Technology
Arts Sciences
TH Köln

5G-FORAN
DFIR IN OPEN RAN

Evaluation of Dashboard Techniques for Digital Forensics and Incident Response (DFIR) in Open RAN

Bericht zum Forschungsprojekt
im Studiengang *Master of Science* Technische Informatik
an der Fakultät für Informations-, Medien- und Elektrotechnik
der Technischen Hochschule Köln

Vorgelegt von: Jonas Levin Weber
Matrikelnummer: 11134601
E-Mail-Adresse: jonas_levin.weber@smail.th-koeln.de

Eingereicht bei: Prof. Dr. Andreas Grebe

Köln, 10. Januar 2024

Abstract

The 5G-FORAN project is a collaboration between the University of Applied Sciences (TH) Cologne and PROCYDE GmbH. The project is supported by the Federal Office for Information Security and aims to address security challenges in Open Radio Access Networks (RANs). This report highlights the project's focus on developing methods for analyzing and resolving cybersecurity incidents within Open RAN. The project encompasses two sub-projects: 5G-FORAN-ATTACK, which evaluates and simulates possible attack scenarios, and 5G-FORAN-DFIR, focusing on attack detection and mitigation.

One outcome of this project is the development of a dashboard for visualizing attack data that is generated in 5G-FORAN-ATTACK. This report details the design and implementation of the prototypical dashboard, which leverages the ApexCharts.js library for visualization. The dashboard is a web application, designed following the Goal Question Metric (GQM) model and push and pull principles. The dashboard is implemented using Go, the Gin web framework, and front-end technologies like Bootstrap, Turbo, and Stimulus.

The report discusses the deployment and the first features of the dashboard as well as its performance with the current attack data volume. It also addresses challenges and proposes enhancements like filtering options and live updates for better data management and visualization. The report concludes with an acknowledgment of the ongoing nature of the dashboard's development, underscoring the need for further enhancements and the potential of the dashboard in comparing the effectiveness of different attack scenarios and defensive measures in Open RAN environments.

Contents

List of Figures

List of Tables

List of Abbreviations

Notation	Description
3GPP	3rd Generation Partnership Project
API	Application Programming Interface
BSI	Federal Office for Information Security
BSON	Binary JSON
CSIRT	Computer Security Incident Response Team
CSS	Cascading Style Sheets
CVE	Common Vulnerabilities and Exposures
DF	Digital Forensics
DFIR	Digital Forensics and Incident Response
DOM	Document Object Model
EDR	Endpoint Detection and Response
FiGHT	5G Hierarchy of Threats
GQM	Goal Question Metric
GUI	Graphical User Interface
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IR	Incident Response
IRP	Incident Response Plan
JSON	JavaScript Object Notation
Near-RT RIC	Near-Real-Time RAN Intelligent Controller
NF	Network Function
Non-RT RIC	Non-Real-Time RAN Intelligent Controller
O-RU	O-RAN Radio Unit
OSS	Open-Source Software
RAN	Radio Access Network
RU	Radio Unit

Notation	Description
SIEM	Security Information and Event Management
SOAR	Security Orchestration, Automation, and Response
SQL	Structured Query Language
SSE	Server-Sent Events
SVG	Scalable Vector Graphics
TH	University of Applied Sciences
UE	User Equipment
VPN	Virtual Private Network
WG1	Work Group 1: Use Cases and Overall Architecture Work Group
WYSIWYG	What You See Is What You Get
XDR	Extended Detection and Response

1. Introduction

The opening of the RAN is a major change in the mobile network industry. It enables new players to enter the market and to develop new products and services. However, the opening of the RAN also introduces new security challenges. To address these challenges, the 5G-FORAN project was initiated to develop a method for the analysis and resolving of cyber security incidents in the context of Open RAN.

1.1. Background

This report is part of the 5G-FORAN project [1] at TH Cologne [2]. The 5G-FORAN project is a research project that is executed in cooperation of TH Cologne with PROCYDE GmbH [3] and funding from the Federal Office for Information Security (BSI) [4]. The aim of the project is to develop a method for the analysis and resolving of cyber security incidents in the context of Open RAN.
Two sub-projects are part of the 5G-FORAN project: 5G-FORAN-ATTACK and 5G-FORAN-DFIR. While 5G-FORAN-ATTACK is concerned with evaluating possible attack scenarios and simulating them, 5G-FORAN-DFIR is concerned with the detection and mitigation of the attacks. As the two sub-projects are closely related and executed in collaboration, the results of this report are relevant for both sub-projects.
The attack simulation in 5G-FORAN-ATTACK is carried out in a test environment that is based on an Open RAN implementation in a Kubernetes cluster as specified by the O-RAN ALLIANCE [5] and provided by the O-RAN Software Community [6]. The attacks are then executed using a tool called Clusterforce that is developed in 5G-FORAN-ATTACK. Clusterforce is a wrapper around open source tools for network analysis and vulnerability detection like nmap [7] and Stratus Red Team [8] as well as Kubernetes and cloud specific security scanners like kube-hunter [9] and Trivy [10].

1.2. Scope

To assist in the evaluation of attack scenarios on Open RAN, a dashboard is developed that visualizes the attacks both for the 5G-FORAN-ATTACK and the 5G-FORAN-DFIR sub-project stakeholders. This report covers the design ideas and the implementation of a prototypical dashboard for 5G-FORAN. The dashboard and the work in this report are only concerned with visualizing the existing attack data that is generated in the 5G-FORAN-ATTACK sub-project. It is not concerned with generating new attack data or detecting attacks as it is done in the 5G-FORAN-DFIR sub-project. It is also not intended that the dashboard development is finished after this report is finished. Instead, the development of the dashboard is an ongoing process and further work will be done in the future that will be based on the ideas presented in this report.

1.3. Structure of the Report

This report first introduces relevant theoretical concepts and general principles of dashboard design. The following chapters then gather the requirements for the dashboard and analyze available technologies for the implementation. After that, design ideas for the dashboard are presented and the prototypical implementation is explained. The report is concluded with a discussion and an outlook on future work.

2. Theoretical Framework

This chapter covers theoretical aspects of O-RAN, DFIR and dashboards. Each section defines one of these terms and describes it.

2.1. Open RAN

The RAN in a mobile network is the interface between User Equipment (UE) like mobile phones, computers and other remotely controlled machines and the mobile network's core [11].
Open RAN is an effort to "open the RAN" [12]. While traditional RAN solutions are developed by a single vendor that provides the entire required hardware stack, Open RAN is aimed to become an open standard for the RAN that vendors can implement. The goal is to create an incentive for competition in the mobile operators market as well as to reduce costs for equipment. Additionally, the Open RAN standard is designed with modularity in mind which yields the possibility of only having to replace one or few components of the RAN instead of the entire hardware stack should requirements change.
In 2018, the O-RAN ALLIANCE was founded by leading mobile network operators with the goal of defining a specification for Open RAN [5]. In this report, Open RAN will from now on refer to the general idea of opening the RAN while O-RAN will refer to the specification of an Open RAN as defined by the O-RAN ALLIANCE.
The O-RAN ALLIANCE is constituted of 11 work groups, each with a specific purpose. The results of the work of each work group are publicly downloadable [13]. One of the work groups is Work Group 1: Use Cases and Overall Architecture Work Group (WG1) which is responsible for identifying O-RAN use cases and for specifying the O-RAN architecture [14]. A high-level overview of the architecture as specified by WG1 in October 2023 is depicted in Figure 2.1.

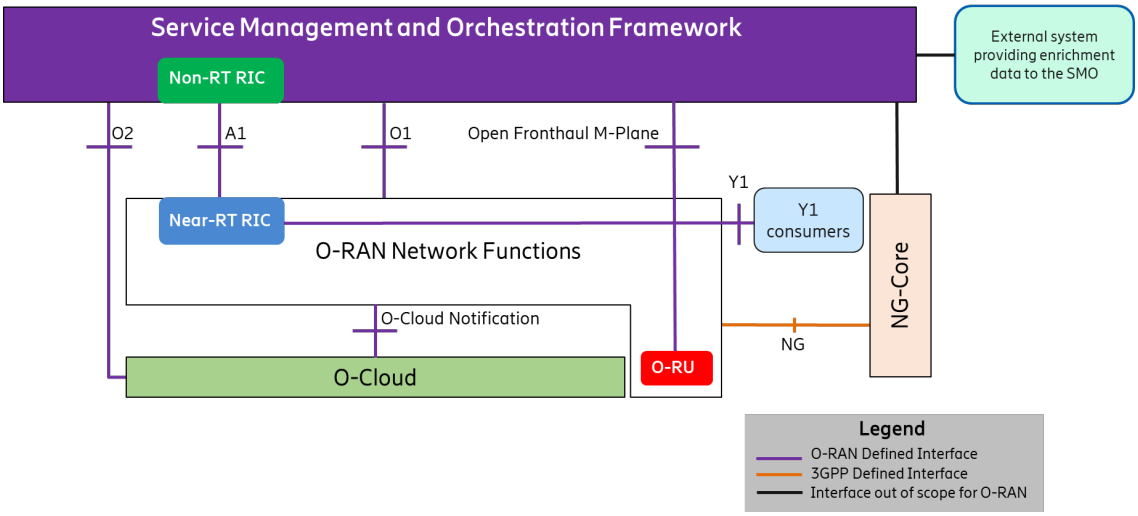

Figure 2.1.: High-level architecture of O-RAN as specified by WG1 of the O-RAN ALLIANCE. Taken from [15].

As seen in the figure, the O-RAN architecture consists of a set of components and interfaces. The components include the Non-Real-Time RAN Intelligent Controller (Non-RT RIC), the Near-Real-Time RAN Intelligent Controller (Near-RT RIC), and the O-RAN Radio Unit (O-RU). The architecture diagram also shows the NG-Core which is the core of the network and not part of O-RAN. Instead, the NG-Core is specified by the 3rd Generation Partnership Project (3GPP) [16] and the RAN merely connects the core with the UE.

Beginning with the O-RU, it is the component of O-RAN that communicates with the UE [15]. In contrast to the Radio Unit (RU) from traditional RAN solutions, O-RU is the O-RAN compatible equivalent.

The Non-RT RIC and Near-RT RIC are responsible for managing the RAN components and resources [15]. The Non-RT RIC's goal is the optimization of the RAN including long-term planning which is achieved by so-called rApps. These applications collect data from the RAN, analyze it, and send instructions to the different RAN components including the O-Cloud which in turn can allocate resources for the RAN. The Non-RT RIC operates within latencies between 10ms and 1s and can utilize long-term data and machine learning models for its decisions.

The Near-RT RIC operates with latencies of less than 10ms and is responsible for controlling the RAN components and resources while adhering to the policies set by the Non-RT RIC that it received over the A1 interface [15]. It can handle short-term optimization tasks such as load balancing or interference management and additionally publishes RAN analytics data via the Y1 interface.

Finally, the O-Cloud is the cloud infrastructure that is responsible for executing the O-RAN Network Functions (NFs) including the Near-RT RIC and Non-RT RIC [15]. It consists of storage, computing, and networking resources that are used by the O-RAN NFs.

2.2. DFIR

Digital Forensics and Incident Response (DFIR) is the union of two concepts: Digital Forensics (DF) and Incident Response (IR) [17]. While DF is concerned with the collection, examination, analysis, and reporting of data that is considered digital evidence, IR is concerned with the preparation, detection and analysis, containment, eradication, recovery, and post-incident review of digital security incidents.

2.2.1. Digital Forensics

The main aspects of DF are the identification and understanding of cyber security incidents. This is realized by following a four step procedure as specified by Kent, Chevalier, Grance, *et al.* [18]: Data collection, examination, analysis, and reporting. The data collection step consists of the identification of relevant data sources and their drainage. Relevant data sources can be file systems, operating systems, memory, network traffic, and applications. The next step is the examination of the collected data which is concerned with the recognition of anomalies and other forensic artifacts. The third step, analysis, involves the interpretation of the examined data and the correlation of the data with previous incidents or other, possibly public, data sources. The last step is the reporting of the analysis results. The report should contain the culprit, if it could be identified, as well as recommendations for future prevention of similar incidents.

2.2.2. Incident Response

Similar to DF, IR is the process of preventing, detecting, and reacting to cyber security incidents [17]. Given an Incident Response Plan (IRP), a Computer Security Incident Response Team (CSIRT) can respond to security incidents in a structured and consistent way. These IRPs are typically created using a six step process: Preparation, detection and analysis, containment, eradication, recovery, and post-incident review. The preparation step is concerned with the identification of risks in the organization and the drafting of IRPs. The next step is very similar to DF in that it is about the monitoring of

forensically relevant data and the filtering thereof. In the containment step, the detected incidents are attempted to be stopped from spreading through the organization's network before they can cause further damage. The eradication step is then the evolution of the containment step with the goal of removing the root cause of the incident. After an incident has been eradicated, the CSIRT can begin recovering the affected systems and undertake data restoration. The final step of the IR process is the post-incident review. In this step, the CSIRT tries to understand how the incident could happen and how it can be mitigated in the future.

2.2.3. DFIR Tools

There are many tools available for DFIR purposes. These tools can be sorted into four categories:

SIEM Security Information and Event Management tools are used to collect and analyze data from a variety of sources in order to detect and respond to security incidents. Originally, Security Information and Event Management (SIEM) tools were log aggregation tools but they have since evolved to include more features like automatic threat detection and IR.

SOAR Security Orchestration, Automation, and Response tools allow the combination of different security related tools into automated workflows that can be triggered by certain events.

EDR Endpoint Detection and Response is concerned with the protection of endpoints like computers, mobile phones, and devices like printers even beyond barriers like antivirus software. Endpoint Detection and Response (EDR) tools collect data from the endpoints in a continuous fashion and analyze these as well as initiate responsive actions to detected threats.

XDR Extended Detection and Response is an expansion of EDR that deals with the collection and analysis of data from more sources than EDR such as cloud services, servers, networks, and even SIEM tools.

2.3. Understanding Dashboards

The term dashboard has evolved in its usage over time. According to Merriam-Webster [19], it has the three following definitions:

The term first appeared in 1842 where it described a screen of wood at the front of a horse carriage that had the purpose of protecting the driver from mud, snow or water that was flung up by the horse's hooves. In more modern times dashboards usually refer to the panels below the windshield of vehicles like cars that host gauges and instruments essential or helpful for the operation of the vehicle.
In the context of this report, however, the third definition is most pertinent: Dashboards are a collection of graphical representations of data that are relevant to a specific group of people. In the context of the internet and websites, dashboards also often provide interactivity such as for sorting, filtering or exporting reports of the presented data. This third definition is used for the remainder of this work.
The primary purpose of a dashboard is to provide quick insights into data that enable informed decision making on the user's side. Typically, this is done by combining data sources in a senseful way and visualizing the results using an appropriate visualization technique that delivers the required information in a clear way.
One example of such a visualization technique is a gauge that visualizes a single value in relation to a scale. This is used in cars to inform the driver about the current speed of the vehicle in comparison to the maximum speed of the vehicle and standstill.

Figure 2.2.: Gauge visualization. Taken from [20].

Figure 2.2 shows an exemplary gauge visualization as it may be used on a digital dashboard showing the progress of some process. Except for the rotation of the gauge, the visualization is very similar to the gauges present in cars, proving their origin.

3. Dashboard Design Principles

This chapter explains general considerations for the design of dashboards which is supported by several principles. These principles are the GQM model, the push and pull principles and the data-ink ratio. The second part of the chapter enumerates special considerations that have to be made when designing a dashboard for the attack visualization in O-RAN.

3.1. Fundamentals of Dashboard Design

The principles of dashboard design are based on the fact that the human brain mainly processes visual information. Additionally, the human brain is tailored to recognize patterns without having to focus on the entirety of information which is called pre-attentive processing. In fact, humans process all information pre-attentively [21]. Therefore dashboards are most effective if they display the most important information in a simple to recognize pattern that the viewer of the dashboard can grasp instinctively.

3.2. Data Considerations and the GQM Model

A key aspect of dashboard design is the consideration of what data should be displayed on the dashboard and in what way. Basili, Caldiera, and Rombach [22] propose the GQM model as an approach to this problem. GQM targets the goal of software quality improvement but can be applied to the design of dashboards as well. The three components of GQM work as follows:

Goal Goals are defined for some object for which measurements can be made. The goals are defined from various viewpoints and with respect to different environments. Each goal has a purpose, an issue, an object or process and a viewpoint.

Question Given a goal, questions are defined that can be answered by measurements. Questions are a means of assessing the defined goals.

Metric Metrics are defined as the collection and processing of available data to answer the defined questions.

The GQM model can be thought of as a tree structure with the goal at the root, questions as branches of the goal and metrics as branches of the questions as displayed in Figure 3.1.

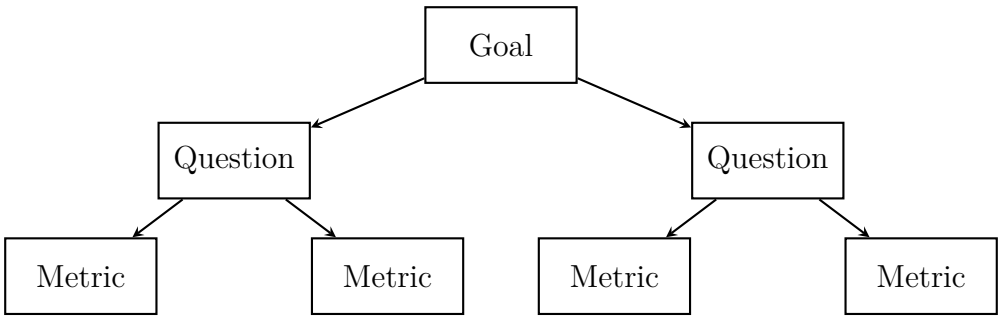

Figure 3.1.: The GQM model [22].

An example in the context of DFIR in O-RAN is the following:

Goal

Purpose Improve

Issue the effectiveness of attacks

Object on the Non-real time RIC

Viewpoint from the perspective of the attacker.

Question How many attacks were successful?

Metric The number of attacks with exit code zero in relation to the number of attacks with a non-zero exit code.

By taking the goals, questions, and metrics into account, the dashboard can be designed in a way that uses the appropriate data from the data source(s) and displays it using a suitable visualization technique that transports the required information. Taking the example from before, the data that is required to answer the question is a collection of attacks each populated with at least an exit code. A suitable visualization is a line chart that displays the progression of the ratio of the number of attacks with exit code zero and the attacks with a non-zero exit code over time. Figure 3.2 shows this exemplary visualization.

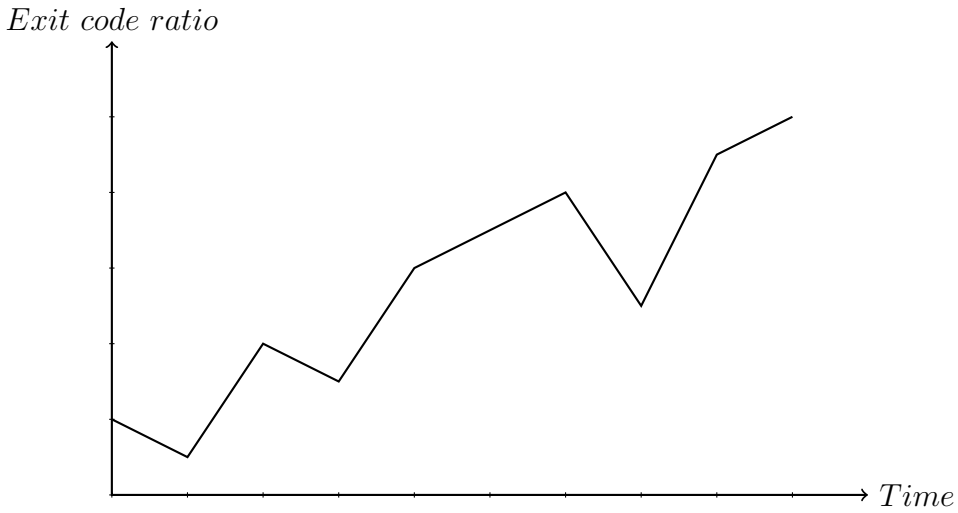

Figure 3.2.: Example visualization of the example GQM question.

As seen in Figure 3.2, the exit code ratio evolution over time can be grasped intuitively and answer the question of how many attacks were successful. Additionally, the visualization shows a trend of an ever increasing exit code ratio which can be used as an indicator for the initial goal.

3.3. Push and Pull Principles

Another important aspect of dashboard design is the understanding of the difference between push and pull principles [23]. Push describes a scheme where the data is pushed to the user without the user having to interact with the dashboard. In contrast, pull describes the scenario where a user actively interacts with the dashboard in order to gather the information they need. The push principle has the advantage of being intuitive and being able to notify the user in case of unforeseen circumstances whereas the pull principle is more suitable for detailed information gathering and discovery of new connections in the data. Each principle can therefore be used in different scenarios which is dependent on the specific requirements each time. Also, a dashboard can certainly contain both push and pull elements if this is determined to be a fitting solution.

3.4. Data-Ink Ratio

Proper visualization of data not only includes choosing a suitable visualization technique but also finding a way of displaying the data on the chosen technique that does not

distract the viewer from the information that shall be transported. Tufte presents five principles for designing visualizations that take this into consideration [24, p. 105]:

1. Above all else, show the data.
2. Maximize the data-ink ratio.
3. Erase non-data-ink.
4. Erase redundant data-ink.
5. Revise and edit.

Tufte defines the data-ink ratio as the quotient of ink used to display the data and the total amount of ink used in the visualization. In more modern times, the term ink can be replaced by pixel as the majority of visualizations is produced digitally. The five principles give a guideline for designing visualizations that have a clear intent and are not cluttered with unnecessary information. It is to be noted, though, that Tufte introduces the principles "maximize the data-ink ratio", "erase non-data ink", and "erase redundant data-ink" all with the relativization "within reason" [24, pp. 96, 100]. Strictly applying all of Tufte's principles would yield visualizations that are too minimalistic and not intuitive to understand anymore.

4. Dashboard Requirements

While the previous chapter introduced general dashboard design principles, this chapter outlines the specific requirements for the dashboard of the 5G-FORAN project. This includes the available data and the suited visualization types.

4.1. General Requirements

Some general requirements have to be kept in mind when choosing a dashboard technology. These requirements include that the technology has to be free such that no license fees have to be paid. Additionally, the technology has to be open source which enables the addition of custom features should the technology not provide them out of the box or provide a way to add them through a plugin system. For now, the dashboard has to work with the available data from the 5G-FORAN-ATTACK sub-project which is stored in a MongoDB database. Therefore, the dashboard technology must at least be able to connect to a MongoDB database.

4.2. Data Considerations

The ATTACK part of the 5G-FORAN project is concerned with the launch of attacks on the DFIR part of the project. These attacks produces a large amount of data about the attacks including the output of each tool execution. See Table A.1 in the appendix for an overview of the produced data.
Potentially all data is useful for the dashboard. However, not all data is collected for every attack. The Common Vulnerabilities and Exposures (CVE) field, for example, currently is only populated if the tool directly reports a CVE. If the CVE can only be inferred in an additional, possibly manual step, the CVE field is left empty. The same holds true for the `tactic`, `mitre_technique` and `microsoft_technique` fields.
Some data is useful for filtering the attacks. These include the `timestamp_start`, `timestamp_end`, `oran_component`, `cve`, `tactic`, `mitre_technique`, and `microsoft_-`

`technique`. The timestamp fields can be used to specify a time range for which attacks should be displayed on the dashboard and the other mentioned fields can be used to further reduce the number of attacks that are displayed.
The ATTACK part of the 5G-FORAN project defines the term campaign which is a set of attacks that are executed in a specific order. These campaigns are also stored in the database and have to be visualized on the dashboard.

4.3. Visualization Techniques

Of the classic visualization techniques, a well suited one for the visualization of the attack data are tables. Tables can display many data records at once and allow the initial investigation of important aspects of the data. In order to reduce the number of displayed records to an interesting set of records, tables can be filtered by the fields mentioned in the previous section. Additionally, tables can be sorted by the values of a specific column which allows the user to quickly find the records that lie within the lower or upper bounds of the range of a specific field. For a more detailed analysis, the user can then select the desired record and be presented with a more detailed view of the record.
Since the attack data is time dependent, a time series visualization is also well suited. This can be a line chart or a bar chart that shows the progression of a specific field over time in a continuous or discrete manner, respectively. This allows the user to quickly identify trends in the data as well as peaks and drops in the data that might be interesting to investigate further.
For the specific case of the 5G-FORAN project, a map visualization can be considered. Since the attacks are executed in a Kubernetes cluster, the cluster can be visualized as a map or network graph. The nodes of the graph can represent the different components of the cluster and the edges can represent the connections between the components that were detected during the attack. Since the ATTACK part of the 5G-FORAN project has full control over the Kubernetes cluster, it can also be visualized using the available information about the cluster. One way of doing this is to embed the Kubernetes Dashboard [25] that is provided by the Kubernetes project.

5. Technology Review

Different technologies for dashboard development were researched and compared. This chapter presents the available dashboard development technologies and compares them considering the requirements defined earlier.

5.1. Available Technologies for Dashboard Development

The available dashboard technologies fall into two categories. The first category is the category of technologies that provide a complete dashboard solution. These technologies allow the user to connect to different data sources and build their dashboard using a graphical user interface.
The second category contains technologies that provide a framework for dashboard development. These technologies contain an Application Programming Interface (API) that can be used programmatically to build a dashboard from a predefined set of visualization techniques and components. It is possible or even required to bring your own data source to these technologies since the framework is not concerned with connecting to any data source. Some of the category two technologies also provide a plugin system that allows the user to extend the framework with custom components. However, if the framework is Open-Source Software (OSS), it is possible to extend the framework by forking the repository and adding the custom components to the fork.
In the following analysis, two technologies from the first category and three technologies from the second category are compared.

5.2. Comparative Analysis

Table 5.1 shows a comparison of the available technologies. For each technology, the table shows whether the technology can be used without any license fees, whether it supports MongoDB as a data source out of the box, what methods of querying the data

are available, and whether the development of custom components and visualizations is supported.

Table 5.1.: Comparison of available dashboard technologies.

Name	Entirely free	MongoDB	Query language	Plugins
Grafana [26]	✗	Only paid	SQL, GUI	✓
Metabase [27]	✗	✓	SQL, GUI	✗
Dash [28]	✗	✗	custom	✓(OSS)
D3.js [29]	✓	✗	custom	✓(OSS)
ApexCharts.js [30]	✓	✗	custom	✓(OSS)

Since one of the requirements of the dashboard is that it has to use free software, Grafana is not a suitable technology because the MongoDB data source is only available as a plugin for Grafana Enterprise which is a paid version of Grafana [31].
Although Metabase provides a MongoDB data source out of the box and is free to use, the features available are too limited for the use case of DFIR in O-RAN. Metabase is a business intelligence tool that enables users to create dashboards and visualizations from their data. However, the visualizations are limited to a few types like bar charts, line charts, and pie charts. The querying of data is also limited as it is implemented using a What You See Is What You Get (WYSIWYG) approach that might fall short for more complex queries.
Dash is a Python framework for building so-called data applications, which dashboards are a subset of. A Dash application is a web application that is entirely built in Python. This includes the Hypertext Markup Language (HTML) code which is generated by Dash instead of being written by the developer directly into HTML files. This means that Dash is not only concerned with the visualization of data but also with the building of the web application itself. While this might be an advantage for smaller and simpler applications, it is a disadvantage for more complex scenarios. Dash is thus not suitable for the dashboard of the DFIR use case.
Finally, D3.js and ApexCharts.js are both JavaScript libraries that provide a framework for building visualizations. Both libraries are OSS and free to use. They do not provide any functionality for connecting to a data source but instead provide an API for building visualizations from the data that the developer has to provide. Since both libraries provide a wide variety of visualization types and the possibility to build custom visualizations, they are suitable for the DFIR dashboard.

5.3. Recommended Technology

D3.js is more flexible than ApexCharts.js because it provides a lower-level API that allows the developer to build custom visualizations from scratch and hook into the rendering process. This flexibility comes at the cost of a steeper learning curve and more development effort. ApexCharts.js, on the other hand, provides a higher-level API that still enables a great degree of customization. For the use case of the DFIR dashboard, the higher-level API of ApexCharts.js is sufficient while the more complex features of D3.js are not needed. ApexCharts.js provides a good balance between flexibility and ease of use. Therefore, ApexCharts.js is the recommended technology for the DFIR dashboard.

6. Dashboard Design

This chapter shows design proposals for the dashboard. The designs are based on the design principles and requirements defined earlier as well as existing DFIR dashboards that are used on the defending site.

6.1. Design Methodology

The dashboard is designed with some general constraints in mind. First of all, the dashboard is intended to be used from the user's web browser. This is done to remove a number of factors that slow down the development process for native or hybrid applications like cross-platform compatibility, installation processes, software dependencies or hardware requirements on the end-user's site. These are all requirements that a web application is often not concerned with as a working internet browser on the user's system already covers them.
A common disadvantage of web applications, however, is the constant need for an active internet connection. The dashboard does not require such a connection per se but only a working connection to the server that hosts the dashboard which can be on the local network or on a Virtual Private Network (VPN).
The dashboard is aimed to be useful on a screen with a resolution of at least 1920 by 1080 pixels which has been the most common resolution for desktop computers since October 2020 [32]. Mobile devices and responsive design are not considered in the process as the users of the dashboard are expected to use desktop computers or laptops with sufficiently large screens.
Regarding the usability of the dashboard, the primary views of the dashboard follow the push principle that was presented in Section 3.3. This way, the user gets an overview of the current state of the attacks and campaigns right away without having to interact with the dashboard. In a next step, the user is able to investigate specific details of each attack, each campaign, and other collection of attacks like all attacks that were performed using a specific tool or all attacks that targeted a specific component of the

O-RAN system. Therefore, the dashboard follows the pull principle to implement these more detailed views.

6.2. Proposed Features

In the following sections, the proposed features of the dashboard are presented. These features are mostly visualizations of the attack data that is stored in the database.

6.2.1. Timeline

To gain insight into what attacks have been performed in a given time frame and how much time each attack took to finish, the dashboard could display visualizations that are based on Gantt charts. Gantt charts are mainly used in project management to show the relation between tasks in a project as well as the time frame in which the tasks are planned to be processed [33, pp. 49-50]. In its essence, a Gantt chart is a modified bar chart that shows time on the x-axis and tasks on the y-axis. Each task is represented by a horizontal bar that spans the time frame of the task. In more advanced Gantt charts, the bars are additionally connected to show dependencies between tasks or the bars are colored to indicate critical tasks. Figure 6.1 shows an example of a timeline visualization that is based on the principles of Gantt charts.

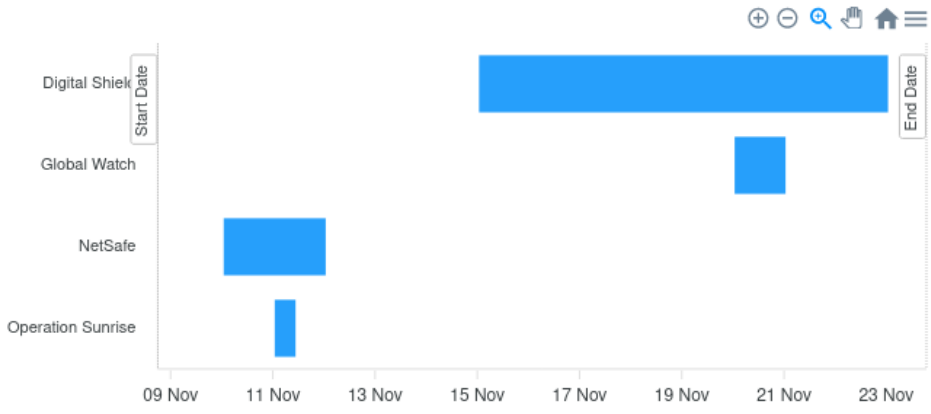

Figure 6.1.: Example timeline chart.

The figure shows four test campaigns that were run in November 2023. Each campaign is represented by a horizontal bar that spans the time frame of the campaign. The y-axis shows the names of the campaigns.

6.2.2. MITRE Matrices

One way of visualizing the attack data is by categorizing the attacks. The MITRE Corporation, for example, categorizes security incidents using so-called techniques which is called MITRE ATT&CK [34]. Each technique can be assigned to one or more tactics which represent general procedures of security breaches in enterprise contexts. MITRE ATT&CK then combines several techniques and tactics into so-called matrices each with a specific purpose. There are two matrices that are relevant for DFIR in O-RAN: The 5G Hierarchy of Threats (FiGHT) [35] matrix and the Threat Matrix for Kubernetes by Microsoft [36].

The FiGHT matrix is a collection of tactics and techniques relevant for 5G systems. It is based on the ATT&CK tactics and techniques but adds FiGHT-specific techniques. As of writing, the majority of techniques in the FiGHT matrix are considered either theoretical or proof of concept but the visualization on the dashboard still is useful, especially for future research in the field.

The Threat Matrix for Kubernetes was published by employees of the Microsoft Defender for Cloud team and is also based on the ATT&CK tactics and techniques. The techniques in the Kubernetes matrix are tailored to the security of Kubernetes clusters and are therefore also relevant for DFIR in O-RAN.

The dashboard could contain views for each matrix that show the tactics and techniques contained in the respective matrix. This could look very similar to how these matrices are displayed by their creators as seen in Figure A.1 and Figure A.2 in the appendix. The techniques in the matrix that have been used to perform attacks could be highlighted and contain links to the detailed view of each respective attack or collection of attacks. Each tactic and technique in the matrix could additionally be linked to the respective entry on the MITRE ATT&CK website for further information.

6.2.3. Cyber Kill Chain

The Cyber Kill Chain by Lockheed Martin [37] is a model that describes the steps that an attacker has to follow in order to compromise a system. It is divided into seven steps that are shown in Figure 6.2. Each step represents a phase of an attack on a system with increasing severity.

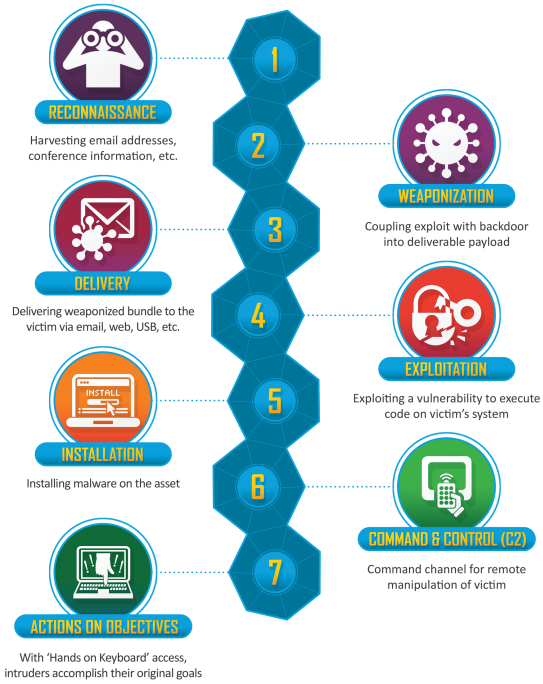

Figure 6.2.: The Cyber Kill Chain by Lockheed Martin. Taken from [37].

A visualization of the Cyber Kill Chain on the dashboard could contain the seven steps and highlight the steps that the attacks have reached. The attacks in each step could contain links to the detailed view of each respective attack.

6.2.4. Network and Cluster Visualization

Since the attacks are performed in the context of a Kubernetes cluster that hosts the O-RAN components which are themselves arranged in a network topology, it is useful to visualize that to gain a better understanding of the components in the network and their interconnectivity. The network visualization could indicate which components were attacked and which components were revealed through attacks. Successful attacks could be used to highlight vulnerable components in the network.

6.2.5. Live Updates

To enable the user of the dashboard to monitor ongoing attacks and campaigns, live updates could be implemented. Using a method of sending events from the server to the browser, the dashboard could receive updates about new attacks and update the visualizations accordingly. Live updates can, for example, be implemented using

WebSockets, Server-Sent Events (SSE) or the dashboard can periodically poll the server for new data. The user can enable and disable live updates as well as configure the update interval.

7. Implementation and Validation

7.1. Implementation

The dashboard is served from a web server that communicates with a MongoDB database and renders HTML pages that are sent to the client. See Figure 7.1 for an architectural overview.

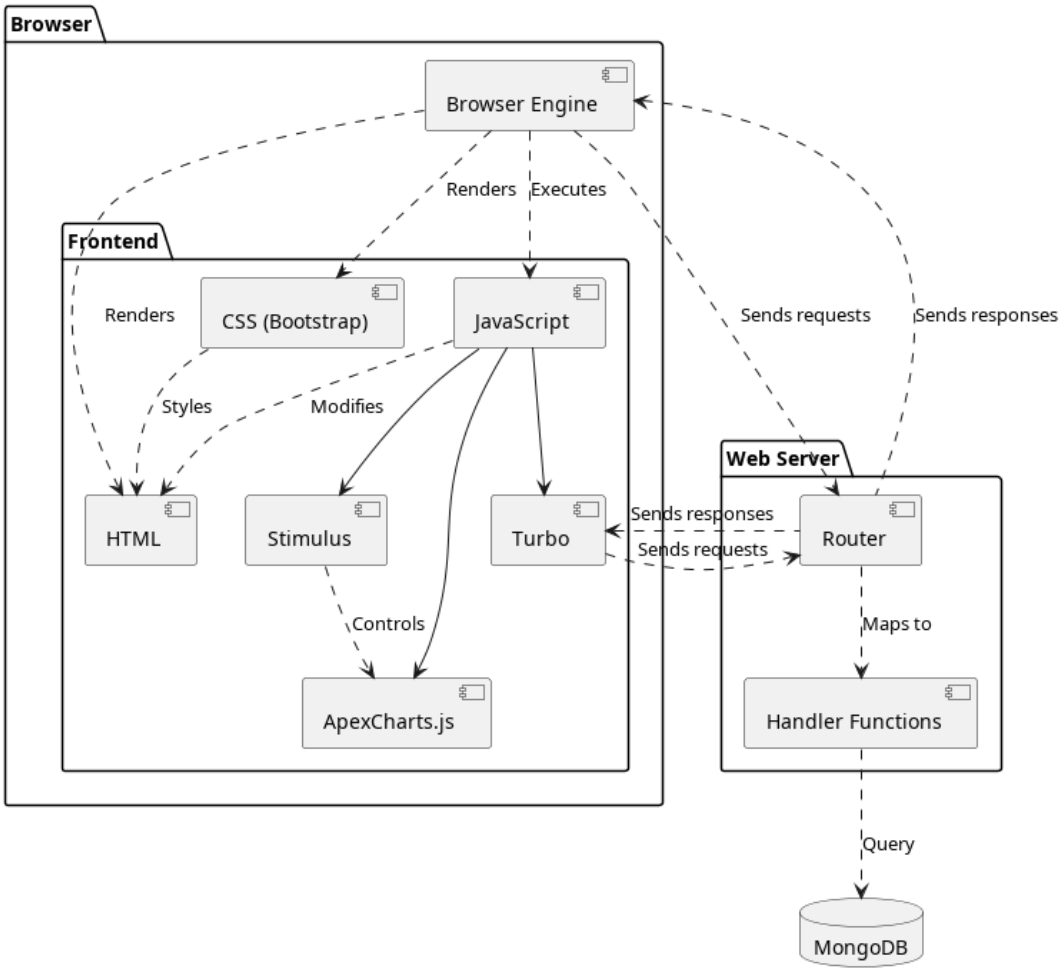

Figure 7.1.: Software architecture of the dashboard.

7.2. Backend

The backend component is implemented in the Go programming language [38] which is a statically typed language mostly used for system programming, servers, and concurrent applications. The HTML pages for the dashboard are produced using server-side rendering which has the advantage of enabling the utilization of the server's processing power for the transformation of the potentially large amounts of data from the database into a displayable format before sending it to the client.
The Hypertext Transfer Protocol (HTTP) server is realized using the Gin web framework [39] that provides a simple interface for developing web servers. The router component of the Gin web framework is responsible for the mapping of HTTP requests to handler functions that in turn process the requests. Each handler function takes the request information, performs the necessary operations including database queries, and renders an HTML page that is sent back to the client.
The communication with the database is handled by the official MongoDB Go driver [40]. The driver provides a convenient interface for querying the database and transforming the results into Go data structures that can then be visualized using the HTML templates.

7.3. Frontend

The frontend of the dashboard is the set of HTML, Cascading Style Sheets (CSS) and JavaScript files that are prepared on the server and sent to the client where they are rendered in a web browser. While the frontend is mainly built off of HTML files on the server, styling is enabled through the Bootstrap library [41]. Bootstrap enhances the appearance of the HTML pages and provides ready-to-use HTML components in a convenient way. Such components include a navigation bar, tables, buttons, and more.
Advanced interactivity is provided using JavaScript in the client's browser. This is implemented using the Turbo [42] and Stimulus [43] libraries from Hotwire [44]. Turbo enables the dashboard to feel like a single-page application by replacing traditional browser behavior: Instead of following links and submitting forms directly, Turbo intercepts these actions in the background, sends the respective requests to the server, and replaces the existing Document Object Model (DOM) nodes without the need for a full page refresh. Turbo can also be used for updating the DOM nodes via so-called Turbo streams. These are small HTML fragments that instruct the browser to update specific DOM nodes with the given content using one of the available actions. The actions include replacing

the content of a node, appending content to a node, or removing nodes. In the current implementation, Turbo streams are not used but they may be in the future when the dashboard is extended e.g. with live updates or notifications.
Stimulus adds an additional layer of flexibility to implement interactivity that neither Bootstrap nor Turbo provide out of the box. Stimulus works by writing small JavaScript classes called controllers that are attached to HTML elements using HTML element attributes. Every Stimulus controller can define a set of targets that it interacts with which are other HTML elements that are relevant to the controller. For example, a controller that displays an alert when the user clicks on a button would define that button as one of its targets. Additionally, the controller can define so-called values that are also passed to the controller via HTML element attributes. These values represent some state and can change over time or after user interaction.
Finally, ApexCharts.js [30] is used to render the charts on the dashboard. ApexCharts.js is equipped with a plethora of different chart types, visualizations, and customization options. The available visualizations include line charts, bar charts, pie charts, and more sophisticated visualizations such as heatmaps and treemaps. ApexCharts.js works by creating Scalable Vector Graphics (SVG) images from the given data and settings that are then embedded into the DOM. Stimulus controllers are used to wrap the chart rendering process and provide additional features such as updating the chart e.g. when the user selects a different set of filters.

7.4. Validation

The dashboard is validated by using test data that is made available by the ATTACK part of the 5G-FORAN project. The test data is produced by the ATTACK part while testing the Clusterforce tool that runs the attacks. The test data mimics the data that will be produced by the real attacks and is therefore appropriate for testing the dashboard.
By connecting the web server to the development database, the test data can be visualized on the dashboard similar to how it would be visualized with the real data. The dashboard's functionalities can therefore be validated by using it as if it was displaying the real attack data.

8. Discussion

The deployment of the dashboard went smoothly which is due to the fact that the dashboard is a web application that can be accessed from any device that has a web browser installed. Using Ansible [45] for the automated deployment of the dashboard also provides a way to easily update the dashboard to a newer version.
With the current number of attacks in the database, the dashboard's performance is acceptable as the dashboard is still responsive and the visualizations are rendered in a timely manner. However, the performance of the dashboard in its current implementation will decrease as the number of attacks in the database increases. In order to mitigate this problem, the dashboard should only load a subset of the attacks from the database and load more attacks as the users requests them, e.g. by setting the correct filters and providing default values for the filters.
One remaining problem is the scaling of the x-axis of the timeline chart. The attacks usually only run for a few seconds such that they would be miniscule in comparison to a time frame of more than one hour. One way to solve this is the addition of a filter that allows the user to specify the time range that they want to investigate and enable the selection of time ranges of a few seconds or minutes. Additionally, if a bigger time frame is selected, the visualization should scale the bars of the attacks up in order for them to be visible in the larger time frame context. Buttons could also be added that allow the user to pre-fill the time range filter with common time ranges such as the last 5 minutes or 10 attacks.
Another shortcoming is that the attack timeline chart is only built by filtering the attack data by the tool that was used to execute the attack, yielding one row for each tool. More ways of building the timeline chart can be thought of, e.g. filtering by the MITRE tactic or technique or by the O-RAN component that was attacked. This would allow the user to investigate the attacks from different perspectives and to find patterns in the attacks that are not visible when only filtering by the tool used.
The design ideas of the dashboard were not fully implemented as of writing this report. The following ideas are a list of improvements that could be implemented in the future.

By running the same campaign against the same set of targets multiple times, it is possible to compare the effectiveness of the attacks and the respective defensive measures. The dashboard could then be used to visualize the comparison of the campaigns. For example, the individual attacks of the campaign can be displayed next to each other enabling the user to immediately identify differences in the results.
In Section 6.2.5, live updates of the dashboard were proposed. By listening to MongoDB events [46] on the web server, it can push updates to the dashboard whenever new data is inserted into the database or when data is changed in the database. Alternatively, the user's web browser can periodically make requests to the web server to check for new or updated data.

9. Conclusion

This report presented the design and implementation of a dashboard for the visualization of attacks on an O-RAN network. The visualizations on the dashboard are implemented using the ApexCharts.js library [30] which provides a wide range of different visualizations that can be customized to the needs of the 5G-FORAN project. The attacks are displayed using two key visualizations: the timeline charts and the MITRE matrices which enable the temporal correlation of attacks and the analysis of the different attack techniques, respectively. The visualizations on the dashboard were designed by following guidelines like the GQM model [22] and the push and pull principles [23]. A web application was implemented using the Go programming language and the Gin web framework [39]. For the frontend of the dashboard, libraries like the Bootstrap CSS framework [41], Turbo [42], and Stimulus [43] were used. Further development effort has to be put into the dashboard as not all proposed features were implemented as of writing this report.

A. Supplementary Figures and Tables

Table A.1.: Data produced by the Clusterforce attack tool.

Name	BSON Type [47]	Description
`id`	ObjectID	Unique identifier of the attack
`ip`	String	IP address of the attacked component
`subnet`	String	Subnet of the attacked component
`file_format`	String	File format of the output files (e.g. JSON)
`file_output_raw`	String	Raw output of the tool
`file_output_parsed`	String	Parsed output of the tool
`command`	Array of strings	Command that was executed
`tool_name`	String	Name of the tool that was used for the attack
`tool_version`	String	Version of the tool that was used for the attack
`timestamp_start`	DateTime	Timestamp when the attack was started
`timestamp_end`	DateTime	Timestamp when the attack was finished
`attack_location`	String	Location of the attack
`perspective`	String	Perspective of the attack
`attack_phase`	String	Phase of the attack
`oran_component`	String	O-RAN component that was attacked
`result_code`	Integer	Result code of the attack
`hostname`	String	Hostname of the attacked component
`cve`	Array of strings	CVE that was exploited
`tactic`	Array of strings	MITRE tactics that were used
`mitre_technique`	Array of strings	MITRE techniques that were used
`microsoft_technique`	Array of strings	MITRE techniques according to the Microsoft Threat Matrix for Kubernetes [36] that were used

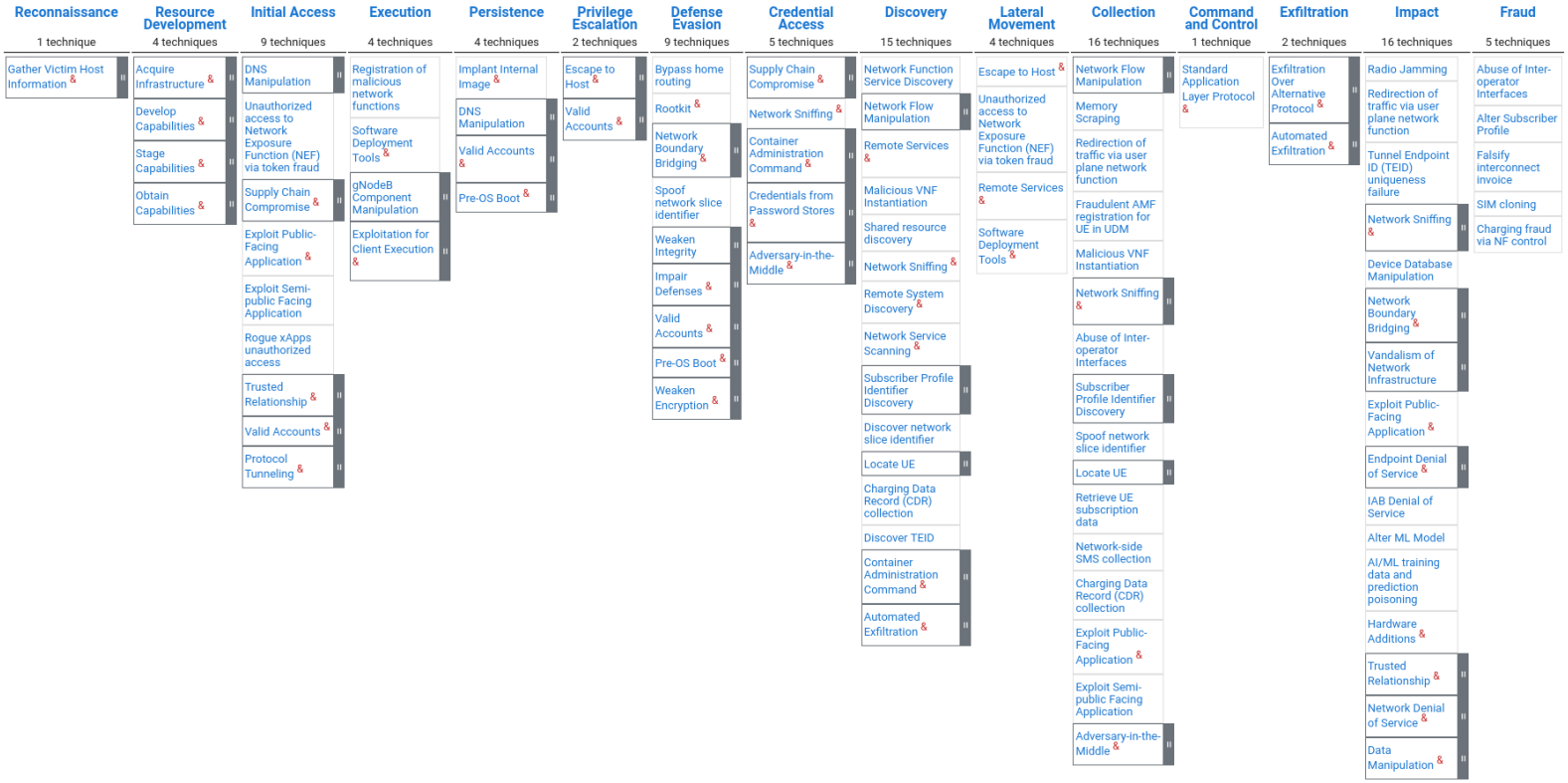

Figure A.1.: MITRE FiGHT matrix. Taken from [35].

Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Collection	Impact
Using cloud credentials	Exec into container	Backdoor container	Privileged container	Clear container logs	List K8S secrets	Access Kubernetes API server	Access cloud resources	Images from a private registry	Data destruction
Compromised image In registry	bash/cmd inside container	Writable hostPath mount	Cluster-admin binding	Delete K8S events	Mount service principal	Access Kubelet API	Container service account	Collecting data from pod	Resource hijacking
Kubeconfig file	New container	Kubernetes CronJob	hostPath mount	Pod / container name similarity	Container service account	Network mapping	Cluster internal networking		Denial of service
Application vulnerability	Application exploit (RCE)	Malicious admission controller	Access cloud resources	Connect from proxy server	Application credentials in configuration files	Exposed sensitive interfaces	Application credentials in configuration files		
Exposed sensitive interfaces	SSH server running inside container	Container service account			Access managed identity credentials	Instance Metadata API	Writable hostPath mount		
	Sidecar injection	Static pods			Malicious admission controller		CoreDNS poisoning		
							ARP poisoning and IP spoofing		

Figure A.2.: Threat Matrix for Kubernetes by Microsoft. Taken from [36]

Bibliography

[1] PROCYDE GmbH. “5G-FORAN”. (2024), [Online]. Available: https://www.5g-foran.com/mehr.html (visited on 01/10/2024).

[2] TH Köln. “TH Köln”. (2023), [Online]. Available: https://www.th-koeln.de/startseite_16.php (visited on 12/18/2023).

[3] PROCYDE GmbH. “PROCYDE GmbH”. (2023), [Online]. Available: https://procyde.com/ (visited on 12/18/2023).

[4] Federal Office for Information Security. “Federal Office for Information Security”. (2023), [Online]. Available: https://www.bsi.bund.de/EN/Home/home_node.html (visited on 12/18/2023).

[5] O-RAN ALLIANCE e.V. “O-RAN ALLIANCE e.V”. (2023), [Online]. Available: https://www.o-ran.org/ (visited on 11/21/2023).

[6] O-RAN Software Community. “O-RAN Software Community”. (2023), [Online]. Available: https://o-ran-sc.org/ (visited on 12/18/2023).

[7] G. Lyon. “Nmap”. (2023), [Online]. Available: https://nmap.org/ (visited on 12/18/2023).

[8] Present Datadog, Inc. “Stratus Red Team”. (2023), [Online]. Available: https://stratus-red-team.cloud/ (visited on 12/18/2023).

[9] Aqua Security Software Ltd. “Kube-hunter”. (2024), [Online]. Available: https://aquasecurity.github.io/kube-hunter/kube-hunter/ (visited on 01/10/2024).

[10] Aqua Security Software Ltd. “Trivy”. (2023), [Online]. Available: https://trivy.dev/ (visited on 12/09/2023).

[11] TechTarget. “What is a Radio Access Network (RAN)?” (2023), [Online]. Available: https://www.techtarget.com/searchnetworking/definition/radio-access-network-RAN (visited on 11/21/2023).

[12] S. Köpsell, A. Ruzhanskiy, A. Hecker, D. Stachorra, and N. Franchi. "Open-RAN Risikoanalyse". (2023), [Online]. Available: https://www.bsi.bund.de/DE/Service-Navi/Publikationen/Studien/Open-RAN/Open-RAN.html?nn=1019518 (visited on 12/18/2023).

[13] O-RAN ALLIANCE e.V. "O-RAN Downloads". (2023), [Online]. Available: https://orandownloadsweb.azurewebsites.net/specifications (visited on 11/21/2023).

[14] O-RAN ALLIANCE e.V. "WG1 - Use Cases and Overall Architecture Workgroup". (2023), [Online]. Available: https://public.o-ran.org/display/WG1/introduction (visited on 11/21/2023).

[15] O-RAN ALLIANCE e.V., *O-RAN Architecture Description v10.00*, Oct. 2023.

[16] 3GPP. "3GPP". (2023), [Online]. Available: https://www.3gpp.org/ (visited on 12/19/2023).

[17] IBM. "What is Digital Forensics and Incident Response (DFIR)?" (2023), [Online]. Available: https://www.ibm.com/topics/dfir (visited on 12/11/2023).

[18] K. Kent, S. Chevalier, T. Grance, and H. Dang, "Guide to Integrating Forensic Techniques into Incident Response", National Institute of Standards and Technology, NIST Special Publication (SP) 800-86, Sep. 1, 2006. DOI: 10.6028/NIST.SP.800-86. [Online]. Available: https://csrc.nist.gov/pubs/sp/800/86/final (visited on 12/11/2023).

[19] Merriam-Webster, Incorporated. "Definition of Dashboard". (Nov. 11, 2023), [Online]. Available: https://www.merriam-webster.com/dictionary/dashboard (visited on 11/14/2023).

[20] ApexCharts. "RadialBar / Circular Gauge Chart", ApexCharts.js. (2023), [Online]. Available: https://apexcharts.com/docs/chart-types/radialbar-gauge/ (visited on 12/11/2023).

[21] A. H. C. van der Heijden, "Two stages in visual information processing and visual perception?", *Visual Cognition*, vol. 3, no. 4, pp. 325–353, 1996, ISSN: 1464-0716. DOI: 10.1080/135062896395625.

[22] V. R. Basili, G. Caldiera, and H. D. Rombach, "The Goal Question Metric Approach", 1994.

[23] A. Janes and G. Succi, "To pull or not to pull", Oct. 25, 2009, pp. 889–894. DOI: 10.1145/1639950.1640052.

[24] E. R. Tufte, *The Visual Display of Quantitative Information*. Graphic Press, 1983, 197 pp., ISBN: 978-0-318-02992-4. Google Books: yl3YzwEACAAJ.

[25] The Kubernetes Authors. "Kubernetes Dashboard". (2024), [Online]. Available: https://kubernetes.io/docs/tasks/access-application-cluster/web-ui-dashboard/ (visited on 01/02/2024).

[26] Grafana Labs. "Grafana". (2023), [Online]. Available: https://grafana.com/ (visited on 12/26/2023).

[27] Metabase. "Metabase". (2023), [Online]. Available: https://www.metabase.com (visited on 12/26/2023).

[28] Plotly. "Dash". (2023), [Online]. Available: https://dash.plotly.com/ (visited on 12/26/2023).

[29] Mike Bostock and Observable, Inc. "D3". (2023), [Online]. Available: https://d3js.org/ (visited on 12/26/2023).

[30] ApexCharts. "ApexCharts.js". (2023), [Online]. Available: https://apexcharts.com/ (visited on 12/05/2023).

[31] Grafana Labs. "MongoDB data source". (2023), [Online]. Available: https://grafana.com/docs/plugins/grafana-mongodb-datasource/latest/ (visited on 12/26/2023).

[32] StatCounter. "Desktop Screen Resolution Stats Worldwide". (2023), [Online]. Available: https://gs.statcounter.com/screen-resolution-stats/desktop/worldwide/ (visited on 12/12/2023).

[33] R. Klein, *Scheduling of Resource-Constrained Projects*. Springer Science & Business Media, 1999, vol. 10.

[34] The MITRE Corporation. "MITRE ATT&CK®". (2023), [Online]. Available: https://attack.mitre.org/ (visited on 12/12/2023).

[35] The MITRE Corporation. "MITRE FiGHT™". (2023), [Online]. Available: https://fight.mitre.org/ (visited on 12/12/2023).

[36] Y. Weizman, D. Patrich, and R. Pliskin. "Threat Matrix for Kubernetes". (2023), [Online]. Available: https://microsoft.github.io/Threat-Matrix-for-Kubernetes/ (visited on 12/10/2023).

[37] Lockheed Martin Corporation. “Cyber Kill Chain®”. (2023), [Online]. Available: https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html (visited on 09/27/2023).

[38] Google. “The Go Programming Language”. (2023), [Online]. Available: https://go.dev/ (visited on 12/05/2023).

[39] Gin Team. “Gin Web Framework”. (2023), [Online]. Available: https://gin-gonic.com/ (visited on 12/05/2023).

[40] MongoDB, Inc. “MongoDB Go Driver”. (2023), [Online]. Available: https://www.mongodb.com/docs/drivers/go/current/ (visited on 12/09/2023).

[41] Bootstrap team. “Bootstrap”. (2023), [Online]. Available: https://getbootstrap.com/ (visited on 12/05/2023).

[42] 37signals. “Turbo”. (2024), [Online]. Available: https://turbo.hotwired.dev/ (visited on 01/10/2024).

[43] 37signals. “Stimulus”. (2024), [Online]. Available: https://stimulus.hotwired.dev/ (visited on 01/10/2024).

[44] 37signals. “HTML Over The Wire”. (2023), [Online]. Available: https://hotwired.dev/ (visited on 12/05/2023).

[45] Red Hat. “Ansible”. (2024), [Online]. Available: https://www.ansible.com (visited on 01/03/2024).

[46] MongoDB, Inc. “Change Streams”. (2023), [Online]. Available: https://www.mongodb.com/docs/manual/changeStreams/ (visited on 12/15/2023).

[47] MongoDB, Inc. “BSON Types”. (2023), [Online]. Available: https://www.mongodb.com/docs/manual/reference/bson-types/ (visited on 12/09/2023).