HLD

Feature Name

Document ID: AAA-BBB-12345

Revision History:

Rev.	Date	Author	Details
1	2024-11-03	Andreas Hoppe	Setup initial version

Abstract:

{3-4 lines about what the feature is, which system/version it is for, which sub-systems it involves.}

[Enter your writing here]

*{ A high-level design document is a high-level **description of the architecture, design, and components of a software system**.*

*The purpose of this document is to provide a **clear and concise understanding** of the system's design before more detailed design and implementation work begins.*

This document will describe the system architecture and organization, the main components and their relationships, the overall data flow and processing, and any external systems and dependencies.

The high-level design document serves as a foundation for more detailed design and implementation work and is used as a reference throughout the software development process.

It is also used to communicate the design to stakeholders, project team members, and other relevant parties, and to ensure that everyone has a clear understanding of the system's design and architecture}

Contents

1. General

1.1 Introduction

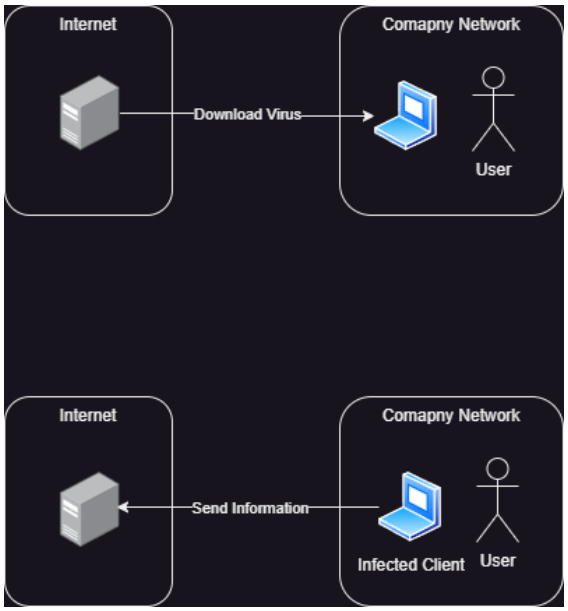
Companies are threatened by malware. Our business analysis reveals a growth of 75% in the previous three years and our analysts assume a further growth of at least additional 5% each year.

Because of this analysis, we want to capitalize on our knowledge about digital threats. Our goal is to sell digital threat detection to our core customers.

The goal is to develop a malware detection mechanism for small to large size companies by the end of 2025 and sell it at least 100 times during 2026 to existing customers.

With the customer experience from 2026, we want to start a marketing campaign to become a leading provider of malware / fraud detection software (among top five) by 2031.

In the initial project described in this HLD we aim for pure detection software. We focus on the following cases:



1. Download infected files: Our solution will detect the download of infected files. In the first version we want our system to be able to inform our customers that such a download happened, which threat was detected (e.g. name of the virus) and to which machine the infected file was downloaded to.
2. Send malicious information: When a client already is infected by a virus, our solutions shall detect malicious communication with external servers.

1.2 Glossary, references

Term	Description
Automatic Cyber Investigator (ACI)	An instance that is invoked when threats are detected. Presents the threat and relevant related information. It is possible to configure for which threats an ACI is invoked.
Cyber Analyst (CA)	Specialist on customer side that works with the product and is automatically informed about threats

Term	Description
SoC	Security Operations Center – component of the product for reviewing cyber-attacks by the Cyber Analyst

2. Requirements

{This chapter should NOT include any design constraints or ideas – only the requirements, as interpreted by dev teams. All design issues must be under "High Level Design".

HLD Requirements are a transformation of Product requirements to technical domain.

Specific requirements should be crossed reference to other specifications (Product, Project etc.)

The breakdown below can be modified according to the specific subject of the HLD

1. *Software requirements are the specific **needs, goals, and constraints** that a software product must meet to be successful.*
2. *These requirements typically describe **what the software should do**, how it should perform, and the conditions under which it should operate.*
3. *They also define the features and functionality that the software must provide, as well as any limitations or restrictions that must be considered.*
4. *Software requirements can come from various sources, such as customers, end users, stakeholders, and project team members.*
5. *The process of defining, documenting, and refining software requirements is a critical step in the software development lifecycle and is essential for creating a high-quality and successful software product.*
6. ***Functional requirements***
    a. *describe what a software system must do and define the specific features and capabilities that the system must provide.*
    b. *These requirements specify what the software should accomplish and how it should behave in response to specific inputs and actions.*
    c. *Examples of functional requirements include user authentication, data input and output, calculations, and reporting.*
7. ***Non-functional requirements***
    a. *describe **how well the software system must perform** and define the constraints and quality attributes that must be met.*
    b. *These requirements specify the **characteristics and qualities** that are expected of the system, such as reliability, scalability, security, and usability.*
    c. *Examples of non-functional requirements include response time, availability, and security standards.*
8. *Functional requirements are often described in terms of inputs, outputs, and specific user interactions with the software, while non-functional requirements are described in terms of performance and quality characteristics.*
9. *Both types of requirements are important for creating a software system that meets the needs and expectations of its users, and it's important to consider both functional and non-functional requirements when designing, building, and testing software.}*

Non-functional requirements

- The solution shall be 99.9% available
- The communication delay for clients shall be below 10ms for each incoming / outgoing http request

1.1 Logical (System Functionality)

{Several items (number each paragraph!) which describe the feature highlights from a user standpoint - should be very similar the feature description in the product requirements specification

If relevant and does not exists in Product Spec – include use cases, state machines }

All requirements should be uniquely identifiable (different section number)

Careful attention should be given to organizing the requirements to maximize readability (short sentences, sub-paragraphs, etc.)

System functionality will break into epics and user story. The dev teams eventually break the user story into tasks}

File Analysis

FA.1: When an infected file was downloaded the CA shall be able to see to which computer the virus was downloaded

FA.2: When an infected file was downloaded the CA shall be able to identify the virus

FA.3: CA shall be able to see how many times a specific infected file was downloaded

HTTP Requests

HR.1: When malicious HTTP request was filed CA shall be able to see from which computer the request came

HR.2: When malicious HTTP request was filed CA shall be able to identify the malware

HR.2: CA shall be able to see on how many machines a specific malware is installed

Threat detection

TD.1: It shall be possible to review the threat detection sensors

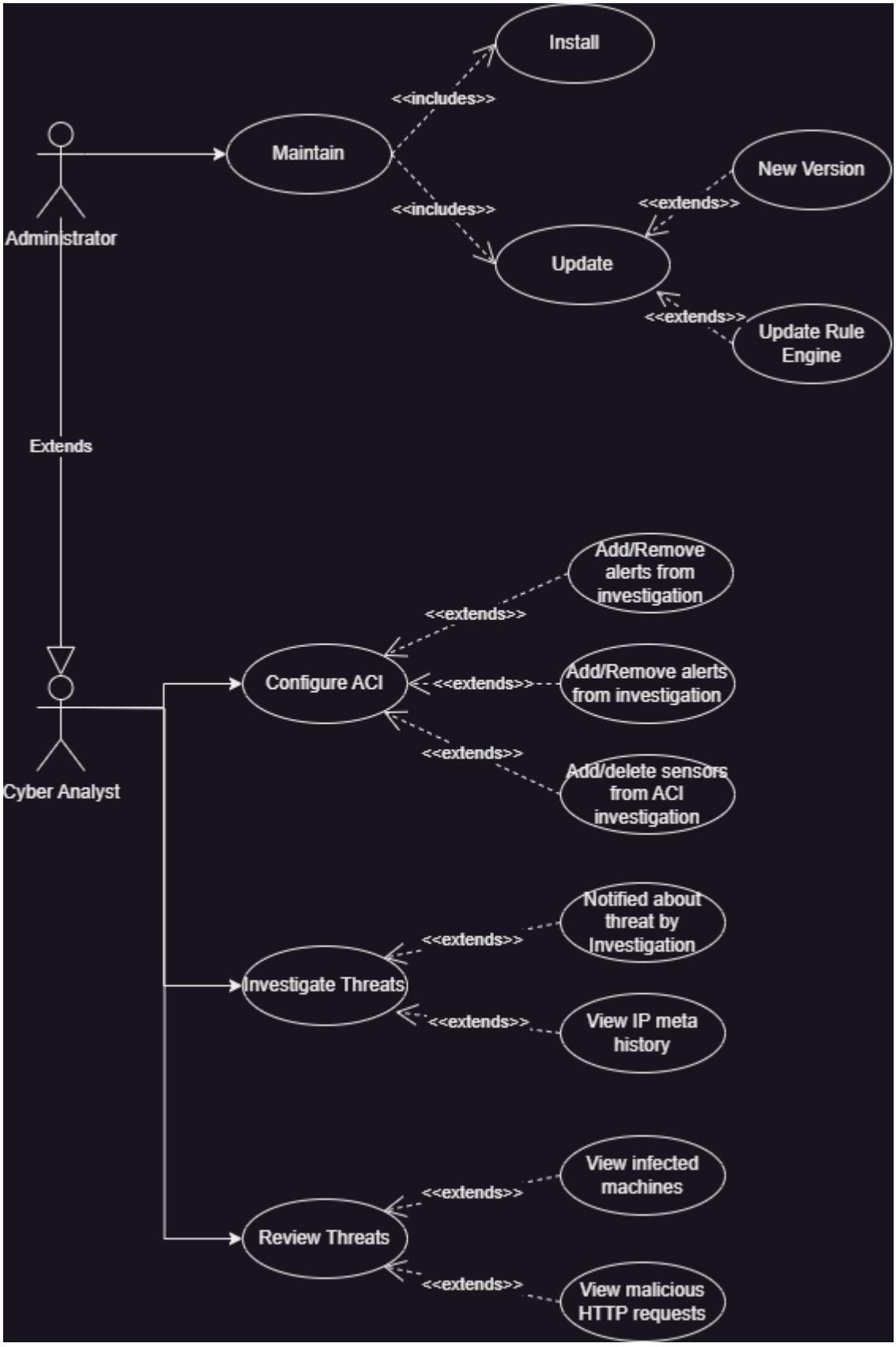
TD.2: It shall be possible to create an investigation in the ACI

TD.3: It shall be possible to assign a threat detection sensor to an investigation

TD.4: It shall be possible to define which information from IP storage is fetched when an investigation is activated

TD.5: When a threat is detected by an investigation the related IP information shall be send to the Control Center

1.1.1 Use Cases



1.2 User Workflow

{If relevant and possible include sample of screen shots}

For a better understanding of the system the following diagram illustrates how the ACI's are invoked based on the analysis of the network traffic:

1.3 Availability and Recovery

{Defines the proportion of time that the system is functional and working. It can be measured as a percentage. It should also state the recovery measures for any relevant failure}

[Enter your writing here]

1.4 Performance & Capacity Requirements

{Performance & capacity requirements should be expressed in measurements, not terms like "very fast". It should include responsiveness of a system to execute any action within a given time interval}

[Enter your writing here]

1.5 Scalability

{System ability to grow, by adding threads or HW. Address issues such as persistency and load balancing}

[Enter your writing here]

1.6 Security

{Address security in all relevant levels – OS, HW, Application level, interfaces (manly external), db}

[Enter your writing here]

1.7 Monitoring and Debugging

{define the monitoring capabilities of the feature. New Metrics, monitoring dashboards, log collection, and the ability of the system to provide any other information helpful for identifying and resolving issues when it fails to work correctly}

[Enter your writing here]

1.8 Deployment

{Describe the deployment environment - how this feature will be deployed, in the cloud native domain, how to deploy the container, what is the replica set, can this feature be deployed on any platforms}

[Enter your writing here]

1.9 Backward Compatibility

{Describe the required backward compatibility versions that should be supported}

[Enter your writing here]

3. High Level Design

1.1 System Architecture

{This section should be filled for features which influence the system architecture: e.g. a new server / platform support, multi-site oriented features, etc.) Use the following views to describe your architecture:

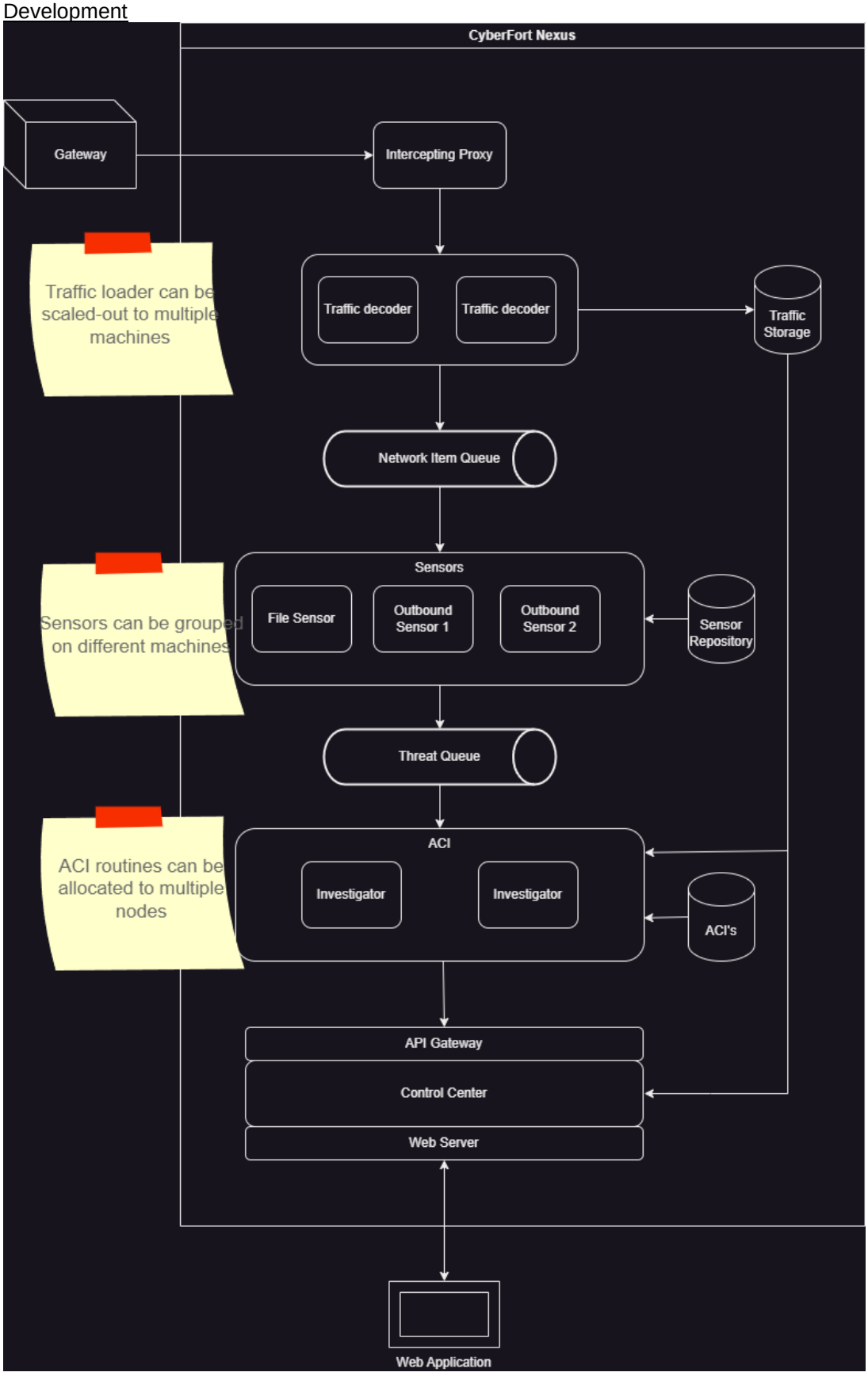
1. *Process – run time behavior – how components communicate, concurrency, distribution, performance and scalability – use sequence diagrams, state machine diagrams, communication and activity diagrams*
2. *Development – implementation view – component diagram*
3. *Physical – deployment view – topology of software components, the physical layer, networking}*

[Enter your writing here]

Process

TBD

Development



1.2 Processes

{Define what the feature should do by identifying inputs, processes and outputs. Specify methods to be used. Interactions to other components & features should be specified or referred to.

This section will also identify the interfaces which will be elaborated in the detail design done by dev teams.

When writing this part, the following principles apply:

Break down the feature to user (or system) processes. Describe, for each process, the information flow between the user, and the subsystems involved. For each process, cover every aspect of the feature.

[Enter your writing here]

1.3 Design Rules and Principles

{Rules for constructing the architecture. Anything that the developers who will work on this feature need to keep in mind when they do low-level design that is not covered in the requirements or interfaces, in example – presentation layer should not direct access to persistency layer, all communication between components should be done over HTTPS. All data at reset should be encrypted.

This section can include guidance for the dev team – example – wherever possible, use asynch flows to improve parallelism}

[Enter your writing here]

1.4 Upgradability

{Describe how to upgrade older version of the system to support the deployment of this feature. What components needs to be upgrade, what kind of dependencies exits. Do we need to under go DB migration, or schema changes}

[Enter your writing here]

1.5 Assumptions and Dependencies

[Enter your writing here]

4. Time Estimation

{Insert the work plan time estimation for development after HLD writing }

Subsystem/team	Workdays

{Insert the work plan time estimation for development after this FRS review }

5. Limitations and Reservations

{This is a very important section. This is the place to put anything that the feature WON'T do that someone might easily think it SHOULD do. If this section is complete, there should be no surprises for the project manager or the client when they get the final product}

[Enter your writing here]

6. Risks

{For each risk, specify the resolution method (i.e., how to minimize the risk, and are there ways to minimize the "damage" of the risk if it materializes)}

[Enter your writing here]

7. Open Issues

{This section should list every item that is not 100% closed in the document. Looking at this section should be enough to know what still needs to be decided or investigated. In the final FRS, this section should not exist. Any time you write in the sections above "DBS will check that..." or "AMS will perform a benchmark that" or "the text of the message will be decided", or "???", or TBD - you MUST add a paragraph to this section!

The final revision of the document should NOT include this section!}

No.	Description	Subsystem	Responsible