate feature importance. For prototypes, we'll use scikit-learn's k-NN with Euclidean distance metrics. Pandas and NumPy will manage data preprocessing, and matplotlib/Seaborn will create visualizations like ROC curves and SHAP plots.

Git/GitHub will store our code, with pytest for unit testing model robustness. For IoT simulation, we'll use Raspberry Pi emulators, Android Studio, and Contiki-NG for network emulation. ONNX will ensure model portability across devices, and cProfile will benchmark performance (e.g., inference time, memory usage). This stack is chosen for its compatibility with low-resource devices and its ability to support our unique blend of explainability and real-time detection.

VII. Potential Challenges and Mitigations

- **Data Scarcity:** Use transfer learning from big datasets to small ones and generate synthetic IoT traffic with Scapy.
- **Overfitting:** Apply cross-validation, dropout, and L2 regularization to keep models generalizable.
- **Recruiting Participants:** Partner with campus cybersecurity clubs and offer incentives like certificates.
- **Device Limits:** Train complex models on cloud servers, then compress for edge deployment using TensorFlow Lite's converter.
- **Ethics:** Ensure data anonymity, obtain informed consent for human studies, and follow GDPR guidelines.

VIII. Conclusion

This project is our shot at making zero-day detection better, clearer, and ready for the IoT world. By blending Autoencoders, Isolation Forests, and One-Class SVMs with SHAP and prototype explanations, we're building a system that's both powerful and easy to understand, especially for IoT devices like sensors and phones. We're super excited to tackle gaps like real-time performance, false positive reduction, and IoT compatibility, which most research overlooks. Our approach is unique because it combines explainable AI with lightweight models tailored for IoT, a rare mix that could redefine cybersecurity for smart homes, healthcare, and beyond. Testing with real people will ensure our explanations help beginners like us trust and use the system effectively. We're not just building a tool—we're creating a game-changer that could protect millions of devices and inspire future innovations. Looking ahead, our pipeline could scale to 5G networks or even quantum-safe systems, making it a foundation for next-gen cybersecurity.

References

[1] S. Pande and A. Khamparia, "Explainable deep neural network based analysis on intrusion detection systems," *Computer Science*, vol. 24, no. 1, 2023.

[2] D. Krishnan, S. Singh, and V. Sugumaran, "Explainable AI for zero-day attack detection in IoT networks using attention fusion model," *Discover Internet of Things*, 2025.

[3] K. Fatema et al., "Federated XAI IDS: An Explainable and Safeguarding Privacy Approach to Detect Intrusion Combining Federated Learning and SHAP," *Future Internet*, vol. 17, no. 6, 2025.

[4] N. Almolhis, "Intrusion Detection Using Hybrid Random Forest and Attention Models and Explainable AI Visualization," *Journal of Internet Services and Information Security*, 2025.

[5] A. E. Muhammad et al., "L-XAIDS: A LIME-based eXplainable AI framework for intrusion detection systems," *Cluster Computing*, 2025.

[6] D. Peisker et al., "Evaluating Explainable AI for Deep Learning-Based Network Intrusion Detection," arXiv preprint arXiv:2506.07882, 2025.

[7] S. Wali and K. Alam, "Explainable AI for Cybersecurity: Interpretable Intrusion Detection in Encrypted Traffic," *Multidisciplinary Frontiers*, 2025.

[8] R. R. Singh, "Exploring the use of Explainable AI for improving intrusion detection in vehicular networks," Master's thesis, National College of Ireland, 2024.

[9] M. Akter et al., "A Comprehensive Survey of Explainable Artificial Intelligence (XAI) in Insider Threat Detection," *IEEE Access*, 2025.