Thesis

NHL Stenden University of Applied Sciences

In the department of:

ICT & CT Information Technology Bachelor Emmen

In association with:

Quality ICT B.V.

Written by:

Christopher Sulistiyo (4850025)

Submission:

1-August-2024

Work placement lecturer(s):

Company Supervisor(s): Manuel Weidijk
Mark Kolk

Summary

SentinelOne is a cybersecurity company based in California, United States that has a product of the same name. It is an EDR platform, which purpose is to detect and respond to threats of an endpoint in real-time, replacing the traditional AV. It utilizes AI and ML on an Agent to detect and respond to threats, and it is deployed on the endpoint itself. The purpose of an EDR platform is to have a centralized

Q-ICT is a small cybersecurity consultant firm based in Emmen, Drenthe, the Netherlands. It is a partner of Pax8, which is a Microsoft distributor, therefore they handle all cybersecurity related to Windows and Microsoft products. They have over 400 clients, all spread throughout the Netherlands, and their main target audience is small to medium-sized businesses. Recently, as of last year with the assistance of their partner, Pax8, Q-ICT has purchased the rights to a SentinelOne subscription. Since then, they have been utilizing the product to provide better endpoint security to their clients. Q-ICT itself

Contents

List of Figures 4
List of Example Code Examples 5
List of Tables 6

Glossary **7**

Acronyms **9**

1 Introduction **12**
1.1 Project Background 12
1.2 Problem Statement 12
1.3 Project Objectives 12
1.4 Reading Guide 12
1.5 The Company 13

2 Research Design **17**
2.1 Research Topic 17
How can Quality ICT B.V. effectively integrate and leverage SentinelOne EDR platform for continuous cybersecurity monitoring? 17
2.2 Research Sub-Questions 17
Sub-question #1: What is the current situation of the QaaS app of Quality-ICT B.V. 17
Sub-question #2: What is SentinelOne EDR platform? 17
Sub-question #3: How can SentinelOne be integrated into the QaaS app environment? 17
Sub-question #4: What are the most suitable visualization techniques for displaying the data received by SentinelOne API in Flutter? 17
2.3 Research Methodology 17
2.3.1 Method of Data Collection 17
2.3.2 Selected Measuring Instruments 18
2.3.3 Method of Data Analysis 18
2.3.4 Reliability, Validity, and General Applicability 18
2.3.5 Research Limitations 19

3 Research Results **20**
3.1 Introduction 20
3.2 Research Sub-Question #1: What is the current state of the QaaS app? 20
3.2.1 The QaaS App Infrastructure 21
3.2.2 Conclusion 27
3.3 Research Sub-Question #2: What is SentinelOne EDR Platform? 28
3.4 Research Sub-Question #3: How can SentinelOne be integrated with the QaaS app? 32
3.4.1 Conclusion 33
3.5 Research Sub-Question #4: What are the best data visualization techniques for SentinelOne? 33
3.5.1 Data Visualization Widgets in Flutter 33
3.5.2 Comparison to other EDR solutions 35
3.5.3 Conclusion 36

4 Realization **37**
4.1 The Back-end 37
4.1.1 Setting up permissions with Google Secret Manager 38

4.1.2 Compliance with ESLint compiler . . . 38
4.1.3 SentinelOne NPM packages . . . 41
4.1.4 Firestore database structure . . . 41
4.1.5 Setting up permissions with Firestore . . . 42
4.1.6 Connection to Algolia . . . 43
4.2 The Front-end . . . 43

5 Conclusion and Recommendation 44
5.1 Conclusion . . . 44
5.2 Recommendation . . . 44
5.2.1 Structure of the front-end codebase . . . 44
5.2.2 Integration with N-Central data . . . 44
5.2.3 Redis middleware NoSQL database caching . . . 44
5.2.4 Compacting everything into one function . . . 45
5.2.5 . . . 45

Bibliography 46

List of Figures

1.1 NIST working methodology that is followed by Q-ICT 15
1.2 Level of structure of MKBiT 16
1.3 Flat (hierarchical) structure of Q-ICT 16

3.1 The login page of the QaaS app, implementing Google 2FA and reCAPTCHA 21
3.2 The infrastructure of the QaaS app 22
3.3 All of the products offered by Firebase (*Kubernetes, 2018*) 23
3.4 All the extensions that are used in the QaaS app, mostly about Algolia 24
3.5 The App Check feature in the QaaS app 24
3.6 The function in the QaaS app that is scheduled to run every 7 days to keep the API data up-to-date (*Cronitor, n.d.*) 24
3.7 Different modules in the QaaS app in which the IT admin can determine which pages can be accessed by which user tags 27
3.8 Different user tags in the QaaS app in which correlates to what actions can a user do within the web application 28
3.9 How all terms related to various technologies in cybersecurity connected to each other 30
3.10 Examples of automatic reports that can be downloaded in SentinelOne 32
3.11 SentinelOne API Documentation 33
3.12 Chart types in Flutter fl_chart 34

4.1 Granting access of read, write, and update from User Preference collection to users in the Firebase Console 42

5.1 How the infrastructure would look if Redis is implemented 45

Listings

3.1 Example of a typical onRequest function 25
3.2 Example of onCall function with authentication check 25
4.1 Example of Class and Interface defined in Models 38
4.2 An example of Firebase HTTP Request onCall Cloud Function version 1.0 39
4.3 onCall Cloud Function version 2.0, where the parameters of the function is less and the way it handles the authorization is different 40
4.4 LogError functionality in the Utilities folder 40
4.5 An example of how that utility is used in the Cloud Function, the reason why exception type is unknown is to comply with the ESLint rules that does not recommend to use the any type to the variables 41
4.6 Example of challenges in JavaScript because of its loosely typed nature that creates lack of type safety, type coercion, and type conversion 41
4.7 Another example of JavaScript challenges 41

List of Tables

3.1 Comparison between the 1st and 2nd Generation of Cloud Functions 23

Glossary

API is a software intermediary that allows two applications to talk to each other. In other words, an API is the messenger that delivers request to the provider that was requested from and then delivers the response back (**Wikipedia, n.d.-b**) . 12

CAPTCHA is a type of challenge-response test used in computing to determine whether or not the user is human (*Wikipedia, n.d.-d*) . 8, 23

CRM is a process or strategy that companies use to manage and analyze customer interactions and data throughout the customer ' lifecycle. CRM technology is a software system that helps organizations easily track all communications and nurture relationships with their leads and clients *Wikipedia, n.d.-f* . 20

ERP is a type of software that organizations use to manage core day-to-day business activities needed to run a company such as accounting, finance, procurement, project management, manufacuring, risk management and compliance, HR, and supply chain operations (*Wikipedia, n.d.-h*) . 20

JSON is a lightweight data-interchange format that is easy for humans to read and write while being also easy for machines to parse and generate too. It is based on a subset of the JavaScript programming language and is commonly used for representing structured data. JSON is often used for transmitting data between a server and a web application, as well as for storing configuration data (*Wikipedia, n.d.-j*) . 7, 17

NoSQL is a category of DB that provides a mechanism for storage and retrieval of data that is modelled in ways other than the tabular relations used in RDMS (*MongoDB, n.d.*). NoSQL DBs are typically designed to handle large volumes of unstructured or semi-structured data, such as JSON, XML, or binary objects, and they offer a flexible data model that can evolve over time. Both databases in Firebase, Firestore, and Real-time Database are NoSQL databases. . 21

Pentest A.K.A. ethical hacking, is a simulated cyberattack where professional ethical hackers break into corporate networks to find weaknesses (Wikipedia, n.d.-l) . 13

REST is a software architectural style for providing standards between computer systems on the web, making it easier to communicate with each other. It is often characterized by how they are stateless and separate the concerns of client and server (**IBM, n.d.**) . 7, 20

RESTful API is an API that follow REST principles (Fielding and Taylor, 2000). It considered the gold standard for scalability and are highly compatible with microservice architecture. It is the often used protocol in the context of building APIs for web-based applications . 20

RMM is a software platform that helps MSPs remotely monitor and manage their clients' endpoints, networks, and computers (Wikipedia, n.d.-m). 20

SOAP API is an API strictly based on XML for the message structure and HTTP for the protocols (*Indeed, 2023*) . 20

XML is a markup language that defines a set of rules for encoding documents in a format that is both human and machine-readable. It provides a way to structure data in a hierarchical format using tags to define elements and attributes within those elements (*Wikipedia, n.d.-n*) . 20

Cron Job is a job scheduler responsible on scheduling tasks on repeated schedule on a server (*Taylor, 2024*) . 26

Network TAP is a simple device that connects directly to the cabling infrastructure to split or copy packets for use ion analysis, security or general network management . 29

reCAPTCHA is a CAPTCHA system owned by Google to enable web hosts to distinguish between human and automated access to websites (*Wikipedia, n.d.-e*) . 23

Threat Hunting A.K.A. cyberthreat hunting, is a proactive approach to identifying previously unknown, or ongoing non-remediated threats, and searching for signs of potential cyber threats within an organization’s network or system (*Wikipedia, n.d.-g*) . 29, 31

Acronyms

2FA Two-Factor Authentication. 4, 21, 23

A.K.A. Also Known As. 7, 8, 29, 34

AI Artificial Intelligence. 1, 12, 28, 29, 32, 33, 42

APA American Psychological Association. 13

API Application Programming Interface. 4, 7, 12, 13, 17–20, 23–26, 28, 31–33, 37, 38, 42

ARP Address Resolution Protocol. 31, 32

AV Anti-Virus. 1, 14, 29, 31, 35, 42

AWS Amazon Web Services. 29

B.V. Besloten Vennootschap. 0, 17

BaaS Back-end as a Service. 21

CAPTCHA Completely Automated Public Turing test to tell Computers and Humans Apart. 4, 7, 8, 21, 23

CDR Content Disarm and Reconstruction. 20

CISO Chief Information Security Officer. 13

CMEK Customer-Managed Encryption Keys. 26

CRM Customer Relationship Management. 7, 20

CT Creative Technology. 0

DB Database. 7, 14, 25, 26, 29, 42

DHCP Dynamic Host Configuration Protocol. 32

DNS Domain Name System. 29

e.g., exempli gratia. 32

EDR Endpoint Detection and Response. 1, 12, 17, 29, 31, 35, 36, 42

EPP Endpoint Protection Platform. 29, 35

ERP Enterprise Resource Planning. 7, 20, 27

etc., et cetera. 20, 21, 29

ETDR Endpoint Threat Detection and Response. 29

GB Gigabyte. 23

GCP Google Cloud Platform. 21, 26, 27

HR Human Resources. 7

HTTP Hypertext Transfer Protocol. 7, 19, 20, 23, 25, 33, 37

HTTPS Hypertext Transfer Protocol Secure. 21

i.e., id est. 15

IAM Identity and Access Management. 27

ICT Information and Communication Technology. 0, 17, 20

IDS Intrusion Detection System. 29

IoT Internet of Things. 28, 31

IP Internet Protocol. 32

IPS Intrusion Prevention System. 29

IT Information Technology. 12–14, 17, 20, 27–29, 42

ITIL Information Technology Infrastructure Library. 13

JS JavaScript. 21, 38

JSON JavaScript Object Notation. 7, 17, 18, 20

MAC Media Access Control. 32

MCDA Multi-Criteria Decision Analysis. 18

MDR Managed Detection and Response. 29, 32

MFA Multi-Factor Authentication. 23

MGMT Management. 31, 33

MIB Management Information Base. 32

MKB Midden- en Kleinbedrijf. 4, 14–16

ML Machine Learning. 1, 12, 29

MS Microsoft. 14, 15, 20

MSP Managed Service Provider. 7, 20

NDR Network Detection and Response. 29, 31

NIC Network Interface Controller. 32

NIST National Institute of Standards and Technology. 4, 15

NoSQL Not Only SQL. 7, 21, 42

NPM Node Package Manager. 37, 41

OAuth Open Authorization. 26

OS Operating System. 35

OTP One Time Password. 21

Pentest Penetration Testing. 7, 13

Pub/Sub Publisher/Subscriber. 26

Q-ICT Quality ICT. 0, 1, 4, 12–20, 26, 27, 31, 33, 37, 42

QaaS Quality as a Service. 4, 12, 14, 15, 17–21, 23–27, 33, 36, 37, 42

RAM Random Access Memory. 23

RDMS Relational Database Management System. 7

REST Representational State Transfer. 7, 20, 25, 32, 33

RMM Remote Monitoring and Management. 7, 20

SaaS Software as a Service. 20

SDK Software Development Kit. 25, 32

SEM Security Event Management. 29

SIEM Security Information and Event Management. 29, 31

SIM Security Information Management. 29

SME Small and Medium-sized Enterprises. 13, 20

SOAP Simple Object Access Protocol. 7, 20

SOAR Security Orchestration, Automation, and Response. 29

SOC Security Operations Center. 29, 31, 32

SWOT Strengths, Weaknesses, Opportunities, and Threats. 18

TAP Test Access Point. 7, 29

TS TypeScript. 21, 38

UX User Experience. 20, 35, 43

vCPU Virtual Central Processing Unit. 23

VLAN Virtual Local Area Network. 31

XML Extensible Markup Language. 7, 18, 20

Chapter 1

Introduction

1.1 Project Background

In today's rapidly evolving digital landscape, cybersecurity remains a paramount concern for organizations across all industries. With the proliferation of sophisticated cyber threats and the increasing complexity of IT infrastructures, business are constantly seeking new and innovative ways to protect their digital assets and fortify their defences and safeguard sensitive data. In this pursuit, cybersecurity consultant firms have emerged as a critical ally for organizations, providing expert guidance and support in the development and implementation of robust cybersecurity strategies, playing a pivotal role in offering expertise and guidance to help organizations navigate the intricate realm of cybersecurity.

One of the key strategies employed by cybersecurity consultants is the integration of third-party security APIs into their arsenal of tools and technologies. These APIs provide invaluable functionalities, ranging from vulnerability assessment and security scans to device health monitoring and threat intelligence analysis by AI. By leveraging these APIs, cybersecurity consultants can enhance their capabilities and provide a more comprehensive and effective security solution to their clients, streamline their operations, provide clients with robust, proactive security measures, and improve their overall service delivery.

SentinelOne emerges as one of the leading organization in the cybersecurity real, offering cutting-edge endpoint protection and threat intelligence solutions. It leverages advance AI and ML algorithms, providing comprehensive protection against malware, ransomware, and other cyber-threats. It is one of the must-have protection tools for business organizations looking to bolster their cybersecurity posture and safeguard their digital assets.

As the main topic of this graduation work placement project, the author has been tasked to integrate SentinelOne endpoint protection capabilities into the client company internal application, the QaaS app. The main objective of this project is to promote transparency more into Q-ICT clients by providing them with real-time data and visibility into their organizations, enabling them to make informed decisions and take proactive measures to protect their own digital assets and mitigate security risks.

1.2 Problem Statement

The company currently manages numerous third-party APIs for the above-mentioned purposes. Currently, Thus, the company has tasked the author with the development of the SentinelOne security threat platform integration for continuous cybersecurity monitoring within the QaaS app as the main topic of his graduation work placement project.

1.3 Project Objectives

In the end of this project which consist of 90-99 working days, the following objectives should be achieved:

1. Effectively integrates and leverages the SentinelOne EDR platform for continuous cybersecurity monitoring within the QaaS app.
2. The QaaS app should have a way to visualize the data retrieved from the SentinelOne API in a user-friendly manner in order for the client users helpdesk support, financial department, cybersecurity department, software development department, and other employees within Q-ICT departments to see the data easier.

1.4 Reading Guide

This report is structured as follows:

- **Summary**: provides a brief and concise overview of the entire report, including the research questions, key findings, and conclusion. Its purpose is to provide readers with a quick and comprehensive understanding of the report.
- **Introduction**: provides an overview of the project background, the research topic of the company, and the project objectives. It introduces the context of the research and outlines the structure of the report.
- **Research Design**: outlines the methodologies employed during the research process. It describes the research approach, data collection methods, selected measuring instruments, data analysis techniques used to address the research questions, reliability, validity, and general applicability.
- **Research Results**: presents the research results, including the research methodology, the findings, and the analysis of the research questions. Firstly, it describes the methodologies employed during the research, and then it provides a detailed account of the research process and the outcomes of the research.
- **Realization**: provides a detailed description of the software end-product developed during this work placement project. It outlines insights into design, development, and implementation phases. It also highlights key features, functionalities, and technology specifications used in the project.
- **Conclusions and Recommendations**: the Conclusion summarizes the key findings and results achieved during the research and realization phases. The Recommendation section outlines the proposed next steps and future research areas to further enhance the project and address any outstanding issues. It discusses potential areas for further exploration or refinement.
- **References**: lists all the sources cited in the report following the appropriate to APA 7th edition citation style.
- **Appendices**: includes any additional supplementary information, data, or materials that are relevant to the report but not included in the main body of the report.

1.5 The Company

Q-ICT, is a small cybersecurity consultancy based in Emmen, northeast of the Netherlands. The company follows a flat organizational structure, which means that there are few or no levels of middle management between the staff and everything communication directly goes with the director. It recognizes the critical importance of proactive API monitoring in safeguarding its clients' digital assets. Their customers are SME companies with employees ranging from 1 to 100. Q-ICT is therefore asked to monitor their clients' devices and ensuring the overall security of their systems, IT infrastructure, and digital assets. As of now, Q-ICT has over 400 active customers. Some of the notable clients are: Kigpolis Verzekeringen, CLS Europe, and Heli Holland.

Departments

The company consists of multiple departments in its behalf, each with their own functions and responsibilities. Those departments are the following:

1. *Service Help Desk Department*: it serves as a frontline support function responsible for addressing client inquiries, troubleshooting technical issues, and providing guidance and support to clients in resolving their technical challenges. It contains 2 subdepartments, the first level help desk and the second level help desk. The First Level Help Desk is the first point of contact for clients seeking technical assistance and support, and it is responsible for managing and resolving client issues in a timely and efficient manner. The Second Level help desk is responsible for providing more advanced technical support and troubleshooting for complex technical issues and consists of Senior System Engineers. It has 2 services, mainly the indoor and outdoor customer services. With the indoor customer services, the company provides remote support to clients, while with the outdoor customer services, the company provides on-site support to clients.
2. *Cybersecurity Department*: it's responsible for implementing procedures that will be used throughout the company's system, especially in Help Desk Department, to ensure that the company's information technology infrastructure is secure. It also develops methodologies and best practices related to cybersecurity, as well as integrating ITIL principles and product management practices into the company's operations. Conducting regular security scans for clients' networks, systems, and applications to identify vulnerabilities and security risks and performing Pentest that involves simulating cyberattacks to identify weaknesses in the client's defences and assess their ability to withstand and respond to real-world cyber threats. This department mainly consist of IT managers, Pentesters, CISOs, and cybersecurity specialists.
3. *Software Development Department*: it addresses Q-ICT clients' need by creating custom software solutions tailored to their specific requirements. It consists of software developers that work closely with clients to understand their unique cybersecurity challenges and design solutions that effectively address those concerns, while utilizing their expertise in programming languages, software frameworks, and cybersecurity principle to develop secure and reliable applications. This is the department where the author

is currently working on his graduation workplacement project. Mainly, this department uses Dart with Flutter as the main front-end development framework, and Node.js with TypeScript template as the main back-end development framework. Furthermore, it utilizes Google Firebase as the main cloud solution for the applications it develops as it works together with Flutter, but it expresses its desires to expand more into MS Azure in the future.

4. *Financial Department*: it is responsible for managing the financial aspects of the company to ensure its financial health and stability. It includes Accountant and Financial Advisor, and they are responsible for analyzing financial data, identifying trends, and making strategic financial decisions to support the company's growth and objectives.

Q-ICT Sister Companies

Q-ICT has 2 other sister companies who are also under the same ownership of Mark Kolk, which are MKBoT and QaaS.nu respectively.

MKBiT

It is a parent-company of Q-ICT who provides IT solutions and consultation to its clients. It was founded to support Q-ICT business operations. It shares the same office with Q-ICT. Its services include cloud solution, Microsoft 365 products, patch-management, online-backup, AV, anti-spam, and monitoring software applications.

Luuk Admiraal is directing this company. Because the company is an IT service provider, it has a vast network of suppliers and partners to provide them with the products and brands to support their business. Here is the list of the suppliers: DrayTek, HP, Microsoft, and Ubiquiti Network.

QaaS.nu

This company is responsible for the software development of Q-ICT. It is still a small company which was set up not a while ago, so there is not a lot of info feasible for the author to write about it. Their upcoming projects are still proof-of-concept. The company is directed by Pierre Kleine Schaars. This terminology is not to be confused with the QaaS app, which is the internal app of all the three companies.

Products and Subscriptions

Besides cybersecurity, together with MKBiT and QaaS.nu, Q-ICT offers a wide range of products and services to its clients. Those products are (*MKBiT, n.d.*):

- Customized software development: this field is one of the main responsibilities of Q-ICT itself, it provides customers with tailor-made software solutions that are designed to meet their specific needs and requirements. The company works closely with clients to understand their unique challenges and develop software applications that effectively address those concerns.
- Online-backup: the company offers a service where the clients can store copies of their files, documents, and data on remote servers via the internet. These remote servers are hosted in secure data centres. Furthermore, MKBiT also provides DB migration in case any event of disasters occurred.
- Active monitoring: the company will offer its customers the chance to oversee, track, and manage their computer systems, networks, or applications in real-time. It serves various purposes, including ensuring system stability, optimizing performance, enhancing security, and providing valuable insights into the usage patterns and behaviours of users or devices. In the future, it wants to bring more automation to its system, providing customers with automatic messages when a disk space becomes full, or when there is an error message in clients' computers.
- Anti-spam software: a specific type of software application that is designed by MKBiT to detect, prevent, and block unsolicited and unwanted e-mail messages, commonly known as spam, from reaching clients' e-mail inboxes. They are typically sent in bulk to many recipients without their consent, often containing advertisements, phishing attempts, malware, or other malicious content. It has features such as whitelist, blacklist, and filter e-mails per e-mail account or e-mail address.
- Microsoft 365 products: the company is also providing access to Microsoft's suite of cloud-based productivity tools and services, including installation services, to its clients. These include Microsoft Word, Excel, PowerPoint, Outlook, and more, which are all accessible online through a web browser or can be installed on local devices such as computers and smartphones.
- AV software: it resells and gives clients advice and comparisons on good antivirus software from manufacturers like McAfee, Bitdefender, and Kaspersky. This software is designed to detect, prevent, and remove malware from computer systems.
- Cloud services: the company also provides clients with a tailor-made cloud solution based on their specific wishes and problems, whether it is using a public, private, or hybrid cloud. It delivers a range of computing resources, applications, and services over the internet through cloud computing technologies. These services enable the clients to access and use computing resources without the need to own or maintain physical hardware and software infrastructure, enabling their employees to access their files

anytime, anywhere, without others having to access them.

In addition, for the software development, Q-ICT also uses other MS technologies such as Power Platform (i.e., Power Automate, Power BI, Power Apps, Power Virtual Agents, and Power Pages), Dynamics 365, and Azure to help with their software development, therefore the clients who are asking for the product to be developed with the help of those MS technologies will also have those subscriptions of the products, therefore paying extra charge.

Unofficial Partnerships

Q-ICT has several unofficial partnerships with other organizations, they are normally located in the same building that Q-ICT rents its office from. These organizations help Q-ICT in their own speciality, and Q-ICT helps them in return. These organizations are:

- MemoICT: is a Shopware company that helps with e-commerce (*MemoICT, n.d.*).
- Ondernemend Emmen: helps with international entrepreneurship, knowledge sharing, and networking (*OndernemmendEmmen, n.d.*).
- Peat Digital: helps with online marketing to promote Q-ICT products more into wider audience (*PeatDigital, n.d.*).
- Webba: helps with web development, especially designing Q-ICT, MKBiT, and QaaS.nu website (*Webba, n.d.*).
- InDiv Solutions: also helps Q-ICT with the front-end side of their website development (*inDiv Solutions, n.d.*).

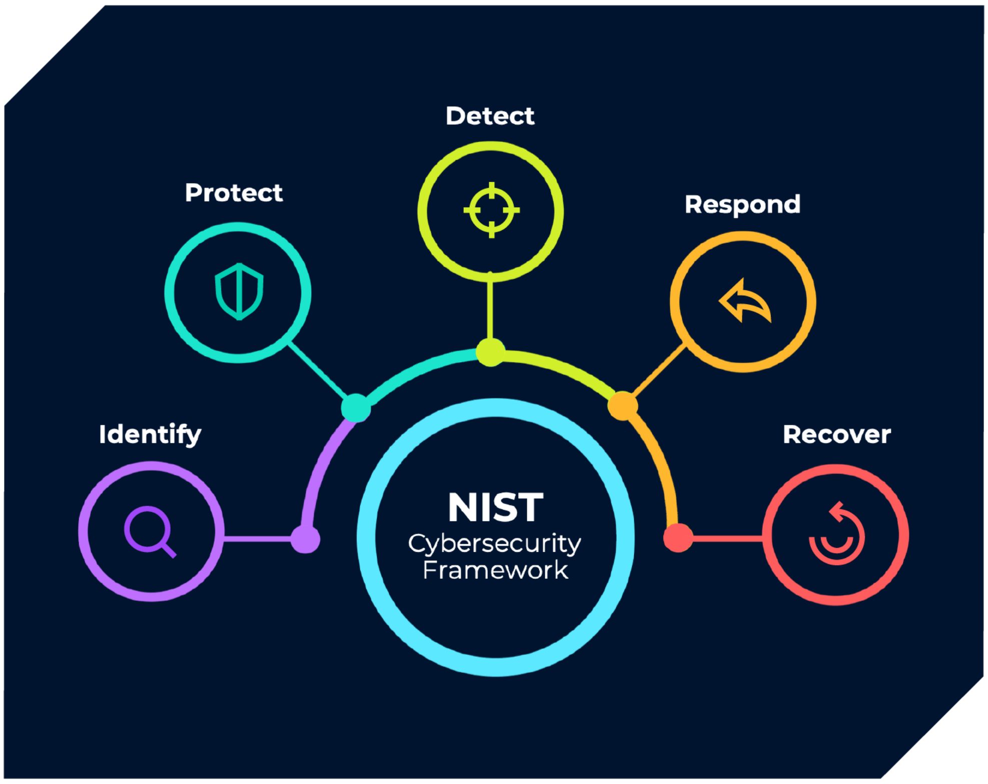


Figure 1.1: NIST working methodology that is followed by Q-ICT

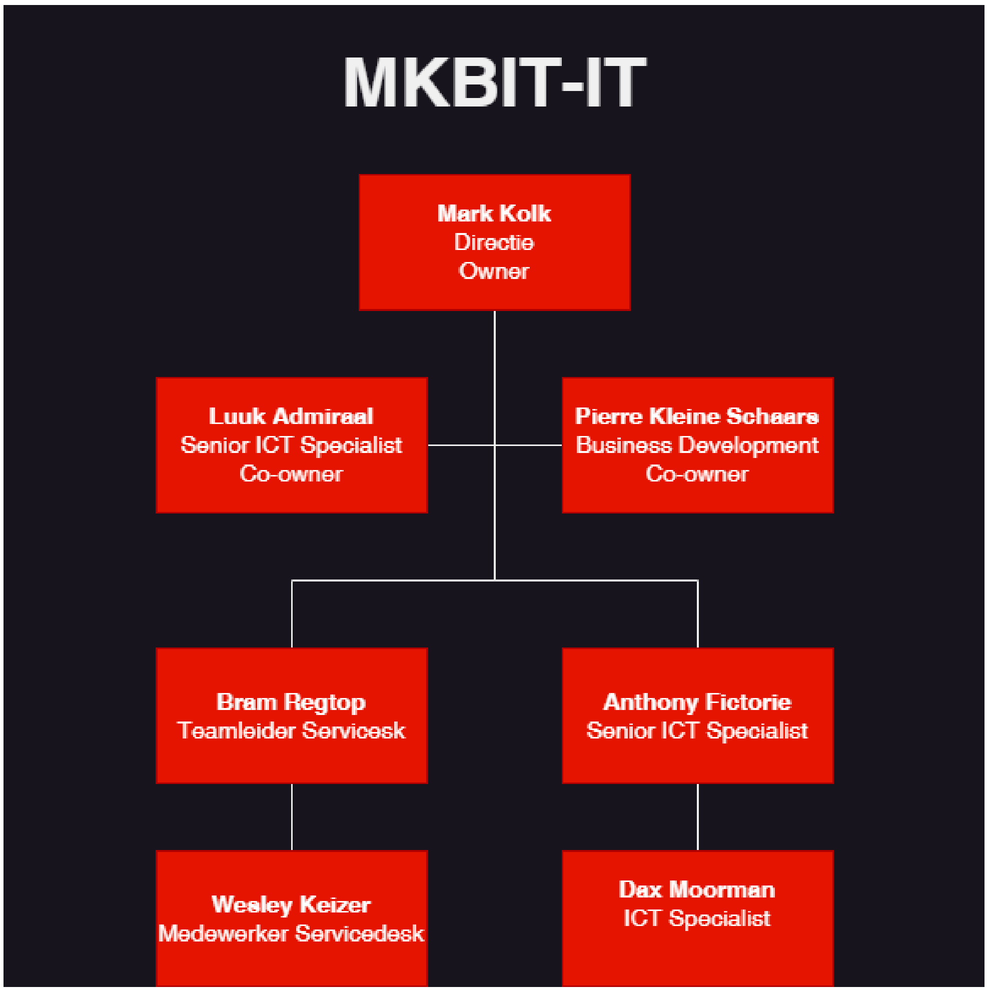


Figure 1.2: Level of structure of MKBiT

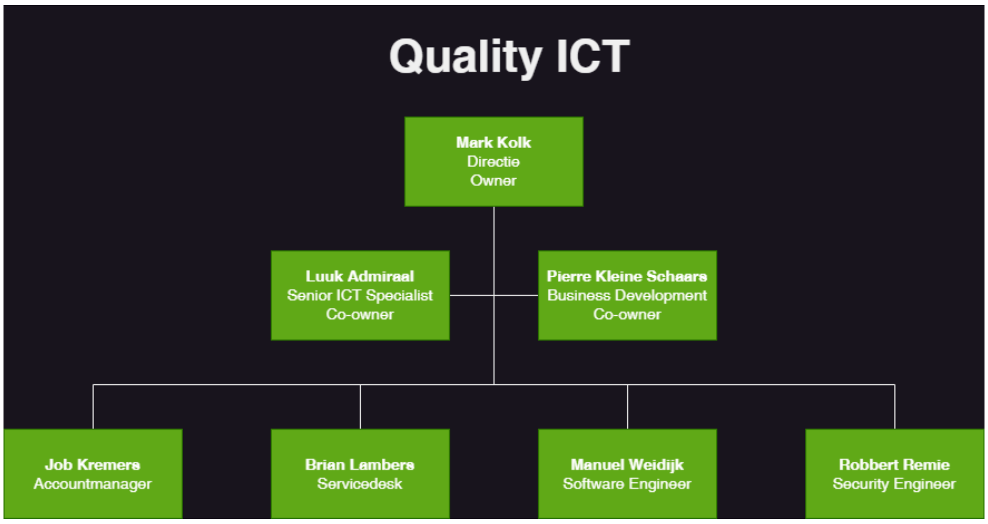


Figure 1.3: Flat (hierarchical) structure of Q-ICT

Chapter 2

Research Design

2.1 Research Topic

In research, it is paramount to have the formulation of a clear research topic, research main question, and research sub-questions. The main question serves as the focal point around which the research revolves, encapsulating the primary objective or purpose of the study. The following main research question will be used throughout the research:

"How can Quality ICT B.V. effectively integrate and leverage SentinelOne EDR platform for continuous cybersecurity monitoring?"

2.2 Research Sub-Questions

The research sub-questions are then used to function as a pathway that dissects the main question into smaller, more manageable components, which can then be addressed individually. This approach allows for a more comprehensive and in-depth analysis of the research topic, ensuring that all relevant aspects are covered and that the research is conducted in a systematic and organized manner. This research main question is therefore expanded in the following research sub-questions:

1. What is the current situation of the QaaS app of Quality ICT B.V.?
2. What is SentinelOne EDR platform, and how does its integration enhance the cybersecurity monitoring capabilities of the QaaS app?
3. How can SentinelOne be integrated into the QaaS app environment, while still utilizing their key features and capabilities in context of cyber-threat detection and remote IT infrastructure management?
4. What are the most suitable visualization techniques for displaying the data processed and received by SentinelOne API in Flutter compare to other Security Threat Platforms to ensure clear and insightful representation of threats?

2.3 Research Methodology

In this research, different research methods have been used to answer the research questions. This research will be based on the six ICT research methods defined by HBO-I (Vogel, 2023). A research method for each sub-question is then defined along with how the results are considered valid and reliable:

2.3.1 Method of Data Collection

- Sub-question #1: desk research of Literature Study will be conducted, with the goal of creating infrastructure information that displays the structure of the QaaS app and all its dependencies. Furthermore, Interview with key stakeholders involved in the development, maintenance, and usage of the QaaS app will be conducted to gain insights into the current situation of the app.

- Sub-question #2: Literature Study on various articles on the Internet, interviews, expert reviews, and requirement elicitation techniques such as use case analysis and user stories. Analysis on the current QaaS app and its capabilities.

- Sub-question #3: technical assessments and will be conducted, under the supervision of the Company Supervisor, to evaluate the technical feasibility, compactibility, and alignment of SentinelOne's features with the QaaS app environment.

- Sub-question #4: research into existing visualization techniques for JSON data coming from SentinelOne API through Literature Study. Analyze existing data visualization tools and platforms that are available in Flutter and Firebase. Gather requirements from project stakeholders regarding data visualization preferences and usability, and do data analysis and usability testing.

2.3.2 Selected Measuring Instruments

- Sub-question #1: structured interview guide, document report checklist analysis and review, observation, analysis tools for codebase and logs, and quite possibly supplemented by surveys or questionnaires.

- Sub-question #2: document analysis tools for literature review. Structured questionnaires for requirement interviews regarding functionality rating scale and compare the response against industry standards and best practices. Observation of existing API monitoring tools. Prioritize functionalities based on importance, feasibility, and impact on the QaaS app.

- Sub-question #3: technical assessments and requirement workshops will be conducted. Furthermore, API documentation review, document analysis tools, security impact risk assessment, and feasibility checklist assessment with the Company Supervisor will also be overseen.

- Sub-question #4: the selected measuring instruments for this sub-question will be through observation of existing data visualization tools, literature review through reading the studies of the best visualization suitability matrix techniques, structured questionnaires to end-user interviews, and usability testing heuristics.

2.3.3 Method of Data Analysis

- Sub-question #1: a qualitative thematic SWOT analysis of interview transcripts and documentation for operational insights to identify strengths, weaknesses, and areas for improvement in the current situation of the QaaS app.

- Sub-question #2: comparative analyze survey/interview responses using MCDA and compare against industry standards and best practices. Prioritize functionalities based on importance, feasibility, and impact on the QaaS app.

- Sub-question #3: evaluate the technical feasibility, compactibility, and alignment of SentinelOne's features with the QaaS app environment. Analyze potential integration challenges and mitigation strategies and assess the performance of the integration through prototyping and testing. Technical analysis for the API documentation and thematic analysis for interview data.

- Sub-question #4: technical tool analysis by reviewing and evaluating the suitability of different visualization techniques for representing the data processed and received by the QaaS app in XML and JSON formats from the API considering factors such as clarity, interpretability, and user engagement. Do a user-centered design and cognitive load analysis by analyzing feedback from company stakeholders, supervisor, and end-users.

2.3.4 Reliability, Validity, and General Applicability

- Sub-question #1: the reliability of the data can be ensured by triangulation of data from multiple sources and conducting interviews with stakeholders from different departments with structure questionnaires to ensure that the data is consistent and accurate. The validity of the data will be ensured by cross-referencing with the existing literature or industry best practices or other sources and through information obtained from interviews with the QaaS app developers to ensure that the data is accurate and reliable. The general applicability of the data will be ensured by ensuring that the information obtained is relevant and applicable to the research question and that it can be used to draw meaningful conclusions and make informed decisions, furthermore by comparing findings with industry standards and best practices or similar case studies or projects.

- Sub-question #2: ensure reliability through sampling techniques and representative stakeholder involvement, with comprehensive literature review and multiple sources of information. Validate priorities against real-world scenarios or case studies involving diverse expert panel, like the Company Supervisor. General applicability can be assessed by comparing prioritization with similar projects or frameworks, and considering scalability and adaptability of the integration with representative user samples.

- Sub-question #3: the validity of this sub-question will be through pilot integration unit testing or proof of concept documents and ensuring alignment with cybersecurity standards and best practices. The reliability will be to consider future needs such as adaptability and scalability of the integration, and focus on Q-ICT user context and needs. General applicability can be assessed by comparing integration strategies with industry standards or expert opinions such as from the Company Supervisor.

- Sub-question #4: reliability can be defined by ensuring future adaptability with comprehensive literature review and multiple sources of information. Validity can be achieved through validating visualization choices through data-driven approach in usability testing or prototyping, ensuring alignment with best practices in data visualization and involving the expert, like the Company Supervisor on the field. General applicability can be assessed by accessibility considerations by comparing proposed visualization techniques with similar applications or domains.

2.3.5 Research Limitations

The project and research in general will be limited on the API request methods, in which the author is allowed to do only GET requests. This is due to the fact that the author is not allowed to do any PATCH, POST, PUT, DELETE, or any other HTTP request methods that could potentially change the state of the QaaS app or the API that is being requested. This limitation is because the author is not a full-time employee of Q-ICT and is not allowed to make any changes to the QaaS app or the API that is being requested. Therefore, the author is limited to do research in the best practices of SentinelOne integration for the GET request method only.

The author is also limited in showing the SentinelOne dashboard data, as it contains clients' sensitive information, and Q-ICT has over 400 clients. Therefore, if any part of the SentinelOne dashboard is shown, it will be with blurred sensitive information.

Moreover, the author is also limited to the non-disclosure agreement signed within the initialization period of the graduation work placement. This means that any confidential information that the company deems as confidential will not be disclosed in this research. This includes any information that is not publicly available, such as any financial data or security data pertaining to the internal system or the QaaS app internal code.

Chapter 3

Research Results

3.1 Introduction

This chapter presents the results of the research conducted to answer the research questions. The chapter is divided into two sections, each corresponding to one of the research questions. The first section presents the results of the research conducted to answer the first research question, while the second section presents the results of the research conducted to answer the second research question.

3.2 Research Sub-Question #1: What is the current state of the QaaS app?

The QaaS app is an ERP web application that is used by Q-ICT and its clients. For Q-ICT's clients, it is a SaaS that is used to For Q-ICT employees themselves, it is an ERP system that is used to manage the clients and their ICT infrastructure. It is made in Dart with Flutter as the front-end framework. There are 2 main parts of the QaaS app, the front-end and the back-end. The front-end is made in Flutter, and the back-end is made in Node.js with TypeScript as the template. The back-end is hosted on Firebase Cloud Functions which are used to connect and make HTTP calls to the internal APIs, and the front-end is hosted on Firebase Hosting.

APIs and Technologies

The QaaS app also utilizes several APIs and technologies to help with its operations. It needs to manage and make connection different sort of APIs to help with the operations of the ERP application. Those APIs are the following:

- Resello: is used for Q-ICT MS subscriptions owned by Pax8 (*LinkedIn*, n.d.). It is a cloud marketplace that simplifies the way SMEs buy, sell, and manage cloud solutions through automation. It provides a single platform to manage the entire cloud customer lifecycle, from quote to cash to support, thus simplifying the process of buying, selling and managing cloud solutions. Furthermore, it normally uses SOAP API for its communication.

- SnelStart: is used for Q-ICT automation of financial and accounting system software, such as managing invoices, etc., for SMEs. It offers a range of products and services to help businesses manage their finances, including accounting software, invoicing software, and financial management tools.

- Bodyguard.io: is a CDR tool used for security tab. It is a product from a Dutch company that filters and scrutinizes downloads from web browsers to detect and prevent malicious files with real-time download scanning capabilities. It normally uses RESTful API for its communication.

- N-Central: is a product from N-Able and is used for monitoring clients' devices and ensuring the overall security of their systems, IT infrastructure, and digital assets. It is a RMM platform designed to help MSP and IT professionals to remotely monitor and manage their clients' devices and networks. It provides a comprehensive set of tools and features for monitoring, managing, and securing clients' devices and networks, including remote monitoring and management, patch management, antivirus, backup and disaster recovery, and network topology mapping. The return response from this API is in XML and JSON format, making it both a REST and SOAP API.

- PerfectView: is used for CRM software (*PerfectView*, n.d.). It is designed to improve business relationships with customers, assist in customer retention, and drive sales growth. In the QaaS app, it is used to manage the relationships and interactions with the app's users, which could include tracking user interactions, managing customer support requests, and analyzing user data to improve the app's functionality and UX.

Besides all the 5 internal APIs that Q-ICT uses, the company also uses several technologies to help them with their operations. Those tools are:

- Computicate (now newly named Acronis): is used as their ticketing system, providing ticketing solution services to customers for a wide range of events and activities (*Acronis, n.d.*).
- TOMTelecom: is used for their company's phone system. It is responsible for structured process of call routing on incoming calls from customers to the appropriate department or individuals, ensuring effective communication and issue resolution (*TOMTelecom, n.d.*).

3.2.1 The QaaS App Infrastructure

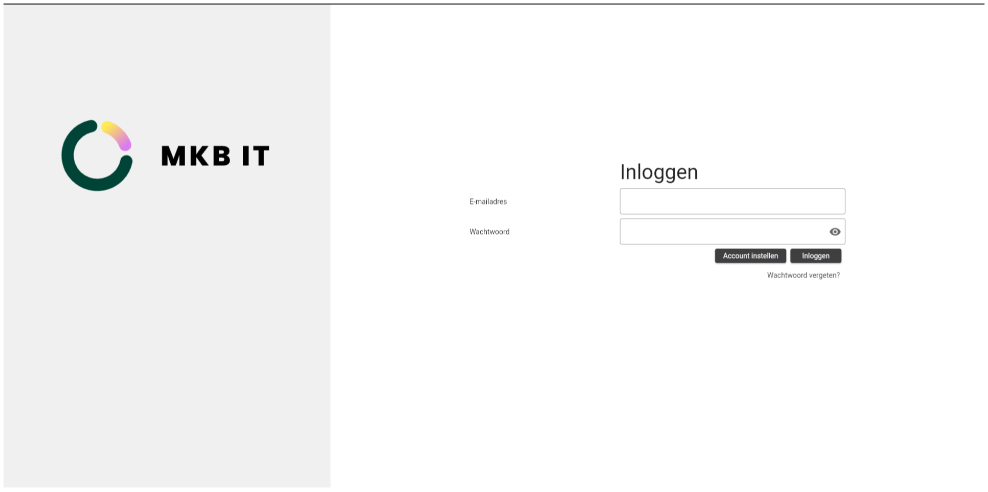


Figure 3.1: The login page of the QaaS app, implementing Google 2FA and reCAPTCHA

Firebase

Firebase is a comprehensive platform for developing and managing web and mobile applications, created by Google and is party of GCP. It was originally an independent company founded by Firebase, Inc. in 2011. It was then acquired by Google in 2014. Since then, it has become an integral part of Google's broader ecosystem of cloud services (Wikipedia, n.d.-i). It is a BaaS that provides developers with a variety of tools and services to help with both back-end infrastructure and front-end capabilities without worrying about managing servers or infrastructures. The services offered by Firebase (*Firebase, n.d.-c*) are many, but for this thesis, it will only discuss the ones that are used by the QaaS app. They are listed in the following:

- Authentication: is an easy-to-understand authentication services that support various authentication methods like email/password, phone number, with identity providers such as Google, Facebook, Twitter, Apple, GitHub, etc., along with utilizing 2FA authentication factors to enhance security by requiring additional factor, such as an OTP code that is sent to the user's phone or security key.
- Database:
  - Firestore Database: Firestore is a NoSQL database that is part of the Firebase platform. It is a flexible, scalable database for mobile, web, and server development. It keeps data in sync across client apps through real-time listeners and offers offline support for mobile and web, so the developers can build responsive apps that work regardless of network latency or Internet connectivity.
- Cloud Functions: often just called Functions in the Firebase console, it allows developers to run back-end code in response to events triggered by Firebase features and HTTPS requests. The code is stored in Google's cloud and runs in a managed environment. It is a serverless framework that allows developers to build and deploy serverless functions that automatically scale up and down based on demand. The available programming languages are Node.js (JS and TS), Python, Go, Java, and .NET (C#). Cloud Functions offers 2 product versions: the original version (1st gen), and the 2nd gen which is built on Cloud Run and Eventarc to provide an enhanced feature set.
  - 1st Generation: Most of the Firebase Cloud Functions that are used in the QaaS app is in this version. The company wishes to migrate all the functions to the 2nd generation in the future. Furthermore, for the integration of SentinelOne with the QaaS app, the company wishes to utilize the 2nd generation of Cloud Functions.
  - 2nd Generation: The company wishes that the author's graduation project will utilize the 2nd generation of Cloud Functions. Features in the 2nd generation including:
    * Longer request processing times

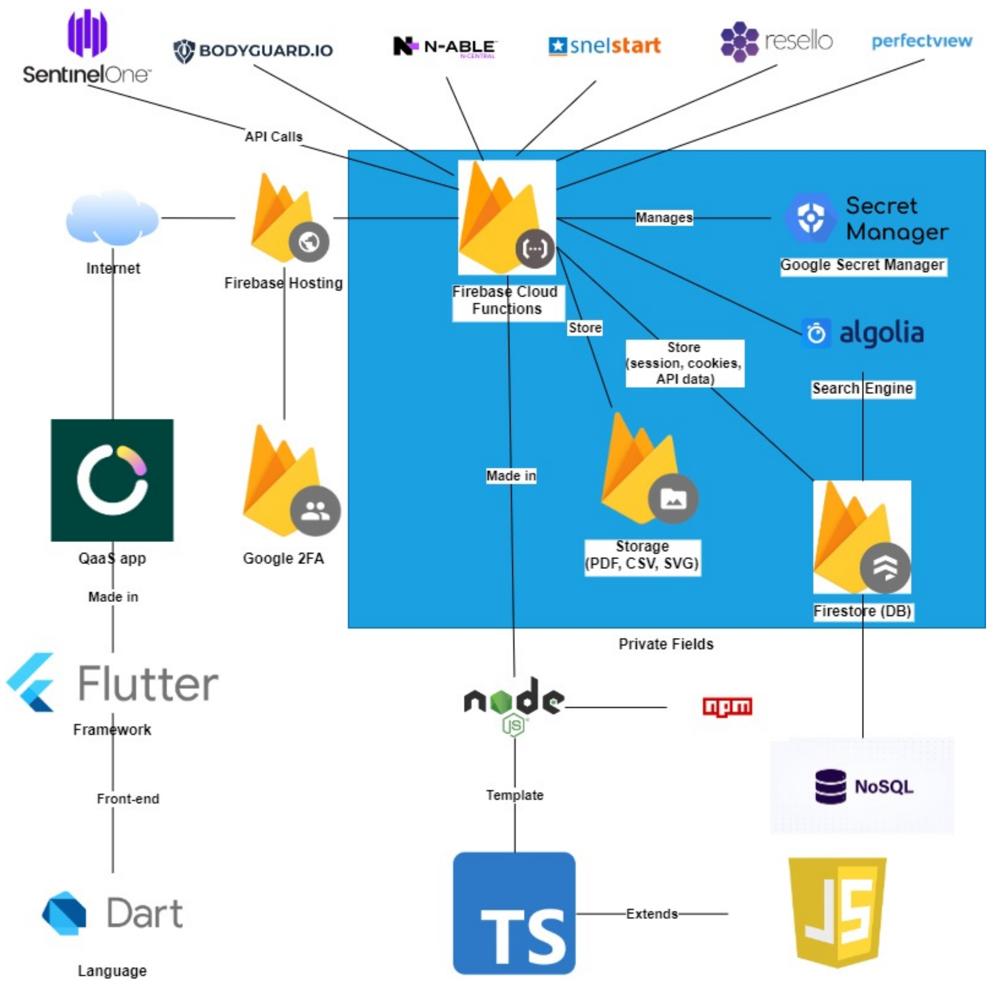


Figure 3.2: The infrastructure of the QaaS app

* Larger instance sizes
* Traffic management
* Eventarc integration
* Broader CloudEvents support

Cloud functions are the main back-end infrastructure of the QaaS app. It is used to connect and make HTTP calls to all the internal APIs. There are different types of functions in Firebase Cloud Functions, and they will be discussed later. Some functions are called within the app, and some outside of the app. Some functions are also used to listen to the Firestore collections, in case of any changes in the data.

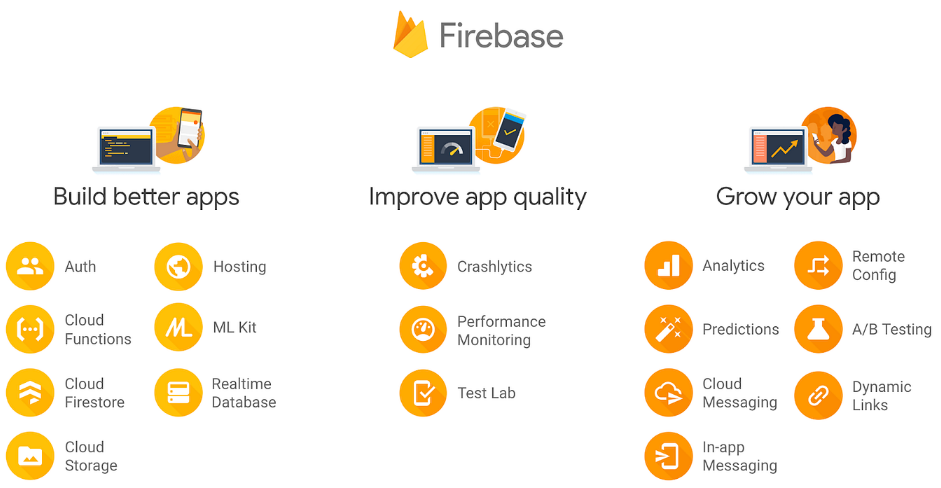
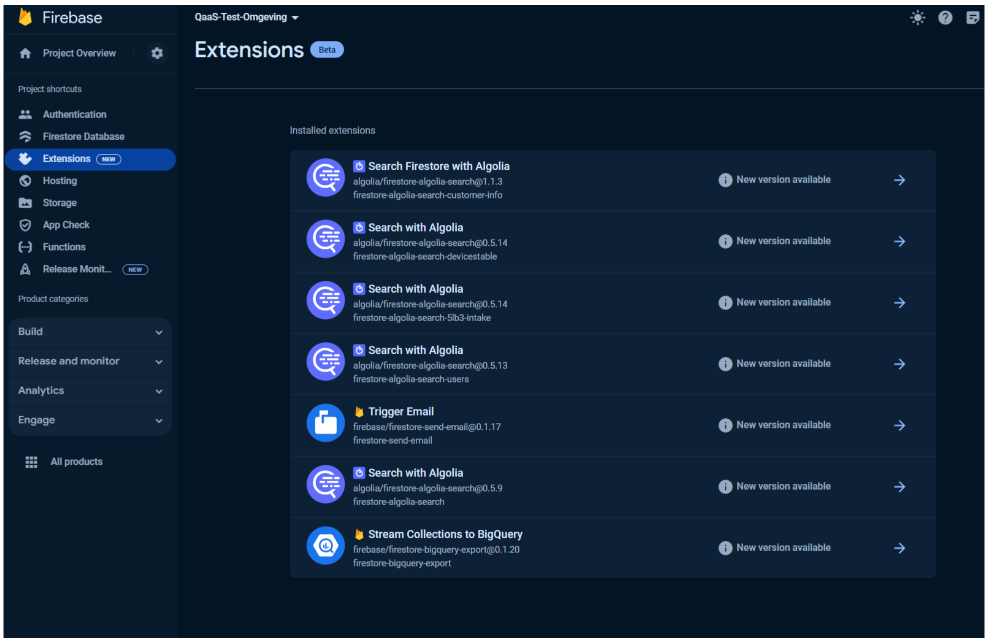


Figure 3.3: All of the products offered by Firebase (*Kubernetes, 2018*)

Feature	1st Gen	2nd Gen
Image registry	Container Registry or Artifact Registry	Artifact Registry only
Request timeout	Up to 9 minutes	• Up to 60 minutes for HTTP-triggered functions • Up to 9 minutes for event-triggered functions
Instance Size	Up to 8GB RAM with 2 vvCPU	Up to 16GB RAM with 4 vCPU
Concurrency	1 concurrent request per functions instance	Up to 1000 concurrent requests per function instance

Table 3.1: Comparison between the 1st and 2nd Generation of Cloud Functions

- Hosting: a service that allows developers to host static websites, dynamic web apps, mobile apps, and microservices on Firebase's infrastructure. The QaaS app is currently hosted on Firebase Hosting.
- Cloud Storage: offers secure, scalable, and reliable file storage and sharing for Firebase apps. It is designed to help developers quickly and easily store and serve user-generated content, such as photos or videos. It is used by the QaaS app to store and serve user-generated content, such as profile pictures, documents, and other files.
- Extensions: it consists pre-built, open-source software packages that extend the functionality of a Firebase project (*Firebase, n.d.-b*). They are designed to automate common development tasks, such as sending notifications, integrating with third-party services, and performing back-end operations, without requiring users to write custom code. The QaaS app uses the Extension mainly for Algolia.
- App Check: it is a security feature that, on top of Firebase 2FA Authentication, that helps protect the project from abuse, such as billing fraud or phishing, by ensuring that only the app that is registered can have access to the Firebase project's resources (*Firebase, n.d.-a*). In the case of the QaaS app, it uses it reCAPTCHA to ensure extra protection in the MFA.

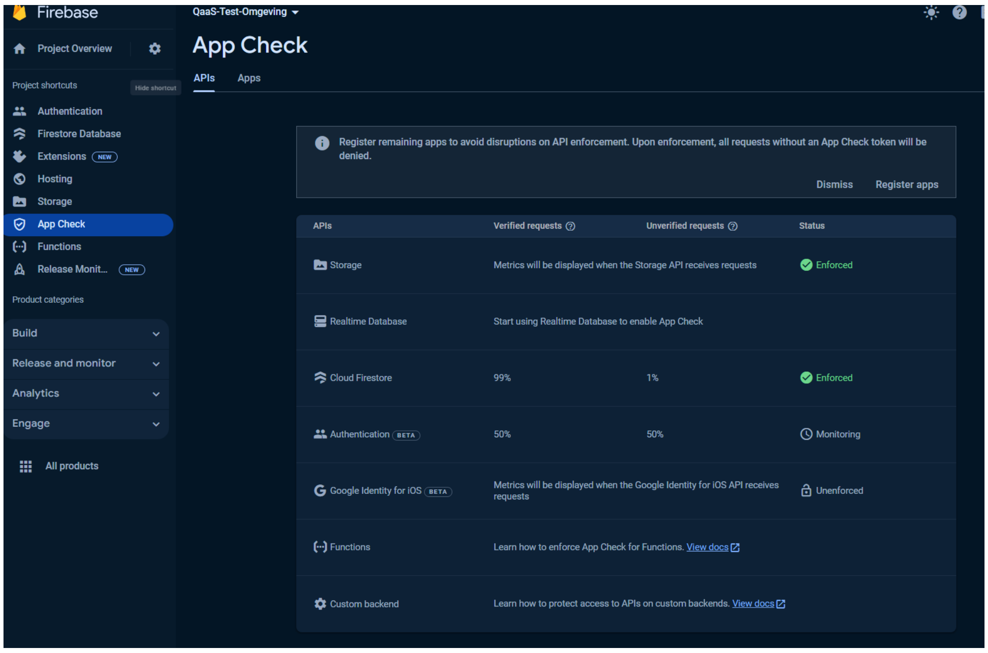


Extensions Beta

Installed extensions

Search Firestore with Algolia algolia/firestore-algolia-search@1.1.3 firestore-algolia-search-customer-info	New version available	→
Search with Algolia algolia/firestore-algolia-search@0.5.14 firestore-algolia-search-devicestable	New version available	→
Search with Algolia algolia/firestore-algolia-search@0.5.14 firestore-algolia-search-5lb3-intake	New version available	→
Search with Algolia algolia/firestore-algolia-search@0.5.13 firestore-algolia-search-users	New version available	→
Trigger Email firebase/firestore-send-email@0.1.17 firestore-send-email	New version available	→
Search with Algolia algolia/firestore-algolia-search@0.5.9 firestore-algolia-search	New version available	→
Stream Collections to BigQuery firebase/firestore-bigquery-export@0.1.20 firestore-bigquery-export	New version available	→

Figure 3.4: All the extensions that are used in the QaaS app, mostly about Algolia



App Check

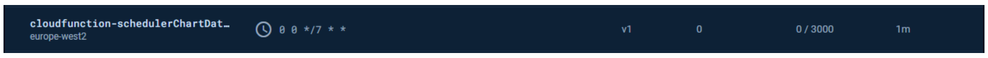
APIs Apps

Register remaining apps to avoid disruptions on API enforcement. Upon enforcement, all requests without an App Check token will be denied.

Dismiss Register apps

APIs	Verified requests	Unverified requests	Status
Storage	Metrics will be displayed when the Storage API receives requests		✓ Enforced
Realtime Database	Start using Realtime Database to enable App Check		
Cloud Firestore	99%	1%	✓ Enforced
Authentication BETA	50%	50%	Monitoring
Google Identity for iOS BETA	Metrics will be displayed when the Google Identity for iOS API receives requests		Unenforced
Functions	Learn how to enforce App Check for Functions. View docs		
Custom backend	Learn how to protect access to APIs on custom backends. View docs		

Figure 3.5: The App Check feature in the QaaS app



```
cloudfunction-schedulerChartDat…   0 0 */7 * *        v1     0     0 / 3000     1m
europe-west2
```

Figure 3.6: The function in the QaaS app that is scheduled to run every 7 days to keep the API data up-to-date (*Cronitor, n.d.*)

Different Types of Cloud Functions in Firebase

Firebase has a lot of different types of Cloud Functions that developers can use. Just like the previous Firebase product explanation, this thesis will only focus on the following types of Cloud Functions that are used in the QaaS app:

HTTP Triggers: these are functions that are triggered by HTTP requests. They are used as API endpoints of the QaaS app, allowing server-side logic execution in response to HTTP requests from client-side applications or external services. The requests are GET, POST, PUT, DELETE, and PATCH, and they are used to creating, reading, updating, and deleting data in either the Firestore DB or the correlated API environment itself.

- onRequest: this method is used to create an HTTP function that is triggered by an HTTP request. They are more general-purpose compared to Callable Functions and can be used to create RESTful APIs, handle form submissions, or perform any other server-side operations that require HTTP requests. Unlike Callable Functions, onRequest functions do not handle authentication or data serialization, so the developers need to manage this aspect manually. This functions only accepts `Request` and `Response`, but not `NextFunction`. In the context of QaaS app, this function is only used for testing purposes, and the developer will need to migrate to onCall when deploying on the Live environment.

- onCall: is a little different from onRequest. Instead of using Request and Response, it uses data and context. In version 2.0, it only accepts request as `CallableRequest<any>` that can get any headers and body of the request sent by user. It is used to create Callable Functions, and they are designed to be called directly from client applications, such as mobile or web apps. They automatically handle authentication and data serialization, making it easier to secure call backend code from client applications, and this is what the QaaS app primarily uses for calling its internal APIs as it ensures that the client is authenticated and authorized to make the call. Callable functionsa are triggered by an HTTP request but are specifically designed to be called from Firebase client SDKs.

Listing 3.1: Example of a typical onRequest function

```
import { Request, Response } from 'express';
import * as functions from "firebase-functions/v2";

const region = "europe-west1";

export.getData = functions.https.onRequest({ region: region }, async (request: Request, response:
    Response): Promise<any> => {
    try {
        const response = SentinelOneAPICall();
        return response.status(200).json(data: response.data);
    } catch (ex: unknown) {
        if (error instanceof Error) {
            // Error object, log message and stack if available
            console.error(`[${context}] An error occurred: ${error.message} \n Stack:
                ${error.stack}`);
        } else {
            // Non-Error object, log with a generic message.
            console.error(`[${context}] An unknown error occurred:`, error);
        }
        response.status(500).send("Failed to retrieve agents");
    }
});
```

Listing 3.2: Example of onCall function with authentication check

```
import * as functions from "firebase-functions/v2";
const region = "europe-west1";

const getData = functions.https.onCall({ region: region }, async (request: CallableRequest<any>) =>
    {
    try {
        // Checking that the user is authenticated.
        if (!context.auth) {
            // Throwing an HttpsError so that the client gets the error details.
            throw new functions.https.HttpsError('failed-precondition', 'The function must be
                called while authenticated.');
        }
```

```
            const response = SentinelOneAPICall();
            return {
                data: response.data,
            };
        } catch (ex: unknown) {
            if (error instanceof Error) {
                // Error object, log message and stack if available
                console.error(`[${context}] An error occurred: ${error.message} \n Stack: ${error.stack}`);
            } else {
                // Non-Error object, log with a generic message.
                console.error(`[${context}] An unknown error occurred:`, error);
            }
            throw new functions.https.HttpsError("unknown", "Failed to retrieve agents", ex);
        }
    });

    async function SentinelOneAPICall() : Promise<any> {
        const apiUrl = 'https://sentinelOne.example.com/users/123';

        // Define the request parameters
        const requestOptions = {
            method: 'GET', // HTTP method (GET, POST, PUT, DELETE, PATCH, etc.)
            headers: {
                'Content-Type': 'application/json', // Set the content type of the request
            }
        };

        // Make the API request
        fetch(apiUrl, requestOptions)
            .then(response => response.json())
            .then(data => console.log(data)) // Process the response data
            .catch(error => console.log('error', error)); // Handle any errors that occurred during the request
    }

    export = getData;
```

Pub/Sub Triggers: are the functions triggered by messages published to a Pub/Sub topic. It is a messaging service that enables decoupling of applications by sending messages between independent components.

Cloud Firestore Functions

Schedule functions: this is a Firebase's own term for Cron Job (*Firebase, n.d.-e*). They are used within the QaaS app to run tasks at regular intervals, such as sending reminders, keeping data up-to-date with the internal APIs (like customer list), or performing maintenance tasks.

Algolia

Algolia is used for search functionality. It is a search-as-a-service platform that enables developers to integrate and build fast, relevant search functionality into their applications and websites (*Wikipedia, n.d.-a*). It provides a range of features and capabilities for building and managing search functionality, including full-text search, typo tolerance, and relevance tuning, as well as analytics and monitoring tools to help developers understand how users are interacting with their search functionality in real-time.

The reason as to why Q-ICT uses Algolia is that the nature of Firebase search engine is quite often proven to be inaccurate and slow.

Google Secret Manager

It is a fully managed service provided by GCP that allows developers and organization to securely store, access, and manage sensitive information such as API keys, passwords, certificates, OAuth credentials, DB credentials and other credentials used in throughout the lifecycle of their applications (*Google, n.d.*). It is not part of Firebase, and it helps the QaaS app to centralize and secure its secrets in scalable and easily manageable way. Key-features of Secret Manager include:

- Secure Storage: it encrypts the secret values using CMEK, ensuring the sensitive data is protected both at rest and in transit.
- Audit Logs: it provides and manages audit logs that record all access and modification of activities, helping developers meet compliance, better accountability and regulatory requirements.
- Versioning and Automatic Rotation: it supports versioning of secrets, allowing developers to store multiple versions of the same secret. This means that the developers get to keep multiple versions of secrets and easily revert or roll back to a previous version if needed, which will help in auditing and tracking changes to secrets over time. This feature enables automatic seamless rotation of secrets at regular in-

tervals without disrupting the applications, which improves the security part of the application by ensuring that secrets are regularly updated without manual intervention.

- Access Control: it provides fine-grained access control using Google IAM, allowing developers to specify who can access and manage the stored secrets and what they can do with them.
- Centralized Management: it stores and manages all secrets in one place, simplifying access and control.

Google Secret Manager comes from GCP, and GCP and Firebase are a separate cloud solutions. But, because both are part of Google, Firebase Cloud Functions can typically access GCP Secret Manager by editing that specific function that the developer wanted to grant access to.

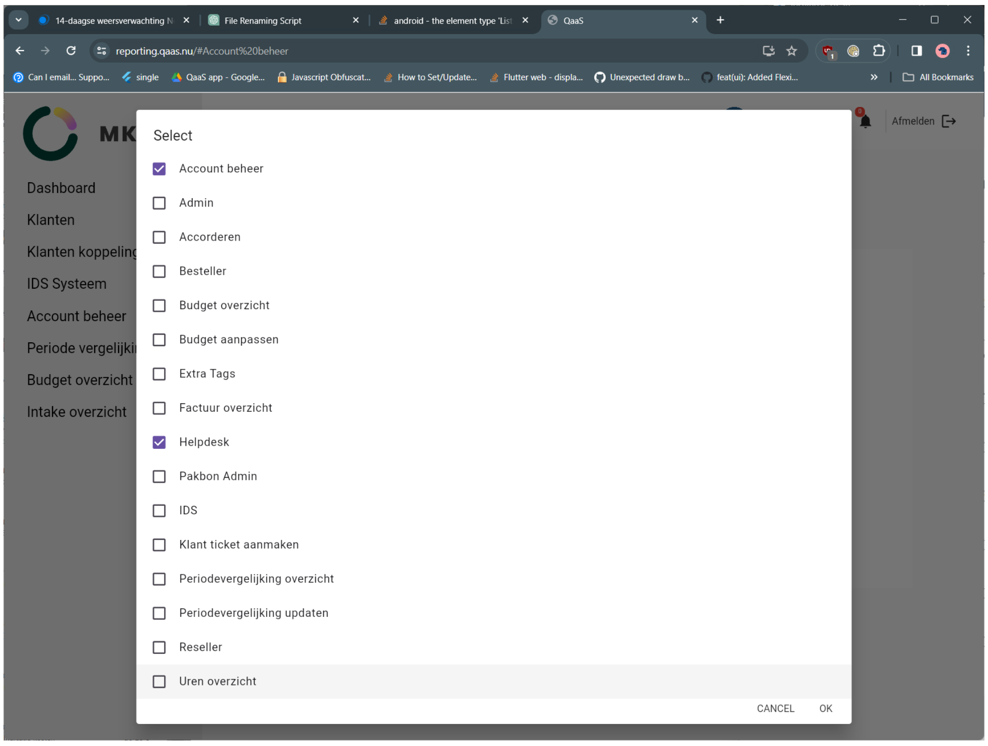
Modules and Templates of the QaaS App

The QaaS has several templates and modules that are used to build the app. The templates can be interpreted as level of access to the app, showing what permission a user has to the app. There are 3 different templates based on 3 different users:

- **Clients**: this template is used by the general users of the app. They have limited access to the app's features and functionalities, such as viewing and updating their profile and accessing the app's resources. They can only see the data relevant to their own account and cannot access or modify data belonging to other client from different company. The Client template is designed for users who have the lowest level of access and control over the app.
- **Helpdesk**: this template is used within the Helpdesk department of Q-ICT. They have more access to the viewing of app functionality than the Clients.
- **IT Admin**: the IT admin have almost the same level of access as the Helpdesk. The only difference is that they have the ability to create, edit, and delete users, manage user permissions, and configure the app settings. They can also grant and revoke access to a user and to the app's features and functionalities, which are called Modules. The admin template is designed for users who have the highest level of access and control over the app, which is to the developers.

Modules are

User tags are used for differentiating different companies.



Select

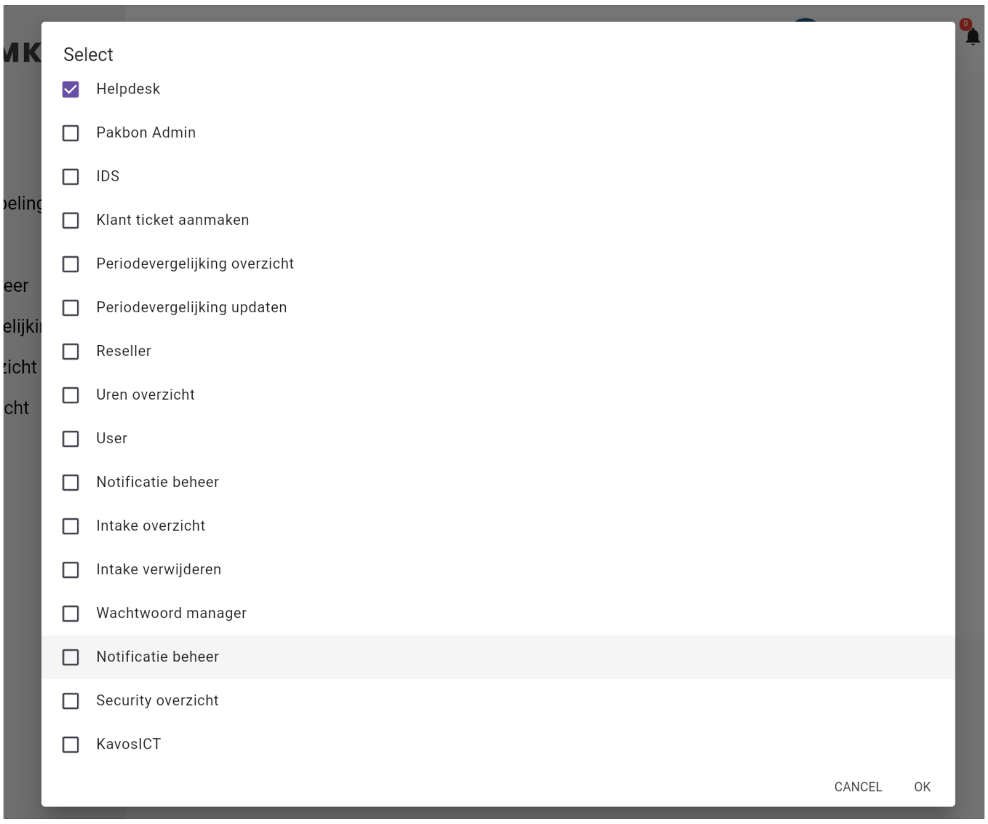
- [✓] Account beheer
- [ ] Admin
- [ ] Accorderen
- [ ] Besteller
- [ ] Budget overzicht
- [ ] Budget aanpassen
- [ ] Extra Tags
- [ ] Factuur overzicht
- [✓] Helpdesk
- [ ] Pakbon Admin
- [ ] IDS
- [ ] Klant ticket aanmaken
- [ ] Periodevergelijking overzicht
- [ ] Periodevergelijking updaten
- [ ] Reseller
- [ ] Uren overzicht

CANCEL OK

Figure 3.7: Different modules in the QaaS app in which the IT admin can determine which pages can be accessed by which user tags

3.2.2 Conclusion

The QaaS app is an ERP web application that is used by Q-ICT and its clients. It is made in Dart with Flutter


Select

- [✓] Helpdesk
- [ ] Pakbon Admin
- [ ] IDS
- [ ] Klant ticket aanmaken
- [ ] Periodevergelijking overzicht
- [ ] Periodevergelijking updaten
- [ ] Reseller
- [ ] Uren overzicht
- [ ] User
- [ ] Notificatie beheer
- [ ] Intake overzicht
- [ ] Intake verwijderen
- [ ] Wachtwoord manager
- [ ] Notificatie beheer
- [ ] Security overzicht
- [ ] KavosICT

Figure 3.8: Different user tags in the QaaS app in which correlates to what actions can a user do within the web application

as the front-end framework, and Node.js with the back-end framework with TypeScript as a template to ensure type safety. The back-end is hosted on Firebase Cloud Functions, and the front-end is hosted on Firebase Hosting. The app uses several internal APIs and technologies to help with its operations, such as Resello, SnelStart, Bodyguard.io, N-Central, and PerfectView. The app also uses several Firebase products, such as Authentication, Firestore Database, Cloud Functions, Hosting, Cloud Storage, Extensions, App Check, and Google Secret Manager. The app also uses Algolia for search functionality. The app has several templates and modules that are used for the authentication and authorization of the app. The templates control what 3 different types of users can do in the app, they are Clients, Helpdesk, and IT Admin. The modules help with the control of the app's features and functionalities, and the IT Admin has the highest level of access and control over the app.

3.3 Research Sub-Question #2: What is SentinelOne EDR Platform?

SentinelOne is a cybersecurity platform that provides endpoint protection, detection, and response capabilities to help organizations defend against advanced cyber threats. It leverages Artificial Intelligence and machine learning to analyze and respond to security threats in real-time, providing organizations with comprehensive protection against malware, ransomware, and other cyber threats. It also provides visibility into clients' IT systems and infrastructure, enabling organizations to gain insights into potential security risks and vulnerabilities and take proactive measures to address them.

Some terminology that the readers need to be familiar with before diving deeper into SentinelOne:

Endpoint

Endpoint can be defined as any remote computing devices that receives incoming communications and sends outgoing messages to the network it is connected to. Examples of endpoints include desktops, laptops, smartphones, tablets, servers, workstations, and other IoT devices that is connected to a network. They are the first-line of defence for the Blue Team today.

Examples of endpoints are:

- Computer (workstations, desktops)
- Laptop
- Server
- Mobile devices

EPP

EPP is an upgrade from legacy AV/anti-malware. EPP consists of solutions that work together to detect and block security threats at the endpoint device level.

Important note that EPP is not to be confused with EDR. EPP is a separate service from EDR, in which in EDR platforms they often include EPP but not vice versa.

EDR

EDR A.K.A. ETDR, is a group of integrated endpoint security solutions that combine data collection, data analysis, forensics, and Threat Hunting with the end-goal of identifying and stopping any potential security breaches in due time. EDR solutions are able to recognize any suspicious patterns that can be investigated later on, as they have been purposefully created to detect and respond in an active manner to advance malware, ransomware, and other cyber threats (the Response EDR). EDR, as the name suggest, were developed specifically for endpoints, and not networks (3.3), thus operate only on endpoint level.

The number one thing that sets apart EDR from legacy (traditional) AV is that traditional AV relies on signature-based detection, usually having a defined set of list in their DB, where known malware signatures are compared against files or processes to identify threats. EDR on the other hand, uses a combination of signature-based detection, such as behavioural analysis, machine learning, and anomaly detection to identify and respond both known and unknown threats. EDR solutions focus on detecting malicious activities at the endpoint level, including file modifications, process execution, and network connections, focusing on malicious behaviour compared to only concerning with malicious software like what traditional AV does.

MDR

MDR is what manage the EDR technology.

NDR NDR products are designed to provide a complete visibility into the network, real-time detection of threats, and guided investigation to accelerate and automate responses (*SentinelOne, n.d.*). NDR takes a feed of raw network traffic from a Network TAP, port mirror, or virtual traffic mirroring in AWS and Azure. By analyzing this traffic in real-time, NDR finally discovers and classify every device communicating on the network. It identifies device roles, such as DNS, web server, medical device, etc., and maps peer groups among those devices.

SIEM

is a software system that collects, aggregates, normalizes security data from a variety of sources within an IT infrastructure, and analyzes it according to pre-set rules, present it in human-readable format and therefore giving a comprehensive picture of the company's information security (Gartner, 2017). SIEM tools evolved from the log management discipline and combine SIM and SEM technologies. A SIEM tool uses AI to automate several manual procedures related to threat detection and incident response. Furthermore, it assists enterprise security teams in spotting anomalies in user behavior.

- Input: logs, threat intel, vulnerability feeds, NDR, firewall, IPS, IDS, and EDR data
- Output: high-fidelity alerts prioritized by severity
- Infused with: AI, ML, and analytics

SOAR

SOAR solutions focus on automating incident response processes and triage capabilities. The key word here is ”orchestration” and ”automation”. In an ideal world, everything. The main goal is to oversee security without human help as much as possible, boosting productivity and shortening the response time. It might use AI and ML to assess security events and automate incident response procedures. These solutions can be standalone product, or it can be added to SIEM solutions since SOAR does not excel in event analysis.

XDR

is a security solution that gathers and analyzes data from multiple sources like endpoints, networks, cloud, emails, app, etc., It offers great visibility into a company's IT infrastructure, helping the security employees to detect more threats, respond efficiently, and deal with fewer false positive alerts Vazquez, 2021.

This solution integrates several tools combining all the gathered data into a single platform to visualize the information. It might incorporate automated processes (even complex ones), ML, and advanced analytics to enable quicker and more effective incident response. It can even deal with hidden and advanced malware.

MXDR

Some people may still call it MDR, managed SOC, or managed security.

SOC

SOC is the organizational context itself. It is a centralized facility or team within an organization that houses a security team responsible for monitoring and analyzing another organization's (client) security position on an ongoing basis. They can be seen as the safety team for client organizations. Please note that SentinelOne company itself is not a SOC, as it is just a cybersecurity company that offers endpoint security solutions.

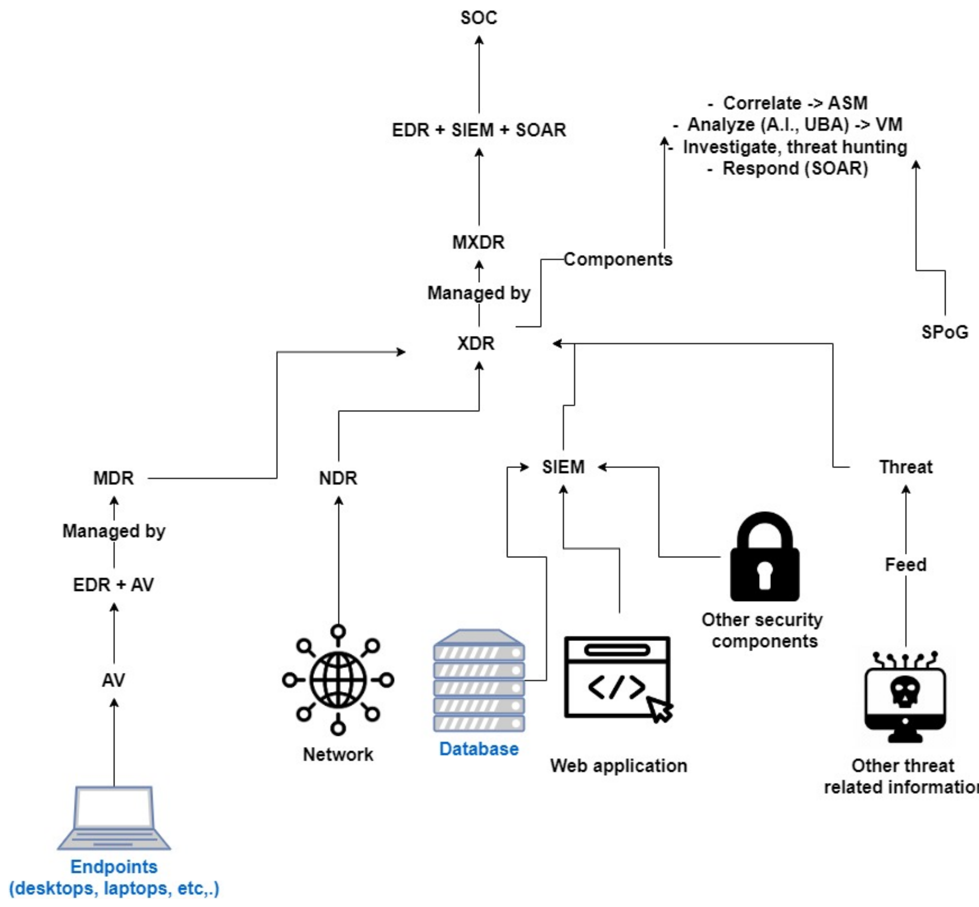


Figure 3.9: How all terms related to various technologies in cybersecurity connected to each other

Please note that as mentioned before, SIEM can also take data from an EDR and NDR but for the sake of simplicity, the diagram puts them as a separate pier systems. In terms of SentinelOne, it has all the features that are mentioned in the diagram, except for SOC, as was mentioned before.

Console

Global

The Global tab refers to the environment of the highest level of advisors. It consists of organizations that have access to global level that propagate down to the tenant level, which means they can put policies, scripts, and other configurations that will be applied to all console and all of their customers.

Tenant

Mentees

Site

Site is just a name

Group

Group refers to logical grouping of devices within SentinelOne MGMT Console. Groups can be created and deleted by the admin user, and

Agents

An Agent is a software program, part of SentinelOne product, that is deployed to each endpoint, including desktop, laptop, server or virtual environment, and runs autonomously on each device, without reliance on Internet connection, enabling data gathering, detection, and response to actions. Agents can be interpreted as an AV, collecting relevant security telemetry such as:

- Running processes
- Connected servers
- Open files

This information can be useful to detect the presence of a threat or to use in forensic analysis and investigation after an attack has occurred (Recovery).

Ranger

Ranger is an add-on SentinelOne product that provides a way of detecting other devices (computers and IoT devices) that are on the client's computer network. If a malicious attacker comes in and plugs his device into the network, all the other SentinelOne agents are going to read the network traffic, determine and classify whether that is a new device or a rogue device. As long as a device has Ranger on that network subnet, SentinelOne can gather and detect technical information regarding the device.

- Not reviewed
- Not trusted
- Under analysis
- Allowed

Ranger is designed to detect and take out malicious actors that are in the local subnet, as they can a lot of information about the devices (*see ARP Poisoning Wikipedia, n.d.-c and man-in-the-middle-attack Wikipedia, n.d.-k*). In the real-business scenario, a lot of the times, a company has a very secure perimeter firewall (*the big outer castle wall in castle-and-moat network security model Taylor, 2021*), but the inside network are wide open (unless they are doing logical separations of their local network, such as using VLAN) for attacks inside the network.

Unfortunately for this project scope, Ranger API calls are off limit, as the company is still..., therefore displaying all the Ranger ability to scan and the API call of the author's account.

Sentinels

Sentinels refer to a term that describes SentinelOne ability to deploy and manage security agents on endpoints within an organization, not part of SentinelOne product, like Ranger. This is part of the SentinelOne EDR capabilities, where the user admin can deploy the agents on the endpoints, and manage them from the SentinelOne console. The admin can also create policies, scripts, and other configurations that will be applied to all the agents in the network.

Visibility

Unfortunately, Q-ICT does not have Visibility turned on its tenant, thus this feature is also off limit for this project scope. But in general, Visibility allows users to do deep querying to be able to identify different things, such as attributes of a machine. It is useful when user is looking for threats or any additional information in Incident Response or Threat Hunting, which in most cases is proactive, Visibility can give the user a lot of power to use queries, and interrogate all the information or telemetry that SentinelOne has pulled off from the connected machines.

Incidents

Incident is just a name of a page in SentinelOne dashboard that provides a list of cyber-incidents overview that have happened on all endpoints detected by Agents in the network connected to SentinelOne by Ranger. The page typically offers an overview of all detected security incidents, categorized based on severity levels of types of threats.

Once a threat is detected, the user can take the following actions in the dashboard, if they have enough privileges to do so:

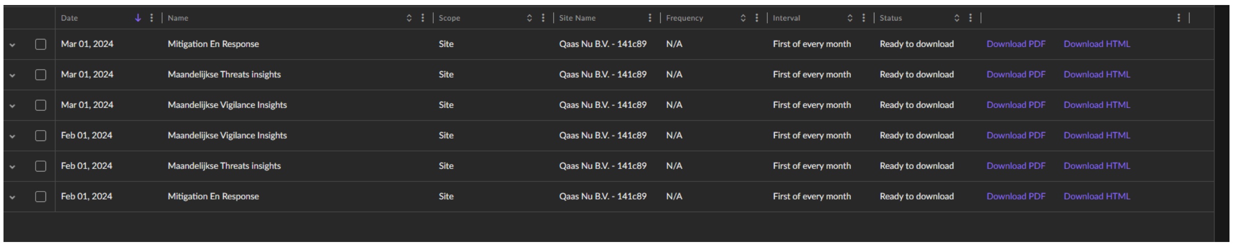
- Kill: stops all processes related to that threat

- Quarantine: encrypts and moves the threat and its executables
- Remediate: deletes all files and system changes created by the threat
- Rollback: restores files and configuration that the threat changed. This step is usually taken when a malware has executed its script and has made changes to the system, e.g., a ransomware has encrypted all the files and asked for a ransom. By taking this step, all the three previous steps will also be undertaken as well. This will then reboot the system and restore it to the safe state before the malware has been executed.

Reports

The reports in SentinelOne provide users with insights into the security posture and threat landscape across their organization's endpoints. The reports offer customizable reporting capabilities, allowing users to generate reports tailored to their specific requirements. Users can then choose from a variety of predefined report templates or create custom reports based on their unique needs.

Furthermore, the users can also choose for the report to be made automatically, instead of manually filling them themselves. They can schedule automated report generation at regular intervals, such as daily, weekly, or monthly.



Date	Name	Scope	Site Name	Frequency	Interval	Status		
Mar 01, 2024	Mitigation En Response	Site	Qaas Nu B.V. - 141c89	N/A	First of every month	Ready to download	Download PDF	Download HTML
Mar 01, 2024	Maandelijkse Threats insights	Site	Qaas Nu B.V. - 141c89	N/A	First of every month	Ready to download	Download PDF	Download HTML
Mar 01, 2024	Maandelijkse Vigilance Insights	Site	Qaas Nu B.V. - 141c89	N/A	First of every month	Ready to download	Download PDF	Download HTML
Feb 01, 2024	Maandelijkse Vigilance Insights	Site	Qaas Nu B.V. - 141c89	N/A	First of every month	Ready to download	Download PDF	Download HTML
Feb 01, 2024	Maandelijkse Threats insights	Site	Qaas Nu B.V. - 141c89	N/A	First of every month	Ready to download	Download PDF	Download HTML
Feb 01, 2024	Mitigation En Response	Site	Qaas Nu B.V. - 141c89	N/A	First of every month	Ready to download	Download PDF	Download HTML

Figure 3.10: Examples of automatic reports that can be downloaded in SentinelOne

Vigilance

It is a MDR service - providing threat monitoring, hunting, and response, to its existing customers. It provides a 24/7 SOC with expert analysts and researchers to give customers near real-time threat monitoring, in-console threat annotations, and response to threats and suspicious events. Vigilance itself is a separate package from SentinelOne.

How does SentinelOne get all this information from a device without having them even connected to the Internet?

On a network, before a machine is connected and talks to other devices and gateways, it is going to do a broadcast and gives up information about itself. This is called an ARP Request. As everything that is talking on Layer 2 is giving out their MAC address for exchange,

A lot of the times, most switches and routers do not change their default credentials

As was said before, every single item that is connected to a network is constantly broadcasting information out to the rest of network, in order to get and maintain an IP address to keep accessing the Internet, even when it is not actively using it (*see DHCP Protocol Gillis, 2023*). In case of an endpoint that has SentinelOne installed on it that has Ranger, that Ranger is going to listen to that NIC on that device, then capture and interpret all the data that is flowing across the network, called MIBs, thus displaying it on the Console.

3.4 Research Sub-Question #3: How can SentinelOne be integrated with the QaaS app?

SentinelOne provides an API to allow users to integrate SentinelOne with other security tools and systems. Additionally, SentinelOne also provide an SDK enables organizations to scan files and detect malicious content among them, called Nexus Embedded AI (*Anfalovas, 2022*), but this thesis will focus solely on SentinelOne APIs.

SentinelOne also provides a comprehensive documentation for its RESTful API that allows users to interact with the SentinelOne platform programmatically. The API provides a wide range of functionalities, including the ability to retrieve information about devices, incidents, and threats, as well as the ability to perform actions such as quarantining devices, remediating threats, and generating reports. The API is designed to work with the Agents, Sentinels, Ranger, and other components of the SentinelOne platform.

The API sends requests to the associated Management Server and responds with the data that the management pulled from the Agents or from the management database. Therefore, all the data that is displayed in the SentinelOne Console dashboard is also available through the API.

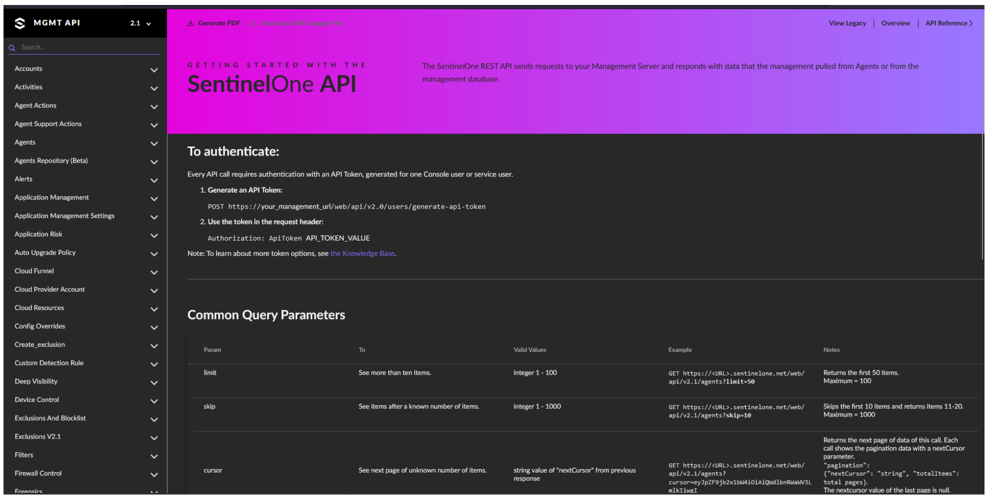
API Token

To be able to access SentinelOne APIs, the user needs to have an API token to prove their identity and that their organization has SentinelOne subscription, therefore has right to access data. API tokens can be created in the SentinelOne MGMT Console.

The API token will be shown only once; therefore the user needs to store it in a secure place. The token generated by the user is time-limited. To renew the token, the user must generate a new one with the same above-steps.

3.4.1 Conclusion

SentinelOne is a cybersecurity platform that provides endpoint protection, detection, and response capabilities to help organizations defend against advanced cyber threats. It leverages AI and machine learning to analyze and respond to security threats in real-time, providing organizations with comprehensive protection against malware, ransomware, and other cyber threats. SentinelOne also provides a RESTful API that allows users to interact with the SentinelOne platform programmatically, enabling integration with other security tools and systems. The API provides a wide range of functionalities, including the ability to retrieve information about devices, incidents, and threats, as well as the ability to perform actions such as quarantining devices, remediating threats, and generating reports. The author will use the official SentinelOne API documentation to integrate SentinelOne with the QaaS app to display the data of Q-ICT and its 400 clients in the QaaS app. The functionality such as filtering, sorting, and pagination will also be taken into account to make the data more user-friendly. The author will utilize different kinds of Firebase Cloud Functions (HTTP onCall, Pub/Sub Triggers, Schedule functions) to call the SentinelOne APIs and display data. Firestore will also be used to store the user's preference from the QaaS app, such as the user's choice of what data they want to see, and visualization types. The author will also utilize Google Secret Manager to store the API token and other sensitive information securely, so it will not be displayed in the code to avoid any security risks. Furthermore, AI powered search engine Algolia will be used to search for specific data in the QaaS app, as well as Firestore



GETTING STARTED WITH THE

SentinelOne API

The SentinelOne REST API sends requests to your Management Server and responds with data that the management pulled from Agents or from the management database.

To authenticate:

Every API call requires authentication with an API Token, generated for one Console user or service user.

1. **Generate an API Token:**

```
POST https://your_management_url/web/api/v2.0/users/generate-api-token
```

2. **Use the token in the request header:**

```
Authorization: ApiToken API_TOKEN_VALUE
```

Note: To learn about more token options, see the Knowledge Base.

Common Query Parameters

Param	To	Valid Values	Example	Notes
limit	See more than ten items.	integer 1 - 100	`GET https://<URL>.sentinelone.net/web/api/v2.1/agents?limit=50`	Returns the first 50 items. Maximum = 100
skip	See items after a known number of items.	integer 1 - 1000	`GET https://<URL>.sentinelone.net/web/api/v2.1/agents?skip=10`	Skips the first 10 items and returns items 11-20. Maximum = 1000
cursor	See next page of unknown number of items.	string value of "nextCursor" from previous response	`GET https://<URL>.sentinelone.net/web/api/v2.1/agents?cursor=eyJpZF9jb2x1bW4iOiAiQWdlbnRWaWV3LmlkIiwgI`	Returns the next page of data of this call. Each call shows the pagination data with a nextCursor parameter. `"pagination": {"nextCursor": "string", "totalItems": total pages}.` The nextcursor value of the last page is null.

Figure 3.11: SentinelOne API Documentation

3.5 Research Sub-Question #4: What are the best data visualization techniques for SentinelOne?

Data visualization is the process of representing complex data (in the form of text and numbers) into a graphical representation, such as: interactive charts, dashboards, pie charts, and so on. In the context of this project, the displays should be able to tell interesting stories and share findings with diverse audience (the client, helpdesk, and cybersecurity analysts).

3.5.1 Data Visualization Widgets in Flutter

In Flutter, there are 3 famous library that supports a wide range of chart types, fl_chart, syncfusion_flutter_charts, and flutter_echarts. Each chart type is suitable for a specific kind of data and provides a different way to visualize the data. Types of charts in both libraries include 30+ plotting series:

- Line chart: is used to display data points connected by straight line segments. They are commonly used to show trends over time.
- Bar chart: it uses rectangular bars to represent data.

The length of each bar corresponds to the value it represents. Bar charts are useful for comparing quantities of different categories.

- Pie chart: represents data in the form of slices of a pie. Each slice corresponds to a category of the data, and the size of the slice is proportional to the quantity it represents.

- Scatter chart: uses dots to represent data points on a two-dimensional plane. They are useful for showing the relationship between two variables.

- Radar chart: A.K.A. spider chart or star chart, is a way of comparing multiple quantitative variables. This makes them useful for seeing which variables have similar values or if there are any outliers among each variable.

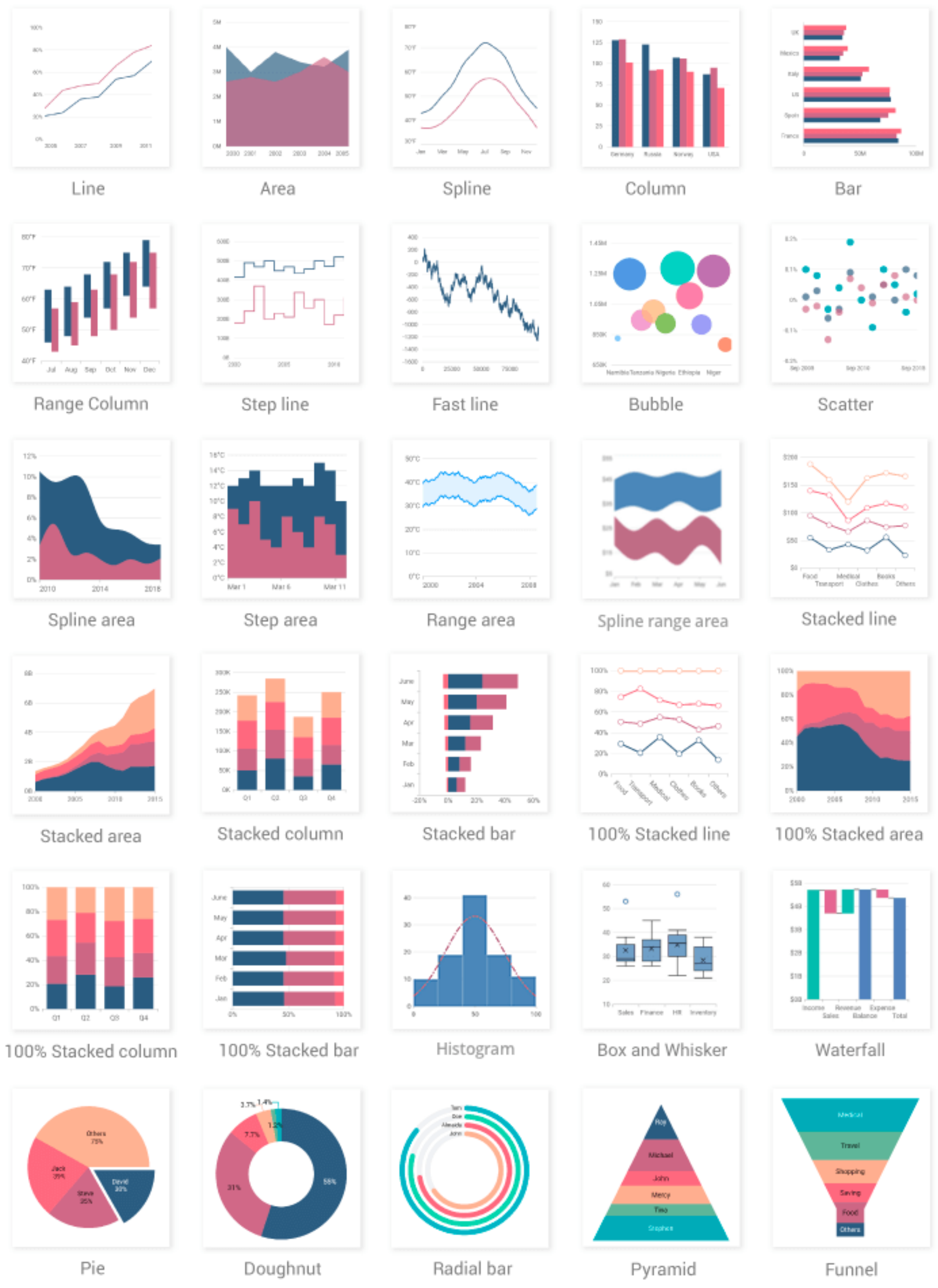


Figure 3.12: Chart types in Flutter fl_chart

Each chart type is easily configured with built-in support for creating stunning visual effects, and any number of series can be added to the chart. Features such as markers, data labels, data label builder, animation, gradient fill, dashes, sorting, and annotations are easy to incorporate and customize.

Several main components of each chart type are:

- Chart data: the most important part is the data wanted to be displayed. This could be a simple list of numbers, or more complex data structure, depending on the type of chart.
- Chart type: the second most important part is to specify the type of chart.
- Axes: the x and y-axes of the chart. The axes provide a reference frame for the data points and can be customized to suit user's needs.
- Grid: the grid lines on the chart. These lines can help users better understand the data by providing a reference frame.
- Touch response: the way the chart responds to user's touch events. It can be customized by providing different types of interactivity, such as highlighting a data point when it is touched.

Axis Types

Axis features such as label intersecting, edge label placement, label rotation, axis opposition, inverse axis, and multiple axis allow users to customize axis elements to make an axis more readable. Four of the axis types that are supported are:

- Numeric
- Category
- Date-time
- Logarithmic

User Interaction

The package greatly enhance UX by adding the following functionalities:

- Zooming and panning
- Crosshairs
- Trackballs
- Drilling down
- Events
- Selection
- Tooltips

Legend

The package also supports legends, which display additional information about a chart series. The legends can be used to collapse the series and can be wrapped or scrolled if items exceed the available bounds.

3.5.2 Comparison to other EDR solutions

To determine the best visualization techniques for SentinelOne, a comparison to other EDR solutions is needed. In this section, the author has looked into other alternatives to SentinelOne, which are a single EPP/EDR solution, created as a complete replacement of legacy AV. Please keep in mind that in this sub-question, only the visualization techniques will be assessed, compared, contrasted, evaluated, and discussed. The other factors such as pricing, technologies, and features will not be discussed in this sub-question.

Heimdal®

On the homepage, information about each of the products/modules that are active under the current customer account is shown. The chart, which can be seen include a variety of Bar charts, Pie charts, and Line charts, include data regarding attacks, vulnerabilities, detections, infected/quarantined files, blocked/allowed processes, third-party application vulnerabilities or OS updates, and quarantined/rejected e-mails.

CrowdStrike

Trend Micro

Trend Micro Apex One has customizable contents and layout of their dashboard. Workload Security uses Session to save user's settings and remember the last view of the dashboard the next time the user log in. The colour here is a bit basic with only the typical red, green, blue, and yellow.

Based on the three examples above, some conclusion can be drawn about the techniques they use in their dashboard:

Visualization techniques

All three solutions use a combination and types of bar charts, pie charts, and line charts to display data. This due to the fact that the type of data they are trying to display is categorical and numerical data. Categorical data is data that can be divided into distinct groups or categories with no inherent order, and one category can only have one value.

Pie charts are used to visualize relative proportions or percentages of different categories within a dataset. Each category is represented by a slice of pie, with the size of each slice corresponding to the proportion of that category relative to the whole dataset.

Bar graphs are used to compare the quantities or frequencies of different categories. Each category is represented by a separate bar, with the height (or length, in horizontal bar graphs) of the bar indicating the value of that category.

In the case of SentinelOne integration to the QaaS app, both bar and pie charts can be used with the addition of line charts to display the timeline of a Threat Incident, and Scatter charts to show the relationship between two variables.

Customizable Widgets

All three solutions allow users to customize the layout and contents of their dashboards. This is important as different users may have different preferences and requirements for the data they want to see. Customizable widgets allow users to choose which data they want to display, how they want to display it, and where they want to display it on the dashboard.

The SentinelOne dashboard of the QaaS app should have this functionality, therefore it should store user's preference **Coloring**

3.5.3 Conclusion

In Flutter, there are two main libraries that support a wide range of chart types, which are: fl_chart and syncfusion_flutter_charts. Each chart type is suitable for a specific kind of data and provides a different way to visualize the data. The SentinelOne data will mainly be displayed by using pie chart and bar graph because the data is categorical and numerical. Comparing to the other EDR platform, the dashboard of the QaaS app should have customizable widgets, allowing users to choose which data they want to display, how they want to display it, and where they want to display it on the dashboard. The dashboard should also store user's preferences and settings, so that the user can see the same view the next time they log in. The colour of visualization charts should not be more than the typical red, green, blue, and yellow, as it can be overwhelming and confusing to the user. Red colour is usually used to indicate a problem or a threat, green colour is used to indicate that everything is okay, while grey colour is used to indicate that the data is neutral or not critical, or that the data is not available. Different colours such as purple, blue, and yellow will also be used to indicate different categories of data. Additionally, the dashboard should also have a legend to display additional information about the chart series.

Chapter 4

Realization

Because the QaaS web application itself is already built on top of existing frameworks, libraries, and technologies, the integration of the SentinelOne was relatively straightforward. The API itself is well-documented, and the NPM package provided by SentinelOne is easy to use. The only challenge was to understand the data structure of the SentinelOne, choosing the right and more important data to display, modelling it in the QaaS database, and then displaying it in the web application.

4.1 The Back-end

As stated before, Q-ICT wishes to utilize the 2nd generation of Firebase Cloud Functions, therefore a new Firebase project was created and stored on the cloud (*using Azure DevOps*). The author also needs to make a decision in regards on how the infrastructure of the codebase should be structured. Because Q-ICT is always critical and open to feedback, the author is given access to the old Firebase codebase (*utilizing version 1.0*), and find any potential upsides and downsides of that project repository. The author then, by an informed decision, is allowed to decide whether to structure the new codebase in the same way as the old one. The author has certainly decided to create some new adjustments to the new codebase. For example, instead of stacking all functionalities that a Firebase Cloud Function might have, the author has decided to improve the codebase especially regarding the separation of concerns. Specifically, the response from the API is modelled using Interfaces and Classes object within the Models directory, allowing for a consistent reuse across different functions. Additionally, distinct Routers and Controllers directories has been established, each with specific responsibilities. The Routers directory primarily handles external communications with the API, utilizing `Axios` (*npm, n.d.*) for handling the GET requests. The Controllers directory, contains the back-end logic of the cloud function, bridging Models and Routers and ensuring comprehensive error documentation. This structure defines the behavior of the Firebase Cloud Function version 2.0 of the QaaS app.

Moreover, there are Utilities and Middlewares directories, which are used to handle common functionalities and errors. The Utilities directory contains functions that are used across the codebase, such as logging errors, and the Middlewares directory contains functions that are used to handle the request and response of the cloud function. For example, the `logError` function in the Utilities directory is used to log errors in the console, and the `handleError` function in the Middlewares directory is used to handle errors in the response of the cloud function. This structure allows for a more organized and maintainable codebase.

The types of functions that are used throughout the project are the following:

- `onCall`: This type of function is used to handle callable functions, which are called from the client-side. It is used to handle requests from the client-side. This is the main function in which the SentinelOne data is fetched and processed.

- `onRequest`: This type of function is used to handle HTTP requests, which are called from the client-side. It is mainly for initial testing, as setting this function up is easier and faster. However, the author does not recommend using this in the production, as it lacks authentication and security for the end-user.

- `onSchedule`: This type of function is used to handle scheduled functions, normally called cron-jobs, which are called at a specific time. It is used to handle scheduled tasks that need to be executed at a specific time. There is only one scheduled function in the project, which is used to replace the old SentinelOne API key with a new one that is securely stored in Google Secret Manager, therefore ensuring proper connection and access to the secret vault, every month.

4.1.1 Setting up permissions with Google Secret Manager

Because every call to the SentinelOne API requires an authorization through a valid SentinelOne API key, it is crucial to store this key securely on the Internet, where not everyone can access it. Following the guideline from the Company Supervisor, in whom is not a big fan of storing sensitive information in .env files or directly in the code, the author stored the SentinelOne API key in a Secret within Google Secret Manager. The number one reason as to why is because Firebase Cloud Functions work well with Secret Manager, as they are both Google Cloud products. Therefore, the author can easily edit the settings of a specific cloud function to have access to the latest version of a specific Secret, providing that they are stored within the same project directory.

Service Account

The first crucial step in setting up the permissions with Google Secret Manager is to create a Service Account. A Service Account is a special type of Google account that belongs to the application or a virtual machine, instead of to an individual end-user. It is used to operate and manage services without sharing user credentials (*Firebase, n.d.-d*). Whenever a function is created in Firebase, it is automatically assigned a Service Account, the admin can edit the permissions of the Service Account to have access to the Secrets stored in the Secret Manager. With the proper access level granted to a specific Service Account assigned to a desired function, the author can then easily configure on which Secrets his cloud functions can have access to.

4.1.2 Compliance with ESLint compiler

Lastly, the back-end project utilizes Node.js as the runtime environment, with TS as the programming language. This has caused longer compilation time as it involves additional steps like type checking and transpiling the code to JS. However, it is still believed to be the better practice as JS in nature is a loosely typed language, and it is easy to make mistakes in the code as its variables do not have a fixed type. While this sometimes can be beneficial as it makes the development faster and more flexible, it also introduces challenges, particularly in debugging and maintaining the code. To prevent this, both the author and the Company Supervisor have decided to use and adhere to TS and ESLint rules, to ensure static typing, code quality and consistency to the coding standards.

Listing 4.1: Example of Class and Interface defined in Models

```

export interface Customer {
  customerId: string;
  customerName: string;
  orgUnitType: string;
  parentId: string;
  city: string;
  stateProv: string;
  country: string;
  county: string | null;
  postalCode: string;
  contactEmail: string;
}

/**
 * Class for CustomerModel.
 */
export class CustomerModel {
  private readonly firestore: FirebaseFirestore.Firestore;
  private readonly collectionName = "customers";
  private readonly context = "CustomerModel";

  constructor() {
    this.firestore = admin.firestore();
  }

  serializeCustomerToJson(customer: Customer): string {
    return JSON.stringify(customer);
  }

  /**
   * Deserializes a JSON string to a Customer object.
   * @param {string} json The JSON string to deserialize.
   * @return {Customer} The deserialized Customer object.
   */
  deserializeJsonToCustomer(json: string): Customer {
    return JSON.parse(json) as Customer;
  }

  /**
   * Creates a new customer in Firestore.
   * @param {Customer} customer Customer to add to Firestore.
   * @return {Promise<FirebaseFirestore.DocumentReference<FirebaseFirestore.DocumentData>> | undefined}.
   */
  async createCustomer(customer: Customer):
      Promise<FirebaseFirestore.DocumentReference<FirebaseFirestore.DocumentData> | undefined> {
    try {
      return this.firestore.collection(this.collectionName).add(customer);
    } catch (ex: unknown) {
      logError(ex, this.context + ".createCustomer");
    }
    return undefined;
  }

  /**
   * Retrieves a customer by its ID from Firestore.
   * @param {string} customerId The ID of the customer to retrieve.
   * @return {Promise<Customer | undefined>} A promise that resolves with the customer object if found,
       or undefined if not found or
   an error occurs.
   */
  async getCustomer(customerId: string): Promise<Customer | undefined> {
    try {
      const doc = await this.firestore.collection(this.collectionName).doc(customerId).get();
      if (!doc.exists) {
        return undefined;
      }
      return doc.data() as Customer;
    } catch (ex: unknown) {
      logError(ex, this.context + ".getCustomer");
      return undefined;
    }
  }

  /**
   * Updates an existing customer in Firestore.
   * @param {string} customerId The ID of the customer to update.
   * @param {Partial<Customer>} customer The partial customer object containing updates.
   * @return {Promise<void>} A promise that resolves when the update is complete.
   */
  async updateCustomer(customerId: string, customer: Partial<Customer>): Promise<void> {
    try {
      await this.firestore.collection(this.collectionName).doc(customerId).update(customer);
    } catch (ex: unknown) {
      logError(ex, this.context + ".updateCustomer");
    }
  }

  /**
   * Deletes a customer from Firestore.
   * @param {string} customerId The ID of the customer to delete.
   * @return {Promise<void>} A promise that resolves when the customer is successfully deleted.
   */
  async deleteCustomer(customerId: string): Promise<void> {
    try {
      await this.firestore.collection(this.collectionName).doc(customerId).delete();
    } catch (ex: unknown) {
      logError(ex, this.context + ".deleteCustomer");
    }
  }
}
```

Listing 4.2: An example of Firebase HTTP Request onCall Cloud Function version 1.0

```
import * as functions from "firebase-functions";
import * as admin from "firebase-admin";
import { Secrets } from "../Firebase/Secrets";
import { FirebaseCall } from "../Firebase/FirebaseCall";
import axios from 'axios';

export default functions
  .region("europe-west1")
  .https.onCall(async (data, context) => {
     try {
       // context.app will be undefined if the request doesn't include a valid
       // App Check token.
       if (context.app === undefined) {
         throw new functions.https.HttpsError(
          "failed-precondition",
          "The function must be called from an App Check verified app."
         );
       }
     } catch (error) {
       console.error("An error occurred in the Firebase HTTP Request onCall Cloud Function: ", error);
       throw new functions.https.HttpsError("internal", "An error occurred while processing the
            request.");
     }
});
```

Listing 4.3: onCall Cloud Function version 2.0, where the parameters of the function is less and the way it handles the authorization is different

```
import axios from "axios";
import * as functions from "firebase-functions/v2";
import { CallableRequest } from "firebase-functions/v2/https";
import admin from "firebase-admin";

import { region, sentinelOneURL, sentinelOneApiVersion } from "../../../config";
import { logError } from "../../../Middleware/LogError";
import { getSecret } from "../../../Util/Secret";

export const getSentinelOneData = functions.https.onCall({ region: region }, async (context:
     CallableRequest<any>) => {
  try {
    // Authentication check
    if (!context.auth) {
      throw new functions.https.HttpsError("unauthenticated", "The function must be called while
           authenticated.");
    }
    const data = context.data;

    // // Checking if the request contains the necessary data
    if (!data || !data.siteId || !data.userId || !data.dataType) {
      throw new functions.https.HttpsError("invalid-argument", "The necessary requirement(s) are
           missing or invalid in your request.");
    }
  } catch(ex: unknown) {

  }
});
```

Listing 4.4: LogError functionality in the Utilities folder

```
export function logError(error: unknown, context: string): void {
  if (isAxiosError(error)) {
    // AxiosError object, log detailed request and response information
    console.error(`[${context}] Axios error occurred: ${error.message}`);
    console.error(`Response data: ${error.response?.data}`);
    console.error(`Status code: ${error.response?.status}`);
    console.error(`Headers: ${JSON.stringify(error.response?.headers, null, 2)}`);
  } else if (error instanceof Error) {
    // Standard Error object, log message and stack
    console.error(`[${context}] An error occurred: ${error.message} \n Stack: ${error.stack}`);
  } else {
    // Non-Error object, log with a generic message
    console.error(`[${context}] An unknown error occurred:`, error);
  }
}
```

Listing 4.5: An example of how that utility is used in the Cloud Function, the reason why exception type is unknown is to comply with the ESLint rules that does not recommend to use the any type to the variables

```
try {

}
catch (ex: unknown) {
  console.log(`An error occured in the getFirestoreData function: ${ex}`);
  logError(ex, "getFirestoreData");
} finally {
  throw new functions.https.HttpsError("internal", "An error occurred while getting the data in
      Firestore.");
}
```

Listing 4.6: Example of challenges in JavaScript because of its loosely typed nature that creates lack of type safety, type coercion, and type conversion

```
function add(a, b) {
  return a + b;
}
console.log(add(5, '2')); // "52" instead of 7
```

Listing 4.7: Another example of JavaScript challenges

```
C:\Users\ChristopherSulistiyo>node
Welcome to Node.js v22.1.0.
Type ".help" for more information.
> true === 1
false
> true + true + true === 3
true
> 0 == "0"
true
> "0" ==- null
true
> -null
-0
> -0 == "0"
true
> parseInt(0.00000005)
5
> typeof(NaN);
'number'

> let a = []
undefined
> a.length == a
true
```

4.1.3 SentinelOne NPM packages

NPM packages are a necessary component when developing a project in JavaScript, as they are used for managing dependencies in Node.js projects. Below are the NPM packages that are used in the project:

•

4.1.4 Firestore database structure

Inside the Firestore, there are various collections (tables) that hold different data. Most of the collections are used to store the data that is fetched from the SentinelOne API, and one collection is used to store the user's session and preference data. SentinelOne data itself is divided into 2 categories: general SentinelOne data and specific data, both of them can have varying fields depending on how SentinelOne structures the data in its JSON format. General SentinelOne data only have 1 specific field in common, which is the siteId. The idea is to store all the clients' data in one collection, therefore to differentiate the data based on client's company, an index of client's site ID (which is unique from each other) is used to separate the data.

The second category is the specific SentinelOne data, win which the data is not only belong to a specific site ID, but also to a specific ID from another SentinelOne data. For example, each threat cam have their own unique timeline of events that have happened. Therefore, another field is required that is used as a second index to reference the desired data. The first and second category will then have different logic in the cloud function to fetch the data.

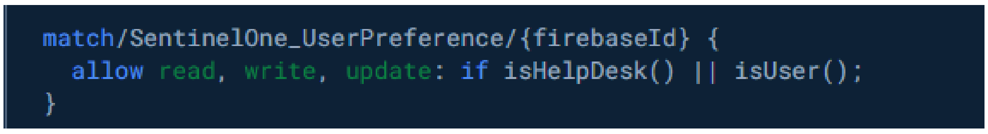
User Preference

A special table in Firestore is created to store the user's preference, each document' ID is the user's ID, therefore ensuring uniqueness. In a document, there are 3 different fields:

- timestamp: containing date time in the format of timestamp. It is used as a reference for the Cloud Function to know when is the data last updated, and if it exceeds a certain threshold, the Cloud Function will fetch the data from the SentinelOne API again. This will be the field that the fetching cloud function needs to do its if-check first to determine if the data is outdated.
- widgetIds: containing an array of strings, each string is the ID of the widget that the user wants to see in the dashboard. This is used for the convenience of adding, updating, and deleting widgets.
- widgetTitle: containing also an array of strings, which serves as the title that the user choose for their widgets. A widget title can have the same name as another widget, as long as the widget ID is different.
- widgets: containing a map of 3 different fields:
  - category: containing a string, which is the category of the widget. The category is used to group the widgets in the dashboard, so the user can easily find the widgets that they want to see. Currently, there are 3 categories: ”Threats”, ”Endpoints”, ”Applications”, and ”Miscellaneous”. In the future, additional 2 categories can be added: ”Ranger (Network Discovery)” and ”Rogues”. Ranger is a feature that is used to discover the network of the client's machine, and Rogues is a feature that is used to detect any unauthorized devices that are connected to the client's machine. These 2 features are currently deactivated in the Q-ICT site environment for internal reasons, thus the author does not include them in the project.
  - graphicType: containing a string, which is the type of the graphic that the user wants to see in the widget. Currently, the graphics that the user can utilize are: ”Doughnout”, ”Pie Chart”, ”Vertical Bar”, ”Stacked Vertical Bar”, ”Horizontal Bar”, ”Stacked Horizontal Bar”, ”Line Chart”, ”Scatter”, ”Bubble”, ”Pyramid”, ”Funnel”, and ”Table”.
  - widgetType: Inside the 4 main categories, the selection is further divided into separate types according to which categories that wanted to be visualized. Agent category can show the status of the agents, such as the number of agents that are connected, disconnected, or in quarantine. Threats category can show the status of the threats that are detected by SentinelOne, such as the number of threats that are detected, resolved, or in quarantine. Applications category show the number of outdated applications that can potentially be a threat to the client's machine, according to the latest CVEs from the NIST and MITRE. The widget categorization is divided as follows:
    - ∗ Endpoint: Agent versions, pending updates, agent by OS, console migration status, domains, encrypted applications, endpoint connected to management, connection status, endpoint health, endpoint list, endpoint summary, installers, load saved filters, local configurations, location IDs, machine types, manual actions required, network health, OS arch, pending uninstall, and scan status.
    - ∗ Threats: Analyst verdicts, confidence levels, engines, external ticket exists, failed actions, incident status, initiator, mitigated preemptively, note exists, pending actions, reboot required, threat classification, threat list, threat status, threat summary of the week, and unresolved.
    - ∗ Applications: Risk levels, and most impactful applications according to SentinelOne Vulnerability Score.
    - ∗ Miscellaneous: SentinelOne news feed, and free text.

4.1.5 Setting up permissions with Firestore

This has got to do with the Firestore database and the access level of permission that a Firebase Cloud Function has. In the SentinelOne integration project, the table that contains SentinelOne data cannot be changed by the client, only by fetching new data from the SentinelOne API and replacing the old data. Therefore, only one table remains that actually can be changed accordingly by the user, the User Preference table. This table can be changed freely by the user as long as they remain logged in to the QaaS app.



```
match/SentinelOne_UserPreference/{firebaseId} {
  allow read, write, update: if isHelpDesk() || isUser();
}
```

Figure 4.1: Granting access of read, write, and update from User Preference collection to users in the Firebase Console

4.1.6 Connection to Algolia

As originally stated, the author and Company Supervisor directly intended on using Algolia search as the SentinelOne API search, although far from being bad, does have type intolerance.

4.2 The Front-end

In the front-end side of this project, it is just a matter of calling the Cloud Functions, modelling the JSON response coming from the Cloud Functions (whether it is from Firestore or SentinelOne), making the visualization graphs using the available Flutter package, setting the logic on how to navigate between pages passing the correct data, the logic for how to display the data correctly in paginated data tables and visualization widgets, and lastly, the searching, filtering, and filtering functionality.

Chapter 5

Conclusion and Recommendation

5.1 Conclusion

Cybersecurity is a complex field and threat actors are ever evolving their trade-craft in clever ways. Every single running business is not too small to be attacked, but they may be too small for them to make the news. Therefore, transparency to the clients is important, because lack of visibility to the clients can lead to a loss of trust. This graduation project serves as the first iteration by Q-ICT about the integration of SentinelOne to its internal application, the QaaS app.

SentinelOne EDR platform is possible to be integrated with the QaaS app, which was made in Flutter with Firebase as the back-end cloud solution, utilizing SentinelOne API to fetch the data. In order that this API to be able to send the correct data, the necessary Agents and Ranger need to be set up properly in a client's machine. This Agent will then serve to replace the traditional AV, as it will gather the data from the client's machine and send it to the management DB, providing real-time monitoring. Additionally, a SentinelOne account with the necessary rights to read the data with its API key is also needed to be configured. In the front-end, the visualization is done using the Flutter package: *fl_chart* and *syncfusion_flutter_charts*. The QaaS app itself has 3 different user roles in its system: the clients, the helpdesk, and the IT admins. The objective of this project is that for the QaaS app to be used by the clients to see the status of their devices, and the helpdesk and IT admins to manage and monitor the client devices that are connected to the SentinelOne EDR platform. For the client user, they can see the status of their devices that are connected to the SentinelOne EDR The user itself can customize which charts and data they want to see in the dashboard, to make it more user-friendly and catching up with other EDR platforms. The user can also see the details of the threats that are detected by SentinelOne, and the user can also see the details of the devices that are connected to the SentinelOne EDR platform.

In the back-end, several functions with different functionalities need to be created to handle the data that is fetched from different SentinelOne API endpoints. The user's session and preference data are stored in the Firebase Firestore DB, and as well as AI powered Algolia search engine to make the search faster and more accurate. Additional features such as filtering, sorting, and pagination will be handled by inputting the correct headers to the API request.

5.2 Recommendation

5.2.1 Structure of the front-end codebase

There are some issues that the author has noticed with the way the QaaS app code was structured. For example, when assigning modules to a user,

5.2.2 Integration with N-Central data

5.2.3 Redis middleware NoSQL database caching

Redis is a powerful tool for DB that uses in-memory data structure to provide caching and message broker. It supports various data structures (*Redis,* n.d.), such as strings, hashes, lists, sorted sets, with range queries, bitmaps, hyperlogs, geospatial indexes with radius queries, and streams.

Because the main idea in Redis is to provide caching of the data, coupled with the fact that it is a NoSQL DB, Redis is exceptionally fast to query due to its in-memory nature in contrast to traditional disk-based DBs. This makes it suitable for use cases where low latency and high throughput are required, making it perfect for getting and storing data that is frequently accessed.

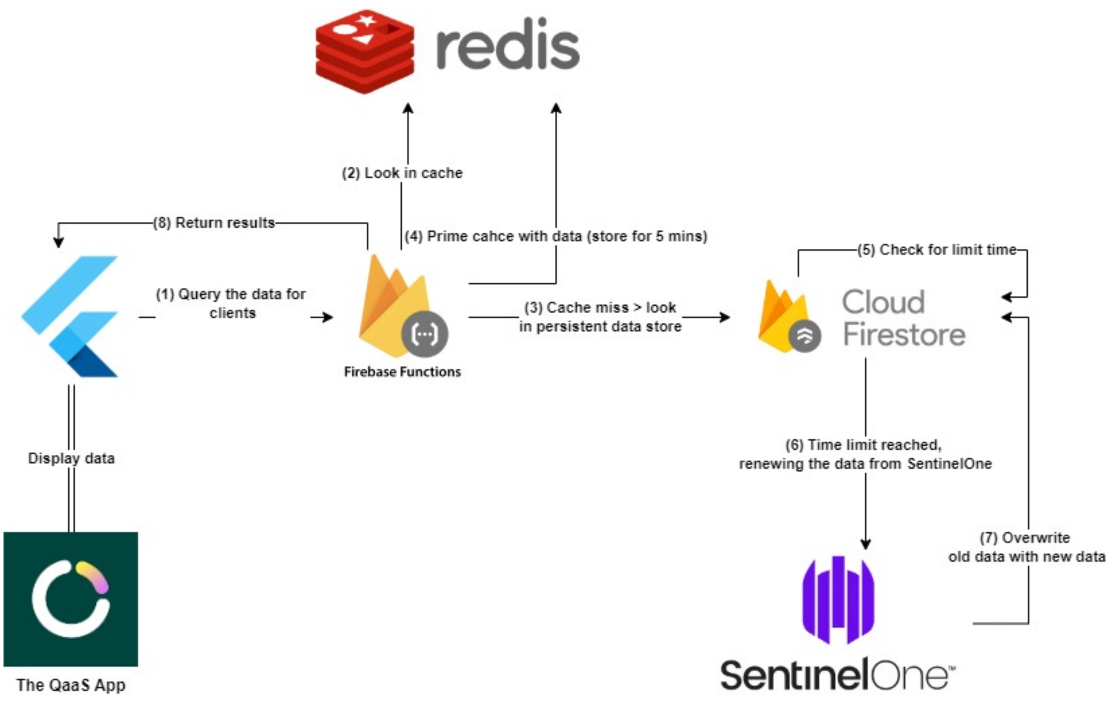
The idea is to put SentinelOne data into the Redis DB too, with the limit of 5-10 minutes before the cache is deleted, and then fetch the data from the Redis DB instead of directly fetching it from Firestore and SentinelOne API constantly. Implementing Redis will make querying the data faster than Firestore, providing better UX for the users.

As of April 2024, Redis (*Trollope, 2024*)

5.2.4 Compacting everything into one function

In the initial phase of the development, the author thought that it is a good idea to make just one function for fetching all data related to SentinelOne, for the purpose of simplicity for other developers, as there are already more than 50 functions that the QaaS app has. However, near the end of the development, the author realize that this may not be the best approach because of the number of

5.2.5



redis
(2) Look in cache
(8) Return results
(4) Prime cahce with data (store for 5 mins)
(5) Check for limit time
(1) Query the data for clients
(3) Cache miss > look in persistent data store
Cloud Firestore
Firebase Functions
Display data
(6) Time limit reached, renewing the data from SentinelOne
(7) Overwrite old data with new data
SentinelOne
The QaaS App

Figure 5.1: How the infrastructure would look if Redis is implemented

Bibliography

Acronis. (n.d.). Computicate. *solutions.acronis.com*. https://solutions.acronis.com/en-us/integrations/computicate/

Anfalovas, I. (2022). What is address resolution protocol? a beginner's guide to ar. *Ipxo*. https://www.ipxo.com/blog/address-resolution-protocol/

Cronitor. (n.d.). Crontab guru. *contab.guru*. https://crontab.guru/#0_0_*/7_*_*

Fielding, R. T., & Taylor, R. N. (2000). Architectural styles and the design of network-based software architectures. *University of California, Irvine*. https://ics.uci.edu/~fielding/pubs/dissertation/top.htm

Firebase. (n.d.-a). Firebase app check. *Firebase.google.com*. https://firebase.google.com/docs/app-check

Firebase. (n.d.-b). Firebase extensions. *Firebase.google.com*. https://firebase.google.com/docs/extensions

Firebase. (n.d.-c). Firebase products - google. *Firebase.google.com*. https://firebase.google.com/products-build

Firebase. (n.d.-d). Firebase service accounts overview. *Firebase.google.com*. https://firebase.google.com/support/guides/service-accounts

Firebase. (n.d.-e). Schedule functions. *Firebase.google.com*. https://firebase.google.com/docs/functions/schedule-functions?gen=2nd

Gartner. (2017). Security information and event management. *Gartner Research*. https://www.gartner.com/en/information-technology/glossary/security-information-event-management

Gillis, A. S. (2023). Dhcp (dynamic host configuration protocol). *TechTarget*. https://www.techtarget.com/searchnetworking/definition/DHCP

Google. (n.d.). Google cloud. *Secret Manager*. https://cloud.google.com/security/products/secret-manager

IBM. (n.d.). What is a rest api? *CyberHoot*. https://www.ibm.com/topics/rest-apis#:~:text=A%20REST%20API%20(also%20called,transfer%20(REST)%20architectural%20style).

Indeed. (2023). What is soap api? (plus comparison to rest api and benefits). *Indeed Editorial Team*. https://www.indeed.com/career-advice/career-development/what-is-soap-api#:~:text=SOAP%20API%2C%20or%20simple%20object,Extensible%20Markup%20Language%20(XML).

inDiv Solutions. (n.d.). Indiv solutions focus op online webshop laten maken. *indiv.nl*. https://webba.nlhttps://indiv.nl/

Kubernetes. (2018). Cronjob. *kunbernetes.io*. https://kubernetes.io/docs/concepts/workloads/controllers/cron-jobs/#:~:text=A%20CronJob%20creates%20Jobs%20on,schedule%2C%20written%20in%20Cron%20format.

LinkedIn. (n.d.). Resello: Powered by pax8. *LinkedIn.com*. https://www.linkedin.com/company/resello-bv/

MemoICT. (n.d.). Home | memo ict | de shopware specialist. *memo-ict.nl*. https://memo-ict.nl/

MKBiT. (n.d.). Service centraal. *MKBiT.nl*. https://mkbit.nl/producten/

MongoDB. (n.d.). What is nosql? *MongoDB.com*. https://www.mongodb.com/nosql-explained

npm. (n.d.). Axios - npm. *npmjs.com*. https://www.npmjs.com/package/axios

OndernemmendEmmen. (n.d.). Ondernemend emmen: Home. *ondernemendemmen.nl*. https://www.ondernemendemmen.nl/

PeatDigital. (n.d.). Home | peat digital| specialisten in online marketing. *peatdigital.nl*. https://peatdigital.nl/

PerfectView. (n.d.). Pefectview crm online | krachtige nederlandse crm. *PerfectViewCRM.nl*. https://www.perfectviewcrm.nl/

Redis. (n.d.). Data structures. *redis.io*. https://redis.io/redis-enterprise/data-structures/

SentinelOne. (n.d.). What is network detection and response (ndr)? *SentinelOne®*. https://www.sentinelone.com/cybersecurity-101/what-is-network-detection-and-response-ndr/

Taylor, C. (2021). Castle-and-moat network security model. *CyberHoot*. https://cyberhoot.com/cybrary/castle-and-moat-network-model/

Taylor, C. (2024). Castle-and-moat network security model. *CyberHoot*. https://cyberhoot.com/cybrary/castle-and-moat-network-model/

TOMTelecom. (n.d.). Wie zijn wij. *tomtelecom.nl*. https://www.tomtelecom.nl/wiezijnwij/

Trollope, R. (2024). Redis adopts dual source-available licensing. *redis.io*. https://redis.io/blog/redis-adopts-dual-source-available-licensing/

Vazquez, M. (2021). What you should consider about extended detection and response (xdr). *IDC*. https://blogs.idc.com/2021/03/18/what-you-should-consider-about-extended-detection-and-response-xdr/

Vogel, J. (2023). Ict research methods — methods pack for research in ict. *HBO-i, Amsterdam*. https://ictresearchmethods.nl/

Webba. (n.d.). Webba | experts in design, techniek en online marketing. *wenbba.nl*. https://webba.nl/

Wikipedia. (n.d.-a). Algolia. *Wikipedia, free encyclopedia*. https://en.wikipedia.org/wiki/Algolia

Wikipedia. (n.d.-b). Api. *Wikipedia, free encyclopedia*. https://en.wikipedia.org/wiki/API

Wikipedia. (n.d.-c). Arp spoofing. *Wikipedia, free encyclopedia*. https://en.wikipedia.org/wiki/ARP_spoofing

Wikipedia. (n.d.-d). Captcha. *Wikipedia, free encyclopedia*. https://en.wikipedia.org/wiki/CAPTCHA

Wikipedia. (n.d.-e). Captcha. *Wikipedia, free encyclopedia*. https://en.wikipedia.org/wiki/ReCAPTCHA

Wikipedia. (n.d.-f). Customer relationship management. *Wikipedia, free encyclopedia*. https://en.wikipedia.org/wiki/Customer_relationship_management

Wikipedia. (n.d.-g). Cyber threat hunting. *Wikipedia, free encyclopedia*. https://en.wikipedia.org/wiki/Cyber_threat_hunting

Wikipedia. (n.d.-h). Enterprise resource. *Wikipedia, free encyclopedia*. https://en.wikipedia.org/wiki/Enterprise_resource_planning

Wikipedia. (n.d.-i). Firebase. *Wikipedia, free encyclopedia*. https://en.wikipedia.org/wiki/Firebase

Wikipedia. (n.d.-j). Json. *Wikipedia, free encyclopedia*. https://en.wikipedia.org/wiki/JSON

Wikipedia. (n.d.-k). Man-in-the-middle attack. *Wikipedia, free encyclopedia*. https://en.wikipedia.org/wiki/Man-in-the-middle_attack

Wikipedia. (n.d.-l). Penetration test. *Wikipedia, free encyclopedia*. https://en.wikipedia.org/wiki/Penetration_test

Wikipedia. (n.d.-m). Remote monitoring and management. *Wikipedia, free encyclopedia*. https://en.wikipedia.org/wiki/Remote_monitoring_and_management

Wikipedia. (n.d.-n). Xml. *Wikipedia, free encyclopedia*. https://en.wikipedia.org/wiki/XML