2025-05

Transformer-Based Intrusion Detection for Securing Medical Applications in 5G IoMT Networks

Provi, Navid Al Faiyaz

IUB

Transformer-Based Intrusion Detection for Securing Medical Applications in 5G IoMT Networks

May 2025

Prepared by:
Navid Al Faiyaz Provi
ID: 2020001
Mohammed Zabir Chowdhury
ID: 1910975
Sabrin Morshed
ID: 2021792

Department of Computer Science and Engineering Independent University, Bangladesh

Supervised by
Md Zahangir Alam, PhD
Assistant Professor
Department of Computer Science and Engineering Independent University, Bangladesh

Attestation

Plagiarism constitutes a strict violation of our university's policies that we fully understand. All our work maintains complete originality. The project contains copyrighted materials and other models that we have properly referenced using international citation protocols.

Author Name:

. . . .

Signature:

. . . .

Author Name:

. . . .

Signature:

. . . .

Author Name:

. . . .

Signature:

. . . .

Evaluation Committee

Supervisor

Name _____ Signature _____

Internal Examiner 1

Name _____ _____

Internal Examiner 2

Name _____ _____

External Examiner

Name _____ _____

Acknowledgement

We, the authors, enjoyed the effective guidance from our loving mentor and the supervisor - Md Zahangir Alam Sir who is an Assistant Professor of Independent University, Bangladesh. His motivational assistance transformed our zeal from obstacles to the moving opportunities of professional development. By his patient mentorship, he offered meaningful feedback that broadened our research into transformer-based intrusion detection paying a heavy price of inquiry. The manner in which he believed in our abilities enabled us to keep our heads in the game and got us motivated as this endeavor proved to be very fruitful.

We have an unpayable debt of love and support to our parents who have earned us this plethora of gratitude inside our hearts. Their sacrifices and the continuous prayers were a source of power that enabled us to survive sleepless nights and perform unbearable obstacles. Our dream supporters gave us the much-needed encouragement that kept our determination stable because they helped us to appreciate the importance of continuous effort coupled with dedication. We extend humbled gratitude to Almighty Allah who bestowed us wisdom and strength and many other blessings that directed our whole effort. Our developments benefitted from divine help that instated deep appreciation.

This thesis is an example of joint commitment to perseverance and patience and knowledge seeking behavior followed by our academic group. Our souls got engrossed in the pursuit of studying 5G IoMT network security and we were pursuing it because of our common interest in saving healthcare systems. The instructive directions of our fellows and mentors were inspirational thanks to the different perspectives that gave this work and this accomplishment substance that established them as a compelling step forward to a digital security platform.

Abstract

The research proposes a transformer-based anomaly detection system that enhances the 5G IoMT system security in that it guards healthcare applications against patient data leakages. WUSTL-HDRL-2024 dataset simulates different 5G network communications along with attack scenarios, while the proposed model makes use of self-attention to detect Distributed Denial of Service (DDoS) and others such as Man-in-the-Middle (MiTM) and Ransomware and Buffer Overflow. The single-layer transformer obtained an AUC-ROC score of 0.9132 and an F1-score of 0.5019 in the task of DDoS attack detection but showed a medium effect in both tasks of MiTM and Ransomware attack detection due to imbalance among labels. As a way of conducting real-time operations and usage in resource-limited environment, it is possible to conduct analysis of aggregated IP flows through a sliding time window framework. Though, the model shows itself a promising safety solution for IoMT systems while encountering difficulties associated with real world data integration and interpretability challenges with unbalanced datasets.

Contents

LIST OF FIGURES

LIST OF TABLES

1 Introduction

1.1 Background of the project

Integration of IoMT system into healthcare resulted in the tracking of the remote patient data at the same time as offering prompt diagnosis services in critical cases. Healthcare facilities have a very difficult task of coming up with stringent security measures that will ensure data accuracy in tandem with privacy mechanisms. Patient care is affected adversely whenever there is data tampering, hence possibly fatal outcomes in life-threatening situations. The complexity of the data modifications in these systems offers the machine learning algorithms a good potential to implement effective intrusion detection systems. Most of the healthcare intrusion detection systems develop their dataset through the use of the network traffic data, and patients' biometric information among other alternatives. The main advantage of IoMT healthcare systems comes at the promise of delivering early detection capabilities, real-time monitoring capabilities and better patient outcomes. Wearable sensors that enable unending monitoring of vital signs permit remote communication between the medical staff and patients and allow the realization of such healthcare benefits. The preservation of these healthcare systems remains essential since their unprotected state would lead to lethal situations when privacy breaches or false medical diagnoses or delayed care occurs [2]. Traditional security measures are not effective against the current complex and advancing cyberthreats that healthcare institutions experience. Internet vulnerabilities are especially dangerous when an

attacker performs Man-in-the-Middle (MitM) attacks by modifying data packets through transmission according to research [3]. Machine learning offers an attractive answer to various security issues. Machine learning provides superior capabilities in analyzing complex data and spotlights patterns to categorize security threats therefore it successfully detects potential vulnerabilities in the system [4].

1.2 Problem Statement

The development of 5G technology has greatly improved the abilities of Internet of Medical Things (IoMT) systems due to a more rapid and reliable transmission of data between medical devices and healthcare infrastructures. This development however brings along a crucial vulnerability terrain considering the high exposure to complex cyber attacks. IoMT systems provide sensitive medical and biometric information across the networks and as the networks are compromisable, in case they get compromised, it can cause catastrophic effects ranging from data breaches, identity theft, wrong diagnosis, and even life-threatening treatment delays. The old-style security mechanisms cannot be effective to resist the spiral of movement of 5G-empowered attacks like Distributed Denial of Service (DDoS), Man-in-the-Middle (MiTM), ransomware, and buffer overflow attacks. These threats take advantages of the network protocol weaknesses and device-level limitations adding unique threats for anomaly detection systems. In addition, a paucity of detailed, real-world IoMT datasets, impairs quality training and assessment of intelligent Intrusion Detection Systems (IDS). Therefore, there is an immediate need for new innovative machine learning-based solutions that are capable in

identifying and handling anomalous behaviors in real-time with computationally challenged 5G-IoMT environments.

1.3 Outcome of the project

This research project's result is the successful building and testing of a transformer-based anomaly detection system, which is optimized for securing 5G-enabled IoMT applications. The outlined model was developed and validated with the help of the WUSTL-HDRL-2024 dataset, which mimics real-world networks with a variety of cyber-attack types. The system installed a single layer transformer and used a sliding window approach and managed to achieve a high Area Under the Curve (AUC) score of 0.9132 and an F1-score of 0.5019 in detecting DDoS attacks indicating highly developed characteristics when it comes to recognizing high-level, disruptive threats. Though the performance for MiTM and Ransomware detection was moderate because of imbalance in the dataset, the model performs similarly under both balanced and imbalanced fashion. The proposed solution turns out to be quite appropriate for a real-time implementation in healthcare settings where latency and resources are critical aspects. It also provides a scalable and flexible approach that could be tweaked for wider industrial cybersecurity purposes. This project is another contribution in the field of intelligent cybersecurity since it presents the feasibility of transformer models for detecting complex anomalies in time critical systems such as IoMT.

2 Literature Review

2.1 Existing NIDS and their issues

Health monitoring system development ideas have raised several times in recent years and some key approaches are presented sequentially below. The researchers at Fotouhi and colleagues established a general framework to monitor healthcare needs in their work [2]. The framework contains three core components that combine a coordinator element with access points and electronic gateways. The body-worn coordinator operates directly below sensors to acquire gathered information. APs attached to room walls operate with identical standards to those used by the sensors which include ZigBee, 6LoWPAN and BLE. Data from the access points reaches the gateway through which the cloud receives the transmitted information using Internet connectivity. Headed security precautions exist within the framework although it omits extensive engineering approaches and evaluation procedures. The authors did not explore how to determine when cyberattacks succeed.

The application of machine learning in healthcare focuses on vital health risk alerting and false alarm reduction through systems developed by Clifton et al. [10]. Sensor readings meet personalized clinical healthcare findings to detect emergency situations before they occur according to their warning mechanism. The evaluation took place at Oxford University Hospital yet did not resolve security vulnerabilities present in the system design.

Rani et al. created an authorized user-accessible cloud-based healthcare system through their research in [11]. The system makes use of SVM algorithm to predict medical situations in patients and forecast potential health issues. This method leverages machine learning technology to process data instead of applying it for security monitoring of data integrity. The healthcare system framework presented by Chakraborty et al. [12] makes use of blockchain technology to boost security measures. The authors failed to conduct tests which would have established any performance standards for their framework.

Alabdulatif and colleagues designed a system with privacy-enhanced cloud-based functionality to monitor vital signs in real time along with their abnormal pattern forecasts [13]. The system consists of three essential modules as its foundation. The Smart Community Resident functions as the first element of the framework which collects data for storage in Cloud Storage after encryption. The main part which handles encrypted data through the Smart Prediction model employs mathematical methods to detect abnormal patterns and forecast threats. The security method depends on traditional approaches instead of evaluating recent innovations such as machine learning for security breach forecasting.

The work of Tao et al. presented [14] demonstrated an IoT security enhancement by utilizing KATAN Hardware techniques which protected healthcare monitoring systems from security threats. The implementation of a secure data collection system utilizes an optimized confidential cipher algorithm running on Field Programmable Gate Array hardware platforms. The implementation encounters standard hardware-based solution problems according to [15].

The authors of [16] introduced a security framework which employs Random Forest (RF) method for aberrant network traffic identification through examination of the KDD 1999 dataset. Anomalous activity detection using the RF method reached 95% accuracy along with a false-positive identification of 1%. The KDD dataset which was employed in data mining competitions since 1999 remains outdated because it lacks health-specific characteristics [17]. The ML method in our system matches the one used in other systems yet we designed a test environment to obtain data more relevant to clinical healthcare tracking situations. The system employs both network flow metrics together with biometric data to perform anomaly detection.

Studies in [18,19] developed their cybersecurity frameworks using KNN as the core element. The IKPDS application from Rao et al. [18] allowed them to detect different attacks with a 99.6% accurate outcome. The research from Shapoorifard and Shamsinejad [19] focused on lowering false alarms and attained 85.2% accuracy. Both research studies employed an updated version of the KDD dataset which fails to overcome the known limitations that originally appeared in the initial KDD dataset.

Multiple approaches emerge from the study of Internet of Medical Things (IoMT) anomaly detection to secure healthcare systems as documented in literature. Different research projects

work with three datasets including KDD Cup 99, NSL-KDD, and WUSTL-EHMS-2020 that contain network traffic patterns alongside biometric information. WUSTL-EHMS-2020 serves as research data for Wagan et al. (2023) because it contains 16,318 samples where normal data accounts for 87.5% of the total samples and attack information constitutes 12.5% of the dataset. This data distribution allows IDS testing with high reliability. The analysis operates through machine learning (ML) traditional models and progressed to deep learning (DL) systems. K-means clustering and fuzzy C-means have clustered heart disease data according to Singh (2019) while convolutional neural networks (CNNs) and deep neural networks (DNNs) attain high accuracy levels (i.e. 99.23% on KDD99 by Khan, 2019). The proposed Fuzzy C-Means and Bi-LSTM approach developed by Wagan et al. (2023) delivered dual advantages of 92% network detection accuracy along with 89.67% biometric accuracy. Wagan's model generated improved F1-scores (for instance 92.02% for network traffic) through the use of class imbalance techniques according to performance metrics that also include precision, recall and accuracy. The use of SMOTE technique to produce synthetic data might decrease medical trust and the possibility of overfitting remains a challenge because of imbalanced datasets. The particular computing limitations found in IoMT devices make it difficult to execute IDS deployment systems. The performance of single-task models suffers in complex situations and people have not thoroughly researched hyperparameter tuning methods. Research should direct efforts toward model optimization together with explainable AI integration to develop trust and reasoning capabilities for anomaly detection.

The application of deep learning technology enables medical experts to discover anomalies in healthcare time series records which reveal complex hidden patterns in medical documentation. The research team conducted a 2023 review that examined different approaches which were evaluated on actual datasets composed of MIMIC-III, PhysioNet and Sleep-EDF databases which hold comprehensive medical information spanning heart rhythms and brain signals and vital signs. Abundant data available permits medical researchers to detect uncommon heart rhythm abnormality together with sleeping irregularities which would otherwise remain undetected.

Multiple networks demonstrate particular success as effective solutions. The spatial information analysis capability of CNNs complements the story comprehension strength of LSTMs and

Transformers in temporal data processing. VAEs demonstrate effective capability of detecting normal patterns and detecting abnormal events. Using semi-supervised FSSL framework developed by Ying yielded high accuracy of 94.8% when processing PhysioNet ECG data. Researchers at Liu's team established advancements in arrhythmia detection by putting together an LSTM autoencoder system. The research by You demonstrates GANs creating artificial anomalies for building more resilient epilepsy monitoring platforms. The evaluation of these models relies on multiple metrics that include accuracy and precision and recall and F1-score and AUC but precision and recall gain critical importance regarding patient life preservation. The analysis showed that Ying managed 90.4% sensitivity whereas Chen's work demonstrated opposed techniques performed superior to traditional approaches. The achieved progress encounters ongoing substantial difficulties. The training process becomes incorrectly biased because anomalous cases exist in significantly lower numbers than normal cases. Data with many dimensions leads to difficulties during computational operations. The model generalization suffers due to signal noise and the lack of consistent definitions regarding anomalies between patients. Healthcare facilities have delayed implementing these systems because patients worry about their private medical information safety and because complex models function in a way that is not transparent.

3 System Design

3.1 Dataset Creation

Testbed Overview

The WUSTL-HDRL-2024 dataset implements a test environment that duplicates flexible 5G network functionality. The goal of this platform surrounds the completion of gaps which exist in current datasets between multiple 5G network actions and security threats. Six core elements make up the 5G Security Testbed which includes the 5G Core together with the Local Network and Multi-access Edge Computing (MEC) servers User Equipment (UE) as well as an Internal Threat Actor and the Routing Infrastructure. The purpose and operational details of each substance will be described below.

Figure 1: WUSTL-HDRL-2024 Workflow

5G Core

The 5G Core is simulated on a Ubuntu 20.04 machine, using Simu5G simulator, which simulates 5G network operations. Simu5G adds on its functionality to allow for more than one MEC server and external hosts. This component deals with critical network management functions and the flow of data between User Equipment (UE) and MECs.

Features of the 5G Core include:

- **Network simulation**: Simulation of both standalone and non-standalone 5G architectures.
- **Interoperability**: Integration with external hosts for diverse case simulations.

Local Network

The **Local Network** incorporates **Stateful IP/ICMP Translation (SIIT)** to bridge IPv6 and IPv4 communications. This functionality ensures compatibility between devices using different Internet protocols, addressing key challenges within the simulated 5G network.

Key components:

- **IPv6/IPv4 bridging**: Facilitates smooth communication between legacy and modern devices.
- **Dynamic environment**: Simulates real-world 5G network interactions.

MEC Servers

The **MEC servers** are strategically placed to perform edge computing tasks. These servers are responsible for monitoring network traffic, hosting critical data, and running intrusion detection systems (IDS).

Functions of MEC Servers:

- **Data processing**: Real-time analysis and storage of traffic data.

- **Security monitoring**: Intrusion detection for network threats.

User Equipment (UE)

The **User Equipment (UE)** includes devices with varying operating systems that connect to the 5G network either directly or via the Local Network. These UEs represent diverse end-user scenarios to simulate realistic network conditions.

Key features:

- **Device diversity**: Simulates a variety of user interactions.
- **End-to-end connectivity**: Ensures seamless communication with the network.

Insider Attacker

An **Insider Attacker** machine is configured to simulate various types of attacks within the network. This component mimics potential malicious activity to evaluate the testbed's defense mechanisms.

Types of simulated attacks:

- **Man-in-the-Middle (MiTM)**: Interception and alteration of communications.
- **DDoS**: Flooding MEC servers with excessive requests to disrupt services.
- **Ransomware**: Encrypting files to demand payment for access.
- **Buffer Overflow**: Exploiting software vulnerabilities to execute unauthorized code.

Routing System

The **Routing System** plays a critical role in managing data flow within the testbed. A central router connects all components, ensuring efficient communication and data management.

Functions:

- **Traffic routing**: Directs data between nodes in the network.
- **Dataset generation**: Facilitates the creation of diverse and unpredictable datasets.

Dataset Collection and Features

The dataset includes detailed records of network interactions and attack instances captured using the **Audit Record Generation and Utilization System (ARGUS)**. ARGUS is an open-source network monitoring tool that provides comprehensive metrics for analysis.

Dataset Characteristics

- **Size**: 81.7 MB
- **Normal samples**: 132,000 (91%)
- **Attack samples**: 13,000 (9%)
- **Total samples**: 145,000

Attack Types

The dataset covers four primary types of cybersecurity breaches:

1. **Man-in-the-Middle (MiTM)**: Compromises data confidentiality and integrity by altering or intercepting communication.
2. **DDoS**: Disrupts services by overwhelming MEC servers with excessive requests.
3. **Ransomware**: Encrypts critical files and demands payment for decryption.
4. **Buffer Overflow**: Exploits software vulnerabilities to execute unauthorized code.

Testbed Workflow

The workflow of the WUSTL-HDRL-2024 testbed involves the following steps:

1. Data generation begins at the **5G Core**, which simulates network interactions using Simu5G.
2. The **Local Network** processes the data, bridging communication between IPv4 and IPv6 devices.

3. Data is transmitted to **MEC servers**, where it is analyzed and monitored for security threats.
4. **User Equipment** interacts with the network, generating realistic traffic patterns.
5. Simulated attacks are introduced by the **Insider Attacker** to test the network's resilience.
6. The **Routing System** oversees the flow of data and ensures its proper logging for analysis.

Data Flow

Captured data is stored in a CSV format, featuring a mix of:

- **Network flow metrics**: Bandwidth, latency, packet loss, etc.
- **Host metrics**: CPU usage, memory utilization, etc.
- **Attack patterns**: Detailed logs of each type of simulated security breach.

Applications

The WUSTL-HDRL-2024 dataset is a critical resource for:

- **Intrusion Detection Systems (IDS)**: Training and evaluating IDS models.
- **5G Security Research**: Understanding and mitigating threats in medical applications.
- **Machine Learning (ML)**: Developing models to detect and respond to network threats in real time.

Types of attacks

Several different cyber threats exist against network security and data protection within the cybersecurity domain. The cyber threat landscape encompasses four type of attacks: Man-in-the-Middle (MitM) attacks and Distributed Denial of Service (DDoS) attacks as well as ransomware attacks and buffer overflow attacks.

Man-in-the-Middle (MitM) Attacks

When cyber criminals conduct Man-in-the-Middle (MitM) attacks they secretly intercept and alter communications that two parties presume to be private. The cybercriminal impersonates each party during an attack to obtain secret data such as login credentials, financial information and private correspondences. These security breaches harm the reliability and privacy of data since hackers could misuse the stolen information to launch unauthorized activities or perform further breaches. The absence of protected transport methods alongside people who become victims of deceptive online approaches lets MitM attackers continue their operations.

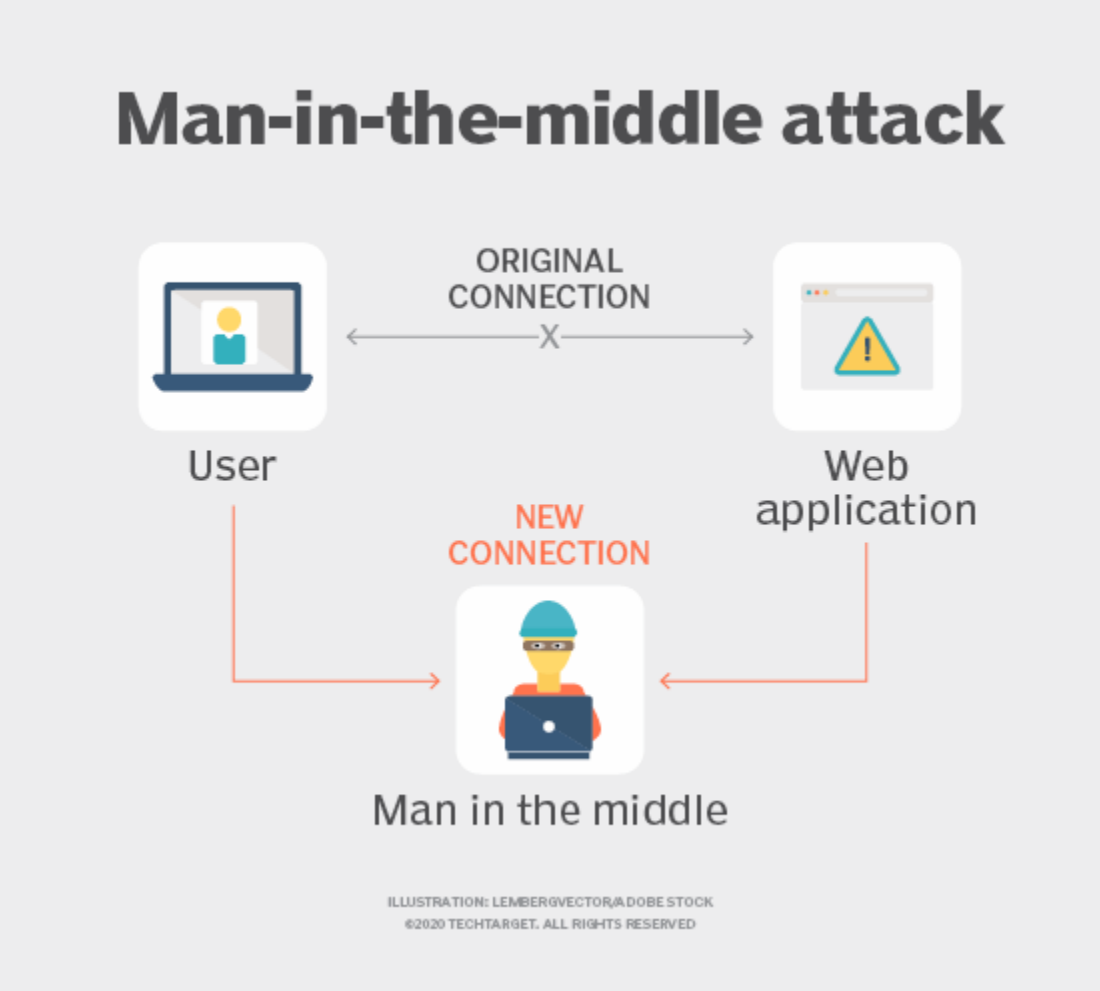


Figure 2: MiTM Attack Scenario

Distributed Denial of Service (DDoS) Attacks

Various devices team up for Distributed Denial of Service (DDoS) Attacks which function through sending abundant internet traffic to render targets inaccessible to their genuine users as displayed in Figure 1.2. Several compromised devices from worldwide locations enable DDoS attackers to deliver their damaging traffic through multiple access points while normal DoS attacks originate from a single machine. Attackers attempt to empty the target's bandwidth and processing capacity and memory stores which results in total system outages. Organizations face disastrous consequences from DDoS attacks through monetary losses together with damage to their reputation which also weakens user confidence in their services.

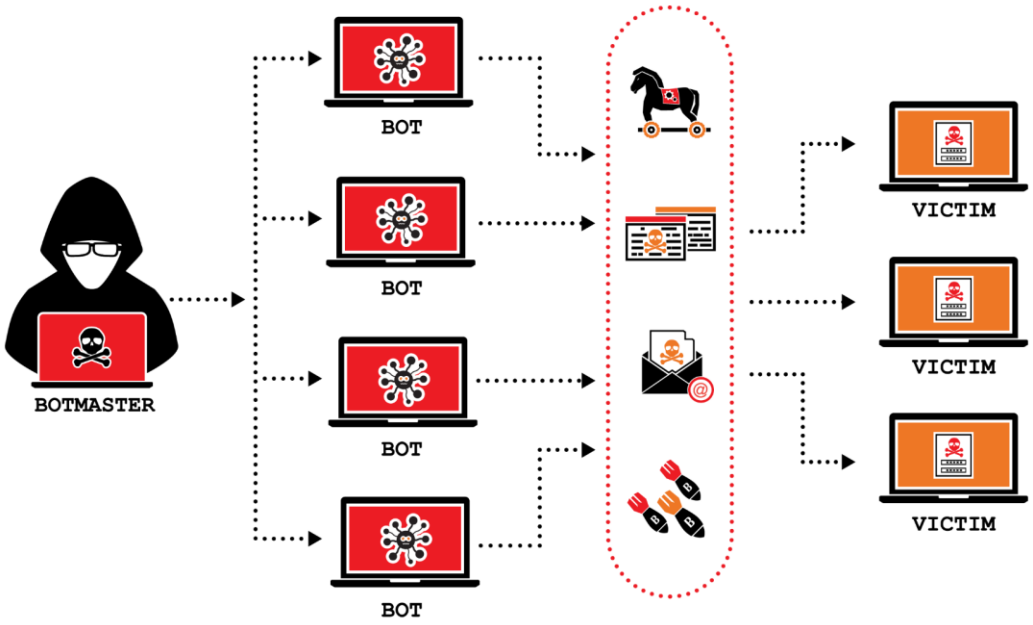


Figure 3: DDoS Attack Scenario

Ransomware Attacks

Computer systems get either restricted by ransomware malware or have their data encrypting while attackers demand cryptocurrency payment for regaining access. Attackers start their operations by breaching systems through phony emails and corrupt downloads together with

software vulnerabilities. Soon after activation this malware begins its encryption process against files before presenting a ransom requirement which usually includes a time constraint for payment pressure. The payment process does not guarantee data retrieval while at the same time potentially creating conditions for additional hacker attacks. The disruptive effect of these incidents results in complete system and data lockdowns which leads to operational shutdowns and major financial losses for organizations.

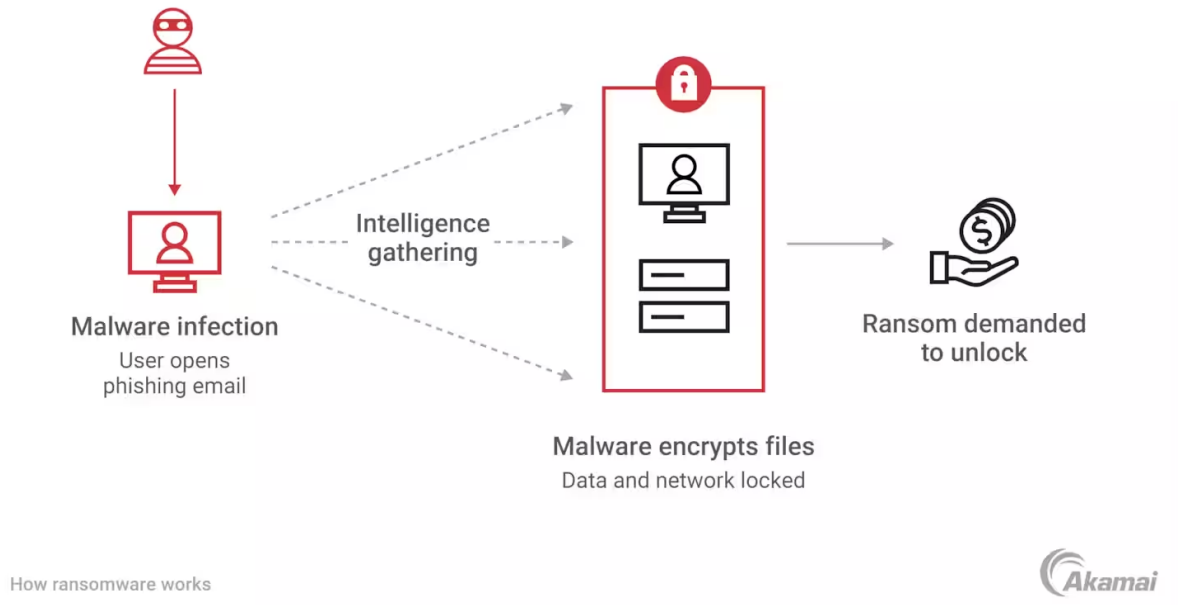


Figure 4: Ransomware Attack Scenario

Buffer Overflow Attacks

The input of data exceeding the memory buffer limit during a program run results in Buffer Overflow Attacks as shown in Figure 1.4. The overflow damage program functions and may cause system crashes that let unauthorized code run. The vulnerabilities in program codes become entry points for cybercriminals who use specific input sequences to execute unauthorized commands while using program privileges. The outcome of such situations includes unauthorized system entry together with data modification and potentially leads to malware infiltration. A buffer overflow attack poses high risk as it will enable an unauthorized access that will enable attackers to enhance their system privileges hence access deeper network places.

Knowledge about such different threats is needed to develop resilient cybersecurity defences. Physical protection employs encryption systems and reinforced routes of communications to prevent MitM attacks. Network security from DDoS attacks requires high resistance at the network activities and adequate filtering mechanisms. Prevention of ransomware covers three elements, which include: data backup scheduling, phishing training of the users and regular updates for system. Protection from buffer overflows necessitates special emphasis to code development coupled with full input validation and also use of security features, that is, address space layout randomization (ASLR) and data execution prevention (DEP).

3.2 Dataset Info

WUSTL-HDRL-2024 is a rich dataset prepared for facilitating a cybersecurity research with special attention to the medical aspects in 5G networks. This dataset has been developed by Washington University in St. Louis, and it provides the much-needed data regarding various 5G network interactions and the related security threats.

Testbed Architecture

The dataset is based on a carefully built 5G Security Testbed that is segmented into six vital components.

1. **5G Core**: Applied to Ubuntu 20.04 system, the 5G Core uses Simu5G, an open source simulator for 5G New Radio (NR) and LTE/LTE-A networks that use OMNeT++ framework. Simu5G enables the simulation of standalone as well as dual connectivity deployments, with heterogeneous gNBs and device-to-device communications.

2. **Local Network**: This segment includes Stateful IP/ICMP Translation (SIIT) in order to enable communication across the IPv6 and IPv4 networks, maintaining compatibility in diverse network protocols within the simulated 5G environment.
3. **Multi-access Edge Computing (MEC) Servers**: Located at the edge of the network, MEC servers play a very crucial role in performing tasks closer to the end-users, thereby decreasing the latency and improving the real time application performance. They also analyze network traffic and host data for intrusion detection systems, which plays a crucial role in maintaining security in the networks.
4. **User Equipment (UE)**: UEs with different operating systems representing end-user devices connect to the 5G network directly or through Local Network simulating the variety of user scenarios and devices' interaction.
5. **Insider Attacker**: Devoted machine with setup to launch several cyber-attacks, which was testing the defence mechanisms of the network and adding the diversity to the set of the attacks patterns in the dataset.
6. **Routing System**: Central router controls data flow in the testbed whereby data management is efficient, and a diverse and unpredictable set of data for sound security analysis is generated.

Simu5G Simulator

Simu5G is relied on very much by the testbed which builds an expert- level replication of 5G network operational practices. The platform supports FDD and TDD operation via heterogeneous gNB (macro/micro/pico/etc) that supports D2D communication and 2G, 3G, LTE & 5G NR base station dual connectivity. The Simu5G platform complies with 3GPP specification and cooperates with INET to allow researchers to come up with generic TCP/IP networks simultaneously with 5G NR layer-2 interfaces.

Security Breaches Simulated

The dataset covers four types of security breach, each minutely simulated to reflect actual cyber threats in the real world:

Man-in-the-Middle (MiTM) Attacks: Being related to the interception or modification of data exchanged between UEs and MECs, these attacks violate the data confidentiality and integrity.
Distributed Denial of Service (DDoS) Attacks: Described by flooding the MEC servers with too many requests thus causing crash of services and denying legitimate access.
Ransomware: These attacks being the encryption of the critical files accompanied by the extortion pleadings to gain access back, pose great threats to the data availability and integrity.
Buffer Overflow Attacks: Exploiting software weaknesses in order to make unauthorized code run, which may cause a system crash, or unwanted access.

Data Collection and Format

Network activities and attacks details were stored using the Audit Record Generation and the Utilization System (ARGUS) which is an advanced program used for the observation of the network traffic. The obtained data is ordered in the CSV format to provide an organized overview of five-generation security threats. The dataset has network flow records, host performance metrics, and attack behavior information, including origin tags for accurate detection use.

Dataset Statistics

Total Size: Approximately 81.7 MB
Number of Samples:

- **Normal Samples**: 132,000 (91%)
- **Attack Samples**: 13,000 (9%)
- **Total Samples**: 145,000

Access and Usage

The WUSTL-HDRL-2024 dataset is available for download, providing researchers with a valuable resource for developing and testing intrusion detection systems and other cybersecurity measures tailored to 5G networks. The dataset's comprehensive nature, encompassing various

Table 1: Abbreviation for the terms

attack types and normal network interactions, makes it an essential tool for advancing cybersecurity research in medical applications within 5G environments.

Abbreviation	Full Form
WUSTL	Washington University in St. Louis
HDRL	Hybrid Deep Reinforcement Learning
5G	Fifth Generation (wireless network technology)
MEC	Multi-access Edge Computing
UE	User Equipment
SIIT	Stateful IP/ICMP Translation
MiTM	Man-in-the-Middle
DDoS	Distributed Denial of Service
CSV	Comma-Separated Values
ARGUS	Audit Record Generation and Utilization System
TDD	Time Division Duplexing
FDD	Frequency Division Duplexing
gNB	Next Generation Node B (5G base station)
LTE	Long-Term Evolution
NR	New Radio (5G standard radio interface)
INET	Internet Network Simulation (framework within OMNeT++)
OMNeT++	Objective Modular Network Testbed in C++ (discrete event simulator)

3.3 Dataset Features

Table 1 displays the complete list of data attributes used in the training and testing phase. The dataset contained 16,000 points that were separated into two categories where 0 meant regular traffic and 1 indicated malicious traffic. The media access control (MAC) address of the source becomes an identifier for the data which identifies attacker-associated samples using value 1 and all other data receives a value of 0. The dataset excludes any samples that do not involve the gateway or attacker or server MAC addresses.

Table 2: Data Science Features.

Feature	Description	Type

Dir	Traffic flow direction	Flow Metric
Flgs	Indicators of connection status	Data stream measure
SrcAddr	Originating IP address	Data stream measure
DstAddr	Target IP address	Data stream measure
Sport	Originating port	Data stream measure
Dport	Target port	Data stream measure
SrcBytes	Bytes sent from source	Data stream measure
DstBytes	Bytes sent to destination	Data stream measure
SrcLoad	Source data volume in bytes	Data stream measure
DstLoad	Destination data volume in bytes	Data stream measure
SrcGap	Bytes absent from source	Data stream measure
DstGap	Absent bytes from target	Data stream measure
SIntPkt	Time gap between source packets	Data stream measure
DIntPkt	Time gap between destination packets	Data stream measure

SIntPktAct	Active time gap for source packets	Data stream measure
DIntPktAct	Active time gap for destination packets	Data stream measure
SrcJitter	Source variability in connection timing	Data stream measure
DstJitter	Destination variability in connection timing	Data stream measure
sMaxPktSz	Largest packet size sent from source	Data stream measure
dMaxPktSz	Largest packet size sent to destination	Data stream measure
sMinPktSz	Smallest packet size sent from source	Data stream measure
dMinPktSz	Smallest packet size sent to destination	Data stream measure
Dur	Length of connection time	Data stream measure
Trans	Total packet accumulation	Data stream measure
TotPkts	Overall packets sent	Data stream measure
TotBytes	Total data volume sent	Data stream measure

network_load	Network capacity consumption	Data stream measure
Loss	Packets lost or resent	Data stream measure
pLoss	Rate of packet loss	Data stream measure
pSrcLoss	Rate of source packet loss	Data stream measure
pDstLoss	Rate of target packet loss	Data stream measure
Rate	Packets transmitted per second	Data stream measure
SrcMac	Originating MAC address	Data stream measure
DstMac	Target MAC address	Data stream measure
IMEI	Device unique identifier	Device Metric
RTime	Live data collection	Platform performance measure
Packet_num	Packet count in the stream	Data stream measure
scputimes_user	Processor user time	Platform performance measure
scputimes_nice	Processor priority scheduling time	Platform performance measure
scputimes_system	CPU system time	Platform performance measure

scputimes_idle	CPU idle time	Platform performance measure
scputimes_iowait	CPU waiting for I/O	Platform performance measure
scputimes_irq	CPU servicing interrupts	Platform performance measure
scputimes_softirq	CPU servicing software interrupts	Platform performance measure
scputimes_steal	CPU time stolen by virtual machine hypervisor	Platform performance measure
scputimes_guest	CPU time spent on guest virtual machine	Platform performance measure
scputimes_guest_nice	Nice priority time on guest virtual machine	Platform performance measure
scpustats_ctx_switches	Context switches in CPU	Platform performance measure
scpustats_interrupts	Number of interrupts serviced	Platform performance measure
scpustats_soft_interrupts	Number of software interrupts serviced	Platform performance measure
scpustats_syscalls	Number of system calls made	Platform performance measure
svmem_total	Total system memory	Platform performance

		measure
svmem_available	Available system memory	Platform performance measure
svmem_percent	Percentage of memory used	Platform performance measure
svmem_used	Total used system memory	Platform performance measure
svmem_free	Free memory	Platform performance measure
svmem_active	Active memory	Platform performance measure
svmem_inactive	Inactive memory	Platform performance measure
svmem_buffers	Buffer memory	Platform performance measure
svmem_cached	Cached memory	Platform performance measure
svmem_shared	Shared memory	Platform performance measure
svmem_slab	Slab memory	Platform performance measure
ram_usage_warning	RAM usage warning indicator	Platform performance measure
sswap_total	Total swap memory	Platform performance

		measure
sswap_used	Used swap memory	Platform performance measure
sswap_free	Free swap memory	Platform performance measure
sswap_percent	Swap memory usage percentage	Platform performance measure
sswap_sin	Swap memory in operations	Platform performance measure
sswap_sout	Swap memory out operations	Platform performance measure
sdiskusage_total	Total disk space	Platform performance measure
sdiskusage_used	Used disk space	Platform performance measure
sdiskusage_free	Free disk space	Platform performance measure
sdiskusage_percent	Disk usage percentage	Platform performance measure
Boot_Time_with_date	System boot time with date	Platform performance measure
DTime	Duration of capture	Data stream measure

4 Methodology

4.1 The Transformer-Based Model

This section is divided into two parts. The first part introduces the concept of attention, while the second part explains the architecture of the original Transformer model presented in [7].

A. The Attention Mechanism and Self-Attention

An attention function is, by definition, a weighted average of values [13]. In its most general form, an attention function maps a query q, a set of keys K, and their corresponding values V, to an output.

The output of the attention function is computed based on a similarity score between the query and each key. These similarity scores define the weights for the weighted sum of the values:

$$\text{Attention}(q, K, V) = \sum_i \text{Score}(q, k_i) v_i \quad \text{.....(4.1)}$$

In essence, the attention mechanism captures how strongly a query relates to each key in the dataset and scales the associated values accordingly.

In [7], the proposed attention function is called Scaled Dot-Product Attention. This mechanism employs the dot product as the similarity measure between the query and the keys. For this, both the query and keys are represented as vectors of the same dimensionality, dk, while each value is represented as a vector of dimensionality dv.

When multiple queries are packed together into a matrix Q, the Scaled Dot-Product Attention is computed as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V \quad \text{.....(4.2)}$$

Here:

- The query vectors exist in the Q matrix
- Key vectors are contained in K.
- Value vectors are in V.
- The √dk term ensures proper normalization. The key vectors are contained inside the K matrix. The value vector storage matrix is designated as V. The normalization factor √dk operates within attention mechanisms to control dot product results thus preventing gradient reduction during learning processes.

The attention function receives transformed matrices for query and key and value from the linear transformation stage. A transformation of the attention function output occurs after the attention function applies a linear transformation to generate the final result. The neural network layers use weight optimization that runs during training through gradient-based backpropagation. The processing of query and key and value matrices runs h times simultaneously which results in h different attention heads. A linear transformation merges final results from all attention calculations in individual heads. Figure 1a shows the multi-head attention mechanism as seen in Figure 1a of the paper.

B. Self-Attention

The term **self-attention** refers to the application of an attention mechanism to an input sequence $x=(x_1,x_2,\ldots,x_n)$, where each element $x_i \in R_{dz}$ serves simultaneously as the queries, keys, and values [14].

A self-attention mechanism transforms the input sequence into a new representation in which each element is weighted by its relationship to every other element in the sequence. This enables the model to capture sequential dependencies without relying on recurrent architecture.

4.2 System Procedure

Effective AD relies fundamentally on the quality combined with proper structure of input data. Computational processing of raw network traffic remains difficult because IP packet collection methods generate detailed network interaction data. A technique named IP flow exporting serves as part of the proposed methodology to group packets into flows through common features including source and destination IP addresses and ports along with protocol metric within time intervals. A series of aggregated statistical elements depicts each flow according to the description provided in [16]. These statistical measurements include packet frequency and byte totals with time-related characteristics. The data aggregation technique both simplifies the data set yet conserves vital detection patterns for AD making it suited for immediate network applications in industrial platforms. Each sequence in the flow collection represents network events chronologically from the perspective of ending times. The need for proper time-based sequencing allows the detection of network traffic changes that often appear through changes in established orders. The transformer model receives standardized data because normalization transforms each flow to have features with mean values equal to zero and standard deviation equal to one. Normalization plays an important role during this step because it minimizes scaling inconsistencies which allows the model to discover significant patterns more efficiently. The processed data flows group into fixed-size windows with length parameter l_w for use as transformer model inputs which allows the analysis of network traffic sequence dependencies. The self-attention mechanisms of transformer models provided through Natural Language Processing (NLP) origins let these models achieve outstanding results when analyzing sequential data. The research presented in this study found inspiration from Marino et al.'s [12] initial application of transformers to network intrusion detection. The proposed approach applies a transformer design to network AD through a sliding window method for processing sequences of network flows. Each window, denoted as ($W = (x_1 \, x\{l_w\})$), consists of (l_w) flows, where each flow (xi) is a vector of aggregated statistics. The model is designed to reconstruct the input window, with anomalies detected based on reconstruction errors. The transformer architecture comprises an encoder and a decoder. The encoder processes the input window (W), generating

a latent representation (E(W)). The decoder, however, receives a shifted version of the window, ($W_{shifted}$), obtained by removing the last (ls) flows, resulting in a sequence of length (lw - ls). The decoder's task is to reconstruct the original window (W) from this shifted input and the encoder's output, (D(E(W), $W_{shifted}$). The reconstruction quality is evaluated using a Mean Squared Error (MSE) loss function, defined as:

$$Loss(W) = | W - D(E(W), W_{shifted}| \quad \text{.....(4.3)}$$

where |.| denotes the Euclidean norm. This loss is optimized through gradient backpropagation, enabling the model to learn patterns characteristic of normal traffic. During inference, an anomaly score is assigned to each flow (x_k) based on the reconstruction loss of its associated window *($Wx = (x\{k\text{-}l_w\}\ x_k)$)*:

$$Score(x) = | W_x - D(E(W_x), W_{x,shifted}) | \quad \text{.....(4.4)}$$

where ($W_{x,shifted}$) contains flows from (x_k-l_w to $x_{k\text{-}lw\text{-}ls}$). The anomaly scores indicate low quality of reconstruction that indicates departure of traffic from the standard patterns (perhaps due to attack activity). It's sequential modeling properties allow such a technique to detect statistical anomalies in traffic efficiently, in turn, performed by a transformer.

4.3 Anomaly Detection Mechanism

The crux of the proposed AD method lies in the premise that usual network traffic has standardized patterns while attacks transaction depicts deviant statistics. The transformer model learns to be expert in accurate normalization of the benign traffic through the training on which it produces low values of MSE loss. Once attack flow passes into the model it fails to yield proper sequence reconstruction hence giving high loss value to yield high anomaly scores. Transformers play attacks detection best by identifying both DDoS that compromises the magnitude of traffic and MiTM that distorts patterns of interaction without much havoc. By using a sliding window approach the model manages to analyse network flow data in its entire context so as to effectively detect anomalies patterns in time sensitive sequences. The transformer identifies DDoS attacks

due to a surge of flow volume due to the heightened level of reconstruction loss against adopted network traffic. By doing the shifted window technique in the decoder the model becomes more attentive to anomalies since it has to predict future flow patterns from insufficient data, increasing anomaly reconstruction errors.

4.4 Configuration of the Transformer Model

As a result of its special network requirements , the transformer model requires very precise tuning of several hyper parameters which will be used for fitting its mode of operation with the network traffic. These hyperparameters include:

Number of Layers (N_{layers}): The combinatorics of the encoder and decoder blocks will be grouped into structures referred to as layers and the total number of these blocks define Nlayers. A single-layer architecture was suitable enough for WUSTL-HDRL dataset as great performance and computational efficiency was obtained.

Window Length (l_w): Number of flows in every input window, usually between 25 – 100. In the experiments 50 of window length was chosen as it successfully infused temporal dependencies without a clutter of noise.

Shift Length (l_s): The number of flows the decoder must predict, determining the size of the shifted window (l_w - l_s). A shift length of 1 was commonly used, simplifying the reconstruction task while maintaining sensitivity to anomalies.

Number of Attention Heads (h): The number of parallel attention mechanisms in each layer. Given the relatively low dimensionality of flow features compared to NLP inputs, a single attention head was used, as additional heads did not improve AD performance.

Number of Epochs (N_{epochs}): The number of training iterations, set to 25 to ensure convergence without overfitting.

Batch Size (bs): The number of windows processed simultaneously, influencing parallelism and convergence speed. Smaller batch sizes accelerated convergence but increased computational time, while larger batches improved efficiency at the cost of slower learning.

These hyperparameters were optimized through experimentation, with the window length and batch size being particularly critical for balancing detection accuracy and computational efficiency. The choice of a single attention head reflects the unique characteristics of network flow data, which have fewer attributes than NLP tokens, reducing the need for multi-head attention. The shift length of 1 was selected to focus the decoder on predicting the immediate next flow, simplifying the reconstruction task while maintaining robustness to anomalies.

The proposed methodology is designed to be adaptable to various network environments, as the hyperparameters can be tuned to suit specific traffic characteristics. For instance, networks with high variability in traffic patterns may benefit from longer window lengths to capture broader temporal contexts, while low-latency environments may require smaller windows for faster processing. The use of a single-layer transformer minimizes computational overhead, making the model suitable for resource-constrained industrial settings, such as healthcare facilities or manufacturing plants. Experiments on the WUSTL-HDRL dataset demonstrated that window lengths between 25 and 100 were effective, with 50 being optimal for the dataset's mix of attack types. Longer windows (e.g., 100) occasionally improved detection for complex attacks like Ransomware, which involve prolonged traffic patterns, but increased computational costs. The single attention head configuration was validated through ablation studies, which showed no significant improvement in AD performance with multiple heads, likely due to the compact feature set of network flows compared to high-dimensional NLP inputs.

5 Experiments and Analysis of the Results

5.1 Performance Metrics

When evaluating NIDS, especially in industrial settings where cyberattacks are rare but critical, it's important to use metrics that focus on detecting those rare events. Research suggests that metrics like precision and recall are better than accuracy because they highlight how well the system identifies actual attacks without being skewed by the large number of normal events. For example, in a dataset where only 9% of samples are attacks, a system could achieve high accuracy by ignoring attacks entirely, which is useless for security. In dataset WUSTL-HDRL-2024, where normal traffic far outweighs attack traffic, standard metrics like accuracy can be misleading. Current research indicates that metrics concentrating on attacks should be implemented thus Precision-Recall curves demonstrate system performance levels across different threshold values. Industrial cybersecurity operations benefit strongly from this analysis because missing attacks might lead to major consequences. Industrial security settings prioritize two main objectives where they aim to reduce false-reactive events and maintain attack detection capabilities. Cost-sensitive evaluation metrics that consider financial viewpoint and operational consequences of errors appear best suited to customize NIDS implementations. A factory would choose to keep its attacks detected instead of reducing false alarms based on its financial capabilities.

Receiver Operating Characteristic (ROC) Curve

Binary classifier assessment often uses ROC curves to display True Positive Rate versus False Positive Rate at multiple decision threshold levels. TPR is measured as.

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \quad \text{.....(5.1)}$$

The TP term indicates correctly detected attacks while FN stands for undetected attacks. FPR is measured as:

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}} \quad \text{.....(5.2)}$$

The total number of correctly detected attacks stands as TP while the total normal cases correctly identified equals TN. The number of attacks falsely detected as attacks counts as FP. AUC-ROC stands as a main assessment metric for classification effectiveness which reports 1.0 for perfect execution but displays a value of 0.5 when the classifier performs only at random level.

A challenge exists with applying ROC curves to highly skewed datasets because the WUSTL-HDRL-2024 dataset demonstrates such severe skew even though this approach proves effective with balanced datasets. When applied to such scenarios a low FPR rate generates numerous false positives because of the overwhelming volume of normal network traffic thus potentially misleading evaluation of AUC-ROC. The reliability of ROC comparison techniques exists only when datasets maintain similar attack-to-normal ratios yet research shows this limitation occurs across different industrial conditions.

5.2 Metrics for Imbalanced Datasets

Given the limitations of ROC and accuracy, metrics that focus on the minority class (attacks) are more appropriate for evaluating NIDS in imbalanced datasets. These metrics include precision, recall, F1-score, and the PR curve, which are particularly relevant for datasets like WUSTL-HDRL-2024.

Precision and Recall

IT measures the parts of predicted attacks that are really attacks:

$$Precision = \frac{TP}{TP + FP} \quad \text{.....(5.3)}$$

Recall (equivalent to TPR) measures the proportion of actual attacks correctly identified:

$$Recall = \frac{TP}{TP + FN} \quad \text{.....(5.4)}$$

There are essential metrics in industrial cybersecurity which require high levels of recall performance for attack detection in addition to optimal precision for eliminating false alerts. For instance, a study evaluating a NIDS on the WUSTL-EHMS-2020 dataset, a similar IoMT cybersecurity dataset, reported precision of 94.94% and recall of 95.01% using a machine learning model, indicating strong performance on the minority class.

F1-score

The F1-score represents the weighted mean calculation between precision and recall to obtain a unified evaluation measure:

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad \text{.....(5.5)}$$

The F1-score is particularly useful when comparing models, as it penalizes imbalances between precision and recall. In imbalanced datasets, where one metric might be high at the expense of the other, the F1-score offers a more holistic evaluation. It is widely used in NIDS research, especially for datasets with low attack prevalence.

Precision-Recall (PR) Curve

The PR curve plots precision against recall across different classification thresholds, providing a visual representation of the trade-off between these metrics. The Area Under the PR Curve (AUC-PR) is a robust metric for imbalanced datasets, as it focuses on the performance of the minority class. Unlike the ROC curve, the PR curve is less affected by the large number of true negatives, making it more suitable for datasets like WUSTL-HDRL-2024. Research highlights that AUC-PR is particularly effective for fraud detection and cybersecurity applications, where the positive class is rare.

In the extreme case of a random classifier, the PR curve for an imbalanced dataset would be a horizontal line at the positive sample ratio, emphasizing the need for models that significantly outperform this baseline.

5.3 Performance Evaluation

Training of transformer models for effective performance occurred on the WUSTL-HDRL dataset through 25 training epochs using a window length set to (lw = 50). The chosen window length decides what sequence of network traffic events should be used to create each input sample for anomaly detection. The model training evaluation included checking how training losses changed over the course of epochs. The training process showed that loss values decreased progressively during the training period until they reached a stable state due to the slowing rate of reduction. The model demonstrates expected behavior during optimal set-ups because it processes training data errors until its parameters achieve optimal performance. The selected hyperparameters including batch size and transformer layers directly affected both system convergence rates as well as computational performance outputs. The model convergence speed increased with smaller batch sizes since weight updates occurred more frequently thus enabling fast adjustment to new data. A drawback of this approach was decreased computational efficiency because parallelization experienced reduction. The number of transformer layers N_{layers} increased when computational time decreased because more trainable parameters in both blocks enabled better processing of intricate patterns. Experiment results showed that one transformer layer (lw = 50) produced effective training data performance which indicates basic archetypes work well for this dataset to optimize both effectiveness and processing time.

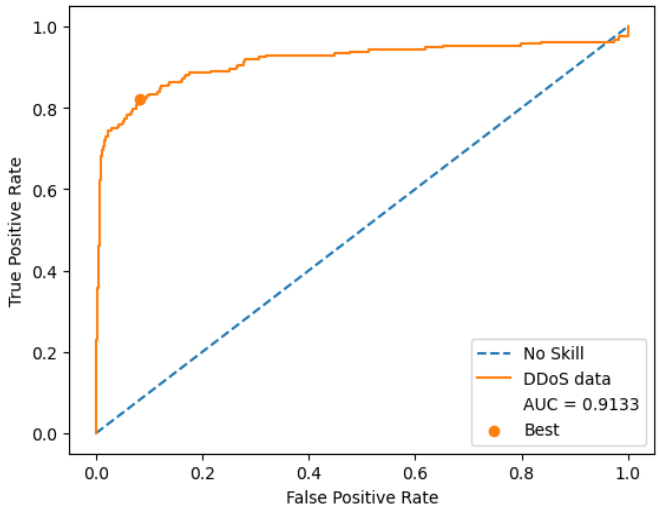


Figure 5: The transformer model on DDoS data ROC curve.

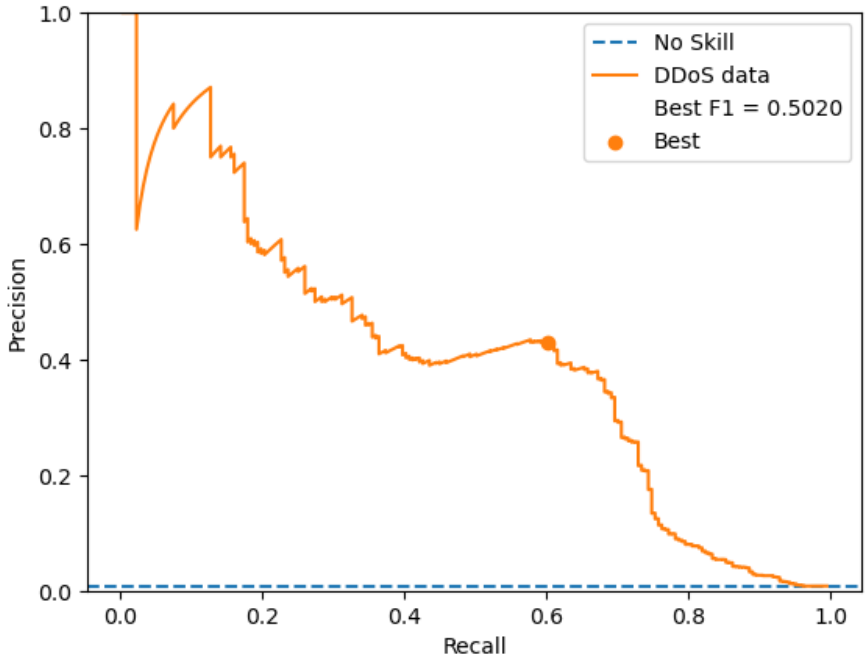


Figure 6: PR curve for MiTM & DDoS Attack Data.

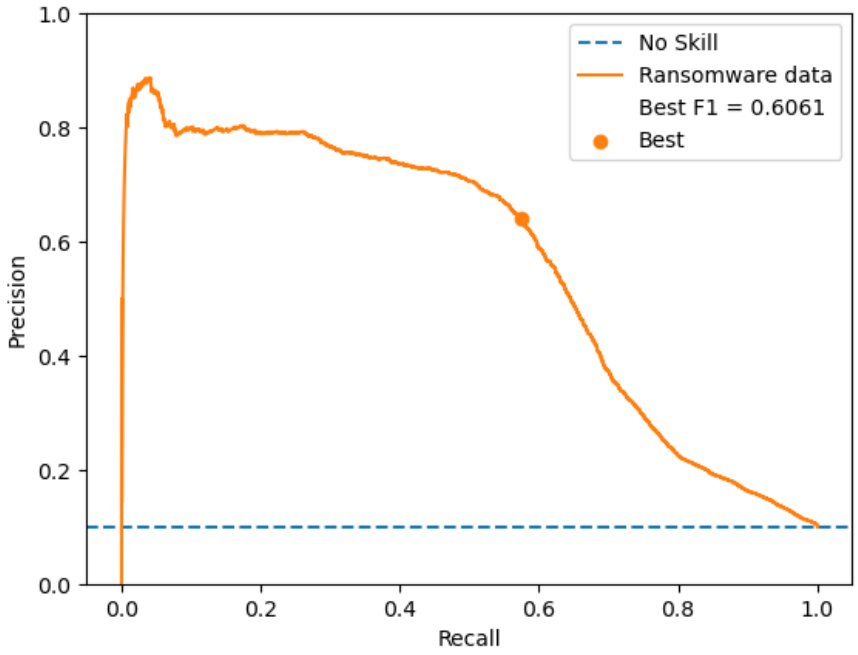


Figure 7: PR curve for Ransomware Attack Data.

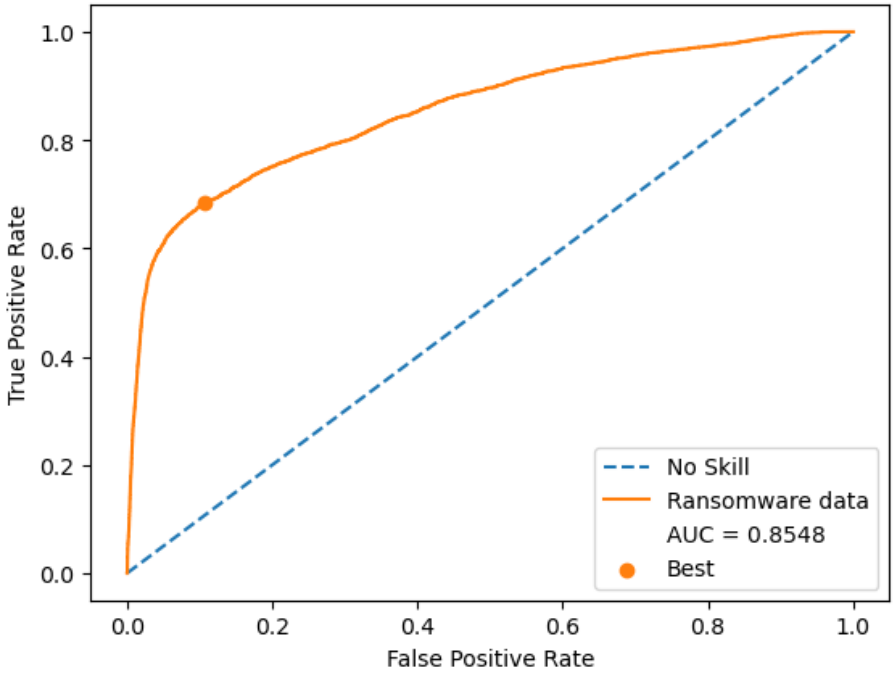


Figure 8: The transformer model on Ransomware data ROC curve.

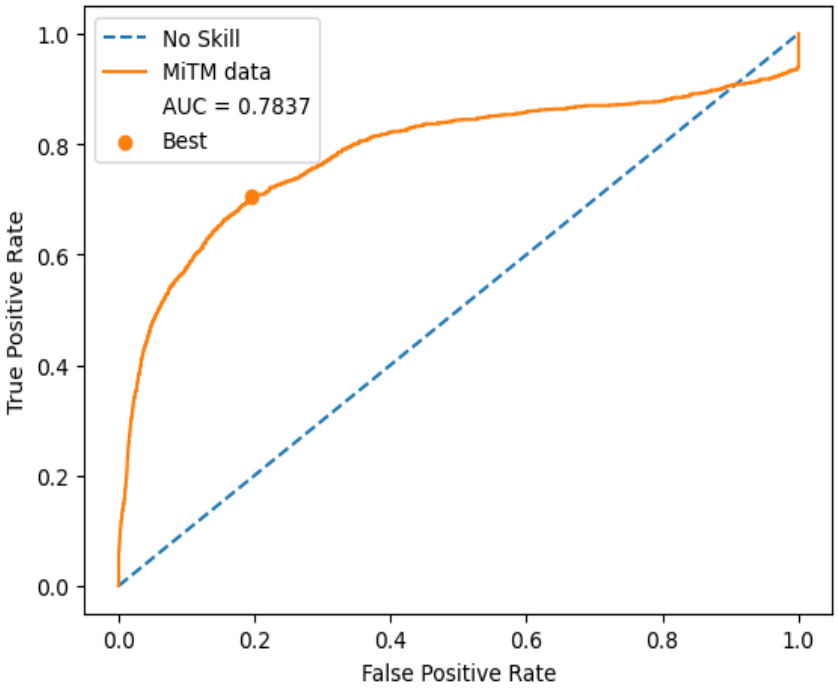


Figure 9: The transformer model on MiTM data ROC curve.

Table 3: Model Scores on the WUSTL-HDRL 2024 Dataset DDoS Attack Data, MiTM (M) And Ransomware (R) Data

Attack Type	**Measure**	**Transformer Score**
DDoS	AUC	**0.9132**
DDoS	F-score	0.5019
MITM (M)	F-score	0.305
MITM (M)	AUC	0.783
Ransomware (R)	F-score	0.606
Ransomware (R)	AUC	**0.854**

The DDoS attack subset of WUSTL-HDRL provides equal data distribution for thorough evaluation using Receiver Operating Characteristic (ROC) and Precision-Recall (PR) curves presented in Figure 4. The ROC curve shows how a model differentiates attack from normal traffic by its ability to detect true positives with various false positive rates at each threshold point. The Transformer model processed DDoS data with an Area Under the ROC Curve (AUC) value of 0.9132 which demonstrates excellent discriminative power. Through its presentation in a precision-recall curve the model demonstrated both high attack detection accuracy and low numbers of false positives. The calculated F1-score was 0.5019 for DDoS data because it represents the harmonic mean of precision and recall as shown in Table I. The transformer model outperformed other attack types during DDoS detection with the highest F1-score and AUC results from the tested evaluations. The detection success can be credited to the DDoS subset's balanced distribution as it minimizes problems that arise from class imbalance. The ROC and PR curves in Figure 4 (a) and (b) visually confirm the model's robust anomaly detection capability, making it a promising solution for industrial environments where DDoS attacks can disrupt critical operations.

In contrast to the balanced DDoS subset, the MiTM, Ransomware, and Buffer Overflow partitions of the WUSTL-HDRL dataset exhibit significant class imbalance, with attack instances constituting a small fraction of the total traffic. This imbalance renders ROC curves less informative, as the large number of true negatives can inflate the AUC, masking poor performance on the minority class. Consequently, only PR curves were plotted for these attack types, as shown in Figure 5, with performance metrics limited to F1-scores and AUC for MiTM and Ransomware, as reported in Table I. For MiTM attacks, the transformer model achieved an F1-score of 0.305 and an AUC of 0.783, indicating moderate performance. The low F1-score reflects the difficulty of detecting MiTM attacks, which are often stealthy and involve subtle deviations from normal traffic. For Ransomware attacks, the model performed better, with an F1-score of 0.606 and an AUC of 0.854, suggesting improved detection capability. The PR curves presented in Figure 5 parts (a) and (b) demonstrate how the model manages the recall-precision relationship during attack detection in imbalanced datasets while treating recall as the main

objective. The extreme partition imbalances demonstrate why F1-score and AUC-PR should be used for NIDS evaluation in industrial cybersecurity deployments. The assessment methods work only with the minority class to guarantee precise evaluation of the model's ability to find rare essential attacks. The transformer model demonstrates better performance in identifying ransomware-monitoring patterns due to its architecture which adapts well to defined ransomware signatures in network traffic.

All measures used for DDoS attack detection demonstrated the transformer model's superiority when it achieved excellent results in highest AUC and F1-score. The model achieved solid F1-scores in its evaluation of unbalanced attack types most notably in Ransomware detection. The model operates efficiently in balanced as well as imbalanced scenarios thus demonstrating versatility that qualifies it as an ideal selection for industrial cyber defense applications. Transformer model achieves success through its architecture because it employs self-attention mechanisms to detect long dependencies in network traffic data. The processing mechanism of transformers allows them to handle complete sequences at once which leads to speeding up parallelization operations and enhancing capabilities to recognize complicated patterns. A single-layer transformer optimized computation time to lw = 50 while maintaining similar performance standards vital for real-time network intrusion detection applications running on resource-limited environments. The transformer model offers major performance benefits for industrial cybersecurity that affect security protocols especially in healthcare sector through its application of the WUSTL-HDRL dataset. The detection rates of DDoS attacks need to be high for minimizing service outages due to their disruptions of essential systems. The calculated AUC value of 0.9132 together with an F1-score value of 0.5019 in detecting DDoS malware demonstrates effective outcome performance. The model operates with limited success indetecting stealthy threats which now commonly appear in industrial networks even though its performance remains insufficient. The balance between how fast predictions form and how efficiently the system operates depends on batch size together with layer quantities as key implementation factors for industrial applications. Security requirements determine whether running smaller batches of training data becomes more important than using larger batches and multiple neural network

layers due to operational needs. Both industrial practitioners and their organisations can achieve effective detection without needing expensive complex systems based on the demonstrated success of the single-layer model in this research.

6 Conclusion and Future Works

6.1 Conclusion

This research shows how machine learning based specifically on transformer models can revolutionize Internet of Medical Things (IoMT) system security within 5G networks. A completely curated WUSTL-HDRL-2024 dataset evolved to build realistic 5G network scenarios and multiple cybersecurity dangers which became the core element to test the proposed anomaly detection techniques. The transformer model applied self-attention mechanisms while detecting Distributed Denial of Service (DDoS) attacks which yielded outstanding results with 0.9132 AUC-ROC and 0.5019 F1-score scores. The detection of Distributed Denial of Service attacks became more efficient through this model because it excels at recognizing long-term dependencies across network traffic data even when they produced sudden traffic bursts. The model demonstrated average success rates when detecting Man-in-the-Middle (MiTM) and Ransomware threats with F1-scores of 0.305 and 0.606 respectively since it performs well in various threat circumstances. The combination of a single-layer transformer with a 50 flows window accomplished the best possible compromise between system efficiency and detection performance thus proving suitable for healthcare facilities and similar resource-limited industrial settings. The method utilized IP flow export along with sequential data analysis to minimize data dimensions until it maintained vital temporal indicators therefore enabling real-time anomaly identification systems. The evaluation utilized precision, recall andPrecision-Recall (PR) curve measurements to give top consideration to the minority class (attacks) because attack data points represented only 9% of the total. The transformer model demonstrates promising capabilities for IoMT system protection against developing cyber threats because it delivers exceptional results in DDoS attack detection and performs well even in imbalanced attack type scenarios. This research contributes to the growing body of evidence supporting the application of advanced machine learning techniques in cybersecurity stickers, reinforcing the critical need for robust cybersecurity measures in healthcare IoMT systems.

6.2 Limitations

The thesis encounters multiple constraints which researchers should take into account despite its positive aspects. The comprehensive nature of WUSTL-HDRL-2024 dataset does not overcome its status as a simulated environment because it fails to deliver a complete representation of actual 5G network complexities. Real-world IoMT systems incorporate various devices and unpredictable network conditions together with human causes which prove difficult to duplicate in testing environments. The attack subspace of MiTM Ransomware and Buffer Overflow contains very few included samples that fail to address common security data skews found in cybersecurity environments thus complicating model learning processes and assessment. Stealthy or rare attacks performed poorly in detection due to the imbalance in the data which reduced F1-scores and indicated that the transformer model detected DDoS attacks more effectively than these concealment methods. The metrics from network flows and devices prove effective but leave out important detection elements found in application-level data as well as user behavior. The single-layer transformer architecture requires more complex attack modeling capability because advanced network attacks can bypass its limited capacity. This limits performance when it comes to the detection of advanced threats. The study is diluted in its effectiveness since it focuses on a specific hyperparameter blend with 50-time window elements as well as a single attention processing unit; thus, it may require further tweaking for different deployments of the IoMT. The chosen evaluation metrics are consistent with the nature of imbalanced datasets but do not take into account the operational costs of false detections and unchecked attacks in medical care systems where resource management, as well as the patient safety, is important.

6.3 Ethical Considerations

Moral standards are vital for the research projects due to their relevance of the healthcare IoT systems. IoMT systems process medical data that comprise of biometric measures and health info therefore leaving patient's to privacy violations as this is the primary ethical risk. When MiTM

attacks are successful in breaking network security it would compromise the sensitive patient information thus causing severe effects such as identity theft given incorrect medical diagnosis. The transformer model needs to achieve a balance between anomaly detection capabilities, as false positive alerts will cause disruption of health services and overloading of the care providers while destroying the patient's confidence. The proper classification of the non-malicious traffic as destructive operations may result in the delay of vital emergency services. The implications of false negative detection in the medical situation where the attacks are not identified are life-threatening since that emphasizes the need for the medical recall thresholds to be high. Medical system implementations with compliance to HIPAA will ensure data confidentiality at the same time data integrity integrity. Implementations of machine learning pose issues with regard to transparency in addition to accountability needs of healthcare systems. The current form of assets utilized by transformer models use sophisticated self-attentive mechanisms generating decision processes which continue being non-transparent thus frustrating health provider and patient trust relationships. The pursuit of ethical excellence requires that, developers should analyze and show every element of every system while being open about the limits and above all prioritizing the safety and the fair treatment of patients, while disallowing possible disparities that might affect particular groups of people.

6.4 Future Works

The future agenda for the present research implies a number of critical changes designed to address the issues currently existing and contribute to the strengthening of the transformer-based anomaly detection system. The first priority includes dataset expansion by acquiring the real-world IoMT network traffic, which also utilizes various possible devices and network scenarios in the context of different attack. Worked up partnerships between healthcare providers will facilitate collection of representative data with ethical and regulatory compliance guarantees of uniformity for anonymized datasets. The transformer architectural design will receive enhancements through model depth trials and multi-head attention adjustments to detect sophisticated and imbalanced attacks which include Man-in-The-Middle and Ransomware. Data

augmentation combined with synthetic attack generation allows the reduction of class imbalance which improves overall model performance. The detection accuracy can be enhanced by adding application-layer data along with contextual patient information into the dataset. The research evaluates cost-sensitive metrics which reflect operational together with financial impact of false negatives and false positives in healthcare environments while modifying the system for realistic needs. Future research will examine explainable AI techniques for transformers to achieve better interpretability in decision-making thus healthcare providers can validate anomaly detection results. These improvements establish connections between research analytics conducted under simulation and practical deployment of transformer-based anomaly detection systems that provide robust and ethical security for IoMT healthcare applications.

Bibliography

[1] B. Marr. "2024 IoT And Smart Device Trends: What You Need To Know For The Future." [Online]. Available: https://www.forbes.com/sites/bernardmarr/2023/10/19/2024-iot-and-smart-device-trends-what-you-need-to-know-for-the-future/?sh=18cac8ae7f34 [Accessed: April 17, 2024].

[2] Grand View Research. "Internet Of Things In Healthcare Market To Reach $169.99 Billion By 2030." [Online]. Available: https://www.grandviewresearch.com/press-release/global- iot-in-healthcare-market [Accessed: April 18, 2024].

[3] IBM Security, "Cost of a Data Breach Report 2023," 2023. [Online]. Available: https://www.ibm.com/reports/data-breach?utm_content=SRCWW&p1=Search&p4=43700077724063991&p5=e&p9=58700008523619412&gclid=CjwKCAjwouexBhAuEiwAtW_Zx92teKfFKXuBhdzKSqhb16G5oHVEmxwZDYv1VTQ9sbrJ2jjnR8Q7ExoCf0IQAvD_BwE&gclsrc=aw.ds

[4] A. Jay. "Number of Internet of Things (IoT) Connected Devices Worldwide 2024: Breakdowns, Growth & Predictions." [Online]. Available:https://financesonline.com/number-of-internet-of-things-connected-devices/ [Accessed: April 18, 2024].

[5] Hady, A. Ghubaish, T. Salman, D. Unal, and R. Jain, "Intrusion Detection System for Healthcare Systems Using Medical and Network Data: A Comparison Study," *IEEE Access*, vol. 8, pp. 106576-106584, 2020.

[6] Ghubaish, Z. Yang, A. Erbad, and R. Jain, "LEMDA: A Novel Feature Engineering Method for Intrusion Detection in IoT Systems," *IEEE Internet of Things Journal*, 2023.

[7] Ghubaish, Z. Yang, and R. Jain, "HDRL-IDS: A Hybrid Deep Reinforcement Learning Intrusion Detection System for Enhancing the Security of Medical Applications in 5G Networks," in *2024 IEEE International Conference on Smart Applications, Communications and Networking (SmartNets)*, Harrisonburg/Washington DC, VA, USA, 2024.

[8] CyberMDX. "2020 Vision: A Review of Major IT & Cyber Security Issues Affecting Healthcare." [Online]. Available:

https://www.healthcareinfosecurity.com/whitepapers/2020-vision-review-major-cybersecurity-issues-affecting-healthcare-w-6017 [Accessed: May 1, 2024].

[9] W. MADDOX. "Why Medical Data is 50 Times More Valuable Than a Credit Card." [Online]. Available: https://www.dmagazine.com/healthcare-business/2019/10/why- medical-data-is-50-times-more-valuable-than-a-credit-card/ [Accessed: May 1, 2024].

[10] United States Naval Academy. "Information Assurance." [Online]. Available: https://www.usna.edu/Users/cs/wcbrown/courses/si110AY13S/lec/l21/lec.html [Accessed: May 1, 2024].

[11] J.-P. A. Yaacoub, M. Noura, H. N. Noura, O. Salman, E. Yaacoub, R. Couturier, and A. Chehaba, "Securing Internet of Medical Things Systems: Limitations, Issues and Recommendations," *Future Generation Computer Systems*, vol. 105, pp. 581-606, 2020.

[12] D. Bhushan and R. Agrawal, "Security Challenges for Designing Wearable and IoT Solutions," in *A Handbook of Internet of Things in Biomedical and Cyber Physical System*, vol. 165, V. Balas, V. Solanki, R. Kumar, and M. Ahad Eds. Intelligent Systems Reference Library: Springer, 2020, pp. 109-138.

[13] F. Pesapane, M. B. Suter, M. Codari, F. Patella, C. Volonté, and F. Sardanelli, "Chapter 52 Regulatory issues for artificial intelligence in radiology," in *Precision Medicine for Investigators, Practitioners and Providers*, J. Faintuch and S. Faintuch Eds.: Academic Press, 2020, pp. 533-543.

[14] J. Sengupta, S. Ruj, and S. D. Bit, "A Comprehensive Survey on Attacks, Security Issues and Blockchain Solutions for IoT and IIoT," *Journal of Network and Computer Applications*, vol. 149, pp. 1-20, 2020.

[15] S. M. Tahsien, H. Karimipour, and P. Spachos, "Machine learning based solutions for security of Internet of Things (IoT): A survey," *Journal of Network and Computer Applications*, vol. 161, pp. 1-18, 2020.

[16] L. Xiao, Y. Li, G. Han, G. Liu, and W. Zhuang, "PHY-Layer Spoofing Detection With Reinforcement Learning in Wireless Networks," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 12, pp. 10037-10047, 2016.

[17] Wikipedia. "Implant (medicine)." [Online]. Available: https://en.wikipedia.org/wiki/Implant_(medicine) [Accessed: May 1, 2024].

[18] J. E. Ferguson and A. D. Redish, "Wireless communication with implanted medical devices using the conductive properties of the body," (in eng), *Expert Rev Med Devices,* vol. 8, no. 4, pp. 427-433, 2011.

[19] Mayo Clinic. "Pacemaker." [Online]. Available: https://www.mayoclinic.org/tests-procedures/pacemaker/about/pac-20384689 [Accessed: May 1, 2024].

[20] A. Phaneuf. "Latest trends in medical monitoring devices and wearable health technology." [Online]. Available: https://www.businessinsider.in/science/news/latest-trends-in- medical-monitoring-devices-and-wearable-health-technology/articleshow/71970305.cms [Accessed: May 1, 2024].

[21] Apple. "Heart health notifications on your Apple Watch." [Online]. Available: https://support.apple.com/en-us/HT208931 [Accessed: May 1, 2024].

[22] Kos, V. Milutinović, and A. Umek, "Challenges in wireless communication for connected sensors and wearable devices used in sport biofeedback applications," *Future Generation Computer Systems,* vol. 92, pp. 582-592, 2019.

[23] H. Jahankhani and J. Ibarra, "Digital Forensic Investigation for the Internet of Medical Things (IoMT)," *Journal of Forensic Legal & Investigative Sciences,* vol. 5, no. 029, 2019.

[24] Medtronic."Pacing Systems - Azure| Medtronic."[Online]. Available: https://europe.medtronic.com/xd-en/healthcare-professionals/products/cardiac-rhythm/pacemakers/azure.html [Accessed: May 1, 2024].

[25] Federal Communications Commission (FCC). "Medical Device Radiocommunications Service (MedRadio)." [Online]. Available: https://www.fcc.gov/medical-device-radiocommunications-service-medradio [Accessed: May 1, 2024].

[26] P. Kasyoka, M. Kimwele, and S. Mbandu Angolo, "Certificateless pairing-free authentication scheme for wireless body area network in healthcare management system," *Journal of Medical Engineering & Technology,* vol. 44, no. 1, pp. 12-19, 2020.

[27] T. Belkhouja, S. Sorour, and M. S. Hefeida, "Role-Based Hierarchical Medical Data Encryption for Implantable Medical Devices," in *2019 IEEE Global Communications Conference (GLOBECOM)*, 9-13 Dec. 2019 2019, pp. 1-6.

[28] Wikipedia. "Chinese remainder theorem." [Online]. Available: https://en.wikipedia.org/wiki/Chinese_remainder_theorem [Accessed: May 1, 2024].

[29] Wikipedia. "Received signal strength indication." [Online]. Available: https://en.wikipedia.org/wiki/Received_signal_strength_indication [Accessed: May 1, 2024].

[30] M. F. Awan, K. Kansanen, S. Perez-Simbor, C. Garcia-Pardo, S. Castelló-Palacios, and N. Cardona, "RSS-Based Secret Key Generation in Wireless In-body Networks," in *2019 13th International Symposium on Medical Information and Communication Technology (ISMICT)*, 2019, pp. 1-6.

[31] Wikipedia. "Cryptographic hash function." [Online]. Available: https://en.wikipedia.org/wiki/Cryptographic_hash_function [Accessed: May 1, 2024].

[32] Wikipedia. "XOR gate." [Online]. Available: https://en.wikipedia.org/wiki/XOR_gate [Accessed: May 1, 2024].

[33] B. A. Alzahrani, A. Irshad, A. Albeshri, and K. Alsubhi, "A Provably Secure and Lightweight Patient-Healthcare Authentication Protocol in Wireless Body Area Networks," *Wireless Personal Communications,* pp. 1-23, 2020.

[34] Z. Xu, C. Xu, W. Liang, J. Xu, and H. Chen, "A Lightweight Mutual Authentication and Key Agreement Scheme for Medical Internet of Things," *IEEE Access,* vol. 7, pp. 53922- 53931, 2019.

[35] Y. Sun and B. Lo, "An Artificial Neural Network Framework for Gait-Based Biometrics,"

[36] *IEEE Journal of Biomedical and Health Informatics,* vol. 23, no. 3, pp. 987-998, 2019.

[37] V. H. Tutari, B. Das, and D. R. Chowdhury, "A Continuous Role-Based Authentication Scheme and Data Transmission Protocol for Implantable Medical Devices," in *2019 Second International Conference on Advanced Computational and Communication Paradigms (ICACCP)*, 25-28 Feb. 2019 2019, pp. 1-6.

[38] S. Maji, U. Banerjee, S. H. Fuller, M. R. Abdelhamid, P. M. Nadeau, R. T. Yazicigil, and

[39] P. Chandrakasan, "A Low-Power Dual-Factor Authentication Unit for Secure Implantable Devices," in *2020 IEEE Custom Integrated Circuits Conference (CICC)*, 2020, pp. 1-4.

[40] Wikipedia. "RSA (cryptosystem)." [Online]. Available: https://en.wikipedia.org/wiki/RSA_(cryptosystem) [Accessed: May 1, 2024].

[41] Wikipedia. "Elliptic-curve cryptography." [Online]. Available: https://en.wikipedia.org/wiki/Elliptic-curve_cryptography [Accessed: May 1, 2024].

[42] V. J. Jariwala and D. C. Jinwala, "Chapter 4 - AdaptableSDA: secure data aggregation framework in wireless body area networks," in *Wearable and Implantable Medical Devices*, vol. 7, N. Dey, A. S. Ashour, S. James Fong, and C. Bhatt Eds.: Academic Press, 2020, pp. 79-114.

[43] T. Bhatia, A. K. Verma, and G. Sharma, "Towards a secure incremental proxy re-encryption for e-healthcare data sharing in mobile cloud computing," *Concurrency and Computation: Practice and Experience,* vol. 32, no. 5, pp. 1-16, 2020.

[44] Wikipedia. "Zettabyte." [Online]. Available: https://en.wikipedia.org/wiki/Zettabyte [Accessed: May 1, 2024].

[45] Wikipedia. "Homomorphic encryption." [Online]. Available: https://en.wikipedia.org/wiki/Homomorphic_encryption [Accessed: May 1, 2024].

[46] G. Kalyani and S. Chaudhari, "An efficient approach for enhancing security in Internet of Things using the optimum authentication key," *International Journal of Computers and Applications,* vol. 42, no. 3, pp. 306-314, 2020.

[47] Wikipedia. "Digital signature." [Online]. Available: https://en.wikipedia.org/wiki/Digital_signature [Accessed: May 1, 2024].

[48] C. Easttom and N. Mei, "Mitigating Implanted Medical Device Cybersecurity Risks," in *2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, 2019, pp. 0145-0148.

[49] Kumari, S. Jangirala, M. Y. Abbasi, V. Kumar, and M. Alam, "ESEAP: ECC based secure and efficient mutual authentication protocol using smart card," *Journal of Information Security and Applications,* vol. 51, pp. 1-12, 2020.

[50] G. Zheng, W. Yang, C. Valli, L. Qiao, R. Shankaran, M. A. Orgun, and S. C. Mukhopadhyay, "Finger-to-Heart (F2H): Authentication for Wireless Implantable Medical Devices," *IEEE Journal of Biomedical and Health Informatics,* vol. 23, no. 4, pp. 1546- 1557, 2019.

[51] G. Zheng, W. Yang, M. Johnstone, R. Shankaran, and C. Valli, "3 - Securing the elderly in cyberspace with fingerprints," in *Assistive Technology for the Elderly*, N. K. Suryadevara and S. C. Mukhopadhyay Eds.: Academic Press, 2020, pp. 59-79.

[52] N. Ibtihel and S. M. Hadj, "Smart ECG Monitoring Through IoT," in *Smart Medical Data Sensing and IoT Systems Design in Healthcare*, C. Chinmay Ed. Hershey, PA, USA: IGI Global, 2020, pp. 224-246.

[53] Ubidots. "Industrial IoT Platform for Energy Management." [Online]. Available: https://ubidots.com [Accessed: May 1, 2024].

[54] W. Youssef, A. O. Zaid, M. S. Mourali, and M. H. Kammoun, "RFID-based System for Secure Logistic Management of Implantable Medical Devices in Tunisian Health Centres," in *2019 IEEE International Smart Cities Conference (ISC2)*, 2019, pp. 83-86.

[55] S. Kulaç, "A New Externally Worn Proxy-Based Protector for Non-Secure Wireless Implantable Medical Devices: Security Jacket," *IEEE Access,* vol. 7, pp. 55358-55366, 2019.

[56] S. Kulaç, "Security Belt for Wireless Implantable Medical Devices," *Journal of Medical Systems,* vol. 41, no. 11, pp. 1-9, 2017.

[57] Mosaif and S. Rakrak, "A Li-Fi based wireless system for surveillance in hospitals," *Biomedical Spectroscopy and Imaging,* vol. 8, pp. 81-92, 2019.

[58] S. Saif, S. Biswas, and S. Chattopadhyay, "Intelligent, Secure Big Health Data Management Using Deep Learning and Blockchain Technology: An Overview," in *Deep Learning Techniques for Biomedical and Health Informatics*, S. Dash, B. R. Acharya, M. Mittal, A. Abraham, and A. Kelemen Eds. Cham: Springer International Publishing, 2020, pp. 187-209.

[59] X. Chen, H. Zhu, D. Geng, W. Liu, R. Yang, and S. Li, "Merging RFID and Blockchain Technologies to Accelerate Big Data Medical Research Based on Physiological Signals," *Journal of Healthcare Engineering,* vol. 2020, pp. 1-17, 2020.

[60] V. Manjula and R. Thalapathi Rajasekaran, "Security Vulnerabilities in Traditional Wireless Sensor Networks by an Intern in IoT, Blockchain Technology for Data Sharing in IoT," in *Principles of Internet of Things (IoT) Ecosystem: Insight Paradigm*, S.-L. Peng,

[61] S. Pal, and L. Huang Eds. Cham: Springer International Publishing, 2020, pp. 579-597.

[62] L. Gupta, T. Salman, M. Zolanvari, A. Erbad, and R. Jain, "Fault and performance management in multi-cloud virtual network services using AI: A tutorial and a case study," *Computer Networks,* vol. 165, 2019, Art no. 106950.

[63] S. C. Sethuraman, V. Vijayakumar, and S. Walczak, "Cyber Attacks on Healthcare Devices Using Unmanned Aerial Vehicles," *Journal of Medical Systems,* vol. 44, no. 1, pp. 1-10, 2019.

[64] Wikipedia. "IP fragmentation attack." [Online]. Available: https://en.wikipedia.org/wiki/IP_fragmentation_attack [Accessed: May 1, 2024].

[65] O. Shwartz, Y. Mathov, M. Bohadana, Y. Elovici, and Y. Oren, "Opening Pandora's Box: Effective Techniques for Reverse Engineering IoT Devices," Cham, 2018: Springer International Publishing, in Smart Card Research and Advanced Applications, pp. 1-21.

[66] Y. Jiang, Y. Shen, and Q. Zhu, "A Lightweight Key Agreement Protocol Based on Chinese Remainder Theorem and ECDH for Smart Homes," (in eng), *Sensors (Basel),* vol. 20, no. 5, pp. 1-13, 2020.

[67] S. Almajali, H. B. Salameh, M. Ayyash, and H. Elgala, "A framework for efficient and secured mobility of IoT devices in mobile edge computing," in *2018 Third International Conference on Fog and Mobile Edge Computing (FMEC)*, 23-26 April 2018 2018, pp. 58- 62.

[68] N. Hassan, S. Gillani, E. Ahmed, I. Yaqoob, and M. Imran, "The Role of Edge Computing in Internet of Things," *IEEE Communications Magazine,* vol. 56, no. 11, pp. 110-115, 2018.

[69] Wikipedia. "Constrained Application Protocol." [Online]. Available: https://en.wikipedia.org/wiki/Constrained_Application_Protocol [Accessed: May 1, 2024].

[70] J. A. Salowey, S. Turner, and C. A. Wood. "TLS 1.3." [Online]. Available: https://www.ietf.org/blog/tls13/ [Accessed: May 1, 2024].

[71] Wikipedia. "ID-based cryptography." [Online]. Available: https://en.wikipedia.org/wiki/ID-based_cryptography [Accessed: May 1, 2024].

[72] Wikipedia. "Certificateless cryptography." [Online]. Available: https://en.wikipedia.org/wiki/Certificateless_cryptography [Accessed: May 1, 2024].

[73] S. S. Gaur, A. Kumar, and B. Mohapatra, "An optimal lightweight cryptographic approach for WSN and its energy consumption analysis," *Int. J. Intell. Eng. Syst,* pp. 59-68.

A Appendices

A.1 Formulation of test.py

```
import torch
torch.cuda.empty_cache()
import pandas as pd
import utils
from pickle import dump
import os

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print(f"Using {device} device")

test_batch_size = 32

dataset = 'ids'
model_name = 'model_1'
model_path = f'models/{model_name}/'
model, data_info = utils.load_model(model_path, device)
assert(data_info['dataset'] == dataset)

ddos_data = pd.read_pickle(f'data/DDoS_{dataset}.pkl')
test_dos = utils.prepare_data(dos_data, dataset, model_path)
assert(data_info['data_columns'] == list(test_dos.columns))

ransom_data = pd.read_pickle(f'data/MiTM_{dataset}.pkl')
test_recon = utils.prepare_data(recon_data, dataset, model_path)
```

```
assert(data_info['data_columns'] == list(test_recon.columns))

mitm_data = pd.read_pickle(f'data/Ransomware_{dataset}.pkl')
test_comm = utils.prepare_data(comm_data, dataset, model_path)
assert(data_info['data_columns'] == list(test_comm.columns))

if __name__ == '__main__':

   dos_loader = utils.get_dataLoader(test_dos, data_info['window_size'], device, batch_size = test_batch_size)
   dos_scores = model.detect(ddos_loader)
   recon_loader = utils.get_dataLoader(test_recon, data_info['window_size'], device, batch_size = test_batch_size)
   recon_scores = model.detect(ransom_loader)
   comm_loader = utils.get_dataLoader(test_comm, data_info['window_size'], device, batch_size = test_batch_size)
   comm_scores = model.detect(mitm_loader)

   output_dir = f"output/{dataset}/output_{model_name}/"
   if not os.path.exists(output_dir):
      os.makedirs(output_dir)
   dump(dos_scores, open(f'{output_dir}DDoS_scores.pkl', 'wb'))
   dump(recon_scores, open(f'{output_dir}MiTM_scores.pkl', 'wb'))
   dump(comm_scores, open(f'{output_dir}Ransomware_scores.pkl', 'wb'))
   dump(data_info, open(f'{output_dir}data_info.pkl', 'wb'))
```

This Python code implements anomaly detection for network intrusion detection using a pre-trained transformer model on the WUSTL-HDRL dataset, focusing on three attack types: DDoS, MiTM, and Ransomware. It leverages the PyTorch library for GPU acceleration and processes data using pandas and custom utilities. The code begins by clearing the GPU cache with

torch.cuda.empty_cache() to free memory, then checks for GPU availability, defaulting to CPU if none is found. It prints the device used (GPU or CPU) for transparency. A test batch size of 32 is set for processing data in batches.The dataset is identified as 'ids', and a model named 'model_1' is loaded from a specified path using a custom utils.load_model function, which returns the model and its metadata (data_info). The code verifies that the model's dataset matches 'ids'. Three datasets—DDoS, MiTM, and Ransomware—are loaded from pickle files using pandas. Each dataset is preprocessed with utils.prepare_data to ensure compatibility with the model, and the code checks that the column names match the model's expected input.In the main execution block, the code creates data loaders for each dataset using utils.get_dataLoader, which organizes the data into batches of size 32, with a specified window size from data_info, and moves them to the selected device. The model's detect method computes anomaly scores for each dataset, producing scores for DDoS, MiTM, and Ransomware.Finally, the code saves the anomaly scores and data_info to pickle files in an output directory (output/ids/output_model_1/). If the directory doesn't exist, it is created using os.makedirs. The scores are stored separately for each attack type, enabling further analysis, such as generating ROC or PR curves to evaluate model performance.This code efficiently processes network traffic data, detects anomalies, and organizes outputs for downstream evaluation, supporting robust cybersecurity research.

A.2 Formulation of train.py

```
import torch
import pandas as pd
import utils

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print(f"Using {device} device")
```

```
dataset = 'ids'
normal_data = pd.read_pickle('./data/train_ids.pkl')
train_data, train_scaler = utils.prepare_data(normal_data, dataset)

d_model = int(len(train_data.columns))

attention = 'single'

N_layers = 6
window_size = 50
batch_size = 256
epochs = 25
dropout = 0
ff_neurons = 512

if __name__ == '__main__':

    train_loader = utils.get_dataLoader(train_data, window_size, device, batch_size = batch_size)

    model = utils.Transformer(d_model, N_layers, attention, window_size, device, dropout,
ff_neurons)
    model.initialize()
    model.train_model(train_loader, epochs=epochs, print_every=1)

    if instance(attention, int):
        attention = f"multi-head {attention}"

    info_dict = {
```

```
    'data_columns' : list(train_data.columns),
    'attention' : attention,
    'N_layers' : N_layers,
    'window_size' : window_size,
    'batch_size' : batch_size,
    'model_info' : str(model),
    'epochs' : epochs,
    'dataset' : dataset,
    'dropout' : dropout,
    'ff_neurons' : ff_neurons
}

utils.save_model(model, info_dict, train_scaler)
```

The code kicks off by importing PyTorch for deep learning, pandas for handling data, and a custom utils module that's probably packed with helper functions. It checks if a GPU is available and sets the device to CUDA if it is, otherwise it falls back to CPU, printing out what it's using so you're not left guessing. Then, it loads a dataset called ids from a pickled file (train_ids.pkl), which is likely a preprocessed pandas DataFrame holding normal network traffic data for training. The utils.prepare_data function processes this data, probably normalizing it and returning the transformed train_data along with a train_scaler to keep track of the scaling for later use. The number of features in the dataset (d_model) is grabbed from the number of columns in train_data, which makes sense since that's what the model will be working with. The script then defines some hyperparameters: it's using a "single" attention mechanism (we'll come back to that), 6 Transformer layers, a 50-timestep window size, a batch size of 256, 25 training epochs, a dropout rate of 0 (no dropout, which is bold!), and 512 neurons in the feed-forward layers. The main block starts by creating a train_loader using utils.get_dataLoader, which likely chops the data into sliding windows of 50 timesteps, batches them, and moves them to the chosen device. Next, it initializes a Transformer model via utils.Transformer, passing in all those

hyperparameters, and calls initialize to set up the weights or whatever else the model needs to get going. The model then trains on the train_loader for 25 epochs, with print_every=1 meaning it'll log progress after every epoch—nice for keeping an eye on things. There's a quirky bit where it checks if attention is an integer, and if so, it reformats it as a string like "multi-head X" for logging, which suggests the code might support multi-head attention in some cases, but here it's just "single" (probably single-head attention). Finally, it bundles up a dictionary called info_dict with all the model's details—data columns, attention type, layer count, window size, batch size, model summary, epochs, dataset name, dropout, and feed-forward neuron count—alongside the trained model and the scaler, and saves everything using utils.save_model. This setup feels like it's built for flexibility: you could swap out the dataset, tweak the attention mechanism, or adjust the model's architecture by changing a few variables. It's handling the IDS scenario specifically, so it's likely looking at sequential network data to learn patterns of normal behavior, but the structure could adapt to other time-series tasks too. The reliance on utils makes it a bit mysterious since we don't see those functions, but it's clearly designed to streamline the preprocessing, training, and saving process for a Transformer-based anomaly detection model.

A.3 Formulation of processing.ipnyb

---

import pandas as pd

data_path = 'raw-data/'

output_path = 'data/'

df0 = pd.read_csv(data_path + 'ids.csv', low_memory=**False**)

mask = list(df0['Attack_categories'])

quarter_mask = len(mask)//4

In [3]:

df0

Out[3]:

In [4]:

columns_to_remove = ['Dir', 'Flgs', 'SrcAddr', 'DstAddr', 'SrcMac', 'DstMac', 'RTime', 'Boot_Time_with_date']

df = df0.drop(columns=columns_to_remove)

In [5]:

df

In [6]:

column_names = df.columns.tolist()

print(column_names)

['Sport', 'Dport', 'SrcBytes', 'DstBytes', 'SrcLoad', 'DstLoad', 'SrcGap', 'DstGap', 'SIntPkt', 'DIntPkt', 'SIntPktAct', 'DIntPktAct', 'SrcJitter', 'DstJitter', 'sMaxPktSz', 'dMaxPktSz', 'sMinPktSz', 'dMinPktSz', 'Dur', 'Trans', 'TotPkts', 'TotBytes', 'network_load', 'Loss', 'pLoss', 'pSrcLoss', 'pDstLoss', 'Rate', 'IMEI', 'Packet_num', 'scputimes_user', 'scputimes_nice', 'scputimes_system', 'scputimes_idle', 'scputimes_iowait', 'scputimes_irq', 'scputimes_softirq', 'scputimes_steal', 'scputimes_guest', 'scputimes_guest_nice', 'scpustats_ctx_switches', 'scpustats_interrupts', 'scpustats_soft_interrupts', 'scpustats_syscalls', 'svmem_total', 'svmem_available', 'svmem_percent', 'svmem_used', 'svmem_free', 'svmem_active', 'svmem_inactive', 'svmem_buffers', 'svmem_cached', 'svmem_shared', 'svmem_slab', 'ram_usage_wraning', 'sswap_total', 'sswap_used', 'sswap_free', 'sswap_percent', 'sswap_sin', 'sswap_sout', 'sdiskusage_total', 'sdiskusage_used', 'sdiskusage_free', 'sdiskusage_percent', 'DTime', 'Attack_categories', 'Label']

In [7]:

MiTM_df = df[:quarter_mask+10000]

MiTM_df['Attack_categories'].value_counts()

Out[7]:

Attack_categories

normal 45083

DDoS 1008

MiTM 147

Ransomware 32

Buffer_overflow 10

Name: count, dtype: int64

In [8]:

```
DDoS_df = df[quarter_mask+10000:2*quarter_mask]
DDoS_df['Attack_categories'].value_counts()
```

Out[8]:

Attack_categories

normal 26069

DDoS 181

MiTM 29

Buffer_overflow 1

Name: count, dtype: int64

In [9]:

```
Ransomware_df = df[2*quarter_mask:3*quarter_mask]
Ransomware_df['Attack_categories'].value_counts()
```

Out[9]:

Attack_categories

normal 32681

DDoS 2815

MiTM 553

Ransomware 200

Buffer_overflow 31

Name: count, dtype: int64

In [10]:

```
train_df = df[3*quarter_mask:]
train_df['Attack_categories'].value_counts()
```

Out[10]:

```
Attack_categories
normal          29051
DDoS             5967
MiTM              943
Ransomware        296
Buffer_overflow    26
Name: count, dtype: int64
In [11]:
train_df.to_pickle(output_path+'train_ids.pkl')
MiTM_df.to_pickle(output_path+'MiTM_ids.pkl')
DDoS_df.to_pickle(output_path+'DDoS_ids.pkl')
Ransomware_df.to_pickle(output_path+'Ransomware_ids.pkl')
```

The code starts by firing up pandas and setting paths for raw data (raw-data/) and output (data/). It loads a CSV file, ids.csv, into a dataframe df0, which is a beast with 145,123 rows and 77 columns, capturing network traffic details like source/destination IPs, ports, bytes, and even system metrics like CPU and disk usage. The low_memory=False flag ensures pandas doesn't choke on the file's size, and a mask is created from the Attack_categories column (values like "normal," "DDoS," "MiTM," etc.), with quarter_mask calculated as a quarter of its length for later slicing. Displaying df0 gives a peek at the data—rows show traffic flows, with columns like SrcBytes (data sent), DstBytes (data received), and Attack_categories labeling each as "normal" or an attack type, paired with a binary Label (0 for normal, 1 for attack). Next, the code trims df0 by dropping columns like Dir, Flgs, SrcAddr, and Boot_Time_with_date that aren't needed for analysis, creating a leaner dataframe df with 69 columns, which is then displayed to confirm the cleanup. The column names are printed to get a sense of what's left—things like Sport (source port), SrcLoad (source data rate), and system metrics like sdiskusage_percent. The real magic happens when the data is split into four chunks for different scenarios: MiTM_df, DDoS_df, Ransomware_df, and train_df. Each slice is based on quarter_mask, roughly dividing the dataset

into four parts, with MiTM_df grabbing the first quarter plus 10,000 rows, DDoS_df taking the next quarter, Ransomware_df the third, and train_df the final chunk. The code checks the Attack_categories distribution in each using value_counts(). For MiTM_df, it's mostly "normal" (45,083) with some "DDoS" (1,008), "MiTM" (147), "Ransomware" (32), and "Buffer_overflow" (10), suggesting a dataset skewed toward normal traffic but with a smattering of man-in-the-middle attacks. DDoS_df shows 26,069 "normal" entries, 181 "DDoS," 29 "MiTM," and just 1 "Buffer_overflow," indicating a focus on distributed denial-of-service attacks but still dominated by normal traffic. Ransomware_df has 32,681 "normal," 2,815 "DDoS," 553 "MiTM," 200 "Ransomware," and 31 "Buffer_overflow," pointing to a ransomware-centric slice with a broader attack mix. Finally, train_df has 29,051 "normal," 5,967 "DDoS," 943 "MiTM," 296 "Ransomware," and 26 "Buffer_overflow," making it a hefty training set with a good spread of attack types. These splits seem designed for specific attack detection tasks—MiTM_df for analyzing man-in-the-middle scenarios, DDoS_df for denial-of-service, Ransomware_df for ransomware, and train_df as a general training set for a machine learning model. The final step saves each dataframe as a pickle file (train_ids.pkl, MiTM_ids.pkl, etc.) in the output directory, preserving them for future modeling or analysis. This setup screams preparation for a supervised learning task, where the Label column will guide a model to distinguish normal from malicious traffic, with each slice tailored to stress-test detection for specific attack types. The code's straightforward but purposeful, transforming a raw dataset into focused subsets for tackling real-world cybersecurity challenges.

A.4 Formulation of visualization.ipnyb

---

import pandas as pd

import numpy as np

import pickle

from sklearn import metrics

import matplotlib.pyplot as plt

import math

In [2]:

dataset = 'ids'

dos_data = pd.read_pickle(f'data/DDoS_**{dataset}**.pkl')

recon_data = pd.read_pickle(f'data/MiTM_**{dataset}**.pkl')

comm_data = pd.read_pickle(f'data/Ransomware_**{dataset}**.pkl')

output = 1

In [3]:

output_name = f'output_model_**{output}**'

output_path = f'output/ids/**{output_name}**/'

dos_in = open(output_path+"DDoS_scores.pkl","rb")

dos_scores = pickle.load(dos_in)

recon_in = open(output_path+"MiTM_scores.pkl","rb")

recon_scores = pickle.load(recon_in)

comm_in = open(output_path+"Ransomware_scores.pkl","rb")

comm_scores = pickle.load(comm_in)

In [4]:

dos_data_scored = dos_data[len(dos_data)-len(dos_scores):].copy()

dos_data_scored['ano_score'] = dos_scores

recon_data_scored = recon_data[len(recon_data)-len(recon_scores):].copy()

recon_data_scored['ano_score'] = recon_scores

comm_data_scored = comm_data[len(comm_data)-len(comm_scores):].copy()

comm_data_scored['ano_score'] = comm_scores

In [5]:

dos_y_true = dos_data_scored['Label']

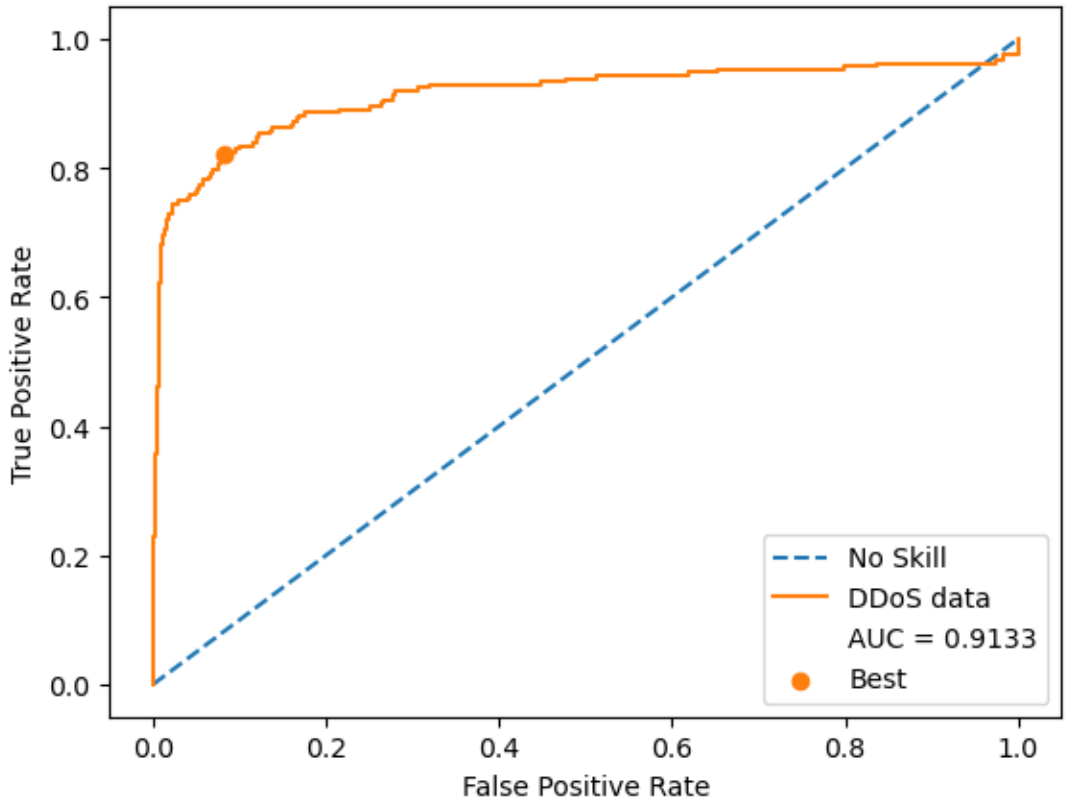
dos_scores = dos_data_scored['ano_score']

```
fpr_dos, tpr_dos, dos_roc_thresholds = metrics.roc_curve(dos_y_true, dos_scores)
dos_roc_auc = metrics.auc(fpr_dos, tpr_dos)
dos_roc_gmeans = np.sqrt(tpr_dos * (1-fpr_dos))
dos_roc_ix = np.argmax(dos_roc_gmeans)
print(f'Best roc-thresh: {dos_roc_thresholds[dos_roc_ix]}, tpr: {tpr_dos[dos_roc_ix]}, fpr: {fpr_dos[dos_roc_ix]}')
print(f'Area under ROC curve: {dos_roc_auc}')

plt.plot([0,1], [0,1], linestyle='--', label='No Skill')
plt.plot(fpr_dos, tpr_dos, label='DDoS data')
plt.plot([], [], ' ', label=f"AUC = {dos_roc_auc:.4f}")
plt.scatter(fpr_dos[dos_roc_ix], tpr_dos[dos_roc_ix], marker='o', color='C1', label='Best')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend()
plt.show()

dos_positive_ratio = dos_y_true.sum()/len(dos_y_true)
dos_precision, dos_recall, dos_pr_thresholds = metrics.precision_recall_curve(dos_y_true, dos_scores)
dos_pr_auc = metrics.auc(fpr_dos, tpr_dos)
eps = 1e-20
dos_pr_f1score = 2/((1/(dos_precision+eps))+(1/(dos_recall+eps)))
dos_pr_ix = np.argmax(dos_pr_f1score)
print(f'Best pr-thresh: {dos_pr_thresholds[dos_pr_ix]}, f1-score: {dos_pr_f1score[dos_pr_ix]}')

plt.axhline(y = dos_positive_ratio, linestyle='--', label='No Skill')
plt.plot(dos_recall[1:-1], dos_precision[1:-1], label='DDoS data', color='C1')
plt.plot([], [], ' ', label=f"Best F1 = {dos_pr_f1score[dos_pr_ix]:.4f}")
```

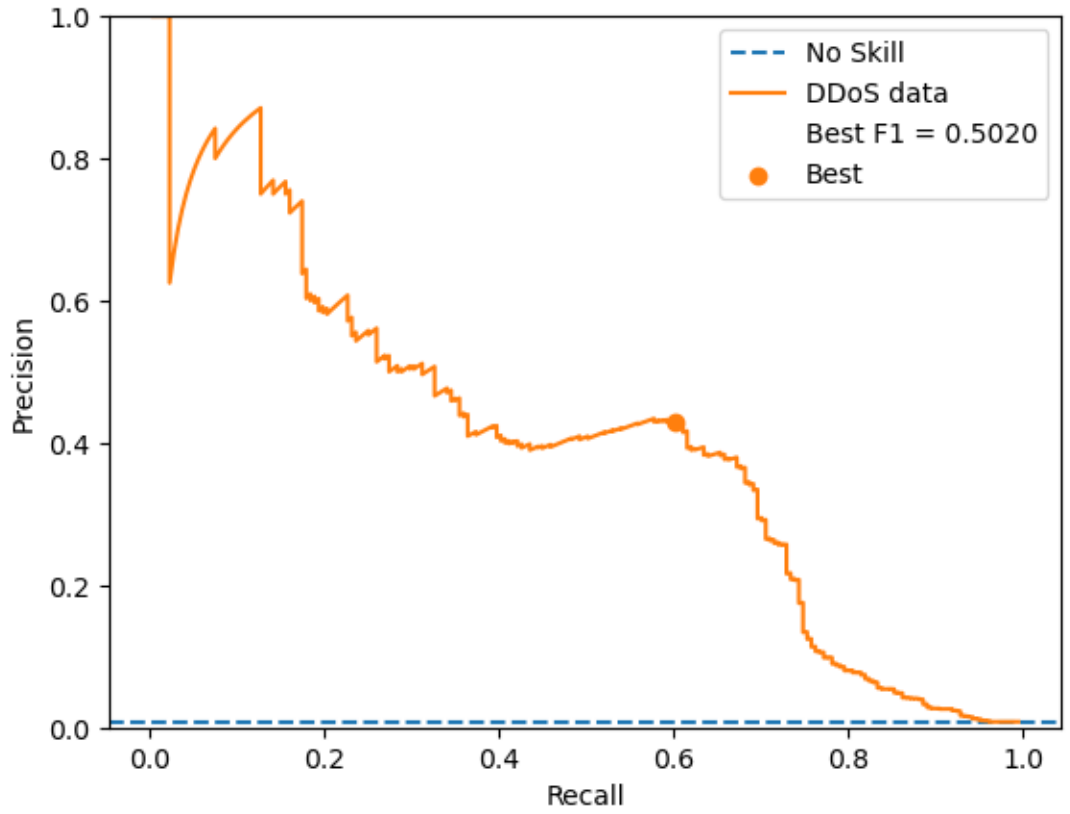
```
plt.scatter(dos_recall[dos_pr_ix], dos_precision[dos_pr_ix], marker='o', color='C1', label='Best')
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.ylim((0,1))
plt.legend()
plt.show()
```

Best roc-thresh: 0.40268608927726746, tpr: 0.8199052132701422, fpr: 0.0828977709454266

Area under ROC curve: 0.9132969170634329



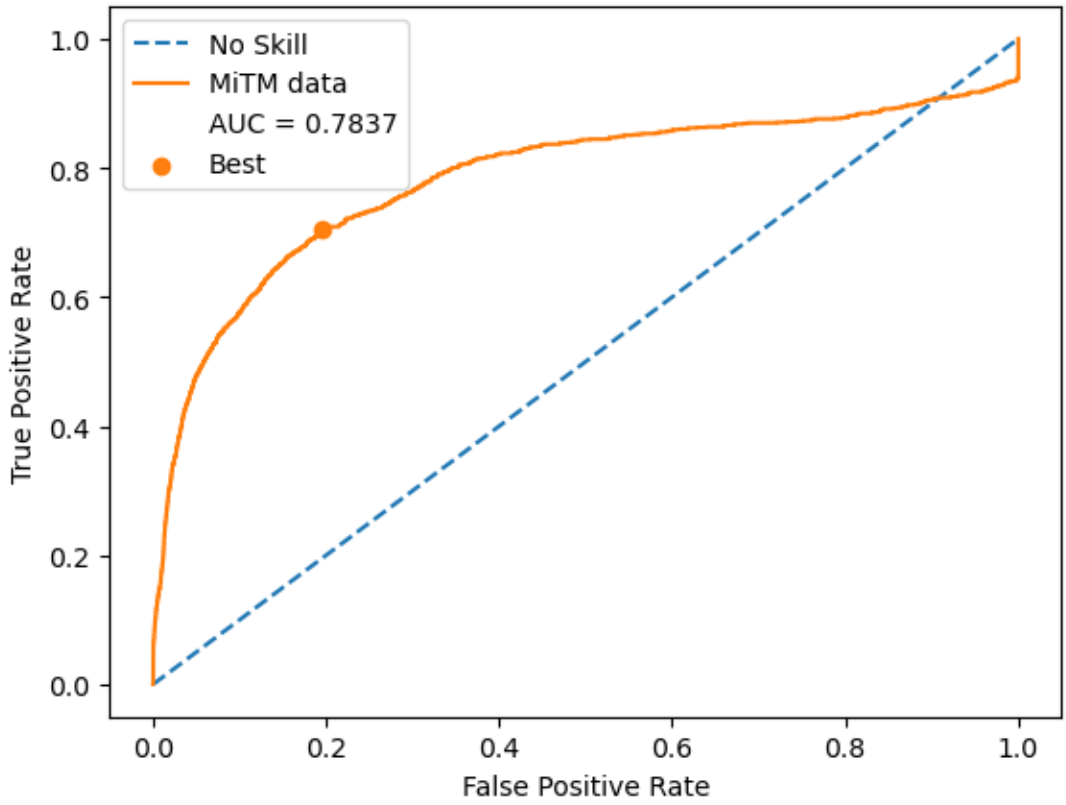
Best pr-thresh: 1.0026147365570068, f1-score: 0.5019762845849802



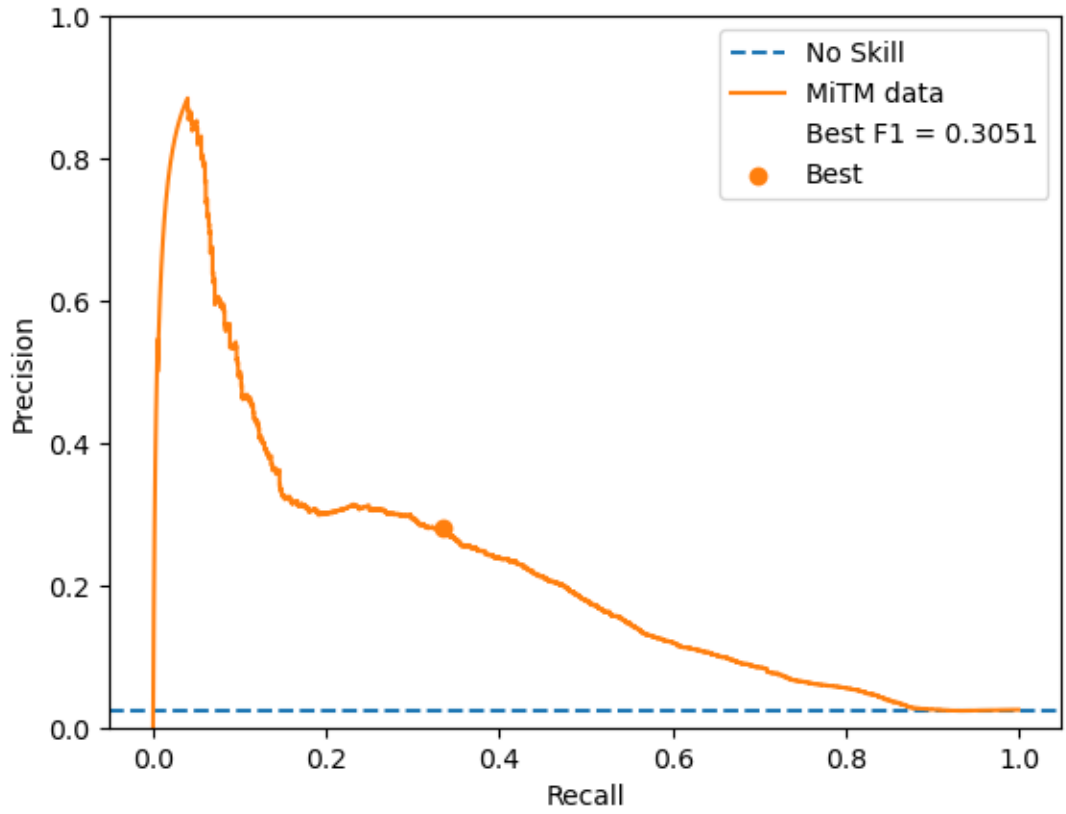
In [6]:

```
recon_y_true = recon_data_scored['Label']
recon_scores = recon_data_scored['ano_score']
fpr_recon, tpr_recon, recon_roc_thresholds = metrics.roc_curve(recon_y_true, recon_scores)
recon_roc_auc = metrics.auc(fpr_recon, tpr_recon)
recon_roc_gmeans = np.sqrt(tpr_recon * (1-fpr_recon))
recon_roc_ix = np.argmax(recon_roc_gmeans)
print(f'Best roc-thresh: {recon_roc_thresholds[recon_roc_ix]}, tpr: {tpr_recon[recon_roc_ix]}, fpr: {fpr_recon[recon_roc_ix]}')
print(f'Area under ROC curve: {recon_roc_auc}')

plt.plot([0,1], [0,1], linestyle='--', label='No Skill')
plt.plot(fpr_recon, tpr_recon, label='MiTM data')
plt.plot([], [], ' ', label=f"AUC = {recon_roc_auc:.4f}")
```

```
plt.scatter(fpr_recon[recon_roc_ix], tpr_recon[recon_roc_ix], marker='o', color='C1', label='Best')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend()
plt.show()

recon_positive_ratio = recon_y_true.sum()/len(recon_y_true)
recon_precision, recon_recall, recon_pr_thresholds = metrics.precision_recall_curve(recon_y_true, recon_scores)
recon_pr_auc = metrics.auc(fpr_recon, tpr_recon)
eps = 1e-20
recon_pr_f1score = 2/((1/(recon_precision+eps))+(1/(recon_recall+eps)))
recon_pr_ix = np.argmax(recon_pr_f1score)
print(f'Best pr-thresh: {recon_pr_thresholds[recon_pr_ix]}, f1-score: {recon_pr_f1score[recon_pr_ix]}')

plt.axhline(y = recon_positive_ratio, linestyle='--', label='No Skill')
plt.plot(recon_recall[1:-1], recon_precision[1:-1], label='MiTM data', color='C1')
plt.plot([], [], ' ', label=f"Best F1 = {recon_pr_f1score[recon_pr_ix]:.4f}")
plt.scatter(recon_recall[recon_pr_ix], recon_precision[recon_pr_ix], marker='o', color='C1', label='Best')
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.ylim((0,1))
plt.legend()
plt.show()
Best roc-thresh: 0.5874118804931641, tpr: 0.7037358818418766, fpr: 0.19496450754214728
Area under ROC curve: 0.7836844162361805
```



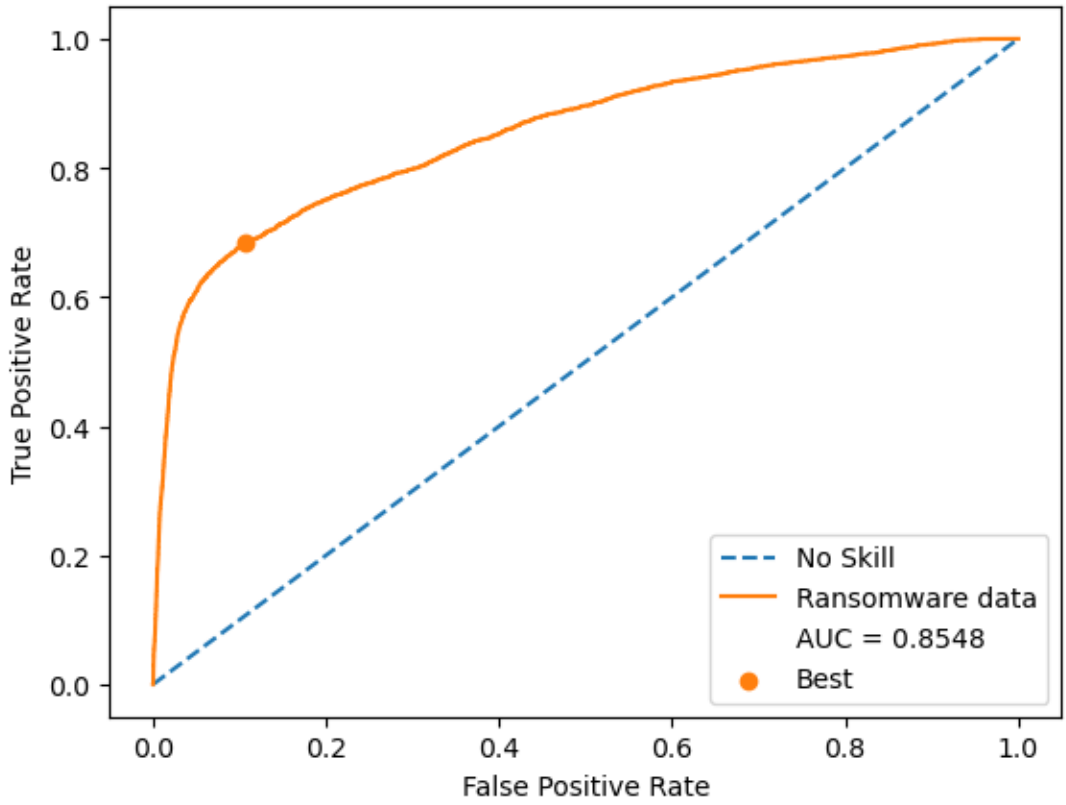
Best pr-thresh: 1.591069221496582, f1-score: 0.30513833992094863



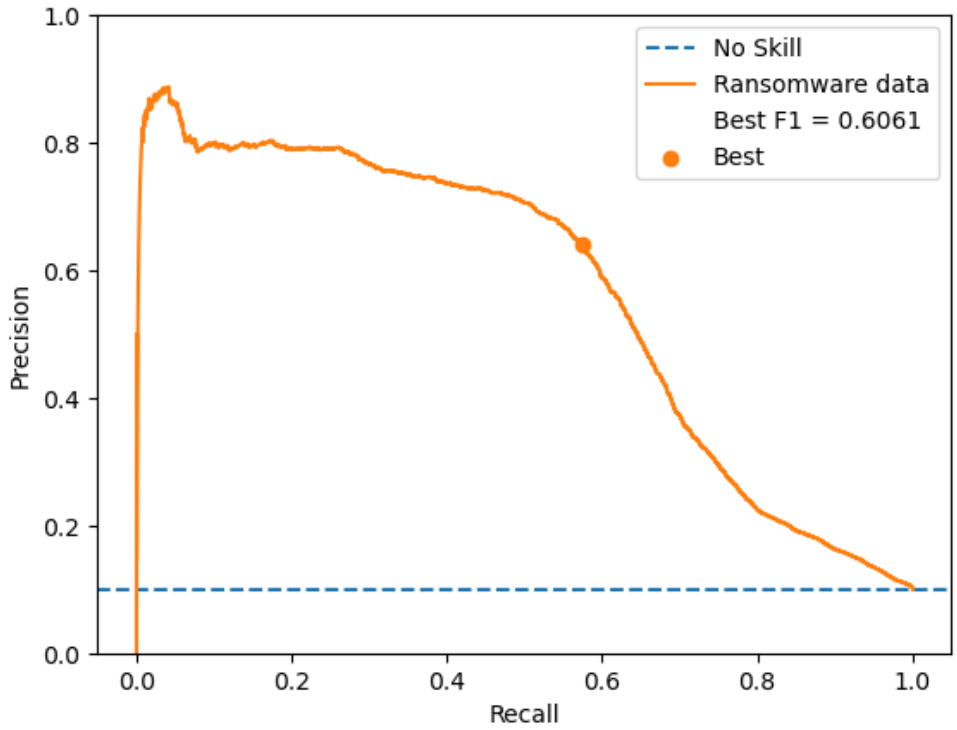
In [7]:

```
comm_y_true = comm_data_scored['Label']
comm_scores = comm_data_scored['ano_score']
fpr_comm, tpr_comm, comm_roc_thresholds = metrics.roc_curve(comm_y_true, comm_scores)
comm_roc_auc = metrics.auc(fpr_comm, tpr_comm)
comm_roc_gmeans = np.sqrt(tpr_comm * (1-fpr_comm))
comm_roc_ix = np.argmax(comm_roc_gmeans)
print(f'Best roc-thresh: {comm_roc_thresholds[comm_roc_ix]}, tpr: {tpr_comm[comm_roc_ix]}, fpr: {fpr_comm[comm_roc_ix]}')
print(f'Area under ROC curve: {comm_roc_auc}')

plt.plot([0,1], [0,1], linestyle='--', label='No Skill')
plt.plot(fpr_comm, tpr_comm, label='Ransomware data')
plt.plot([], [], ' ', label=f"AUC = {comm_roc_auc:.4f}")
```

```
plt.scatter(fpr_comm[comm_roc_ix], tpr_comm[comm_roc_ix], marker='o', color='C1', label='Best')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend()
plt.show()

comm_positive_ratio = comm_y_true.sum()/len(comm_y_true)
comm_precision, comm_recall, comm_pr_thresholds = metrics.precision_recall_curve(comm_y_true, comm_scores)
comm_pr_auc = metrics.auc(fpr_comm, tpr_comm)
eps = 1e-20
comm_pr_f1score = 2/((1/(comm_precision+eps))+(1/(comm_recall+eps)))
comm_pr_ix = np.argmax(comm_pr_f1score)
print(f'Best pr-thresh: {comm_pr_thresholds[comm_pr_ix]}, f1-score: {comm_pr_f1score[comm_pr_ix]}')

plt.axhline(y = comm_positive_ratio, linestyle='--', label='No Skill')
plt.plot(comm_recall[1:-1], comm_precision[1:-1], label='Ransomware data', color='C1')
plt.plot([], [], ' ', label=f"Best F1 = {comm_pr_f1score[comm_pr_ix]:.4f}")
plt.scatter(comm_recall[comm_pr_ix], comm_precision[comm_pr_ix], marker='o', color='C1', label='Best')
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.ylim((0,1))
plt.legend()
plt.show()
Best roc-thresh: 0.36072924733161926, tpr: 0.6829674909697138, fpr: 0.10547928413826918
Area under ROC curve: 0.854847502993974
```



Best pr-thresh: 0.6487600803375244, f1-score: 0.6061493411420205



In [8]:

print(f'Summary of results for model_**{output}**')

print(f'ROC AUC (dos/recon/comm): {metrics.auc(fpr_dos, tpr_dos):.6f}, {metrics.auc(fpr_recon, tpr_recon):.6f}, {metrics.auc(fpr_comm, tpr_comm):.6f}')

print(f'Best F1-score (dos/recon/comm): **{dos_pr_f1score[dos_pr_ix]:.6f}**, **{recon_pr_f1score[recon_pr_ix]:.6f}**, **{comm_pr_f1score[comm_pr_ix]:.6f}**')

Summary of results for model_1

ROC AUC (dos/recon/comm): 0.913297, 0.783684, 0.854848

Best F1-score (dos/recon/comm): 0.501976, 0.305138, 0.606149

The code kicks off by importing essentials like pandas, numpy, sklearn's metrics, matplotlib for plotting, and pickle for loading data. It loads three pre-processed datasets from pickle files—DDoS_ids.pkl, MiTM_ids.pkl, and Ransomware_ids.pkl—each representing network traffic data tailored for a specific attack type, as prepared in the earlier processing.ipynb notebook. These datasets contain features like source/destination ports, bytes, and system metrics, with a Label column (0 for normal, 1 for attack). The script also loads corresponding anomaly scores from pickle files (DDoS_scores.pkl, MiTM_scores.pkl, Ransomware_scores.pkl) generated by a model, stored in a directory structure for model output (output/ids/output_model_1/). For each dataset, the code aligns the scores with the most recent data rows (since scores might cover only a subset), creating new dataframes (dos_data_scored, recon_data_scored, comm_data_scored) with an ano_score column. The goal is clear: evaluate how well the model's anomaly scores distinguish attacks from normal traffic across these three scenarios, using ROC and Precision-Recall curves to measure performance.

Starting with the DDoS scenario, the script extracts true labels (dos_y_true) and anomaly scores (dos_scores) from dos_data_scored. It computes the ROC curve, yielding false positive rates (FPR), true positive rates (TPR), and thresholds, then calculates the Area Under the Curve (AUC) as 0.9133, indicating strong model performance (closer to 1 is better). To find the optimal threshold, it uses the geometric mean of TPR and (1-FPR), identifying a threshold of 0.4027 with TPR of 0.8199 and FPR of 0.0829—pretty good at catching attacks while keeping false alarms low.

A plot visualizes this, showing the ROC curve against a diagonal "no skill" line, with the AUC and best threshold marked. Next, it tackles the Precision-Recall curve, computing precision, recall, and thresholds, and calculates the F1-score (harmonic mean of precision and recall) to find the best balance. The optimal PR threshold is 1.0026, yielding an F1-score of 0.5020, which is decent but suggests precision and recall could be better balanced. Another plot shows the PR curve against a baseline (positive ratio of attacks), highlighting the best F1-score. This scenario, focused on denial-of-service attacks, shows the model is effective but struggles slightly with precision-recall trade-offs, likely due to the imbalanced dataset (few attacks vs. many normal instances).

For MiTM and Ransomware, the script repeats this process with recon_data_scored and comm_data_scored. In the MiTM scenario, the ROC AUC is 0.7837, lower than DDoS, indicating weaker performance in detecting man-in-the-middle attacks. The best ROC threshold is 0.5874 (TPR 0.7037, FPR 0.1950), and the ROC plot confirms the model's moderate ability to separate attacks from normal traffic. The PR curve gives a best threshold of 1.5911 with an F1-score of 0.3051, quite low, suggesting the model struggles with precision or recall for MiTM attacks, possibly due to their stealthy nature or fewer attack instances (147 MiTM vs. 45,083 normal in MiTM_df). The Ransomware scenario fares better, with an ROC AUC of 0.8548, showing good discrimination. The optimal ROC threshold is 0.3607 (TPR 0.6830, FPR 0.1055), and the PR curve yields a threshold of 0.6488 with a solid F1-score of 0.6061, indicating a better balance of precision and recall. Plots for both scenarios mirror the DDoS setup, with ROC and PR curves clearly labeled. The script wraps up by summarizing results: ROC AUCs of 0.9133 (DDoS), 0.7837 (MiTM), and 0.8548 (Ransomware), and F1-scores of 0.5020, 0.3051, and 0.6061, respectively. The DDoS and Ransomware models perform well, but MiTM detection needs work, likely due to data imbalance or attack complexity. This code paints a vivid picture of a model's strengths and weaknesses across distinct cybersecurity threats, setting the stage for further tuning or feature engineering to boost performance, especially for tricky MiTM attacks.

Turnitin Report Summary

Transformer-Based Intrusion Detection for Securi ng Medical Applications in 5G IoMT Networks__MZ A-G01 (Dr. Md. Zahangir Alam.docx

American Education Centre Ltd

Document Details

Submission ID
trn:oid:::28592:93651949

Submission Date
Apr 30, 2025, 2:10 PM GMT+6

Download Date
Apr 30, 2025, 2:15 PM GMT+6

File Name
Transformer-Based Intrusion Detection for Securing Medical Applications in 5G IoMT Networks__MZA-G01 (Dr. Md. Zahangir Alam).docx

File Size
4.1 MB

77 Pages

13,796 Words

89,538 Characters

Page 1 of 85 - Cover Page Submission ID trn:oid:::28592:93651949

turnitin

13% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Filtered from the Report

- Bibliography
- Quoted Text

Match Groups

- **103** Not Cited or Quoted 12%
  Matches with neither in-text citation nor quotation marks
- **4** Missing Quotations 0%
  Matches that are still very similar to source material
- **0** Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- **0** Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

9% Internet sources
7% Publications
7% Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

No suspicious text manipulations found.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.