Protecting IoT Networks Against Routing Attacks

Philokypros P. Ioulianou

PhD

University of York
Computer Science

December 2021

Abstract

The rapid development of Internet of Things (IoT) will offer great benefits for both individuals and companies. However, as smart devices are widely deployed, they become attractive to hackers. Some recent examples are the 25 critical vulnerabilities discovered, known as "BadAlloc", which allow the execution of Denial-of-Service (DoS) attacks, as well as the existence of IoT malware such as Mozi which affect network operation. Therefore, new solutions should be developed to protect the computationally-limited devices.

In this work, a new Security Framework for IoT-based networks (SRF-IoT) is proposed. Our focus is on detecting and isolating attackers that exploit routing protocols which are used in 6LoWPAN IoT networks for packet routing. Although, many works study the security of routing protocols such as the IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL), they are still vulnerable to various attacks. We study the impact of well-known routing attacks such as DoS, rank and blackhole attacks in IoT networks. To investigate the impact of routing attacks, we design and develop the algorithms in ContikiOS, a popular Operating System, and using Cooja simulator we simulate the different scenarios. The obtained simulation results help us understand the characteristics of an RPL-based IoT network under its normal operation and devise effective countermeasures against malicious activity. The SRF-IoT framework contains a trust-based mechanism that identifies and isolates malicious attackers with the help of an external Intrusion Detection System.

Evaluation is based on simulations on a new simulator tool called Whitefield framework that combines both Contiki-NG and NS-3 simulator. This new simulator is used in this project as it allows large scale (over 100 nodes) realistic simulations using real-world stacks such as Contiki-NG. The analysis of the results showed the effectiveness of SRF-IoT in a network under combined rank and blackhole attacks with 92.8% Packet Delivery Ratio, and 8.2% packets dropped. Moreover, parent switches are kept low, reaching almost a hundred. Simulation results demonstrate that SRF-IoT is an efficient and promising solution to protect an IoT network against routing attacks.

Table of Contents

List of tables

List of figures

Acknowledgements

Throughout my studies and the writing of this thesis I have received a great deal of support and assistance.

I wish to express my sincere appreciation to my supervisor, Dr. Vassilios Vassilakis, whose expertise was invaluable in formulating the research problem and methodology. Your guidance and feedback pushed me to sharpen my thinking and brought my work to a higher level. Without your help, the objectives of this project would not have been achieved.

I would like also to thank my co-supervisor, Dr. Siamak Shahandashti, whose feedback during my thesis writing was really helpful. In addition, I would like to thank my examiners Dr. Poonam Yadav and Dr. Emmanouil Panaousis for reviewing my thesis and giving me really useful feedback.

In addition, I wish to acknowledge the support and great love of my parents for their wise counsel. I could not have completed this dissertation without the moral and financial support of my parents Panayiotis and Despina. I also thank my brothers who provided stimulating discussions as well as happy distractions to rest my mind outside of my research.

Finally, I would like to also gratefully thanks my future wife Stephanie who gave me a great courage and support throughout these years.

Declaration

I declare that this thesis is a presentation of original work and I am the sole author. This work has not previously been presented for an award at this, or any other, university. All sources are acknowledged as References.

Some of the material presented within this thesis has previously been published in the following papers:

- Ioulianou, Philokypros P., Vassilios G. Vassilakis, Ioannis D. Moscholios and Michael D. Logothetis. "A signature-based intrusion detection system for the Internet of Things." Information and Communication Technology Forum (2018).
- Ioulianou, Philokypros P., Vassilios G. Vassilakis, and Michael D. Logothetis. "Battery drain denial-of-service attacks and defenses in the Internet of Things." Journal of Telecommunications and Information Technology (2019).
- Ioulianou, Philokypros P., and Vassilios G. Vassilakis. "Denial-of-Service Attacks and Countermeasures in the RPL-Based Internet of Things" CyberICPS/SECPRE/SPOSE/ADIoT@ ESORICS (2019).

Philokypros P. Ioulianou
December 2021

Chapter 1

Introduction

1.1 Motivation

Recently, more and more smart devices are connecting to the Internet to improve our daily lives. According to predictions, there will be over 75 billion Internet of Things (IoT) linked devices operating by 2025 [1]. The growth of IoT will provide multiple benefits for companies or individuals. Different domains such as smart homes, industrial control, health monitoring, intelligent transportation, and smart grids are adopting IoT [2]. IoT devices are usually resource-constrained, with low processing power, small batteries, limited memory and processing resources. Nevertheless, they can support Internet connectivity, exchange small pieces of data with each other or with a server, and perform lightweight computations. An Operating System (OS) is responsible to manage energy, computing, communications and storage resources of a device [3]. As smart devices have limited resources, IoT OSes should be designed to efficiently manage devices' available resources while providing an interface for software developers to implement an energy-efficient software. An OS implements various protocols that allow devices to communicate with others in a network. Some of them are low-power Wi-Fi, Bluetooth, and IEEE 802.15.4. Using any of these links, IoT devices connect with each other and create the so-called Low-Power and Lossy Network (LLN). As devices in LLNs have limited resources, no existing routing protocol was suitable for those networks. Therefore, new routing protocols have been developed specifically for LLNs. The main routing protocol for LLNs introduced by IETF in 2012 was the IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL) [4].

As IoT becomes more popular, Things become attractive to malicious actors. Smart devices have little to no security defences and, therefore, security issues start to appear [5]. For example, IoT devices use simple default passwords, connect to weak wireless networks, and have outdated software. The lack of security protections in smart platforms allows

attackers to easily exploit and use them in external attacks such as Distributed Denial of Service (DDoS). Such attacks could severely impact the availability and integrity of large scale networks like smart grids or industry-based networks.

Apart from external network attacks, malicious actors may attempt to cause damage in internal networks. Attacks may occur at different levels of the IoT stack. The network or communication layer is mostly concerned in this thesis. Some well-known attacks of this level are Man-in-the-Middle (MitM), eavesdropping, routing attacks and Denial of Service (DoS). These types of attacks usually aim at causing service disruption or leaking sensitive information. Moreover, devices under attack might be unavailable, and routing information could be exposed to bad actors.

Some real-world issues were discovered recently in several IoT devices. Microsoft's security researchers recently discovered and publicly warned for 25 critical vulnerabilities in IoT devices, known as "BadAlloc" [6]. The vulnerabilities were found in the implementation of memory allocation of several OSs and Software Development Kits (SDKs). An attacker could exploit these vulnerabilities to successfully execute malicious code or crash the vulnerable device.

Another set of 33 vulnerabilities in TCP/IP stacks were found recently, affecting millions of IoT devices of over 150 manufacturers [7]. Four open source TCP/IP stacks are affected and found in multiple OSes such as Contiki and Nut/OS [8]. This set of new flaws is called "Amnesia:33" from the fact that the number of vulnerabilities is 33, and most of them cause memory corruption. Forescout researchers explain that in order "to exploit 'Amnesia:33' vulnerabilities, an attacker needs a communication path to a vulnerable device or a routed path to an internal network" [9]. In other words, attackers need to send malformed packets in order to exploit these vulnerabilities. Therefore, a suggested mitigation method is to monitor network packets for malicious packets and try to block them. Another solution is the use of internal Domain Name System (DNS) to avoid exploitation of DNS protocol. According to Forescout's data, IoT devices were the most affected by "Amnesia:33" bugs with 46%.

Another recent disclosure of multiple vulnerabilities from Forescout researchers is known as "NAME:WRECK" [10], [11]. In this disclosure, researchers found DNS vulnerabilities in open source TCP/IP stacks. These flaws were found in DNS parsing implementation, and could give attackers control over affected devices, execute malicious code or launch DoS attacks. The affected IoT devices were estimated to be over 10 billion [12].

Similar critical vulnerabilities have been found and disclosed by several researchers in recent months, including "NUMBER:JACK" [13], [14], "Ripple20" [15] and "Urgent/11" [16]. All these flaws could put in danger billions of devices. As mentioned by Samuel [17], the discovered vulnerabilities might be exploited by the following attacks:

- Remote Code Execution (RCE), in which attackers could remotely take full control of a vulnerable IoT device.
- DoS attacks, in which attackers could stop the operation of the device or a whole business.
- Information leak could allow unauthorised attackers to gather critical information about a network from affected IoT devices.
- DNS cache poisoning, in which attackers could inject false information in DNS cache to forward devices' traffic to a malicious domain.

Some of the above real-world threats are studied and implemented in this thesis. For instance, DoS and routing attacks impose serious threats for IoT networks. Therefore, our research focuses on those threats to design and suggest new mitigation methods. Using a real-world OS such as ContikiOS which is widely used by IoT manufacturers allows immediate evaluation of the system in a real-world environment.

As already mentioned, smart devices might be used for further attacks in external networks. Botnets, for example, usually target IoT devices for carrying out complex attacks. A recent real-world example is Mozi botnet [18] which targets IoT devices with poor or default telnet passwords running Secure Socket Shell (SSH) service. Its goal was to subvert a large number of devices and coordinate DDoS attacks to multiple targets. It combines the power of three different malwares; Gafgyt, Mirai, and IoT Reaper. Mirai [19] was the first IoT botnet that infected millions of devices (mostly IP cameras, home routers, and digital video recorders) and used them for DDoS attacks. The novelty of Mozi is that it attempts to form a peer-to-peer (P2P) botnet using the infected devices. Mirai and Gafgyt were using a centralized command and control server. Taking down a decentralized botnet is much more difficult than a centralized one. According to [20], Mozi infected over 15000 devices the last four months.

Taking into account the previously mentioned reasons, IoT networks should be protected from both internal and external attackers. It is important to safeguard smart devices as they are deployed in critical networks without proper security measures. The majority of IoT devices have limited computation power and operate using batteries. Thus, traditional security approaches such as Intrusion Detection Systems (IDSes) and cryptographic mechanisms are not always applicable [21]. As a consequence, many IoT devices are left with weak or no security measures [5] and become targets of cyber-attacks. Such attacks have multiplied over the last years [22].

As IoT networks consist of different devices that use multiple protocols and various technologies, an IDS solution should handle data packets coming from smart devices and

respond quickly to possible incidents. In addition, IoT devices have limited storage capacity, and low processing power while IoT-based protocols are designed differently from conventional ones. The data packet rate in IoT networks is higher than traditional networks due to multiple applications running simultaneously. As a result, protecting an IoT environment requires designing new security solutions that consider the resource-constrained nature of smart devices.

All in all, the above-mentioned and other similar incidents indicate that many security issues exist in IoT networks. Therefore, appropriate solutions should be developed to protect businesses, consumers, and critical infrastructure. Taking into account the Confidentiality, Integrity, and Availability (CIA) triad, the most important issue to address is to ensure data availability and integrity. In this thesis, our motivation is to protect IoT networks from various attackers and ensure that data obtained from sensors or other IoT devices is consistent, and available when needed. Therefore, DoS and routing attacks should be prevented or eliminated from creating problems to IoT devices and networks. For implementation and experimentation with IoT devices, we chose to emulate the behaviour of a real-world hardware platform along with an OS using a simulator tool. In this way, our designed solution will be evaluated using different configuration and in several IoT environments prior to real-world evaluation.

1.2 Problem Overview

The main goal of this thesis is to propose a new security solution to protect IoT-based networks from a combination of routing attacks. The detection of multiple combined routing attacks is one of the novelties of our work. Usually in IoT environments, smart devices are connected with each other to form a network. An IoT-based routing protocol is used to create routes from nodes to the root of the network. As smart devices have limited capabilities, routing protocols share knowledge and create routes in an efficient way. However, attacks may occur in a device and the routing protocol could be exploited.

As the focus of this thesis is the IoT network layer, routing attacks affecting the resources, network traffic and topology of the network are mostly considered. For example, resource-based routing attacks such as DODAG Information Solicitation (DIS) and DODAG Information Object (DIO) flooding attacks are mostly based on sending a large batch of requests to flood the network with control packets [23]. Those types of flooding attacks belong to the DoS attacks subcategory. Therefore, from this point and on we consider DoS attacks as resource-based routing attacks that drain the device's energy resources. Other types of resource-based attacks including version number modification and local repair attacks attempt to exhaust devices' power resources by changing packet parameters which

initiate further actions. In addition, attacks affecting network traffic such as decreased rank attack may cause routing loops, increase end-to-end delay, and waste device's energy [24]. Cyber-attacks targeting network topology are also important. For example, topology-based attacks such as blackhole attacks may lead to unoptimised paths and disruption in network operation.

In this thesis, routing attacks were chosen based on the following criteria:

- Limited number of research papers study them in the literature
- Type and impact in Availability, Integrity or Confidentiality of the network
- Limited or absence of mitigation method
- Ease of implementation

Having in mind the above criteria, the following routing attacks have been studied; "Hello" or DIS flooding and version number modification from DoS-based routing attacks, as well as blackhole and decreased rank attack from other routing attacks categories.

Therefore, the objectives of our thesis are the following:

1. Study the impact of DoS attacks and design a novel detection method.
2. Investigate the impact that a combined rank and blackhole attack has on a network, and implement a new mitigation method.
3. Improve the designed detection methods by using Machine Learning models to detect both known and unknown attackers.

According to [23], DoS attacks target WSN and affect the availability by flooding the IoT network with data packets. Moreover, DoS attacks and especially "Hello" or DIS flooding, which is an RPL-specific attack, have no mitigation method until today. This means, a malicious actor can compromise a device and flood a network with packets to cause traffic overhead and resource exhaustion. Although this is an RPL-specific attack, detecting attacks targeting network resources is still a problem under research, and novel solutions need to be proposed.

As long as topology and traffic-based attacks are concerned, a combination of blackhole and decreased rank attacks is utilised. Blackhole attack may disrupt network topology while decreased rank attack may use network traffic to impersonate root node leading to more severe attacks. This combination creates a complex environment where network availability and topology could be disrupted when the attack is initiated.

Although some security mechanisms exist in IoT-based routing protocols, many attacks cannot be mitigated with existing methods. For example, routing attacks such as blackhole attack cannot be easily detected. Thus, an IDS is deployed in IoT networks to detect any malicious attempts. A traditional IDS is not suitable for IoT networks as smart devices have limited capabilities. Apart from that, the majority of state-of-the-art studies propose solutions to detect one or more routing attacks. An IoT-based IDS is needed to secure smart devices from internal and external attackers. Identifying those specific attacks using an IoT-based IDS has already been extensively studied by research community. To our knowledge, none of the available solutions in the literature can detect a combination of different routing attacks in large scale IoT networks. Moreover, most of the solutions have centralised detection modules or require complex firmware modifications. A proper mitigation scheme should be easy to deploy and compatible with various devices and protocols. Last but not least, known and unknown attacks should be detected to better protect the networks. These requirements are taken into account before designing the proposed framework.

1.3 Proposed Solution Overview

A new solution that aims to mitigate a combination of routing attacks is proposed in this thesis. Figure 1.1 shows the proposed solution that is developed in this thesis. The novel Security Framework for IoT-based networks (SRF-IoT) can identify and isolate attackers by using two main components; an Anomaly-based Intrusion Detection System (SRF-IDS) and a Trust-based Objective Function (SRF-OF). The SRF-IDS is the main component that contains multiple internal modules while SRF-OF is embedded into the routing protocol.

Looking at the SRF-IDS component, the *Communication* module is mainly responsible for sniffing network traffic and communicating with neighbouring nodes. Specifically, it interacts with other SRF-IDS devices, captures network packets, and communicates with monitored nodes that use SRF-OF. Apart from that, it collects packets from different interfaces and forwards them to the *Preprocessor* module. In this module, data packets are organised together and decoded before further processing. After this step, the *Trust Monitoring (TM)* module receives the packets for extracting useful measurements such as the number of packets forwarded. The TM module sends trust metrics to monitored nodes via the *Communication* module after decoding them in the *Preprocessor* module. Exported metrics from analyses are passed to a critical component, the *Detection* module. This module is responsible for identifying attackers using predefined thresholds and signatures obtained from the *Configuration* module. Suspicious nodes' details are forwarded to the *Decision* module of the SRF-IDS root node, called SRF-IDS root, for final decision-making. The

SRF-IDS root takes into account the behaviour history of a node and the information from the *Configuration* module. If a malicious attacker is found by the SRF-IDS *Decision* module, a firewall rule is automatically created for blocking suspicious node traffic. Moreover, an alert is shown to the system administrator by the *Alerting System* module.

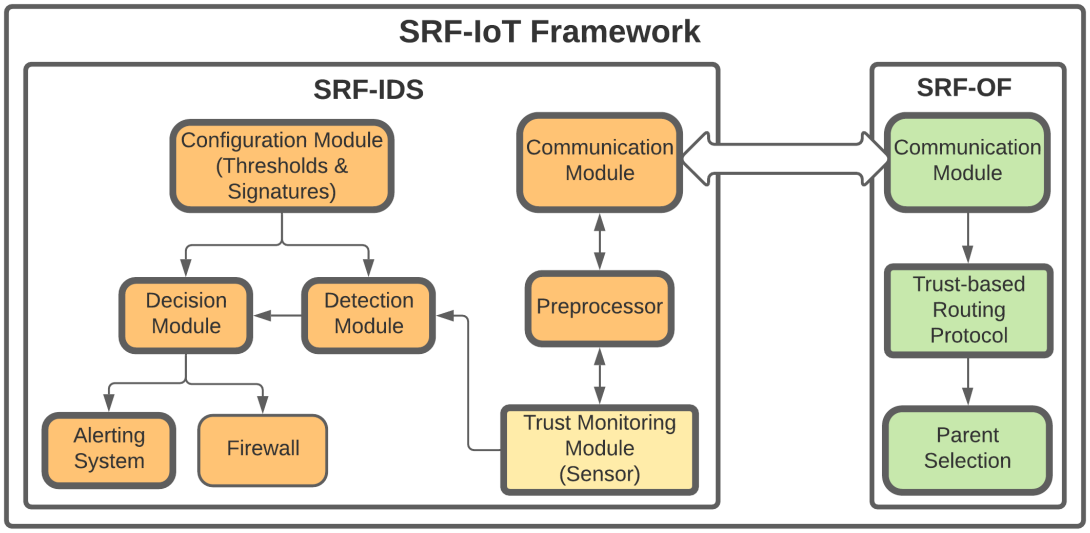


Fig. 1.1 Proposed solution overview

As regards to the SRF-OF component, it is an important module used by nodes to increase SRF-IoT framework's effectiveness. Trust concept is introduced in our work as an extra security mechanism to enhance the detection capabilities of the suggested framework. Trust score is a value indicating whether a device forwards received packets to next hop or not. The calculation process is embedded into the routing protocol and is done in the *Trust-based Routing protocol* module. Then, the *Parent selection* module executes an algorithm to choose the best parent based on trust score value. Using this trust-based approach aims to detect routing attacks such as blackhole and rank attacks as well as other complex attacks.

A Machine Learning (ML) approach is also explored by the SRF-IDS *Detection* module in this thesis. Using ML algorithms will significantly enhance the detection abilities of SRF-IDS as it will enable the detection of both known and unknown attacks after proper training. Before deploying the module, received data will be provided as an input into the model, and proper ML algorithms will be used to train the model. The output of the model will indicate if a node is malicious or not. Another feature is that DIS or "Hello" flooding can be detected from this module. Communication between SRF-IDS and deployed smart devices is done with the help of special control messages sent from network layer. We assume these packets are encrypted to avoid leakage of sensitive information.

One of the novelties of our approach is that a combination of trust-based and IDS-based method is achieved. Most studies propose mitigation schemes for various routing attacks using the former or the latter. Combining the two approaches is a great innovation in the field of IoT security as it enhances the detection performance and allows detecting attackers faster. Furthermore, mitigation methods for additional routing protocol attacks can be easily added. Apart from that, monitored nodes' energy consumption is minimised. Nodes operate normally without any watchdog mechanism as in other trust-based solutions, so no additional energy is consumed. SRF-IDS detector nodes have embedded a watchdog mechanism to monitor network and provide monitored nodes with trust metrics.

Another novelty of our approach is the deployment of SRF-IDS along with the normal network. The SRF-IDS consists of a centralised router that hosts the configurations with known signatures, thresholds, the Detection module, the Decision module, and acts as a firewall. Apart from that, decentralised SRF-IDS detectors are responsible for traffic monitoring, and local detection module. Alerting monitored nodes with trust metrics is also the task of SRF-IDS detectors. In order to monitor neighbouring network without interruptions, the SRF-IDS is deployed in a different RPL Instance than the normal network. All components of SRF-IoT indicated in Figure 1.1 are implemented in this thesis except for the firewall. We assume that a traditional network firewall can be deployed along with the SRF-IDS, so that appropriate rules will be created to filter malicious traffic.

The three significant contributions of this thesis that address the objectives discussed in Section 1.2 are the following:

a) An Anomaly-based IDS has been designed and developed using an overlay approach to easily monitor and help IoT devices isolate malicious actors. Evaluation results showed 100% True Positive rate when 9 detectors are deployed, less than 1% False Positive rate, and low traffic overhead in scenarios under routing attacks such as DoS attacks.

b) A novel Security Framework for IoT-based networks (SRF-IoT) has been designed and proposed to avoid several routing attacks including DoS, rank and blackhole attacks. The framework consists of the anomaly-based IDS, called SRF-IDS, and a trust-based OF, called SRF-OF. It has been implemented in RPL protocol and evaluated under routing attacks. Experimental results depict that SRF-IoT framework can effectively detect and help nodes to avoid malicious nodes. Our proposed framework showed 92.8% Packet Delivery Ratio (PDR), 97 parent switches, and 8.2% packets dropped in scenarios with active blackhole and rank attacks.

c) An innovative ML-based detection module for SRF-IDS has been designed and evaluated for detecting known and unknown attacks. It has been trained and tested using simulation results generated by SRF-IoT evaluation phase. Results from evaluating the ML model showed Precision of 93.3%.

As the proposed solution aims to protect IoT networks from malicious actors, the contributions are achieved.

Regarding our published works, the papers “A signature-based intrusion detection system for the Internet of Things” (2018) [25] and “Battery drain denial-of-service attacks and defenses in the Internet of Things” (2019) [26] are related to Contribution (a), and paper “Denial-of-Service Attacks and Countermeasures in the RPL-Based Internet of Things” [27] is related to Contributions (a) and (b).

1.4 Project Methodology

In order to understand a problem and design a new solution, it is required to create a process that will be followed and eventually will help mitigate the problem. In this section, processes done in both general and adopted approaches are analysed and explained.

1.4.1 General Approach

As in many research studies, studying a problem and proposing the proper solution is a multi-step task. Thus, many small tasks are required to be done before implementing and evaluating a new system. Figure 1.2 shows the general methodology that is usually adopted by researchers when studying an experiment-based project. As it is shown, the general methodology is divided into five steps; defining the problem, studying the problem, designing the solution, implementing the solution and evaluating the solution.

Fig. 1.2 General methodology of experiment-based projects

The first step is to define the research problem that researcher wants to study. In this step, a literature review is done to study and identify exactly the problem. After this step, a more detailed study of the identified problem is needed. Finding the aspect and proposed

solutions for the problem are the important outcomes from this procedure. The next step is to start designing a solution for the problem. Having in mind the weaknesses of current solutions, researchers propose novel methods to mitigate the current problem. When the design of the solution finishes, the implementation phase starts. In computer science research field, implementation usually means coding or development of the suggested software in a platform that was defined in the previous phase. Evaluating the developed solutions in the final step and most important. This process shows if the suggested solution works as expected and mitigates the problem that was initially studied. If the developed solution cannot be evaluated in a real platform, performance can be tested using a simulator.

1.4.2 Adopted Project Methodology

The methodology followed in this project is very similar to the general methodology described previously. Figure 1.3 shows the six defined steps; define the top IoT-based threats, study selected attacks through simulations, analyse results from simulations, design the suggested SRF-IoT solution based on previous results, implement the solution, and evaluate the solution.

Beginning from the first step, the initial goal was to define the most severe IoT threats. This required extensively studying the literature so that the problem is identified. In our case, the problem is IoT attacks that disrupt network operations. Specifically, version number modification, DIS flooding, decreased rank attack and blackhole attacks were studied. After finding the four most dangerous IoT-based attacks, the next task was to study their impact through simulations. Many works in the literature have been simulating attacks to find out how much damage they may cause in a network. Therefore, the same approach was adopted, and an implementation of each attack was done in a real-world platform. Then, several simulation scenarios were generated to simulate the behaviour of real-world devices using various configurations. These scenarios contained different cases where both benign and malicious devices were deployed.

Once simulations were finished, results were exported and analysed. Simulator tool generated log files which were analysed to extract useful metrics, check nodes' behaviour once attacks launched, and investigate the type of transmitted packets. Moreover, network topology was tested to ensure that attackers successfully deployed attacks as expected. The next task was to design the proposed SRF-IoT solution based on the simulation results. The components of SRF-IoT are the IDS, called SRF-IDS, and the trust-based routing protocol, called SRF-OF. SRF-IDS was designed to detect DIS flooding attack as well as decreased rank attack and blackhole attacks. Regarding DoS attacks, malicious attackers are detected based on a threshold. The threshold value was extracted from the results of the previous step. Moreover, analysis from previous step helped us define the normal and malicious

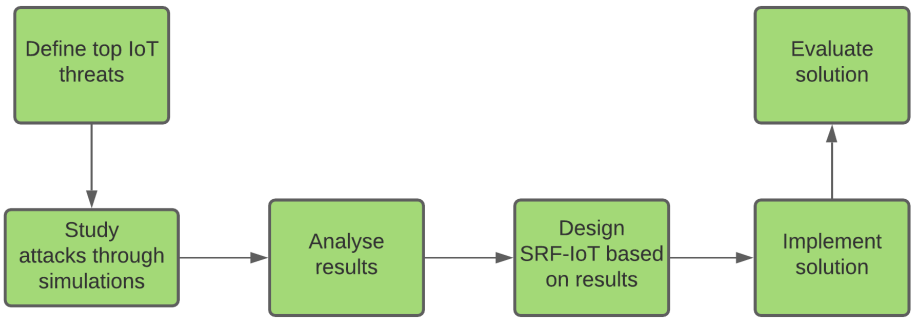


Fig. 1.3 The methodology followed throughout this project

packet forwarding behaviour as well as a trust-based metric. These metrics are utilised by SRF-IDS as detection method for rank and blackhole attacks. Specifically, SRF-IDS was designed to interact with the trust-based SRF-OF so that rank and blackhole attackers can be easily identified by individual monitored nodes. As a result, benign nodes in the monitored network isolate attackers with the help of SRF-IDS. In addition, communication between the two components is achieved by using special control messages in the routing protocol. Further design decisions are explained in the following chapters. Implementing the actual SRF-IoT scheme was the next procedure. The two components, SRF-IDS and SRF-OF, were developed in a real-world OS. As IoT devices have limited capabilities, efficient coding methods were used. Testing the developed software is a crucial step. SRF-IoT framework was extensively evaluated using simulations of a real-world platform. The solutions were tested to achieve the expected detection rates and results. Further simulations were done to evaluate the performance of the solution in different network topologies.

1.5 Thesis Structure

The chapters of the thesis are the following:

In Chapter 2, some useful background information is provided. In Section 2.1, we introduce the IoT reference model that is used throughout the thesis. OSes available for IoT devices are discussed in Section 2.2. Moreover, in Section 2.3 we review and compare simulators that are available for IoT research. Later in Section 2.4 we introduce the RPL routing protocol that is available for IoT networks. Section 2.5 reviews the most significant attacks in IoT networks. We analyse the attack vectors in each layer of IoT stack as well as routing attacks such as blackhole and rank attacks. In Section 2.6 background information about solutions to detect IoT attacks such as Intrusion Detection System (IDS) are discussed.

Section 2.7, discusses supervised ML algorithms for IDS systems. In Section 2.8, the related research work is presented. In Section 2.8.1 we briefly present recent works in the field of RPL security. The latest studies in the field of routing attacks including DoS attacks and prevention methods in IoT networks are described in Section 2.8.2. Trust-based IDS and RPL protocol-based mitigation techniques are discussed in Section 2.8.3. In Section 2.8.4 we talk about current IDS solutions for IoT networks. A chapter overview is given in Section 2.9.

In Chapter 3, we study and implement resource as well as traffic -based routing attacks in IoT networks. Section 3.1 provides a brief description of the problem. Then, in Section 3.2 two popular types of IoT network layer-based DoS attacks are studied: "Hello" or DIS flooding attack and version number modification. Both attacks may drain the batteries of IoT devices. These attacks are implemented in ContikiOS and simulated using Cooja simulator. We demonstrate how these attacks can impact the power consumption of IoT devices and can constitute some devices unreachable. Following the presentation of simulation results, in Section 3.3 we focus on RPL-specific DoS attack, called "Hello" or DIS flooding, which is simulated in large networks. Mean packet interval and mean number of messages were utilised to show the effects of the attack. Exported results are analysed and IDS thresholds are defined to combat the IoT-specific attack. Section 3.4 discusses the implementation details of other resource and traffic-based routing attacks such as rank and blackhole attacks. Malicious devices attack the network by advertising false rank and dropping packets of child nodes. The last Section 3.5 briefly summarises the chapter.

In Chapter 4, a novel Security Framework for IoT-based networks (SRF-IoT) is introduced. Section 4.1 presents the operational details of the framework such as trust concept. Also, a high-level architecture of SRF-IoT scheme is presented. The two main components of SRF-IoT framework are introduced in Section 4.2. An external Anomaly-based Intrusion Detection System (SRF-IDS) is the first main component that aims to detect attackers, and provide smart devices with trust metrics. A Trust-based Objective Function (SRF-OF) which is embedded into the routing protocol is the second component. This OF allows nodes choose the most trusted parent to route packets, and avoid malicious actors in a network. The modular Anomaly-based SRF-IDS for detecting DoS attacks is presented in detail in Section 4.3. Initially, a brief discussion of current problem is given and then, the design of SRF-IDS is discussed. SRF-IDS comprises a set of distributed lightweight detection modules and a border router acting as a centralised detection module. The architecture and several components of the proposed SRF-IDS are explained in detail. In Section 4.5, implementation details of SRF-IDS communication module are presented. Moreover, Section 4.6 provides more information about SRF-OF and trust-based protocol implementation. The development of detection module using thresholds is presented in Section 4.7. Design phase,

and implementation of algorithms are explained with details. An initial version of detection module is evaluated using Cooja simulator tool. Section 4.8 describes an improved version of threshold-based detection module. The weaknesses of the first version are initially described. Later, changes in the design as well as simulation scenarios are presented. Results from evaluating the improved detection module are also illustrated. SRF-IoT design decisions as well as algorithms for information collection, trust calculation, trust monitoring, and detection procedure are explained in detail in Section 4.4. A brief summary of chapter information is given in Section 4.9.

In Chapter 5, performance evaluation of SRF-IoT framework is presented for three different cases. The first case described in Section 5.1, evaluates SRF-IoT framework when rank and blackhole attackers are deployed in a medium-size network. As a second case, a complex attack scenario is presented in Section 5.2. Specifically, the proposed scheme is evaluated in a medium-size environment where malicious nodes attack the network using a combination of blackhole, rank and DIS flooding attacks. In Section 5.3, SRF-IoT framework is tested in large scale networks where the three routing attacks are launched together. Results show that DIS flooding and other routing attackers are detected and isolated with the help of SRF-OF. In each different case, the specific scenarios, configurations, and metrics are analysed, and results are presented. Attack simulation was done in Whitefield framework; a new simulator that combines Contiki-NG with NS-3 to allow faster simulations, easier deployment of large scale scenarios with minimum effort, and more realistic results than Cooja simulator. In Section 5.4, an ML model that is embedded into the Detection module of SRF-IDS detectors is introduced. Initially, the aim of deploying an ML model into SRF-IDS component is described. Then, the related design decisions are discussed. In addition, the general approach followed from creating the dataset until training and testing the ML model is described. The actual implementation steps taken for creating the datasets are also explained. Configurations and metrics as well as experimental results are presented. The trained model shows precision of 93.3% when is validated. In Section 5.5, observed results are discussed and compared with other related works. Section 5.6 presents a brief summary of the chapter.

In the last Chapter 6, conclusions along with limitations and future steps are discussed.

Chapter 2

Background and Literature Review

It is important for the reader to understand the basic concepts of IoT networks before proceeding to next chapters. The aim of this chapter is to describe the most important concepts and definitions used in this thesis. First, the IoT reference model and the available Operating Systems for smart devices are introduced. Then, simulators used for IoT research as well as the RPL routing protocol are explained along with the potential attacks. The definition of Intrusion Detection System (IDS) for IoT is given as well as the supervised ML algorithms implemented in SRF-IDS are discussed. Last, the related state-of-the-art studies are presented.

2.1 IoT Reference Model

In traditional networks, a reference model is a theoretical model that is used as a basis for communication among various network systems. The same concept is used for IoT. A number of different IoT reference architectures have been proposed for IoT [28]. Among them, CISCO's 7-layer model [29] provides sufficient level of detail and has been considered in this study.

It consists of the following layers: Starting from Level 1 or the "edge level", physical devices are the smart devices and machines, which send or receive generated/censored data. A device can be of any size and ,thus, the IoT reference model describes generally the basic capabilities for devices. For example, a device should be able to convert signal from analog to digital, and generate data.

Level 2 refers to the communication and connectivity between the devices, within the same network or across different networks. In many implementations, IoT devices are able to reliably transmit data using the existing network infrastructure. Apart from that, in this

level routing, switching and translation between protocols occurs. So, most of Level 2 functionalities concern the communications within a network.

Level 3 activities include data analysis and transformation. In other words, network packets are processed in that level to be understandable to the higher levels. Specifically, in this level data are filtered, formatted and aggregated to reduce their size and be suitable for storage. Moreover, as the processing is done per network packet, their content is inspected. Generally, Level 3 functionalities focus on communication packets entering or leaving the network.

Level 4 is where data are converted from data in motion to data at rest. In other words, network packets or event-based data are formatted into database relational tables. This allows application to use those data when needed. The size of data is also largely reduced by filtering them.

In the next level, data abstraction occurs. This means that in Level 5, data gathered from different sources can be combined and simplified for use in applications. Data abstraction is achieved by creating schemas of data in a way that are useful for applications. Moreover, techniques such as data filtering, projecting, and reformatting are used to transform data into proper format for application use. In addition, proper data protection methods such as authentication and authorisation are implemented in this level to secure data.

Level 6 is the application level and is the place where information is read by IoT applications. The latter can vary from analytics to system management and control. For example, control applications and business intelligence applications are likely to use the data in various ways. Applications in this level communicate with Level 5 and data stored in databases.

The highest level, Level 7, is where the end users and business processes live. Making the IoT system and its generated data useful requires people to collaborate and use IoT applications and their data.

In general, Levels 1-3 are called the “Edge-side layer”, Levels 4-6 are called the “Server/Cloud-side layer”, and Level 7 is called the “User-side layer”. In this work, we are mostly concerned with “Edge-side” layer or Layers 1-3.

2.2 Operating Systems in IoT

Many operating systems have been used in the IoT research field. An Operating System (OS) is a vital component for a device in order to run applications and manage its resources [30]. For instance, OS is responsible for monitoring and managing energy consumption of the device, executing instructions, and enabling the device to communicate with other devices. IoT networks operate on resource-constrained devices. As discussed before, smart devices

have limited storage capacity, low processing power, and operate on batteries. Customised IoT-based OS have been designed to meet these constrains and enable programs to run on IoT devices. There are two types of IoT OS; Linux-based and non-Linux based. LiteOS, Pyrix and ARM Mbed are some examples of Linux-based OS, while ContikiOS/NG, TinyOS, RIOT and Openthread are non-Linux OS [31, 32]. As this project focuses on resource constrained devices, non-Linux OS are studied mostly.

One of the well-known OS in the literature for Wireless Sensor Networks (WSN) is ContikiOS [33]. It has been under development since 2003, and follows a modular architecture. In this type of architecture, applications can be loaded and unloaded dynamically during runtime. Moreover, ContikiOS is based on Protothread which features multi-threading and event-driven programming model. It also supports dynamic memory allocation, and full TCP/IP stack via the uIP. ContikiOS is using C programming language for implementation. It is a flexible OS and it has been integrated into many different hardware platforms. In 2017, Contiki-NG [34] started as a fork of the original ContikiOS. Its goal is to focus on dependable low-power communication, implement standard protocols such as IPv6/6LoWPAN, CoAP and RPL, and improve the documentation. Moreover, Contiki-NG has an active community which releases regular updates. Regarding RPL protocol, ContikiOS provides an implementation that is called ContikiRPL [35] while Contiki-NG implements a lighter version of ContikiRPL, called RPL-Lite. The latter version of RPL removes support for storing mode in favour of non-storing mode, and removes the complexity of handling multiple instances. Usually, RPL-Lite shows better performance and has a considerably smaller ROM footprint than ContikiRPL.

Another popular OS for constrained devices is TinyOS [36]. One of the difference with other OS is that it follows a monolithic architecture. This means that all processes run in kernel space and in case of application bug, the whole OS crashes. Another difference is that memory is fixed and it is statically allocated. TinyOS applications run as independent components which are written in NesC programming language and have an event-driven execution model. Commands, Events and Tasks are three elements written in C that exist in each component. Commands are usually queries to a component that demand something to be executed. Events are signals sent from the component once a Command is finished. Both Commands and Events are executed immediately while Tasks are not. If a program is running and a new Task is created, the task will be added to the queue and executed later by the scheduler, after the execution of the program is finished.

RIOT is a Real-Time Operating System (RTOS) for embedded smart devices [37]. It is based on microkernel architecture with multi-threading. This means it has small kernel size and a small number of context switches. This reduces the amount of memory needed

by hardware devices. RIOT is organised in software modules that are combined together during compile, and allows dynamic memory allocation. Low memory and power resources are needed by devices because only required modules are compiled in the system. Various stacks such as standard IP protocols, 6LoWPAN, IPv6, RPL are supported in RIOT. The programming language for applications in RIOT is C.

FreeRTOS is another RTOS designed for embedded and devices with limited capabilities [38]. It is deployed in many industrial/commercial environments and also used in many research studies. It is an easy to use, small and portable OS that has been integrated into many hardware platforms including IoT-LAB test-bed. Similarly to RIOT, it is based on microkernel architecture and supports multi-threading programming model. In addition, dynamic memory allocation is supported while third-party libraries are needed for Internet communication. Applications are written in C programming language but C++ can also be used.

In this project, an OS is used to help us design, develop and demonstrate the proposed solution into a real environment. A comparison among the different IoT OSes is presented in Table 2.1. As it is shown, ContikiOS/NG is better for IoT research as it supports both multi-threading and event-driven approaches, and allows modular development of applications as it has a modular architecture. Another advantage is that it can be easily emulated in a simulation tool such as Cooja [39]. For these reasons, ContikiOS/NG is chosen for implementing the proposed solution in our thesis.

Table 2.1 Comparison of IoT Operating Systems

Operating System	**Architecture**	**Programming Model**	**Programming Language**	**Memory Allocation**
TinyOS	Monolithic	Event-driven	NesC	Static
ContikiOS/NG	Modular	Multithreading and Event-driven	C	Dynamic
RIOT	Microkernel	Multithreading	C	Dynamic
FreeRTOS	Microkernel	Multithreading	C and Assembly	Dynamic

2.3 Simulators for IoT Research

When designing and testing new solutions, many researchers rely on simulation tools. In recent years, a number of open-source simulators has been made available for WSN and IoT research [40]. The following features should be supported by a simulator:

- Generate Real Network Traffic: It is essential to have and use the actual traffic that the network would have when it is implemented in order to study the attack impact. This means, if simulator does not support a real software, it will not create real traffic but will generate traffic patterns such as distributions.
- Simulate Real Software: Each simulated node should behave in the same way as a real software. So, it is important to simulate real software using a simulator.
- Support IoT OS: Simulating a real-world OS running on IoT device is really important because simulator can produce realistic results.
- Calculate Power Consumption: Estimating the energy consumption is a great metric to have it in a simulator.

Two popular general-purpose simulators for computer networks are NS-3 [41] and OMNeT++ [42]. Both are discrete-event simulators implementing many real-world network protocols. They have been developed in C++ and can be embedded into existing projects as libraries. Castalia [43] is an extension of OMNeT++ to support WSN simulations. It is mainly used by researchers to study MAC and PHY layers as well as test algorithms in a realistic wireless channel and radio model. TOSSIM [44] and Cooja [39] simulators are also gaining popularity among WSN/IoT researchers. Both are particularly suitable for real-world experiments, since the developed applications can be uploaded directly to real hardware. TOSSIM is the simulator of TinyOS [45] and allows writing application in nesC language. Cooja can be used to simulate the behaviour of ContikiOS [46] and is flexible in the sense that nodes of the same network can run different software or have different underlying hardware platforms.

A recent simulator that combines NS-3 and real-world OS is the Whitefield Framework [47]. According to its author, Whitefield provides a simulation environment for WSNs by combining realistic PHY/MAC layer simulation with the native mode use of popular IoT Operating Systems. In our case, we use Contiki-NG as the OS which provides the network layer and above, while NS-3 simulator provides the PHY/MAC/RDC layer. Moreover, Whitefield generates log and pcap files for each simulation. This is really useful for monitoring and auditing simulation results. The only drawback of using Whitefield is that deployed nodes are native processes running in NS-3 and not emulated hardware. Therefore, monitoring energy consumption or other hardware-specific metrics is not possible.

A comparison of the aforementioned simulators is presented in Table 2.2. As it can be seen, most simulators do not generate real traffic by running real applications. Real network traffic is created by Cooja, for example, as it is based on ContikiOS system. Other simulators,

Table 2.2 Comparison of Popular Simulators for IoT

Simulator	Main Usage	Traffic Generation	IoT Devices Support/ OS	Power Consumption	Limitations
Cooja	Wireless sensors	Real	Yes/ ContikiOS	Yes with Powertracker	Supports less than 100 nodes
TOSSIM	Wireless sensors	Statically or Dynamically	Yes/ TinyOS	Yes with PowerTOSSIM	Restricted to TinyOS
OMNeT++	Internet protocols	Events	No/ -	Yes	Lack of software code
NS-3	Network and application layers	Traffic patterns	No/ -	Yes	Lack of real traffic
Castalia	Focus on radio models	Real	Yes/No	Yes	General Simulator
Whitefield Framework (based on NS-3)	Wireless sensors	Real/Traffic patterns	Yes/ Contiki-NG	No	Lack of energy consumption model

such as TOSSIM and NS-3, create dynamic traffic or using patterns such as exponential distributions. Another important feature that is missing from the half of simulators is the support of real-world OSes, and IoT devices. Simulating with a real OS will allow us to generate realistic and accurate results regarding the impact of attacks in various IoT networks. In addition, the proposed solution should be good to be implemented in an OS that can be easily uploaded into a real hardware device.

Given the characteristics and capabilities of the above simulators, we have chosen Cooja for implementing the developed IDS and the Whitefield Framework for evaluating the SRF-IoT framework. Cooja simulator allowed us measure the energy consumption of our initial experiments as it supported this feature. Moving into Whitefield framework, this feature was not available anymore. Therefore, we changed our focus and we did not measure energy consumption. However, we were able to experiment with large networks. Simulating more than 30 nodes in Cooja was a limitation using our current setup but in Whitefield we mitigated this problem. Cooja was the first choice to simulate ContikiOS in small networks for evaluating IDS while Whitefield framework is used for more realistic, large scale scenarios that are simulated without any memory or computing power limitations. Simulating ContikiOS is preferred than TinyOS because the former is a dynamic system, allows allocating resources in run-time, and supports the communication with external networks using the TCP/IP protocol suite.

2.4 RPL Routing Protocol for IoT Networks

A routing protocol aims to achieve communication between devices belonging to a network or in different networks. Nodes in IoT networks deploy different types of routing protocols to exchange packets effectively with other devices [48].

The IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL) [4] is a standardised lightweight routing protocol for 6LoWPAN networks. A Destination-Oriented Directed Acyclic Graph (DODAG) is created between the nodes. One or more DODAGs can be grouped together and form an RPL Instance. In an RPL Instance, all DODAGs share the same RPL Instance Identification (RPL InstanceID). The RPL InstanceID must be included in a field in data packets so that nodes can identify destination. Moreover, an instance may include one DODAG with one root node or multiple DODAG with different root nodes. However, a node can participate in one RPL Instance only at a time. A single 6LoWPAN network may have many global RPL instances which may also include several local RPL DODAGs.

An Objective Function (OF) is utilised in RPL protocol to select and optimise routes within an RPL instance. Usually, OF is based on some metrics or constraints (e.g., energy, latency, and throughput). The result of the function is the rank of the node, which is an indication of the node's distance from a DODAG root (hops from the root node). Two OFs are supported by RPL; Zero Objective Function (OF0) and Minimum Rank with Hysteresis Objective Function (MRHOF) namely. OF0 is based on hop count to calculate rank while MRHOF uses the Expected Transmission Count (ETX). That is the expected number of transmissions that a node needs to make to a destination in order to deliver a message successfully. Rank increases with every hop from the root, thus root has the minimum rank. This metric is also needed in the preferred parent selection process of a node. Nodes with lower rank are preferred. Parents are nodes that forward packets from a child node. Parents are by definition, closer to the root, which is translated to a lower rank.

RPL supports two modes; non-storing and storing. In non-storing mode, the root is responsible for storing information about each node in the DODAG network. Each packet has to go upwards to the root and then, the root calculates the routing path to the destination based on the stored information. The root inserts this routing path in packet header and sends it to the next node. When a node receives a packet, it forwards the packet to the next hop node (as indicated in the packet header). In storing mode, each nodes maintains a list that stores routing information about all nodes belonging to its sub-DODAG. When a packet arrives, the recipient node finds the packet destination from the list and forwards it to the next node. If the destination is not listed, the packet is forwarded to the preferred parent.

A number of ICMPv6 control messages are used in RPL. DODAG Information Option (DIO) messages are sent by the root. These messages are needed to maintain the DODAG, and contain information about the metrics used for routing, the rank of the broadcasting node, and the identity of the DODAG. If a DIO message is received by a node, the node will determine its rank, based on received rank, and the cost of getting to the node from itself.

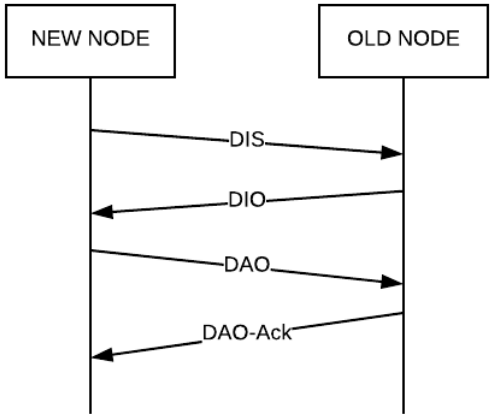


Fig. 2.1 RPL control messages

DIO messages are sent regularly from each node, triggered by a timer called trickle timer [49]. Using a timer reduces the redundant transmissions of DIO messages.

DODAG Information Solicitation (DIS) messages are used to solicit a DIO from RPL node. In other words, it is used as a neighbour discovery. When a new node joins the DODAG, it multicasts a DIS message and waits to hear for a DIO. A Destination Advertisement Object (DAO) message is sent to propagate information upwards, and to create downward routes. Nodes send this message to the root, so the message is propagated or forwarded by parent nodes until root node is reached. Then, a Destination Advertisement Acknowledgement (DAO-ACK) is sent as an answer from the root node when receiving a DAO. The transmission sequence of RPL control messages between a new node and an old one is shown in Figure 2.1.

Each node has a unique ID, based on its IPv6 address, and maintains a list of its DODAG neighbours. A node has one or more parents, except for the root that has no parents. Additionally, RPL nodes use a rank-based system to indicate their position with respect to the root, who has the lowest rank. The direction from root to other nodes is called *downward route*, while the direction from other nodes to root is called *upward route*.

2.5 IoT Attacks

Attacks take place at different levels in the IoT infrastructure. The combination of routing attacks such as DoS, rank and blackhole attacks is the novelty of our work. Those attacks as well as other attack vectors are explained in this section. The content of this section is related to **Contributions (a) and (b).**

2.5.1 Attack Vectors

The majority of IoT devices have weak or no security at all making it easy for an attacker to exploit them [5]. As a result, critical information can be stolen from devices or they can be used to cause harm in other networks. Below, we briefly review the most significant attack types against IoT devices in each layer of the IoT stack. For referencing purposes, Cisco's 7-layer model is used [29].

At the Physical Devices layer, changing the firmware of a smart device with a malicious one could permit an attacker to read data in transit or stored in the device. Another method of hardware exploitation is the non-network side-channel attack. In that attack, electromagnetic signals of the device are monitored by an attacker so as to expose the status of the device. DoS attack constitutes another threat for smart devices. Resource exhaustion and battery draining are some examples of DoS attacks [50]. In these attacks an attacker may prevent a device from sleeping by periodically transmitting "Hello" messages or may drain the limited power resources by submitting heavy computation tasks. Apart from DoS attacks, an adversary could attack the network by cloning a node. This means, the node will normally participate in the network but node will be controlled by the attacker. In this way, packets received by the node could be redirected or modified.

At the Connectivity level, eavesdropping is a popular attack in which the goal of the attacker is to export confidential information including usernames and passwords. Therefore, the attacker can learn about the network infrastructure, enter and modify device's data, or steal important information. Also, at this level devices are vulnerable to MitM attack in which attackers become the proxy between nodes and root, controlling all the communication. Routing attacks are also well known in IoT networks. Usually, infected nodes redirect or change routing paths of neighbouring nodes to disrupt network operation. Some routing attacks examples are selective forwarding, sinkhole and blackhole attacks. Moreover, replay attacks [51] can be very dangerous as attackers try to spoof and drop packets or even modify routing information. This leads to network disruption. Another type of attack is device spoofing. In this case, a malicious node impersonates a normal device to steal credentials and gain unauthorised access to an IoT application. In addition, DoS attacks at the level of Connectivity may have a negative impact on the performance of an IoT network. Some examples of DoS attacks are packet flooding and signal jamming which have as a goal to corrupt device's communication signal. Last but not least, IoT devices can be exploited and transformed into bots to carry out DDoS attacks against selected targets. Chalubo and Mirai botnets are the most recent examples of this threat [52, 53].

At the Edge Computing level, servers could be exploited by injecting malicious input into them and stealing important data. Similarly, attackers may try to leak information

from a device or server in order to learn which services are used in a vulnerable IoT network. Database warnings or errors, for example, provide valuable information to attackers. Additionally, access control attack can cause significant damage to IoT network if is not prevented. IoT applications utilising access control do not allow illegitimate users or devices to access sensitive data. Once access control in a critical IoT application is compromised, then anyone could get access to it.

2.5.2 Routing Attacks

Routing attacks may degrade the performance of IoT devices. Thus, extensive study is needed to understand how they actually occur. Below, we present some background information about routing attacks in IoT networks.

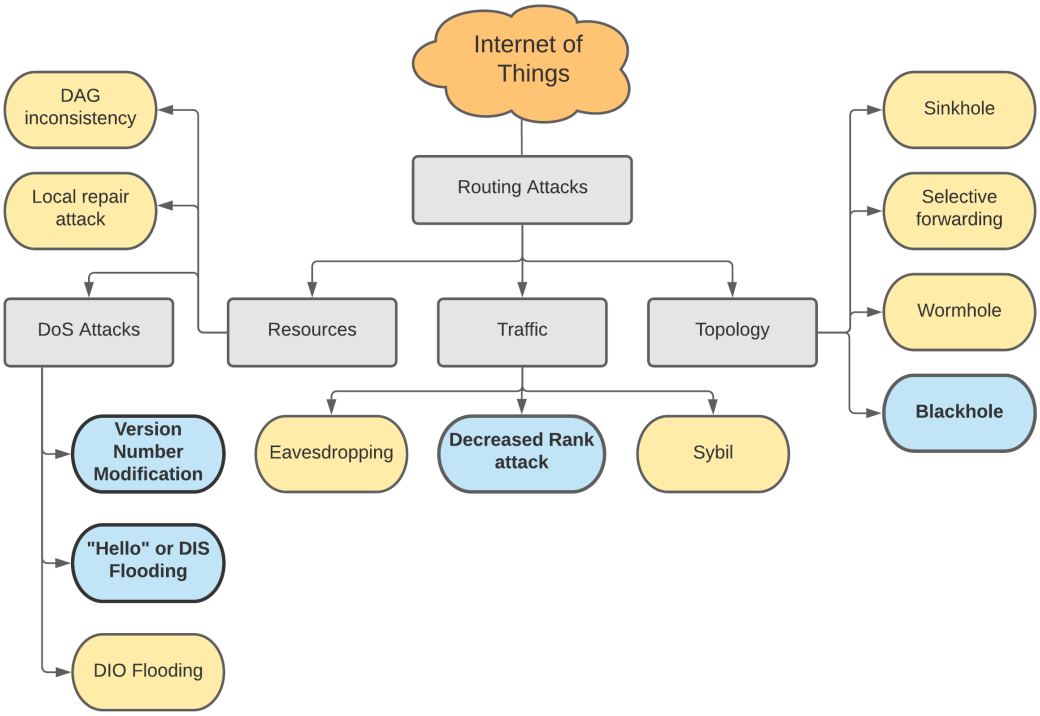


Fig. 2.2 IoT-based routing attacks overview. Blue colour: Attacks studied in this work, Yellow colour: Attacks not studied in this work.

Figure 2.2 presents an overview of current IoT-based routing attacks. The figure was created based on a combination of our own categorisation of routing attacks and the taxonomy found in [54]. The main three categories of routing attacks (Resources, Traffic and Topology) are adopted from [54] while the subcategories are modified to represent our main focus.

For instance, in the category of Resources-based attacks we consider RPL-specific flooding attacks as a sub-type of DoS attacks. The reason is that DoS attacks can be caused by either flooding the network with packets or by making a modification in the network which will cause nodes to send a large number of packets.

Attacks studied in this work are coloured in blue in Figure 2.2. As it is shown, routing attacks can be classified to resource-based, topology-based and traffic-based. Resource-based attacks such as Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks, pose a threat to IoT networks. Malicious actors attempt to flood a system with millions of packets using a large number of compromised devices. This may cause resource exhaustion and network disruption making the targeted system unavailable.

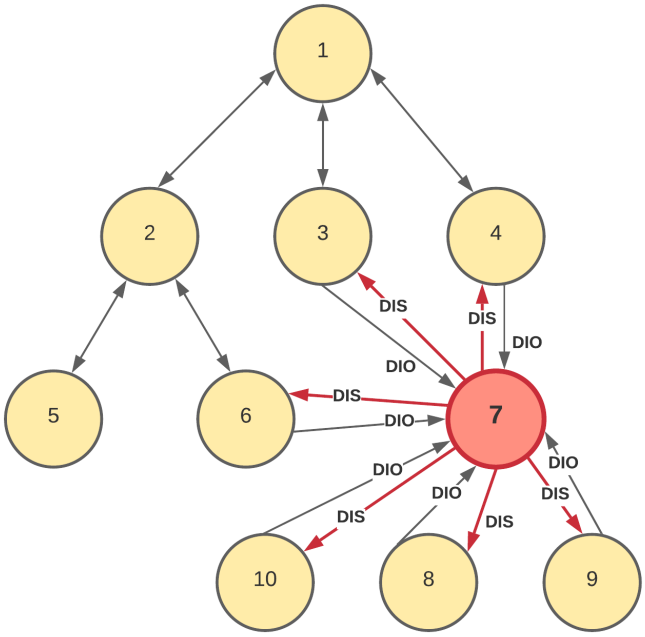


Fig. 2.3 DIS flooding attack. Node 7 (red node) represents DIS Attacker. Red arrows show transmitted DIS packets while DIO packets are sent from neighbours as reply to DIS requests.

In IoT networks, DoS attack may occur using various methods. For instance, version number modification attack [55] is an RPL-specific DoS attack in which the malicious node changes the DODAG version number of the received DIO packet. Then, it transmits to next hop the DIO packet with the malicious version number causing a reset of node's trickle timer. The version number is usually used by root node to control the so-called "global repairs" of the RPL network, and to ensure that the latest routes are available to nodes in the DODAG. When the version number changes, root node initiates global repair to reconstruct

the whole DODAG network. As a result, nodes are energy exhausted performing unnecessary computations.

Another example of RPL-specific DoS attack is "Hello" or DIS flooding attack [23]. Normally, an RPL DIS control packet is sent from nodes initially to join the network. However, DIS flooding attack occurs when a malicious node unicasts a batch of DIS requests to neighbouring RPL devices. Nodes receiving DIS packets, have to process and acknowledge the messages with DIO packets. Thus, node's battery energy is wasted by replying all incoming requests, network is congested, and nodes become unavailable. Figure 2.3 illustrates an example network where RPL protocol is used and DIS flooding attack is launched. Node 7 represents the malicious attacker which sends a large amount of DIS requests. Then, neighbouring devices reply to DIS requests with DIO control packets. As a result, attacker affects network and devices' availability.

In topology and traffic -based attacks, malicious actors target routing protocols and attempt to change or advertise fake routing information to attract traffic. Sinkhole, selective forwarding, rank and blackhole attacks belong to those two categories of routing attacks. Sinkhole attack happens when a node advertises fake routing metric to attract traffic from neighbouring nodes. This attack is used as the basis of launching other attacks. For instance, after sinkhole attack, the selective forwarding attack can be launched in which only chosen packets are forwarded. In this case, attacker will drop all data packets and forward only control messages.

A severe attack that is studied through this project is rank attack. According to [24], in rank attack a malicious node may intentionally advertise lower rank in order to attract neighbouring devices to select it as preferred parent. A parent node is needed in order to form the DODAG network and allow creation of routes reaching Border Router (BR). In cases where networks are small, the best parent is the BR itself. In other cases, metrics such as rank and ETX are used to select the best parent. If a malicious node manages to be chosen as the best parent of several nodes, it can attack the network affecting its topology, availability and integrity. As a result, the malicious node is a single point of failure of the network. An example of rank attack is presented in Figure 2.4. Node 6 (red node) represents the attacker which tries to attract neighbouring nodes. The network topology is formed using RPL protocol. When the attack is launched, attacker sends multiple DIO control packets to advertise a low rank value. Network under attack is shown on the top right figure, in which the attacker (node 6) sends DIO packets (red arrows) to neighbouring devices which are located inside the doted line rectangle. As can be seen from the illustration, a successful rank attack leads neighbouring devices to choosing the malicious node as their best parent. Thus, network topology is altered, and network performance is degraded.

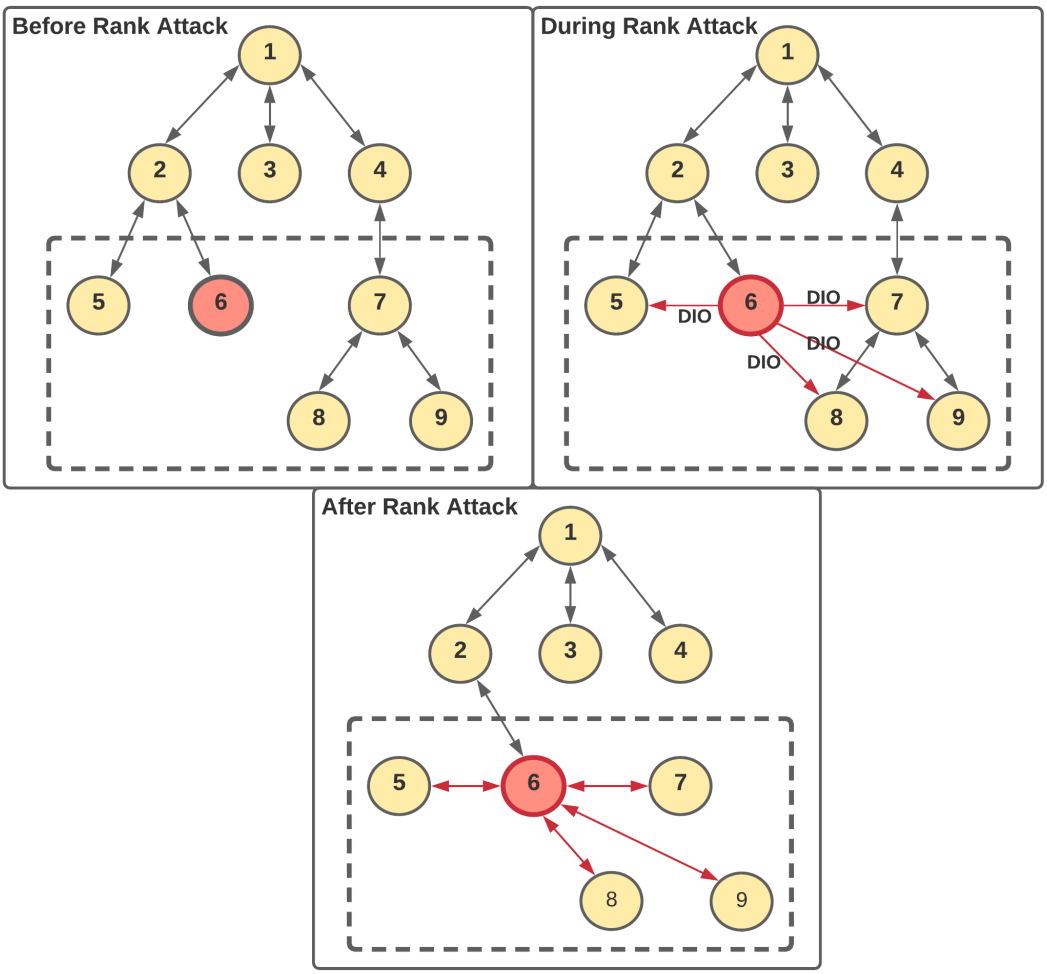


Fig. 2.4 Rank attack example. Node 6 (red node) represents the Attacker. Nodes inside the doted line rectangle may be affected by the malicious node attack. Red lines symbolise RPL DIO messages with a fake rank.

Another interesting routing attack is blackhole attack. In blackhole attack, the compromised node drops all incoming traffic. Therefore, no packets are forwarded from this node, and network disruption is achieved by the attacker. By dropping data packets, retransmission rate of child nodes is increased and an internal DoS attack occurs. This attack could isolate children nodes from the rest of the network, degrading the performance of the network. Figure 2.5 illustrates the result of blackhole attack in RPL network. Node 7 is the compromised node which accepts incoming traffic but does not forwards any packet. The arrows in the figure represent the packets sent while those with red crosses show the packets that attacker node discards. This attack can be combined with rank attack to attract more child nodes and achieve greater network disruption.

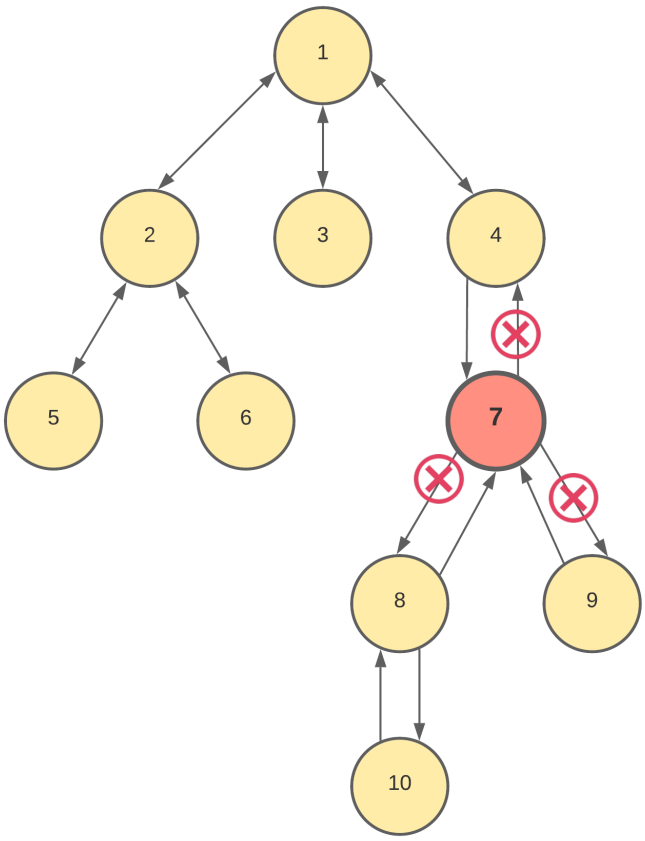


Fig. 2.5 Blackhole attack example. Node 7 (red node) represents the Attacker. Arrows show the communication between nodes. Red crosses symbolise blocked messages that the attacker is not forwarding.

2.6 Intrusion Detection for IoT

As already discussed in previous section, IoT networks are vulnerable to many attacks. Security attacks may cause severe damage to IoT devices and networks. Although security attacks exist in these networks, solutions focusing on securing the IoT ecosystem have already been developed. The analysis and discussion of existing security solutions for IoT networks are under the goals of **Contributions (a) and (b).**

In recent years, IDSes have attracted the attention of security researchers and practitioners for protecting IoT devices [56]. An IDS is a security technology that monitors networks or systems for malicious activity or policy violations. In the literature, three types of detection methods are typically distinguished, namely signature-based, anomaly-based, and specification-based [57]. Hybrid approaches, that combine two or more methods, are also gaining popularity.

A signature-based IDS may detect an attack/intrusion if the attack's signature is already stored in a database. These systems can detect known attacks very accurately and this is the reason why they are widely used in the industry. Anomaly-based detection tries to recognise malicious behaviour by distinguishing it from normal/expected behaviour. It needs the previous creation of profiles for defining the normal behaviour of users, hosts, or networks. Such profiles may be pre-loaded or built in real time by collecting data during the normal operation. Specification-based detection shares some similarities with anomaly-based detection in that it is able to recognise attackers by distinguishing it from the expected behaviour. However, the specification-based approach is based on manually creating the specification of legitimate behaviour rather than using a machine learning method as happens in anomaly-based detection. The disadvantage of this method it the time-consuming task of creating a detailed specification for a device.

2.7 Supervised Machine Learning

Machine Learning (ML) is a trending research field that helps computers complete tasks without human intervention. ML algorithms try to learn from existing data so that future actions will be done based on the experience gained. This section explores the supervised ML technique. ML-based detection methods are covered by **Contribution (c).** The use of ML algorithms within SRF-IDS will be explored in later chapters. Supervised, semi-supervised, and unsupervised learning are described [58]. These three categories are the most known learning procedures in ML field.

A well-known data processing technique is supervised learning. In this category, the model learns to map input variables (X) to an output (Y) using a function $f(Y = f(X))$. Both input and output datasets are labelled so that the model can learn the relation between the two. During training, input data is provided along with their correct output so that the algorithm learns the patterns. Learning procedure continues until a satisfactory level of performance is reached. After training, the supervised learning algorithm is tested to evaluate its performance. Specifically, unknown data is given as input to the model, and it tries to predict the output value based on the relationships learned from training procedure. Two main categories of supervised learning problems exist; regression and classification problems. Regression problems have as output variable a real value, while classification problems have a category as an output variable.

An example of classification algorithm that is also used in this work is the *2-class Decision Forest* [59]. The specific algorithm is a fast supervised ensemble learning model that is usually used for predicting two outcomes. The ensemble approach is the one that

numerous related models are created and merged in some way together to create a more generalised model instead of depending on a single model. In this way, better results can be obtained. Creating individual models and combining them together in an ensemble can be achieved by multiple ways. The Decision Forest implementation used in this project builds multiple decision trees and merges them together using *Voting* to produce more accurate and stable results.

Another classification algorithm that is used in this project is the *2-class Support Vector Machine (SVM)* [60]. According to Yang and Li [61], SVM is one of the best classification algorithms. It is a reliable algorithm which can achieve high accuracy on predicting the class of unseen data. It is based on the principle of structural risk minimisation. This principle aims in finding a hypothesis h for which one can be sure that the lowest error is observed while other methods are using the empirical risk principle which tries to improve the performance of the training set.

2.8 Related Work

The aim of this section is to discuss the latest related research work in the field of IoT attacks and defences. RPL security related studies are presented in the first subsection while IoT-based DoS attacks and mitigation techniques are described in the second subsection. Then, solutions for detecting attackers in IoT networks are discussed. These are split into two categories: IDS-based and trust-based solutions.

2.8.1 RPL Security

Even though IoT security is a field studied extensively, this section aims to summarise state-of-the-art research in securing the RPL protocol. Studying the impact of attacks in IoT, and mitigation methods are mostly related to **Contributions (a), (b) and (c).** Routing attacks such as blackhole, sinkhole, rank, version number, selective forwarding, DIS and DAO attacks are extensively studied in [22, 62, 63, 24]. Below, we briefly present recent works in the field of RPL security in IoT networks.

A detailed study about RPL security is presented by Verma et al. in [64]. Authors provide a comprehensive overview of RPL attacks, and categorise them based on their targets, including resources, topology, and traffic. Moreover, they evaluate existing RPL security solutions for several attacks, and compare their performance based on various metrics. Authors conclude that specific IDS-based and RPL-based mitigation techniques are still in the early stages. Thus, more research is needed to completely secure IoT networks.

Another extensive study of RPL security attacks and their impact on network performance is conducted by Wallgren et al. [65]. Authors implement well-known RPL routing attacks, including rank attack, in ContikiOS and demonstrate that the protocol is vulnerable to these attacks. A heartbeat protocol is also developed as a security mechanism to protect the IoT network against selective-forwarding attack. However, their solutions works well only if IPSec is used in the network because an attacker may choose to not filter ICMPv6 packets and thus, avoid being detected.

Ribera et al. [66] studied blackhole and greyhole attacks in RPL network. Authors analysed the impact of the attacks in Contiki-NG using metrics such as CPU, memory usage and TX/RX rates. Then, a novel UDP-based heartbeat detection technique is implemented and evaluated using the Cooja simulator. Results show high accuracy with low overhead in terms of CPU usage and battery consumption.

Boudouaia et al. [24] discuss the latest works about RPL-based rank attack. The rank property of the RPL protocol can be exploited and may cause poor network performance and waste of energy. Mitigation methods as well as the damage caused on network parameters are explained in their work. In addition, authors compare different attacks including rank attack using the Friedman test, and emphasise on the importance of rank attack.

Secure mode for RPL protocol is studied by Raoof et al. [67]. RPL operates in Unauthenticated mode by default. However, it can also operate in Pre-installed Secure Mode (PSM), using pre-installed symmetrical encryption keys to secure RPL control messages. Another mode is the Authenticated Security Mode (ASM) in which nodes utilise the pre-installed keys to join the network. Authors compare the performance of RPL protocol using the authenticated modes under blackhole, selective-forward, and neighbour attacks. Simulation results indicate that RPL can protect network from external attackers. Yet, secure modes require more memory and storage spaces than the unsecured mode, so it cannot work for all smart devices.

All in all, many studies investigate the security and impact of routing attacks to RPL protocol. The protocol is still vulnerable to many attacks including a combination of blackhole, rank and DIS flooding attacks. Moreover, existing solutions require more resources from IoT devices.

2.8.2 Denial of Service (DoS) Attacks and Defences

The latest works in the field of DoS attacks and prevention methods in IoT networks are described below. This subsection covers the objectives of **Contribution (a).**

Raoof et al. [23] present a comprehensive study of RPL attacks. They classify RPL attacks into those inherited by Wireless Sensor Networks (WSN) and into RPL-specific attacks. The

latest mitigation methods are also discussed and classified for RPL-based networks. Authors report that while there are some IoT-based IDSes, RPL-specific attacks, such as DIS flooding attacks, have no appropriate mitigation method to date. Moreover, the majority of studies do not present complete implementations for the mitigation of the various RPL-specific attacks.

Islam et al. [68] discuss the security issues for IoT devices in the healthcare domain. The authors analyse DoS attacks and focus on the *permanent DoS attack* in which the functionality of the device is permanently affected due to the execution of specific Linux commands. BrickerBot is the name of a bot that spreads the malicious code and transforms IoT devices into "bricks" [69]. Raymond et al. [70] examined MAC layer denial-of-sleep attacks on WSNs, where the power supply of a sensor is exploited. This type of attack can have a disastrous effect in sensor's lifetime.

A number of DoS prevention techniques for WSN have been suggested [70–72], which could potentially be used in the IoT security domain. A recent work by Muna et al. [73] focuses on detecting malicious activities in industrial IoT. Authors highlight the issue of zero-day and DoS attacks in critical infrastructure and suggest an anomaly-based IDS. Their solution uses deep learning models. It learns the TCP/IP traffic's normal behaviour and detects any abnormal behaviour automatically. It also aims to reduce false-positive rates. Their system was evaluated using two well-known datasets and then compared to other anomaly-based techniques.

Perakovic et al. [74] focus on DDoS attacks, which may negatively affect the quality of service (QoS) and the Service Level Agreement (SLA) between end user and service provider. The authors examined data from application layer attacks and they discovered that the most popular attack involves the "HTTP GET" method. They also report that nowadays, network and transport layers are exploited. Therefore, it is suggested that researchers should study detection methods across different layers.

Regarding the detection of DDoS attacks in IoT, Misra et al. [75] suggest using Learning Automata (LA) to build a prevention strategy by utilising the Service Oriented Architecture (SOA). This strategy determines the packet sampling rate from the environment. Specifically, it receives sampling rates as input from the environment and responds with the best action based on the given inputs. Each device has a DDoS prevention mechanism. During the detection phase, this mechanism counts the number of requests for each layer. If the requests in a layer exceed the maximum number of requests that this layer can handle, an alert is generated. Following the alert, devices record the IP addresses and flag the one with the most requests as the attacking device. Then, an Attacker Information Packet (AIP) is spread among the nodes, so that traffic is filtered and malicious packets are dropped. The best sampling rate

is defined by LA component to reduce delay and power consumption during the sampling stage. This process continues until packet interval falls below an acceptable threshold.

A multi-level DDoS detection framework was proposed by Yan et al. [76]. Authors use traditional mechanisms in different levels including fog computing, edge computing and cloud computing to defend against DDoS attacks. Bhuyan et al. [77] suggest an entropy-based DDoS detection. This approach is one of the most effective in the literature because of the low computing overhead.

Although many solutions have been proposed for detecting DoS/DDoS attacks in IoT, there have been very few real-world implementations and tests. Moreover, most of the works do not perform detailed simulations either. Anomaly-based IDSes require realistic training data from IoT attacks. The lack of detailed datasets makes this task difficult. Another limitation of current solutions is their energy efficiency. IDSes should not significantly affect sensor's battery performance. Last but not least, current solutions are usually designed to detect a single type of IoT-based attacks and lack generality.

2.8.3 RPL and Trust-based Mitigation Methods

Many works exist in the literature implementing mitigation methods for RPL attacks. This section discusses trust-based and RPL protocol-based solutions. This subsection covers the objectives of **Contribution (b).**

A comprehensive survey by Airehrour et al. [78] discusses the security of RPL protocol. Authors emphasize the lack of security mechanisms in the current IoT routing protocols and study the various methods proposed to achieve secure and sustainable routing among IoT devices. Among the various recommended security methods, embedding trust into the routing protocol could aid in protecting IoT networks from malicious attacks.

A Metric-based RPL Trustworthiness Scheme (MRTS) for addressing RPL attacks is introduced by Djedjig et al. [79]. In this scheme, every node evaluates the behaviour of its neighbouring nodes based on indirect suggestions and direct observations. Then, nodes will need to calculate the Extended RPL Node Trustworthiness (ERNT) for their neighbours. The node with higher trust value, more energy and better link quality is selected as preferred parent. MRTS uses the ERNT as a routing metric to form the network. Results show that it helps nodes to avoid malicious nodes. In addition, it has low energy consumption and high packet delivery ratio. However, nodes need to be in promiscuous mode to observe neighbour's behaviour.

Seyyed and Fereidoon [80] introduced a Dynamic and Comprehensive Trust Model for IoT (DCTM-IoT). Authors utilise several metrics such as packet forwarding indicator, ETX, energy and mobility in their scheme to calculate trust. DCTM-IoT supports direct and indirect

observations from neighbouring nodes to determine trust value. Then, a new OF is used in RPL to mitigate rank, sybil and blackhole attacks with mobility. Simulations were done using MRHOF, OF0 and the new OF in RPL. Results show good performance in small networks. In spite of good performance, DCTM-IoT needs to keep a record of the history of each node to calculate trust, resulting to not so lightweight solution for IoT devices.

Glissa et al. present a Secure-RPL (SRPL) protocol [81]. The aim of the proposed solution is to prevent malicious nodes from changing their rank multiple times, creating fake topologies. SRPL introduces a threshold to limit the number of rank changes. A hash chain authentication technique is also utilised to authenticate nodes when moving in the DODAG and modifying rank values. Authors suggest that SRPL can be used to detect other RPL attacks such as sinkhole, blackhole, selective forwarding attacks etc. For these attacks, they recommend to deploy anomaly-based algorithms to improve detection rate. Simulation results show that SRPL successfully protects the network from rank attacks. However, there is an increase in RPL control messages.

Belavagi and Muniyal [82] study the different RPL attacks using simulations, and try to identify multiple intrusions for varied network size by using an RPL-based IDS. Rank attack, selective forwarding, wormhole and Denial of Service (DoS) attack are identified by the algorithms discussed in their work. Cooja simulator with ContikiOS and ETX as objective function is used for simulating scenarios of multiple attacks. Evaluation results show that as the number of attackers increases, network performance is reduced. Therefore, a machine learning approach will be more suitable to identify various RPL routing attacks.

A Secure RPL Routing Protocol (SRPL-RP) for identifying, and isolating rank and version attacks is proposed by Almusaylim et al. [83]. Authors extend the works in [84, 85] in which mitigation methods identify both rank and version attacks. For monitoring purposes, all nodes have a monitoring table that store legitimate nodes including all their details. For rank detection, if a node's rank is greater than node's parent rank, it is considered malicious. Then, it is removed from monitoring table as a legitimate node and it is added in the blacklist table. Then, an alert is sent to all DODAG nodes to avoid the malicious node and isolate the attacker. For version attack, a similar threshold-based approach is followed. SRPL-RP is implemented and simulated in Cooja with ContikiOS using various topologies. SRPL-RP showed better detection and mitigation accuracy in comparison with other solutions.

Airehrour et al. [86] implemented the SecTrust-RPL protocol to overcome RPL routing attacks such as rank and sybil attacks. The suggested secure framework enhances RPL protocol by using a trust-based system to detect attackers. The concept is to have each node in the RPL network to be in promiscuous mode and sniff neighbour packets. Then, they compute direct and recommended trust values for each of its neighbours. Direct trust is calculated

based on the packet forwarding behaviour of the neighbour of the node. Recommended trust is given by another node and it is an estimation of how reliable and trustful is a node located at 2-hops or more. Penalty is added in cases where a node misbehaves during operation. Evaluation is based on Cooja simulator using ContikiOS, and test-bed experiments. Comparing results with standard RPL protocol show that their solution protects against rank and sybil attacks, is more reliable and consumes less energy than standard RPL protocol.

Perrey et al. [87] propose a Trust Anchor Interconnection Loop (TRAIL) scheme. It is a topology authentication scheme that is used to identify and isolate rank and version attackers in RPL networks. It aims to minimise control message overhead and power consumption. Sink node plays the role of trust anchor while the rest of the nodes validate ranks of their upward path to the root and drop invalid values. Validation is done using a round trip message. Evaluation on RIOT platform shows that attackers are isolated and only a small number of messages is exchanged.

Iuchi et al. [88] propose a secure parent node selection scheme. It aims to allow nodes choose legitimate nodes only as parents and avoid attackers. In this scheme, nodes firstly exclude the best candidate if more than one candidates exist. The reason for that is because attackers usually claim better rank than normal nodes to attract neighbours. Every node can decided if a node's rank is legitimate or not based on the average obtained ranks from neighbouring nodes. Therefore, nodes select parents that don't have too low rank, and avoid malicious nodes. Simulation results show that the proposed scheme avoids attackers and network operates better than standard RPL protocol.

Generally, many studies focus on embedding trust into RPL protocol to detect routing attackers. Yet, existing solutions have some limitations such as the requirement to have nodes in promiscuous mode, detect only one or two attacks, and require big storage capacity. A novel solution should be designed to address these issues.

2.8.4 IDS solutions for IoT

Over time IDSes have been considered by researchers as security measures for keeping IoT networks secured. However, traditional network detection algorithms have different requirements than those based on IoT. Thus, adapting traditional methods in IoT environments is a challenging task. Certain IoT characteristics such as the limited processing power of intelligent devices, different network structures and a variety of IoT device protocols, introduce new challenges which an IoT-based IDS must take into account [56]. Below, we present the latest IDS solutions for IoT. This subsection covers the objectives of **Contribution (a).**

A survey by Zarpelao et al. [56] discusses the latest intrusion detection approaches. It compares the various studies that propose IDS for IoT in terms of detection method, placement strategy, security threats, and validation strategy. Even though thera are many IDS solutions, the research field is still developing. Among the several issues, they suggest to investigate different detection methods as well as to detect more IoT attacks.

The first developed IDS that aims at protecting smart devices irrespective of specific IoT protocol or application is Kalis [89]. Kalis is a network-based, hybrid signature/anomaly-based, hybrid centralized/distributed, online IDS. The selected detection strategy depends on the specific features of the protected network. Furthermore, Kalis obtains knowledge from network-installed modules and tries to prevent intrusion by taking into account the current topology of the network and by conducting traffic analysis. Moreover, it can be extended to support new protocol standards and may improve detection performance by allowing knowledge sharing between the nodes. It is implemented on routers using the OpenWRT firmware [90]. Evaluation is done using 6 TelosB devices programmed in TinyOS [45]. Experimental results show that Kalis achieves 100% accuracy in detecting most of the attacks. Thus, it has better detection performance than Snort [91] and other traditional IDS solutions.

Strainer-based Intrusion Detection of Blackhole in 6LoWPAN for the Internet of Things (SIEWE) is proposed by Patel and Jinwala [92] to detect blackhole attacks in RPL networks. Blackhole attackers attract nodes by advertising a greater routing metric to neighbouring nodes so it is selected as the preferred parent. SIEWE uses this fact and adds the nodes IDs in a suspicious list. Then, smart devices which have suspicious nodes in their vicinity, analyze the behaviour of these nodes and inform BR about their findings. Thus, SIEWE includes only those nodes that have suspected nodes in their vicinity rather than requiring each node in the network to process and check suspicious nodes. Evaluation results on Cooja simulator indicate that SIEWE increases PDR of the network. A drawback of this approach is that only RSSI is utilised as parent selection metric. Attackers could send with same signal power in order to bypass detection mechanism.

Arshad et al. [93] propose the Collaborative Intrusion Detection for IoT (COLIDE) for detecting DoS and botnet attacks. Their host-based IDS enables collaboration between sensors and edge router to minimise energy consumption and improve detection performance. In order to achieve that, they introduce a device-level and edge-router components. Devices monitor traffic and run lightweight algorithms for detecting local attacks. Edge routers receive traffic from IoT devices and determine if a malicious node exists or not in the network. Moreover, smart devices need to join the network through one of the edge routers to enable packet monitoring. Implementation and evaluation of the framework is done in ContikiOS, showing effective and cost-effective collaboration. Although this solution seems

promising, evaluation does not show the number of edge routers needed and its performance in networks with more than 50 nodes.

A Le et al. [94] study the impact of RPL routing attacks in Cooja using ContikiOS. The four studied attacks are rank, sinkhole, local repair, neighbour and DIS flooding attacks. Results show that internal threats can degrade RPL network performance in many cases, by reducing packet delivery ratio, increasing end-to-end delay or creating more control overhead. Authors designed a mitigation method in [95] based on the results of their previous work [94]. A specification-based IDS is implemented and tested. The IDS uses a finite-state machine to define the behaviour of RPL protocol and detect malicious activity. A benefit of using this approach is that no additional control overhead is introduced in the network.

A Sink-Based Intrusion Detection System (SBIDS) is presented by Shafique et al. [85] for detecting rank attacks in RPL networks. Authors use a rule-based approach to compare node's current rank with the node's parent rank as well as the minimum rank of their siblings. If a node advertises a greater rank than its parent, it is considered malicious. Evaluation of the scheme showed that SBIDS achieves good detection performance of rank attacks.

SVELTE IDS is another interesting work in the field [96]. This is a anomaly- and signature-based IDS, developed to prevent RPL-based routing attacks in IoT devices [4]. Some of the attacks considered include selective forwarding, sinkhole attack and spoofed or altered information. Regarding node's placement approach, SVELTE has a centralised module, called 6LoWPAN Border Router (6BR), which carries out heavy calculations, and a number of resource-restricted modules monitoring network devices. The 6BR consists of three components. The first one is the 6LoWPAN Mapper, which gathers information from sensors to regenerate the network. The second component is the detection system, which uses the obtained information to detect possible intrusions. The third component is a mini firewall that prevents the entry of malicious traffic into the network. In IoT devices, the first and third components are integrated. Authors develop and evaluate SVELTE in ContikiOS. Results indicate that SVELTE has a high True Positive (TP) rate but also some false alarms during simulations. However, SVELTE shows high traffic overhead due to the reconstruction of network initiated by the router.

Even though a few IoT-based IDSes have recently been developed, current solutions have certain constraints. Kalis, for example, requires deployment of specific detection modules per attack type. This could create a complex network resulting in poor detection performance. Additionally, evaluation was done only with 6 real world devices. No results are available for larger networks. SVELTE has also some limitations as it is a host-based IDS and thus, sensor's software must be modified. This, however, would be very challenging for larger networks, which is a typical case in many IoT application domains. Another major issue is

Table 2.3 Scientific works that study RPL attacks

Study	Implementation Method	Detected Attacks
SRPL-RP [83]	RPL integrated with threshold-based detection	Rank and version attacks
Secure-RPL (SRPL) [81]	RPL integrated with threshold-based detection	Rank attack
SecTrust-RPL [86]	RPL integrated with trust scheme	Rank and sybil attacks
Trust Anchor Interconnection Loop (TRAIL) [87]	RPL integrated with rank authentication scheme	Rank and version attacks
Secure Parent Node Selection Scheme [88]	RPL with integrated threshold-based detection	Rank attack
Kalis [89]	Signature/Anomaly-based IDS	DoS attacks
SVELTE [96]	Signature/Anomaly-based IDS	Sinkhole, and selective forwarding
Sink-Based Intrusion Detection Systems (SBIDS) [85]	Rule-based IDS	Rank attack
Specification-based IDS [95]	Finite-state machine IDS	Rank, sinkhole, local repair, neighbour and DIS attacks
COLIDE [93]	Host-based IDS	DoS and botnet attacks
SIEWE [92]	Anomaly-based IDS	Blackhole
RPL-based IDS [82]	Threshold-based IDS	Rank, selective forwarding, wormhole and DoS attacks
Our SRF-IoT	RPL with trust scheme and external IDS collaboration	Rank, blackhole, and DIS attacks

Table 2.4 Comparison of SRF-IoT features versus related works.

	Rank	Blackhole	Version number	DIS flooding	Combined attacks	IDS deployment [1]	Threshold-based or Anomaly-based [2]	Trust-based	Total nodes
SRPL-RP [83]	✓	-	✓	-	✓	-	T	-	20
SRPL [81]	✓	✓	-	-	✓	-	T	-	22
SecTrust-RPL [86]	✓	-	-	-	-	-	A	✓	30
TRAIL [87]	✓	-	✓	-	-	-	T	-	20
Secure Parent Node Selection Scheme [88]	✓	-	-	-	-	-	T	-	32
Kalis [89]	-	-	-	✓	-	E	T	-	6
SVELTE [96]	-	-	-	-	-	E	A	-	8/16/32
SBIDS [85]	✓	-	-	-	-	E	T	-	20
COLIDE [93]	-	-	-	✓	-	E	A	-	7
SIEWE [92]	-	✓	-	-	-	E	A	-	Up to 50
RPL-based IDS [82]	✓	-	-	✓	-	E	T	-	10/40/100
SRF-IoT	✓	✓	✓	✓	✓	S	A	✓	36/42/144

[1] Type of IDS deployment. **S: S**eparate RPL Instance, **E: E**mbedded into existing devices.
[2] Detection method. **T**: Threshold-based, **A**: Anomaly-based.

that SVELTE has high false detection rate. This was proved by Matsunaga et al. [97] who proposed a scheme to reduce false detection rate. However, further experiments are needed to ensure that the solution is robust and scalable.

In conclusion, most of the suggested solutions are either IDS-based or protocol-based utilising the features of RPL protocol. A new technologically enhanced method should consider combining these two fields to achieve higher and better detection performance of multiple attacks. Table 2.3 depicts scientific works that study and propose a mitigation method for RPL attacks. As it is depicted, most studies implement an IDS solution or integrate the IDS into RPL protocol for detecting various attacks. In this project, a collaboration between RPL-based devices and IDS is achieved. Our aim is to identify and avoid routing as well as other types of attacks by embedding trust concept into parent selection algorithm of RPL protocol. This algorithm will allow nodes to securely select a parent. The trust-based method works along with an external IDS that provides the monitored devices with useful metrics. The aforementioned limitations have been taken into account during the design of the proposed SRF-IoT scheme.

Table 2.4 compares existing solutions with the proposed SRF-IoT framework. As it is shown, the different features supported by the related studies are compared. Specifically, SRF-IoT framework is able to detect all the four routing attacks - rank, blackhole, version number modification and DIS flooding - as well as a combination of them. Looking at the IDS deployment feature, we refer with **S** to an IDS deployed in a Separate RPL Instance and with **E** the IDS which is Embedded into existing devices. SRF-IoT framework is the only solution that IDS is deployed in a separate RPL Instance. Another important feature is the type of detection method. Usually, IDS systems detect attacks using a threshold-based (refer with **T**) or an anomaly-based method (refer with **A**). Our framework uses a combination of an anomaly-based IDS and a trust-based system to detect and isolate attackers. The total number of deployed nodes during evaluation is another significant parameter. The maximum number of simulated nodes in all the existing related works is 100 nodes. This number does not include any additional IDS nodes. In the evaluation of SRF-IoT, a maximum of 144 nodes are deployed excluding SRF-IDS detectors. It is the highest number of deployed nodes in a network simulation.

2.9 Chapter Summary

The current chapter explains some useful concepts that help the reader understand the rest of the thesis. The IoT reference model, the available OSes and simulators for IoT researchers are presented. Then, routing protocols and current IoT-based attacks are explained in details.

Some existing security solutions for IoT networks are discussed as well as machine learning concepts are introduced to the reader. Current attacks and defence methods in IoT networks are also presented. Existing suggested mitigation methods are divided into trust-based and IDS-based. The main goal of this chapter is to give the reader some background information for IoT networks, an overview of the current problems in those networks, and present the state-of-the-art mitigation methods.

Chapter 3

IoT Routing Protocol Attacks

As the basic concepts and latest works are already discussed, the next step is to study the impact of IoT-based attacks in the network. This chapter presents the design and implementation of IoT-based attacks in RPL protocol. Simulation results are also explained. Then, DIS flooding attack is extensively studied in larger networks as it causes more impact. Evaluation steps are described and a method of defence is defined.

3.1 Problem Statement

Nowadays, an increasing number of malicious actors target IoT devices to carry out large-scale cyber-attacks. The reason for that is the limitations that IoT platforms have. Limited processing power, capacity, and energy are some of the IoT constraints. Unfortunately, in many cases the traditional solutions do not consider these limitations, and cannot protect IoT networks effectively, leaving the devices unsecured. Apart from that, newly developed protocols for IoT platforms introduce new vulnerabilities which can be exploited by malicious actors. As already discussed in previous chapters, these attacks may disrupt IoT network availability, confidentiality and integrity. In cases where smart networks are operating in critical infrastructures such as Smart Grids, an attack may have severe impact in the infrastructure of the whole country. Therefore, solutions should be suggested to detect attacks and prevent any future attacks in IoT devices.

Studying the impact of IoT attacks is the first step before designing and implementing a defence method. As the focus of this work is on network layer, related attacks usually target routing protocols. One of the most popular routing protocols for IoT devices is RPL. Thus, a version of the RPL protocol is used in ContikiOS, an OS of real-world platforms, to implement and check the actual impacts of these attacks.

3.2 Battery Drain DoS Attacks

In this section, the details about designing and implementing RPL-specific DoS attacks are presented. The goal is to study the impact of several attacks on each device and on the whole network. After that, by launching attacks using various configuration parameters and intensities, different detection methods can be implemented, tested, and enhanced. Implementing and simulating RPL-based DoS attacks is a requirement to study the causes of the attack to the network, and to achieve **Contributions (a) and (b).**

3.2.1 Design

As discussed in Chapter 2, RPL organises nodes along a DODAG [98]. The root node initiates the creation of graphs by regularly generating DIO messages, which are advertised via link-local multicasts. The "Hello" or DIS flooding attack in RPL occurs when a large number of DIS messages are transmitted by malicious node to other nodes. This causes the recipient nodes to reply by sending DIO messages. Consequently, network floods with packets and node's batteries are drained.

Similarly, in version number modification attack [55], the malicious node changes the DODAG version number before forwarding the received DIO messages to the next hop. Nodes receiving a malicious DIO message with the modified version number reset their trickle timer, store the new version number in their memory and advertise it to their neighbours via DIO messages. Note that root uses the version number to control the so-called "global repairs" of the RPL network and to ensure that the latest routes are available to nodes in the DODAG. Global repair is the repair mechanism that is initiated by the root to rebuild the network. During this process, it increases the version number of RPL DODAG and the whole DODAG is reconstructed. This method ensures a loop-free and optimised tree based on the used objective function. Still, this makes the IoT nodes to perform useless computations and waste their energy. Thus, modifying the version number will cause unnecessary global rebuilds of the DODAG, create loops in the topology as well as exhaust the nodes.

3.2.2 Implementation

Two IoT-specific DoS attacks have been implemented in ContikiOS, namely version number modification and DIS flooding (or "Hello" flooding) attacks. These attacks exploit the RPL protocol's features and affect the power consumption of IoT devices. Cooja provides an implementation of the RPL protocol, called ContikiRPL [35].

Algorithm 1 shows the DIS flooding attack implementation in ContikiRPL. Specifically, the malicious node launches the attack by sending a batch of DIS packets to its neighbours. This is implemented as a *While* loop which calls the *dis_output()*. The *total_packets* variable defines the number of iterations. In this scenario, its value is 10. The *dis_output()* function, called inside the loop, is responsible for building and transmitting DIS packets. Therefore, the *While* loop calls the function 10 times to transmit 10 DIS packets. In this scenario, sending just 10 DIS packets is enough to study and understand the effects of DIS flooding attack. In future scenarios, this number is increased.

Algorithm 1 "Hello" or DIS flooding attack

```
1: int i ← 0;
2: while i < total_packets do
3:     i ← i + 1;
4:     dis_output(NULL);
5: end while
```

Version number modification is also implemented in ContikiRPL. Algorithm 2 depicts the code used to develop the attack. As shown, the function *dio_output()* which is responsible for creating and sending DIO packets, is modified to increase the version number. The malicious code is added in the function at a point where packet payload is prepared to be loaded into packet buffer. Lines 2-6 is the initialisation of variables used in the specific function. Then, in line 7 a buffer is created to insert packet payload. In line 10, DAG version is incremented by one, and in line 11 the value is inserted into packet buffer. The version value is chosen to be incremented by one because it only matters to have a different version number, and not a specific value. Neighbouring nodes will receive a DIO packet with different version number and this, will trigger global repair.

3.2.3 Scenarios and Topologies

Below are two scenarios, simulated in Cooja, which show the effects of the DoS attacks mentioned above. Applications of the UDP client-server model are used on top of each node. Seven Tmote Sky nodes [99] were simulated running ContikiOS. The network, depicted in Fig. 3.1, consists of one server (root node with ID 1) and six client nodes with IDs from 2 to 7.

In the first scenario, no compromised nodes exist. Each node is configured to send messages to the server at intervals. These messages contain various information about the sending node, such as its battery indicator and temperature. In the second scenario, node 7 is malicious/compromised and performs DoS attacks. Specially, node 7 has been configured to

Algorithm 2 Version number modification attack

1: **Function** $void\ dio\_output(rpl\_instance * instance)$: {Code added at the point where payload is loaded into packet buffer}
2: $unsigned\ char\ *buffer$;
3: $int\ pos$;
4: $int\ is\_root$;
5: $rpl\_dag\ *dag \leftarrow instance.current\_dag$;
6: $pos \leftarrow 0$;
7: $buffer \leftarrow UIP\_ICMP\_PAYLOAD$;
8: $buffer[pos] \leftarrow instance.instance\_id$;
9: $pos \leftarrow pos + 1$;
10: $dag.version \leftarrow dag.version + 1$;
11: $buffer[pos] \leftarrow dag.version$;
12: $pos \leftarrow pos + 1$;

transmit a big number of DIS messages to its neighbours. In addition, it changes the DODAG version number so that global repairs are initiated.

3.2.4 Simulation Parameters

Cooja simulator [39] was used for testing and experimentation, which is becoming increasingly popular among IoT researchers. It is also particularly suitable for experiments in real-world, since the developed applications can be directly uploaded to real hardware. Cooja can be used to simulate the behaviour of ContikiOS [33].

The node types and configuration used for each node is shown in Table 3.1. Server is the receiver of all messages exchanged in the network. As a result, it is always powered on. The Radio Duty Cycle (RDC) driver is responsible for saving as much as possible power for the device. ContikiOS implements several RDC drivers, but the server uses NullRDC which does not save power. The Medium Access Control (MAC) driver is responsible for reliably transferring packets at the radio medium. If any collisions occur, it re-transmits the packets until they are delivered. All nodes in our scenarios use the Carrier Sense Multiple Access (CSMA) driver at the MAC layer to guarantee packet delivery. In contrast with benign and malicious nodes, the server does not transmit DIS messages.

Benign nodes are sending data to the server. They are configured to send a DIS message every 60 seconds until they successfully join the network. The malicious node broadcasts 80 DIS messages every second, thus launching the “Hello” or DIS flooding attack. Benign and malicious nodes in this scenario are configured to use NullRDC as RDC driver and CSMA as MAC driver. This setup will keep the devices always on.

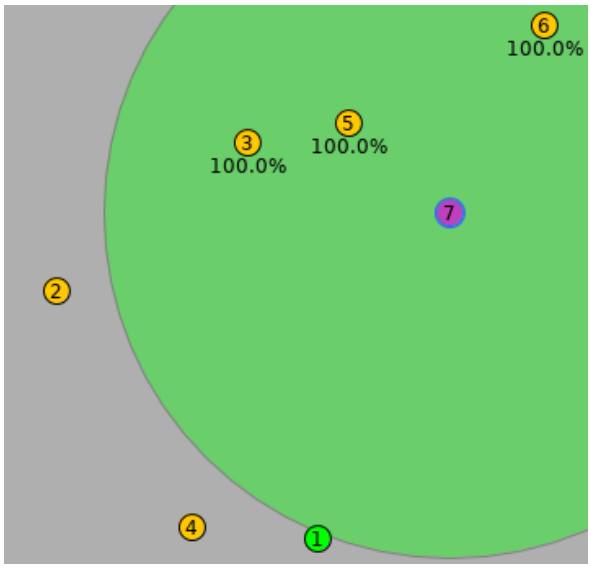


Fig. 3.1 Network topology of Cooja simulations

Various parameters can be configured in Cooja simulator. Firstly, the simulation time in our experiments for each scenario is 10 minutes. Another feature provided by simulator is mote startup delay. This parameters allow nodes to boot at random time and not all at the same time. A startup delay of 1000 ms is configured by default in Cooja. An important parameter is random seed. This value is chosen to be autogenerated by Cooja simulator during the initialisation phase. Seed number affects nodes' behaviour such as packet transmission times.

Regarding radio medium, there are four propagation models in Cooja; constant loss Unit Disk Graph Medium (UDGM), distance loss UDGM, Directed Graph Radio Medium (DGRM), and Multipath Ray-tracer Medium (MRM), namely. Constant loss UDGM assumes an optimal transmission range disk in which nodes inside the transmission disk capture packets while those outside of it do not. Distance loss UDGM is the constant loss UDGM

Table 3.1 Node types and configuration

Node type	**Description**	**Radio Duty Cycle driver**	**DIS sending interval**
Server	Acts as a sink node. Receives messages without doing any processing or sending replies.	NullRDC	N/A
Benign node	Uses RPL to create a mesh network and sends data periodically to server.	ContikiMAC or NullRDC	60 seconds
Malicious node	Uses RPL to broadcast DIS control messages to neighbours (DIS attack).	ContikiMAC or NullRDC	1 second

version including radio interference. In that model, packets are sent with a probability of "Success ratio TX" and received with a probability of "Success ratio RX". The third model is the Directed Graph Radio Medium (DGRM) in which user specifies RSSI and RX success ratios for each link between motes. The last and most advanced model, MRM, calculates receiver power using ray tracing methods such as the Friis formula. It also supports calculation of diffractions, refractions and reflections along radio links.

Having in mind the above propagation models, we chose to use distance loss UDGM in our Cooja simulations. The reason is that data loss caused by interference must be included in our experiments because we will understand better the impact of attacks when interference exist in simulation environment. Therefore, more realistic results will be exported. For the experiments, simulator's default transmission and interference range between nodes is 50m. In our scenarios, we place nodes within the range of all other nodes.

3.2.5 Results

In the first scenario, the network topology is formed as shown in Fig. 3.2. The numbers displayed on each link indicate the ETX. That is the expected number of transmissions that a node needs to make to a destination in order to deliver a message successfully. For instance, the ETX value of node 4 next to the server (node 1) is 8. In Fig. 3.2 we also note that node 7's messages must be transmitted via nodes 3, 2 and 4 to get to the server. Note that node 7 is not malicious and runs the same code as all other nodes in this scenario. In Figure 3.3 the power consumption of each node is shown. Measurements have been gathered using the PowerTracker tool in Cooja. As expected, all nodes are almost always on (average 99.87% of the time) and have very low values of Radio TX and Radio RX. This is normal for small-sized networks.

In the second scenario, nodes use the same RDC and MAC driver configuration as before. Node 7 has, however, been modified to transmit 80 DIS messages and increase the DODAG version number before transmitting the received DIO messages to the server. Changing the

Table 3.2 Simulation parameters

Parameter	**Value**
Radio medium	Unit Disk Graph Medium (UDGM): Distance Loss
Mote startup delay	1000 ms
Random seed	Autogenerated
Range	Transmission range: 50 m Interference range: 50 m
Duration	10 min

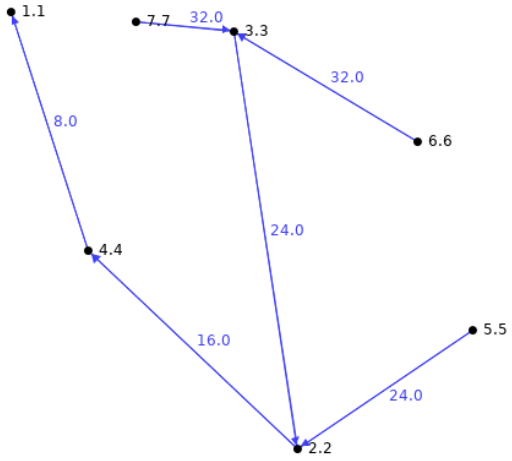


Fig. 3.2 Scenario 1 (normal operation): Network topology

version number leads to global repair and the formation of two different DODAGs. Every few minutes, global repair is triggered. As a result, the routes change quickly. The topology of the network is therefore not stable and some nodes may be disconnected from the server or other nodes. There is one such situation in Fig. 3.4, where at that particular moment nodes 5 and 6 do not have a route to the server. The impact of the attack is demonstrated in Fig. 3.5, which shows the measurements of power consumption. In adjacent nodes 3, 5, and 6 the attack caused high Radio RX and for node 7 high Radio TX. As a result, both malicious/compromised and neighbouring nodes are depleted with energy.

Mote	Radio on (%)	Radio TX (%)	Radio RX (%)
Sky 1	99.91%	0.01%	0.06%
Sky 2	99.79%	0.05%	0.11%
Sky 3	99.92%	0.03%	0.10%
Sky 4	99.86%	0.06%	0.06%
Sky 5	99.91%	0.02%	0.11%
Sky 6	99.79%	0.02%	0.07%
Sky 7	99.88%	0.02%	0.07%
AVERAGE	99.87%	0.03%	0.08%

Fig. 3.3 Scenario 1 (normal operation): Power consumption measurements

In the previous scenarios, nodes used the same RDC and MAC drivers. However, using a different RDC driver may produce different results. In ContikiOS, ContikiMAC is another option for RDC driver. For this reason, the two scenarios were repeated using the ContikiMAC RDC driver in benign and malicious nodes while keeping the same MAC driver. Starting with the normal scenario, the nodes' power consumption is shown in Fig. 3.6. As it is expected, ContikiMAC enables sleep mode and this is clearly shown with the very low percentage of the Radio On for all nodes except for the server. In addition, Radio TX is on average 0.45% which means that nodes sleep most of the time and send very few packets in the

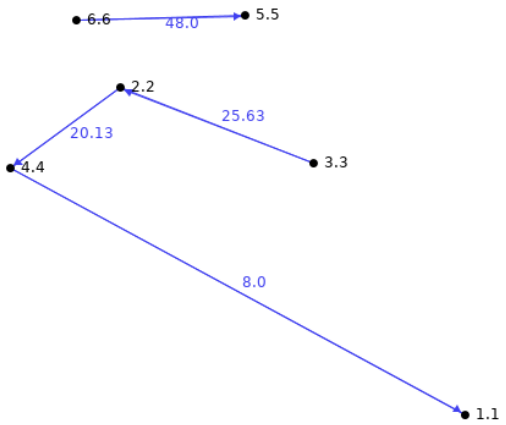


Fig. 3.4 Scenario 2 (attack): Network topology

network. Radio RX is even lower on average than Radio TX because packets have mostly as destination the server. The corresponding network topology is shown in Fig. 3.7. As it can be seen, all nodes communicate with the server by using their next hop.

Mote	Radio on (%)	Radio TX (%)	Radio RX (%)
Sky 1	99.88%	0.04%	0.12%
Sky 2	99.89%	0.11%	0.36%
Sky 3	99.80%	0.12%	2.12%
Sky 4	99.93%	0.14%	0.14%
Sky 5	99.80%	0.15%	2.09%
Sky 6	99.83%	0.09%	2.04%
Sky 7	99.91%	1.82%	0.32%
AVERAGE	99.86%	0.35%	1.03%

Fig. 3.5 Scenario 2 (attack): Power consumption measurements

Mote	Radio on (%)	Radio TX (%)	Radio RX (%)
Sky 1	99.82%	0.11%	0.18%
Sky 2	1.93%	0.87%	0.05%
Sky 3	1.94%	0.87%	0.06%
Sky 4	1.16%	0.24%	0.06%
Sky 5	1.25%	0.35%	0.07%
Sky 6	1.20%	0.30%	0.04%
Sky 7	1.29%	0.39%	0.06%
AVERAGE	15.34%	0.45%	0.07%

Fig. 3.6 Scenario 1 using ContikiMAC : Power consumption measurements

Using the same nodes' configuration, the second scenario with a malicious node was repeated. In this case, the power consumption of nodes is affected by the malicious node as shown in Fig. 3.8. Although nodes should be sleeping most of the time, they are ON for 50% of the time, including the server. The difference with the normal scenario is about 35%, which is significant. The reason for this behaviour is because the malicious node 7 broadcasts

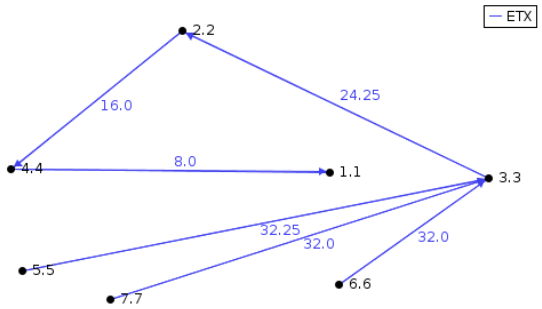


Fig. 3.7 Scenario 1 using ContikiMAC : Network Topology

DIS messages requiring a DIO reply from its neighbours. This is also the reason why nodes 3, 5 and 6 have the highest Radio RX percentage in comparison with other nodes. These nodes are closer to node 7 and are more affected than the others. Furthermore, node 7 transmits all the time and thus it has the highest percentage of Radio TX. Looking at the network topology in Fig. 3.9, we can see that some links have unusually high ETX values. The reason for that are the global repairs, which are initiated and assign different ETX values to links. Nodes located near the malicious node have worse ETX value in their links in comparison with other nodes. Moreover, the malicious node is not shown in the presented network topology because it never joins the DODAG and, therefore, no information is sent to the server.

Mote	Radio on (%)	Radio TX (%)	Radio RX (%)
Sky 1	100.00%	0.07%	0.24%
Sky 2	3.78%	2.08%	0.19%
Sky 3	59.37%	2.20%	35.70%
Sky 4	4.28%	2.67%	0.03%
Sky 5	59.60%	1.99%	36.06%
Sky 6	60.17%	2.42%	36.08%
Sky 7	63.98%	47.33%	0.94%
AVERAGE	50.17%	8.39%	15.61%

Fig. 3.8 Scenario 2 using ContikiMAC : Power consumption measurements

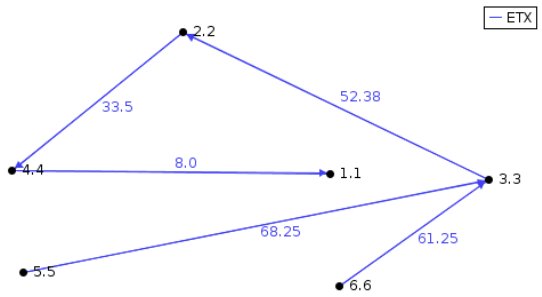


Fig. 3.9 Scenario 2 using ContikiMAC : Network Topology

3.3 Flooding Attack in Large Networks

The current section provides an extensive study of DIS flooding attack using different parameters and calculating more metrics to achieve **Contributions (a) and (b).** Specifically, further simulations were made for "Hello" or DIS flooding attack. Scenarios with larger networks were simulated to extract useful results. Later, exported results are analysed and SRF-IDS algorithms are implemented to combat the IoT-specific attack.

3.3.1 Design

The goal of these simulations is to examine RPL-specific DoS attacks and export some useful metrics that will help us design and implement mitigation algorithms. To this end, a detector or sniffer node was designed to capture traffic exchanged in the network and store some useful metrics for further investigation.

Detector operates in promiscuous mode to gather packets and calculates metrics every 5 seconds. Once a UDP packet or RPL control packet such as DIS, DIO, or DAO is captured, detector stores node IP, number of packets captured, packet type, and packet interval. Then, every 5 seconds, collected data are aggregated and printed in the console by detector, to allow further processing. Data aggregation was configured at 5 seconds interval to allow us easily inspect the differences in measurements before and after the launch of attack.

"Hello" or DIS flooding attack is configured as in the previous section. In this scenario, ten DIS packets are transmitted by malicious nodes per second. Storing information in detector such as packet interval, and the number of packets captured, will allow us to export useful results. For example, when a network is under "Hello" or DIS flooding attack, we expect a large number of packets to be exchanged, and high packet interval from malicious nodes. Therefore, statistics provided by detector will help us define the behaviour that nodes should have in normal and malicious scenarios.

3.3.2 Implementation

Attack implementation is discussed in this section. As "Hello" or DIS flooding attack is studied in this section, implementation of the attack is the same as illustrated in the previous Subsection 3.2.2. Malicious nodes are configured to send 10 DIS packets per second. The development of detector required a structure to store important information of neighbouring nodes. The structure implemented in detector for storing neighbour's metrics is depicted in Algorithm 3. As shown, detector creates a structure in memory for each node to store the IP address of neighbour node, a counter for DIS packets received, a counter for DIO/DAO/UDP

packets received, the type of last message type, packet intervals and timestamp of last received packet of neighbour node. This information is kept until the end of simulation time.

Algorithm 3 Neighbour structure stored in detector

```
1: struct neighbour{
2:     int address;
3:     int counterMsg;
4:     int counterDIS;
5:     int msgtype;
6:     unsigned long intervals;
7:     unsigned long timestamp;
8: };
```

After collecting the measurements, detector has to calculate metrics using the Algorithm 4. The function *Calculate_Metrics()* is called by detector to process metrics. Specifically, it starts by iterating over the list of neighbours and checks the statistics of each node. Root node is excluded from calculations as it only receives packets. Each metric, such as packet interval and counters for DIS and other messages, are updated with the latest value. Several checks are made to ensure that values are correctly collected and aggregated. After aggregating metrics, if the aggregated value of metric is above zero, the average value of it is calculated. At the end of the algorithm, the average value of each metric is printed in the console. Later, results will be analysed based on these values.

3.3.3 Scenarios and Topologies

Below, we demonstrate two main scenarios, *normal* and *malicious*, simulated in Cooja. One of the goals is to examine the behaviour of sensors in an environment without a malicious/compromised node. In addition, this behaviour is compared to a scenario where one or more nodes are compromised. The obtained results have been used to design and implement a *detector* for identifying malicious nodes and stopping DoS attacks. The application used in IoT nodes/sensors is based on the UDP client-server model. The hardware used for each node is Zolertia Z1 [100] running ContikiOS. In the normal scenario, each node regularly sends messages to the root node. These messages contain various information about the sending node, such as node ID and battery indicator. In the malicious scenario, one or more nodes are malicious/compromised and have been modified to launch “Hello” or DIS flooding attack in which a large number of DIS messages are sent to their neighbours. This causes IoT nodes to perform unnecessary computations and consume energy.

Algorithm 4 Calculation of metrics

```
1: Input: Neighbours array
2: Output: Average value of packet interval, DIS and other RPL messages.
3: Function Calculate_Metrics() :
4: for each node in Neighbours do
5:    if node.address ≠ 1 then
6:       if node.intervals ≠ 999 then
7:          total_interval ← node.intervals + total_interval;
8:          c_int ← c_int + 1;
9:       end if
10:      if node.counterDIS ≠ 0 then
11:         total_dis ← node.counterDIS + total_dis;
12:         c_dis ← c_dis + 1;
13:      end if
14:      if node.counterMsg ≠ 0 then
15:         total_othermsg ← node.counterMsg + total_othermsg;
16:         c_othermsg ← c_othermsg + 1;
17:      end if
18:   end if
19: end for
20: if c_int ≠ 0 then
21:    AVG_TIME = total_interval/c_int;
22: end if
23: if c_dis ≠ 0 then
24:    AVG_DIS = total_dis/c_dis;
25: end if
26: if c_othermsg ≠ 0 then
27:    AVG_MSG = total_othermsg/c_othermsg;
28: end if
29: print (AVG_TIME, AVG_DIS, AVG_MSG)
```

Each scenario uses a variable number of nodes. Table 3.1 describes most of node types used in the experiments, while Table 3.3 below describes the configuration of the additional Detector node. This node is configured to sniff network traffic from neighbouring nodes to detect malicious nodes. Moreover, it is able to store information about packets exchanged and packet intervals of each neighbour. As RDC driver, the NullRDC driver was used to have always ON the node and monitor the network.

The number of nodes used in each scenario is depicted in Table 3.4. As it is indicated, malicious motes are used only in the malicious scenario, while other nodes are used in both scenarios. Specifically, in the normal scenario, we have one server, one detector, and a varying number (2 to 8) of benign nodes. In the malicious scenario, we have one server, one detector, and a varying number of benign (2 to 7) and malicious (1 to 6) nodes.

Concerning benign motes, three different cases were created. In the first case, one malicious mote was used in each scenario while the number of benign motes varies. In the second case, 3 benign motes were deployed while malicious motes range from 1 to 6. In the third case, benign motes are fixed to 4 while again malicious motes vary from 1 to 6. The total number of nodes for each scenario is shown in the last column of the table. Overall, we performed 7 and 6 simulations for the normal and malicious scenarios, respectively. Each simulation was repeated 25 times. In this way, a large sample is gathered for analysis.

The topology used in simulations is depicted in Figure 3.10. All scenarios use mesh topology. The normal scenario in the shown topology consists of three client nodes with IDs from 2 to 4, one detector with ID 5, and one server/root node with ID 1. A simple malicious scenario is very similar to the normal one, with the exception that one benign node is converted into a malicious one. Thus, two benign and one malicious node exist in the malicious scenario. Nodes use RPL for multi-hop routing in order to build a single-parent routing tree on top of the mesh. However, in both scenarios each node is placed within the transmission range of other nodes so that packets are delivered directly to the destination. Figure 3.10 has a “10m background grid”. This is useful because it displays a grid of 100 m^2 squares in the background. The distance between benign nodes in both scenarios is 10m. In the normal scenario, each node sends 1 DIS message per minute. This is the default sending rate implemented in ContikiOS and was also used by Le et al. [95]. In contrast, the malicious scenario has a malicious node which sends 10 DIS messages per second.

3.3.4 Simulation Parameters and Metrics

The configuration used in our simulations is the default of ContikiOS. Specifically, RPL is configured in storing mode, using Minimum Rank with Hysteresis Objective Function (MRHOF) as OF and ETX as metric. Simulation duration is set to 25 minutes and each

Table 3.3 Additional node type

Node type	Description	Radio Duty Cycle driver	DIS sending interval
Detector	Sniffs traffic from neighbours in order to detect malicious nodes. Stores information about other nodes (messages exchanged, packet interval).	NullRDC	N/A

Table 3.4 Number of node types in each scenario

	Servers	Benign nodes	Malicious nodes	Detectors	Total
Normal scenario	1	2 to 8	-	1	4 to 10
Malicious scenario	1	First case: 2 to 7 Second case: 3 Third case: 4	First case: 1 Second case: 1 to 6 Third case: 1 to 6	1	First case: 5 to 10 Second case: 6 to 11 Third case: 7 to 12

scenario is repeated 25 times. By repeating scenarios we get more accurate results and a large sample for analysis. Apart from duration and number of repetitions, the rest simulation parameters were kept the same as in Table 3.2.

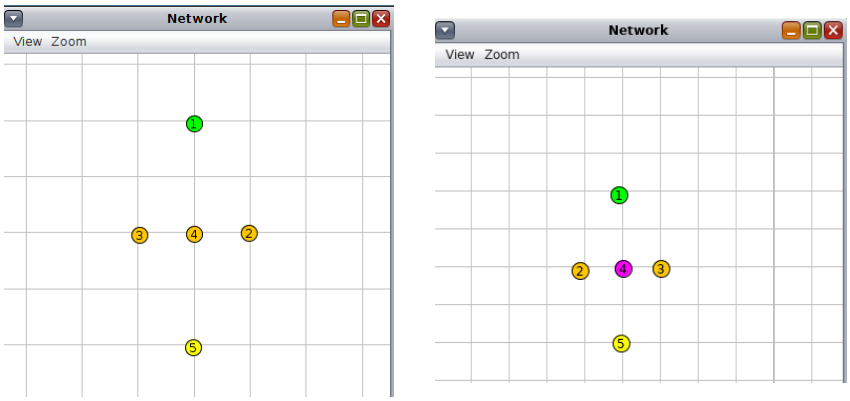
In our scenarios, we place nodes within the range of all other nodes. Detector is in promiscuous mode to gather packets and checks the network every 5 seconds. This means that the total number of checks is 300 over the whole duration of each simulation.

Various metrics, described below, were utilised to understand the characteristics of a normal sensor network. These metrics represent the average numbers taken from each simulation over 25 repetitions. IDS root or server is not included in the calculations because it receives packets only. The metrics and the equations used for each one are explained below:

- *Mean packet interval:* Indicates how often on average a node sends a packet, measured in seconds.
- *Mean number of DIS messages:* Indicates the number of DIS messages sent by a node on average.
- *Mean number of other messages:* Indicates the number of other messages sent by a node on average, apart from DIS messages. These could be DIO, DAO, or UDP messages.

Below we provide the equations for the aforementioned metrics. The mean packet interval, $E[I_{pkt}]$, is measured in seconds and is given by:

$$E[I_{pkt}] = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} \frac{p_{ij}}{n}}{m \cdot r} \tag{3.1}$$



(a) Normal scenario (b) Malicious scenario

Fig. 3.10 Network topology used in simulations. Colours for node types: green = server, orange = benign node, purple= malicious node, yellow = detector.

where: i is the i-th IDS code execution, m is the total number of IDS code executions, j is the j-th neighbour, n is the total number of neighbours, p_{ij} is the packet interval of j-th neighbour at the i-th IDS code execution, and r is the total number of repetitions of each simulation scenario.

The mean number of DIS messages, $E[N_{dis}]$, is given by:

$$E[N_{dis}] = \frac{\sum_{k=1}^{r} \sum_{i=1}^{m} d_{ki}}{r} \tag{3.2}$$

where: k is the repetition number, and d_{ki} is the number of DIS messages detected during i-th IDS code execution in the k-th repetition.

The mean number of other node's messages, $E[N_{other}]$, is given by:

$$E[N_{other}] = \frac{\sum_{k=1}^{r} \sum_{i=1}^{m} o_{ki}}{r} \tag{3.3}$$

where: o_{ki} is the number of other messages detected during i-th measurement in the k-th repetition.

3.3.5 Results

In this section, we present the simulation results. Measurements were taken from the detector which was in promiscuous mode and recorded packet intervals, DIS and other messages (DIO, DAO, UDP) exchanged by network nodes. We reference to the first case of malicious scenario with one malicious and 2 to 7 benign motes, the second case with 3 benign and 1 to 6 malicious motes, and the third case with 4 fixed benign motes and 1 to 6 malicious motes.

Starting with the *Mean packet interval* metric, in normal scenarios (4 to 10 nodes) the mean interval is 63.67 seconds. On the other hand, the first case of malicious scenario (5 to 10 nodes), the mean interval is 3.34. This big difference in packet interval is due to the malicious node. This node sends DIS packet every second making other nodes to respond and thus, reducing the packet interval of the simulation. Packet interval of second and third cases of malicious scenario is depicted in Figure 3.11. The given graph outlines the trend of both simulated cases. Both start with a mean interval of about 3 seconds having one malicious node in the network. Then, as the number of malicious nodes increases, the interval drops below 2 seconds. This happens because more and more malicious nodes send packets frequently, reducing the overall packet interval.

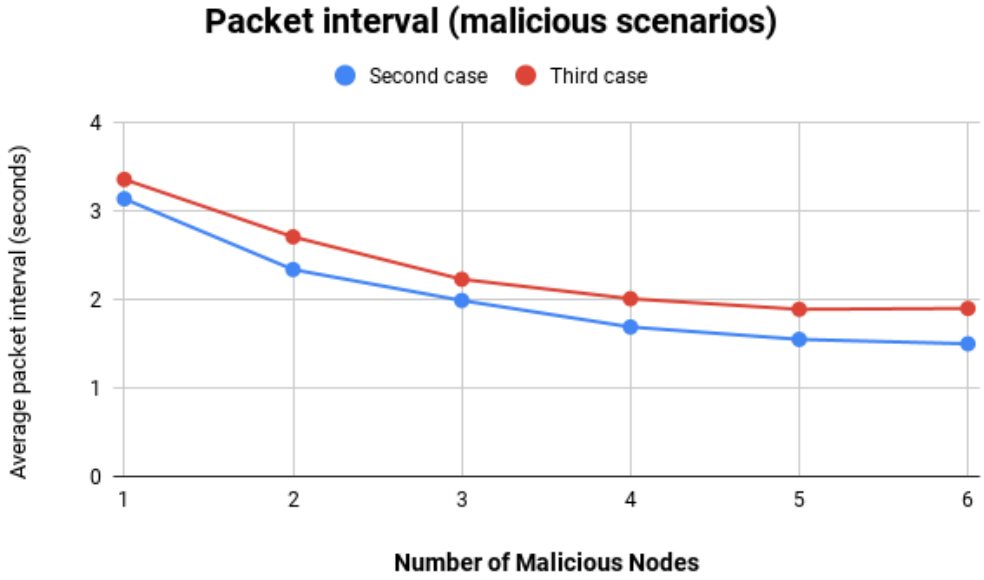
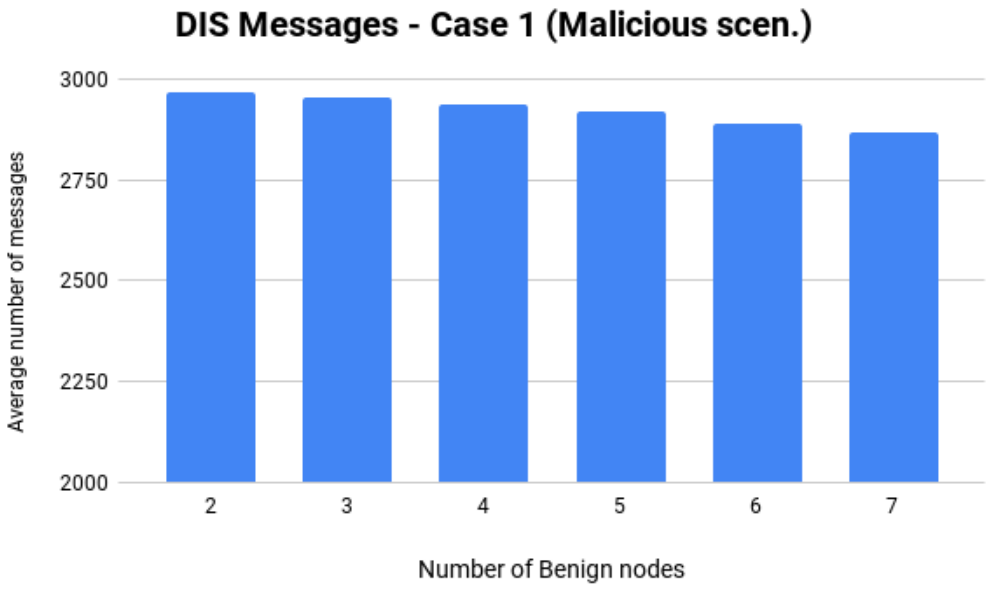


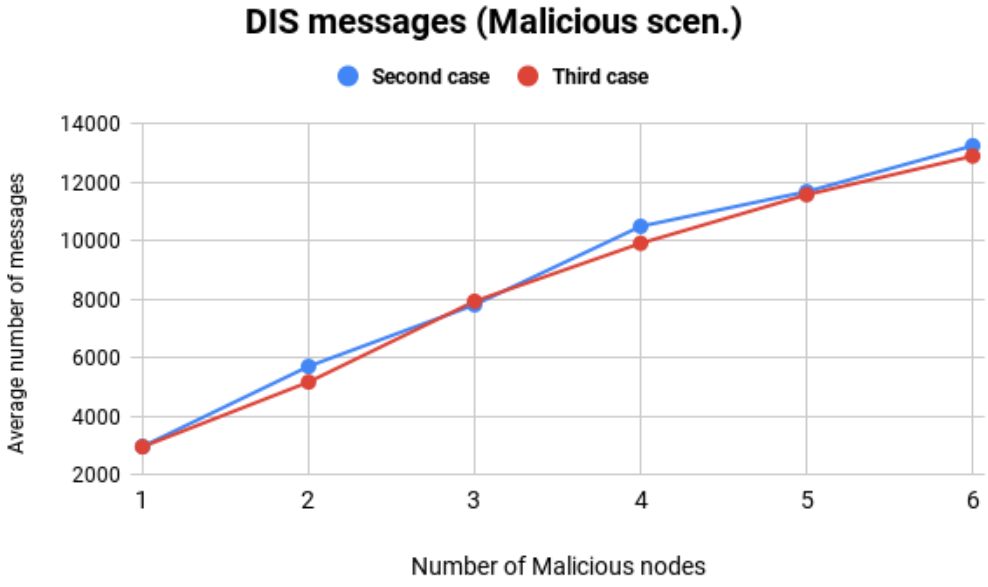
Fig. 3.11 Mean packet interval in malicious scenario (cases 2 and 3)

Examining the *Mean number of DIS messages*, several conclusions can be drawn. First of all, we found that in normal scenarios with one and two benign nodes we have an average of 1 DIS message. Increasing the number of benign nodes leads to the increase of the mean number of detected DIS messages up to 3.5 messages (i.e., some nodes send 3 or 4 DIS messages). This is expected because nodes are configured to send 1 DIS message every 60 secs in case they do not find a parent. Therefore, as the number of nodes increases, the chances a node needs to send a DIS message also increases.

On the other hand, the number of DIS messages in the malicious scenario (cases 1-3) is rocketed as shown in Figure 3.12. The first case is depicted in Figure 3.12a while cases 2 and 3 are shown in Figure 3.12b. Looking at Figure 3.12a (1st case), it starts from almost



(a) Malicious scenario (case 1)



(b) Malicious scenario (cases 2 and 3)

Fig. 3.12 Mean number of DIS messages

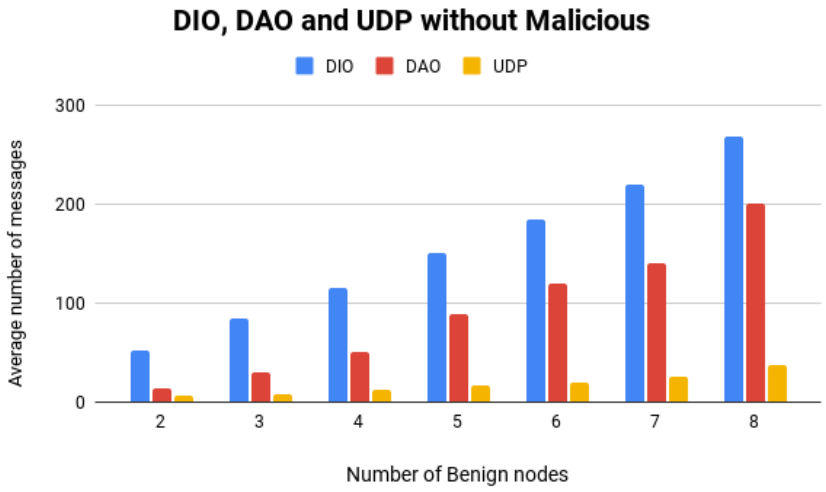
3,000 messages and gradually falls to 2,850 while the number of benign nodes rises. The lines of second and third cases as shown in Figure 3.12b upsurge dramatically as the number of compromised nodes goes up. Specifically, they start from 3,000 and climb up to almost 14,000 messages. Considering that one malicious node sends 10 DIS messages per second, this means that 15,000 DIS messages are sent during 25 minutes of simulation. As a result, the detector cannot process this number of packets but it detects a reasonable number which can be used later to detect the malicious node.

Another useful metric is the *Mean number of other messages*. The results presented in Figure 3.13 are the average number of DIO, DAO and UDP messages in each simulation. According to Figure 3.13a, the normal scenario starts with a very low number of DIO and DAO messages, while UDP messages constitute the minority. This is an expected behaviour because nodes setup the network without any interference from malicious node and send UDP packets to the root. The average number of messages rise up to 269 at the scenario with 8 normal nodes. Thus, the more the nodes, the more messages are exchanged.

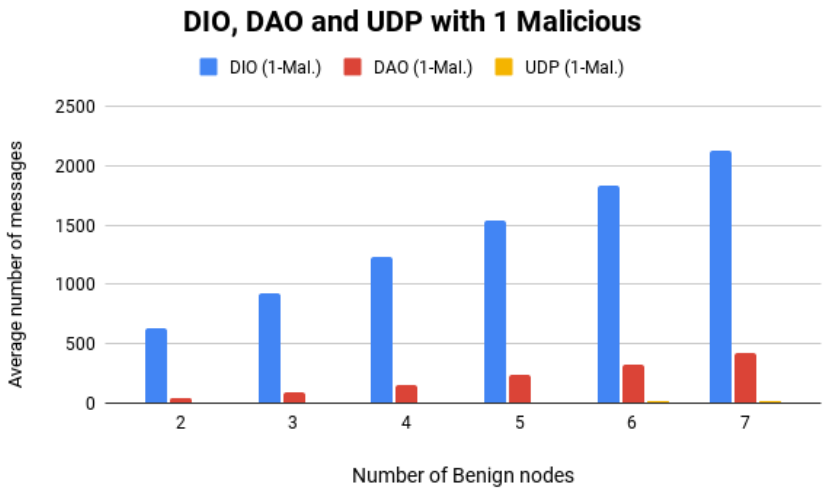
On the other hand, there is a significant rise in the number of other messages in the case with one malicious node as depicted in Figure 3.13b. The same trend is followed by the rest two cases of the malicious scenario as shown in Figure 3.13c. This is because malicious nodes send a large number of DIS messages. This can also be confirmed from the chart, where the DIO messages constitute the majority of the exchanged messages. Note that DIO messages are sent as a response to DIS messages originated from the malicious node. This demonstrates that a malicious node can degrade network performance and affect the operation of benign nodes.

3.4 Traffic and Resource-based Attacks

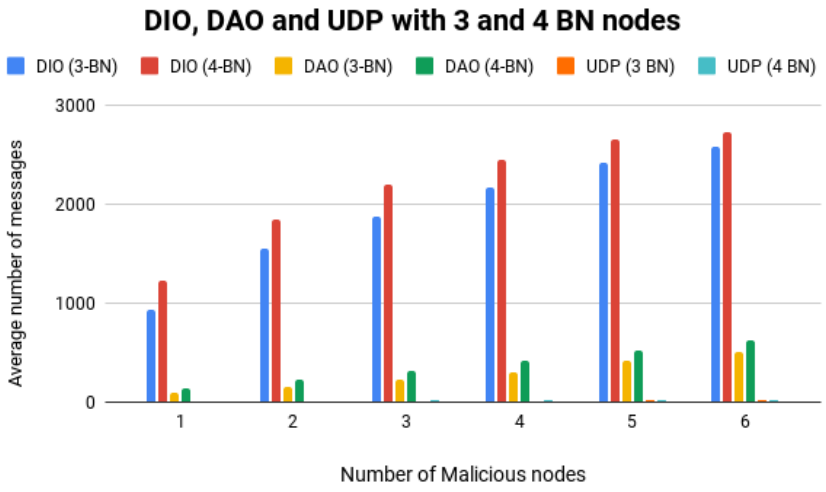
In previous sections, the impact of IoT-specific DoS attack, namely DIS flooding, was extensively studied. DoS attack is based on the RPL routing protocol and affects the availability of the network. In this section, attacks affecting the traffic and resources of IoT devices are examined. Specifically, a combination of rank and blackhole attacks is implemented in the newer version of ContikiOS, called Contiki-NG. Moreover, measurements are gathered from simple log collectors, called IDS detectors. Those detectors are configured to operate in separate RPL Instance which has a separate root node, called IDS root, from Border Router (BR). The current section is related to **Contribution (b).**



(a) Normal scenario



(b) Malicious scenario with 1 malicious node (case 1)



(c) Malicious scenario with 3 and 4 benign nodes (cases 2 and 3)

Fig. 3.13 Mean number of other messages (UDP, DAO, DIO)

3.4.1 Design

Apart from DoS attacks, we wanted to explore other IoT-based attacks. Rank attack is an example of attack impacting the packets exchanged in IoT networks. According to [24], in rank attack usually a malicious node may intentionally advertise a lower rank in order to attract neighbouring devices to select it as preferred parent. A parent node is needed in order to form the DODAG network and allow creation of routes reaching BR. In cases where networks are small, the best parent is the BR itself. In other cases, metrics such as rank and ETX are used to select the best parent. If a malicious node is chosen as the best parent of several nodes, it can attack the network affecting its availability and integrity. As a result, the malicious node will be the single point of failure of the network. Another severe attack is Blackhole. A blackhole attacker can degrade the performance of the IoT network by dropping all incoming packets. As a result, no packets are forwarded from this node.

Both attacks can cause severe problems to networks. For this reason, a combination of blackhole and rank attacks is implemented and tested in our experiments. They are combined to achieve the highest impact in the network by attracting neighbouring nodes and dropping all packets. Specifically, malicious node launches rank attack after 2 minutes of simulation by sending DIO packets advertising a fake low rank. Then, nodes will exchange control packets and will select the malicious node as the best parent. Once parent is chosen, nodes will start transmitting packets to BR through the attacker. However, all packets arriving to the malicious device and need to be forwarded, will be dropped. This is the general idea of how blackhole and rank attacks achieve network disruption. Attack is launched after 2 minutes to allow nodes form the network. A special node, called *udp-client-malicious*, is configured to execute these attacks in simulated scenarios.

The attacks are implemented in RPL-lite protocol which is supported by Contiki-NG. RPL-lite protocol is a lightweight version of RPL protocol which removes some features so that memory footprint is minimised. The transition of the project to the newer Contiki-NG version was decided due to several factors explained in the next chapters.

3.4.2 Implementation

The implementation of rank attack is presented in Algorithm 5. As can be seen, the malicious code is implemented inside the *rpl_icmp6_dio_output* function. This function is called when DIO packet will be sent. After the packet is created, attacker advertises fake rank with value 129. The minimum default rank is 128 and it is assigned to root. Therefore, nodes will consider attacker as a good parent to route their packets.

Algorithm 5 Rank attack implementation

1: **Input:** IP address of neighbour
2: **Function** $rpl\_icmp6\_dio\_output(ip\_address)$:
3: $dio\_packet \leftarrow create\_dio\_packet(ip\_address)$;
4: $dio\_packet.dag\_rank \leftarrow 129$;
5: **Send packet**

Algorithm 6 shows blackhole attack implementation in Contiki-NG. All incoming packets are dropped only if need to be forwarded, and after 2 minutes of simulation are passed. The code is implemented inside an existing function of Contiki-NG source code called *uip_process*. The function is called when a new packet arrived to the node.

Algorithm 6 Blackhole attack implementation

1: **Input:** *Packet* Pkt_{ij}
2: **Function** $uip\_process(Pkt_{ij})$:
3: $next\_hop \leftarrow Pkt_{ij}.next\_hop$;
4: $time\_attack \leftarrow 2$;
5: **if** $next\_hop \neq NULL$ and $current\_time \geq time\_attack$ **then**
6: **drop packet**
7: **end if**

3.4.3 Scenarios

Two main scenarios were examined; normal and malicious. Nodes in normal scenario are operating normally and no malicious attackers exist. Malicious scenario has one or more attackers. This scenario is studied to understand the impact that rank and blackhole attacks have in devices and network as a whole. Devices in both scenarios are using the default MRHOF as OF to choose a parent.

There are 5 types of nodes in the scenarios; BR, IDS root, benign nodes, malicious nodes and IDS detectors. BR and IDS Root form the two different networks, and play the role of sink for both networks. IDS nodes have different RPL InstanceID than other nodes. In addition, benign and malicious nodes are configured to join the network and start sending UDP packets to BR once DODAG network is formed. However, malicious nodes start attacking the network after 2 minutes.

Benign nodes are 30 in both scenarios while malicious nodes are 6 in the malicious scenarios. A varying number of IDS detectors is deployed in the network to study and find the optimised number of detectors to avoid the attackers. However, IDS nodes are deployed in all scenarios only for capturing traffic and calculating metrics.

3.4.4 Simulation Parameters and Metrics

The metrics used for studying the impact of the mentioned attacks are the following:

- *Mean Packet Delivery Ratio (PDR):* Indicates the ratio of the total number of unicast packets received by BR up to the total number of unicast packets generated by all benign and malicious nodes. It does not include UDP re-transmitted packets.
- *Mean Parent Switch:* Shows the number of parent switches that benign and malicious nodes do during the simulation. A parent switch happens to select a better route to the BR. Changing parent results to changing the rank of a node.
- *Mean Packets Dropped:* Percentage of packets dropped by the attackers during the simulation. Malicious nodes drop all types of packets that should be forwarded to next hop.

Below, the mathematical definitions for the metrics are provided. Let S_r be the packet delivery ratio for repetition r. It is calculated by:

$$S_r = \frac{Rcvd}{\sum_{k=1}^{n} P_k} \tag{3.4}$$

where *Rcvd* is the total number of packets received at BR, k is the k-th node, n is the total number of nodes, and TP_k is the total number of packets sent from node k. The *Mean Packet Delivery Ratio (PDR)*, $E[DRpkt]$, is given by:

$$E[DRpkt] = \frac{\sum_{r=1}^{m} S_r}{m} \tag{3.5}$$

where r is the r-th repetition, and m is the total number of repetitions for each simulation.

The *Mean Parent Switch*, $E[PS]$, is given by:

$$PS_r = \sum_{k=1}^{n} P_k \tag{3.6}$$

$$E[PS] = \frac{\sum_{r=1}^{m} PS_r}{m} \tag{3.7}$$

where P_k the parent switch of node k, PS_r the sum of parent switches for repetition r, and r the number of repetitions for each simulation.

The *Mean Packets Dropped*, E[Dpkt], is given by:

$$D_r = \frac{Drop_r}{\sum_{k=1}^{n} TA_k} \tag{3.8}$$

$$E[Dpkt] = \frac{\sum_{r=1}^{m} D_r}{m} \tag{3.9}$$

where $Drop_r$ the total packets dropped in repetition r, TA_k the total number of packets transmitted from node k including multicast and unicast packets, D_r the percentage of packets dropped in repetition r, and m the number of repetitions for each simulation.

Table 3.5 Simulation configurations

Parameter	**Value**
Grid size	70x70
Topology	Random
Simulation time	60 minutes
Seed number	Random in each execution
Max MAC packet retries	3
Max buffered packets in MAC layer	20
Operating System	Contiki-NG 4.4
Simulator	Whitefield simulator

Configuring simulations was done using the Whitefield's framework configuration file. The file allows user to define several options. The options used are shown in Table 3.5. The grid size is 70x70 while topology is random in each execution. Simulations run for 60 minutes. Malicious scenarios are repeated 4 times while normal scenarios are repeated 10 times. IDS detectors are used only to capture traffic and calculate metrics using the measurements taken. Using more IDS detectors allow us to collect large sample for analysis. A parameter playing a significant role in simulations is seed number. It affects the behaviour of the nodes regarding processing and packet transmission times. Therefore, we use random seed in each repetition so that Whitefield simulator produces random results in each run. Nodes are configured by default to retry MAC packets up to 3 times while the maximum number of packets waiting in the buffer in MAC layer is 20. As already mention in previous sections, Whitefield framework combines the NS-3 simulator for running the actual simulations and uses Contiki-NG to emulate the behaviour of a real world OS.

3.4.5 Results

The maximum values of mean PDR is shown in Figure 3.14. Normal scenarios have a maximum averaged value of almost 95% while in BHR scenario this percentage drops almost by 14%. As indicated, there is a standard error of 10% in each scenario. The difference between the two scenarios shows that blackhole and rank attackers can actually degrade the performance of the network.

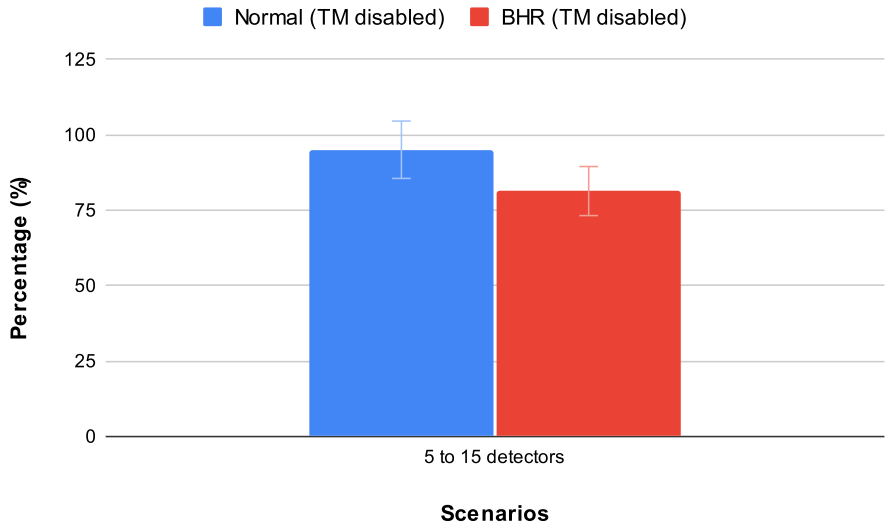


Fig. 3.14 Comparison of maximum values of Mean PDR for BHR and Normal scenarios

Figure 3.15 compares the maximum value of *Mean Parent Switch* of two cases; Normal and BHR namely. Normal scenarios have maximum only 32 parent switches on average, while attacking the network causes over 600 parent changes. Parent switch is vital as the node basically chooses the best route to root. Having such as big difference in the two scenarios means that malicious actors successfully attacked the network using rank and blackhole attacks with great impact. The impact is translated into large energy consumption, and increased traffic overhead.

The *Mean Packets Dropped* obtained by simulating the BHR scenario in various cases are depicted in Figure 3.16. The bar chart shows only the results from deploying an even number of IDS detectors. IDS detectors operate in passive mode for gathering data. Their operation mode remains the same in all scenarios. As indicated, there are some scenarios where more than 40% of the packets on average are dropped due to the attacks launched in the network. This percentage gradually declines as more IDS detectors are deployed, reaching almost 34%. Therefore, blackhole and rank attackers may largely affect the network operation by dropping almost 42% of packets sent in some cases.

3.5 Chapter Summary

In this chapter, the impact of IoT-based attacks was studied using several simulations. DoS and other routing attacks were designed and implemented in RPL protocol of ContikiOS. DoS attackers were simulated in small and larger network using various configurations.

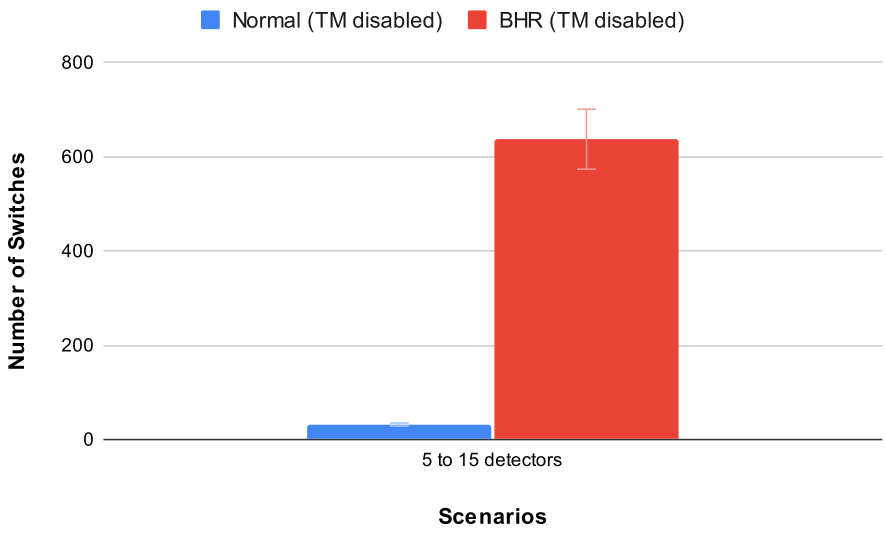


Fig. 3.15 Comparison of maximum values of Mean Parent Switch

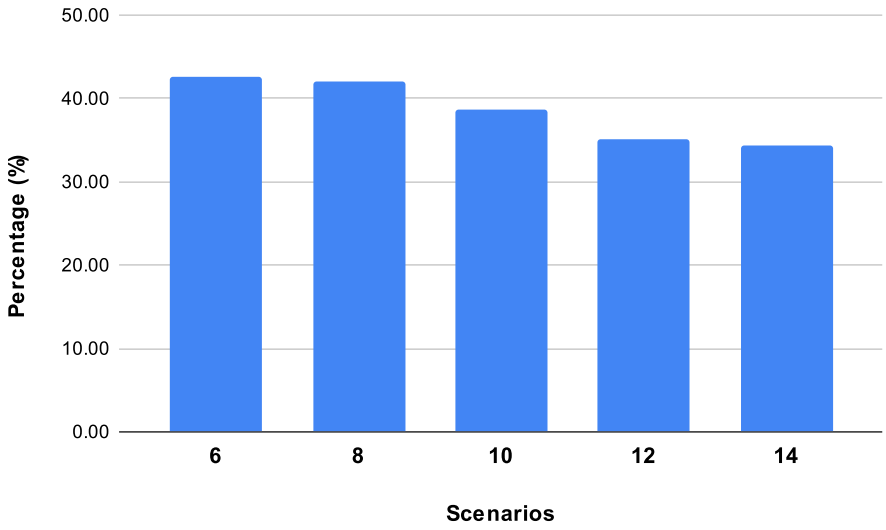


Fig. 3.16 Mean Packets Dropped for BHR scenario

Results showed that energy consumption increased in IoT devices. Moreover, packet sent interval and other related statistics were studied to understand the impact of RPL-specific DoS attack. Regarding other routing attacks, rank and blackhole attacks were implemented in Contiki-NG. Simulation results showed a degraded performance of IoT network with increased packet dropped and reduced PDR.

Chapter 4

Security Framework for IoT-based Devices (SRF-IoT) Overview

The current chapter presents the design and implementation details of Security Framework for IoT-based Devices (SRF-IoT). Firstly, the operational details such as trust calculations and various components of the scheme are described. Then, each component of the SRF-IoT framework is presented. The external Anomaly-based SRF-IDS is introduced and described in detail. Additionally, the implementation steps followed to link SRF-IDS communication component with SRF-OF are described. Apart from that, the implementation of SRF-IDS detection module in both ContikiOS and Contiki-NG is described. Simulation details and evaluation results are depicted. Last, the processes executed by SRF-IoT framework are presented.

4.1 Operation Overview

SRF-IoT is a Security Framework designed for IoT RPL-based networks. It aims to support the security of an IoT network by identifying and avoiding malicious devices. This is achieved by embedding trust in RPL protocol for choosing the most trusted node as parent. Additionally, an external IDS, called SRF-IDS, is used to shield the network from internal attackers. SRF-IDS is presented in detail in Section 4.3. DIS flooding can be detected by SRF-IDS. SRF-IoT framework is proposed as an extra security measure for RPL-based networks. The combination of rank and blackhole attacks are mostly considered as those attacks could severely disrupt network operation [78].

4.1.1 Trust Concept

In IoT networks, smart devices provide services to their peer connected devices. Evaluating the reliability of a device would enhance networks' security and performance [48],[101]. Bearing this in mind, trust is used in our proposed solution to form a secure network by avoiding malicious actors. According to [102], trust in wireless networks may be defined as a degree of belief to forecast a node's forthcoming actions which depend on its previous experience and information gained from device's behaviour. SRF-IoT uses trust concept to evaluate the reliability of deployed nodes.

Trust value is calculated as the number of successfully forwarded packets between the node and its neighbours for at least 5 seconds. As discussed in other trust systems [86, 87, 79, 103], there are two types of trusts; direct and recommended trust. Direct trust is calculated by the node after monitoring its direct neighbour's packet forwarding behaviour [86]. On the other hand, recommended trust can be seen as a recommendation from a third party node. Basically, recommended trust is the trust value given by a third node that is 2-hops away and it recommends its direct neighbour to other nodes. However, recommended values cannot always be trusted, and a third-party node might provide with wrong information. Therefore, only selected nodes can be used to provide nodes with trust recommendations.

In our work, we use direct trust for securing the RPL protocol. Each node computes the trust value of its direct neighbours based on the information received from an SRF-IDS detector. There is a trade-off between active network monitoring and saving energy in the sense that smart devices have limited energy resources and cannot monitor network for long time periods to detect attackers. The advantage of our approach is that monitored nodes do not waste energy on monitoring neighbours. The SRF-IDS is used as an information collector entity. Resulting trust value is used by each node to select the most trusted parent. Devices with high trust value are selected as parents while those with lower trust values are avoided. Nodes which fall below a trust threshold are blacklisted. The main goal is to secure the network by: i) selecting a path in the network to send packets where nodes have high trust score, and ii) avoid malicious nodes or nodes with low trust score. The proposed framework consists of four procedures: gathering information from SRF-IDS, calculating trust, assessing trust, and identifying malicious nodes.

Trust is utilised in this work to improve the RPL routing protocol. Although some security mechanisms exist in RPL, attacks such as blackhole and rank attack may occur in a real-world IoT network. Embedding trust in RPL will enable devices to learn their neighbours and choose the best parent based on this value. This knowledge is gained through IDS, which processes sniffed packets and sends various metrics to monitored smart devices. In the monitored network, IoT devices get the data from SRF-IDS, and based on some algorithms

they calculate the trust value. More weight is applied on the current behaviour of a node than its history. This is to avoid cases where a malicious node has a high trust value initially, and then starts to attack others. Ranking and selecting neighbouring nodes based on their latest trust value helps to avoid malicious actors. Therefore, trust and SRF-IDS concepts are utilised as a method of defence to detect and avoid these attacks.

4.1.2 Architecture Components

A high-level architecture of SRF-IoT framework is presented in Figure 4.1. On the left side, the external IDS, called SRF-IDS, is shown along with the internal components. SRF-IDS is responsible for packet sniffing, monitoring the behaviour of nodes, and updating monitored nodes with trust metrics. The transfer of trust metrics from SRF-IDS to monitored nodes is done by transmitting special control packets. These three procedures belong to the *TM* module which is embedded into SRF-IDS detectors. Moreover, SRF-IDS root has an embedded detection module for detecting attacks such as DIS flooding. A traditional firewall is also used to block external attackers. On the right side, the basic functionality of SRF-OF which is embedded into monitored nodes is represented. Basically, a monitored node receives trust metrics from SRF-IDS. Then, it calculates the new trust values, and updates the blacklist with suspicious nodes. The parent selection algorithm is based on the calculated trust value of the specified node.

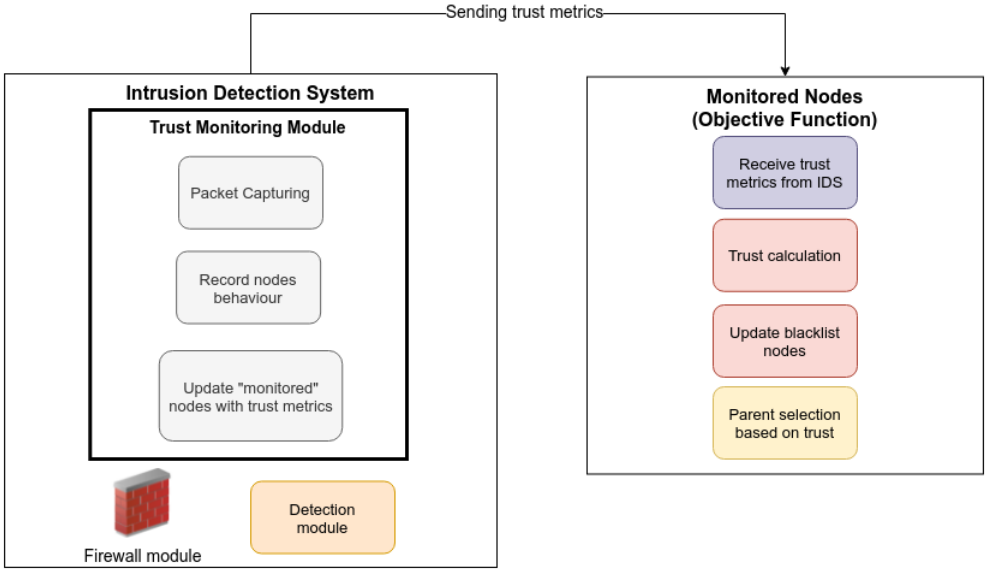


Fig. 4.1 SRF-IoT Framework high level architecture

4.2 Components

In this section, the two main components of SRF-IoT framework are discussed in more detail. The external SRF-IDS is the first component that helps the system to capture and analyse traffic, while the SRF-OF is the second component which is embedded into the routing protocol.

4.2.1 External Security Framework Anomaly-based Intrusion Detection System (SRF-IDS)

A main component of SRF-IoT scheme is the external Anomaly-based Intrusion Detection System (SRF-IDS). It consists of two types of devices; SRF-IDS root and SRF-IDS detectors. SRF-IDS root is actually a router responsible to take final decisions about malicious nodes, while SRF-IDS detectors are sensor devices operating in promiscuous mode to gather information from monitored network. SRF-IDS detectors are usually deployed near TX range of monitored nodes to allow packet sniffing. Each device type contains several internal modules such as communication, detection, decision and other modules. These modules are described extensively in the next section. SRF-IDS has detection mechanisms for IoT-based DoS attacks as well as topology and resource-based attacks such as rank and blackhole attacks. The former attack can be identified by SRF-IDS without any intelligence from monitored nodes. However, the latter attack can be detected only after gathering information such as the number of packets forwarded from monitored nodes.

One of the features of SRF-IDS is its ability to operate independently in a different RPL Instance without interfering with the monitored network. This is achieved by using a unique RPL InstanceID in packets, only for SRF-IDS nodes. In addition, SRF-IDS has the possibility to communicate with monitored network to update nodes with trust metrics. This is achieved by implementing a packet parser in benign devices to export trust metrics, and configuring the proper RPL InstanceID in packet header. Consequently, SRF-IDS network and monitored network operate in different RPL Instances, while SRF-IDS nodes can send packets to monitored nodes.

Moreover, SRF-IDS root' decision module identifies malicious nodes both in a centralised and distributed way. A centralised decision is taken for DoS attacks as the SRF-IDS root needs to collect reports from several SRF-IDS detectors, analyses data and takes a decision. In case of topology attacks such as rank and blackhole attacks, SRF-IDS detectors act individually, and report to SRF-IDS root suspicious nodes for further actions. Actual detection of rank and blackhole attackers happens by benign nodes and not SRF-IDS detectors. SRF-IDS detectors in this case work as a source of intelligence for monitored network' nodes.

As a last security measure, our proposed SRF-IDS is designed to support a host-based firewall to block malicious IPs in SRF-IDS root. Firewall rules are created by SRF-IDS root once a malicious node is reported by SRF-IDS detectors. Attackers are blacklisted to stop their communication to external networks. As our focus is on designing and developing the SRF-IoT framework, the firewall module is not in the scope of this project. Therefore, reader can deploy any typical firewall to block network traffic of malicious attackers.

4.2.2 Security Framework Objective Function (SRF-OF)

The second component of the SRF-IoT framework is the SRF-OF which uses trust concept, and operates independently in each monitored node. It was designed to accomplish three tasks: 1) gather intelligence about candidate parents from SRF-IDS, 2) evaluate candidate parents by calculating trust value for each of them based on the information received, 3) choose the most trusted node as the preferred parent of a node. All those operations are required to protect IoT networks from attacks such as blackhole, rank and other types of routing attacks. In other words, monitored nodes will receive information from SRF-IDS about the packet forwarding behaviour of their neighbours. This metric will allow them to calculate a neighbour's trust and select the parent node with the highest trust value. The higher the value, the more trustful it is. If all monitored nodes perform this procedure, nodes will avoid blackhole and rank attackers as those drop incoming packets, and network will be more secure. SRF-OF was implemented as an OF which can be executed by any node deployed in the network. For this reason, SRF-OF is implemented as part of the routing protocol. In our project, SRF-OF is embedded into RPL protocol so that it can be evaluated using simulations with ContikiOS. Furthermore, SRF-OF was designed and implemented based on Minimum Rank with Hysteresis Objective Function (MRHOF) except the implementation of selecting the node's best parent.

As the trust value is based on packet forwarding behaviour of a node, designing SRF-OF required some important decisions. The first and most important issues that had to be resolved was the time synchronisation between SRF-IDS and monitored nodes. Metrics such as packets sent and forwarded from the monitored node are calculated by SRF-IDS detectors for a time period, and then forwarded to monitored nodes for further processing. This time period should be the same between SRF-IDS detector and monitored node so that both nodes refer to the same time period, and consequently, calculate the trust value based on the proper metrics. For this reason, a modification was made to calculate metrics and transmit them from SRF-IDS detector to monitored nodes every 15 minutes. Then, statistics in both ends will be reset. For instance, SRF-IDS detector will sniff packets sent and forwarded from a monitored node. Then, every 15 minutes, these metrics will be transmitted to the

monitored node. Once they are successfully received, monitored node will compare its own metrics of packets sent with the packets counted from SRF-IDS detector. In cases where a difference in metrics is found, proper calculations will be done to correct the values. An example of such difference is when the packets captured by SRF-IDS detector are less than the packets actually sent by the monitored node. That means, SRF-IDS detector missed some packets sent by nodes. Those cases are detected by monitored nodes which assign their own actual value for the specific metric. After evaluating the metrics received by SRF-IDS versus monitored node's own metrics, measurements and counters stored in both devices will be cleared from memory so that storage requirements are minimised. In addition to that, SRF-IDS detectors are designed and implemented with proper mechanisms to detect and fix any problematic metrics such as receiving duplicate packets. In those cases, measurements are updated accordingly without calculating duplicate packets.

The time period of 15 minutes was chosen to process metrics because after several experiments, we found that benign nodes usually transmit data packets every minute. Thus, in order to minimise the amount of space needed to store measurements of each neighbour, a time interval was introduced for calculating and resetting recorded statistics. Apart from that, processing overhead is kept in low levels because in a time period of one hour, the process will be executed only four times. Reducing execution time interval would lead to lower CPU and energy overhead. An additional feature that was designed and implemented to save nodes' processing resources is that no trust calculations are executed in case a monitored node does not notice any difference between previous and new packet metrics received. In this way, monitored node will compare the metrics received by SRF-IDS before proceeding to a parent switch.

Another design decision that had to be made was to reduce storage requirements. Monitored nodes should be able to store metrics of their neighbours. After careful investigation, we found out that a node has an average two or three candidate parents to choose from. As a result, the structure for storing metrics of candidate parents was configured to allow a maximum of 5 nodes. This means that a node could store packet forwarding behaviour of 5 neighbouring nodes, which is not the usual case but was designed to fit the worst case scenario.

The last issue that had to be solved was the packet parsing from SRF-IDS detectors. Packets containing information about the packet forwarding behaviour of neighbouring nodes, arrive to monitored nodes using special RPL control packets. Therefore, a parsing process had to be designed and implemented so that information is exported from packets. Our solution for this issue was to implement a packet parser that, after verifying packet header to ensure packet is from SRF-IDS, the node exports the data from each packet field

and stores it in a structure for each neighbour. Metrics and trust calculations are executed for each neighbour using the data stored in the node itself.

4.3 Anomaly-based Intrusion Detection System (SRF-IDS)

This section presents in detail the external Anomaly-based SRF-IDS. The novelty of the proposed SRF-IDS is that it can be deployed in a separate RPL Instance to monitor network and detect routing attackers without interfering with monitored nodes. In the first section, details about the design of SRF-IDS are given. Then, detection module implementation steps in both ContikiOS and Contiki-NG are described. The last section presents simulations and results of SRF-IDS evaluation.

4.3.1 Problem Statement

As already discussed in previous chapter, attacks may occur in different parts of a network. For example, an attacker may compromise a Border Router (BR) to take control of the whole network or may target internal devices to attack the network itself. Network layer is mostly considered in this work. Several attacks such as eavesdropping, and various routing attacks may cause severe disruption to network operation, or may leak sensitive information.

For the above reasons, a proper solution should be able to secure those IoT networks that deploy thousands of smart devices. Compromising a single device in a critical network could cause a chaos in the economy of a country. Therefore, solutions should be designed and developed to protect those networks. Traditional IDSes are not applicable to IoT networks as they have different requirements from current networks. Smart devices have constraints in processing, memory and energy, making them vulnerable and easy target for attackers. As a result, it is a challenge to suggest a solution that will protect IoT devices that operate using multiple technologies and protocols. The proposed SRF-IDS should have the ability to protect IoT networks by supporting different technologies, handling large amount of data packets, and responding quickly to possible security events.

4.3.2 SRF-IDS Design

The design of SRF-IDS component is discussed in this section. The architecture and several components of the proposed SRF-IDS are explained in detail in the following subsections.

SRF-IDS Architecture and Components

In this subsection, the proposed SRF-IDS solution is described.

In addition to the typical sensor nodes, we consider two new types of devices: i) a router, called SRF-IDS root, for running both the detection module and a firewall, and ii) sensor-like devices, called SRF-IDS detectors, for monitoring and sending suspicious traffic to the router. In a typical scenario of an IoT network, there will be one SRF-IDS root and up to 100 SRF-IDS detectors. The SRF-IDS root may also play the role of the BR of the network.

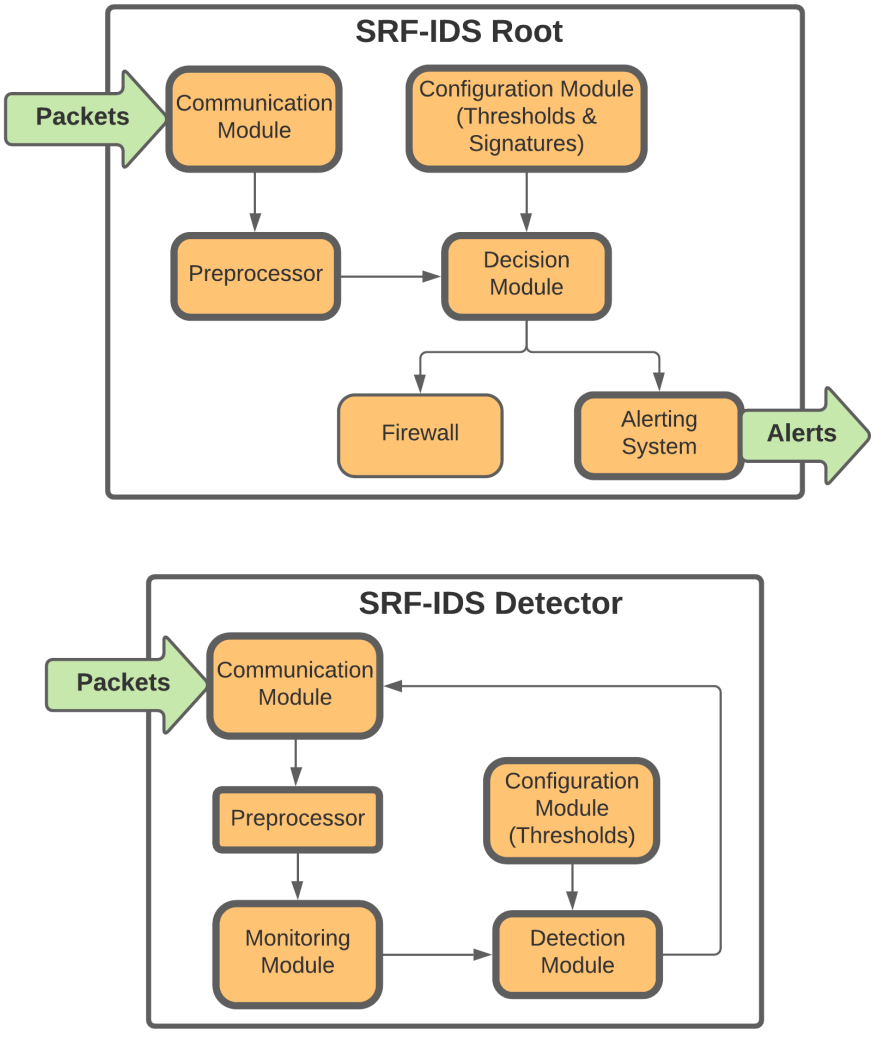


Fig. 4.2 IDS internal components

Figure 4.2 illustrates the two types of SRF-IDS devices along with their internal components.Both devices have some components in common. *Communication module* is a common

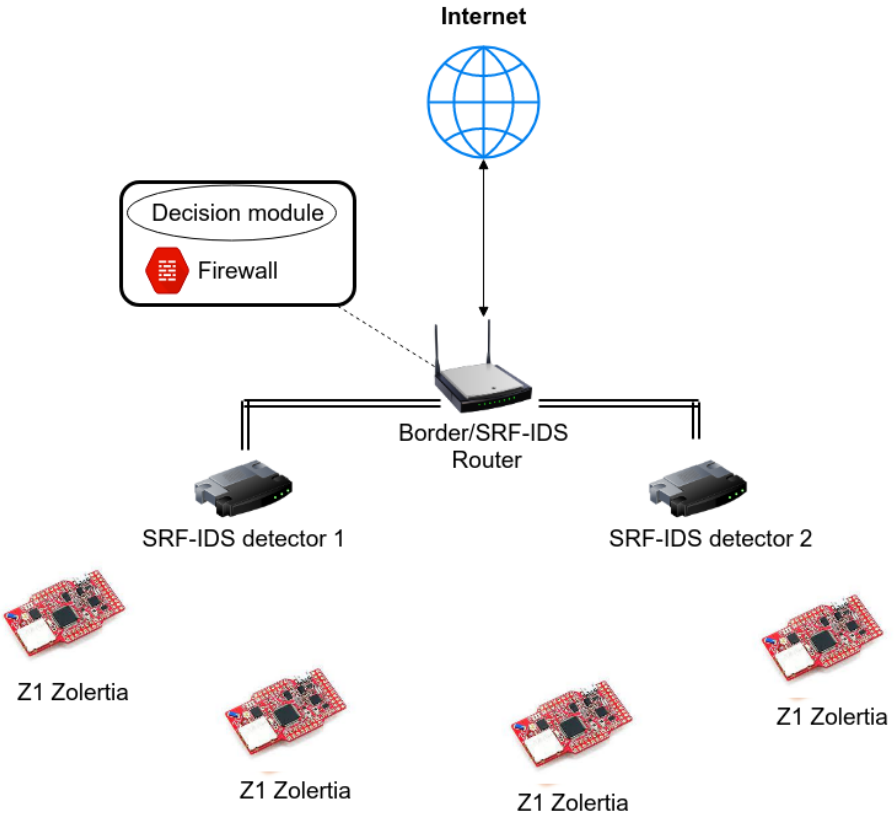


Fig. 4.3 High-level SRF-IDS architecture

and critical component for both SRF-IDS root and detectors. It is the entry point of a network packet. It is responsible for sniffing network traffic, and communicating with other SRF-IDS or monitored devices. In case SRF-IDS devices have multiple network interfaces, communication module captures packets from all interfaces. *Preprocessor* is another common component that processes packets arriving from multiple network interfaces. Specifically, it exports useful data from packets and feeds next module with information such as source IP, packet type and other useful metrics.

As depicted in the figure, each device has its own modules. In SRF-IDS detector, there are two extra modules plus the configuration module that require small processing power. *Monitoring module* was designed to be able to monitor nodes' behaviour. This is done by saving the metrics and data coming from *Preprocessor* into a temporary table for comparison purposes. Then, metrics are forwarded to *Detection module* for further decision making. In SRF-IoT, *Monitoring module* is evolved into a *Trust-based Monitoring module* which is also responsible to monitor and assess trust. *Detection module* is the heart of SRF-IDS detector. Here, the actual detection of suspicious nodes happens. Lightweight algorithms are executed to identify malicious devices. Threshold values are provided by the *Configuration module*. If

the algorithm result indicates that a node is malicious, immediately node' details are sent to SRF-IDS Root via communication module. The *Configuration module* is responsible for providing and storing thresholds that are required to check during algorithm execution.

Looking at SRF-IDS Root, *Decision module* is an important component that identifies malicious nodes using complex algorithms and data provided from multiple SRF-IDS detectors. *Preprocessor module* and *Configuration module* provide the data input to the algorithms. SRF-IDS Root's *Configuration module* provides thresholds, signatures of known attacks and other configurations needed for decision making. In contrast, the module in SRF-IDS detectors provides only the predefined thresholds needed for executing the lightweight algorithms. The values of those thresholds are defined from the experimental results of Chapter 3. The output of *Decision module* is forwarded to the last two components of SRF-IDS Root, the *Firewall* and *Alerting System*. If a node is identified as malicious, proper rules are created in the *Firewall* to block traffic from specific IP. Moreover, an alert is generated by the SRF-IDS Root to the system administrator to take further mitigation steps.

SRF-IDS detectors monitor sensors' traffic to help in detecting malicious nodes. Compromised devices may attempt to interrupt the network internally without having to communicate with the BR or external networks. For such cases, SRF-IDS detectors will log network traffic and, if a node's behaviour resembles a known attack, the related information will be forwarded to the SRF-IDS root for further analysis and decision making.

An example of high-level SRF-IDS architecture is shown in Figure 4.3. This is the first version of our SRF-IDS architecture, in which the BR is designed to be the SRF-IDS root as well. That means SRF-IDS detectors report to the same device, the BR/SRF-IDS root, as the monitored nodes. As depicted, there are four Zolertia Z1 sensors and the SRF-IDS consists of one SRF-IDS root or BR and two detectors. The BR/SRF-IDS root is connected to the Internet and includes two components: a *Firewall* and a *Decision module*. These two components help in protecting the network from both internal and external attacks. The *Decision module* runs algorithms to help decide if a node is malicious or not, while the firewall creates and enforces rules for blocking malicious sensor requests to external destinations. The detectors are wired connected to the BR to avoid jamming or eavesdropping via a wireless channel. Appropriate secure wireless communication scheme will be in place (e.g., [104]) if a wireless channel between the BR and the detectors is necessary or desirable. For simulation purposes, we assume that a secure wireless channel is established in the networks. So there is no need to wired connection between SRF-IDS devices.

Any packet exchanged between the sensors is captured by the nearest detector. Afterwards, a lightweight algorithm determines if traffic should be forwarded or not to the BR. We assume that detectors will be devices with limited resources. Hence, algorithms that require heavy

computations, would not be suitable. The combination of SRF-IDS root and detectors helps in capturing traffic from both internal and external communications. For example, some compromised devices may try to communicate with a remote server in order to download commands. Other compromised devices may exchange traffic locally. Our design considers all types of communications so that malicious nodes can be blocked. The BR/SRF-IDS root captures traffic from both WiFi and IEEE 802.15.4 interfaces. The BR is also able to detect attacks from Zigbee/6LoWPAN devices.

Communication Module

A critical component that allows IoT nodes to interact with others in the network is the *Communication module*. This module is designed to transmit and receive raw packets from other nodes using the network interface. *Communication module* is able to capture all types of packets even from IoT devices that have multiple network interfaces. However, in the case of SRF-IDS detectors, the module is configured to work in promiscuous mode so that network packets are sniffed from all neighbouring nodes. Normal operation such as replying to requests destined for SRF-IDS detector is unaffected. In SRF-IDS root, *Communication module* operates in normal mode without any modification. Both SRF-IDS root and detectors are able to communicate within the DODAG network using this module.

Preprocessor Module

Once a packet is received and disassembled at the *Communication module* of the node, packet details are forwarded to the next component, the *Preprocessor module*. In this component, an initial processing of incoming packets is done. Basically, packets received at multiple network interfaces are combined, and useful data is exported. Exported information may include source IP, next-hop node IP, final destination IP, packet type, timestamp and other useful data. Minor changes are made in the *Preprocessor module* of SRF-IDS detectors which sniffs network traffic. Specifically, the module in those detectors is responsible for properly parsing packets destined for detector or any packet sniffed from the network. Therefore, modifications were made in the parsing procedure so that no data is discarded in case a packet is not the final destination for SRF-IDS detector. Once packets are processed and useful metrics are exported, those metrics are forwarded to the next module for further processing.

Configuration Module

Detecting malicious attackers requires a special module to have in place the proper information and configuration. In SRF-IDS, the *Configuration module* is responsible to keep

and provide nodes with the proper configuration so that detection algorithms are executed correctly. Both SRF-IDS devices have a *Configuration module*. SRF-IDS detector uses this module to provide *Detection module*'s algorithms with thresholds that are needed to detect suspicious nodes. On the other hand, SRF-IDS root uses *Configuration module* to provide the *Decision module* with thresholds and attacks' signatures. In both cases, provided thresholds are used mostly for DoS attacks while other configurations are used during algorithm execution. Additional thresholds such as the number of reports to receive from SRF-IDS detectors to classify a node as malicious, are utilised by SRF-IDS root. More details about thresholds are given in the next sections. Signatures stored in root's *Configuration module* are used to compare and identify possible attack attempts from data received by SRF-IDS detectors. SRF-IDS detectors use less configuration parameters than SRF-IDS root because they report suspicious nodes only. Final decision making is done by the SRF-IDS root. For this reason, it can have enough computational power to execute algorithms for detecting different types of attacks.

Monitoring Module

Monitoring module is a vital component of SRF-IDS detectors. This module allows detectors to monitor the behaviour of neighbouring nodes. After the initial processing of data packets at *Preprocessor module*, metrics and exported data arrive at *Monitoring module*. Here, information such as source IP, final destination IP, packet type, timestamp and other important information are stored for each monitored node in a temporary table. Values from the table are compared in specific time intervals to decide if the behaviour of a node is normal or not. The normal behaviour means that a node sends or replies to requests as they are defined by the routing protocol. In cases where the node behaves suspiciously such as sending a batch of requests or sending packets in small time intervals, detector flags this node and forwards information to *Detection module* for further actions.

As SRF-IDS needs to collaborate with SRF-OF in order to mitigate more routing attacks, *Monitoring module* is evolved into a *Trust-based Monitoring (TM) module*. This means that apart from monitoring a node's packet sending behaviour, it also monitors the packet forwarding behaviour of monitored nodes. Metrics such as the number of packets send and forwarded by a node are also stored in a temporary table. Then, in specific time intervals, SRF-IDS detectors send those metrics, called trust metrics, to monitored nodes so that they are informed and calculate a trust value for each neighbour. At the end, a trusted parent is selected to create routes from a node to root. Moreover, trust metrics are also reported to SRF-IDS root for creating proper rules in the embedded firewall. As the goal is to help

SRF-OF component of a monitored node to calculate trust of candidate parents, we renamed this module to *TM module*.

Decision and Detection Modules

An important part of the proposed SRF-IDS is the *Decision module* within the SRF-IDS Root. This module is responsible for classifying a node as malicious or not. The decision is based on the individual information collected by SRF-IDS detectors for each node deployed in the network. *Decision module* is designed to analyse reports received by SRF-IDS detectors, aggregate measurements from various reports, and classify nodes based on thresholds. The output result of *Decision module* is forwarded to *Alerting system* to warn the system administration, and to the *Firewall* for creating the right rules if needed. In our project, the result of the *Decision module* is taken as the result of SRF-IoT framework. That means, the final classification of a node is based on the result of this module. For instance, if a node sends too many packets to other nodes, or a node sends packets in smaller intervals than a threshold, then this node may be considered as malicious. In that case, the node might be removed from the network, its IP will be blacklisted, an appropriate firewall rule will be created, and the network administrator will be alerted.

A similar module that has some differences from the previous, is the *Detection module*. This component resides within SRF-IDS detectors. *Detection module* is also extensively investigated in this project so that right formulas and thresholds are defined to detect suspicious nodes with high confidence. The difference of *Detection module* from the *Decision module* is that *Detection module* runs lightweight algorithms on detectors using less thresholds than in *Decision module* to decide if a node is malicious or not. In addition to that, *Detection module* has an input the local measurements taken from *Monitoring module* while *Decision module* considers reports sent from several detectors to take a decision. The output result of the *Detection module* is forwarded to the *Communication module* to be sent to SRF-IDS root for taking the final decision. The classification result of *Detection module* is not considered as final result.

Alerting Module

The component responsible for alerting the system or network administrator for potential attackers is the *Alerting module*. This module is located in SRF-IDS root, and its main goal is to inform user about malicious nodes. As shown in Figure 4.2, it accepts the result of *Decision module* and announces to the network administrator if any malicious node exists in the network. In case that no malicious behaviour is detected, the *Alerting module* does not

show any message. The user console of simulator tool is used to display warning messages in our case because evaluation is done using simulations.

4.3.3 Routing Attacks Mitigation

The proposed SRF-IDS aims at detecting and preventing a wide range of routing attacks. For example, DoS attacks such as DIS flooding attacks may occur inside IoT networks to achieve resource exhaustion of the sensor nodes. In addition to that, blackhole attacks, sinkhole attacks, selective forwarding, and clone ID are widely known traffic and topology -based routing attacks [65, 105] which usually exploit the RPL protocol.

The above mentioned attacks can be mitigated using existing methods such as measuring DIS message sending rate, the Received Signal Strength (RSS), packet interval, and packet data drop rate [106]. Specifically, DIS message sending rate is used in this work to detect DIS flooding attack. The packet rate of smart devices is usually very low. A device that behaves abnormally and sends more packets than others, could be considered as a malicious one. The rate of false positives must be kept sufficiently low to avoid generating many false alerts [107].

Another promising metric is the packet interval. Each device is configured to sleep most of the time. Malicious devices could exploit this feature and wake up the device to send more requests in the network. Our proposed SRF-IDS detects this behaviour by taking into account the sending intervals of all nodes in the network and calculating the average packet interval which will constitute the "normal" behaviour. Thus, any node exceeding the packet interval threshold will be considered as malicious. Those mitigation methods are implemented in *Detection* and *Decision* modules of SRF-IDS. According to reports [5, 108], DIS flooding attacks are the ones most commonly used and may affect the availability as well as the integrity of IoT systems. Therefore, designing and developing an efficient SRF-IDS to protect IoT networks from these attacks is currently an open problem.

As regards to other types of routing attacks, using SRF-IDS component alone cannot achieve protection from these attacks. Blackhole, rank and other complex routing type attacks require new methods of defence. For this reason, in our project SRF-OF component is introduced and implemented to mitigate these attacks. As SRF-OF needs information to operate, SRF-IDS was designed with the ability to collaborate with such components. SRF-IDS detectors are added with the responsibility to inform neighbouring monitored nodes with trust metrics collected by *Decision module*. Further details of how SRF-OF and SRF-IDS operate together were discussed in Section 4.2.

As far as the scalability of the proposed SRF-IDS is concerned, even in large networks good efficiency is expected. To ensure that, SRF-IDS detectors perform certain calculations

(e.g., packet sending rate and packet interval) and only if the metric of interest is above a threshold, node's traffic will be forwarded to the BR for further investigation (e.g., decision making).

4.3.4 Firewall and Other Design Decisions

The firewall inside the SRF-IDS root serves as an additional layer of protection. The firewall contains rules for blocking IP addresses of malicious nodes. Nodes are blocked only if the *Detection module* has information of malicious behaviour. In that case, a new rule with the node's IP is created and the node cannot send or receive data from the Internet.

As far as the placement strategy of SRF-IDS modules is concerned, a hybrid approach has been adopted. The centralized node (i.e. BR) stores signatures, analyses traffic, and detects attacks originating from the sensors or coming from the Internet. The decentralized nodes (i.e., SRF-IDS detectors), perform lightweight tasks such as monitoring and reporting network data to the BR. This placement strategy helps in capturing traffic and detecting attacks from all network segments. Furthermore, deploying detectors in close proximity to the sensors aims at detecting attack attempts faster and more efficiently rather than waiting the attack traffic to pass via the BR.

4.4 SRF-IoT Processes

The proposed framework consists of SRF-IDS, and an improved trust-based version of RPL protocol. The routing protocol is modified to consider trust values as method of evaluation for parent selection. We define the following terms:

Monitored network: The network that is being monitored by the SRF-IDS.

Monitored nodes: The nodes that belong to the network that is being monitored by SRF-IDS.

Neighbour: The node N_b is a neighbour of N_a only if N_b is in transmission (TX) range of N_a. That means N_b could provide a route to sink/BR.

In the following subsections, design details of various SRF-IoT processes are described in detail.

4.4.1 Collecting Information From SRF-IDS

The first and most important process of the system is the network information collection. SRF-IDS is responsible for this, using the deployed SRF-IDS detectors. Specifically, SRF-IDS detectors capture network traffic from monitored network and save packet metadata in a local database. Then, an algorithm runs to confirm if packets are forwarded or not to the next hop. The outcome of the algorithm is communicated with the devices belonging to the monitored network for further processing. Only monitored devices that are in TX range of SRF-IDS detectors will receive the metrics. Operating SRF-IDS in promiscuous mode in a different RPL Instance is a novelty that allows easy SRF-IDS deployment like a plug-n-play system. Additionally, no extra energy is consumed from smart devices in the monitored network to sniff any traffic. All the processing is done in SRF-IDS detectors and only useful metrics are transferred to monitored network. It is important to note here that nodes in the monitored network can be benign or malicious. SRF-IDS detectors will try to communicate with any type of device in their TX range because it is impossible to know which node is an attacker or not. We assume that SRF-IDS packets will be encrypted to avoid being exploited by attackers.

4.4.2 Calculating Trust

The calculation of trust value of a node until time T, occurs in this process. The formula to calculate the direct trust that device D_a holds for device D_b until time period T is given by $DT(D_a,D_b)_T$. The number of packets successfully transmitted between devices D_a and D_b until time period T is given by $PT_{ab}(T)$. The total number of packets forwarded by device D_b on behalf of device D_a until time period T is given by $PF_{ba}(T)$. Intuitively, the higher the number of packets D_b drops (i.e. $PT_{ab}(T)$ - $PF_{ba}(T)$) the lower the trust D_a has in D_b should be. In order to be able to compare between devices, we normalised this by dividing it to the total number of packets that D_b forwards. Therefore, now the higher the proportion of the number of packets D_b drops to the total number of packets forwarded by D_b (i.e. $[PT_{ab}(T)$ - $PF_{ba}(T)] / PF_{ba}(T)$) , the lower the trust D_a has in D_b should be. The outcome of this formula, i.e. $[PT_{ab}(T)$ - $PF_{ba}(T)] / PF_{ba}(T)$, ranges between 0 (if all packets are forwarded) and infinity (if D_a sends lots of packets, but none are forwarded). Normalisation was the next step so that we get a value between 1 and 0, with 1 corresponding to the former case (maximum trust) and 0 corresponding to the latter case (minimum trust). In order to achieve it, the following function has been used:

$$f(x) = \frac{1}{(1+x)} \tag{4.1}$$

The above function takes *x=0* to *f(x)=1* and $x = \infty$ to *f(x)=0*. The value achieved from *[PT_{ab}(T) - PF_{ba}(T)] / PF_{ba}(T)* needs to be scaled to reflect varying degrees of trust in different situations. For example, if a device is initially forwarding all packets it has high trust value. In a later moment, if it behaves maliciously and drops the packets, its trust value should be reduced. For this purpose, a weight factor w is added before applying the f function. A weight factor is added to the equation to punish or reward nodes that may change packet forwarding behaviour accordingly.

Based on the previous explanations, direct trust calculations are computed as follows:

$$DT(D_a, D_b)_T = \frac{PF_{ba}(T)}{PF_{ba}(T) + w \cdot [PT_{ab}(T) - PF_{ba}(T)]} \tag{4.2}$$

which is the result of multiplying the numerator and denominator of initial formula f by $PF_{ba}(T)$. Weight factor w can take the following values:

$$w = \begin{cases} 0.6, & \text{if node\_verified}_{\text{T}} \equiv 0 \text{ and PFI(t)} \equiv 0 \\ 0.8, & \text{if node\_verified}_{\text{T}} \equiv 0 \text{ and PFI(t)} > 0 \\ 0.85, & \text{if node\_verified}_{\text{T}} \equiv 1 \text{ and PFI(t)} \equiv 0 \\ 0.5, & \text{if node\_verified}_{\text{T}} \equiv 1 \quad \text{and PFI(t)} > 0 \text{ and } \text{PF}_{\text{ba}}\text{(T)} > \\ & \qquad\qquad\qquad\qquad \text{minimum\_fw} \\ 0.0, & \text{otherwise} \end{cases}$$

where *PFI(t)* represents the number of packets forwarded, sniffed by SRF-IDS, for the specified node at time t. To calculate the total number of packets forwarded until time T we use:

$$PF_{\text{ba}}(T) = \sum_{n=1}^{T} PFI(n) \tag{4.3}$$

The parameter *node_verified$_T$* means that a node is verified by SRF-IDS that is behaving normally until moment T. We use this indicator to increase the weight factor so that a node is trusted by benign nodes, while an unverified node has a smaller weight. The fourth case has a condition that w is smaller if node is verified, forwards packets, and the total number of packets forwarded are greater than the minimum number of forwarded packets (*minimum_fw*). The value of *minimum_fw* is 5, and it is used as an indicator to check if a node keeps forwarding packets after the initial verification. If a node is verified but the total number of forwarded packets are less than this parameter, weight becomes zero and trust is 100%. This ensures that a verified benign node will be fully trusted until it reaches the threshold *minimum_fw*. Then, weight is applied in the formula.

Weight factor plays an important role because trust value depends on the obtained behaviour of the node. The values were chosen after various experiments so that a fair and balanced value is calculated for each node by the OF. The general idea of weight factor is to use it in the OF formula so that high trust score is calculated for an unverified node that behaves normally, keep the trust score at same levels if a node keeps forwarding packets to avoid unnecessary parent switches, and assign low trust score once a node does not or selectively forwards data packets. The verification of a node is done with the help of SRF-IDS component. SRF-IDS detector keeps the following fields in a structure for a monitored neighbour node:

$\mathbf{PF_{ba}}$ = the total number of packets forwarded from device D_b on behalf of device D_a.

$\mathbf{A_{Da}}$ = a set of IP addresses that device D_a is usually sending the packets. The value is taken from the destination IP field in the packet.

$\mathbf{verifiedIP_a}$ = a field indicating if the IP address of a node is verified or not. Initially, all nodes are unverified until SRF-IDS verifies them. A node is verified if it forwards a packet to next hop.

4.4.3 Assessing Trust

Keeping up to date the trust value is significant to avoid interruption of network operation from malicious actors. Hence, SRF-IDS constantly evaluates the network and sends updated metrics to the monitored nodes. SRF-IDS sends the updated metrics in two modes; interval-based and trickle-based. Interval-based transmission occurs every 3 minutes from SRF-IDS detectors to devices in monitored network so that malicious nodes are identified in short time. Before packet transmission, SRF-IDS detectors verify that new metrics are actually available to send to monitored nodes else they skip the procedure.

Trickle-based transmission is based on RPL trickle timer implementation for transmitting DIO packets [49]. Trickle timer is a dynamic mechanism embedded into RPL that tries to minimise the transmission of RPL control packets. SRF-IDS detectors send packets with updated metrics each time trickle timer resets. Sending packets in two different modes ensures that SRF-IDS packets arrive successfully and at the proper time to monitored nodes to choose their best parent. Without any metrics, monitored nodes use MRHOF. Once a benign monitored node receives a packet from SRF-IDS, it calculates candidate parent's trust value based on the new measurements. Metrics for monitored nodes are stored in SRF-IDS detectors and they are reset every 15 minutes to avoid storage capacity problems.

Trust values have different scales as shown in Table 4.1. If trust falls below the *min_threshold* (less than 26) then a device is considered malicious and it is blacklisted. In addition, RPL local repair is triggered to allow nodes find new parents. In the opposite case, if a node is

blacklisted and trust value is above 50, node is removed from the list of malicious nodes. Initially, trust value 63 is assigned by default to all nodes joining the network. This is to allow nodes choose the best parent using other metrics, such as rank, until trust metrics become available. Lists with malicious nodes are stored locally in each benign node so that parent selection algorithm avoids blacklisted nodes. In case that a node stops attacking, the SRF-IDS will recalculate it's trust value. However, weights are adjusted and node will gradually become fully trusted.

Table 4.1 Trust scale

Values	Explanation	Actions
≥ 0 and ≤ 25	No Trust	Avoid node
≥ 26 and ≤ 50	Low Trust	Select only if no other option exist
≥ 51 and ≤ 75	Medium Trust	Select only if other nodes are below this rank
≥ 76 and ≤ 86	High Trust	Best candidate parent
≥ 87 and ≤ 100	Full Trust	Ideal parent, select without comparisons

4.4.4 Identifying Malicious Nodes

In case a node starts attacking the network using blackhole and rank attacks, immediately the SRF-IDS will notice this behaviour in the packet forwarding metric. Then, a packet is sent to the monitored network so that nodes will assign a low trust value for these neighbours.

The algorithm used by SRF-IDS detectors to detect the various routing attacks is shown in Figure 4.4. The procedure starts by enabling promiscuous mode in SRF-IDS detectors to start sniffing network traffic. Once a packet is captured, its packet type is checked and proper thresholds are used to determine if the network is under DIS flooding attack. Specifically, if the packet interval is above the predefined $threshold_{DIS}$ then SRF-IDS detector alerts the network administrator for possible DIS flooding attack, the node is reported to SRF-IDS root for blacklisting, and SRF-IDS detector continues packet sniffing. If packet interval is at normal levels, the packet is checked if it needs to be forwarded by the received node. This is done by checking the destination IP (*dstIP*) field to be different with the next hop IP (*next_hopIP*). In case that *dstIP* is equal to the received node IP, the packet is discarded and the procedure restarts. Otherwise, the *dstIP* is stored in a table for further checks. SRF-IDS detectors continue to sniff packets in order to decide if the neighbouring node actually forwards the packets. This is translated into the following condition: if the captured packet source IP (*srcIP*) equals to *next_hopIP* then it means the neighbour (next hop) node transmitted the packet. Next check is to validate destination node. If the stored packet IP (*stored_pkt_dstIP*) equals to the new destination IP (*dstIP*) then the packet is forwarded

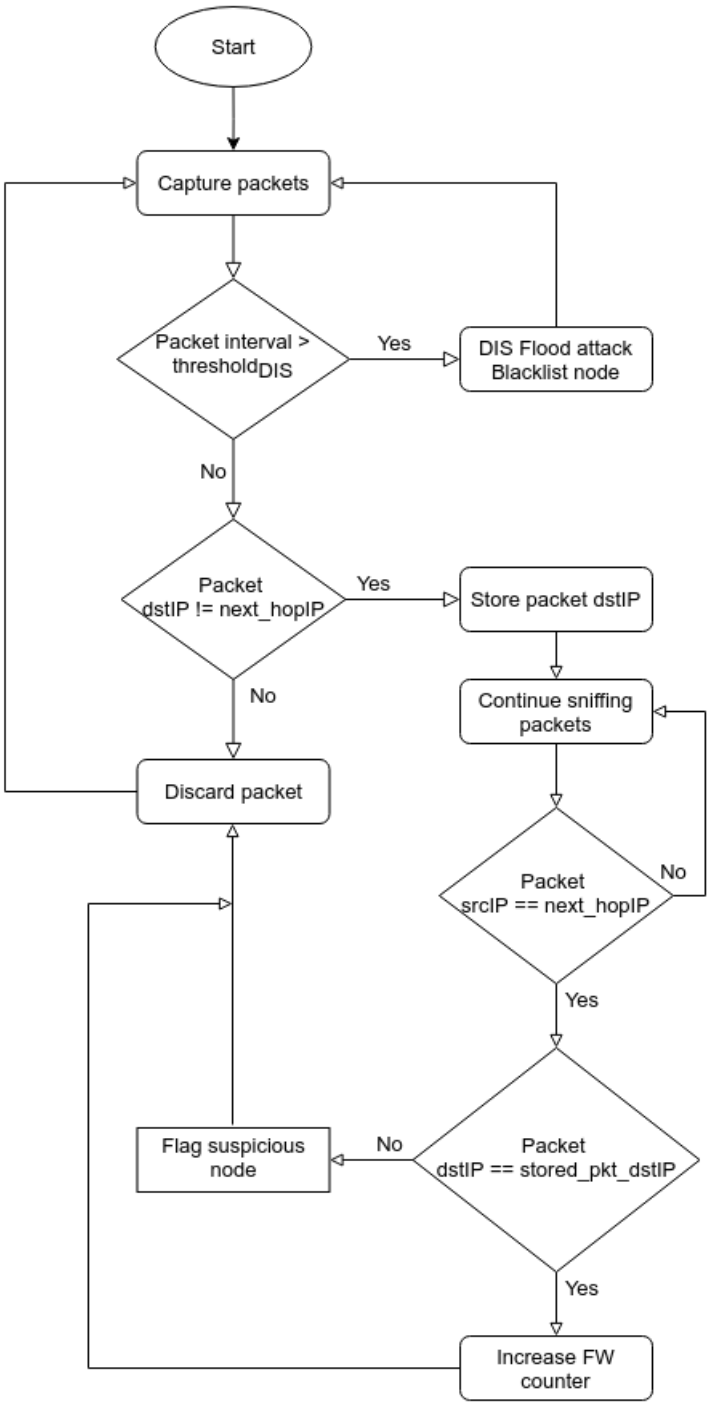


Fig. 4.4 SRF-IDS detectors operation flow chart

correctly, and the node's packet counter is increased by one point. In case *srcIP* equals to *next_hopIP* but the *dstIP* is not the expected, it means packet is not forwarded, and SRF-IDS

flags the device as possible malicious node. The last step is to discard the packet and continue capturing network packets for other nodes.

4.5 SRF-IDS Communication Component

This section presents the implementation of other SRF-IoT components. The full SRF-IDS and SRF-OF source code that has been developed as part of this work is available on Github repository [109].

Communication and routing among devices is handled by the RPL-lite protocol. SRF-IDS and monitored networks operate as usual in two different RPL instances. Nodes accept only packets destined for their RPL instance. Therefore, a solution was needed to allow SRF-IDS alert monitored devices about attackers. As discussed in Subsection 4.4.3, SRF-IoT scheme works with the help of an external IDS. A first prototype of SRF-IDS is presented and evaluated for detecting DIS flooding attacks in Section 4.7. However, an improved version with additional features is also implemented and discussed in Subsection 4.8. The new implementation includes the possibility of monitoring devices operating in a different RPL instance, detecting packet forwarding behaviour of a monitored node, and communicating with neighbouring devices. Apart from that, the detection mechanism for DIS flooding attacks is not affected.

Allowing SRF-IDS to communicate with monitored nodes is essential to enable SRF-OF calculate trust correctly. Trust is calculated by each node using the metrics received from SRF-IDS detectors. To achieve this, we implemented a new ICMPv6 control message in Contiki-NG. The implementation was done for SRF-IDS detectors and benign/malicious nodes which could be monitored. SRF-IDS detectors are able to send special packets to nodes in RPL Instance ID zero, and include the trust metrics mentioned in Subsection 4.4.2.

Algorithm 7 presents the actual implementation that is used by SRF-IDS detectors to send packets to neighbouring devices. As it is shown, SRF-IDS detectors iterate over their neighbours and create a buffer that contains the metrics. For each neighbour, SRF-IDS detectors send the IP address of the node, the *verifiedIP* flag and the number of forwarded packets. After adding the metrics into the buffer, SRF-IDS detector sends the ICMPv6 message using a custom RPL code to the neighbouring device. Moreover, detector sends the same information to SRF-IDS root for detecting potential blackhole attackers. If no information is available for a device, SRF-IDS detector does not send any packets to neighbours. Custom ICMPv6 packets are parsed by benign/malicious nodes to extract metrics and calculate trust using the appropriate formulas. We assume that these ICMPv6 packets are encrypted and only nodes participating in the monitored network can read its contents. Although malicious nodes can

receive and read packet contents, the metrics will be related to their parents. So, it will be useful even for them to select a benign parent based on metrics.

Algorithm 7 SRF-IDS communication function

```
1:  Function ids_output_to_benign(ip_addr) :
2:  for each N in nbr_table do
3:     linkaddr*lladr ← nbr_table_get_lladr(nbr_table,N);
4:     uip_ipaddr_t*ipaddr2 ← NULL;
5:     ipaddr2 → u8[0] ← 254; {Add address prefix}
6:     ipaddr2 → u8[1] ← 128;
7:     unsigned char *buffer ← UIP_ICMP_PAYLOAD; {Create new buffer}
8:     uint16_t pos ← 0; {Set RPL instance ID}
9:     buffer[pos++] ← 0;
10:    buffer[pos++] ← N → destParents; {Number of neighbours}
11:    for int j = 0; j < N → destParents; j++ do
12:       buffer[pos++] ← N → destIP[j];
13:       buffer[pos++] ← N → verifiedIP[j];
14:       buffer[pos+2] ← N → count_fw_packets[j];
15:       N → count_fw_packets[j] ← 0;
16:       N → verifiedIP[j] = 0;
17:       N → destIP[j] = 0;
18:    end for
19:    if N → destParents > 0 then
20:       uip_icmp6_send(ipaddr2, ICMP6_RPL, RPL_CODE_IDS_NORM, sizeof(buffer));
21:       uip_ipaddr_t addr2;
22:       if get_root_ipaddr(addr2)! = NULL then
23:          uip_icmp6_send(addr2, ICMP6_RPL, RPL_CODE_IDS2, sizeof(buffer));
24:       end if
25:    else
26:       printf("No information available");
27:    end if
28: end for
```

In cases where a monitored node receives metrics from multiple SRF-IDS detectors, appropriate mechanism is in place to handle these cases. For example, let M_a be the monitored node, IDS_a and IDS_b the SRF-IDS detectors, and D_a a candidate parent. If IDS_a sends a packet with metrics to M_a for device D_a that contains $verified_a$=*1* and $packets\_forwarded_a$=*5*, the M_a will store it normally. In a later moment, if IDS_b sends another packet that contains $verified_a$=*0* and $packets\_forwarded_a$=*0*, node M_a will aggregate the knowledge and calculate trust with the proper weight factor. Node M_a will consider its actual transmitted packets to check if $packets\_forwarded_a$=*0* is correct or not. A candidate parent is verified after consecutive notifications arrive by multiple SRF-IDS detectors.

4.6 Security Framework Objective Function (SRF-OF)

The trust concept is implemented as a new OF in RPL-lite protocol, called Security Framework Objective Function (SRF-OF). The SRF-OF algorithm for choosing the best parent is presented in Algorithm 8. Firstly, nodes check if neighbours are acceptable as parents. An acceptable node has low link metrics or path cost. Then, checks are done to avoid blacklisted and malicious nodes that were detected in previous attempts. A neighbour node with high trust value and smaller rank than current node's rank, is selected as parent (lines 17-20). In case Algorithm 8 reaches the last condition (line 27), it returns the parent with the lowest ETX value. The last condition as well as the whole SRF-OF implementation includes appropriate mechanisms to achieve stability in parent selection and to avoid unnecessary parent switches. It is important to have a stable network and minimise parent switches to reduce energy consumption and packet overhead.

The trust value for each neighbour is computed in Algorithm 9 which is based on the formula presented in Subsection 4.4.2. Specifically, monitored nodes receive the packet at time T from SRF-IDS, extract metrics from the packet for a specific neighbour, and store the measurements to the corresponding variables. For trust calculation, a node uses the formula shown in Subsection 4.4.2 which takes into account the actual number of packets sent to the neighbour until time T to the number of packets forwarded by neighbour and captured by SRF-IDS detectors until time T.

SRF-OF utilises the following metrics during best parent calculations: trust value, rank and ETX. A combination of these metrics would allow nodes choose the most trusted and reliable parent. SRF-OF is implemented in Contiki-NG for both benign and malicious nodes. SRF-IDS is using the default MRHOF as objective function while the rest nodes use SRF-OF.

4.7 Threshold-based Detection Module

The development of *Detection module* using thresholds is presented in this section. Firstly, design decisions are discussed and then, implementation steps are explained. Moreover, simulation configurations as well as metrics are introduced. An initial version of *Detection module* is evaluated using a simulator tool, and results are presented.

4.7.1 Design

One of the most important decisions is to define the most appropriate threshold values for the *Detection module*. In order to define the most suitable thresholds and identify DIS flooding attacks, results from Subsection 3.3.5 were utilised. Specifically, two thresholds are going to

Algorithm 8 SRF-OF algorithm

Input: Neighbour nodes nbr1 and nbr2 from nbr table
Output: Best neighbour/parent to route packets
Function $best\_parent(nbr1, nbr2)$:
int $nbr1\_is\_acceptable \leftarrow (nbr1 != NULL$ and $nbr\_is\_acceptable\_parent(nbr1))$;
int $nbr2\_is\_acceptable \leftarrow (nbr2 != NULL$ and $nbr\_is\_acceptable\_parent(nbr2))$;
if nbr1!=NULL **and** nbr1_is_acceptable **and** is_blacklisted(nbr1) **then**
  if nbr2_is_acceptable **then**
    return $nbr2$;
  end if
  return $NULL$;
else if nbr2!=NULL **and** nbr2_is_acceptable **and** is_blacklisted(nbr2) **then**
  if nbr1_is_acceptable **then**
    return $nbr1$;
  end if
  return $NULL$;
end if
if $((nbr1 \rightarrow trust\_value > nbr2 \rightarrow trust\_value)$ **or** $(nbr2 \rightarrow trust\_value < 38))$ **and** $(nbr1 \rightarrow rank < current\_rank))$ **then**
  return $nbr1$;
else if $(((nbr2 \rightarrow trust\_value > nbr1 \rightarrow trust\_value)$ **or** $(nbr1 \rightarrow trust\_value < 38))$ **and** $(nbr2 \rightarrow rank < current\_rank))$ **then**
  return $nbr2$;
end if
if $nbr1 \equiv current\_preferred\_parent$ **and** $within\_hysteresis(nbr1)$ **then**
  return $nbr1$;
else if $nbr2 \equiv current\_preferred\_parent$ **and** $within\_hysteresis(nbr2)$ **then**
  return $nbr2$;
end if
if $nbr\_link\_metric(nbr1) < nbr\_link\_metric(nbr2)$ **then**
  return $nbr1$;
else
  return $nbr2$;
end if

Algorithm 9 Trust calculation at time T

Input: Read buffer received from SRF-IDS
for each neighbour **do**
  $nbr \leftarrow$ Get neighbour node details from SRF-IDS buffer
  $node\_verified_T \leftarrow nbr \rightarrow verified_T$
  $node\_pkts\_forwarded_t \leftarrow nbr \rightarrow pf\_from\_ids$
  if $node\_verified_T == 0$ **then**
    if $node\_pkts\_forwarded_t == 0$ **then**
      $direct\_trust \leftarrow (nbr{\rightarrow}total\_packets\_fw/(nbr{\rightarrow}total\_packets\_fw + 0.6 * (nbr{\rightarrow}total\_packets\_tx - nbr{\rightarrow}total\_packets\_fw)))$
    else if $node\_pkts\_forwarded_t > 0$ **then**
      $direct\_trust \leftarrow (nbr{\rightarrow}total\_packets\_fw/(nbr{\rightarrow}total\_packets\_fw + 0.8 * (nbr{\rightarrow}total\_packets\_tx - nbr{\rightarrow}total\_packets\_fw)))$
    else
      $direct\_trust = 0$
    end if
  else if $node\_verified == 1$ **then**
    if $node\_pkts\_forwarded_t == 0$ **then**
      $direct\_trust \leftarrow (nbr{\rightarrow}total\_packets\_fw/(nbr{\rightarrow}total\_packets\_fw + 0.85 * (nbr{\rightarrow}total\_packets\_tx - nbr{\rightarrow}total\_packets\_fw)))$
    else
      $direct\_trust \leftarrow (nbr{\rightarrow}total\_packets\_fw/(nbr{\rightarrow}total\_packets\_fw + 0.5 * (nbr{\rightarrow}total\_packets\_tx - nbr{\rightarrow}total\_packets\_fw)))$
    end if
  end if
end for

be used in the IDS; packet interval and number of DIS messages. The packet interval variable, called $threshold_{time}$, is defined to be 30 seconds while the threshold for the number of DIS packets, called $threshold_{DIS}$, is defined to be 3 packets. Thresholds values were defined by taking the measurements of each metric from the simulations of several attacks. The $threshold_{detectors}$ is defined to be 3 so that a node is reported by detectors at least 3 times.

Regarding DIS flooding attack, it was configured to be launched by compromised nodes after 30 seconds so that the network is properly formed. Then, compromised nodes attack the network by sending 10 DIS messages every 30 seconds.

4.7.2 Implemented Algorithms

Following the previous design considerations, an implementation of SRF-IDS was done in ContikiOS and was tested using Cooja simulator. Particularly, SRF-IDS detector was configured to collect information from nodes and forward suspicious traffic to SRF-IDS root/BR. BR was programmed to receive reports from detectors and decide if malicious sensors exist or not in the network based on thresholds.

The algorithms for detection, information collection and reporting are given in Algorithms 10, 11 and 12, respectively. In our experiment, a sensor configured as SRF-IDS detector executed Algorithms 11 and 12, while a node configured as BR runs Algorithm 10. SRF-IDS detectors run Algorithm 10 to capture and calculate the number of DIS packets, other messages and packets interval from neighbouring nodes. Based on predefined thresholds, it will decide if they should be forwarded to BR or not. Algorithm 12 shows the thresholds used for deciding if a node will be reported to BR. This procedure is executed every 5 seconds by SRF-IDS detector. Based on the algorithm's output, SRF-IDS detector forwards a list with details of possible malicious nodes to BR. Algorithm 10 is executed by BR for malicious node detection. It is repeated every 3 seconds so that malicious nodes are detected fast enough. The detection algorithm considers a node as compromised if the reports sent by different SRF-IDS detectors exceed the threshold detectors variable which is equal to 3. This means a node is considered malicious if it is reported by detectors at least 3 times. In case that a node is in range of only one SRF-IDS detector, BR will count the number of reports sent by the specific detector to determine if a node is compromised or not.

4.7.3 Metrics

Simulations were created using the same configuration as described in Subsection 3.3.4. The only difference is that each scenario is repeated 5 times for cross-validation. Moreover, the DIS flooding attack was configured to be launched by compromised nodes after 30 seconds,

Algorithm 10 Centralised detection module

Function $CheckNodes(Monitored)$:
$Monitored[NumNodes] \leftarrow$ array with monitored Nodes
for each node in Monitored **do**
  $node.countDetect++$;
  if $node.interval \leq threshold_{time}$ **and** $node.totalDIS \geq threshold_{DIS}$ **then**
    if $node.countDetect \geq threshold_{detectors}$ **then**
      alarm Node compromised
    end if
  end if
end for

Algorithm 11 Local monitoring module: Updating metrics

On capturing a new packet:
for each node in Monitored **do**
  if $packet == DIS_{type}$ **then**
    $node.totalDIS++$;
  else
    $node.otherMes++$;
  end if
  $node.interval = clockNow - node.timestamp$;
  $node.timestamp = clockNow$;
end for

Algorithm 12 Local monitoring module: Reporting to BR

Function ids_detector_output(*Monitored*):
count_suspicious_nodes=0;
for each node in *Monitored* **do**
  if $node.packet\_interval \leq threshold_{time}$ and $node.totalDIS \geq threshold_{DIS}$ **then**
    $count\_suspicious\_nodes+ = 1$;
    $suspicious\_list.add(node)$;
  end if
end for
send_report_to_BR(suspicious_list, count_suspicious_nodes);

so that the network is properly formed. Then, compromised nodes attack the network by sending 10 DIS messages every 30 seconds. The metrics used for SRF-IDS evaluation are the following:

- *True positive (TP) rate:* Percentage of malicious nodes that are correctly detected as malicious.
- *False positive (FP) rate:* Percentage of normal nodes that are incorrectly detected as malicious.
- *IDS incidents:* Indicates how many times the SRF-IDS generated a warning for a malicious node.
- *Messages sent to SRF-IDS root:* Indicates the number of messages sent to SRF-IDS root by SRF-IDS detectors.
- *Precision:* A metric that presents the total number of malicious nodes that are correctly classified as malicious (TP) divided by the total number of nodes classified as malicious. It can be calculated based on (4.4).

$$Precision = \frac{TP}{TP+FP} \tag{4.4}$$

Each metric is calculated after taking into account results from all the repetitions of individual scenario.

4.7.4 Network Topology Selection

SRF-IDS should be able to detect compromised nodes in different environments. For this reason, various scenarios were tested with the help of Cooja simulator.

Specifically, three topologies were generated; Ellipse, Random and Linear. In each topology, 30 benign nodes, 4 SRF-IDS detectors, 6 malicious nodes and one SRF-IDS root were deployed.

Figure 4.5 illustrates the topologies created along with the different types of nodes. Benign nodes were deployed differently in each topology while rest nodes were in close proximity. After repeating 15 times, results showed that Random topology had the highest TP rate with 94.4% followed by Ellipse topology with 83.3% and Linear topology with 72.2%. Although deploying nodes randomly had better detection rate, the FP rate was as high as in Linear topology. Deploying nodes in ellipse had zero FP. Ellipse topology had the second best detection rate, had no FP, and it was easier to form it in Cooja because nodes remain

at fixed locations while in random topology nodes change location in every experiment increasing the complexity of the network. As a consequence, Ellipse topology was chosen for our experiments.

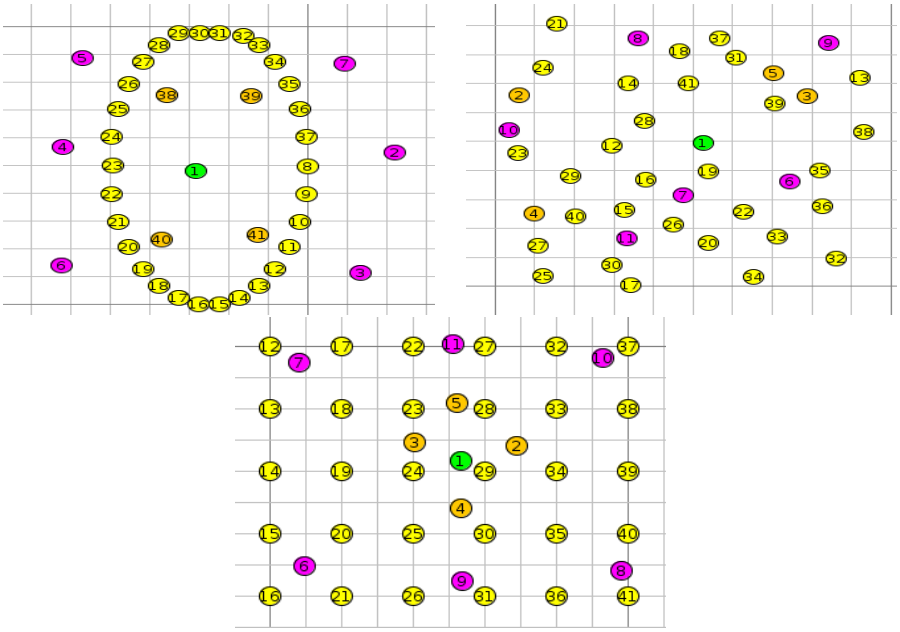


Fig. 4.5 Ellipse (upper left), Random (upper right) and Linear topologies. Colours for node types: green = server, yellow = benign node, purple= malicious node, orange = detector

4.7.5 Scenarios and Configurations

Evaluating an SRF-IDS requires testing various topologies. DIS flooding attack was configured to be launched by compromised nodes after 30 seconds so that the network is properly formed. In other words, compromised nodes attack the network by sending 10 DIS messages every 30 seconds. A large network with 30 benign nodes, a variable number of SRF-IDS detectors, 6 malicious nodes and one SRF-IDS root was created. The number of malicious nodes remains fixed, whereas the number of SRF-IDS detectors varies. Using a large number of sensors helps in evaluating the scalability of proposed IDS. Sensors were deployed throughout the network as depicted in Fig. 4.6. SRF-IDS detectors increase by one in each scenario. We created 10 scenarios that are repeated 5 times each for better accuracy. Each scenario is simulated for 25 minutes.

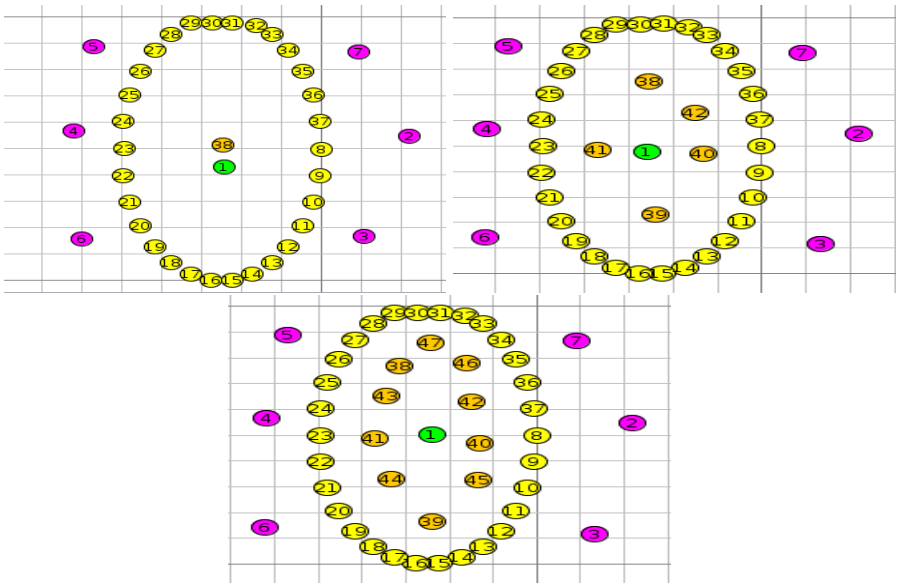


Fig. 4.6 Topology used in SRF-IDS simulations. Scenarios with 1,5 and 10 SRF-IDS detectors shown. They increase up to 10. Colours for node types: green = server, yellow = benign node, purple= malicious node, orange = detector

4.7.6 Evaluation Results

Results from 5 repetitions of each simulation are very encouraging. Figure 4.7(a) illustrates the TP and FP rates using different SRF-IDS detectors. As it is expected, the TP rate is 100% if 3 or more detectors are used in the network. This percentage falls to 97% and 83% when detectors are one and two respectively. However, FP rate increases if detectors are 8 or more. Looking more closely in the log files, we discovered that SRF-IDS detectors could be treated as malicious by other SRF-IDS detectors. This is the reason that FP rate increases when more SRF-IDS detectors are deployed in the network.

Figure 4.7(b) shows the number of warnings generated by SRF-IDS root. These warnings are the output of the *Detection module* algorithms. As it is depicted, the number is below 2,500 when SRF-IDS detectors are less than 3. However, this number is rocketed to over 7,000 when 3 or more detectors are deployed. The number remains at similar levels when detectors are between 5 and 8, with a small increase when 9 or more detectors exist in the network.

In conclusion, the SRF-IDS achieves high detection rate in almost all cases. This means that all 6 malicious nodes are detected even in large networks. Furthermore, results suggest that more than 3 and less than 8 SRF-IDS detectors should be deployed for best performance and low overhead. However, this could be different if more compromised nodes exist.

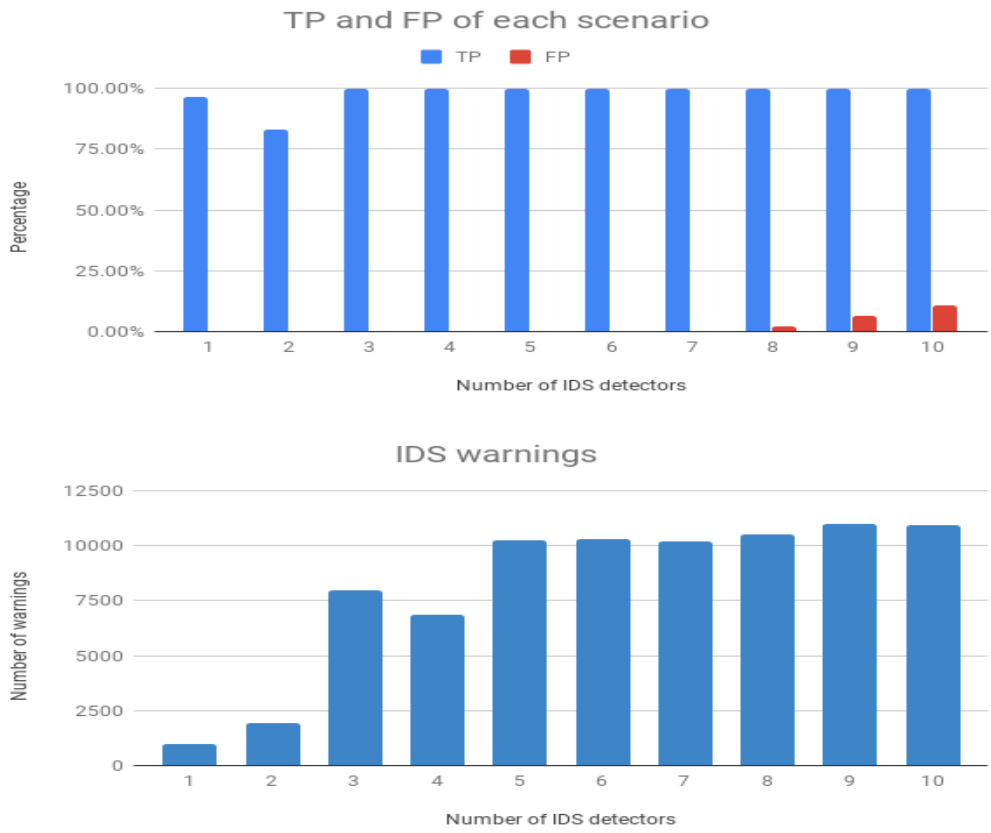


Fig. 4.7 a) TP and FP rates for each scenario, b) Number of warnings generated by SRF-IDS root

4.8 Improved Threshold-based Detection Module

An enhanced version of the *Detection module* is presented in this section. The problems and disadvantages of the first version of this module are discussed. Then, several fixes and features are added and explained. Also, scenarios and results are presented in the subsections below.

4.8.1 Problem Statement

The first version of *Detection module* introduced several issues. Firstly, a vulnerability existed in SRF-IDS detectors which could falsely treat other SRF-IDS detectors as malicious. Another problem is that attackers could attack SRF-IDS by advertising false rank, forcing SRF-IDS detectors to choose them as parents. This could disrupt the whole SRF-IDS operation and block SRF-IDS traffic. Apart from that, an issue with benign nodes and SRF-IDS detectors was discovered. SRF-IDS detectors could be selected as parents by monitored nodes requiring

further communication overhead with other neighbouring nodes. As a result, SRF-IDS detectors become part of the benign network, and while memory and storage resources are used only for network operation and not for monitoring purposes such as packet capturing and analysis that are needed by IDS.

4.8.2 Design Improvements

We have chosen to implement our threshold-based SRF-IDS along with DIS flooding attack in Contiki-NG mostly because it has an improved version of RPL protocol, better documentation and an active community which releases regular updates [34].

The SRF-IDS was implemented with the following extra functionalities:

1. Detectors were modified to always select SRF-IDS root as parent. Otherwise, malicious nodes may become parents of some detectors and this will affect the detection mechanism. This was achieved by setting a flag in DIS packet of RPL. In this way, when SRF-IDS detectors send a DIS packet, SRF-IDS root would recognise it and respond accordingly.

2. Detectors were configured to operate as leaf nodes in a DODAG network. A leaf node is a node that never becomes a parent. This is to ensure that they communicate only with BR.

3. A time-window concept is introduced to SRF-IDS. Measurements stored in detectors are reset at configurable time intervals (e.g., every 3 minutes). This time-window is expected to allow detection of multiple malicious nodes during simulation and to reduce false alarms.

4. Detectors are in promiscuous mode and they were programmed to communicate only with SRF-IDS root at intervals to keep communication link updated. This feature ensures the SRF-IDS root that SRF-IDS detectors stay online during operation.

5. A security vulnerability discovered in previous experiments described in Subsection 4.8.1 was that SRF-IDS detectors could be treated as malicious by other SRF-IDS detectors. Sending too many packets to SRF-IDS root may cause other detectors to treat this as an attack. Hence, we introduced configurable time intervals for the message reports and a new message code in the RPL protocol to distinguish traffic coming from the SRF-IDS detector. In this way, we expect to reduce the traffic overhead and minimize FP rate. We assume that packets are encrypted and the RPL message code cannot be sniffed from attackers.

6. Detectors were configured to begin detection after a certain time period (e.g., 1 minute). This was done to allow time for the benign nodes to form the DODAG network.

7. SRF-IDS root was modified to take measures and avoid unnecessary computations. Once a malicious node is detected, SRF-IDS root saves its IP address in the *blacklist*. When a packet arrives from a node that exists in the blacklist, the SRF-IDS root drops it. The blacklist can be reset at regular time intervals (e.g., every 10 minutes).

Regarding implemented code, Algorithms 11, 12 and 13 were implemented in Contiki-NG. Algorithm 13 below is an improved version of Algorithm 10. We introduced a new *detected* flag that is enabled once a node is detected. Additionally, the $threshold_{detectors}$ is decreased and set to 2. Experiments in ContikiOS showed that receiving reports from 3 detectors didn't improve the detection rate. Thus, we decreased the value to enhance SRF-IDS performance.

Algorithm 13 Centralised detection module

```
1: Monitored[NumNodes]←array with monitored Nodes
2: Function CheckNodes(Monitored):
3: for each node in Monitored do
4:     node.countDetect++;
5:     if node.packet_interval ≤ threshold_time and node.totalDIS ≥ threshold_DIS then
6:         if node.countDetect ≥ threshold_detectors and node.detected == 1 then
7:             alarm "Node compromised"
8:         end if
9:     end if
10: end for
```

With the introduction of the time-window on the metrics kept by the SRF-IDS detectors, a change in the number of suspicious nodes reported in large networks is expected. For this reason, the SRF-IDS root was modified to take decisions after receiving at least two reports for a specific node. Algorithm 13 is executed every 20 seconds by the SRF-IDS root instead of 3 seconds that was in the previous evaluation. Detectors execute the same Algorithm 11 when a new packet is captured. In addition, detectors report to SRF-IDS root, as shown in Algorithm 12, every 10 seconds instead of 5 seconds to minimize traffic overhead.

4.8.3 Scenarios and Topologies

Regarding simulation scenarios, we kept the same scenarios and topologies as described in Subsection 4.7.5. We also keep the same metrics to evaluate our system in both versions of Contiki.

4.8.4 Evaluation Results

Results from Contiki-NG simulation are illustrated in Figs. 4.8 and 4.9. According to Fig. 4.8, the TP rate for almost all scenarios is lower than before. A reason for that is because the detectors report fewer incidents than before and some attacks are missed. As the number of detectors increases, the TP rate increases as well. In regards to the FP rate, the highest value is 0.87%, which is very low in comparison to 10% that we had ContikiOS. Looking at Fig. 4.9, we see fewer incidents to be reported by SRF-IDS root than previously. The reason is that detectors report every 10 seconds to SRF-IDS root instead of 5 seconds. This minimized the traffic overhead in the node's network but lowered TP rate. Figure 4.10 compares the precision between the SRF-IDS implementation in ContikiOS and its improved version in Contiki-NG. Note that two implementations have similar precision in all scenarios. The only difference is that the one in Contiki-NG has even higher precision in cases with 8 or more detectors. That means that the SRF-IDS can handle traffic from more detectors and can detect malicious nodes in medium to large scale networks. The precision metric was calculated using (4.4).

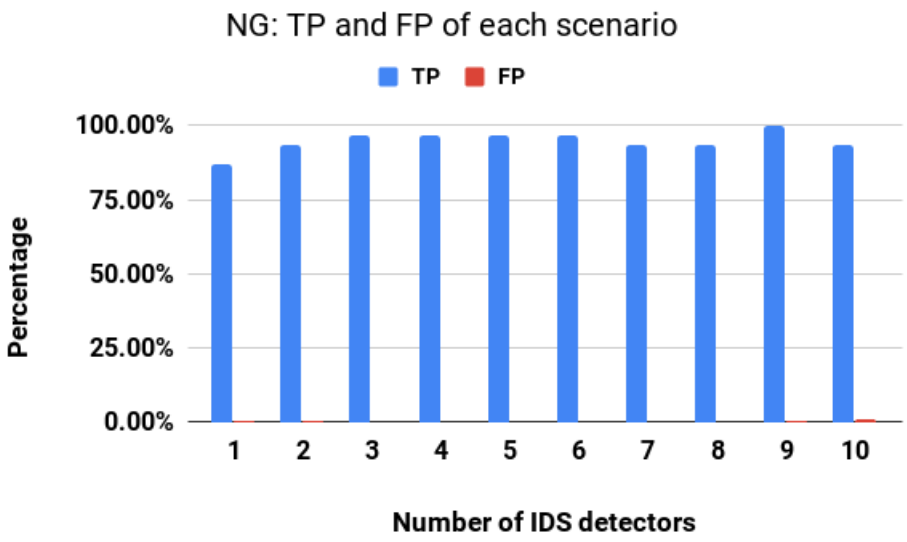


Fig. 4.8 TP and FP rates for each scenario in Contiki-NG

4.9 Chapter Summary

In this chapter, the design and implementation details of SRF-IoT framework are discussed. Firstly, an overview of trust concept as well as the high-level architecture of the two compo-

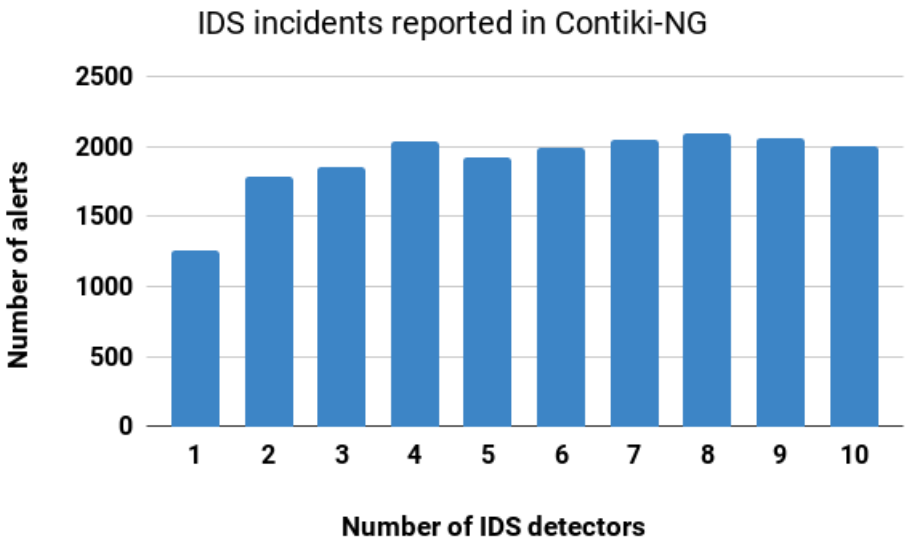


Fig. 4.9 Number of incidents reported by BR in Contiki-NG

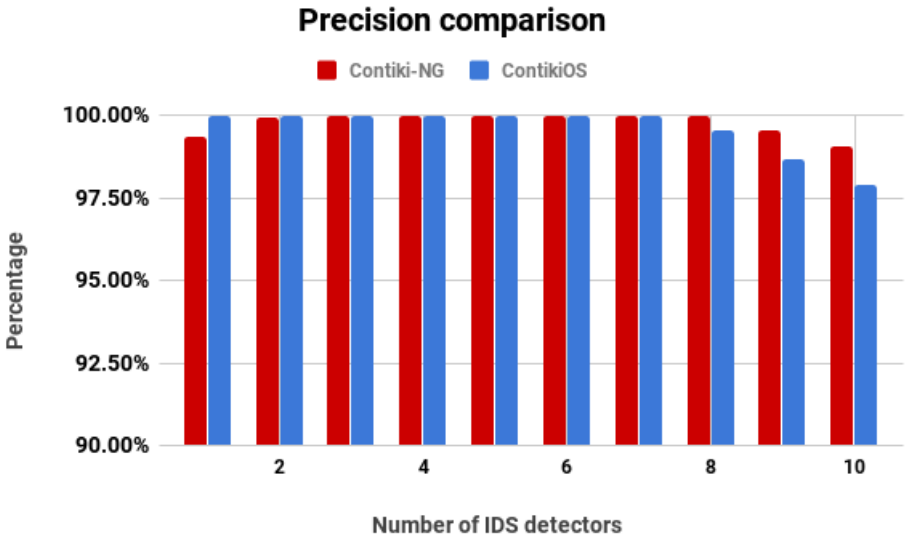


Fig. 4.10 Comparison of Precision between ContikiOS and Contiki-NG

nents, SRF-IDS and SRF-OF, are explained. Each component of the SRF-IoT framework is presented. As our work is implemented in a real-world OS, high-level algorithms were coded and explained in this chapter. Specifically, the communication protocol implemented for linking SRF-IDS with the SRF-OF running in monitored nodes is described. Moreover, the implementation of SRF-IDS detection module in both ContikiOS and Contiki-NG is

described. Simulation details and evaluation results are depicted. At the end, each operation done by SRF-IoT scheme is presented and discussed.

Chapter 5

SRF-IoT Performance Evaluation

In this chapter, the SRF-IoT framework is evaluated under three different cases. As a first case, SRF-IoT along with rank and blackhole attackers are deployed in a medium-size network. Then, a case with a combination of different routing attacks including rank, blackhole and DIS flooding attacks is presented in again a medium scale network. As a last extreme scenario, we tested SRF-IoT in a larger network in which the combination of the three routing attacks are launched. In each case, the specific scenarios, tools and metrics are analysed, and results are presented. Apart from those scenarios, new detection methods using ML models are also explored. Supervised learning algorithms are trained and evaluated with the help of cloud platforms. Evaluation results from ML model as well as the different scenarios are presented. Finally, we compare our solution against the related works.

5.1 Evaluation in Medium Network Under Rank and Blackhole Attacks

The proposed framework is evaluated under different environments and scenarios. The following section explains the main scenarios as well as the configurations used throughout simulations. Detailed information about deployed nodes and metrics is given.

5.1.1 Scenarios

We examined three main scenarios; normal, malicious using Minimum Rank with Hysteresis Objective Function (MRHOF) as OF, and SRF-IoT using SRF-OF as OF. Specifically, normal scenario is an environment without any attackers. Nodes are operating normally and are using the default MRHOF to choose a parent. In the malicious scenario, one or more attackers

exist, and all nodes are using the default MRHOF. This scenario is studied to understand the impact that rank and blackhole attacks have in the network. The SRF-IoT scenario is similar to the malicious one, but the difference is the OF that nodes are using. In that scenario, nodes are using the new implemented objective function called SRF-OF, in an environment where one or more compromised nodes exist. SRF-IDS operates in a different RPL Instance, and, therefore, nodes are using the default MRHOF. SRF-IDS detectors capture metrics only and do not help in parent selection in Normal and malicious scenarios.

Table 5.1 Node types and configuration

RPL InstanceID	**Node type**	**Description**
0	Sink/BR	Acts as a sink node. Receives messages and sends only UDP replies.
0	Benign node	Uses RPL-lite to create a mesh network and sends data periodically to BR.
0	Malicious node	Uses RPL-lite to join the network and advertises low rank (rank attack) and drops all incoming packets (blackhole attack).Also, it sends UDP packets like benign nodes.
1	SRF-IDS Root	Plays the role of sink in IDS and collects all the information from SRF-IDS detectors.
1	SRF-IDS Detector	Sniffs traffic of monitored network to detect malicious nodes. Stores information about messages exchanged and packets forwarded.

The different types of nodes used in our simulations are shown in Table 5.1. As it can be seen, SRF-IDS nodes have different RPL InstanceID than other nodes. In addition, Sink/BR and SRF-IDS Root play the role of sink for both networks. Benign and malicious nodes are configured to join the network and start sending UDP packets to Border Router (BR) once DODAG network is formed. However, malicious nodes start attacking the network after 2 minutes.

The number of nodes deployed in each scenario is presented in Table 5.2. Benign nodes are 30 in all scenarios while 6 malicious nodes are deployed in the malicious scenarios. A varying number of SRF-IDS detectors is deployed on the network to study and find the optimised number of SRF-IDS detectors to avoid the attackers. In normal and malicious scenario with MRHOF, SRF-IDS detectors are deployed with *TM* module disabled. This means, SRF-IoT framework is not operating as SRF-IDS does not provide monitored nodes with trust metrics. SRF-IDS is deployed in these two scenarios only for comparison purposes, and to gather measurements for generating results. The *TM* module, as mentioned in previous

Table 5.2 Number of node types in each scenario

	Sink/BR	IDS root	Benign nodes	Malicious nodes	IDS Detectors	Total
Normal scenario with MRHOF	1	1	30	-	5 to 15, Trust Module Disabled	37 to 47
Malicious scenario with MRHOF	1	1	30	6 BH and Rank	5 to 15, Trust Module Disabled	43 to 53
SRF-IoT scenario with SRF-OF	1	1	30		5 to 15, Trust Module Enabled	

sections, is enabled only when SRF-IoT scheme is evaluated and thus, monitored nodes use the SRF-OF.

5.1.2 Topology

IoT networks usually are deployed in mesh topology. Devices in mesh topology route packets with each other directly. In our case, the topology is shown in Figure 5.1. This topology uses the feature of RPL protocol that allows one DODAG network to have two different RPL Instances. It is a different architecture from our initial topology shown in the first version of SRF-IDS. This is because having two RPL Instances allows the operation of two networks at the same time. The RPL Instance with ID equal to zero, is the one that is monitored for suspicious activity and we call it *monitored* network. It has one sink/router that acts as BR, and several devices that can be benign or malicious.

SRF-IDS forms a second network inside the DODAG with RPL InstanceID equal to one, which contains the IDS root and IDS detectors. These two networks belong to the same DODAG but are two different RPL Instances. This helps IDS to distinguish packets coming from neighbour monitored network easily. Our approach takes advantage of RPL specification that allows operation of one DODAG with two different RPL Instance IDs [4]. This is already supported in Operating Systems (OS) such as Contiki-NG that implement RPL protocol, so no extra modifications are needed.

5.1.3 Configuration and Metrics

The settings and metrics utilised for evaluation purposes of SRF-IoT framework are discussed in this subsection. The metrics used are the following:

- *Median Packet Delivery Ratio (PDR):* Indicates the median value of the ratio of the total number of unicast packets received by BR up to the total number of unicast packets

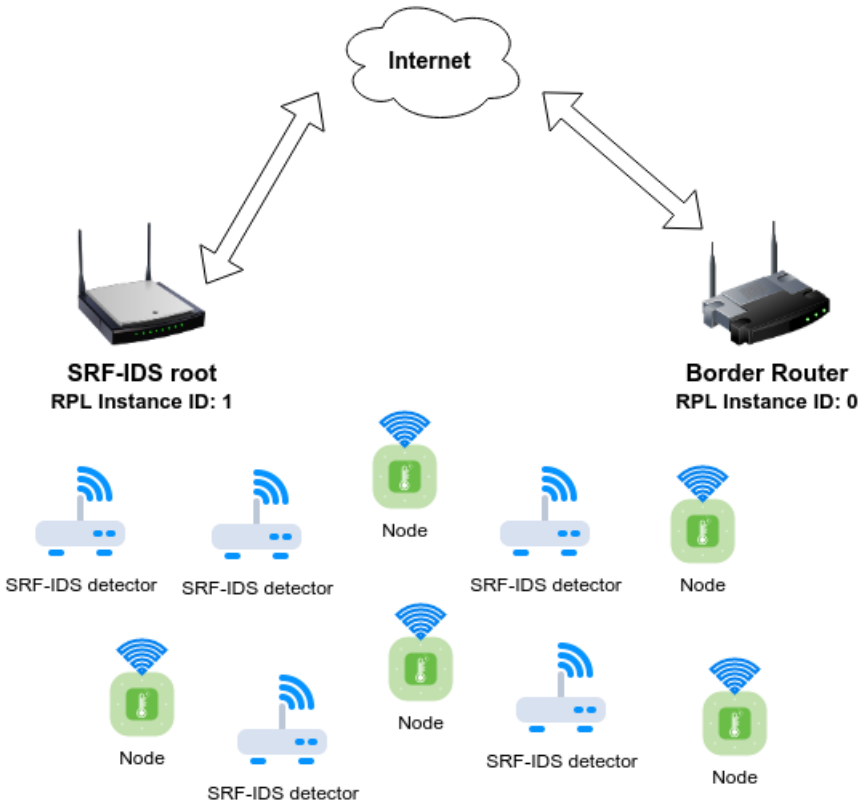


Fig. 5.1 Network topology used in simulations

generated by all benign and malicious nodes. It does not include UDP re-transmitted packets.

- *Median Parent Switch:* Presents the median value of the number of parent switches that benign and malicious nodes execute during the simulation. A parent switch happens to select a better route to the BR. Changing parent results to changing the rank of a node.
- *Median Packets Dropped:* Shows the median percentage of packets dropped by the attackers during the simulation. Malicious nodes drop all types of packets that normally should be forwarded to next hop.
- *Median IDS Packet Overhead:* Indicates the median percentage of SRF-IDS detector's packets sent to SRF-IDS root and the monitored network during simulation. The value is calculated from the number of packets exchanged in N repetitions.

Below, the mathematical definitions for the metrics are provided. Let

$$Median(X) = \begin{cases} X_{(\text{n+1})/2}, & \text{if n is odd,} \\ \\ \dfrac{X_{n/2} + X_{(n/2)+1}}{2}, & \text{if n is even} \end{cases} \tag{5.1}$$

be the median value of parameter X for cases when n (the total number of parameters) is odd or even number, and let X be a set of calculated values sorted from smallest to largest. Also, let

$$S_r = \frac{Rcvd}{\sum_{k=1}^{n} P_k} \tag{5.2}$$

be the packet delivery ratio for repetition r where $Rcvd$ is the total number of packets received at BR, k is the number of node sending the packets, n is the total number of nodes, and P_k is the total number of packets sent from node k. The *Median Packet Delivery Ratio (PDR)*, E[DRpkt], is given by:

$$E[DRpkt] = Median(S) \tag{5.3}$$

where r is the repetition number, m is the total number of repetitions for each simulation, and *Median(S)* is the median value of the set S. Packet delivery ratio from all repetitions are included in the set S to calculate the *Median(S)* value.

The *Median Parent Switch*, E[PS], is given by:

$$PS_r = \sum_{k=1}^{n} P_k \tag{5.4}$$

$$E[PS] = Median(PS) \tag{5.5}$$

where P_k the parent switch of node k, PS_r the sum of parent switches for repetition r, r the number of repetitions for each simulation, and *Median(PS)* is the median value of the set PS. The set PS contains all parent switches for all repetitions so that the *Median(PS)* value is calculated.

The *Median Packets Dropped*, E[Dpkt], is given by:

$$D_r = \frac{Drop_r}{\sum_{k=1}^{n} TA_k} \tag{5.6}$$

$$E[Dpkt] = Median(D) \tag{5.7}$$

where $Drop_r$ the total packets dropped in repetition r, TA_k the total number of packets transmitted from node k including multicast and unicast packets, D_r the percentage of packets dropped in repetition r, m the number of repetitions for each simulation, and *Median(D)* is the median value of set D. The total packets dropped from all repetitions are included in set D so that the *Median(D)* is calculated.

The *Median IDS Packet Overhead*, E[IDS], is given by:

$$Sent_r = \sum_{k=1}^{n} TA_k \tag{5.8}$$

$$I_r = \frac{\sum_{a=1}^{b} IDS_a}{\sum_{r=1}^{m} Sent_r} \tag{5.9}$$

$$E[IDS] = Median(I) \tag{5.10}$$

where k is the number of node sending the packets, n is the total number of nodes, TA_k in (5.8) is the total transmitted packets from node k including multicast and unicast packets, $Sent_r$ is the sum of packets sent in repetition r, a is the number of SRF-IDS detector, b is the total number of SRF-IDS detectors IDS_a is the total number of SRF-IDS packets sent from SRF-IDS detector a to SRF-IDS root, m is the total number of repetitions for each simulation, I_r the SRF-IDS packet overhead percentage in repetition r, and *Median(I)* is the median value of set I. The calculated values from all repetitions are added in set I to calculate the *Median(I)* value.

Table 5.3 Simulation configurations

Parameter	**Value**
Grid size	70x70
Topology	Random
Simulation time	60 minutes
Seed number	Random in each execution
Max MAC packet retries	3
Max buffered packets in MAC layer	20
Operating System	Contiki-NG 4.4
Simulator	Whitefield simulator

Regarding simulation configuration, the simulator called Whitefield framework provides a configuration file in which proper settings were defined. In that file, various simulation options can be defined by the user. The options used are shown in Table 5.3. Simulation execution time is defined to 60 minutes. Malicious scenarios using MRHOF are repeated 4

times while Normal scenarios are repeated 10 times because SRF-IDS is not active in parent selection. SRF-IDS detectors in those two scenarios are used only to capture traffic and calculate metrics using the measurements taken. Using more IDS detectors allow us to collect large sample for analysis. In addition, SRF-IoT scenarios using SRF-OF are repeated 10 times. This allow us to collect large sample for analysis. The seed number plays a significant role in simulations. It affects the behaviour of the nodes regarding processing and packet transmission times. Therefore, we use random seed in each repetition so that Whitefield simulator produces random results in each run. The simulator's default configuration of MAC packets retransmissions is 3 times while the maximum number of packets waiting in the buffer in MAC layer is 20.

5.1.4 Evaluation Results

Evaluation results from simulating the SRF-IoT scheme are presented in this subsection. From this point, we reference to malicious scenario using MRHOF as BHR scenario, normal scenario using MRHOF as Normal scenario and malicious scenario using SRF-OF as SRF-IoT scenario.

Starting with Figure 5.2, the *Median Packet Delivery Ratio (PDR)* of SRF-IoT framework is presented for scenarios where 5 to 15 SRF-IDS detectors are deployed. Overall, PDR in monitored network is kept at high levels with the help of SRF-IoT framework. This can be clearly identified from the figure as the median PDR starts from 90.8% when 5 detectors are deployed, then reaches the maximum of 92.8% in scenario with 7 detectors, and then PDR declines as the number of detectors increases, reaching the minimum of 82.4% with 15 SRF-IDS detectors. This is expected because multiple SRF-IDS detectors generate extra overhead in the network and monitored nodes may receive other nodes' metrics, not related to their direct neighbours. Therefore, monitored nodes drop more unrelated packets, reducing the PDR metric.

A comparison of the *Median PDR* of different scenarios is shown in Figure 5.3. Normal scenario has a median value of almost 95% in all simulated cases while in BHR scenario this percentage drops more than 15%. This big difference shows how attackers can degrade the performance of the network. Comparing the median number of PDR in SRF-IoT scenarios versus Normal scenarios, we can see that a small difference of a minimum 5% and maximum 13% exist. This means that as more SRF-IDS detectors are deployed, SRF-IoT framework's performance declines. On the other hand, those results indicate that SRF-IoT framework can assist nodes to avoid extra processing and energy overhead, and operate in the same levels as in normal scenarios.

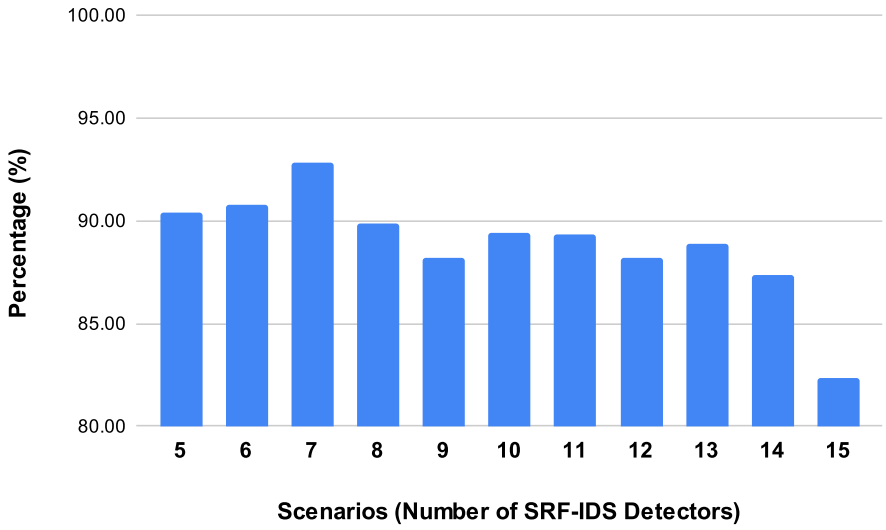


Fig. 5.2 Median Packet Delivery Ratio (PDR) results in SRF-IoT scenario after deploying 5 to 15 SRF-IDS detectors

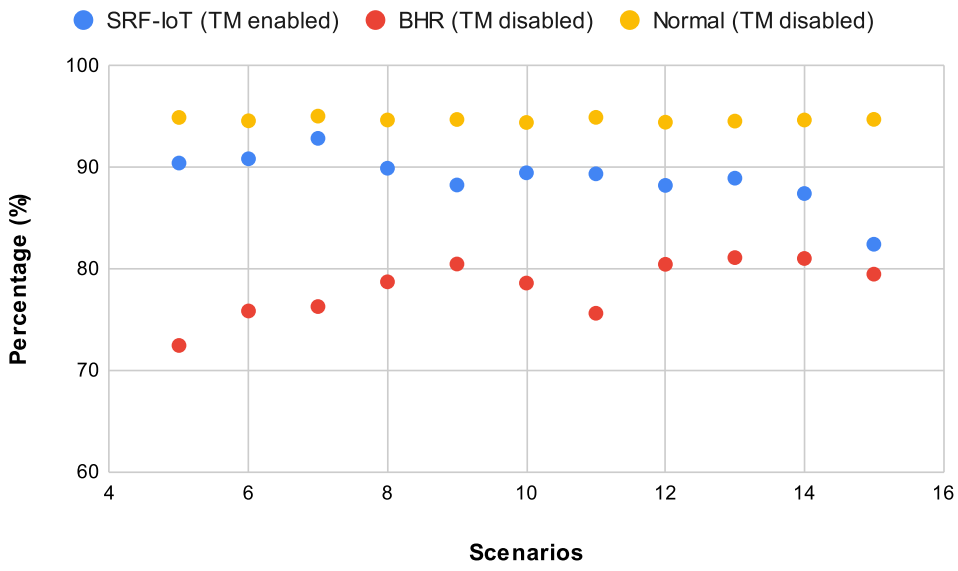


Fig. 5.3 Comparing Median Packet Delivery Ratio (PDR) per scenario after deploying 5 to 15 SRF-IDS detectors

Looking at the *Median Parent Switch* in Figure 5.4, results depict that our proposed framework requires less than 200 parent switches when deploying less than 15 SRF-IDS detectors. Specifically, the bar chart shows that SRF-IoT scheme has a median of 140 parent changes with 5 deployed SRF-IDS detectors, declining to 97 changes with 7 detectors, and

then going up to 258 when SRF-IDS detectors are 15. It is clear that having more than 12 SRF-IDS detectors, parent changes are almost doubled. An explanation is that as we increase SRF-IDS nodes, multiple detectors monitor similar nodes, and they send multiple metrics for the same monitored nodes. This could lead to high or low trust values which in turn leads to parent changes. The normal scenario indicates that parent switch should occur less than 3 times per node on average.

Figure 5.5 compares the values of *Median Parent Switch* metric for three different scenarios; SRF-IoT, BHR and Normal namely. Generally speaking, the scatter chart indicates that SRF-IoT framework achieves low parent changes, with a small increase from the levels of Normal scenarios. Looking more closely, using the SRF-OF, nodes change parents almost 3 times less than in BHR. Attacking the network causes approximately 600 parent changes on average while in SRF-IoT we have less than 190 switches and in Normal scenarios we have a median of 30 switches. The only exception is the case with 15 SRF-IDS detectors in which SRF-IoT sees a surge in parent changes. The high parent switch number explains the low PDR that we previously saw in that scenario. Assuming the selected parent is an attacker, we expect to have more dropped packets, affecting the PDR metric. This means malicious actors successfully affect the network performance using rank and blackhole attacks. Another conclusion is that SRF-IoT scheme drastically reduces parent switches in almost all cases, and even in the worst case it keeps the network more stable than in BHR scenario in which both PDR and parent switches metrics are high.

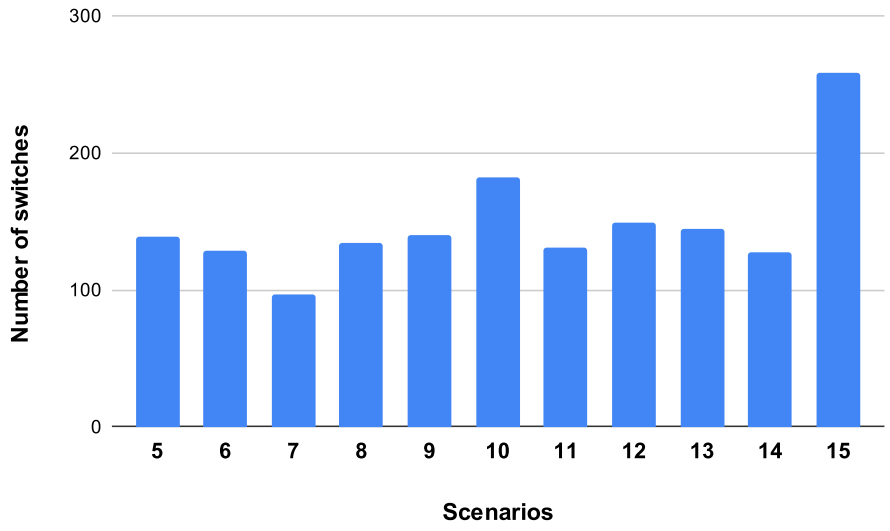


Fig. 5.4 Median Parent Switch results in SRF-IoT scenario after deploying 5 to 15 SRF-IDS detectors

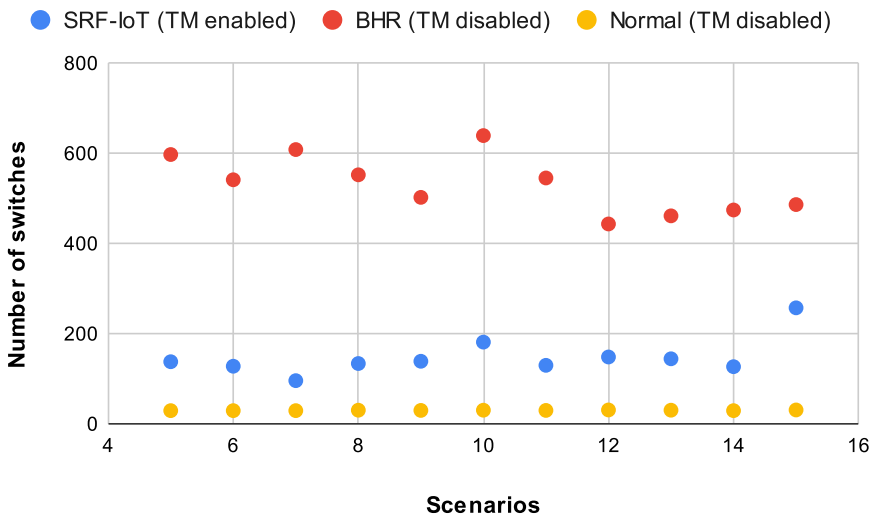


Fig. 5.5 Comparing Median Parent Switches per scenario after deploying 5 to 15 SRF-IDS detectors

The *Median Packets Dropped* obtained after deploying 5 to 15 detectors in SRF-IoT and BHR scenarios are displayed on Figure 5.6. Looking at BHR scenario, the median percentage of packets dropped is initially 38.6% when 5 SRF-IDS detectors are deployed. The percentage fluctuates around 40% until the scenario with 12 SRF-IDS detectors in which the percentage declines at 35%. This is related to the previous parent switch metric because in scenarios with higher packets dropped, more parent changes occur. Regarding SRF-IoT scenario, it has an increasing trend of dropping packets as the number of SRF-IDS detectors becomes bigger. A median of 14.2% of the packets are dropped when 5 SRF-IDS detectors are deployed, falling to 8.2% with 7 detectors and then goes up to 16.5% with 8 detectors. Then, the median value remains at the same level apart from the scenarios with 12 and 15 SRF-IDS detectors in which the median dropped packets climb up to 20.2% and 25% respectively. Therefore, our framework may assist the network to avoid blackhole attackers and reduce dropped packets.

As a last metric, the *Median SRF-IDS Packet Overhead* is illustrated in Figure 5.7. In all scenarios, SRF-IDS generates less than 2.5% traffic overhead in the network. The only case where the SRF-IDS packet overhead is relatively high is at the scenario with 15 SRF-IDS detectors. The median value reaches 2.2% because the monitored nodes are trying to avoid attackers - we have the highest median parent switches in this scenario- the number of dropped packets is also increasing, and thus, SRF-IDS detectors attempt to help monitored nodes by sending them trust metrics. All the previous metrics indicate that monitored network is greatly affected by attackers in that specific scenario. Generally, the number of packets

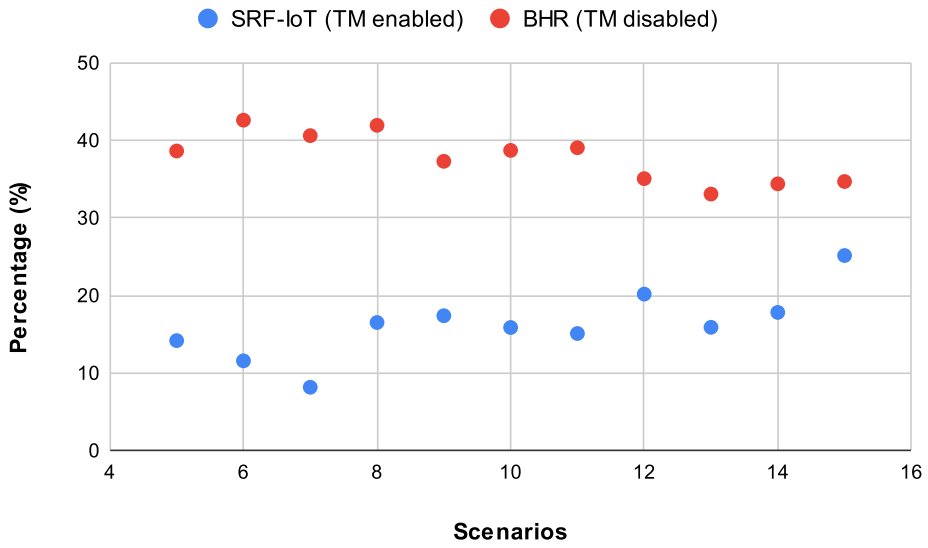


Fig. 5.6 Median Packets Dropped per scenario after deploying 5 to 15 SRF-IDS detectors

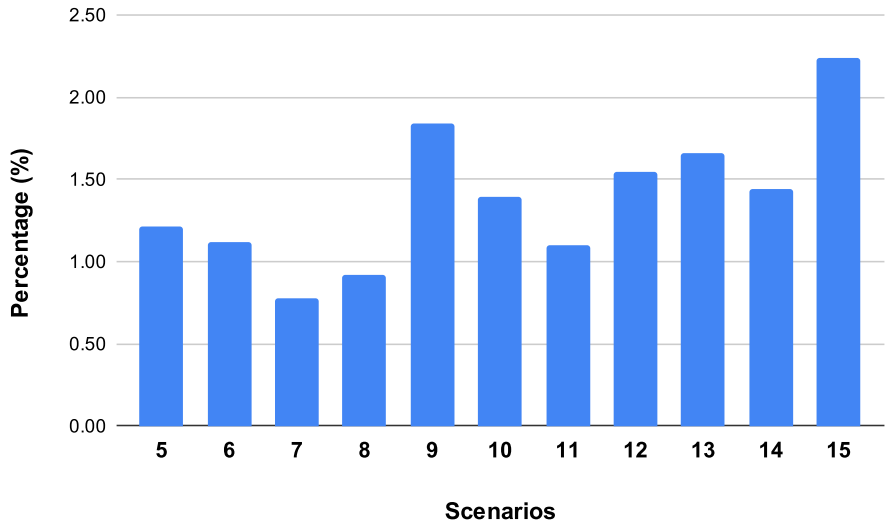


Fig. 5.7 Median SRF-IDS Packet Overhead in SRF-IoT scenario after deploying 5 to 15 SRF-IDS detectors

sent from SRF-IDS detectors depends on the number of monitored nodes. For example, SRF-IDS detectors which are deployed near multiple nodes of the monitored network, send more packets in order to update nodes with trust metrics. In our case, SRF-IDS nodes are randomly deployed in the simulated scenarios. As depicted in the column chart, SRF-IDS helps monitored nodes avoid attackers with very low packet overhead.

In conclusion, the experimental evaluation of SRF-IoT framework against rank and blackhole attackers showed higher PDR, lower packets dropped as well as lower parent switches in comparison with the malicious scenarios that SRF-IoT has TM disabled. Results indicate that the proposed framework can aid nodes to choose the proper nodes as parents and avoid the compromised ones. According to the evaluation results, deploying 7 SRF-IDS detectors in a network with at least 36 nodes generates the best results assuming the one sixth of them might be compromised. On the other hand, results depicted that deploying 5 to 10 SRF-IDS detectors still helps the IoT network to isolate and avoid attackers.

5.2 Evaluation in Medium Network Under Combined Rank, Blackhole and DIS Flooding Attacks

In this section, the evaluation of SRF-IoT framework in a complex attack scenario is presented. Specifically, the proposed scheme is evaluated in a simulation environment where malicious nodes attack the network using a combination of blackhole, rank and DIS flooding attacks. The aim of this experiment is to explore and evaluate the system under complex attacks.

5.2.1 Scenarios and Configuration

Similarly to the previous section's scenarios, Table 5.4 indicates the scenarios and the number of nodes deployed in each scenario. Three different cases were studied; normal scenario with MRHOF as OF, malicious using MRHOF as OF, and SRF-IoT using SRF-OF as OF. Similarly to previous experiment, all three scenarios have 1 Sink/BR, 1 IDS root and 30 Benign nodes. The difference with the experiments in previous section is the addition of 6 DIS flooding attacker nodes in malicious and SRF-IoT scenarios. As the table indicates, those nodes increase the total nodes deployed in those two scenarios. The rest number of deployed nodes remain the same as in the previous section. Normal scenario is used just for comparison purposes as it has the same configuration as before. DIS flooding attackers are configured to broadcast a batch of 50 RPL DIS packets every 30 seconds. The DIS flooding configuration is changed in comparison with the previous section because we wanted to increase the impact of the attack to the network. The attack is launched after the first simulation minute is passed.

The malicious scenario in which nodes use the default MRHOF is studied to understand and collect measurements regarding the impact of an attack that combines DIS flooding with rank and blackhole attacks. Moreover, the level of difficulty increases as there are two types of malicious attackers. SRF-IDS detectors are deployed to collect statistics, but TM module is not enabled in this specific scenario. SRF-IoT scenario is simulated to evaluate the

Table 5.4 Number of node types in each scenario

	Sink/BR	IDS root	Benign nodes	Malicious nodes	IDS Detectors	Total
Normal scenario with MRHOF	1	1	30	-	5 to 15, Trust Module Disabled	37 to 47
Malicious scenario with MRHOF	1	1	30	6 BH and Rank 6 DIS flooding	5 to 15, Trust Module Disabled	49 to 59
SRF-IoT scenario with SRF-OF	1	1	30		5 to 15, Trust Module Enabled	

implemented SRF-IoT framework that attempts to protect IoT devices from attackers using the SRF-OF and SRF-IDS. Having two different types of routing attackers, DIS flooding and rank/blackhole attackers, increases complexity of detection mechanism to identify and avoid them. Thus, SRF-IoT framework is evaluated in a complex scenario with larger number of attackers than any previous scenario. SRF-IDS detectors are deployed in fully a operational state with TM module enabled in this scenario.

Generally, SRF-IDS operates in a different RPL Instance using the default MRHOF as in previous experiments. Moreover, SRF-IDS is deployed in all scenarios but only in SRF-IoT scenario the TM module is enabled. In other two scenarios, SRF-IDS does not operate as part of SRF-IoT scheme, and therefore, SRF-IDS is deployed to allow us collect various measurements.

As regards to the simulation settings and metrics, for all the scenarios we have used almost the same configuration and metrics defined in Subsection 5.1.3. The difference with the previous section's experiments is the number of repetitions. In those new experiments, SRF-IoT scenario with SRF-OF, malicious scenario with MRHOF, and normal scenario are repeated 10 times.

5.2.2 Evaluation Results

Results from evaluating the SRF-IoT framework are discussed in this subsection. We utilise the same notation as before; the malicious scenario using MRHOF is referenced as BHR scenario, the normal scenario using MRHOF as Normal scenario and the malicious scenario using SRF-OF as SRF-IoT scenario.

The *Median Packet Delivery Ratio* (PDR) of SRF-IoT scenario is depicted in Figure 5.8. Generally, SRF-IoT achieves high median PDR in all scenarios. As it is shown, the median PDR starts with less than 90%, gradually rises up to 93% as the number of SRF-IDS detectors increases and then falls down to 84% when 15 SRF-IDS detectors are deployed.

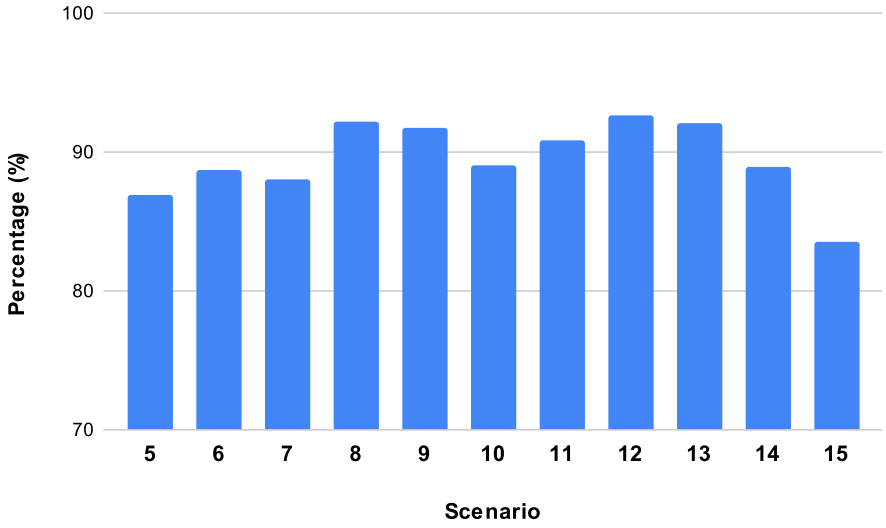


Fig. 5.8 Median Packet Delivery Ratio (PDR) for combined attacks in SRF-IoT scenario

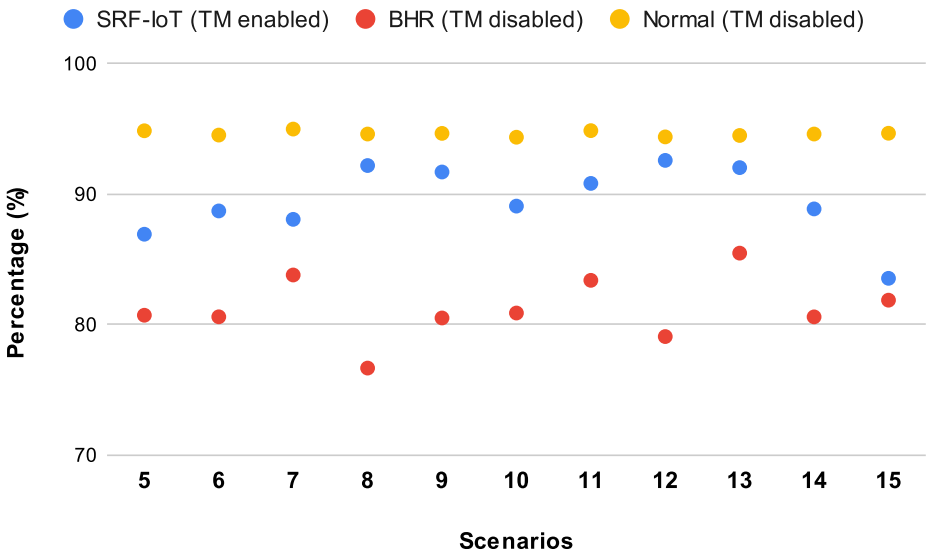


Fig. 5.9 Comparing Median Packet Delivery Ratio (PDR) for combined DIS flooding with rank/blackhole attacks

A comparison of *Median PDR* for the three different scenarios is depicted in Figure 5.9. As it can be seen, the median PDR values for SRF-IoT and BHR scenarios fluctuate around 80% and 90% as the SRF-IDS detectors increase, while in Normal scenario the median PDR is steady throughout the different simulated cases. The cause of those fluctuations is the randomness that simulator introduces in each simulation as well as the multiple attackers

which cause a random behaviour of nodes. It is obvious that BHR scenario has the lowest PDR in comparison with SRF-IoT and Normal cases. The highest difference in PDR between BHR and SRF-IoT scenarios is 16% when 8 SRF-IDS detectors are deployed. The outcome of this figure is that attacks clearly affect PDR of IoT network and by deploying SRF-IoT, smart devices may successfully limit the impact of complex attacks.

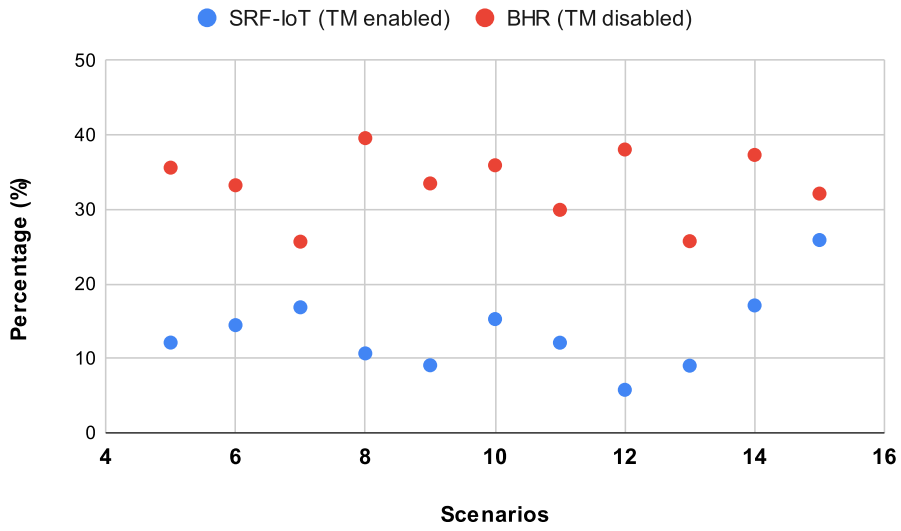


Fig. 5.10 Comparing Median Packets Dropped in complex attacks scenarios

Results comparing the *Median Packets Dropped* in SRF-IoT and BHR scenarios are depicted in Figure 5.10. It is clear that attackers in BHR scenario drop more packets than in SRF-IoT scenario. The fluctuations observed in PDR metric, exist also in packets dropped figures. In BHR scenarios where 7, 11, and 13 SRF-IDS detectors are deployed, the median dropped packets are around 28%. Moreover, as expected, the median PDR for the scenarios mentioned had the highest values. Looking at SRF-IoT scenario, in most cases the median packets dropped are less than 20% which proofs the good performance of the framework. The only case that SRF-IoT does not perform well is the one with 15 SRF-IDS detectors in which the median packets dropped are about 26%.

Another important metric is the *Median Parent Switch* and is presented in Figure 5.11. From an initial analysis, we observe that Normal scenario has a steady median parent switch of 31, while SRF-IoT framework works well by keeping the number of parent switches under 200 in most cases. In BHR scenario, attackers have huge negative impact on the network by dramatically increasing the number of parent switches by two and sometimes three times of those occurring in SRF-IoT scenario. In the most cases of SRF-IoT scenarios, the parent

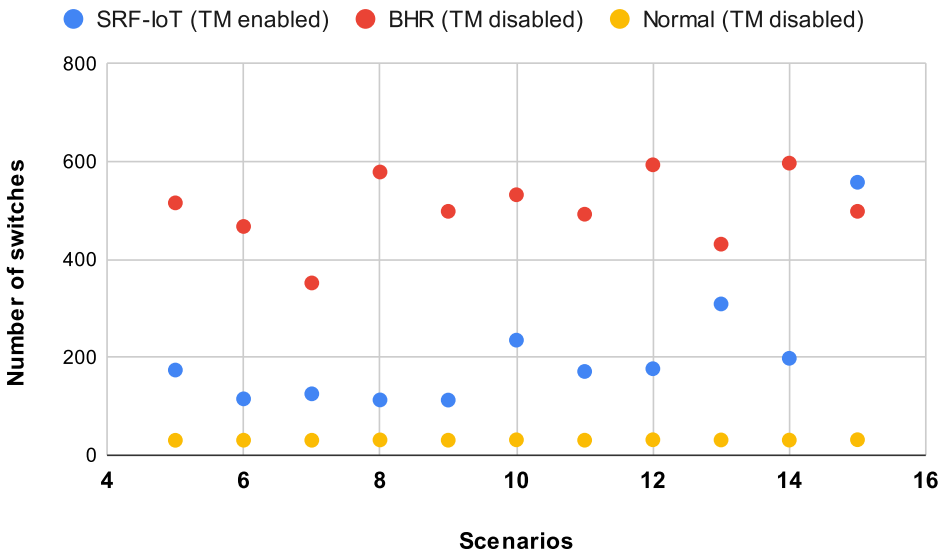


Fig. 5.11 Median Parent Switch in complex attacks scenarios

changes are less or equal to 200. This means that the proposed framework can actually help monitored nodes to have a stable network and avoid attackers.

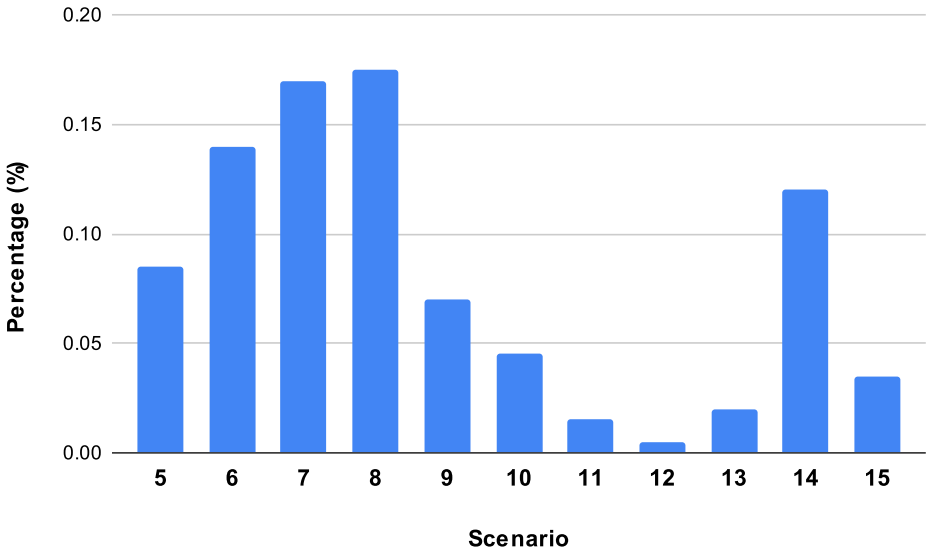


Fig. 5.12 SRF-IDS Median Packet Overhead in complex attacks scenarios

The last metric depicted in Figure 5.12 is the *Median Packet Overhead* introduced by SRF-IDS component during the various simulations. It is important to keep traffic overhead caused by SRF-IDS in low levels so that it does not affect the performance of the neighbouring

monitored network. The figure shows the overhead to climb up to a maximum median of 0.18% in the case with 8 SRF-IDS detectors which is very low. This means SRF-IDS component does not introduce extra traffic overhead. This happens because it sends the special packets only when information is available for a monitored node.

In conclusion, results from simulating the SRF-IoT framework in a network with complex routing attacks indicated that it can effectively improve network conditions. In comparison with the scenarios that SRF-IoT is deployed with TM disabled, higher PDR, lower packets dropped, and lower parent switches are recorded in all tested scenarios with 5 to 15 SRF-IDS detectors. However, to achieve an optimised performance, based on the analysed results 8, 9 or 12 SRF-IDS detectors should be deployed in networks with at least 42 nodes.

5.3 Evaluation in Large Network under Combined Rank, Blackhole and DIS Flooding Attacks

One of the challenges of a newly developed system is to deploy and evaluate it in multiple environments. In this section, the evaluation of SRF-IoT framework in networks with a large number of nodes is described. A scenario of over 120 nodes in a network that is attacked by various attackers is explored and results are discussed.

5.3.1 Scenarios

Multiple scenarios were used to simulate and compare the performance of the proposed system. Table 5.5 depicts the various scenarios along with the number of nodes in each scenario. As it is depicted, normal with MRHOF, malicious with MRHOF, and SRF-IoT with SRF-OF scenarios are created. Each scenario has 1 Sink/BR, 1 IDS root and 120 benign nodes deployed. Regarding malicious nodes, in the malicious and SRF-IoT scenarios, we deployed 12 blackhole and rank attackers as well as 12 additional nodes launching DIS flooding attacks. We have increased the number of benign and malicious nodes so that a larger network is simulated under multiple combined attacks. Following the same idea from previous experiments, SRF-IDS detectors are deployed with TM module disabled in both normal and malicious scenarios so that measurements can only be collected. In SRF-IoT scenario, TM module is enabled on SRF-IDS detectors to allow SRF-IoT scheme operate normally. The aim of those scenarios are to explore the behaviour and efficiency of our proposed SRF-IoT framework in large scale networks where DoS as well as other routing attacks are launched. This goal has been achieved with the use of Whitefield framework

Table 5.5 Number of node types in each scenario

	Sink/BR	IDS root	Benign nodes	Malicious nodes	IDS Detectors	Total
Normal scenario with MRHOF	1	1	120	-	23 to 27, Trust Module Disabled	145 to 149
Malicious scenario with MRHOF	1	1	120	12 BH and Rank 12 DIS flooding	23 to 27, Trust Module Disabled	169 to 173
SRF-IoT scenario with SRF-OF	1	1	120		23 to 27, Trust Module Enabled	

simulator because it allows the simulation of large networks using real-world hardware platforms.

The number of deployed detectors was defined based on the results from previous experiments. Considering that an average of 7 SRF-IDS detectors were needed to achieve good detection performance in previous experiments in Section 5.1 for a network with 36 nodes, we decided to deploy 23 to 27 SRF-IDS detectors for monitoring at least 144 nodes. Specifically, after several experiments in medium scale networks we defined that the ratio of monitored nodes to SRF-IDS detectors is 5:1. For this reason, we expect to see a similar ratio pattern in the experiments with large number of nodes in order to achieve good results. DIS flooding attackers are configured to send 50 DIS packets every 30 seconds. This configuration is similar to the one presented in Subsection 5.2.1.

5.3.2 Configuration and Metrics

Simulation settings for testing the SRF-IoT framework remain the same as those presented in Subsection 5.1.3. The only difference is the number of repetitions. Normal scenario with MRHOF, malicious scenario with MRHOF, and SRF-IoT scenario with SRF-OF are repeated four times in the experiments presented in the current section. The reason for reducing the number of repetitions is the increase in processing time that was needed to simulate the large number of nodes in each experiment.

Regarding the metrics, Subsection 5.1.3 describes most of them. Some additional metrics are the following:

- True Positive (TP) rate: The percentage of malicious DIS flooding attackers that are correctly detected as malicious, and reported by SRF-IDS nodes more than 2 times. The reporting value is defined by the threshold_detectors variable which is implemented in Section 4.8.

- False Positive (FP) rate: The percentage of benign nodes that are incorrectly detected as malicious.

The two metrics above are calculated as an average from the four repetitions per scenario. Moreover, the two metrics are used for examining the performance of SRF-IDS for detecting DIS flooding attackers in large networks. The mathematical definitions for above metrics are described below. The True Positive (TP) rate, E[TP], is given by:

$$TP_i = \frac{Detected\_MN}{Actual\_MN} \tag{5.11}$$

$$E[TP] = \sum_{i=1}^{4} TP_i \tag{5.12}$$

where i is the repetition number, $Detected\_MN$ is the number of malicious DIS flooding attackers detected more than 2 times by SRF-IDS detectors, $Actual\_MN$ is the number of actual DIS flooding attackers deployed in the scenario, and $E[TP]$ the average TP percentage that is calculated from the results of four repetitions. The False Positive (FP) rate, E[FP], is given by:

$$FP_i = \frac{Detected\_MN}{Total\_Nodes} \tag{5.13}$$

$$E[FP] = \sum_{i=1}^{4} FP_i \tag{5.14}$$

where $Detected\_MN$ is the number of incorrectly identified benign nodes as malicious nodes, $Total\_Nodes$ is the total number of nodes deployed in the scenario, $E[FP]$ is the average FP percentage that is calculated from the results of four repetitions.

5.3.3 Evaluation Results

The results produced from the simulated scenarios are presented in this subsection. The same notations are used to reference scenarios; malicious scenario using MRHOF is referenced as BHR scenario, normal scenario using MRHOF as Normal scenario and the malicious scenario using SRF-OF as SRF-IoT scenario.

Figure 5.13 presents the *Median Packet Delivery Ratio* (PDR) of SRF-IoT scenario. As it is depicted, the highest median PDR value is 52% and is achieved when deploying SRF-IoT framework along with 23 SRF-IDS detectors. As the number of SRF-IDS detectors increases, the percentages slightly falls by 7% and then climbs up to 48%, maintaining the same levels for 25 and 26 SRF-IDS detectors. In the last scenario with 27 SRF-IDS detectors, the median PDR drops down to the same levels as the scenario with 24 SRF-IDS detectors.

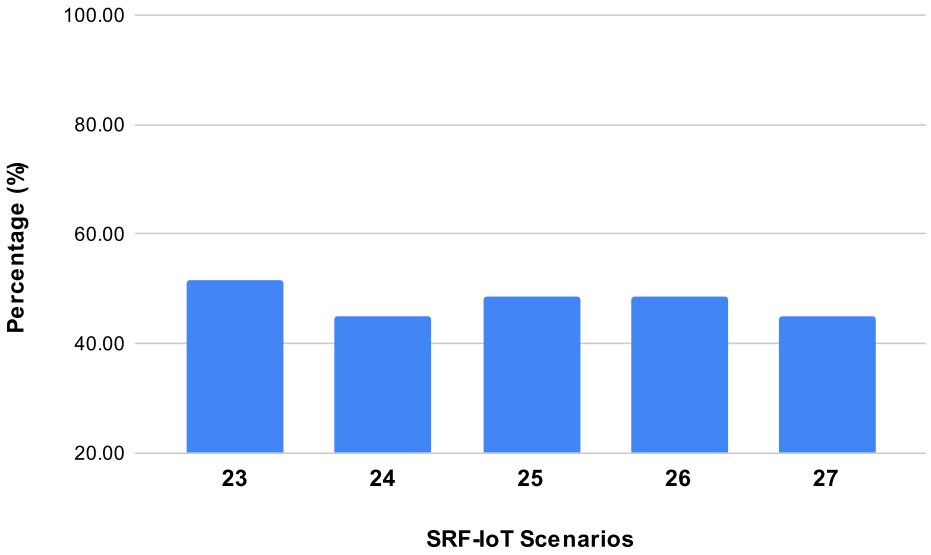


Fig. 5.13 Median Packet Delivery Ratio (PDR) in large networks for combined attacks

A comparison of *Median PDR* among the three different scenarios in which the number of SRF-IDS detectors varies, is showed in Figure 5.14. It is clear that the the median PDR in all cases falls below 80%, and in both SRF-IoT and BHR scenarios the recorded results have no big difference. Looking more closely, Normal scenario maintains a steady median PDR at around 74% in all scenarios. Regarding SRF-IoT scenario, the median PDR ranges from 45% to 52% which is very low in comparison with the Normal scenario. However, it shows a slightly better performance in most cases if we compare it with BHR scenarios in which the median PDR is around 41%. Two exceptions are observed in SRF-IoT scenarios; the scenario with 24 and 27 SRF-IDS detectors in which the median PDR in both BHR and SRF-IoT scenarios has less than ~1% difference. This means SRF-IoT fails to detect attackers and help monitored nodes to avoid them in those scenarios. All in all, the best results for SRF-IoT framework are observed in the scenario with 23 deployed SRF-IDS detectors, following the scenarios with 25 and 26 SRF-IDS detectors.

The *Median Packets Dropped* metric of SRF-IoT and BHR scenarios is depicted in Figure 5.15. Overall, in BHR scenario the packets dropped are more than 48% and remain at similar levels in all cases with slight increase. Regarding SRF-IoT scenario, the median remains at lower levels than BHR scenario but there are some fluctuations. As it is shown, SRF-IoT scenario starts with a median of 23%, declines dramatically to 9%, then surges to 24% and after a fall of 9% it climbs again up to 30%. On the opposite side, BHR scenario starts with a median of 40%, then slightly falls to 38% with 24 SRF-IDS detectors and then goes up to 40% in the rest cases. Therefore, in SRF-IoT scenarios we have less packets dropped in all

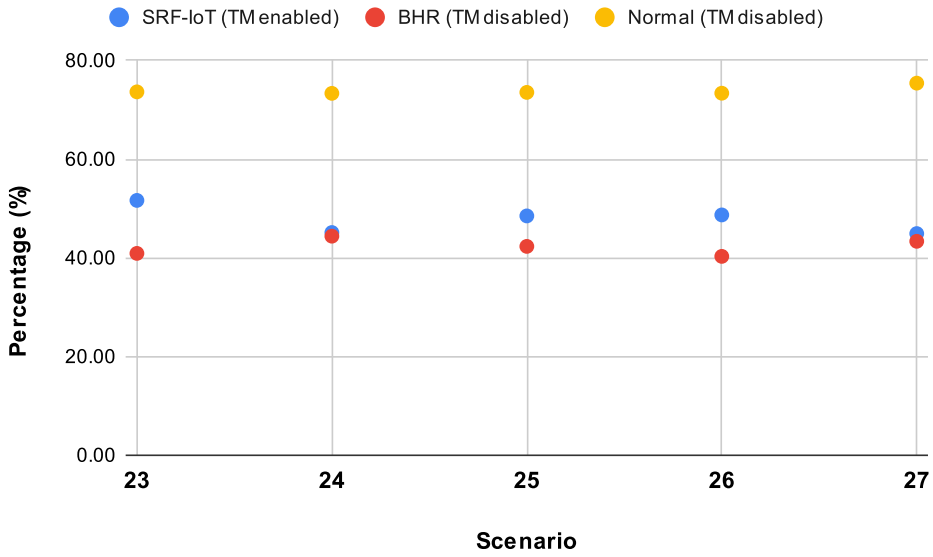


Fig. 5.14 Comparing Median Packet Delivery Ratio (PDR) in different scenarios for combined attacks

scenarios which means, if we consider also the PDR metric, the proposed framework works as expected.

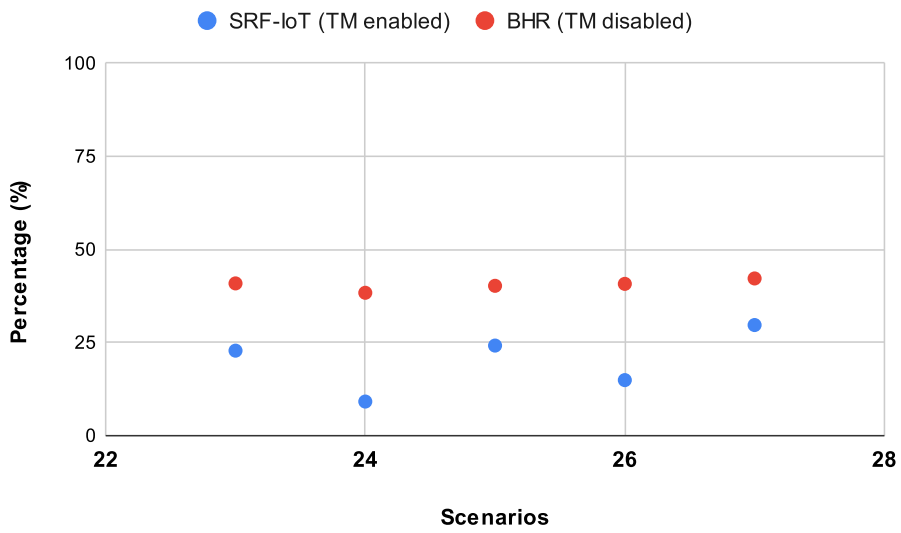


Fig. 5.15 Median Packets Dropped in large networks with complex attacks

Another important metric is *Median Parent Switch*. In Figure 5.16, the results from the three scenarios is presented. From a first analysis, the number of parent switch in both SRF-IoT and BRH scenarios is relatively high in comparison with Normal scenario. However,

SRF-IoT shows a small decrease in the numbers which means the framework could help nodes avoid attackers with the proper number of deployed detectors. In the Normal scenario, a starting median of 274 parent changes occurs with 23 SRF-IDS detectors and after a small increase to the metric using 24 detectors, the number remains at similar levels in the rest cases. Looking at SRF-IoT scenario, the median parent switches start from 3567 using 23 SRF-IDS detectors, then the median reaches 4643 in the next case and in the next two cases it declines until it reaches 3248 parent switches. The last case of SRF-IoT scenario with 27 SRF-IDS detectors shows a dramatic increase of 5277 parent switches. As regards to BHR scenario, a significant rise in parent switches is depicted with a minimum median of 6655 and a maximum value of 8648. As already mentioned in previous sections, SRF-IoT does not have TM enabled in BHR and thus, the fluctuations observed in the simulations is caused by the random behaviour of the nodes. Summarising, SRF-IoT can still help an IoT network to reduce parent switches and keep a network stable by deploying 23 to 26 SRF-IDS detectors. This is proved by the results of median PDR, packets dropped and parent switches.

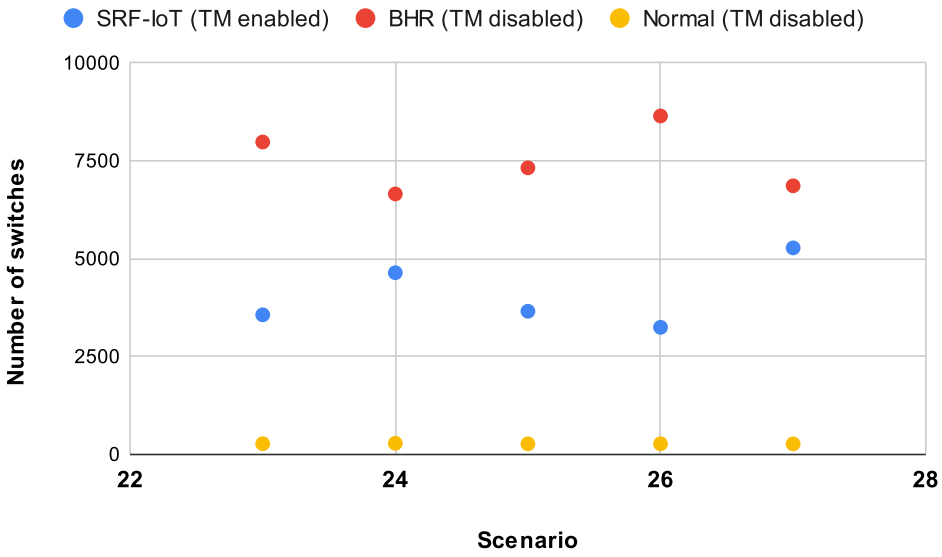


Fig. 5.16 Comparing Median Parent Switch of three scenarios under complex attacks

Results for the *SRF-IDS Median Packet Overhead* are shown in Figure 5.17. It is obvious that SRF-IDS component of SRF-IoT framework produces very low traffic overhead in the network during its operation. The median packet overhead is less than 0.25% in all cases while in two cases the median value falls down to 0.07% and 0.12% when 24 and 26 SRF-IDS detectors are deployed respectively. Hence, SRF-IoT framework does not generate too many packets to interfere with the monitored network.

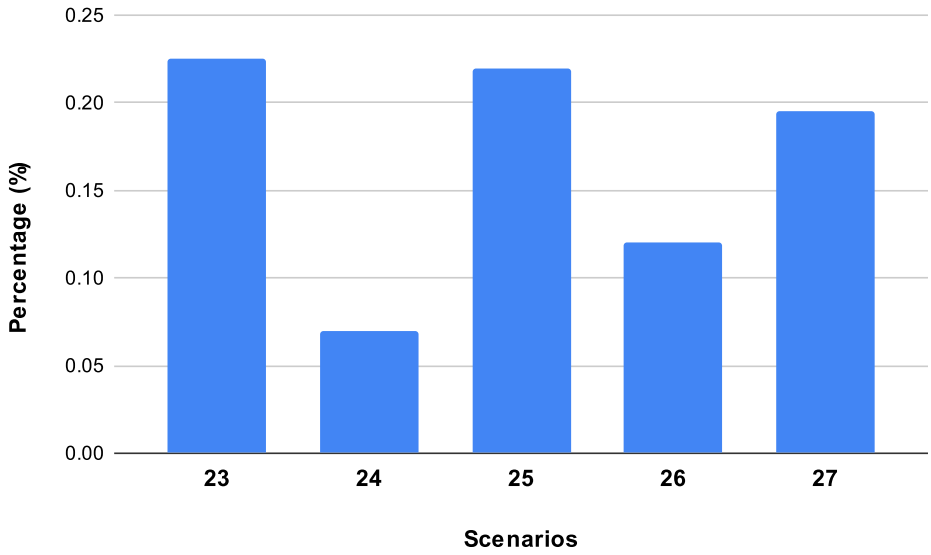


Fig. 5.17 SRF-IDS Median Packet Overhead in large networks under complex attacks

The aim of SRF-IDS component of SRF-IoT framework is to identify DIS flooding attackers as shown in Chapter 4. Thus, we find useful to evaluate the component in the scenario with large scale network. A metric used to examine the performance of SRF-IDS for detecting DIS flooding attackers is the *TP and FP* rates per scenario. Figure 5.18 presents the average TP and FP rates obtained by examining the results from four repetitions. The TP trend starts with an initial high value of 87.5% and then, as the number of SRF-IDS detectors increases, the TP percentage declines, reaching a minimum of 50%. Regarding the FP rate, it starts with a very low value of 1.4% and as the number of SRF-IDS detectors rises up, the value reaches a maximum of 3.4%. Therefore, results indicate that SRF-IDS can effectively detect DIS flooding attackers with an average TP of 87.5% and relatively low FP rate.

To sum up, the evaluation of SRF-IoT in a large network with over 120 nodes showed that it can improve network performance and assist monitored nodes to avoid malicious actors. Several metrics depicted that it is a challenge to avoid complex attacks in large networks and thus, the improvement from BHR scenarios was minimised. However, the deployment of less than 27 SRF-IDS detectors, and specifically 23 SRF-IDS detectors, showed that it can actually detect DIS flooding attackers efficiently, and enhance PDR performance by avoiding routing attackers with the minimum number of parent switched. Consequently, 23 SRF-IDS detectors are needed to effectively detect attackers with SRF-IoT framework, defining the ratio of monitored nodes to SRF-IDS detectors to be 6:1.

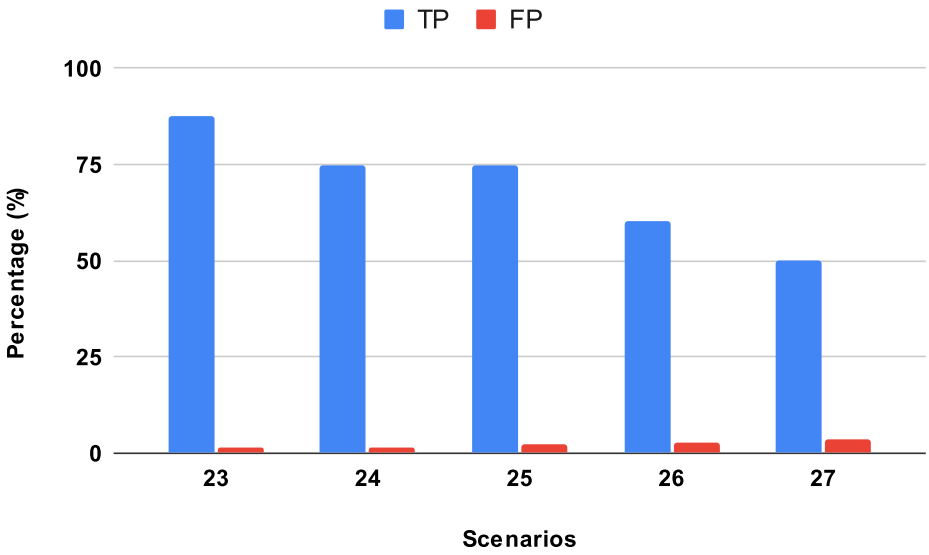


Fig. 5.18 SRF-IDS TP and FP per scenario

5.4 Machine Learning SRF-IDS Detection Module

The latest advancement in SRF-IDS component is the addition of a Machine Learning (ML) feature in the Detection module that will allow detecting unknown attacks. The novelty of our ML approach is that ML model is trained using the datasets created from simulations presented in previous sections. In this way, ML model is trained using more realistic data. ML algorithms can be used to train a model for predicting and classifying network traffic. Our main goal of creating an ML-based Detection module for SRF-IDS is to detect known complex attacks such as rank and blackhole attackers as well as unknown attacks. As we have seen in previous sections, simple routing attacks such as DIS flooding can be detected by SRF-IDS component using thresholds. Moreover, other routing attacks such as blackhole and rank attacks can be mitigated with the use of trust-based OF in SRF-IoT framework. However, a combination of routing, flooding and other types of attacks may bypass our trust-based SRF-IoT scheme and cause serious damage to the network. Therefore, having an embedded ML module trained to detect complex attacks will significantly enhance SRF-IoT's detection performance.

Nowadays, many ML algorithms exist in the literature for training a model from existing datasets. In this project, Google AutoML [110] and Microsoft Azure ML [111] are used to train our ML model which will be embedded into SRF-IDS TM module. Google AutoML is an automated and simple to use tool that provides researchers the opportunity to study and deploy different ML approaches. It has the possibility to scale up an ML model based on the

needs of the user. Moreover, it automatically chooses the most appropriate algorithm based on the dataset available. A similar ML as a service provider is MS Azure ML. This service allows users to generate and handle custom ML solutions. In addition, it assist users with extending, and deploying their workloads to the cloud. An advantage of MS Azure ML is that it gives the flexibility to the user to choose a specific ML algorithm from a list of well-known supervised, unsupervised or semi-supervised algorithms. Both platforms are explored so that the one with the best performance will be adopted in SRF-IoT framework.

5.4.1 Design

The SRF-IDS component of SRF-IoT scheme at its current design does not detect unknown attacks. It was designed using threshold and trust-based mechanisms to detect DIS flooding, rank and blackhole attackers. For this reason, an ML-based module would be a great feature to add and improve its detection performance. Learning from the current behaviour of the nodes will allow SRF-IDS to later identify known and unknown attacks, achieving high detection rates. As we have already examined and evaluated SRF-IDS in several simulations, the results obtained from the scenario with 30 benign nodes in Section 5.1 have been used as the basis of the learning procedure for the ML-based module. Specifically, packet capture (pcap) files generated by Whitefield framework for the referenced scenario were used. A pcap file contains all network packets exchanged from a specific node. Creating a realistic dataset would need realistic packet captures. Thus, only the pcap files of SRF-IDS detectors were collected and analysed during the learning procedure. Any packet sniffed by a SRF-IDS detector is recorded into its pcap file.

Through the learning procedure, it was discovered that configuration used for each simulated scenario included specific IDs for each deployed node. For example, if node with ID 3 is an attacker, ML model will learn that attacker is always the node with ID 3. In order to avoid this problem and allow the ML model to intelligently detect malicious nodes based on their behaviour and not based on their IDs, a different ID should be assigned at each malicious node in the test and training datasets. If we had the same node ID in both training and testing sets, the ML model would give wrong prediction results.

A new simulation configuration was created with the same parameters described in the previous chapter. The only difference is that node IDs are randomly assigned to attackers so that nodes have different IDs from the first configuration. Then, the new simulation configuration was used in the Whitefield framework simulator to evaluate the SRF-IoT. Scenarios were repeated 10 times; the same number of repetitions as in previous simulations. The resulting pcap files from this experiment are used to create the testing dataset.

5.4.2 General Approach

The approach followed for building the ML model is described in this section. All processes described in the subsections below are depicted in Figure 5.19.

Dataset Creation

The first step before training and deploying an ML model, is to produce a dataset for training the model. In order to create a dataset, we have used the network traffic produced during our simulations explained in Section 5.1. Whitefield framework has the option to export pcap files for each node of a simulation. Regarding SRF-IDS detectors, they had promiscuous mode enabled in all the experiments so that every packet received in their RX range was recorded in the pcap file. A large number of pcap files were generated during each simulation. The next step is to collect only the pcap files of SRF-IDS detectors for each simulated scenario. This was done because the general concept is to create an ML-based module for SRF-IDS using a realistic approach. Achieving that, required us to collect data from devices that actually sniff traffic, and avoid using the simulator's functionalities. Therefore, the ML model is trained and evaluated using the packets received by SRF-IDS detectors.

Pre-processing

After collecting pcap files from SRF-IDS detectors, all files had to be merged into one. For each repetition of each simulated scenario, a pcap file per SRF-IDS detector is generated. Therefore, all those files from different simulations were combined together so that a large pcap file per scenario is created. Assuming, for example, that *sim5-rep1IDS1.pcap* is the file of SRF-IDS detector 1 (IDS1) in the scenario with 5 SRF-IDS detectors (sim5) in first repetition (rep1), and *sim5-rep1IDS2.pcap* is the file for SRF-IDS detector 2 in first repetition, a new file called *sim5-rep1Merged.pcap* is created which contains the packets of all SRF-IDS detectors, of the first repetition from the scenario with 5 SRF-IDS detectors. The output of this process is one pcap file per repetition. Then, the same process is repeated to merge together the pcap files of all repetitions of each scenario. The result of this task is one pcap file per scenario.

Packet Classification and File Conversion

Once the dataset is formed, packet labelling or classification is the next step. Each packet is classified as *Malicious* or *Benign*. Classification is done based on conditions that are explained in the next sections. Packets contain several fields such as source IP, destination IP,

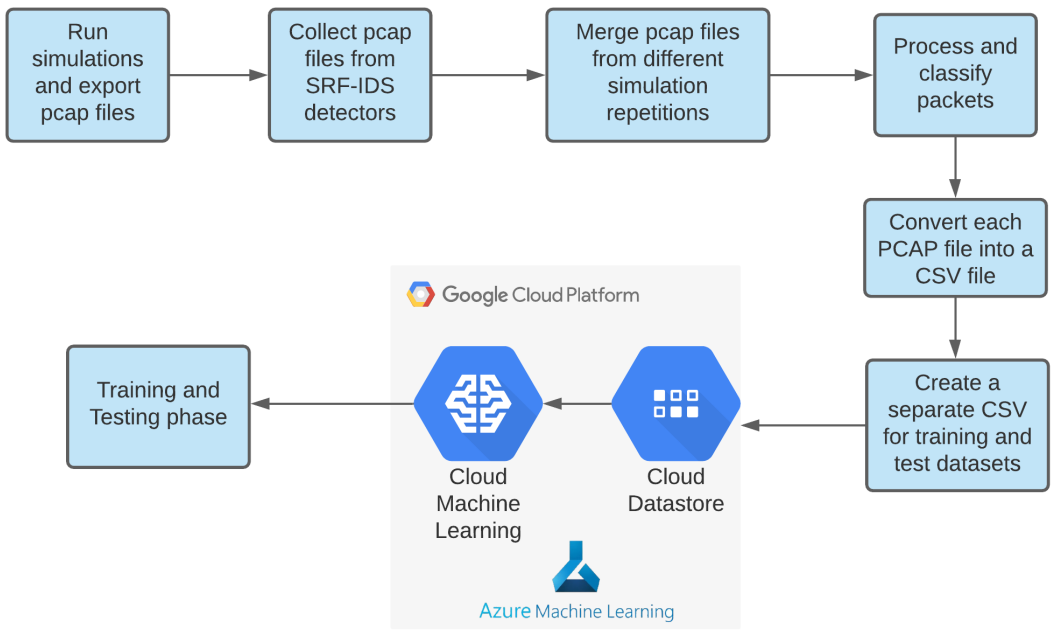


Fig. 5.19 Procedure followed

protocol etc. An analysis was carried out on network packets to understand and choose the packet fields that contain useful information. For example, rank field of RPL DIO packet was selected to identify rank attack. Packets with rank 129, the rank value of malicious nodes, were classified as *Malicious*. Regarding blackhole attackers, RPL packets with destination a malicious node are classified also as *Malicious*. As for other attacks such as DIS flooding, several simulations indicated that SRF-IDS was able to detect them with high detection rate. Therefore, the ML model is designed to focus on complex attacks such as rank and blackhole attacks. As a last step of this task, processed packets have to be merged together so that one pcap file is created for all the simulated scenarios. The outcome of this process are two files only, the training and testing datasets.

The next task is the file conversion from pcap format into Comma-separated values (CSV) format. This was necessary as the ML tools allow only CSV files extensions as datasets. Thus, the merged pcap files had to be converted into CSV files. At the end, we have a training dataset and a testing dataset in CSV format.

Learning and Testing Phases

Learning procedure starts right after the datasets are created. Google's Cloud Datastore and Microsoft's Azure Datastore are used to store the datasets in each platform in the cloud. It

allows the user to manage, edit and analyse datasets before proceeding to any training task. Once data pre-processing is finished, the learning procedure starts. Google AutoML handles all training parameters so that it automatically adjust them to achieve the highest detection performance. Testing dataset is used as the evaluation method. Results are exported and analysed in the dashboard provided by the Google platform.

The same process is repeated in MS Azure ML. The difference is that in MS Azure ML user has to design the experiment which includes defining the flow to be followed from feature selection to evaluation model as well as configuring the actual parameters of the deployed ML algorithms. The default parameters provided by the platform have been used in AzureML. After the evaluation phase, MS Azure ML allows the user to visualise results, check different metrics, and compare the performance of the deployed algorithms.

5.4.3 Implementation

The implementation steps for creating and validating the ML model are discussed in this section. Dataset implementation procedure is explained in detail. Also, packet fields and feature selection method are discussed.

Dataset Creation Process

Both training and test datasets are created from the experiments executed in Section 5.1. In those scenarios, 30 benign nodes, 6 blackhole attackers, and a varying number of 5 to 15 SRF-IDS detectors are deployed. Basically, simulations of 11 scenarios are repeated 10 times and are analysed for creating the training set. Similarly, 110 simulations, 11 scenarios repeated 10 times each, are analysed for generating the test set. We divided training and testing datasets so that ML model is trained using the former, and tested using the latter. The first 5 repetitions are used for the training set while the rest 5 are used for test set. Only pcap files generated by SRF-IDS detectors are used to train and evaluate the ML model. This is to verify that ML model will be as realistic as possible by using the packets captured actually from the SRF-IDS detectors during the simulations.

After an analysis of the datasets, the number of *Malicious* packets are less than the *Benign* packets. This is expected because the number of attackers deployed in the network is less than benign nodes. Therefore, there is an imbalance in the created datasets which the selected classification algorithms in the training phase will need to handle.

In order to create both training and testing datasets, a script was written in Python language so that pcap files are parsed and analysed. A prerequisite for generating the datasets is to export and parse pcap files created by the simulations explained in previous section.

Algorithm 14 presents the steps involved in order to create training and testing datasets. As both datasets follow the same procedure, the following describes the creation of training dataset. Assuming pcap files are available, the first step as shown in Algorithm 14 is to merge all pcap files of training set into one file by calling the *merge_train_data_into_pcap()* function. Inside this function, the script iterates over each scenario to combine and merge pcap files from the first 5 repetitions created by the simulated scenario. As already mentioned before, only pcap files generated by SRF-IDS detectors were used for dataset creation procedure.

Algorithm 14 Dataset creation process

```
1: Input: Pcap files exported from simulations
2: Output: Training and test datasets as CSV files
3: merge_train_data_into_pcap();
4: merge_test_data_into_pcap();
5: for each sim in SCENARIOS do
6:     create_training_csv(sim);
7:     create_test_csv(sim);
8: end for
9: for each file in SCENARIOS_CSV do
10:    merge_and_filter_train_csv(file);
11:    merge_and_filter_test_csv(file);
12: end for
```

The output of this function is a pcap file per scenario that contains all packets captured from SRF-IDS detectors. Next step is to call the *create_training_csv()* for each scenario to rename packet fields, classify packets, filter packet types, and finally create a CSV file from the pcap file. Specifically, this function renames packet headers for better understanding. Then, it classifies packets into “Malicious” or “Benign”. This is very significant as it will allow the ML algorithm to learn when a packet belongs to each category. We created a new field in the CSV file called “Category”. This field indicates if a packet is “Malicious” if any of the following conditions occurs:

- Rank field has a value equal to 129.
- Next-hop destination IP belongs to a rank/blackhole attacker.

The first condition means that a rank attacker advertises false rank while second condition means that the next-hop node is a blackhole attacker. Packets that do not fulfil any of the above conditions, they are treated as “Benign”. The last task of the function is to filter packets so that only ICMPv6 and UDP packets are contained in the CSV file. This aims to remove any unnecessary packets captured by devices The resulting file is a CSV file that contains

Table 5.6 Packet fields

Field name	**Description**
Rank	DODAG rank value of the node sending the DIO packet
wpanDst	MAC layer destination address in hexademical
wpanDst16Int	MAC layer destination address in decimal
Src	Source IP address of the sending node
Dst	Destination IP address of the sending node
Parent	Parent IP of the sending node
Code	RPL message type
Flag	Packet flags
wsInfo	Additional information about the packet
ipv6Plen	IPv6 packet payload length
Length	Frame length
ipv6Nxt	IPv6 packet next header
Time	Absolute time when this frame was captured
DAOSequence	A sequence number incremented at each unique DAO message from a node and echoed in the DAO-ACK packet
Reserved	Reserved flag, must be zero
rplInstance	Shows which RPL Instance the DODAG is part of

many useful information such as source IP, destination IP, protocol, timestamp and other fields. As each scenario generated a different CSV file, we had to merge all CSV files into one large file. This process is done by *merge_and_filter_train_csv()* function. The output of this function is a CSV file with many fields available for training our ML model. As indicated in the algorithm,the procedure described for creating the training dataset, it is repeated for creating the test dataset.

Packet Processing

Each network packet sent by a node contains a large amount of fields from PHY, MAC, 6LoWPAN, network, transport and application layers. However, for our experiments a total of 16 fields were chosen to be included in the datasets. The packet fields contained in both training and testing datasets are shown in Table 5.6. As it is depicted, some fields contain basic network information while some of them contain RPL-specific information. The selection of those packet fields was based on detailed packet analysis. Several pcap files were analysed to decide which fields contain critical information. Apart from that, the way of how routing attacks work was considered to create the list of 16 packet fields. For example, the *Rank* and *ipv6Nxt* fields are useful to attackers as they are usually modified by routing attackers to advertise fake rank or route a packet to a blackhole node.

5.4.4 Configuration and Metrics

The configuration and metrics used for evaluating the ML model are discussed in this section. Transformation of data was used in both platforms to normalise data and make them appropriate for the ML algorithms. Moreover, the threshold for the learning rate was configured at 0.5 in both platforms. In MS Azure ML, several ML algorithms are available to be chosen by the user in order to train and test the ML model. As the current task was to classify packets into two categories, the decision for choosing the training algorithms was based on accuracy and training time [112]. The first algorithm chosen is *2-class Decision Forests* which usually shows high accuracy but needs moderate training time [59]. The second ML algorithm is the *2-class Support Vector Machine (SVM)* [60] which is good at training the model with large feature sets but usually shows less accuracy than DF.

Regarding Google AutoML, the user does not select any algorithm as Google uses its own ML algorithms. Therefore, the only available configuration for the user is to choose the structure of the data and set the input datasets.

In regard to the metrics, the following are used to evaluate the ML model:

- **AUC**: It is the Area under the Receiver Operating Characteristic (ROC) Curve. AUC value ranges from zero to one and the higher it is, the better.
- **Accuracy:** It is the ratio of the number of true predictions to the total number of cases.
- **Precision:** It is the ability of a model to avoid labelling negative samples as positive. Low precision indicates high FP rate.
- **Recall:** It is the ratio of correctly classified as positive samples divided by the total number of positive samples. It is actually the TP rate.
- **F1 score:** It tells how precise and robust is the classifier. Actually, F1 score is the harmonic mean of precision and recall metrics. It provides a fair representation of both false positives and false negatives. However, true negatives are not taken into account in the calculations.

5.4.5 Evaluation Results

Results obtained from the two ML platforms are presented in this section. Table 5.7 depicts the three ML algorithms used in the experiments along with the calculated metrics. It is obvious that Google AutoML algorithm shows superior performance in comparison with 2-class SVM and 2-class Decision Forest (DF). Evaluation showed a precision of 93.3% in

Table 5.7 ML algorithms evaluation results

Metric	**2-class SVM**	**2-class Decision Forest**	**Google AutoML**
Accuracy	76.1%	92.2%	-
Precision	3.5%	76.7%	93.3%
Recall	2.8%	62%	93.3%
F1 Score	3.1%	68.6%	93.3%
AUC	0.49	0.84	0.92

Google AutoML, followed by 76.7% and 3.5% in 2-class DF and 2-class SVM respectively. Although Google AutoML does not provide Accuracy metric, the SVM achieves 76.1% while DF's is increased by 16.2%. The F1 Score is one of the most important metrics to look for when a classification algorithm is evaluated because is a mean of Recall and Precision metrics. The highest F1 Score is 93.3% achieved by Google AutoML, followed by 68.6% of DF and the lowest one is 3.1% of SVM. Another metric is AUC which shows if the model can discriminate between malicious and benign packets. Based on the results, Google AutoML has 0.92 while 0.84 and 0.49 are recorded for DF and SVM algorithms respectively.

Another interesting figure generated by Google AutoML is the feature importance. As depicted in Figure 5.20, Google AutoML analysed the dataset to find the most important features after the training phase of the ML model. As illustrated, only 3 out of 16 features are important to consider. Specifically, *Rank* feature is more than 80% important, *wpanDst* field is less than 20% important and *Dst* has very low importance. *Rank* field is obviously significant to have it as rank attack modified this specific packet field. For the other two fields, further analysis should be made to determine if they are really useful for the training of ML model. Although the rest fields seem to have low importance, some might be needed to avoid data overfitting. Therefore, careful analysis of the features should be done before proceeding to feature selection which will enhance ML model's performance.

Generally, the low performance of SVM and DF algorithms could be due to the imbalance datasets that were created for training purposes. The default settings of the ML algorithms trained the model in a way that low performance is achieved as depicted by the metrics. On the other hand, Google AutoML produced great results and handled the imbalanced dataset in a way that it didn't affect its performance. For this reason, the ML model created by Google AutoML will be deployed as part of the SRF-IDS component of SRF-IoT framework.

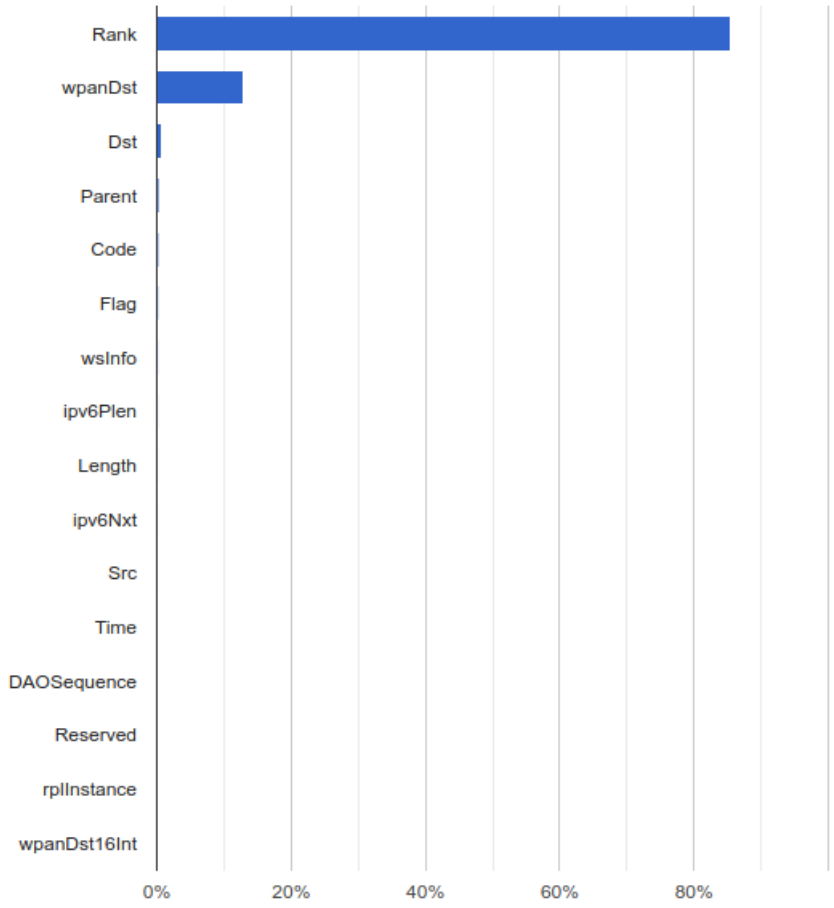


Fig. 5.20 List of features ordered by importance

5.5 Comparison with Related Works

This section focuses on the discussion and the comparison of the results obtained from the proposed SRF-IoT framework with other similar studies.

Table 5.8 presents an overview of the results obtained from the evaluation of SRF-IoT framework in the current chapter. As it can be seen, we have evaluated the SRF-IoT framework in medium and large scale networks under blackhole, rank and DIS flooding attacks. The first row represents the results of Section 5.1 in which SRF-IoT demonstrates a PDR of 92.8%, while only 8.2% of packets are dropped and the overall parent changes are just 97. The next results from Section 5.2 show high PDR of 93%, while 20% of packets are dropped and nodes change on average 200 parents. Although, DIS flooding attackers are added in those scenarios, SRF-IoT framework show good performance. The last results from Section 5.3 depict a decrease in the metrics. This is due to the large network and the challenges faces by SRF-IoT to keep the network stable when it is under attack. The analysis

of results show 52% PDR, just 9% of packets are dropped and 3248 parent switches on average. All the calculated metrics are median values.

Table 5.8 Summary of SRF-IoT framework results from the scenarios explored in the current chapter. BH: Blackhole attack, Rank: Rank attack

Attacks	**Network size**	**PDR**	**Packets Dropped**	**Parent Switches**
BH and Rank	Medium (30 BN nodes plus 6 attackers)	92.8%	8.2%	97
DIS flooding, BH and Rank	Medium (30 BN nodes plus 12 attackers)	93%	< 20%	200
DIS flooding, BH and Rank	Large (120 BN nodes plus 24 attackers)	52%	9%	3248

In Table 5.9, a comparison of our work with four similar works is presented. Specifically, the evaluation results from the studies of MRTS [79], SRPL-RP [83], SecTrust-RPL [86],[83] and RPL Pre-installed Secure Mode (PSM) [67] are compared with our proposed SRF-IoT framework. The metrics used in the comparison are the PDR, parent switches and packets dropped. In addition, the total number of nodes deployed in each experiment is compared.

The related works that are used for comparison purposes explore rank or blackhole attackers in medium scale networks. As we wanted to have a similar basis for comparing all the related studies, the results from Section 5.1 are referenced and used. That specific section evaluates SRF-IoT framework in a scenario where 30 benign nodes are deployed and 6 nodes launch a combination of blackhole and rank attacks. Thus, simulations have similar configurations with the other works.

As it can be seen from Table 5.9, most of the studies provide the PDR metric, while only one of them provides the parent switch and another one the packets dropped metrics. In our work, we provide all of the aforementioned metrics. Looking at PDR metric, the highest PDR value is achieved by SRPL-RP with 98.48% in a small network of 16 nodes plus the 4 rank attackers. The second most higher PDR value is from our SRF-IoT framework which achieves 92.8% in a network of 30 benign nodes plus 6 attackers that launch combined blackhole and rank attacks. MRTS study follows with a PDR up to 90% in a network with 27 nodes plus the 3 blackhole attackers. The SecTrust-RPL and RPL PSM studies exhibit the lowest PDR with a value of 80%. Both of them deploy 27 nodes but the former has 3 rank attackers while the latter has only 1 blackhole attacker. Comparing the rest metrics, parent switches of SRF-IoT framework are slightly more than the 80 switches observed in MRTS work. Regarding packets dropped, our work keeps the percentage near 8% which is very low in comparison with the 22-23% recorded in SecTrust-RPL study.

Table 5.9 Comparison of results with similar IoT-related studies. BH: Blackhole attack, Rank: Rank attack

Study (Attack)	PDR	Parent Switches	Packets Dropped	Number of nodes and attackers
MRTS (BH)	up to 90%	>80	-	27 nodes, plus 3 attackers
SRPL-RP (Rank)	98.48%	-	-	16 nodes, plus 4 attackers
SecTrust-RPL (Rank)	80%	-	22-23%	27 nodes, plus 3 attackers
RPL Preinstalled Secure Mode (BH)	80%	-	-	27 nodes, plus 1 attacker
Our proposed SRF-IoT (Rank and BH)	92.8%	97	8.2%	30 nodes, plus 6 BH+Rank attackers

All in all, it is obvious that the proposed SRF-IoT is an effective solution that achieves the best results among the current related studies that deploy less nodes and study only single attacks. SRF-IoT was evaluated in a larger network than other works, using a combination of blackhole and rank attacks and demonstrates superior performance in most cases.

5.6 Chapter Summary

In the current chapter, experimental evaluation of SRF-IoT framework was presented.In the first case SRF-IoT was evaluated under rank and blackhole attacks in a medium scaled network. The second case explored the performance of SRF-IoT under a combination of different routing attacks including rank, blackhole and DIS flooding attacks. The last case tests SRF-IoT framework in a large network in which the combination of the three routing attacks are launched. The specific scenarios, tools and metrics are analysed for each of the explored cases. New methods of intrusion detection using ML algorithms are also explored. Specifically, various ML models are studied, trained and evaluated using the datasets created from previous simulations. Evaluation results from both ML models and different scenarios are discussed. A comparison of experimental results with other related works is also presented. SRF-IoT showed improved performance along the different metrics for almost all scenarios.

Chapter 6

Conclusion and Future work

This chapter summarises the results of the research carried out in this project. In addition, limitations and future improvements are proposed.

6.1 Conclusions

In this project, a novel security framework is proposed, called SRF-IoT, for protecting IoT-based networks against network layer attackers. The main focus of our framework is to detect and identify attackers that exploit routing protocols and especially the RPL protocol. Assisting monitored nodes to avoid attackers, we minimise the impact of an attack in IoT network. We studied the impact of some severe routing attacks in IoT networks including "Hello" or DIS flooding, rank and blackhole attacks.

As our goal was to develop a solution with real-world impact, we have chosen to design and implement the solution in a known OS, ContikiOS/NG that is being used by many manufacturers. The Cooja simulator was used to simulate the initial scenarios and investigate the characteristics of an RPL-based IoT network under its normal operation. Then, we studied and implemented DIS flooding attack, and evaluate a network in Cooja under this specific attack. Specifically, experimental results of "Hello" or DIS flooding attack showed an increase in power consumption of deployed devices by 35%. This is due to the large number of DIS packets exchanged in the network. Effective defence methods based on thresholds were designed, implemented and tested in ContikiOS. Evaluation of the threshold-based approach showed 100% True Positive (TR) rate when 3 or more detectors are used in the network and less than 1% False Positive (FP) rate.

The next step was to explore other routing attack such as rank and blackhole attacks. Those two routing attacks were implemented in Contiki-NG and their impact on network performance was explored. Evaluation results depicted that blackhole and rank attackers may

largely affect the network operation by reducing the mean Packet Delivery Ratio (PDR) by 14% while almost 42% of packets sent were dropped in some cases. Based on the obtained results, we designed and developed a new security framework in Contiki-NG called SRF-IoT for detecting RPL routing attackers. In this framework, a trust-based solution, called SRF-OF, aims to identify and isolate traffic and topology -based routing attackers while an external IDS, called SR-IDS, aims to detect resource-based routing attackers. The proposed SRF-OF method is a trust-based system that makes use of the external SRF-IDS to get intelligence and chooses the best route for network packets based on a trust score.

SRF-IoT framework is evaluated in a new simulator, called Whitefield framework, that is able to run Contiki-NG on top of NS-3 simulator. Obtained results indicate superior performance of SRF-IoT framework with 92.8% PDR, 8.2% packets dropped, and under 100 parent switches in a scenario under routing attacks. Moreover, in malicious scenarios with SRF-IoT framework deployed, nodes change parents 3 times less than in malicious scenarios where SRF-IoT framework is not deployed. Simulation results demonstrate that our proposed SRF-IoT framework can efficiently protect devices against routing attacks in medium as well as large networks. This is achieved by assisting nodes to select proper devices as parents so that compromised nodes are avoided, reducing also the extra processing and energy overhead.

6.2 Limitations and Future Work

In this project, SRF-IoT framework has been designed and developed to protect IoT networks against routing attackers. However, some limitations of our work are the following:

- Experiments through simulations are only used to evaluate the SRF-IoT framework. The behaviour of the proposed solution is not evaluated in a real testbed, and it could be different in a real-world scenario. Although the software is implemented in Contiki-NG which is a real-world OS, the hardware platform might have constraints that in simulations do not exist. Moreover, simulations do not have the possibility to emulate exactly the functionalities of a real hardware platform. For instance, SRF-IDS detectors are implemented in such way to operate in promiscuous mode and capture network traffic from multiple interfaces. However, only specific real hardware devices might support promiscuous mode or sniffing traffic from multiple wireless interfaces. These issues should be explored when deploying the system in a real testbed.

- Only routing attacks have been explored in this project. A limitation of the project is that only attacks affecting RPL routing protocol are studied. Therefore, SRF-IoT framework protects IoT networks only from those types of attacks, leaving the network

vulnerable to other kinds of threats such as application layer attacks. Althrough routing attack can cause severe impact on IoT networks, the SRF-IoT framework cannot help the network if attackers exploit the application that run on top of network stack.

- SRF-IDS detectors use a fixed number as a threshold to detect DIS flooding attackers. The current implementation of DIS flooding detection algorithm works with the use of predefined thresholds. Several metrics including the packet interval and the number of packets sent from nodes have been studied to define the proper thresholds. However, there is a security risk that malicious attackers could learn the value of the threshold after some time. As a result, they can exploit the detection mechanism and avoid being detected by changing their attack behaviour. At that point, SRF-IDS will be unable to detect resource-based attacks such as DIS flooding attacks.

- SRF-IoT framework works mostly as reactive security measure. For example, routing attackers are identified after the attack has already been launched. Therefore, SRF-IoT tries to detect and minimise the impact from attackers by informing monitored nodes. Even though the proposed security measure provides a level of security by detecting and smartly isolating attackers, in cases that IoT network is part of a critical infrastructure there is a risk that damage might have already be done by the attackers.

- If no SRF-IDS detector is in TX/RX range of a monitored node, SRF-IoT will not be fully operational. There are cases in which SRF-IDS detectors are randomly deployed away from a monitored node. This means SRF-IoT framework will not be able to transmit trust metrics to a neighbouring node, and thus, MRHOF will be used in place of SRF-OF by the monitored node. The risk is that in case a malicious node exist in the network, the monitored node will not be able to avoid the attacker. These limitations required the network administrator in real world environments to deploy SRF-IoT nodes near the monitored network's nodes.

Having in mind the previous limitations, further improvements on the framework can be done. For future work, the author suggests the following:

- Extend the work by detecting a combination of routing and application layer attacks. The research of more routing attacks could include sinkhole, wormhole and selective forwarding attacks. Those attacks can impose severe threat to an IoT network. A combination of them should be also explored. Apart from network layer attacks, some application layer threats such as SQL injection could be useful to study. The implementation of SQL injection attack could be achieved by deploying of a Constrained Application Protocol (CoAP) server in Contiki-NG which will be intentionally

vulnerable. Then, SRF-IoT nodes will attempt to find this vulnerability using several known SQL injection commands. Initially, the attacks could be developed in Contiki-NG, and the Whitefield framework simulator could be used to simulate and study the impact of the attacks. Results from attack simulations should be analysed to develop effective detection and mitigation methods. Then, implemented algorithms should be evaluated using both simulations and real-world experiments. Detecting a combination of network and application layer attacks using the SRF-IoT framework will make it a great tool for protecting IoT networks at all layers. Moreover, mitigation methods for each studied attack should also be developed so that the impact of attacks will be minimised.

- Improve SRF-IDS detection algorithm for DIS flooding attackers so that it utilises a dynamic-based threshold. Currently, SRF-IDS uses static thresholds to detect DIS flooding attackers. An idea is to change the approach from static to dynamic so that a threshold is defined after some period of time. This could be achieved by using an SRF-IDS detector to sniff packets and monitor the behaviour of monitored network for a specific time period. For example, metrics such as packet intervals and number of messages exchanged during this time period could be recorded. Later, collected metrics could be analysed and a formula could be used to determine the normal and malicious thresholds. Those thresholds can be changed after some period of time so that malicious nodes do not exploit this mechanism. A problem of this approach is again the memory constrains of the node. A special node with better memory and processing capabilities could be used to solve this issue. After data collection, the special node could update SRF-IoT nodes with the new values, and therefore the threshold will be updated dynamically.

- Performance of ML model can be improved by using a balanced dataset for training. A drawback of the current produced dataset is that the scenario included 30 benign nodes and 6 attackers, creating an imbalanced dataset. Having a small number of attackers generates small number of malicious packets. Therefore, SRF-IDS detectors capture mostly benign traffic. If the same number of malicious nodes is deployed, the dataset will contain a similar number of benign and malicious packets, creating a balanced dataset. Having a balanced dataset, will allow the ML algorithm to make better predictions, avoiding data overfitting. Moreover, feature selection could be used to choose only the important packet fields and then, evaluate the performance of ML model.

- Deploy the ML model for real-world use. The trained ML model could be deployed in Google or MS cloud so that a prediction service will be available for use by SRF-IoT. Specifically, SRF-IDS could interact with the deployed model using an API. The request of SRF-IDS could contain the packet fields of a sniffed packet. Then, API will response by predicting if a packet is malicious or not. A limitation of this approach is that internet connection is required for SRF-IoT framework. In case having internet connectivity is an issuse, the trained ML model can be deployed in a local server which will server SRF-IoT with the prediction service.

- Enhance the SRF-IoT framework by adding some pro-active features that could improve the security of IoT networks. In order to achieve that, a penetration testing module could be designed and implemented in SRF-IDS detectors. For example, instead of just updating monitored nodes with trust metrics, a penetration test could be executed by SRF-IDS detectors against monitored nodes to check for exploitable vulnerabilities. In case a vulnerability is found, SRF-IDS detector could send the relevant information to the *Communication Module* which will transmit them to monitored node along with the trust metrics. However, one of the limitations of creating such a tool is the limited memory of SRF-IDS sensors. In order to overcome the memory issues, a custom network tool could be designed that will execute simple tests against known network ports and known vulnerable web applications without requiring to store a lot of information. Moreover, a check for sensitive data in the sniffed packets that already captured by SRF-IDS detectors could be a useful feature for this penetration module.

Nomenclature

Glossary

BHR scenario : A scenario where blackhole and rank attacks are launched.

Blackhole attack : A routing attack in which the compromised node drops all incoming traffic.

DIS flooding attack : An RPL-specific DoS attack in which an attacker transmits a large batch of DIS control packets to a device preventing from sleeping.

DoS attack : A resource-based routing attack in which an attacker may prevent a device from sleeping by periodically transmitting packets.

IDS incidents : Indicates how many times the IDS generated a warning for a malicious node.

Mean number of DIS messages : Indicates the number of DIS messages sent by a node on average.

Mean number of other messages : Indicates the number of other messages sent by a node on average, apart from DIS messages. These could be DIO, DAO, or UDP messages.

Mean packet interval : Indicates how often on average a node sends a packet, measured in seconds.

Median IDS Packet Overhead : Indicates the median percentage of SRF-IDS detector's packets sent to SRF-IDS root and the monitored network during simulation.

Median Packet Delivery Ratio (PDR) : Indicates the median value of the ratio of the total number of unicast packets received by BR up to the total number of unicast packets generated by all benign and malicious nodes. It does not include UDP re-transmitted packets.

Median Packets Dropped : Shows the median percentage of packets dropped by the attackers during the simulation.

Median Parent Switch : Presents the median value of the number of parent switches that benign and malicious nodes execute during the simulation. A parent switch happens to select a better route to the BR.

Messages sent to IDS root : Indicates the number of messages sent to IDS root by IDS detectors.

Monitored network : The network that is being monitored by the SRF-IDS.

Monitored nodes : The nodes that belong to the network that is being monitored by SRF-IDS.

Neighbour : The node N_b is a neighbour of N_a only if N_b is in transmission (TX) range of N_a. That means N_b could provide a route to sink/BR.

Precision : A metric that presents the total number of malicious nodes that are correctly classified as malicious (TP) divided by the total number of nodes classified as malicious.

Rank attack : A routing attack in which a malicious node may intentionally advertise lower rank in order to attract neighbouring devices to select it as preferred parent.

$RPL-lite$: A lightweight implementation of the IPv6 Routing Protocol for Low-Power and Lossy Networks.

RPL InstanceID : A unique ID that is used to identify the RPL Instance that DODAG is part of.

$SRF-IDS$: The Intrusion Detection System (IDS) that is part of the SRF-IoT framework. It consists of SRF-IDS Detector and SRF-IDS Root.

$SRF-IDS\ Detector$: A sensor device deployed in SRF-IDS that operates in promiscuous mode and gather information from monitored network.

$SRF-IDS\ Root$: The root node of SRF-IDS that is responsible to take final decisions for malicious nodes.

$SRF-OF$: Security Framework Objective Function is an objective function used by SRF-IoT to calculate neighbouring node's trust.

TM Module : Trust Monitoring Module is an internal component of SRF-IDS that is used by SRF-IDS Detector to monitor and update neighbours' trust.

Version number modification attack : An RPL-specific DoS attack in malicious node changes the DODAG version number before forwarding the received DIO messages to the next hop. This causes unnecessary global rebuilds of the DODAG.

Acronyms / Abbreviations

6*BR* 6LoWPAN Border Router

6*LoWPAN* The IPv6 over Low -Power Wireless Personal Area Networks

AIP Attacker Information Packet

AODV Ad-hoc On Demand Distance Vector

ASM Authenticated Security Mode

AUC Area Under the *ROC* Curve

BH Blackhole topology attack

BR Border Router

CIA Confidentiality, Integrity and Availability

CoAP Constrained Application Protocol

COLIDE Collaborative Intrusion Detection for IoT

CSMA Carrier Sense Multiple Access

CSV Comma-separated values

DAG Directed Acyclic Graph

DAO Destination Advertisement Object

DAO − *ACK* Destination Advertisement Object Acknowledge

DCTM − *IoT* Dynamic and Comprehensive Trust Model for IoT

DDoS Distributed Denial of Service attack

DF Decision Forest

DIO DODAG Information Object

DIS DODAG Information Solicitation

DNS Domain Name System

DODAG Destination-Oriented Directed Acyclic Graph

DoS Denial of Service attack

ERNT Extended RPL Node Trustworthiness

ETX Expected Transmission Count

FP False Positive

ICMP Internet Control Message Protocol

*ICMPv*6 Internet Control Message Protocol version 6

IDS Intrusion Detection System

IETF Internet Engineering Task Force

IoT Internet of Things

IP Internet Protocol

IPSec Internet Protocol Security

*IPv*6 Internet Protocol version 6

LA Learning Automata

LLN Low Power and Lossy Network

MAC Media Access Control

MitM Man-in-the-Middle attack

ML Machine Learning

MRHOF Minimum Rank with Hysteresis Objective Function

MRTS Metric-based RPL Trustworthiness Scheme

MS Microsoft

OF Objective Function

*OF*0 Zero Objective Function

OS Operating System

OSI Open Systems Interconnection

P2P Peer to peer network

PCAP Packet capture file

PDR Packet Delivery Ratio

PHY Physical layer

PSM Pre-installed Secure Mode

QoS Quality of Service

RCE Remote Code Execution

RDC Radio Duty Cycling

RERR Route Error

ROC Receiver Operating Characteristic

RPL The IPv6 Routing Protocol for Low-Power and Lossy Networks

RREP Route Reply

RREP_ACK Route Reply Acknowledgement

RREQ Route Request

RSS Received Signal Strength

RTOS Real-time Operating System

RX Radio receiver

SBIDS Sink-Based Intrusion Detection System

SDK Software Development Kit

SIEWE Strainer-based Intrusion Detection of Blackhole in 6LoWPAN for the Internet of Things

SLA Service Level Agreement

SOA Service Oriented Architecture

SRF – IoT Security Framework for IoT-based devices

SRPL Secure RPL

SRPL – RP Secure RPL Routing Protocol

SSH Secure Socket Shell

SVM Support Vector Machine

TM Trust Monitoring Module

TP True Positive

TRAIL Trust Anchor Interconnection Loop

TX Radio transmitter

UDP User Datagram Protocol

WSN Wireless Sensor Networks

Notations

$DT(D_a,D_b)_T$ Direct trust that device D_a holds for device D_b until time period T.

$E[N_{dis}]$ Mean number of DIS messages.

$PF_{ba}(T)$ The total number of packets forwarded by device D_b on behalf of device D_a until time period T.

$E[I_{pkt}]$ Mean packet interval.

$E[N_{other}]$ Mean number of other node's messages.

$PT_{ab}(T)$ The number of packets successfully transmitted between devices D_a and D_b until time period T.

$E[DR_{pkt}]$ Mean/Median Packet Delivery Ratio (PDR)

E[PS] Mean/Median Parent Switch

$E[D_{pkt}]$ Mean/Median Packets Dropped

E[IDS] Median IDS Packet Overhead.

E[TP] True Positive (TP) rate

E[FP] False Positive (TP) rate

References

[1] Statista Research Department. Number of iot devices 2015-2025. https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/, Nov 2016.

[2] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The Internet of Things: A survey. *Computer Networks*, 54(15):2787–2805, Oct. 2010.

[3] Prabal Dutta and Adam Dunkels. Operating systems and network protocols for wireless sensor networks. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 370(1958):68–84, 2012.

[4] T. Winter et al. RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. RFC 6550, March 2012.

[5] Gemalto. The state of IoT security. http://www2.gemalto.com/iot/index.html, 2018.

[6] Chris Williams. BadAlloc: Microsoft looked at memory allocation code in tons of devices and found this one common security flaw. https://www.theregister.com/2021/04/29/microsoft_badalloc_iot/, April 2021.

[7] O'Donnell Lindsey. 'Amnesia:33' TCP/IP Flaws Affect Millions of IoT Devices. https://threatpost.com/amnesia33-tcp-ip-flaws-iot-devices/161928/, December 2020.

[8] egnite GmbH. Nut/OS. http://www.ethernut.de/en/firmware/index.html, July 2009.

[9] Forescout Research Labs. *Amnesia:33 How TCP/IP stacks breed critical vulnerabilities in IoT, OT and IT devices*. Forescout, 2020.

[10] Forescout Research Labs & JSOF. *NAME:WRECK Breaking and fixing DNS implementations*. Forescout, September 2021.

[11] Spring Tom. How the NAME:WRECK Bugs Impact Consumers, Businesses. https://threatpost.com/namewreck-bugs-businesses/165385/, April 2021.

[12] Arghire Ionut. At Least 100 Million Devices Affected by "NAME:WRECK" DNS Flaws in TCP/IP Stacks. https://www.securityweek.com/least-100-million-devices-affected-namewreck-dns-flaws-tcpip-stacks, February 2021.

[13] Jessica Davis. CISA Warns More Critical Flaws Found in Open Source TCP/IP Stacks. https://healthitsecurity.com/news/cisa-warns-more-critical-flaws-found-in-open-source-tcp-ip-stacks, February 2021.

[14] Arghire Ionut. Vulnerabilities in TCP/IP Stacks Allow for TCP Connection Hijacking, Spoofing. https://www.securityweek.com/vulnerabilities-tcpip-stacks-allow-tcp-connection-hijacking-spoofing, December 2021.

[15] JSOF research lab. Ripple20. https://www.jsof-tech.com/disclosures/ripple20/, 2020.

[16] Townsend Kevin. Critical Industries at Risk from Eleven Zero-day Flaws in Real Time Operating System. https://www.securityweek.com/critical-industries-risk-eleven-zero-day-flaws-real-time-operating-system, July 2019.

[17] Greengard Samuel. Understanding TCP/IP Stack Vulnerabilities in the IoT. https://www.darkreading.com/edge/theedge/understanding-tcp-ip-stack-vulnerabilities-in-the-iot/b/d-id/1339888, January 2021.

[18] Sergiu Gatlan. New mozi p2p botnet takes over netgear, d-link, huawei routers. "https://www.bleepingcomputer.com/news/security/new-mozi-p2p-botnet-takes-over-netgear-d-link-huawei-routers/", Dec 2019.

[19] Georgios Kambourakis, Constantinos Kolias, and Angelos Stavrou. The Mirai botnet and the IoT zombie armies. In *Military Communications Conference (MILCOM)*, pages 267–272. IEEE, 2017.

[20] Jai Vijayan. New malware family assembles iot botnet. https://www.darkreading.com/iot/new-malware-family-assembles-iot-botnet--/d/d-id/1337578, Apr 2020.

[21] Mahmoud Ammar, Giovanni Russello, and Bruno Crispo. Internet of Things: A survey on the security of IoT frameworks. *Journal of Information Security and Applications*, 38:8–27, 2018.

[22] Mukrimah Nawir, Amiza Amir, Naimah Yaakob, and Ong Bi Lynn. Internet of Things (IoT): Taxonomy of security attacks. In *Proc. 3rd International Conference on Electronic Design (ICED)*, pages 321–326. IEEE, 2016.

[23] Ahmed Raoof, Ashraf Matrawy, and Chung-Horng Lung. Routing attacks and mitigation methods for rpl-based internet of things. *IEEE Communications Surveys & Tutorials*, 21(2):1582–1606, 2018.

[24] M. A. Boudouaia, A. Ali-Pacha, A. Abouaissa, and P. Lorenz. Security against rank attack in rpl protocol. *IEEE Network*, 34(4):133–139, 2020.

[25] Philokypros P. Ioulianou, Vassilios G Vassilakis, Ioannis D Moscholios, and Michael D Logothetis. A signature-based intrusion detection system for the Internet of things. In *Proc. IEICE Information and Communication Technology Forum (ICTF)*, pages 1–6, Graz, Austria, July 2018.

[26] Philokypros P Ioulianou, Vassilios G Vassilakis, and Michael D Logothetis. Battery drain denial-of-service attacks and defenses in the Internet of Things. *Journal of Telecommunications and Information Technology*, (2):37–45, January 2019.

[27] Philokypros P. Ioulianou and Vassilios G. Vassilakis. Denial-of-service attacks and countermeasures in the RPL-based Internet of Things. *2nd International Workshop on Attacks and Defenses for Internet-of-Things (ADIoT) in conjunction with ESORICS*, Sept. 2019.

[28] Arsalan Mosenia and Niraj K Jha. A comprehensive study of security of Internet-of-Things. *IEEE Transactions on Emerging Topics in Computing*, 5(4):586–602, Dec. 2017.

[29] CISCO. The Internet of Things Reference Model. http://cdn.iotwf.com/resources/71/IoT_Reference_Model_White_Paper_June_4_2014.pdf, 2014.

[30] Yousaf Bin Zikria, Heejung Yu, Muhammad Khalil Afzal, Mubashir Husain Rehmani, and Oliver Hahm. Internet of things (iot): Operating system, applications and protocols design, and validation techniques. *Future Generation Computer Systems*, 88:699–706, 2018.

[31] O. Hahm, E. Baccelli, H. Petersen, and N. Tsiftes. Operating systems for low-end devices in the internet of things: A survey. *IEEE Internet of Things Journal*, 3(5):720–734, 2016.

[32] N. Al-Taleb and N. Min-Allah. A study on internet of things operating systems. In *2019 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT)*, pages 1–7, 2019.

[33] Adam Dunkels, Bjorn Gronvall, and Thiemo Voigt. Contiki - A lightweight and flexible operating system for tiny networked sensors. In *Proc. 29th IEEE Int. Conf. on Local Computer Networks*, pages 455–462, Tampa, FL, USA, Nov. 2004.

[34] Contiki-ng: The os for next generation iot devices. https://www.contiki-ng.org/, note = Accessed: 2021-03-25, 2017.

[35] Nicolas Tsiftes, Joakim Eriksson, and Adam Dunkels. Low-power wireless IPv6 routing with ContikiRPL. In *Proc. 9th ACM/IEEE Int. Conf. on Information Processing in Sensor Networks*, pages 406–407, 2010.

[36] TinyOS: An OS for Embedded, Wireless Devices. https://github.com/tinyos/tinyos-main, March 2021. Accessed: 2021-03-13.

[37] Emmanuel Baccelli, Oliver Hahm, Mesut Günes, Matthias Wählisch, and Thomas C Schmidt. Riot os: Towards an os for the internet of things. In *2013 IEEE conference on computer communications workshops (INFOCOM WKSHPS)*, pages 79–80. IEEE, 2013.

[38] Richard Barry et al. Freertos. *Internet, Oct*, 2008.

[39] Fredrik Osterlind et al. Cross-level sensor network simulation with Cooja. In *Proc. 31st IEEE Int. Conf. on Local Computer Networks*, pages 641–648, Tampa, FL, USA, Nov. 2006.

[40] Alvaro Diaz and Pablo Sanchez. Simulation of attacks for security in wireless sensor network. *Sensors*, 16(11):1932, 2016.

[41] George F Riley and Thomas R Henderson. The ns-3 network simulator. In *Modeling and tools for network simulation*, pages 15–34. Springer, 2010.

[42] OMNeT++: Discrete Event Simulator. http://www.omnetpp.org, 2020. Accessed: 2018-10-25.

[43] Athanassios Boulis. Castalia: Revealing pitfalls in designing distributed algorithms in WSN. In *Proc. 5th International Conference on Embedded Networked Sensor Systems*, pages 407–408, Sydney, Australia, 2007. ACM.

[44] Philip Levis and Nelson Lee. TOSSIM: A simulator for TinyOS networks. *UC Berkeley, September*, 24, 2003.

[45] Philip Levis, Samuel Madden, Joseph Polastre, Robert Szewczyk, Kamin Whitehouse, Alec Woo, David Gay, Jason Hill, Matt Welsh, Eric Brewer, et al. Tinyos: An operating system for sensor networks. In *Ambient intelligence*, pages 115–148. Springer, 2005.

[46] Contiki: The Open Source OS for the Internet of Things. http://www.contiki-os.org/, 2018. Accessed: 2018-10-25.

[47] Rahul Jadhav. Whitefield framework. https://github.com/whitefield-framework/whitefield, 2020.

[48] Erol Gelenbea, Edith Ngaib, and Poonam Yadavc. Routing of high-priority packets in wireless sensor networks. *IEEE Second International Conference on Computer and Network Technology, IEEE*, 2010.

[49] P. Levis, T. Clausen, J. Hui, O. Gnawali, and J. Ko. The Trickle algorithm. *Internet Engineering Task Force (IETF)*, 2011.

[50] Yuchen Yang et al. A survey on security and privacy issues in Internet-of-Things. *IEEE Internet of Things Journal*, 4(5):1250–1258, Oct. 2017.

[51] Fadele Ayotunde Alaba, Mazliza Othman, Ibrahim Abaker Targio Hashem, and Faiz Alotaibi. Internet of Things security: A survey. *Journal of Network and Computer Applications*, 88:10–28, June 2017.

[52] Symantec Security Response. Mirai: What you need to know about the botnet behind recent major DDoS attacks, Oct. 2016.

[53] Timothy Easton. Chalubo botnet wants to DDoS from your server or IoT device. https://news.sophos.com/en-us/2018/10/22/chalubo-botnet-wants-to-ddos-from-your-server-or-iot-device/, Oct. 2018.

[54] Anthea Mayzaud, Remi Badonnel, and Isabelle Chrisment. A taxonomy of attacks in rpl-based internet of things. *International Journal of Network Security*, 18(3):459–473, 2016.

[55] Anthéa Mayzaud, Anuj Sehgal, Rémi Badonnel, Isabelle Chrisment, and Jürgen Schönwälder. A study of RPL DODAG version attacks. In *Proc. IFIP International Conference on Autonomous Infrastructure, Management and Security*, pages 92–104, Brno, Czech Republic, June/July 2014. Springer.

[56] Bruno Bogaz Zarpelão, Rodrigo Sanches Miani, Cláudio Toshio Kawakani, and Sean Carlisto de Alvarenga. A survey of intrusion detection in Internet of Things. *Journal of Network and Computer Applications*, 84:25–37, April 2017.

[57] Hung-Jen Liao et al. Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications*, 36(1):16–24, Jan. 2013.

[58] D Praveen Kumar, Tarachand Amgoth, and Chandra Sekhara Rao Annavarapu. Machine learning algorithms for wireless sensor networks: A survey. *Information Fusion*, 49:1–25, 2019.

[59] Y. Ioannou, D. Robertson, D. Zikic, P. Kontschieder, J. Shotton, M. Brown, A. Criminisi, Antonio Criminisi, Darko Zikic, and Jamie Shotton. Decision forests, convolutional networks and the models in-between. In *arXiv:1603.01250*, March 2016.

[60] Shan Suthaharan. Support vector machine. In *Machine learning models and algorithms for big data classification*, pages 207–235. Springer, 2016.

[61] Qing Yang and Fangmin Li. Support vector machine for intrusion detection based on lsi feature selection. In *2006 6th World Congress on Intelligent Control and Automation*, volume 1, pages 4113–4117. IEEE, 2006.

[62] Arvind Kamble, Virendra S Malemath, and Deepika Patil. Security attacks and secure routing protocols in rpl-based internet of things: Survey. In *2017 International Conference on Emerging Trends & Innovation in ICT (ICEI)*, pages 33–39. IEEE, 2017.

[63] Faiza Medjek, Djamel Tandjaoui, Imed Romdhani, and Nabil Djedjig. Security threats in the internet of things: Rpl's attacks and countermeasures. In *Security and privacy in smart sensor networks*, pages 147–178. IGI Global, 2018.

[64] Abhishek Verma and Virender Ranga. Security of rpl based 6lowpan networks in the internet of things: A review. *IEEE Sensors Journal*, 20(11):5666–5690, 2020.

[65] Linus Wallgren, Shahid Raza, and Thiemo Voigt. Routing attacks and countermeasures in the RPL-based Internet of things. *International Journal of Distributed Sensor Networks*, 9(8):1–11, 2013.

[66] Eric Garcia Ribera, Brian Martinez Alvarez, Charisma Samuel, Philokypros P Ioulianou, and Vassilios G Vassilakis. Heartbeat-based detection of blackhole and greyhole attacks in rpl networks. In *2020 12th International Symposium on Communication Systems, Networks and Digital Signal Processing (CSNDSP)*, pages 1–6. IEEE, 2020.

[67] Ahmed Raoof, Ashraf Matrawy, and Chung-Horng Lung. Secure routing in iot: Evaluation of rpl's secure mode under attacks. In *2019 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE, 2019.

[68] SM Riazul Islam, Daehan Kwak, MD Humaun Kabir, Mahmud Hossain, and Kyung-Sup Kwak. The Internet of things for health care: A comprehensive survey. *IEEE Access*, 3:678–708, 2015.

[69] Pascal Geenens. BrickerBot - The Dark Knight of IoT. https://blog.radware.com/security/2017/04/brickerbot-dark-knight-iot/, 2017.

[70] David R Raymond and Scott F Midkiff. Denial-of-service in wireless sensor networks: Attacks and defenses. *IEEE Pervasive Computing*, 7(1):74–81, 2008.

[71] Prabhakaran Kasinathan, Claudio Pastrone, Maurizio A Spirito, and Mark Vinkovits. Denial-of-Service detection in 6LoWPAN based Internet of Things. In *Proc. 9th IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pages 600–607, 2013.

[72] Tobias Heer, Oscar Garcia-Morchon, René Hummen, Sye Loong Keoh, Sandeep S. Kumar, and Klaus Wehrle. Security Challenges in the IP-based Internet of Things. *Wireless Personal Communications*, 61(3):527–542, Dec 2011.

[73] AL-Hawawreh Muna, Nour Moustafa, and Elena Sitnikova. Identification of malicious activities in industrial Internet of things based on deep learning models. *Journal of Information Security and Applications*, 41:1–11, 2018.

[74] Dragan Peraković, Marko Periša, and Ivan Cvitić. Analysis of the IoT impact on volume of DDoS attacks. In *Proc. 33rd Symposium on New Technologies in Postal and Telecommunication Traffic (PosTel)*, pages 295–304, 2015.

[75] Sudip Misra, P Venkata Krishna, Harshit Agarwal, Antriksh Saxena, and Mohammad S Obaidat. A learning automata based solution for preventing distributed denial of service in Internet of things. In *Proc. International Conference on Internet of Things and 4th International Conference on Cyber, Physical and Social Computing*, pages 114–122. IEEE, 2011.

[76] Qiao Yan, Wenyao Huang, Xupeng Luo, Qingxiang Gong, and F Richard Yu. A multi-level ddos mitigation framework for the industrial internet of things. *IEEE Communications Magazine*, 56(2):30–36, 2018.

[77] Monowar H Bhuyan, DK Bhattacharyya, and Jugal K Kalita. An empirical evaluation of information metrics for low-rate and high-rate ddos attack detection. *Pattern Recognition Letters*, 51:1–7, 2015.

[78] David Airehrour, Jairo Gutierrez, and Sayan Kumar Ray. Secure routing for internet of things: A survey. *Journal of Network and Computer Applications*, 66:198–213, 2016.

[79] Nabil Djedjig, Djamel Tandjaoui, Faiza Medjek, and Imed Romdhani. Trust-aware and cooperative routing protocol for iot security. *Journal of Information Security and Applications*, 52:102467, 2020.

[80] Seyyed Yasser Hashemi and Fereidoon Shams Aliee. Dynamic and comprehensive trust model for iot and its integration into rpl. *The Journal of Supercomputing*, 75(7):3555–3584, 2019.

[81] Ghada Glissa, Abderrezak Rachedi, and Aref Meddeb. A secure routing protocol based on rpl for internet of things. In *2016 IEEE Global Communications Conference (GLOBECOM)*, pages 1–7. IEEE, 2016.

[82] Manjula C Belavagi and Balachandra Muniyal. Multiple intrusion detection in rpl based networks. *International Journal of Electrical & Computer Engineering (2088-8708)*, 10, 2020.

[83] Zahrah A Almusaylim, NZ Jhanjhi, and Abdulaziz Alhumam. Detection and mitigation of rpl rank and version number attacks in the internet of things: Srpl-rp. *Sensors*, 20(21):5997, 2020.

[84] Ahmet Arış, Sıddıka Berna Örs Yalçın, and Sema F Oktuğ. New lightweight mitigation techniques for rpl version number attacks. *Ad Hoc Networks*, 85:81–91, 2019.

[85] Usman Shafique, Abid Khan, Abdur Rehman, Faisal Bashir, and Masoom Alam. Detection of rank attack in routing protocol for low power and lossy networks. *Annals of Telecommunications*, 73(7-8):429–438, 2018.

[86] David Airehrour, Jairo A. Gutierrez, and Sayan Kumar Ray. Sectrust-rpl: A secure trust-aware rpl routing protocol for internet of things. *Future Generation Computer Systems*, 93:860 – 876, 2019.

[87] Heiner Perrey, Martin Landsmann, Osman Ugus, Thomas C Schmidt, and Matthias Wählisch. Trail: Topology authentication in rpl. *arXiv preprint arXiv:1312.0984*, 2013.

[88] Kenji Iuchi, Takumi Matsunaga, Kentaroh Toyoda, and Iwao Sasase. Secure parent node selection scheme in route construction to exclude attacking nodes from rpl network. In *2015 21st Asia-Pacific Conference on Communications (APCC)*, pages 299–303. IEEE, 2015.

[89] Daniele Midi, Antonino Rullo, Anand Mudgerikar, and Elisa Bertino. Kalis - A system for knowledge-driven adaptable intrusion detection for the Internet of Things. In *Proc. IEEE 37th Int. Conf. on Distributed Computing Systems (ICDCS)*, pages 656–666, Atlanta, GA, USA, June 2017.

[90] OpenWRT: A Linux OS for Embedded Devices. https://openwrt.org/, 2004. Accessed: 2018-10-25.

[91] Martin Roesch et al. Snort: Lightweight intrusion detection for networks. In *Lisa*, volume 99, pages 229–238, 1999.

[92] Himanshu B Patel and Devesh C Jinwala. Blackhole detection in 6lowpan based internet of things: an anomaly-based approach. In *TENCON 2019-2019 IEEE Region 10 Conference (TENCON)*, pages 947–954. IEEE, 2019.

[93] Junaid Arshad, Muhammad Ajmal Azad, Muhammad Mahmoud Abdeltaif, and Khaled Salah. An intrusion detection framework for energy constrained iot devices. *Mechanical Systems and Signal Processing*, 136:106436, 2020.

[94] Anhtuan Le, Jonathan Loo, Yuan Luo, and Aboubaker Lasebae. The impacts of internal threats towards routing protocol for low power and lossy network performance. In *2013 IEEE Symposium on Computers and Communications (ISCC)*, pages 000789–000794. IEEE, 2013.

[95] Anhtuan Le, Jonathan Loo, Kok Keong Chai, and Mahdi Aiash. A specification-based IDS for detecting attacks on RPL-based network topology. *Information*, 7(2):1–19, 2016.

[96] Shahid Raza, Linus Wallgren, and Thiemo Voigt. SVELTE: Real-time intrusion detection in the Internet of Things. *Ad Hoc Networks*, 11(8):2661–2674, June 2013.

[97] Takumi Matsunaga, Kentaroh Toyoda, and Iwao Sasase. Low false alarm rate RPL network monitoring system by considering timing inconstancy between the rank measurements. In *Proc. 11th International Symposium on Wireless Communications Systems (ISWCS)*, pages 427–431, Barcelona, Spain, August 2014.

[98] Emmanuel Baccelli, Mathias Philipp, and Mukul Goyal. The P2P-RPL routing protocol for IPv6 sensor networks: Testbed experiments. In *Proc. 19th International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, pages 656–666, Split, Croatia, Sept. 2011.

[99] Joseph Polastre, Robert Szewczyk, and David Culler. Telos: Enabling ultra-low power wireless research. In *Proc. 4th Int. Symp. on Information Processing in Sensor Networks*, pages 364–369, Boise, Idaho, USA, April 2005.

[100] Zolertia technical documentation. https://github.com/Zolertia/Resources/wiki/Zolertia-Technical-documentation, 2017.

[101] George Rontidis, Emmanouil Panaousis, Aron Laszka, Tasos Dagiuklas, Pasquale Malacaria, and Tansu Alpcan. A game-theoretic approach for minimizing security risks in the internet-of-things. In *2015 IEEE International Conference on Communication Workshop (ICCW)*, pages 2639–2644, 2015.

[102] Weiwei Yuan, Donghai Guan, Young-Koo Lee, Sungyoung Lee, and Sung Jin Hur. Improved trust-aware recommender system using small-worldness of trust networks. *Knowledge-Based Systems*, 23(3):232–238, 2010.

[103] Emmanouil Panaousis, Eirini Karapistoli, Hadeer Elsemary, Tansu Alpcan, MHR Khuzani, and Anastasios A Economides. Game theoretic path selection to support security in device-to-device communications. *Ad Hoc Networks*, 56:28–42, 2017.

[104] Bashar A Alohali, Vassilios G Vassilakis, Ioannis D Moscholios, and Michael D Logothetis. A secure scheme for group communication of wireless IoT devices. In *Proc. 11th IEEE/IET Int. Symp. on Communication Systems, Networks, and Digital Signal Processing (CSNDSP)*, pages 1–6, Budapest, Hungary, July 2018.

[105] Pavan Pongle and Gurunath Chavan. A survey: Attacks on RPL and 6LoWPAN in IoT. In *Proc. International Conference on Pervasive Computing (ICPC)*, pages 1–6. IEEE, 2015.

[106] Anass Rghioui, Anass Khannous, and Mohammed Bouhorma. Denial-of-Service attacks on 6LoWPAN-RPL networks: Threats and an intrusion detection system proposition. *Journal of Advanced Computer Science & Technology*, 3(2):143–153, 2014.

[107] Wenjuan Li, Weizhi Meng, Xiapu Luo, and Lam For Kwok. MVPSys: Toward practical multi-view based false alarm reduction system in network intrusion detection. *Computers & Security*, 60:177–192, 2016.

[108] McAfee Labs Threats Report. https://www.mcafee.com/enterprise/en-us/assets/reports/rp-quarterly-threats-mar-2017.pdf, April 2017.

[109] Philokypros P. Ioulianou. Srf-iot framework source code. https://github.com/philok93/IoTIDS/, 2021.

[110] Google cloud automl custom machine learning models. https://cloud.google.com/automl/, 2021.

[111] Microsoft azure machine learning. https://azure.microsoft.com/en-us/services/machine-learning/, 2021.

[112] https://docs.microsoft.com/en-us/azure/machine-learning/how-to-select-algorithms, 2021.