Artificial Immune Systems for Detecting Unknown Malware in the IoT

Author:
Hadeel Saleh ALRUBAYYI

A thesis submitted in fulfillment
of the requirements for the degree of
Doctor of Philosophy

School of Electronic Engineering and Computer Science
Queen Mary University of London
London, United Kingdom

January 2023

Artificial Immune Systems for Detecting Unknown Malware in the IoT, January 2023

Author:
Hadeel Saleh Alrubayyi

Institute:
Queen Mary University of London, London, United Kingdom

STATEMENT OF AUTHORSHIP

I, Hadeel Saleh Saleh Alrubayyi, confirm that the research included within this thesis is my own work or that where it has been carried out in collaboration with, or supported by others, that this is duly acknowledged below and my contribution indicated. Previously published material is also acknowledged below. I attest that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge break any UK law, infringe any third party's copyright or other Intellectual Property Right, or contain any confidential material.

I accept that the College has the right to use plagiarism detection software to check the electronic version of the thesis.

I confirm that this thesis has not been previously submitted for the award of a degree by this or any other university.

The copyright of this thesis rests with the author and no quotation from it or information derived from it may be published without the prior written consent of the author.

Details of collaboration and publications:

- Alrubayyi, H., Goteng, G., Jaber, M., & Kelly, J. (2021). Challenges of Malware Detection in the IoT and a Review of Artificial Immune System Approaches. Journal of Sensor and Actuator Networks, 10(4), 61. [1].
- Alrubayyi, H., Goteng, G., Jaber, M., & Kelly, J. (2021, May). A novel negative and positive selection algorithm to detect unknown malware in the IoT. In IEEE INFOCOM 2021-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS) , IEEE., 2021, pp. 1-6. [2].

Hadeel Saleh Alrubayyi January 2023

DEDICATION

This thesis work is dedicated to my best friend, Sumaya, who has been a constant source of support during the challenges of graduate school and life.

ACKNOWLEDGEMENTS

I believe that the Ph.D. experience is very unique for each graduate student, for sure mine is! Not only going through the graduate school experience but also living through the difficult times of the pandemic. The pandemic has made this experience special on its own, throwing extra challenges into the equation. There were sleepless and hopeless nights, as well as happiness and satisfaction. This journey has come to an end as I am taking the last lap, which seems to be the most difficult one of all, emotionally speaking. For that, I would like to thank the people without whom I would not have been able to complete this research, and without whom I would not have made it through my Ph.D. degree!

My first thanks go to my first supervisor, Dr. Gokop Goteng, who has been the greatest guide and support throughout my journey. He was always there to help and was patient with me in difficult times.

To Dr. Mona Jaber, my second supervisor, I would like to say Thank you! Dr. Mona has been exceptional in the ways she has shown support throughout my journey. She has shown ambition, passion, and of all leadership.

Also, I would like to thank Dr. James Kelly, my independent assessor, for his contributions to this project.

I would like to thank everyone at Queen Mary University of London, specifically in the Department of Electronic Engineering and Computer Science, who was part of this project and provided help of any sort. Special thanks go to Melissa Yeo, who was a great support at all stages of my Ph.D.

To my mom, Norah, Thank you for all the unconditional love, the patience, and unwavering faith you have in me. Thank you for letting me grow my own wings and fly!

Thanks to my dad, Saleh, my sisters, Alaa, Renad and Hend, and my brother, Zaid, for all their love and support.

My biggest thanks goes to my soulmate Sumaya! There are no words that can describe how thankful I am to have such an amazing friend in my life. You are always the first one to stand right next to me in both happy and difficult times. For that I am grateful, and I love you so much!

I would like to thank my friend Manal for being by my side all these years, through all the good times and the bad. Truly great friends are hard to find!

Many thanks go to my best friends in London, Benny and Courtney! Truly thankful for making the experience of doing my Ph.D. during the pandemic so memorable. Love you guys!

Many thanks go to Agab, who was my first cheerleader ever! Here I am, becoming a Dr, as you always said "it is going to happen someday". It is today!

I would like to thank my friend Abdullah for his continuous support and encouragement in the past years.

Tghreed, Suzan, Sarah, and Deema, thanks for always believing in me, love you always!

Many thanks to all my friends in London, especially Anja and Sakshi! Thanks for all the trips, the laughs, and the early morning struggles.

ABSTRACT

With the expansion of the digital world, the number of the Internet of Things (IoT) devices is evolving dramatically. IoT devices have limited computational power and small memory. Also, they are not part of traditional computer networks. Consequently, existing and often complex security methods are unsuitable for malware detection in IoT networks. This has become a significant concern in the advent of increasingly unpredictable and innovative cyber-attacks. In this context, artificial immune systems (AIS) have emerged as effective IoT malware detection mechanisms with low computational requirements. In this research, we present a critical analysis to highlight the limitations of the AIS state-of-the-art solutions and identify promising research directions. Next, we propose Negative-Positive-Selection (NPS) method, which is an AIS-based for malware detection. The NPS is suitable for IoT's computation restrictions and security challenges. The NPS performance is benchmarked against the state-of-the-art using multiple real-time datasets. The simulation results show a 21% improvement in malware detection and a 65% reduction in the number of detectors. Then, we examine AIS solutions' potential gains and limitations under realistic implementation scenarios. We design a framework to mimic real-life IoT systems. The objective is to evaluate the method's lightweight, fault tolerance, and detection performance with regard to the system constraints. We demonstrate that AIS solutions successfully detect unknown malware in the most challenging IoT environment in terms of memory capacity and processing power. Furthermore, the systemic results with different system architectures reveal the AIS solutions' ability to transfer learning between IoT devices. Transfer learning is a critical feature in the presence of highly constrained devices in the network. More importantly, we highlight that the simulation environment cannot be taken at face value. In reality, AIS malware detection accuracy for IoT systems is likely to be close to 10% worse than simulation results, as indicated by the study results.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ACO	Ant Colony Optimization
AI	Artificial Intelligence
AIN	Artificial Immune Networks
AIS	Artificial Immune Systems
ANNs	Artificial Neural Networks
ATT	All Attack Types
AWA	Artificial Awareness Architecture
AWS	Amazon Web Services
Amazon EC2	Amazon Elastic Compute Cloud
Amazon VPC	Virtual Private Cloud
CA	Certificate Authority
CFG	Control Flow Graph
COVID 19	Coronavirus 2019
CPU	Central Processing Unit
DCA	Dendritic Cell Algorithm
DNS	Domain Name System
DOS	Denial of Service
FNN	Feed-Forward ANN
FN	False Negative
FP	False Positive
IPv6	Internet Protocol Version 6
IRPs	I/O request packets
IoMT	Internet of Medical Things
IoT	Internet of Things
MCC	Matthews Correlation Coefficient
NDS	Negative Detector Set
NHS	National Health Services
NPS	Negative-Positive Selection

NSNN	Negative Selection and Neural Networks
OS	Operating System
PDS	Positive Detector Set
PSCA	Positive Selection Classification Algorithm
PSO	Particle Swarm Optimization
RCBM	R-Continuous Bit Matching
SNN	Self-Normalizing Neural Networks
SSL	Secure Socket Layer
TLS	Transport Layer Security
TN	True Negative
TP	True Positive
UNSW	The University of New South Wales

1 INTRODUCTION

1.1 RESEARCH INTRODUCTION

Today's world is more connected than ever before. Societies are reliant on technology which has become inextricable from their daily lives. Smart cities, smart homes, and e-government are applications used to improve life quality for communities [3]. The internet of things (IoT) is a system of connected physical objects/things embedded with small sensors with small memory capacity and low processing power. IoT devices are able to collect and exchange data over the internet, usually without human interaction [4]. Unlike the internet, where an internet protocol (IP) address is used to connect each device to the internet for functionality, IoT devices can still function and collect data without an internet connection. For instance, a smartwatch could collect health data, such as heart rate, without an internet connection. IoT systems provide inventive solutions to daily life challenges. For instance, with the increasing need to develop a more cost-efficient and personalized healthcare system, IoT devices play a massive role in achieving this vision [5]. Today's situation due to the Coronavirus 2019 (COVID-19) pandemic has accelerated the adoption of such technologies in various ways. For instance, e-health applications are developed to support the depleted healthcare staff and systems [6]. Moreover, IoT devices are employed to improve energy efficiency and reduce environmental impacts of energy use [7].

The continuous growth of IoT systems and the direct interaction with the physical world make it an excellent target for cybercrimes [8]. For instance, in IoMT (Internet of Medical Things) systems, a high volume of patients' data is exchanged, raising serious security concerns. Consequently, many standards are established to address these issues, such as implementing a secure socket layer (SSL) and transport layer security (TLS) to prevent leakage of confidential information [9]. Cybercrime is any illegal action committed against computers or traditional crimes targeting individuals using the internet [10]. Getting hold of confidential information, such as credit card information, was the motivation behind a data breach targeting EasyJet, the airline company [11]. Evidently, a weak security configuration enables hackers to get access to critical data [12]. One of the major distributed denial of service cyberattacks was in 2016, targeting the Domain Name System provider (DYN). The type of malware used for the attack, which used IoT devices rather than computers, resulted in significant services being unavailable for many users in different countries. [13] Health, educational, financial, and governance institutes were affected, which makes malware attacks a global risk factor.

Malware attack is one of the significant security threats in the IoT, and malware detection, specifically detecting unknown malware files, is one of the ongoing investigations. IoT devices have constrained resources, such as small memory and processing powers, which makes applying security solutions challenging. Also, IoT architecture allows minimal control for the user over the IoT device, which leads to major security concerns [14]. In addition to the security challenges in the IoT systems, hackers became more creative and use innovative tools to form an attack [15]. Intrusion detection systems (IDS) are installed to prevent such attacks. IDS works either as a network-based intrusion detection system (NIDS) or as a host-based intrusion detection system (HIDS) [16]. NIDS detects attacks over the network, ex., a network port, while HIDS detects attacks within the system, ex. An infected operating system. Malware detection generally is done in two phases, the malware analysis phase and the malware detection phase. In the malware analysis phase, static, dynamic, and hybrid analyses are the three main methods to extract malware file features. After analyzing the file, the results are passed to the next phase, which is the malware detection phase. Three different techniques are used to detect malware files, signature-based, behavioral-based, and specification-based techniques. The signature-based technique reads the file signature and runs it against an existing database, which makes it unable to detect new malware files. The behavioral-based and the specification-based techniques monitor the file behavior in general or read some of its features, such as the application interface, without reading its signature. The behavioral-based and the specification-based techniques are proven to be able to detect unknown malware files; however, they are computationally expensive [17].

Artificial Immune System (AIS) methods are inspired by the human immune system methodology in fighting attacks [18]. They are proven to be adaptive, distributed, robust, and not computationally expensive, which makes them suitable for securing the IoT. This research focuses on applying AIS strategies, inspired by adaptive immunity in the human body, to computing algorithms to secure the internet of things. The main goal is to detect unknown malware attacks to secure the internet of things systems.

1.1.1 *Research Motivation*

Since the IoT environment is dynamic and interconnective, a high risk of malware and intrusion attacks is presented. Such an attack could cause massive damage to the network devices and the system data. The primary motivation of this project is designing a method to enhance the detection performance with less false negative detection for unknown malware in the IoT systems. Given the IoT systems properties, we design the method to be lightweight, adaptive and distributed to meet the system requirements.

1.1.2 *Research Questions*

- What are the main security challenges in IoT systems, and what are the IoT-specific requirements to overcome these challenges?

  Chapter 2

- What are the limitations and restrictions of the current malware detection solutions?

  Chapter 4

- How to design a security method to detect unknown malware that overcomes the challenges of IoT systems' requirements and AIS solutions' limitations?

  Chapter 5

- What are the opportunities and challenges of running AIS solutions in realistic IoT systems?

  Chapter 6

- How to improve the efficiency of AIS solutions for unknown malware detection in constrained real-time IoT systems?

  Chapter 7

1.1.3 *Research Objectives*

- IoT perspective to cybersecurity threats and solutions requirements.
- Design a malware detection algorithm that is lightweight and can detect unknown malware attacks.
- Validate the results of the malware detection algorithm using multiple real-time datasets.
- Investigate the impact of IoT hardware limitations on the efficacy of the malware detection algorithm.
- Design an AWS-enabled validation framework for the evaluation of AIS malware detection solutions, under realistic architecture and characteristics.
- Design real-life system scenarios to investigate the algorithm's detection performance, lightweight, distribution, adaptivity, and fault tolerance capabilities with respect to system constraints.
- Design the first trial of transfer learning within and across the IoT systems to combat the constrained memory in IoT devices.

1.1.4 *Research Challenges*

(a) **IoT in a Dynamic Environment**
The IoT paradigm is dynamic and lightweight with constrained resources. These devices often have limited computational powers and small memory. IoT devices are interconnective, which means they are connected directly to the cloud and/or other IoT devices. They are heterogeneous, which means the connected IoT devices could be run on different platforms with different requirements and specifications. Furthermore, connected IoT devices can exchange services within the constraints of things. Finally, the increasing number of IoT devices leads to generating an enormous scale of data in a massive-scale network. The IoT System architecture consists of three main layers: the perception layer, which is the physical layer, the network layer, which is responsible for access control between the IoT device and the cloud, and the application layer, which is the front-end layer. Each layer is a target for a different set of attacks. For instance, malicious node injection and battery draining from security threats to the physical

layer. Man in the middle and denial of service are threats to the network layers. Malware attacks are one of the main threats to the application layer. For all these reasons, securing IoT devices is challenging. Applying traditional security solutions is very computationally expensive and applying minimal security measures is very high risk. Therefore, it is critical to design IoT-specific security solutions.

(b) **Artificial Immune Systems Characteristics**
Even though many researchers implemented and used AIS solutions to secure the IoT, it still has its own challenges. For instance, applying negative selection algorithms using different formats and data representations to secure the IoT has improved the depth of detection capabilities, yet high false negative is one of the main limitations of this technique. Also, the negative selection algorithm is unsuitable for a dense environment and cannot cope with changing dynamics of the system with respect to time. Consequently, it has some latency in response time. Furthermore, few implementations have been done using the positive selection algorithm for malware detection in the IoT. The reason is that this method has many limitations when used as the primary technique in malware detection, one of which is high false positives.

1.2 THESIS STRUCTURE

Chapter 2: IoT security challenges and vulnerabilities to malware attacks introduction

Chapter 3: Artificial Immune Systems methods- background

Chapter 4: A critical evaluation and analysis of the state-of-the-art of AIS solutions for malware detection in the IoT

Chapter 5: A novel AIS-based method (NPS) to secure the IoT - results analysis in comparison to the state-of-the-art

Chapter 6: Implementing the NPS in realistic IoT systems Using AWS

Chapter 7: Implementing the NPS in realistic IoT systems Using AWS - transfer learning across IoT systems challenges and opportunities

Chapter 8: Conclusion and future research directions

1.3 CONTRIBUTIONS TO KNOWLEDGE AND ASSOCIATED PUBLICATION

- A critical analysis of detection mechanisms in the context of IoT with focus on AIS (Chapter 4).
  Alrubayyi, H., Goteng, G., Jaber, M., & Kelly, J. (2021). Challenges of Malware Detection in the IoT and a Review of Artificial Immune System Approaches. Journal of Sensor and Actuator Networks, 10(4), 61. [1].

- A new AIS method that addresses the IoT challenges and outperforms the state-of-the-art (Chapter 5).
  Alrubayyi, H., Goteng, G., Jaber, M., & Kelly, J. (2021, May). A novel negative and positive selection algorithm to detect unknown malware in the IoT. In IEEE INFOCOM 2021-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS) , IEEE., 2021, pp. 1-6. [2].

1.4 TALKS AND PRESENTATIONS

- Enhancing AWS IoT Gateway Security Using Adaptive Immunology. WMC Lab at QMUL (April 2019)

- AWS re/Start and AWS Academy: closing the digital skills gap. Institute of Coding Conference (March 2020)

- AIS for Malware Detection in the IoT Using AWS: Knowledge Transfer Across Networks Study. IEEE WIE UKI Career Development Day (September 2022).

- IoT Security Challenges and AIS Advances in Detecting Unknown Malware. IEEE WIE UKI Ambassadors Programme, Seventh Event of Early Career Talk (October 2022)

2 IOT SECURITY CHALLENGES AND VULNERABILITIES TO MALWARE ATTACKS

2.1 INTRODUCTION TO IOT SYSTEMS

IoT is a system of interconnected machines with unique identifier numbers that can communicate and share data within a network without human interaction. The IoT system consists of devices (often referred to as IoT devices) with unique identifiers that integrate seamlessly into the information network by using intelligent interfaces [19].

2.1.1 *Internet of Things Characteristics*

These IoT devices are physical entities interacting to form the IoT system with essential features as follows (see Figure. 2.1):

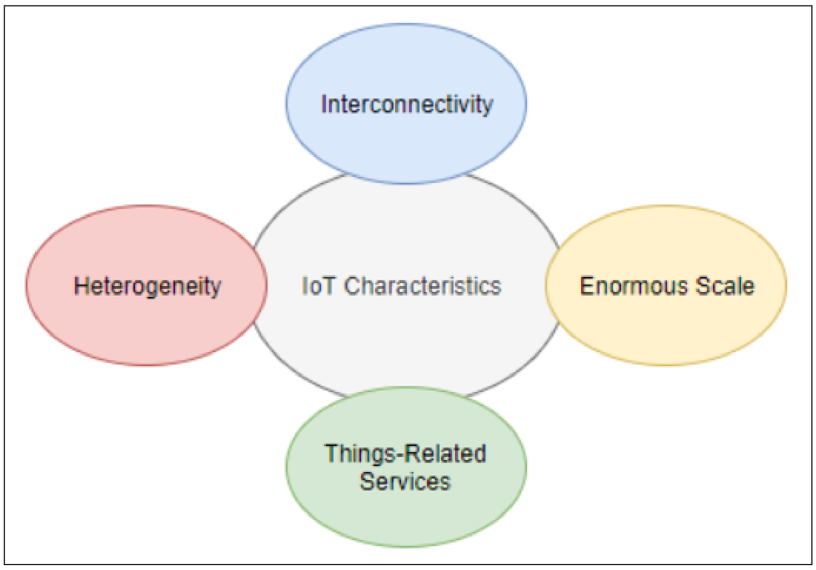


Figure 2.1: IoT Devices Unique and Challenging Characteristics

- Interconnectivity: Interconnectivity is about connecting the device to the cloud and/or other devices. The connectivity is needed to enable the control of the device remotely but mainly to access the data collected by the IoT device's sensor(s). For example, an IoMT device for heart disease prediction is remotely controlled to monitor patients' heart rate [20]. The health parameters are collected in real-time and transmitted to a data center in the cloud.

- Heterogeneity: The IoT devices are heterogeneous as these may be built on different platforms and have different specifications. Various hardware, such as a simple sensor to monitor the heart rate in [20], or a data center built on the cloud, could be supplied by different vendors. These integrated IoT devices could use different security measures, which leads to a lack of standardization in the network. Each connected device could use different security protocols with its security bugs and limitations, which expose the system to different ways of hacking.

- Things-related services: In the IoT environment, devices are capable of exchanging services within the constraints of things. Since the communication between different IoT devices is not controlled by a central processor/human, this could form a serious threat. A malicious device is disguised as an accepted IoT device that could start disturbing other devices, for example, by installing malicious files.

- Enormous scale: The number of IoT devices is increasing exponentially and is generating an unprecedented amount of data. The expected number of IoT devices by 2025 is between 25 billion and 50 billion [21]. The scale is simply enormous, and data privacy and integrity are critical challenges in massive-scale networks. For instance, IoMT-based COVID-19 applications are creating massive amounts of real-time data that gets stored in the cloud. However, as the generated data continues to increase, the network pressure increases, which might lead to occurrences of erroneous interpretations [22].

2.1.2 *IoT Systems Architecture*

The IoT architecture consists of three main layers, the perception layer, the network layer, and the application layer.

- Perception layer: This first layer is a physical layer that involves connected smart devices and sensors to collect data, such as temperature, humidity, and sound.
- Network layer: A middle layer is responsible for securing the connection between the perception layer and the cloud via routers and gateways.
- Application layer: This front-end layer is responsible for delivering specific services to endpoints, such as end-users, servers, and the cloud.

The gateway layer, which sits between perception and network layers in IoT architecture, is introduced by authors in [23, 24]. The main role of the gateway layer is to process and classify the enormous amount of data generated by smart devices and sensors in the perception layer. This layer has a significant value from a security perspective. Since it is the main link between the IoT devices and the network layer, which is mainly linked to the cloud, it is vulnerable to many security threats. The gateway layer processes a massive amount of system data, which makes it a target to malicious files aiming to harm the system or gain access to valuable data. Consequently, securing this layer improves the safety and reliability of both the perception layer, where the limited capacity IoT devices are installed, and the main network hub, the cloud.

2.2 IOT SECURITY THREATS

Cybercrimes involve a wide range of illegal activities that could target different layers within the IoT architecture and be categorized as follows [25]:

(a) Physical attacks that target the hardware level, some of which:

- interference on Radiofrequency Identification (RFIDs): RFID works by using radio communication to identify hardware and send noise signals to interfere with radio communication causing a denial of service,
- malicious Node Injection: Connecting a malicious node between two interconnected nodes and injecting the communication messages with falsified information,
- battery Draining: Maximizing the node's power consumption will break the node, minimize its lifetime, and shut it down due to malicious control of the device. A physical attack could use a node injection to send false messages to vulnerable nodes that would cause them to drain the battery. Such messages may claim that

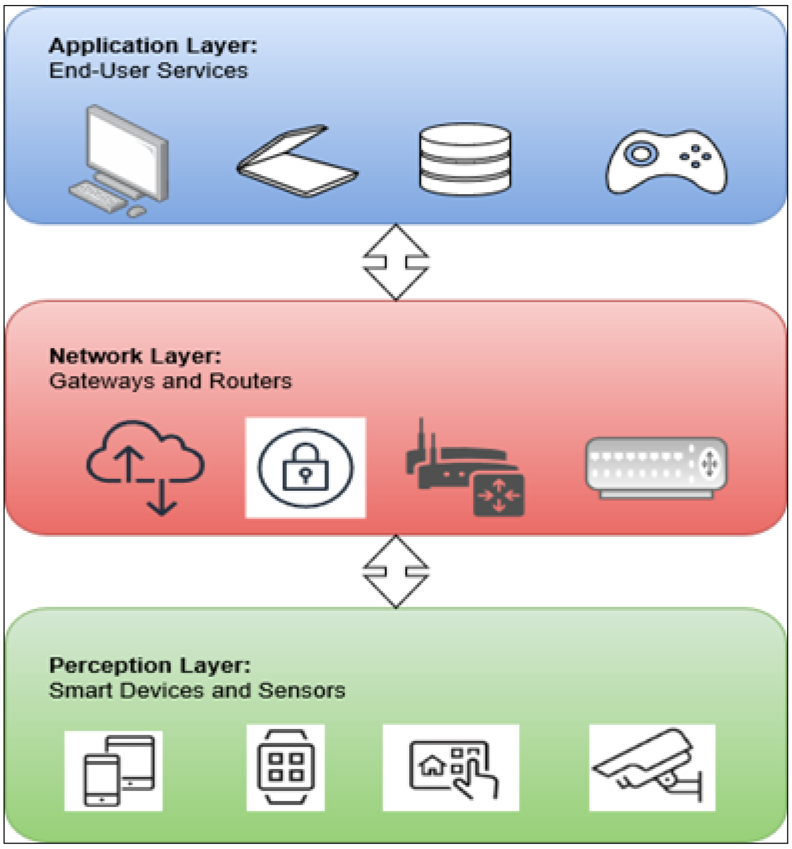


Figure 2.2: IoT Systems Architecture

(i) the reception is bad, pushing the transmission power to increase and increase retransmissions until the battery is dead,

(ii) claim that the server requires data to be uploaded every millisecond instead of an hour, thus consuming the battery power at an accelerated rate of 10^6.

(b) Network attacks, some of which:

- eavesdropping attack: Intruders intercept network information by examining messages between nodes, such as gateways, and get unauthorized access to system information,

- RFID Spoofing: Obtain transmitted data from an RFID tag and inject the system with falsified information,

- RFID cloning: Copying data from existing tags to a new one without the original ID to inject the signal with falsified information,

- man in the middle: Hack the communication between two nodes to get access to network information,

- denial of service: Overwhelming the system so it is unavailable to the user.

(c) Software attacks that target the application layer, some of which:

- Phishing: Obtaining access to sensitive and private data records, such as usernames, passwords, and credit card information, through email spoofing or fake websites.

- Virus, worms, trojan Horse, and spyware: Malicious software that attacks the system to obtain sensitive information, harm the hardware level or the software level.

Given the IoT characteristics discussed in Section 2.1, it is very challenging to address the security threats presented in Section 2.2 holistically. This is further elaborated on in the next section.

2.3 IOT SECURITY CHALLENGES

The IoT involves smart devices and sensors, some of which use non-chargeable batteries, making battery life one of the predominant challenges in IoT security. Running security rules will drain the battery resources; applying minimum security requirement measures is not a smart idea, especially if these devices are responsible for collecting sensitive information. Increasing battery size and capacity is not a solution as well because these devices are designed to be lightweight. Domain Name System (DNS), which is used to identify objects and their attributes, is another IoT security challenge; data integrity is problematic here due to the possibility of being hacked by a man in the middle or a DNS cash poisoning attack. In addition to device limitation and object identification, device authentication and authorization is one of the IoT security challenges. Issuing certificates for each object in the IoT is extremely challenging due to the number of connected objects and not having a global root certificate authority (CA). The threat of malware attacks arises in IoT due to these security challenges. Antivirus is the main line of defense to detect known malware in a real-time paradigm. However, the traditional security solutions /have not been efficient and do not provide decentralized and strong security solutions in the IoT [26]. Due to the IoT device limitation and computing power, shifting similar solutions from traditional platforms to IoT might not be affordable [27]. Battery size and expected durability is a challenge that makes the implementation of security measures more limited as it has to be energy efficient as well as secure. Moreover, in IoT systems, network resources are integrated into devices that were never previously anticipated to be part of computer networks [24]. Integrating IoT devices into traditional networks introduces a new paradigm of security. The integrated system inherent the traditional network security issues besides the ones targeting the IoT devices [26]. Consequently, using traditional security measures is not enough to endow IoT systems with malware detection capabilities.

2.4 MALWARE ATTACKS ANALYSIS AND DETECTION TECHNIQUES

Based on the analysis presented in Section 2.3, malware is a major security threat to the IoT, and detecting unknown malware is one of the key challenges. First, IoT devices' limitations form a significant challenge when aiming to apply security solutions. Second, introducing new ways of integrating network resources into devices that were not part of a traditional computer network, such as smart homes, opens the door to many "unknown" security threats, such as newly developed malware files. For these reasons, traditional malware detection mechanisms are unsuitable for the IoT environment.

This section presents a brief background of malware followed by classification based on their reproduction behavior and action. We also examine IoT-related malware attacks, which have significantly increased in recent years and require imminent attention. Next, we present a study of existing methods for analyzing and detecting malware in general and how they apply to IoT systems.

2.4.1 *Malware Attacks*

Malware is malicious software that gets executed within the system without the user's permission and has harmful intentions. Black hats, hackers, and crackers are all names for malware writers and developers, who have different intentions when creating this malicious executable software, some of which are internal threats, governance purposes, and competitor's spies. In the past, the malware was written using simple techniques, and for simple reasons, which we could call "traditional" [28]. Nowadays, hackers have more resources and technical knowledge to develop more complex malware for one or multiple reasons, which we call "next-generation malware". The Figure below (Fig. 2.3) shows the predicted damage cost of malware attacks in the IoT networks.

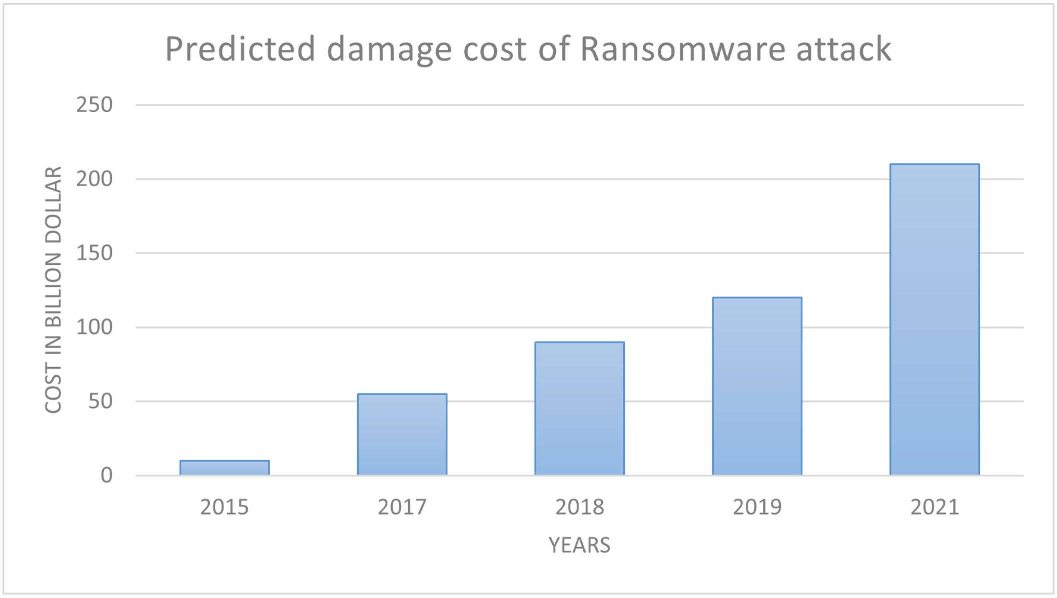


Figure 2.3: Predicted Damaged Cost of Malware Attacks [29]

2.4.2 *Malware Analysis Techniques*

Malware analysis techniques are essential to developing effective malware detection methods. These techniques involve the analysis of the process and functionality of the malware to build a suitable defense method. Three main malware analysis techniques achieve the same goal of determining how the malware works and how the attack will affect the network (see Figure 2.4).

(a) Static analysis, also called code analysis: In this technique, the infected file is inspected and analyzed without executing it. Low-level information is extracted, such as the control flow graph (CFG), data flow graph, and system calls. Static analysis is fast at analyzing data and safe to use; also, it has a low level of false positives, which means a higher detection rate. Moreover, the static analysis tracks all possible paths, which gives it a global view; however, it fails in detecting unknown malware using code obfuscation [30].

(b) Dynamic analysis, also called behavioral analysis: In dynamic analysis, the infected file is inspected during execution, which is usually conducted on an invisible virtual machine, so the malware file does not change its behaviors. Dynamic analysis is time-consuming and vulnerable, and it can only detect a few paths based on triggered files. Furthermore, it is neither safe nor fast, and it suffers from a high level of false positives. However, dynamic analysis is known for its good performance in detecting new and unknown malware [31].

(c) Hybrid analysis: This technique was designed to overcome the challenges and limitations of the previous two techniques. First, it analyzes the signature descriptions of any malware code and then combines that with other dynamic parameters to improve the analysis of malware [32].

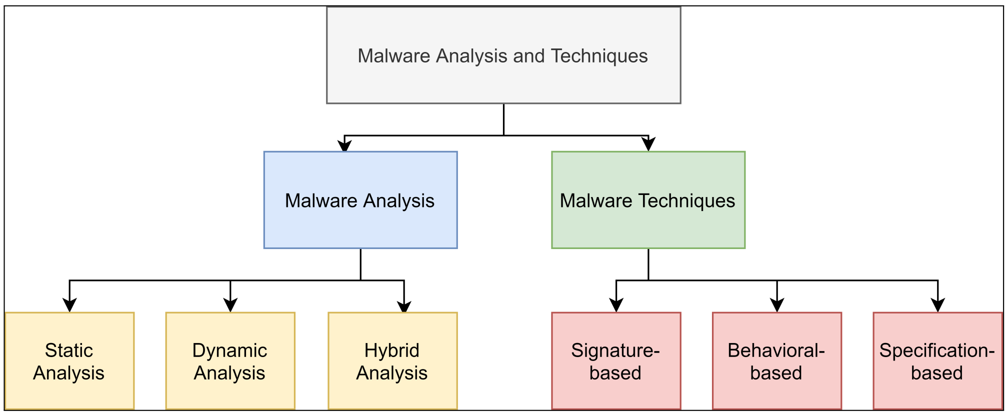


Figure 2.4: Malware Files Analysis and Detection Methods

The connection in IoT networks is currently enabled via cloud services. Static, dynamic, and hybrid malware analyses are mostly applied in the cloud to protect IoT devices.

2.4.3 *Malware Detection Techniques*

Three main methods are used in malware detection: signature-based detection technique, behavior-based detection technique, and specification-based detection technique [33].

(a) Signature-based method: In the signature-based technique, files are analyzed and compared to an existing list, and if they are listed in the list, they are classified as malware. This way is ineffective in recognizing all malware that enters the network because some malware is encrypted, so extracting the signature takes time and a lot of processing energy. Also, it is not effective for new or unknown malware. Some of the signature-based technique applications are presented in [34–36].

(b) Behavioral-based method: This technique monitors the program behavior rather than reading its signature. This technique follows three steps:

- data collector which collects information about the program
- interpreter which converts collected data to intermediate representations
- matcher which compares intermediate representations with behavior signatures.

There are two approaches to this technique:

- simulates the behavior of legitimate programs and compares any new program to that model. This approach works to detect most malware, even new ones. However, it is hard to implement because of the different behaviors of each program in the network. For example, a video reader will use different services than a mail or a web client
- simulates the behavior of known malware and compare them to new programs, which means new malware could not be identified.

Some of the Behavioral-based technique applications are presented in [37–39]

(c) Specification-based method: This technique was introduced to overcome the disadvantages and limitations that the first two techniques have. This technique uses different features for malware detection, some of which:

- API Calls: Hofmeyr et al. were among the first to propose using application interface and system calls sequences for malware detection [40]
- OpCode: Executable files are made of a series of assembly codes, and in this method, researchers used this operational code to detect malware [41]
- N-Grams: This method uses executable programs' binary codes for malware detection [42]
- Control flow graph (CFG): It is a graph that shows the control flow of programs, and it has been used to analyze malware behavior [43]
- Hybrid feature: In this machine learning method, researchers combine different techniques for malware detection to get better results. For example, Eskandari et al. in [44] used CFG and API calls for metamorphic malware detection.

2.5 MALWARE IN THE IOT

The malware detection techniques presented in the previous section have been followed to implement malware detection methods in the IoT; for instance, SVELTE, which is a signature and anomaly-based intrusion detection method, has been used to protect the IoT from routing attacks based on the IPv6 routing protocol [44]. On one hand, applying a signature-based technique for malware detection in the IoT is not the best approach because it is not designed to detect unknown/newly developed malware files; on the other hand, designing a behavioral-based or specification-based method to secure the IoT is computationally expensive due to the long simulation process it requires.

Major AI solutions to securing the IoT fall under either behavior or specification-based techniques, which are complex to implement in IoT systems. For instance, the authors in [45] evaluate the recent advances in AI/ML techniques in securing the IoT. They use 80% of the dataset only to train the module, which is computationally expensive, and state that, despite the advances in AI techniques in the IoT, the security method is still vulnerable when implemented in a real IoT system. Other AI and ML-based solutions for malware detection in the IoT are presented in [46–48]. Furthermore, the authors in [49–51] published surveys about AI solutions enhancing IoT security by presenting the challenges and limitations of algorithms. Besides the weak probability and instability of AI algorithms, they are computationally complex, with high resource consumption. Therefore, in this work, we analyze the AIS solutions to secure the IoT that are less complex for implementation with high detection probabilities.

As businesses and consumers continue to connect devices to the Internet without proper security measures, IoT devices are increasingly leveraged by cybercriminals to dispense malware payloads [52]. In the first half of 2019, SonicWall observed a 55% increase in IoT attacks—a number that outpaces the first two quarters of the previous year. A security vendor detected over 100 million attacks on IoT devices in the first half of 2019, highlighting the continued threat to unsecured IoT devices [53]. Kaspersky, the Russian Anti-Virus vendor, has claimed to detect 106 million attacks from 267,000 unique IP addresses in the first half of 2019 [53]. This number of attacks was almost nine times more than reported for the first quarter of 2018, when only 12 million were detected, originating from 69,000 IP addresses. According to the authors in [53], a major reason driving this surge is consumers' increased propensity to buy smart home solutions without due diligence regarding security measures. Due to all the reasons listed above, malware attacks are major security threats in the IoT and thus require an IoT-specific security solution.

The best way to secure the IoT based on its characteristics and architecture is to implement a distributed, dynamic, adaptive, and self-monitoring method. This leads us to investigate the AIS solutions in the next chapter (Chapter 3) and how these can be applied to secure the IoT against malware attacks.

2.6 SUMMARY OF IOT SECURITY CHALLENGES

IoT networks consist of interconnected devices with unique identifiers that provide real-time interaction. These devices often have limited computational capacity and small memory. IoT devices are interconnective, which means they are connected to the cloud and/or other IoT devices to enable remote control. They are heterogeneous, meaning the connected devices might run on different platforms and have different specifications. Moreover, connected IoT devices are capable of exchanging services within the constraints of things. Finally, the increasing number of connected IoT devices generates an enormous scale of data in a massive-scale network. The IoT System architecture consists of three main layers: the perception layer, which is the physical layer for the connected devices, the network layer, which is responsible for securing the connection between the device and the cloud, and the application layer, which is the front-end layer to deliver the service. Each layer is a target for a different set of attacks. For instance, malicious node injection and battery draining from security threats to the physical layer. Man in the middle and denial of service are threats to the network layers. Malware attacks are one of the main threats to the application layer.

Based on recent attack analysis, malware forms a huge security threat to IoT systems, and detecting unknown malware files is one of the key challenges. Static, dynamic, and hybrid analysis are three ways to analyze a malware file. Signature and behavioral-based techniques are used to detect malware files. First, the signature-based technique reads a unique part of the file for detection. This method is efficient in detecting known malware files with known signatures; however, it can not be used to detect unknown malware files. Second, the behavioral-based technique simulates the behavior of a malicious or benign file for detection. This technique is efficient in detecting unknown malware files; however, it is expensive to run.

Based on the IoT characteristics and malware detection techniques presented, the best way to secure the IoT is by implementing a lightweight and adaptive method. For these reasons, we investigate the AIS methods for securing IoT systems.

3 ARTIFICIAL IMMUNE SYSTEMS METHODS -BACKGROUND-

3.1 INTRODUCTION TO ARTIFICIAL IMMUNE SYSTEMS

Nature has crafty ways of solving problems. The knowledge retrieved from its observation has been a source of inspiration for computer scientists throughout the years for devising solutions to challenging problems. In particular, problems where the traditional methods fail to provide a suitable solution or would result in a complex solution requiring high computational power. In the cases where analytic expressions are not available, nature-inspired computing may be able to find sub-optimal solutions efficiently. Nature-inspired algorithms abstract the phenomena found in the wild and are subject to evolutionary steps or computing layers to converge to a solution. Examples include Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), Artificial Neural Networks (ANNs) [54], and AIS [55]. AIS is a field composed of different methods inspired by many theories of the biological immune system. The immune system is responsible for protecting the body from any intrusions, and any possible danger called an antigen. In this work, we consider malware to be an unwanted foreign intrusion, and we examine the application of the defense mechanisms followed by the adaptive immune system in fighting antigens. The following figure (Fig. 3.1) shows five different topics discussed among AISs.

3.2 HUMAN BODY IMMUNE SYSTEM

The first defense line in the body is the Innate immune system; it detects and kills any malicious activity. If the system gets attacked by an unknown source, it kills and keeps the information about it. This system is mostly responsible for sending alerts to cells and directing them to the infected area. If the Innate system fails to eliminate the threat (antigen), it is time for the

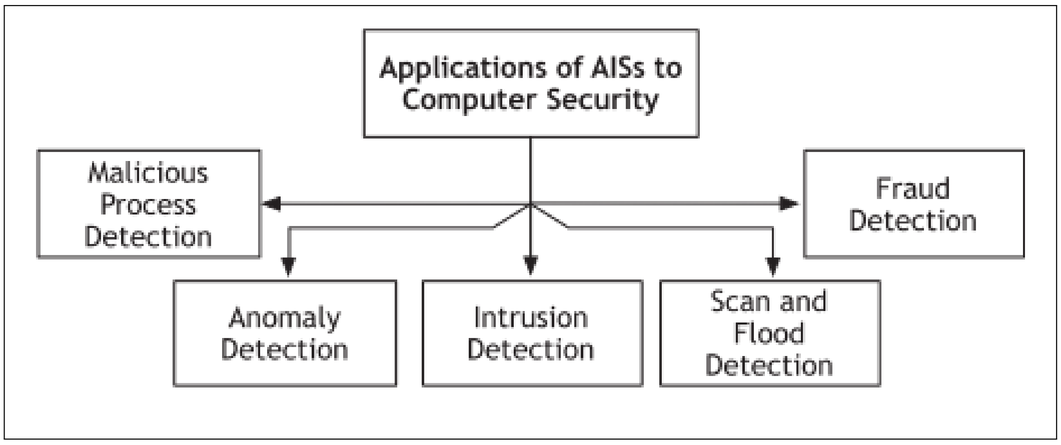


Figure 3.1: Artificial Immune Systems Main Topics and Area of Research [56]

Adaptive Immune system to fight the (antigen). It gets information about attacks from the Innate system to prevent the same attack from happening again [57]. Innate immune system: it is composed of outside layers to protect the body, for example, skin, and inside defense layer, for example, the acid in the stomach. Also, blood cells such as:

(a) Neutrophils: if this type of cell encounters an antigen, it kills it and then die

(b) Macrophages: this type of cell can kill up to 100 germs before they die (can also kill the infected body cells -cancer-)

Adaptive immune system: it has two lymphocyte cell types, B and T cells [58]. The figure below (Fig. 3.2) shows the T cell immunity and the antibody immunity.

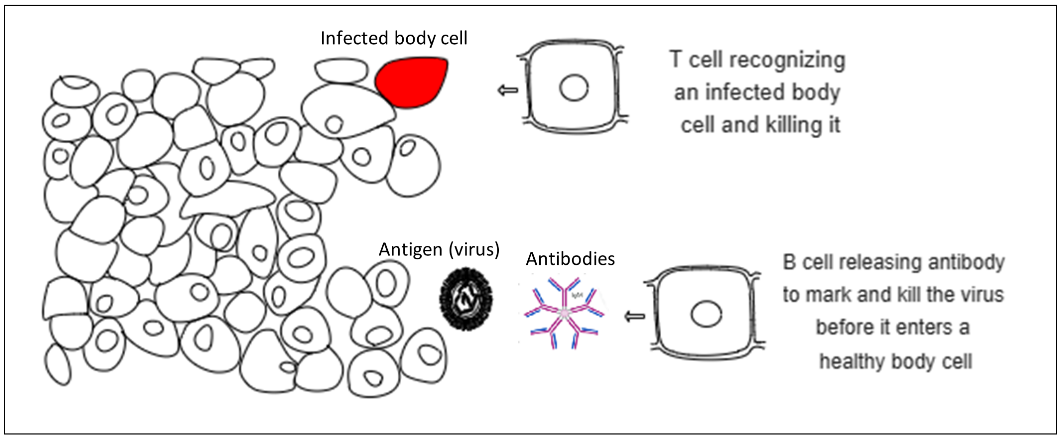

Infected body cell
T cell recognizing an infected body cell and killing it
Antigen (virus)
Antibodies
B cell releasing antibody to mark and kill the virus before it enters a healthy body cell

Figure 3.2: T cell Immunity and Antibody Immunity

(a) B cells come when a pathogen enters the body and before the disease occurs. They provide antibodies to stick to the antigen and "mark it" as a sign for the Macrophages to kill it. B cells have memory B cells which keep information about the antigen.

(b) T cells: they come when the infection occurs, and it consists of:

- Helper T cells: take information about the antigen, which consists of effector T cells, which call everyone and tell them about the antigen, and memory T cells, which keep information about the antigen.

- Cytotoxic T cells: kill the infected body cells that cannot be treated

The figure below (Fig. 3.3) shows how B and T cells get activated.

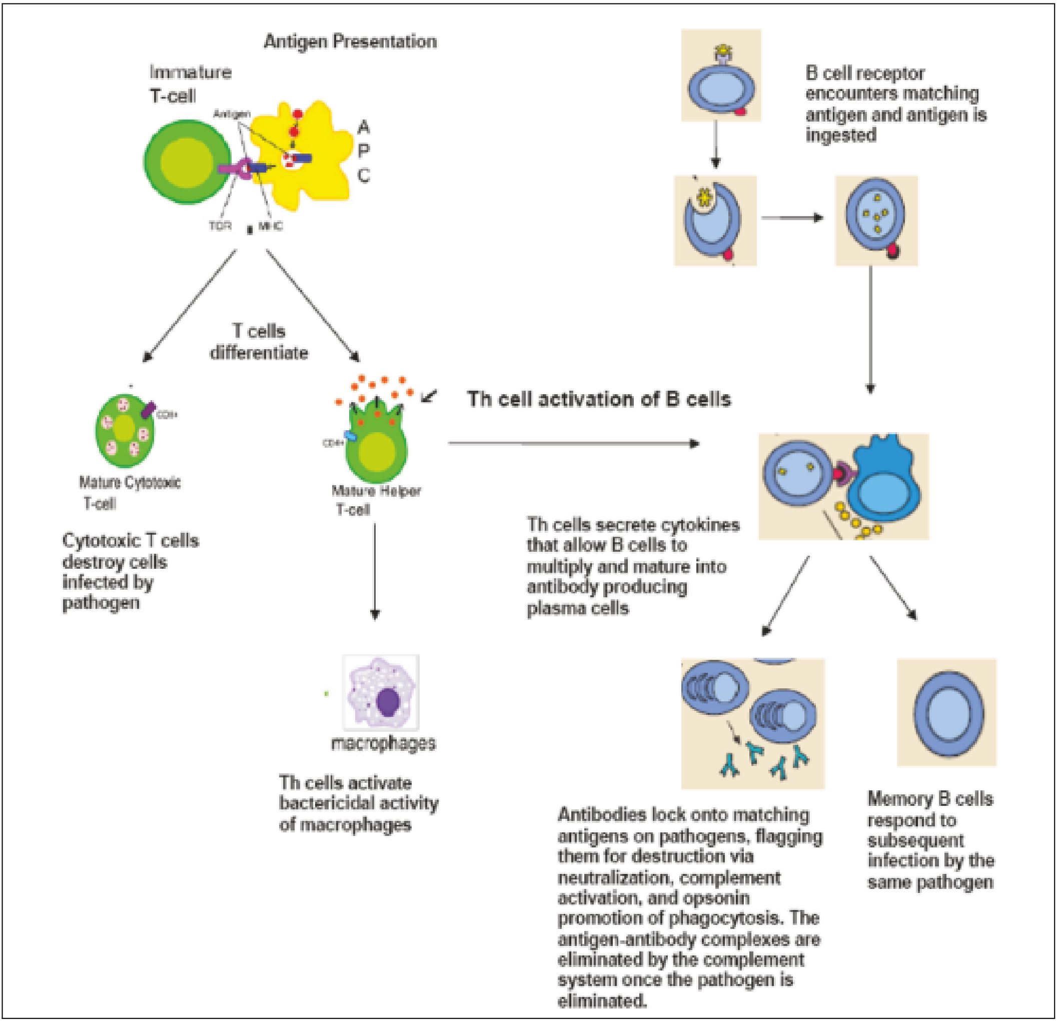

Antigen Presentation
Immature T-cell
Antigen
A P C
TCR
MHC
T cells differentiate
CD8+
Mature Cytotoxic T-cell
Cytotoxic T cells destroy cells infected by pathogen
CD4+
Mature Helper T-cell
Th cell activation of B cells
macrophages
Th cells activate bactericidal activity of macrophages
B cell receptor encounters matching antigen and antigen is ingested
Th cells secrete cytokines that allow B cells to multiply and mature into antibody producing plasma cells
Antibodies lock onto matching antigens on pathogens, flagging them for destruction via neutralization, complement activation, and opsonin promotion of phagocytosis. The antigen-antibody complexes are eliminated by the complement system once the pathogen is eliminated.
Memory B cells respond to subsequent infection by the same pathogen

Figure 3.3: T-cell and B-cell Activation and Function [59]

3.2.1 *Adaptive Immune System Methodology*

The main part of our adaptive immune system is B cells which generate the antibodies. There are 100 million types of B cells in the human body, and the reason is that each kind of B cell generates different antibodies to catch any possible attack because different antibodies handle different antigens. Consequently, when a specific type of antigen enters the body that requires a particular type of B cells to handle, the body starts generating more of that specific B cell [60–62].

Antibodies generation:
It is made of a light and thick chain of different DNA. A mix and match will be done here to generate different types of antibodies that can mark any type of antigens. Consequently, each B cell will have its own kind of antibodies after mixing and matching.

Clonal Selection:

(a) B cells will generate a test patch of their own antibodies that go to the surface as "bait" called B cells receptors. B cells will be floating around in their zone, trying to find a matching antigen (which their specific antibodies can catch).

(b) When a B cell bond with a cognate antigen, it doubles its size and divide into two B cells, and these two B cells will double in size and divide, which makes it four B cells in total.

(c) B cells will send all generated antibodies to the bloodstream, and most B cells die after all the hard work.
The main job for antibodies is to mark the antigen (opsonize), not to kill it!

(d) Now the antigen is marked with antibodies, so it is phagocyte's job (such as Macrophages) to eat it and kill it. The antibody forms a bridge between antigens and macrophages.

Neutralizing antibodies : when an antigen enters the body, it uses its receptor to hang on a cell and then enter that cell. What happens is that the antigen uses the cell to generate duplicate copies of itself, then kill the cell and moves to neighbor cells. Antibodies can hang on the antigen receptors preventing them from entering the cell or making more copies.

3.3 Artificial Immune Systems Methods

Understanding how the adaptive immune system works to defend the human body, researchers have started developing different methods that imitate a similar process to protect computer networks. The use of AIS in security applications is mostly in detecting security incidences, such as intrusions at the host or the network carried out by malicious actors, using low-level scripts, automated tools, or malware [63].

(a) **Negative Selection Method:** supervised learning classification algorithm, which was inspired by the "process of self-tolerance of B-cells, and CLONALG, which is inspired by clonal selection theory and consists of mutation and selection processes" [64]. The method works in two phases: the detector generation phase (see Figure. 3.4) and the matching and detection phase (see Figure. 3.5). First, it generates detectors that do not match the protected data, and then it keeps matching these detectors with that data. If a match occurs, it means a change has happened in the protected data, and action must be taken. This method was first introduced in [56], and the main idea was to come up with a method that has similar techniques to the human immune system, where the system is capable of distinguishing between self-cells (the body cells) and non-self-cells (antigens). In computer networks, we map the self-cells to authorized system files and non-self cells to malicious files [65]. This approach relies on three main points:

- in the negative selection, the detector generation stage is run when the method is deployed/activated on a new site. For this reason, different detectors that do not match self are generated each time this stage is run. Consequently, if a copy of a detector set at one site is found, the other sites still have different copies.
- we match the self-data of each site to generate the required negative detectors, which means we have different sets of detectors to protect each entity based on its own data.
- unlike the signature-based method, the negative selection method should detect any foreign activities that do not match the self-data rather than checking for a certain pattern in each file.

The technique in the negative selection method relies on two main factors:

- data representation: this is a fundamental difference between many models of negative selection algorithms. It changes the matching rule process, detectors' generation, and the detection process. The main data representation for this method is binary, assuming that all datasets are eventually implemented as binary bits. Other representations include numeric data, categorical data, boolean data, and textual data. These different representations could be grouped into two different categories: String Representation and Real-Valued Vector representation.

- matching rule: matching rule defines matching or recognition, which is the distance measured between tested data and generated detectors. It is used in both detectors' generation stage and detection stage. For all data presentation, matching rule M can be formally defined as a distance measure between d and x within a threshold, where d is a detector, and x is a data instance [66]. This matching rule introduces the concept of partial matching, where the detector and the data instance do not have to be exactly the same in every single bit to be matched. For example, if we have this data: 11001100, and we are applying matching distance = 3, matched detectors could be (11001100, 11001111, 11001000, 00101100, etc.) where at least 5 bits match in the original data the detector.

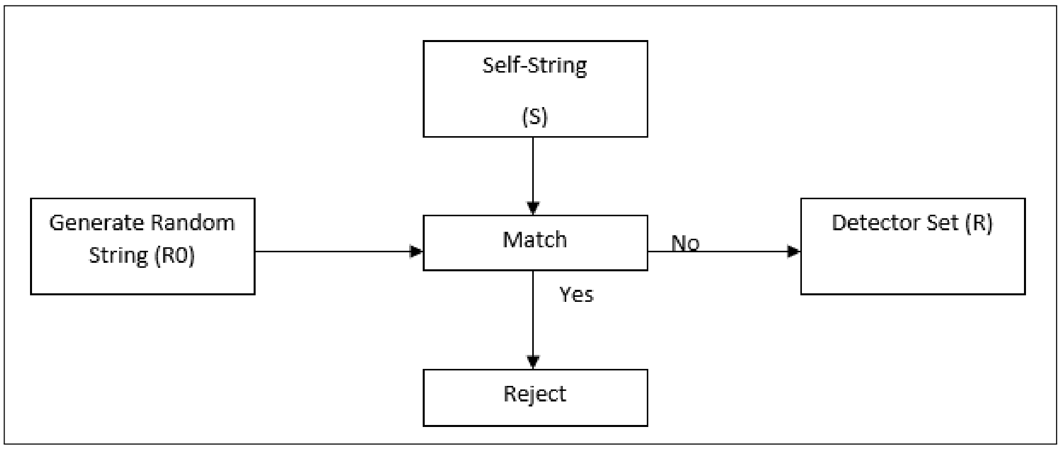


Figure 3.4: Negative Selection Method's Detectors Generation Stage [67]

(b) **Positive Selection Method (inspired by negative selection):** the positive selection method is inspired by the process of T-cells selection where only T-cells that can recognize self-molecules (body cells) will be used in the immune system. Unlike the negative selection method, this positive selection will generate detectors that recognize and match with self-protected data (see Figure. 3.7). Then, during the detection stage, is a detector that does not match the protected data, which means some changes have occurred to the protected data. The positive Selection Classification algorithm (PSCA) is a general classification algorithm that classifies unknown data using classifiers that can recognize self-class (system files) data. Authors in [64] applied PCSA in malware detection following the next steps:

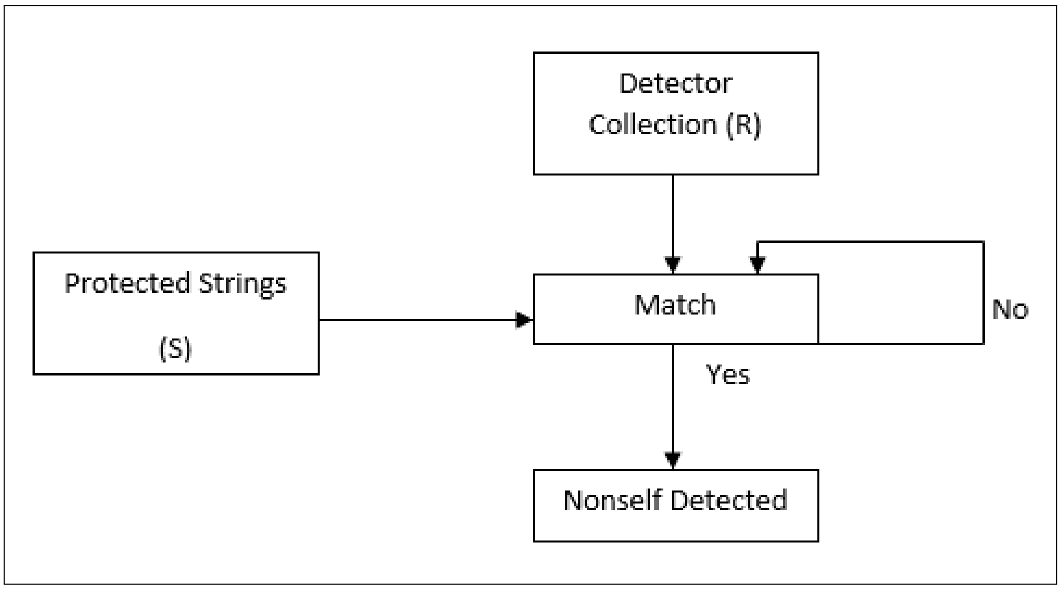


Figure 3.5: Negative Selection Method's Detection Stage [67]

- learning stage: In this stage, the method learns how to classify data into two different classes (self and non-self)
- classification stage: The authors implemented the positive selection classification stage using radius presentation. Unlike the binary presentation of data where minimal distance is applied using a matching threshold, in the radius presentation, the distance between a detector and a system file is presented in a circle format. To simplify, we show the radius presentation in the figure below (Fig. 3.6). Detectors are generated given a fixed radius (R) for coverage (presented in a black border circle). All files close to any detector less than R are classified as benign/self-data (presented in green). Other files that are outside the range of the given radius of the detectors are classified as non-self-data (represented in red). The authors in [68] present more examples on fixed and flexible radius range for detector generation.

(c) **Clonal Method:** the clonal selection theory was proposed in [70], and states that B-cells undergo cloning, variation, and selection to mature affinity. The CLONALG method was proposed by Castro and Zuben, and it is inspired by the clonal selection theory; the CLONALG method was initially designed for optimization and pattern recognition issues [71]. According to authors in [55], CLOALG requires the definition of five main factors:

- size of receptors population
- selection strategy

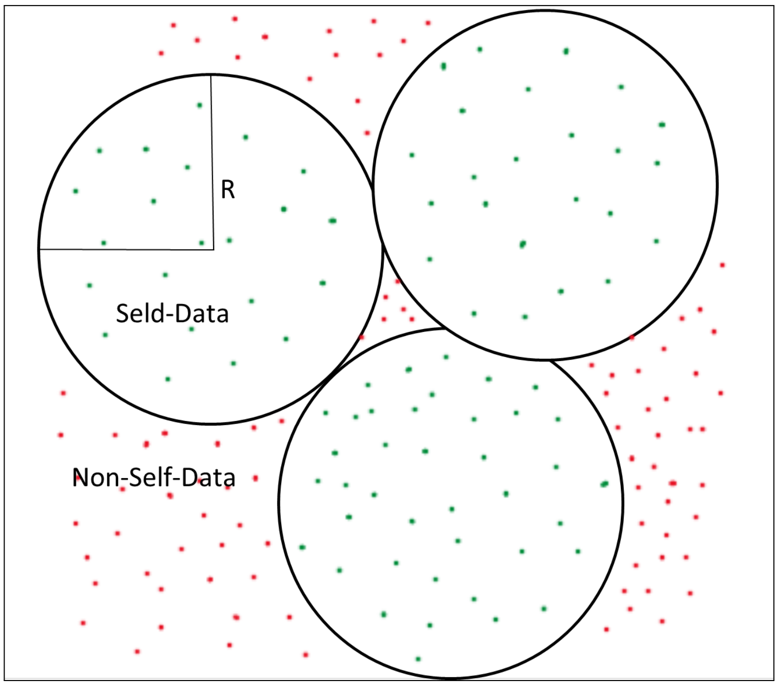


Figure 3.6: Radius Presentation of Detectors and Self/Non-Self Data

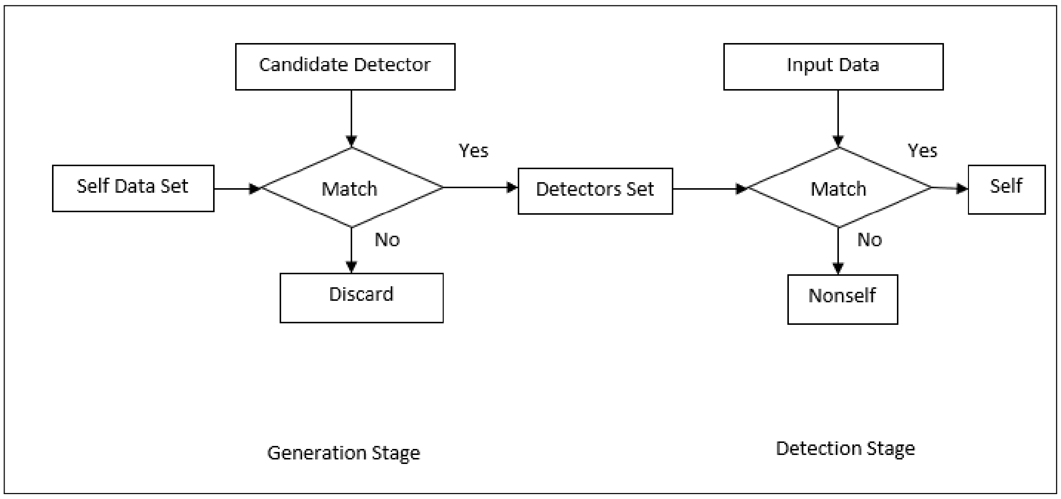


Figure 3.7: Detector Generation and Matching in Positive Selection [69]

- number of receptors
- affinity function that returns real-valued measures
- function to assign the rate of mutation and the number of clones according to the affinity.

To simplify, the cloning method is a supervised data mining technique. When an antigen enters the body, B-cells start cloning specific antibodies for that type of antigen. However, if it is new, the immune system clones the most stimulated lymphocytes. Similarly, in CLONALG method generate a set of receptors R that can recognize a set of patterns P.

(d) **Artificial Immune Networks Method:** artificial immune networks (AIN) theory was proposed in [67]. AIN is an unsupervised learning algorithm inspired by B-cells' immunological memory due to the existence of a mutually reinforcing network of themselves. This process means that B-cells interact to spread information so that memory can be preserved, displaying active behavior even when no immune response is taking place [55]. AIN mimics the immune network theory and parts of clonal selection as well. The AIN system process aims to set up a collection of repertoires for a given issue, where better-performing cells stifle low-similarity (comparable) cells in the system. This standard is accomplished through an intuitive procedure of presenting the population to outer data, to which it reacts with both a clonal selection reaction and inner meta-elements of intra-population reactions that balances out the reactions of the population to the outside boosts.

As the human immune system can detect and react to antigens in our body, the AIS can determine and respond to malicious files different from system files used in the training phase [69]. AIS can detect discrepancies in the system behavior and identify attacks without prior knowledge, making them ideal candidates for detecting unknown malware files. In the next section, we investigate the state-of-the-art AIS solutions in malware detection and securing the IoT.

(e) **Danger Theory**: against the self and non-self theories, danger theory is based on the idea that the immune system in the human body is not necessarily capable of detecting self and non-self cells. However, the immune system is capable of detecting cells that might cause danger to the body, which triggers an immune response by sending a danger signal [72]. These cells could be infectious self, such as cancer cells, or infectious non-self cells, such as virus infection. The recognition of the infectious cells happens by analyzing the cell context, "tissue context," and the balance between two types of cell death, necrosis, and apoptosis. On one hand, in apoptosis cell death, the cell contents are easily reduced, breaking the cell from the inside out. Dendritic cells are sensitive to an increase in the signals of apoptosis and are attracted to the dying cell. Eventually, the dead cell is found by a dendritic cell and absorbed. On the other hand, in necrosis cell death, the cell could die because of cell stress, such as irradiation or lack of oxygen. In this case, the cell contents degrade chaotically in the body, which might cause irregular fragments of DNA to be produced and become uric acid crystals. Dendritic cells are sensitive to changes in the concentration of the molecules released as a result of necrosis death. Dendritic cells move from the tissue and present and collect debris as antigen to T cells to trigger an immune response [73].

3.4 SUMMARY OF ARTIFICIAL IMMUNE SYSTEM METHODS

AIS is a field based on mimicking the human immune system mechanisms. Mainly, AIS is based on how the B and T cells defend the human body during an attack. B cells get activated when an attack (antigen) enters the body. B cells' main role is to provide antibodies to mark the antigen by getting attached to it. T cells get activated when an infection occurs, and t cells kill the infected body cells and keep a record of the attack using memory T cells. AIS applications are used widely in the area of security. For instance, the AIS methods are used in malicious process detection, anomaly detection, intrusion detection, scan and flood detection, and fraud detection.

We discuss for main AIs methods used in security. The first is the negative selection method which is a supervised learning classifying method. Negative selection is based on the B cells technique in marking an antigen. The second is the positive selection method based on the T cells technique in defending the human body. Next is the clonal method, based on the B cells cloning antibodies for specific antigens. Finally, artificial immune networks are an unsupervised classification method based on B memory cells for identifying an attack.

4 A CRITICAL EVALUATION AND ANALYSIS OF THE STATE-OF-THE-ART OF AIS SOLUTIONS

4.1 INTRODUCTION TO AIS METHODS IN SECURING THE IOT

AIS applications are artificial intelligence (AI) techniques inspired by the intelligence of the human body's immunology. Given its ability to detect unseen attacks and its low complexity, various AIS-based methods are proposed in the literature for IoT security. An immune-based architecture was presented in [69] to secure the IoT using edge technologies based on IoT system requirements. As highlighted by the authors, the architecture meets IoT security requirements, such as adaptability and lightweight, and can secure IoT nodes from various security threats and attacks. However, the proposed method is to secure the IoT using edge technologies, which means it is limited to a certain IoT system architecture. Moreover, the availability of this method has not been considered during the evaluation process. In addition, to secure Internet protocol version 6 (Ipv6) in the IoT, a bio-inspired method was presented in [74]. An AIS-based method is implemented in the routing protocol for low-power and lossy networks to enhance the security level and performance with the given limited resources in the IoT. The main limitation of this approach is that it is time and energy-consuming, which makes it difficult to secure IoT devices with limited resources. The following section reviews AIS methods for IoT malware detection, including negative and positive selection algorithms and immune and artificial immune-based methods.

4.2 AIS IN MALWARE DETECTION IN THE IOT

This section highlights the work conducted in malware detection using AIS in the IoT. The original negative selection algorithm uses Binary Encoding to represent self and non-self-datasets. Later on, real-valued methods were proposed, and some researchers adopted different types of malware detection techniques such as variable-sized detectors [75], hypercube detectors [76], hyper-ellipsoid detectors [77], and multi-shaped detectors [78]. Deeper investigations have been conducted using a Hypersphere detector because it has simple mathematic calculations compared to the other types. These different data representation methods have not been applied to securing the IoT since they are not sufficiently lightweight to meet the IoT system requirements.

4.2.1 *Negative and Positive Algorithms*

One of the objectives of the main concept of negative selection is to produce enough detectors to cover the non-self-area. Most approaches generate these detectors randomly in different ways to cover holes and overlaps and improve the detection rate. Many researchers have proposed combining two AIS methods to overcome this challenge.

The authors in [79] proposed the MNSA algorithm, a combination of negative and positive selection detectors. The first set of detectors can recognize self-data, and the other set of detectors is used to detect non-self-data. Combining the results of these two detector sets is supposed to improve the system's detection rate for unknown malware files. To test the method's efficiency, randomly generated 12-bit long strings are used for both the training and detecting stages of the algorithm. As a result, it was claimed in [79] that the MNSA algorithm could detect up to 34% of all intrusions without any prior knowledge about the non-self, and it can confirm more than 90% of those detected files. The main limitation of this research is that it was tested on random strings and not actual malware files. Furthermore, this method uses too many detectors in both negative and positive sets.

The authors in [55] proposed using the positive selection algorithm (PCSA) for malware detection. They define the PCSA as a general classification algorithm for unknown data classification. Positive selection and clonal selection algorithm techniques were applied to secure the IoT. The algorithm has different stages, starting with the learning stage to produce classifiers: self and non-self. The main goal of this algorithm is to recognize self-data, and after the learning stage, the authors claim that all classifiers are available to classify unknown data. They also define two states after classification: overlap, where more than two kinds of classifiers recognize the unknown data, and hole, where the unknown data cannot be recognized by any classifier. To evaluate the proposed algorithm, the researchers in [55] compared their solution to another algorithm in [80]. In total, 3721 Windows malicious executables and 3458 benign Windows executables were collected for the experiment. There are four types of malicious files: backdoor, spyware, trojans, and worms. The main feature captured and used for malware detection here is I/O request packets (IRPs), for which they developed an MBMAS tool presented in [81] that can associate a process with its child process in run time. Researchers claimed a 99.30% accuracy result for the PSCA algorithm that they developed. The only limitation that this paper claimed is that IRP traces of programs vary from one host to another, and some IRPs repeat sometimes. This method has not been implemented in an IoT system, and we find this work not sufficiently robust to cope with the interconnective environment of the IoT.

4.2.2 *Negative and Neural Networks*

An artificial neural network-based algorithm for intrusion detection in the IoT is presented in [82]. The algorithm uses three neural network layers, input, hidden, and output layer. The input layer feeds the model with data passed to the hidden layer, which can not be accessed outside the environment. The third layer is to show the output of the hidden layer. The method achieves an 84% detection accuracy rate and less than 8% false negatives. The authors only presented the results of a simulation of the algorithm; no real-time data or platform was used.

The authors in [83] proposed using a negative selection algorithm combined with neural networks (NSNN) for intrusion detection in the IoT. The research goal is to develop an algorithm that meets IoT requirements, is lightweight enough to apply to a wide range of IoT use cases, is capable of detecting previously unknown intrusion vectors, and provides an acceptable detection rate. The dataset used in this experiment is KDD NSL [84]. The authors use only the basic traffic features, which provide most of the needed information. The different types of intrusions are divided into 23 different sets (22 types of attacks and one normal). Then, the attack types are divided into three attack sets: denial of service (DOS), PROBE, and All Attack Types (AAT). They tested the algorithm against different percentages of normal and attacks of each type (10%, 20% ...90% attack and subsequently 10%, 20%...90% of normal). Each one of the 27 sets iterated 100 times with different test data sets every time. The trained NSNN algorithm was tested against the dataset. The following coefficients were calculated: positive predictive value, negative predictive value, sensitivity, specificity, accuracy, Matthews correlation coefficient (MCC), and F1-Score (the harmonic mean of the precision and recall). This research achieved an F1-Score of 0.77 in the DOS simulation, 0.72 in the PROBE simulation, and 0.73 in all AAT simulation results. The researchers in [83] claimed that their work is limited to creating the negative selection and neural network algorithm only. Currently, they make no claims about the best way to implement an online learning mechanism for it. Furthermore, they noted that the test set used in the experiment is dated, and the results should be used only for comparison purposes and not to demonstrate the actual performance of the algorithm. In addition to the presented shortcomings, we find the F1-Score of this algorithm to be unreliable in securing the IoT systems.

4.2.3 *Danger Theory Algorithms*

There are different implementations of danger theory to detect malware attacks with minimizing false negative detection. However, most of these solutions are complicated enough not to fit the constrained IoT devices [85]. For instance, authors in [86] presented a hybrid solution (MANET) for malware detection using negative selection and danger theory. They use the idea of memory cells to provide a better structure for the generated detector sets, which increases the expandability of the detectors. Meaning having a larger scope to detect malware files. Even though the proposed method is promising for malware detection, the structure is complicated enough, making it expensive for IoT systems. Moreover, authors in [87] presented a method to detect and extract danger signals for malware detection with the goal of minimizing the false positive detection rate. The proposed method collects information from network traffic, analyzes the collected data, and then marks each file as either good or bad. The files that are marked as bad then get deleted from the system. Considering the massive scale of traffic data produced and managed in the IoT systems, this approach is not lightweight to fit the IoT-specific requirements.

4.2.4 *Immune and Artificial Immune Based Algorithms*

An AIS-based approach for malware detection in the IoT is presented in [88]. The authors develop an algorithm for intrusion detection in smart homes using negative selection and random input parameter selection. The experiment uses a Raspberry Pi connecting smart home devices to the internet via a router. One of the drawbacks of this project is that no results were published for the experiment.

Celosia is an immune-inspired intrusion detection technique for the IoT devices presented in [89]. Celosia is a non-supervised method that consists of subsystems of many network structures that are individually trained. This method is evaluated only for detecting botnet attacks that enable the attacker to access devices connected to the internet.

The authors in [90] presented an AIS-based algorithm for malware detection (DeepDCA). DeepDCA uses a dendritic cell algorithm (DCA), a danger theory technique, and Self-Normalizing Neural Networks (SNN). The proposed approach focuses on the preprocessing phase, presenting the feature selection, the SNN signal categorization, signal processing, and anomaly metrics steps. The Bot-IoT dataset was used in the experiment, converting some categorical variables to apply the feature selection method easily. The method was evaluated using different file features, resulting in an F1-Score less than 50% when using imbalanced data for the best ten file features in the dataset. When using balanced data for the ten best file features in the dataset, the F1-Score increased to over 90%. Although this method achieves a high detection accuracy rate with low false negatives, it is neither sufficiently lightweight nor distributive to be implemented in IoT devices.

The authors proposed the artificial awareness architecture (AWA) in [91] as a model for artificial immune ecosystems. Their experiment shows that the proposed algorithm can detect intrusions in specific given IoT architectures; however, it does not detect outliers–anomalies.

Moreover, the researchers in [92] proposed a novel approach to securing the IoT based on immunology techniques. The proposed method adopts dynamic and circular defense processes against a security threat. It incorporates five links: security threat detection, danger computation, security response, security defense strategy formulation, and security defense. The first link collects and analyzes IoT network traffic, and the other links function based on the produced results. The method simulates AIS techniques for intrusion detection based on the following mechanisms: capturing the

IoT traffic data and simulating the data to antigens in AIS; representing the detector simulation for the detection elements, such as the lifetime and the number of recognized antigens; thirdly, implementing a matching mechanism to determine if there is a match between a detector and an antigen. Also, the evolution process is represented by classifying the detectors into immature detectors, mature detectors, and memory detectors. In the experiment, cloning attacks, mutated cloning attacks, replay attacks, and mutated replay attacks were simulated. Even though this method can detect security threats and change detectors to adapt to the dynamic IoT environment, no real malware files were used in this experiment. In addition, this work was not implemented in a real IoT scenario.

Furthermore, the authors in [93] proposed an artificial immune-based method for intrusion detection in the IoT. The method involves many local intrusion detection sub-models that share their learning attainments. The signature information in the IoT sense layer represents antigens in this method as binary strings. Detector sets are generated, including a number of antigens matched by the detector and the generation life of the detector. One of the main limitations of the proposed method is that it is not sufficiently lightweight to meet the IoT system requirements.

Finally, the authors in [94] proposed an AIS-based algorithm for intrusion detection in the IoT. It was claimed that the main signature information on the IoT datagram is extracted to be switched to a binary character string for experiment purposes. Different detector stages are identified as immature, mature, and memory detectors. The authors stated that immature detectors meet the recognition diversity of intrusion detection, while mature detectors evolve to be immature detectors. Although this paper presents a new method of detecting unknown malware in the IoT environment, no simulation results were given. In addition, we find this method to be memory space and time-consuming for IoT devices.

Table 4.1 shows a comparison of the AIS-implemented solutions for securing the IoT.

Table 4.1: Comparison of AIS applications for securing the IoT.

Method	Year	Experiment Results Included	Malware Files Used in the Experiment	Limitations and Shortcoming Presented	Method Covers Holes and Overlaps
Danger theory-based [87]	2003	✘	NA	✔	✘
PCSA [55]	2011	✔	✔	✔	✔
MANET [86]	2014	✘	NA	✔	✘
MNSA [79]	2017	✔	✘	✔	✘
Neural network-based [82]	2019	✔	✔	✘	✘
NSNN [83]	2018	✔	✔	✘	✘
AIS-based [88]	2020	✘	NA	✘	✘
Celosia [89]	2020	✔	✔	✘	✘
DeepDCA [90]	2020	✔	✔	✘	✘
AWA [91]	2017	✔	✘	✔	✘
Immune-based [92]	2013	✔	✘	✘	✘
AIS-based [93]	2012	✘	NA	✘	✘
Immune-based [94]	2011	✘	NA	✘	✘

4.3 IOT SYSTEM SECURITY REQUIREMENTS

In the previous section, various implementations of AIS for securing the IoT were reviewed. Our study shows a revived interest in addressing malware detection using the AIS method accompanying the spread of IoT systems. Table 4.2 highlights five main properties to be considered when applying AIS applications to the IoT.

Table 4.2: IoT Systems' Properties.

Property	Definition
Robust	The capability of a system to cope with issues during execution and continue operating despite data conditions
Lightweight	The capability to operate and execute with minimal computational complexity
Fault tolerance	The capability to function given a defect within hardware or software in the system, and adapt to the changing environment to build up a trustworthy network
Adaptive	The capability to adapt and learn the system behavior over runtime
Distributed	The capability to run and communicate within a distributed environment

4.3.1 *Immune-Based Implementations Challenges*

Many AIS applications contain some of these properties, but implementing an AIS algorithm that meets all the requirements remains unsolved. For instance, designing an immune-based method results in implementing a robust and adaptive solution for securing the IoT; however, the method is neither lightweight nor fault-tolerant and not necessarily distributed [92–94].

4.3.2 *AIS Hybrid Solution Challenges in the IoT*

Implementing a method based on AIS techniques is difficult. For instance, clonal selection algorithms are adaptive but computationally expensive. Moreover, clonal selection suffers from high false positives, and the degree of damage cannot be inferred instantly. On the other hand, the negative selection algorithm has high false negatives and is unsuitable for dense environments. Combining two or more AIS algorithms might be the solution to overcome some of these challenges, such as applying negative selection and neural network techniques in NSNN, which results in fault-tolerant, adaptive, and distributed solutions; however, it is not lightweight [83]. Furthermore, negative and positive selection algorithm techniques were combined in MNSA to improve the detection rate in the IoT [79]. Even though the goal of implementing this method was met, the solution does not meet all the IoT system's requirements, such as robustness. The same scenario applies to PCSA, which is not fault-tolerant as well [55]. Based on the characteristics of AIS methods and IoT system properties, we contemplated the reviewed AIS solutions in IoT and investigated which properties are applied in each solution. Table 4.3 below shows the result of this analysis.

Table 4.3: IoT system properties adopted in AIS solutions.

Method/ Properties	**Robust**	**Lightweight**	**Fault Tolerant**	**Adaptive**	**Distributed**
Danger Theory + Negative Selection [86]	✘	✘	✘	✔	✔
PCSA: Positive Selection [55]	✘	✔	✘	✔	✔
Danger Theory [87]	✘	✘	✘	✔	✔
MNSA: Negative Selection + Positive Selection [79]	✘	✘	✘	✔	✔
Neural Network based [82]	✘	✘	✘	✔	✔
NSNN: Negative Selection + Neural Network [83]	✘	✘	✔	✔	✔
Artificial Immune based method [88]	✘	✔	✘	✔	✘
Celosia: Immune System based [89]	✘	✔	✘	✔	✘
DeepDCA: Artificial Immune-based [90]	✔	✘	✘	✔	✘
AWA: Artificial Immune Ecosystem [91]	✔	✘	✘	✔	✔
Immune System based method [92]	✔	✘	✘	✔	✘
Artificial Immune based method [93]	✔	✘	✘	✔	✔
Immune System based method [94]	✔	✘	✘	✔	✔

4.4 SUMMARY OF AIS APPLICATIONS

AIS methods are generally attractive for malware detection owing to their ability to detect unknown attacks and intelligently keep records of any attack for future use. In addition, they are a prime contender in the design of IoT malware detection because the offered features best match IoT system characteristics. The features of AIS methods, such as their adaptivity, distributed implementation, lightweight computation, and robustness, are compatible with the IoT devices' specific requirements. To this end, we survey recent research in the field of AIS for malware detection. We critically analyze existing works, draw key insights, and identify promising future research directions in which novel AIS techniques can be developed to address imminent and increased IoT security challenges.

5 A NOVEL AIS-BASED METHOD TO SECURE THE IOT - SIMULATION RESULTS

5.1 INTRODUCTION TO A NOVEL AIS-BASED METHOD

In this chapter, we present a Negative-Positive Selection (NPS) algorithm, a novel AIS method, for detecting unknown malware in the area of IoT. NPS addresses two dominant challenges in IoT security: the dynamic and ever-changing nature of IoT malware attacks and the lightweight restrictions of IoT devices that limit the choice of security measures. The NPS algorithm is a hybrid method inspired by negative and positive selection techniques to overcome the challenges and limitations in the state-of-the-art. Specifically, the NPS algorithm is designed to overcome the challenges in the negative selection and the multiple negative selection algorithms [79]. First, in the NPS, we use the exhaustive detector generation method because time and space complexity are proven to be better than other methods, e.g., self-linear and greedy. Second, we generate the negative and positive detector sets separately, in parallel, to produce the exact number of required detectors. This is done to overcome generating an unnecessary number of detectors. Moreover, we overcome generating premature detectors in the detectors generation stage by increasing the detector's size from 12-bit to 16-bit. Finally, in the detection stage of the NPS, we match the incoming traffic first with the negative detectors and then with the positive detectors as opposed to matching the files to each set separately and then combining the results. Thus, the NPS algorithm is lightweight and suitable for IoT systems.

5.2 METHODOLOGY

As shown in Fig. 5.1, the NPS method works in two stages: a detector generation stage and a detection stage. The first stage is done once when running the algorithm. The second stage runs in a loop whenever a new file is introduced to the system for detection.

In this work, we propose to generate negative and positive detector sets separately, in parallel processes, to avoid the risks of previous AIS methods (see Fig. 5.2). Negative selection algorithms commonly use randomly generated 12 bits strings as input files to the malware detection mechanism. Differently, our proposed method allows input files to go up to 16-bit strings in order to capture the destination port address. This approach is thus designed to produce better detection results for the state-of-the-art. The NPS generates the same number of detectors in each set (the negative and the positive detector sets).

The detectors generation stage shown in Fig. 5.2 and Algorithm 1 comprises the generation of two different sets of detectors, a positive detector set (PDS) and a negative detector set (NDS). PDS contains detectors that match self-data based on the positive selection concept, and NDS contains detectors that do not match self-data based on the negative selection concept. A matching threshold (see Algorithm 1 variable R) is used in both stages of this algorithm, which defines the level of similarity between two strings to be considered matching.

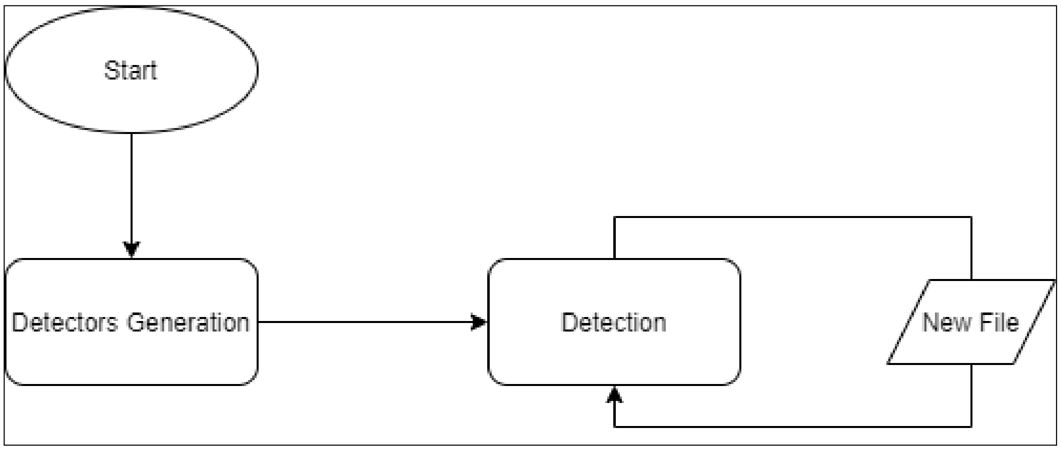


Figure 5.1: The NPS Workflow

Once the first stage of detector generation is completed, both sets of PDS and NDS are ready for detection. The second stage is the detection stage described in Fig. 5.3 and Algorithm 2. The proposed method is designed to store information related to previously detected attacks in the first layer

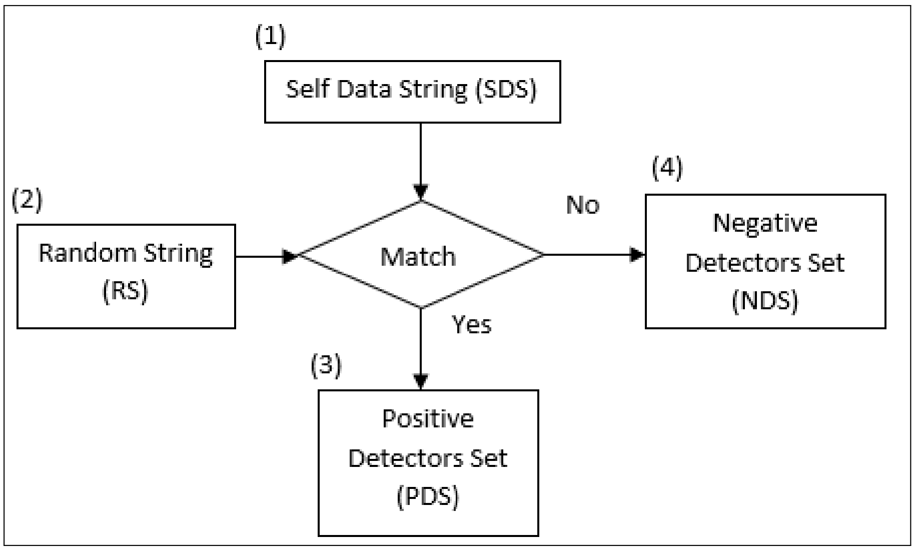


Figure 5.2: The NPS Detectors Generation Stage

(see Fig. 5.3 process (1)). Access to this data renders the proposed method faster and more accurate in detecting known malware. The first step in this stage is comparing an incoming file to the existing database on known attacks. If there is no match, it indicates that either the file is self-data or is an unknown malware file. As shown in Fig. 5.3 process (3), the file is first processed by the NDS, which is able to detect an unseen malware file. In this case, the system is alerted, and the new data is stored in the database. Otherwise, the file is processed by the PDS. If the incoming file matches any detector in PDS, it is considered benign. Else, it is flagged as a malware attack.

5.2.1 *Detectors Generation Calculations*

Captured data conversion:

$$S = d||d| = L, d = toString(OriginalDataset) \tag{5.1}$$

Where

$$L \in N \tag{5.2}$$

L is the length of self-data
N is nature number set
toString() is the function to convert captured self-data into binary

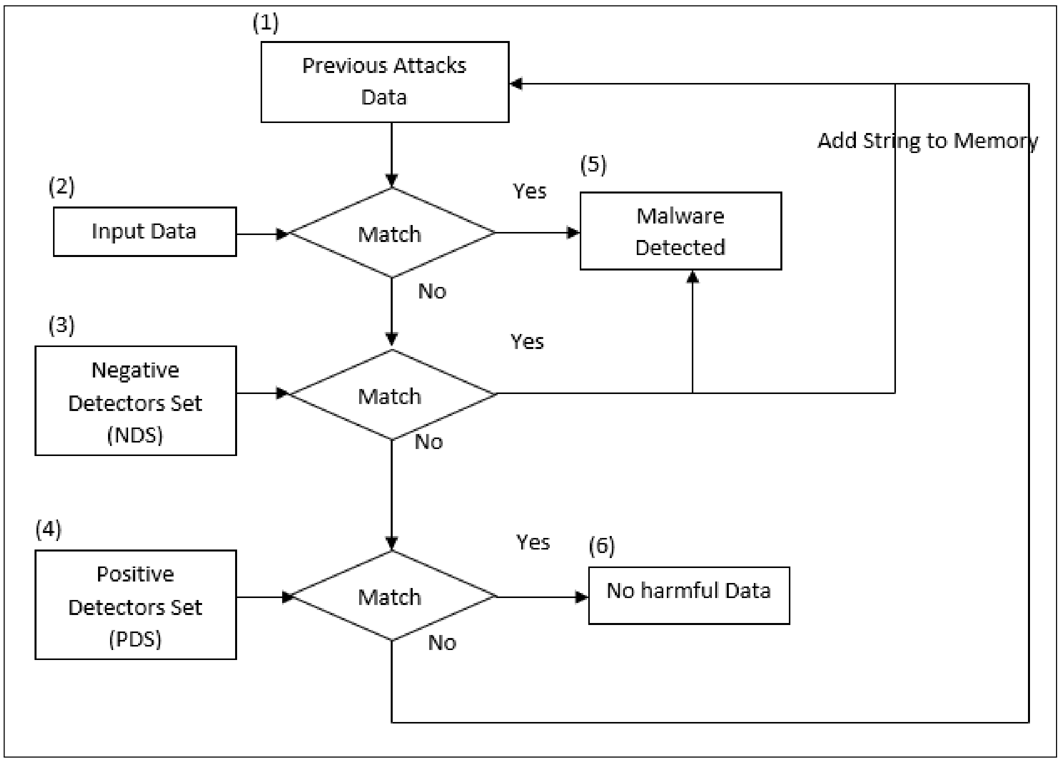


Figure 5.3: The NPS Detection Stage

Algorithm 1: The NPS Detectors Generation Stage

```
Input
NS = set of self-data
R = the number of contiguous matches required for
a match
DS = Detectors set size
Output
PDS= set of detectors capable of classifying self-data
NDS= set of detectors capable of classifying non-self-data
begin
while $i \leq DS$ do
   $D$ = a randomly generated detector
   if $D \in NS$ then
      $PDS = PDS + Detector$
   else
      $NDS = NDS + Detector$
   end if
   $i \leftarrow i + 1$
end while
```

Algorithm 2: The NPS Detection Stage

```
Input
PA = previous attacks data
Dataset = dataset to be recognized
Dataset = dataset to be recognized
R = the number of contiguous matches required for
a match
PDS= set of detectors capable of classifying self-data
NDS= set of detectors capable of classifying non-self-data
Output
Attack = malicious files
Benign = systems files

begin
while $i \leq DatasetSize$ do
  if $Dataset[i] \in PA$ then
    $Attack \leftarrow 2$
    $EXIT$
  else if $Ddataset[i] \in NDS$ then
    $Attack \leftarrow Attack + 1$
  else
    $Benign \leftarrow Benign + 1$
  else if $Dataset[i] \in PDS$ then
    $Benign \leftarrow Benign + 1$
  else
    $Attack \leftarrow Attack + 1$
  end if
  $i \leftarrow i + 1$
end while
switch $Attack$ do
  case $1$ do
    Low detection
  case $2$ do
    Malware detected
switch $Benign$ do
  case $1$ do
    Low detection
  case $2$ do
    Benign file detected
```

Initialization:
Data will be represented in Binary: 0,1
We will define self data as S and non self data as N where:

$$D = S \cup N \tag{5.3}$$

And

$$S \cap N = \varnothing \tag{5.4}$$

D = Training Data set
d = a training data set

$$d \in D \tag{5.5}$$

Detectors Generation:

In this research, we will use the Exhaustive detector generation method, which is the original method proposed by authors in [95].

m = the number of alphabet symbols (m = 2 in string representation)
L = the number of symbols in a string (length of the string)
r = the number of contiguous matches required for a match
Pm = the matching probability between a detector string and a randomly chosen self string
N_r0 = The number of initial detector strings (before censoring)
N_r = The number of detector strings after censoring (size of the repertoire)
N_s = The number of self string
f = The probability of a random string not matching any of the N_s (self strings)

$$f = (1 - Pm)^{N_S} \tag{5.6}$$

Pf = The probability that N_r detectors fail to detect an intrusion

THEN

$$Pm \approx m^{-r} \frac{(L-1) \times (m\breve{}1)}{m+1} \tag{5.7}$$

$$Pm \approx \frac{1}{N_s} \tag{5.8}$$

$$N_r 0 = \frac{-LnPf}{Pm \times (1 - Pm)^{N_s}} \tag{5.9}$$

$$N_r = \frac{-LnPf}{Pm} \tag{5.10}$$

$$Pf \approx e - Pm \times N_r \tag{5.11}$$

Positive Detection:

all detectors D_p are chosen among S (self data)

$$Dp = Dp1, Dp2, \ldots\ldots, Dpi \tag{5.12}$$

Negative Detection:

all detectors D_n are chosen among N (non self data)

$$Dn = Dn1, Dn2, \ldots\ldots, Dni \tag{5.13}$$

Where

$$Dpn = Dp \cup Dn \tag{5.14}$$

RCB (r-contiguous bits) matching rule:

In this algorithm, we will use a "Matching threshold."

Dpni M d $\leftrightarrow$ distance measure between a detector and a training data set is within a threshold

The matching between the two strings will be calculated using the Euclidean equation:

$$D = \sqrt{\sum_{i=1}^{l} (Di - Dpni)^2} \tag{5.15}$$

5.3 DESIGN AND IMPLEMENTATION

In order to validate and evaluate the effectiveness of NPS in IoT malware detection, we have implemented the methodology using a real dataset. We present the dataset and implementation setup in the following paragraphs.

5.3.1 *Dataset*

There are different datasets used to evaluate IoT security solutions. As listed in [96], the most used datasets are the NSL-KDD, the Bot-IoT, the Botnet, and the Android malware datasets. In this project, we chose to use the NSL-KDD [84] for two reasons. First, unlike the other datasets, the NSL-KDD eliminates the redundant records in the previous dataset (KDD'99), reducing the number of borderline records compared to any other dataset [96]. This leads to more accurate results when evaluating an AIS-based security solution. Also, by eliminating the borderline records, we reduce the total number of records (see details in Table 5.1), unlike the Bot-IoT [97], which has 72,000,000 records. Using a larger number of records to evaluate an IoT security solution might overwhelm the system when running the solution in an actual IoT system setup. Please refer to Appendix A A.1 for the dataset records extraction.

Table 5.1: NSL-KDD Dataset Used in the Experiment.

Total number of records used	1,074,992
Number of attack files	262,178
Number of benign files	812,814
List of attacks	Brute-force, Heartbleed attack, Botnet, Denial of service, Distributed Denial of Service, Web attacks, and infiltration of the network from inside
Number of traffic features	80
Some of the traffic features	Destination port, flow duration, average size of packet, number of forward packets per second, number of backward packets per second

5.3.2 *Simulation Setup*

The experiment has two stages: a detector generation and a detection stage. The first stage is a training model based on a data file that includes benign (harmless) data records. The data file is extracted from the system's files. Within the file, the destination port is used to generate and test the detectors, thus, resulting in 16-bit string detectors. If the generated detector matches one of the records in the file, it is added to the positive detector set. Otherwise, it is added to the negative detector set (as shown in Fig. 5.2). The second stage of this experiment is the detection stage, where a file with benign and malicious records (different than the ones used in the training stage) is used to test the method.

To evaluate the method detection rate and efficiency, we use the following calculations:

- **True positive (TP):** malware is detected as a malicious application
- **True negative (TN):** benign is detected as non-malicious application
- **False positive (FP):** benign is detected as a malicious application
- **False negative (FN):** malware is detected as non-malicious application

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.16}$$

$$Precision = \frac{TP}{TP + FP} \tag{5.17}$$

$$Recall = \frac{TP}{TP + FN} \tag{5.18}$$

A malware detection method is first evaluated based on its achievable accuracy rate as defined in Equation5.16, with higher accuracy as the objective. A more revealing performance indication is precision and recall as defined in Equation5.17 and Equation5.18. For instance, let us consider a dataset that includes 80 benign and 20 malware files. If Method A results in all benign files being correctly identified but all malware incorrectly identified, the result is 80% accuracy. However, the performance of Method A is unacceptable as it is incapable of detecting malware with precision and recall being null. Method B, on the other hand, detects 60 benign files and all 20 malware files correctly, the accuracy is still 80%, but the performance is much more reliable in detecting malware. The precision for method B is 50%, and the recall is 100%. The ultimate goal in securing the IoT is to maximize the recall percentage in detecting malware attacks.

The experiment undergoes multiple iterations where each employs a different number of detectors. As such, comparing the results of these iterations reveals the combination that yields the highest detection rate with the minimal number of detectors. Using fewer detectors reduces time and space complexity in the implementation of the NPS, as discussed in the results section.

5.4 THE NPS SIMULATION RESULTS AND DISCUSSION

5.4.1 *Detection Accuracy*

As aforementioned, to increase the detection rate and decrease the number of used detectors, we use 16-bit strings and the same number of detectors in both positive and negative sets. In the first iteration, we create 40 detectors, which means 20 detectors in each set, and we achieve close to 92% detection rate. When increasing the number of detectors to 60 in total, 30 detectors in each set, the detection rate is increased to 99% as shown in Fig. 5.4. To compare our method to the state-of-the-art, we run different iterations using the MNSA [79] technique which uses 12-bit strings and a different number of detectors (5/10, 10/50, 15/100, 20/150, and 30/200) positive negative detectors, respectively. The highest detection rate is close to 89% with 30/200 positive and negative detectors. When increasing the string size to 16-bit, the detection rate increase to up to 99% with 20/150 positive and negative detectors as shown in Fig. 5.4.

5.4.2 *Detection Precision and Recall*

One of the main goals of this hybrid method is to reduce the FN rate since it is one of the negative selection challenges. The risk of a FN in malware detection is obvious; it means that a malware file is allowed into the network, which would cause serious harm to any application, e.g., traffic signals, e-health, and smart homes. On the other hand, a FP does not have a direct security impact, but a high rate of FP would reduce the usability of any application. In other words, if many benign files are treated as

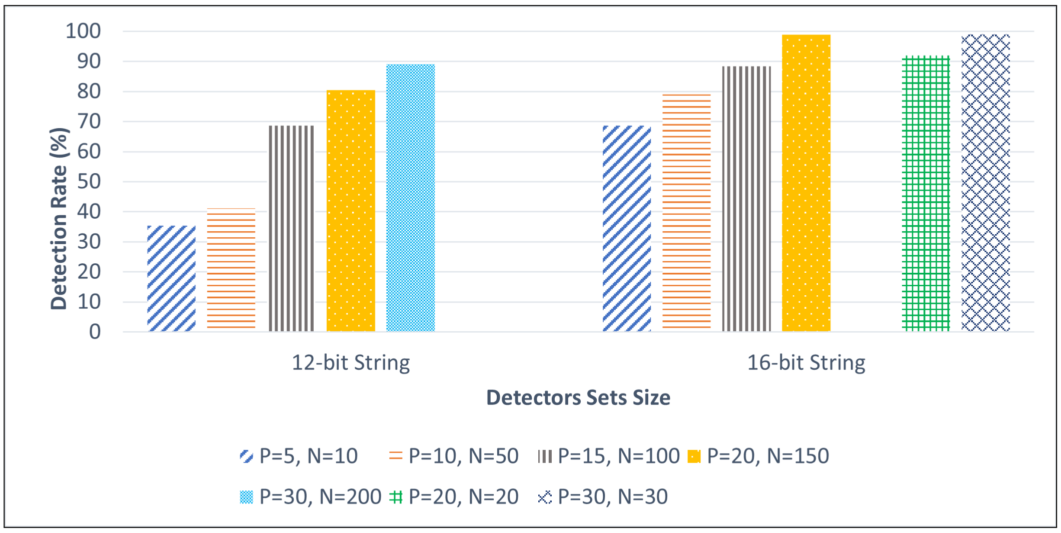


Figure 5.4: The NPS Simulation Results

malware, the efficiency of the information exchange gets negatively impacted at a high cost, e.g., unjustified long traffic queues and emergency health services being triggered when not needed. This would eventually limit the progress in IoT-enabled applications. To this end, starting the detection process by comparing incoming files to the negative detectors, then confirming the detection results with the positive detectors improves the classification of malware as a non-malicious application rate. Running NPS for unknown malware detection, we achieve almost 0% FN and less than 1% FP detection rates. Moreover, we achieve up to 95% precision rate and 96% recall rate when generating 20 positive and 20 negative detectors using the precision and recall formulations in (2) and (3), respectively. The precision and recall rates both increase to 99% when 30 detectors are generated in both sets, as shown in Fig. 5.5. This means that the NPS method could be implemented on its own to protect IoT devices, which require less complicated security measurements taken. Even though this is an important metric to evaluate intrusion detection methods' preciseness, there were no results given in this regard in the MNSA algorithm [79].

5.4.3 *Space and Time Complexity*

As aforementioned, IoT devices are lightweight, with less computational power and memory capacity than traditional networks, which makes implementing security solutions challenging. In this section, we calculate the time and space complexity to validate that the NPS algorithm is not computationally expensive, which makes it suitable to be implemented in the

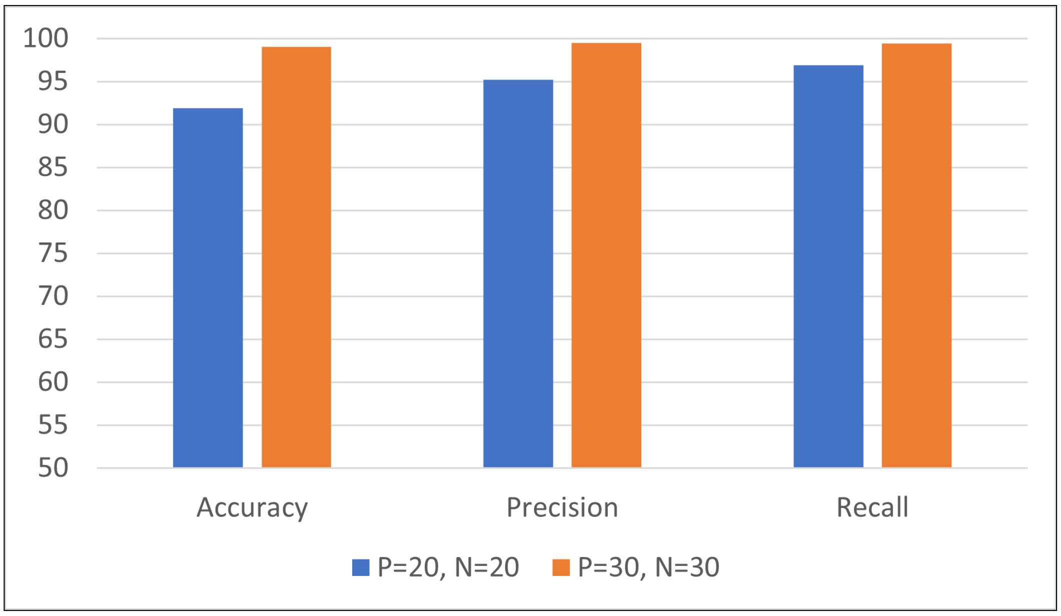


Figure 5.5: The 12-bit & 16-bit String Results

IoT. After running these two stages, the time and space complexity is calculated using the following equations [56]:

$$Time = (m^L \times N_S \times N_R) \tag{5.19}$$

$$Space = (L \times N_S \times N_R) \tag{5.20}$$

Where: m is alphabet size (m = 2 in binary representation), L is string size, N_S is a number of self-data, and N_R is a number of detectors. Space complexity based on (5) is reduced by 1.4% when the number of detectors N_R is reduced for two settings of N_S=12,16. Using an equal number of detectors with 16-bit string results in a 65% decrease in space complexity than using the 12-bit string technique. 64% is the decrease in time complexity when using the 16-bit string with an equal number of detectors than using a 12-bit string with larger detectors set.

5.5 QUANTITATIVE PERFORMANCE ANALYSIS

In this section, we highlight the main criteria to evaluate the performance of the most promising AIS methods in the literature for malware detection in the IoT [79, 83] and the presented novel method [1]. The most recent AIS solutions for securing the IoT are selected to present a quantitative performance analysis. These methods are selected because of their promising results (accuracy and false negatives), which we were able to reproduce to enable the quantitative performance analysis. A false-negative denotes malware that is falsely classified as benign. It follows that a better malware detection method results in fewer false negatives.

The NSL-KDD dataset is used to evaluate the NPS [1] and NSNN [83] methods. Consequently, in order to enable a quantitative performance analysis, we reproduce the results of the MNSA using the same NSL-KSS dataset. The traffic data were captured by running 420 machines and 30 servers in 5 different departments. Although the NSL-KDD dataset is not IoT-specific, it contains various malware attack types. It offers different file features to test security solutions, which makes it a good fit for this experiment's purposes. In contrast to other machine learning approaches presented in the literature review (Chapter 4), AIS requires minimal data to create necessary detectors later used in the detection phase. In our case, 10% of randomly selected samples of the dataset are used in the detector generation phase, and the remaining 90% are used for testing. We compare the performance from two perspectives: in Section 5.5.1, we analyze the detection accuracy and F1-Score of each; in Section 5.5.2, we examine the complexity of each algorithm from both time and memory perspectives.

5.5.1 *Detection Accuracy and F1-Score*

The NPS [1] uses both negative and positive detectors and overcomes two main challenges in securing IoT applications. First, the method is lightweight, as it generates a smaller number of detectors compared to other AIS algorithms, such as the MNSA [79], with a higher detection rate accuracy, calculated using Equation (5.16). With 40 detectors in total (20 negative and 20 positive detectors), the NPS achieves up to a 92% detection rate and a rate of up to 99% when using 60 detectors in total (30 negative and 30 positive detectors; see Figure 5.6). When reproducing the results of the MNSA, the detection rate accuracy increases to 80% when using 170 detectors in general (150 negative and 20 positive detectors). The mean detection accuracy rate for the NSNN [83] is close to 73%, which is lower than both NPS and MNSA algorithms. Second, it overcomes the false-negative detection challenge. As explained earlier, accuracy alone does not fully capture the detection performance as it does not highlight the false negatives. In other words, a detection accuracy of 75% may result from a 100% misclassification of malware (since 25% of the records are labeled as attacks—262.178/1,074,992, as shown in Table 5.1). To this end, we calculate the F1-Score (see Equation (5.21)), which is more representative of the performance when the data are not balanced.

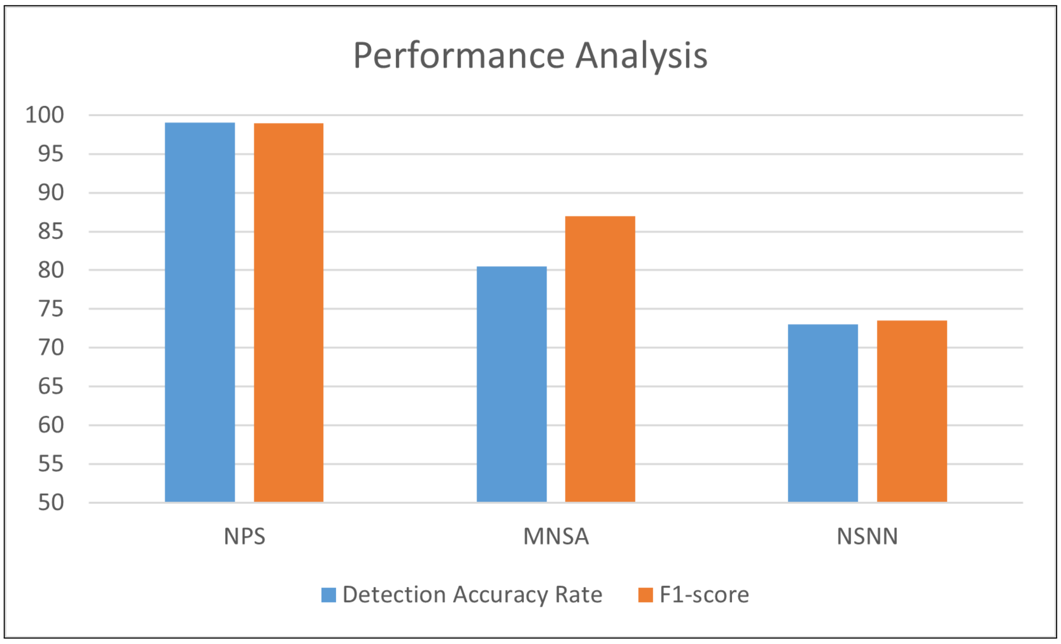


Figure 5.6: Accuracy and F1-Score Results of NPS, MNSA, and NSNN using NSL-KDD dataset

As shown in Figure 5.6, calculating the F1-Score for the NPS, we obtain a score of 96% when using 40 detectors in total. When using 60 detectors, the F1-Score for the NPS algorithm increases to 99%. The F1-Score for the MNSA increases to 87% when using 170 detectors, and the F1-Score for the NSNN is 73%. Overall, the NPS achieves almost a 14% improvement.

We calculate the detection rate accuracy, Precision, and recall using Equations (5.16),(5.17), and (5.18), respectively. We calculate the F1-Score using the following equation (Equation 5.21).

$$F1 - Score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{5.21}$$

5.5.2 *Memory and Time Complexity*

IoT devices are lightweight with limited computing power; therefore, reducing memory usage and computing time when applying security methods is essential. We calculate the space complexity for the NPS, MNSA, and NSNN using Equations (5.19) and (5.20), where m is the alphabet size (m = 2 in binary representation), L is the string size, N_S is the amount of self-data, and N_R is the number of detectors. Table 5.2 shows the values of the three methods for each parameter.

Using 16-bit strings with an equal number of detectors in both negative and positive sets in the NPS results in a 65% decrease in memory usage compared to generating 12-bit strings with larger detector sets in the MNSA. To calculate the space complexity for the NSNN, we assume that the string length is $\geq$7 since the R-Continuous Bit Matching (RCBM) is 7. RCBM is the number of matching bits between two strings: self and non-self. In this case, the NPS uses 90% less memory space than the NSNN.

When calculating the time complexity using Equation (5.19), the results show that the NSNN needs less computing time than the other two methods—MNSA and NPS. The following Figure 5.7 shows the result of the space and time complexity analysis.

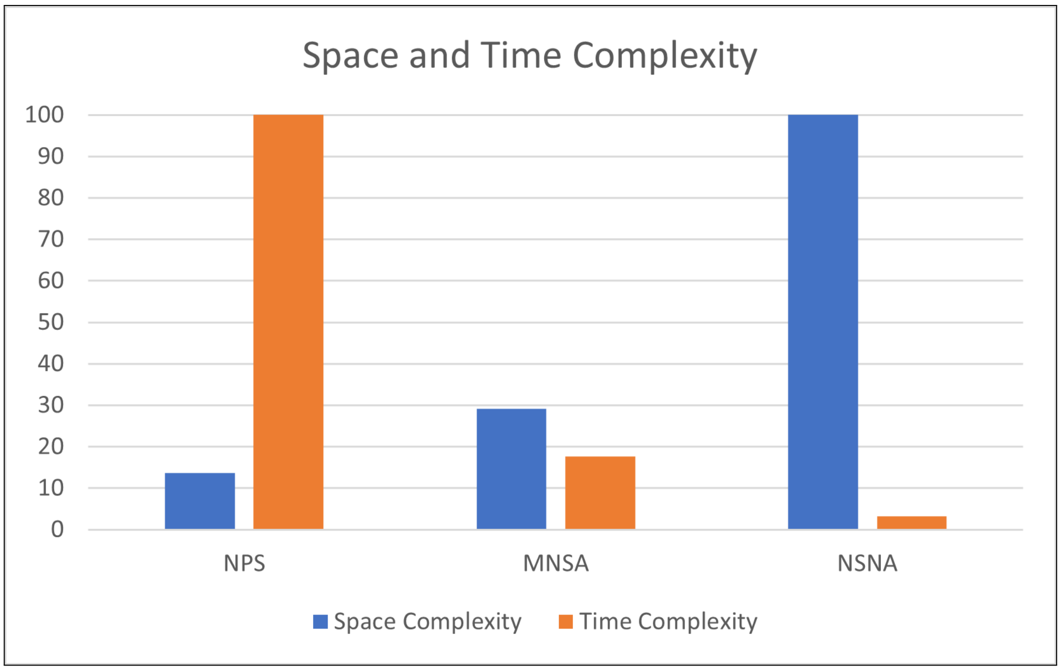


Figure 5.7: Memory and Time Complexity of NPS, MNSA, and NSNN

Table 5.2: Space and time complexity calculations.

Method	M	L	N_S	N_R
NPS	2	16	1000	60
MNSA	2	12	1000	170
NSNN	2	7	1000	1000

5.6 Validating the NPS simulation results using the Bot-IoT and UNSW-NB15 datasets

In this chapter, we use two state-of-the-art datasets widely used to evaluate intrusion detection methods in the IoT. For the first dataset, we use the Bot-IoT dataset (see details in 5.6.1), and the second one is the UNSW-NB15 (see details in 5.6.2). The two datasets have been created by the intelligent security group at the University of New South Wales (UNSW) Canberra. We run the NPS using the two datasets and conduct a performance analysis. Next, we benchmark the obtained results against the state-of-the-art intrusion detection methods that use the same datasets.

5.6.1 *Bot-IoT Dataset*

The Bot-IoT dataset was created by designing a realistic network environment incorporating both normal and botnet traffic. The dataset includes DDoS, DoS, Operating System (OS) and Service Scan, Keylogging, and Data exfiltration attacks (see Table 5.3). The authors introduced and explained the Bot-IoT in [98–102]. The dataset records are available to be extracted in different formats, including the original pcap files. Please refer to Appendix A A.2 for the dataset records extraction.

Table 5.3: Bot-IoT Dataset

Total number of records	over 73,000,000
Number of attack files	73,360,900
Number of benign files	9543
List of attacks	DDoS, DoS, Operating System (OS) and Service Scan, Keylogging and Data exfiltration attacks
Number of traffic features	46
Some of the traffic features	Destination port, flow duration, average size of packet, number of forward packets per second, number of backward packets per second

5.6.2 *UNSW-NB15 Dataset*

The UNSW-NB15 dataset was created by the IXIA PerfectStorm tool for generating a hybrid of real modern normal activities and synthetic contemporary attack behaviors. The dataset includes nine types of attacks: Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, and Worms (see Table 5.4). The UNSW-NB15 is introduced and explained by the authors in [103–108]. The dataset records are available to be extracted in different formats, including the original pcap files. Please refer to Appendix A A.3 for the dataset records extraction.

Table 5.4: UNSW-NB15 Dataset

Total number of records	over 25000,000
Number of attack files	321,283
Number of benign files	2,218,761
List of attacks	Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, and Worms
Number of traffic features	49
Some of the traffic features	Destination port, flow duration, average size of packet, number of forward packets per second, number of backward packets per second

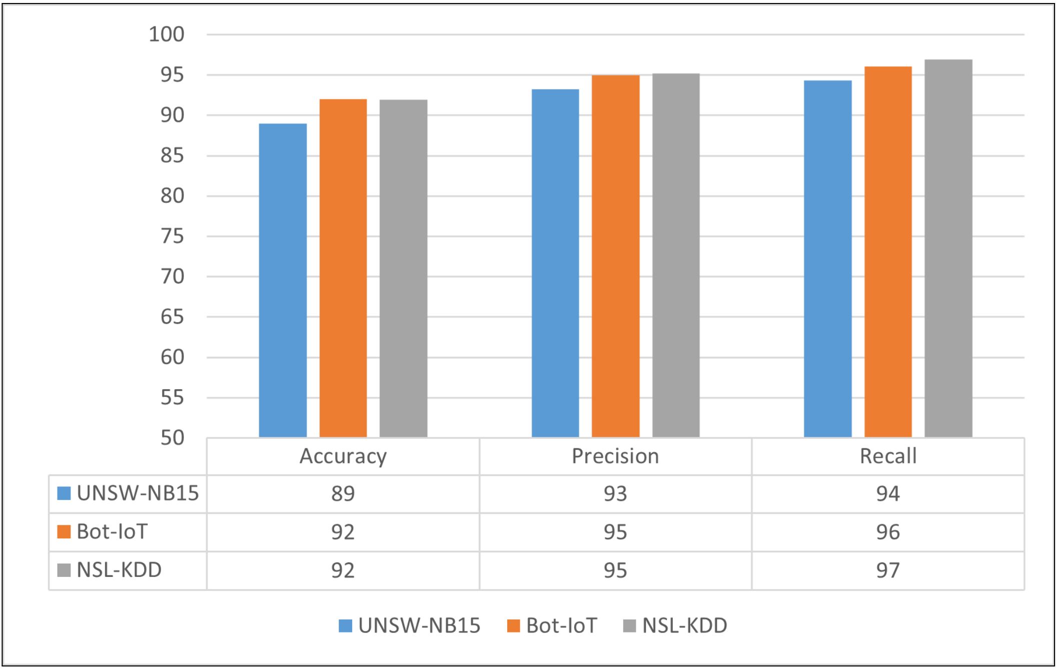
5.7 RESULTS AND DISCUSSION OF USING MULTIPLE DATASETS

This section presents the performance results of running the NPS using the Bot-IoT and UNSW-NB15 datasets. We then benchmark the results against the state-of-the-art advances in malware detection in the IoT using these two datasets. We use Equations 5.16,5.17,5.18, and 5.21 presented in Chapter 5 to calculate the presented results.

5.7.1 *The NPS Results and Discussion*

The figure below (Fig. 5.8) shows the performance analysis of running the NPS using the NSL-KDD, Bot-IoT, and UNSW-NB15 datasets with 20 detectors in each set, negative and positive. The NPS achieves high detection performance using all three datasets. The detection accuracy is up to 92%, the precision rate is up to 95%, and the recall rate is up to 97%. The results of creating 30 detectors in each set, negative and positive, using multiple datasets are shown in Fig. 5.9. The detection accuracy goes up to 99%, the precision rate is 99%, and the recall rate is close to 100%.

We can see a negligible gap when running the NPS using the three datasets. This difference is justifiable by the different number of records and types of attacks in each of the datasets. Therefore, this performance analysis shows the effectiveness of the NPS in detecting unknown malware attacks in IoT systems.

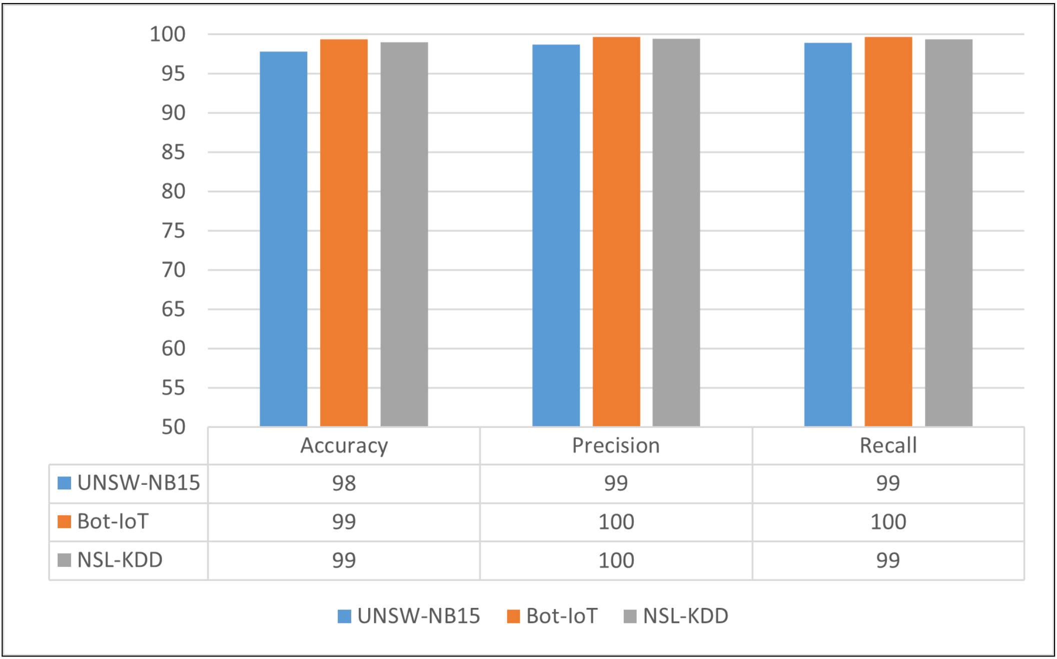


	Accuracy	Precision	Recall
■ UNSW-NB15	89	93	94
■ Bot-IoT	92	95	96
■ NSL-KDD	92	95	97

Figure 5.8: Running the NPS Using Multiple Datasets - 20 Detectors Results

5.8 STATE-OF-THE-ART USING THE UNSW-NB15 AND THE BOT-IOT DATASETS

In this section, we present a compression of the results we obtain running the NPS using multiple datasets and state-of-the-art solutions for malware detection in the IoT. Since detection accuracy is the common factor to highlight the performance in the published work, we use the detection accuracy rate in this comparison. First, the authors in [109] presented the LSTM-based unsupervised deep learning model for malware detection in the IoT. The authors reported 96% detection rate accuracy for running their model using both the Bot-IoT and UNSW-NB15 datasets combined. The



	Accuracy	Precision	Recall
🟦 UNSW-NB15	98	99	99
🟧 Bot-IoT	99	100	100
⬜ NSL-KDD	99	100	99

Figure 5.9: Running the NPS Using Multiple Datasets - 30 Detectors Results

main focus of this work is on detecting DoS and DDoS attacks. Second, the method presented in [110] achieves a 99% detection accuracy rate using the UNSW-NB15 dataset. However, the number of classes used for classification is unclear. Next, the method presented in [111] achieves a 70% detection accuracy rate using the UNSW-NB15 dataset. The results were obtained using only 10% of the total number of records of the dataset. Similarly, the authors in [112] used only 10% of the total number of records of the UNSW-NB15 dataset and reported 89% detection rate accuracy. Finally, feed-forward ANN (FNN) and self-normalized neural network (SNN) presented in [113] use the Bot-IoT for the experiment. They reported a 95% detection accuracy rate for FNN and 91% for SNN. They only use 20% of the total number of records and the ten best features to run the proposed solutions.

As anticipated, the NPS outperforms the state-of-the-art solutions detection accuracy rate by up to 42%. This validates the claim that the NPS is fit to secure the IoT from unknown malware attacks.

5.9 SUMMARY OF THE NOVEL AIS-BASED METHOD

AIS methods are adaptive, distributive, and lightweight, which makes them an ideal fit to secure IoT networks. This chapter presents the NPS algorithm based on negative and positive selection techniques to improve the unknown malware detection rate. The performance of NPS is benchmarked against state-of-the-art malware detection schemes using a real dataset. We achieve close to a 22% increase in the accuracy of detection rate using 16-bit strings with an almost 65% decrease in space complexity, which resolves one of the main challenges in securing the IoT. Another limitation to be overcome by the NPS is the false negative detection which was reduced to almost 0% of detected files resulting in a 99% recall rate.

Moreover, we run the NPS using multiple datasets to validate the obtained results and benchmark the performance against state-of-the-art advances. We create 20 and 30 detectors in each set, negative and positive. We use the Bot-IoT and USNW-NB15 datasets, two of the most recent and widely used datasets, to evaluate malware detection solutions in the IoT. We obtain similar results running the NPS using both datasets compared to the ones we obtain from the NSL-KDD dataset. There is a negligible difference in the detection performance due to the differences in the number of records and the number of attacks in each one of the datasets. We benchmark the results we achieve against the state-of-the-art methods in malware detection in the IoT using the same datasets.

6 IMPLEMENTING THE NPS IN REALISTIC IOT SYSTEMS USING AWS

6.1 INTRODUCTION TO THE IMPLEMENTATION

In Chapter 4, we present the IoT systems' properties in Table 4.2. Robust, lightweight, fault-tolerant, adaptive, and distributed are the five main characteristics of a security solution in IoT systems. This chapter highlights the AIS solutions' lightweight and distributed abilities.

The advantages of using AIS to secure the IoT systems are discussed in the previous chapters, showing that the NPS is the most promising method compared to state-of-the-art solutions. This was further demonstrated when simulation results showed the superiority of the NPS with respect to the recent promising methods in [79, 83]. However, these methods' results are insufficient to fully validate the method under a realistic setup. Real-life IoT systems follow different system architectures and employ devices with different characteristics.

In this work we implement the NPS and MNSA using a real-time platform. We create different IoT system architectures to test the AIS solutions efficiency in detecting unknown malware attacks with minimizing the memory utilization in the IoT device. We conduct a quantitative analysis studying the detection behavior of AIS solutions in a realistic setup. In the next sections, we present the implementation methodology, IoT system architectures, problem formulation, and results and discussion.

Section 6.2 presents the AIS solutions implementation in a realistic setup. We describe the methodology of this research, the dataset used, IoT systems architectures, and the problem formulation. Section 6.3 presents the implementation results, performance analysis, and discussion.

6.2 AIS SOLUTIONS IMPLEMENTATION

This work investigates the relationship between, on one hand, the achievable performance of AIS malware detection algorithms and the hardware and system architecture limitations, on the other hand. In this section, we first present the AIS solutions methodology in Section 6.2.1. Second, we present the dataset we use in this implantation in Section 6.2.2. Then, we present the IoT system architectures. We identify the hardware and software factors and their realistic range in Section 6.2.3. Finally, we formulate the malware detection problem as a function using these factors and the related labeled dataset in Section 6.2.4.

6.2.1 *AIS Solutions Methodology*

We implement the NPS and the MNSA in a real-time platform using AWS. This allows us to conduct a quantities performance analysis and study the behavior of the AIS solutions in realistic setups. Both the NPS and MNSA work in two stages: the detector generation stage and the detection stage. In the detectors generation stage, two different sets of detectors are generated: the negative detectors, represented as D_{Neg}, which do not match self-data. The positive detectors, represented as D_{Pos}, match self-data. In the detection stage, if an incoming file matches one of the negative detectors, it is tagged as malicious. In contrast, if an incoming file matches one of the positive detectors, it is tagged as benign. The size of each detector is represented as D_{Size}. Where:

$$D_{Pos+Neg} = D_{Pos} \cup D_{Neg} \tag{6.1}$$

The table below (Table 6.1) shows the size and the number of detectors used in this implantation for each method. The aim of choosing different detector sizes for both methods is to test the method's detection rate accuracy while also considering the method's complexity. The goal is to achieve high detection accuracy rate by minimizing the space and time complexity of running the method.

Table 6.1: Detectors Size

	NPS	MNSA
D_{Pos}	30	20
D_{Neg}	30	150
D_{Size}	16	12

6.2.2 *Dataset Used*

We use the NSL-KDD dataset (available in [84]) in this implementation. We use this particular dataset for mainly two reasons, first is the fact that it was used in the simulation experiment for both methods, NPS and MNSA. The other reason for using this dataset is that AWS computing platforms were used to collect the traffic data for this dataset. Consequently, we run the implementation on AWS using the IDS2018 dataset to obtain more coherent and accurate results. This dataset contains seven different malware attacks, brute-force attacks, heartbleed attacks, botnet, denial-of-Service, distributed denial-of-service, web Attacks, and infiltration of the network from inside [2].

We represent the total number of dataset records as D_{Total}, and we represent each record in D_{Total} as D_i where:

$$i = [1 to |D_{Total}|] \tag{6.2}$$

The size of D_i is represented as S, and each D_i includes the same number of features represented as F_S where:

$$F = [f_1, f_2, f_3, \ldots F_S] \tag{6.3}$$

Each record D_i is associated with the ground truth Y_i where $Y_i = 0, 1$. $Y_i = 0$ indicates that the record is benign and $Y_i = 1$ indicates that it is malicious. We define Y as the vector including all the labels of Y_i for:

$$i = [1 to |D_{Total}|] \tag{6.4}$$

The dataset D_{Total} is split into two parts: D_{Train} and D_{Test} where:

$$D_{Total} = D_{Train} \cup D_{Test} \tag{6.5}$$

We use a 25:75 ratio of D_{Total} to define the size of the D_{Train} and D_{Test}, respectively. D_{Train} is used to train the model using a supervised learning technique. The second part, D_{Test}, is used to test the model performance. We compare the actual label Y_i to the predicted label by the model represented as $\hat{Y}_i$.

6.2.3 *IoT System Architecture*

In this work, we use AWS to create the desired architecture for the implementation. We introduce the main services used in this section and briefly describe each service [114].

- Amazon Elastic Compute Cloud (Amazon EC2): providing scalable computing capacity in the AWS Cloud. Users have total control over the EC2 configuration as it is not an AWS-managed service.
- Virtual Private Cloud (Amazon VPC): creating a logically isolated virtual network when launching AWS resources. This service provides an extra layer of security to the implementation by using public and private subnets and Network Access Control List configuration.
- AWS IoT Core: enabling IoT devices connected to the AWS cloud. It supports a large number of devices and messages by providing reliable and secure services.
- Device Gateway: the entry point for IoT devices connected to AWS
- AWS Lambda: running programming code in a serverless computing service in response to events and automatically managing the underlying computing resources.
- Cloudwatch: monitoring and management services for AWS resources.

In this implementation, since we have total control over the configuration of the EC2, we use it to configure the IoT device to be connected to the network. We create five different system architectures to run this experiment to mirror real system scenarios and conduct a performance analysis. The system configuration and the memory size range are inspired by the IoMT devices presented in [115] for heart monitoring. First, the volume size represents the memory size of the IoT device used in each system. The memory size ranges between 30GB and 128GB. Since IoT devices are lightweight with small memory and computation capacity, we only increase the volume size to 128GB to fit the IoT devices' requirements and mimic real-life scenarios. Second, the value of D_{Total} ranges between 12,000 to 40,000 records. This is decided based on the memory capacity, method performance, and CPU utilization. In all five systems, we use only one memory and one CPU in the IoT device. We set the memory performance to moderate to low and use only the TCP protocol. This setup mimics the lightweight with low memory capacity IoT devices often connected to the network.

Next, we explain the setup and the variable values for each system. We set the values for the following variables in Table 6.2: volume size, D_{Total}, and the number of devices to be connected to the network.

Table 6.2: IoT System Specifications

	System1	**System2**	**System3**	**System4**	**System5**
Volume Size	30GB	32GB	64GB	128GB	30GB
D_{Total}	12,000	14,000	28,000	40,0000	24,000
Number of IoT devices	One	One	One	One	Two

We use the system architecture shown in Fig. 6.1 for the first four systems, where we connect only one IoT device. In the fifth setup of this implementation (System 5), we connect two IoT devices to the IoT core, as shown in Fig. 6.2. As demonstrated in the Figure, The traffic goes both ways between the two IoT devices. The algorithm is implemented and trained on one IoT device, and then it is tested using the traffic coming from the IoT Core and the other IoT device. Since the load is divided between two devices in this setup, we use the total number of D_{Total} in one of the IoT devices in the detector generation stage of the method. Then we use the total number of D_{Total} in the other IoT device in the detection stage. Meaning D_{Train} and D_{Test} both have the size of 12,000 records.

6.2.4 *Problem Formulation*

This work examines the realistic implementation of AIS solutions in constrained IoT systems. The objective of malware detection mechanisms is surely to maximize the detection rate of unknown malware while reducing false alarms (when benign files are wrongly classified as malicious).

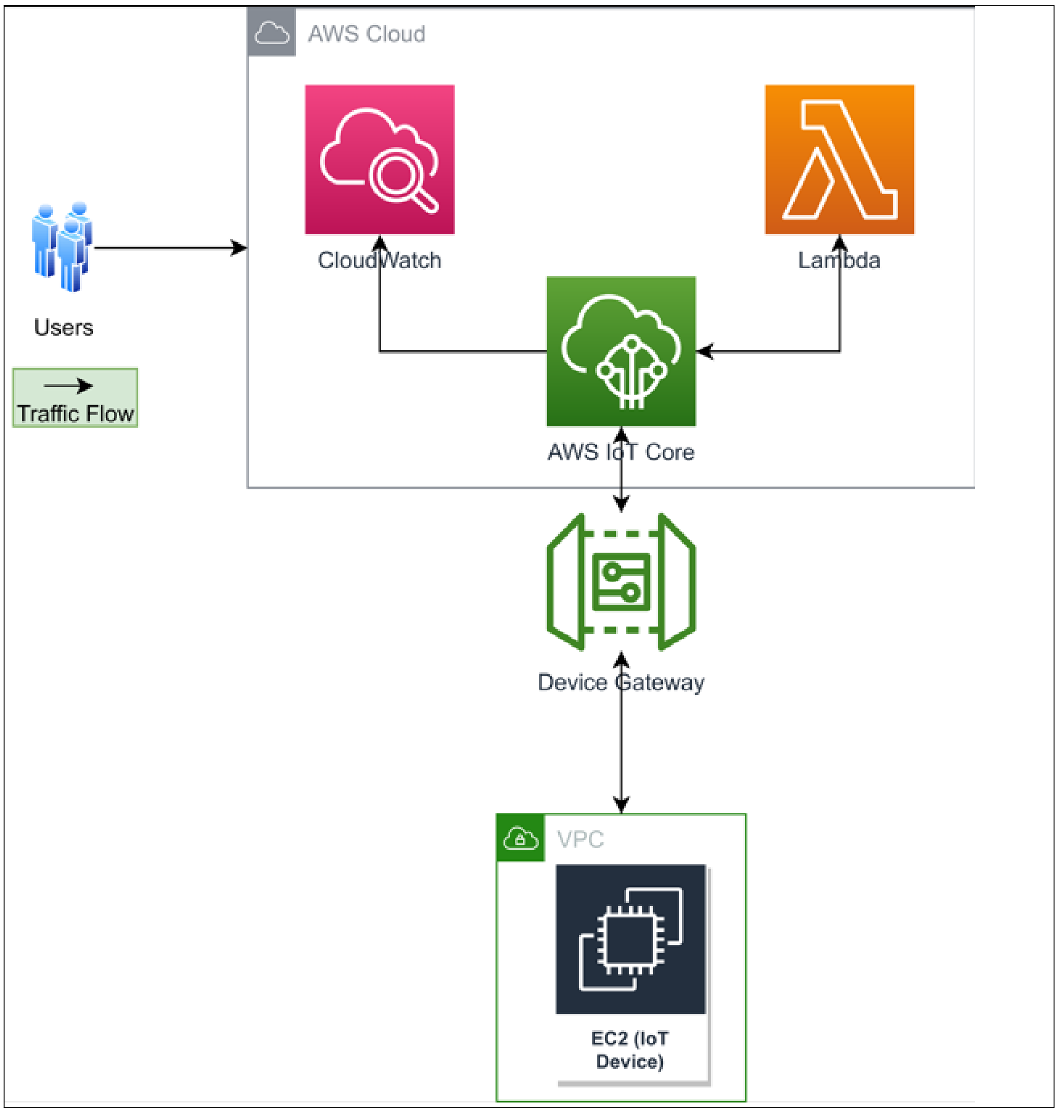


Figure 6.1: AWS Architecture Implementation-1-

Meaning, we maximize the number of correct predictions ($\hat{Y}$) by the AIS solutions as:

$$\hat{Y} == Y \tag{6.6}$$

In this work, we study how the different parameters of AIS algorithms can be tuned to accommodate the given constrained conditions of the IoT systems. While we still achieve a high detection performance of unknown malware. In particular, we find the suitable number of positive and negative detectors, D_{Pos} and D_{Neg}, the size of each detector D_{Size}, and the possible size of D_{Train} (indicated as $|D_{Train}|$), that would allow the highest number of correct predictions. We define the minimum and the maximum number of detectors as ND function, and the minimum and the maximum

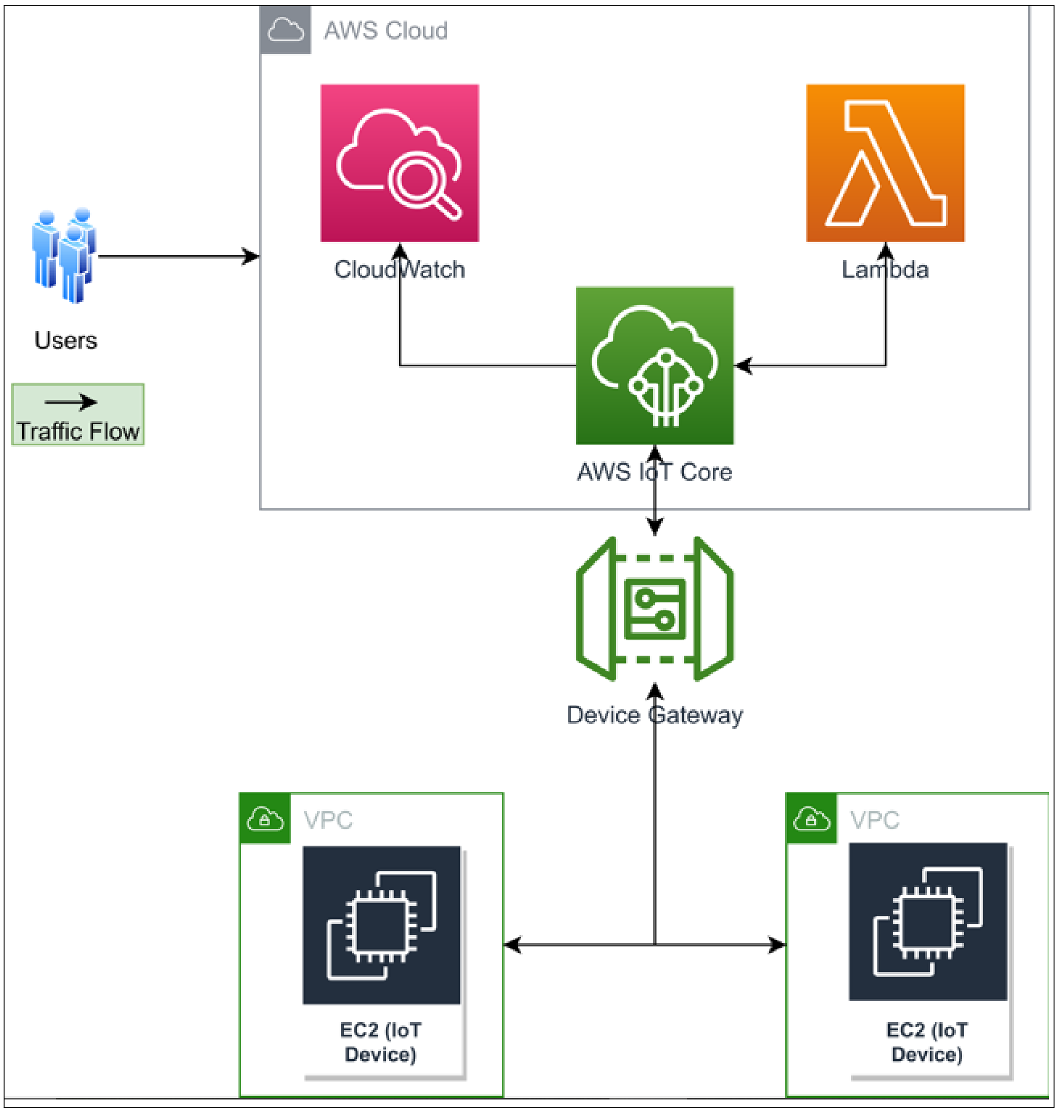


Figure 6.2: AWS Architecture Implementation-2-

number of D_{Train} as NT. Equation (6.7) presents the problem formulation and the optimization constraint defined by the total memory TM.

$$\max_{|D_{Train}|, D_{Pos}, D_{Neg}, D_{Size}} \sum_{k=1}^{|D_{Test}|} \hat{Y}_k == Y_k \tag{6.7}$$

s.t.

$$ND((D_{pos} + D_{neg}) \times D_{size}) + NT(D_{Train} \times S) <= TM \tag{7.7.(a)}$$

6.3 THE IMPLEMENTATION RESULTS AND DISCUSSION

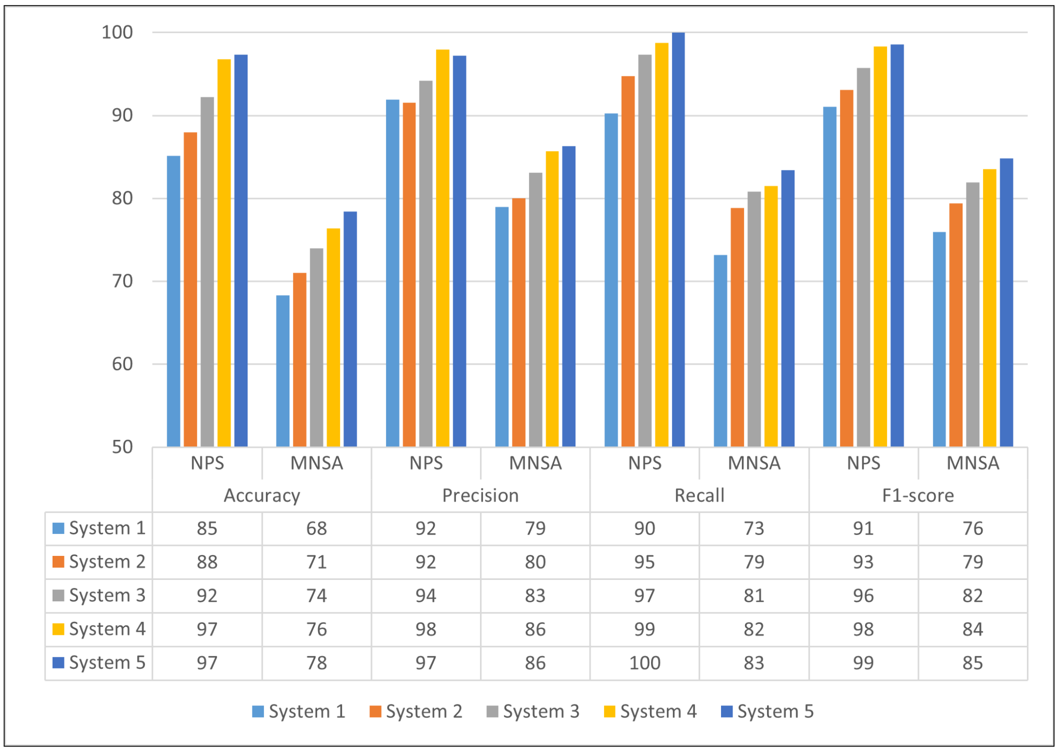
This section presents the results of our implementation, followed by a discussion and interpretation. We calculate the detection accuracy, precision, recall, and F1-Score using the equations presented in Chapter 5. An important characteristic in the context of malware detection is to reduce false alarms, and this metric is referred to as detection recall. An increase in false alarms may slow down the data acquisition process and may affect the acceptance of a malware detection algorithm. Also, it is critical that any algorithm successfully identifies all malicious files as malware, and this metric is referred to as detection precision. To this end, the F1-score of the proposed methods is measured as this captures the accuracy of detecting malware and the rate of false alarms jointly. The detection performance results are presented in Section 6.3.1. Then, we present the CPU Utilization for the NPS and MNSA in Section 6.3.2. Finally, we present the implementation and simulation performance analysis in Section 6.3.3.

6.3.1 *Detection Performance*

The figure below (Fig. 6.3) shows the results for the five system scenarios implemented in this project for both the NPS and MNSA. Since the F1-Score takes both negative and positive detection into account, we use it as the main metric to evaluate the performance in this analysis. First, we start with Systems 1 to 4, using only one IoT device in the implementation. As predicted in Chapters 4 and 5, the NPS succeeds in better malware detection than MNSA in all four systems, as evidenced by the higher F1-Score by up to 20% than the MNSA. As anticipated, the performance of both the NPS and MNSA improves when we move from System 1 to 4 by 8% and 10% for the NPS and MNSA, respectively. Increasing the volume size allows for an increase in the number of D_{Train}. Therefore, this results in better detection performance in all four systems for both methods.

In System 5, we connect two IoT devices to the system. This system architecture is used to evaluate the method's transfer learning abilities within the network. Each IoT device has its own incoming data traffic from the IoT gateway and/or other IoT devices connected to the same network. Transfer learning in this context means testing the method's ability to be trained and create detectors using an IoT device traffic, then running the detection stage using another IoT device traffic. The aim is to test the generated detectors' ability to detect malware attacks targeting both IoT devices connected to the system with being only trained using one of the IoT devices. Since we train the method on one IoT device, we can use the total number of D_{Total} to create the detectors. Both AIS solutions, the NPS and MNSA, show the capability of transfer learning and protecting both IoT devices connected to the system. However, as anticipated, the NPS succeeds in a better detection performance protecting the two IoT devices than the MNSA by 16%. This shows that the NPS has better transfer learning abilities than the MNSA. Therefore, the NPS is better for protecting distributed and robust IoT systems.

We demonstrate the NPS and MNSA detection performance and lightweight abilities on one IoT device using Systems 1 to 4. System 5 demonstrates the transfer learning abilities. In this implementation, by increasing the volume size of the device, we can use a larger number of dataset records in the detector generation stage. This results in a better learning curve, thus, a better classification accuracy when detecting malware files. Furthermore, AIS solutions can secure the IoT system with multiple IoT devices if only installed on one IoT device, as demonstrated in System 5.

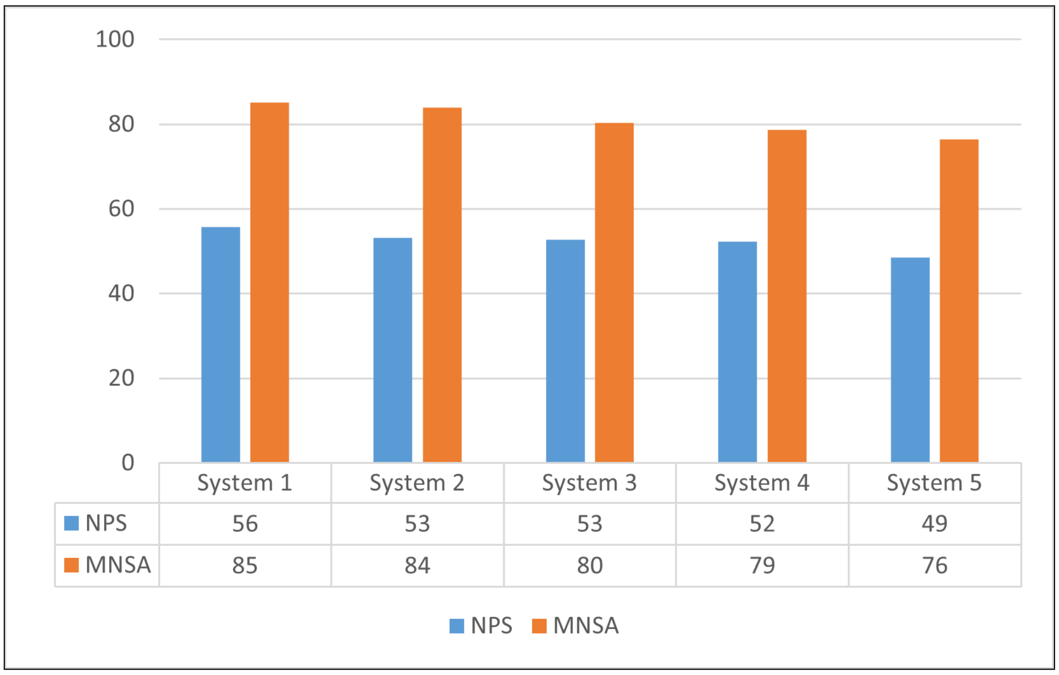


	Accuracy		Precision		Recall		F1-score	
	NPS	MNSA	NPS	MNSA	NPS	MNSA	NPS	MNSA
System 1	85	68	92	79	90	73	91	76
System 2	88	71	92	80	95	79	93	79
System 3	92	74	94	83	97	81	96	82
System 4	97	76	98	86	99	82	98	84
System 5	97	78	97	86	100	83	99	85

Figure 6.3: The NPS Implementation Using AWS Results

6.3.2 *CPU Utilization*

One of the AWS services we use in this implementation is cloudwatch which is for monitoring and managing services. Cloudwatch shows the CPU Average utilization for AWS resources, the implemented IoT devices in this case. The objective is to validate the claims in Chapters 4 and 5 that AIS solutions are lightweight in realistic settings. To this end, the CPU utilization is measured in each system for both the NPS and MNSA, as shown in Fig. 6.4. The NPS requires less CPU utilization by up to 37% than the MNSA, hence is more suitable for lightweight IoT systems under all conditions depicted by all five systems. The results also show that the CPU utilization drops systematically when moving from System 1 to System 5 for both the NPS and MNSA. We see a decrease close to 13% and 10% in CPU utilization for the NPS and MNSA, respectively. By increasing the volume size of the IoT device, we can decrease the CPU utilization in the device, which is one of the main objectives when implementing security methods in IoT devices.



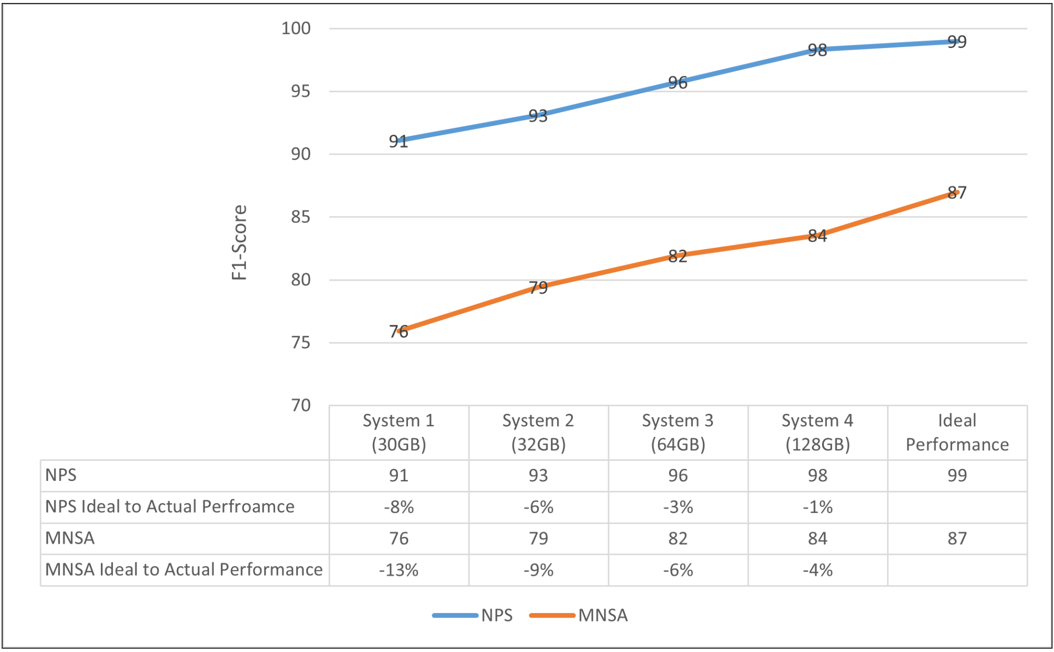
	System 1	System 2	System 3	System 4	System 5
NPS	56	53	53	52	49
MNSA	85	84	80	79	76

Figure 6.4: CPU Average Utilization for Each System

6.3.3 *Algorithm Simulation and Implementation Performance Analysis*

In this section, we compare the actual implementation results shown in Fig. 6.3 and the ideal simulation results in Chapters 4 and 5. In this work, we use the F1-Score as the main factor to compare the implementation and the simulation results of the AIS solutions. In this implementation, The size of $D_{Pos+Neg}$ when we run the NPS is 60 detectors, and it is 170 detectors for the MNSA. Consequently, to present an accurate quantitative analysis, we compare the results for the same size of detectors in the simulation. Moreover, since the simulation is run as one entity, we analyze the performance of the first four systems architectures where we run the implementation using one IoT device. Fig. 6.5 shows the analysis of the results for both the ideal simulation and the actual implementation results for the NPS and MNSA.

We compare the simulation results with System 1 results since we use one IoT device with the most constrained memory size. The MNSA shows a decrease close to 13% from the simulation and System 1 results, while the NPS performance decreases by 8% when compared to the simulation results. As anticipated, both AIS solutions, the NPS and the MNSA, underperformed compared to their simulation results. This validates the claim that IoT security solutions should be tested in a realistic environment to demonstrate sufficient findings.



	System 1 (30GB)	System 2 (32GB)	System 3 (64GB)	System 4 (128GB)	Ideal Performance
NPS	91	93	96	98	99
NPS Ideal to Actual Perfroamce	-8%	-6%	-3%	-1%	
MNSA	76	79	82	84	87
MNSA Ideal to Actual Performance	-13%	-9%	-6%	-4%	

Figure 6.5: Comparing the Actual to the Ideal Performance of AIS for Different IoT Systems

6.4 SUMMARY OF IMPLEMENTING AIS USING AWS

In the previous chapters, we demonstrated that AIS solutions are an excellent fit to secure IoT systems. However, the published results are based on computer simulations only. This chapter presents the first implantation of AIS solutions using AWS. We use AWS to create different IoT system architectures mirroring real-life scenarios. We demonstrate the lightweight, transfer learning, and detection capabilities of the AIS methods. The results show that increasing the size of the IoT device memory allows us to increase the dataset size to train the module, which leads to better detection performance. The results also show that running an AIS solution on one of the IoT devices could secure the device itself from malware attacks and any other IoT devices connected to the same network. Finally, we validate the claim that AIS security solutions should be tested in a real setup to obtain accurate results.

7 IMPLEMENTING THE NPS IN REALISTIC IOT SYSTEMS USING AWS -2-

7.1 INTRODUCTION TO THE IMPLEMENTATION METHODOLOGY

In this chapter, we investigate the transfer learning abilities of the NPS across IoT networks. Transfer learning or adaptive learning is defined in [116] as utilizing the knowledge from one or more site to facilitate learning in a targeted site. In this context, we examine adapting the knowledge by a different IoT device connected to a different IoT network. We do not only deploy the module from one site to another; rather, we build on the existing knowledge. The first layer of the detection stage of the NPS is a memory of previously detected malware files. Therefore, once the NPS is run on one site, it collects information and adds that knowledge to the memory. Then, when the module is transferred to another IoT device to test incoming traffic, it first matches the file with its own memory of malware files before it is matched with the negative and positive detectors. We create two IoT systems with the same architecture presented in 6.1. We use the first IoT system to run the NPS and create the required number of detectors in both sets, D_{Pos} and D_{Neg}. Then, we copy the created detectors into another IoT device connected to the second IoT system and run the detection stage. The objective of this experiment is to examine the following:

- The knowledge transfer of the NPS across networks
- Maximizing the malware detection performance of the NPS
- Minimizing the average utilization of the CPU in the IoT device
- The fault-tolerance abilities of the NPS in case of a network failure

We do that by following these steps:

- Optimizing the number of detectors created and copied by the NPS, which means less processing time for both devices
- Minimizing the CPU average utilization by using different IoT devices to run each stage, detectors generation stage, and detection stage
- Maximizing the number of D_{Train} for a better learning curve. Thus, better detection performance

7.2 PROBLEM FORMULATION

In this chapter, we continue the experiment of implementing the NPS in a realistic setup. We follow the implantation methodology, dataset, and problem formulation presented in Sections 6.2.1, 6.2.2, and 6.2.4. For the problem formulation, the objective is still to maximize the number of correctly predicted labels by optimizing the D_{Train} and the number of detectors. Unlike the experiment in Chapter 6, in this experiment, we optimize not only the number of created detectors but also the number of copied/transferred detectors. To this end, we present the number of created detectors as N_{CRD} and the number of copied detectors as N_{COD}. Also, we refer to the first IoT device's total memory as TM_{IoT1} and TM_{IoT2} for the second IoT device. We present the problem formulation for this experiment below:

$$\max_{|D_{Train}|, N_{CRD}, N_{COD}, D_{Size}} \sum_{k=1}^{|D_{Test}|} \hat{Y}_k == Y_k \tag{7.1}$$

$$\text{s.t.}$$

$$N_{CRD}((D_{pos} + D_{neg}) \times D_{size}) + (D_{Train} \times S) <= TM_{IoT1} \tag{8.1.(a)}$$

$$N_{COD}((D_{pos} + D_{neg}) \times D_{size}) + (D_{Test} \times S) <= TM_{IoT2} \tag{8.1.(b)}$$

7.3 COPYING DETECTORS RESULTS AND DISCUSSION

We use Equations 5.16,5.17,5.18, and 5.21 presented in Chapter 5 to calculate the presented results.

7.3.1 *Experiments Results*

(a) **Copying the same number of created detectors** This experiment aims to evaluate the transfer of the knowledge of the NPS and make the method as lightweight as it could be. We create 20 detectors in each set during the training stage of the NPS on one IoT device, then copy all 20 detectors to run the detection stage of the NPS on another IoT device. Next, we create 30 detectors in each set, then copy all 30 detectors to run the detection stage. We use 25:75 for D_Train and D_Test, respectively.

When we run the detection stage for both 20 and 30 detector sets, we get similar results to training and testing the method on the same IoT device for both sets in Chapter 6. However, the CPU average utilization of the IoT device in the detection stage is lower in both experiments, with 20 and 30 detector sets (see Table 7.1 and Table 7.2).

Table 7.1: CPU Average Utilization for 20 Detectors

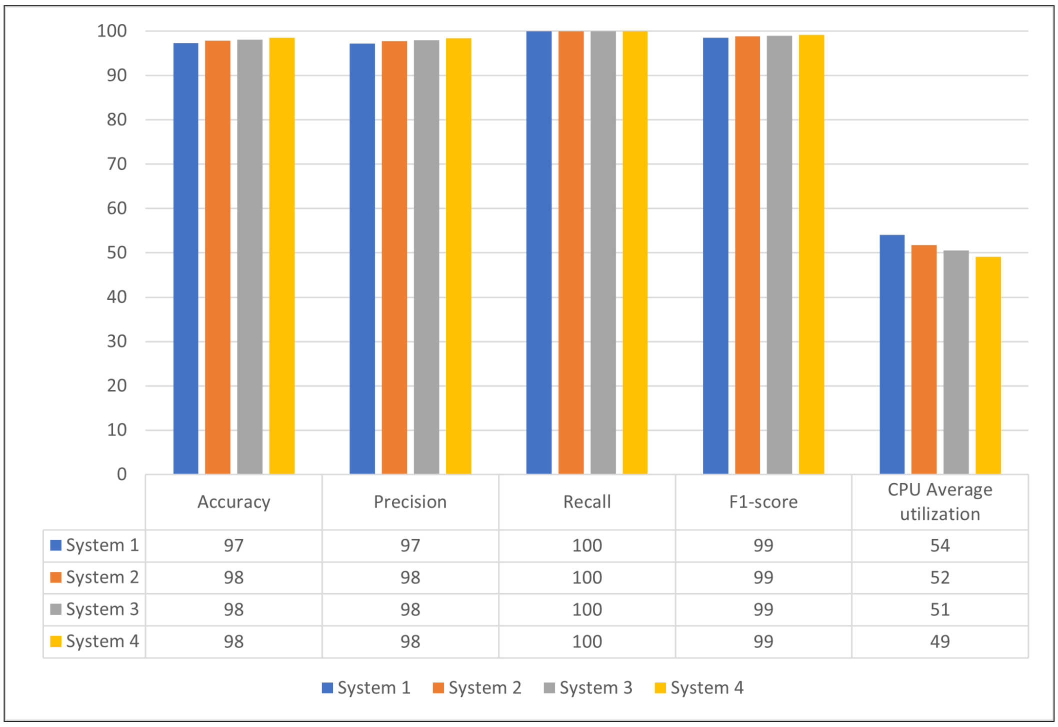
System 1	51%
System 2	49%
System 3	48%
system 4	45%

Table 7.2: CPU Average Utilization for 30 Detectors

System 1	54%
System 2	52%
System 3	50%
system 4	49%

(b) **Copying the same number of created detectors with a larger number of records used in D_{Train}** Since creating 30 detectors in each set results in better detection performance, we experiment more with copying detectors from one IoT device to another. The only thing we change here is the number of records used to train the method in D_Train. Instead of doing 25:75 of the dataset records, we use 100% of the total number of records in D_{Total} to train the method. Also, we use 100% of D_{Total} in the second IoT device for D_{Test}.

The CPU average utilization is the same as copying 30 detectors (Table 7.2). The figure below shows the performance results for the four IoT system architectures (Fig. 7.1).



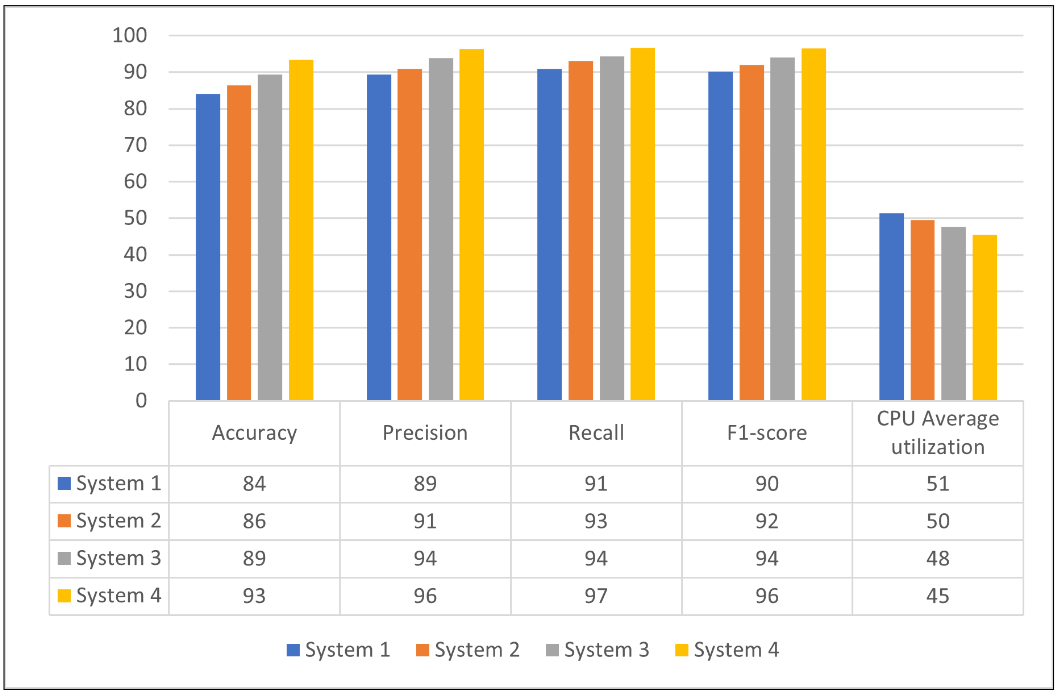
	Accuracy	Precision	Recall	F1-score	CPU Average utilization
🟦 System 1	97	97	100	99	54
🟧 System 2	98	98	100	99	52
⬜ System 3	98	98	100	99	51
🟨 System 4	98	98	100	99	49

Figure 7.1: Copying the Same Number of Generated Detectors Results

(c) **Copying Partial Number of Detectors** We experiment more with the 30 detector sets. In this experiment, we also use 100% of the total dataset record in D_{Train} and D_{Test}. We create 30 detectors in each set, negative and positive, then copy only a partial number of detectors to another IoT device to run the detection stage. We copy 20 detectors in each set out of the total number of 30 detectors. The figure below (Fig 7.2) shows the performance results of copying 20 detectors out of the created 30 detectors.

7.3.2 *Discussion*

From the first experiment, creating the required number of detectors on one IoT device and then copying that exact number to another IoT device is very efficient. The NPS achieved the same performance as it was installed and trained on the same device. More importantly, as anticipated, we reduced the CPU average utilization to 45% of the total capacity. This proves the claim that the NPS is adaptive and can transfer knowledge with excellent detection results. Next, when we create 30 detectors in each set, D_{Pos} and D_{Neg}, with the maximum number of records in D_{Train}, we achieve



	Accuracy	Precision	Recall	F1-score	CPU Average utilization
🟦 System 1	84	89	91	90	51
🟧 System 2	86	91	93	92	50
⬜ System 3	89	94	94	94	48
🟨 System 4	93	96	97	96	45

Figure 7.2: Copying Partial Number of the Generated Detectors Results

a 100% detection recall rate and up to 99% F1-Score. These remarkable detection results show that the NPS achieved zero false negative detection and almost correctly labeled all the records in D_{Test} with low CPU average utilization.

In the last experiment, the NPS detection performance is better than creating 20 detectors but less than creating 30 detectors because of the increased number of records in D_{Train}. The CPU average utilization is still low, as we aimed. The CPU average utilization is similar to copying 20 detectors out of the created 20 detectors (Table 7.1), which means it is lightweight with better malware detection performance.

The results for all the presented experiments demonstrate the method's ability to detect unknown malware attacks with low computational power. Moreover, creating detectors using one IoT device and running the detection stage on another IoT device demonstrate the method's ability to transfer knowledge with high detection performance and low memory utilization. Also, the results show that the NPS is fault-tolerant. This means that if part of the network fails, another part of the network will still be able to detect any malware attack during the failure.

7.4 SUMMARY OF IMPLEMENTING AIS USING AWS -2-

We present the implementation of the NPS in realistic IoT systems. This chapter demonstrates the NPS's transfer learning across different IoT networks and fault tolerance abilities. We use the same system architectures created in Chapter 6 and the same dataset. The objective is to investigate the AIS solutions' adaptivity by maximizing the malware detection performance and minimizing the CPU average utilization of the IoT device. We run multiple experiments, creating the detectors on one IoT device connecting to the first IoT system. Then, we optimize copying the created detectors to another IoT device connected to a different IoT system.

The performance analysis shows an excellent ability to transfer knowledge across IoT networks while achieving a remarkable detection performance. We achieve a 100% detection recall rate, which means zero false negatives. Also, we achieve up to 99% F1-Score and minimize the CPU average utilization to less than 45% of its total capacity. As anticipated, the NPS is adaptive, lightweight, and fault-tolerant, which makes it a strong candidate to secure IoT systems.

8 CONCLUSION AND FUTURE RESEARCH DIRECTIONS

8.1 RESEARCH CONCLUSION

The number of IoT devices is increasing rapidly, and the amount of data exchanged between these devices is enormous and expected to reach 175 zettabytes by 2025 [117]. Nevertheless, cyberattacks are rapidly increasing alongside the adoption of IoT applications. A malware attack is a form of cyberattack in the IoT, consisting of malicious software executed without user permission. Hackers continuously develop new malware files that are hard to detect using common techniques such as anti-virus applications. To this end, it is vital to devise a real-time IoT security solution that addresses this challenge adequately.

8.1.1 *Research Structure*

To secure the IoT systems, the method should be lightweight, adaptive, distributed, and fault-tolerant to meet the IoT requirements as presented in Chapters 2 and 4. Therefore, we present the NPS method for unknown malware detection that fits the IoT systems' requirements. We demonstrate the NPS's high detection performance with low computational power. Then, we validate the obtained results using multiple real-time datasets in Chapter 5. Next, we demonstrate the method's distribution ability in Chapter 6. Finally, we demonstrate the method's adaptivity and fault-tolerance abilities in Chapter 7.

8.1.2 *Research Discussion*

In this research, we investigate the AIS solutions for malware detection the IoT systems. We present the NPS algorithm based on negative and positive selection techniques to improve the unknown malware detection rate. We achieve a 21% increase in the accuracy of detection rate using 16-bit strings with an almost 65% decrease in space complexity, which resolves one of the main challenges in securing the IoT. Another limitation to be overcome by the NPS is the false negative detection which was reduced to almost 0% of detected files resulting in a 99% recall rate.

Next, we validate the results of the NPS using multiple real-time datasets, and we benchmark the results against state-of-the-art methods. We initially use the NSL-KDD dataset to evaluate the NPS, and then we use the UNSW-NB15 and the Bot-IoT datasets. The NPS performance is remarkable using all three datasets with a negligible difference due to the different number of records and types of attacks in each one of the datasets.

Then, we implement the AIS solutions in a realistic setup. We present the implantation of the NPS method using AWS. We create different system architectures mirroring real-life scenarios to demonstrate the method's lightweight, distribution, and detection capabilities. The results show that running the algorithm on one of the IoT devices could secure the device itself from malware attacks and any other IoT devices connected to the same network. We achieve up to 97% detection accuracy rate, 100%, and 97% recall and precision rate, respectively, which test the method's sensitivity and reliability. Furthermore, we obtain a high F1-Score up to 98% and reduce the average CPU use in the IoT device to less than 50% of its total capacity.

Finally, we conduct a performance analysis for the different systems implemented. The results show better detection performance by increasing the volume size of the IoT device and the number of dataset records used to generate detectors. Furthermore, we run different experiments using AWS to demonstrate the method's adaptivity, transfer of knowledge, and fault tolerance abilities. We create detectors using one IoT device and copy the detectors to another IoT device. We achieve high detection performance up to 99%. We also manage to reduce the CPU average utilization to almost 45% of its total capacity. The method is proven effective and outperforms the state-of-the-art methods in detecting malware files with less computational power needed.

8.1.3 *Limitations and Future Research Directions*

As with any research work, our project has some limitations. One of the main limitations of this project was the lack of conducted experiments using AIS for malware detection in IoT systems. Although this topic is well-researched theoretically with exceptionally promising research directions, implemented solutions in IoT environments are limited. Therefore, in this research, we thoroughly analyze the state-of-the-art solutions and present detailed performance analysis, both theoretically and pragmatically, to be a reference to build on for future AIS implementation in the IoT. The second limitation of this research is the IoT devices' constrained resources, making the implementation for research purposes especially more challenging. For instance, using a large number of dataset records to train the module on an IoT device with small memory and computing power is not feasible. Therefore, we suggest training the module with larger dataset records and perhaps different malware attack types on the IoT gateway for future research since it usually has more computing power. Then, deploying the module to be tested on constrained IoT devices. Next, in this research, we adjust the detectors and detector sets sizes to present the analysis of detection accuracy with regard to the module complexity. Even though we achieve a high detection rate with small space complexity to fit the IoT systems requirements, the NPS still has high time complexity. Consequently, more research could be done to adapt different detector sizes to improve the time complexity of the module. In addition to the time and space complexity, the features used to test the AIS solutions in this project are also one of the limitations. Although we succeeded in achieving a high detection rate, the NPS method is implemented and tested on the destination port. Therefore, more research could be done on other network traffic features to detect different types of attacks to improve the classification results. Moreover, in the implementation of the NPS, we use only one or two IoT devices connected to the system. More research could be done connecting more IoT devices to the same network to investigate further the method's robustness to meet the IoT systems' requirements. Finally, in this research, we use AWS to implement the NPS. Even though we mimic real-life IoT systems, for future research, physical IoT networks could be used to test and run the module in an environment with different challenges, such as connectivity and access control.

A

APPENDIX 1

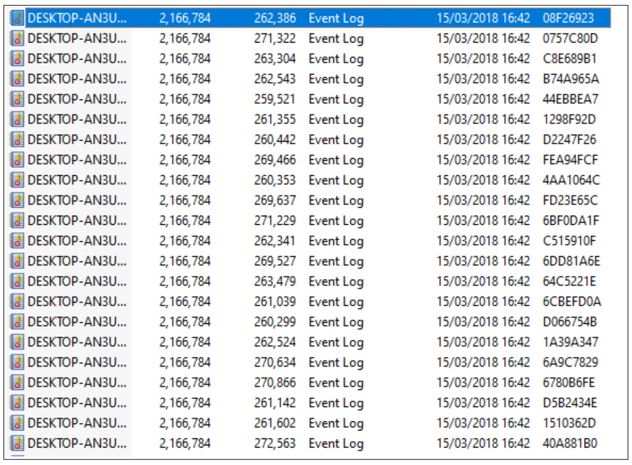
This appendix contains the features extraction process for the NSL-KDD, Bot-IoT, and UNSW-NB15 Datasets used in the implementation of this project.

A.1 NSL-KDD DATASET

To extract features needed from the NSL-KDD dataset, we used CICFlow-Meter [118] which is a network traffic flow generator. Comma-separated values files are generated with more than 80 network traffic features including destination port, source IP, flow ID, and protocol. As advised by the dataset authors, we install the Amazon Web Services Command Line Interface (AWS CLI) and run the following command:

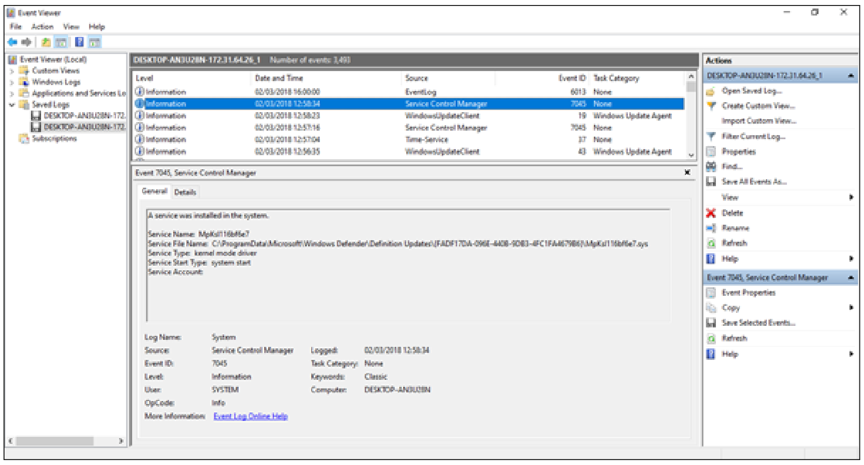
$awss3sync - -no - sign - request - -regioneu - west - 2$
$''s3 : //cse - cic - ids2018/''dest - dir$

That enable us to download events log files and extracted features to use in Machine Learning (ML) -saved in .csv format- as shown in the following figures (Fig. A.1 - A.5).



DESKTOP-AN3U...	2,166,784	262,386	Event Log	15/03/2018 16:42	08F26923
DESKTOP-AN3U...	2,166,784	271,322	Event Log	15/03/2018 16:42	0757C80D
DESKTOP-AN3U...	2,166,784	263,304	Event Log	15/03/2018 16:42	C8E689B1
DESKTOP-AN3U...	2,166,784	262,543	Event Log	15/03/2018 16:42	B74A965A
DESKTOP-AN3U...	2,166,784	259,521	Event Log	15/03/2018 16:42	44EBBEA7
DESKTOP-AN3U...	2,166,784	261,355	Event Log	15/03/2018 16:42	1298F92D
DESKTOP-AN3U...	2,166,784	260,442	Event Log	15/03/2018 16:42	D2247F26
DESKTOP-AN3U...	2,166,784	269,466	Event Log	15/03/2018 16:42	FEA94FCF
DESKTOP-AN3U...	2,166,784	260,353	Event Log	15/03/2018 16:42	4AA1064C
DESKTOP-AN3U...	2,166,784	269,637	Event Log	15/03/2018 16:42	FD23E65C
DESKTOP-AN3U...	2,166,784	271,229	Event Log	15/03/2018 16:42	6BF0DA1F
DESKTOP-AN3U...	2,166,784	262,341	Event Log	15/03/2018 16:42	C515910F
DESKTOP-AN3U...	2,166,784	269,527	Event Log	15/03/2018 16:42	6DD81A6E
DESKTOP-AN3U...	2,166,784	263,479	Event Log	15/03/2018 16:42	64C5221E
DESKTOP-AN3U...	2,166,784	261,039	Event Log	15/03/2018 16:42	6CBEFD0A
DESKTOP-AN3U...	2,166,784	260,299	Event Log	15/03/2018 16:42	D066754B
DESKTOP-AN3U...	2,166,784	262,524	Event Log	15/03/2018 16:42	1A39A347
DESKTOP-AN3U...	2,166,784	270,634	Event Log	15/03/2018 16:42	6A9C7829
DESKTOP-AN3U...	2,166,784	270,866	Event Log	15/03/2018 16:42	6780B6FE
DESKTOP-AN3U...	2,166,784	261,142	Event Log	15/03/2018 16:42	D5B2434E
DESKTOP-AN3U...	2,166,784	261,602	Event Log	15/03/2018 16:42	1510362D
DESKTOP-AN3U...	2,166,784	272,563	Event Log	15/03/2018 16:42	40A881B0

Figure A.1: Event Log Screenshot



DESKTOP-AN3U2BN-172.31.64.26_1 Number of events: 3,493

Level	Date and Time	Source	Event ID	Task Category
Information	02/03/2018 16:00:00	EventLog	6013	None
Information	02/03/2018 12:58:34	Service Control Manager	7045	None
Information	02/03/2018 12:58:23	WindowsUpdateClient	19	Windows Update Agent
Information	02/03/2018 12:57:16	Service Control Manager	7045	None
Information	02/03/2018 12:57:04	Time-Service	37	None
Information	02/03/2018 12:56:35	WindowsUpdateClient	43	Windows Update Agent

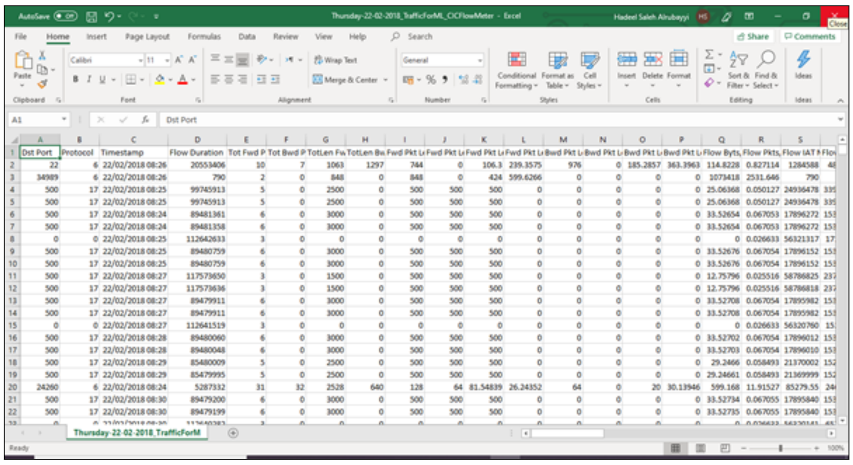
Event 7045, Service Control Manager

```
A service was installed in the system.

Service Name:  MpKsl116bf6e7
Service File Name:  C:\ProgramData\Microsoft\Windows Defender\Definition Updates\{FADF17DA-096E-4408-9DB3-4FC1FA4679B6}\MpKsl116bf6e7.sys
Service Type:  kernel mode driver
Service Start Type:  system start
Service Account:

Log Name:          System
Source:            Service Control Manager    Logged:         02/03/2018 12:58:34
Event ID:          7045                       Task Category:  None
Level:             Information                Keywords:       Classic
User:              SYSTEM                     Computer:       DESKTOP-AN3U2BN
OpCode:            Info
More Information:  Event Log Online Help
```

Figure A.2: Event Log Details Screenshot

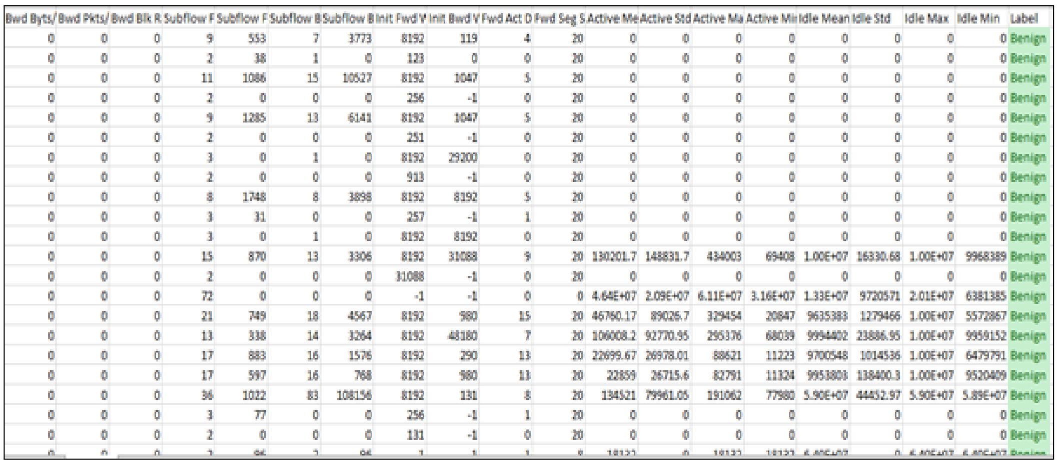


Thursday-22-02-2018_TrafficForML_CICFlowMeter - Excel

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1	Dst Port	Protocol	Timestamp	Flow Duration	Tot Fwd P	Tot Bwd P	TotLen Fw	TotLen Bw	Fwd Pkt L	Fwd Pkt L	Fwd Pkt L	Fwd Pkt L	Bwd Pkt L	Bwd Pkt L	Bwd Pkt L	Bwd Pkt L	Flow Byts,	Flow Pkts,	Flow IAT M	Flo
2	22	6	22/02/2018 08:26	20553406	10	7	1063	1297	744	0	106.3	239.3575	976	0	185.2857	363.3963	114.8228	0.827114	1284588	48
3	34989	6	22/02/2018 08:26	790	2	0	848	0	848	0	424	599.6266	0	0	0	0	1073418	2531.646	790	
4	500	17	22/02/2018 08:25	99745913	5	0	2500	0	500	500	500	0	0	0	0	0	25.06368	0.050127	24936478	335
5	500	17	22/02/2018 08:25	99745913	5	0	2500	0	500	500	500	0	0	0	0	0	25.06368	0.050127	24936478	335
6	500	17	22/02/2018 08:24	89481361	6	0	3000	0	500	500	500	0	0	0	0	0	33.52654	0.067053	17896272	153
7	500	17	22/02/2018 08:24	89481358	6	0	3000	0	500	500	500	0	0	0	0	0	33.52654	0.067053	17896272	153
8	0	0	22/02/2018 08:25	112642633	3	0	0	0	0	0	0	0	0	0	0	0	0	0.026633	56321317	17
9	500	17	22/02/2018 08:25	89480759	6	0	3000	0	500	500	500	0	0	0	0	0	33.52676	0.067054	17896152	153
10	500	17	22/02/2018 08:25	89480759	6	0	3000	0	500	500	500	0	0	0	0	0	33.52676	0.067054	17896152	153
11	500	17	22/02/2018 08:27	117573650	3	0	1500	0	500	500	500	0	0	0	0	0	12.75796	0.025516	58786825	237
12	500	17	22/02/2018 08:27	117573636	3	0	1500	0	500	500	500	0	0	0	0	0	12.75796	0.025516	58786818	237
13	500	17	22/02/2018 08:27	89479911	6	0	3000	0	500	500	500	0	0	0	0	0	33.52708	0.067054	17895982	153
14	500	17	22/02/2018 08:27	89479911	6	0	3000	0	500	500	500	0	0	0	0	0	33.52708	0.067054	17895982	153
15	0	0	22/02/2018 08:27	112641519	3	0	0	0	0	0	0	0	0	0	0	0	0	0.026633	56320760	15
16	500	17	22/02/2018 08:28	89480060	6	0	3000	0	500	500	500	0	0	0	0	0	33.52702	0.067054	17896012	153
17	500	17	22/02/2018 08:28	89480048	6	0	3000	0	500	500	500	0	0	0	0	0	33.52703	0.067054	17896010	153
18	500	17	22/02/2018 08:29	85480009	5	0	2500	0	500	500	500	0	0	0	0	0	29.2466	0.058493	21370002	152
19	500	17	22/02/2018 08:29	85479995	5	0	2500	0	500	500	500	0	0	0	0	0	29.24661	0.058493	21369999	152
20	24260	6	22/02/2018 08:24	5287332	31	32	2528	640	128	64	81.54839	26.24352	64	0	20	30.13946	599.168	11.91527	85279.55	244
21	500	17	22/02/2018 08:30	89479200	6	0	3000	0	500	500	500	0	0	0	0	0	33.52734	0.067055	17895840	153
22	500	17	22/02/2018 08:30	89479199	6	0	3000	0	500	500	500	0	0	0	0	0	33.52735	0.067055	17895840	153

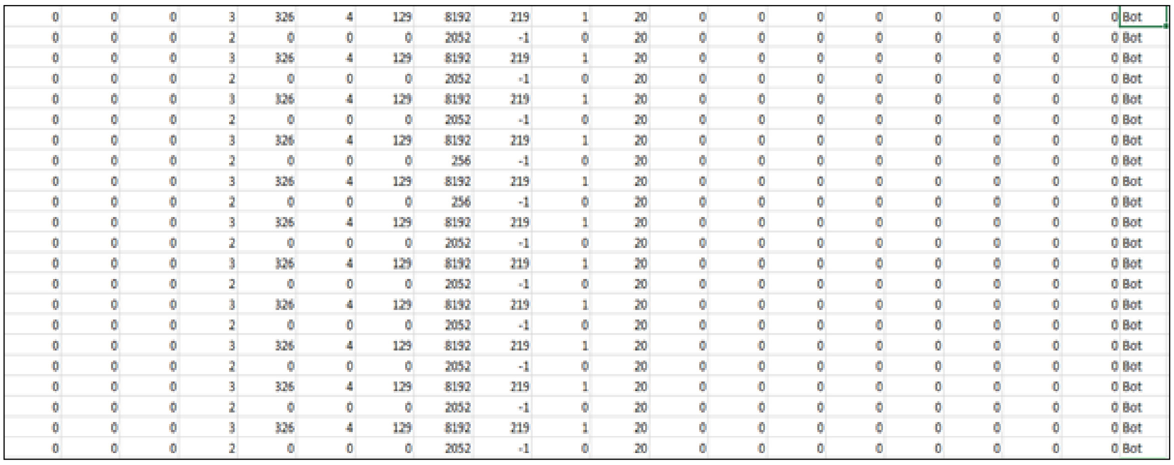
Thursday-22-02-2018_TrafficForM

Figure A.3: Some Extracted Records Screenshot-1-



Bwd Byts/	Bwd Pkts/	Bwd Blk R	Subflow F	Subflow F	Subflow B	Subflow B	Init Fwd V	Init Bwd V	Fwd Act D	Fwd Seg S	Active Me	Active Std	Active Ma	Active Mi	Idle Mean	Idle Std	Idle Max	Idle Min	Label
0	0	0	9	553	7	3773	8192	119	4	20	0	0	0	0	0	0	0	0	Benign
0	0	0	2	38	1	0	123	0	0	20	0	0	0	0	0	0	0	0	Benign
0	0	0	11	1086	15	10527	8192	1047	5	20	0	0	0	0	0	0	0	0	Benign
0	0	0	2	0	0	0	256	-1	0	20	0	0	0	0	0	0	0	0	Benign
0	0	0	9	1285	13	6141	8192	1047	5	20	0	0	0	0	0	0	0	0	Benign
0	0	0	2	0	0	0	251	-1	0	20	0	0	0	0	0	0	0	0	Benign
0	0	0	3	0	1	0	8192	29200	0	20	0	0	0	0	0	0	0	0	Benign
0	0	0	2	0	0	0	913	-1	0	20	0	0	0	0	0	0	0	0	Benign
0	0	0	8	1748	8	3898	8192	8192	5	20	0	0	0	0	0	0	0	0	Benign
0	0	0	3	31	0	0	257	-1	1	20	0	0	0	0	0	0	0	0	Benign
0	0	0	3	0	1	0	8192	8192	0	20	0	0	0	0	0	0	0	0	Benign
0	0	0	15	870	13	3306	8192	31088	9	20	130201.7	148831.7	434003	69408	1.00E+07	16330.68	1.00E+07	9968389	Benign
0	0	0	2	0	0	0	31088	-1	0	20	0	0	0	0	0	0	0	0	Benign
0	0	0	72	0	0	0	-1	-1	0	0	4.64E+07	2.09E+07	6.11E+07	3.16E+07	1.33E+07	9720571	2.01E+07	6381385	Benign
0	0	0	21	749	18	4567	8192	980	15	20	46760.17	89026.7	329454	20847	9635383	1279466	1.00E+07	5572867	Benign
0	0	0	13	338	14	3264	8192	48180	7	20	106008.2	92770.95	295376	68039	9994402	23886.95	1.00E+07	9959152	Benign
0	0	0	17	883	16	1576	8192	290	13	20	22699.67	26978.01	88621	11223	9700548	1014536	1.00E+07	6479791	Benign
0	0	0	17	597	16	768	8192	980	13	20	22859	26715.6	82791	11324	9953803	138400.3	1.00E+07	9520409	Benign
0	0	0	36	1022	83	108156	8192	131	8	20	134521	79961.05	191062	77980	5.90E+07	44452.97	5.90E+07	5.89E+07	Benign
0	0	0	3	77	0	0	256	-1	1	20	0	0	0	0	0	0	0	0	Benign
0	0	0	2	0	0	0	131	-1	0	20	0	0	0	0	0	0	0	0	Benign

Figure A.4: Some Extracted Records Screenshot-2-

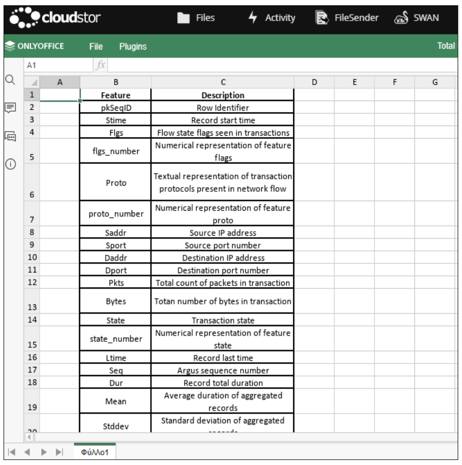


0	0	0	3	326	4	129	8192	219	1	20	0	0	0	0	0	0	0	0	Bot
0	0	0	2	0	0	0	2052	-1	0	20	0	0	0	0	0	0	0	0	Bot
0	0	0	3	326	4	129	8192	219	1	20	0	0	0	0	0	0	0	0	Bot
0	0	0	2	0	0	0	2052	-1	0	20	0	0	0	0	0	0	0	0	Bot
0	0	0	3	326	4	129	8192	219	1	20	0	0	0	0	0	0	0	0	Bot
0	0	0	2	0	0	0	2052	-1	0	20	0	0	0	0	0	0	0	0	Bot
0	0	0	3	326	4	129	8192	219	1	20	0	0	0	0	0	0	0	0	Bot
0	0	0	2	0	0	0	256	-1	0	20	0	0	0	0	0	0	0	0	Bot
0	0	0	3	326	4	129	8192	219	1	20	0	0	0	0	0	0	0	0	Bot
0	0	0	2	0	0	0	256	-1	0	20	0	0	0	0	0	0	0	0	Bot
0	0	0	3	326	4	129	8192	219	1	20	0	0	0	0	0	0	0	0	Bot
0	0	0	2	0	0	0	2052	-1	0	20	0	0	0	0	0	0	0	0	Bot
0	0	0	3	326	4	129	8192	219	1	20	0	0	0	0	0	0	0	0	Bot
0	0	0	2	0	0	0	2052	-1	0	20	0	0	0	0	0	0	0	0	Bot
0	0	0	3	326	4	129	8192	219	1	20	0	0	0	0	0	0	0	0	Bot
0	0	0	2	0	0	0	2052	-1	0	20	0	0	0	0	0	0	0	0	Bot
0	0	0	3	326	4	129	8192	219	1	20	0	0	0	0	0	0	0	0	Bot
0	0	0	2	0	0	0	2052	-1	0	20	0	0	0	0	0	0	0	0	Bot
0	0	0	3	326	4	129	8192	219	1	20	0	0	0	0	0	0	0	0	Bot
0	0	0	2	0	0	0	2052	-1	0	20	0	0	0	0	0	0	0	0	Bot
0	0	0	3	326	4	129	8192	219	1	20	0	0	0	0	0	0	0	0	Bot
0	0	0	2	0	0	0	2052	-1	0	20	0	0	0	0	0	0	0	0	Bot

Figure A.5: Some Extracted Records Screenshot-3-

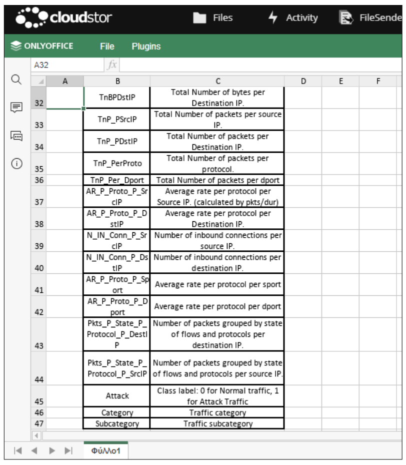
A.2 BOT-IOT DATASET

The Bot-IoT dataset records are available on [98]. We use the .CSV formatted files for this experiment. The following figures (Fig. A.6 - A.9) show some examples of the dataset records and data features that are extracted and used to run the NPS.



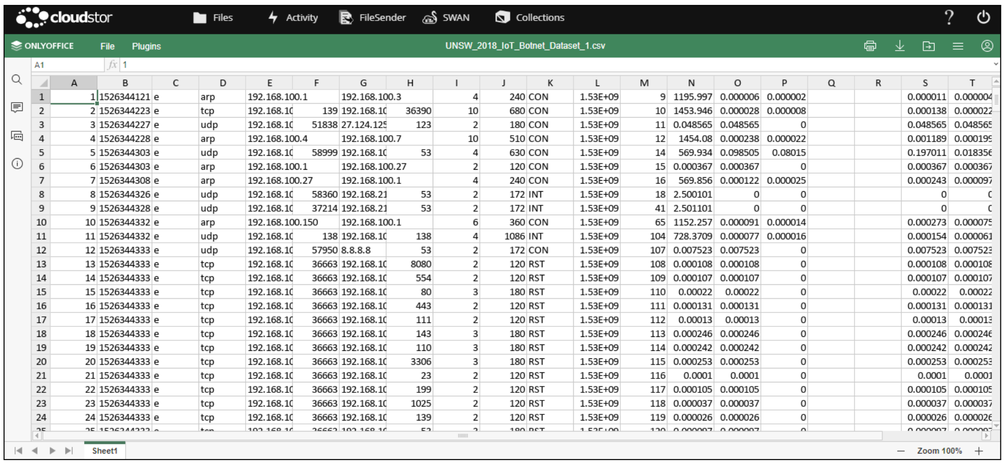
Feature	Description
pkSeqID	Row Identifier
Stime	Record start time
Flgs	Flow state flags seen in transactions
flgs_number	Numerical representation of feature flags
Proto	Textual representation of transaction protocols present in network flow
proto_number	Numerical representation of feature proto
Saddr	Source IP address
Sport	Source port number
Daddr	Destination IP address
Dport	Destination port number
Pkts	Total count of packets in transaction
Bytes	Totan number of bytes in transaction
State	Transaction state
state_number	Numerical representation of feature state
Ltime	Record last time
Seq	Argus sequence number
Dur	Record total duration
Mean	Average duration of aggregated records
Stddev	Standard deviation of aggregated

Figure A.6: Data Features Distribution Screenshot-1-



	A	B	C	D	E	F
32		TnBPDstIP	Total Number of bytes per Destination IP.			
33		TnP_PSrcIP	Total Number of packets per source IP.			
34		TnP_PDstIP	Total Number of packets per Destination IP.			
35		TnP_PerProto	Total Number of packets per protocol.			
36		TnP_Per_Dport	Total Number of packets per dport			
37		AR_P_Proto_P_SrcIP	Average rate per protocol per Source IP. (calculated by pkts/dur)			
38		AR_P_Proto_P_DstIP	Average rate per protocol per Destination IP.			
39		N_IN_Conn_P_SrcIP	Number of inbound connections per source IP.			
40		N_IN_Conn_P_DstIP	Number of inbound connections per destination IP.			
41		AR_P_Proto_P_Sport	Average rate per protocol per sport			
42		AR_P_Proto_P_Dport	Average rate per protocol per dport			
43		Pkts_P_State_P_Protocol_P_DestIP	Number of packets grouped by state of flows and protocols per destination IP.			
44		Pkts_P_State_P_Protocol_P_SrcIP	Number of packets grouped by state of flows and protocols per source IP.			
45		Attack	Class label: 0 for Normal traffic, 1 for Attack Traffic			
46		Category	Traffic category			
47		Subcategory	Traffic subcategory			

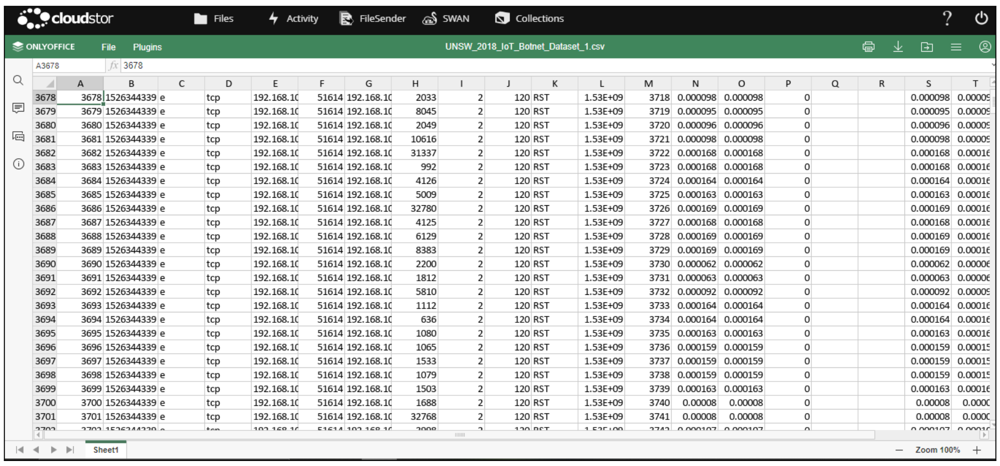
Figure A.7: Data Features Distribution Screenshot-2-



UNSW_2018_IoT_Botnet_Dataset_1.csv

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1	1	1526344121	e	arp	192.168.100.1		192.168.100.3		4	240	CON	1.53E+09	9	1195.997	0.000006	0.000002			0.000011	0.000004
2	2	1526344223	e	tcp	192.168.10	139	192.168.10	36390	10	680	CON	1.53E+09	10	1453.946	0.000028	0.000008			0.000138	0.000022
3	3	1526344227	e	udp	192.168.10	51838	27.124.125	123	2	180	CON	1.53E+09	11	0.048565	0.048565	0			0.048565	0.048565
4	4	1526344228	e	arp	192.168.100.4		192.168.100.7		10	510	CON	1.53E+09	12	1454.08	0.000238	0.000022			0.001189	0.000199
5	5	1526344303	e	udp	192.168.10	58999	192.168.10	53	4	630	CON	1.53E+09	14	569.934	0.098505	0.08015			0.197011	0.018356
6	6	1526344303	e	arp	192.168.100.1		192.168.100.27		2	120	CON	1.53E+09	15	0.000367	0.000367	0			0.000367	0.000367
7	7	1526344308	e	arp	192.168.100.27		192.168.100.1		4	240	CON	1.53E+09	16	569.856	0.000122	0.000025			0.000243	0.000097
8	8	1526344326	e	udp	192.168.10	58360	192.168.21	53	2	172	INT	1.53E+09	18	2.500101	0	0			0	0
9	9	1526344328	e	udp	192.168.10	37214	192.168.21	53	2	172	INT	1.53E+09	41	2.501101	0	0			0	0
10	10	1526344332	e	arp	192.168.100.150		192.168.100.1		6	360	CON	1.53E+09	65	1152.257	0.000091	0.000014			0.000273	0.000075
11	11	1526344332	e	udp	192.168.10	138	192.168.10	138	4	1086	INT	1.53E+09	104	728.3709	0.000077	0.000016			0.000154	0.000061
12	12	1526344333	e	udp	192.168.10	57950	8.8.8.8	53	2	172	CON	1.53E+09	107	0.007523	0.007523	0			0.007523	0.007523
13	13	1526344333	e	tcp	192.168.10	36663	192.168.10	8080	2	120	RST	1.53E+09	108	0.000108	0.000108	0			0.000108	0.000108
14	14	1526344333	e	tcp	192.168.10	36663	192.168.10	554	2	120	RST	1.53E+09	109	0.000107	0.000107	0			0.000107	0.000107
15	15	1526344333	e	tcp	192.168.10	36663	192.168.10	80	3	180	RST	1.53E+09	110	0.00022	0.00022	0			0.00022	0.00022
16	16	1526344333	e	tcp	192.168.10	36663	192.168.10	443	2	120	RST	1.53E+09	111	0.000131	0.000131	0			0.000131	0.000131
17	17	1526344333	e	tcp	192.168.10	36663	192.168.10	111	2	120	RST	1.53E+09	112	0.00013	0.00013	0			0.00013	0.00013
18	18	1526344333	e	tcp	192.168.10	36663	192.168.10	143	3	180	RST	1.53E+09	113	0.000246	0.000246	0			0.000246	0.000246
19	19	1526344333	e	tcp	192.168.10	36663	192.168.10	110	3	180	RST	1.53E+09	114	0.000242	0.000242	0			0.000242	0.000242
20	20	1526344333	e	tcp	192.168.10	36663	192.168.10	3306	3	180	RST	1.53E+09	115	0.000253	0.000253	0			0.000253	0.000253
21	21	1526344333	e	tcp	192.168.10	36663	192.168.10	23	2	120	RST	1.53E+09	116	0.0001	0.0001	0			0.0001	0.0001
22	22	1526344333	e	tcp	192.168.10	36663	192.168.10	199	2	120	RST	1.53E+09	117	0.000105	0.000105	0			0.000105	0.000105
23	23	1526344333	e	tcp	192.168.10	36663	192.168.10	1025	2	120	RST	1.53E+09	118	0.000037	0.000037	0			0.000037	0.000037
24	24	1526344333	e	tcp	192.168.10	36663	192.168.10	139	2	120	RST	1.53E+09	119	0.000026	0.000026	0			0.000026	0.000026

Figure A.8: Some Extracted Records Screenshot-1-



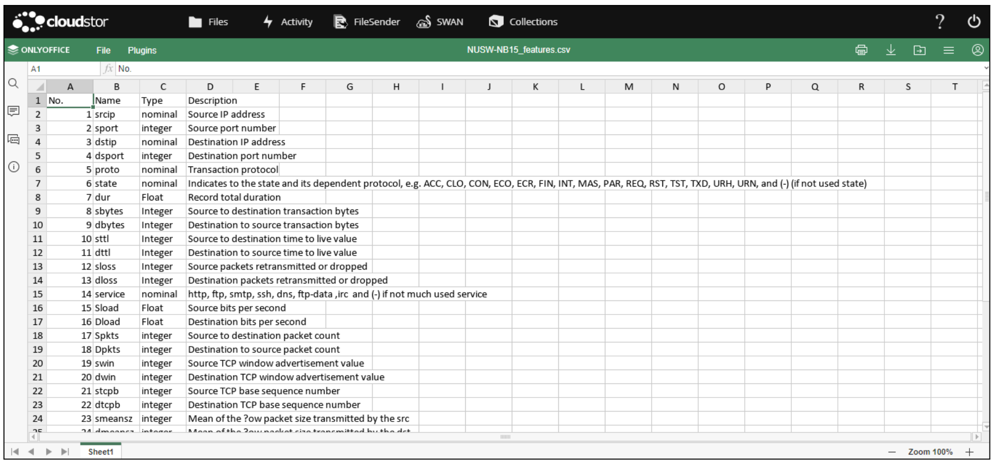
UNSW_2018_IoT_Botnet_Dataset_1.csv

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
3678	3678	1526344339	e	tcp	192.168.10	51614	192.168.10	2033	2	120	RST	1.53E+09	3718	0.000098	0.000098	0			0.000098	0.00009
3679	3679	1526344339	e	tcp	192.168.10	51614	192.168.10	8045	2	120	RST	1.53E+09	3719	0.000095	0.000095	0			0.000095	0.00009
3680	3680	1526344339	e	tcp	192.168.10	51614	192.168.10	2049	2	120	RST	1.53E+09	3720	0.000096	0.000096	0			0.000096	0.00009
3681	3681	1526344339	e	tcp	192.168.10	51614	192.168.10	10616	2	120	RST	1.53E+09	3721	0.000098	0.000098	0			0.000098	0.00009
3682	3682	1526344339	e	tcp	192.168.10	51614	192.168.10	31337	2	120	RST	1.53E+09	3722	0.000168	0.000168	0			0.000168	0.00016
3683	3683	1526344339	e	tcp	192.168.10	51614	192.168.10	992	2	120	RST	1.53E+09	3723	0.000168	0.000168	0			0.000168	0.00016
3684	3684	1526344339	e	tcp	192.168.10	51614	192.168.10	4126	2	120	RST	1.53E+09	3724	0.000164	0.000164	0			0.000164	0.00016
3685	3685	1526344339	e	tcp	192.168.10	51614	192.168.10	5009	2	120	RST	1.53E+09	3725	0.000163	0.000163	0			0.000163	0.00016
3686	3686	1526344339	e	tcp	192.168.10	51614	192.168.10	32780	2	120	RST	1.53E+09	3726	0.000169	0.000169	0			0.000169	0.00016
3687	3687	1526344339	e	tcp	192.168.10	51614	192.168.10	4125	2	120	RST	1.53E+09	3727	0.000168	0.000168	0			0.000168	0.00016
3688	3688	1526344339	e	tcp	192.168.10	51614	192.168.10	6129	2	120	RST	1.53E+09	3728	0.000169	0.000169	0			0.000169	0.00016
3689	3689	1526344339	e	tcp	192.168.10	51614	192.168.10	8383	2	120	RST	1.53E+09	3729	0.000169	0.000169	0			0.000169	0.00016
3690	3690	1526344339	e	tcp	192.168.10	51614	192.168.10	2200	2	120	RST	1.53E+09	3730	0.000062	0.000062	0			0.000062	0.00006
3691	3691	1526344339	e	tcp	192.168.10	51614	192.168.10	1812	2	120	RST	1.53E+09	3731	0.000063	0.000063	0			0.000063	0.00006
3692	3692	1526344339	e	tcp	192.168.10	51614	192.168.10	5810	2	120	RST	1.53E+09	3732	0.000092	0.000092	0			0.000092	0.00009
3693	3693	1526344339	e	tcp	192.168.10	51614	192.168.10	1112	2	120	RST	1.53E+09	3733	0.000164	0.000164	0			0.000164	0.00016
3694	3694	1526344339	e	tcp	192.168.10	51614	192.168.10	636	2	120	RST	1.53E+09	3734	0.000164	0.000164	0			0.000164	0.00016
3695	3695	1526344339	e	tcp	192.168.10	51614	192.168.10	1080	2	120	RST	1.53E+09	3735	0.000163	0.000163	0			0.000163	0.00016
3696	3696	1526344339	e	tcp	192.168.10	51614	192.168.10	1065	2	120	RST	1.53E+09	3736	0.000159	0.000159	0			0.000159	0.00015
3697	3697	1526344339	e	tcp	192.168.10	51614	192.168.10	1533	2	120	RST	1.53E+09	3737	0.000159	0.000159	0			0.000159	0.00015
3698	3698	1526344339	e	tcp	192.168.10	51614	192.168.10	1079	2	120	RST	1.53E+09	3738	0.000159	0.000159	0			0.000159	0.00015
3699	3699	1526344339	e	tcp	192.168.10	51614	192.168.10	1503	2	120	RST	1.53E+09	3739	0.000163	0.000163	0			0.000163	0.00016
3700	3700	1526344339	e	tcp	192.168.10	51614	192.168.10	1688	2	120	RST	1.53E+09	3740	0.00008	0.00008	0			0.00008	0.0000
3701	3701	1526344339	e	tcp	192.168.10	51614	192.168.10	32768	2	120	RST	1.53E+09	3741	0.00008	0.00008	0			0.00008	0.0000

Figure A.9: Some Extracted Records Screenshot-2-

A.3 UNSW-NB15 DATASET

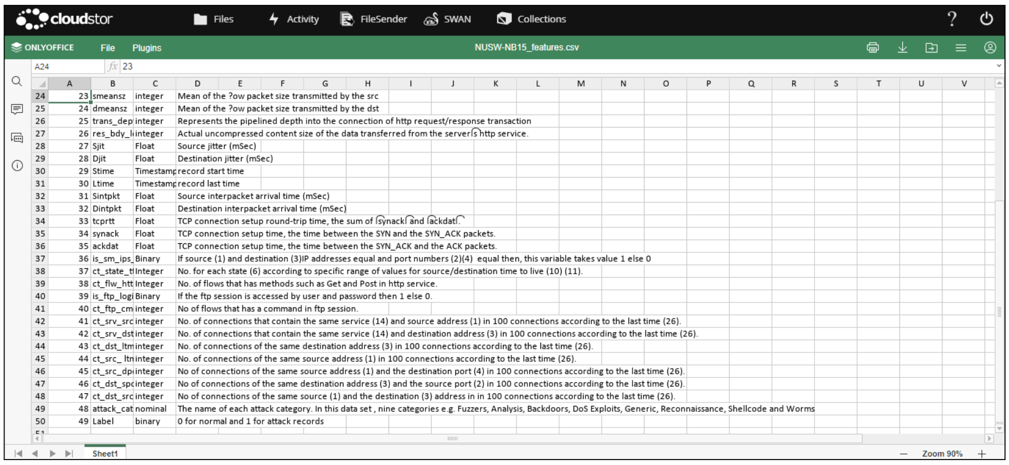
The UNSW-NB15 records are available on [103]. We use the .CSV formatted files for this experiment. The following figures (Fig. A.10 - A.13) show some examples of the dataset records and data features that are extracted and used to run the NPS.



NUSW-NB15_features.csv

No.	Name	Type	Description
1	srcip	nominal	Source IP address
2	sport	integer	Source port number
3	dstip	nominal	Destination IP address
4	dsport	integer	Destination port number
5	proto	nominal	Transaction protocol
6	state	nominal	Indicates to the state and its dependent protocol, e.g. ACC, CLO, CON, ECO, ECR, FIN, INT, MAS, PAR, REQ, RST, TST, TXD, URH, URN, and (-) (if not used state)
7	dur	Float	Record total duration
8	sbytes	Integer	Source to destination transaction bytes
9	dbytes	Integer	Destination to source transaction bytes
10	sttl	Integer	Source to destination time to live value
11	dttl	Integer	Destination to source time to live value
12	sloss	Integer	Source packets retransmitted or dropped
13	dloss	Integer	Destination packets retransmitted or dropped
14	service	nominal	http, ftp, smtp, ssh, dns, ftp-data ,irc and (-) if not much used service
15	Sload	Float	Source bits per second
16	Dload	Float	Destination bits per second
17	Spkts	integer	Source to destination packet count
18	Dpkts	integer	Destination to source packet count
19	swin	integer	Source TCP window advertisement value
20	dwin	integer	Destination TCP window advertisement value
21	stcpb	integer	Source TCP base sequence number
22	dtcpb	integer	Destination TCP base sequence number
23	smeansz	integer	Mean of the ?ow packet size transmitted by the src

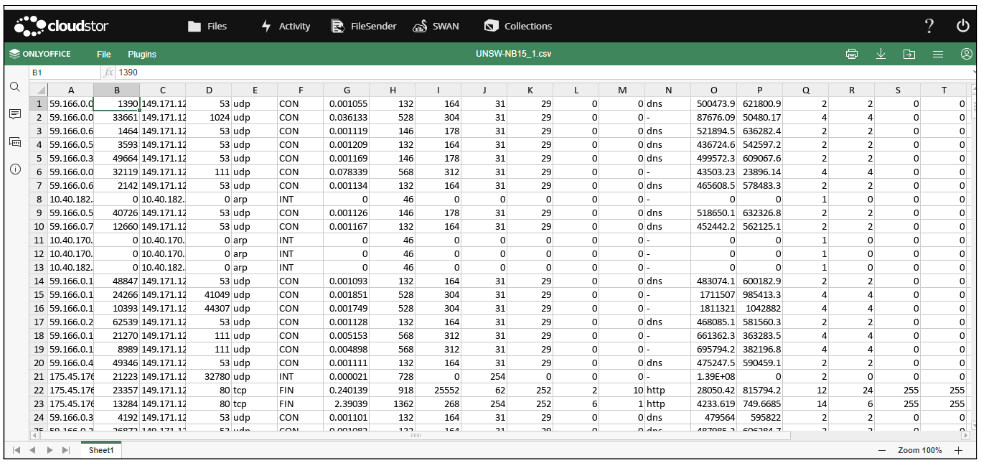
Figure A.10: Data Features Distribution Screenshot-1-



NUSW-NB15_features.csv

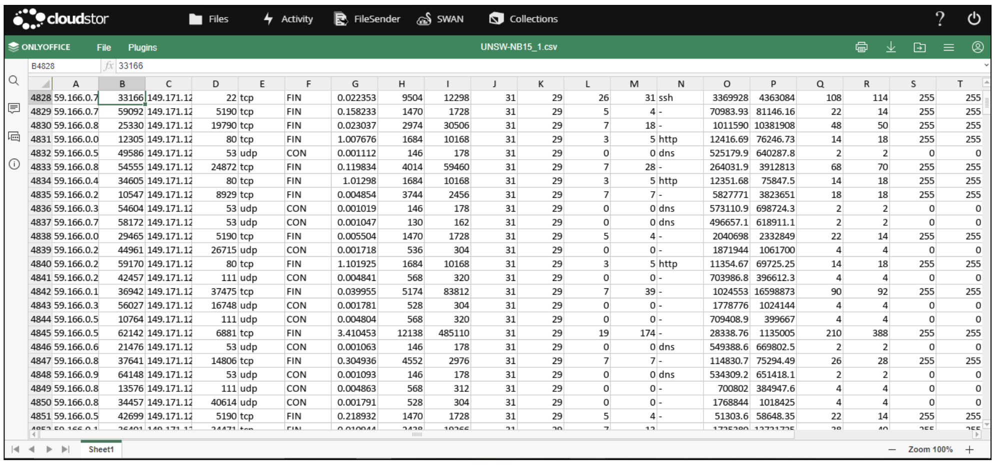
23	smeansz	integer	Mean of the ?ow packet size transmitted by the src
24	dmeansz	integer	Mean of the ?ow packet size transmitted by the dst
25	trans_dep	integer	Represents the pipelined depth into the connection of http request/response transaction
26	res_bdy_l	integer	Actual uncompressed content size of the data transferred from the server's http service.
27	Sjit	Float	Source jitter (mSec)
28	Djit	Float	Destination jitter (mSec)
29	Stime	Timestamp	record start time
30	Ltime	Timestamp	record last time
31	Sintpkt	Float	Source interpacket arrival time (mSec)
32	Dintpkt	Float	Destination interpacket arrival time (mSec)
33	tcprtt	Float	TCP connection setup round-trip time, the sum of 'synack' and 'ackdat'.
34	synack	Float	TCP connection setup time, the time between the SYN and the SYN_ACK packets.
35	ackdat	Float	TCP connection setup time, the time between the SYN_ACK and the ACK packets.
36	is_sm_ips_	Binary	If source (1) and destination (3)IP addresses equal and port numbers (2)(4) equal then, this variable takes value 1 else 0
37	ct_state_t	Integer	No. for each state (6) according to specific range of values for source/destination time to live (10) (11).
38	ct_flw_htt	Integer	No. of flows that has methods such as Get and Post in http service.
39	is_ftp_logi	Binary	If the ftp session is accessed by user and password then 1 else 0.
40	ct_ftp_cm	integer	No of flows that has a command in ftp session.
41	ct_srv_src	integer	No. of connections that contain the same service (14) and source address (1) in 100 connections according to the last time (26).
42	ct_srv_dst	integer	No. of connections that contain the same service (14) and destination address (3) in 100 connections according to the last time (26).
43	ct_dst_ltm	integer	No. of connections of the same destination address (3) in 100 connections according to the last time (26).
44	ct_src_ ltn	integer	No. of connections of the same source address (1) in 100 connections according to the last time (26).
45	ct_src_dp	integer	No of connections of the same source address (1) and the destination port (4) in 100 connections according to the last time (26).
46	ct_dst_spc	integer	No of connections of the same destination address (3) and the source port (2) in 100 connections according to the last time (26).
47	ct_dst_src	integer	No of connections of the same source (1) and the destination (3) address in in 100 connections according to the last time (26).
48	attack_cat	nominal	The name of each attack category. In this data set , nine categories e.g. Fuzzers, Analysis, Backdoors, DoS Exploits, Generic, Reconnaissance, Shellcode and Worms
49	Label	binary	0 for normal and 1 for attack records

Figure A.11: Data Features Distribution Screenshot-2-



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1	59.166.0.0	1390	149.171.12	53	udp	CON	0.001055	132	164	31	29	0	0	dns	500473.9	621800.9	2	2	0	0
2	59.166.0.0	33661	149.171.12	1024	udp	CON	0.036133	528	304	31	29	0	0	-	87676.09	50480.17	4	4	0	0
3	59.166.0.6	1464	149.171.12	53	udp	CON	0.001119	146	178	31	29	0	0	dns	521894.5	636282.4	2	2	0	0
4	59.166.0.5	3593	149.171.12	53	udp	CON	0.001209	132	164	31	29	0	0	dns	436724.6	542597.2	2	2	0	0
5	59.166.0.3	49664	149.171.12	53	udp	CON	0.001169	146	178	31	29	0	0	dns	499572.3	609067.6	2	2	0	0
6	59.166.0.0	32119	149.171.12	111	udp	CON	0.078339	568	312	31	29	0	0	-	43503.23	23896.14	4	4	0	0
7	59.166.0.6	2142	149.171.12	53	udp	CON	0.001134	132	164	31	29	0	0	dns	465608.5	578483.3	2	2	0	0
8	10.40.182.	0	10.40.182.	0	arp	INT	0	46	0	0	0	0	0	-	0	0	1	0	0	0
9	59.166.0.5	40726	149.171.12	53	udp	CON	0.001126	146	178	31	29	0	0	dns	518650.1	632326.8	2	2	0	0
10	59.166.0.7	12660	149.171.12	53	udp	CON	0.001167	132	164	31	29	0	0	dns	452442.2	562125.1	2	2	0	0
11	10.40.170.	0	10.40.170.	0	arp	INT	0	46	0	0	0	0	0	-	0	0	1	0	0	0
12	10.40.170.	0	10.40.170.	0	arp	INT	0	46	0	0	0	0	0	-	0	0	1	0	0	0
13	10.40.182.	0	10.40.182.	0	arp	INT	0	46	0	0	0	0	0	-	0	0	1	0	0	0
14	59.166.0.1	48847	149.171.12	53	udp	CON	0.001093	132	164	31	29	0	0	dns	483074.1	600182.9	2	2	0	0
15	59.166.0.1	24266	149.171.12	41049	udp	CON	0.001851	528	304	31	29	0	0	-	1711507	985413.3	4	4	0	0
16	59.166.0.1	10393	149.171.12	44307	udp	CON	0.001749	528	304	31	29	0	0	-	1811321	1042882	4	4	0	0
17	59.166.0.2	62539	149.171.12	53	udp	CON	0.001128	132	164	31	29	0	0	dns	468085.1	581560.3	2	2	0	0
18	59.166.0.1	21270	149.171.12	111	udp	CON	0.005153	568	312	31	29	0	0	-	661362.3	363283.5	4	4	0	0
19	59.166.0.1	8989	149.171.12	111	udp	CON	0.004898	568	312	31	29	0	0	-	695794.2	382196.8	4	4	0	0
20	59.166.0.4	49346	149.171.12	53	udp	CON	0.001111	132	164	31	29	0	0	dns	475247.5	590459.1	2	2	0	0
21	175.45.176	21223	149.171.12	32780	udp	INT	0.000021	728	0	254	0	0	0	-	1.39E+08	0	2	0	0	0
22	175.45.176	23357	149.171.12	80	tcp	FIN	0.240139	918	25552	62	252	2	10	http	28050.42	815794.2	12	24	255	255
23	175.45.176	13284	149.171.12	80	tcp	FIN	2.39039	1362	268	254	252	6	1	http	4233.619	749.6685	14	6	255	255
24	59.166.0.3	4192	149.171.12	53	udp	CON	0.001101	132	164	31	29	0	0	dns	479564	595822	2	2	0	0

Figure A.12: Some Extracted Records Screenshot-1-



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
4828	59.166.0.7	33166	149.171.12	22	tcp	FIN	0.022353	9504	12298	31	29	26	31	ssh	3369928	4363084	108	114	255	255
4829	59.166.0.7	59092	149.171.12	5190	tcp	FIN	0.158233	1470	1728	31	29	5	4	-	70983.93	81146.16	22	14	255	255
4830	59.166.0.8	25330	149.171.12	19790	tcp	FIN	0.023037	2974	30506	31	29	7	18	-	1011590	10381908	48	50	255	255
4831	59.166.0.0	12305	149.171.12	80	tcp	FIN	1.007676	1684	10168	31	29	3	5	http	12416.69	76246.73	14	18	255	255
4832	59.166.0.5	49586	149.171.12	53	udp	CON	0.001112	146	178	31	29	0	0	dns	525179.9	640287.8	2	2	0	0
4833	59.166.0.8	54555	149.171.12	24872	tcp	FIN	0.119834	4014	59460	31	29	7	28	-	264031.9	3912813	68	70	255	255
4834	59.166.0.4	34605	149.171.12	80	tcp	FIN	1.01298	1684	10168	31	29	3	5	http	12351.68	75847.5	14	18	255	255
4835	59.166.0.2	10547	149.171.12	8929	tcp	FIN	0.004854	3744	2456	31	29	7	7	-	5827771	3823651	18	18	255	255
4836	59.166.0.3	54604	149.171.12	53	udp	CON	0.001019	146	178	31	29	0	0	dns	573110.9	698724.3	2	2	0	0
4837	59.166.0.7	58172	149.171.12	53	udp	CON	0.001047	130	162	31	29	0	0	dns	496657.1	618911.1	2	2	0	0
4838	59.166.0.0	29465	149.171.12	5190	tcp	FIN	0.005504	1470	1728	31	29	5	4	-	2040698	2332849	22	14	255	255
4839	59.166.0.2	44961	149.171.12	26715	udp	CON	0.001718	536	304	31	29	0	0	-	1871944	1061700	4	4	0	0
4840	59.166.0.2	59170	149.171.12	80	tcp	FIN	1.101925	1684	10168	31	29	3	5	http	11354.67	69725.25	14	18	255	255
4841	59.166.0.2	42457	149.171.12	111	udp	CON	0.004841	568	320	31	29	0	0	-	703986.8	396612.3	4	4	0	0
4842	59.166.0.1	36942	149.171.12	37475	tcp	FIN	0.039955	5174	83812	31	29	7	39	-	1024553	16598873	90	92	255	255
4843	59.166.0.3	56027	149.171.12	16748	udp	CON	0.001781	528	304	31	29	0	0	-	1778776	1024144	4	4	0	0
4844	59.166.0.5	10764	149.171.12	111	udp	CON	0.004804	568	320	31	29	0	0	-	709408.9	399667	4	4	0	0
4845	59.166.0.5	62142	149.171.12	6881	tcp	FIN	3.410453	12138	485110	31	29	19	174	-	28338.76	1135005	210	388	255	255
4846	59.166.0.6	21476	149.171.12	53	udp	CON	0.001063	146	178	31	29	0	0	dns	549388.6	669802.5	2	2	0	0
4847	59.166.0.8	37641	149.171.12	14806	tcp	FIN	0.304936	4552	2976	31	29	7	7	-	114830.7	75294.49	26	28	255	255
4848	59.166.0.9	64148	149.171.12	53	udp	CON	0.001093	146	178	31	29	0	0	dns	534309.2	651418.1	2	2	0	0
4849	59.166.0.8	13576	149.171.12	111	udp	CON	0.004863	568	312	31	29	0	0	-	700802	384947.6	4	4	0	0
4850	59.166.0.8	34457	149.171.12	40614	udp	CON	0.001791	528	304	31	29	0	0	-	1768844	1018425	4	4	0	0
4851	59.166.0.5	42699	149.171.12	5190	tcp	FIN	0.218932	1470	1728	31	29	5	4	-	51303.6	58648.35	22	14	255	255

Figure A.13: Some Extracted Records Screenshot-2-

B APPENDIX 2

B.1 THE NPS STEP BY STEP

In this section, we present the NPS algorithm step by step. Starting by reading the training data file, and then generating two sets of detectors, negative and positive. Next is reading the testing data file, followed by the detection stage.

Algorithm 3: The main class of the NPS

```
1: Read the destination port from the training data file
   (Algorithm4)
2: Create the required number of negative and positive detectors
   (Algorithm5)
3: Read the destination port and record label from the testing data file
   (Algorithm6)
4: Run the detection stage
   (Algorithm7)
5: Calculate detection results of the NPS (Algorithm8)
```

Algorithm 4: Read the destination port from the training data file

```
1: Input Training data file
2: Output
   $DestinationPort$ =
   Destination port for each record in the training file in binary
3: while $i \leq$ training data file length do
4:    Read destination port
5:    convert destination port to binary
6: end while
```

Algorithm 5: Create the required number of negative and positive detectors

```
1:  Input
    $DestinationPort$ =
    Destination port for each record in the training file in binary
    $NumberofMatches$ =the required number of matches
    $DetectorSize$ = the detector size
    $DetectorSetSize$ = the detector set size
2:  Output
    $NegativeDetectors$ = Negative detectors sets
    $PositiveDetectors$ = Positive detectors sets
3:  while $i \leq DetectorSetSiz$ do
4:     $D$ = a generated random string using $detector_size$
5:     Match $D$ with $DP$
6:     if $D \in DP$ then
7:        add $D$ to $PositiveDetectors$
8:     else
9:        add $D$ to $NegativeDetectors$
10:    end if
11: end while
```

Algorithm 6: Read the destination port from the testing data file

```
1: Input Testing data file
2: Output
   $DestinationPort$ =
   Destination port for each record in the testing file in binary
   $RecordLabel$ = the label of each record (malware or benign)
3: while $i \leq$ testing data file length do
4:    Read destination port
5:    Convert destination port to binary
6:    Read the label of the record
7: end while
```

Algorithm 7: Detection Stage

```
1: Input $DestinationPort$ =
   Destination port for each record in the testing file in binary
   $RecordLabel$ = the label of each record (malware or benign)
   $PreviouslyDetectedFiles$ =
   a database of previously detected malware files
   $NegativeDetectors$ = Negative detectors sets
   $PositiveDetectors$ = Positive detectors sets
2: Output
   $TheNPSDetection$ = marking each file as malware or benign
3: while $i \leq DP$ do
4:    if $DP \in PreviouslyDetectedFiles$ then
5:       the file is labeled as malware
6:       $EXIT$
7:    else if $DP \in NegativeDetector$ then
8:       the file is labeled as malware
9:       $EXIT$
10:   else if $DP \notin PositiveDetector$ then
11:      the file is labeled as malware
12:      $EXIT$
13:   end if
14: end while
```

Algorithm 8: Calculate detection results of the NPS

```
 1: Input $TheNPSDetection$ =
    Detection results of each file (malware or benign)
    $RecordLabel$ = the label of each record (malware or benign)
 2: Output
    $TP$ = malware is detected as a malicious application $TN$ =
    benign is detected as non-malicious application $FP$ =
    benign is detected as a malicious application $FN$ =
    malware is detected as non-malicious application
 3: while $i \leq RecordLabel$ do
 4:   if $TheNPSDetection = RecordLabel$ then
 5:
 6:     if $RecordLabel = Malware$ then
 7:       $TP$
 8:     else
 9:       $TN$
10:     end if
11:   end if
12:   if $TheNPSDetection \neq RecordLabel$ then
13:
14:     if $RecordLabel = Benign$ then
15:       $FP$
16:     else
17:       $FN$
18:     end if
19:   end if
20: end while
```

BIBLIOGRAPHY

[1] H. Alrubayyi, G. Goteng, M. Jaber and J. Kelly, 'A novel negative and positive selection algorithm to detect unknown malware in the IoT', in *IEEE INFOCOM 2021-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, IEEE, 2021, pp. 1–6.

[2] H. Alrubayyi, G. Goteng, M. Jaber and J. Kelly, 'Challenges of malware detection in the IoT and a review of artificial immune system approaches', *Journal of Sensor and Actuator Networks*, vol. 10, no. 4, 2021, ISSN: 2224-2708. [Online]. Available: https://www.mdpi.com/2224-2708/10/4/61.

[3] S. Kumar, P. Tiwari and M. Zymbler, 'Internet of things is a revolutionary approach for future technology enhancement: A review', *Journal of Big data*, vol. 6, no. 1, pp. 1–21, 2019.

[4] V. A. Thakor, M. A. Razzaque and M. R. Khandaker, 'Lightweight cryptography for iot: A state-of-the-art', *arXiv preprint arXiv:2006.13813*, 2020.

[5] H. Habibzadeh, K. Dinesh, O. Rajabi Shishvan, A. Boggio-Dandry, G. Sharma and T. Soyata, 'A survey of healthcare internet of things (hiot): A clinical perspective', *IEEE Internet of Things Journal*, vol. 7, no. 1, pp. 53–71, 2020. DOI: 10.1109/JIOT.2019.2946359.

[6] V. Chamola, V. Hassija, V. Gupta and M. Guizani, 'A comprehensive review of the COVID-19 pandemic and the role of IoT, drones, AI, blockchain, and 5G in managing its impact', *IEEE Access*, vol. 8, pp. 90 225–90 265, 2020. DOI: 10.1109/ACCESS.2020.2992341.

[7] N. Hossein Motlagh, M. Mohammadrezaei, J. Hunt and B. Zakeri, 'Internet of things (IoT) and the energy sector', *Energies*, vol. 13, no. 2, 2020, ISSN: 1996-1073. DOI: 10.3390/en13020494. [Online]. Available: https://www.mdpi.com/1996-1073/13/2/494.

[8] V. A. Thakor, M. A. Razzaque and M. R. A. Khandaker, 'Lightweight cryptography algorithms for resource-constrained iot devices: A review, comparison and research opportunities', *IEEE Access*, vol. 9, pp. 28 177–28 193, 2021. DOI: 10.1109/ACCESS.2021.3052867.

[9] G. Hatzivasilis, O. Soultatos, S. Ioannidis, C. Verikoukis, G. Demetriou and C. Tsatsoulis, 'Review of security and privacy for the internet of medical things (IoMT)', in *2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, 2019, pp. 457–464. DOI: 10.1109/DCOSS.2019.00091.

[10] C. Donalds and K.-M. Osei-Bryson, 'Toward a cybercrime classification ontology: A knowledge-based approach', *Computers in Human Behavior*, vol. 92, pp. 403 –418, 2019, ISSN: 0747-5632. DOI: https://doi.org/10.1016/j.chb.2018.11.039.

[11] *The biggest data breaches in the first half of 2020*, https://www.keepnetlabs.com/the-biggest-data-breaches-in-the-first-half-of-2020/, Oct. 2020.

[12] S. Hilt, V. Kropotov, F. Mercês, M. Rosario and D. Sancho, 'The internet of things in the cybercrime underground', *Trend Micro Research*, 2019.

[13] *The role of helper t cells*. [Online]. Available: https://www.savemyexams.co.uk/a-level/biology/aqa/17/revision-notes/2-cell-structure/2-5-cell-recognition--the-immune-system/2-5-6-the-role-of-helper-t-cells/.

[14] A. E. Omolara, A. Alabdulatif, O. I. Abiodun, M. Alawida, A. Alabdulatif, W. H. Alshoura and H. Arshad, 'The internet of things security: A survey encompassing unexplored areas and new insights', *Computers & Security*, vol. 112, p. 102 494, 2022, ISSN: 0167-4048. DOI: https://doi.org/10.1016/j.cose.2021.102494. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404821003187.

[15] S. Saeed, N. Jhanjhi, M. Naqvi, M. Humayun and S. Ahmed, 'Ransomware: A framework for security challenges in internet of things', in *2020 2nd International Conference on Computer and Information Sciences (ICCIS)*, 2020, pp. 1–6. DOI: 10.1109/ICCIS49240.2020.9257660.

[16] M. Wazid, A. K. Das, J. J. P. C. Rodrigues, S. Shetty and Y. Park, 'Iomt malware detection approaches: Analysis and research challenges', *IEEE Access*, vol. 7, pp. 182 459–182 476, 2019. DOI: 10.1109/ACCESS.2019.2960412.

[17] A. A. A. Samra, H. N. Qunoo, F. Al-Rubaie and H. El-Talli, 'A survey of static android malware detection techniques', in *2019 IEEE 7th Palestinian International Conference on Electrical and Computer Engineering (PICECE)*, 2019, pp. 1–6. DOI: 10.1109/PICECE.2019.8747224.

[18] S. Alhasan, G. Abdul-Salaam, L. Bayor and K. Oliver, 'Intrusion detection system based on artificial immune system: A review', in *2021 International Conference on Cyber Security and Internet of Things (ICSIoT)*, 2021, pp. 7–14. DOI: 10.1109/ICSIoT55070.2021.00011.

[19] M. Othman and A. El-Mousa, 'Internet of things cloud computing internet of things as a service approach', in *2020 11th International Conference on Information and Communication Systems (ICICS)*, 2020, pp. 318–323. DOI: 10.1109/ICICS49469.2020.239503.

[20] M. Khan and F. Algarni, 'A healthcare monitoring system for the diagnosis of heart disease in the IoMT cloud environment using MSSO-ANFIS', *IEEE Access*, vol. 8, pp. 122 259–122 269, 2020. DOI: 10.1109/ACCESS.2020.3006424.

[21] J. Zhang, G. Li, A. Marshall, A. Hu and L. Hanzo, 'A new frontier for IoT security emerging from three decades of key generation relying on wireless channels', *IEEE Access*, vol. 8, pp. 138 406–138 446, 2020. DOI: 10.1109/ACCESS.2020.3012006.

[22] H. Lin, S. Garg, J. Hu, X. Wang, M. J. Piran and M. S. Hossain, 'Privacy-enhanced data fusion for COVID-19 applications in intelligent internet of medical things', *IEEE Internet of Things Journal*, pp. 1–1, 2020. DOI: 10.1109/JIOT.2020.3033129.

[23] F. Aloul, I. Zualkernan, S. Shapsough and M. Towheed, 'A monitoring and control gateway for IoT edge devices in smart home', in *2020 International Conference on Information Networking (ICOIN)*, 2020, pp. 696–701. DOI: 10.1109/ICOIN48656.2020.9016465.

[24] J. Greensmith, 'Securing the internet of things with responsive artificial immune systems', in *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO '15, Madrid, Spain: Association for Computing Machinery, 2015, 113–120, ISBN: 9781450334723. DOI: 10.1145/2739480.2754816. [Online]. Available: https://doi.org/10.1145/2739480.2754816.

[25] J. Deogirikar and A. Vidhate, 'Security attacks in IoT: A survey', in *2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, 2017, pp. 32–37. DOI: 10.1109/I-SMAC.2017.8058363.

[26] S. Aldhaheri, D. Alghazzawi, L. Cheng, A. Barnawi and B. Alzahrani, 'Artificial immune systems approaches to secure the internet of things: A systematic review of the literature and recommendations for future research', *Journal of Network and Computer Applications*, vol. 157, p. 102 537, 2020, ISSN: 1084-8045. DOI: https://doi.org/10.1016/j.jnca.2020.102537.

[27] J. Jeon, J. Park and Y. Jeong, 'Dynamic analysis for IoT malware detection with convolution neural network model', *IEEE Access*, vol. 8, pp. 96 899–96 911, 2020. DOI: 10.1109/ACCESS.2020.2995887.

[28] Ö. Aslan and R. Samet, 'A comprehensive review on malware detection approaches', *IEEE Access*, vol. 8, pp. 6249–6271, 2020.

[29] M. Humayun, N. Jhanjhi, A. Alsayat and V. Ponnusamy, 'Internet of things and ransomware: Evolution, mitigation and prevention', *Egyptian Informatics Journal*, vol. 22, no. 1, pp. 105–117, 2021, ISSN: 1110-8665. DOI: https://doi.org/10.1016/j.eij.2020.05.003. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1110866520301304.

[30] A. Moser, C. Kruegel and E. Kirda, 'Limits of static analysis for malware detection', in *Twenty-Third Annual Computer Security Applications Conference (ACSAC 2007)*, 2007, pp. 421–430. DOI: 10.1109/ACSAC.2007.21.

[31] A. Afianian, S. Niksefat, B. Sadeghiyan and D. Baptiste, 'Malware dynamic analysis evasion techniques: A survey', *ACM Computing Surveys (CSUR)*, vol. 52, no. 6, pp. 1–28, 2019.

[32] K. A. Roundy and B. P. Miller, 'Hybrid analysis and control of malware', in *International Workshop on Recent Advances in Intrusion Detection*, Springer, 2010, pp. 317–338.

[33] T. Alsmadi and N. Alqudah, 'A survey on malware detection techniques', in *2021 International Conference on Information Technology (ICIT)*, 2021, pp. 371–376. DOI: 10.1109/ICIT52682.2021.9491765.

[34] S. Mukkamala and A. Sung, 'Detecting denial of service attacks using support vector machines', in *The 12th IEEE International Conference on Fuzzy Systems, 2003. FUZZ '03.*, vol. 2, 2003, 1231–1236 vol.2. DOI: 10.1109/FUZZ.2003.1206607.

[35] C. Kruegel and T. Toth, 'Using decision trees to improve signature-based intrusion detection', in *International workshop on recent advances in intrusion detection*, Springer, 2003, pp. 173–191.

[36] F. I. Shiri, B. Shanmugam and N. B. Idris, ‘A parallel technique for improving the performance of signature-based network intrusion detection system’, in *2011 IEEE 3rd International Conference on Communication Software and Networks*, IEEE, 2011, pp. 692–696.

[37] B. Arrington, L. Barnett, R. Rufus and A. Esterline, ‘Behavioral modeling intrusion detection system (bmids) using internet of things (IoT) behavior-based anomaly detection via immunity-inspired algorithms’, in *2016 25th International Conference on Computer Communication and Networks (ICCCN)*, IEEE, 2016, pp. 1–6.

[38] C. Zhu, W. Sheng and M. Liu, ‘Wearable sensor-based behavioral anomaly detection in smart assisted living systems’, *IEEE Transactions on automation science and engineering*, vol. 12, no. 4, pp. 1225–1234, 2015.

[39] A. B. Sallow, M Sadeeq, R. R. Zebari, M. B. Abdulrazzaq, M. R. Mahmood, H. M. Shukur and L. M. Haji, ‘An investigation for mobile malware behavioral and detection techniques based on android platform’, *IOSR Journal of Computer Engineering (IOSR-JCE)*, vol. 22, no. 4, pp. 14–20, 2020.

[40] D. Bilar, ‘Opcodes as predictor for malware’, *Int. J. Electron. Secur. Digit. Forensic*, vol. 1, no. 2, 156–168, Jan. 2007, ISSN: 1751-911X. DOI: 10.1504/IJESDF.2007.016865. [Online]. Available: https://doi.org/10.1504/IJESDF.2007.016865.

[41] M. Schultz, E. Eskin, F. Zadok and S. Stolfo, ‘Data mining methods for detection of new malicious executables’, in *Proceedings 2001 IEEE Symposium on Security and Privacy. S P 2001*, 2001, pp. 38–49. DOI: 10.1109/SECPRI.2001.924286.

[42] P. Jalote, *An integrated approach to software engineering*. Springer Science & Business Media, 2012.

[43] M. Eskandari and S. Hashemi, ‘Metamorphic malware detection using control flow graph mining’, *Int. J. Comput. Sci. Network Secur*, vol. 11, no. 12, pp. 1–6, 2011.

[44] S. Raza, L. Wallgren and T. Voigt, ‘Svelte: Real-time intrusion detection in the internet of things’, *Ad hoc networks*, vol. 11, no. 8, pp. 2661–2674, 2013.

[45] A. Abusnaina, A. Anwar, S. Alshamrani, A. Alabduljabbar, R. Jang, D. Nyang and D. Mohaisen, ‘Systemically evaluating the robustness of ml-based IoT malware detectors’, in *2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks-Supplemental Volume (DSN-S)*, IEEE, 2021, pp. 3–4.

[46] V. Rey, P. M. SÃ¡nchez SÃ¡nchez, A. Huertas CeldrÃ¡n and G. Bovet, 'Federated learning for malware detection in IoT devices', *Computer Networks*, vol. 204, p. 108 693, 2022, ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2021.108693. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128621005582.

[47] J. Abawajy, A. Darem and A. A. Alhashmi, 'Feature subset selection for malware detection in smart IoT platforms', *Sensors*, vol. 21, no. 4, 2021, ISSN: 1424-8220. DOI: 10.3390/s21041374. [Online]. Available: https://www.mdpi.com/1424-8220/21/4/1374.

[48] G. L. Nguyen, B. Dumba, Q.-D. Ngo, H.-V. Le and T. N. Nguyen, 'A collaborative approach to early detection of IoT botnet', *Computers & Electrical Engineering*, vol. 97, p. 107 525, 2022, ISSN: 0045-7906. DOI: https://doi.org/10.1016/j.compeleceng.2021.107525. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0045790621004717.

[49] H. Wu, H. Han, X. Wang and S. Sun, 'Research on artificial intelligence enhancing internet of things security: A survey', *Ieee Access*, vol. 8, pp. 153 826–153 848, 2020.

[50] B. Vignau, R. Khoury, S. HallÃ© and A. Hamou-Lhadj, 'The evolution of IoT malwares, from 2008 to 2019: Survey, taxonomy, process simulator and perspectives', *Journal of Systems Architecture*, vol. 116, p. 102 143, 2021, ISSN: 1383-7621. DOI: https://doi.org/10.1016/j.sysarc.2021.102143. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1383762121001053.

[51] R. Kawasoe, C. Han, R. Isawa, T. Takahashi and J. Takeuchi, 'Investigating behavioral differences between IoT malware via function call sequence graphs', ser. SAC '21, New York, NY, USA: Association for Computing Machinery, 2021, 1674–1682, ISBN: 9781450381048. DOI: 10.1145/3412841.3442041. [Online]. Available: https://doi.org/10.1145/3412841.3442041.

[52] *Sonicwall 2019 report: 55 rise in IoT malware attacks*, https://www.openaccessgovernment.org/iot-malware-attacks/69870/, Jul. 2019.

[53] P. Muncaster, *Over 100 million IoT attacks detected in 1h 2019*, https://www.infosecurity-magazine.com/news/over-100-million-iot-attacks/, Oct. 2019.

[54] K. Naveed and H. Wu, 'Poster: A semi-supervised framework to detect botnets in IoT devices', in *2020 IFIP Networking Conference (Networking)*, 2020, pp. 649–651.

[55] Z. Fuyong and Q. Deyu, 'Run-time malware detection based on positive selection', *Journal in computer virology*, vol. 7, no. 4, p. 267, 2011.

[56] Z. Ji and D. Dasgupta, 'Revisiting negative selection algorithms', *Evolutionary Computation*, vol. 15, no. 2, pp. 223–251, 2007.

[57] M. Riaz, F. Yousaf, M. Akram, M. I. Ullah, G. Rasool, C. Egbuna, K. C. Patrick-Iwuanyanwu, C. Z. Uche and J. C. Ifemeje, 'Chapter 16 - immunology and immunochemistry', in *Analytical Techniques in Biosciences*, C. Egbuna, K. C. Patrick-Iwuanyanwu, M. A. Shah, J. C. Ifemeje and A. Rasul, Eds., Academic Press, 2022, pp. 251–268, ISBN: 978-0-12-822654-4. DOI: https://doi.org/10.1016/B978-0-12-822654-4.00014-2. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128226544000142.

[58] S. Ghosh, *Human Immune System*. Walter de Gruyter GmbH & Co KG, 2022.

[59] J. S. Marshall, R. Warrington, W. Watson and H. L. Kim, 'An introduction to immunology and immunopathology', *Allergy, Asthma & Clinical Immunology*, vol. 14, no. 2, pp. 1–10, 2018.

[60] P. Moss, 'The t cell immune response against sars-cov-2', *Nature immunology*, pp. 1–8, 2022.

[61] J. S. Low, D. Vaqueirinho, F. Mele, M. Foglierini, J. Jerak, M. Perotti, D. Jarrossay, S. Jovic, L. Perez, R. Cacciatore *et al.*, 'Clonal analysis of immunodominance and cross-reactivity of the cd4 t cell response to sars-cov-2', *Science*, vol. 372, no. 6548, pp. 1336–1341, 2021.

[62] J. Varadé, S. Magadán and Á. González-Fernández, 'Human immunology and immunotherapy: Main achievements and challenges', *Cellular & Molecular Immunology*, vol. 18, no. 4, pp. 805–828, 2021.

[63] M. Abdullahi, Y. Baashar, H. Alhussian, A. Alwadain, N. Aziz, L. F. Capretz and S. J. Abdulkadir, 'Detecting cybersecurity attacks in internet of things using artificial intelligence methods: A systematic literature review', *Electronics*, vol. 11, no. 2, 2022, ISSN: 2079-9292. DOI: 10.3390/electronics11020198. [Online]. Available: https://www.mdpi.com/2079-9292/11/2/198.

[64] S. Forrest, A. S. Perelson, L. Allen and R. Cherukuri, 'Self-nonself discrimination in a computer', in *Proceedings of 1994 IEEE Computer Society Symposium on Research in Security and Privacy*, 1994, pp. 202–212. DOI: 10.1109/RISP.1994.296580.

[65] K. D. Gupta and D. Dasgupta, ‘Negative selection algorithm research and applications in the last decade: A review’, *IEEE Transactions on Artificial Intelligence*, vol. 3, no. 2, pp. 110–128, 2022. DOI: 10.1109/TAI.2021.3114661.

[66] M. Burnet, *The clonal selection theory of acquired immunity*. Vanderbilt University Press Nashville, 1959, vol. 3.

[67] G. Scaranti, L. Carvalho, S. Barbon and M. Proença, ‘Artificial immune systems and fuzzy logic to detect flooding attacks in software-defined networks’, *IEEE Access*, vol. 8, pp. 100 172–100 184, 2020. DOI: 10.1109/ACCESS.2020.2997939.

[68] T. Yang, W. Chen and T. Li, ‘A real negative selection algorithm with evolutionary preference for anomaly detection’, *Open Physics*, vol. 15, no. 1, pp. 121–134, 2017.

[69] R. Roman, R. Rios, J. Onieva and J. Lopez, ‘Immune system for the internet of things using edge technologies’, *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4774–4781, 2019. DOI: 10.1109/JIOT.2018.2867613.

[70] L. De Castro and F. Von Zuben, ‘Learning and optimization using the clonal selection principle’, *IEEE transactions on evolutionary computation*, vol. 6, no. 3, pp. 239–251, 2002.

[71] N. Jerne, ‘Towards a network theory of the immune system’, *Ann. Immunol.*, vol. 125, pp. 373–389, 1974.

[72] S. Alhasan, G. Abdul-Salaam, L. Bayor and K. Oliver, ‘Intrusion detection system based on artificial immune system: A review’, in *2021 International Conference on Cyber Security and Internet of Things (ICSIoT)*, 2021, pp. 7–14. DOI: 10.1109/ICSIoT55070.2021.00011.

[73] J. Greensmith, ‘Securing the internet of things with responsive artificial immune systems’, in *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO ’15, Madrid, Spain: Association for Computing Machinery, 2015, 113–120, ISBN: 9781450334723. [Online]. Available: https://doi.org/10.1145/2739480.2754816.

[74] K. Saleem, J. Chaudhry, M. Orgun and J. Al-Muhtadi, ‘A bio-inspired secure IPv6 communication protocol for internet of things’, in *2017 Eleventh International Conference on Sensing Technology (ICST)*, 2017, pp. 1–6. DOI: 10.1109/ICSensT.2017.8304428.

[75] Z. Ji and D. Dasgupta, 'Real-valued negative selection algorithm with variable-sized detectors', in *Genetic and Evolutionary Computation Conference*, Springer, 2004, pp. 287–298.

[76] D. Dasgupta and F. Gonzalez, 'An immunity-based technique to characterize intrusions in computer networks', *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 3, pp. 281–291, 2002. DOI: 10.1109/TEVC.2002.1011541.

[77] J. Shapiro, G. Lamont and G. Peterson, 'An evolutionary algorithm to generate hyper-ellipsoid detectors for negative selection', in *Proceedings of the 7th annual conference on Genetic and evolutionary computation*, 2005, pp. 337–344.

[78] S. Balachandran, D. Dasgupta, F. Nino and D. Garrett, 'A framework for evolving multi-shaped detectors in negative selection', in *2007 IEEE Symposium on Foundations of Computational Intelligence*, 2007, pp. 401–408. DOI: 10.1109/FOCI.2007.371503.

[79] M. Pamukov and V. Poulkov, 'Multiple negative selection algorithm: Improving detection error rates in IoT intrusion detection systems', in *2017 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, vol. 1, 2017, pp. 543–547. DOI: 10.1109/IDAACS.2017.8095140.

[80] K. Igawa and H. Ohashi, 'A negative selection algorithm for classification and reduction of the noise effect', *Applied Soft Computing*, vol. 9, no. 1, pp. 431–438, 2009.

[81] F. Zhang, D. Qi and J. Hu, 'MBMAS: A system for malware behavior monitor and analysis', in *2009 International Symposium on Computer Network and Multimedia Technology*, 2009, pp. 1–4. DOI: 10.1109/CNMT.2009.5374613.

[82] S. Hanif, T. Ilyas and M. Zeeshan, 'Intrusion detection in IoT using artificial neural networks on unsw-15 dataset', in *2019 IEEE 16th International Conference on Smart Cities: Improving Quality of Life Using ICT IoT and AI (HONET-ICT)*, 2019, pp. 152–156. DOI: 10.1109/HONET.2019.8908122.

[83] M. Pamukov, V. Poulkov and V. Shterev, 'Negative selection and neural network based algorithm for intrusion detection in IoT', in *2018 41st International Conference on Telecommunications and Signal Processing (TSP)*, 2018, pp. 1–5. DOI: 10.1109/TSP.2018.8441338.

[84] *NSL-KDD dataset*, https://www.unb.ca/cic/datasets/nsl.html.

[85] M. E. Pamukov, 'Application of artificial immune systems for the creation of iot intrusion detection systems', in *2017 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, vol. 1, 2017, pp. 564–568. DOI: 10.1109/IDAACS.2017.8095144.

[86] A. Khannous, A. Rghioui and E. Fatiha, 'Manet security: An intrusion detection system based on the combination of negative selection and danger theory concepts', May 2014, pp. 88–91. DOI: 10.1109/NGNS.2014.6990233.

[87] U. Aickelin, P. Bentley, S. Cayzer, J. Kim and J. McLeod, 'Danger theory: The link between ais and ids?', in *Artificial Immune Systems*, J. Timmis, P. J. Bentley and E. Hart, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 147–155, ISBN: 978-3-540-45192-1.

[88] E. D. Alalade, 'Intrusion detection system in smart home network using artificial immune system and extreme learning machine hybrid approach', in *2020 IEEE 6th World Forum on Internet of Things (WF-IoT)*, 2020, pp. 1–2. DOI: 10.1109/WF-IoT48130.2020.9221151.

[89] K. Naveed and H. Wu, 'Celosia: An immune-inspired anomaly detection framework for IoT devices', in *2020 IEEE 45th Conference on Local Computer Networks (LCN)*, 2020, pp. 13–23. DOI: 10.1109/LCN48667.2020.9314839.

[90] S. Aldhaheri, D. Alghazzawi, L. Cheng, B. Alzahrani and A. Al-Barakati, 'DeepDCA: Novel network-based detection of IoT attacks using artificial immune system', *Applied Sciences*, vol. 10, no. 6, 2020, ISSN: 2076-3417. [Online]. Available: https://www.mdpi.com/2076-3417/10/6/1909.

[91] P. Parrend, P. David, F. Guigou, C. Pupka and P. Collet, 'The AWA artificial emergent awareness architecture model for artificial immune ecosystems', in *2017 IEEE Congress on Evolutionary Computation (CEC)*, 2017, pp. 403–410. DOI: 10.1109/CEC.2017.7969340.

[92] C. Liu, Y. Zhang and H. Zhang, 'A novel approach to IoT security based on immunology', in *2013 Ninth International Conference on Computational Intelligence and Security*, 2013, pp. 771–775. DOI: 10.1109/CIS.2013.168.

[93] R. Chen, C. M. Liu and C. Chen, 'An artificial immune-based distributed intrusion detection model for the internet of things', in *Advanced materials research*, Trans Tech Publ, vol. 366, 2012, pp. 165–168.

[94] C. Liu, J. Yang, R. Chen, Y. Zhang and J. Zeng, 'Research on immunity-based intrusion detection technology for the internet of things', in *2011 Seventh International Conference on Natural Computation*, vol. 1, 2011, pp. 212–216. DOI: 10.1109/ICNC.2011.6022060.

[95] S. Hofmeyr, S. Forrest and A. Somayaji, 'Intrusion detection using sequences of system calls', *J. Comput. Secur.*, vol. 6, no. 3, 151–180, Aug. 1998, ISSN: 0926-227X.

[96] K. Albulayhi, A. A. Smadi, F. T. Sheldon and R. K. Abercrombie, 'IoT intrusion detection taxonomy, reference architecture, and analyses', *Sensors*, vol. 21, no. 19, 2021, ISSN: 1424-8220. [Online]. Available: https://www.mdpi.com/1424-8220/21/19/6432.

[97] *The bot-IoT dataset*, https://research.unsw.edu.au/projects/bot-iot-dataset.

[98] *The bot-IoT dataset*. [Online]. Available: https://research.unsw.edu.au/projects/bot-iot-dataset.

[99] N. Koroniotis, N. Moustafa, E. Sitnikova and B. Turnbull, 'Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-IoT dataset', *Future Generation Computer Systems*, vol. 100, pp. 779–796, 2019, ISSN: 0167-739X. DOI: https://doi.org/10.1016/j.future.2019.05.041. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X18327687.

[100] N. Koroniotis, N. Moustafa, E. Sitnikova and J. Slay, 'Towards developing network forensic mechanism for botnet activities in the IoT based on machine learning techniques', in *Mobile Networks and Management*, J. Hu, I. Khalil, Z. Tari and S. Wen, Eds., Cham: Springer International Publishing, 2018, pp. 30–44, ISBN: 978-3-319-90775-8.

[101] N. Koroniotis, N. Moustafa and E. Sitnikova, 'A new network forensic framework based on deep learning for internet of things networks: A particle deep framework', *Future Generation Computer Systems*, vol. 110, pp. 91–106, 2020, ISSN: 0167-739X. DOI: https://doi.org/10.1016/j.future.2020.03.042. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X19325105.

[102] N. Koroniotis, N. Moustafa, F. Schiliro, P. Gauravaram and H. Janicke, 'A holistic review of cybersecurity and reliability perspectives in smart airports', *IEEE Access*, vol. 8, pp. 209 802–209 834, 2020. DOI: 10.1109/ACCESS.2020.3036728.

[103] *The unsw-nb15 dataset*. [Online]. Available: https://research.unsw.edu.au/projects/unsw-nb15-dataset.

[104] N. Moustafa and J. Slay, 'Unsw-nb15: A comprehensive data set for network intrusion detection systems (unsw-nb15 network data set)', in *2015 Military Communications and Information Systems Conference (MilCIS)*, 2015, pp. 1–6. DOI: 10.1109/MilCIS.2015.7348942.

[105] N. Moustafa and J. Slay, 'The evaluation of network anomaly detection systems: Statistical analysis of the unsw-nb15 data set and the comparison with the kdd99 data set', *Information Security Journal: A Global Perspective*, vol. 25, no. 1-3, pp. 18–31, 2016. DOI: 10.1080/19393555.2015.1125974.

[106] N. Moustafa, J. Slay and G. Creech, 'Novel geometric area analysis technique for anomaly detection using trapezoidal area estimation on large-scale networks', *IEEE Transactions on Big Data*, vol. 5, no. 4, pp. 481–494, 2019. DOI: 10.1109/TBDATA.2017.2715166.

[107] N. Moustafa, G. Creech and J. Slay, 'Big data analytics for intrusion detection system: Statistical decision-making using finite dirichlet mixture models', in *Data Analytics and Decision Support for Cybersecurity: Trends, Methodologies and Applications*, I. Palomares Carrascosa, H. K. Kalutarage and Y. Huang, Eds. Cham: Springer International Publishing, 2017, pp. 127–156, ISBN: 978-3-319-59439-2. DOI: 10.1007/978-3-319-59439-2_5. [Online]. Available: https://doi.org/10.1007/978-3-319-59439-2_5.

[108] M. Sarhan, S. Layeghy, N. Moustafa and M. Portmann, 'Netflow datasets for machine learning-based network intrusion detection systems', in *Big Data Technologies and Applications*, Z. Deze, H. Huang, R. Hou, S. Rho and N. Chilamkurti, Eds., Cham: Springer International Publishing, 2021, pp. 117–135, ISBN: 978-3-030-72802-1.

[109] M. Zeeshan, Q. Riaz, M. A. Bilal, M. K. Shahzad, H. Jabeen, S. A. Haider and A. Rahim, 'Protocol-based deep intrusion detection for dos and ddos attacks using unsw-nb15 and bot-IoT data-sets', *IEEE Access*, vol. 10, pp. 2269–2283, 2022. DOI: 10.1109/ACCESS.2021.3137201.

[110] X. Larriva-Novo, V. A. VillagrÃ¡, M. Vega-Barbas, D. Rivera and M. Sanz Rodrigo, 'An IoT-focused intrusion detection system approach based on preprocessing characterization for cybersecurity datasets', *Sensors*, vol. 21, no. 2, 2021, ISSN: 1424-8220.

DOI: 10.3390/s21020656. [Online]. Available: https://www.mdpi.com/1424-8220/21/2/656.

[111] N. Guizani and A. Ghafoor, 'A network function virtualization system for detecting malware in large IoT based networks', *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 6, pp. 1218–1228, 2020. DOI: 10.1109/JSAC.2020.2986618.

[112] Y. Yang, K. Zheng, C. Wu and Y. Yang, 'Improving the classification effectiveness of intrusion detection by using improved conditional variational autoencoder and deep neural network', *Sensors*, vol. 19, no. 11, 2019, ISSN: 1424-8220. DOI: 10.3390/s19112528. [Online]. Available: https://www.mdpi.com/1424-8220/19/11/2528.

[113] O. Ibitoye, M. O. Shafiq and A. Matrawy, 'Analyzing adversarial attacks against deep learning for intrusion detection in IoT networks', *2019 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, 2019.

[114] L. Perlitz and S. G. Elliott, *The products*, https://aws.amazon.com/products/, 2000.

[115] *9 best heart rate monitors of 2022*, =https://www.healthline.com/health/fitness/heart-rate-monitor, 2022.

[116] W. Zhang, L. Deng, L. Zhang and D. Wu, 'A survey on negative transfer', *IEEE/CAA Journal of Automatica Sinica*, pp. 1–25, 2022.

[117] A. Alahmed, A. Alrasheedi, M. Alharbi, N. Alrebdi, M. Aleasa and T. Moulahi, 'Machine learning for real-time data reduction in cloud of things', in *2020 2nd International Conference on Computer and Information Sciences (ICCIS)*, IEEE, 2020, pp. 1–6.

[118] *Cicflowmeter*. [Online]. Available: https://www.unb.ca/cic/research/applications.html.