Thesis

Implementing Intrusion Detection System (IDS) in Internet of Things (IoT) Networks

Stefanos Kyriakou

University of Cyprus

Department of Computer Science

June 2020

University of Cyprus

Department of Computer Science

Implementing Intrusion Detection System (IDS) in Internet of Things (IoT) Networks

Stefanos Kyriakou

Research Supervisor
Dr. Vasos Vassiliou

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Bachelor of Science at the University of Cyprus

June 2020

Acknowledgements

I would first like to thank my professor and research supervisor Dr. Vasos Vasiliou for not only teaching my various courses throughout the years but also as a mentor to this dissertation. I would also like to give my gratitude to the University of Cyprus and the Department of Computer Science for giving me an academic roof for 4 years and the opportunity to study what I have always loved.

Summary

The rapid adoption of inter-connected devices from the “Internet of Things” or else IoT, has brought to light the crucial part of detecting attacks against our networks and ecosystems. The aggressive production of devices to meet the demands of the consuming and marketing mass has resulted in reduced timeframes of manufacturing. The grade of security that is implemented during the lifecycle of development, as well as the establishment of production standards has become questionable. IoT devices, user interfaces and wireless protocols are targeted through several vulnerabilities that are showing in frequent occurrences. It is of vital importance that intrusion analysts not only understand the underlying IoT ecosystem, but are also in the position to evaluate threats and develop appropriate detections. This paper will cover applications of vulnerabilities of the IoT network, the malicious attacks and threats that take advantage of those, as well as several ways to apply intrusion detection, with a hands-on intrusion detection system deployment through Snort and a MikroTik router.

Table of Contents

Chapter 1

Introduction

1.1 Motivation

The devices known as the “Internet of Things” are categorized through an endless list of wireless sensors, smart meters, security cameras and appliances. The explosive growth of the IoT has pushed development to extreme lengths concerning the rapid rate of products coming into the market. As such, a gap has been created in the traditional security of endpoints and although not new, it is still a very hot topic due to the lack of standards and ability to keep security development on par with the demand for new technology. As the IoT continues to grow, more and more weaknesses will surface on multiple levels of the network stack. [17]

“The 2019 Global PKI and IoT Trends Study, conducted by research firm the Ponemon Institute and sponsored by nCipher Security, is based on feedback from more than 1,800 IT security practitioners in 14 countries and regions.” The study showcases the four main IoT threats that concern the security sector, as well as the lack of encryption usage. The estimated usage of digital certificates for identification and authentication by 2021 will be 42% of the devices. [2]

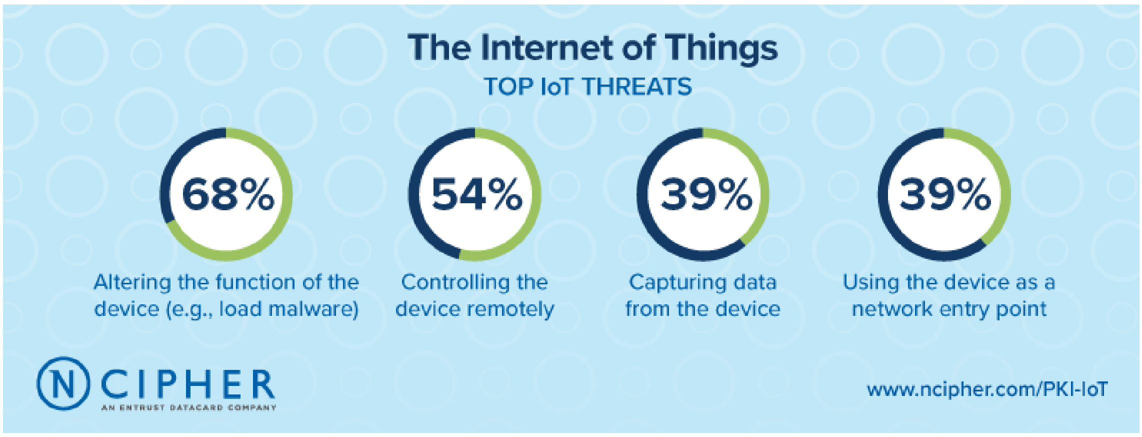


Figure 1.1 – Top IoT Threats by ncipher [2]

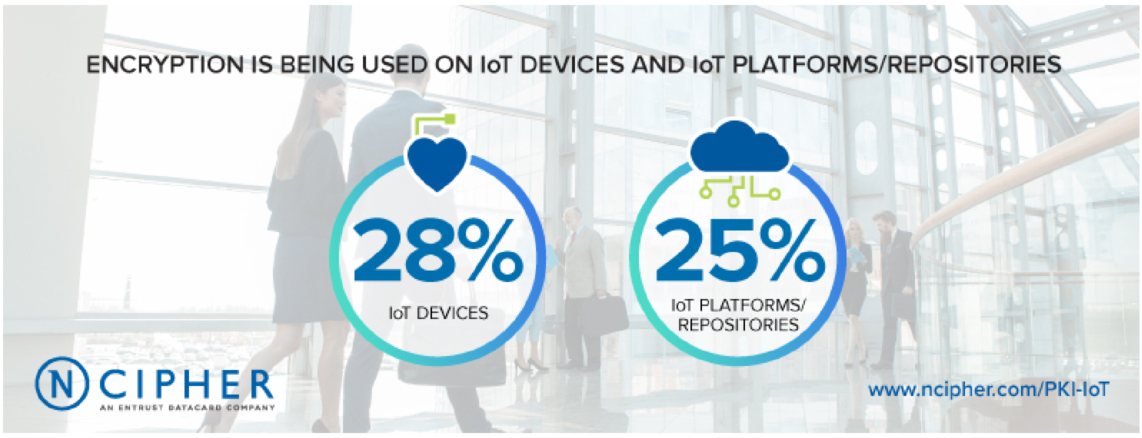


Figure 1.2 – Encryption used on IoT devices by ncipher [2]

1.2 Goal

IoT devices face threats and vulnerabilities in proportion to the attack surfaces of its extensive range. Physical interfaces are being attacked, wireless communication and routing protocols, as well as the traditional IP attacks on top. The growing market and related vulnerabilities are more than enough to show the severity of threats and the risks that come along with them. Not only do intrusion analysts need to be able to research and develop detection mechanisms for the IoT, but also make resources known to the consumers that are actually using the devices. This paper will discuss common threats and vulnerabilities of the over-growing IoT ecosystem, how detection can be applied and where, as well as hands-on deployment of an Intrusion Detection System, Snort, combined with a MikroTik router as the IoT gateway.

1.3 Methodology

First of all, extensive research and applied usage of the MikroTik router was done, through the purchase of a physical model and use in a home environment. Through several guides and documentation, successful understanding and setup of the “programmable” router was achieved from scratch.

Then, an Intrusion Detection System was needed to combine with the gateway at hand. Snort fit right into this spot, with relative support and documentation to integrate the one within the other in a router-to-machine relationship. The machine at hand could have been anything from a server to a Raspberry Pi. In this case, it was an Ubuntu 18.04 machine, on which Snort was installed and studied from the extensive guidelines that are freely available on the supported website snort.org.

Having the appropriate tools on hand and the needed knowledge on their use, there was a need for datasets of malicious traffic. Before it was possible to work malicious datasets, more research into IoT was needed. Through studying several academic papers that describe and explain in detail of how the IoT network stack works, the comprehension of the vulnerabilities and attack surfaces were next.

The first target was ContikiOS through its simulator Cooja. An open source repository of the routing protocol RPL exists, that has available three types of RPL attacks. Two more datasets were used from different institutions that created a controlled IoT environment with real devices and executed malicious code. The attacks were of various types, some of which are further explained in this paper.

1.4 Structure

In Chapter 1 the motivation and goal of the project have been given, as well as the complete path that was followed in order to be successful in carrying out the dissertation.

In Chapter 2 the underlying IoT network is explained as a first requirement in the equation of detecting and dealing with attacks. Following in the next part, common threats and vulnerabilities are shown as well as the discovery model of threats.

In Chapter 3 dissection of the communication and monitoring is done through common application protocols that are used, as well as link layer communication technologies combined with some intrusion detection systems that were developed.

Following in Chapter 4, intrusion detection systems are put in action against different cases of attacks on the IoT network.

Chapter 5 showcases a hands-on practical project of implementing the IDS Snort in combination with the MikroTik router as the IoT gateway in a private IoT network.

Lastly, in Chapter 6 the conclusions and future considerations are showcased based on the research and practical experience that came from this project.

Chapter 2

IoT Threats

A solid comprehension of the environment and associated threats are essential to the development and implementation of intrusion detection systems. This entails understanding the technology involved, relevant communications, the overall architecture and the impact of having the system compromised. An intrusion analyst can make use of these to understand the threats that may linger, whether already out there or those that have yet to be discovered. Another part of the equation are the intrusion detection sensors that can help with ingress attacks originated from anywhere in the world wide web, between an internet point and the IoT gateway. In wireless protocols communication, the sensors will need to be between the IoT nodes and the gateway for countermeasures.

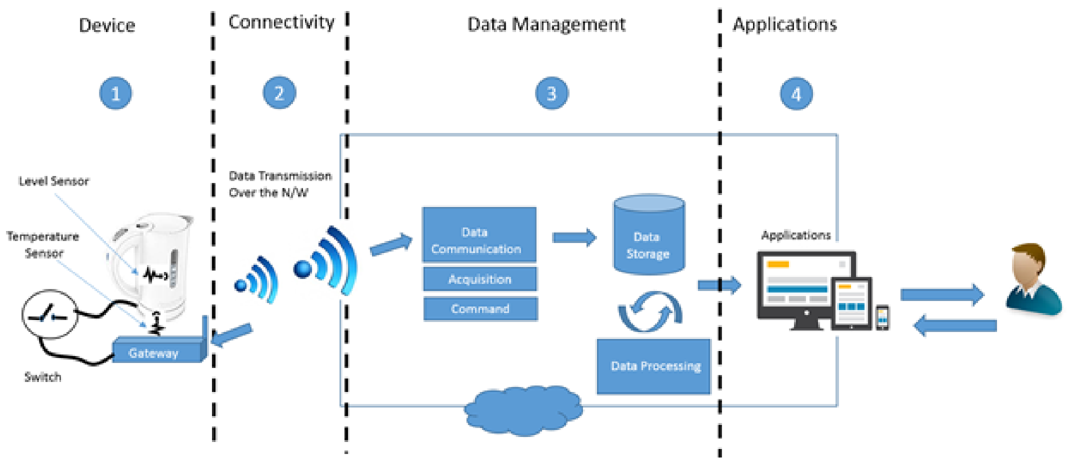
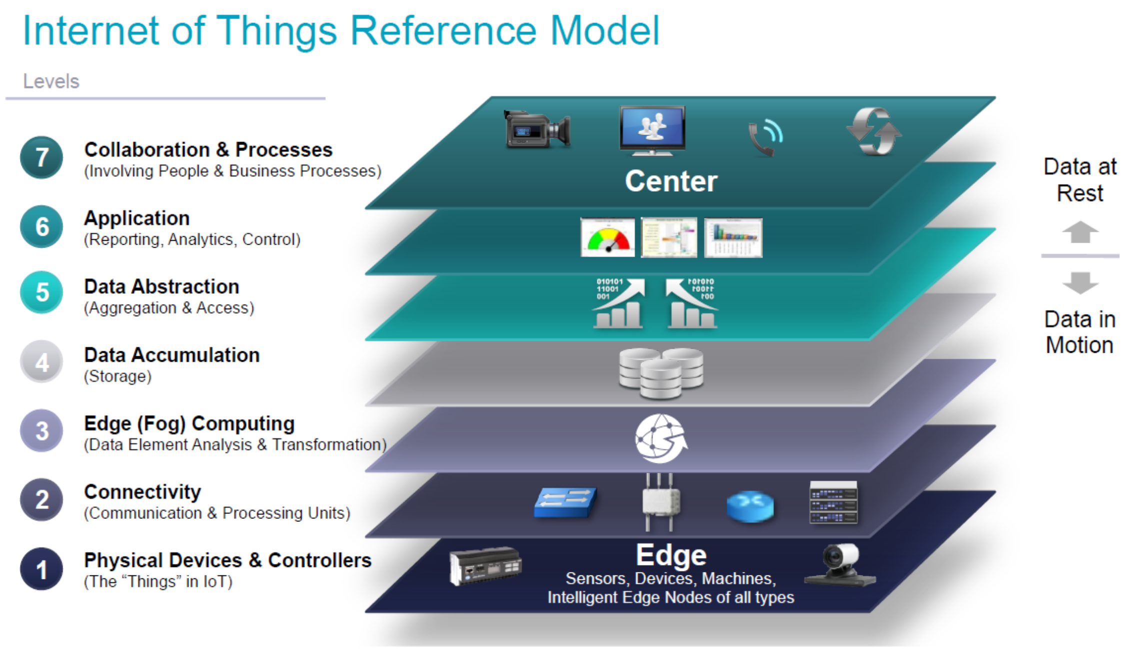


Figure 2.1 – Basic IoT architecture [19]

2.1 IoT Model

In traditional TCP/IP networked environments, technologies, communication, and protocols are mapped to either the OSI reference model, or the TCP/IP stack (or both) since it helps to visualize the communication. Jim Green from Cisco published in 2014 the Internet of Things Reference Model shown in Figure 2.2. It provides a nice representation in the IoT space to better understand where threats and vulnerabilities exist and where intrusion detection can be beneficial. [15]


Internet of Things Reference Model
Levels
7 Collaboration & Processes
(Involving People & Business Processes)
6 Application
(Reporting, Analytics, Control)
5 Data Abstraction
(Aggregation & Access)
4 Data Accumulation
(Storage)
3 Edge (Fog) Computing
(Data Element Analysis & Transformation)
2 Connectivity
(Communication & Processing Units)
1 Physical Devices & Controllers
(The "Things" in IoT)
Center
Edge
Sensors, Devices, Machines,
Intelligent Edge Nodes of all types
Data at Rest
Data in Motion

http://cdn.iotwf.com/resources/71/IoT_Reference_Model_White_Paper_June_4_2014.pdf

Figure 2.2 – IoT Reference Model [15]

The 1st layer shows the physical infrastructure focused on the actual tangible things and includes attacks against devices using interfaces like Joint Test Action Group, Universal Asynchronous Receiver/Transmitter, Serial Wire Debug, Serial Peripheral Interface among many more. Attacks on this layer give the attacker direct access to the device console, with the ability to control execution of firmware, leakage of firmware secrets (passwords, hard-coded keys etc.). JTAG and UART mentioned above are mainly debugging tools during development and troubleshooting but can be indeed abused to gain physical access to IoT devices from a third party. Interface access to these devices can be taken advantage of by sniffing and modifying TDI/TDO signals, control TMS and TCK signals, while also giving access to keys used during testing. [34]

At the next layer, which is the link layer, a variety of protocols and technologies is included, such as common wireless local area networks or WiFi, personal area networks (RFID, NFC, Bluetooth, ZigBee, 6LoWPAN, Z-Wave), cellular technologies like LTE or GPRS and more. IEEE 802.15.4 based ZigBee and Z-Wave are affected by current wireless attacks. “ZigBee is a mesh network specification for low-power wireless local area networks (WLANs) that cover a large area” [35]. At events such as BlackHat, ZigBee attacked were showcased on how to capture encryption keys by sniffing ZigBee devices in the process of joining the network [48]. Joshua Wright developed KillerBee1, which allows eavesdropping on ZigBee networks, replay traffic, attack cryptosystems and more [46]. Similarly, Z-wave is described on its website as "a wireless radio frequency technology that lets smart devices talk to and connect with one another" [43]. A popular tool called EZ-Wave was used to attack Z-Wave, created by Joseph Hall and Ben Ramsey. "EZ-Wave uses commodity Software Defined Radio to exploit Z-Wave networks” [43], employing DoS attack where Z-wave controlled lights are destroyed or disabled.

At the 3rd layer exist the edge devices or in other wards the Fog computing. Fog computing is referred to as the layer that brings the cloud closer to the edges of a network, addressing the enormous chunks of data that IoT devices can and will be sending for processing towards the cloud. Taking into consideration the current and continuous growth of the IoT market, Fog computing is proven to make more sense. Moreover, the setup of such environment is very fertile with regards to intrusion detection. Having the computation at the edge devices means having intrusion detection capabilities closer as well, making the containment and control of attacks much more efficient and effective. [8]

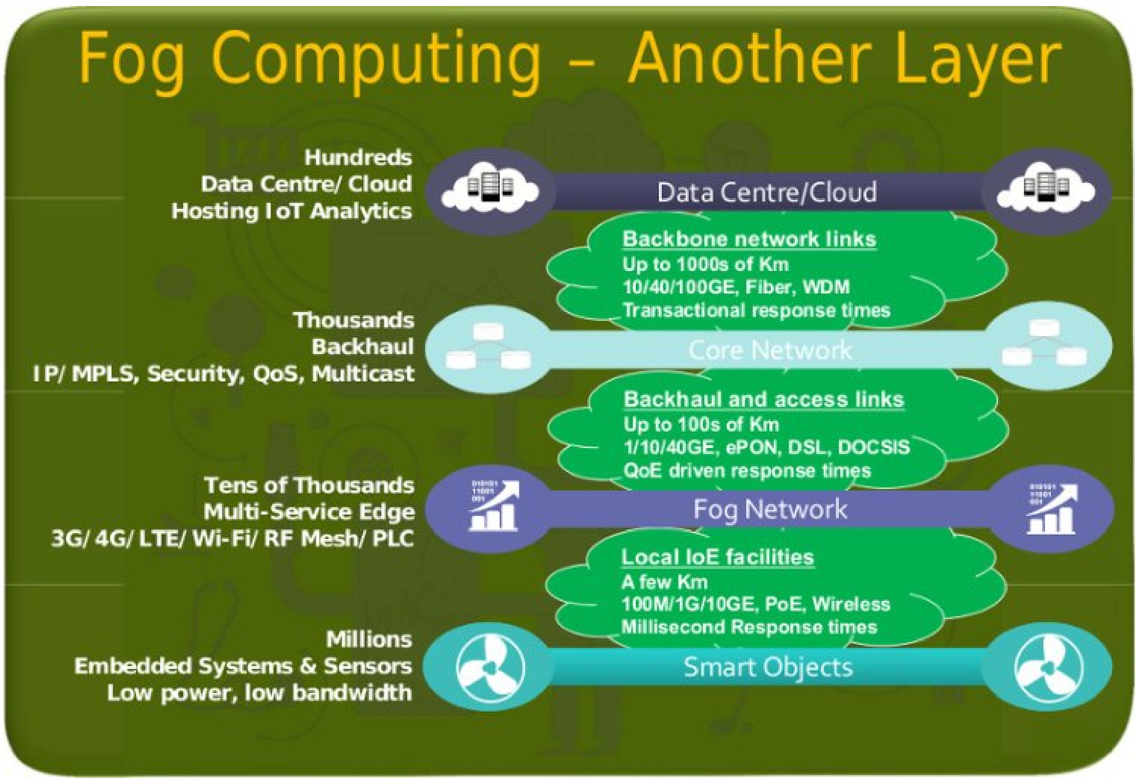


Figure 2.3 – Fog Computing [8]

The remaining levels above of the layers that were just explained can be incorporated into a single application layer, as further explanation is not currently required. Multiple points of intersection exist in the application layer between the IoT and related supporting application technologies. The cloud accommodates smart devices using REST APIs, WebSockets, HTTP, which are under the control of generic web application threats and vulnerabilities. Another set of attack vectors is introduced with protocols such as telnet, SSH and HTTP/S being used through app services to manage IoT devices. Of course, there are more applications and protocols that can be used such as Message Queuing Telemetry Transport), Constrained Application Protocol (CoAP), Extensible Messaging and presence Protocol (XMPP) and JavaScript (node.js mainly) among many others. There are different perspectives that are taken when attacking the application layer; application data, database of users or personal information of a service or device, or simply to bypass mechanisms to gain access to lower layers (buffer overflows or injections exploits). In the end the weaknesses of the application layer are taken advantage of to gain access to a physical device. [38]

2.2 Common Threats and Vulnerabilities

Common threats and vulnerabilities can be identified from well-documented sources readily available throughout the internet. Among the most trusted and examined sources is the Open Web Application Security Project (OWASP). A list of the top IoT vulnerabilities is maintained by the OWASP IoT Project.

This gives the industry some insight into security issues and vulnerabilities that are actually out there. A more comprehensive list of vulnerabilities and threats can be made out of this by mapping the attack surface and identifying recent attacks. Some categories of the OWASP IoT Attack Surface Areas list showcases other types of attacks, particularly on IoT networking, mentioning routing attacks, DoS (Denial of Service), attacks, Sybil attacks and others. [28]

2.3 Threat Modeling: Discovering

A key method to identify threats from a more calculated perspective is Threat Modeling. This typically entails numerous steps to orderly understand the system being modeled, which consequentially identifies potential vulnerabilities.

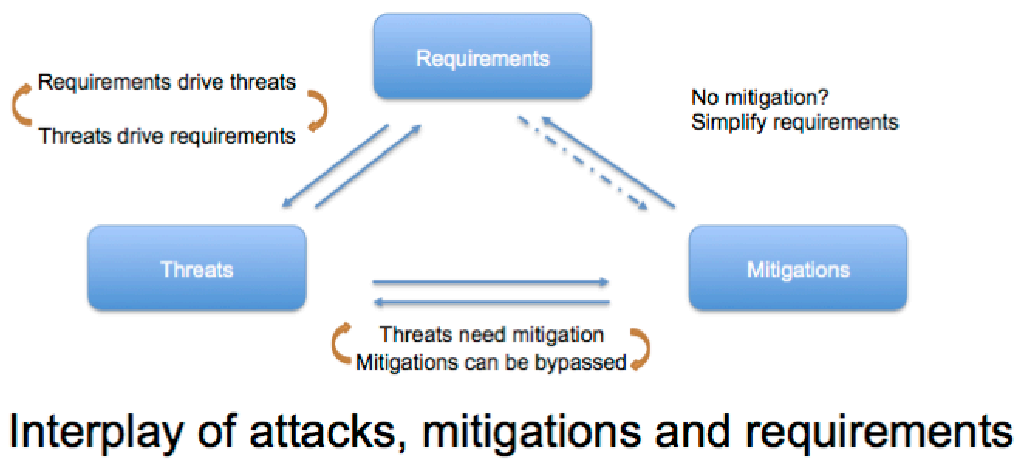

Requirements
Requirements drive threats
Threats drive requirements
No mitigation?
Simplify requirements
Threats
Mitigations
Threats need mitigation
Mitigations can be bypassed
Interplay of attacks, mitigations and requirements

Figure 2.4 – Threat Modeling [1]

In a perfect scenario a given IoT application would be taken apart based on function and labeled as a device, a cloud gateway, a field gateway or a service. Every single one is distinct by its own trust borders and has unrelated requirements around authorization and authentication, data used etc., which will all take part in the threat model procedure. Once modeled, each part of the system can be computed for threats using the STRIDE model. STRIDE is an acronym for Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service and Elevation of Privileges. Establishing potential threats can also be achieved through attack trees. When identified, an intrusion analyst can better comprehend what to execute and where. [33]

Chapter 3

Understanding Intrusion Detection in IoT

In IoT networks intrusion detection usually gets categorized in either traditional IP networks or Low-power Wireless Personal Area Networks – LoWPAN. In both cases there are some requirements in order for network intrusion detection to be successful. This encompasses the means to capture traffic, detect attacks in real time and to be able to understand the IoT communication architecture, where the actual faulty mechanisms exist.

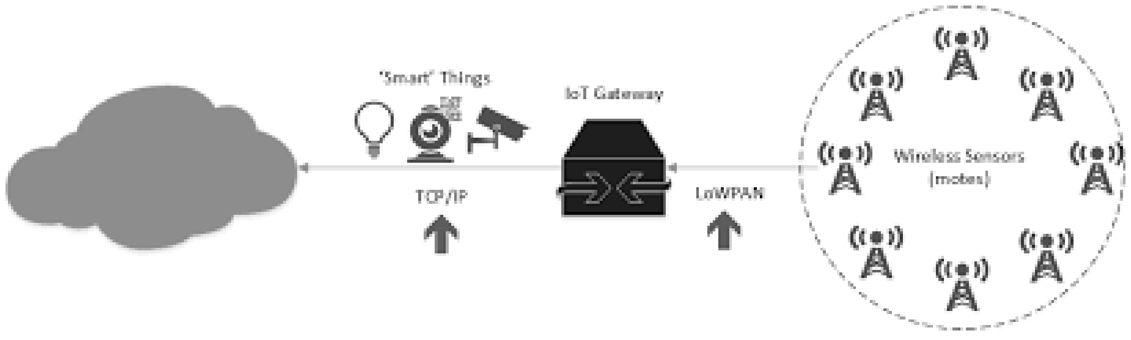


Figure 3.1 – TCP/IP vs LoWPAN IDS [24]

3.1 IoT Communication Stack

IoT communication covers a broad spectrum of objects such as devices, widespread applications and an extensive range of circumstances. There are the traditional transport and network layers of TCP, UDP, IPv4 and IPv6, meanwhile at the lower levels exist a bit more unique link layer and application protocols and technologies, where additional research is required. The IoT communication stack and widespread protocols are shown in Figure 3.2.

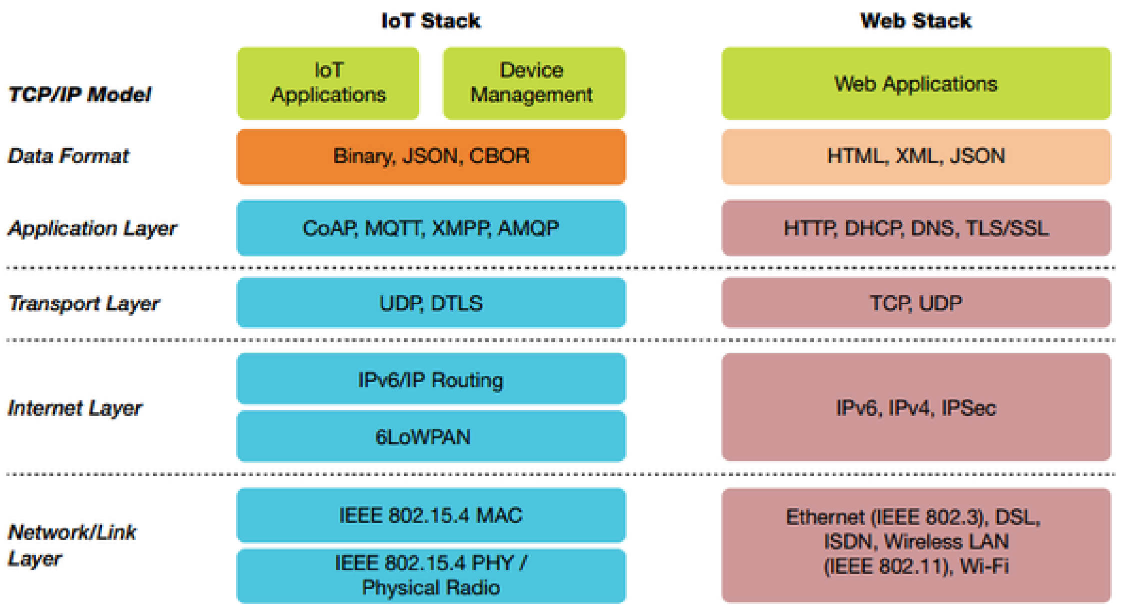


Figure 3.2 – IoT Communication Stack [16]

3.1.1. Frequent IoT Application Protocols

A publish and subscribe based messaging protocol designed for M2M comms is Message Queuing Telemetry Transport, or otherwise MQTT, which its main use is in monitoring and measuring applications, such as home automation projects. Communication in MQTT is done over TCP/IP. Kindred to the common client – server communication, MQTT uses pub/sub – publish/subscribe to communicate in between clients who are serving content to others that are subscribed to them. An MQTT broker is needed in this situation, which in simpler terms acts as a proxy. It breaks down publishing clients from subscribing clients, so there is no knowledge of the existence of each other, but only of the MQTT broker. Thus messages are sent without knowing the destination and with no restrains in time. The broker receives a 'connect' packet from clients and responds back to them with 'connack' packets. [38]

Specialized IoT applications such as smart energy and building automation use CoAP as a web transport protocol (Constrained Application Protocol). It makes use of UDP at the transport layer, so the entirety of the communication state is handled by the applications. CoAP follows the traditional client-server model, using the four main HTTP methods of GET, PUT, POST and DELETE. Utilizing UDP, CoAP makes use of constrained resources, giving it the ability to communicate via other packet-based communication protocols or even SMS. [38]

The Extensible Messaging and Presence protocol known as XMPP offers encryption and authentication, being an open and standards-based protocol. It originally started as a Jabber messaging protocol, but it is mainly used in chats, voice and video calls, instant messaging and generic user-defined XML data. Being relatively popular means that it is more likely for its weaknesses and vulnerabilities to be exposed. For example, nmap scanner encompasses scripts to enumerate users by checking for XMPP. A gateway called XMPPloit was developed to be situated between client-server and is able to endanger their communication. [38]

3.1.2 Frequent Link Layer Protocols and Technologies

Resource constrained devices make up the IoT environments. Such environments need different protocols in order to communicate at the lower levels. Bluetooth Low Energy was developed with that goal in mind (Bluetooth LE or BLE). BLE is based on RF (radio frequency) communication, where small bursts of RF are used where energy conservation is a concern and continuous RF is not needed. There are two modes of operation, the first one being the simple client-server communication, in which case one device acts as an edge device and talks to a different central one. A simple bi-directional example is a wireless sensor network between two devices. The second mode being a unidirectional communication where a device is either listening for data or broadcasting, but never both. The communication in both cases is done with beacon frames. [25]

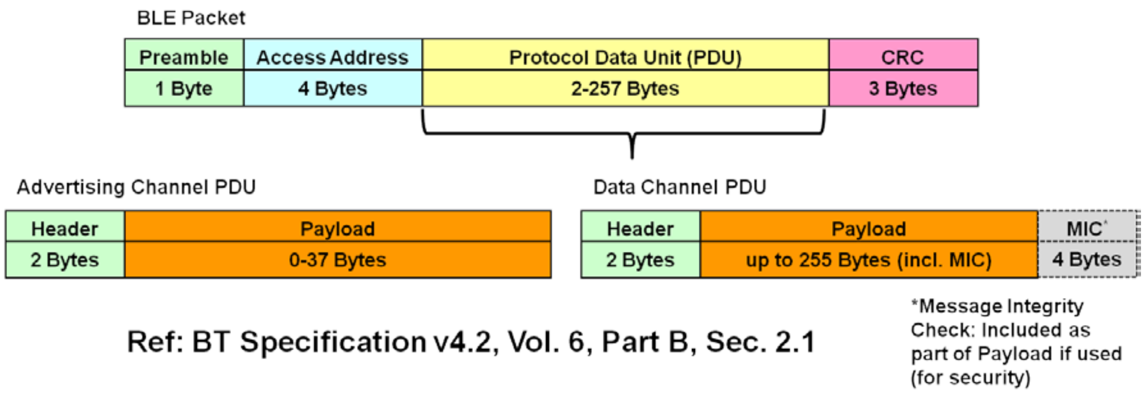


Figure 3.3 – Bluetooth Low Energy packet [27]

An example of BLE beacon technology is Apple’s iBeacon. It was designed to aid retail areas, in an effort to improve user experience iBeacon transmitters are supposed to be sending messages in malls, stores etc.

ZigBee is a wireless mesh technology, akin to BLE, that had the goal of addressing the low power RF communications needed by IoT edge devices. It is based on the IEEE 802.15.4 standard and extends along the full stack from application layer down to the link layer. It is usually applied in smart home environments, as well as the utility industry. Attributes of the 3.0 version comprise device unique authentication, secure firmware upgrades over the air, logical link-based encryption and runtime key updates. The frame of ZigBee is shown in figure 3.4. [47]

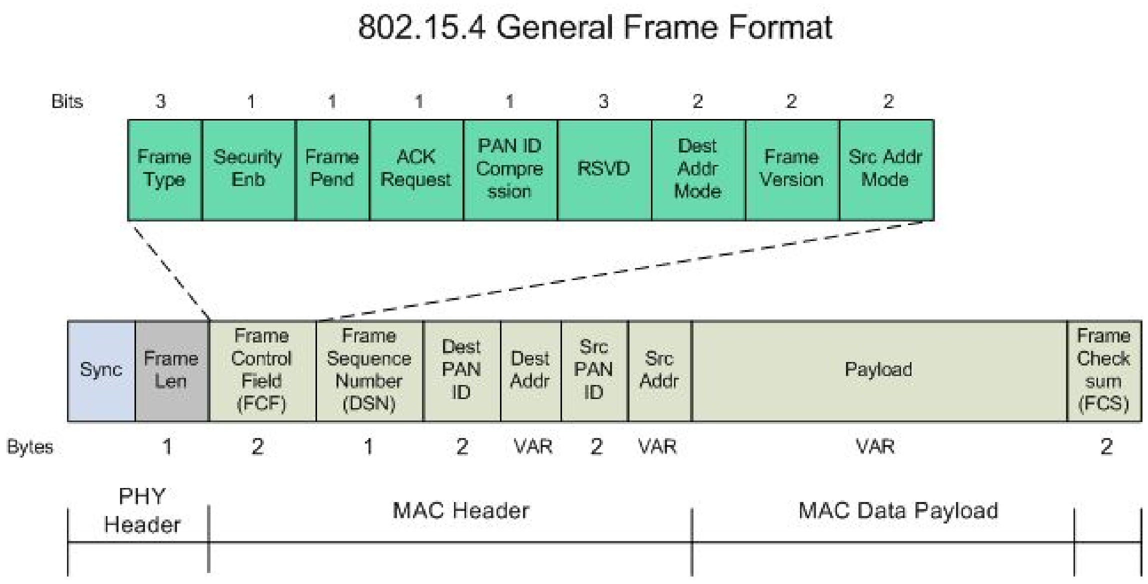


Figure 3.4 – ZigBee frame format [4]

3.2 Monitoring Communication

An important segment of intrusion detection in networks is the capacity to sniff or monitor traffic. Conventional TCP/IP networks have endless resources available, while for IoT this part becomes difficult. Dissectors are incorporated in tools like Wireshark to comprehend a broad spectrum of protocols, when in contrast, specialized RF tools are required in order to sniff traffic in-between IoT devices.

For example, using Wireshark alongside a BLE adapter traffic between BLE devices can be monitored. Below in figure 3.5, a ‘BLE Friend’ from Adafruit is displayed to work in combination with Nordic BLE sniffer application, making it possible to be monitored and analyzed in Wireshark, while at the same time detecting BLE devices and signal strength. [44]

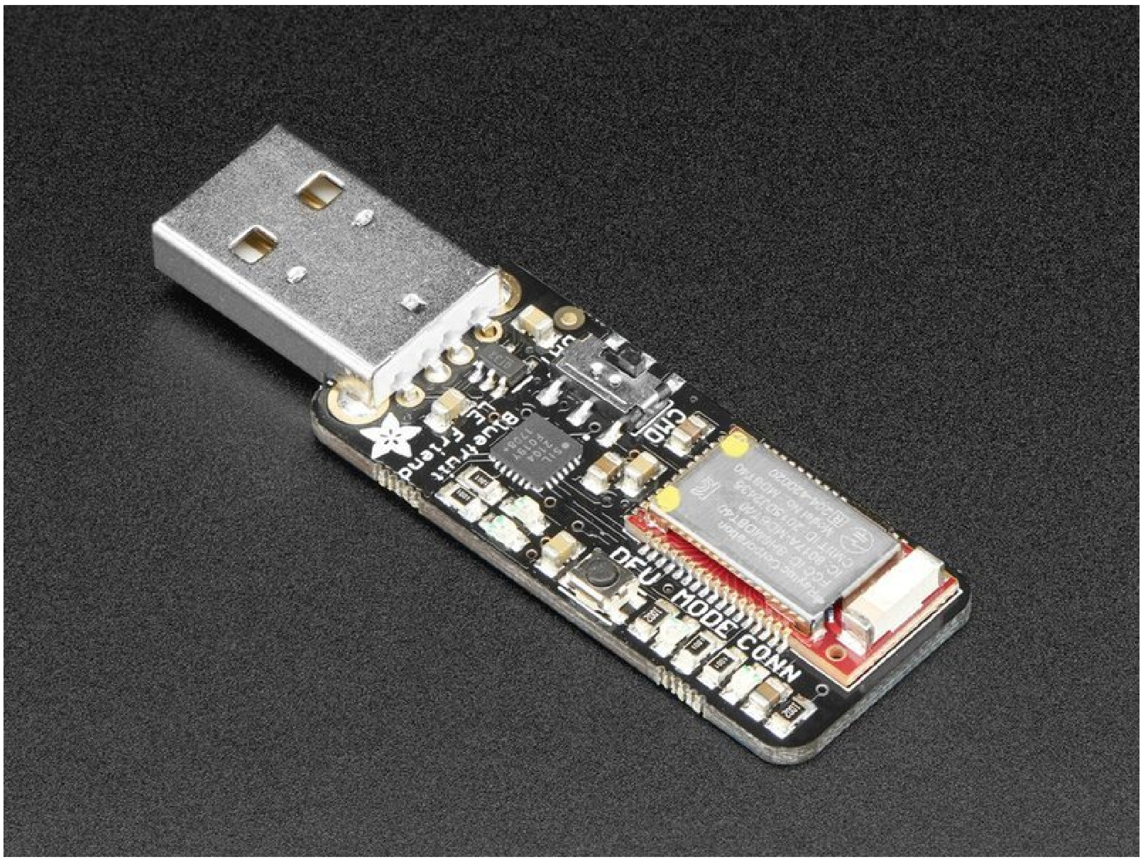
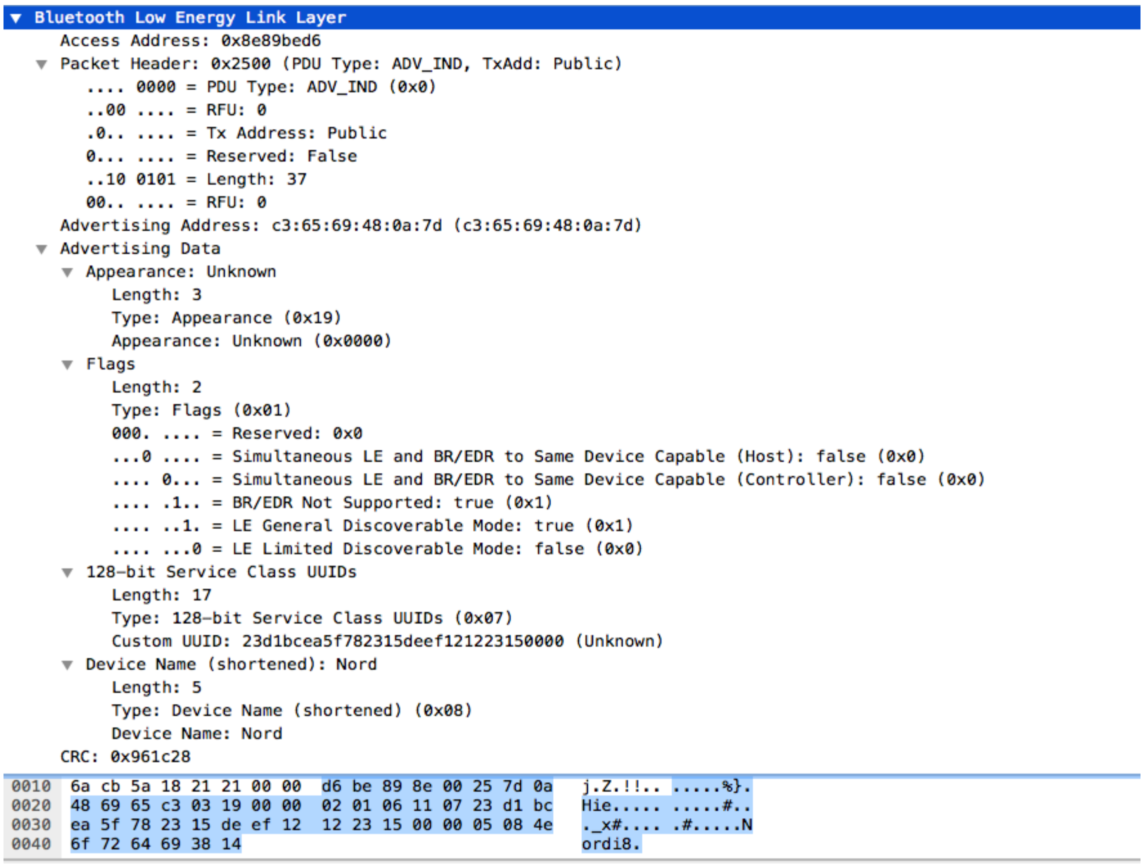


Figure 3.5 – Bluefruit LE Adapter from Adafruit [44]

The start of the communication will be show accessible BLE devices and appropriate RF signal data. To limit transactions that are displayed filtering needs to be done on specific devices. The BLE beacons that are advertising are shown in figure 3.6, informing the intrusion analyst extensively of the packets that may provide useful to figuring out intrusions.



```
▼ Bluetooth Low Energy Link Layer
    Access Address: 0x8e89bed6
  ▼ Packet Header: 0x2500 (PDU Type: ADV_IND, TxAdd: Public)
      .... 0000 = PDU Type: ADV_IND (0x0)
      ..00 .... = RFU: 0
      .0.. .... = Tx Address: Public
      0... .... = Reserved: False
      ..10 0101 = Length: 37
      00.. .... = RFU: 0
    Advertising Address: c3:65:69:48:0a:7d (c3:65:69:48:0a:7d)
  ▼ Advertising Data
    ▼ Appearance: Unknown
        Length: 3
        Type: Appearance (0x19)
        Appearance: Unknown (0x0000)
    ▼ Flags
        Length: 2
        Type: Flags (0x01)
        000. .... = Reserved: 0x0
        ...0 .... = Simultaneous LE and BR/EDR to Same Device Capable (Host): false (0x0)
        .... 0... = Simultaneous LE and BR/EDR to Same Device Capable (Controller): false (0x0)
        .... .1.. = BR/EDR Not Supported: true (0x1)
        .... ..1. = LE General Discoverable Mode: true (0x1)
        .... ...0 = LE Limited Discoverable Mode: false (0x0)
    ▼ 128-bit Service Class UUIDs
        Length: 17
        Type: 128-bit Service Class UUIDs (0x07)
        Custom UUID: 23d1bcea5f782315deef121223150000 (Unknown)
    ▼ Device Name (shortened): Nord
        Length: 5
        Type: Device Name (shortened) (0x08)
        Device Name: Nord
    CRC: 0x961c28

0010  6a cb 5a 18 21 21 00 00  d6 be 89 8e 00 25 7d 0a   j.Z.!!.. .....%}.
0020  48 69 65 c3 03 19 00 00  02 01 06 11 07 23 d1 bc   Hie..... .....#..
0030  ea 5f 78 23 15 de ef 12  12 23 15 00 00 05 08 4e   ._x#.... .#.....N
0040  6f 72 64 69 38 14                                   ordi8.
```

Figure 3.6 – BLE Advertising packet [20]

For networks over IEEE 802.15.4, such as ZWave and ZigBee, a variety of tools can be used to sniff traffic: a RaspBerry Pi mounted with Raspbee, XBee and a set device of APIMOTE (version 4).

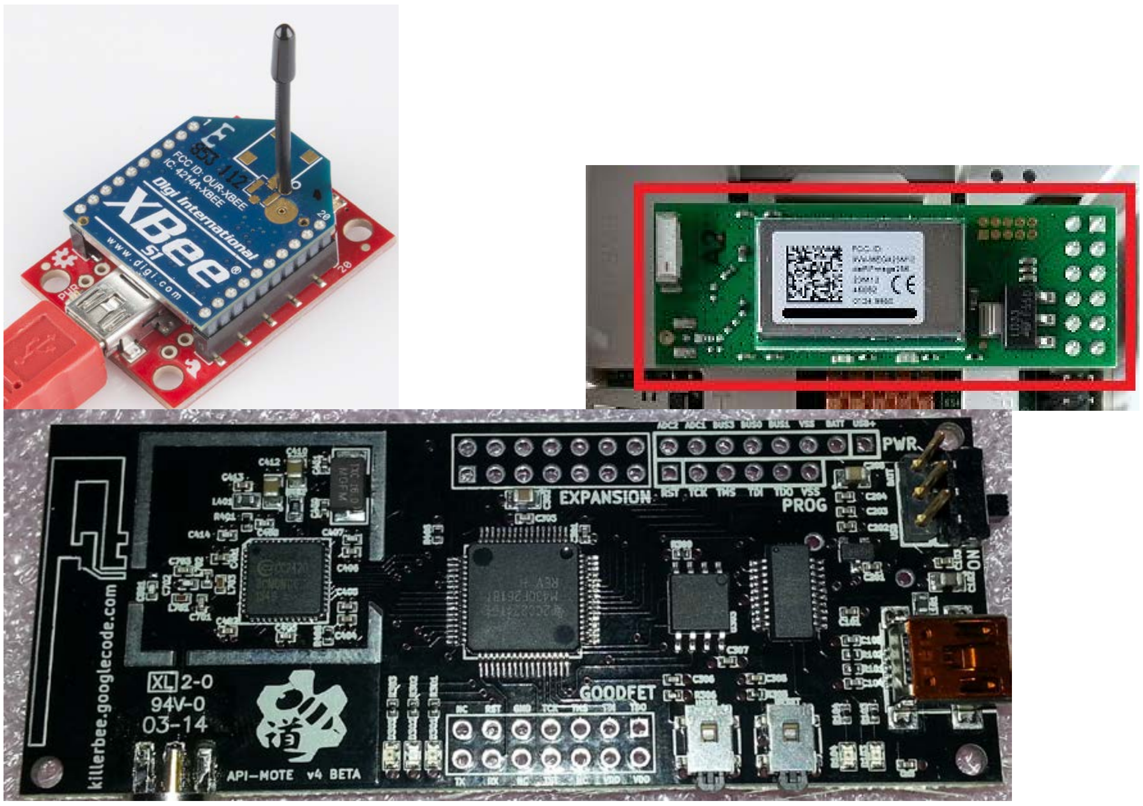


Figure 3.7 – Xbee, RaspBee, APIMOTEv4 [21, 18, 37]

3.3 IDS for IoT

After addressing the need for sniffing and having the ability to monitor traffic, we can now look deeper into intrusion detection implementations. There are various solutions readily accessible when IoT is concerned. IoT environments are usually different from one another and different tools should be used based on the use case of each scenario. Explored below are the most popular tools: Bastille, BeeKeeper, Raspberry Pi IDS – RPiDS and Snort.

3.3.1 Bastille

Bastille is a commercial product of the Bastille Networks company focused on software defined radio threat detection. To pinpoint RF based threats in an IoT environment it cans frequency from 60MHz up to 6GHz. Bastille managed to stand out in their security research, recognizing the KeyJack and MouseJack vulnerabilities. Their technology comprises of RF spectrum monitoring, discovery of connected devices and the physical location of those, along with their respective names for each as follows: "Collaborative

Bandit Sensing", "Bayesian Device Fingerprinting" and "Distributed Tomographic Localization". [10]

3.3.2 BeeKeeper

Developed by River Loop Security, BeeKeeper was a Wireless Intrusion Detection System - WIDS based on IEE 802.15.4 standard, to work together with the KillerBee framework. It was designed to detect attacks in 6LoWPAN, ZigBee and 802.15.4 networks, sniffing the RF spectrum with sniffers mentioned previously, such as APIMOTEv4.Then, the data is processed through a drone responsible for managing channels and interfaces. As of right now, the original repository was last updated on 2014. [36]

3.3.3 Snort

Snort was developed at first by Marty Roesch in 1998. It is an open source intrusion detection system, very popular within its industry and maintained at Snort.org. It is capable of performing real-time traffic analysis and packet logging on IP networks. It can perform protocol analysis, content searching/matching, and can be used to detect a variety of attacks and probes, such as buffer overflows, stealth port scans, CGI attacks, SMB probes, OS fingerprinting attempts and many more. With build-in preprocessors and rules, Snort can be used as a packet logger, sniffer and intrusion prevention/detection system. [42]

3.3.4 Raspberry Pi IDS - RPiDS

A paper was released regarding the use of Snort inside of Raspberry Pi’s. A group of 4 German researchers, Conti, Marmol, Sforzin and Bohli, have published an outline of applying such architecture as seen in figure 3.8. The main focus was for it to be an IoT deployment with easy of use and portability capabilities, designed as such that a node could be relocated independently, based on the needs of the network, while maintaining scalability where more Raspberry Pi’s could be deplayed and run either collaboratively or independently, processing data from other node sensors. This results in the ability to correlate traffic data from local sensors, lower the rates of false positives and even detecting network topology related attacks, that would be overlooked in other cases. [39]

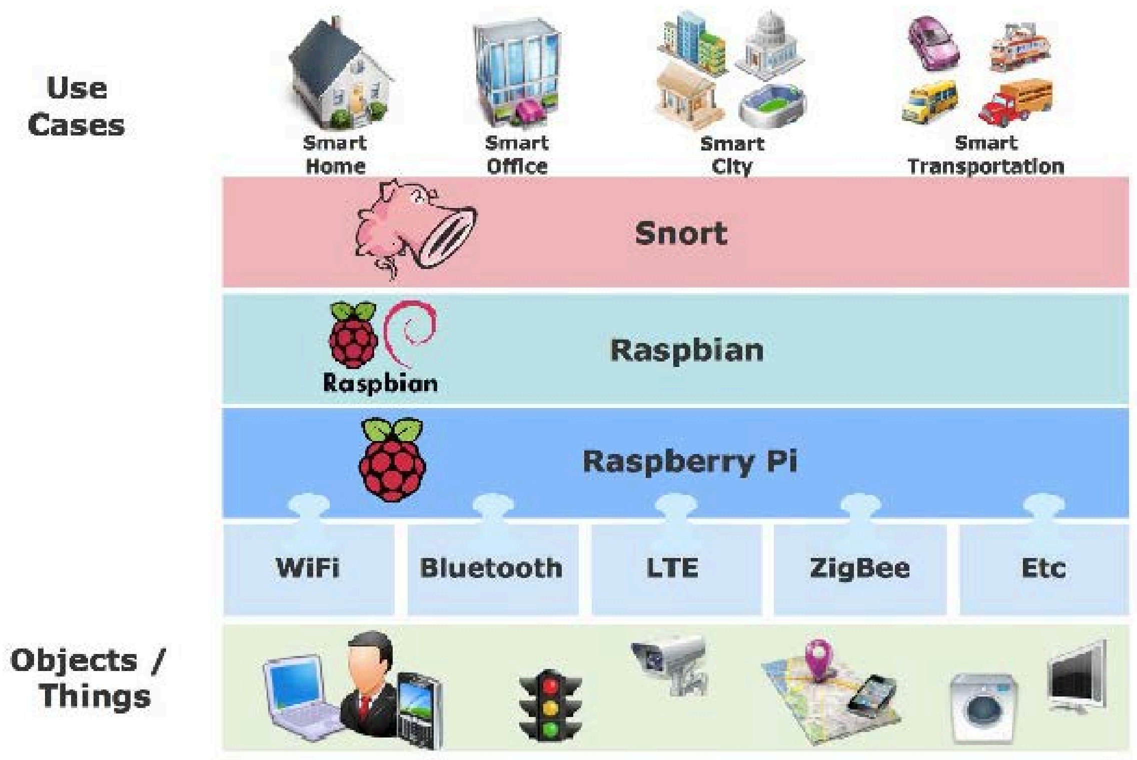


Figure 3.8 – RpiDS Architecture [39]

Chapter 4

IoT Attacks and Detection

Intrusion detection is better understood when divided into traditional IP networks and LoWPANs, where different technologies exist to make detection and identification of attacks possible. With the common IP networks we have a more mature environment with established standards in detection mechanisms, but in regards to wireless networks, not so much. Each wireless environment can differentiate from the previous and having a less mature and studies industry results in specialized scenarios. Following will be Mirai, a major attack within the IP network and afterwards some more distinct attacks in the wireless environments with BLE, 6LoWPAN and ZigBee.

4.1 IoT Mirai Botnets

On 2016 the Mirai botnet hit a good number of IP connected devices. The botnet took advantage of the lack of standards of manufacturers, that speed up their processes to hit the market faster, making it easier for the attack to target default credentials on interconnected devices. When the source code of the attack was released to the public in 2016, intrusion analysts were able to develop intrusion detection tools around it. [5]

In simple terms, the attack scanned for vulnerable devices and then compromised them. This was achieved through a network of bots, of which initiated a brute force scan of IP addresses, trying to match login credential against a database of known device credentials. [9]

The most important file of the release was “scanner.c”, where the setup of the network packet was shown for both IP and TCP headers, along with the complete credentials list that was being tried against scanned devices. Knowing this information, it is relatively easy for Snort to work against this threat through its content matching capabilities, identifying the attack based on its packet data.

When running the command “strings” against a binary network capture file generated from Mirai, there are over 4000 readable strings to the human eye. Those being related to the login attempts that are run against devices, it is then possible to search for said packet through Wireshark using the “find packet” feature. Now it is possible to analyze the details of the specific packet and its payload, correlating the data with the source code. It is then possible to create Snort rules based on the information gathered. The scanner uses TCP as its transportation protocol on port 23 and having the content to match against, a rule can be created like in Figure 4.1. The purpose of this rule is to alert for content between any source and destination, where the destination port would be tcp/23 and the content to match against “vizxv”. The result alert of this rule run against the Mirai capture file is shown in Figure 4.2. [24]

```
alert tcp any any -> any 23 (msg:"Mirai Test Rule – inbound login attempt";
flow:to_server,established; content:"vizxv";depth:6; sid:1000000;)
```

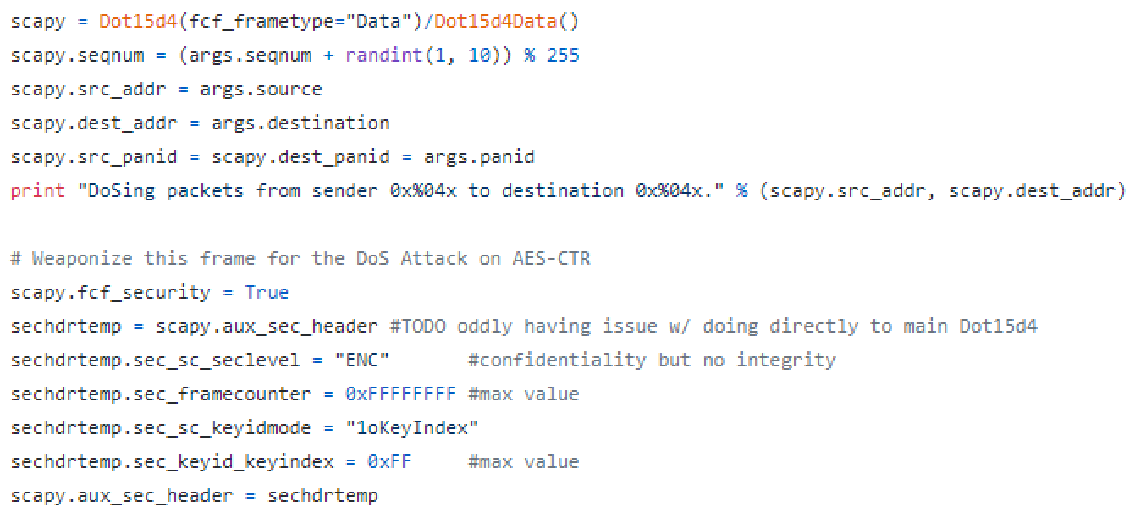
Figure 4.1 – Snort test rule for Mirai login attempt [24]

```
[**] [1:1000000:0] Mirai Test Rule – inbound login attempt [**] [Priority: 0] 01/30-11:15:30.534758
192.168.88.240:35124 -> 192.168.89.10:23 TCP TTL:45 TOS:0x20 ID:54321 IpLen:20 DgmLen:46 DF
***AP*** Seq: 0x7611E4E7 Ack: 0x40DCBD2 Win: 0x5AC TcpLen: 20
```

Figure 4.2 - Result of Snort against Mirai pcap file using test rule [24]

4.2 ZigBee DoS AES-CTR Attack

In order for ZigBee to include encryption with replay protection AES-CTR security mode was implemented. Inspection of the frame counter is done to make sure that identical values do not exist, thus protecting from replay attacks. If such a frame is found with the same counter value, it is deleted. A smart way to work around this is to send a packet with frame counter value of FFFFFFFF, which is the maximum value for the frame counter field, making any subsequent and potentially “real” packet to be deleted by the inspection. This is how the DoS attack against ZigBee AES-CTR kills the ensuing of the communication. [36]



```
scapy = Dot15d4(fcf_frametype="Data")/Dot15d4Data()
scapy.seqnum = (args.seqnum + randint(1, 10)) % 255
scapy.src_addr = args.source
scapy.dest_addr = args.destination
scapy.src_panid = scapy.dest_panid = args.panid
print "DoSing packets from sender 0x%04x to destination 0x%04x." % (scapy.src_addr, scapy.dest_addr)

# Weaponize this frame for the DoS Attack on AES-CTR
scapy.fcf_security = True
sechdrtemp = scapy.aux_sec_header #TODO oddly having issue w/ doing directly to main Dot15d4
sechdrtemp.sec_sc_seclevel = "ENC"       #confidentiality but no integrity
sechdrtemp.sec_framecounter = 0xFFFFFFFF #max value
sechdrtemp.sec_sc_keyidmode = "1oKeyIndex"
sechdrtemp.sec_keyid_keyindex = 0xFF     #max value
scapy.aux_sec_header = sechdrtemp
```

Figure 4.3 – AES-CTR Attack code [36]

With the use of two APIMOTEv4 devices and the BeeKeeper WIDS framework, a detection system can be setup. The devices capture traffic, of which is then directed to drone modules that work with the WIDS. The drones handle the interface for packet capture, channel selection and analysis of captured data to take a decision against potential malware and in turn the WIDS manages the drones. [36]

4.3 6LoWPAN Networks Routing Attacks

Lastly, we have IPv6 over Low Power Wireless Personal Area Networks, or else 6LoWPAN, which was created as a wireless standard for IP networks with constrained resources, and more specifically Low-Power and Lossy Networks – LLNs. Allows devices with limited power and processing capabilities to transmit data through the air as an internet protocol, being able to communicate with 802.15.4 as well as various other types on an IP network link. The main protocol used by 6LoWPAN is the Routing Protocol for Low-Power and Lossy Networks – RPL -, which is threatened by many routing attacks, few of them being Sybil, selective forwarding, wormhole and sinkhole. We are going to take a look at the latter two.

Both sinkhole and wormhole attacks work in a similar manner to bring destruction to the routing of 6LoWPAN ecosystems. A node inside the network is affected by malicious code, giving it access to an outsider and making it declare the knowledge of the shortest path to a target host. This gives it the ability to control and manipulate traffic either by discarding packets or re-directing. [45]

As mentioned, wormhole attacks work more or less the same way. Their difference is the fact that more than one node is compromised. A tunnel is established between the two nodes, either them being both inside the same 6LoWPAN network or even communicating from outside the border router. Traffic is again monitored and controlled, but the impact of re-transmission is even bigger in this case, tiring and consuming the limited power both in battery and processing of the several existing devices. [14]

Now, due to the nature of these attacks, countermeasures against them are not simple at all. Starting with the sinkhole attack, Intrusion Detection for Sinkhole Attacks Over 6LoWPAN for Internet of Things – INTI – was develop, which can isolate the nodes affected on top of detecting the attack, using watchdog, node reputation and trust that is developed between the communicated nodes throughout the environment. The results of the detection stems from four modules. Route monitoring, cluster configuration, attack detection and isolation. A hierarchical configuration is achieved between the nodes themselves with leaders and members. Incoming and outgoing streams are being monitored for abnormalities outside of the expected one on one communication. Pairing this along with the reputation based on trust within the hierarchy, INTI takes action towards isolating and reconstructing the routes. [11]

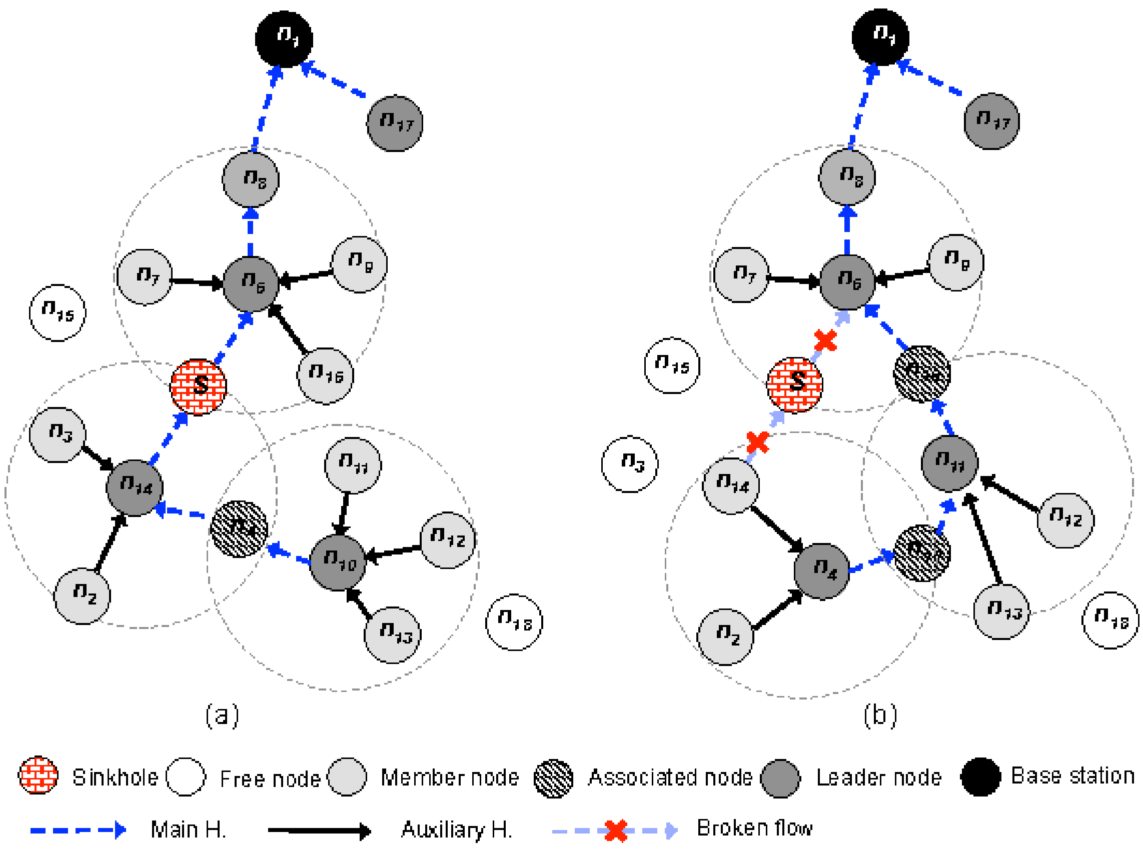


Figure 4.4 – Sinkhole detection [11]

Wormhole on the other hand is even more complicated. Similar route and node communication monitoring intrusion detection technology is used, within a distributed system working along with the IPv6 border router gateway to achieve distributed detection with the wireless devices. The border gateway provides a stronger processing unit and resources, being able to achieve better detection of abnormalities as the main node that analyzes the hierarchy placement. The devices within the system have the role of monitoring and informing the gateway. RSSI levels are included which take part in the decision making. For example, on Figure 4.6 the router gateway is able to conclude that the route advertised between nodes 10 and 13 should be out of range of each other and thus declaring the detection of an attack. [32]

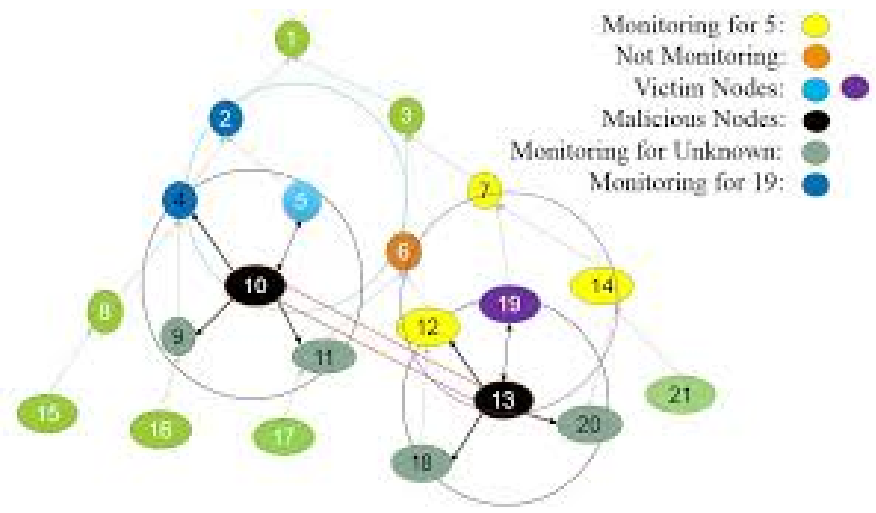


Figure 4.5 – Wormhole attack [32]

Chapter 5

Implementing Intrusion Detection in private IoT Network

How do all the components come into place when there is the need to deploy an IDS in a small and private IoT network, most likely to be used in a personal home or non-corporate project. In this final chapter we will take a look into implementing the Snort IDS within an IoT network of our own using a MikroTik router as the gateway.

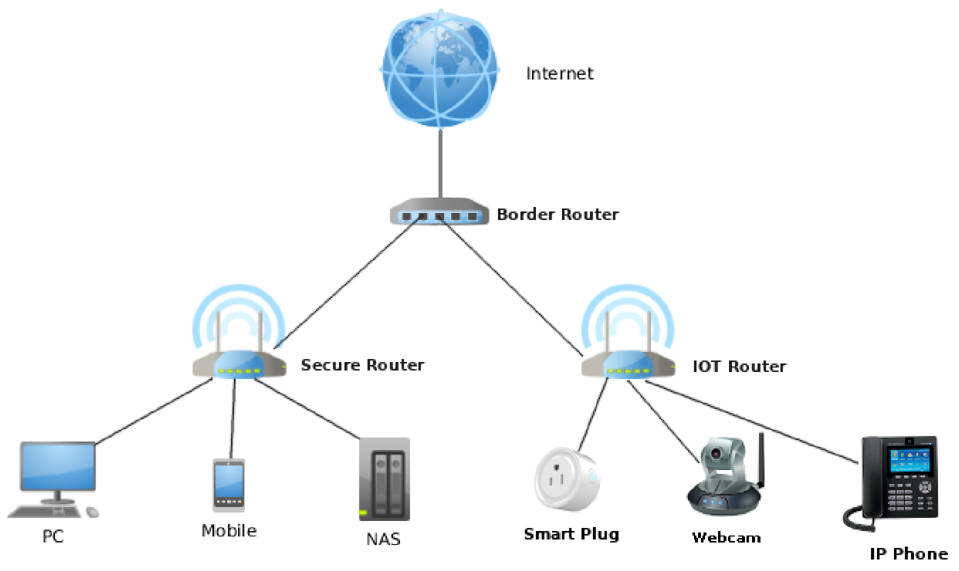


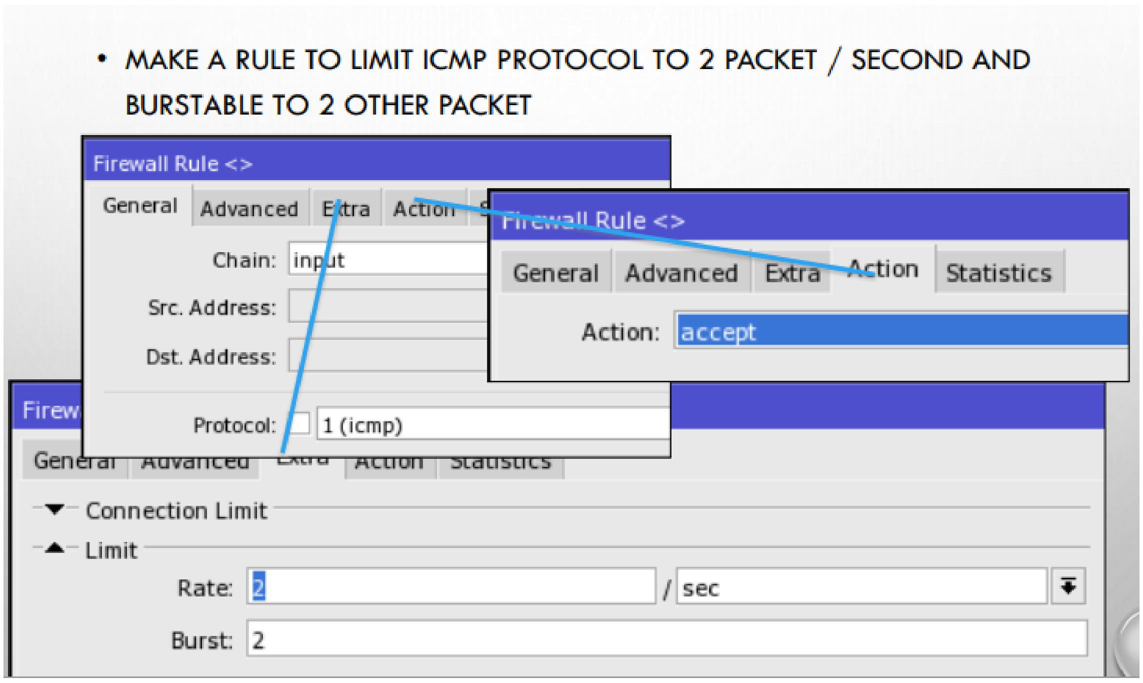
Figure 5.1 – Gateway topology [31]

5.1 MikroTik Router

MikroTik is a company originated from Latvia, that develops and sells both wireless and wired network solutions such as routers, switches access points and operating systems. Using their proprietary RouterBOARD hardware to run the RouterOS operating system, they provide a stand-alone OS based on the Linux v2.6 kernel that can be installed and tested on a PC, with all the included features. Some of them being routing first of all, firewall, wireless access point, bandwidth management, QoS, VPN server, and many more. [41]

The main way of configuration that I used was through their custom GUI config tool called Winbox, that uses their API along with a simple Web based interface. The specific setup of the router will not be explored in this paper, as there are many resources and different approaches. In the Figures below, some basic features of the router can be observed, such as the firewall, bandwidth management with QoS Queue Trees and DHCP server. The main focus of the IDS will be the use of the firewall for an active intrusion detection and prevention system. [40]

Detection through the router is possible, although limited. Thresholds can be set to limit the rate and burst of traffic, mitigating ping or ICMP flooding. The same limitation rules can be used for brute forcing router credentials, along with content matching rules of expected error messages on login attempts. [6]


- MAKE A RULE TO LIMIT ICMP PROTOCOL TO 2 PACKET / SECOND AND BURSTABLE TO 2 OTHER PACKET

Firewall Rule <>

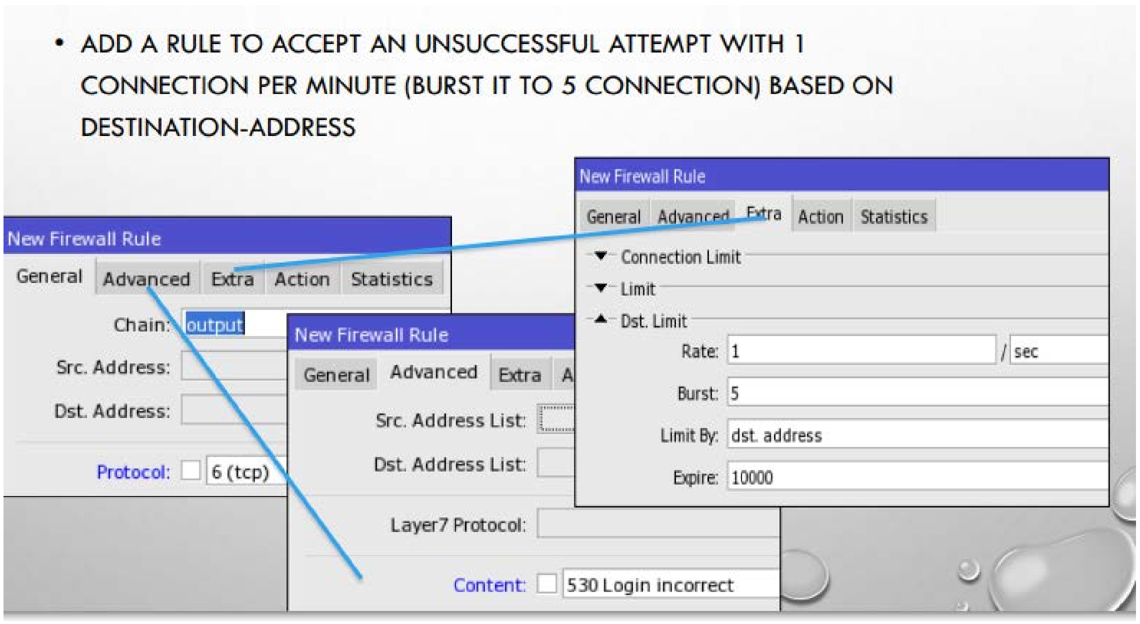
Chain: input
Src. Address:
Dst. Address:
Protocol: 1 (icmp)

Firewall Rule <>

Action: accept

Connection Limit
Limit
Rate: 2 / sec
Burst: 2

Figure 5.2 – ICMP Flooding threshholds [6]



- ADD A RULE TO ACCEPT AN UNSUCCESSFUL ATTEMPT WITH 1 CONNECTION PER MINUTE (BURST IT TO 5 CONNECTION) BASED ON DESTINATION-ADDRESS

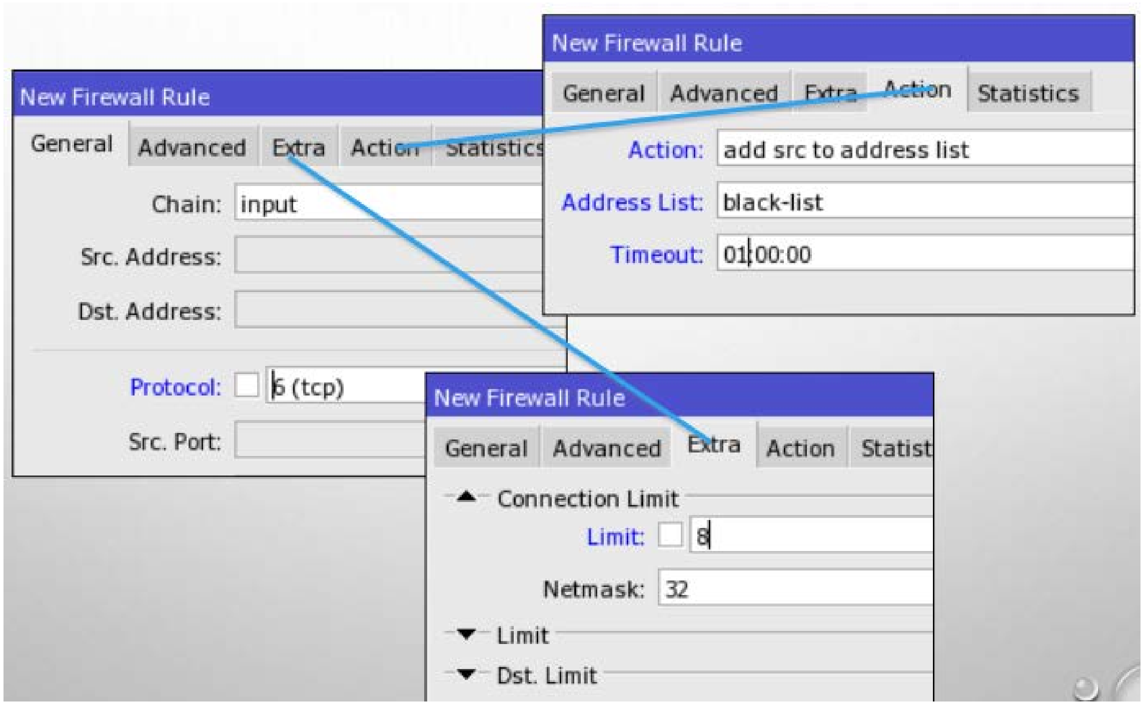
New Firewall Rule (General): Chain: output; Src. Address: ; Dst. Address: ; Protocol: 6 (tcp)

New Firewall Rule (Advanced): Src. Address List: ; Dst. Address List: ; Layer7 Protocol: ; Content: 530 Login incorrect

New Firewall Rule (Extra): Connection Limit; Limit; Dst. Limit — Rate: 1 / sec; Burst: 5; Limit By: dst. address; Expire: 10000

Figure 5.3 – Brute force ftp content match [6]

A different mitigation mechanism is the use of weighted rules against low and high ports, using timeframes and many other parameters to counter port scanning. Furthermore, use of the “tarpit” is considered better in this situation, where instead of dropping incoming traffic from port scanning addresses, they remain ignored, thus timing out the connection without wasting more resources. Tarpit can also be used against DoS attacks, where limits on concurrent connections are set. [6]



New Firewall Rule (General): Chain: input; Src. Address: ; Dst. Address: ; Protocol: 6 (tcp); Src. Port:

New Firewall Rule (Action): Action: add src to address list; Address List: black-list; Timeout: 01:00:00

New Firewall Rule (Extra): Connection Limit — Limit: 8; Netmask: 32; Limit; Dst. Limit

Figure 5.4 – DoS connections limit set [6]

A useful usage of the address list feature is to hold a temporary list (timeout timer) of addresses accessing a certain port, which are then expected to knock on a second (or more) port(s) to gain access through the gateway (port knocking).

5.2 Snort

As mentioned in a previous chapter, Snort is an open-source intrusion detection system, able to perform real-time analysis on traffic and packet logging with its content matching capabilities both in ASCII and HEX form. In this project it was deployed on a Linux machine hosting the Ubuntu 18.04 OS, but in general it can even be deployed on an independent Raspberry Pi or any device that supports it. The installation of Snort was done following a comprehensive guide from snort.org. [13]

To be able to analyze the traffic Snort needs direct access to it, but since the router and the Linux machine are separate entities, a method of re-directing is needed. In this case, a firewall entry on the router allows for it to send traffic towards a sniff target as shown in Figure 5.5. This is achieved from the CALEA (Communications Assistance for Law Enforcement Act) feature of the router that intercepts and logs network traffic.

```
/ip firewall calea add action=sniff chain=forward sniff-target="Snort device IP" sniff-target-
port=37008

/ip firewall calea add action=sniff chain=input sniff-target="Snort device IP" sniff-target-port=37008
```

Figure 5.5 - MikroTik firewall rules for traffic sniffing towards Snort

For Snort and the Linux machine to be able to intercept this traffic as a sniffer, the tool Trafr is needed. Trafr is MikroTik’s program that is able to receive mirrored traffic from a RouterOS machine. A firewall rule needs to be added to the Linux machine as well, to accept the incoming traffic in the specific port.

```
wget http://www.mikrotik.com/download/trafr.tgz

tar -zvxf trafr.tgz

sudo apt-get install libc6-i386 (Ubuntu specific)

iptables -L --line-numbers

iptables -I INPUT 13 -p udp --dport 37008 -j ACCEPT -m comment --comment "Accept Sniffed traffic
from RouterBoard"
```

Figure 5.6 - Linux machine setup for Snort

The final step is to test the tools and run them at start-up. Because of trafr's and snort's nature, it is not possible to be set as a daemon at start-up, so the workaround is to set it up as a screen, which then can be set as a start-up script. The utility "screen" is a terminal multiplexer, allowing to start sessions of any number of windows/virtual terminals inside the same session and processes in "screen" keep running even if the window is disconnected. With the following flags we can start a named screen in detached mode, where in simpler terms, it means that a process is created and started running on the background.

```
./trafr -s | tcpdump -r - -n

./trafr -s | snort -r –

cp trafr /usr/local/bin/


sudo apt-get install screen

screen -dmS mytrafr /usr/local/bin/trafr
```

Figure 5.7 - Trafr & Snort test / Screen set-up

5.3 Detection and Prevention

Snort is setup and running, but what does it actually do? It analyzes traffic based on rules and produces alerts. To test the capabilities of Snort in an IoT network 3 datasets were used.

5.3.1 Datasets

The first dataset originates from an open-source RPL attacks framework, developed on ContikiOS. Simulations and deployment of malicious notes set across a Wireless Sensor Network using RPL as its network layer. A lot of configuration can be made but for the purpose of this paper the default settings were used to generate simple and malicious traffic. [12]

The attacks that are included are flooding attack, versioning attack and blackhole attack. Flooding attacks in RPL are about "HELLO" messages sent to nodes within range. This is the initial message that a node must send in order to join the topology. A broadcasted message is created from the attacker's node, where they simply present themselves to their neighbors. This can be a simple overhead attack in terms of packets, eating through the constraint resources of a few devices, or when using a device with stronger signal than the surrounding "neighbors", then the topology is in a state of confusion as their responses simply cannot reach the malicious node. [29]

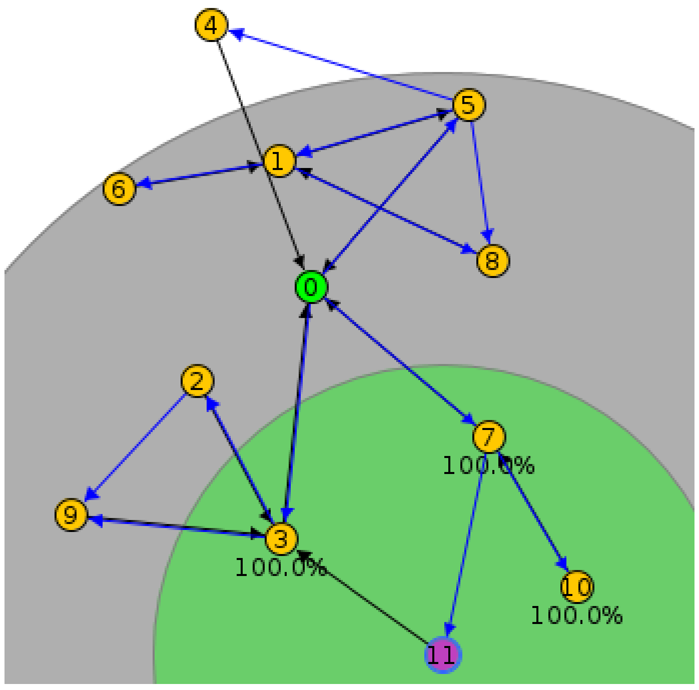


Figure 5.8 – Flooding attack [12]

In order to explain the versioning attack, a better understanding of RPL is needed. In order to have a hierarchical structure of only one root, loop-free topologies are formed with the term Destination Oriented Directed Acyclic Graphs – DODAG. There are certain parameters that can define how the topology may become optimized through objective functions and multiple instances of DODAGs can exist, but nodes can only participate in only one at the same time frame. Several control messages exist and are used and broadcasted during the lifetime of the topology, where creating a stable DODAG results in less transmissions of said control messages. Due to the nature of low-power and lossy networks, nodes can disconnect from the topology due to battery or connection issues, thus requiring a repair function This is called “global repair” mechanism, where the DODAG is rebuilt in increments. Control messages are sent with the DODAG version number and each node validates their own version, if the received number is higher than the one they have, a new procedure is initiated to join the new DODAG. [26]

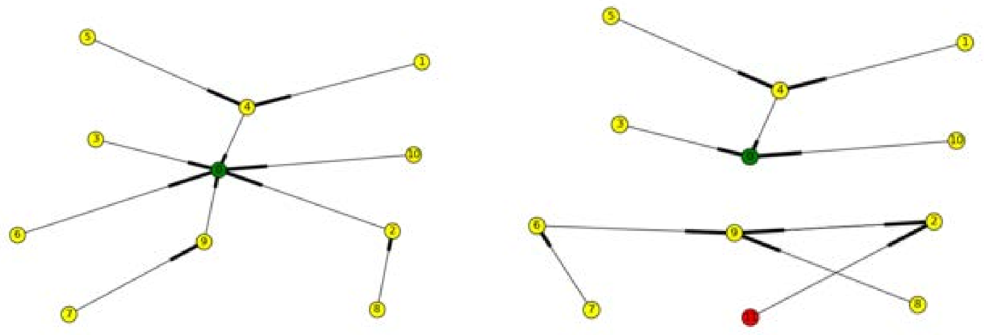


Figure 5.9 – Legitimate DODAG & Versioning attack – global repair [12]

Only the root node uses the versioning number for global repair so that the routing state of the topology is up-to-date. Control messages carry this number for out of date checks and updates, where if any node has an older version means they did not join in the new instance of DODAG and as such, it is perceived unfavorably by neighbor nodes in terms of choosing it as a parent of a certain part of the tree topology. To avoid inconsistencies whenever 2 versions of a DODAG coexist during repairs, it is considered that the version number transferred to be unchanged. However, there is no function in place to check for such cases and when an attacker’s node transmits different number of version in their control messages, causing all sorts of chaos and unpredicted behavior in the propagation and repair of more than 2 DODAG versions that the topology is trying to converge from the previous to the newer one. Manipulation of the version number causes rebuilds and loops into the topology, negatively impacting routing of traffic, channel usage and a lot of unnecessary energy consumption. [7]

Lastly, the blackhole attack is an iteration of the previously mention sinkhole attack. A malicious node advertises to their neighbors of a false or even truly best route towards their target destination (the root of the DODAG most likely). In case of a successful placement of several nodes under the attacker's node, it then proceeds to drop the packets received in a silent manner, ghosting a section of the network as far as its signal strength and possibly extending the damage towards children of its neighbor nodes. [3]

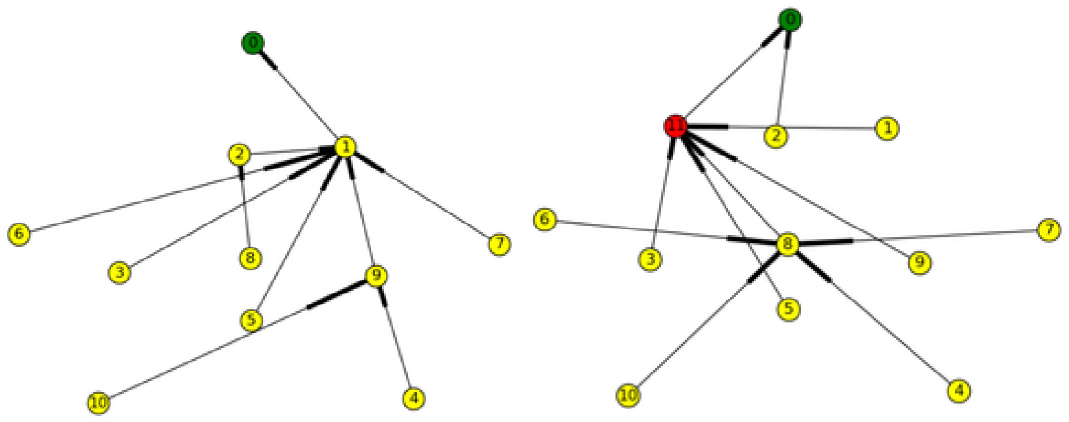
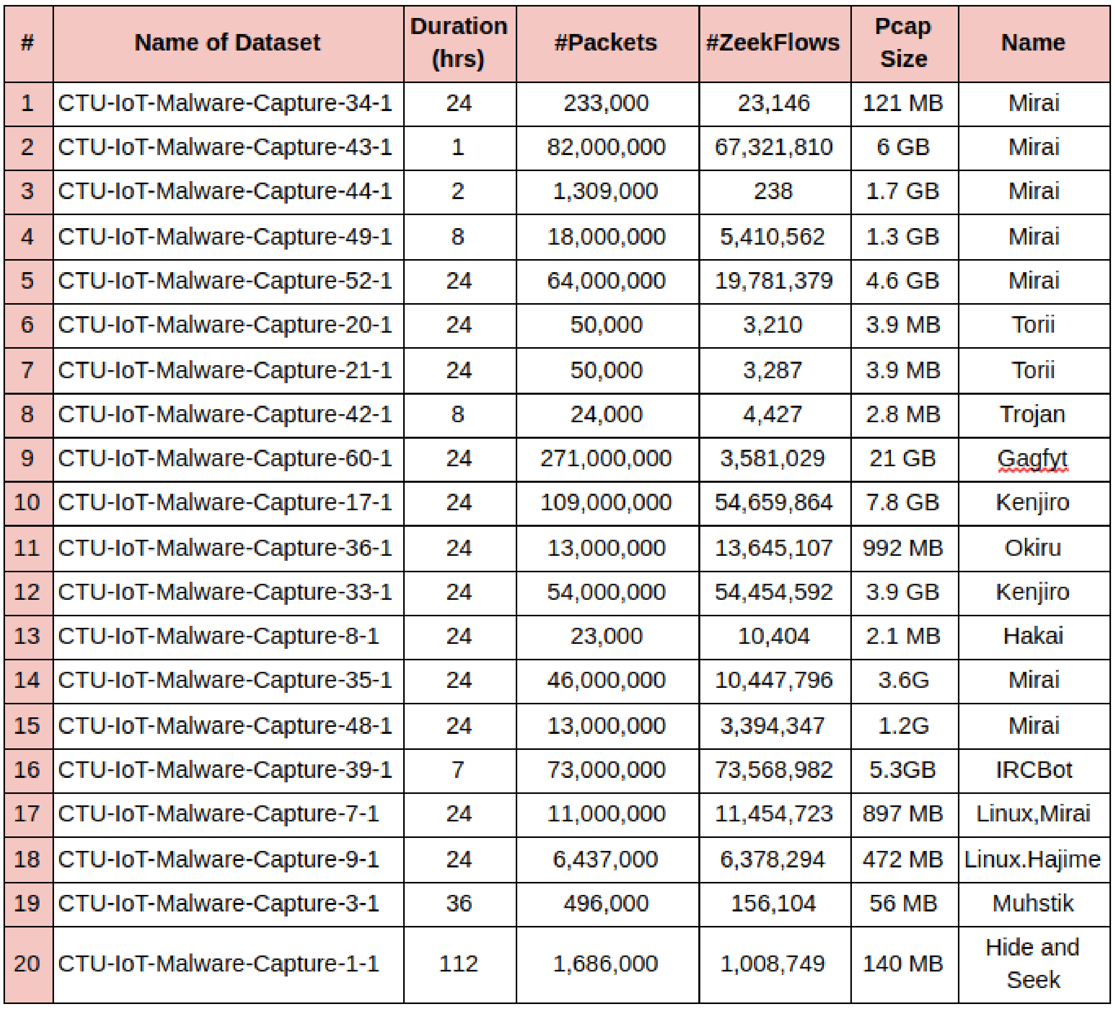


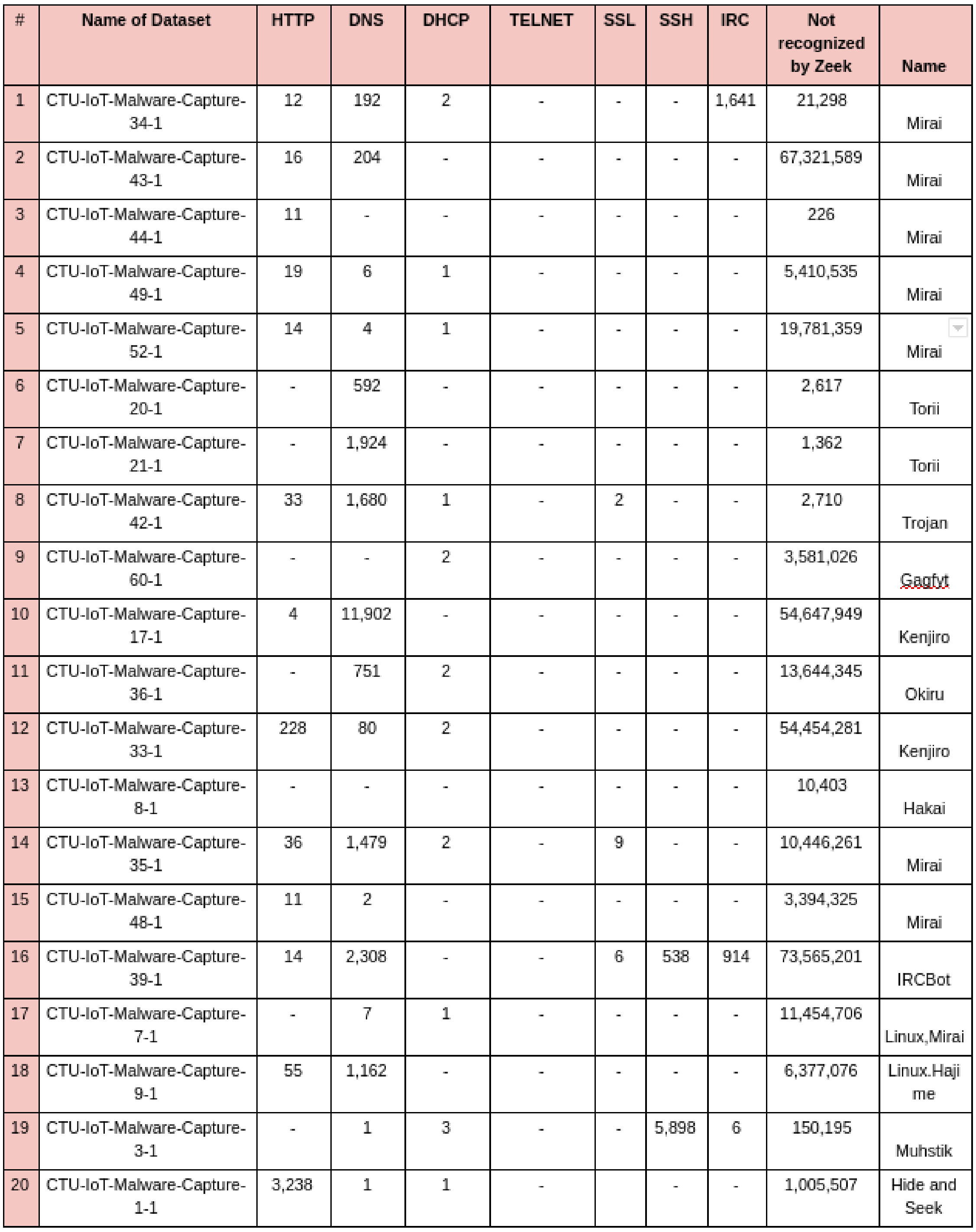
Figure 5.10 – Legitimate DODAG & Blackhole attack [12]

The second dataset that was used contains 20 malicious and 3 benign IoT traffic captures, created as part of the Avast AIC laboratory. The network was artificially created for the purpose of researchers developing machine learning algorithms and as such consists of scenarios divided into pcap files. Three devices were used: Amazon Echo home intelligent personal assistant, Somfy smart doorlock and Philips HUE smart LED lamp. The malware originated from a Raspberry Pi inside of the network and the devices run with internet access in a controlled network environment. Information about each file can be seen in the below figures. [30]



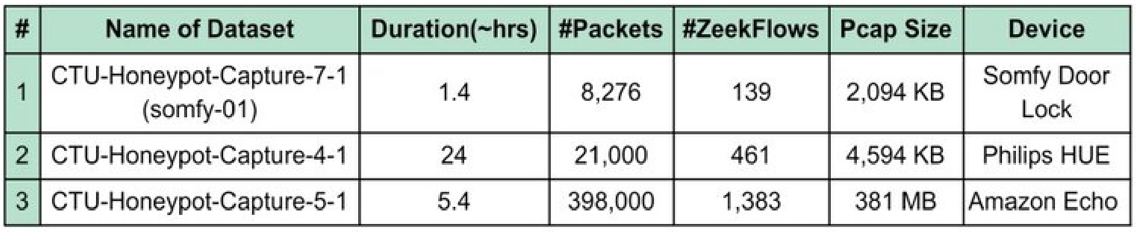
#	Name of Dataset	Duration (hrs)	#Packets	#ZeekFlows	Pcap Size	Name
1	CTU-IoT-Malware-Capture-34-1	24	233,000	23,146	121 MB	Mirai
2	CTU-IoT-Malware-Capture-43-1	1	82,000,000	67,321,810	6 GB	Mirai
3	CTU-IoT-Malware-Capture-44-1	2	1,309,000	238	1.7 GB	Mirai
4	CTU-IoT-Malware-Capture-49-1	8	18,000,000	5,410,562	1.3 GB	Mirai
5	CTU-IoT-Malware-Capture-52-1	24	64,000,000	19,781,379	4.6 GB	Mirai
6	CTU-IoT-Malware-Capture-20-1	24	50,000	3,210	3.9 MB	Torii
7	CTU-IoT-Malware-Capture-21-1	24	50,000	3,287	3.9 MB	Torii
8	CTU-IoT-Malware-Capture-42-1	8	24,000	4,427	2.8 MB	Trojan
9	CTU-IoT-Malware-Capture-60-1	24	271,000,000	3,581,029	21 GB	Gagfyt
10	CTU-IoT-Malware-Capture-17-1	24	109,000,000	54,659,864	7.8 GB	Kenjiro
11	CTU-IoT-Malware-Capture-36-1	24	13,000,000	13,645,107	992 MB	Okiru
12	CTU-IoT-Malware-Capture-33-1	24	54,000,000	54,454,592	3.9 GB	Kenjiro
13	CTU-IoT-Malware-Capture-8-1	24	23,000	10,404	2.1 MB	Hakai
14	CTU-IoT-Malware-Capture-35-1	24	46,000,000	10,447,796	3.6G	Mirai
15	CTU-IoT-Malware-Capture-48-1	24	13,000,000	3,394,347	1.2G	Mirai
16	CTU-IoT-Malware-Capture-39-1	7	73,000,000	73,568,982	5.3GB	IRCBot
17	CTU-IoT-Malware-Capture-7-1	24	11,000,000	11,454,723	897 MB	Linux,Mirai
18	CTU-IoT-Malware-Capture-9-1	24	6,437,000	6,378,294	472 MB	Linux.Hajime
19	CTU-IoT-Malware-Capture-3-1	36	496,000	156,104	56 MB	Muhstik
20	CTU-IoT-Malware-Capture-1-1	112	1,686,000	1,008,749	140 MB	Hide and Seek

Figure 5.11 – Summary of Malicious scenarios [30]



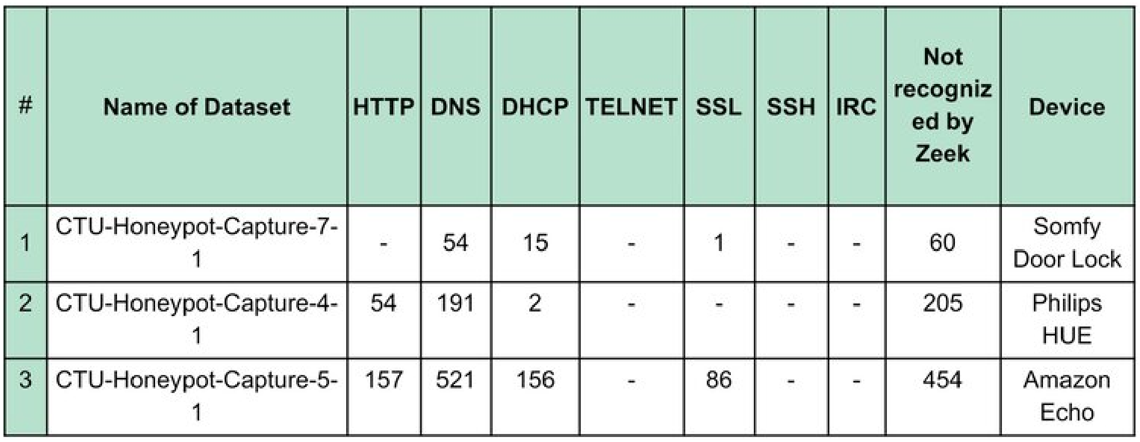
#	Name of Dataset	HTTP	DNS	DHCP	TELNET	SSL	SSH	IRC	Not recognized by Zeek	Name
1	CTU-IoT-Malware-Capture-34-1	12	192	2	-	-	-	1,641	21,298	Mirai
2	CTU-IoT-Malware-Capture-43-1	16	204	-	-	-	-	-	67,321,589	Mirai
3	CTU-IoT-Malware-Capture-44-1	11	-	-	-	-	-	-	226	Mirai
4	CTU-IoT-Malware-Capture-49-1	19	6	1	-	-	-	-	5,410,535	Mirai
5	CTU-IoT-Malware-Capture-52-1	14	4	1	-	-	-	-	19,781,359	Mirai
6	CTU-IoT-Malware-Capture-20-1	-	592	-	-	-	-	-	2,617	Torii
7	CTU-IoT-Malware-Capture-21-1	-	1,924	-	-	-	-	-	1,362	Torii
8	CTU-IoT-Malware-Capture-42-1	33	1,680	1	-	2	-	-	2,710	Trojan
9	CTU-IoT-Malware-Capture-60-1	-	-	2	-	-	-	-	3,581,026	Gagfyt
10	CTU-IoT-Malware-Capture-17-1	4	11,902	-	-	-	-	-	54,647,949	Kenjiro
11	CTU-IoT-Malware-Capture-36-1	-	751	2	-	-	-	-	13,644,345	Okiru
12	CTU-IoT-Malware-Capture-33-1	228	80	2	-	-	-	-	54,454,281	Kenjiro
13	CTU-IoT-Malware-Capture-8-1	-	-	-	-	-	-	-	10,403	Hakai
14	CTU-IoT-Malware-Capture-35-1	36	1,479	2	-	9	-	-	10,446,261	Mirai
15	CTU-IoT-Malware-Capture-48-1	11	2	-	-	-	-	-	3,394,325	Mirai
16	CTU-IoT-Malware-Capture-39-1	14	2,308	-	-	6	538	914	73,565,201	IRCBot
17	CTU-IoT-Malware-Capture-7-1	-	7	1	-	-	-	-	11,454,706	Linux,Mirai
18	CTU-IoT-Malware-Capture-9-1	55	1,162	-	-	-	-	-	6,377,076	Linux.Hajime
19	CTU-IoT-Malware-Capture-3-1	-	1	3	-	-	5,898	6	150,195	Muhstik
20	CTU-IoT-Malware-Capture-1-1	3,238	1	1	-		-	-	1,005,507	Hide and Seek

Figure 5.12 – Application Layer Protocols on Malicious scenarios [30]



#	Name of Dataset	Duration(~hrs)	#Packets	#ZeekFlows	Pcap Size	Device
1	CTU-Honeypot-Capture-7-1 (somfy-01)	1.4	8,276	139	2,094 KB	Somfy Door Lock
2	CTU-Honeypot-Capture-4-1	24	21,000	461	4,594 KB	Philips HUE
3	CTU-Honeypot-Capture-5-1	5.4	398,000	1,383	381 MB	Amazon Echo

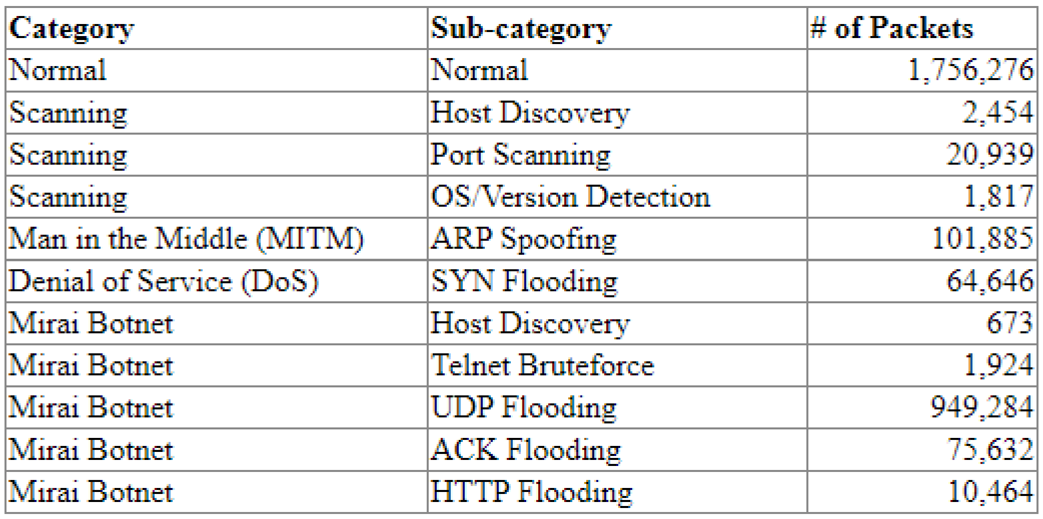
Figure 5.13 – Summary of Benign scenarios [30]



#	Name of Dataset	HTTP	DNS	DHCP	TELNET	SSL	SSH	IRC	Not recognized by Zeek	Device
1	CTU-Honeypot-Capture-7-1	-	54	15	-	1	-	-	60	Somfy Door Lock
2	CTU-Honeypot-Capture-4-1	54	191	2	-	-	-	-	205	Philips HUE
3	CTU-Honeypot-Capture-5-1	157	521	156	-	86	-	-	454	Amazon Echo

Figure 5.14 – Application Layer Protocols on Benign scenarios [30]

The third dataset is similar to the previous one, where a network of two smart home devices was setup with SKT NUGU (NU 100), an AI voice recognition speaker and a EZVIZ (C2C Mini O Plus 1080P) Wi-Fi camera. The respective files and attacks can bee seen in Figure 5.15. [23]



Category	Sub-category	# of Packets
Normal	Normal	1,756,276
Scanning	Host Discovery	2,454
Scanning	Port Scanning	20,939
Scanning	OS/Version Detection	1,817
Man in the Middle (MITM)	ARP Spoofing	101,885
Denial of Service (DoS)	SYN Flooding	64,646
Mirai Botnet	Host Discovery	673
Mirai Botnet	Telnet Bruteforce	1,924
Mirai Botnet	UDP Flooding	949,284
Mirai Botnet	ACK Flooding	75,632
Mirai Botnet	HTTP Flooding	10,464

Figure 5.15 – Summary of Dataset [22]

5.3.2 Action Taken

Having all these pcap files, a command like the one below can be run for Snort to analyze.

```
snort -c /usr/local/etc/snort/snort.lua -r
/media/ubuntu/Elements/Diplomatiki/iot_23_datasets_full/opt/Malware-
Project/BigDataset/IoTScenarios/CTU-IoT-Malware-Capture-34-1/2018-12-21-15-50-14-
192.168.1.195.pcap -A alert_fast -s 65535 -k none > ~/Desktop/test.txt
```

Figure 5.16 - Snort single file analysis

In this case, the result would be a file with all the relevant information that is being tracked along with the alerts. Alerts are classified with priorities, starting from 0 to N, 0 being the most severe.

```
Application Stats

1549218716,DNS,1576,1778

1549218716,Facebook,8727,630912

1549218716,OpenSSH,19523,2344

1549218716,SSH,19523,23445

1549218716,HTTPS,8727,630912

1549218716,SSL client,8727,630912

1549218716,ICMP,392,392

1549218716,__unknown,11028,139100

03/16-16:06:43.480000 [**] [125:3:1] "FTPP_FTP_PARAMETER_LENGTH_OVERFLOW" [**]
[Classification: Attempted Administrator Privilege Gain] [Priority: 1] [AppID: FTP] {TCP}
192.168.202.96:47739 -> 192.168.28.101:21
```

Figure 5.17 - Snort sample output

Taking the priorities into consideration and assuming that priorities 0 and 1 are most likely breaches of our network security, a script was created to find those alerts, take the incoming addresses and block them on the router. A command like the one shown in Figure 5.18, prints the incoming address of the alert with the respective priority of all the files in the current directory.

```
grep "Priority: 0" * | sed 's#\(.* \)\([0-9]\+\.[0-9]\+\.[0-9]\+\.[0-9]\+\)\(:[0-9]\+ ->.*\)#\2#g'
```

Figure 5.18 - Command to match incoming address of an alert

Outputting the addresses into the file and redirecting that to a command, prepares the appropriate script to be run on the router.

```
$ printf "address1\naddress2\n" | \
awk '{print "ip firewall address-list add list=block_list address=" $0}' > ./block_addresses.auto.rsc
ip firewall address-list add list=block_list address=address1
ip firewall address-list add list=block_list address=address2
```

Figure 5.19 - Creation of router script to block addresses

The file generated can now be uploaded through FTP to the router and because of the ".auto.rsc" extension, the script will be automatically executed upon upload. In order for the address list to actually get blocked, a separate firewall rule has to be made to explicitly drop traffic from said address list.

```
add action=drop chain=forward comment=\
   "Drop traffic from internet and in block list" in-interface=ether1 \
   log=yes log-prefix=!public src-address-list=block_list
add action=drop chain=forward comment=\
   "Drop traffic from LAN and in block list" dst-address-list=\
   block_list in-interface=bridge log=yes log-prefix=!public_from_LAN \
   out-interface=!bridge
```

Figure 5.20 - Firewall rules for dropping traffic from addresses in block list

It is worth noting that Snort has a majority of its rules commented out to reduce false positives. In research environments such as this, they are better turned on. Using just the default rules the results were pretty devastating at a 0% rate of any alert with priority 1 or 0. But, with all the rules, from the 2nd dataset of 23 files, 3 out of which were benign, 11 files were detected to have alerts of at least priority 1, but one of which was a benign pcap file. From the 3rd dataset of 42 files, 1 out of which was benign, 21 files were found. Unfortunately, none of the RPL attacks were detected and keeping in mind that the rules used may very well result in false positives.

```
ubuntu@ubuntu1804:/diplomatiki/logs/run 2/23$ grep -Ril "Priority: 1" *.txt
104.248.160.24-80.pcap.txt
104.248.160.24.port80TCP.pcap.txt
2018-07-25-10-53-16-192.168.100.111.pcap.txt
2018-09-14-13-40-25-Philips-Hue-Bridge.pcap.txt
2018-12-21-15-33-59-192.168.1.196.pcap.txt
2018-12-21-15-50-14-192.168.1.195.pcap.txt
2019-01-10-19-22-51-192.168.1.198.pcap.txt
2019-01-10-21-06-26-192.168.1.199.pcap.txt
2019-02-28-19-15-13-192.168.1.200.pcap.txt
2019-02-28-20-50-15-192.168.1.193.pcap.txt
2019-03-08-13-24-30-192.168.1.197.pcap.txt
```

Figure 5.21 – All Snort rules, 2nd dataset

```
ubuntu@ubuntu1804:/diplomatiki/logs/run 2/other$ grep -Ril "Priority: 1" *.txt
mirai-ackflooding-1-dec.pcap.txt
mirai-ackflooding-2-dec.pcap.txt
mirai-ackflooding-3-dec.pcap.txt
mirai-ackflooding-4-dec.pcap.txt
mirai-hostbruteforce-1-dec.pcap.txt
mirai-hostbruteforce-2-dec.pcap.txt
mirai-hostbruteforce-3-dec.pcap.txt
mirai-hostbruteforce-4-dec.pcap.txt
mirai-hostbruteforce-5-dec.pcap.txt
mirai-httpflooding-1-dec.pcap.txt
mirai-httpflooding-2-dec.pcap.txt
mirai-httpflooding-3-dec.pcap.txt
mirai-httpflooding-4-dec.pcap.txt
mirai-udpflooding-1-dec.pcap.txt
mirai-udpflooding-2-dec.pcap.txt
mirai-udpflooding-3-dec.pcap.txt
mirai-udpflooding-4-dec.pcap.txt
scan-portos-1-dec.pcap.txt
scan-portos-2-dec.pcap.txt
scan-portos-3-dec.pcap.txt
scan-portos-6-dec.pcap.txt
```

Figure 5.22 - All Snort rules, 3rd dataset

Chapter 6

Future Considerations and Challenges

General considerations to improve upon IoT IDS solutions entail encryption, deployment complications, scalability and management.

Intrusion detection is solely dependent on inspecting packet payloads and content matching, and as progress in the security of IoT is increasing, more applications make use of encryption and privacy mechanisms, rendering inspection useless. A key offload process is required, where decryption is done first and then the traffic is handled by the IDS. This introduces a lot more cost and complexity.

And as complexity is concerned, IoT deployment of such systems will vary case to case. The placement of an IDS will heavily depend on the location of the risk, if that can be pinpointed, based on a threat model of the deployment.

In the specific case of this paper, it can be further developed through different IDS systems like Suricata or Bro and even distributed solutions on Raspberry Pi’s. In the latter case, a more complex system will surely be needed, like an AI system, which in this case can co-exist with other IDS, like Snort, working in unison. A smarter and costlier system can focus on the grey traffic that a simpler IDS cannot find malicious intents.

Chapter 7

Conclusion

The market of the “Internet of Things” has been predicted to grow at a fast pace and has already been proven as such in the last few years. The development of new products and technology stacks is not halting, as to be able to meet the demand. Security implications arise from these rapid operations and the rush of production. Standards are greatly lacking, making this a more imminent problem as security is not within what is seen as fit in the manufacturing stages. IoT endpoint management and intrusion detection is tricky due to the lack of network support and constrained resources. Understanding and dissecting the underlying communication and technology in use is required by intrusion analysts to be able to pinpoint the location of risks. The correct deployment of intrusion detection is heavily dependent on such research, as the end goal is to correctly identify and defend the monitored assets.

Bibliography

[1] adam, “Threat Modeling & IoT – Adam Shostack & friends.” https://adam.shostack.org/blog/2017/05/threat-modeling-iot/ (accessed Jun. 11, 2020).

[2] ADUK GmbH, “Enterprises are leaving IoT devices vulnerable to cybersecurity threats, finds nCipher Security,” ADUK GmbH, Oct. 09, 2019. https://aduk.de/industry-news/enterprises-are-leaving-iot-devices-vulnerable-to-cybersecurity-threats-finds-ncipher-security/ (accessed Jun. 11, 2020).

[3] F. Ahmed and Y.-B. Ko, “Mitigation of black hole attacks in Routing Protocol for Low Power and Lossy Networks: Mitigation of black hole attacks in RPL,” Security Comm. Networks, vol. 9, no. 18, pp. 5143–5154, Dec. 2016, doi: 10.1002/sec.1684.

[4] Akiba, “IEEE 802.15.4 in the Context of Zigbee - Part 2 - MAC Layer.” https://archive.freaklabs.org/index.php/blog/zigbee/ieee-802154-in-the-context-of-zigbee-part-2-mac-layer.html (accessed Jun. 11, 2020).

[5] Anna-senpai, “[FREE] World’s Largest Net:Mirai Botnet, Client, Echo Loader, CNC source code release,” Hack Forums, Sep. 30, 2016. https://hackforums.net/showthread.php?tid=5420472 (accessed Jun. 11, 2020).

[6] ANTONIUS DUTY SUSILO, “IDS Implementation with Mikrotik.” https://mum.mikrotik.com/presentations/VN17/presentation_4172_1494480243.pdf (accessed Jun. 10, 2020).

[7] A. Aris, S. F. Oktug, and S. Berna Ors Yalcin, “RPL version number attacks: In-depth study,” in NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium, Istanbul, Turkey, Apr. 2016, pp. 776–779, doi: 10.1109/NOMS.2016.7502897.

[8] Banafa Ahmed, “Fog Computing is Vital for a Successful Internet of Things,” Datafloq. https://datafloq.com/read/fog-computing-vital-successful-internet-of-things/1166 (accessed Jun. 11, 2020).

[9] C. J. Barker, “Mirai (DDoS) Source Code Review,” Medium, Oct. 13, 2016. https://medium.com/@cjbarker/mirai-ddos-source-code-review-57269c4a68f (accessed Jun. 11, 2020).

[10] bastille.net, “bastille_security_white_paper.pdf.” https://static1.squarespace.com/static/57320daef699bbe627689544/t/58042ddf6a49634b9ebecbfb/1476668897632/bastille_security_white_paper.pdf (accessed Jun. 11, 2020).

[11] C. Cervantes, D. Poplade, M. Nogueira, and A. Santos, “Detection of sinkhole attacks for supporting secure routing on 6LoWPAN for Internet of Things,” in 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), May 2015, pp. 606–611, doi: 10.1109/INM.2015.7140344.

[12] A. dhondta, dhondta/rpl-attacks. 2020.

[13] N. Dietrich, “Snort_3_on_Ubuntu.pdf,” Snort 3.0.1 on Ubuntu 18 & 20, Jul. 05, 2020. https://snort-org-site.s3.amazonaws.com/production/document_files/files/000/000/251/original/Snort_3_on_Ubuntu.pdf?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIXACIED2SPMSC7GA%2F20200609%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20200609T040100Z&X-Amz-Expires=172800&X-Amz-SignedHeaders=host&X-Amz-Signature=6feb04311de86f81ef082681d206b62f55d0151f97c6c945e2787d9cd248fdb5 (accessed Jun. 09, 2020).

[14] Fazeldehkordi Elahe, Sadegh Iraj, Oluwatobi Amiri, and Akanbi Ayodeji, “A Study of Black Hole Attack Solutions - 1st Edition.” https://www.elsevier.com/books/T/A/9780128053676 (accessed Jun. 11, 2020).

[15] Green Jim, “IoT_Reference_Model_White_Paper_June_4_2014.pdf.” http://cdn.iotwf.com/resources/71/IoT_Reference_Model_White_Paper_June_4_2014.pdf (accessed Jun. 11, 2020).

[16] Gupta Anil, “(6) What is a good approach for designing an IOT architecture? - Quora.” https://www.quora.com/What-is-a-good-approach-for-designing-an-IOT-architecture (accessed Jun. 11, 2020).

[17] Hayes Eric, “The Underwhelming Security of the Internet of Things (IoT),” Jacobian Engineering, Feb. 21, 2017. https://jacobianengineering.com/blog/2017/02/underwhelming-security-internet-things/ (accessed Jun. 11, 2020).

[18] J. Horst, “Raspberry, turn the light on! - Voice assistant used to manage Philips Hue light bulb.” http://www.devjhorst.com/2018/02/raspberry-turn-the-light-on-voice-assistant.html (accessed Jun. 11, 2020).

[19] hvrsupreet, “IoT Architecture Basics (01),” Realm, May 07, 2017. https://supreethvr.wordpress.com/2017/05/07/iot-architecture-basics-01/ (accessed Jun. 11, 2020).

[20] R. IDIL, “A Brief Overview of Bluetooth Low Energy,” Medium, Oct. 23, 2018. https://medium.com/rtone-iot-security/a-brief-overview-of-bluetooth-low-energy-79be06eab4df (accessed Jun. 11, 2020).

[21] JIMBLOM, “RETIRED - Exploring XBees and XCTU - learn.sparkfun.com.” https://learn.sparkfun.com/tutorials/retired---exploring-xbees-and-xctu/selecting-an-explorer (accessed Jun. 11, 2020).

[22] Kang Hyunjae, “IoT Network Intrusion Dataset - Hacking and Countermeasure Research Lab.” http://ocslab.hksecurity.net/Datasets/iot-network-intrusion-dataset (accessed Jun. 10, 2020).

[23] H. Kang, D. H. Ahn, G. M. Lee, J. D. Yoo, K. H. Park, and H. K. Kim, “IoT network intrusion dataset.” IEEE DataPort, Sep. 27, 2019, doi: 10.21227/Q70P-Q449.

[24] A. Kliarsky, “Detecting Attacks Against The Internet of Things,” p. 36, 2017.

[25] J. Lindh, “Bluetooth® low energy Beacons,” p. 15, 2015.

[26] A. Mayzaud, A. Sehgal, R. Badonnel, I. Chrisment, and J. Schönwälder, “A Study of RPL DODAG Version Attacks,” in Monitoring and Securing Virtualized Networks and Services, vol. 8508, A. Sperotto, G. Doyen, S. Latré, M. Charalambides, and B. Stiller, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 92–104.

[27] Microchip Technology, Inc., “Bluetooth® Low Energy Packet Types - Developer Help.” https://microchipdeveloper.com/wireless:ble-link-layer-packet-types (accessed Jun. 11, 2020).

[28] M. Nawir, A. Amir, N. Yaakob, and O. B. Lynn, “Internet of Things (IoT): Taxonomy of security attacks,” in 2016 3rd International Conference on Electronic Design (ICED), Aug. 2016, pp. 321–326, doi: 10.1109/ICED.2016.7804660.

[29] T. Nguyen, T. Ngo, T. Nguyen, D. Tran, H. A. Tran, and T. Bui, “The Flooding Attack in Low Power and Lossy Networks: A Case Study,” in 2018 International Conference on Smart Communications in Network Technologies (SaCoNeT), El Oued, Oct. 2018, pp. 183–187, doi: 10.1109/SaCoNeT.2018.8585451.

[30] Parmisano Agustin, Garcia Sebastian, and Jose Erquiaga Maria, “IoT-23 Dataset: A labeled dataset of Malware and Benign IoT Traffic.,” Stratosphere IPS. https://www.stratosphereips.org/datasets-iot23 (accessed Jun. 10, 2020).

[31] pcwrt, “Beyond Three Dumb Routers,” pcWRT, Jun. 20, 2018. https://www.pcwrt.com/2018/06/beyond-three-dumb-routers/ (accessed Jun. 11, 2020).

[32] P. Pongle and G. Chavan, “Real Time Intrusion and Wormhole Attack Detection in Internet of Things,” IJCA, vol. 121, no. 9, pp. 1–9, Jul. 2015, doi: 10.5120/21565-4589.

[33] robinsh, “IoT Security Architecture.” https://docs.microsoft.com/en-us/azure/iot-fundamentals/iot-security-architecture (accessed Jun. 11, 2020).

[34] K. Rosenfeld and R. Karri, “Attacks and Defenses for JTAG,” IEEE Des. Test. Comput., vol. 27, no. 1, pp. 36–47, Jan. 2010, doi: 10.1109/MDT.2010.9.

[35] Rouse Margaret, “What is Zigbee? - Definition from WhatIs.com,” IoT Agenda. https://internetofthingsagenda.techtarget.com/definition/ZigBee (accessed Jun. 11, 2020).

[36] R. L. Security, riverloopsec/beekeeperwids. 2020.

[37] R. L. Security, riverloopsec/apimote. 2020.

[38] P. Sethi and S. R. Sarangi, “Internet of Things: Architectures, Protocols, and Applications,” Journal of Electrical and Computer Engineering, vol. 2017, pp. 1–25, 2017, doi: 10.1155/2017/9324035.

[39] A. Sforzin, F. G. Marmol, M. Conti, and J.-M. Bohli, “RPiDS: Raspberry Pi IDS — A Fruitful Intrusion Detection System for IoT,” in 2016 Intl IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld), Toulouse, Jul. 2016, pp. 440–448, doi: 10.1109/UIC-ATC-ScalCom-CBDCom-IoP-SmartWorld.2016.0080.

[40] SIA Mikrotīkls, “Manual:RouterOS features - MikroTik Wiki.” https://wiki.mikrotik.com/wiki/Manual:RouterOS_features#Tools (accessed Jun. 11, 2020).

[41] SIA Mikrotīkls, “MikroTik.” https://mikrotik.com/ (accessed Jun. 11, 2020).

[42] Snort, “Snort - Network Intrusion Detection & Prevention System.” https://www.snort.org/ (accessed Jun. 11, 2020).

[43] M. Szczys, “Shmoocon 2016: Z-Wave Protocol Hacked With SDR,” Hackaday, Jan. 16, 2016. https://hackaday.com/2016/01/16/shmoocon-2016-z-wave-protocol-hacked-with-sdr/ (accessed Jun. 11, 2020).

[44] Townsend Kevin, “Introducing the Adafruit Bluefruit LE Friend,” Adafruit Learning System. https://learn.adafruit.com/introducing-adafruit-ble-bluetooth-low-energy-friend/introduction (accessed Jun. 11, 2020).

[45] L. Wallgren, S. Raza, and T. Voigt, “Routing Attacks and Countermeasures in the RPL-Based Internet of Things,” International Journal of Distributed Sensor Networks, vol. 9, no. 8, p. 794326, Aug. 2013, doi: 10.1155/2013/794326.

[46] Wright Joshua, Speers Ryan, and Melgares Ricky, “KillerBee.” https://tools.kali.org/wireless-attacks/killerbee (accessed Jun. 11, 2020).

[47] ZigBee Alliance, “docs-05-3474-21-0csg-zigbee-specification.pdf.” https://zigbeealliance.org/wp-content/uploads/2019/11/docs-05-3474-21-0csg-zigbee-specification.pdf (accessed Jun. 11, 2020).

[48] Zillner Tobias and Strobl Sebastian, “us-15-Zillner-ZigBee-Exploited-The-Good-The-Bad-And-The-Ugly.pdf.” https://www.blackhat.com/docs/us-15/materials/us-15-Zillner-ZigBee-Exploited-The-Good-The-Bad-And-The-Ugly.pdf (accessed Jun. 11, 2020).