Monitoring Network Traffic and Responding to Malicious Traffic for IoT Devices in 5G Network Slices

Martijn N.F. de Redelijkheid
m.n.f.deredelijkheid@student.utwente.nl
M.Sc. Computer Science
March 14th 2023

Supervisors:
Dr. ir. Andrea Continella
a.continella@utwente.nl
Msc. Chakshu Gupta
c.gupta@utwente.nl
Ir. Frank Fransen
frank.fransen@tno.nl
Msc. Luca Morgese
luca.morgese@tno.nl

Telecommunication Engineering Group
Faculty of Electrical Engineering,
Mathematics and Computer Science
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

Contents

Chapter 1

Introduction

The Internet of Things (IoT) has been an ever growing presence in our daily lives. New devices and products are created almost on a daily basis, and the world of IoT is not showing any signs of stopping its immense growth anytime soon. IoT devices are also being introduced into many new domains, from automotive to industrial to the healthcare domain. These different domains present new challenges and goals for IoT to satisfy. For example, privacy is especially important in the healthcare domain and availability is especially important in the automotive and industrial domains. Problematically, many devices in the IoT ecosystem suffer from subpar security due to several limitations present in such devices. IoT devices are often not designed with security in mind and they often have low computational power, which inhibits the use of host-based features that add security, as well as the use of costly encryption techniques [1]–[5]. Additionally, they often do not have robust updating systems in place [6]. As such, IoT devices have become an interesting target for malicious actors. The Mirai botnet attacks in 2016 are a good example of the impact these devices can have when many of them are compromised. The Mirai botnet attacks were a series of distributed denial of service (DDoS) attacks performed by an enormous botnet of IoT devices which took down several popular websites [7]. These incidents also indicate that it is imperative to increase the security of the IoT ecosystem to prevent more large-scale attacks and malicious events.

There has been a lot of research to improve the security of IoT devices using various approaches to detect and react to threats. One of the most interesting options is the use of network based traffic monitoring. Such an approach does not require changes to the software running on the IoT devices, and can be administered by the network operator. Many previous works present methods for performing threat detection in this way for home and enterprise networks [4], [6], [8]–[19]. There have also been several works investigating the possibilities for securing IoT devices connected to mobile networks i.e., fourth generation (4G),fifth generation (5G) and similar networks [2], [20]–[26]. However, there are no widely adopted solutions that provide IoT threat detection in mobile networks. As such, IoT devices connected in this manner are not well protected.

In the context of mobile networking standards we have now arrived at the starting point of widespread 5G implementation. This generation of mobile networks has brought many new technologies and possibilities, such as network slicing and higher bandwidth capabilities. 5G is envisioned to use these technologies to support many different use cases. One of the main new visions for 5G is that it will allow for many new applications for IoT devices, as network speeds and new technologies are geared towards enabling these additional uses for IoT. Examples include the automotive industry and healthcare domain. 5G supports these use cases better than previous mobile networks because it is much faster, enables lower latency, and generally supports more devices connected at the same time. In addition, 5G has three modes of operation, one of which is massive machine-type communication (MMTC). MMTC uses various Low-power wide-area network (LPWAN) technologies, including Narrowband IoT (NB-IoT). These technologies are ideal for low-powered devices. MMTC is also focused on intermittent transmissions of small sizes, making it ideal for smart sensors. Another of the modes of operation of 5G is ultra-reliable low latency communication (URLLC), which aims to provide extremely low latency communication. This would be ideal for various IoT devices to support emergency services, smart vehicles, and (industrial) robotics [27]. The adoption of 5G has been accelerating in the recent years, and it is expected to keep increasing in the coming years. Currently, there are over 500 million active 5G subscriptions. By 2027, it is expected that there will be roughly 4.4 billion 5G subscriptions. More than 180 service providers have launched commercial 5G services, mainly in Asia and North America [28].

While the spread of 5G becomes ubiquitous, it is important to investigate what it brings. 5G provides many new possibilities in network management and allows for providing connectivity in different ways. One of the most interesting additions to the 5G architecture is network slicing. Network slicing is a technology that allows for creating so-called network slices. These network slices are isolated logical networks that run on generic networking hardware, which may also be shared between multiple slices. The idea of these slices is that they may be used to provide different properties required by different customers. These properties include latency, bandwidth, geographic coverage, and more. Network slicing also allows for interesting new additions to the security arsenal available for securing networks against different attacks, such as different options for traffic monitoring and responding to attacks by redirecting traffic flows [29].

Network slicing provides an opportunity to the mobile network operators to logically group (IoT) devices in their mobile networks by creating network slices reserved for (different types of) (IoT) devices. An IoT manufacturer or operator might request this from a mobile network operator and supply them with the unique identifiers of the concerned devices. This grouping helps traffic monitoring efforts as it prevents ”contamination” from other types of devices (e.g., mobile phones, laptops, etc.). This also allows the traffic monitoring system to be tailored to the types of IoT devices present in the network slice. Several previous works proposed this idea and investigated the implementation of several security functions in this context (see section 3.1). However, these works have several relevant limitations. Several of them do not contain any implementation using existing standards and network components [22], [23]. Others only focus on the performance of the system, and do not consider the security benefits [30]. Several other works only present frameworks and architectures for orchestrating such traffic monitoring, but do not provide implementation or research into effectiveness [24], [25], [31], [32]. Finally, these works are mostly heavily focused on detection and lack implementations and experiments with response actions that may be applied in scenarios where malicious activity is detected.

The current work aims to investigate one of the options for improving the security of a 5G network slice containing IoT devices from the point of view of the mobile network operator i.e., by monitoring traffic and automatically responding to detected malicious behaviour. We also aim to investigate the speed of a traffic monitoring solution where the traffic monitoring agent runs separately from the routing device and the speed of applying a specific response action (PDR/FAR change) automatically.

We use 5G components that comply with the 3rd Generation Partnership Project (3GPP) specifications to set up a testing infrastructure. On top of this testing infrastructure we implement a traffic monitoring system using Suricata [1]. We do this by adding a separate monitoring agent where Suricata runs as an intrusion detection system (IDS) as opposed to running Suricata as an intrusion prevention system (IPS) on the device that routes traffic. The efficiency of this approach is evaluated by simulating a set of attacks and measuring the detection for latency. In addition, we suggest several response actions available to the 5G network provider using components of the5G core network (see section 3.2.1). We implement one of these response actions (PDR/FAR change) in the detection infrastructure. The latency components present in such an automated response setup are also inves-

[1] https://suricata.io/

tigated by testing with a simulated attack. We find that the proposed system works and can detect attacks in a reasonable timeframe ($\sim$11,32 milliseconds on average, with slight variation depending on the attack). The automated response also works in a reasonable timeframe ($\sim$37,10 milliseconds between alert and isolation). Generalising these results, we find that the combined solution is only effective against attacks that take more than $\sim$50 milliseconds on average. Attacks that complete in a shorter timespan are not detected in time to apply an immediate response, but the device is isolated shortly after the attack ends.

Chapter 2 further explains some of the relevant concepts for the project, and goes further in depth on network slicing and IoT. Chapter 3 investigates existing solutions for network monitoring of IoT, both with regards to mobile networks and network slicing. Chapter 4 describes the theoretical approach taken for this work, while chapter 5 describes how this approach was implemented. Chapter 6 describes how the implementation is evaluated. Chapter 7 describes the results of testing the monitoring solution and the response action using the testbed implementation. Chapter 8 concludes the paper with a summary of results, limitations and suggestions for future work.

Chapter 2

Background

The sections below describe some general background relevant to the project. Section 2.1 describes some of the properties and trends of security in IoT devices. Section 2.2 describes the structure of 5G networks and further introduces network slicing, as well as some of the technologies that facilitate it. Section 2.3 describes what 5G network slicing can mean for the IoT domain.

2.1 Security of Internet of Things devices

There are several reasons for the subpar security of many IoT devices. Firstly, IoT devices often have low computational power as this makes them cheaper and easier to produce. This means that there is often not much room for adding security features. Such constrained IoT devices can not use heavy cryptographic protocols or authentication protocols, as there is simply not enough computational power available to run these protocols [1], [4], [6]. Host-based malware detection is usually not an option for IoT devices because of this lack of computational power [3], [5]. Besides these inherent issues, adding security features tends to be an afterthought in IoT development [2]. This is partially due to the fact that in the IoT domain developments are so rapid that companies must be fast with releasing new products in order to stay competitive. This also leads to a lack of (security) software updates, as it is difficult for companies to cheaply supply updates for a large catalogue of devices [3]. Manufacturer constraints also often impede the use of host-based malware detection. These factors cause IoT devices to generally be vulnerable to malware and attacks.

2.2 5G & Network Slicing

5G networks have two main components: the access technology and the core network. The access technology is the system that allows devices to connect to the network. There are several different options for this (see figure 2.1). The 5G Base Station (BS) is a radio base station and is often called a gNodeB (gNb). This is intended to become the most common access technology for mobile devices. Access technologies are local, as they rely on physical phenomena (e.g., radio waves) to connect to the devices. Devices or services that run close to the access technologies (in terms of latency) are often said to run in the ”edge”.

The core network receives communication from the access technology. This communication is split into two planes: the control plane and the user plane. The control plane handles all control functions, such as authenticating devices with the network and handing over connections to other broadband networks. One of the most important control plane functions is the access and mobility management function (AMF). It handles many of the authentication and registration functions, as well as various mobility tasks. It is also the function responsible for communication between the gNb and the control plane of the core network. The user plane is simply a channel that transfers user traffic it receives from the gNb to its end destination. It is embodied by one or more user plane functions (UPFs) which sends this traffic to any of the connected data networks. These may be the internet, an adjacent core network of a different provider, or various smaller target networks. The core network is much more centralized than the access technologies, although components may also be placed at the edge (especially the user plane). See figure 2.1 for a diagram depicting the various components of the 5G architecture and their interactions.

One of the most important technologies of the new 5G network is network slicing. Network slicing allows defining isolated end-to-end logical networks that run on generic hardware (which may be shared between multiple slices). Slices are customised based on the needs of the devices present in the slice. Slices can increase revenue for the providers, as they may more efficiently use the available network resources [29]. The clients procuring a slice are called tenants. They may run various applications (such as traffic monitoring) within their slice [34]. These applications may also be offered by the network provider as a service [35].

Network slicing is well developed for 5G core networks, and has been proven to work for radio access networks (RANs). However, it has not yet been widely implemented and requires further development for usage in RANs. [36], [37] Network

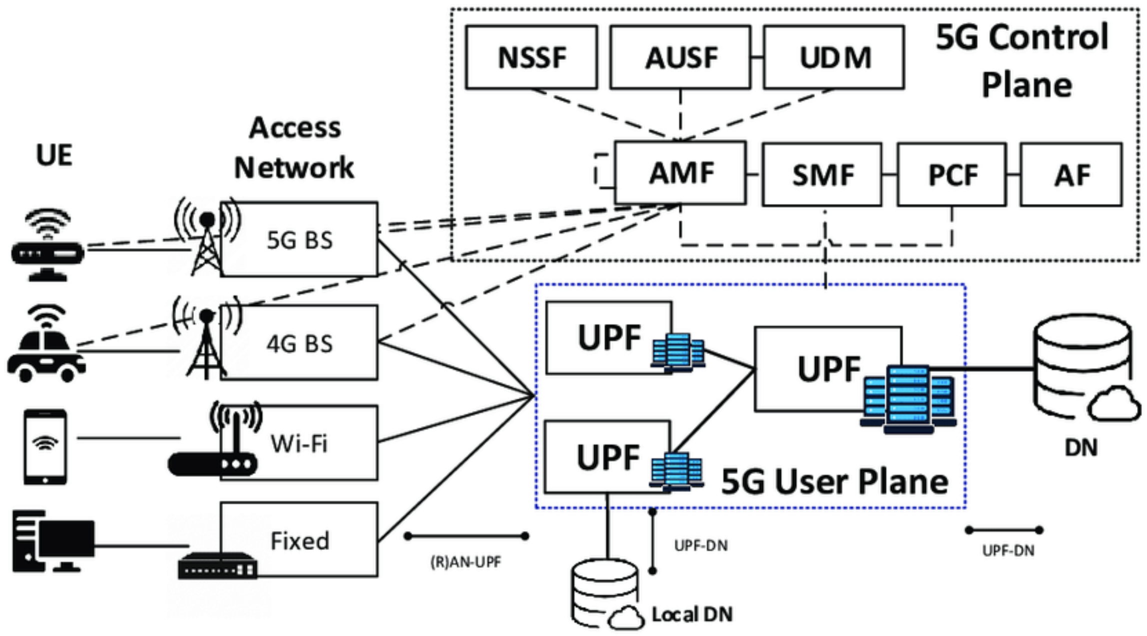

Figure 2.1: 5G architecture diagram picturing UEs, access technologies, and core network components [33].

slicing makes use of two essential paradigms: network function virtualization (NFV) and software-defined networking (SDN).

2.2.1 Network Function Virtualization

NFV aims at providing flexibility, agility, and scalability to the network. NFV basically entails running network functions on general-purpose hardware in a virtualized manner, instead of using specialized hardware. Examples of security functions that can be run in such a way are firewalls and packet inspection systems. The resulting functions are called virtualized network functions (VNFs). VNFs are then chained together to create complete network services [34], [38].

2.2.2 Software-Defined Networking

SDN aims at making connectivity programmable e.g., directing traffic flows to improve performance. The paradigm allows for managing network resources through an abstraction layer. SDN is implemented on traditional networking capabilities through the addition of hardware that allows for inserting software into the networking components i.e., white-box routers and switches. This kind of hardware is improving rapidly, with many devices available today [34], [38].

2.2.3 Security of 5G Network Slicing

It is important to consider the security issues affecting network slicing, as this new technology exposes new vulnerabilities and brings various security benefits.

Network slicing provides several benefits to security. Isolation between slices means that threats that may affect a specific slice can possibly be contained, preventing the compromise of devices in other slices. Network slicing may also allow for adding custom security functions to slices. These security functions can add security in different ways, from monitoring traffic to performing system hardening checks. It may also be useful to create slices with different security levels, with some devices being more strictly monitored than others. Network slicing also offers some sandboxing possibilities, in the way of creating a quarantine slice. Devices that are presumed to be malicious may be placed in this slice to prevent them from harming other devices. This slice may restrict internet access or perform deeper inspection [39]. Furthermore, network slicing also allows for adding security functions with finer granularity. Some examples of such security functions include antivirus solutions, application control systems, anti-spyware, domain filtering, logging, and intrusion detection. Unique security policies may also be added on a per-slice basis [40].

Besides providing additional security, network slicing also introduces several vulnerabilities. Impersonation attacks may allow adversaries to act as the network slice manager. This would give complete control over the routing and configuration of the network traffic. MitM attacks may occur when traffic between components facilitating network slicing is done in an unprotected manner. This could allow eavesdropping or impersonation attacks. Diversity in slice security means that attackers may target slices with low security to later reach slices with higher security. Proper isolation mechanisms would prevent such an attack path. At the hardware level, side-channel attacks may compromise slice activities. Hosting several network functions on the same virtual hardware may allow attackers to observe the CPU usage patterns and gain information on the system. DoS attacks may prevent slices and slice security functions from working. This may also be done by placing many mobility requests to the slice manager, causing an overload of the respective control plane functions. Jamming and flooding attacks may be used to deplete resources and reduce the availability of the system, both by targeting the network slicing infrastructure or by targeting resources in the network directly [39], [41].

There has been work on the improvement of security for network slicing technology. Some examples include efforts towards better load balancing in RANs, anomaly detection in RANs, different DoS protection mechanisms, co-residence prevention systems, and privacy preserving protocols. Side-channel attacks and UE attacks are still somewhat under-researched for network slicing [41].

2.3 IoT & 5G Network Slicing

The heterogeneous nature of IoT devices makes 5G network slicing very interesting for supporting IoT, as this technology allows supporting all of the diverse service requirements posed by IoT devices. In addition, any IoT devices that aim to use 5G will inherently run in a network slice (which may simply be the default slice with no customisation).

As previously mentioned, there are many benefits that network slicing brings to the IoT ecosystem. Scalability is improved, as a lot more devices can be connected to the same network by smartly dividing resources based on slice requirements. Since many types of IoT devices often have very minimal communication patterns (only sending a status update regularly) this is especially relevant. The capacity of the hardware can also be more efficiently used by more precisely allocating resources based on the needs of the devices in the slice, which may increase revenue for the providers. Dynamicity is improved, since network slicing allows for rapidly deploying custom (temporary) networks by allocating resources dynamically. This dynamic resource allocating may also allow for improving the quality of service (QoS) of varying IoT applications. Privacy can also be improved, for example when a dedicated network slice is used for private data transferring [1].

However, besides all of these benefits there are also several challenges that network slicing brings to the IoT ecosystem. Firstly, the network slice orchestrator must be able to handle rapidly scaling network slice requests. The dynamic goals of network slicing must be supported efficiently to prevent impact on network connectivity and latency. There have been some efforts towards creating machine learning techniques to handle such scaling. Recursion is another possibility that could provide benefits by allowing subslices to inherit the properties of parent slices. This would allow for a more granular approach to defining slices. Slice function chains may need to be dynamically implemented to allow for adding and removing network functions assigned to a slice to support changing requirements of IoT devices. Finally, the complexity of managing and providing slices is not to be underestimated. These systems consist of many functions for e.g., resource management and routing, which must be orchestrated carefully to handle possible failure scenarios [1].

Chapter 3

Related Work

The following sections explore the previous work that has been done in a similar direction to the current work. Section 3.1 describes several approaches that have been investigated for securing mobile networks. Subsection 3.1.1 mentions several works that explicitly use SDN and NFV, and subsection 3.1.2 mentions several works that explicitly use network slicing. Section 3.2 explores the response actions available for use in network security, as well as how these may translate to operations in the 5G core network based on the 3GPP specifications. Section 3.3 describes the research gap that is left open by the previous works.

3.1 Securing Mobile Networks

This section presents several works that have proposed solutions to secure mobile networks. This includes both the core network and the edge. The subsections contain works specifically using SDN and NFV and network slicing.

Ranaweera et al [31] propose an architecture for providing security-as-a-service (SaaS) in Mobile Edge Computing (MEC). Mobile edge hosts are devices that can run services at the edge. The framework proposes adding a mobile edge host that specialises in providing SaaS. It provides three security functions: IDS as a service, authentication as a service, and secure transmission channels as a service. The architecture also contains a component called a security analyser, which stores credentials, collects statistics, and performs threat analysis. The authors have partially implemented the system in a virtual machine with several Docker instances. Only the IDS as a service function is present in the paper.

Ageyev et al [20] propose a system with two components: detectors in the RAN, and a detection system in the core network (a 4G core network in their case). The detector sends the anomalies it finds through a set of functions that extract some information and determine a set of response actions, which are then forwarded to the VNF manager, which executes the response. The entire anomaly detection system is separate from the devices that route the user traffic.

3.1.1 SDN/NFV-based Approaches

This section contains works that explicitly make use of the SDN and NFV paradigms for their solutions. This means that some of these can also be applied in network slices, as the combination of these paradigms supports this technology.

Alejandro et al [24] propose an architecture for security orchestration in IoT networks with SDN and NFV. The system manages so-called security enablers. These are VNFs that provide security in various ways (e.g., a virtual firewall). The orchestration system allows for a user to define security policies in a high level language, which then gets translated to a medium level language and finally into configurations for the security enablers. The architecture also has a monitoring module and a response module. The monitoring module sends alerts when devices show anomalous behaviour, and the response module changes configurations of security enablers when an alert is received. All of these components are managed by the security orchestrator, which also analyses reaction outcomes. They implement a proof of concept which is used to test a single scenario: adding a rule to a firewall. The authors test the translation from the high level language to the application of the rule in three systems: iptables, Onos[1], and OpenDayLight[2]. The authors test the translation of these policies and find that 1000 rules can be translated in 30 seconds in all tested cases.

Alejandro et al [25] and Farris et al [26] propose a framework for orchestrating security functions in networks with NFV and SDN. The system has three levels of policy definitions, from high level policy language defined by the users to low level configurations of security components. They also provide a reasoning approach that can handle conflicts in policies. The project was performed in connection with the ANASTACIA[3] project and is based on the architecture defined in ANASTACIA. No implementation was provided and no testing was performed.

Khettab et al [30] propose an architecture for implementing security VNFs in an

[1]https://opennetworking.org/onos/
[2]https://www.opendaylight.org/
[3]http://www.anastacia-h2020.eu/

SDN/NFV-enabled network. They use Onos as their SDN controller. They identify two types of security VNFs to test: IDSs/IPSs and deep packet inspection (DPI) systems. They suggest that traffic may be mirrored to these services to prevent impact on latency. Alerts from the detection systems are sent to a central security orchestrator, which may then block the traffic flow in question or limit its bandwidth to still retain some level of service. The authors test the performance of Suricata, Snort[4], and Ntopng[5]. Suricata and Snort are IDSs/IPSs. Ntopng is a DPI system. They find that Suricata outperforms Snort, and Ntopng performs similarly to Suricata. Performance was measured in CPU usage, packet loss, and packet processing speed. They conclude that different security VNFs should have different scaling policies, as this is relevant to maximize security and performance. The paper does not further evaluate the security of the selected solutions (e.g., by evaluating detection accuracy).

3.1.2 Network Slicing-based Approaches

This section contains works that either explicitly make use of network slicing in parts of their solution, or that specifically mention how their approach may be applied in a network slice.

Candal-Ventureira et al [2] propose a two-step SDN architecture for detecting and mitigating IoT DDoS attacks. The first step is that the traffic of a set of devices is briefly analysed and classified as suspicious or not suspicious. This is done on the basis of simple flow rules present in the SDN switch. The accuracy of this approach depends on the granularity of the flow rules, as well as the size of the device sets. In the case of this study, they used throughput as a simple indicator of suspicious traffic. A threshold can be selected to reduce false positives or increase the strictness of this system. The authors mention that there has been research into the traffic patterns of IoT devices, but that a simple approach suffices for their purposes. Then, if suspicious traffic is found, the set of devices has its traffic routed to a quarantine slice, where DPI is performed. This dynamic reallocation requires an SDN application that is able to duplicate the state of any user-plane functions between the slices, to allow devices to continue functioning. The authors use a 4G network based on OpenAirInterface and simulate a set of factory automation IoT devices to test their approach. They find that their detection technique is viable and their dynamic reallocation system works well. The time required to reallocate a set of devices is roughly 1 second, but can be reduced by proactively performing some of the allocation steps

[4]https://www.snort.org/

[5]https://www.ntop.org/products/traffic-analysis/ntop/

(e.g., already instantiating a slice).

Chafika et al [32] propose a high level orchestration framework for secure network slicing. The project was performed in context of the EU H2020 MonB5G[6] project. This project has its own network slicing orchestration architecture, in which the authors have fitted their framework. The framework describes the placing of various components in relation to each network slice. The system has two types of security orchestrators: local security orchestrators and end-to-end security orchestrators. The local secrity orchestrators implement security functions at the network slice level. The end-to-end security orchestrators use a global view of the network to manage security at a higher level. These may decide to move slices and make decisions on where slices may be deployed. It also describes the general steps taken by the local orchestrator to analyse vulnerabilities, identify security goals for slices, and implement protective measures. The framework has not been implemented. The authors plan to start implementing parts of the framework in their future work.

Jain et al [22] created a Python-based simulation suite for simulating an environment with different types of IoT devices, several base stations, and network slices. Device configuration is input in YAML format. The simulation suite also implements a rudimentary IDS based on configuration parameters of the entire system e.g., number of allowed failed authentication attempts. The parameters are set manually. The IDS detects attacks on the devices, base stations, and slices themselves. The only response action in their implementation is removing a device from the network entirely. They test with several manually triggered anomalies and find that their proof-of-concept works. Five types of IoT devices were used in the simulation, with different mobility and usage patterns.

In a later work, Jain et al [23] expand their slicing simulation suite to incorporate a machine learning component for detecting anomalies based on bandwidth usage. In this paper they narrow the scope of their research to the healthcare use case, which adds an interesting additional challenge in the form of life-critical devices. They solve this by creating a backup slice for these devices and changing the response policy to only alert an administrator in case a life-critical device shows anomalies (instead of removing it from the network entirely). They use a neural network and achieve a good accuracy (99,4%) at detecting anomalies. Attacks were once again manually triggered, and also included the previously used attacks on base stations and slices.

Lam et al [42] propose a software-defined security (SDS) system that inspects traf-

[6]https://www.monb5g.eu/

fic coming in to and going out of the core network. It makes use of a convolutional neural network. The system reads traffic from the backhaul link (the link between cloud RAN and core network) and the interconnect link (the link between core network and internet). It also has an IDS module in the core network. The system can be adapted per-slice by customising the SDS system connected to each slice. They use the CICIDS2018 dataset, which contains many different devices and several protocols. Anomaly detection is done based on the inter-arrival time of each traffic flow. This is the average amount of flows that reaches a host in a given time frame. Some general flow statistics are also used. These features are mapped to 224x224x3 images and fed into the neural network. The authors find a very high precision (98,9%) when predicting anomalous and benign traffic in their dataset and conclude that machine learning techniques seem very suitable for the purpose.

Sattar et al [43] investigate the possibilities for using slice isolation to protect against DDoS attacks in 5G networks. DDoS attacks in network slicing may target either the hosts resources or the communication links. The authors consider two types of attacks: DDoS flooding attacks against communication links, and slice-initiated attacks, where the adversary runs a set of VNFs at maximum capacity to exhaust the slice resources. To handle these attacks, they present an optimization system that allows for defining isolation requirements for each slice. The optimization system then calculates a scheme which describes how to best allocate resources to the slices. They implement a testbed with 12 slices and allocate resources according to the calculated scheme. Then, they attack one of the slices using a DDoS flood and measure the impact on one of the other slices. They find that higher isolation reduces the impact the DDoS attack has on the second slice in terms of response time, bandwidth availability, and round trip time. As such, important services may be hosted on more isolated slices, to prevent impact of DDoS attacks. They also perform a slice-initiated attack, and find that all isolation levels in their approach prevent this type of attack from having any impact.

3.2 Response Actions in Network Security

The paper by van Leeuwen [44] presents an overview of containment actions that may be applied in case of an attack. They present a topology of response actions based on the goal of each action and the general tactic of each action. They identify three main goals and various subgoals:

- Incident containment
    - Isolation

    - Shrinking the attack surface

- Slow down the adversary
    - Disrupt the attack
    - Offensive defense
    - Deceptive defense

- Managing information
    - Gathering
    - Preserving
    - Sharing

In addition, the paper provides a matrix for determining which containment actions are available in a given network, based on the security actuators in place. In the scenario of a 5G core operator using the previously described monitoring approach we have two security actuators to our disposal:

1. OSI level 3 capable security actuator (i.e. the UPF with routing capabilities)
2. Network-based security agent (i.e., the IDS)

According to the matrix in the paper these actuators give us the following options for containment actions:

- Traffic filtering/rerouting
- Disconnect/isolate host
- Disconnect/isolate network
- Add additional logging
- Scan for known indicators of compromise
- Enable remote logging
- Logging to unchangeable media
- Generate an alarm
- Generate a report

Various of the other papers above also mention response actions. For the current scenario the most notable response actions are the following: Candal Ventureira et al [2] suggest moving a malicious UE to a different network slice where its traffic is more closely monitored. Nobakht et al [6] make use of flow filters to filter traffic. Specifically, they use thresholds for throughput rates on suspicious devices. Jain et al [22] have two response actions in their simulated environment: disabling a device, and disabling a network slice. Khettab et al [13] have two responses similar to the ones by Nobakht et al [6], namely stopping the traffic flow (i.e., resetting the connection) and limiting the bandwidth of a suspicious device.

Several other papers we examined during the literature review also suggested interesting response actions. Midi et al [45] present a paper focused on wireless sensor networks, which are quite different from the envisioned telecom network. However, several response actions are suggested: collecting more information, alerting neighbors and the device itself, discarding network traffic, suspending a host, and permanently blocking the host. Hafeez et al [46] make use of adhoc overlay networks (AONs). These are virtual network which are overlaid on existing networks and can restrict hosts from communicating to other hosts. They suggest three types: no access, restricted access, and full access. Mishima et al [47] present the results of a system put in place where devices displaying suspicious behaviour are isolated from the network. This is mainly aimed at user devices such as laptops, and works by publishing the isolation information on a website with instructions on how to access the network again. The user is expected to perform tasks such as scanning for malware before they are allowed on the network again.

For traffic manipulation there seem to be three levels of severity:

- Doing nothing
- Restricting traffic (limited target hosts, bandwidth, rerouting etc.)
- Disconnecting/isolating

For any of these options there is the additional axis of time to consider i.e., how long should we isolate the device. Generally this should be a permanent action as the device is compromised and it will not simply be fixed over time. However, if the owner of the device removes the threat (e.g., by removing malware) the restriction could be lifted. Another option is to isolate a device for a predetermined timespan, which may be a solution to stop a current attack but keep the device available at a later time.

Then there is the option of taking an entire part of the network (or slice) offline or isolating it. This solution also may have a large effect on other devices, meaning it should only be used if such impact is acceptable and no other response would suffice. There are also the one-time disconnect actions, which only reset the session of the connected device, but do not permanently disconnect it. The device can immediately try to connect again and will go through registration procedures as normal. This only impacts the session that is terminated. Finally, there are several information-related actions that may be taken. For example, additional scanning of traffic, collecting more logs, and alerting involved parties may be done in this space.

3.2.1 Response Actions in 5G Core Networks

The 5G core network is standardised by the 3GPP foundation. Within the scope of these specifications various methods of influencing traffic exist. Some of these can be used as response actions by making use of the service-based architecture of the control plane i.e., interacting with the control plane functions to initiate such response actions.

The UPF contains several mechanisms that can be used to restrict traffic. When packets reach the UPF from either side of the network (gNb/core or data network) they are matched against a packet detection rule (PDR). This PDR may contain various information elements that uniquely identify the traffic. This includes the IP address of the UE, the interface on which the packet arrives, and various flow identifiers and filters. Each PDR also contains a link to a forwarding action rule (FAR). A FAR is a rule that indicates what the UPF should do with the arrived packets. The options Forward, Drop, Duplicate, and Buffer are available. These PDRs and FARs are provided by the session management function (SMF). This is a control plane function that handles session management. Applying a new PDR and FAR can lead to restricting the traffic of a given UE or network, or the complete isolation of a given UE or network [48], [49].

The UPF can also restrict the bandwidth of the UE. This is done by enforcing the maximum bitrate for the device. This may be done by setting a QoS enforcement rule (QER) linked to the PDR. This QER contains a value for the maximum bitrate, which can be one of several types, with subtly different specifications [48], [49].

The 3GPP specifications mention a single main method for influencing traffic from outside the 5G core network control plane. This method makes use of the application function (AF). The AF is a function that offers ”application services”, which may include many things, such as video streaming services. An application service for monitoring and managing IoT devices may be set up using the AF. The capabilities of the AF for influencing the traffic in the 5G core network make use of the policy system provided by the policy control function (PCF). This policy system allows the AF to set up filters for the data streams the function is concerned with. It can also apply various policies on these filters, from dropping traffic to restricting bandwidth. A trusted AF may directly interact with the PCF, while an outside AF has to use the functions provided by the network exposure function (NEF). The NEF performs some checks and forwards the request. The PCF communicates changes in policy to the SMF, which translates it to rules used by the UPF, such as PDRs and FARs. This action can lead to filtering and/or routing traffic, isolating a host, or isolating a network [48], [50]–[53].

In addition to this main method there are several options for slightly altering network functions to accept requests from the monitoring agent. The SMF may be altered to accept requests for changes in PDR and FAR for a target UE. This would provide an easy way to essentially handle traffic filtering and isolating at the lowest level .i.e., closest to the UPF. The SMF would translates the requests and directly forward them to the UPF. A similar approach could be taken for restricting device bandwidth. This would be done by modifying the unified data management function (UDM) to accept a request. The UDM would then inform the SMF, which would in turn communicate with the PCF and AMF. The AMF would inform the UE and await a response, which may be an issue if the UE is compromised. One could possibly add an option to the AMF to not wait for a response. If a response is obtained, the AMF informs the SMF, which updates the UPF to restrict the bandwidth. This action only leads to restricting traffic of a given UE [48], [50], [53].

Another option for a response action is a network-initiated de-registration, i.e., the network forces the UE to disconnect. This action may be initiated by various functions, but the 3GPP specifications suggest the following. First the UDM de-registers the UE in its internal database. This may include the entire subscriber being removed, or possibly that the information is still stored, but the state is set to ”de-registered”. The UDM then sends a notification to the AMF which the UE is connected to. The AMF then sends a de-registration request to the UE and informs the SMF. The SMF then instructs the UPF to release the session. Various confirmations are then sent between the control plane components, after which the AMF releases the signalling context with the RAN, after which all connections are closed. This action only leads to isolation of a given UE [48], [50], [53].

A final interesting response action may be to perform a network-initiated inter-slice change. That is, the network forces the UE to connect to a different network slice with different properties. The new slice may allocate lower bandwidth per device, have additional monitoring systems, or simply be completely isolated. This action could be done if the UDM was changed to accept a request. The subscription data of the UE would be changed in the UDM, indicating the UE is no longer allowed to be connected to the original network slice. The AMF sends a command to the UE to update its config. It then waits for a response from the UE, which may once again be an issue if the UE is compromised. If a response is obtained the AMF informs the SMF and gNb to release the original sessions. The UE then has to attempt to reconnect, which leads to it connecting to the new network slice. This action could lead to filtering traffic, routing traffic differently, or isolating the device completely [48], [50], [54].

3.3 Research Gap

There have been many studies into the securing of IoT devices from the network perspective. Several of these papers have attempted to do this in the context of network slicing. However, there is still a significant research gap to be filled. Some of the previous works [22], [23] did not use 5G components that comply with the existing specifications. Other works [24]–[26], [30], [32] only present frameworks, without further testing the efficiency of the solutions. Other papers [42] are only concerned with the detection of threats, and not acting on alerts. Sattar et al [43] and Candal-Ventureira et al [2] do investigate some response actions, but none of the actions suggested in this project (they investigated isolating groups of devices in a network slice and using slice isolation to protect against DDoS attacks, respectively). Lam et al [42], Khettab et al [30], and Ageyev et al [20] use a separate monitoring host for their systems. Khettab et al [30] mention some of the trade-offs for this approach, but do not perform any additional tests. The response actions suggested in section 3.2.1 have not been investigated before.

As such, this research project contributes to the body of research by further investigating another (automated) response action available in 5G network slices with IoT devices, namely the PDR/FAR change. The project combines this action with an implementation using 5G network components that comply with the 3GPP specifications, as well as investigating the efficiency of using a separate monitoring agent in this context.

Chapter 4

Approach

The following sections describe the selected approach for the current work. Section 4.1 shortly describes the practical goals of the approach. Section 4.2 describes the approach to monitoring the IoT network traffic. Section 4.3 describes the approach for adding the response action to the monitoring setup. Section 4.4 describes how we approach the evaluation of the proposed solutions to reach the goals of the project.

4.1 Goals

We have two main goals for the testing approach: Firstly, we want to find out whether a network traffic monitoring system that uses a separate monitoring agent can detect attacks in a timely manner when applied in a realistic 5G network slice containing IoT devices. Note that this work does not focus on the accuracy of the detection, but the speed of detection in the complete architecture, as the aforementioned accuracy has been extensively researched in the past. Secondly, we want to find out whether the response action (PDR/FAR change) has the desired effect and can be automatically deployed in a timely manner. These two results will indicate whether the proposed solution would be suitable for protecting IoT devices in a 5G network slice.

4.2 Network Traffic Monitoring

We set up a basic testbed using 5G components that comply with the 3GPP specifications. The architecture for the testbed can be observed in figure 4.1. For the access technology we use a gNb, as this represents a common scenario that likely supports network slicing in the future. Both the UEs and gNb are simulated, as no 5G hardware is available for the project. We also do not use real IoT devices, but use real traffic traces to replay our attacks. There is a UPF for handling user traffic

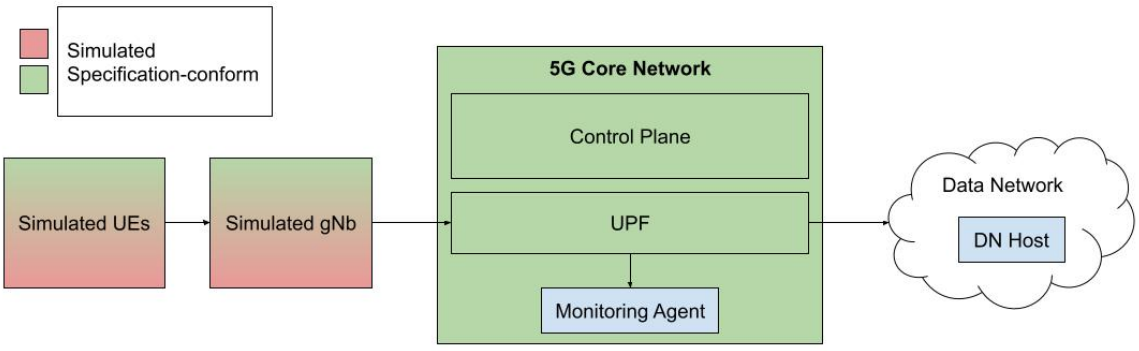

Figure 4.1: Architecture for basic testbed using 5G components that conform to the 3GPP specifications.

and a separate host for the control plane functions. In addition, we added a host outside of the 5G network (i.e., somewhere in the data network) to use for testing attack scenarios that require two-way communication. This host will from now on be referred to as the ”DN Host”.

We place a monitoring agent on a separate host to prevent load on the UPF. The UPF mirrors traffic to the monitoring agent, which then analyses it. The main benefit envisioned for this approach is that the UPF does not have to wait for the Suricata analysis to forward the packets, which means there should be no impact on the latency between the UE and the target of the communication. A drawback of this approach is that the monitoring agent does not immediately act as an IPS, which means attacks will pass through until the monitoring agent triggers a response. Whether this is a problem depends on the attack. We use a simple default network slice for the IoT devices in our test. The UPF can discern between traffic coming from the IoT slice and traffic coming from other slices, and only mirrors the traffic from the IoT slice to the monitoring agent.

4.3 Response Actions

Implementing the response actions described in section 3.2.1 allows us to investigate how they work and how quickly they can be applied to a threat. We implement one of the response actions (PDR/FAR change) and integrate it with the network monitoring setup from the previous section. The response is automatically applied by interacting with the 5G control plane when an alert is fired by the monitoring system.

4.4 Evaluation Approach

We use the testbed to test the network monitoring solution for a (set of) IoT device(s) by sending some simple traffic through the setup. Then, we observe the latency components present in the setup and analyse them to find the expected times for various events during an attack (packets being sent, packets arriving at the monitoring agent, etc.). In addition, we test the system using several replayed attacks to evaluate the efficiency of the monitoring setup in real attack scenarios. Once again we collect timestamps to analyse how the detection compares to the baseline investigated in the previous part. Finding this timing information provides valuable information on whether such an approach is feasible to use in practice, and whether it will perform well in improving the security of the devices that are monitored.

We test the automated response with one of the replayed attacks. Once again, we collect and analyse timestamps to investigate the speed of the solution. We also investigate whether the expected outcome of the response action is reached. Finding this information will allow us to determine the whether the response action would be useful in practice.

Chapter 5

Implementation

The following sections describe the implementation of the approach from the previous chapter. Section 5.1 describes the overarching architecture of the testbed and the technologies used to implement it. Section 5.2 describes the various components in further details, as well as various flows of events that occur on the testbed.

5.1 System Architecture

The architecture of the system can be observed in figure 5.1. The system runs on a total of 5 virtual machines connected through a private network in OpenStack[1].

Virtual machine #1 contains the UEs and the gNb. We simulate the gNb and UEs using the open source UERANSIM[2]. The virtual machine sends user traffic from the IoT UEs to the UPF, and exchanges control plane traffic with the 5G control plane.

Virtual machine #2 contains the UPF. The UPF is part of the 5G core network, which we implement using the open source Free5GC[3] project. The Free5GC project is based on the 3GPP specifications release 15. The virtual machine receives user traffic from the UERANSIM virtual machine and routes it to the DN Host. It also mirrors the user traffic of the IoT network slice to the monitoring agent and exchanges control traffic with the 5G control plane.

Virtual machine #3 contains the 5G control plane. This includes all control plane functions (e.g., AMF), which can all be separately addressed within the machine. The control plane is also implemented using Free5GC. The virtual machine exchanges control traffic with the UERANSIM virtual machine, the UPF, and the mon-

[1]https://www.openstack.org/

[2]https://github.com/aligungr/UERANSIM

[3]https://www.free5gc.org/

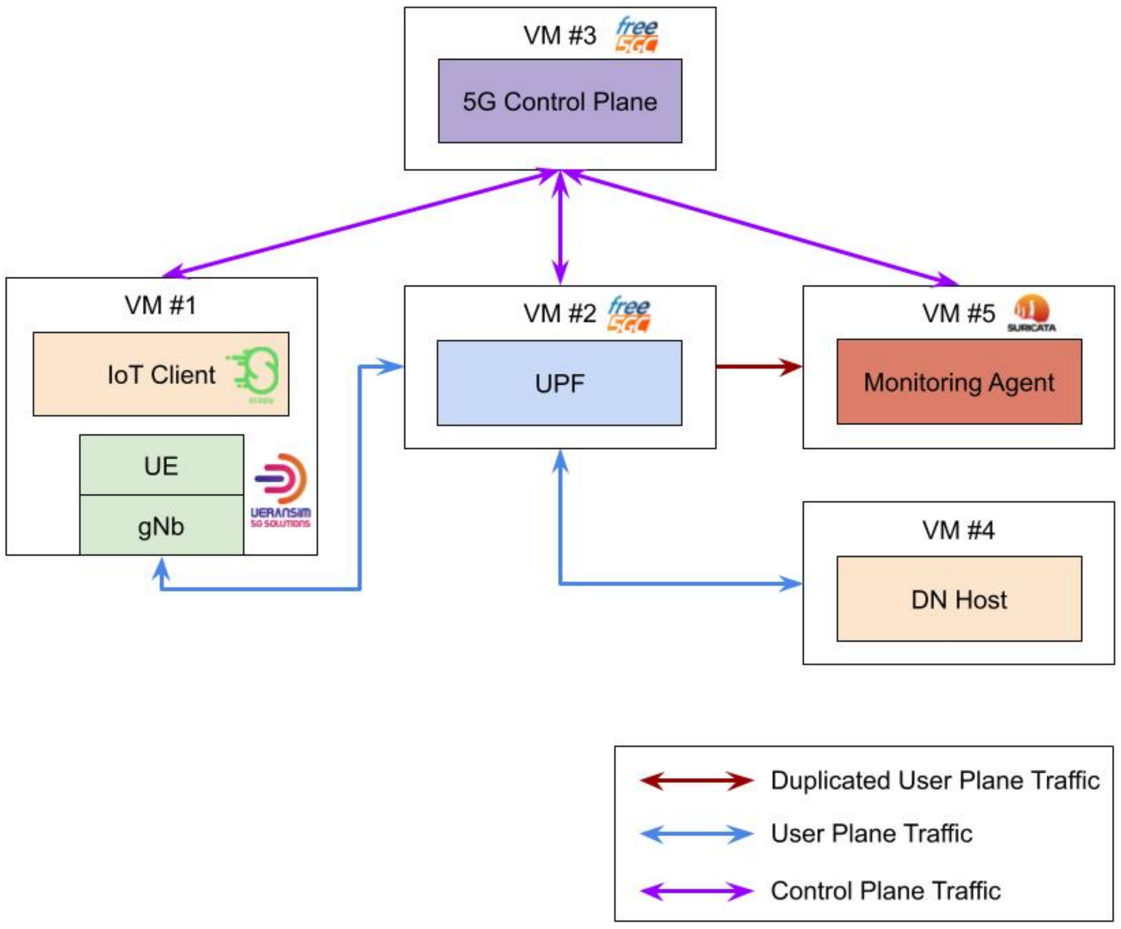

Figure 5.1: Architecture for testbed with monitoring solution.

itoring agent. The monitoring agent may send a request for applying a response action, to which the control plane responds with a confirmation or an error code.

Virtual machine #4 runs the DN Host. This virtual machine simply sends and receives user traffic by communicating with the UPF. It is used for two-way communication in the experiments. In a realistic scenario this would be any host on the internet or outside network.

Virtual machine #5 runs the traffic monitoring functions. Traffic received by this virtual machine is analysed using Suricata[4]. The virtual machine receives mirrored user traffic from the UPF. It also exchanges control plane traffic with the 5G control plane, which includes sending a request to apply a PDR/FAR change.

5.2 System Details

Each virtual machine has a single network interface that is actually connected to the private OpenStack network. This is the *ens4* interface. Any traffic that leaves or reaches a virtual machine does so through this interface. However, there are various interfaces inside each virtual machine that handle functions of the 5G core and the traffic mirroring operations. See figure 5.2 for an overview of the interfaces a packet passes through during a simple request/response process. Various encapsulation steps are used in the process, some inherent to the 5G network and some used for the traffic mirroring system. See figure 5.3 for an overview of the steps a user traffic packet goes through during a simple request/response process.

The UERANSIM virtual machine contains the simulated UE and gNb. When starting up, the gNb connects to the control plane by contacting the AMF through the *ens4* interface. The AMF handles the registration request and connects the gNb. Then, when a UE is started, it will use the local loopback interface to connect to the gNb. When a UE connects to the core network through the gNb, the AMF instructs it to connect to a UPF that corresponds to the slice requested by the UE. As such, the gNb passes any traffic coming from the UE to this UPF. UE traffic is sent into the *uesimtun0* interface on the virtual machine running UERANSIM. This traffic is then passed through the local loopback interface to the gNb on the same machine, which encapsulates it in a GPRS tunneling protocol user data (GTP-U) tunnel and passes it through the *ens4* interface to the UPF. For incoming user traffic the gNb removes the GTP-U encapsulation and passes the traffic to the destination UE using

[4]https://suricata.io/

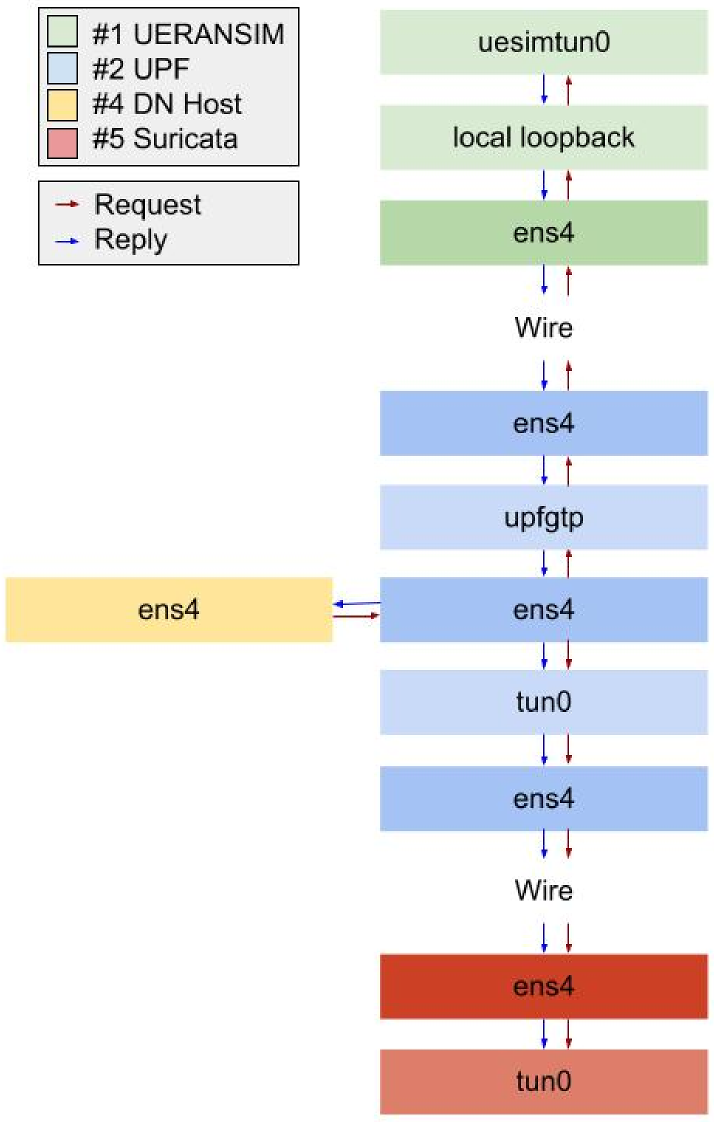

Figure 5.2: Interfaces each request/reply packet passes through during the testbed operation.

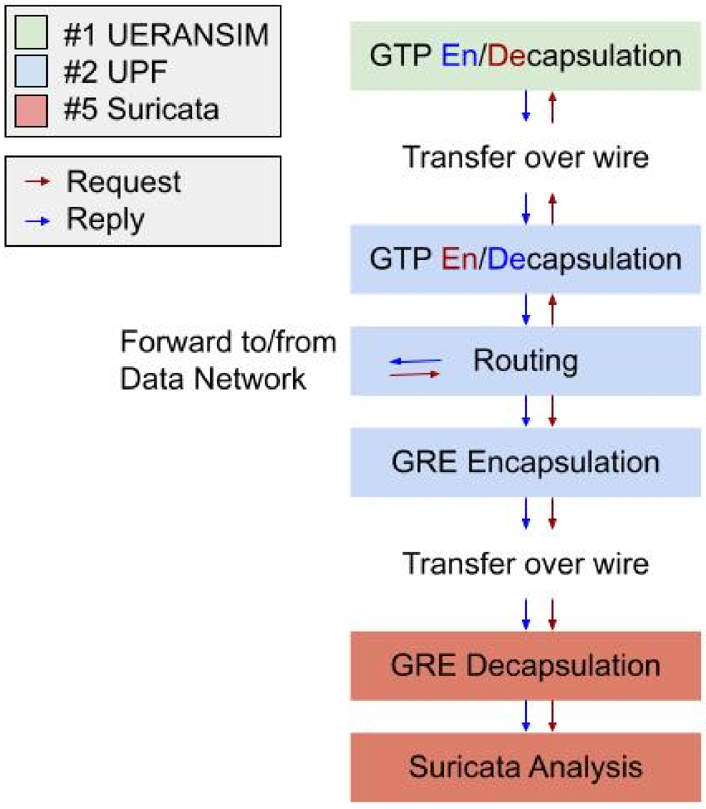

Figure 5.3: Steps each request/reply packet goes through during the testbed operation.

the same interfaces.

The UPF routes user traffic to or from the 5G network. User traffic from the gNb is first received on the *ens4* interface, which passes it to the *upfgtp* interface. Then, the GTP-U encapsulation is removed and the packet is routed to the target destination. This could be the DN Host, which means the packet will leave the UPF from the *ens4* interface. The UPF does not add any encapsulation when it sends packets to the DN Host, or any other host outside of the 5G network. In addition to routing, the machine also performs traffic duplication. The traffic on the *ens4* interface of the virtual machine is duplicated into a tunnel interface (*tun0*) leading to the virtual machine running Suricata. This is done by using the traffic control configuration of the Linux kernel, which allows defining of ingress and egress queue disciplines, as well as defining filters that can specify where to mirror the traffic. The system only mirrors network traffic originating from and travelling to a specific range of IP addresses. This range is linked to the IoT network slice. This requires some configuration in the 5G core, namely by giving each network slice a different data network configuration. Each slice may still route to the same destination (e.g., the internet) but will provide different IP addresses to the devices inside the slice. The tunnel interface the traffic is duplicated to uses the generic routing encapsulation (GRE) protocol. This simply provides an encapsulation layer around the original packet, which is stripped at the destination. As such, the monitoring virtual machine receives the packets exactly as they are on the *ens4* interface of the UPF.

The virtual machine used for traffic monitoring runs the Suricata IDS on the incoming tunnel interface (*tun0*). Packets arriving at the other interfaces on the device are ignored. Note that packets destined for interface *tun0* still pass through interface *ens4*, but are not analysed by Suricata twice. Suricata is a signature-based IDS, which means it detects threats by comparing incoming traffic to a set of signatures. Each signature is part of a rule which contains the signature, an action Suricata performs when a packet matches the signature, and some optional information on the threat. We configured the IDS with a small set of rules comprising of rules from the default emerging threats ruleset and several custom rules defined for the specific attacks tested in the project. See section 6.2 for the custom rules.

Only one of the response actions mentioned in 3.2.1 is implemented for this project. This response action is the PDR/FAR change. The other response actions have a larger time cost as more additional features need to be implemented in Free5GC before they can be used. As such, they are left as future work. The PDR/FAR change response is implemented by adding a feature to the SMF in the control plane. This feature allows an outside monitoring agent such as the one used in the project to request changes to the PDRs and FARs of a given UE, which essentially allows the system to set up traffic filters. This is done based on the IP address of the UE, but may be extended to more 5G-appropriate identifiers (e.g., PDU session ID) if the monitoring agent is integrated into the 5G core further. A smaller program on the monitoring agent continuously checks the Suricata log and sends a response request when an alert is read from the log. The request is a simple HTTP POST message containing the requested update in JSON format. The SMF then reads the request and starts the update procedure. See figure 5.4 for a sequence diagram of the steps in the response process.

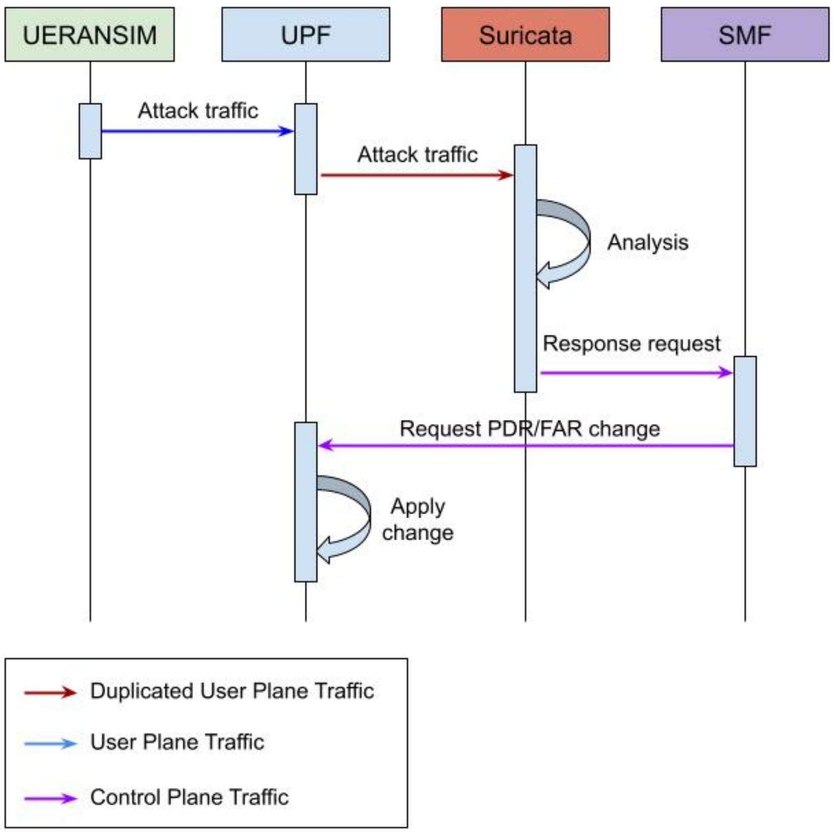

Figure 5.4: Steps during the response process for the PDR/FAR change action.

Chapter 6

Evaluation

The following sections describe how the implemented system is evaluated. Section 6.1 describes how the latency components in the system are tested and analysed. Section 6.2 describes how various attacks are replayed and detected by the system. Section 6.3 describes how the response action is tested.

6.1 Latency Testing

We use a simple method for testing the latency of the various steps in the transfer and duplication process. We run a network traffic capture (tcpdump[1]) on each relevant interface on each host that is part of the traffic routing process. See table 6.1 for an overview of these interfaces. Then, we start a simple ICMP ping session from the virtual machine running UERANSIM, targeting the DN Host. The ICMP packet travels through the UPF, to the DN Host, and back through the UPF. The packets are also duplicated by the UPF and sent to the virtual machine running Suricata. By analyzing the timestamps for each captured packet at the different interfaces, we find an average latency for each part of the process. Figure 5.3 shows the steps each ping packet goes through during the test.

6.2 Attack Detection Testing

We select several of attacks from the IoT network intrusion dataset collected by Kang et al [55] and edit them for the purpose of this project. See table 6.2 for a description of the selected attacks. We specifically select the attacks that can be performed between two hosts and reduce the traffic captures to the traffic between those two hosts. Then, we change the IP addresses to the ones used in the system architecture of this project.

[1] https://www.tcpdump.org/

Table 6.1: Network traffic capture interfaces for latency testing.

VM	**Interface**
UERANSIM	uesimtun0
UERANSIM	ens4
UPF	ens4
UPF	upfgtp
UPF	tun0
DN Host	ens4
Suricata	tun0

Table 6.2: Attacks for testing effectiveness of monitoring approach.

Attack	**Direction**	**Malware**	**Description**
ACK Flood	UE $\rightarrow$ DN Host	Mirai	The UE performs a flooding attack using TCP ACK packets.
Telnet Bruteforce	DN Host $\leftrightarrow$ UE	Mirai	The DN Host host attempts to break into the UE using default telnet passwords.
UDP Flood	UE $\rightarrow$ DN Host	Mirai	The UE performs a flooding attack using UDP packets.
Port scan	DN Host $\leftrightarrow$ UE	Generic	The DN Host host scans the UE for open ports using nmap[2].
Port and OS scan	DN Host $\leftrightarrow$ UE	Generic	The DN Host host scans the UE for open ports, then attempts to identify the OS using nmap.

Table 6.3: Custom rules for detecting flooding and scanning attacks using Suricata.

Attack	**Rule**	**Suricata syntax**
ACK Flood	Triggers an alert when 100 TCP ACK packets are sent between two hosts in 1 second.	alert tcp any any $\rightarrow$ any any (msg: "Device performing ACK flood."; flags:A; threshold: type threshold, count 100, seconds 1, track by_src; flow:to_server;)
UDP Flood	Triggers an alert when 1000 UDP packets are sent between two hosts in 1 second.	alert udp any any $\rightarrow$ any any (msg: "Device performing UDP flood."; threshold: type threshold, count 1000, seconds 1, track by_src; flow:to_server;)
Port scan	Triggers an alert when 50 TCP SYN packets are sent from a single host in 1 second.	alert tcp any any $\rightarrow$ any any (msg:"SYNSTEALTH SCAN DETECTED"; flow:stateless; flags:S,12; threshold:type threshold, track by_src, count 50, seconds 1;)

Table 6.4: Interfaces for network traffic capture for response testing.

VM	**Interface**
UERANSIM	uesimtun0
Suricata	tun0
Suricata	ens4
Control Plane	ens4
UPF	ens4

To replay the attacks as they are in the dataset it is necessary for the hosts to have a dialogue i.e., each host waits for the response from the other side as it is in the traffic capture. This is mainly relevant for the telnet bruteforce (as it is expected the host responds to attempted logins). We use Scapy[3] to create a tool that can do this. Both hosts read the traffic capture file and replay the traffic exactly as it was originally recorded. We test several captures for each attack to be able to find a mean time to detect.

Some of the attacks in our testing repertoire are recognized by Suricata using the default emerging threats ruleset. However, for the remaining attacks we create some additional rules. Suricata does not provide rules against denial of service (DoS) attacks in its default ruleset, as the definition of a DoS attack may differ between scenarios. See table 6.3 for the custom rules.

6.3 Response Action Testing

We test the response using a single ACK flood pcap from the selection of attacks mentioned previously. The update instruction sent to the SMF is a request to change the existing FAR rule to simply block all incoming and outgoing traffic. We reset the state of the PDR and FAR associated with the UE for every test. We also collect timestamps for the detection steps and the response steps. The interfaces used for this can be observed in table 6.4. We also test the response action with various IP ranges to determine whether the devices that should stay unaffected can indeed still transmit traffic to the data network. We also test this for traffic coming towards the IoT devices from the data network.

[3]https://scapy.net/

Chapter 7

Results

This chapter contains the results of the research. Section 7.1 presents the results of the latency testing of the monitoring system. Section 7.2 presents the results of testing the monitoring system against the attacks described in the previous chapter. Section 7.3 presents the results of testing with the implemented response action. Section 7.4 shortly discusses the results.

7.1 Latency

Table 7.1 shows the latency results for transmission between the UE and the DN Host. The latency results were recorded over 200 ICMP packets (100 request, 100 response) using the timestamps from the tcpdump captures. There are small differences in various components with regards to outgoing or incoming packets. However, the sum of the averages for each direction is very close ($\sim$1,41ms for outgoing and $\sim$1,34ms for incoming). This may indicate that various operations (e.g., encapsulation and decapsulation) are happening in different components based on whether they apply to incoming or outgoing packets. This could also explain the 0 value for the outgoing packets between the ens4 interface and the upfgtp interface on the UPF, as both the decapsulation and processing of the packet may only occur when it has been passed to the upfgtp interface. Nonetheless, this gives an idea of the normal latency components present in the simple 5G core network example.

Table 7.2 shows the latency results for the additional steps required for the monitoring of traffic. Here we also see small differences in whether packets are incoming or outgoing. Note that from the perspective of the Suricata machine all packets are technically ”incoming”, as all packets are mirrored from the UPF. This means the differences in latency are caused by the type of packet (request or response) and not by the direction the packet is traveling. The sum of averages for each direction

Table 7.1: Latency components of transmission between DN Host and UE.

VM	Interfaces	Latency request (ms)	Latency reply (ms)
UERANSIM	uesimtun0 → ens4	0,695306	0,573517
-	ens4 (UERANSIM) → ens4 (UPF)	0,413568	0,471501
UPF	ens4 (incoming) → upfgtp	0	0,011100
UPF	upfgtp → ens4 (outgoing)	0,053120	0,036416
-	ens4 (UPF) → ens4 (DN Host)	0,248233	0,248445
		1,410277	1,340979

Table 7.2: Additional latency components in monitoring setup.

VM	Interfaces	Latency request (ms)	Latency reply (ms)
UPF	ens4 (outgoing) → tun0	0,023693	0,005998
UPF	tun0 → ens4 (tunnel)	0,013094	0,009498
-	ens4 (UPF) → tun0 (Suricata)	1,411878	1,228301
Suricata	ens4 → alert	21,579295	25,872155
		24,189954	28,208484

is different here, which can mainly be attributed by the difference in analysis times required by Suricata. The difference in analysis time may be due to various factors outside the scope of this research, such as the way Suricata processes these packets. The main time cost comes from the analysis done by Suricata (90,87% of the total). This takes ∼24,2ms for outgoing packets and ∼28,2ms for incoming packets. The transfer between the virtual machine running the UPF and the virtual machine running Suricata also takes a large amount of time in comparison to the other components (4,71% of the total).

7.2 Attacks

Table 7.3 shows the mean time to detection (MTTD) for the tested attacks. This metric is split up into two different variants. The main MTTD is the time between the moment the first packet is seen on the outgoing interface of the attacker (i.e., the start of the attack) and the moment an alert shows up in the Suricata log. The second option measures from the moment the packet that triggers the firing of the alert (i.e., the 100th packet in 1 second) is seen on the outgoing interface of the attacker to the moment an alert shows in the Suricata log, and is denoted "MTTD from threshold". This way of looking at the detection time gives some insight into what the numbers would look like for single-packet attacks, as the time taken to reach the threshold for the flooding rules is not taken into account using this method.

Table 7.3: Mean time to detection (MTTD) for the tested attacks.

	MTTD (ms)		**MTTD from threshold (ms)**	
	AVG	STDEV	**AVG**	**STDEV**
Mirai ACK flood	68,25	14,05	**12,69**	**3,72**
Mirai Telnet bruteforce	120536,52	44536,62	**119500,70**	**44481,62**
Mirai UDP flood	530,48	78,06	**14,42**	**4,72**
Port scanning	224,69	17,90	**12,06**	**5,34**
Port and OS scanning	225,37	15,91	**12,73**	**4,91**

Note that there is a serious increase in time for the Telnet bruteforce. This is an anomaly that occurred where it took Suricata several minutes to show the alert in the log after receiving the alerting packet. We suspect this is due to the specifics of the rule used for detecting this attack. The rule uses the ”stream:established” keyword. This causes Suricata to make use of its TCP re-assembly system. This system keeps track of the state of the TCP stream by looking at occurrences of the three-way handshake and the four-way termination handshake. The streams used in the attack simulation all use the three-way handshake, send some traffic, and then do the four-way termination handshake. The system then knows the state is closed. For each state of the stream the system has a timeout, which indicates when the stream will be taken out of memory. For a closed state this timeout is 120 seconds. However, normally this timeout would not be relevant, as the data sent in the stream would be analysed immediately. However, since Suricata is using the TCP re-assembly system, it also waits for a chunk of data to arrive before analysing it. The chunk limit is not reached by the short communication in the attack, and therefore the data is only analysed when the timeout triggers. One of the bruteforce pcaps triggered the chunk limit and lead to lower detection times, hence the total average time slightly below 120 seconds. Note that the above explanation has not been confirmed, but is assumed to be the likely cause of the outlier.

In a real implementation this rule should be changed so it would immediately analyse the data in the stream. Alternatively, Suricata could be run in IPS mode, which forces it to immediately analyse the data (i.e., not wait for a chunk to reach the limit). If the time between receiving the alerting packet and the alert timestamp is zeroed to approximate this scenario we get table 7.4, which shows that the average for the Telnet bruteforce is in the same order of magnitude as the other averages, but still an outlier. This gives an average over all attacks of $\sim$11,32ms between the triggering packet being sent and an alert showing up in the Suricata log.

Table 7.4: Mean time to detection (MTTD) for the tested attacks, corrected for anomalous Telnet bruteforce values.

	MTTD (ms)		**MTTD from threshold (ms)**	
	AVG	STDEV	**AVG**	**STDEV**
Mirai ACK flood	68,25	14,05	**12,69**	**3,72**
Mirai Telnet bruteforce	1040,48	56,64	**4,67**	**1,64**
Mirai UDP flood	530,48	78,06	**14,42**	**4,72**
Port scanning	224,69	17,90	**12,06**	**5,34**
Port and OS scanning	225,37	15,91	**12,73**	**4,91**

Variance between the different pcaps for each attack was observed. For the flooding attacks this was mainly due to the different traffic traces being more or less aggressive at the start of the attack. An ACK flooding attack that sends 100 packets within 10ms starts the reporting process after those 10ms, whereas a less aggressive attack might send the 100 packets in 20ms and start the reporting process later, resulting in a longer time to detection. The detection time for the bruteforce attack depended on when a specific credential was attempted by the attacker. The rule in the standard set would trigger on a packet containing that credential. Some of the pcaps would have this credential early in the bruteforce, and others later. This means there is a large variance in the times to detection for the bruteforce attack. In addition to this variance between different pcap files, we also found some variance when using the same pcap file. Most of this variance occurred between the first packet being sent and the alerting packet being sent. There are many factors that may play into this phenomenon, including the replay scripts, the state of the network interfaces, and the various encapsulation steps.

We can also obtain some information on the scenario where Suricata would have run as an IPS on the UPF. We do this by looking at the time Suricata takes between receiving a triggering packet (i.e., a packet that triggers an alerting rule) and showing an alert in the log. This gives a clue as to the time Suricata requires for analysing the (stream of) packet(s) and generating an alert. See table 7.5 for this data for each attack. If we assume that the analysis times would be similar when running Suricata in IPS mode, we can see that each packet would take an additional $\sim$10,81ms to reach its destination. However, in the current scenario the monitoring agent has to strip the GRE encapsulation from the packets before they are passed to the Suricata analysis engine. This time cost is also included in those $\sim$10,81ms, and would not be present when using the system in IPS mode on the UPF machine. We can also look at table 7.2 to see what the analysis time looks for for ICMP packets. Here we

Table 7.5: Time taken by Suricata between receiving a triggering packet and showing an alert in the log.

	Suricata Analysis Time (ms)	
	AVG	**STDEV**
Mirai ACK flood	11,99	1,88
Mirai Telnet bruteforce	3,83	1,42
Mirai UDP flood	14,00	1,89
Port scanning	11,76	2,53
Port and OS scanning	12,46	2,23

Table 7.6: Time to response for PDR/FAR change.

	MTT (ms)		MTT from threshold (ms)	
	AVG	**STDEV**	**AVG**	**STDEV**
Detection	126,84	24,91	13,35	11,95
Response	37,10	50,05	37,10	50,05
Total	**163,94**	**57,10**	**50,45**	**51,87**

find a much higher time cost of ~21,58ms for ICMP request packets and ~25,87ms for ICMP reply packets. Here the same issue with removing the GRE encapsulation is present. Note that we have not tested Suricata in IPS mode, and we imagine there are many factors that influence this latency (such as packet type and packet size).

7.3 Response

Table 7.6 shows the latency observed for the response testing with a single ACK flood pcap. Once again the results are split up into two variants. The variants only differ in the Detection portion of the process. The variants are the same as described before, namely excluding or including the time it takes for the attack to reach the threshold. The Response portion is the same for both variants and records the time from the moment the log shows up until the traffic block is in place in the UPF.

We note that the detection results during this test differed slightly from the ones found when only the detection system was tested. The average time until the attack was detected was slightly higher and the standard deviation was much larger. This is caused by a single outlier where detection took ~45,85ms. Without this outlier we find a detection time of ~9,74ms on average with a standard deviation of ~3,72ms.

The response portion of the process has a relatively large standard deviation. This is caused by a single outlier where the SMF took over 160 milliseconds between receiving the request and sending a request to the UPF. The other tests (i.e., disregarding the outlier) resulted in an average of $\sim$21,32ms with a standard deviation of $\sim$4,05ms.

We also test the response action in a few different configurations. We set the PDR change to only affect a certain range of IP addresses and find that the IP addresses targeted by the PDR have their traffic blocked, whereas the other devices can still communicate. Traffic is blocked at the UPF incoming interface i.e., the upfgtp interface or the ens4 interface, depending on the direction of traffic.

7.4 Discussion

Suricata was able to detect all of the attacks in a reasonable time with the additional flooding rules. This means the system could be used to respond to such attacks in a timely manner, reducing the impact on availability. The rules would need to be adapted to the given network and the expected traffic.

The response action brings the average time to neutralize an attack to $\sim$50,45ms on average. For various types of attack this is an acceptable time. Flooding attacks that are stopped in $\sim$50,45ms are severely cut short, as many of these attacks are meant to intended to be active for hours or days. However, attack that are meant to complete faster than $\sim$50,45ms are not stopped in time by the system. The response action is still applied in this case, which could soften the impact of the completed attack by preventing the newly infected device to attack other devices. The response action also produces the expected result i.e., the isolation of the device.

The response action also works as expected, allowing for defining both internal and external IP address range to block traffic from. This means we could allow devices to still talk to a subset of external addresses, and we could specify which devices should be targeted with this rule.

Chapter 8

Conclusion

The following section contains some concluding words. Subsection 8.1 shortly summarizes the results of the project. Section 8.2 mentions some limitations of the project. Section 8.3 suggests some avenues for future investigations.

8.1 Summary

The project has found that the suggested monitoring approach works well and is able to detect attacks in a reasonable timeframe (∼11,32ms on average). Any attacks that take longer than this timeframe can be detected before completing and may be stopped. This means that many flooding and scanning attacks can be cut short and mitigated. However, any attacks that are shorter will complete before the monitoring system detects the attack, and will not be stopped in time. The system will still have an alert which indicates the attack has occurred, which might still be enough to prevent further propagation of the attack.

One of the responses using the 5G core network components (PDR/FAR change) was implemented and tested. The response successfully blocks the specified traffic and does so within an acceptable timeframe. The response is applied on average within ∼37,10ms from the moment the alert is generated, or ∼50,45ms of the triggering packet being sent. This once again means that attacks that take longer than that timeframe can be stopped, and shorter attacks are not stopped. However, the system will still apply the response action, which may simply mean that the compromised device is now isolated, and can not propagate an infection or attack.

In conclusion, the approach is feasible and may be suitable for the use case of protecting large-scale IoT deployments in a 5G network slice.

8.2 Limitations

There are some limitations to the system implementation. First, the mirroring of traffic to the monitoring virtual machine is done by using the Linux traffic control, instead of the functions built into the UPF. A 5G UPF has a mechanism that allows for duplicating traffic to another location. This is done by using PDRs and FARs. The FAR contains instructions for what action to take on an arriving packet matching the corresponding PDR. One of the options specified in the current standards is to duplicate the packet. This can happen in combination with some of the other options e.g., Forward. This means the packets is forwarded normally to the final destination and duplicated to the destination specified by the FAR. The reason this project did not make use of this capability is that it is not implemented in any of the open source UPFs or 5G cores. It may be interesting for future research to look into the differences between using the built-in capabilities and the Linux traffic control method.

The results also contained various outliers that are not completely explained. The Telnet bruteforce attack only being detected after several minutes would be a serious issue in a real implementation and should have been more closely investigated. The same goes for the outlier observed during the response action testing, where the SMF took much longer than usual to forward a request to the UPF. These outliers somewhat muddy the collected data, as they have serious impact on the observed average times and standard deviations. However, they may not be simply discarded as they may pose risks if these occur more often.

Another limitation is the use of simulated UEs and a simulated gNb. Using real 5G-capable IoT devices could be an interesting addition of realism to similar projects.

A final limitation is the fact that only a single response action was implemented during the project. The other response actions mentioned in 3.2.1 would have been an interesting addition to the testing repertoire. Especially the inter-slice change would have been interesting regarding the original focus on network slicing.

8.3 Future Work

The project has unearthed several interesting leads for future work. Mainly the various response actions that are technically possible to apply in the 5G core form an interesting avenue for future research. In addition, the question whether applying these actions would be the best course of action should also be answered, as it might be possible that simply running an IPS on the UPF is preferable in most cases.

It is also possible to run Suricata on the UPF virtual machine itself, either in IDS mode or IPS mode. For future research it may be interesting to compare this to running the monitoring system separately. Since Suricata scales rather well with throughput due to it being a multi-threaded IDS, this may not cause as much load on the machine as expected. Running Suricata in IPS mode may also be interesting, but this inevitably means a delay is added to each packet, which hinders low latency applications.

It may also be interesting to experiment with more custom slices. However, this may require some significant work, depending on the software used for the 5G components, as several of the open source candidates do not really support customising slices. Custom slices may be combined with various security functions and evaluated e.g., by seeing if custom latency constraints can still be satisfied when traffic is sent through and IPS, and finding where the boundaries for such solution lie.

Chapter 9

List of acronyms

3GPP	3rd Generation Partnership Project
4G	fourth generation
5G	fifth generation
AMF	access and mobility management function
AF	application function
BS	Base Station
DDoS	distributed denial of service
DoS	denial of service
DPI	deep packet inspection
FAR	forwarding action rule
gNb	gNodeB
GRE	generic routing encapsulation
GTP-U	GPRS tunneling protocol user data
IDS	intrusion detection system
IoT	Internet of Things
IPS	intrusion prevention system
LPWAN	Low-power wide-area network
MEC	Mobile Edge Computing

MitM	man-in-the-middle
MMTC	massive machine-type communication
MTTD	mean time to detection
NB-IoT	Narrowband IoT
NEF	network exposure function
NFV	network function virtualization
PCF	policy control function
PDR	packet detection rule
QER	QoS enforcement rule
QoS	quality of service
RAN	radio access network
SaaS	security-as-a-service
SDS	software-defined security
SDN	software-defined networking
SMF	session management function
UDM	unified data management function
UE	user equipment
UPF	user plane function
URLLC	ultra-reliable low latency communication
VNF	virtualized network function

Bibliography

[1] S. Wijethilaka and M. Liyanage, “Realizing internet of things with network slicing: Opportunities and challenges,” in *2021 IEEE 18th Annual Consumer Communications Networking Conference (CCNC)*, 2021, pp. 1–6.

[2] D. Candal-Ventureira, P. Fondo-Ferreiro, F. Gil-Castiñeira, and F. J. González-Castaño, “Quarantining malicious iot devices in intelligent sliced mobile networks,” *Sensors*, vol. 20, no. 18, 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/18/5054

[3] T. Trajanovski and N. Zhang, “An automated and comprehensive framework for iot botnet detection and analysis (iot-bda),” *IEEE Access*, vol. 9, p. 124360–124383, 2021. [Online]. Available: http://dx.doi.org/10.1109/ACCESS.2021.3110188

[4] V. Casola, A. De Benedictis, A. Riccio, D. Rivera, W. Mallouli, and E. M. de Oca, “A security monitoring system for internet of things,” *Internet of Things*, vol. 7, p. 100080, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2542660519300289

[5] S. S. Chawathe, “Monitoring iot networks for botnet activity,” in *2018 IEEE 17th International Symposium on Network Computing and Applications (NCA)*, 2018, pp. 1–8.

[6] M. Nobakht, C. Russell, W. Hu, and A. Seneviratne, “Iot-netsec: Policy-based iot network security using openflow,” in *2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, 2019, pp. 955–960.

[7] H. Sinanović and S. Mrdovic, “Analysis of mirai malicious software,” in *2017 25th International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, 2017, pp. 1–5.

[8] J. R. Rose, M. Swann, G. Bendiab, S. Shiaeles, and N. Kolokotronis, “Intrusion detection using network traffic profiling and machine learning for iot,” in *2021*

IEEE 7th International Conference on Network Softwarization (NetSoft), 2021, pp. 409–415.

[9] R. Hattarki, S. Houji, and M. Dhage, “Real time intrusion detection system for iot networks,” in *2021 6th International Conference for Convergence in Technology (I2CT)*, 2021, pp. 1–5.

[10] Y. Afek, A. Bremler-Barr, D. Hay, R. Goldschmidt, L. Shafir, G. Avraham, and A. Shalev, “Nfv-based iot security for home networks using mud,” in *NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium*, 2020, pp. 1–9.

[11] A. Sivanathan, “Iot behavioral monitoring via network traffic analysis,” 2020. [Online]. Available: https://arxiv.org/abs/2001.10632

[12] M. Ge, X. Fu, N. Syed, Z. Baig, G. Teo, and A. Robles-Kelly, “Deep learning-based intrusion detection for iot networks,” in *2019 IEEE 24th Pacific Rim International Symposium on Dependable Computing (PRDC)*, 2019, pp. 256–25 609.

[13] P. Matoušek, O. Ryšavý, and M. Grégr, “Security monitoring of iot communication using flows,” in *Proceedings of the 6th Conference on the Engineering of Computer Based Systems*, ser. ECBS ’19. New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: https://doi-org.ezproxy2.utwente.nl/10.1145/3352700.3352718

[14] T. D. Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan, and A.-R. Sadeghi, “DÏot: A federated self-learning anomaly detection system for iot,” in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, 2019, pp. 756–767.

[15] E. Anthi, L. Williams, M. Słowińska, G. Theodorakopoulos, and P. Burnap, “A supervised intrusion detection system for smart home iot devices,” *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 9042–9053, 2019.

[16] M. Hasan, M. M. Islam, M. I. I. Zarif, and M. Hashem, “Attack and anomaly detection in iot sensors in iot sites using machine learning approaches,” *Internet of Things*, vol. 7, p. 100059, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2542660519300241

[17] S. Chawla and G. Thamilarasu, “Security as a service: Real-time intrusion detection in internet of things,” in *Proceedings of the Fifth Cybersecurity Symposium*, ser. CyberSec ’18. New York, NY, USA:

Association for Computing Machinery, 2018. [Online]. Available: https://doi.org/10.1145/3212687.3212872

[18] D. He, S. Chan, X. Ni, and M. Guizani, "Software-defined-networking-enabled traffic anomaly detection and mitigation," *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 1890–1898, 2017.

[19] E. Hodo, X. Bellekens, A. Hamilton, P.-L. Dubouilh, E. Iorkyase, C. Tachtatzis, and R. Atkinson, "Threat analysis of iot networks using artificial neural network intrusion detection system," in *2016 International Symposium on Networks, Computers and Communications (ISNCC)*, 2016, pp. 1–6.

[20] D. Ageyev, T. Radivilova, and O. Mohammed, "Traffic monitoring and abnormality detection methods analysis," in *2020 IEEE International Conference on Problems of Infocommunications. Science and Technology (PIC S T)*, 2020, pp. 823–826.

[21] D. Ageyev, T. Radivilova, O. Bondarenko, and O. Mohammed, "Traffic monitoring and abnormality detection methods for iot," in *2021 IEEE 4th International Conference on Advanced Information and Communication Technologies (AICT)*, 2021, pp. 250–254.

[22] A. Jain, T. Sing, S. K. Sharma, and V. Prajapati, "Implementing security in iot ecosystem using 5g network slicing and pattern matched intrusion detection system: A simulation study," *Interdisciplinary Journal of Information, Knowledge, and Management*, vol. 16, pp. 1–38, 2021.

[23] A. Jain, T. Sing, and S. K. Sharma, "Security as a solution: An intrusion detection system using a neural network for iot enabled healthcare ecosystem," *Interdisciplinary Journal of Information, Knowledge, and Management*, vol. 16, pp. 331–369, 2021.

[24] A. Molina Zarca, J. Bernal Bernabe, I. Farris, Y. Khettab, T. Taleb, and A. Skarmeta, "Enhancing iot security through network softwarization and virtual security appliances," *International Journal of Network Management*, vol. 28, no. 5, p. e2038, 2018, e2038 nem.2038. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/nem.2038

[25] A. Molina Zarca, M. Bagaa, J. Bernal Bernabe, T. Taleb, and A. F. Skarmeta, "Semantic-aware security orchestration in sdn/nfv-enabled iot systems," *Sensors*, vol. 20, no. 13, 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/13/3622

[26] I. Farris, J. B. Bernabe, N. Toumi, D. Garcia-Carrillo, T. Taleb, A. Skarmeta, and B. Sahlin, “Towards provisioning of sdn/nfv-based security enablers for integrated protection of iot systems,” in *2017 IEEE Conference on Standards for Communications and Networking (CSCN)*, 2017, pp. 169–174.

[27] “5g and iot in 2022.” [Online]. Available: https://www.thalesgroup.com/en/markets/digital-identity-and-security/iot/resources/innovation-technology/5G-iot

[28] C. Patrik, L. Anette, J. Peter, C. Stephen, M. Richard, S. Rachit, L. Dave, and C. Ilyas, “Ericsson Mobility Report,” Ericsson Group, Tech. Rep., 2021.

[29] P. Rost, C. Mannweiler, D. S. Michalopoulos, C. Sartori, V. Sciancalepore, N. Sastry, O. Holland, S. Tayade, B. Han, D. Bega, D. Aziz, and H. Bakker, “Network slicing to enable scalability and flexibility in 5g mobile networks,” *IEEE Communications Magazine*, vol. 55, no. 5, pp. 72–79, 2017.

[30] Y. Khettab, M. Bagaa, D. L. C. Dutra, T. Taleb, and N. Toumi, “Virtual security as a service for 5g verticals,” in *2018 IEEE Wireless Communications and Networking Conference (WCNC)*, 2018, pp. 1–6.

[31] P. Ranaweera, V. N. Imrith, M. Liyanag, and A. D. Jurcut, “Security as a service platform leveraging multi-access edge computing infrastructure provisions,” in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, 2020, pp. 1–6.

[32] B. Chafika, T. Taleb, C.-T. Phan, C. Tselios, and G. Tsolis, “Distributed ai-based security for massive numbers of network slices in 5g amp; beyond mobile systems,” in *2021 Joint European Conference on Networks and Communications 6G Summit (EuCNC/6G Summit)*, 2021, pp. 401–406.

[33] I. Pupo, A. Santoyo-Gonzalez, and C. Cervelló-Pastor, “A framework for the joint placement of edge service infrastructure and user plane functions for 5g,” *Sensors*, vol. 19, 09 2019.

[34] F. Z. Yousaf, M. Bredel, S. Schaller, and F. Schneider, “Nfv and sdn—key technology enablers for 5g networks,” *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 11, pp. 2468–2478, 2017.

[35] “An introduction to network slicing,” GSM Association, 2020. [Online]. Available: https://www.gsma.com/futurenetworks/resources/an-introduction-to-network-slicing-2/

[36] S. D’Oro, F. Restuccia, T. Melodia, and S. Palazzo, “Low-complexity distributed radio access network slicing: Algorithms and experimental results,” 2018. [Online]. Available: https://arxiv.org/abs/1803.07586

[37] A. Nakao, P. Du, Y. Kiriha, F. Granelli, A. A. Gebremariam, T. Taleb, and M. Bagaa, “End-to-end network slicing for 5g mobile networks,” *Journal of Information Processing*, vol. 25, pp. 153–163, 2017.

[38] J. Ordonez-Lucena, P. Ameigeiras, D. Lopez, J. J. Ramos-Munoz, J. Lorca, and J. Folgueira, “Network slicing for 5g with sdn/nfv: Concepts, architectures, and challenges,” *IEEE Communications Magazine*, vol. 55, no. 5, pp. 80–87, 2017.

[39] A. Mathew, “Network slicing in 5g and the security concerns,” in *2020 Fourth International Conference on Computing Methodologies and Communication (ICCMC)*, 2020, pp. 75–78.

[40] Palo Alto Networks, Inc., “Mobile network infrastructure getting started,” pp. 84–87, 2021.

[41] J. Cáceres-Hidalgo and D. Avila-Pesantez, “Cybersecurity study in 5g network slicing technology: A systematic mapping review,” in *2021 IEEE Fifth Ecuador Technical Chapters Meeting (ETCM)*, 2021, pp. 1–6.

[42] J. Lam and R. Abbas, “Machine learning based anomaly detection for 5g networks,” 2020. [Online]. Available: https://arxiv.org/abs/2003.03474

[43] D. Sattar and A. Matrawy, “Towards secure slicing: Using slice isolation to mitigate ddos attacks on 5g core network slices,” in *2019 IEEE Conference on Communications and Network Security (CNS)*, 2019, pp. 82–90.

[44] R. M. Van Leeuwen, “Cyber-attack containment through actionable awareness,” Master’s thesis, Technical University of Eindhoven, May 2022.

[45] D. Midi, S. Sultana, and E. Bertino, “A system for response and prevention of security incidents in wireless sensor networks,” *ACM Trans. Sen. Netw.*, vol. 13, no. 1, dec 2016. [Online]. Available: https://doi-org.tudelft.idm.oclc.org/10.1145/2996195

[46] I. Hafeez, M. Antikainen, A. Y. Ding, and S. Tarkoma, “Iot-keeper: Detecting malicious iot network activity using online traffic analysis at the edge,” *IEEE Transactions on Network and Service Management*, vol. 17, no. 1, pp. 45–59, 2020.

[47] K. Mishima, T. Nemoto, Y. Hagiwara, and T. Tsujisawa, "First year's actual operational results of efficient security measure system with automatic isolation in tuat," in *Proceedings of the 2019 ACM SIGUCCS Annual Conference*, ser. SIGUCCS '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 121–124. [Online]. Available: https://doi-org.ezproxy2.utwente.nl/10.1145/3347709.3347809

[48] 3GPP, "5g; system architecture for the 5g system (5gs)," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 123.501, 03 2022, version 15.13.0. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/123500_123599/123501/15.13.00_60/

[49] 3GPP, "LTE; 5G; Interface between the Control Plane and the User Plane nodes," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 129.244, 11 2020, version 15.10.0. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/129200_129299/129244/15.10.00_60/

[50] 3GPP, "5G; Procedures for the 5G System (5GS)," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 123.502, 07 2020, version 15.16.0. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/123500_123599/123502/15.16.00_60/

[51] 3GPP, "5g; 5g system; policy and charging control signalling flows and qos parameter mapping; stage 3," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 129.513, 04 2021, version 15.10.0. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/129500_129599/129513/15.10.00_60/

[52] 3GPP, "5g; 5g system; policy authorization service; stage 3," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 129.514, 09 2022, version 15.13.0. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/129500_129599/129514/15.13.00_60/

[53] 3GPP, "5g; 5g system; unified data management services; stage 3," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 129.503, 01 2022, version 15.10.0. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/129500_129599/129503/15.10.00_60/

[54] M. M. Sajjad, C. J. Bernardos, D. Jayalath, and Y. Tian, "Inter-slice mobility management in 5g: Motivations, standard principles, challenges and research directions," *CoRR*, vol. abs/2003.11343, 2020. [Online]. Available: https://arxiv.org/abs/2003.11343

[55] H. Kang, D. H. Ahn, G. M. Lee, J. D. Yoo, K. H. Park, and H. K. Kim, “Iot network intrusion dataset,” 2019. [Online]. Available: https://dx.doi.org/10.21227/q70p-q449