Mobile Agent based Intrusion Detection for Smart and Connected Medical Devices

Adedayo Odesile

A thesis
submitted in partial fulfillment of the
requirements for the degree of

Masters of Science in Computer Science & Software Engineering

University of Washington

2017

Committee:

Geethapriya Thamilarasu

Munheiro Fukuda

Dong Si

Program Authorized to Offer Degree:
Department of Computing and Software Systems

University of Washington

Abstract

Mobile Agent based Intrusion Detection for Smart and Connected Medical Devices

Adedayo Odesile

Chair of the Supervisory Committee:
Dr Geethapriya Thamilarasu
Department of Software and Computing Systems

The advent of wearable and implantable devices have fostered recent advances in healthcare. Medical devices equipped with wireless connectivity to remote monitoring features are increasingly becoming connected to each other and the internet. Such smart and connected medical devices referred to as the Internet of Medical Things have enabled continuous real-time patient monitoring, increase in diagnostic accuracy, and effective treatment. Inspite of their numerous benefits, these devices open up newer attack surfaces thereby introducing multitude of security and privacy concerns. In this research, we design and develop a mobile agent based intrusion deteciton system to secure the network of connected medical devices. In particular, the proposed system is hierarchical, autonomous, and employs machine and regression algorithms to detect network level intrusions as well as anomalies in sensor data. Our simulation results reflect a relatively high detection accuracy with minimal resource overhead.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

INTRODUCTION

Recent advances in healthcare have led to the development of smart medical devices to facilitate un-interrupted real-time patient diagnosis, and care procurement. This was fostered by the advent of the Internet of Things (IoT); a compound network topology characterized by sensors attached to regular devices (things) for remote telemetry across the local network/internet. In the context of healthcare, the Internet of Things (IoT) driven medical telemetry is commonly known as Internet of Medical Things (IoMT) [28, 14, 11]. IoMT networks comprises of all diagnostic, routing, and data analytic entities relevant towards delivering adequate health-care services to corresponding patients.

Several improvements have been achieved in the quality of medical services by the advent of Internet of Things. Cost of treatment is significantly reduced by providing a means of continuous patient diagnosis and monitoring without requiring their presence in the medical facility. Other significant benefits are facilitation of timely incidence response to dangerous health situations, reduction in injury or death of patients, optimized dispatch and routing of ambulances, and much more. In spite of its abundant benefits, patient privacy and security concerns have impaired its general acceptance. Several factors such as the open-nature of IoMT networks, and computational and communication resource constraint on the diagnostic devices have posed challenges in the implementation of an adequate security system for the network, and most importantly, the patient under diagnosis. A recent study by IBM reveals that medical data has become to the most valuable class of information in the cyber black market[9]. This has resulted into an alarming surge of illegal data breaches in medical institutions, with the sensing and diagnostic layer being a primary launch point due to several security vulnerabilities that characterize this layer [23]. The sensitivity of information transmitted in IoMT networks warrants the need for an efficient security system without impeding the primary functionality of the medical devices. As a result, several research has been done to mitigate security challenges within the domain. They are broadly classified into cryptographic, intrusion detection, and trust management systems [31, 32, 24, 34].

Cryptography is a data protection technique that utilizes a series of reversible complex mathematical functions initialized by a cryptographic key to scramble information. The entire data security lies on the attacker's inability to reproduce the cipher key. Trust based systems define models of rewarding and malicious behavior. They update node's reputation/trust-worthiness based on how cooperative they have been in the network. Nodes whose reputation fall below a certain threshold are black-listed from the network. Intrusion detection systems employ multi-dimensional models of network and device level activities to represent benign and anomalous patterns. Relevant network/device information is continuously extracted and analyzed either centrally or in a distributed fashion. Analysis could be real-time or offline depending on system requirements. The analytical entity uses its designated detection algorithm to determine if collected network information reflects normal or anomalous activities within the network.

Cryptographic solutions are usually computationally intensive and are mostly infeasible for lower powered sensing devices. Although crypto-based security is relatively matured and standardized in comparison with other categories, it only provides external defense, and is usually incapable of recovering from an internal breach. Trust systems possess an extra advantage in the sense that they address internal security breaches. However, they are less versatile than intrusion detection systems as they require construction of behavioral models for each normal and anomalous network/device activity. The space and computational complexity of such models increase exponentially with additional behaviors. Secondly, the models are usually based on exact/absolute computation which limits their ability to handle attacks similar to the ones already modelled. Intrusion detection systems are similar to the their trust-based counterparts in the sense that they also rely on models. However, they employ statistical methods such as pattern recognition or machine learning which are easily extensible without incurring much additional complexity. The use of statistics makes intrusion detection systems probabilistic in nature, as opposed to the absolute trust based models. This increases their ability to handle a wider range of attacks that may not fall within the model.

Several challenges such as resource constraints, scalability, and fault tolerance permeate existing medical network security solutions. The severity of each challenge depends on

the system architecture which is broadly classified into centralized and distributed systems. Scalability and fault-tolerance are commonly found in centralized systems, while the distributed variants suffer from communication over-head, and synchronization challenges. In this research, we examine newer security paradigms that could mitigate the stated challenges without compromising on system security. We design a hierarchical, semi-centralized, autonomous agent security protocol for both network and device level intrusion detection for the most vulnerable layer in the Internet of Medical Things which is the diagnostic/imaging layer. We ran extensive simulations that cover several possible network and environmental scenarios, and got promising results that reflect significant improvements in the security-resource tradeoffs.

1.1 Our Work In Perspective

While several intrusion detection systems have been developed for various computing domains, they still possess inherent limitations in their application to low-powered network devices. Traditional Intrusion Detection Systems (IDS) were either fully centralized with a monolithic entity in form of a server/router carrying out network data analysis, or distributed with the responsibility of intrusion detection being dispersed across designated nodes within the network. In addition to scalability and redundancy problems arising from the centralized systems, the distributed counterpart is usually permeated with high communication overhead as network packets are required to be mirrored to designated nodes for distributed analysis.

Exploration of newer paradigms in distributed detection was pioneered by Balasubramaniyan et al. where they employed the use of autonomous agents. The agents are concrete instances of program code capable of independent execution and state management [4]. Their system was a layered architecture comprising of static agents executing autonomously in each hierarchy of the network. Other researchers proceeded further by equipping the agents with migration capabilities[17, 6]. By making them mobile, the agents were automatically propagated across the network eradicating the need for manual installations on each device. Secondly, redundancies were eliminated as code was not duplicated but was rather migrated at intervals across network nodes.

Although significant progress was made by the use of mobile agents in addressing scalability

and redundancy challenges for traditional computer networks, its application in the medical space is currently unexplored to the best our knowledge. With some enhancements and adaptive measures in place, we extend the use of mobile agents for distributed intrusion detection across low powered medical devices. We implemented a prototype of our system on a network simulation software, and carried out several experimental procedures with corresponding quantitative results. Based on our observations, we were able to achieve a high level of detection accuracy with minimal impact on computational resources and network overhead. For future research, we intend to actualize our system on real hardware to further prove our hypothesis. We foresee challenges such as dealing with agent portability across heterogeneous hardware platforms. Our future research will be mainly comprised of addressing these challenges.

1.2 Our Contributions

To the best of our knowledge, we consider this research to be the first attempt at utilizing mobile agents to facilitate low-footprint intrusion detection within the medical space. Our work consists of the following contributions:

- Introduction of a new security paradigm for protecting smart and connected medical devices.

- Design of a scalable, fault-tolerant, and robust architecture for mobile agent driven intrusion detection.

- Provide an insight into how complex tasks like machine learning and regression could be distributed over low-powered devices.

- Development of a data model to help train Machine Learning (ML) algorithms for detecting the prevalent, DoS and privacy breach attacks within the medical space.

- Development of a polynomial model for detecting anomalies in device data using statistical regression.

- Implementation of a simulation for our system, with established metrics, and rigorous assessments to prove the possible benefits that could be realized from utilizing mobile agents.

For the purpose of providing a fundamental background for this article, we provide a brief description of the relevant terms in the next section.

1.3 Background

The terms fundamental to the comprehension of our research work are described in the next set of subsections.

1.3.1 IoMT/SCH

The Internet of Medical Things/Smart Connected Health is a variant of the IoT networks adapted to the healthcare space. It is a hierarchical network constituted by a diagnostic/sensing, data aggregation, routing, and data service layer. The diagnostic/sensing layer comprises of two broad categories of devices which are the sensing and smart imaging devices. Sensing devices range from a network of low-powered sensors attached/implanted in patient for direct observation of physiological phenomena, known as Wireless Body Area Networks (WBANs), to sensors attached to smart-beds, smart ambulances, etc, relaying data in form of numeric signals. On the other hand, imaging devices are usually higher powered, and utilize various forms of physical media to approximate visual or acoustic images of patient's internal organs. Typical examples are smart MRI and Ultra-Sound scanners, smart X-Ray machines, etc.

The routing layer typically consists of internet gateways, internal and external routers, while the data service layer encompasses the different servers used to either analyze, redact, or persist patient's medical information.

1.3.2 Wireless Body Area Network (WBAN)

A WBAN is a wireless network of sensing devices within the proximity of a single human body, continuously observing and relaying physiological data to appropriate destinations. They are the fundamental medium for real-time patient diagnosis. Earlier wired variants of

BANs, although considered more secure are usually inconvenient for the end users, thereby resulting in their replacement by wireless devices. A typical WBAN follows a star topology with sensors relaying data to a central cluster-head as shown in Figure 1.1. Other variants

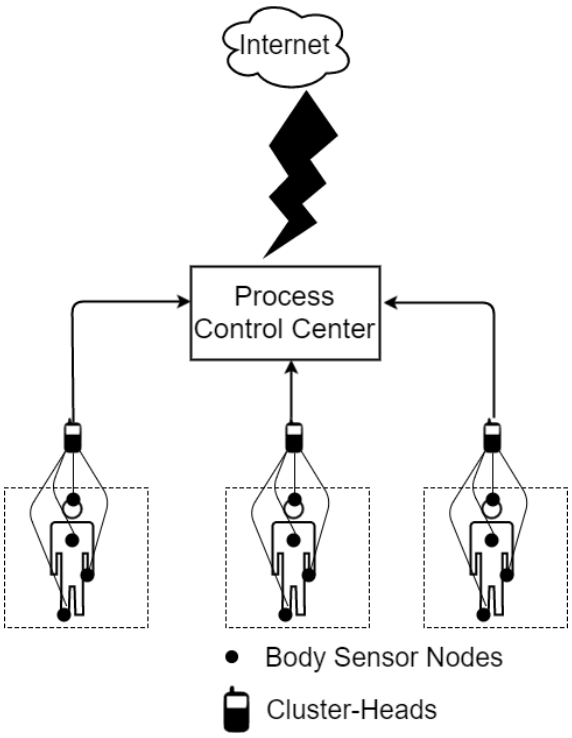

Figure 1.1: Wireless Body Area Network

of WBANs include macro wearables (e.g: MS Band) that have their sensing and aggregating circuits on a single chip. Most WBANs run on the IEEE 802.15/4 zigbee network standard. However, a newer standard (IEEE 802.15/6) was developed for modern devices to handle more persistent high bandwidth, short range transmission, using the human tissue as a modulating medium [20]. Secondly, certain security measures were put in place by creating multiple network profiles which are open (no security), authentication, and both authentication with confidentiality.

1.3.3 Cluster-Head

A cluster-head is a data aggregation and communication control device found in WBANs. They are relatively powerful compared to the actual sensor nodes, and are mostly installed on

as a running app on smart phones or equivalent dedicated devices. They serve as a gateway between the sensors and external networks that run on traditional network protocols. They control communication by using beacon signals to announce availability of wireless bands for transmission, or re-transmissions. While this is required to be manually implemented on classic Zigbee networks, it is an intrinsic feature of the modern WBAN network standard.

1.3.4 Digital Imaging and Communication in Medicine (DICOM)

DICOM is a suite of a TCP/IP based application layer protocol and message exchange format used in medical imaging devices for transmission, handling, and printing of patient's multi-media data. They are used in smart MRI, ultrasound or CT scanners.

1.3.5 Autonomous Agents

Autonomous agents are concrete pieces of program capable of independent execution and state management. They come in two variants which are the static and mobile agents. Unlike their mobile counterparts, static agents are not capable of autonomous migration from one device to another.

In the next chapter, we discuss existing solutions that are relevant to our research.

Chapter 2

LITERATURE REVIEW

We begin our study by examining security challenges that permeate smart medical networks.

2.1 Security Problems in Smart and Connected Medical Devices

A study carried out by Dimitrou et al. discusses privacy breach attacks through passive eavesdropping on a WBAN network [10]. Adversary uses a long range antennae tuned to the radio transmission frequency for the network, and eventually capture radio signals. This could be decoded offline which inadvertently results into exposing patient's private medical information illegally.

Kumar et al. examined attacks that threaten patient's data integrity [19]. They consider an adversary capable of intercepting physiological data from the network channels, and subsequently altering them to induce errors in diagnostic inferences. This kind of attack was considered relatively consequential as wrong diagnosis could lead to physical injuries or death of patient in extreme cases.

The selective forwarding attack was studied by Kambourakis et al[16]. This type of attack was more prevalent in the legacy WBAN IEEE 802.15.4 standard where multi-hop transmission was present. It involves a malicious sensor node dropping packets based on certain criteria as opposed to relaying them. This eventually prevents important information from getting to its destination, which can result into delay in diagnosis and incidence response.

Nasser et al. studied sink-hole attacks in medical sensor networks [26]. The sink-hole is another multi-hop related attack where a malicious node manipulates routing information to attract packets. This open doors to subsequent attacks like the eavesdropping or selective forwarding.

The sybil attack was also investigated by Kambourakis et al [16]. This attack is focused on the routing layer in the legacy WBAN networks. A compromised node presents multiple fake identities which could trick other devices into transmitting information to the same

node.

Another example of privacy related attacks is activity tracking. A variant of WBAN used for intelligent personalized sport training was proposed by Alonso et al. [37]. Eavesdropping on such networks could allow adversary to infer what type of activity the user is engaged in. Vitals such as heart rate, blood pressure, glucose levels are clear indicators of user's level of activity.

Denial of Service (DoS) constitutes any illegal attempt to render the network in-operational or reduce throughput. They cut across several layers of the network stack. Wood et al. carried out an extensive study on DoS attacks in sensor networks including WBANs [1]. They classified them into physical layer attacks which includes signal jamming and tampering, link layer attacks, routing attacks, and transport layer attacks.

2.2 Existing Security Solutions

Several solutions have been proposed to address the aforementioned security concerns in smart medical networks [31, 32, 24, 34, 3, 39, 38, 29]. They are classified into cryptographic, trust based, and intrusion detection systems.

The IEEE provided a pre-installed cryptographic security suite for the modern 802.15.6 WBAN standard. It consists of three profiles which are unsecured, authentication, and authenticity with confidentiality. However, the inherent cryptographic protocol still possess design flaws that exposes other vulnerabilities [35].

A security suite for WBANs was proposed by Sampangi et al. [31]. Their solution takes a no key exchange approach by the use of reference frames. It was implemented in two levels of hierarchy namely the IAMKeys (Independent and Adaptive Management of Keys) and KEMESIS (Key Management and Encryption for Securing Inter-Sensor Communication). The earlier is used for securing communication between the cluster heads and base station while the latter was used for node-to-cluster head transmission. One major caveat with their solution is that it requires buffering of pair-wise communication backlog which incurs significant memory footprints.

Sangari et al. implemented a public-key cryptographic protocol similar to the Elliptic-Curve Dife-Hellman (ECDH) for inter-node communication [32]. They used electrocardio-

gram (ECG) waves data read by both communicating nodes to independently generate the same random number x at both communicating end points. This is made possible by examining specic features of the ECD waves like amplitude, intervals, and digital slope, with some subsequent noise filtering techniques. One major drawback in their solution is that it only applies to sensors measuring ECG data.

Li et al. proposed a lightweight trust system based on self correlation techniques [24]. The protocol was designed such that every node maintains a vector of its individual trust values for every node in the network. A sender queries its neighbors for their opinion about the intended recipient. It ascribes weights to the received opinions based on their similarity of opinion vectors which is computed by an inner product. Their solution could be susceptible to malicious recommendations made by compromised nodes.

Anandkumar et al. implemented a distributed trust scheme to detect sybil/clone attacks within a complex WBAN (e.g. multiple patients in an ICU) [2]. Their protocol was designed to be a self-healing, randomized, and efficient protocol where legitimate nodes cooperatively evaluate neighboring nodes and isolate replicas. A significant caveat in their solution is the assumption evaluating nodes do not provide biased recommendations.

Salem et. al. developed a centralized machine learning based intrusion detection system using support vector machines for classification of activities and linear regression for assessing outliers [30]. Their solution focused on detecting irregularities in values aggregated by the cluster-head from the sensors. They converted individual snapshots of values reported by the sensors at time t to a row vector which was fed into the pre-trained SVM classifier. They compared data sets marked as suspicious with a row vector generated by linear regression. Once a threshold between the suspicious and forecasted row vector was exceeded, an alarm was raised. The cluster head performs all detection activities ranging from training to classification and regression. The major drawback in their solution was the cluster-head being a single point of failure and contention.

Thamilarasu proposed iDetect, a multi-objective WBAN intrusion detection system based on genetic algorithms [34]. The researcher defined three core metrics to address the trade-off between security and resource overhead which are the detection rate R, false positive rate P, and total energy usage E. Three objectives which were to maximize R, and minimize both

P and E were used to facilitate the selection, recombination, mutation, and replacement procedures of the genetic algorithms. However, the time taken for the algorithm to arrive at an optimal convergence point between R, P, and E is relatively high in certain scenarios.

Additional inherent limitations also exist in the examined security systems. These are usually fundamental to cryptographic, trust, and intrusion detection systems, as discussed in the previous chapter. In order to address these limitations, we employ the use of mobile agents to carry out distributed intrusion detection. We discuss the evolution of mobile agents in the next section, which is ensued by a study of existing agent based intrusion detection systems.

2.3 Autonomous Agents

The authors in [27] were one of the pioneering researchers in the design of software agents. They define agents as a class of software that acts on behalf of its users, like a proxy. Such agents were known as static agents, since they do not possess migration capabilities. Gray et. al [12] went further to provide deeper insights on how mobilizing agents could significantly improve the efficiency of parallel and distributed data intensive systems. By giving an agent the capability of serializing its code and state into a byte stream, it could be transmitted via the network channel resulting into a migration of the agent from one computing node to another. On reception at the destination node, the byte stream is decompressed and de-serialized into a concrete program which resumes execution based on its saved state.

Implementation of mobile agents was initially attempted on Java platforms due to its cross-compatibility across heterogeneous systems running a JVM. Danny et al. coined the term 'Aglets' in their research work on a successful implementation of a Java-based mobile agents framework. However, implementing such on less portable languages like C/C++ posed challenges resulting from differences in source and destination execution environments. Researchers such as Chen et al. addressed the portability challenges by including an embedded C/C++ interpreter in the agent framework for portable execution [8].

2.4 Agent based Intrusion Detection

The use of autonomous code for intrusion detection was originally conceived by Balasubramaniyan et.al [4]. The authors implemented a hierarchical system that carries out multi-level

detection across different echelons of the network for traditional computer systems. A system known as DIDMA was proposed in [17]. The autonomy of the agents were relatively higher with a combination of both static and mobile agents. It was also designed for traditional systems where resource availability is not a significant constraint. While static agents run on each node in the network, intervention requests could be sent to a manager system for resolving uncertainties. The manager responds with a mobile agent to carry out higher level correlations on the requesting node. A major drawback in their approach is the manager system being a single failure point. Another mobile agent based intrusion detection was proposed by Bernades et al. to run on resource constrained networks [6]. However, their system was still flawed by its centralized architecture.

A versatile security protocol was proposed by the authors in [41]. The researchers designed a compound static agent that aggregates three sub-agents with each carrying out anomaly detection in file, user, and network access patterns respectively. Agents were also cryptographically secured with symmetric and asymmetric encryption and authentication schemes. While the system is considered to be robust, it was designed for traditional systems with abundant computational resources. A decentralized mobile agent based intrusion detection system was proposed Kachirski et al. for clustered wireless sensor networks [15]. While data gathering happened on a per-sensor scale by a dedicated agent, actual classification and detection takes place at the cluster-heads based on the aggregated data. Khanum et al. proposed a similar architecture with an increased level of sophistication in [18]. Their system included a database that comprises of known attack signatures which serves as a comparison basis for real-time network activity patterns. The database also continually evolves with continuous update of signatures for adapting to newer unknown attacks. While the system is adaptive, resilient, and versatile, it incurs extra overhead in continuous communication with the external database.

Security solutions for connected medical devices are required to cope with network mobility, computational power, and communication constraints. To address these challenges permeating existing solutions, we propose a distributed mobile agent IDS framework. We employ a layered and decentralized hybrid architecture with mobile agents performing intrusion detection at different hierarchies of the network.

Chapter 3

ATTACK MODEL

In this chapter, we define the possible attacks that demonstrate the capabilities of our experimental threat models. Based on our evaluations, we identified three classes of attacks as the ones most tangential to the medical domain. In addition to modelling possible attacks, we establish different adversarial strategies.

3.1 Types of Attacks

3.1.1 Denial of Service (DoS)

DoS can be defined as any illegal attempt made to impede or totally sabotage the functional operations of a system in any capacity. This is the most endemic attack in the computing space, and as a result have been carried out through several means. The following are considered to be examples of DoS attacks common in IoMT networks.

- **Sender Radio Exhaustion**: This is usually targeted at nodes transmitting information. It is carried out by increasing the rate of transmission, thereby resulting into increased energy usage by the radio, and eventual battery exhaustion.

- **Receiver Radio Exhaustion**: This focuses on the receiver of packets. It is carried out by hijacking multiple transmitters and concentrating a barrage of packets on the receiver. This will inadvertently lead to increased energy usage and exhaustion due to continuous reception and processing of packets.

- **Decoy Packets**: This is also focused on the network recipient. In such attacks, the malicious transmitter sends noise to act as decoys for the receiver. By pre-occupying the receiver with the burden of filtering the noise, it inadvertently prevents the receiver from carrying out its primary functionality which involves processing actual packets. The cluster-head is a common victim of such attacks.

- **Sink Holes**: This is an example of DoS that affects the attacker's neighboring nodes. The attacker prevents information from getting to its destination, or slows down the network in general by dropping most or all packets in transit.

We consider DoS to be a leading concern within the medical space based on the following motives behind such an attack:

- Adversary could use this as a means to deny victim of timely medical response to an urgent situation, thereby causing victim physical harm or loss of life. This is mostly fomented by personal grudges, blackmail, a means of victim coercion, or ransom benefits.

- Adversary might be interested in sabotaging devices' manufacturer brand by making them seem faulty in order to give another competing brand, a marketing advantage.

We consider our adversary to possess the ability to hijack and reprogram sensor nodes to pump data at a faster/slower rate, or transmit noise.

3.1.2 Data Fabrication and Falsification

Data fabrication and falsification are considered the most common techniques used in carrying out data related attacks. Fabrication in the context of a sensor network could be described as computational fabrication of invalid data that is not related to the physical phenomena being observed by sensing data. Falsification on the other hand, is the illegal modification of valid data. This kind of attack is fairly easy to execute in open networks like WBANs which communicate through connectionless protocols, thereby not requiring any sort of pre-authentication or handshake. Any device that could transmit within the radio frequency of the respective sensors could be used to generate false data for the cluster-head. However, execution of such attacks on wired devices such as the smart scanners are more challenging as the computing chip needs to be physically accessed and reprogrammed.

Data driven attacks could be highly consequential because they tend to provide misleading results which could lead to wrong and potentially dangerous actuator response, prescriptions, or even lack of any required medical response. They are mostly carried out for the following reasons:

- To inflict harm by causing patient to receive misinformed/misdirected medical response to certain conditions.

- To increase operational costs of medical organizations by introducing a lot of false alarms and unnecessary incidence response.

In this capacity, our adversary is considered to be powerful enough to hijack a sensor and modify sensed data.

3.1.3 Privacy/Data Breach

This encompasses all form of attacks that results into illegal or unauthorized access to private medical information. The most common way of executing such attacks is through a passive listening radio device tuned to the same broadcast frequency of the wireless medical devices. More sophisticated adversaries could remotely reprogram a node in the network to route private data to a certain location. Although data breaches may not have immediate consequences, it could be used as a medium for blackmail on larger scales. The following are three common motives behind data breaches:

- To gain leverage for coercing compromised individuals into performing certain deeds including parting with some finances.

- For financial gains in the cyber black market.

- Inducement of medical institutions into providing certain amount of money in order to avoid facing disciplinary action from certain health regulation bodies.

Our adversary is assumed to be capable of mirroring packets to illegal destinations. Detecting passive listeners are outside the scope of our research.

3.2 Adversarial Strategies

We establish three different strategies that define the level of sophistication for our experimental adversaries.

3.2.1 A Malicious Adversary

This type of adversary takes an aggressive approach resulting in a lot of damage within a short period of time. They are the easiest to detect due to their overt strategy whereby they attack at almost every opportunity, and do so on a large scale.

3.2.2 A Suspicious Adversary

Unlike their malicious counterparts, we model a suspicious adversary to focus primarily on long term survivability in the network. They take a more subtle approach, thereby causing little bits of damage over a long period of time. They do their best to stay out of the radar by ensuring they operate covertly. Detecting suspicious adversaries is considered more challenging than the malicious variants. However, they eventually get detected after a longer time period.

3.2.3 An Elusive Adversary

The elusive adversaries are designed to be the most sophisticated and challenging to detect. They oscillate between the benign, malicious, and suspicious states at random intervals. This makes them increasingly difficult to classify accurately.

Chapter 4

PROPOSED INTRUSION DETECION SYSTEM (IDS)

This chapter provides an overview of the proposed intrusion detection for connected medical devices. We discuss the intended network structure, system requirements and design architecture in the following sections.

4.1 Network Architecture

A smart medical network usually consist of various devices ranging from patient wearable sensors to smart imaging devices. For the purpose of this research, we design a concise but generalizable network topology representative of these devices.

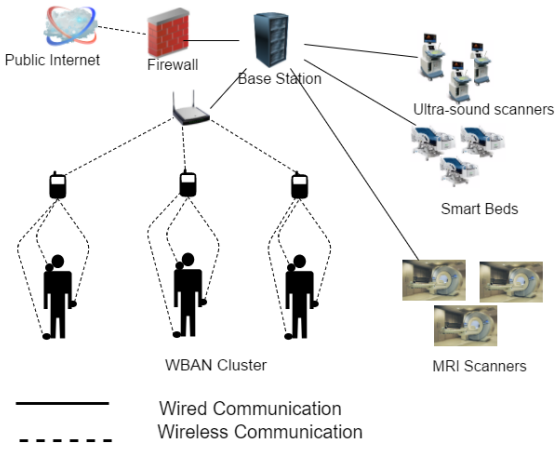

Figure 4.1: IoMT Network Architecture

WBAN sensor devices communicate via the IEEE 802.15/6 WBAN wireless network standard, and send streams of numeric physiological data to their respective cluster-heads. The cluster-heads and other classes of sensing devices such as the smart-beds utilize regular ethernet based TCP/IP. However, the imaging devices communicate using the DICOM network standard. They stream frames of images compressed according to the DICOM message exchange format [7]. Based on the above network architecture, the WBAN has lower throughput and range compared to the wired ethernet and DICOM networks.

4.2 IDS Requirements

In addition to addressing the threats posed by our established adversarial model, other non-functional requirements are considered in order to guide our design process and ensure it improves on the limitations characterizing existing systems. It is paramount that our system should be scalable, fault-tolerant, conservative, easily deployable, and extensible without significant compromise on its efficiency in providing adequate security. In order to achieve these requirements, we design a hierarchical hive of mobile agents independently but cooperatively carrying out intrusion detection across different segments of the network. The following outlines each of the requirements and describes how they are addressed by mobile-agents:

- **Scalability:** In an IoMT network, a potential point of contention is the cluster-head for the sensing devices. However, mobile agents are executed and propagated autonomously without continuous reliance on the cluster-head. Hence, an increasing number of nodes within the network will only lead to increase in number of required agents. This impacts the cluster-heads solely at point of agent instantiation and initial dispatch.

- **Fault Tolerance:** The system could survive multiple node failures considering that mobile agents could adjust their itineraries according to changes in available routes. However, if the cluster-head fails, the WBAN (sensing layer of the IoMT network) ceases to exist considering that the cluster-head is responsible for data aggregation and communication coordination across the sensors.

- **Conservativeness:** The use of mobile agents ensures that the computationally demanding task for intrusion detection is distributed across the network. This is done such that available resources are used optimally without overloading or underutilizing any particular node. Secondly, the agents are considerably smaller than the size of aggregated network traffic data as demonstrated in the next chapter. Hence, there is less communication overhead in transmitting agents.

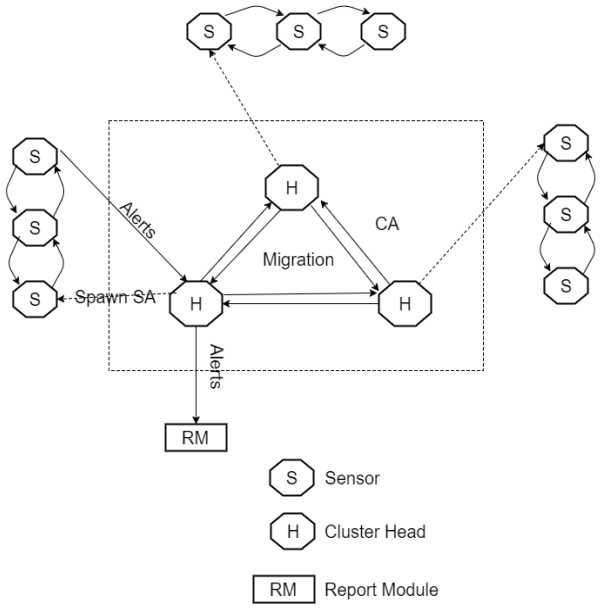

Figure 4.2: System Architecture

- **Ease of Deployment:** The entire system could be deployed on the cluster-head while the agents automatically propagate into the network.

- **Extensibility:** Existing agents could be terminated by control commands from the cluster-head while newly extended ones could be dispatched.

4.3 IDS Architecture

Figure 4.2 illustrates our proposed security system architecture. We propose two levels of detection namely, sensor and cluster-head level detection respectively. The former is carried out by a group of agents referred to as sensor agents (SA). The sensor agents carry out localized intrusion detection per device. The latter is carried out by cluster-head agents (CA), which are designed to be relatively sophisticated. The CAs carry out intrusion detection per cluster using a distributed tally system explained in the next section.

Having considered an overview of the proposed system architecture, we now present further details and algorithms for the agents which form the essence of the security suite.

4.3.1 Sensor Agent

A sensor agent (SA) is a single instance of mobile code designed to carry out network/device level detection for a group of network nodes. They are spawned from the cluster-heads and preloaded with two sets of parameters. The first set includes variables that guide the agent's trajectory and state management, while the latter is derived from training algorithms to be used for detection purposes. The parameters are outlined in table 4.1 and the ensuing code structure.

Table 4.1: Sensor Agent Parameters

Parameter	**Category**	**Description**
Id	Operational	A numeric identifier used for tracking a single SA instance
Clique Index	Operational	A numeric identifier for the group of sensors SA is assigned to within the WBAN cluster.
Detection Model	Security	An encapsulation of the parameters and algorithms used in carrying out network/device intrusion detection.
Cluster-Head Id	Operational	A numeric identifier for the originating cluster-head.
Aggregation Delay Period	Operational	The duration for which a SA waits for logs to accumulate on a sensor before aggregating and running them through its detection algorithm.
Detection Type	Security	A field that identifies SA's detection algorithm as Machine learning (ML) or Polynomial Regression (PR).
Itinerary	Operational	A list of sensors within the SA's network clique which defines its trajectory.

According to the code structure, the fixed size of an agent's state is exactly 10 bytes, while the size of the variable parameters are dependent on the number of devices designated to agent and the type of detection algorithm being used. For example, a support vector classifier needs to store the vector coordinates of its hyperplane whose dimensions are determined by the number of data features. On arriving at an arbitrary sensor (SN) within its set trajectory, the SA waits for a preset delay period to cumulate network/sensor data.

Subsequently, the SA aggregates them into a single analytical entity and runs them through its pre-trained detection algorithm which labels activities as benign, malicious or suspicious. In the first case, the SA migrates to the next node within its itinerary, while a malicious or suspicious classification results in a direct alarm outbreak or cluster-head intervention request respectively. The intervention process is described in detail under the special agents subsection. Algorithm 1 describes the sensor agent protocol.

Algorithm 1 Sensor Agent Protocol Algorithm

Require: $SA.isTrained \equiv true$

```
wait(aggregationDelay)
for all logEntry FROM entries[lastAggIndex] do
    SA.cumulate(logEntry)
end for
result ← SA.analyzeEntries()
if result ≡ MALICIOUS then
    SA.triggerAlarm()
else if result ≡ SUSPICIOUS then
    SA.triggerCHInterventionRequest()
end if
SA.hop(next_node_in_itinerary)
```

4.3.2 Special Agent

A special agent is an instance of mobile code that is dispatched by the cluster-head in response to an intervention request from a sensor agent. They sweep the entire cluster, gathering network activity data in the process, and reporting back to the originating CH.

The collected data is run through a conflict resolution detection algorithm which is trained for a group of sensors as opposed to the per-sensor SA (sensor agent) training set. A negative classification of data implies the possibility of a distributed intrusion across nodes within the scanned cluster. Such attacks have very subtle impacts when examined on a per-device scale. Special agents possess the same attributes with SA with a shorter aggregation delay period.

An intervention request is made by a SA to its originating CH if it is unable to classify a sensor's network activities as malicious or benign. The request consists of the agent and sensor ID. On reception of such request, the CH creates a signature from the request attributes, and caches it to prevent responding to duplicate requests. The CH subsequently spawns a special agent, and populates its itinerary with the addresses of every sensor within the local cluster. Following the classification of data reported by the special agent, the CH leaves the intervention request signature in its cache for a set duration before invalidation to prevent successive response to further requests. The special agent protocol is summarized in algorithm 2.

Algorithm 2 Special Agent Protocol Algorithm

Require: $SA.interventionRequest.isNew()$

$CH.train(SP)$

for all *sensors IN cluster* **do**

$SP.cumulate(logEntry)$

end for

$SP.hop(CH)$

$result \leftarrow CH.analyzeEntries()$

if $result \equiv MALICIOUS$ **then**

$CH.triggerAlarm()$

end if

$invokeAfter(cacheInvalidationFunc, validPeriod)$

4.3.3 Cluster-Head Agent

A cluster-head agent (CA) is another instance of autonomous code which could either be static or mobile depending on the network configuration. Static agents are used on single clustered WBANs comprising of one cluster-head (CH). CAs are designed to play the role

of a sensor agent for cluster-heads. However, there are some key differences in their mode of operations. A cluster-head only spawns a single instance of CA. Secondly, CAs only perform network intrusion detection considering that the CH is not a sensing or imaging device requiring device data anomaly detection. A CA is also trained with a different data set that reflects network traffic from a CH's perspective. A static CA resides on its originating CH and carries out intrusion detection at set intervals, while a mobile CA works in a similar fashion to a regular SA where it traverses nodes (cluster-heads) within its defined trajectory and carry out localized detection. Algorithm 3 summarizes the cluster-head agent execution flow.

Algorithm 3 Cluster-Head Agent Protocol Algorithm

Require: $CA.isTrained \equiv true$

$wait(aggregationDelay)$
for all $logEntry\ FROM\ entries[lastAggIndex]$ **do**
  $CA.cumulate(logEntry)$
end for
$result \leftarrow SA.analyzeEntries()$
if $CA.getCHId() \neq currCH.getId()$ **then**
  $CA.broadcast(\{result, currCH.getId()\})$
else
  $finalResult \leftarrow CA.computeMajorityVote()$
  if $finalResult \equiv MALICIOUS$ **then**
    $CA.triggerAlarm()$
  end if
end if
$CA.hop(next_node_in_itinerary)$

The proposed system is not designed to rely on the higher network entities such as the medical servers, hence there is no higher level service that resolves suspicious (uncertain) results such as the special agents. As a result, CAs employ a distributed voting tally system to determine their final classification results. On visiting a foreign CH (one that is different from its originating CH), C_i, the CA performs regular network analysis and broadcasts its

result about C_i to other reachable CAs. Every other CA regardless of where they are in the network examines the received broadcast message, and checks if C_i is its originating CH. In such a case, the CA updates its opinion cache with the received result broadcast while the other CAs discard theirs. The CA which originated from C_i eventually cycles back to its originating CH and computes a majority vote from opinions provided by other CAs on C_i before deciding to raise an alarm or not.

Chapter 5

SYSTEM DESIGN AND METHODOLOGY

This chapter outlines the operational procedures of the proposed system. We also give details on the detection algorithms, their roles, and the rationale behind using them.

As discussed in the previous chapter, the sensor and cluster-head agents are responsible for carrying out intrusion detection at the diagnostic and aggregating hierarchy of the network respectively. For the sake of clarity, we delineate the WBAN sensing devices and the other types of smart medical devices by classifying them as category A and B respectively. The process by which the agents operate in unison is outlined as follows:

- The cluster-heads (CH) are preloaded with training sets for both network and device level detection.

- In the category A network (WBANs), two separate instances of mobile code instance known as sensor agents (SAs) are created and trained for network and device level detection respectively.

- Both instances of SAs are duplicated until there is one network and device detection sensor agents for every group of devices within the cluster.

- The SAs are propagated throughout the WBAN to their respective trajectories to perform localized detection.

- The SAs only travel with the state of their trained detection algorithm, excluding the training dataset. This ensures agents are minimal in size, thus minimizing communication overhead with each hop.

- On arrival at a sensor node, a SA aggregates network activity or device data depending on its role as a network or device intrusion detection agent.

- It runs aggregated information through its detection algorithm and classifies them as either benevolent, in which it migrates to the next node, malicious thereby triggering an alarm, or suspicious where an intervention request is made to the CH.

- Once a CH receives an intervention request, a special agent is instantiated and dispatched to sweep the entire cluster for network activity or device data. This is done to ensure more sophisticated adversaries that tend to distribute their attack vectors across the network thereby appearing subtle could still be detected.

- It delivers the aggregated information to the CH which in turn runs it through an instance of the algorithm trained for cluster-scale detection. In this case, the classification result is binary which could either be benign or malicious which will result into a red alert.

- In a single cluster WBAN, an instance of a static (no migration) cluster-head agent (CH) is spawned to carry out localized network intrusion detection at set intervals on the CH.

- Device-level intrusion detection is not executed at this hierarchy of the network as the cluster-heads do not measure any data.

- The training sets for CAs are different based on the cluster-head's communication medium being Ethernet-based.

- Additional measures were taken in scenarios consisting of multiple interconnected clusters, especially in a medical facility housing multiple patients.

- One mobile CA is created and trained for every CH with each of them independently traversing the networks of CHs for localized cluster-head intrusion detection.

- A CA does not trigger alarms immediately after detecting malicious network patterns. Instead, it shares its detection results with other CAs, which in turn discard the message except the CA which originated from the cluster-head being analyzed.

- The receiving CA adds the result to its cache (a data structure designed to store multiple detection results from every CA).

- On complete culmination of every CA's opinion about its originating CH, it runs a vote and takes its next course of action based on the majority. If half or less of the other CAs reported the CH as benevolent, then an alarm is triggered.

- A similar process is carried out for the category B devices. However, it differs in the sense that the agents do not travel directly from one device to another. The agents migrate through the central network router which serves as a communication gateway across the devices.

The proposed system consists of two detection types which are network and device intrusion detection. In the former, we focus on identifying aberrant changes in network traffic patterns. We use this to address attacks such as DoS and privacy breaches where adversaries rely on the network channels as a medium for operation. The latter is used for detecting faulty sensing devices or anomalous readings due to adversarial activities on a device scale such as data falsification/fabrication.

5.1 Network Intrusion Detection with Machine Learning

Network level threats are usually launched from any arbitrary part of the network, but executed through the communication protocol/media across connected devices. A DoS attack is a direct attempt to reduce network throughput or render it non-operational, while privacy breaches refer to any effort made towards acquiring private information from the network illegally. Hence, we consider both categories of attacks to be network level. Such attacks are known to always impact the state of the network, therefore detecting them requires a representation of the normal and anomalous operational state of the network to be maintained respectively as a comparison reference. Obtaining the network state requires the definition of a generalizable multivariate model to encode it for persistence and analytical purposes. There are several variables that could serve as possible indicators for the state of a typical network. Examples of such variables include the rate of packet influx/efflux, network throughput, Received Signal Strength Indicator (RSSI), etc. To the best of our knowledge,

analysis and interpretation of multi-variate data demands statistical techniques such as a trained Machine Learning (ML) algorithm. As a result, we examine multiple variations of standardized ML algorithms and carried out experiments to determine the optimal choice for network level intrusion detection for the proposed system.

Our Machine Learning Algorithm was developed through the following steps:

5.1.1 Data Collection and Transformation

We define a standard Internet of Medical Things (IoMT) network topology as shown in Figure 4.1. We randomly distribute the attack vectors across the network and run repeated simulations for both malicious and suspicious adversaries, while simultaneously extracting and labelling network trace information.

We define a wide range of features characterizing network traffic such as total packet size, number of packets, etc. Several features had disparate ranges of data. In order to unify their scales, we normalize each feature with the following series of equations. $\{\forall j : j \in F\}$ and all data entry (rows) i of the training set with n entries, where j is a data dimension/feature and F is the defined feature set:

$$x_{(i,j)} = x_{(i,j)} - \mu(j) \tag{3}$$

$$\sigma(j) = \sqrt{\frac{1}{n}\sum_{i=0}^{n}(x^2)} \tag{4}$$

$$x_{(i,j)} = \frac{x_{(i,j)}}{\sigma(j)} \tag{5}$$

where $x_{(i,j)}$ is a scalar value at data entry i and feature j, $\mu(j)$ is the mean for all $x_{(i,j)}$ for a feature j, and σ is the corresponding standard deviation. The result of Equations 3 to 5 executed in order is the z-score of each data point.

5.1.2 Principal Feature Extraction

We use the principal component analysis (PCA) technique to identify redundant features from the initial set. PCA is a multi-step mathematical transformation that identifies strongly correlated features which could either be combined or the one with highest relevance selected. We carry out PCA by deriving the co-variance matrix C from the normalized training data

D through equation 6. We derive the eigenvalues and eigenvectors for matrix C using the OpenCV library. The matrix is subsequently sorted in order of decreasing eigenvalue and eigenvector columns. $\{\forall v : v \in V\}$ where V is the set of eigenvectors for C, we compute the cumulative energy content $g|v|$ as presented in Equation 7. In the final thinning process, we chose a value L such that $\frac{g|L|}{g|v|} \geq 0.9$. Any eigenvector with a $g|v|$ lesser than $g|L|$ is discarded as representing a redundant dimension with little or no significant impact on the data trend. The final output $D^|$ of the PCA was each data entry projected on the remaining eigenvectors which has been normalized as unit vectors since all computations were carried out on z-scores not actual data values. The projection is a dot product operation presented in Equation 8.

$$C = \frac{1}{n-1} DD* \tag{6}$$

$$g|v| = \sum_{i=0}^{v} g|i| \tag{7}$$

$$D^| = \begin{pmatrix} \vec{x_1}.\vec{v_1} & \vec{x_1}.\vec{v_2} & \cdots & \vec{x_1}.\vec{v_k} \\ \vec{x_2}.\vec{v_1} & \vec{x_2}.\vec{v_2} & \cdots & \vec{x_2}.\vec{v_k} \\ \vdots & \vdots & \ddots & \vdots \\ \vec{x_n}.\vec{v_1} & \vec{x_n}.\vec{v_2} & \cdots & \vec{x_n}.\vec{v_k} \end{pmatrix} \tag{8}$$

D^* is the conjugate transpose of training set matrix D, x_i is a row vector representing a data entry in D, v_i is the ith eigenvector from the selected k eigenvectors.

5.1.3 Algorithm Assessment and Selection

We establish certain metrics by which we determine the most suitable machine learning algorithm for our proposed system. We train and test five supervised machine learning algorithms commonly used in intrusion detection (SVM, DT, RF, KNN, and NBC). The detailed description for each established metric is defined in the following sub-sections:

Accuracy

This is the ratio of correct to the total number of test classifications as depicted in equation 9.

$$x_a = \frac{t_n + t_p}{T} * 100\% \tag{9}$$

where x_a is the percentage accuracy, t_n and t_p are the number of true negatives and positives respectively, and T is the total number of classifications carried out. Classification accuracy is a metric used in every assessment of ML algorithms regardless of the context in which they are applied. It gives a first level estimate for the algorithm's correctness based on the training and test data sets.

Cost-Benefit Ratio

This is a rather contextual metric derived by Ulvilla et al. for evaluation of a ML based intrusion detection algorithm based on its operational consequences [36]. It is defined as the ratio of false positives to negatives or its inverse depending on which classification error incurs more costs. In the context of medical networks, a false negative is considered more consequential as it could lead to loss of lives. Although, sensor networks connected to actuators such as insulin pumps could also result in significant harm to the patient. An example is a false positive classification triggering excess injection of insulin. Equation 10 shows the initial formula used in [36] to compute CBR x_c.

$$x_c = \frac{f_n}{f_p} \tag{10}$$

where f_n and f_p are the total number of false positives and negatives respectively. However, we augment their definition of x_c by considering cases two extreme cases.

The first case is such that an algorithm may have a relatively low amount of false positives and some false negatives, resulting in a large CBR that approaches infinity as false positives approach zero. Secondly, an algorithm might possess a relatively low amount of false negatives and a considerable amount of false positives which results in a low CBR that approaches zero as false negatives gets smaller. In both cases, there are tendencies for an algorithm with a relatively high amount of classification errors to possess a low or even zero CBR, or the reverse case where low amount of errors still produces a high CBR. This does

not accurately reflect the fact that a high number of errors should result into a high cost or vice-versa.

As a result, we define cost benefit ratio x_c in Equation 11 as:

$$x_c = \frac{2f_n + f_p}{f_p + t_n + t_p} \tag{11}$$

In this case, the accuracy is also factored in by including t_n and t_p. As t_p increases, the influence of f_p in x_c decreases with that of f_n increasing, causing a resultant but more controlled increase in x_c. Conversely, a controlled decrease in x_c is realized as t_n increases.

Feedback Reliability Ratio

The feedback reliability ratio (FBR) was defined by Banerjee et al. as an inverse measure of the reliability of a ML based intrusion detection system [5]. It is defined by the ratio of a weighted error sum to the total number of observations as depicted in equation 12.

$$x_f = \frac{W_n * f_n + W_p * t_p}{T} \tag{12}$$

where W_n and W_p are weights assigned to the false positives and negatives respectively on a scale of 0 - 1. We experimented with several weight combinations, which led to the following conclusion; the condition $W_p < Wn \bigwedge W_p > 0.5W_n$ must hold to reflect an accurate estimate of their inverse reliability while maintaining the property of associating higher risks with false negatives.

Training Time

We examined a simplistic model of micro-controller/processor computational energy usage in equation 13 [40]. It is evident that runtime is the only variable within the model, implying that energy usage is a function of time.

$$x_e = C * f * V^2 * t \tag{13}$$

where C is the computing chip's capacitance, f is the clock frequency, V is the total voltage dissipated per unit time, and t is the computation time.

Machine learning consists of two phases of computation which are the training and classification. Our experimental results show that each ML algorithm took roughly the same

time for classification but varied significantly in training time. As a result, we establish the training time as the dominant factor in computing power usage.

Rank

This is a normalized aggregation of results from the other metrics which we use to rank each algorithm. The rank x_r is presented in equation 14 as:

$$x_r = \frac{x_a}{100} - \frac{x_c}{max(x_c)} - x_f - \frac{x_t}{max(x_t)} + 3 \tag{14}$$

where x_t is the training time and the value 3 is used for inverting the three negatives.

5.2 Device Intrusion Detection with Polynomial Regression

Device level threats constitute attacks that are launched and executed within a device. Data falsification/fabrication falls under this class of threats considering it involves illegal modification or synthesis of device data by a malware or other possible means. Contrary to network level attacks, their impact is constrained to the compromised node(s) without having any significant effect on other connected devices or the network in general. A similar approach is employed by maintaining a profile of the device state. While several indicators also exist for device state, most of them are not directly obtainable from the sensing devices since they do not possess a rich system API like traditional computers. Amidst the device state indicators, we consider the sensor data and timestamp to be most accessible and relevant to detecting anomalies introduced by data falsification/fabrication and device faults. We also define a model that estimates the normal trend of sensing device data, and use it as a baseline for detecting irregular device readings. While the network state model is dependent on several indicators, the equivalent for device state is solely dependent on timestamp and previous sensor readings. This corresponds to a time-dependent regression problem, which we choose to address with polynomial regression (PR).

The following steps were taken in implementing polynomial regression for the proposed system:

5.2.1 Data Collection

We implement the same network configuration used in the machine learning development. The data collection is done separately for sensing and imaging devices respectively. Data collection is done under benign network conditions to prevent the eventual model from being corrupted by malicious data. Data is extracted as a tuple of timestamp and sensor scalar value for sensing devices, while imaging devices have their data transformed in real-time from a pixel matrix P to a scalar value s through Equations 15 to 18.

$$V(P_{i,j}) = \vec{V}_{i,j} = \begin{pmatrix} \mu_r \\ \mu_g \\ \mu_b \end{pmatrix} \tag{15}$$

$$f(\vec{V}_{i,j}) = h(\ \vec{(i,j)}).\vec{V}_{i,j} \tag{16}$$

$$\vec{h}(\vec{(i,j)}) = \begin{pmatrix} g(|\vec{(i,j)}|)Mod255 \\ gg(|\vec{(i,j)}|)Mod255 \\ ggg(|\vec{(i,j)}|)Mod255 \end{pmatrix} \tag{17}$$

$$s = \mu(f(\vec{V}_{i,j})) \tag{18}$$

where the pair (i, j) represents the 2-dimensional index of each sub-matrix of the entire pixel. We divide the pixel matrix P into 16 sub-matrices $P_{1,1}$ to $P_{4,4}$. In Equation 15, we convert each sub-matrix to a 3-D vector with each dimension representing the average red, green, and blue channel values respectively. We subsequently define a function f in Equation 16 that converts the 3D vector into a scalar by computing its inner-product with another function h. h is defined to compute a hash of the sub-matrix indices i, j in form of a 3-D vector to ensure the position of the sub-matrix is factored in. The first element of the hashed vector is the output of a function g generating a 32-bit hash of the magnitude of vector (i,j) modulo 255, where 255 is the maximum value of a color channel within the 24-bit color depth specification. The second and third elements were derived from g being applied twice and thrice to the vector (i,j) respectively. The scalar value of the entire matrix P is subsequently computed in Equation 18 as the mean of all scalars from the sub-matrices. Although the

conversion process is fairly complex, it incurs fewer computational resources when compared to other matrix representations such as the eigenvalues.

5.2.2 Model Construction

Our initial attempt was the use of time-series analysis (TSA) which models the data as a linear time function, and supplements its accuracy by a season multiplier m. m is computed for every change in data trend as a real number. It is used to change the resultant gradient of the derived linear equation in order to adapt it to sudden changes in data trend. One major drawback of TSA is the data being required to be periodic such that the entire data trend is expected to be repeated at some point. As with most natural phenomena, physiological data is approximately closer to varying sine/cosine functions which are considerably difficult to approximate linearly.

We decide to address both challenges with a dynamic polynomial regression. Using a polynomial model ensures we get a closer approximation of the data trend. Secondly, its dynamism keeps it up to date with the latest valid changes in data trends. Our polynomial model is built as a function of time in Equation 19.

$$y^{|} = m_n t^n + m_{n-1} t^{n-1} + ... + m_1 t + m_0 \tag{19}$$

where m_n to m_0 are derived coefficients, and n is the polynomial order. In computing the coefficients, we construct a coefficient matrix C which is presented in Equation 20. The matrix is subsequently row-reduced to its echelon form with the element of its right-most column vector being the desired coefficients m_{0-n}.

$$C = \begin{pmatrix} n & \sum_{i=0}^{n} t & \cdots & \sum_{i=0}^{n} t^n & \sum_{i=0}^{n} y \\ \sum_{i=0}^{n} t & \sum_{i=0}^{n} t^2 & \cdots & \sum_{i=0}^{n} t^{n+1} & \sum_{i=0}^{n} yt \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \sum_{i=0}^{n} t^n & \sum_{i=0}^{n} t^{n+1} & \cdots & \sum_{i=0}^{n} t^{2n} & \sum_{i=0}^{n} yt^n \end{pmatrix} \tag{20}$$

For continuous update of the model to newer trends, every agent carrying out device-level detection maintains state information containing unique elements ($\sum_{i=0}^{n} t ... \sum_{i=0}^{n} t^{2n}$) of the coefficient matrix. Subsequent data samples classified as benign are cumulated with

the existing state matrix. The matrix is eventually expanded and row-reduced to its echelon form, thereby generating a new set of coefficients for the model.

In the next chapter, we discuss the results gotten from the individual algorithm assessments and the proposed system in entirety.

Chapter 6

EXPERIMENTAL PROCESS AND RESULTS ANALYSIS

This chapter describes in detail the tools used in carrying out our simulated experiments, data collection and pre-processing, definition of system assessment metrics, detection algorithm calibration, and the various simulation scenarios with corresponding results and their analysis.

A smart and connected medical network is known to consist of heterogeneous devices communicating on different network protocols. An example could be wireless sensing devices running on either traditional zigbee 802.15/4 or the modern 802.15/6 WBAN. Other categories of smart devices could also include smart and connected ultra-sound scanners or MRI machines running on the DICOM network protocol. To address the communication protocol disparity, we ensure the proposed system was designed to be oblivious of the differences between these network standards. For proof of concept, we run our tests on a network simulation software with a very close approximation to heterogeneous hardware in different real world scenarios. This also gives us the flexibility to emulate multiple usage contexts without the complexities involved in modifying/tuning actual hardware to get the same effect.

Prevalent attacks in IoMT networks such as DoS could also be simulated accurately on simulation software. We utilize the OMNeT based Castalia-3.2 WBAN simulator for our experiments as it provides an accurate emulation of network implementations across all stacks with full customizability. Additionally, its inherent radio model is versatile enough to simulate constructive and destructive interference, RX to TX to sleep transitions, and energy usage computation.

6.1 Tools

Table 6.1: Tools used in the course of this research

Hardware	**Specifications**	**Purpose**
Laptop 1	Memory: 8GB Processor: Intel Core i3 OS: Ubuntu 16.0	Used for development and simulation execution.
Laptop 2	Memory: 8GB Processor: Intel Core i7 OS: Windows 10	Used for result transformation and analysis.
Software	**Device Associated**	**Purpose**
Castalia-3.2	Laptop	Used at the core of the experiment to emulate different network scenarios, execute IDS protocols and algorithms, and results generation.
OMNeT 4.6	Laptop	Served as a framework with network related APIs for the Castalia simulator.
Open CV	Laptop	Provided APIs for several lightweight machine learning algorithms used in implementing the IDS.
Visual Studio Code	Laptop	Used as an editor in the development phase.

Although the experiment is conducted on a network simulation software, there are other tools utilized in different facets of the research such as the development environment, software libraries, the hardware platform used to execute the software tools, etc. The tools are described in Table 6.1.

The simulation was carried out on Castalia-3.2 running on Ubuntu 16.4. The software

utilizes APIs from the OMNeTPP framework to run background timers used for synchronizing communication within network nodes [25]. Castalia was built specifically for WBAN simulations. In addition to the preloaded IEEE 802.15/(6/4) implementations, it allows for total customization of the entire network stack from the wireless channel to the application layer. It also provides other features such as radio energy usage metrics, radio interference model, physical process modelling, etc.

While the entire simulation runs in a black-box terminal, it generates a detailed trace of the entire simulation. The generated trace can be parsed by the Castalia result module for graphical visualization and further analysis.

Table 6.2 shows the parameters used to run the simulation [7, 13, 20].

Table 6.2: Radio level network simulation parameters

Parameter	**Sensor Value**	**Cluster-head Value**	**Wired Devices Values**
Data rate.	1,024 kbps	147,456 kbps	1,002,400 kbps
Modulation Type.	DIFFQPSK	DIFFQPSK	N/A
TX range/power usage.	-10 dBm/3.0 mW	-1 dBm/29 mW	-0.001dBm/14 mW
Available Energy.	1,872 kJ	Rechargeable	Connected
Retransmission Interval.	100 ms	100 ms	100ms

Based on these parameters, we were able to emulate benign and compromised network scenarios, generate training data, and develop corresponding algorithms for both network and device-level detection. Emulating a compromised network required us to implement our adversarial model in the simulation. We discuss this implementation in the following section.

6.2 Adversarial Model Implementation

We identify launch points for each of the stated attack and briefly outline the measures taken to carry out such attacks in table 6.3.

Table 6.3: Attack Implementations

Attack	**Category**	**Launch Point(s)**	**Implementation**
Sender Radio Exhaustion	DoS	During packet transmission	Increasing the sampling and transmission rate of a sensing device exponentially.
Receiver Radio Exhaustion	DoS	During packet transmission	A group of malicious sensors target a victim and flood it with data packets.
Decoy Packets	DoS	During packet transmission	Sent noise to the CH as opposed to meaningful data packet.
Sink-holes	DoS	Both packet reception and transmission	Selective or no transmission of sensed data to CH.
Data falsification	Data driven attack	Right after an incoming sample of sensor/image data	Modifying sensed data before transmission to the CH.
Illegal Transmitter	Privacy Breach	During packet transmission.	Transmission of data to an illegal destination other than the CH.

The different attack strategies (malicious, suspicious, and elusive) discussed in Chapter 3 were implemented with the use of attack probabilities. The attack probability P is a number between 0 and 1 that determines the likelihood of an adversary launching an attack on getting

to a probable launch point. For example; the probability of a data falsifier modifying a sensor data sample right after its measurement. We assign a value P of 0.7 and 0.2 to malicious and suspicious adversaries respectively. On the other hand, the P value for an elusive adversary is variable in the sense that it changes at random intervals across a uniform distribution between 0 to 1. This was done to make them increasingly difficult to classify accurately, making them the most sophisticated adversaries.

In the next, section we discuss the results observed from the machine learning and regression algorithm assessment procedures.

6.3 Machine Learning

Machine Learning (ML) is primarily used for network intrusion detection. In terms of our adversarial model, ML addresses both DoS and privacy breach attacks by examining anomalous patterns in network traffic and triggering alerts on confirmation of such aberrations.

Five commonly used ML algorithms for intrusion detection were evaluated which are the SVM (Support Vector Machines), DT (Decision Trees), NBC (Naive Bayes Classifier), KNN (K-Nearest Neighbor), and RF (Random Forests).

The results observed from our defined metrics and their final aggregation are presented in Figures 6.1 to 6.4.

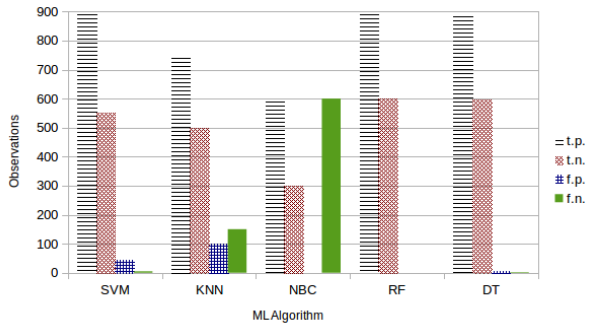

Figure 6.1: True Positives vs True Negatives

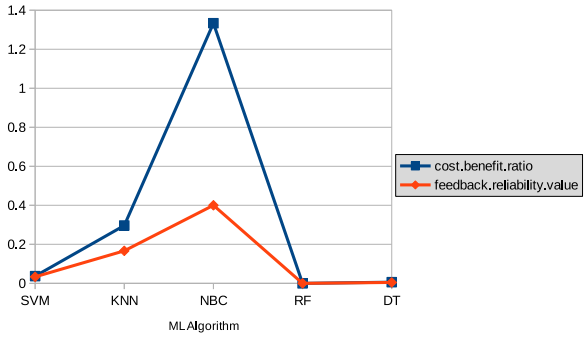

Figure 6.2: Cost Ratio and Feedback Reliability Value

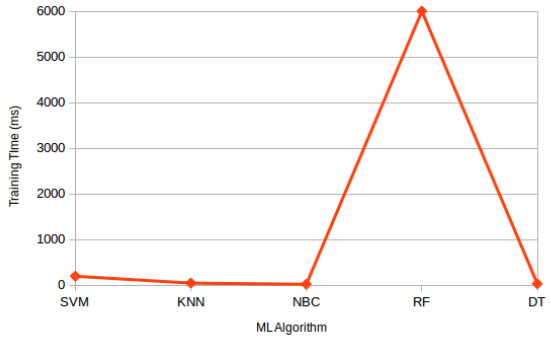

Figure 6.3: Training Time

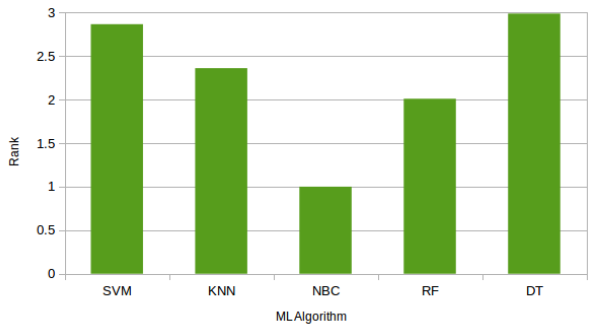

Figure 6.4: Final Rank Score

6.3.1 Classification Accuracy

The accuracy is shown in Figure 6.1 in form of the true/false positives and negatives. 1,500 total observations are made for the sensor and cluster-head traffic respectively. For sensor accuracy tests, we use 600 malicious, 300 suspicious, and 600 benign classes of data respectively. On the other hand, the cluster-head traffic was tested with 900 malicious, and 600 benign network data considering that classification at the cluster-head level is binary. The combined results are presented in the previously referenced Figures 6.1 to 6.4. The KNN and NBC algorithms incurred a high amount of false classifications, rendering them infeasible for the system. We proceed further in assessing other metrics relating to resource usage, etc to establish a more concrete measure of their suitability. The remaining three algorithms (SVM, DT, and RF) performed with significantly higher level of accuracy, with RF having a classification accuracy of approximately 100%. DT stands right next to RF with a few false positives and fewer negatives. The SVM on the other hand incurred a relatively higher number of false positives.

6.3.2 Cost-Benefit Ratio

As depicted in Figure 6.2, the NBC failed miserably in this context by incurring a large number of false negatives, which happens to be the strongest factor in determining the resultant CBR value. Although, KNN's CBR is order of magnitudes lesser than that of its NBC counterpart, it is still considered high enough to render it infeasible in comparison with the remaining three algorithms. The RF maintained 0 CBR as it incurred no false

classification with the SVM being slightly higher than that of DT due to its higher number of false negatives.

6.3.3 Feedback Reliability Value

This is also presented in Figure 6.2 with NBC and KNN possessing scores, order of magnitudes higher than their remaining counterparts. The FRV is a direct inverse representation of how reliable each respective ML classification algorithm could be with respect to the data model that characterizes the simulated networks.

6.3.4 Training Time

Figure 6.3 presents the time taken to train in milliseconds over a total of 11,059 training data entries for each ML algorithm. The training time is believed to be the major determinant of computational resources a ML algorithm is postulated to incur on an actual computing chip. The RF algorithm performs poorly in this context with a training time exponentially longer than its counterparts. Although the SVM incurred a significantly longer training time than the NBC, KNN, and DT algorithms, it is still considered to be within feasible estimates of what could be executed on the cluster-heads. The KNN and NBC performed excellently incurring a training time lesser than 20 ms.

6.3.5 Final Ranking

This is aggregated based on Equation 14, with its results presented in Figure 6.4. As expected, the NBC and KNN algorithms fell short extensively in classification accuracy which resulted in a lower score. RF incurred a relatively high computational overhead which has a significant negative impact on its final score regardless of its accuracy, CBR, and FRV. The SVM came close to par with DT but had a higher amount of false positives. We subsequently proceed with the Decision Tree (DT) ML algorithm as most suitable for the system.

6.4 Polynomial Regression (PR)

PR is utilized for device-level intrusion detection which addresses data falsification and fabrication attacks. This involves building a model as an n-order polynomial equation which is a

function of one or more independent variables. The equation is used to forecast/predict sensor data values that should be reported at certain times in the future. When time progresses to the forecast period, the sensed data value is compared with its predicted equivalent. Once their difference exceeds a set threshold, an alarm is flagged reporting an anomaly. At the same time, the model is continually updated from benign device data in real-time to ensure model adapts to changing trends.

Following the data collection and model construction process, we run tests to find the optimal value of the polynomial order n. While higher order polynomials are considered to be more accurate, they could suffer from over-fitting. Secondly, incrementing n implies an exponential increase in coefficient matrix size, and the time it takes to construct it. Figure 6.5 shows the resulting accuracy for each polynomial order. The coefficients derived for each polynomial order used in are presented in Table 6.4.

Table 6.4: Polynomial Coefficients

Order	**Coefficients**
1st	$m_0 = 104.527$, $m_1 = 0.000198$
2nd	$m_0 = 106.736$, m_1 = -0.00229, $m_2 = 5.197 * 10^{-7}$
3rd	$m_0 = 73.599$, $m_1 = 0.058684$, $m_2 = -1.817 * 10^{-5}$, $m_3 = -3.565 * 10^{-12}$
4th	$m_0 = 71.132$, $m_1 = 0.0678$, $m_2 = -2.51 * 10^{-5}$, $m_3 = 1.4 * 10^{-9}$, $m_4 = 2.227 * 10^{-15}$
5th	$m_0 = 53.526$, $m_1 = 0.176$, m_2 = -0.000174, $m_3 = 7.246 * 10^{-8}$, $m_4 = -1.090 * 10^{-11}$, $m_5 = 2.237 * 10^{-20}$

We observe a slight dip in accuracy between the linear and quadratic model due to the inherent property of quadratic equations possessing a single global maxim/minim. This impedes the possibility of approximating periodic data which usually have several minima and maxima. The increase in prediction accuracy from the cubic to the fifth order model

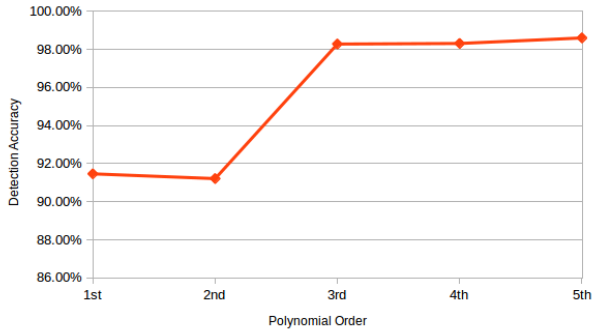

Figure 6.5: Order vs Accuracy

is considered not significant enough to warrant incurring the extra computational resources required. As a result, we proceed with the cubic model for the actual system implementation.

6.5 System Simulation

Our simulation is divided into three broad categories which are discussed in the next set of subsections. We employ a 2 x 4 x 9 factorial experimental design for each category in order to exhaust a significant amount of possible network compositions. Each dimension of the factorial design is outlined as follows:

1. **Detection Type:** Network Intrusion Detection, Device Intrusion Detection.

2. **Adversarial Composition:** Dominantly malicious, dominantly suspicious, dominantly elusive, randomly distributed.

3. **Percentage of Compromised Nodes:** 10 - 90%.

We define an attack probability as the chances that a node will launch an attack when in a possible launch state. While an attack probability of 0.7 and 0.2 is defined for malicious and suspicious attackers, the elusive ones are designed to be relatively difficult to classify. The elusive adversary constantly changes its attack probability over a uniform distribution between 0 - 1. Hence, they oscillate erratically across the the benign, suspicious, and malicious states.

In each category of simulation, the system is assessed by its detection accuracy, and energy usage.

6.5.1 WBAN Cluster (Category A Devices)

A WBAN cluster consists of multiple sensing devices attached to parts of the patients body, constantly monitoring physiological phenomena like ECG signals, blood pressure, etc, and relaying it to a higher powered central cluster-head. Figure 6.6 shows the network topology used in the simulation.

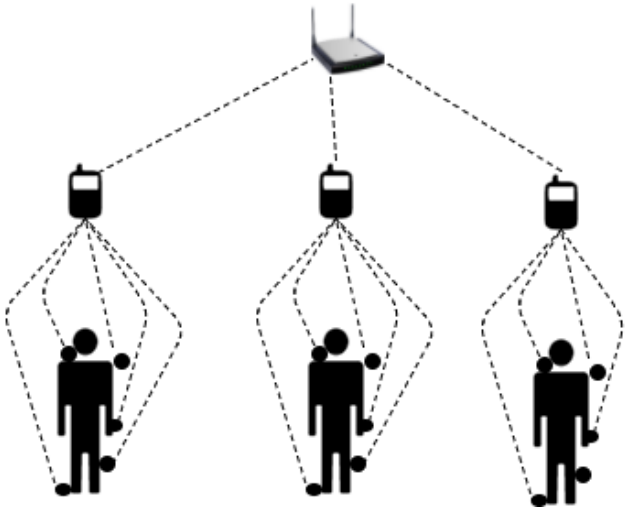

Figure 6.6: WBAN Cluster Topology

Each cluster consists of 5 sensors and 1 cluster-head. Several simulations are executed over the course of a simulation hour, which is equivalent to about 3 minutes in real-time. We vary the percentage of compromised nodes along with their adversarial composition, and present the observed results for both network and device intrusion detection in Figures 6.7 to 6.12.

As expected from figures 6.11 and 6.12, the classification accuracy of malicious adversaries is significantly higher than its suspicious, and elusive counterparts due to their overt approach. The system is less efficient against the elusive adversaries due to their erratic transitions from benign to malicious roles. The same trend is observed for device-level detection with polynomial regression (PR). However, the average accuracy is lower for PR consider-

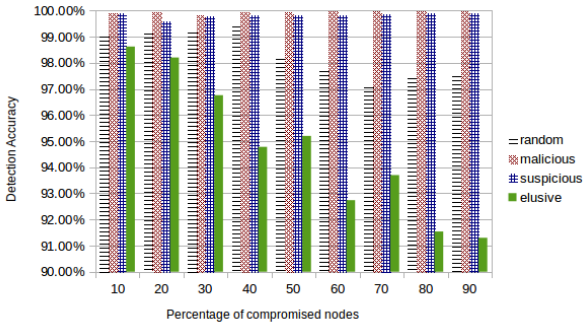

Figure 6.7: Network Intrusion Detection Accuracy (WBAN Clusters)

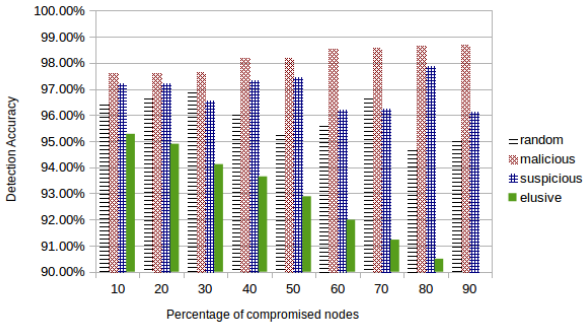

Figure 6.8: Device Intrusion Detection Accuracy (WBAN Cluster)

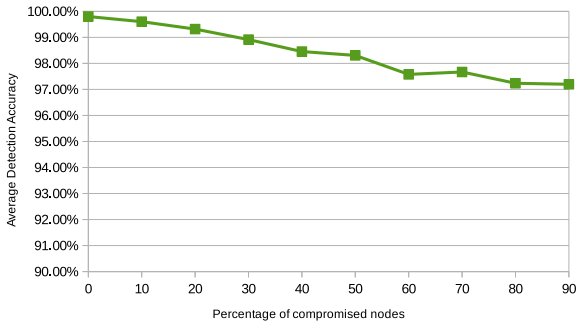

Figure 6.9: Network Intrusion Detection Average Accuracy (WBAN Cluster)

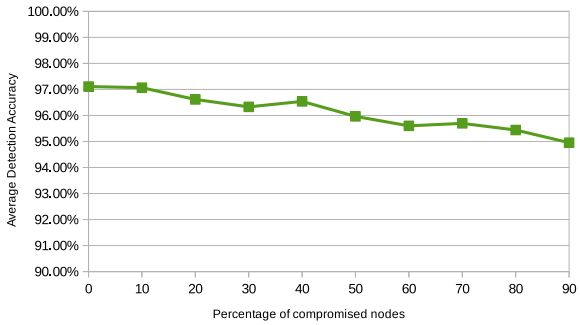

Figure 6.10: Device Intrusion Detection Average Accuracy (WBAN Cluster)

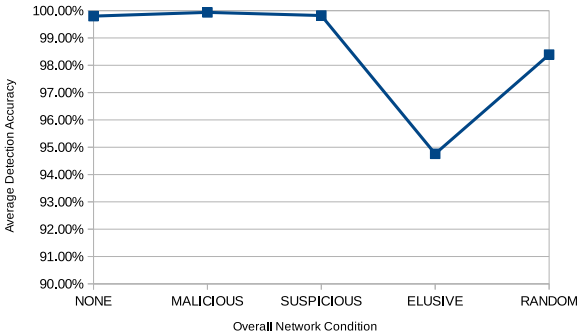

Figure 6.11: Network Intrusion Detection Accuracy Per Adversary (WBAN Cluster)

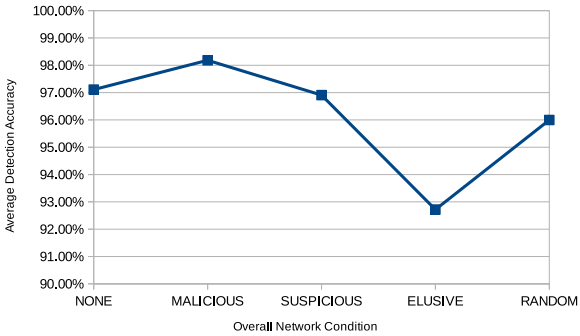

Figure 6.12: Device Intrusion Detection Accuracy Per Adversary (WBAN Cluster)

ing the highly irregular nature of the simulated patient ECG data. We believe real world ECG data is more uniform and structured compared to the simulated version. By using a

more stochastic data trend, we test the limits of our detection system in areas of extreme uncertainties.

Figures 6.9 and 6.10 shows the average accuracy of the detection system declining steadily for both network and device intrusion detection. This is due to the increasing number of elusive adversaries in the network as percentage of compromised nodes increased. This is further bolstered by the results presented in Figures 6.7 and 6.8 where the detection accuracy drops significantly with increasing percentages of elusive adversaries.

From our observations in Figures 6.11 and 6.8, the system performs best at an accuracy of 99.6% and 98.2% for network and device-level intrusion detection respectively. The best case is realized with the highest percentage being the malicious nodes due to their overt aggressive adversarial strategy. On the other hand, the worst case resulted in accuracy of 94.9% and 92.8% for network and device-level intrusion detection respectively. This occurs when a greater portion of the network adversaries are elusive in nature.

In addition to testing the system accuracy, we also measure energy consumption. The two main processes that usually result in significant energy usage is network communication and device computations. We extract the values of energy consumed by each device's radio directly from Castalia. The computational energy was derived as a function of computation time in Equation 13. The duration of each simulation is such that it is equivalent to one hour in the actual time unit. We run two sets of simulations with and without the intrusion detection system in place respectively, and got the following results presented in Figure 6.13.

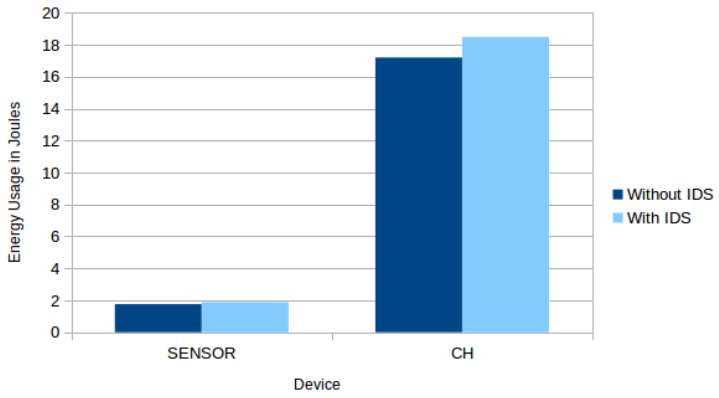

Figure 6.13: Energy Usage per Device (WBAN Cluster)

The system incurred an average of 5.2% and 7.03% energy overhead on sensors and cluster-heads respectively. We consider the results gotten in terms of both detection accuracy and energy in the context of WBANs to be amenable, and worthy of pursuing further by implementing the system on actual hardware.

In the next subsection, we discuss our experimental results from simulating a network of higher powered sensing and imaging devices (category B devices).

6.5.2 Connected Diagnostic/Imaging Devices (Category B Devices)

As depicted in Figure 6.14, a network of category B devices is composed of smart and connected higher powered devices such as smart beds, smart MRI scanners, etc. They differ in the sense that they are not constrained in energy supply. However, they are not primarily considered to be computing devices. The smart and connected parts of such devices are mostly add-ons or peripherals. As a result, the available processing or storage resources on the category B devices are lesser in comparison to traditional computers. Secondly, the devices are always connected directly to the network gateway either through wired or wireless media, without direct connection to each other.

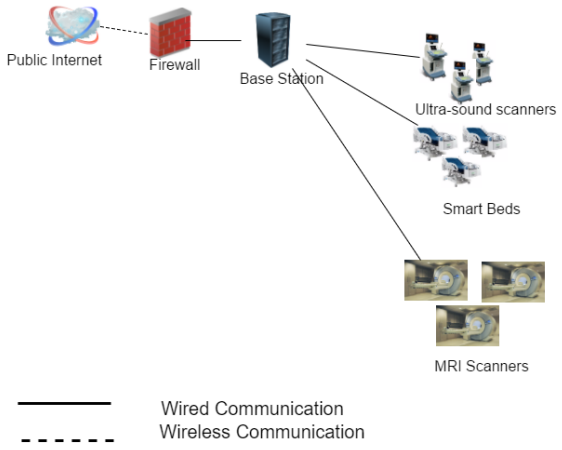

Figure 6.14: Simulated Cat B Network Topology

We continue with the factorial experimental design where we measure system accuracy for network and device intrusion detection under varying adversarial compositions. The resultant detection accuracy in the context of all simulated scenarios are presented in Figures 6.15 to

??.

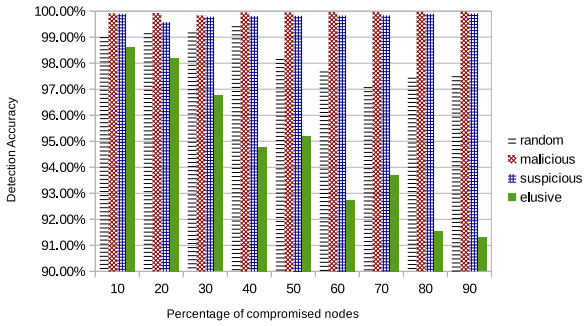

Figure 6.15: Network Intrusion Detection Accuracy (Category B Devices)

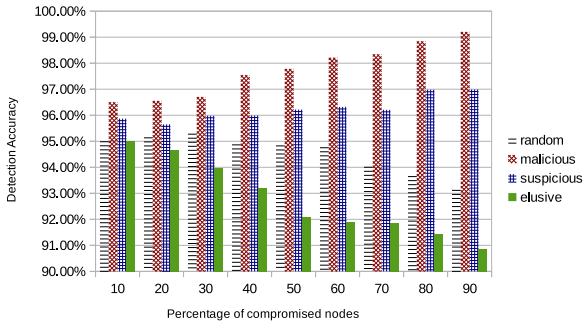

Figure 6.16: Device Intrusion Detection Accuracy (Category B Devices)

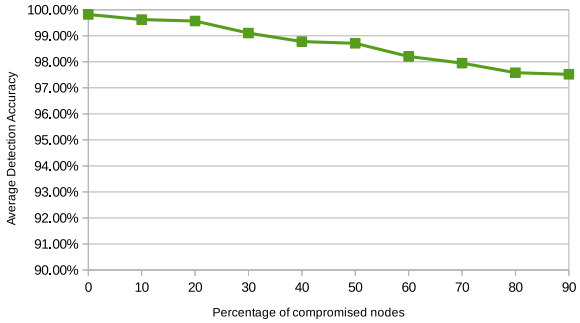

Figure 6.17: Network Intrusion Detection Average Accuracy (Category B Devices)

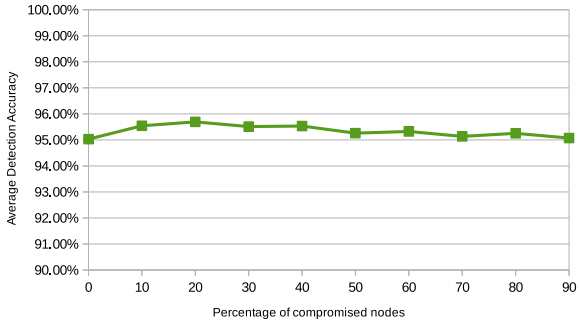

Figure 6.18: Device Intrusion Detection Average Accuracy (Category B Devices)

A trend similar to the simulation results from the WBAN cluster is also observed with the category B devices. The system performed best against malicious nodes and was significantly less effective against elusive adversaries for both network and device intrusion detection. However, the overall network intrusion detection accuracy was slightly higher than that of the WBAN clusters. This was due to the fact that we simulate category B devices to emulate wired connections to the cluster-head/base station. Hence they possess increased network bandwidth (Table 6.2), little or no interference, less packet retransmissions and a more regulated flow of traffic. One notable observation is the significant decrease in accuracy from the network to device intrusion detection. This is due to the application of our regression

algorithm on image data from imaging devices. We use an approximation of the pixel matrix into a scalar value derived by Equations 15 to 18. This has a slight negative impact on the accuracy of our regression algorithm, but resulted in exponential decrease in computational time from using Eigenvalues. The best case accuracy is 99.9% and 97.81% for network and device-level detection respectively. In the worst case scenario, an accuracy of 95.72% and 92.91% is realized for both levels of detections respectively. The relationship between the adversarial composition and system accuracy is very similar to that of the WBAN Cluster.

We consider energy foot-printing to be irrelevant in this context as the devices involved rely on continuous AC power supply. In the next subsection, we examine the complete network architecture presented in Figure 6.21 and extracted relevant results.

6.5.3 Combined IoMT Network

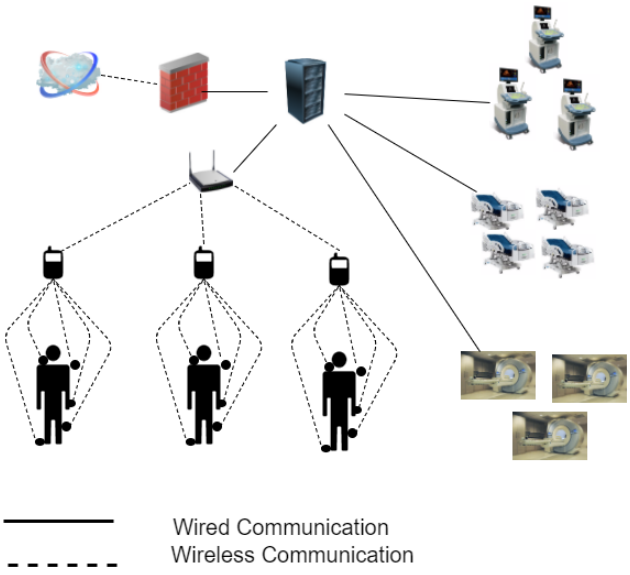

Figure 6.21: Simulated Smart Health Network Topology

In this phase of the experiment, we combine both classes of devices to form a complex network of multiple variants of medical sensing and imaging devices. As expected, we run the simulations under the same set of scenarios and observe the results presented in Figures 6.22 to 6.27.

The category B devices generate most of the network traffic. As a result, they ended

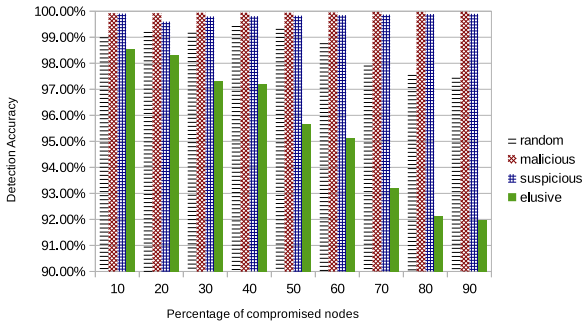

Figure 6.22: Network Intrusion Detection Accuracy (Smart Health Grid)

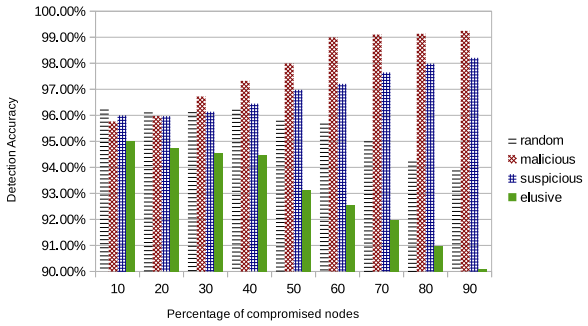

Figure 6.23: Device Intrusion Detection Accuracy (Smart Health Grid)

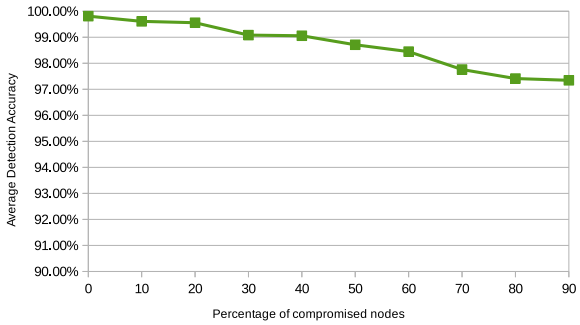

Figure 6.24: Network Intrusion Detection Average Accuracy (Smart Health Grid)

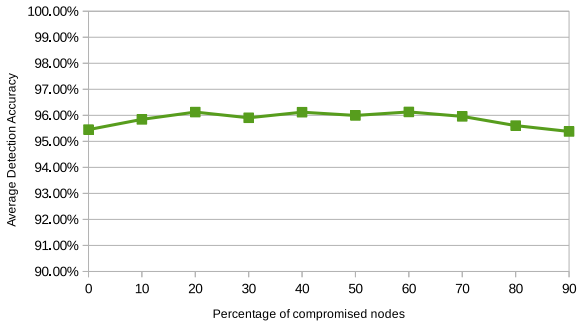

Figure 6.25: Device Intrusion Detection Average Accuracy (Smart Health Grid)

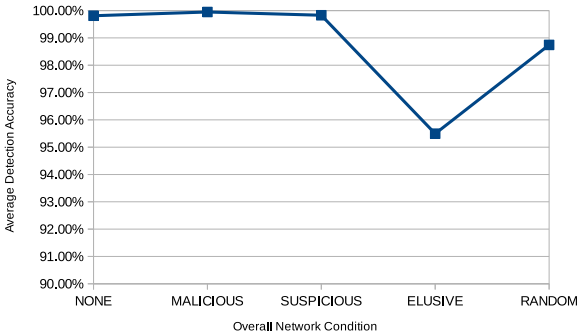

Figure 6.26: Network Intrusion Detection Accuracy Per Adversary (Smart Health Grid)

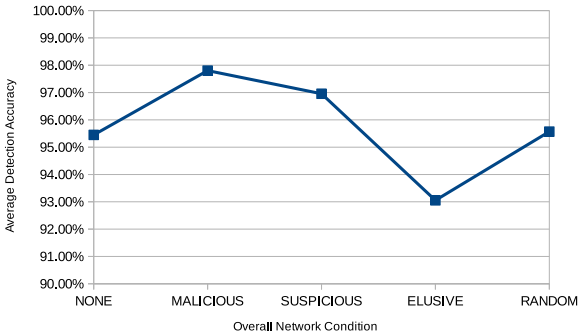

Figure 6.27: Device Intrusion Detection Accuracy Per Adversary (Smart Health Grid)

up as the major determinant in the network IDS accuracy trend. In the combined IoMT network, the system had a best case accuracy of 99.8% and 97.93% for network and device

intrusion detection respectively. The worst case accuracy was recorded to be 95.21% and 93.17% for both detection levels respectively.

Table 6.5 displays the comparative system performance for each kinds of network, in increasing order. The WBAN cluster had the best performance in device-level IDS and lowest in network IDS. Conversely, the category B devices performed best in network IDS, and worst in device IDS. While the combined IoMT network ended up mid-way for both types of detection.

Table 6.5: Comparative System Performance

Network Type	**Network IDS**	**Device IDS**
WBAN-Cluster	1	3
Category B Devices	3	1
Smart-Health-Grid	2	2

6.5.4 Scalability Test

In the final phase of the experiment, we test the system for scalability by running repeated simulations with increasing number of devices. In each run, we measure the average energy consumed by the cluster-head and observe its relationship to the number of devices. The result presented in Figure 6.28 shows that our system scales well with increasing number of devices. The difference in CH energy usage between a 2 and 5 cluster network (10 and 25 devices) is approximately 1 Joule. The single cluster network consumed a relatively low amount of energy due to the absence of extra communication amongst multiple cluster-heads.

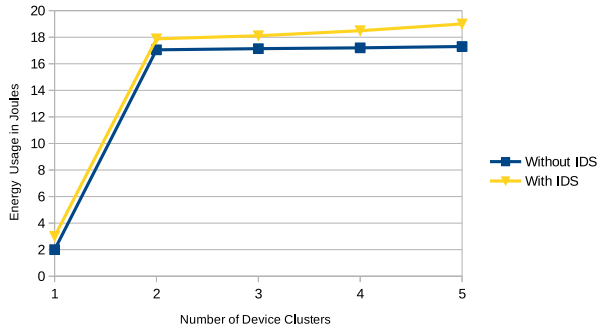

Figure 6.28: Energy Usage With Increasing Clusters (5 devices per Cluster)

We test a total of 72 varying simulations for each network type with an overall best and worst case detection accuracy of 99.9% and 92.91% respectively out of 216 simulations in total. The system also incurred an energy overhead between 5-7% for both network devices and cluster-heads. We believe our results proves the feasibility of our system in providing adequate security for IoMT networks at the sensing and diagnostic layer. Several challenges were encountered along the design of our proposed system, that warranted some of the techniques in the overall security protocol. These challenges are discussed in the next chapter along with their corresponding mitigation measures.

Chapter 7

POST-EVALUATION DISCUSSION

In this chapter, we discuss some challenges experienced during the development of our IDS and the corresponding measures taken to address them within the system. We also give possible insights into implementing our simulation on actual hardware.

7.1 Design Challenges

7.1.1 Execution of Detection Algorithms on Resource Constrained Networks

The primary challenge in our approach is the feasibility of executing computationally demanding algorithms on a resource constrained network such as the WBAN subset of the IoMT network. Both detection algorithms (network and device) are divided into a training, detection, and update phase. The former requires enough storage, memory, and processor clock cycles to process the training data set, thereby making it the most demanding in terms of space and run-time complexity. Detection involves processing and interpreting data relevant to the type of threats being diagnosed, which is order of magnitudes lesser the training data set. Secondly, data interpretation always require fairly constant time as long as data input format is maintained. This part of the algorithm is considered the least complex amongst the three phases. The update phase involves adaptation of current training model to emerging trends within the network and its constituent devices. Additional data is always included or used to supplant existing training set. The training phases is subsequently triggered to persist the new changes to the algorithm state. This requires more resources than the detection phase, however, it is still less demanding compared to the training phase.
We consider the two classes of devices within a WBAN which are the cluster-heads (CH), and sensor nodes (SN). A CH possesses higher computational and storage resources which is comparable to an average modern smart-phone. On the other hand, the SNs vary in computational power with the macro variants like FitBit possessing a decent amount of CPU speed and memory, and micro-chips variants having roughly 20 kB of memory, and 1 MB of flash

storage. In order to maximize the throughput of the intrusion detection algorithms with minimal footprint on available resources, we distribute the phases such that the computationally demanding training and update phase is executed on the higher-powered cluster-heads while actual detection is carried out locally on SNs.

Mobile agents are considered excellent candidates for inter-device distributed algorithms considering that they could maintain their execution code and state autonomously.

7.1.2 Parallel Execution of Network and Device Level Intrusion Detection

Both types of intrusion detection are mutually exclusive as we employ different algorithms and models for each one respectively. However, the system had to be designed such that they could be executed independently without conflicts, interference, or any other form of contentions. Most sensing devices within the network do not possess multi-core processors or operating systems complex enough to handle thread scheduling. As a result, we employ a simple FIFO (first-in first-out) serial process scheduling without any form of inter-leaving or interrupts that will require maintenance of process states.

Multiple agents could be deployed in parallel for network and device level intrusion detection respectively, as they are considered independent and autonomous in execution and termination. There is also no need to worry about terminal waits for either processes since agents are guaranteed to relinquish all resources (both computational and memory) before migrating to other nodes.

7.1.3 Handling Outcome of Detection Algorithms

To reduce the possibility of recurring false positives, we had to come up with a flexible approach towards interpreting the output from analysis of network/device data by respective detection algorithms. In most cases, the delineation between what is considered acceptable and anomalous within the network or device is not clearly defined, but rather blurred by several unaccounted/unobservable random factors. Results that fall between such gray areas require correlation with data from multiple devices to arrive at a definite conclusion on the state of the network/device.

To address this, we design a conflict resolution protocol in form of an intervention request

and response. It involves making a request from the sensor under analysis to the cluster-head for a larger scale correlation across the entire cluster. Details of the protocol is explained under the Special Agents subsection.

7.1.4 Mobile Agents Lost in Transit

Similar to data packets, agents are also susceptible to getting lost in transmission due to radio interference from neighboring devices. To address this, we create a checkpoint and recovery protocol that ensures agents lost in transit could be restored without any significant loss in their current state.

Within a cluster-head, we maintain a timer T for state dump inspection. Each agent dispatched from a particular CH C_a dumps their state information on C_a on completion of a single cycle of its itinerary. The state data dump includes all the parameter values specified in table 4.1. Each dump is tagged by agent ID and its timestamp. State dump is an update operation as it replaces the earlier tagged with the same ID. The timer T runs at set intervals to ensure agents have updated their state within time period I_s and I_c for sensors and CH agents respectively.

$$I_s = g_{max}(a_s + \alpha) \tag{1}$$

$$I_c = n_c(a_c + \beta) \tag{2}$$

Where g_{max} is the maximum number of nodes within a single sensor agent's itinerary, n_c is the number of cluster-heads within the entire network, a_s and a_c is the set aggregation period for the sensor and CH agents respectively, α and β are constants approximating the time it takes for detection algorithms to analyze and classify data for the sensor and CH agents respectively. Once the difference between the current time on CH C_a and the last agent's state dump timestamp exceeds the update window (I_s and I_c), agent is marked as missing and re-spawned from the CH.

7.2 Hardware Implementation

We discuss the implementation of the proposed system on actual hardware devices in terms of feasibility, techniques, possible challenges, and ameliorative measures.

7.2.1 System Feasibility

Typical sensor hardware could vary from implanted nano-devices to macro-wearables, each with varying computational and networking capacities. Based on our simulations, the typical size of a network intrusion detection agent ranges between 220-235 bytes, while a device level agent is around 350 bytes due to the extra state information for the coefficient matrix. While this may not be considered feasible on nano devices possessing less than 1KB of storage and memory, regular sensing devices with 20K RAM, and roughly 1 MB of flash storage are quite capable of handling such agents.

Secondly, low level sensing devices do not possess sophisticated operating systems that expose APIs for accessing granular hardware operational details. As a result, we ensure our system rely on directly computable hardware metrics like packet influx/efflux rate or sensor data sample value. Lastly, all mobile agent execution environments require some sort of listener that deflates the received agent packet before it could resume execution. Therefore the hardware environment is required to be programmable. Fortunately, majority of modern sensors and smart diagnostic devices are designed to be extensible via an open-source or proprietary SDKs.

Machine learning algorithms could be implemented from the scratch to avoid the verbose features that characterize the library editions which are designed for a more generalized purpose.

7.2.2 Techniques

Regardless of the method of implementation or platforms, all mobile agent based frameworks consist of the following entities in order as presented in Figure 7.1.

Spawning

This involves the instantiation of an agent code object with its required operational parameters of which must include its unique identifier, deflation function, and itinerary. Other bootstrapping functions such as the training of agent's machine learning or regression algorithm is carried out in this phase. In practice, this is done by mere instantiation of a class or struct.

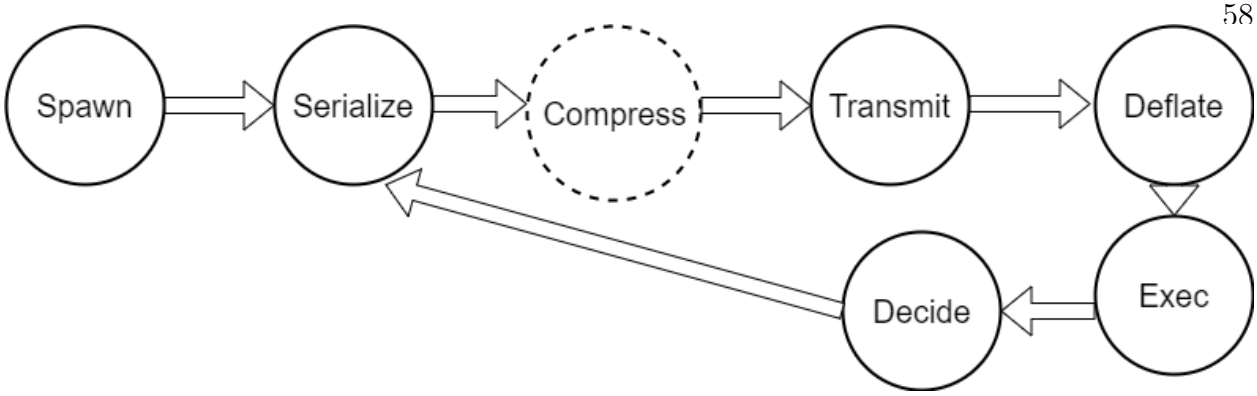

Figure 7.1: Mobile Agent Execution Workflow

Serialization

This is the process whereby the agent is converted from object code to a stream of bytes containing its current execution state, reference to its on-arrival function, destination node address, and other parameters required by the network protocol being used to transmit agent. Depending on the execution environment and programming language, this could be carried out in several ways. While devices running the JVM could implement this with the inherent class-loader serialization APIs, other devices running on C++ or C platforms require manual serialization or use of third party libraries like the Boost.

Compression

This is an optional part of the workflow as it also incurs extra computational overhead. There are several platform agnostic byte compression algorithms that could be used to carry out this procedure.

Transmission

Depending on the sophistication of the communication protocol between source and destination node, a large part of this process is usually abstracted into the transport and network layer of the protocol stack. For unreliable protocols like the WBAN and other UDP based variants, agents are split into parts if they exceed maximum unit packet size. The sequencing, acknowledgement, and retransmission of the agent parts are required to be manually

implemented at the application layer.

Deflation

This involves both decompression and de-serialization of agent packets at the receiving node. The executable code is reconstructed from the decompressed byte stream through an embedded interpreter. Again, JVM compatible devices provide inbuilt APIs for object code reconstruction, while other platforms require manual implementations or 3rd party libraries.

Resumption of Execution

On completion of agent deflation, the execution of agent could either be triggered immediately or deferred depending on the agent protocol. The deflation and initiation of the resumed execution is usually triggered by the agent listener program on the computing environment. The agent's on-arrival function is what usually gets invoked at this point.

Travel Decision

This is the point where agent decides on its next destination on completion of its execution phase. Agents could have static or adaptive itineraries. The static variants have a fixed trajectory, and report a failure once a node within its path is faulty or down. On the other hand, the adaptive counterparts could still possess a fixed set of destinations. However, they are capable of adjusting their paths based on varying network conditions. Our system employs the adaptive approach which is coded into the migration function.

Rinse, Wash, and Repeat

On successful selection of a destination, the entire process from serialization to travel decision making is repeated until the expiration of agent's designated lifetime.

7.2.3 Expected Challenges and Possible Mitigations

Although, the process of a mobile agent's life cycle is clearly defined, their implementation on real hardware still comes with certain challenges.

Device Heterogeneity

IoMT devices are relatively diverse in both hardware and software configurations. The task of deploying a unified mobile agent platform on such disparate devices becomes more challenging if they are programmed with different languages. Three possible scenarios exist when dealing with device heterogeneity. Differences in hardware structure, in device programming language, and a combination of both. A typical example of hardware differences could be a Java-compatible smart insulin pump from manufacturer A transmitting agents to a Java-compatible wearable band from manufacturer B. Different programming languages could mean a C++ programmable device trying to execute an agent transmitted from a Java-based counterpart.

Different variants of hardware with compatible programming SDKs could communicate seamlessly, once they have the appropriate network ports open for communication. However, devices with different programming SDKs require a language agnostic means (such as XML) of encoding an agent's state, and code base. A dynamic parser and code generator will have to be implemented for each programming language which runs each time an agent is received in the agreed unified representation. However, extra computational overhead is incurred by the continuous parsing and generation. A caching method could be employed which could store a previously received agent code along with the hash of its unified representation. The hash could be used for comparison with incoming agents in order to determine if its codebase exists in the cache, thereby eliminating the need for regenerating agent's code.

Disparity in Network Protocol

Although the IEEE 802.15/6 network standard has been defined for the WBAN sensors, its adoption has not been widely propagated. Secondly, the category B devices mostly communicate either through DICOM (for the imaging devices), Zigbee, or Ethernet-based TCP or UDP/IP. As a result, most IoMT networks are always composed of devices communicating through different protocols.

Addressing this will require an abstraction layer to be created, such that it defines a standard interface for required network operations.

Threading Model

Agents are typically dispatched to a separate asynchronous thread or even process entirely after deflation on traditional systems. Unlike traditional computers, most IoMT devices do not possess complex operating systems or multi-core processors capable of multitasking/multithreading.

Execution of agents could be deferred by holding them in memory until processor is idle.

7.3 Conclusion and Future Work

In this research, we design a mobile agent driven intrusion detection prototype for IoMT (Internet of Medical Things) networks. We employ both machine learning and polynomial regression for network and device level intrusion detection respectively. Different polynomial orders were tested for accuracy and efficiency by which we concluded that the 3rd order polynomial was most appropriate for approximating the model without incurring much resources. We run various set of simulations emulating different use-case scenarios and with network and device level detection executing in parallel. We got promising results in terms of accuracy and energy overhead. While we achieve a higher level of accuracy than the initial prototype, the energy incurred is slightly higher due to the inclusion of device level intrusion detection.

In future research, we plan to test our system on actual hardware. We gave some insights into how we intend to address the challenges that are expected when deploying a mobile agent based system on heterogeneous hardware. We also plan to examine alternative algorithms such as state graphs for analyzing change in sensor data trends as opposed to regression which deals with actual sensor data samples.

BIBLIOGRAPHY

[1] J.A. Stankovic A.D. Wood. Denial of service in sensor networks. *IEEE Computer Science*, (1):5462, October 2002.

[2] M Anandkumar, C Jayakumar, Arun Kumar, M Sushma, and R Vikraman. Intrusion Detection and Prevention of Node Replication Attacks in Wireless Body Area Sensor Network. *International Journal of UbiComp*, 3, July 2012.

[3] K Anandumar, C Jayaumar, Kumar Arun, M Sushma, and R Vikraman. Intrusion Detection and Prevention of Node Replication Attacks in Wireless Body Area Sensor Network. *International Journal of UbiComp (IJU*, 3(3), July 2012.

[4] J. S. Balasubramaniyan, J. O. Garcia-Fernandez, D. Isacoff, E. Spafford, and D. Zamboni. An architecture for intrusion detection using autonomous agents. In *Computer Security Applications Conference, 1998. Proceedings. 14th Annual*, pages 13–24, December 1998.

[5] Usha Banerjee, Ben Greenstein, and K Arya. Feedback Reliability Ratio of an Intrusion Detection System. *Journal of Information Security*, pages 238–244, June 2012.

[6] M. C. Bernardes and E. dos Santos Moreira. Implementation of an intrusion detection system based on mobile agents. In *International Symposium on Software Engineering for Parallel and Distributed Systems, 2000. Proceedings*, pages 158–164, 2000.

[7] W. Dean Bidgood, Steven C. Horii, Fred W. Prior, and Donald E. Van Syckle. Understanding and Using DICOM, the Data Interchange Standard for Biomedical Imaging. *Journal of the American Medical Informatics Association*, 4(3):199212, 1997.

[8] Bo Chen, Harry H. Cheng, and Joe Palen. Mobile-C: A Mobile Agent Platform for Mobile C-C++ Agents. *Softw. Pract. Exper.*, 36(15):1711–1733, December 2006.

[9] IBM Corp. IBM 2016 Cost of Data Breach Study - United States, September 2016.

[10] T. Dimitriou and K. Ioannis. Security issues in biomedical wireless sensor networks. In *2008 First International Symposium on Applied Sciences on Biomedical and Communication Technologies*, page 15, October 2008.

[11] Dimiter V. Dimitrov. Medical Internet of Things and Big Data in Healthcare. *Healthcare Informatics Research*, 22(3):156–163, July 2016.

[12] R. Gray, D. Kotz, S. Nog, D. Rus, and G. Cybenko. Mobile agents: the next generation in distributed computing. In *Proceedings of IEEE International Symposium on Parallel Algorithms Architecture Synthesis*, pages 8–24, March 1997.

[13] The IEEE. IEEE SA - 802.11n-2009 - IEEE Standard for Information technology– Local and metropolitan area networks– Specific requirements– Part 11: Wireless LAN Medium Access Control (MAC)and Physical Layer (PHY) Specifications Amendment 5: Enhancements for Higher Throughput.

[14] S. M. R. Islam, D. Kwak, M. H. Kabir, M. Hossain, and K. S. Kwak. The Internet of Things for Health Care: A Comprehensive Survey. *IEEE Access*, 3:678708, 2015.

[15] O. Kachirski and R. Guha. Intrusion detection using mobile agents in wireless ad hoc networks. In *IEEE Workshop on Knowledge Media Networking, 2002. Proceedings*, pages 153–158, 2002.

[16] G. Kambourakis, E. Klaoudatou, and S. Gritzalis. Securing Medical Sensor Environments: The CodeBlue Framework Case. In *The Second International Conference on Availability, Reliability and Security, 2007. ARES 2007*, page 637643, April 2007.

[17] P. Kannadiga and M. Zulkernine. DIDMA: a distributed intrusion detection system using mobile agents. In *Sixth International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing and First ACIS International Workshop on Self-Assembling Wireless Network*, pages 238–245, May 2005.

[18] Surraya Khanum, Muhammad Usman, and Alaa Alwabel. Mobile agent based hierarchical intrusion detection system in wireless sensor networks. January 2012.

[19] Pardeep Kumar and Hoon-Jae Lee. Security Issues in Healthcare Applications Using Wireless Medical Sensor Networks: A Survey. *Sensors (Basel, Switzerland)*, 12(1):5591, December 2011.

[20] K. S. Kwak, S. Ullah, and N. Ullah. An overview of IEEE 802.15.6 standard. In *2010 3rd International Symposium on Applied Sciences in Biomedical and Communication Technologies (ISABEL 2010)*, November 2010.

[21] Danny B. Lange, Mitsuru Oshima, Gnter Karjoth, and Kazuya Kosaka. Aglets: Programming Mobile Agents in Java. In *Proceedings of the International Conference on Worldwide Computing and Its Applications*, WWCA '97, pages 253–266, London, UK, UK, 1997. Springer-Verlag.

[22] G. Leroy, H. Chen, and T. C. Rindflesch. Smart and Connected Health [Guest editors' introduction]. *IEEE Intelligent Systems*, 29(3):2–5, May 2014.

[23] Nicole Lewis. Worries rise about security breaches in healthcare as endpoints expand. nov 2016.

[24] W. Li and X. Zhu. Recommendation-Based Trust Management in Body Area Networks for Mobile Healthcare. In *2014 IEEE 11th International Conference on Mobile Ad Hoc and Sensor Systems*, pages 515–516, October 2014.

[25] OpenSim Ltd. Omnet++ WBAN Projects | Wireless body area network Projects.

[26] Chen Y. Nasser N. SEEM: Secure and energy-efficient multipath routing protocol for wireless sensor networks, 09 2007.

[27] Hyacinth S. Nwana. Software agents: an overview. *The Knowledge Engineering Review*, 11(3):205–244, September 1996.

[28] Margaret Rouse. What is IoMT (Internet of Medical Things) or healthcare IoT? *IoT Agenda*, aug 2015.

[29] Michael Rushanan, Aviel D. Rubin, Denis Foo Kune, and Colleen M. Swanson. SoK: Security and Privacy in Implantable Medical Devices and Body Area Networks. In *Proceedings of the 2014 IEEE Symposium on Security and Privacy*, SP '14, pages 524–539, Washington, DC, USA, 2014. IEEE Computer Society.

[30] Osman Salem, Alexey Guerassimov, and Ahmed Mehaoua. Anomaly Detection in Medical Wireless Sensor Networks using SVM and Linear Regression Models. LIPADE Laboratory, University of Paris Descartes, France, June 2014.

[31] Raghav V. Sampangi, Saurabh Dey, Shalini R. Urs, and Srinivas Sampalli. A security suite for wireless body area networks. *arXiv:1202.2171 [cs]*, February 2012.

[32] A. S. Sangari and J. M. L. Manickam. Public key cryptosystem based security in wireless body area network. In *2014 International Conference on Circuit, Power and Computing Technologies (ICCPCT)*, pages 1609–1612, March 2014.

[33] G. Thamilarasu and Z. Ma. Autonomous mobile agent based intrusion detection framework in wireless body area networks. In *2015 IEEE 16th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, pages 1–3, June 2015.

[34] Geethapriya Thamilarasu. iDetect: an intelligent intrusion detection system for wireless body area networks. *International Journal of Security and Networks*, 11(1/2):82, 2016.

[35] Mohsen Toorani. On Vulnerabilities of the Security Association in the IEEE 802.15.6 Standard. *arXiv:1501.02601 [cs]*, 8976:245–260, 2015.

[36] Jacob Ulvila and John Gaffney Jr. Evaluation of Intrusion Detection Systems. *Journal of Research of the National Institute of Standards and Technology*, 108:453–473, November 2003.

[37] Javier Vales-Alonso, Pablo Lpez-Matencio, Francisco J. Gonzalez-Castao, Honorio Navarro-Helln, Pedro J. Baos-Guirao, Francisco J. Prez-Martnez, Rafael P. Martnez-lvarez, Daniel Gonzlez-Jimnez, Felipe Gil-Castieira, and Richard Duro-Fernndez. Ambient Intelligence Systems for Personalized Sport Training. *Sensors*, 10(3):2359–2385, March 2010.

[38] H. Wang, H. Fang, L. Xing, and M. Chen. An Integrated Biometric-Based Security Framework Using Wavelet-Domain HMM in Wireless Body Area Networks (WBAN). In *2011 IEEE International Conference on Communications (ICC)*, pages 1–5, June 2011.

[39] F. Xu, Z. Qin, C. C. Tan, B. Wang, and Q. Li. IMDGuard: Securing implantable medical devices with the external wearable guardian. In *2011 Proceedings IEEE INFOCOM*, pages 1862–1870, April 2011.

[40] Yifan Zhang, Yunxin Liu, Li Zhuang, Xuanzhe Liu, Feng Zhao, and Qun Li. Accurate CPU Power Modeling for Multicore Smartphones. *Microsoft Research*, February 2015.

[41] Shao-Chun Zhong, Qing-Feng Song, Xiao-Chun Cheng, and Yan Zhang. A safe mobile agent system for distributed intrusion detection. In *2003 International Conference on Machine Learning and Cybernetics*, volume 4, pages 2009–2014 Vol.4, November 2003.