João Miguel Coimbra Barros da Silva

Attack and Intrusion Detection on the IoT

Dissertation Report
Master in Informatics Engineering
advised by Prof. Doutor António Jorge da Costa Granjal
and presented to the Department of Informatics Engineering
of the Faculty of Sciences and Technology of the University of Coimbra

January 2018

• U C •

UNIVERSIDADE DE COIMBRA

Acknowledgements

I would like to thank to my supervisor, Prof. Doutor Jorge Granjal, who guided me through this thesis and for always giving me precious suggestions. Special gratitude to Prof. Doutor Nuno Lourenço, whose experience in machine learning was very helpful. To all my family and friends, thank you for always being there for me, and for putting up with me on my most grumpy moments.

"The best way out is always through"

by Robert Frost

Abstract

Nowadays, the Internet of Things (IoT) has a big impact on our way of living, making security a huge concern. In many cases, standard techniques like firewalls are unable to prevent intrusions, and thus, Intrusion Detection Systems (Intrusion Detection System (IDS)s) are required, either for detecting external or internal breaches of security. Currently, for the IoT, there are no ready-to-use IDS solutions based on pattern recognition.

The main goal of this work is to develop an Anomaly-based IDS for the IoT, operating at the Constrained Application Protocol (CoAP).

The new Anomaly-based IDS, AnIDS, is presented along with the architecture, software libraries, platforms, scenario and performance evaluation. Client CoAP level misbehaviors are programmed in Contiki, and features are calculated with Wireshark, a packet dissector. These features are pre-processed by standardization and the feature extraction algorithms Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA). The resulting data is processed using Support Vector Machine (SVM), a machine learning algorithm based on the Scikit-Learn package, in order to train a classifier. This classifier is evaluated with the plotting of Confusion Matrices and Receiving Operator Characteristic (ROC) Curves.

AnIDS allows to detect the signs of an intrusion, by cross checking the normal behavior, known *a priori* of the nodes, with a known data pattern of intrusion. AnIDS is also able to identify the type of misbehavior present on a node based on *a priori* data patterns of the specific intrusion.

The Client CoAP misbehavior implemented are: implicit Denial of Service (DoS), repeatedly Acknowledgement (ACK) sending, wrong URI requesting, and invalid CoAP content accept option requesting.

Results demonstrate that AnIDS can obtain an F_Measure and a Recall scores of $\sim 98\%$, demonstrating that the system is capable to distinguish between the normal and the intruder behavior. As CoAP is the most disseminated open source application protocol on the IoT, AnIDS can have a significant impact on IoT security.

Keywords— IoT, 6LoWPAN, CoAP, Machine Learning, IDS

Resumo

Na civilização atual, a Internet das Coisas (IoT) rege a nossa forma de viver, tornando a segurança desses aparelhos uma preocupação. Em muitos casos, técnicas standard como as *firewalls* são insuficientes, exigindo o recurso a Sistemas de Deteção de Intrusões (IDS) de forma a identificar intrusões externas e internas. Atualmente, para a IoT, não se encontram IDSs baseados em reconhecimento de padrões que estejam disponíveis.

O objetivo desta tese é desenvolver um novo IDS baseado em anomalias para a IoT, que opere ao nível da camada protocolar de aplicação, CoAP (*constrained application protocol*).

É apresentado um novo IDS, AnIDS, sendo descrita a sua arquitetura, bibliotecas de software, plataforma, cenário e as técnicas usadas para a avaliação da performance. Os comportamentos erróneos de clientes CoAP são programados em Contiki, e as características (*features*) são calculadas via Wireshark. Estas características são pré-processadas com standardização, para média 0 e desvio padrão 1, e são aplicados os algoritmos de extração de *features* PCA e LDA. Os dados resultantes permitem treinar o algoritmo de classificação SVM, presente na biblioteca Scikit-Learn. O classificador é avaliado através do cálculo de matrizes de confusão e curvas ROC.

O AnIDS permite detetar sinais de intrusões cruzando dados característicos de comportamento normal com dados de nós com um padrão conhecido de intrusão. O AnIDS também é capaz de identificar o tipo de comportamento erróneo presente num nó baseando-se em padrões de dados de intrusões específicas, previamente conhecidas. Implementam-se os seguintes comportamentos erróneos: DoS implícito, envio constante de ACKs, pedidos a URIs inválidos e pedidos com a *flag* de CoAP *accept* não suportada.

Os resultados obtidos para os indicadores *F_Measure* e *Recall* de $\sim$ 98%, demonstram que o AnIDS pode distinguir com elevado grau de certeza o comportamento normal do comportamento erróneo. Dado que o CoAP é o protocolo aberto mais disseminado da IoT, o AnIDS poderá ter um grande impacto na segurança nos dispositivos da IoT.

Plavras Chave— IoT, 6LoWPAN, CoAP, Machine Learning, IDS

Contents

Acronyms

6BR 6LoWPAN Border Router.

ACK Acknowledgement.

BLE Bluetooth Low Energy.

CoAP Constrained Application Protocol.

DoS Denial of Service.

GTS Guaranteed Time Slots.

IDS Intrusion Detection System.

IEEE Institute of Electrical and Electronics Engineers.

IETF International Engineering Task Force.

IoT Internet of Things.

IPv6 Internet Protocol version 6.

KNN K-Nearest Neighbours.

LDA Linear Discriminant Analysis.

LQI Link Quality Indication.

MAC Media Access Control.

ND Neighbour Discovery.

OS Operating System.

PCA Principal Component Analysis.

rbf Radial Basis Function.

ROC Receiving Operator Characteristic.

RPL Routing Protocol for Low Power and Lossy Networks.

SCP Secure Copy.

SM Security Manager.

SSH Secure Shell.

SVD Singular Value Decomposition.

SVM Support Vector Machine.

UART Universal asynchronous receiver/transmitter.

List of Figures

List of Tables

Chapter 1

Introduction

This chapter provides an introduction to this thesis. It describes the context, the research goals, the scheduling of the work, and the document structure.

1.1 Context

Our way of life is particularly dependent on the Internet of Things (IoT). IoT is everywhere, bringing computerization and connectivity to billions of devices such as private and public transports, home appliances, medical devices, webcams, power plants, or just smart light bulbs.

The low cost of these microcomputers comes with a price: reduced computing power and consequent difficulties in installing protection software. Another problem is that IoT network security is more complex than traditional network security because of the diversity of the communication protocols, standards, and device capabilities. Moreover, as they are connected to the internet, they are open to intruders through the Internet Protocol version 6 (IPv6).

IoT security breaches have become a hot topic, particularly since WikiLeaks's intrusion in CIA documents and the Dyn attack. These episodes revealed that the security vulnerabilities in the IoT are deep and pervasive, and may threat not only people's personal life but may also have a world-level impact. A recent paper demonstrates that even products apparently protected by standard cryptographic techniques are vulnerable [35]. So, in spite of the many communication protocols that have been developed, security remains an issue and Intrusion Detection Systems (IDSs) need to be applied.

Pattern Matching algorithms have been extensively used, namely *Signature Based* IDS Snort. Currently, for the IoT, there are no ready-to-use IDS solutions based on pattern recognition. Anomaly-based IDS is a promising solution, and it will be explored in this work, targeting the applicational layer protocol of the IoT CoAP.

1.2 Research Goals

The main gold of this Master thesis is to develop a new Anomaly-based IDS, AnIDS, for the IoT working at the Application protocolar Layer Level, CoAP. Secondary aims include:

- selection of an IoT platform to conceive a scenario necessary for training the classifier;
- identification of different types of intrusions;
- development an IDS with a higher performance, namely good accuracy, recall and F_Measure.

1.3 Document Structure

The document has the following structure:

Chapter 1 – Introduction

The first chapter is an introduction to familiarize the reader with the used concepts, and to contextualize the problem.

Chapter 2 – State of the Art

The second chapter provides the state-of-the-art on the security of IoT protocols that operate on the Internet.

Chapter 3 – System Architecture and Methodologies

The third chapter contains the Architectural features of the IDS, namely an high level view of the System, message formats exchanged among the devices, and the assumptions of the IDS being developed.

Chapter 4 – Implementation and Evaluation Strategy

Chapter four describes the implementation and the evaluation of the proposed solution.

Chapter 5 – Results and Analysis

Chapter five presents and discuss the performance results of the proposed solution.

Chapter 6 – Conclusion and Future Work

This final chapter presents the major conclusions, discusses impact and limitations of this work, and proposes future developments.

Chapter 2

State of the Art

2.1 Basic Concepts

For clarity, some important concepts used on this work are defined next according to [27]:

1. *Intrusion*: The action of trying to compromise Confidentiality, Integrity and Authentication (CIA) policies or to bypass the security techniques of a computer or a network;

2. *Intrusion Detection*: The process of monitoring the events occurring in a computer system or network and analyzing them for signs of intrusions. Since the Wireless medium is more susceptible to attacks, namely WDoS (Wireless Denial of Service) ones, the IDS are categorized as belonging to one of the two groups: Wireless-based or other technology types;

3. *Intrusion Detection System*: The hardware or the software that is responsible for automating the Intrusion Detection Task;

4. *Intrusion Prevention System* (IPS): A system that not only inherits all the capabilities of the IDS, but also has the capability of stopping the intrusion events. Normally, the authors fuse both the terms IDS and IPS, resulting on the acronym IDPS (Intrusion Detection and Prevention System) to refer to an IPS.

2.2 Protocols on the IoT and their Security

Limitations of the IoT devices make cryptographic algorithms and communication protocols to be hindered, and thus, they influence the design of the architectural structure. Those limitations include:

- low computing power capability;
- low available storage;
- constant power drain;
- non trusted random generators;

- short term relationships between the device and the others, that may never had communicated with them before;

Such limitations must be taken into account when defining the protocol standards for the IoT. Devices run their own Internet stack using different protocols depending on the layer. In this Section, the main protocols for the IoT and their security mechanisms are explained.

2.2.1 IEEE 802.15.4-2011

IEEE802.15.4-2006 is a physical layer protocol, defined by the Institute of Electrical and Electronics Engineers (IEEE) to support low-rate wireless personal area networks. It was originally planned to support critical industrial very low cost communication to nearby devices, with little or no underlying infrastructure, intending to exploit lower power consumption.

IEEE802.15.4 is also used on the Media Access Control (MAC) layer (that interacts with the radio) and defines the format of the Header of the MAC packet and the communication between motes (low computing nodes). IEEE 802.15.4 devices are recognized either by their 16-bit short identifier or by their 64-bit IEEE EUI-64. The 16-bit identifiers are employed by devices that are short on computing resources or operate on restricted environments.

The mechanisms for securing IEEE802.15.4 are applied only to the MAC layer (layer 2), though these mechanisms imply that the upper layers will be more protected.

From the standard [12], *ad hoc* networks are not significantly different from any other wireless network, being susceptible to passive eavesdropping attacks and active tampering, for the requirement of a wired connection is not mandatory in order to participate on the communication.

Security mechanisms on the PHY/MAC layer inherited by higher layer protocols, being based on symmetric-key cryptography, providing combinations of services as, data confidentiality, data authenticity and replay protection.

It is important to make clear that an IEEE 802.15.4 device is not required to implement security, since it is optional. The MAC Layer of the IEEE 802.15.4 establishes security procedures for:

- outgoing and incoming frames;
- KeyDescriptor, SecurityLevelDescriptor and DeviceDescriptor lookup;
- incoming security level checking;
- incoming key usage policy checking.

There are built-in tools for making intrusions on IEEE802.15.4 protocol such as KillerBee from the OS Kali Linux.

The IEEE 802.15.4 does not hang on a specific keying model, leaving to the administrator the choice of the most adequate model [22]. This selection depends on the threat model and on the scenario. A scenario is characterized, for example, by the applications that are currently running on the device and by its computing resources.

There are also issues regarding the reuse of the same Nonce value when the same Key is used in more than one ACL (Access Control List) entry. When using Stream Ciphers, the previous situation can facilitate the process of reverse engineering plain texts out of ciphered texts. The reuse of the keys may pose a huge problem, because group keying makes the repetition of the same key in multiple ACL entries more frequent, making the reuse of the Nonce value a security breach. In a similar way, network shared keying and replay protection are incompatible.

Moreover, IEEE 802.15.4 is not robust against a Distributed Denial of Service Acknowledgment (DDoS Acknowledgement (ACK)) attack, as the integrity or confidentiality is not applied to ACK packets. This allows the forgery of Acknowledgments, for which the attacker only needs to learn the sequence number of the packet to be confirmed, being the packet sent in plain text. Following [22], the use of key management mechanisms at higher layers of the protocolar stack, is a workaround on the management of the ACL limitations at the link layer in respect to the support of group and network shared keying. The optimized hardware of the nodes, that have the cryptography mechanisms optimized at hardware level, provides higher layers, application and network, strongly securing their payload.

2.2.2 International Engineering Task Force (IETF) 6LoWPAN

IETF 6LoWPAN is an adaptation layer, placed right between IPv6 and IEEE802.15.4e MAC layers, that allows link layer forwarding and fragmentation. The IPv6 header and Next Headers can be compressed, by crossing out redundant information that can be inferred from other layers of the stack. As in RFC 4944 [28], the adaptation layer itself does not provide any security procedures, depending of the IEEE 802.15.4 on the MAC layer. As stated on RFC 4944, although the Interface Identifiers from EUI-654 MAC addresses are intended to be unique, their replay or their forgery will not be detected by the IETF 6loWPAN protocol.

2.2.3 IETF Routing Protocol for Low Power and Lossy Networks (RPL)

RPL is a free of charge network layer protocol, assuring routing on most IoT devices. In the most common scenario involving RPL, the nodes of the network are connected through multi-hop paths to a small number of root devices, which are responsible for data gathering and management tasks. For each root device, a Destination Directed Oriented Acyclic Graph (DODAG) is built based on different parameters, such as link costs, motes' attributes and Objective Function. The Objective Function must be defined on each implementation of the IETF ROLL standard, since it is not provided by the protocol. Each DODAG is identified by an ID, the DODAGID, and the graph is built based on a Rank metric.This metric is monotonically decreasing along the DODAG and towards the target mote, based on a gradient approach.

According to RFC 6550 [14], RPL supports three security modes, namely:

- Unsecured, in which the basic messages like DIS (DODAG Information Solicitation), DIO (DODAG Information Object), DAO (DODAG Advertisement Object), and DAO-ACK for meshing configuration, do not carry Security Sections, relying on the lower level layer protocols for securing the frames;
- Pre-installed, in which a node that is expected to join the network has a pre-installed

key, becoming either a host or a router, guaranteeing message, confidentiality, integrity and authenticity.

- Authenticated, which similarly to the previous mode, it is based on a pre-installation of a key, being only allowed to become a host. In order to get promoted to router, it must obtain a second key from a key authority, which can authenticate that the requester is allowed to be a router before providing it with the second key.

The secured network uses the pre-installed or the authenticated mode; in either way, it will be signaled with a specific bit, 'A' bit, on the payload of the layer 3 or (network layer) packet. All RPL implementations must have the algorithm CCM (Counter with Cipher Block Chaining-Message Authentication Code). CCM requires AES-128 as the cryptographic algorithm.

When a node intends to join a secure Network, it is assumed it has been configured with a shared key for communicating with its future neighbours and the root of RPL. To do this, either the node listens for secure DIOs or triggers secure DIOs by sending a secure DIS. There are rules on RFC 6550 [14] that define the specific values of the DIS and DIO messages that must be present on each field of each packets.

The most problematic attacks that can bypass RPL protocol are the following [26]:

- *Rank Attacks*, which can lead to the construction of optimized paths, loop creations or optimized path disruption
- *Local Repair Attacks* in which an attacker would be able to change its rank to the highest value possible and broadcast it to all its neighbours, making them to find a new parent towards the root. The other alternative for the attacker would be to change its DODAG ID value, making the current network to assume the node changed to a different network, because each DODAG ID on a network is unique.
- *Resource Depleting Attacks*, that consists on depleting resources, such as computational and battery life, by imposing the execution of a large number of processes, for in RPL there is no limit to the number of actions.

There are open-source tools for simulating and generating attacks at RPL level: Contiki-IDS and RPL-attacks. The Contiki-IDS is a repository of an IDS at RPL level. It has built-in Cooja attack simulations, like worms and sinkholes. The Contiki version in which the IDS was built (2.6) is outdated. The RPL-attacks consists on a Python based framework built for simulating Hello-Flooding, Increased Rank andor Blackhole attacks.

2.2.4 Constrained Application Protocol (CoAP)

The IETF Constrained RESTful Environments (CORE) working group developed the Constrained Application Protocol (CoAP), a protocol that easily translates to HTTP for integration with the web, while respecting the specialized needs, such as low overhead and simplicity under constrained environments. Therefore, a set of mechanisms must be applied to compress application layer protocol packet, payload. CoAP works on top of UDP, holding the unreliable nature of UDP. HTTP and CoAP have different client/server models: synchronous for HTTP and asynchronous for CoAP.

CoAP messages fall in one of the following categories, Confirmable, Non-confirmable (being used for multicasting), Acknowledgment and Reset.

CoAP inherits the same CRUD (Create, Read, Update, Delete) operations as Representational State Transfer (REST): GET, (read), POST (create), PUT (update), DELETE. It also has the same HTTP code errors.

CoAP URI is similar to HTTP URI, allowing an easy migration to IPv6 networks. As in RESTful architectures, CoAP resources are organized hierarchically being the URI in the form of:

coap-URI = "coap:" "//" host [":" port] path-abempty ["?" query]

Proxying and caching are elements of men-in-the-middle attacks. These attacks ruin IPsec or DTLS, which are protection of systems of CoAP messages. Attacks to proxying and caching break confidentiality and/or integrity of messages exchanged. For the case of caching proxies, the scenario gets worse, for, contrarily to HTTP/1.1, CoAP has no caching control suppressing options.

Another security breach is the malicious increase of packet size, turning CoAP susceptible to DDoS attacks. If *NoSec* mode is used in a node, there is no way to verify the source address given to the request packet, as UDP does not provide any mechanism for it. So, an attacker needs only to type the destination IP address of its victim in a request packet, and amplify it. A network composed of many constrained devices may cause the impression that the network would not be able to generate a large ammount of traffic. However the target of the attacker is to constraint the network. The use of Multicasting on CoAP networks could also compromise the security of all network. Yet, on RFC 7252, all CoAP servers can only accept multicast requests either if their source is authenticated cryptographically, or if they do not cross a multicasting boundary limit. CoAP servers should limit the use of multicast to specific resources, depending on the scenario.

Regarding packet spoofing attacks (in which the attacker tries to assume another node's identity), as CoAP uses the handshake free UDP protocol, a rogue node is free to read and write any type of messages namely *reset*, *confirmable*, *request*, *response* or *observe*. However, forged response packets can be detected and fought without recurring to transport layer security, by choosing a randomized token in the request. "CON Packet Attacks" can be deployed towards the victims with the purpose of their Power Depletion. IP spoofing can also be used to block the CoAP server valid IPs that are being spoofed by an attacker. These valid IPs are wrongly considered as fake by the CoAP server. To avoid these attacks, CoAP servers should not use NoSec mode [13].

One of the most significant problems on CoAP security is the possibility of Cross Protocolar Attacks, even when an attacker, named *Mallory*, wants to target a node, Alice, but Mallory is currently blocked via a firewall rule from Alice. Also *Mallory* knows that *Alice* usually talks to *Bob*, a CoAP endpoint. A workaround to this security measure is:

1. *Mallory* sends a spoofed packet with Alice address to Bob;
2. *Bob* receives the spoofed packet sent by Mallory, and replies to *Alice*'s address;
3. *Alice* receives the packet from Bob, for it *"was"* Alice who sent the request;

This way, a DDoS attack can be easily executed, making *Alice* to block *Bob* instead of *Mallory*.

One more problematic scenario is the key installation process. For generating a good key, its entropy should be as high as possible, and preferably externally generated. Creating a Key Group for *N* clients will reduce the number of keys to be managed. However, it will increase the ratio at which entropy decreases, for same key is distributed by N clients. This makes easier for the attacker to crack the keys, as each single user of the group is a bottleneck of security.

Since we are only interested in open (non-proprietary) protocols, it is important to note that the Transport Layer of the IoT is currently merged with the Application layer. So, the Application Layer is considered filled with CoAP protocol. In [22] different security techniques and applications, exploring security properties of CoAP targeting transparent end-to-end security, are extensively described.

There are other IoT protocols, like Bluetooth Low Energy (BLE), WirelessHART, ZWave, LoRWaN, IEEE 802.11ah, ZigBee and MQTT. Besides not being open source, they all have security problems and have no clear advantages over open source protocols, like CoAP. Some examples of security problems are described in [21] for BLE, [33] for WirelessHART, [29] for IEEE 802.11ah, [37] for ZigBee. These protocols will be not discussed as they are not on the scope of this thesis.

2.3 IoT Operating Systems

For selecting the most appropriate Operating System (OS) for our work, some characteristics were taken into consideration, namely: project activity, community size, programming languages supported, devices supported in terms of chipsets and hardware requirements. The characteristics of the main Operating Systems for IoT are summarized in tables from 2.1 to 2.7, based on current literature. ([1, 3–10, 17–20]).

Table 2.1: ARM MBed analysis

Name:	*ARM Mbed*
Sum up:	platform and operating system for internet-connected devices based on 32-bit ARM Cortex-M microcontrollers;
Quality of Documentation	very good and guided documentation;
Programming Language	C/C++ language
Min. Hardware Requirments	256KB of Flash; 32KB of SRAM;
Devices & chipset Supported:	devices: mbed NXP LPC1768; chipsets: ARM, Atmel, BBC, Make it Digital Campaign, CQ Publishing Co.,Ltd.; Delta; Embedded Artists, Espotel; JKSoft; Maxim Integrated; MultiTech; NXP Semiconductors; Nordic Semiconductor ASA; Nuvoton; Outrageous Circuits; RedBearLab; Renesas; STMicroelectronics; SeeedStudio; Semtech; Silicon Labs; Switch Science Inc.; WIZnet; communitycontributors; u-blox AG;
Project / Community Activity	more than 10.000 questions put by developpers; Blog constantly updated, having 3 posts on March 2017; more than 6000 posts on forum;
Other Aspects:	develop applications with the free online code editor and compiler, using the ARMCC C/C++ compiler, providing private workspaces; Applications can be developed; The mbed software consists on core libraries that provide the microcontroller peripheral drivers, networking, RTOS and runtime environment, build tools and test and debug scripts; A components database provides driver libraries for components and services; Vast number of real life examples

Table 2.2: Contiki analysis

Name:	*Contiki*
Sum up:	uIP TCP/IP stack, uIPv6 stack, Rime stack. Multitasking kernel; Windowing system and GUI; Networked remote display using Virtual Network Computing; small web browser; Personal web server; Simple telnet client Screensaver
Quality of Documentation	easy instalation via Instant Contiki; very good and guided documentation;
Programming Language	Partial C - use of Protothreads, multithreading and event driven programming.
Min. Hardware Requirements	$RAM < 2kB$ (+ 30kB for GUI); $ROM < 30kB$ embeded systems to 8-bit computers;
Devices & chipset Supported:	Atmel – ARM, AVR NXP Semiconductors – LPC1768, LPC2103, MC13224Microchip – dsPIC, PIC32, Texas Instruments – MSP430, CC2430, CC2538, CC2630, CC2650, STMicroelectronics – STM32 W
Project / Community Activity	Over 150 contributors; last updated blog was on August 2015; Average of 150 mails each past 3 months;
Other Aspects:	Low battery consumption mechanism; Contiki instalation comes in a VMWare image, with a network simulator, cooja.

Table 2.3: Apache Mynewt analysis

Name:	*Apache Mynewt*
Sum up:	BLE and 6LoWPAN+CoAP protocols supported; preemptive multithreading; Semaphores; Mutexes; Event Queues; Memory Management; data buffers.
Quality of Documentation	instalation steps well guided; amateur style documentation;
Programming Language	C languages for coding apps; higher level languages to build apps.
Min. Hardware Requirements	Mynewt size: approx. 7.5kB ROM and 1.2kB RAM
Devices & chipset Supported:	nRF51—2 DK; RuuviTag; BLE Nano; BLE Nano2 and Blend2; BMD-300-EVAL-ES; STM32F4DISCOVERY; STM32-E407; Arduino Zero (Pro); Arduino M0 Pro; NUCLEO-F401RE; FRDM-K64F from NXP; Creator Ci40 IoT Kit; BBB micro:bit; Adafruit Feather.
Project / Community Activity	Over 30 contributors; 3 events on current year; Developers&Users contributions: Around 700 posts Posts a month.
Other Aspects:	Pros: Newtron flash file system (nffs) for minimal RAM usage and reliability; Cons:Non friendly documentation for advanced programmers.

Table 2.4: Raspbian analysis

Name:	Raspbian
Sum up:	Debian-based computer operating system, officially provided by the Raspberry Pi Foundation, as the primary operating system for the family of Raspberry Pi single-board computers; highly optimized for the Raspberry Pi line's low-performance ARM CPUs; uses PIXEL, Pi Improved Xwindows Environment, Lightweight as its main desktop environment as of the latest update; composed of a modified LXDE desktop environment and the Openbox stacking window manager with a new theme and few other changes;
Quality of Documentation	Extensively Documented, with very good start up guide, for any experience level; very extensive FAQ;
Programming Language	D language for developpers; python, scratch, mathematica for apps;
Min. Hardware Requirements	$RAM < 1MB$; $ROM < 1MB$;
Devices & chipset Supported:	over 6500 topics on the forum; very community support;
Project / Community Activity	very good reputation
Other Aspects:	

Table 2.5: RIOT OS analysis

Name:	*RIOT OS*
Sum up:	multithreading and realtime capabilities; network stacks: IPv6, 6LoWPAN, Content centric networking; standard protocols: RPL, User Datagram Protocol (UDP), Transmission Control Protocol (TCP), and CoAP
Quality of Documentation	Average quality of Documentation for installation : github wiki based; very good for developers
Programming Language	fully support for C and C++
Min. Hardware Requirements	$RAM < 1.5kB$; $ROM < 5kB$;8-bit, 16-bit, and 32-bit processors.
Devices & chipset Supported:	devices: Airfy Beacon, Arduino Due, Arduino Mega 2560, Atmel samr21-Xplained Pro, f4vi, HiKoB FOX, mbed NXP LPC1768, MSB-IoT, Nordic nrf51822 (DevKit), OpenMote, Spark-Core, STM32F4DISCOVERY, STM32F3DISCOVERY, STM32F0DISCOVERY, ScatterWeb MSB-A2, +K1ScatterWeb MSB-430H, TelosB, Texas Instruments cc2538 Developer Kit, Texas Instruments EZ430-Chronos, UDOO Board (Cortex-M3 part), WSN 430 (v1.3b and v1.4), yunjia-nrf51822, Zolertia Z1; chipsets: MSP430, ARM7, Cortex-M0, Cortex-M3, Cortex-M4, MIPS32, x86
Project / Community Activity	140 contributors; around; around 1500 issues on the past 4 years, in which over 1100 were closed; Twitter blog in constant update;
Other Aspects:	very well documented; native port enables RIOT to run as a Linux or OS X process, enabling use of standard development and debugging tools such as GNU Compiler Collection (GCC), GNU Debugger, Valgrind or Wireshark; RIOT is partly Portable Operating System Interface (POSIX) compliant; cooja integration

Table 2.6: TinyOS analysis

Name:	*TinyOS*
Sum up:	fully non-blocking; based on software components, being some hardware abstractions. Components are connected to each other using interfaces. TinyOS provides interfaces and components for common abstractions such as packet communication, routing, sensing, actuation and storage
Quality of Documentation	sparse documentation;
Programming Language	Optimized C language exploting sensors' limits
Min. Hardware Requirements	$RAM < 1kB$; $ROM < 4kB$
Devices & chipset Supported:	Devices: btnode3; epic; exp_msp432; eyesIFXv1; eyesIFXv2; intelmote2; iris; mega2560; mica2; mica2dot; micaz; mulle; null; nxp_jn516; sam3s_ek; sam3u_ek; shimmer; shimmer2; shimmer2r; span; telosa; telosb; tinynode; ucbase; ucmini; ucprotonb; z1;
Project / Community Activity	39 contributors; over 270 issues where over 200 were closed as of 2012; claiming to have '35,000 downloads each year'
Other Aspects:	Java and shell-scripts interfaces for easy programming.

Table 2.7: Zephyr analysis

Name:	*Zephyr*
Sum up:	early members and supporters of Zephyr include Intel, NXP Semiconductors, Synopsys, and UbiquiOS Technology; Single address-space, where both the application code and kernel code execute in one shared address space as from version 1.6.0; Highly configurable; Resources defined at compile-time; Minimal error checking; Development services
Quality of Documentation	Documentation consolidated in one place;
Programming Language	C language
Min. Hardware Requirements	$RAM < 2kB$
Devices & chipset Supported:	X86 Boards, Arduino/Genuino 101, Galileo Gen1/Gen2, MinnowBoard Max, X86 Emulation (QEMU), Quark D2000 Development Board, tinyTILE ARM Boards, 96Boards Carbon, 96Boards Nitrogen, Arduino/Genuino 101 (BLE), Arduino Due, CC3200 LaunchXL, CC3220SF LaunchXL, Curie (BLE), ST Disco L475 IOT01, NXP FRDM-K64F, NXP FRDM-KL25Z, NXP FRDM-KW41Z, Hexiwear, Hexiwear KW40Z,ARM V2M MPS2, nRF51-PCA10028, nRF52840-PCA10056, Redbear Labs Nano v2, Nordic nRF5x Segger J-Link, nRF52-PCA10040, ST Nucleo F401RE, ST Nucleo F411RE, ST Nucleo, L476RG, OLIMEXINO-STM32, ARM Cortex-M3 Emulation (QEMU), SAM E70 Xplained, ARM V2M Beetle, ARC Boards, Arduino/Genuino 101 (Sensor Subsystem), DesignWare(R) ARC(R) EM Starter Kit, NIOS II Boards Altera MAX10, XTENSA Boards, Xtensa simulator
Project / Community Activity	143 contributors; over 2100 issues, with over 1500 solved and 597 open; 6 recorded events in 2016, according to the blog
Other Aspects:	Real life examples derived from Zephyr - a motion sensing tooth brush, called Grush;

After comparing all OS we select Contiki for the 6BR, CoAP server and CoAP clients. This choice is justified by the documentation and simulation tools provides, by its software stability and hardware compatibility, as described in Table 2.2 and also because it is adopted by many researchers.

2.4 Intrusion Detection on the Internet

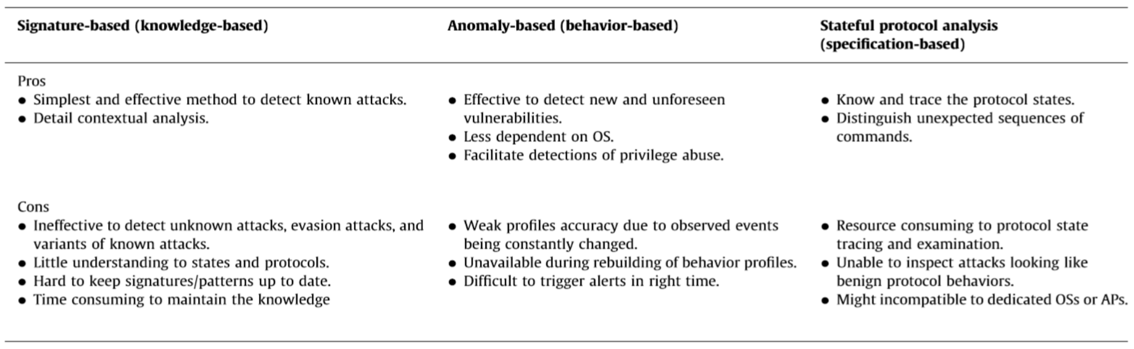
This section describes the techniques required to detect an Intrusion in the standard Internet.

2.4.1 IDS Methodologies

An IDS can be classified in one of three major categories:

- SD *(Signature-based Detection)*, also known as *Knowledge-based Detection* or *Misuse Detection*, in which a signature is a pattern or a string that corresponds to a known attack or threat. SD is the process of comparing patterns against captured events for recognizing possible intrusions;
- AD *(Anomaly-based Detection)* or *behavior based*; an anomaly is a deviation from a normal behavior; a normal behavior, also known as profile, represents the expected behavior derived from monitoring regular activities, network connections, hosts or users over a period of time. Profiles can be either static or dynamic and is defined by a set of attributes. Typical attributes of a profile include, for example, the frequency of key strokes, the number of different files accessed on a given time or the number of incorrect attempts on the login screen. By comparing the profiles with observed events, intrusions are recognized with a good accuracy;
- SPA *(Stateful Protocol Analysis)* or *Specification-based*, which is based on tracing the protocol states, depends on the generic profiles developed to specific protocols.

There is also the Hybrid approach in which complementary methodologies are followed, like using both SD and AD, in order to detect known attacks and to warn of unknown threats, respectively. Figure 2.1 states the advantages of the three major IDS methodologies. Though theoretically SPA could be the most effective, it is almost impractical, as the state machine of the protocol implementation can be very large. Also, the analysis of the state machine can take a very high execution time.



Signature-based (knowledge-based)	**Anomaly-based (behavior-based)**	**Stateful protocol analysis (specification-based)**
Pros		
• Simplest and effective method to detect known attacks. • Detail contextual analysis.	• Effective to detect new and unforeseen vulnerabilities. • Less dependent on OS. • Facilitate detections of privilege abuse.	• Know and trace the protocol states. • Distinguish unexpected sequences of commands.
Cons		
• Ineffective to detect unknown attacks, evasion attacks, and variants of known attacks. • Little understanding to states and protocols. • Hard to keep signatures/patterns up to date. • Time consuming to maintain the knowledge	• Weak profiles accuracy due to observed events being constantly changed. • Unavailable during rebuilding of behavior profiles. • Difficult to trigger alerts in right time.	• Resource consuming to protocol state tracing and examination. • Unable to inspect attacks looking like benign protocol behaviors. • Might incompatible to dedicated OSs or APs.

Figure 2.1: Pros and Cons of IDS Methodologies, from [27].

In general, the Anomaly-based methodology is the elected one. The stateful-protocolar-analysis is by far the least used methodology.

According to [27], there are five different IDS technologies: HIDS (Host-based IDS), NIDS (Network based IDS), WIDS (Wireless Based IDS), NBA (Network Based IDS) and MIDS

(Mixed based IDS). The MS is responsible for processing captured incidents, whereas the DS is responsible for storing event information.

An HIDS monitors and collects the characteristics of hosts containing sensitive information, servers running public services, and suspicious activities. A NIDS captures network traffic at specific network segments through sensors, and subsequently, analyzes the activities of applications and protocols to recognize suspicious incidents. WIDS is similar to NIDS, but captures wireless network traffic, such as *ad hoc* networks, wireless sensor networks and wireless mesh networks. When multiple technologies are used, as MIDS, the security goal can be fulfilled for a more complete and accurate detection.

The previous authors [27] also compare IDS detection approaches, classifying them in five groups. These approaches are characterized according to the detection methodology, technology type, and other features. The detection approaches described are:

- Statistics-based, consisting on analysis of network packets, audited data and user profiles. This approach uses Anomaly Detection as the prevailing detection methodology, and is based on Bayesian and distance analysis. The NIDS technology is preferred and its source is mostly audited data. These kind of approaches are stated as simple, and real-time active, though less accurate;
- Pattern-based, in which Signature Detection is more frequently used than Anomaly Detection. This approach is simple but less flexible; it makes use of keystroke monitoring and file system checking. The HIDS technology is the most prevailing on this approach;
- Rule-based, having lower false positive rate and a high accuracy, though it is very hard to have its rules created or updated. The NIDS technology is the elected choice on this approach. Support Vector Machine (SVM) is one of the algorithms used on this approach, along with other data mining techniques;
- State-based, in which Anomaly Detection is more common than Signature-based, having been used with a Stateful-Protocol-Analysis methodology. State-based approach is described as having a low false positive rate but also to be less effective. It recurs to Markov Chains and user intention identification. The HIDS and the NIDS have been equally used.
- Heuristic-based, being fault tolerant, self learning and scalable flexible, have their foundations based on machine learning techniques, like neural-networks, fuzzy logic and genetic-algorithms.

The components of an IDS include a *sensor* and an *agent*. One major drawback of IDS technologies is that they cannot guarantee 100% accuracy in intrusion detection. In real world scenarios, security administrators prefer to reduce false negatives (the errors in classification where malicious events were classified as being benign), even if that means an increase of false positives. Figure 2.2 describes the taxonomy of IDS [27] with great detail. IDSs can be categorized according to the type of System Deployment, Data Source, Timeliness and Detection Strategy.

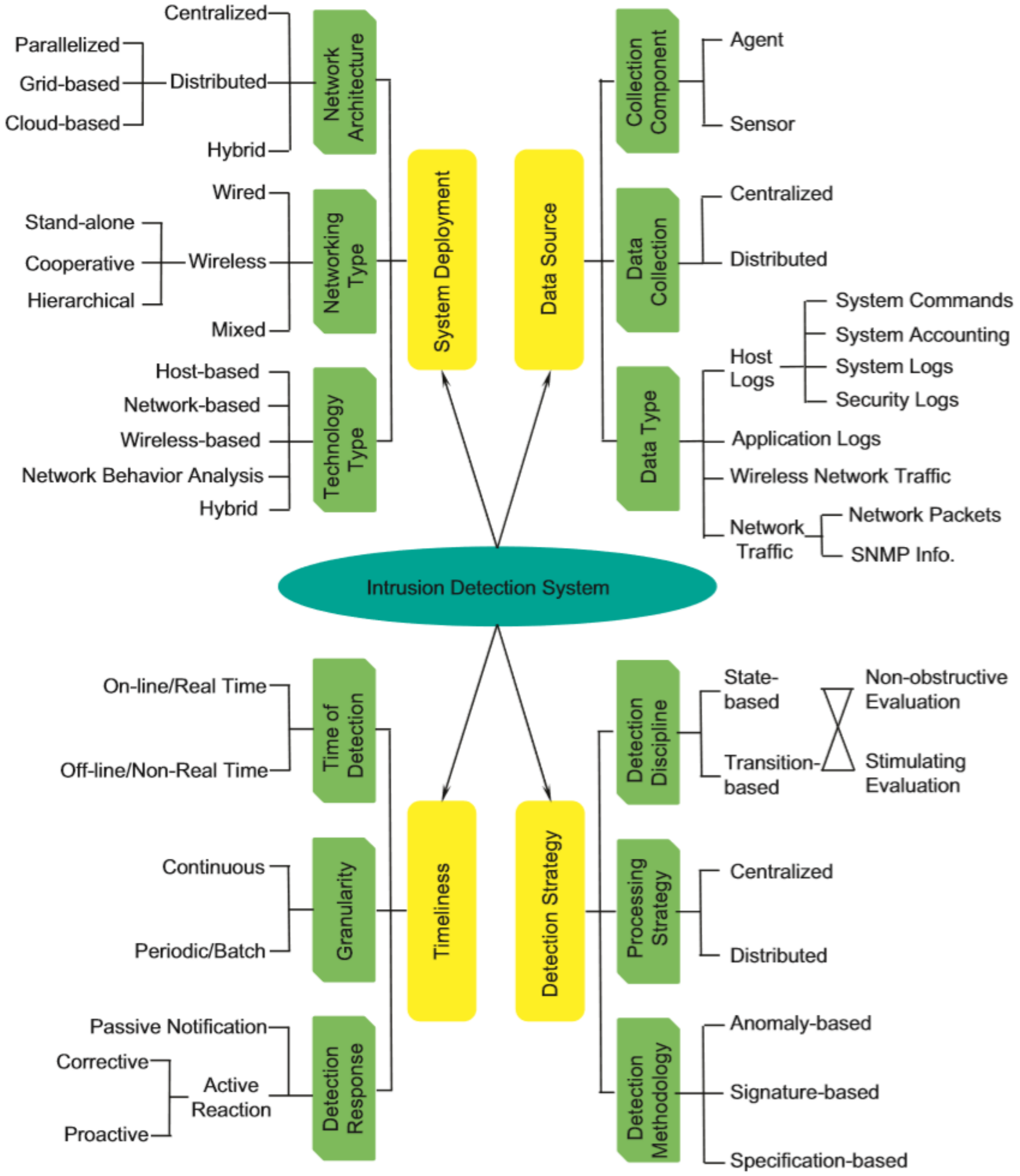


Figure 2.2: IDS Taxonomy, from [27].

2.5 Intrusions and Detection Systems on the IoT

Figure 2.3 gives a simplified overview of IoT intrusions, classified by protocol layers (Physical, MAC, Network, Transport and Application) and groups of protocols (IEEE 802.154, 6LoWPAN RPL and CoAP, corresponding to the three coloured areas).

An example of an attack is Sybil that consists in overpowering a mote in order to gain access to his resources. When the Sybil node is localized on MAC Layer, the effect is to change the transmission link, but when localized on the Network Layer, the effect is to create a fake route. In some cases, like with Misdirection and Internet Smurf Attack, the intrusions have different purposes, but the procedure is the same. The Flooding, the De-synchronization and the Internet Smurf Attack operate by flooding the target.

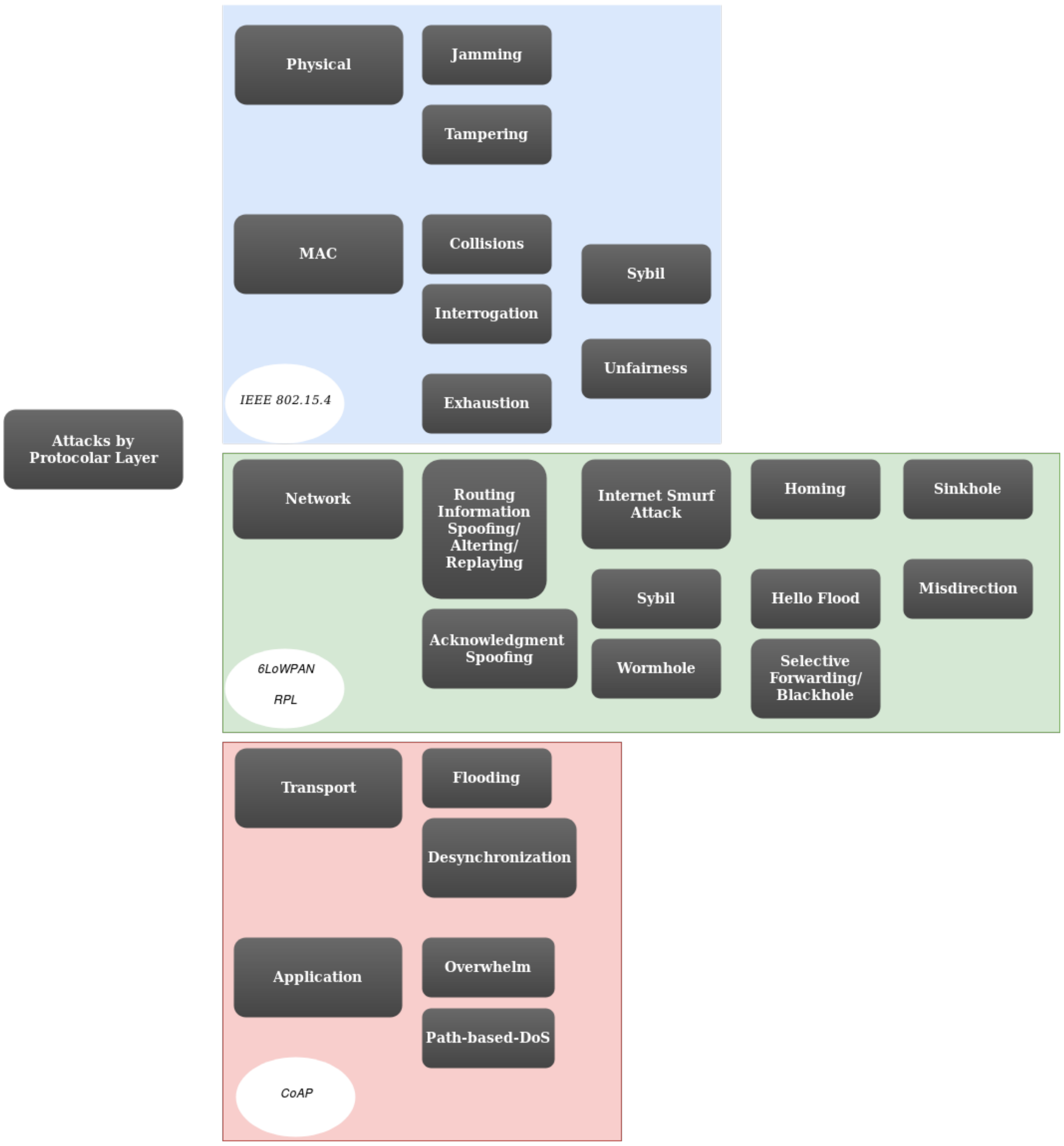


Figure 2.3: Intrusion Taxonomy by Protocolar Layer. Adapted from [26].

Many of these intrusions may be considered as Denial of Service intrusions (DSI). A Denial of Service (DoS) attack consists of bombarding a central resource located on a public network with normal requests, in such a way that resources rapidly become depleted. These intrusions are particularly difficult to be detected by Signature-based IDS. DoS is easily detected by the Anomaly-based IDS, the approach chosen on this thesis.

2.5.1 Denial of Service Detection

There are several types of DoS [25]:

- *Jamming attacks*, exploiting transmission of radio signal interference, frequencies being used by the sensor network;
- *Cloning of Things attacks*, which consists on replacing nodes during their maintenance phase, like a software update, by another pre-programmed version of the same physical node. A node can also be captured during its operational phase. In both situations, an avalanche of new security failures can occur, such as extraction of security parameters or firmware injections;
- *Eavesdropping attacks*, which relies on a sloppiness of the communicating parties, like exchanging security keys in plain text; and, even if the exchanged material is ciphered, one can deploy a cryptographical analysis or a brute force attack, reducing the entropy of the keys, and easing the decipherment of the messages. This can possibly evolve to a Man-in-the-Middle attack.
- *Routing attacks*, by spoofing, altering or relaying routing information of a grid of devices in order to create communication loops or repelling network traffic. Flooding, sinkhole, selective forwarding, wormhole attacks also fall into this category.
- *Application attacks*, as the most used applicational protocol, CoAP, is still very young, there are many security issues that arise and will arise in the future; some of the known vulnerabilities are protocol parsing, SYN flood, proxying, or cross-protocol attacks.

A DDoS (Distributed DoS) is a slightly different approach to the DoS attack. In DDoS the number of attackers is higher and the sending frequency of each of the attackers is lower, making it more difficult to be detected.

Two solutions for DoS attacks have been proposed, but neither of them uses CoAP: one is based in 6LoWPAN and another is based in detection in ebbits networks.

The first solution, proposed by [30], prevents DoS attacks against low powered devices, originated from the Internet, by putting all the protection on the more capable devices, the edge routers. The mechanism is based on the address registration process defined in the Neighbour Discovery (ND) protocol of 6LoWPAN. Accordingly to [30], the traffic is forwarded from the Internet to the 6LoWPAN devices if the following rules are met:

1. The destination node address must be registered at the edge router, so that traffic does not become forwarded to non existing nodes;
2. The nodes must previously declare the willingness to accept data from the Internet. In this way, nodes that should not be addressed from outside, will not be contacted, like the 6LoWPAN routers.
3. The transport protocol to be used by the device must be registered at the edge router, so that no unexpected traffic can reach the device.

4. Nodes must previously inform the edge router about the accepted traffic rate limit, in order to avoid flooding attacks.

This mechanism is fairly easy to implement, being only required to change the three fields on specific types of messages. As this solution uses stateless traffic processing, it supports a scalable architecture, by giving the chance of implementing the mechanism on multiple edge routers, thus turning the network more robust. Lastly, the compression rates of the 6LoWPAN are not compromised, for the non-compliant messages have their reserved fields with a 0 value.

The second solution [25] proposes an architecture for an automated system that detects intrusions on IoT devices connected to ebbits networks [24]. These networks provide a platform for connection IoT devices remotely located, offering a security manager service. The architecture of ebbits network is given in Figure 2.4.

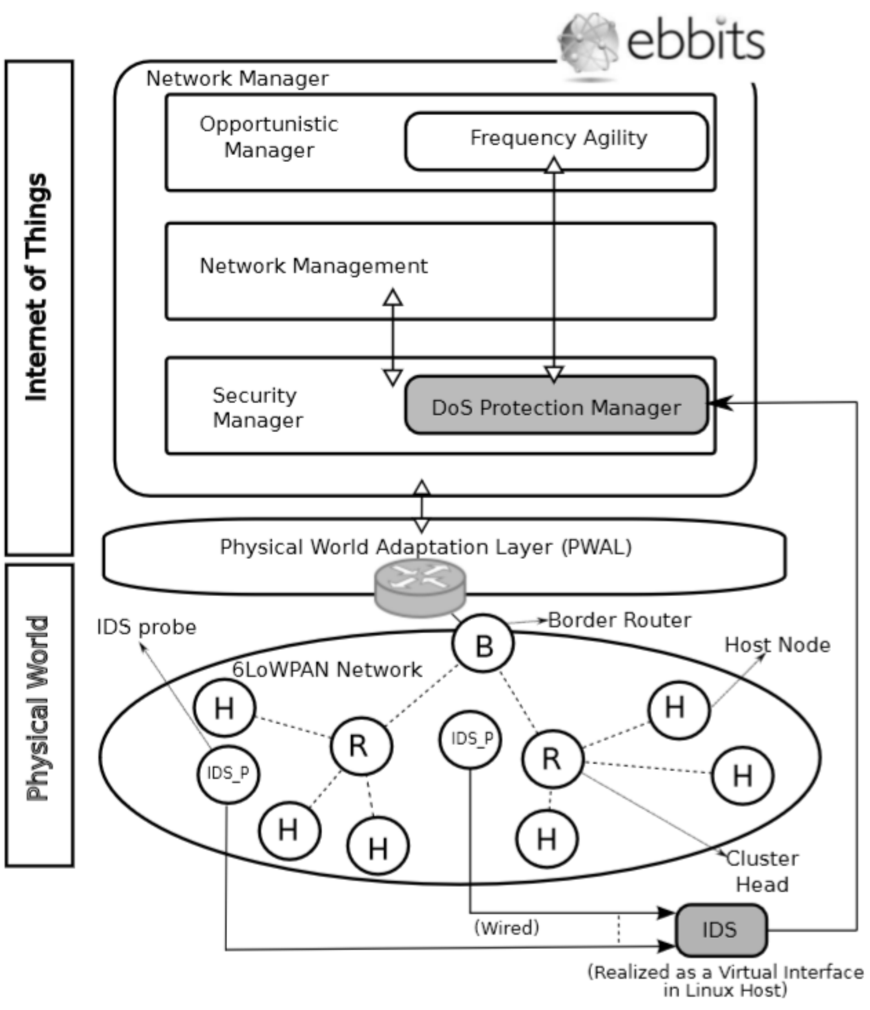


Figure 2.4: DoS Detection Architecture, from [25].

The architecture includes the Physical world and the ebbits Network Manager. The Network Manager includes a Security Manager that provides security mechanisms, encryption and policy enforcement. It also enables secure communication between ebbits Network Manager and the Physical World. The Security Manager has two main components, the IDS and the DoS protection Manager. The IDS is composed by a set of probes, the IDS_P devices, that have *promiscuous* mode on and capture wireless packets. Probes are connected to a machine that runs an open-source IDS, Suricata. The DoS protection Manager receives alerts from IDS when intrusion attempts occur. It also collects feature parameters such as interference rate and packet drops from the other ebbits managers (opportunistic and Network Management) and performs data cross to confirm an actual attack.

The efficiency of the ebbits Network security system was evaluated using penetration testing. For creating the scenario of the penetration test, the actual attacker was a node running Contiki Operating System, flooding the network via Wireless medium. The IDS used was Suricata. Suricata had a rule relying on packet rate from any source to any destination, so that if a pre-defined rate threshold was exceeded, an event in Suricata would be triggered.

In the ebbits security system, the DoS Protection Manager relies on both the IDS and data from the Opportunistic and Network Management, thus reducing the false alarm ratio. But, Suricata is a Signature-based detection IDS, and so, it depends on a set of predefined rules to detect such attacks. Such rules still need to be inserted manually by an expert on security. This restricts the ebbits solution to the rules created by the expert.

2.5.2 Hybrid IDS

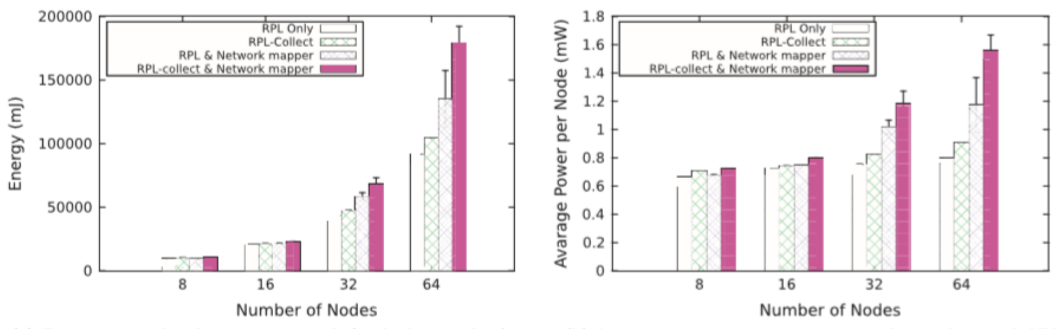
As mentioned before, hybrid systems combine different approaches, like Signature-based detection and Anomaly-based detection, to identify known attacks and to warn about unknown threats, respectively. Two main hybrid systems have been described: SVELTE [34], and RIDES (Robust Intrusion Detection System) [11].

SVELTE combines Signature and Anomaly-based approaches. It is composed by three main centralized modules:

- The 6LoWPAN Mapper, placed on the 6BR, reconstructs the RPL DODAG in the 6LoWPAN Border Router (6BR). The Mapper starts by sending requests to nodes in the 6LoWPAN network at regular intervals, being the request constituted by the RPL Instance ID, DODAG ID, the DODAG Version Number and a timestamp. In return, each node responds by prepending its Node ID to the request message, and by appending its own rank, its parent ID and all its neighbour IDs and their ranks.
- The IDS Component, installed on the 6BR and on all the constrained nodes, follows an hybrid approach. The IDS detects and corrects the RPL DODAG inconsistencies, detects Filtered Nodes, find rank inconsistencies and adapts to End-to-End Losses. As shown in [34], this IDS can detect spoofed/altered information, sinkhole and selective forwarding attacks.
- A distributed mini-firewall, installed on each constrained node and on the 6BR, provides blocking functionality against well-known external attackers, specified by the network administrator. The firewall blocks external malicious hosts communications to the nodes inside the 6LoWPAN network. The hosts are specified in real-time. When a constrained node notices it is being attacked by an external host, the node sends a packet with the host IP to the firewall module in the 6BR. However, it could happen that the attacker would target a different node and in that way, circumvent the blockade of communication to a specific node on the 6LoWPAN network. To prevent such situation, the mini-firewall is extended to adapt and block any external host, providing a minimum set of nodes complains about this same external host.

In [34] an experiment with Contiki's simulator Cooja in Tmoke Sky nodes was described.

The attacker was given the possibility to spoof or alter information, and/or perform sinkhole or selective forwarding attacks. It was verified that the TP (true positives) for sinkhole attacks in a lossless network were nearly 100% and no false positives were detected during simulation. As for the lossy network, the TP were almost 90%. The TP ratio decreased as the network size (number of motes) increased. The energy overhead of SVELTE running RPL protocol (Figure 2.5) was measured with Contiki's Powertrace application. It is interesting to note that there is an exponential growth of the energy consumed by the entire network during the 30 minutes of operation, as the number of nodes increases. Also, the average consumption remains constant until a certain amount of nodes is reached.



(a) Energy usage for the entire network (with *duty cycling*) in 30 minutes.

(b) Average power consumption per node in a *duty cycled* RPL-based network

Figure 2.5: SVELTE Energy overhead, from [34].

In terms of memory consumption, SVELTE only produces an overhead of 1.76k of RAM, which is quite below the ROM of constrained devices such as the 48k in Tmote sky (Figure 2.6).

Configuration	Total ROM (byte)	Overhead (byte)
6Mapper client	44,264	1414
Firewall client	43,556	0246
Packet loss improvement	43,264	0122
6Mapper server (1 node, 1 neighbor)	46,798	3580
6Mapper server (8 node, 1 neighbor)	46,798	3846
6Mapper server (16 nodes, 1 neighbor)	46,800	4152
6Mapper server (16 nodes, 8 neighbors)	46,924	4724

Figure 2.6: SVELTE Memory Overhead, from [34].

RIDES is a Hybrid IDS, that combines Signature-based detection relying on a predefined ruleset, with Anomaly-based detection where the IDS can learn the malicious patterns of an attacker. This hybrid solution was conceived to deal with attacks, such as PoD (Ping-of-Death), that only require a small amount of packets to subvert the target. In such cases, the Anomaly-based detection would fail, leading to a high percentage of false negatives, for single packet attacks are not detected. This solution is based on two main components:

- *SCG* (Signature Code Generator), which uses Bloom Filters for storing Snort signature codes of different sizes, in an attempt to minimize false positives, while occurrences of false negatives are close to 0%.

- *NAD* (Network Anomaly Detector), which uses CUSUM (cumulative sum control chart) for detecting the abnormal network activity. The control limits of CUSUM can be defined by two methods: V-mask, which is a visual procedure, and Tabular CUSUM. Tabular CUSUM is preferred for computer implementations.

The Snort is an open source Signature-Based IDS. RIDES' architecture is illustrated in Figure 2.7. The evaluation of the performance of RIDES show that with a storage memory

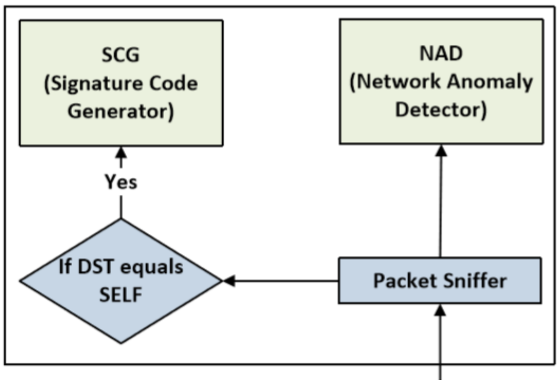

SCG (Signature Code Generator)
NAD (Network Anomaly Detector)
Yes
If DST equals SELF
Packet Sniffer

Figure 2.7: RIDES Architecture, from [11].

of 8 kbits, a sensor can decrease false positives to 10%. The probability of not experiencing a hash collision will increase as the number of signatures decreases. It is also shown [11] that the signature codes can reduce energy consumption by 8 µJ. Finally, this work shows that the true positive ratio increases as the time interval between packets also increases. The number of attackers is negligible.

2.5.3 Anomaly-based IDS

Anomaly-based IDSs can recognize normal and erroneous behavior by learning the data patterns of normal and erroneous network activity. This methodology creates a model that can classify traffic as either normal or erroneous. Unlike Signature-based IDS, Anomaly-based IDS are not based on a predefined set of rules, saving the Security Manager the effort to find the appropriated signatures for blocking malicious traffic.

Another Anomaly-based IDS, [15], provides a centralized solution for preventing forgery of traffic that is forwarded to a potentially infected node, the botnet, that acts as a router to the remaining network. This solution is based on the assumption that such device is also communicating with the source of the attacker, and so, packets that go through the infected node are expected to be larger than the non tampered ones, for the source must update the bot in order to make the desired forgery. For such solution to be executed, three conditions are essential to the scenario, namely, restraining protocolar communications only to TCP, limiting the number of services running on each node to one and assuming the normality on the communication between the final user and the 6LoWPAN devices. The detection mechanism must be placed on the gateway, the edge router. The architectural components are depicted in Figure 2.8.

As the TCP protocol is not complying with the CoAP protocol, the [15] does not fulfills the aim of this work. Moreover, it is not clear in what application layer protocol it operates.

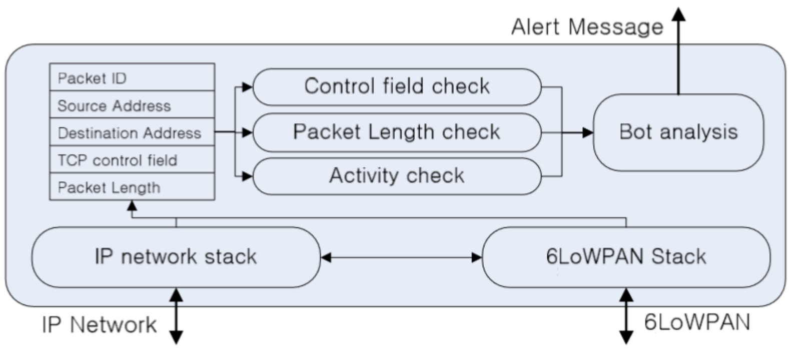


Figure 2.8: Solution against Botnet, from [15].

A different Anomaly-based solution [23], named WIDS (Wireless IDS), uses agents with intelligence that constantly perceive and analyze events. Figure 2.9 illustrates the architecture of this solution. The advantages of this methodology include, fewer errors due

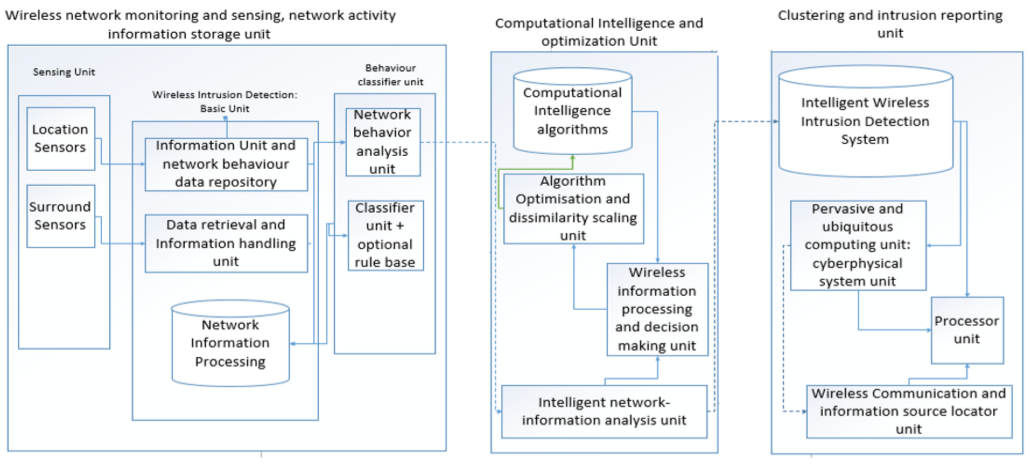

Wireless network monitoring and sensing, network activity information storage unit
Sensing Unit
Location Sensors
Surround Sensors
Wireless Intrusion Detection: Basic Unit
Information Unit and network behaviour data repository
Data retrieval and Information handling unit
Network Information Processing
Behaviour classifier unit
Network behavior analysis unit
Classifier unit + optional rule base
Computational Intelligence and optimization Unit
Computational Intelligence algorithms
Algorithm Optimisation and dissimilarity scaling unit
Wireless information processing and decision making unit
Intelligent network-information analysis unit
Clustering and intrusion reporting unit
Intelligent Wireless Intrusion Detection System
Pervasive and ubiquitous computing unit: cyberphysical system unit
Processor unit
Wireless Communication and information source locator unit

Figure 2.9: WIDS architectural scheme, from [23].

to the use of computational intelligence, easier to detect newer malicious activities, no need for a rule-based dataset and intrusions are treated as a continuous stream in a system. This solution, however, is not described with enough details in order to allow its implementation.

2.5.4 Wormhole Attack Detection Solution

A wormhole attack consists on disrupting communications through the creation of a tunnel between two or more nodes that are geographically far away. By communicating through such tunnel, a group of nodes, the selfish nodes, make the remaining nodes of the network believe that they are closer, in terms of topology, than they really are. If succeeded, the selfish nodes will disrupt routing, by making routes to pass through the tunnel, due to the fact that each of the selfish nodes seems to be closer. But in fact, they are significantly geographically separated. So, the flow of all packets through the tunnel will cause a bottleneck, increasing the delay on each of the selfish nodes' networks.

In [31], an IDS to detect wormhole attacks of packet encapsulation and relay types is described. Its architecture is shown in Figure 2.10.

The IDS network architecture is hybrid (Figure 2.2), consisting in a centralized module

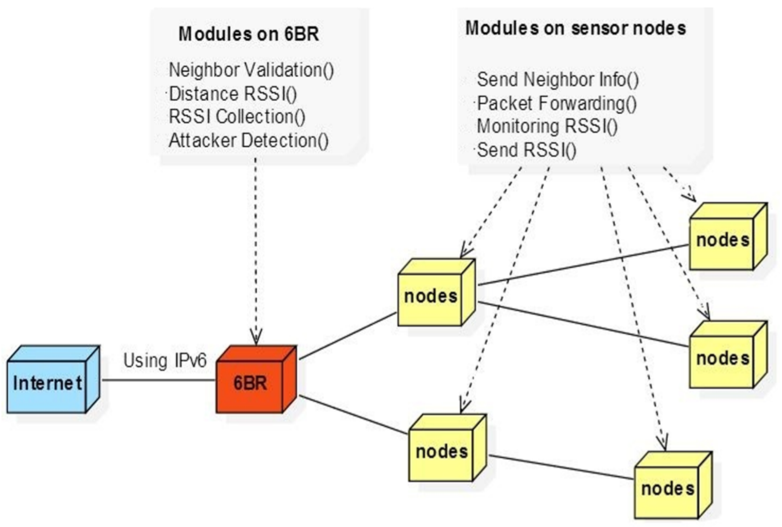


Figure 2.10: Pongle architectural scheme for detecting wormhole attacks, from [31].

on the 6BR, preforming the following operations: neighbour validation, distance RSSI (Received Signal Strength Indicator), RSSI collection and attacker detection. Concerning distributed network architecture, the sensor nodes perform the following operations: neighbour information sending, packet forwarding, RSSI monitoring and RSSI sending.

The scenario was simulated in Cooja/Contiki simulator. The true positive ratio obtained is $\sim 90\%$. This IDS is not suited to an IoT scenario. There is no point in creating a tunnel between a user and an IoT device on IEEE802.15.4-CoAP stack, for, not only the sensor is already connected to the Internet (and so it has open access), but also there are computing power limitations on IoT devices to support tunneling.

2.5.5 A Deep Packet Inspection (DPI) based Solution

Deep Packet Inspection (DPI) consists on observing the payload for each protocolar layer of the packet for required information. The solution described in [38], claims to be a high-performance ultra-lightweight deep-packet Anomaly-based detection approach. Thus, this solution can be run on small IoT devices. This DPI-based solution uses n-gram bit-patterns to model payloads in an efficient and flexible manner and allows the n-gram size to vary by dimension. By using a direct representation of the feature space for the discrimination function, the detector can make a fast packet classification decision. The approach can be implemented in hardware or software and has abundant parallelism to be exploited for an effective implementation. The approach can be deployed in an IoT device's network interface or protocol stack, or it can be built into network appliances and firewalls. The DPI-based solution can operate in a wide-range of network environments and is highly configurable and scalable.

The analysis of results have shown that small IoT devices use few and relatively simple protocols, leading to network payloads that are highly similar and therefore amenable to Anomaly detection with an extremely low occurrence of false positives. The detectors proved to be highly effective in identifying abnormal packets from a wide-range of attacks, and they have an excellent ability to discriminate device-specific traffic from other types of Internet traffic.

2.5.6 An IDPM Solution

A multilayer and effective intrusion detection and prevention mechanism (IDPM) is an IPS designed for low-power IoT systems [36]. The system has two layers. In the first layer, an intelligent Anomaly detection model is built using a RNN (Random Neural Network) to handle performance degradation attacks. In the second layer, a lightweight compile-time code instrumentation technique is implemented to protect programs from illegal memory accesses. The proposed solution also acts as a sensor node health monitoring system, and can detect the failure of a valid sensor node.

The feasibility of the proposed solution is demonstrated for a wireless sensor nodes–based IoT system, with a detection accuracy of 97.23% [36]. This solution was further tested by adding an attacker sensor node and generating different security attacks that are eventually detected. The proposed IDPM does not require dedicated hardware resources and presents negligible performance overhead with 10.45% increase in the power consumption.

This solution is not described with enough detail, preventing its implementation. Also, it does not comply to CoAP.

2.6 Conclusions

As perceived by the state of the art, both major used approaches, Signature and Anomaly-based, have pros and cons. Signature-based approaches, although with a faster decision mechanism, do not recognize new threats and have the need for creating/updating the rules (signatures) in order to block undesired traffic. On the contrary, Anomaly-based IDS can learn new data patterns from normal profiles, and so, they make easier the task of Security Managers, freeing them from generating models. In fact, the expertise required to map manually the models generated by Anomaly-based into rules (signatures) would be daunting.

In the next chapters, we describe a new Anomaly-based IDS, AnIDS, providing its architecture, implementation and validation.

Chapter 3

System Architecture and Methodologies

This chapter describes the structure of the IDS to be implemented and the selected sign of intrusions detection, misbehavior. Misbehaviors are implemented on Contiki CoAP client. We choose to program these misbehaviors because there were no available open-source intruder tools on IEEE802.15.4-CoAP stack. Additionally, no dataset containing data of intrusions on the IoT stack was found.

3.1 Architecture

The architecture is based on the assumption that a single node, the one that will be in charge of the IDS module, is capable of running a network sniffer application, in conjunction with a high level language interpreter. Chapter 4 describes the tools that are used on the process. The main point is that the Controller is a high processing and storage capacity node, a sink, such as a Raspberry PI, that makes decisions regarding routing and forwarding on behalf of the other nodes. It is also assumed that the sink has every node on the 6LoWPAN network radio covered.

In a CoAP application context, the most common scenario, when one 6BR serves as the door to the Internet, and one or multiple CoAP server nodes have a set of finite number of resources, is configured. Client nodes are linked to server nodes and they can request, actuate or observe, server resources. The proposed architecture of the IDS is depicted in Figure 3.1.

A multi-hop scenario, in which any pair of in-range nodes could communicate, was initially considered. However, due to physical restrictions related to the placement of the nodes, it was not implemented. Moreover, this work considers only GET requests due to limitations of the hardware. Section 4 explains the tools that are used and their limitations.

The IDS's role is to detect a compromised node, and to stop communication to and from that node, by spreading alert messages to the remaining nodes and/or filtering the packets to and from the compromised node. So, the IDS must be centralized (Figure 3.1).

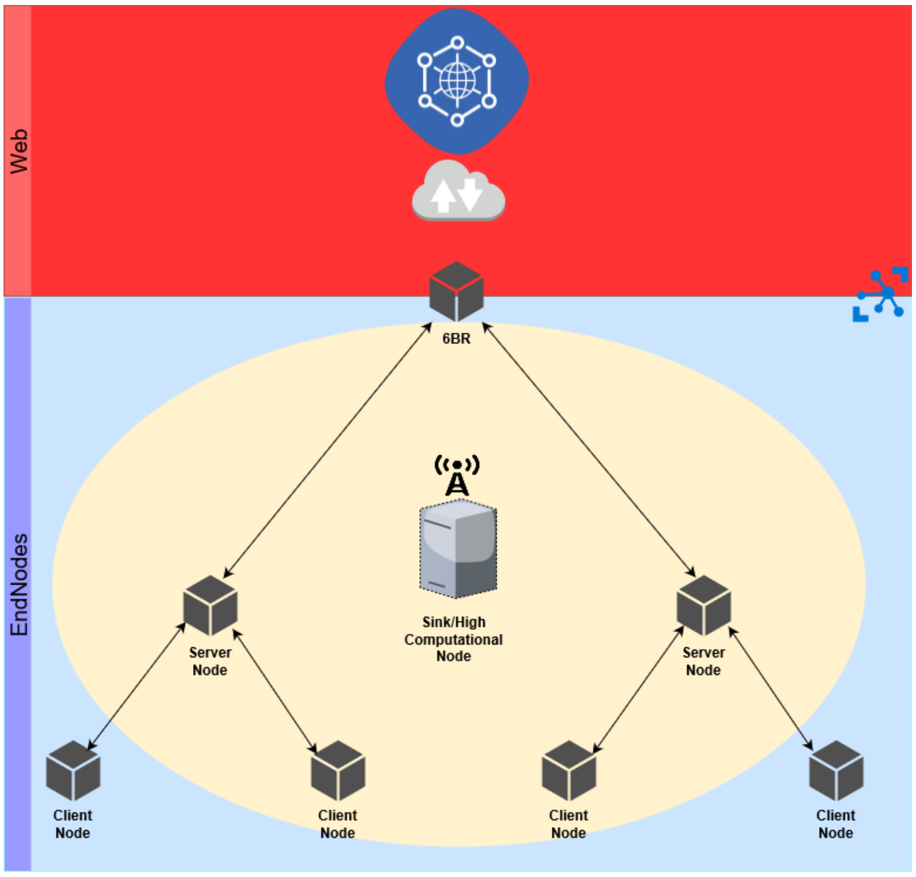


Figure 3.1: High Level Architecture of the proposed solution

3.2 Misbehavior to be detected

Compromised nodes are detected by the presence of erroneous behaviors. Examples of erroneous behaviors include: draining all power battery from the hacked node; and/or forcing the remaining normal nodes to process messages that should not be processed. These misbehaviors could lead to energy depletion in all nodes and/or compromise network performance. The following erroneous behaviors are considered for CoAP clients:

- Having a higher rate of CoAP requests at the respective Server, named here by *DOS_FREQ*, label 2;
- Sending CoAP acknowledgements to the CoAP server with no previous request to CoAP requests, named here by *DOS_ACK*, label 3;
- Requesting resources that are not supported/not existent on the CoAP Server, named here by *WRONG_URI*, label 4.
- Sending requests with an invalid *ACCEPT* option to the CoAP Server, named here by *WRONG_ACCEPT*, label 5.

We define that the normal behavior, named as NORMAL, corresponds to label 1. The misbehaviors implemented are known for not being detected by simple Signature-based IDS approaches.

3.3 Model Learning and Detection Methodology

The architecture for the IDS solution is described in Figure 3.2.

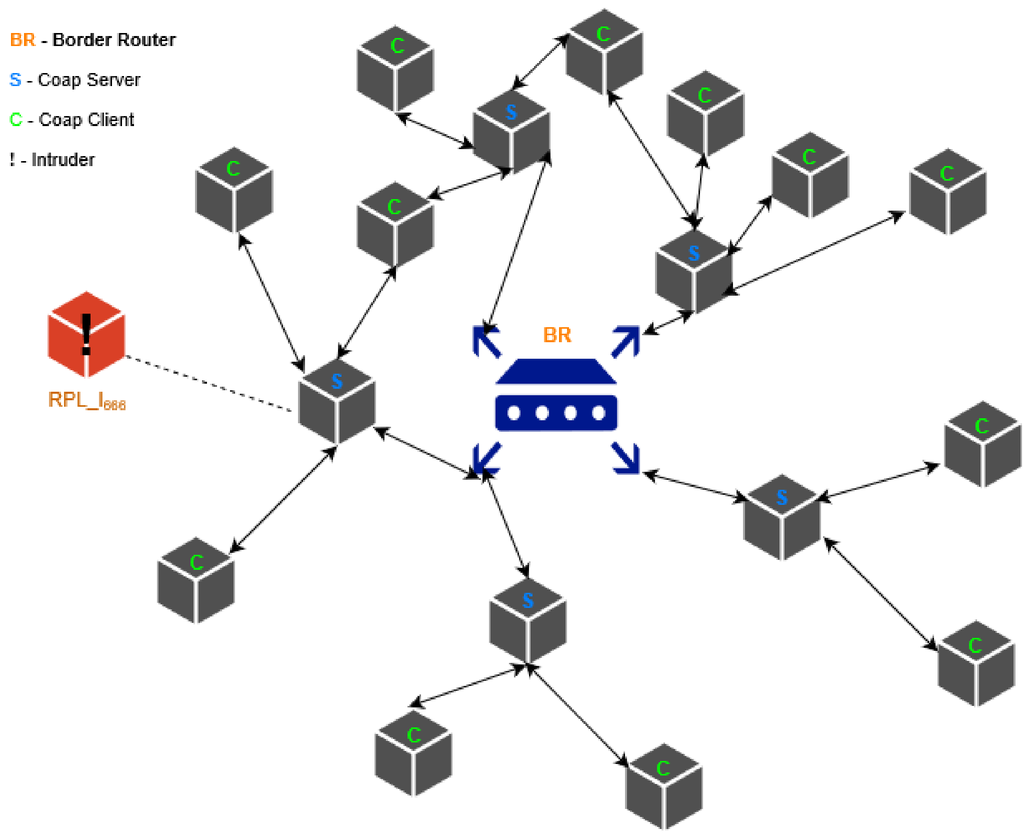


Figure 3.2: Probing Scheme for the proposed solution

The components of the architecture are:

- The 6BR node, responsible for interconnecting all the nodes to the Internet.
- The Server node (in case of just one), with its clients, each Server Node providing access to its own CoAP resources. Pairs of Server Nodes do not communicate with each other.
- The Client nodes, the ones who request CoAP resources, being the request rate controlled by a Timer.

The previous scenario is simplified in Chpater 4, restricting the resource requests of clients to one server.

Of the three different *approaches* for the intrusion detection problem, *Multi-Class*, *Binary-Class* and *One-Class*, only the first two are used. On *Multi-Class* we are interested on knowing which intrusion the system has detected, assuming that we know data patterns of specific intrusions. The *Binary-Class* approach is a simplification of *Multi-Class* that assumes the problem is simply to detect if there is an intrusion or not. The *One-Class* approach is used when we only know the data pattern of non-intruded nodes or the data pattern of intruded nodes, exclusively. This approach is not considered here, because it is less powerful than other approaches. Also, for a better performance of the classifier, the security manager should know the intrusion and the non-intrusion patterns.

For *Multi-Class* approach, the pre-processing algorithms used are Standardization and MinMax, and the feature extraction algorithms used are Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA). The algorithm for the learning IDS is the SVM, as it is a state of the art classifying algorithm and used for IDS of the Standard Internet. According to [27], SVM are classified as Rule-Based and Heuristic-Based approaches. Neural Networks were not considered, because of their excessive computational requirements and their time consuming learning and classification process. K-Nearest Neighbours (KNN) algorithm, due to the lazy learning characteristic and the relatively high number of observations of the training data set that are required for this work [32], would be too slow and is not considered. On the contrary, SVM have a faster classification procedure, allowing a real time implementation and also have the advantage of being non-linear classifiers.

For the *Binary-Class*, we follow the same procedures used on the Multi-Class approach, except for the number of labels used. For reducing the number of labels to two, misbehavior labels were converted to a single label. For example, the array of labels $[1, 2, 3, 2, 1, 4, 5, 3, 2]$ is converted into $[1, 2, 2, 2, 1, 2, 2, 2, 2]$, following the nomenclature as of Section 3.2.

On both classification approaches, *Multi-Class* and *Binary-Class*, the whole data is pre-processed, either via Standardized methodology with mean 0 and variance 1, or scaled to the interval $[0, 1]$.

The calculation of the features and the learning methodology are depicted in Figure 3.3.

The steps marked as 3 occur simultaneously, meaning that when the calculation of the features are completed, the new data will be available for classification. At the same time, the network starts to be sniffed again by the Sink node. In step 4, a label will be returned. This label is computed by applying the learned model to the test data. The *splitting of packets*, the calculation of features and their aggregation will be discussed on the next chapter. Performance results of the IDS system are presented and discussed on chapter 5.

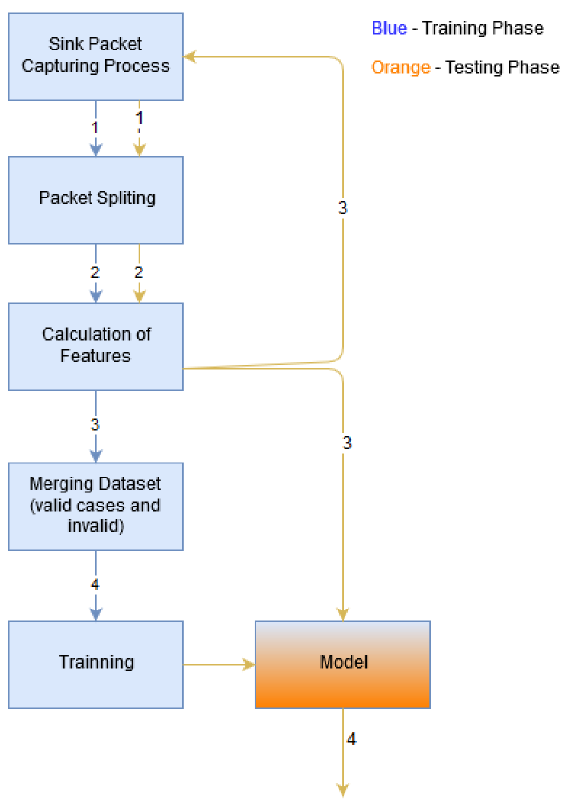


Figure 3.3: Learning Methodology Approach chosen

On this chapter, it was stated that the architecture of the system would follow a CoAP common design, with one 6BR, one CoAP server and multiple CoAP clients, being each of the clients making requests to the same server. Also, client misbehaviors were implemented, in order to train a classifier that can make a distinction between normal and erroneous traffic. Lastly, the overview of the model building and of the detection methodology was defined.

Chapter 4

Implementation and Evaluation Strategy

4.1 Implementation Strategy

For the implementation of the specified IDS, we need to select: the operating system(s) for nodes (clients, servers and 6BR), the platform in which the scenario is built, the statistical data gatherer and the intrusion detection component.

For the IDS operating system, the IoT Contiki was selected because of its widely acceptance within the research community, good simulation tools, software stability and hardware compatibility, as previously described (Table 2.2). The simulation tool for WSN NeSSi2 was discarded, for it does not simulate IEEE802.15.4-CoAP stack.

For the experimental environment (set up of nodes) and platform, we first tested the physical Hardware Micaz and Telos motes for client and server nodes. These devices have a built in IEEE802.15.4 radio chip and very little ROM and Flash memory. For the 6BR, the first choice was a Raspberry PI with the 802.15.4 radio module. The PI was directly connected via ethernet cable to a router supporting Internet Protocol version 6 (IPv6). The process of compiling a Contiki program and sending the image to each of the nodes to form a 6LoWPAN network was problematic. Occasionally, motes could not connect with each other, imposing to rely on logging information (obtained by connecting the PC directly to each mote and by capturing the printed messages that were being sent via the Universal asynchronous receiver/transmitter (UART) output).

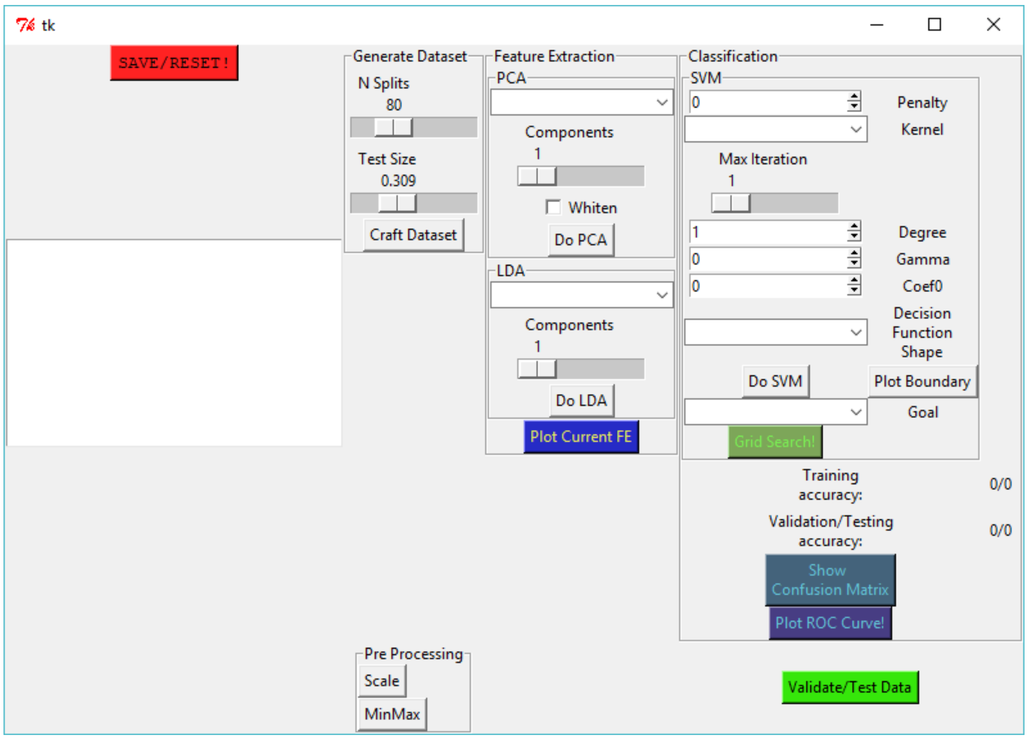
Searching for a better solution, we used Cooja, the simu/emulation environment embedded on Contiki project. Although the number of simulated types of nodes was smaller than the physical nodes, *configuring, placing and debugging* was easier. Contiki has a *radio capture plugin*, which can capture the packets sent by each of the simulated nodes. Still, there was a problem: the CoAP server and client firmwares from the version 3.0, the most up to date, did not fit into some of the supported Cooja motes, like Sky or Micaz. Additionally, a simulation environment oversimplifies real scenarios.

As a final choice, we used the platform that combined the advantages of both physical and the Simu/Emulation scenarios: IoT-LAB [2]. IoT-LAB is a platform that allows researchers to build real physical scenarios without having to handle neither with the difficulties of flashing the motes nor with the need of configuring the gathering of the resultant experimental data. As the nodes are physical and have real sensors, real time values from any of the available nodes can be read. The available sensors are: pressure, temperature, magnetometer, accelerometer, gyrometer and light. The motes used on IoT-LAB were the M3 Open Node for the availability of network sniffers and power tracing features.

For debugging purposes we kept using Cooja before launching each experiment in IoT-LAB, for it allows logging output messages, and thus, has advantages to evaluate the execution of the program.

Wireshark version 1.12 was chosen as the Packet Statistical Gatherer and Exporter. This tool is one of the best open source packet dissectors for IEEE802.15.4 networks. The version 1.12 was firstly chosen in order to be compatible with the Latest version available for Instant Contiki3.0, the virtualized environment with the most up to date Contiki software distribution building tools.

The Machine Learning application, running on the local computer, intends to classify nodes as being *Normal* or *Compromised*. Depending on the misbehavior of the compromised node, the node is labeled following the label nomenclature described on Chapter 3. This application is programmed on Python version 2, consisting on a GUI designed in Tkinter, via the pygubu designer. The library for the machine learning application is the Scikit-learn, and the plots were generated with matplotlib. On the GUI, by tuning the adequate parameters, one can choose to do stratified splitting of the dataset into traintest parts with the desired proportion and preform pre-processing, feature extraction and classification. There is also a functionality for performing a Grid Search for the best parameters of the classifier regarding the desired Scoring goal. Figure 4.1 shows the GUI developed for training the IDS.



N Splits 80
Test Size 0.309
PCA Components 1
LDA Components 1
Penalty 0
Max Iteration 1
Degree 1
Gamma 0
Coef0 0
Training accuracy: 0/0
Validation/Testing accuracy: 0/0

Figure 4.1: Developed GUI for tuning the Detection Module of the IDS

4.2 Implementing and Automating the Experiments

As stated in the previous Section, IoT-LAB was the elected experimental platform for the current solution. The access to my home directory of the linux server supporting IoT-LAB was made via Secure Copy (SCP) and Secure Shell (SSH). Scripting via Python 2 made possible a high degree of automation in configuring the nodes and collecting data. Figure 4.2 illustrates the methodology for building the experimental scenario used in conjunction with IoT-LAB.

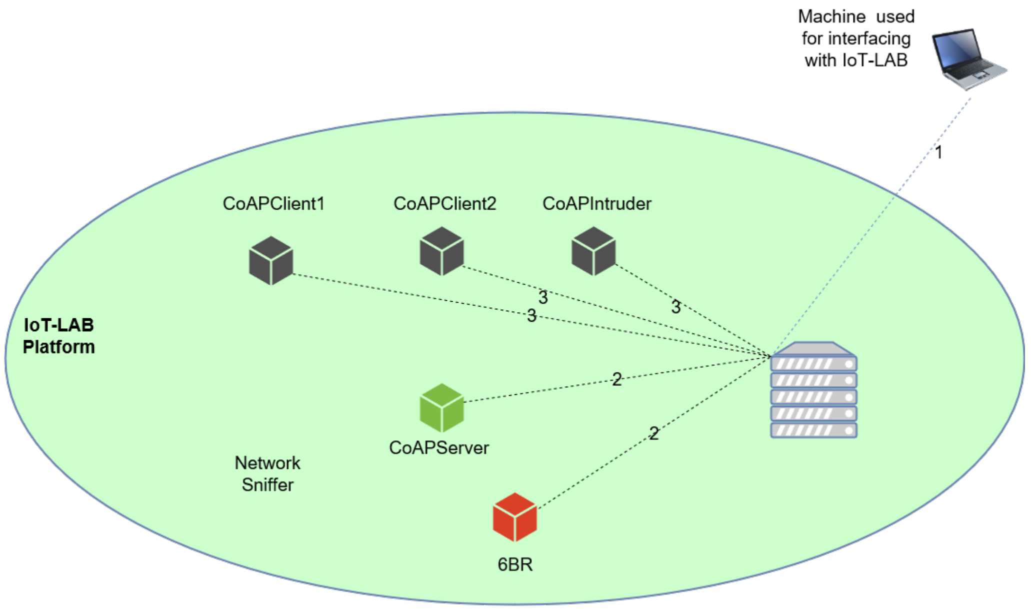


Figure 4.2: Steps used for connecting the local computer to the IoT-LAB server and for launching the experiments.

The experiments require the following steps:

- Step 1 - connect the local computer to the IoT-LAB server and select the nodes from the platform located in Grenoble, France. Grenoble was chosen for it has a large number of available M3 Open nodes.
- Step 2 - submit the experiment to the IoT-LAB, specifying the firmware and a profile for the server and the 6BR nodes. On the case of the clients, only a profile is associated, because the OS image of the client nodes is flashed after knowing the IPv6 address of the CoAP Server.
- Step 3 - replace the hard-coded IPv6 address on the client nodes code by the IPv6 address of the CoAP server, compiling the source code of the client and flashing the resultant firmware to the nodes.

When the experiment starts, an Secure Shell (SSH) console is opened and a daemon command is run to make a bridge on the 6BR node, between the 6LoWPAN network and the Internet. This step is mandatory, otherwise the server will not respond to the clients. For capturing the packets, another shell must be open to run a daemon command that aggregates all traffic of the experiment.

Each experiment consisted in a 6BR, a CoAP Server and three CoAP clients, being one of the three an intruder. The number of CoAP servers was reduced, due to node positioning limitations of IoTLAB platform. Each experiment had a duration of 30 minutes, and a total of four experiments were run, each with its misbehavior node.

After the experiment is finished, the PCap file is analyzed by TShark, the terminal application of Wireshark. TShark is executed with the statistical flag via Python 2 language scripting and calculates the features. TShark can collect network data during a customized period of time, and at the end of this period, performs an analysis. For the training phase, the approach was to split the entire training PCap file into subfiles with the same duration of the sampling period, using Editcap utility from Wireshark.

Currently, only client misbehaviors have been implemented. However, the Python scripts we have developed can also detect misbehaviors from servers and 6BR. This can be performed by calculating features of the network packets. It is only needed to specify what is the role of the intruder, client, server or 6BR. In each experiment a set of features are computed from the PCap file associated with the analyzed traffic network. The meaning of the following features is explained by the protocol Standard documents: [12] for IEEE802.15.4, [28] for 6LoWPAN, [16] for IPv6, and [13] for CoAP.

For IEEE802.15.4 traffic these features are:

- $wpan-nonask-phy.frame\_length$ - Frame Length
- $wpan.aux\_sec.frame\_counter$ - Frame Counter
- *wpan.bcn.gts.count* - Guaranteed Time Slots (GTS) Descriptor Count
- *wpan.cmd.gts.length* - GTS Length
- *wpan.correlation* - Link Quality Indication (LQI) Correlation Value
- $wpan.frame\_length$ - Frame Length
- *wpan.gtsreq.length* - GTS Length
- $wpan.sec\_frame\_counter$ - Frame Counter
- $wpan.sec\_key\_sequence\_counter$ - Key Sequence Counter

For 6LoWPAN traffic, the features are:

- $6lowpan.frag.size$ - Datagram size
- $6lowpan.fragment.count$ - Message fragment count
- $6lowpan.hc2.udp.length$ - Length
- $6lowpan.hops$ - Hop limit
- $6lowpan.iphc.hlim$ - Hop limit
- $6lowpan.mesh.hops$ - Hops left
- $6lowpan.nhc.ext.length$ - Header length
- $6lowpan.reassembled.length$ - Reassembled 6LoWPAN length
- $6lowpan.udp.length$ - Length

For IPv6 traffic, the features are:

- *ipv6.flow* - Flow Label
- *ipv6.fragment.count* - Fragment count
- *ipv6.hlim* - Hop Limit
- *ipv6.opt.calipso.cmpt.length* - Compartment Length
- *ipv6.opt.jumbo* - Payload Length
- *ipv6.opt.length* - Length
- *ipv6.opt.rpl.sender_rank* - Sender Rank
- *ipv6.plen* - Payload Length
- *ipv6.reassembled.length* - Reassembled IPv6 length
- *ipv6.shim6.len* - Length
- *ipv6.shim6.opt.elemlen* - Element Length
- *ipv6.shim6.opt.len* - Length
- *ipv6.shim6.opt.total_len* - Total Length

Finally, for CoAP traffic, the features are:

- *coap.opt.block_size* - Encoded Block Size
- *coap.opt.length* - Options Length
- *coap.opt.length_ext* - Options extended Length
- *coap.opt.max_age* - Max-age
- *coap.token_len* - Token Length
- *coap.code* - Status Code

From each of these features, except for the *coap.code*, the tuple $(sum, min, max, average)$ is calculated. For *coap.code* we count the number of message codes, from and to each node of the network, containing the following specific status: *error*, *not supported* and *valid*.

4.3 Previous Approaches

For calculating traffic statistics we also tested the following utilities found on the web:

- Netmate based on Flowcalc;
- Argus;
- Tcpstat;

- Tcptrace;
- Tstat;
- Tcpdstat
- Capanalysis
- YAF-Yet Another Flowmeter;
- CapLoader - Not Free to use
- Nop-ng - Not Free to use
- Scapy -Python framework for pcap analysis and packet manipulation.

However, none of these tools could capture or recognize IEEE802.15.4-CoAP stack, and so could not be used for the implemented IDS.

When planning the best strategy for obtaining data from attacks in order to train the IDS, a search on the Internet was performed to verify if there were open to the public intrusion datasets for IEEE802.15.4-CoAP available. The following sites were evaluated:

- Netresec - Not Relevant;
- KDD-Cup-99 - Not Relevant;
- UNB - Not Relevant;
- CAIDA - Not Relevant;
- ADFA - Not Relevant;
- MIT - Not Relevant;
- WAND - Not Relevant;
- RIPENCC - Not Relevant;
- UMASS - Not Relevant;
- Predict - Not Relevant;
- AWID - Request for dataset is pending;
- HS-Coburg - Not Relevant;
- secrepo - Not Relevant;
- ISI - Not Relevant;
- Honeynet - Not Relevant;
- FitnessLab - Public dataset, although the protocol used for Routing is AODV;
- UTWENTE - Not Relevant;

As none of these sites contained dataset for IEEE802.15.4-CoAP, we had to build our own intrusions.

4.4 Final Product Characteristics

According to the Internet IDS taxonomy described by [27] (Figure 2.2), the characteristics of the IDS developed are:

- System Network Architecture: Centralized, for it is chosen is the Client-Server CoAP model.
- Networking type: Wireless Hierarchical
- Technology Type: Wireless Based
- Collection Component: Agent, for IoT-LAB sniffer captures the traffic in a specific node of the Network.
- Data Collection: Centralized for the capture of the network traffic is done in a specific node of the network.
- Data Type: Wireless Network Traffic, stored in pcap file format.
- Time of Detection: currently off-line;
- Granularity: Periodic/Batch
- Detection Response: Planned to be Passive Notification
- Detection Discipline: state based and Stimulating Evaluation, for the IDS tells if a node is in the state NORMAL or INTRUSION.
- Processing Strategy: Centralized
- Detection Methodology: Anomaly-based

4.5 Evaluation Strategy of the IDS

The classifiers of the IDS are evaluated during the training phase. Scoring metrics of performance include:

- $Accuracy = \frac{TP+TN}{n}$
- $Recall = \frac{TP}{TP+FN}$
- $Precision = \frac{TP}{TP+FP}$
- $F\_Measure_{\beta} = \frac{Precision*Recall}{\beta^2*Precision+Recall}$

were evaluated in confusion matrices and Receiving Operator Characteristic (ROC) curves, either for the Multi or for the Binary Class approaches, (see Chapter 3). TP stands for the number of true positives, TN for the number of true negatives, FN for the number of false negatives, FP for the number of false positives and n the number of observations.

Concerning the Binary Class approach, the system can be in one of two states or conditions: either with an intrusion present, I, or with no intrusion present, NI. The prior probability of an intrusion is called p. The IDS reports either an intrusion alarm, A, or no alarm,

NA. The parameters of the IDS's ROC curve are: the probability of an alarm given an intrusion $P(A|I) = 1 - \beta$, and the probability of an alarm given no intrusion, the false alarm probability, $P(A|NI) = \alpha$.

Our approach is just a proof of concept. In a real scenario, the security manager would have to attribute costs to wrong classified observations. This implies the creation of an IDS's Decision Tree expected cost, like the one illustrated on Figure 4.3.

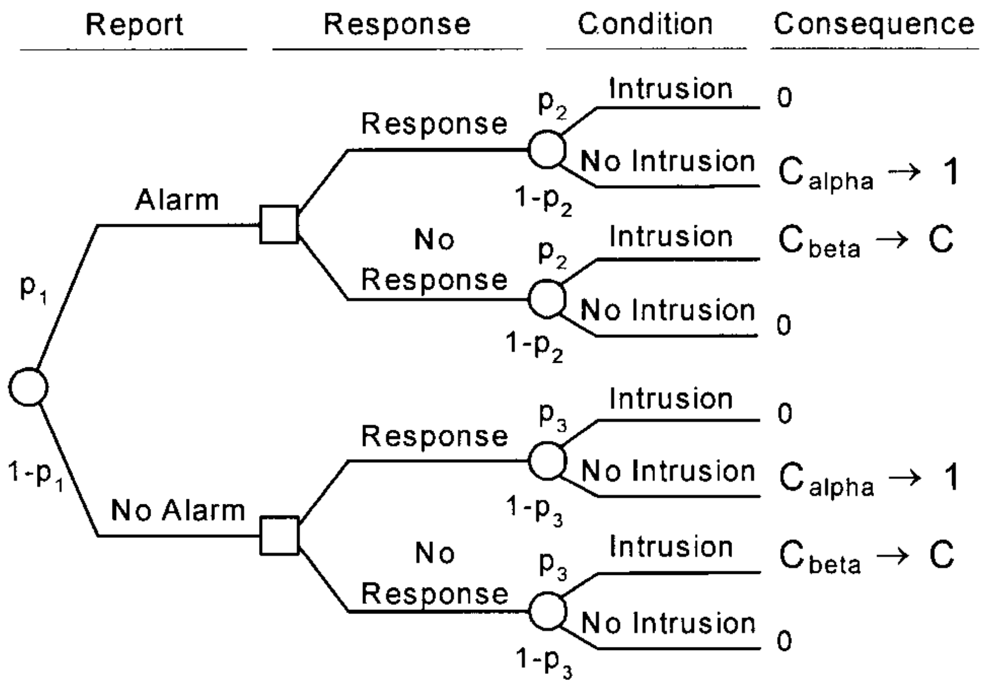


Figure 4.3: Decision Tree of IDS's expected cost, from [39].

Figure 4.3 shows the sequence of actions (squares) and uncertain events (circles) that describe the operation of the IDS and of the actions or responses that can be taken, based on reports. It also shows the consequences of the combinations of actions and events. The costs shown correspond to the consequences. The convention in a decision tree is to read it from left to right. The path leading to any point in the tree is shown to the left of the point and is assumed to be determined. Paths to the right of any point show all subsequent possibilities, which are not yet determined.

Chapter 5

Results and Analysis

The results, obtained from the analysis of the PCap files, are constituted by observations, consisting on arrays of features organized as a rectangular matrix. These observations are submitted to data pre-processing, dimensionality reduction and classifier training. Along with the feature extraction algorithms, and SVM related parameters presented in each table, a brief contextualization of the data obtained is presented. Finally, an insight of the obtained results are provided. The proportions of the number of NORMAL observations vs. the erroneous observations (resultant from the misbehaved node) is 2 : 1, meaning that for each two NORMAL observations, there is one erroneous observation.

5.1 Multi-Class Problem Approach

For the Multi-Class problem approach, the labels for the classes of the NORMAL and misbehaved nodes are organized as follows:

- **NORMAL** - Label 0;
- **DOS_FREQ** - Label 1;
- **DOS_ACK** - Label 2;
- **WRONG_ACCEPT** - Label 3;
- **WRONG_URI** - Label 4.

Two feature extraction algorithms are used, PCA and LDA. For both, we trained classifiers regarding the scoring metrics of accuracy and F_Measure.

For PCA, the data is pre-processed with a Standard Scaler, mean 0 and standard deviation 1. Figure 5.1 illustrates, in two dimensions, the feature reduction applied to the Multi-Class data. We may observe how clustered the resultant data is after the transformation.

The features of the data are reduced by PCA to 3 components, with whitening, and the method is set to *auto*. Then, a *grid search* for the best parameters of the SVM classifier, with a K-fold of 3, setting the scoring metric to accuracy is performed. The following SVM Kernels used are: Linear, RBF, polynomial and sigmoidal. The results are described in Figures 5.2 to 5.5.

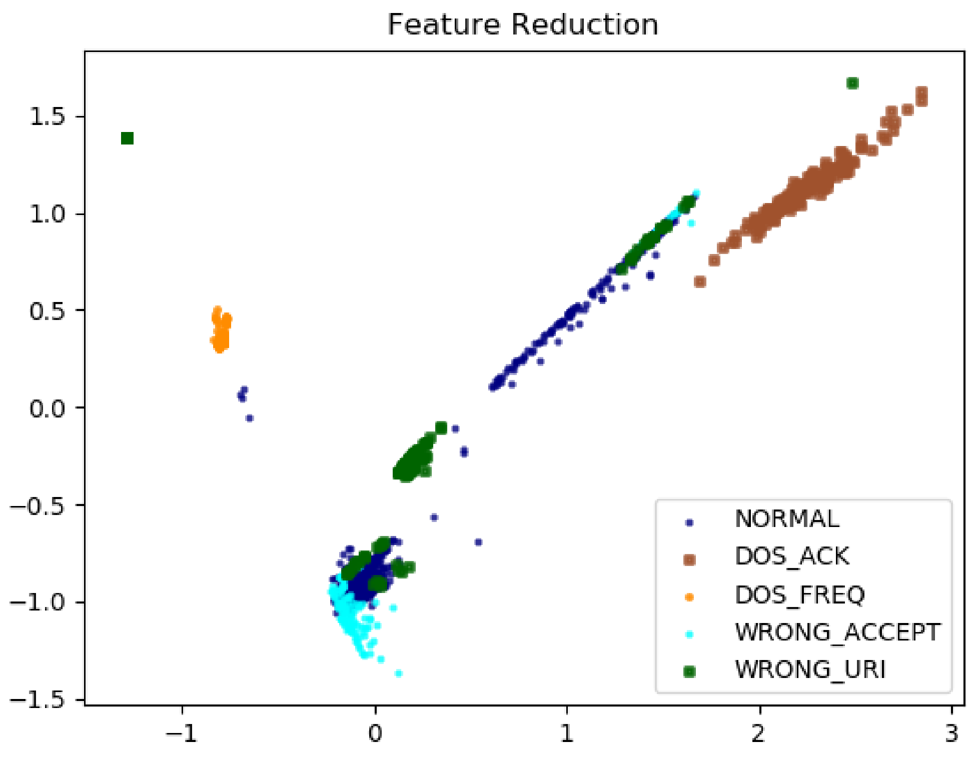
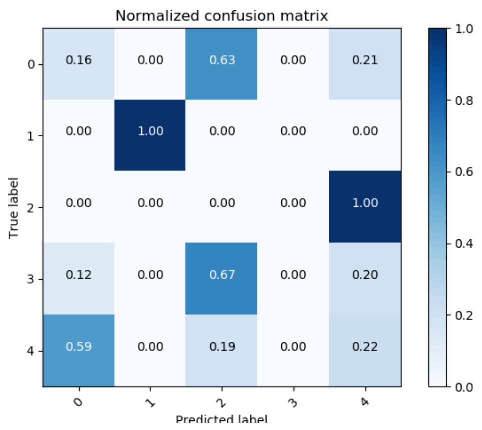
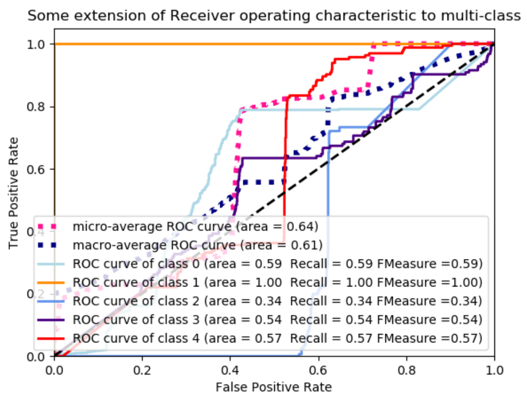


Figure 5.1: Multi-Class problem - PCA with auto solver, 3 Components and Data Whitening

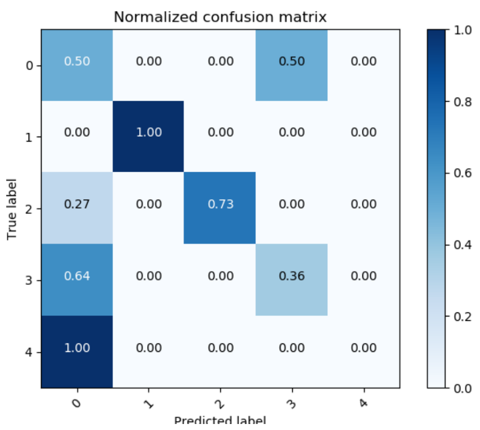


(a)

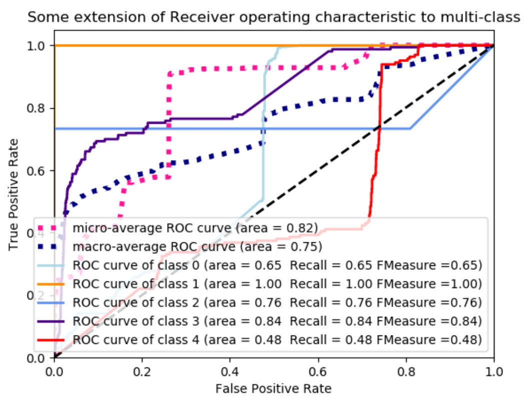


(b)

Figure 5.2: Multi-Class problem - Grid Search SVM with linear Kernel, One vs. Rest decision function shape, 30 iterations, and scoring criterion of accuracy, Confusion Matrix 5.2a and ROC Curves 5.2b

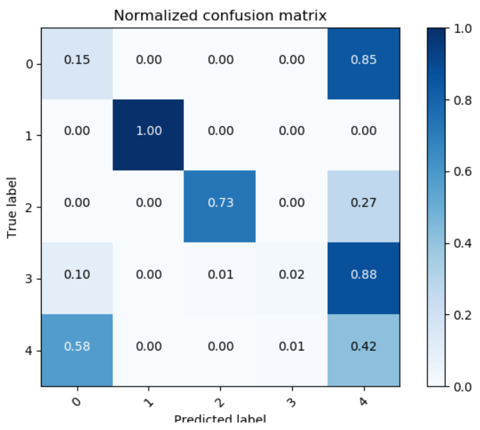


(a)

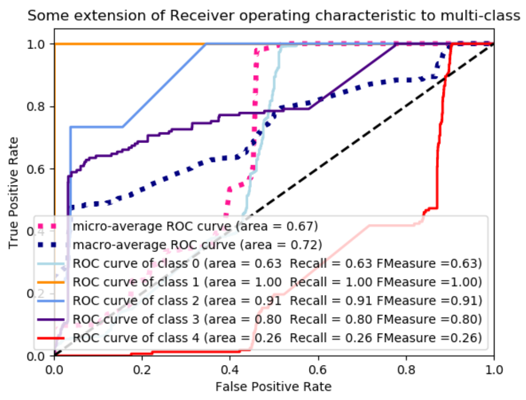


(b)

Figure 5.3: Multi-Class problem - Grid Search SVM with: RBF Kernel, One vs. Rest decision function shape, 30 iterations and scoring criterion of Weighted F_Measure, Confusion Matrix 5.3a and ROC Curves 5.3b

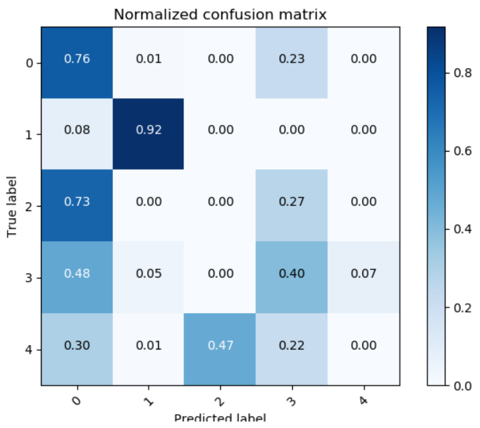


(a)

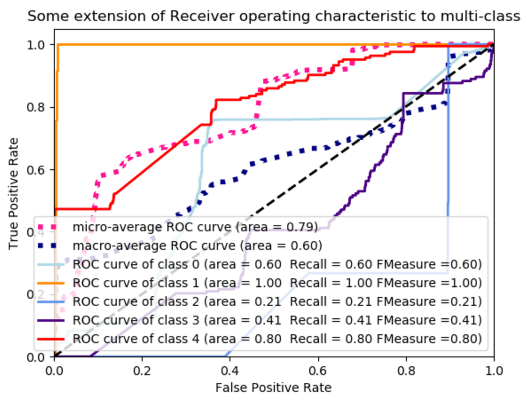


(b)

Figure 5.4: Multi-Class problem - Grid Search SVM with: polynomial Kernel, One vs. Rest decision function shape, 30 iterations and scoring criterion of Weighted F_Measure, confusion Matrix 5.4a and ROC Curves 5.4b



(a)



(b)

Figure 5.5: Multi-Class problem - Grid Search SVM with: sigmoidal Kernel, One vs. Rest decision function shape, 30 iterations, and a scoring criterion of accuracy, confusion Matrix 5.5a and ROC Curves 5.5b

For each Kernel, the best parameters and accuracy results are found by a grid search under the previously described conditions. Results are presented on Table 5.1.

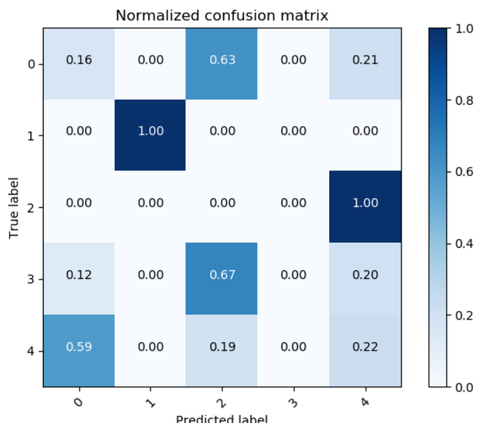
Table 5.1: SVM Parameters obtained using accuracy as the scoring metric.

Scoring Metric: accuracy

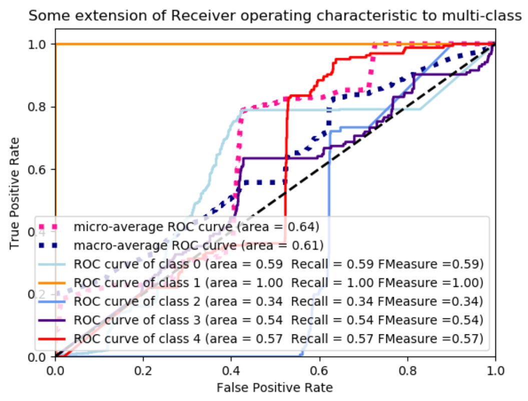
C	Kernel	Iterations	Degree (valid on Polynomial only)	Gamma	Coef0	Decision Function Shape	accuracy
0.957895	'linear'	30	1	0.100000	0.000000	'ovr'	0.212513
0.957895	'rbf'	30	1	1.000000	0.000000	'ovr'	0.509824
0.410526	'poly'	30	2	0.621053	0.105263	'ovr'	0.283868
0.200000	'sigmoid'	30	1	0.194736	0.947368	'ovr'	0.618408

The results from Figure 5.2 to 5.5 and from Table 5.1 show, that an SVM learner with Linear Kernel has 100% misclassification of the labels of DOS_ACK observations, and a misclassification of 59% of WRONG_URI observations. RBF Kernel has a completely wrong classification of WRONG_URI observations, considering them as NORMAL. For wrongly observations classified as NORMAL, polynomial Kernel has a good score, although 58% of the WRONG_URI observations are misclassified as Normal. sigmoidal Kernel achieved the best accuracy (Table 5.1), but classified many misbehaviors as NORMAL (Figure 5.5a).

Then, a *grid search* for the best parameters of the SVM classifier, setting the scoring metric to F_Measure, is performed. The SVM Kernels used are: Linear, RBF, polynomial and sigmoidal. The results are described in Figures 5.6 to 5.9.

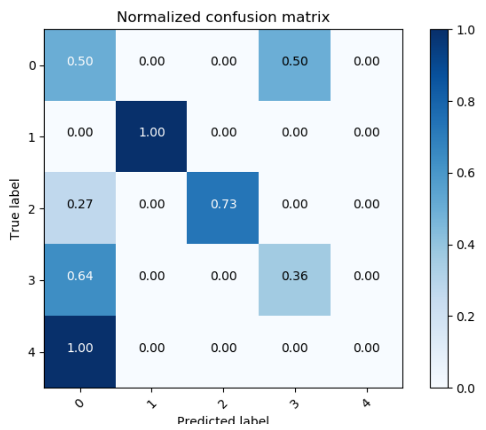


(a)

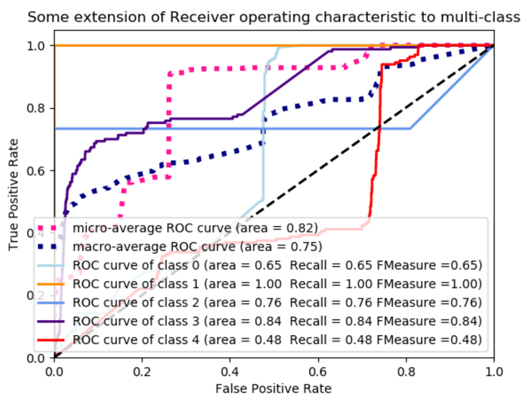


(b)

Figure 5.6: Multi-Class problem - Grid Search SVM with: linear Kernel, One vs. Rest decision function shape, 30 iterations, and scoring criterion of Weighted F_Measure, confusion Matrix 5.6a and ROC Curves 5.6b

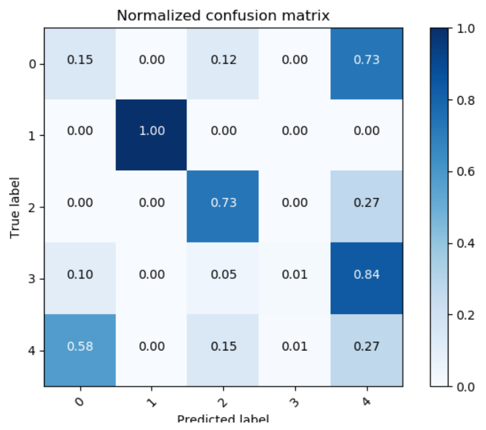


(a)

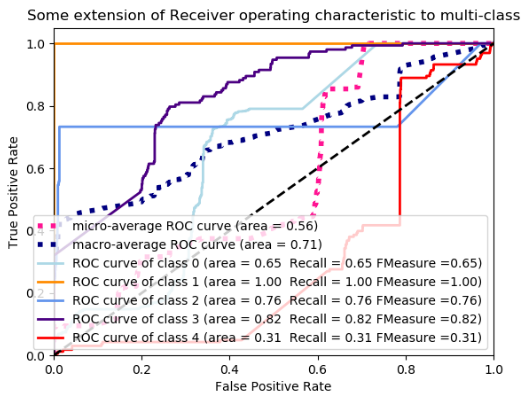


(b)

Figure 5.7: Multi-Class problem - Grid Search SVM with: rbf Kernel, One vs. Rest decision function shape, 30 iterations and scoring criterion of Weighted F_Measure, Confusion Matrix 5.7a and ROC Curves 5.7b

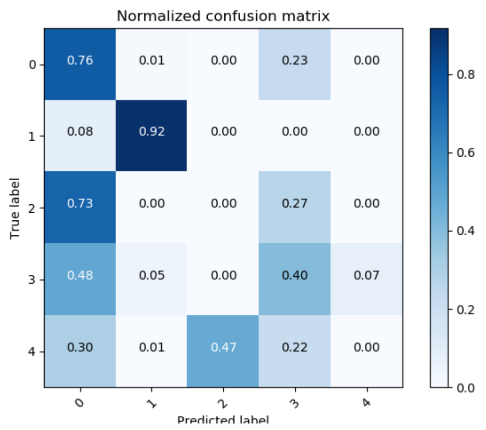


(a)

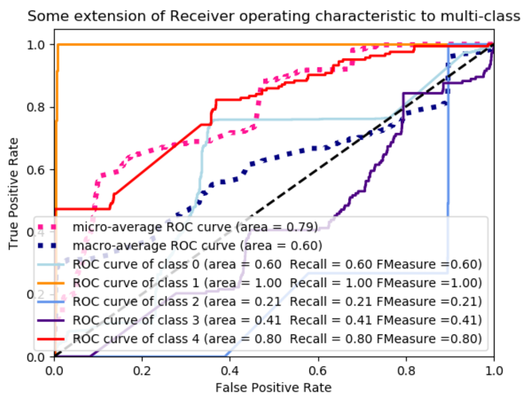


(b)

Figure 5.8: Multi-Class problem - Grid Search SVM with: polynomial Kernel, One vs. Rest decision function shape, 30 iterations and scoring criterion of Weighted F_Measure, confusion Matrix 5.8a and ROC Curves 5.8b



(a)



(b)

Figure 5.9: Multi-Class problem - Grid Search SVM with: sigmoidal Kernel, One vs. Rest decision function shape, 30 iterations, and a scoring criterion of Weighted F_Measure, confusion Matrix 5.9a and ROC Curves 5.9b.

For each Kernel, the best parameters and accuracy results are found by a grid search under the previously described conditions. Results are presented on Table 5.2.

Table 5.2: SVM Parameters obtained using F_Measure as the scoring metric.

Scoring Metric: F_Measure

C	Kernel	Iterations	Degree (valid on polynomial only	Gamma	Coef0	Decision Function Shape	F_Measure
0.957895	'linear'	30	1	0.100000	0.000000	'ovr'	0.212513
0.957895	'rbf'	30	1	1.000000	0.000000	ovr'	0.509824
0.326316	'poly'	30	2	0.715790	0.315790	'ovr'	0.270424
0.200000	'sigmoid'	30	1	0.194737	0.947368	'ovr'	0.618407

Although the scoring Metric is changed from accuracy to F_Measure, the results did not seem to change significantly. The sigmoidal and RBF Kernel still have the best accuracy of the four Kernels, though classifying wrongly the majority of the erroneous behaviors as NORMAL. polynomial, linear and RBF Kernels have the best score on accuracy regarding DOS_FRE, though sigmoidal Kernel does not fall very behind.

LDA is used with the method of singular value decomposition and 6 components. Again, the data is pre-processed with a *Standard Scaler*, mean 0 and standard deviation 1. Figure 5.10 illustrates the feature reduction applied to the Multi-Class data in two dimensions.

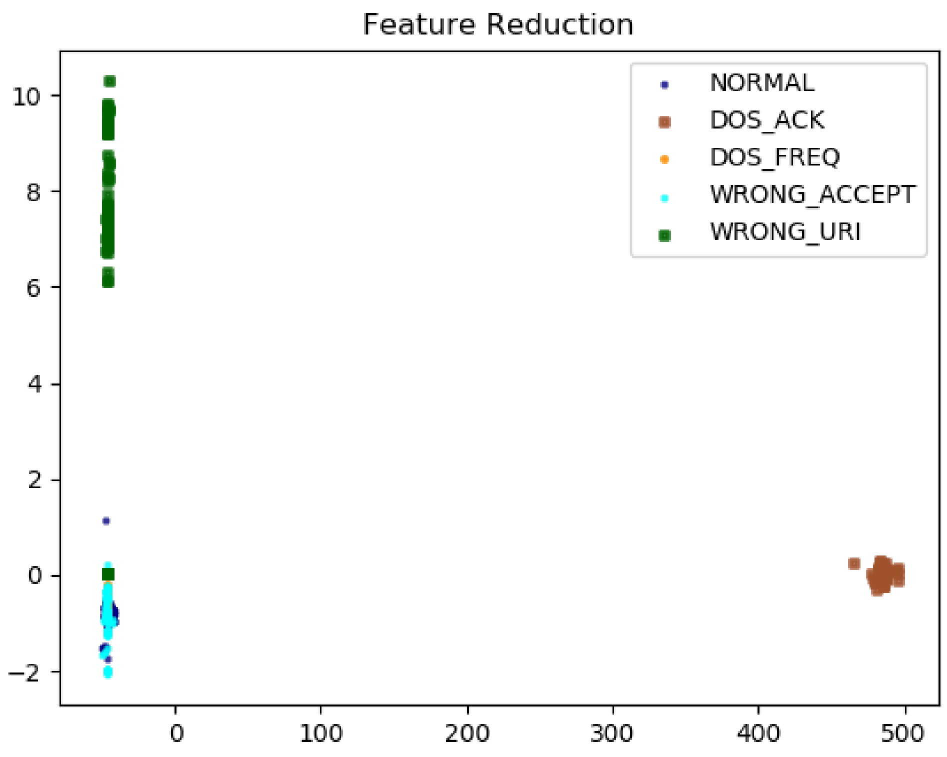
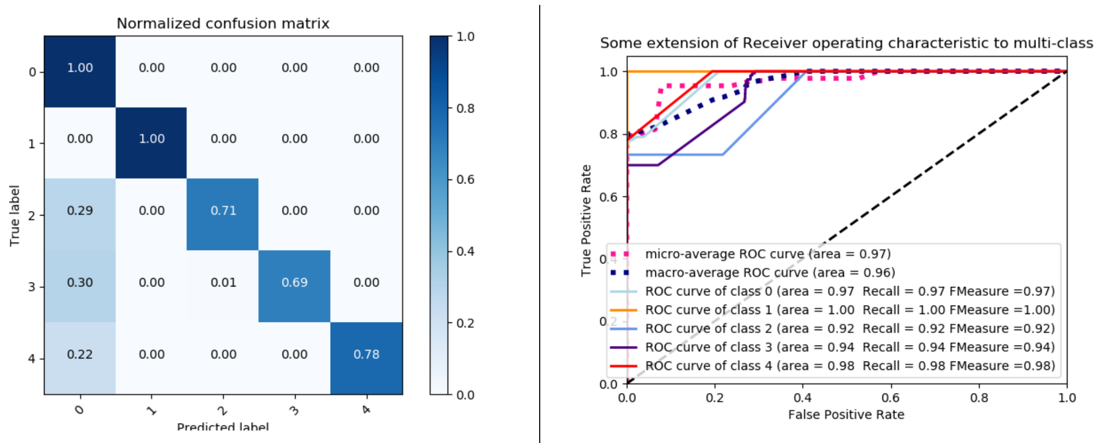


Figure 5.10: Multi-Class problem - LDA with Singular Value Decomposition (SVD) solver, 6 Components.

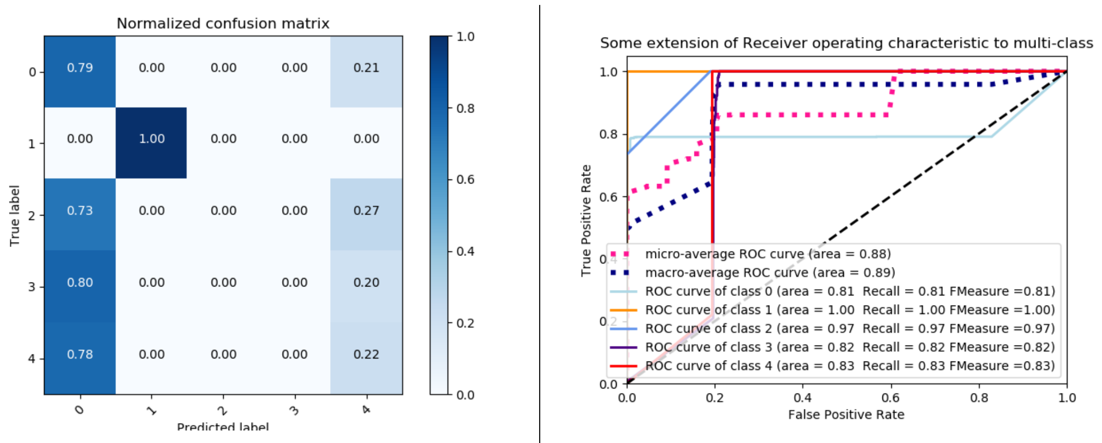
The features of the data are reduced by LDA to 6 components, and the method is set to *svd*. Then, we performed a *grid search* for the best parameters of the SVM classifier, with a K-fold of 3, setting the scoring metric to accuracy.

The following SVM Kernels used are: Linear, RBF, polynomial and sigmoidal. The results are described in Figures 5.11 to 5.14.



(a) (b)

Figure 5.11: Multi-Class problem - Grid Search SVM with linear Kernel, One vs. Rest decision function shape, 30 iterations, and scoring criterion of accuracy, Confusion Matrix 5.11a and ROC Curves 5.11b



(a) (b)

Figure 5.12: Multi-Class problem - Grid Search SVM with RBF Kernel, One vs. Rest decision function shape, 30 iterations, and scoring criterion of accuracy, Confusion Matrix 5.12a and ROC Curves 5.12b

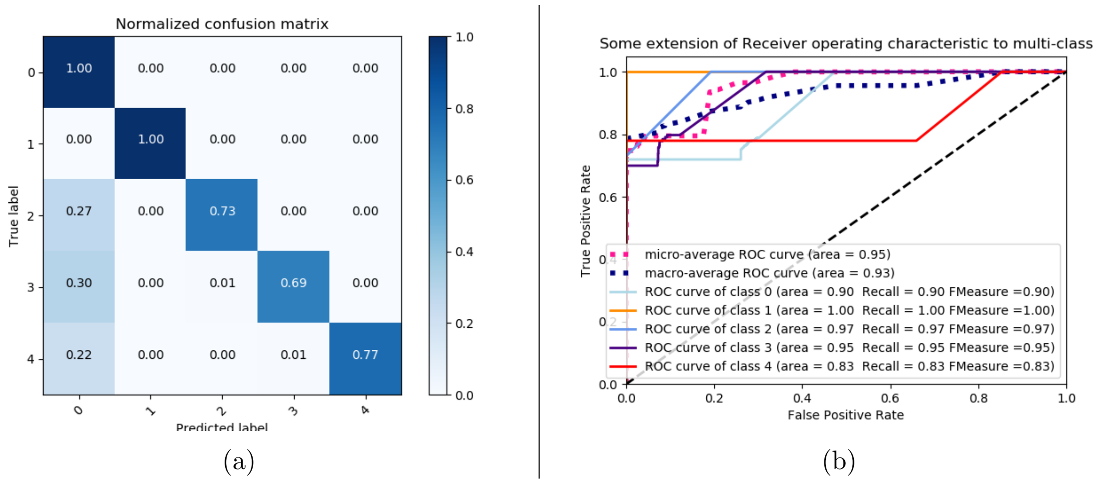


Figure 5.13: Multi-Class problem - Grid Search SVM with polynomial Kernel, One vs. Rest decision function shape, 30 iterations, and scoring criterion of accuracy, Confusion Matrix 5.13a and ROC Curves 5.13b,

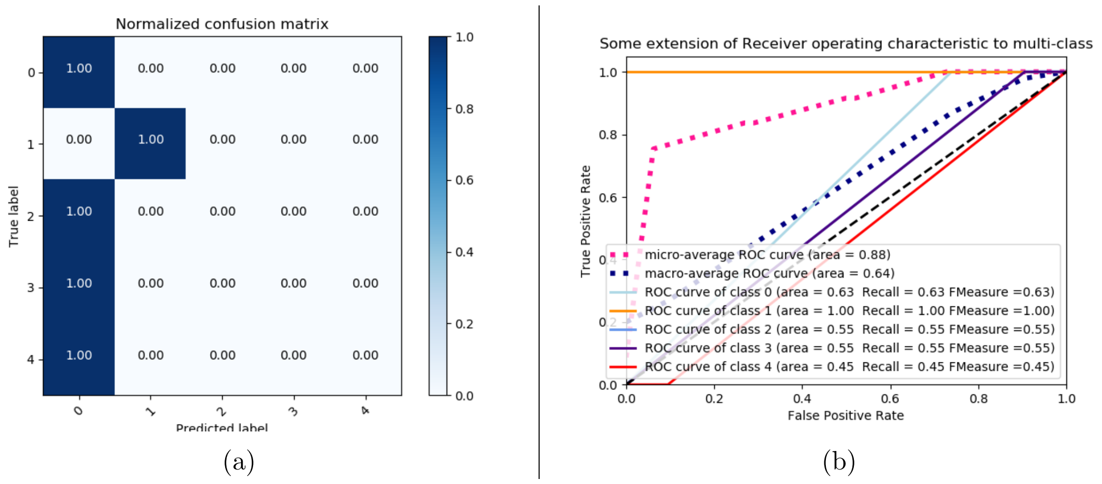


Figure 5.14: Multi-Class problem - Grid Search SVM with sigmoidal Kernel, One vs. Rest decision function shape, 30 iterations, and a scoring criterion of accuracy, confusion Matrix 5.14a and ROC Curves 5.14b

For each Kernel, the best parameters and accuracy results are found by a grid search under the previously described conditions. Results are presented on Table 5.3.

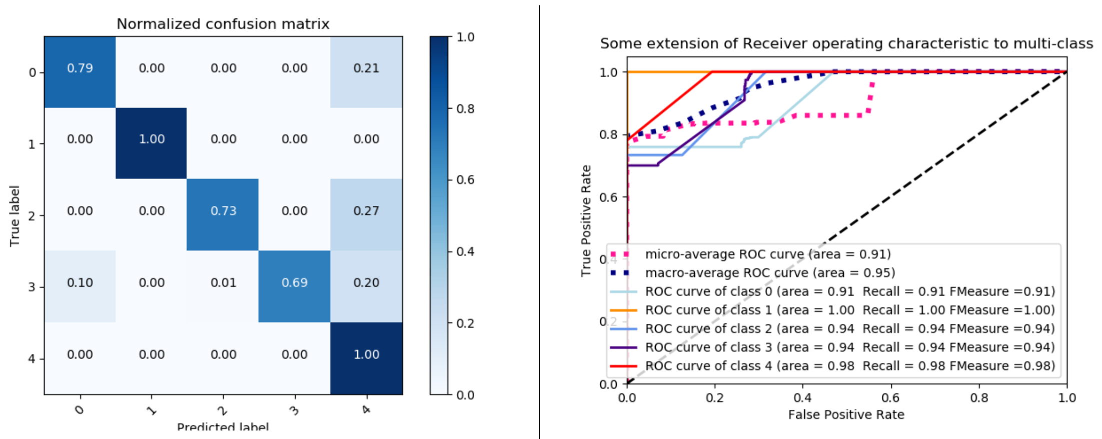
Table 5.3: SVM Parameters obtained using accuracy as the scoring metric.

Scoring Metric: accuracy

C	Kernel	Iterations	Degree (valid on Polynomial only)	Gamma	Coef0	Decision Function Shape	accuracy
0.452632	'linear'	30	1	0.100000	0.000000	'ovr'	0.932782
0.578947	'rbf'	30	1	0.810526	0.000000	'ovr'	0.632368
0.200000	'poly'	30	1	0.147368	0.000000	'ovr'	0.933816
0.284211	'sigmoid'	30	1	0.100000	0.000000	'ovr'	0.753361

With LDA the accuracy improved substantially. The polynomial and Linear Kernels reached 93% as can be seen on Table 5.3. RBF and sigmoidal Kernels showed a bad performance with an accuracy below 80% according to Table 5.3.

The features of the data are reduced by LDA to 6 components, and the method is set to *svd*. Then, we performed a *grid search* for the best parameters of the SVM classifier, with a K-fold of 3, setting the scoring metric to F_Measure. The following SVM Kernels used are: Linear, RBF, polynomial and sigmoidal. The results are described in Figures 5.15 to 5.18.



(a) (b)

Figure 5.15: Multi-Class problem - Grid Search SVM with linear Kernel, One vs. Rest decision function shape, 30 iterations, and a scoring criterion of Weighted F_Measure, confusion Matrix 5.15a and ROC Curves 5.15b

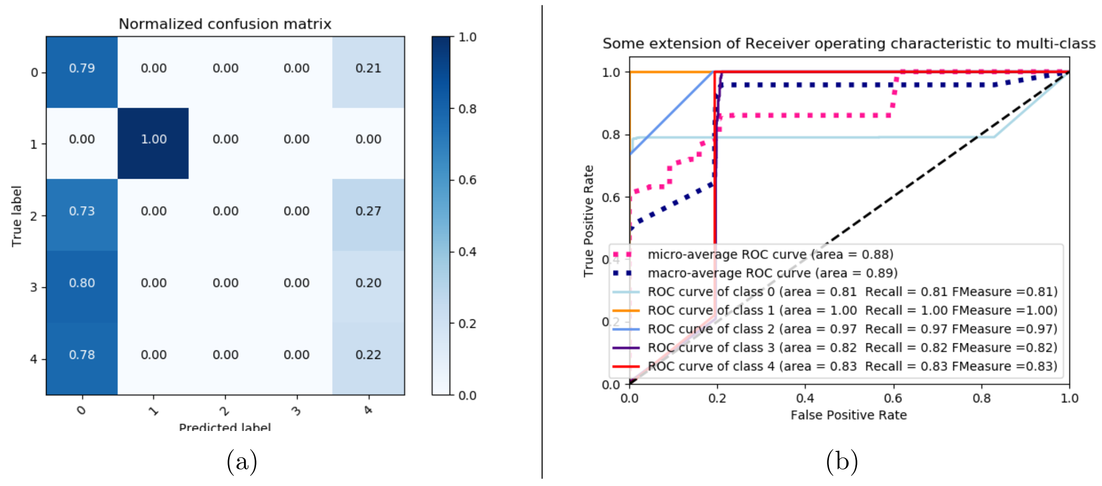


Figure 5.16: Multi-Class problem - Grid Search SVM with Radial Basis Function (rbf) Kernel, One vs. Rest decision function shape, 30 iterations, and a scoring criterion of Weighted F_Measure, confusion Matrix 5.16a and ROC Curves 5.16b

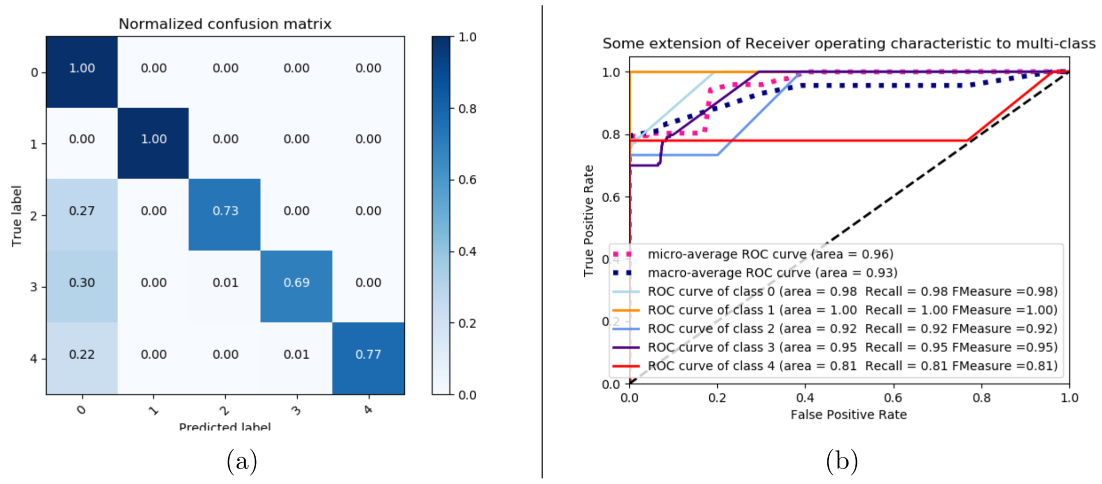


Figure 5.17: Multi-Class problem - Grid Search SVM with polynomial Kernel, One vs. Rest decision function shape, 30 iterations, a scoring criterion of Weighted F_Measure, confusion matrix 5.17a and ROC Curves 5.17b

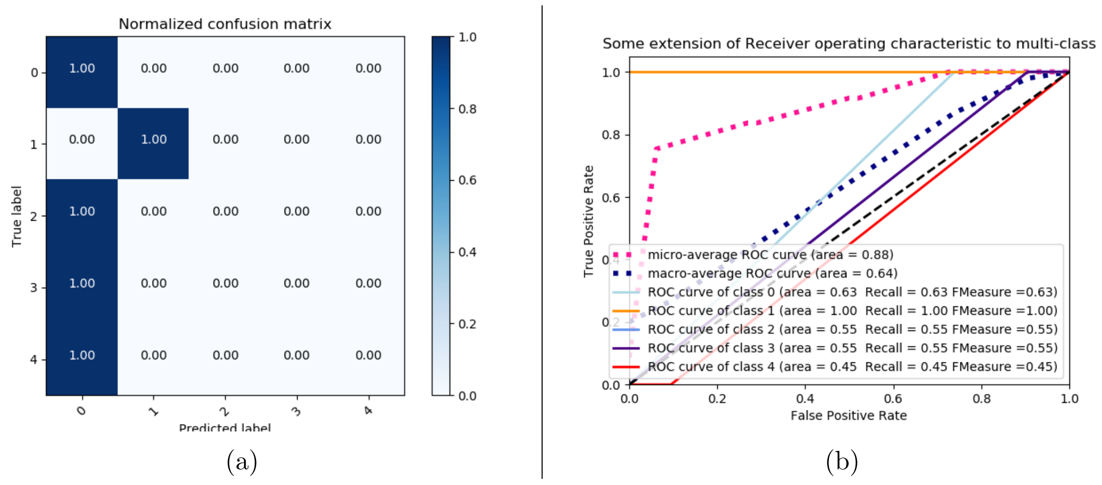


Figure 5.18: Multi-Class problem - Grid Search SVM with sigmoidal Kernel, One vs. Rest decision function shape, 30 iterations, and a scoring criterion of Weighted F_Measure, confusion matrix 5.18a and ROC Curves 5.18b

For each Kernel, the best parameters and F_Measure results are found by a grid search under the previously described conditions. Results are presented on Table 5.4.

Table 5.4: SVM Parameters obtained using F_Measure as the scoring metric.

Scoring Metric: F_Measure

C	Kernel	Iterations	Degree (valid on polynomial only	Gamma	Coef0	Decision Function Shape	F_Measure
0.915793	'linear'	30	1	0.100000	0.000000	'ovr'	0.813340
0.578947	'rbf'	30	1	0.810526	0.000000	'ovr'	0.632368
0.200000	'poly'	30	1	0.147368	0.000000	'ovr'	0.933816
0.284210	'sigmoid'	30	1	0.100000	0.000000	'ovr'	0.753361

Changing the scoring metric from accuracy to F_Measure seemed to not improve the F_-Measure score. Also, this change greatly sacrificed the accuracy of the classifier with linear and polynomial Kernels, reducing it to 82%. sigmoidal and RBF Kernels did not suffer significant alterations.

We conclude that LDA is a better feature extraction algorithm than PCA. An SVM with a polynomial Kernel also proved to be more accurate.

5.2 Binary Class Problem Approach

For Binary Class approach, the following two labels are defined:

- **NORMAL** - Label 0;
- **ERRONEOUS** - Label 1;

For this approach, only the most accurate feature extraction method, LDA is used. The method of singular value decomposition and 2 components is applied. Once more, data is pre-processed with a *Standard Scaler*, with mean 0 and standard deviation 1. Figure 5.19 illustrates the feature reduction applied to the binary class data in two dimensions.

The features of the data are reduced by LDA to 2 components, and the method is set to *svd*. Then, a *grid search* for the best parameters of the SVM classifier, with a K-fold of 3, setting the scoring metric to accuracy is performed.

The following SVM Kernels used are: Linear, RBF, polynomial and sigmoidal. The results are described in Figures 5.20 to 5.23.

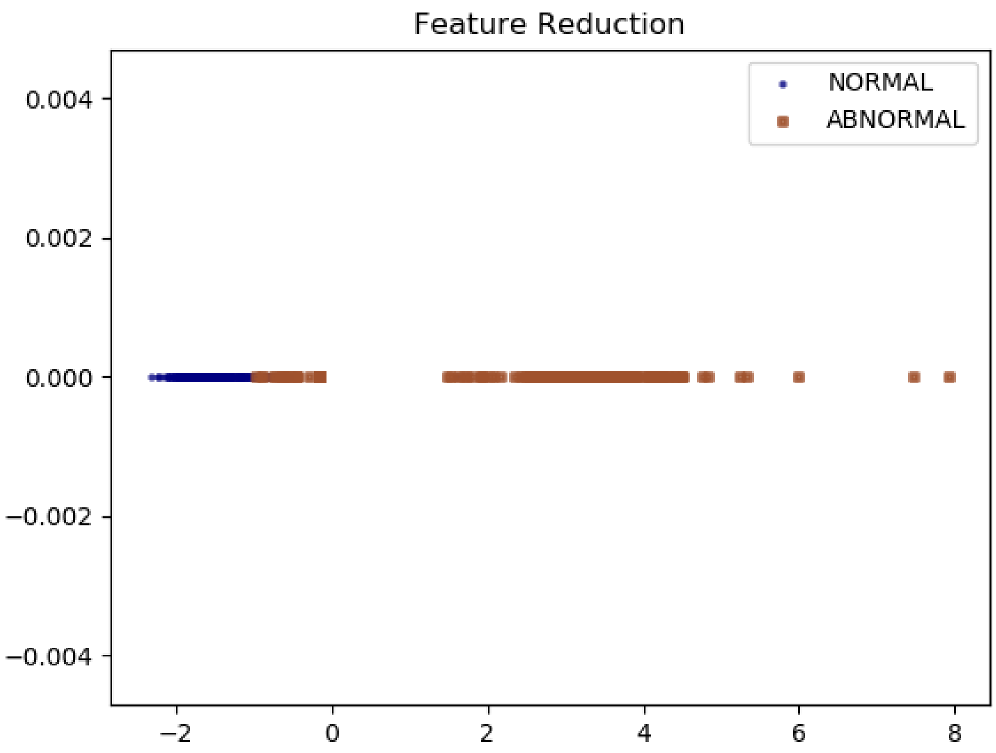
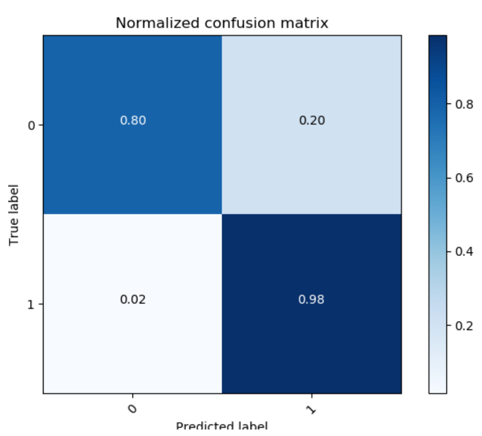
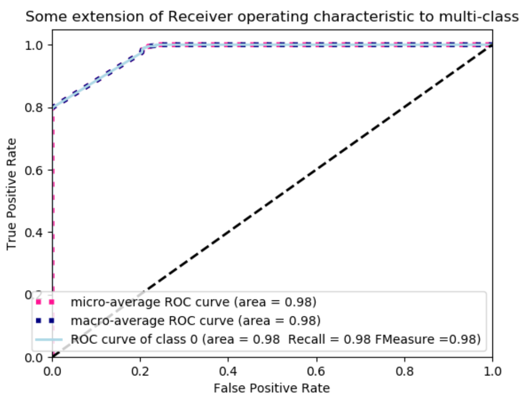


Figure 5.19: Binary Class problem - LDA with SVD solver, 2 Components

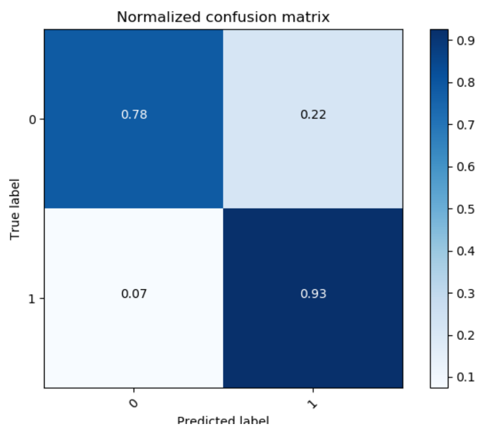


(a)

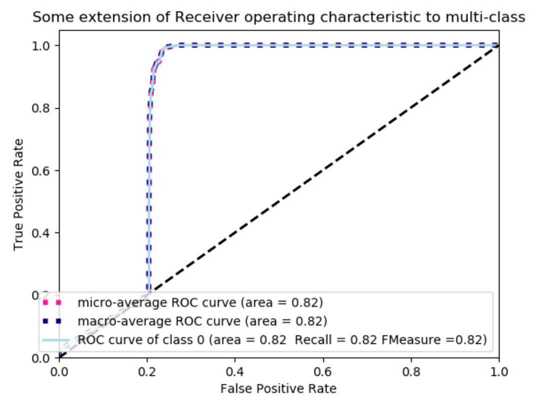


(b)

Figure 5.20: Binary Class problem - Grid Search SVM with linear Kernel, One vs. Rest decision function shape, 30 iterations, with scoring criterion of accuracy, confusion Matrix 5.20a and ROC Curves 5.20b

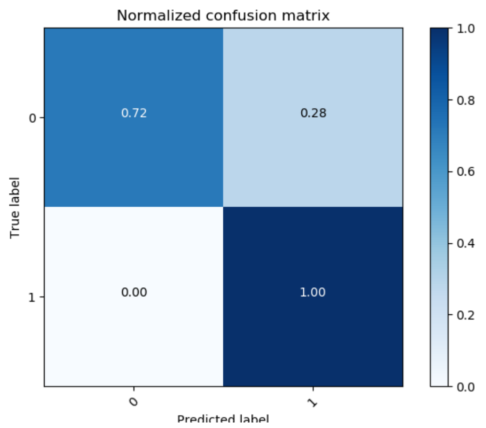


(a)

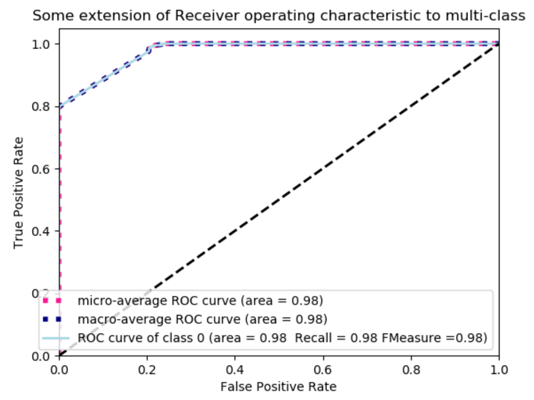


(b)

Figure 5.21: Binary Class problem - Grid Search SVM with RBF Kernel, One vs. Rest decision function shape, 30 iterations, with scoring criterion of accuracy confusion Matrix 5.21a and ROC Curves 5.21b

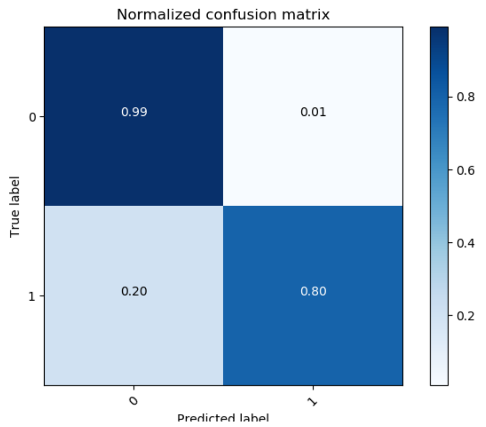


(a)

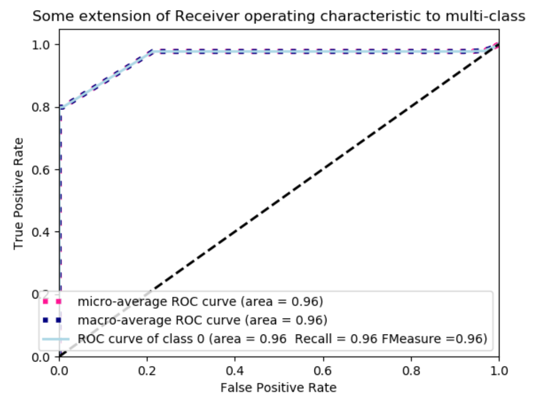


(b)

Figure 5.22: Binary Class problem - Grid Search SVM with polynomial Kernel, One vs. Rest decision function shape, 30 iterations, with scoring criterion of accuracy, confusion Matrix 5.22a and ROC Curves 5.22b



(a)



(b)

Figure 5.23: Binary Class problem - Grid Search SVM with sigmoidal Kernel, One vs. Rest decision function shape, 30 iterations, and a scoring criterion of accuracy confusion matrix 5.23a and ROC Curves 5.23b

For each Kernel, the best parameters and F_Measure results are found by a grid search under the previously described conditions. Results are presented on Table 5.5.

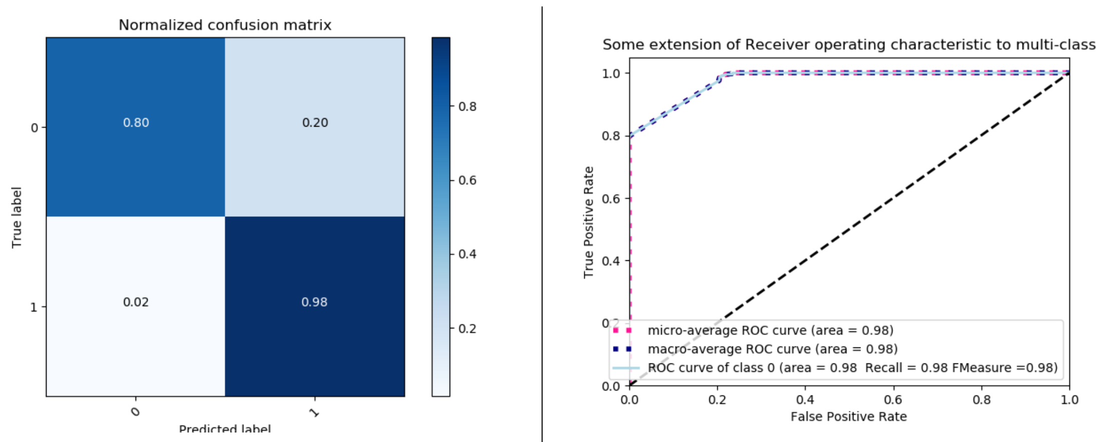
Table 5.5: SVM Parameters obtained using accuracy as the scoring metric.

Scoring Metric: accuracy

C	Kernel	Iterations	Degree (valid on Polynomial only)	Gamma	Coef0	Decision Function Shape	accuracy
0.200000	'linear'	30	1	0.100000	0.000000	'ovr'	0.858325
0.200000	'rbf'	30	1	0.100000	0.000000	'ovr'	0.830403
0.915790	'poly'	30	3	0.621053	0.000000	'ovr'	0.810238
0.200000	'sigmoid'	30	1	0.9052632	0.315710	'ovr'	0.927094

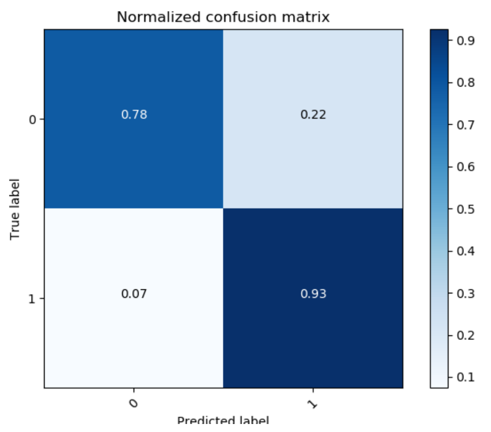
For the linear Kernel, the accuracy of these results do not show significant changes from those obtained with accuracy as the scoring metric, in Multi-Class LDA approach (Table 5.3). However, according to Figure 5.20b, good results of recall and accuracy are obtained for linear Kernel and for a polynomial Kernel of degree 3. Even more, on polynomial Kernel, the false negatives (observations wrongly classified as NORMAL), is 0%.

The features of the data are reduced by LDA to 2 components, and the method is set to *svd*. Then, a *grid search* for the best parameters of the SVM classifier, with a K-fold of 3, setting the scoring metric to F_Measure, is performed. The SVM Kernels used are: Linear, RBF, polynomial and sigmoidal. The results are described in Figures 5.24 to 5.27.

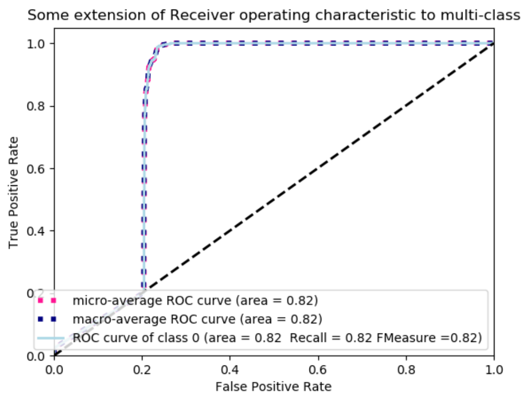


(a) (b)

Figure 5.24: Binary Class problem - Grid Search SVM with linear Kernel, One vs. Rest decision function shape, 30 iterations, and a scoring criterion of accuracy confusion Matrix 5.24a and ROC Curves 5.24b

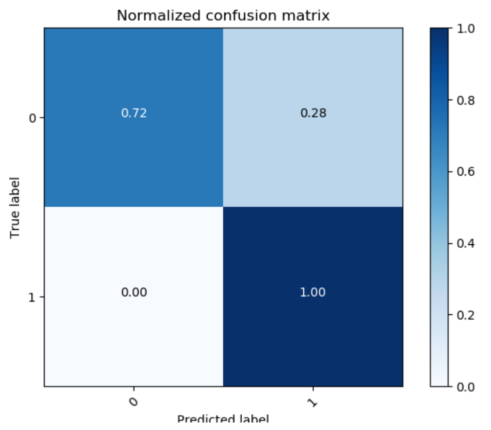


(a)

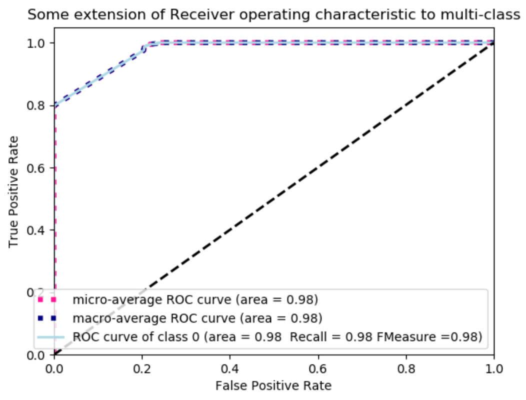


(b)

Figure 5.25: Binary Class problem - Grid Search SVM with RBF Kernel, One vs. Rest decision function shape, 30 iterations, and a scoring criterion of accuracy, confusion matrix 5.25a and ROC Curves 5.25b

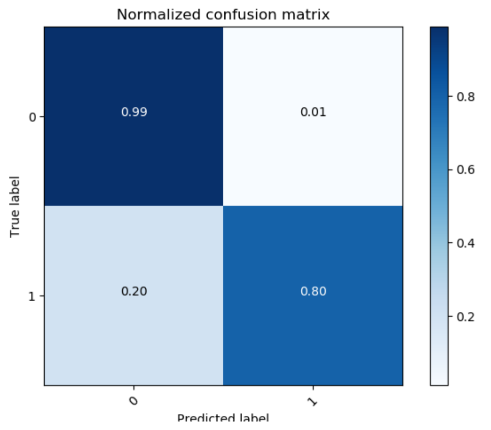


(a)

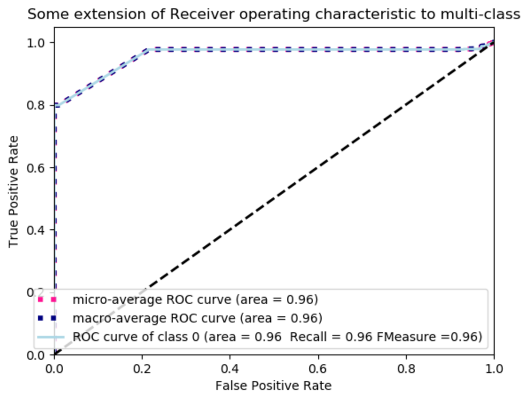


(b)

Figure 5.26: Binary Class problem - Grid Search SVM with polynomial Kernel, One vs. Rest decision function shape, 30 iterations, and a scoring criterion of accuracy, confusion Matrix 5.26a and ROC Curves 5.26b



(a)



(b)

Figure 5.27: Binary Class problem - Grid Search SVM with sigmoidal Kernel, One vs. Rest decision function shape, 30 iterations, and a scoring criterion of accuracy, confusion Matrix 5.27a and ROC Curves 5.27b

For each Kernel, the best parameters and F_Measure results are found by a grid search under the previously described conditions. Results are presented on Table 5.6.

Table 5.6: SVM Parameters obtained using F_Measure as the scoring metric.

Scoring Metric: F_Measure

C	Kernel	Iterations	Degree (valid on polynomial only	Gamma	Coef0	Decision Function Shape	F_Measure
0.200000	'linear'	30	1	0.100000	0.000000	'ovr'	0.858325
0.200000	'rbf'	30	1	0.100000	0.000000	'ovr'	0.830403
0.915790	'poly'	30	3	0.621053	0.000000	'ovr'	0.810238
0.452632	'sigmoid'	30	1	0.810526	0.4210526	'ovr'	0.925543

With the metric score set to F_Measure, the sigmoidal Kernel reaches the best accuracy score, of about 92% (Table 5.6). Sigmoidal Kernel accurately classifies NORMAL observations, with a ratio of false positive of 1% (Figure 5.27a), though with considerable false negatives of 20%. For polynomial Kernel, there are 0% false negatives, but 28% of false positives (Figure 5.26a). Considering IDSs aims, it is crucial to prevent the most intrusions as possible. So, the main concern is to assure a low false negatives rate even at a cost of increasing false positives. This must be taken into account when searching for the maximum accuracy.

Regarding the Multi-Class approach, using confusion matrices described in Figures 5.12a and 5.14a, we verify that RBF and sigmoidal Kernels have more false negatives (erroneous observations wrongly classified as NORMAL) than SVMs (with linear and polynomial Kernels).

If the Security Manager (SM) is interested in identifying specific intrusions, he should choose the Multi-class problem approach, since results show that a linear Kernel or a polynomial Kernel with degree one have the best results regarding accuracy. If the SM is more interested in capturing the largest number of ERRONEOUS behaviors, he should use the Binary class approach, using the parameters for the polynomial Kernel described in Table 5.5.

Chapter 6

Conclusion and Future Work

The area of security research on IoT is of paramount importance on today's technological paradigm, where all kind of devices can be connected and open to malicious intrusions. Currently, there are no ready-to-use open source Anomaly-based IDS solutions for IoT security. We propose an Anomaly-based Intrusion Detection System (IDS), AnIDS, for the IoT, operating at the CoAP level.

We faced some difficulties along this project. The major one was related to the key problem of finding a mechanism to generate intrusions on IEEE802.15.4-CoAP, or, in alternative, to find a dataset of intrusions for this stack. As none of these solutions were available, we had to generate the misbehaviors on the IoT-LAB platform. The next big challenge was how to calculate features from the PCap files. Most of the available programs of Internet PCap statistics calculators do not support IEEE802.15.4-CoAP, or not even mention what protocolar stack they support. Therefore, we had to experiment those programs one by one. Fortunately, Wireshark, with the TShark command line utility, proved to be able to analyze the majority of Internet protocols, including IEEE802.15.4, 6LoWPAN, RPL and CoAP. Thus, TShark could be used to calculate the features from PCap files.

AnIDS uses pre-processing algorithms and training classifiers for building the intrusion model, either on a Multi or Binary class approaches. For the Multi-Class problem approach, AnIDS is capable of distinguish between the four types of implemented misbehaviors, reaching an accuracy of 93% for the best SVM classifier, with accuracy as the scoring metric, when using the LDA feature extraction algorithm. For the Binary Class problem approach, with the best SVM parameters, a classification of NORMAL and ERRONEOUS can reach an accuracy of 92%, and an F_Measure of 98%.

Currently, features are calculated using a TShark call from a python script via *popen* command. This solution can be improved to reduce execution time. As future work, we plan to develop an application for calculating features from a PCap file, without 3$^{\text{rd}}$-party command system calls. Although only SVM was used as a classifier algorithm, AnIDS can be easily extended to include additional algorithms, like K-NN, Neural Networks or Random Forests. Additionally, this work also provides a framework for launching IoT experiments on the IoT-LAB cloud platform.

Using a CoAP scenario, AnIDS can handle misbehaviors that are difficult to detect by a pure Signature-based IDS. Additional intrusions can be considered in future developments.

We hope this work can contribute to make our digital world more secure.

References

[1] Contiki: The open source operating system for the internet of things. `http://www.contiki-os.org/index.html`, March 2003. (Accessed on 05/02/2017).

[2] Fit/iot-lab • very large scale open wireless sensor network testbed, 2014.

[3] Apache mynewt os - wikipedia. `https://en.wikipedia.org/wiki/Apache_Mynewt_OS`, March 2017. (Accessed on 05/01/2017).

[4] Boards — mbed. `https://developer.mbed.org/platforms/`, April 2017. (Accessed on 05/03/2017).

[5] Contiki - wikipedia. `https://en.wikipedia.org/wiki/Contiki`, April 2017. (Accessed on 05/02/2017).

[6] Home — mbed. `https://www.mbed.com/en/`, January 2017. (Accessed on 05/03/2017).

[7] mbed - wikipedia. `https://en.wikipedia.org/wiki/Mbed`, April 2017. (Accessed on 05/03/2017).

[8] Riot (operating system) - wikipedia. `https://en.wikipedia.org/wiki/RIOT_(operating_system)`, April 2017. (Accessed on 05/02/2017).

[9] Riot · github. `https://github.com/RIOT-OS/`, April 2017. (Accessed on 05/02/2017).

[10] TinyOS Alliance. Github - tinyos/tinyos-main: Main development repository for tinyos (an os for embedded, wireless devices). `https://github.com/tinyos/tinyos-main`, 2000. (Accessed on 05/02/2017).

[11] Syed Obaid Amin, Muhammad Shoaib Siddiqui, Choong Seon Hong, and Sungwon Lee. Rides: Robust intrusion detection system for ip-based ubiquitous sensor networks. *Sensors*, 9(5):3447–3468, 2009.

[12] IEEE Standards Association et al. Ieee std 802.15. 4-2011, ieee standard for local and metropolitan area networks—part 15.4: Low-rate wireless personal area networks (lr-wpans), 2011.

[13] Carsten Bormann, Klaus Hartke, and Zach Shelby. The Constrained Application Protocol (CoAP). RFC 7252, June 2014.

[14] Anders Brandt, JP Vasseur, Jonathan Hui, Kris Pister, Pascal Thubert, P Levis, Rene Struik, Richard Kelsey, Thomas H. Clausen, and Tim Winter. RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. RFC 6550, March 2012.

[15] Eung Jun Cho, Jin Ho Kim, and Choong Seon Hong. Attack model and detection scheme for botnet on 6lowpan. In *Asia-Pacific Network Operations and Management Symposium*, pages 515–518. Springer, 2009.

[16] Dr. Steve E. Deering and Robert M. Hinden. Internet Protocol, Version 6 (IPv6) Specification. RFC 8200, July 2017.

[17] Linux Foundation. Home — zephyr project. `https://www.zephyrproject.org/`, March 2017. (Accessed on 05/02/2017).

[18] Raspberry Pi Foundation. Frontpage - raspbian. `https://www.raspbian.org/`, June 2012. (Accessed on 05/16/2017).

[19] The Apache Software Foundation. Apache mynewt. `https://mynewt.apache.org/`, March 2017. (Accessed on 05/01/2017).

[20] The Apache Software Foundation. incubator-mynewt-dev mailing list archives. `http://mail-archives.apache.org/mod_mbox/incubator-mynewt-dev/201704.mbox/browser`, May 2017. (Accessed on 05/01/2017).

[21] Carles Gomez, Joaquim Oller, and Josep Paradells. Overview and evaluation of bluetooth low energy: An emerging low-power wireless technology. *Sensors*, 12(9):11734–11753, 2012.

[22] Jorge Granjal, Edmundo Monteiro, and Jorge Sá Silva. Security for the internet of things: a survey of existing protocols and open research issues. *IEEE Communications Surveys & Tutorials*, 17(3):1294–1312, 2015.

[23] Abhishek Gupta, Om Jee Pandey, Mahendra Shukla, Anjali Dadhich, Samar Mathur, and Anup Ingle. Computational intelligence based intrusion detection systems for wireless communication and pervasive computing networks. In *Computational Intelligence and Computing Research (ICCIC), 2013 IEEE International Conference on*, pages 1–7. IEEE, 2013.

[24] Nick Jones. ebbits - articles: The ebbits platform. `http://www.ebbits-project.eu/articles.php?article_id=2`, 2002. (Accessed on 04/26/2017).

[25] Prabhakaran Kasinathan, Claudio Pastrone, Maurizio A Spirito, and Mark Vinkovits. Denial-of-service detection in 6lowpan based internet of things. In *Wireless and Mobile Computing, Networking and Communications (WiMob), 2013 IEEE 9th International Conference on*, pages 600–607. IEEE, 2013.

[26] Anhtuan Le, Jonathan Loo, Aboubaker Lasebae, Mahdi Aiash, and Yuan Luo. 6lowpan: a study on qos security threats and countermeasures using intrusion detection system approach. *International Journal of Communication Systems*, 25(9):1189–1212, 2012.

[27] Hung-Jen Liao, Chun-Hung Richard Lin, Ying-Chih Lin, and Kuang-Yuan Tung. Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications*, 36(1):16–24, 2013.

[28] Gabriel Montenegro, Jonathan Hui, David Culler, and Nandakishore Kushalnagar. Transmission of IPv6 Packets over IEEE 802.15.4 Networks. RFC 4944, September 2007.

[29] Ken Munro. Wi-fi halow 802.11ah for iot. 802.11uh-oh? — pen test partners. `https://www.pentestpartners.com/blog/wi-fi-halow-802-11ah-for-iot-802-11uh-oh/`, 2016. (Accessed on 16/03/2017).

[30] Luís ML Oliveira, Joel JPC Rodrigues, Amaro F Sousa, and Jaime Lloret. Denial of service mitigation approach for ipv6-enabled smart object networks. *Concurrency and Computation: Practice and Experience*, 25(1):129–142, 2013.

[31] Pavan Pongle and Gurunath Chavan. Real time intrusion and wormhole attack detection in internet of things. *International Journal of Computer Applications*, 121(9), 2015.

[32] JS Raikwal and Kanak Saxena. Performance evaluation of svm and k-nearest neighbor algorithm over medical data set. *International Journal of Computer Applications*, 50(14), 2012.

[33] Shahid Raza, Adriaan Slabbert, Thiemo Voigt, and Krister Landernäs. Security considerations for the wirelesshart protocol. In *Emerging Technologies & Factory Automation, 2009. ETFA 2009. IEEE Conference on*, pages 1–8. IEEE, 2009.

[34] Shahid Raza, Linus Wallgren, and Thiemo Voigt. Svelte: Real-time intrusion detection in the internet of things. *Ad hoc networks*, 11(8):2661–2674, 2013.

[35] E. Ronen, A. Shamir, A. O. Weingarten, and C. O'Flynn. Iot goes nuclear: Creating a zigbee chain reaction. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 195–212, May 2017.

[36] Ahmed Saeed, Ali Ahmadinia, Abbas Javed, and Hadi Larijani. Intelligent intrusion detection in low-power iots. *ACM Transactions on Internet Technology (TOIT)*, 16(4):27, 2016.

[37] Daniel AJ Sokolov. Deepsec: Zigbee macht smart home zum offenen haus, 2015.

[38] Douglas H Summerville, Kenneth M Zach, and Yu Chen. Ultra-lightweight deep packet anomaly detection for internet of things devices. In *Computing and Communications Conference (IPCCC), 2015 IEEE 34th International Performance*, pages 1–8. IEEE, 2015.

[39] Jacob W Ulvila and John E Gaffney Jr. Evaluation of intrusion detection systems. *Journal of Research of the National Institute of Standards and Technology*, 108(6):453, 2003.

UNIVERSITY OF COIMBRA   Attack and Intrusion Detection on the IoT   João Miguel Coimbra Barros da Silva