Enhanced Anomaly Detection in Iot Using FPGA Hardware acceleration

Sai Prudvi Ela
Dept of Computer Engineering
Florida Institute of Technology
Melbourne,Florida,32901
Sela2023@my.fit.edu

Aparna Peddireddy Nadipolla
Dept of Electrical Engineeering
Florida Institute of Technology
Melbourne,Florida,32901
apeddireddyn2023@my.fit.edu

Ajay Tammuluri
Dept of Electrical Engineeering
Florida Institute of Technology
Melbourne,Florida,32901
atammuluri2023@my.fit.edu

Chaturta Papajjagari
Dept of Electrical Engineeering
Florida Institute of Technology
Melbourne,Florida,32901
cpapajjagari2024@my.fit.edu

Harshitha Juvvala
Dept of Computer Engineering
Florida Institute of Technology
Melbourne,Florida,32901
hjuvvala2023@my.fit.edu

***Abstract*—The widespread adoption of the Internet of Things (IoT) has significantly influenced various aspects of daily life and business operations, leading to billions of devices connected to the internet. However, the rapid integration and proliferation of IoT devices have also increased vulnerabilities to cyber attacks. Traditional intrusion detection systems (IDS), which are not designed for the dynamic and diverse nature of IoT environments, struggle to identify new or sophisticated attacks. Anomaly-based intrusion detection systems, which learn normal behaviors and detect deviations, offer a promising solution. This paper explores the use of Field-Programmable Gate Arrays (FPGAs) as an efficient means to handle the computational demands of real-time anomaly detection in IoT networks. FPGAs are highlighted for their high throughput and configurability, which are critical in managing the vast and fast-moving data within IoT environments. This study proposes a novel anomaly-based IDS that uses a GPU for training neural networks and a System on Chip (SoC) Zynq platform for efficient real-time inference. This approach utilizes the recently released IoT-23 dataset for validation, demonstrating the potential of hardware acceleration in enhancing IoT security through real-time, efficient anomaly detection.**

***Index Terms*—FPGA, IoT-23 dataset, IDS, CPU, GPU, Neural Network, Anomaly Detection, Internet of Things (IoT), Machine Learning, Cybersecurity, Real-time Processing, Hardware Acceleration, System on Chip (SoC), IoT Security, Data Analysis in IoT.**

I. Introduction

The Internet of Things' (IoT) widespread adoption has changed many facets of our everyday lives, ranging from how we run our homes to how businesses function. By 2021, billions of gadgets will be connected to the internet, gathering and sharing data to improve functionality and efficiency. But this quick integration and growth has also made things more vulnerable to cyber attacks. Because they were initially created for less dynamic contexts, traditional intrusion detection systems (IDS) find it difficult to handle the massive volume and variety of data generated by Internet of Things devices. These systems pose serious security vulnerabilities since they frequently are unable to identify new or complex assaults. An effective method to deal with these issues is anomaly-based intrusion detection. Anomaly-based systems learn what constitutes typical behavior and signal variations, which may suggest a potential breach, in contrast to signature-based systems, which rely on established patterns. Not with standing their benefits, these systems must overcome significant computational obstacles to be implemented successfully in an IoT setting, particularly in real-time applications.since of their high throughput and configurability, Field-Programmable Gate Arrays (FPGAs) present a feasible alternative since they may be used to expedite the intricate computations required for real-time anomaly identification. Because of their efficiency and capacity for parallel processing, FPGAs are especially well-suited to handle the high-volume, high-velocity data that characterizes Internet of Things networks.

For IoT gateways and edge devices, this study suggests an effective anomaly-based intrusion detection system. More precisely, a GPU is used during the training phase to create neural network models, which are subsequently put to the test using a hierarchical decision-making methodology. The chosen model is contained in the Programmable Logic (PL) path of the SoC Zynq platform, which speeds up the inference phase. The experiment also targets the recently released IoT-23 dataset .

II. LITERATURE

Anomaly-based detection systems have been utilized to detect previously unidentified attacks. Prominent methods to safeguard IoT networks against security flaws have been examined in a number of survey studies, including data mining, machine learning, and deep learning. Based on the generated dataset, a literature has presented a federated self-learning anomaly detection in Internet of Things networks. Similarly, researchers have successfully detected IoT anomalies with 98% accuracy by using machine learning algorithms on a produced dataset.

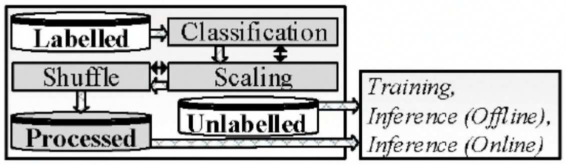


Fig. 1. System overview

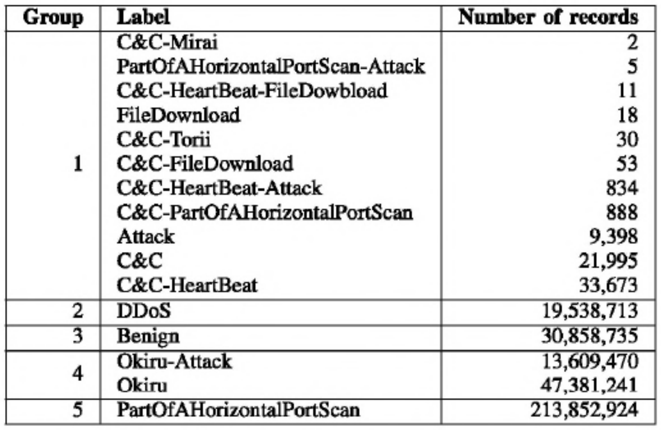
Recently, supervised learning techniques have been evaluated on the IoT-23 dataset. For example, in order to increase accuracy, authors have assessed an ensemble approach that combines Long Short-Term Memory (LSTM) with Deep Neural Networks (DNN). In order to do anomaly prediction, additional study has additionally merged IoT-23 with two other datasets. A uniform characteristic set to identify botnet attacks has been proposed. However, this system's feature extraction procedure is tool-based, which could lead to problems with dependencies and bottlenecks. On the IoT-23 dataset, malware and botnet analysis have been implemented, respectively. Additionally, it has been determined that the optimum method for identifying anomalies is Random Forest (RF).

Many writers have suggested that SoC will be a platform of the future for protecting IoT devices. IoT security monitoring has made use of FPGA-accelerated semantic caching. The work uses studies from the CAESAR competition to compare ASIC and FPGA for IoT security. An SoC platform is used to propose ANN for intrusion detection systems, and a comparison with the developed FPGA-based system is shown.

III. Methodology

Artificial Neural Networks (ANNs) are computational systems that excel at finding complex patterns in data. Like our brains, ANNs learn and improve through experience. The fig-1 utilizes multiple ANN models on a new dataset to achieve this. In a two-stage process, training exposes the models to labeled data, allowing them to select the best architecture and fine-tune internal parameters. Once trained, these models can then analyze new, unseen data and generate corresponding outputs. To tackle complex problems, the system leverages a hierarchical decision-making approach, employing multiple ANNs for improved accuracy, particularly when dealing with unbalanced data. However, this approach comes with trade-offs. Using multiple models can increase processing overhead, leading to slower computation times and higher power consumption.

The IoT-23 dataset is a valuable collection of network traffic data specifically for Internet of Things (IoT) devices. It's a large dataset with over 325 million records, each labeled into one of sixteen categories. Each record contains eighteen features describing the traffic. These features fall into two categories: static and flow-based. Static features come directly from the packet header information, such as IP addresses, ports, and protocols used. Flow-based features are calculated by analyzing packets that are part of the same flow or occur within a specific time window.



Group	Label	Number of records
1	C&C-Mirai	2
	PartOfAHorizontalPortScan-Attack	5
	C&C-HeartBeat-FileDowbload	11
	FileDownload	18
	C&C-Torii	30
	C&C-FileDownload	53
	C&C-HeartBeat-Attack	834
	C&C-PartOfAHorizontalPortScan	888
	Attack	9,398
	C&C	21,995
	C&C-HeartBeat	33,673
2	DDoS	19,538,713
3	Benign	30,858,735
4	Okiru-Attack	13,609,470
	Okiru	47,381,241
5	PartOfAHorizontalPortScan	213,852,924

Fig. 2. labelled flow in the IoT-23 dataset

The labels in the dataset are grouped into five main categories as shown in fig-2. The first group includes attack types with a smaller number of records (less than 35,000 each) and mainly consists of traffic infected with Command and Control (C&C) malware. The second and third groups are dedicated to DDoS attacks and benign traffic, respectively. The last two groups contain records identified as Okiru and PartOfAHorizontalPortScan (PHPS). This comprehensive dataset allows researchers to analyze the details of IoT network traffic, which can be helpful in developing stronger security measures for these devices.

IV. Proposed Framework

The first step in implementing an anomaly detection system on the PYNQ Z2 board is to design the FPGA logic using the Vivado Design Suite. This involves crafting VHDL code to implement the specific anomaly detection algorithms suitable for your research. Crucially, this design includes input and output ports for clock, reset, data, and the anomaly detection flag. Additionally, a constraints file (XDC) is necessary to map these I/O ports to the correct physical pins on the PYNQ Z2 board. Once the design is complete, a comprehensive testbench is developed within Vivado to simulate various data scenarios, including those representing both normal and anomalous conditions. These simulations ensure that the FPGA logic correctly identifies anomalies before deployment.

Since FPGAs cannot directly process binary data files, a method for loading data into the system must be devised. One approach involves pre-loading data as initial memory content within Vivado, which is suitable for simulation purposes. However, for real-world scenarios, the PYNQ Z2's onboard ARM processor offers flexibility. Python code can be written to read the binary file from the SD card and transmit the data to the FPGA, likely through an efficient AXI-based interface for high-speed data transfer. After the FPGA design is synthesized and implemented in Vivado, a bitstream file is generated. This bitstream configures the FPGA and is transferred to the PYNQ Z2 board.

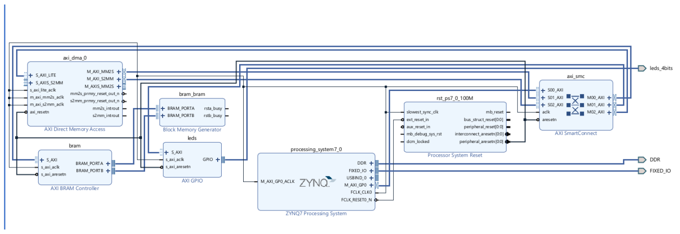
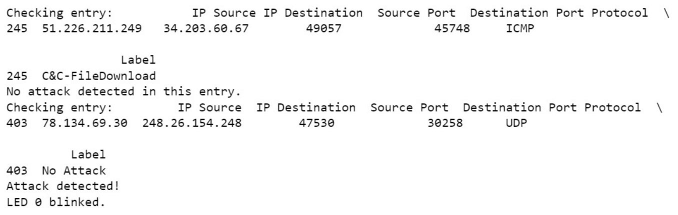


Fig. 3. Block Design



```
Checking entry:         IP Source IP Destination  Source Port  Destination Port Protocol  \
245  51.226.211.249   34.203.60.67       49057            45748     ICMP

                Label
245  C&C-FileDownload
No attack detected in this entry.
Checking entry:       IP Source  IP Destination  Source Port  Destination Port Protocol  \
403  78.134.69.30  248.26.154.248       47530            30258      UDP

         Label
403  No Attack
Attack detected!
LED 0 blinked.
```

Fig. 4. Result

With the bitstream loaded, the PYNQ environment provides Python-based control of the FPGA. A Python script or Jupyter Notebook is used to manage the interaction, sending data from the binary file to the FPGA and monitoring its anomaly detection output. This stage involves real-time testing to evaluate the system's accuracy and responsiveness. The output flag (potentially an LED) is observed to determine the system's success in identifying anomalies. If anomalies are missed or false positives occur, debugging tools within Vivado can be used to analyze waveforms and refine the VHDL or Python code as needed.

Below is the Python code used for detecting anomalies within the dataset and controlling an LED based on the detection:

```
    # Function to randomly check for attacks
    within the dataset
def random_check_for_attacks
(df, num_checks=10):
    for _ in range(num_checks):
        # Randomly select an
        entry from the dataset
        random_entry = df.sample()
        print(f"Checking entry:
        {random_entry}")

        # Check if the randomly selected
        entry is labeled as 'Attack'
        if 'Attack' in random_entry
        ['Label'].values[0]:
            print("Attack detected!")
            blink_led(0)  # Blink the
            first LED (LED 0)
            return  # Stop checking
            after detecting an attack
        else:
            print("No attack detected
            in this entry.")

    # If no attack is found after all checks
    blink_led(1, times=2, interval=1)
    # Blink the second LED (LED 1) twice
```

V. RESULT

The anomaly detection system was successfully implemented on the PYNQ Z2 board, demonstrating its ability to process binary data and identify anomalies in real-time. Anomalies were defined based on specific criteria, and upon detection, a visual indicator (blinking LED) was triggered on the board. This confirmed the correct operation and rapid response of the FPGA design.

The system consistently detected anomalies within microseconds, highlighting the benefits of hardware acceleration for time-critical security applications within IoT networks. The FPGA's speed and computational efficiency outperformed traditional processing approaches. Moreover, the integration of the PYNQ framework and Jupyter Notebooks provided a flexible development environment for rapid prototyping and testing. These results demonstrate the potential of FPGAs in enhancing the security and responsiveness of IoT networks through real-time anomaly detection.

REFERENCES

[1] Xilinx Inc. (2021). *Vivado Design Suite User Guide*. Available at: urlhttps://www.xilinx.com/support/documentation/sw$_{m}anuals/xilinx2021_{1}/ug973-vivado-release-notes-install-license.pdf$

[2] Stratosphere Laboratory. (2021). *A labeled dataset with malicious and benign IoT network traffic*. Available at: urlhttps://www.stratosphereips.org/datasets-IoT23

[3] Chen, Y., & Luo, Y. (2019). FPGA-based implementation for real-time anomaly detection using machine learning. *Sensors*, 19(14), 3075. DOI:10.3390/s19143075

[4] Wang, B., & Srinivasan, R. (2018). A review of anomaly detection systems in cloud networks and IoT using machine learning techniques. *IEEE Access*, 6, 2169-3536. DOI:10.1109/ACCESS.2018.2870642

[5] IEEE Standards Association. (2020). *IEEE Standard for Floating-Point Arithmetic*. IEEE Std 754-2019 (Revision of IEEE 754-2008), 1-84. DOI:10.1109/IEEESTD.2019.8766229

[6] Kumar, S., & Spafford, E. H. (1995). A pattern matching model for misuse intrusion detection. *17th National Computer Security Conference*, 11-21.

[7] Python Software Foundation. Python Language Reference, version 3.8. Available at: urlhttps://www.python.org

[8] Jupyter Project. (2021). *Jupyter Notebook User Documentation*. Available at: urlhttps://jupyter-notebook.readthedocs.io/en/stable/

[9] Harris, M., & Harris, S. (2016). *Digital Design and Computer Architecture: ARM Edition*. Elsevier Science.

[10] Popovici, E., & Mencer, O. (2020). FPGA in IoT: Opportunities and challenges for machine learning in resource-constrained environments. *IEEE Internet of Things Journal*, 7(5), 3922-3934. DOI:10.1109/JIOT.2020.2973758

VI. ROLES

Data preparation and model training –Aparna,Harshitha
Hardware Implementation –Chaturya,Ajay
Software Implementation – Prudvi