Integrating self-collected health data by patients with diabetes in Infodoc

HØGSKOLEN
I BERGEN

UNIVERSITAS BERGENSIS

Anders Steen Nilsen

Supervisor: Pål Ellingsen
Co-Supervisor: Alain Giordanengo

Department of Computing, Mathematics and Physics
Western Norway University of Applied Sciences

Master in Software Engineering

November 2018

Dedicated to my sister.

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 65,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 150 figures.

Anders Steen Nilsen
November 2018

Acknowledgements

I would, first of all, thank Pål Ellingsen for his supervision and guidance throughout this master thesis. No problem has been too small or too big to discuss, and the door has always been open.

Alain Giordanengo has also been a tremendous resource. I am grateful for our weekly meetings and continuously counseling.

I would acknowledge the effort Astrid Grøttland, and Eirik Åsand at Norwegian Center for E-health Research (NSE) has done to facilitate cooperation between Infodoc and me NSE, and Alain.

This thesis would not be possible without the numerous people at Infodoc. Thanks to Endre Dyrøy and Sølvi Førde Lindås for giving me this thesis, Kristian Sandven for being my primary contact person for technical issues and also Irene Fagerheim for good break conversations.

Thanks to Joakim Kårstad Tran for joining me in meetings and for helping me do security testing in Bergen and Tromsø. I would also wish him luck on his thesis.

I would also acknowledge Isabel Francis help in proofreading this thesis.

Abstract

NSE's project Full Flow of Health Data Between Patients and Health Care Systems (FullFlow) tries to give diabetes patients data to general practitioners and specialists.

A prototype extension that can show this content in Infodoc's Electric Health Record (EHR) Plenario has been made.

In order to make such a prototype, all technologies that have been used, or has been considered used, was thoroughly examined. All design questions have been justified.

How to do the security testing was evaluated, and the choice was to do Threat Modeling followed by penetration testing.

Table of contents

List of figures

List of tables

Acronyms

AAU	Aalborg University. 3
ACL	access control lists. 26
ADA	Application Decomposition and Analysis. 41, 44, 46
alg	Algorithm. 54
AMQP	Advanced Message Queuing Protocol. xii, 14, 28, 29, 39
API	Application Programming Interface. 10, 14, 50, 51, 55
ASCII	American Standard Code for Information Interchange. 28
ASF	Application Security Frame. 46
CEF	Chromium Embedded Framework. xv, 33, 34, 55
CI	Continuous Integration. 32
CSS	Cascading Style Sheets. 28
CTU	Czech Technical University in Prague. 3
CVSS	Common Vulnerability Scoring System. 25
DFD	Data Flow Diagram. 11, 44, 46, 49
DoS	Denial Of Service. 51
DPA	Data Protection Act. 9
DPAPI	Data Protection Application Programming Interface. 54

DREAD Damage potential, Reproducibility, Exploitability, Affected users, Discoverability. 39, 42, 44

EHR Electric Health Record. ix, 2, 3, 10, 17, 18, 43

FAT File Allocation Table. 26

FHIR Fast Healthcare Interoperability Resources. 31

GDPR General Data Protection regulation. 10

GP General Practitioner. 2, 31, 50

GPEN Global Privacy Enforcement Network. 1

HMAC Hash-based Message Authentication Code. 54, 55

HTML Hypertext Markup Language. 21, 23, 27, 28, 31, 32, 33

HTTP Hypertext Transfer Protocol. 11, 16

ID Identity. 13

IE Internet Explorer. 27

IEC the International Electrotechnical Commission. 13

ISO the International Organization for Standardization. 13, 28

JOSE JSON Object Signing and Encryption. 54

JS JavaScript. 26, 27, 31, 32

JSON JavaScript Object Notation. 14, 16

JWA JSON Web Algorithm. 54

JWE JSON Web Encryption. 19

JWS JSON Web Signatures. 19, 54

JWT JSON Web Tokens. 19, 28, 53, 54

LAN Local Area Network. 11

MIME	Multipurpose Internet Mail Extensions. 28
MVP	Minimal Viable Product. 32
NDE	The Norwegian Directorate of eHealth. 2, 3, 10, 28, 42, 47
NFR	Norwegian Research Council. 3
NHN	Norsk Helsenett. 47
NSE	Norwegian Center for E-health Research. vii, ix, 2, 3, 11
NTNU	Norwegian University of Science and Technology. 3
NVD	National Vulnerability Database. 25
OASIS	Organization for the Advancement of Structured Information Standards. 28
OCTAVE	Operationally Critical Threat, Asset, and Vulnerability Evaluation. xii, 41, 45, 58
OIDC	OpenID Connect. 19, 20
OP	OpenID Provider. 20
OWASP	The Open Web Application Security Project. xiii, 10, 11, 46
PASTA	Process for Attack Simulation and Threat Analysis. xiii, 41, 46
PDF	Portable Document Format. 20, 21, 22, 31, 44
PHA	Personal Health Archive. 2, 5
PHR	Personal Health Record. 3
REST	Representational State Transfer. 10, 14, 15, 16, 50, 51
RP	Relying Party. 20
RSA	Rivest–Shamir–Adleman. 54, 55
SDL	Security Development Lifecycle. 41, 44, 58
SEI	The Software Engineering Institute. 45
SIR	Security Intelligence Report. 23

SOAP Simple Object Access Protocol. 14

SSL Secure Sockets Layer. 36

STRIDE Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, Elevation of Privilege. 39, 41, 44, 46, 49

TC Task Committee. 28

TLS Transport Layer Security. 36, 51

UIT University of Tromsø. 3

UNN University Hospital of North Norway. 3

URI Uniform Resource Identifier. 16, 28

URL Uniform Resource Locator. 16, 33

URN Uniform Resource Name. 16

VAST visual, agile, and simple threat modeling. 41

XML eXtensible Markup Language. 14, 16, 29

Chapter 1

Introduction

1.1 Motivation

With a smartphone, a patient can measure his blood pressure and pulse, and he can log his activity levels, calorie intake and several other values. Smartphone applications can detect concussion and skin cancer. [11, 12]

Using a smartphone to collect health data is fast, cheap and convenient, but how good is the data integrity? Who owns the data? Is the information gathered and used afterward? Is it securely stored? Can the user delete the data? This is a handful of questions The Norwegian Data Protection Authority (Datatilsynet) asked when they tested six different health applications' Privacy Policy. As seen in figure 1.1, the result is that five out of six did not provide sufficient information about how the data is handled. This report was part of the 2016 Global Privacy Enforcement Network (GPEN) Privacy Sweep that showed that 60% did not provide sufficient information in their privacy policy. [9]

Table 1.1 Result from The Norwegian Data Protection Authority [9]

Product	1	2	3	4	5	6
Privacy Policy	✓	✓	✗	✗	✗	✗
Information on how is the data used	✗	✓	✗	✗	✗	✗
Information on storing and security	✗	✗	✗	✗	✗	✗
Contact information	✓	✓	✗	✗	✗	✗
Infrormation regarding deletion	✗	✗	✗	✗	✗	✗

Note 1.1. Extraction from the Personal Data Act [13]

Section 28 *Prohibition against storing unnecessary personal data*
The controller shall not store personal data longer than is necessary to carry out the purpose of the processing. If the personal data are not after that stored in pursuance of the Archives Act or other legislation, they shall be *erased*.
The data subject may demand that data which are strongly disadvantageous to him or her shall be blocked or erased . . .

Note 1.1, is only one of many strict Norwegian laws regarding personal data. The applications on the market today do not comply with these laws. The Norwegian Directorate of eHealth (NDE) has created *kjernejournalen* at helsenorge.no, a Personal Health Archive (PHA) that complies with Norwegian laws and regulations (more information in chapter 1.4). With a ‘Kjernejournal’ in helsenorge.no, the patients owns their data and can choose who has access. [14]

There is a need for a better way of handling user-generated health data. By sending all data to helsenorge.no, the data would be securely managed and follow all the Norwegian laws and regulations.

1.2 Infodoc Plenario

Infodoc has been delivering Electric Health Record (EHR) systems since 1979. In 2007 Infodoc delivered the system Infodoc Plenario, an EHR that can be tailored for General Practitioner (GP)s, specialist Health stations, and communal health services. Besides the Medical journal function, it also has an appointment planner, economy and messaging capabilities. [15, 16]

Infodoc’s Plenario architecture is relying on a server on-site in each health care facility. But with a new feature called Infodoc SKY, more and more functionalities are ported towards a cloud environment eliminating the need of an on-site server.

Infodoc is collaborating with Norwegian Center for E-health Research (NSE) to integrate FullFlow into Infodoc Plenario.

1.3 The Norwegian Directorate of eHealth

“NDE is a subordinate institution of our Ministry of Health and Care Services. Established on January 1st, 2016, The Norwegian Directorate of eHealth (NDE) is a subordinate institution of our Ministry of Health and Care Services. The former eHealth division of The Norwegian Directorate of Health provided the nucleus from which the new Directorate of eHealth has evolved.

The Norwegian Directorate of eHealth will implement the national policy on eHealth, establish the requisite standards, and administrate the use of eHealth methodology nation-wide."[17]

NDE is responsible for the operations and development of Helsenorge.no [18], this in turns makes them responsible for how to receive and distribute data they get from the Full Flow server.

1.4 Helsenorge.no

Helsenorge.no is a portal for health services for residents in Norway that delivers health information and self-service operations. A service Helsenorge.no is providing is 'Kjerne-jounal,' a Personal Health Record (PHR) for every Norwegian resident. With 'Kjernejournal,' medical workers will have access to key information about the patient's health. Following strict Norwegian laws, patients have the right to look at data that is stored about them and can choose access rights to different medical personnel.

Helsenorge.no's 'Kjernejournal' in the FullFlow project will serve as the portal where patients will have full control over their data. FullFlow will send the data to Helsenorge.no, and Helsenorge.no will, after the patients' approval, send it to the EHR systems.[18]

1.5 FullFlow

FullFlow, or it's full title "Full Flow of Health Data Between Patients and Health Care Systems" is a project that has the goal of sending data between patients, primary health care EHR systems and secondary health care EHR systems [19]. The project is orchestrated by NSE and financed by Norwegian Research Council (NFR) through the IKTPLUSS program.

Several actors are active in the FullFlow project. Infodoc, Dips ASA and Hove Medical Systems AS are EHR vendors. Cooperating universities are University of Tromsø (UIT), Czech Technical University in Prague (CTU), Norwegian University of Science and Technology (NTNU) and Aalborg University (AAU). University Hospital of North Norway (UNN) are also partitioning in the project.

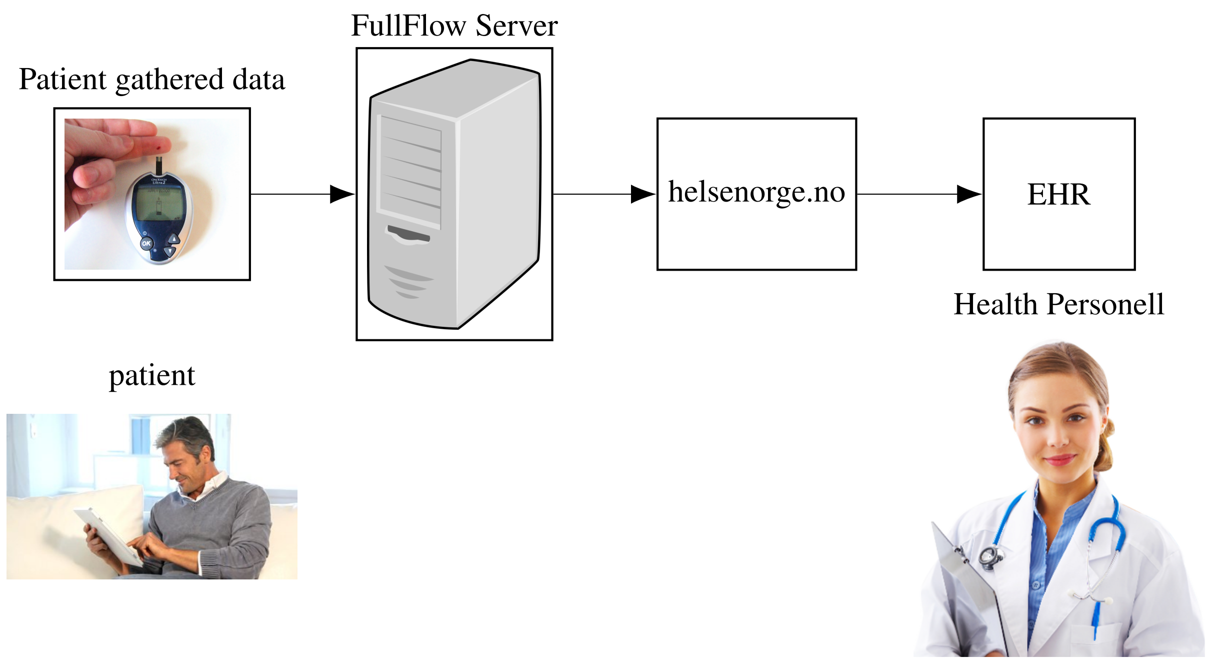


Figure 1.1 Simplified flow of data

1.5.1 Purposes

The end goal of FullFlow is to improve health outcomes. The data should flow between patients, primary health care and secondary health care EHRs. With available patient gathered data from diabetes patients, GPs will be more effective and efficient. [20]

1.5.2 FullFlow Architecture

Explanation to figure 1.2

Stage 0: **EHR initiation**

- A medical worker (GP) initiates the flow of data by sending a request to Helsenorge.
- Helsenorge gathers a FHIR schema and alerts the patient

Stage 1: **Data collection**

- The patient logs into FullFlow server with ID-Porten. (This may be done automatically by the smartphone's application)
- FullFlow gathers the data and goes automatically to the next stage.

Stage 2: **Data Analysis**

- FullFlow generates reports.
- FullFlow attatches the report to the Schema.

Stage 3: **Data Consulting**

- The EHR software gathers the scheme from the message queue at helsenorge.no.
- EHR presents the scheme.
- From the Personal Data Act, “The controller shall not store personal data longer than is necessary to carry out the purpose of the processing.” The Data should only be stored if it is interesting for the clinical procedure.
- If the data is not interesting, delete it.

Stage 4: **Data Export**

- The data shall be stored into PHA.
- The Patient will be alerted.

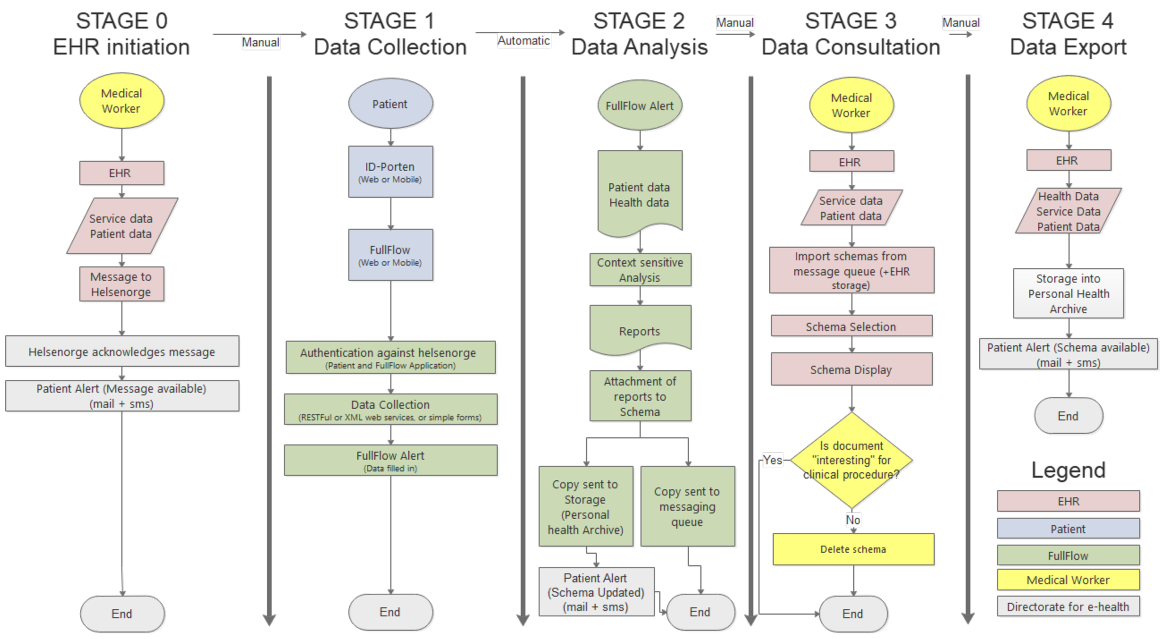

STAGE 0
EHR initiation
Manual
STAGE 1
Data Collection
Automatic
STAGE 2
Data Analysis
Manual
STAGE 3
Data Consultation
Manual
STAGE 4
Data Export
Medical Worker
EHR
Service data Patient data
Message to Helsenorge
Helsenorge acknowledges message
Patient Alert (Message available) (mail + sms)
End
Patient
ID-Porten (Web or Mobile)
FullFlow (Web or Mobile)
Authentication against helsenorge (Patient and FullFlow Application)
Data Collection (RESTFul or XML web services, or simple forms)
FullFlow Alert (Data filled in)
End
FullFlow Alert
Patient data Health data
Context sensitive Analysis
Reports
Attachment of reports to Schema
Copy sent to Storage (Personal health Archive)
Copy sent to messaging queue
Patient Alert (Schema Updated) (mail + sms)
End
Medical Worker
EHR
Service data Patient data
Import schemas from message queue (+EHR storage)
Schema Selection
Schema Display
Is document "interesting" for clinical procedure?
Yes
No
Delete schema
End
Medical Worker
EHR
Health Data Service Data Patient Data
Storage into Personal Health Archive
Patient Alert (Schema available) (mail + sms)
End
Legend
EHR
Patient
FullFlow
Medical Worker
Directorate for e-health

Figure 1.2 Overview of main flow of data
Used with permission from the owner, Alain Giordanengo.

1.5.3 Project goal

Figure 1.3 shows a proposed solution on how to receive the data coming from a message queue from helsenorge.no.

A message comes from Helsenorge.no's message queue through the internal helsenett network. Inside the message is a base64 encoded .zip file which contains one or several html pages with optional javascript and .css files. These files will get temporarily stored encrypted in a database. A REST API will serve the different Plenario instances.

Each health worker will have its instance of Plenario and Plenario will fetch and show the web pages gathered from the REST API.

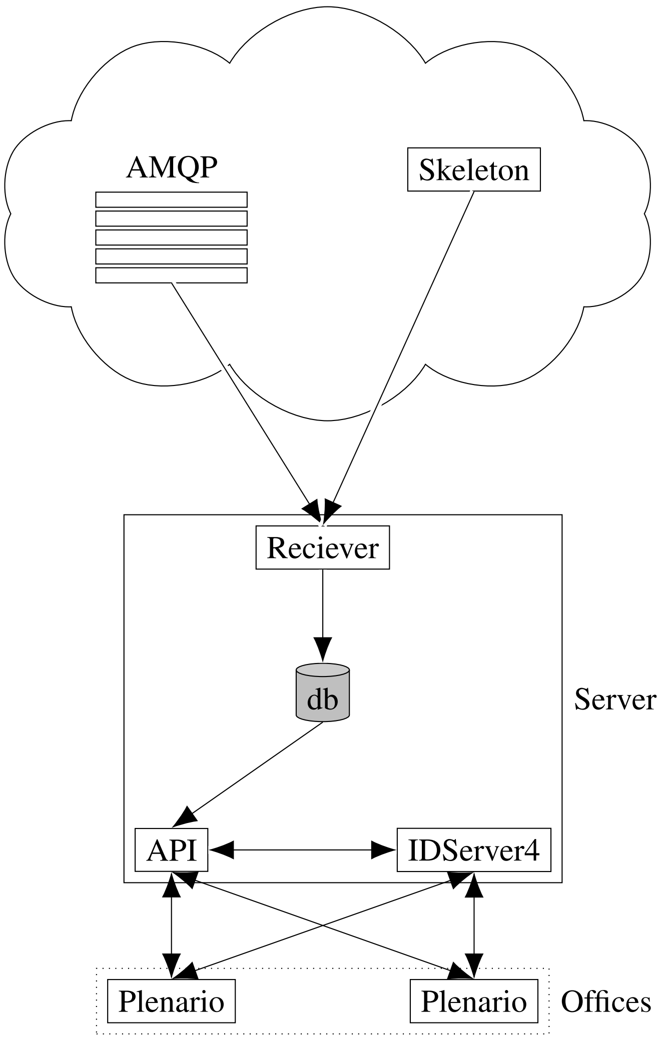


Figure 1.3 Scheme of internal architecture

1.6 Research question

The software that is going to be developed will handle sensitive health information about patients. It is important that this information is not accessible for attackers. It is also significant that attackers can not get access to the systems running the software as this could lead to unauthorized access to other data.

Note 1.2. Research question

How to integrate FullFlow into Infodoc's Plenario and how to carry out the risk assessment?

1.7 Thesis Outline

The introduction to the thesis has been made in this chapter. Chapter 2 Theoretical Background will list related work and explain the methodology used. Chapter 3 Technologies will go in-depth on technologies used in the integration prototype as well as alternatives.

In chapter 4 Design and Implementation explains how the prototype was designed and how the different technologies were implemented.

In chapter 5 Evaluation of Security, Testing Methods investigates different methods and frameworks while chapter 6 and 7 explain how threat modeling and penetration testing was carried out. Finally, the thesis has a conclusion in chapter 8.

Code snippets will be in `Computer Modern font and in a light gray box`. And longer Code examples will be in its own listing.

Listing 1.1 Example of multiline code

```
public static void Main(){
    var ans = 2+2
    ans = QuickMaths(ans);
}

public static int QuickMaths(var inn){
    return inn-1;
}
```

Note 1.3. Example of a note

Notes will be displayed inside a blue colored box.

Chapter 2

Theoretical Background

2.1 Related Works

This chapter will go through related works, different solutions to transport self-gathered health data to health care systems.

2.1.1 Vivify

Samsung did a pediatric remote patient monitoring program with Children's Health in Dallas. The solution used vivify to send data from patients tablet with different connect biometric sensors such as blood pressure monitors, weight scales and pulse oximeters.

This is closely related to what FullFlow is trying to accomplish, patient self-gathered health data sent directly to health care workers.

The result was more satisfied health care workers and patients that were able to live at home.[21]

Vivify (founded 2009) Claims to be the first end-to-end Remote Care Management platform to utilize consumer electronics. Vivify has different applications that cover 70+ conditions, eg. Heart failure and diabetes, but also includes an API for developers to integrate their applications. [22]

Something that differs Vivify and FullFlow is that FullFlow is complying with strict Norwegian health care laws while Vivify follows the less strict Data Protection Act (DPA). No Privacy Policy for Vivify's patients could be found.

Vivify can serve as an example of what is possible with a system like FullFlow, e.g., what kind of conditions that can be monitored.

2.1.2 Validic

In a study '*Connecting Smartphone and Wearable Fitness Tracker Data with a Nationally Used Electronic Health Record System for Diabetes Education to Facilitate Behavioral Goal Monitoring in Diabetes Care*' [23], researchers used Validic to transfer health data from diabetic patients to Chronicle Diabetes, a web-based system to allow health care workers to help diabetic patients assess their metabolic management.

Validic was used in the study because of its high-level security and experience working with large Electric Health Record (EHR) systems. Validic provides a Representational State Transfer (REST) Application Programming Interface (API) to allow EHR systems to gather data from 200+ application and devices. [24]

This study shares many similarities with the FullFlow project where Validic is used as a medium where the solution is to integrate health apparatus on the patient side and an easy way for the EHR systems to get that data out. Validic has focused on user Privacy and complies with US Privacy shield (See Note 2.1). They let the user look at their data, delete their data and has to give permission to 'health care portal providers' to access their data. [24]

Note 2.1. EU-US Privacy Shield

General Data Protection regulation (GDPR) does not allow the exchange of user information to countries that has inadequate Privacy laws. The United State is one of those countries. To circumvent this problem, GDPR allows transferring data to US companies that comply with the EU-US Privacy Shield.[25]

2.2 Methodology

A prototype addition to Infodoc's Plenario was created to receive data from the FullFlow servers. The data is HTTP files together with Javascript that shows graphs and other visualizations generated with data diabetes patients has gathered.

In order to make the prototype, different technologies needed investigation. Well-known technologies were first investigated, and alternatives were found through searching the Internet and reading articles. The Open Web Application Security Project (OWASP), a well-known not-for-profit organization that delivers tools, libraries, and documentation about security, was a source in finding different threat modeling frameworks. Infodoc, The Norwegian Directorate of eHealth (NDE) and the FullFlow project was also used as sources when investigating what technologies to use.

The implementation of FullFlow was evaluated with Threat Modeling and Penetration testing. When doing a risk assessment, the goal is to quantify the risk. Since the factors that made up risk, severity, and probability, are scored based on limited data, this is a way of quantizing a qualitative method. Looking at the example in table 2.1, the qualitative factors that make up risk is scored and combined to give a quantitative risk score.

	Text (Qualitative)	Score (Quantitative)
Threat	Emloyee loses memory stick with sensitive material.	
Probability	At least once a year	4
Consequence	Damaged reputation	3
Risk (P x C)	High	12

Table 2.1 Risk Scoring Example

In order to find the best-fitting methodology to assess the risk, different methods were studied. Sources were OWASP and The Code of Conduct for information security in the healthcare and care services (‘Normen’). ‘Normen’ covers all aspects of information security in health care to be compliant with Norwegian law. In order to be connected to the ‘Norwegian Health Network,’ as Infodoc’s Plenario is, the organization needs to comply with ‘Normen.’ Therefore the risk scoring was done following ‘Normen’s guidelines. [26]

The threat modeling was conducted with the help of Microsoft Threat Modeling Tool. Microsoft Threat Modeling Tool enables users to draw a Data Flow Diagram (DFD) with assets, processes and trust boundaries.

Risks of using threat modeling are that it is hard to find and mitigate all security issues. Threat modeling was formalized on best practices on already existing methods and therefore lack academic research. [27]

After the threat modeling, a chosen number of the threats was then penetration tested. The penetration testing was done at Norwegian Center for E-health Research (NSE) in Tromsø with co-supervisor, Alain Giordanengo and fellow student Joakim Tran. A server and client was set up on to different computers in a closed Local Area Network (LAN). Postman, a tool to send and receive Hypertext Transfer Protocol (HTTP) packages was used. To test for threats against malicious JavaScript, different attacks was carried out with the help of Joakim Tran.

Chapter 3

Technologies

To develop the suggested solution (see figure 1.3), different technologies needed investigation. Underlying technologies such as Base64 encoding, REST and URL as well as the more complex solutions such as authorization servers and web engines. Different alternatives were also explored to ensure that the most applicable technology was used.

Technologies that were already in use by Infodoc were preferred since this meant easier integration and the ability to use Infodoc's expertise.

3.1 Definitions

This chapter will make use of the following Definitions. All definitions are gathered from the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC) since ISO is the leading organization to make international standards with over 160 members and 21 000 international standards. [28]

Definition 3.1 (ISO/IEC 18014-2)**.** Authentication: Provision of assurance in the claimed identity of an entity.

Definition 3.2 (ISO/IEC 2382-8:1998)**.** Authorization: granting of rights, which includes the granting of access based on access rights

Definition 3.3 (ISO/IEC 2382-17:1999)**.** Entity: any concrete or abstract thing that exists, did exist, or might exist, including associations among these things.
Examples: A person, an object, an event, an idea, a process, etc.

Definition 3.4 (ISO/IEC 24760)**.** Identity (ID): Set of attributes related to an entity

Definition 3.5 (ISO/IEC 2382-8:1998)**.** Identity authentication: performance of tests to enable a data processing system to recognize entities
Example: The checking of a password or of an identity token.

3.2 REST

REST and SOAP are the leading architectures for transferring data over a network. Since REST is more modern and straightforward, this is what will be used in this project.

Representational State Transfer (REST) is an architectural style to make an Application Programming Interface (API). It makes data available as resources that are acceded with nouns. REST does not specify in which format the server should respond, but JavaScript Object Notation (JSON) and eXtensible Markup Language (XML) are the most common. REST REST defines containing six interaction constraints (see chapter 3.2.2). [29].

REST is chosen since it is the industry standard and allows for microservice architecture, which is on par with how Infodoc constructs their software infrastructure.

REST will be used in most of the communication throughout the system except for the communication with helsenorge.no which will use Advanced Message Queuing Protocol (AMQP).

Let us look at the REST API for Google Gmail. To get a message (mail) from a user, one would merely use `GET https://www.googleapis.com/gmail/v1/users/userId/messages/id` and the API would respond with the particular message from the particular user. Also, notice the directory like structure.

3.2.1 History

Simple Object Access Protocol (SOAP) was made by Dave Winer in 1998, and is a protocol that allows systems to exchange information with XML. Systems that communicate with SOAP are tightly coupled and rigid. This was one of the problems Roy Fielding wanted to fix when he in 2000 created the specification for REST.
One of the first big sites to use REST API was eBay, Amazon, and Flickr. [30]
Programmableweb.com has a gathered API's since it was founded in 2005. The graph in figure 3.1 show that internet-based API becomes more and more popular with nearly 2000 new API's added yearly on average [31].

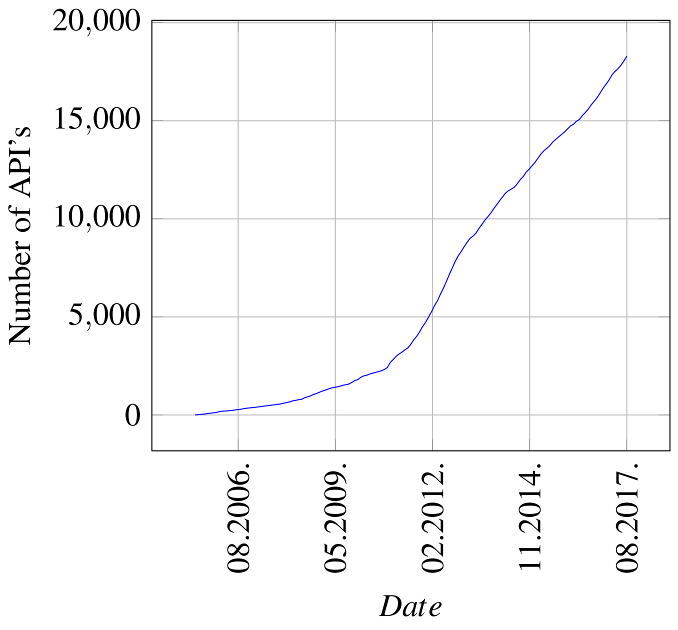


Figure 3.1 Programmableweb's API directory

Table 3.1 Six restrictions in REST

Client-Server
Stateless
Cache
Uniform Interface
Layered System
Code on Demand

3.2.2 Constraints

REST consists of 6 constrictions, as seen in table 3.2.2. The first restriction, *Client-Server*, is a separation of client interface and server data.

The server should also be *stateless*, in other words, it should not contain any information about what state the client is. This means in practice that the client must have control over what state it is.

Responses from the server must be labeled as either *cacheable* or *not cacheable*. This can in many cases eliminate responses from the server.

The *Uniform Interface* is in the heart of REST architecture and describes the interface between server and client. «REST is defined by four interface constraints: identification of resources; manipulation of resources through representations; self- descriptive messages; and, hypermedia as the engine of application state.» [1]

The *Layered System* constraint enables the client to not care if it is connected directly to the server or an intermediary. This allows for scalability since this allows the server to enable load-balancing.

An optional constrain within REST is *Code on Demand*. This allows clients to download and executing code in the form of applets or scripts. [29]

3.2.3 Data format

Table 3.2 REST data formats

Data element	*Example*
Resource	The intended conceptual target of a hypertext reference
Resource identifier	URL, URN
Representation	XML, JSON, HTML document, JPEG image
Representation metadata	Media type, last-modified time
Resource metadata	Source link, alternates, vary
Control data	if-modified-since, cache-control

> “The key abstraction of information in REST is a *resource*. Any information that can be named can be a resource: a document or image, a temporal service (e.g. today’s weather in Los Angeles), a collection of other resources, a non-virtual object (e.g. a person), and so on. In other words, any concept that might be the target of an author’s hypertext reference must fit within the definition of a resource” [29, p. 88]

When looking back to our example in chapter 3.2, the *Assets* was the resource.

Resource identifier, or as it’s called in Hypertext Transfer Protocol (HTTP), Uniform Resource Identifier (URI), is an identifier that can either be an Uniform Resource Locator (URL) or Uniform Resource Name (URN). A URL locates a resource, e.g. `https://www.facebook.com/anders.s.nilsen` while URN names a resource, e.g `urn:isbn:9780007117116` wich is the isbn code for The Lord of the Rings book. `foo://example.com:8042/over/there?name=ferret#nose` can be broken down into components as seen in table 3.3 [32].

REST does not specify what kind of data *representation* should be used. XML and JSON are the de facto standard.

A *representation* consists of data and representation metadata describing the data as name-value pairs (e.g.in HTTP it comes in the header).

Table 3.3 Components of an URI

scheme	*authority*	*path*	*query*	*fragment*
foo:	//example.com:8042	/over/there/	?name=ferret	#nose

Resource metadata describe the resource and may be included in the response message. It could contain links to additional information or other resources. [29]

Control data could be information in the request that stated you only want information if the representation has changed. As Fieldman described it: “*Control data* defines the purpose of a message between components”.[29]

3.3 OAuth2

Plenario has plans to implement OAuth into their Electric Health Record (EHR) software. OAuth will therefore be used in this project.

OAuth is used for giving access to resources to one server from another server, without the need to give away credentials (username and password).

OAuth defines three roles, the User, who grants access to a resource on one server, called the Service Provider for another server called the Consumer.

An example would be a mailing site wants a user, Anna, to give access to her contacts from a social site. OAuth solves this by letting Anna login to her social site and authorize the request to her contacts by the mailing site. Note that the mailing server does not get Anna’s user credentials but only an access token. [33]

3.3.1 Terminology

OAuth defines four roles, resource owner, resource server, the client and the authorization server.[34]

Resource owner

“An entity capable of granting access to a protected resource.”
Usually, a person (end-user) that wants to grant access to some information (resource) for an application (client).[34]

Client

“An application making protected resource requests on behalf of the resource owner and with its authorization.”
Not to be confused with the resource owner, the client is often an application that wants access to the owner’s resource.[34]

Resource server

“The server hosting the protected resources, capable of accepting and responding to protected resource requests using access tokens.”[34]

Authorization server

“The server issuing access tokens to the client after successfully authenticating the resource owner and obtaining authorization.”
Maybe the same server as the resource server.[34]

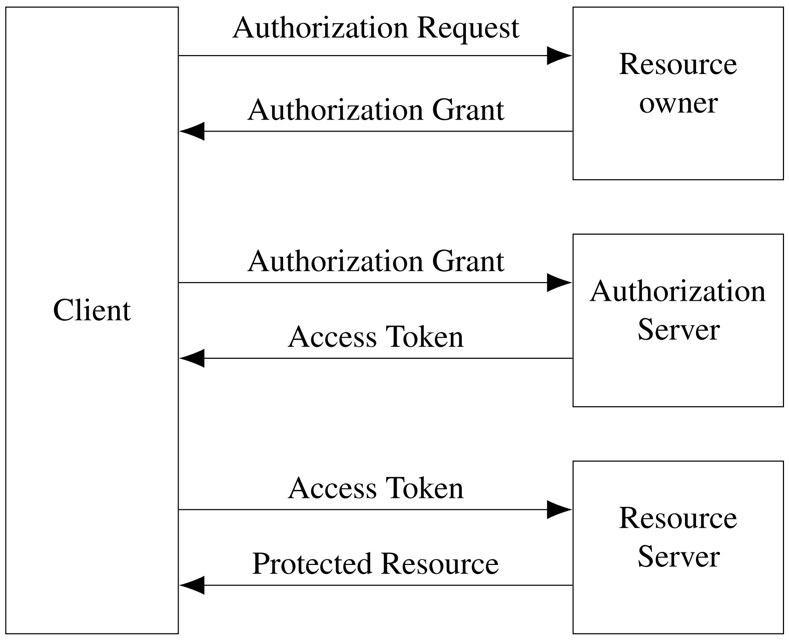


Figure 3.2 OAuth authorization flow

Looking at figure 3.2, an end-user (resource owner), Bob, wants to share tweets (Protected Resource) from his Twitter (Resource server) account.

3.4 OpenID Connect

Plenario has plans to implement OpenID Connect into their EHR software. No other alternatives will, therefore, be examined and OpenID Connect will be used in this project.

OpenID Connect (OIDC) 1.0 is an identity layer on top of OAuth 2.0. While Oauth 2.0 only provides authorization, OIDC provides also authentication. [1]
OIDC ensures that an entity is what it claims to be, e.g. authentication validation (see definition 3.5).

OAuth 2.0 describes Authorization Request as a request from the client to the authorization server. It usually contains a scope on what needs to be authorized. OIDC extends OAuth's Authorization Requests to not only authorize but to authenticate. This is called an *Authentication request*.[1]

A JSON Web Tokens (JWT) (pronounced 'jot') is a standardized way of transferring tokens. JWT can be encrypted and can then be called a JSON Web Encryption (JWE), or signed and called a JSON Web Signatures (JWS). To clarify, a JWE and JWS are different implementations of JWT.

Listing 3.1 JWE serialization

```
BASE64URL(JWE Encrypted Key) || '.' ||
BASE64URL(UTF8(JWE Protected Header)) || '.' ||
BASE64URL(JWE Initialization Vector) || '.' ||
BASE64URL(JWE Ciphertext) || '.' ||
BASE64URL(JWE Authentication Tag)
```

Listing 3.2 JWS serialization

```
BASE64URL(UTF8(JWS Protected Header)) || '.' ||
BASE64URL(JWS Payload) || '.' ||
BASE64URL(JWS signature)
```

[35]

The Payload claims in the form of a JSON object whose members are the claims.

An OIDC *ID Token* is a JWT where some required claims in the payload.[1]

In OpenID Connect an *OpenID Provider* is an OAuth 2.0 Authorization Server that is capable of Authenticating the End-User. It also provides Claims to a Relying Party about the Authentication event and the End-User. [1]

The *Relying Party* is an OAuth 2.0 is a Client application that requires End-User Authentication and claims form an OpenID Provider. In the Infodoc addition, this would be server-side API. [1]

An *user agent* is in our case the Client side of Infodoc, but often it is just a web browser. [36]

Userinfo Endpoint: "Protected Resource that, when presented with an Access Token by the Client, returns authorized information about the End-User represented by the corresponding Authorization Grant. The UserInfo Endpoint URL MUST use the https scheme and MAY contain port, path, and query parameter components."[1]

3.4.1 Flow

OIDC specifies authentication using three different flows (see table 3.4), Authorization Code Flow, Implicit Flow, and Hybrid Flow. Below follows the general flow applicable to all three different flows (see table 3.4).
The Relying Party (RP) (Client) asks the OpenID Provider (OP) If it can Authenticate an End-User. The OP authenticates the End-User and is given authorization. OP Authenticates the End-User for RP (e.g., confirms that the End-User is whom he claims) with an ID Token and usually also an Access Token. RP requests UserInfo with the Access token. The UserInfo Endpoint in OP will give information about the End-User in return.[1]

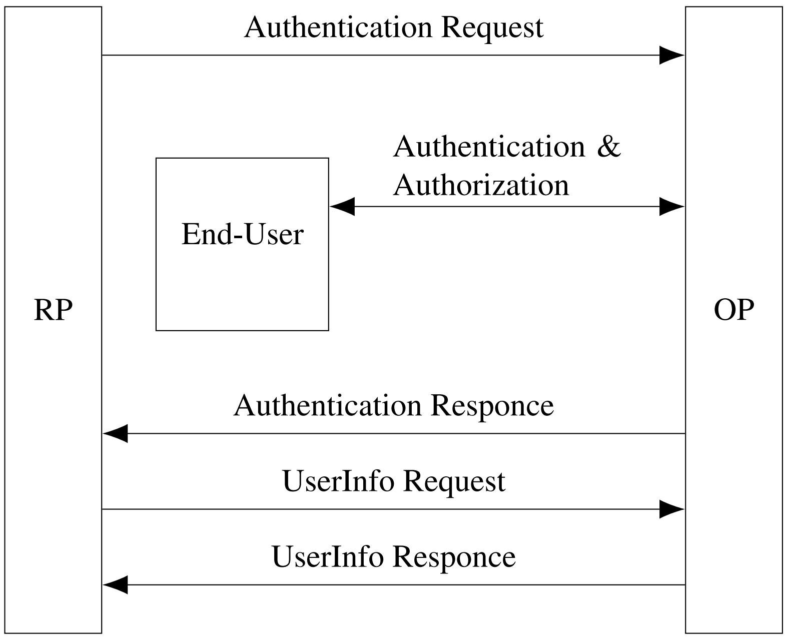


Figure 3.3 OpenID Connect 1.0 Flow [1]

3.5 PDF

Portable Document Format (PDF) is a popular file format for documents. It was created by Adobe in 1993 and made as an open standard in 2008. One of the reasons it has gotten so

Table 3.4 Summarization of the three OIDC flows [1]

Property	Authorization Code Flow	Implicit Flow	Hybrid Flow
All tokens returned from Authorization Endpoint	no	yes	no
All tokens returned from Token Endpoint	yes	no	no
Tokens not revealed to User Agent	yes	no	no
Client can be authenticated	yes	no	yes
Refresh Token possible	yes	no	yes
Communication in one round trip	no	yes	no
Most communication server-to-server	yes	no	varies

popular is that most platforms can easily export documents to PDF and that most systems have software that supports viewing PDF files.[37–39]

PDF could be used to view health data, but will not be chosen since it offers fewer capabilities to view data dynamically as opposed to Hypertext Markup Language (HTML) technologies.

3.5.1 File Structure

A PDF file consists of 7-bit ASCII characters. The file is divided into a Header, Body, Cross-Reference Table and a Trailer.
The header is a 5 character line stating the version e.g. `%PDF-1.3`.

The *body* consists of number of *objects*.

The *objects* can be boolean values, Integer and Real numbers, Strings, Names, Arrays, *Dictionaries*, Streams, and the null object.

A *dictionary* are objects written as key-value pairs encapsulated in double angle brackets. Note that Dictionary can also contain other dictionaries.

Listing 3.3 PDF Dictionary example

```
<< /Type /Example
/Subtype /DictionaryExample
/Version 0.01
/IntegerItem 12
/StringItem (a string)
```

```
>>
```

[40]

Each page in a PDF file is represented as a page object - a dictionary with references to its content and also attributes to that page. The pages are then tied together with a Page tree. However, in the root is the Catalog Dictionary. The Catalog dictionary contains

The Catalog Dictionary may include *Name Dictionary*. Using the key `JavaScript`, it is possible to link strings to document-level JavaScript.

3.5.2 Example of JavaScript Execution

Listing 3.4 JavaScript Execution inside PDF

```
<<
/Type /Catalog
/Pages 5 0 R
 /Names << % the Javascript entry
    /JavaScript <<
      /Names [
        (EmbeddedJS)
        <<
          /S /JavaScript
          /JS (
            app.alert('Hello, World!');
                        app.launchURL\("http://localhost:8000/altered.pdf", true\);
                        )
        >>
      ]
    >>
  >> % end of the javascript entry
>>
```

When this (Listing 3.4) is inserted into a PDF file, depending on the software, will redirect to an alternative PDF file.

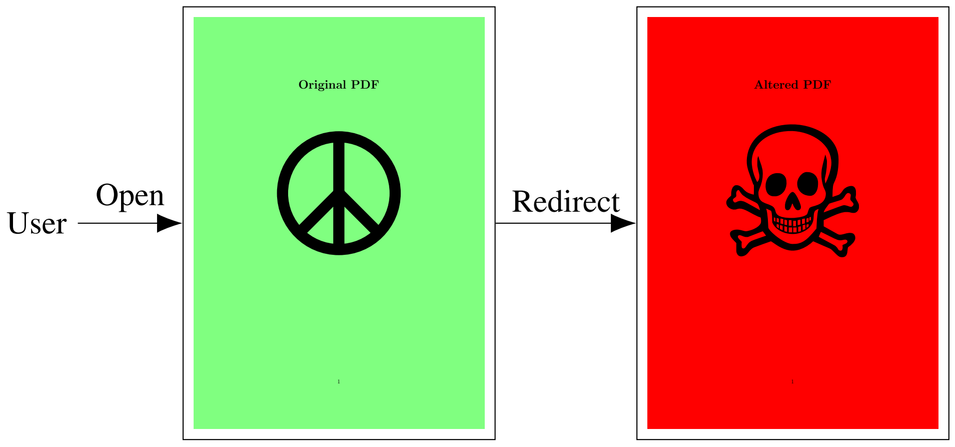


Figure 3.4 Execution

3.5.3 Acrobat Reader

Adobe created Acrobat Reader (formerly Reader) in June 1993 and released together with the PDF standard. It is widely used to view PDF files maybe because it is free software.

As seen in figure 3.5, Adobe Acrobat has had a high number of vulnerabilities, and the percentage of high risks are significant.

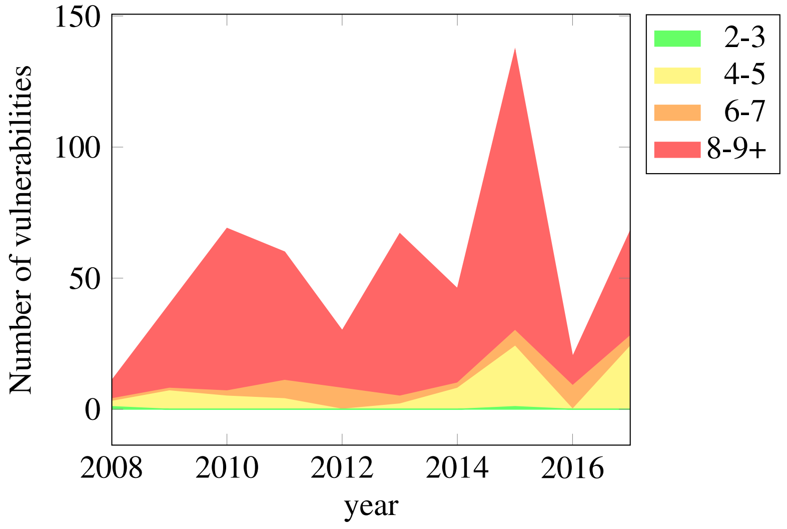


Figure 3.5 Adobe Acrobat Reader vulnerabilities grouped after CVSS Score [2]

Since 2010 Adobe Acrobat Reader has incorporated the same sandbox solution as Chromium (See Chapter 3.6.3). This helps protect against JavaScript attacks [41]. The attack showed in 3.5.2 will only raise a messagebox asking if you want to run the JavaScript, there is also a choice to never get the warning message.
According to Microsoft security Intellegence Report Security Intelligence Report (SIR), attacks using JavaScript in PDFs (win32/Pdfjsc) are dropping. But as Synatics report shows [41] it is possible to use multiple of different obscurifications, making detection hard.

3.6 Chromium

“Chromium is an open-source browser project that aims to build a safer, faster, and more stable way for all Internet users to experience the web.”[42]
Chromiums rendering engine is sandboxed and detached from the kernel. The sandbox only allows the rendering engine to communicate with the kernel.

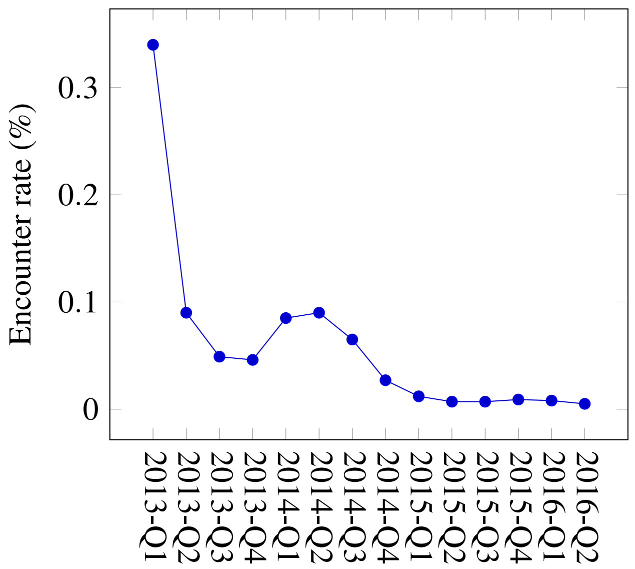


Figure 3.6 Encounter rate of Win/Pdfjsc as percentage of all reporting computers [3–8]

Note 3.1. Engine choice

Chromium will be used in the Plenario addition so the doctors can view the HTML files.

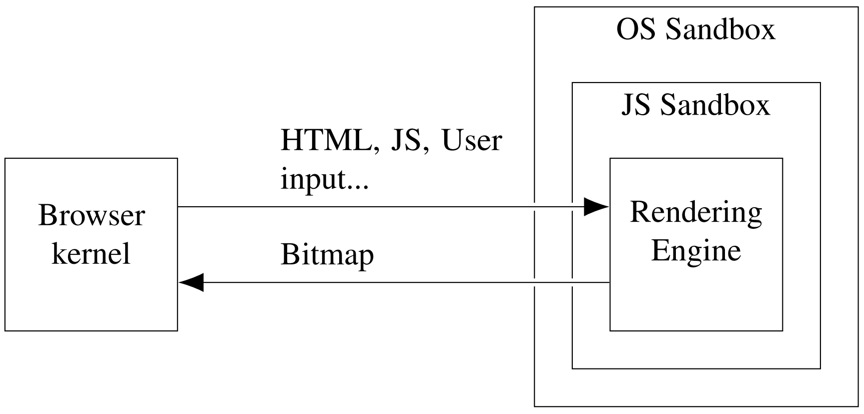


Figure 3.7 Chromiums architecture

3.6.1 Rendering Engine

The rendering engine takes care of all potential harmful code in a low privileged sandbox and generates a bitmap which it sends to the high-privileged browser kernel. If the rendering engine gets compromised because of harmful JS and wants to get files, it cannot since the only way of getting files from the system is through the Kernel.

Table 3.5 Responsibilities [10]

Rendering Engine	*Browser Kernel*
URL Parsing	URL Parsing
Unicode Parsing	Unicode Parsing
HTML parsing	Cookie database
CSS parsing	History database
Image decoding	Password database
JavaScript interpreter	Window management
Regular expressions	Location bar
Layout	Safe Browsing blacklist
Document Object Model	Network stack
Rendering	SSL/TLS
SVG	Disk cache
XML	parsing Download manager
XSLT	Clipboard

3.6.2 Browser Kernel

The Browser Kernel is the glue between the Rendering Engine and the operating system and takes care of what needs to be sent too and from the system as well as storing data.
When the Rendering engine wants a file from the user, it can ask the Kernel to show the file upload control. [10] [43]

3.6.3 Sandbox

"The sandbox is a C++ library that allows the creation of sandboxed processes". Chromium uses it to launch rendering engines, but the sandbox can also be used with other applications. It restricts the sandboxed processes from using the disk.
The process that creates a sandboxed process is called a 'Broker' while the restricted, sandboxed process, is called a 'Target.'

"The sandbox relies on four Windows security mechanisms:

- A restricted token
- The Windows job object
- The Windows desktop object
- Windows Vista and above: The integrity levels"[44]

[44]

3.6.4 Vulnerabilities

In Chrome, 172 vulnerabilities was gathered by National Vulnerability Database (NVD) in 2016. 24 Vulnerabilities had a Common Vulnerability Scoring System (CVSS) score of 9, as seen in the figure 3.8. [2] [45]

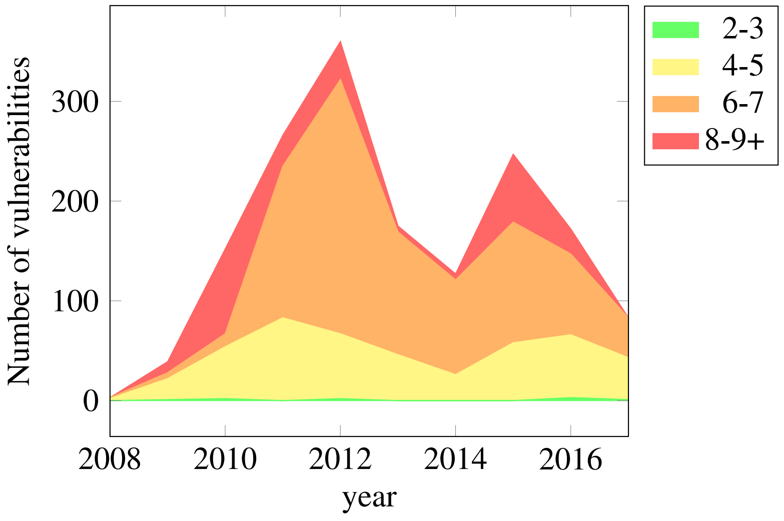


Figure 3.8 Chrome vulnerabilities grouped after CVSS Score [2]

Opening local HTML files - Same origin vulnerability.

To open a local file creates a security problem. This is because of the Same-origin policy wich is described by Jesse Rudman in MDN Web docs:

> "The same-origin policy restricts how a document or script loaded from one origin can interact with a resource from another origin. It is a critical security mechanism for isolating potentially malicious documents." [46]

As pointed out by Adam Barth, if opening a local HTML file, it could contain an `<iframe>` that links to a site with private information (figure 3.10). Since the frame is in the same origin as the malicious JavaScript (JS), it can access all its content with `frames[0].document.documentElement.innerHTML`. This would not be possible with a HTML document on the web, since the site malicious code and the private information would be in different origins (figure 3.9).[47][48]
This attack is can only be used on web pages that does not set the `X-Frame-Options` to `DENY` or `SAME ORIGIN`, wich disallows the site to be shown in a `<Iframe>`.[49]

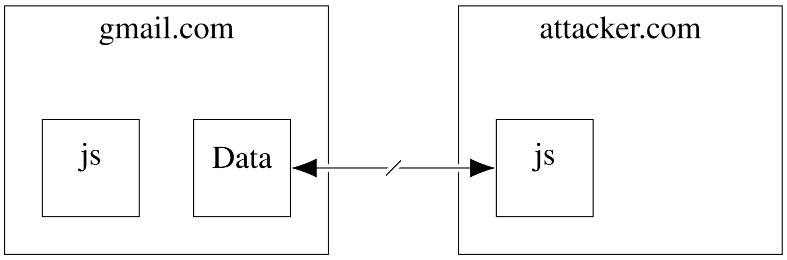


Figure 3.9 Example of different origin

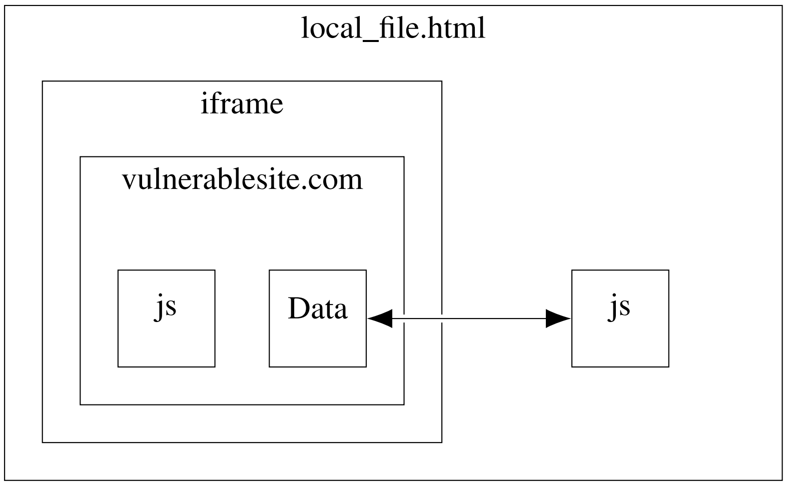


Figure 3.10 Example of Same origin

FAT filesystem

Files in File Allocation Table (FAT) file systems does not support access control lists (ACL). Since this is required by the sandbox to determine whether or not the rendering engine is allowed access, a compromised rendering engine could both write and read from a FAT file system as long as it knows/guesses the file path.

3.7 Alternative Web Engines

To present the HTML and JS a web browser will be embedded in Infodoc's Plenario, other alternatives was considered but not chosen since Chromium was most applicable

The Windows WebBrowser, Gecko and Chromium will be evaluated. The inbuilt .Net *WebBrowser* is easy to implement and but uses the Internet Explorer (IE) already installed on the clients machine[50]. If a client uses an insecure version of IE this will lead to security issues.

Gecko is Firefox embedded framework. With poor documentation and examples, this will not be used.

Electron is a framework for creating native applications with HTML Cascading Style Sheets (CSS) and JavaScript. It appends Chromium as the web-engine. It was originally developed for Github's Atom editor but has also been used to build Slack and Visual Studio Code.
Since Electron is not compatible with Windows Forms and is not designed to integrate with .Net, Electron will not be useful for this integration.

Even though it uses Chromium, it does not use Chromium's sandbox technology. Making a breach much more serious.[51]

3.8 Base64 encoding

Base 64 encodes data in the form of a base 64 alphabet. This allows encoding data to a readable text file.
Email [RFC 822] [52] supported only US-American Standard Code for Information Interchange (ASCII) characters.Multipurpose Internet Mail Extensions (MIME) [RFC 2045] [53] was created to allow for character set other than US-ASCII and also "non-textual message bodies" (images). Using Base64 encoding, emails could be written with UTF-8 [RFC 3629] [54][55].

Several technologies that are mentioned in this chapter are using Base64 encoding, but most importantly all the JWTs are encoded in this fashion.

The alphabet exists of 64 value-char pairs with an edition of the padding character '=' (see table 3.6). Some alternatives exists, such as an URI safe option where '+' and '/' are changed with '-' (minus) and '_' (underscore).[55]

Base64 encoding takes binary data in groups of 24 bits, divides it into 4 groups of 6 bit each and represent them with the base64 alphabet. See table 3.7 [55]

3.9 Advanced Message Queuing Protocol (AMQP)

The Norwegian Directorate of eHealth (NDE) might use AMQP to send data from Helsenorge.no. This is however not decided, but since NDE has used AMQP in other solutions such as Digital Dialog, we will set up an Azure Service Bus that implements AMQP.

AMQP is an application protocol that sits on top of TCP/IP. It became an ISO standard in 2014 written by Organization for the Advancement of Structured Information Standards (OASIS) Task Committee (TC)

Table 3.6 The Base64 alphabet

value	*Character*	*value*	*Character*	*value*	*Character*	*value*	*Character*
0	A	16	Q	32	g	48	w
1	B	17	R	33	h	49	x
2	C	18	S	34	i	50	y
3	D	19	T	35	j	51	z
4	E	20	U	36	k	52	0
5	F	21	V	37	l	53	1
6	G	22	W	38	m	54	2
7	H	23	X	39	n	55	3
8	I	24	Y	40	o	56	4
9	J	25	Z	41	p	57	5
10	K	26	a	42	q	58	6
11	L	27	b	43	r	59	7
12	M	28	c	44	s	60	8
13	N	29	d	45	t	61	9
14	O	30	e	46	u	62	+
15	P	31	f	47	v	63	/

Table 3.7 Encoding *piw* to *cGl3*

Message to encode	p								i								w							
Message in binary	0	1	1	1	0	0	0	0	0	1	1	0	1	0	0	1	0	1	1	1	0	1	1	1
6-bit grouping in decimal	28						6						37						55					
Base64	c						G						l						3					

The Protocol is divided in 5 main layers.

1. Type system and encoding
2. Transport layer
3. Message format
4. Transactions
5. Security

AMQPs describe their types in XML. The XML contains information about the name, length, and a Type System code. AMQP describes four different Type Systems. Primitive types, Restrictive types Described types and composite types. Primitive types describe common types such as boolean and int. Restrictive types are a new type in AMQP that describes a subset of another type. Described types are custom annotations that are used for other types. The composite type is a superset of other types.

The Transport layer is a low-level peer-to-peer protocol for transporting messages between two nodes. The Transport layer also defines concepts such as connection, session and link to name a few, each with their own clearly defined responsibilities.

Chapter 4

Design and Implementation

This chapter will explain the design and implementation process.

4.1 Data format

To send patient gathered content, one could choose different options to send the data, and some are shown in table 4.1.

The FullFlow server has the ability to send either option 4 or 5 but the following options was discussed.

Table 4.1 Options on how to send data

Option	*Description*
1 All FHIR Raw Data	The data sent to the data server would just be forwarded.
2 Relevent FHIR Data	The data that data server uses to generate charts.
3 HTML	A simple HTML for viewing the data.
4 HTML + JS	Allowing dynamic graphs.
5 PDF	An PDF file for viewing data.

These formats all have their downsides and upsides. Option 1, to send everything, is out of the question since this would be too much unnecessary traffic. To only send relevant data, option 2, would result in making graphs locally and present it to the General Practitioner (GP), this is also a bad idea since every time the FullFlow server changed their logic, the graphs on Infodoc's side would have to change as well. Another option would be to get the Fast Healthcare Interoperability Resources (FHIR) data with an HTML file. The Hypertext Markup Language (HTML) file would contain simple graphs. This is a good option since it

Figure 4.1 option 1,2

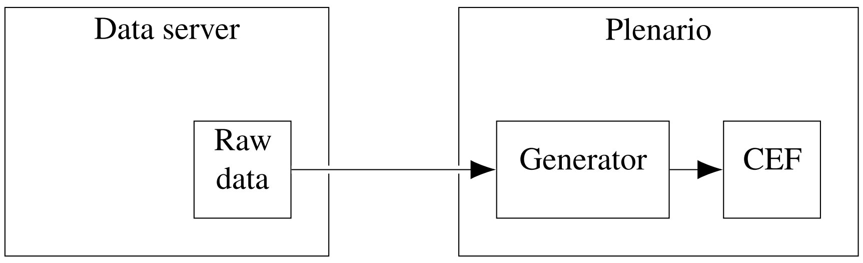
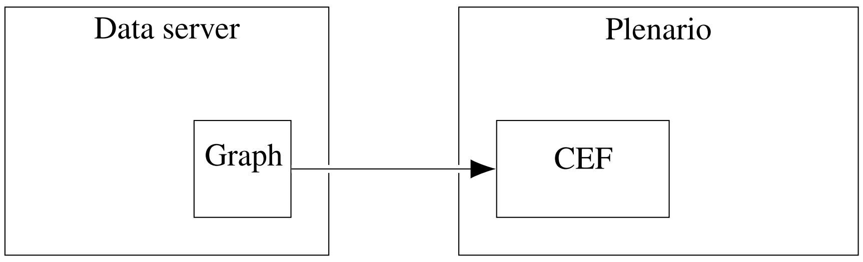


Figure 4.2 option 3,4,5



is both secure and offers the possibility for the GP to easily change between the different graphs.
In addition, it is also possible to send JavaScript (JS), this would enhance the user experience, but could also compromise the security. Since the JS is sent from someplace other than Infodoc's server, it has to be classified as untrusted code.
The last option (5) is to send it as a Portable Document Format (PDF) file. This is perhaps the easiest solution.

Option 4 was chosen since it raises essential security questions on how to run untrusted JS and also gives the best user experience.

4.2 Data Flow

Throughout the FullFlow projects life cycle, the architecture of how the data should flow changes. Initially the data together with the visualizations (HTML + JS) should be sent together with the data through Helsenorge.no Data should be sent from Helsenorge.no in a message queue (AMQP), and the template for viewing the data (HTML + JS) will be gathered from the FullFlow server. The templates that are gathered from the FullFlow server will not contain any sensitive data and can, therefore, be gathered without authentication. It is however important that the template is not tampered with.

4.3 Implementation

It took some time to implement a solution that could receive and show html files. The method used to develop this software was first to make a Minimal Viable Product (MVP) and then do Continuous Integration (CI).

This section does therefore not mirror in what order the development was done but is divided into different parts of the solution for better readability.

4.3.1 Plenario addition

First a standalone program that implemented chromium, extracted a HTML from a Base64 String and showed it's content was developed (see figure 4.3).

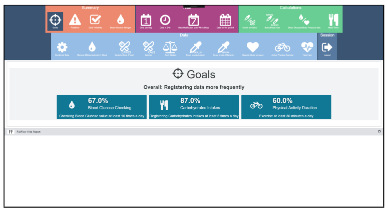

Goals
Overall: Registering data more frequently
67.0%
Blood Glucose Checking
87.0%
Carbohydrates Intakes
60.0%
Physical Activity Duration

Figure 4.3 Chromium Embedded Framework (CEF)

Then this program was integrated in Infodoc's Plenario (see figure 4.4). In CEF It is possi-

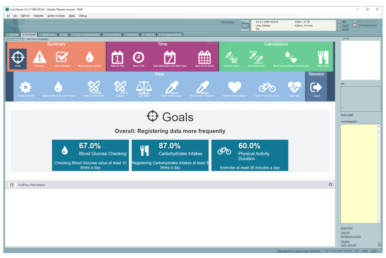

Goals
Overall: Registering data more frequently
67.0%
Blood Glucose Checking
87.0%
Carbohydrates Intakes
60.0%
Physical Activity Duration

Figure 4.4 Chromium Embedded Framework (CEF) in Plenario

ble to create a custom scheme (scheme as in part of an Uniform Resource Locator (URL), see chapter 3.2.3). Creating a `CefCustomScheme` with the help of `FolderSchemeHandlerFactory`, you need to specify a root folder, scheme name and a hostname. The `FolderSchemeHandlerFactory` would then check the code as follows.

Listing 4.1 Snippet from `IResourceHandler`'s method `Create`

```
//Check the file requested is within the specified path and that the file exists
if(filePath.StartsWith(rootFolder, StringComparison.OrdinalIgnoreCase) && File.Exists(filePath))
{
    var fileExtension = Path.GetExtension(filePath);
    var mimeType = ResourceHandler.GetMimeType(fileExtension);
    var stream = File.OpenRead(filePath);
    return ResourceHandler.FromStream(stream, mimeType);
}
```

This allows only files in a given `rootFolder` to be rendered.
A tighter coupling between the HTML files and code could be made by linking it into the program's resources. This would, however, make it difficult to update the Viewer.

Since the application does not need any data or information from the Internet or the rest of the filesystem, it is safest to give CEF access to just the custom scheme and nothing else.

A quick search on the web showed that it is possible from the Developer tools in Google Chrome to simulate an offline network. This did not work in CEF.

Listing 4.2 Proxy settings

```
{settings.CefCommandLineArgs.Add("proxy-server", "http = x; https = x; ftp = x; file = x");
```

Adding a non existing proxy for http, https, file and ftp schemes seems to work, but this is not the best solution as it does not explicit disallow theses schemes.

The solution was to create a `IRequestHandler` so that Plenario will have full control over what gets sent to and from chromium. The Requesthandler itself is added to the browser `myBrowser.RequestHandler = new RequestHandler();` . The Requesthandler returns default values except in the method `GetResourceResponseFilter` (as seen in listing 4.3) where it checks the current scheme and returns either a Blocking filter (Listing 4.5) or Pass-through filter (Listing 4.4).

Listing 4.3 Requesthandlers' method GetResourceResponseFilter

```
IResponseFilter IRequestHandler.GetResourceResponseFilter(IWebBrowser browserControl, IBrowser browser,
       IFrame frame, IRequest request, IResponse response)
{

    var url = new Uri(request.Url);
    if (url.Scheme == MySchemeHandlerFactory.schemeName)
    {
        return new PassThruResponseFilter();
    }

    //return new PassThruResponseFilter();
    return new BlockingResponseFilter();
}
```

Listing 4.4 Pass-through filter

```
public class PassThruResponseFilter : IResponseFilter
{
    bool IResponseFilter.InitFilter()
    {
        return true;
    }

    FilterStatus IResponseFilter.Filter(Stream dataIn, out long dataInRead, Stream dataOut, out long
            dataOutWritten)
    {
        if (dataIn == null)
        {
            dataInRead = 0;
            dataOutWritten = 0;

            return FilterStatus.Done;
        }
        dataInRead = dataIn.Length;
        dataOutWritten = Math.Min(dataInRead, dataOut.Length);

        dataIn.CopyTo(dataOut);

        return FilterStatus.Done;
    }

    public void Dispose()
    {

    }
}
```

Listing 4.5 Blocking filter

```
FilterStatus IResponseFilter.Filter(Stream dataIn, out long dataInRead, Stream dataOut, out long
        dataOutWritten)
{
    dataInRead = 0;
    dataOutWritten = 0;
    return FilterStatus.Error;
}
```

4.3.2 Message Queue Reciever

The system needs to be able to receive messages from helsenorge.no. An Azure Service bus has been created for testing purposes, this message queue will simulate helsenorge.no. In order to send messages to the queue. Service Bus Explorer will be used to examine and send messages to the queue, as well as a simple self-written program, able to send messages and files. It is crucial to implement this functionality as there are many security issues associated with how to receive and handle the data. SQLite will be used as it is a simple one file flat database that requires minimal setup.

Listing 4.6 Receiver.cs - extract

```
static void Main(string[] args)
{
    InitializeDb(); //Initialize SQLite
    var client = QueueClient.CreateFromConnectionString(connectionString, queueName);
    client.OnMessage(OnBrokeredMessage);
}


private static void OnBrokeredMessage(BrokeredMessage message)
{
    SaveMessageBodyInDb(message.GetBody<string>());
}

private static void SaveMessageBodyInDb(string body)
{
    using (DbConnection connection = dbFactory.CreateConnection())
    {
        if (connection != null)
        {
            connection.ConnectionString = $"Data Source={MyDatabaseDb3}";
            connection.Open();
            var sqliteCmd = connection.CreateCommand();
            sqliteCmd.CommandText = $"INSERT INTO {TableName} ({BodyColoumn}) VALUES (@value);";
            sqliteCmd.Parameters.Add(new SQLiteParameter("@value", body));
            sqliteCmd.ExecuteNonQuery();
        }
    }
}
```

At this stage, the receiver is safe from SQL-injections as it uses Parameters that treats input only as values. If someone generates a body: `This is an attack'); DELETE FROM bodies where 1=1')` this will simply input `This is an attack'); DELETE FROM bodies where 1=1');` Into the database.

Note 4.1. Azure Service Bus Storage Capacity

Azure Service Bus has a 256kb limit on Message Size. This can be a problem as there might be much more data from the patient.

The Receiver stores the data unencrypted, meaning whomever to have access to the server the Receiver runs in, has access to the data.

4.4 IdentityServer 4 + API

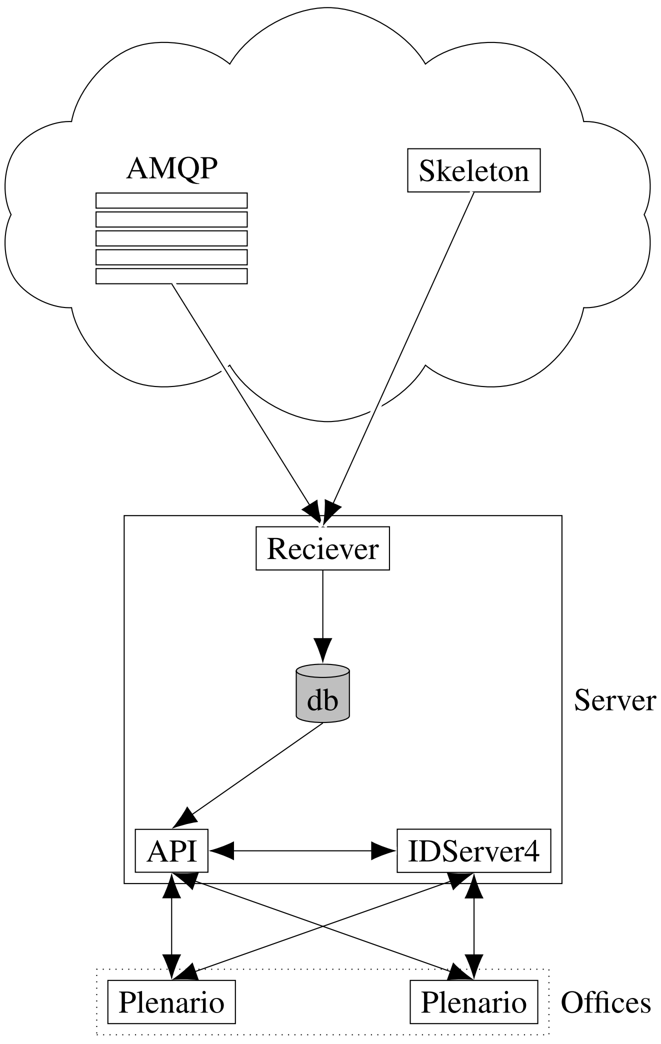
This implementation will contain an API server that uses IdentityServer 4. IdentityServer 4 uses OpenIDConnect (chapter 3.4) and Oauth 2.0 (chapter 3.3).
See figure 4.5
This solution is arguably more advanced than it needs to be, but it allows for decentralized authentication. Meaning that it is simple to implement other authorizing services such as BankID. Identityserver 4 is also used by Norwegian Health Network. [56] The Implementation keeps the secret hard coded in the software
`var tokenClient = new TokenClient(disco.TokenEndpoint, "client", "600FA056C06...");`
This is, according to 'OAuth 2.0 Threat Model and Security Considerations' (RFC6819), a threat.

> "A secret, burned into the source code of the application or an associated resource bundle, cannot be protected from reverse engineering. Secondly, such secrets cannot be revoked, since this would immediately put all installations out of work."[57, Chapter 5.3.1]

A solution to this could be to use 'Azure Key Vault,' a cloud base service, to keep this key safe, or have the Server (see fig 4.5) operate from the cloud. Another solution could be to use the computer's certificate. This is easy to implement in IdentityServer4 according to its documentation.

No data is sent encrypted, so it would be possible for an eavesdropper to peek at the data being sent, both health data from the API, but also the Access token from IDServer. A solution would be to use Secure Sockets Layer (SSL) or Transport Layer Security (TLS). TLS will be used since this is a requirement by OAuth2.0 “Access token credentials MUST only be transmitted using TLS as described in Section 1.6 with server authentication as defined by [RFC2818].”[34]

Figure 4.5 Architecture



Chapter 5

Evaluation of Security Testing Methods

This chapter will evaluate security testing methods and techniques. It will also evaluate different risk categorizations like Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, Elevation of Privilege (STRIDE) and Damage potential, Reproducibility, Exploitability, Affected users, Discoverability (DREAD).

Multiple techniques exist for testing security, each with their advantages and disadvantages. It is therefore essential to do this evaluation since Security testing can be done in so many different ways and there are no clear guidelines on what methods to be used.

5.1 Requirements

Requirement have been specified in collaboration with Infodoc.

- The system must not allow unauthorized reading from the database.
- The transmission of data must not allow others to eavesdrop.
- In ”Plenario addition“, no one other than the privileged doctors can get authorization to view its patients.
- The application should withstand harmful data and attacks coming from the Advanced Message Queuing Protocol (AMQP) server, Helsenorge.no.

The four requirements describe what the system should *not* do. Such requirements are often called negative requirements. The problem with negative requirements is that it is harder to find out how the system should not behave rather than how it should. Testing needs, therefore, to be driven by risk analysis and threat modeling [58].

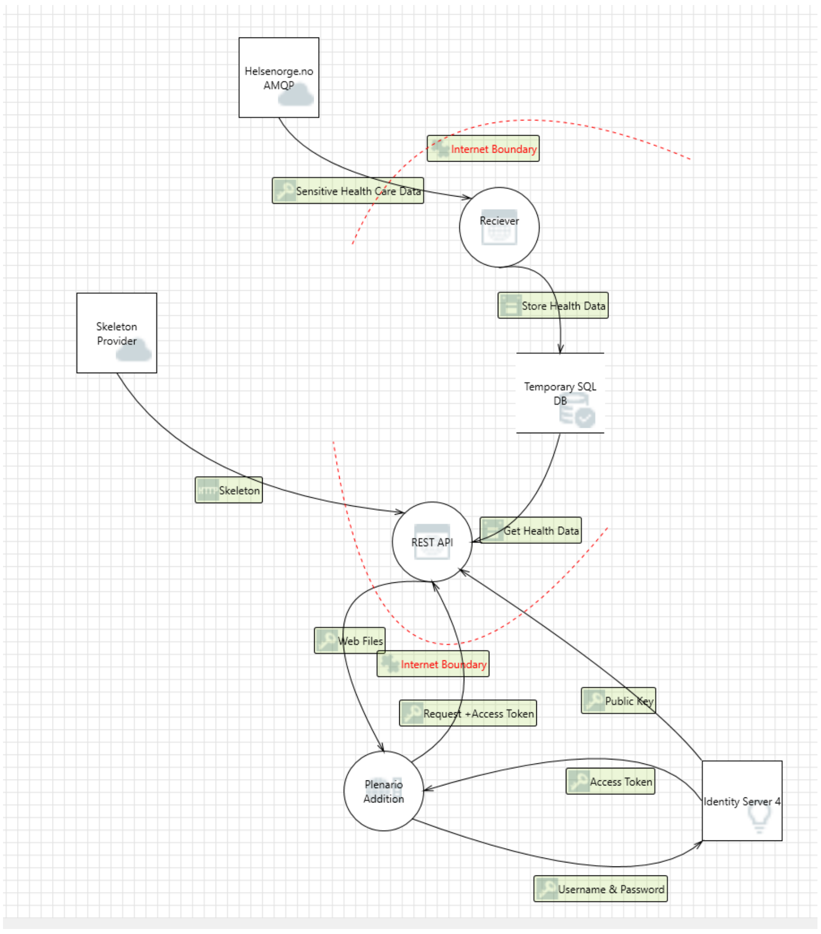


Figure 5.1 Data Flow Diagram

5.1.1 Out of scope

In figure 5.1 only the Reciever, database, REST API, Plenario Addition and Identity Server 4 will be tested. The testing tools, as well as core Plenario functions, such as long-term data storage, will not be tested. This is code is not part of the system developed in connection with this thesis.

- Saving the data securely after viewing it.
  - Infodoc already has a system for saving sensitive data.
- Test tools:
  - Azure Service Bus, acting as a substitution of helsenorge.no.
  - Fetching test-data.
  - Sending test-data to Azure Service Bus.

5.2 Penetration testing

Penetration testing tests network applications without any notion of how the application works. This is why Penetration Testing is also referred to as black-box testing or ethical hacking. [58]

The tester acts like an attacker and tries to penetrate the system. This form for testing only tests the attack surface and is thus quick and cheap. This form is also used after the application has been fully developed. Disadvantages are that this test does not reveal the source of the problem and there are also problems that can be hard to find. Penetration testing could also be used for already known vulnerabilities to test if the vulnerability is mitigated or not. [58]

5.3 Threat modeling

As described in OWASP's code review guide[59], thread modeling is defined as:

> "Threat modeling is an in-depth approach for analyzing the security of an application. It is a structured approach that enables employees to identify, quantify, and address the security risks associated with an application. Threat modeling is not an approach to reviewing code, but it complements the secure code review process by providing context and risk analysis of the application."

There exists several approaches for doing Threat Modeling: Microsoft's SDL, PASTA, VAST, trike, OCTAVE [60, 61]. The thread modeling techniques all use a risk quantifier scheme.

OWASP Code Review Guide decompose threat modeling into three general steps:

1. Application Decomposition and Analysis (ADA).
2. Determine and rank threats.
3. Determine countermeasures and mitigation.

While Microsoft threat modeling with STRIDE being the most widely accepted threat modeling process, it is important not to choose Microsoft STRIDE only because it is the most popular.[62]
A study by Riccardo Scandariato et al. [63] showed that Threat modeling with Stride overlooked 64–69% of the threats.

Table 5.1 DREAD classification

Damage potential:	How great is the damage if the vulnerability is exploited?
Reproducibility:	How easy is it to reproduce the attack?
Exploitability:	How easy is it to launch an attack?
Affected users:	As a rough percentage, how many users are affected?
Discoverability:	How easy is it to find the vulnerability?

Table 5.2 DREAD Example

Threat	D	R	E	A	D	Total	Rating
Attacker obtains authentication credentials by monitoring the network.	3	3	2	2	2	12	High
SQL commands injected into application.	3	3	3	3	2	14	High

5.3.1 Threat and risk categorizations

Microsoft developed STRIDE. It focuses on what the attacker tries to achieve and consists of the following categories. Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service and Elevation of Privilege.

DREAD is also developed by Microsoft and is a way of quantativly rate Risk. Table 5.1 shows the full form of the DREAD abbreviation along with its belonging question. Each Threat is given a DREAD score by answering each of the DREAD questions with a subscore of 1-3. All the subscores are summed up to a total. A total score of 12–15 is classified as High risk, 8–11 as Medium risk, and 5–7 as Low risk. An example can be seen in table 5.2.

David Labland, one of the creators of STRIDE and DREAD, criticized STRIDE and DREAD in a blog post. Here Labland pointed out that STRIDE and DREAD were not ‘developed with any real academic rigor.’ He also pointed out that it can sometimes be hard to classify a threat with STRIDE as the classes overlap. E.g. “ escalation of privilege (E) tends to imply spoofing and loss of non-repudiation and could imply tampering, information disclosure, and denial of service” [64]

Later in his post, he also points out that it is hard to give a score of severity from 1-10, and suggest a way of translating a score from low-medium-high to 1-10.

5.3.2 The Code of Conduct for information security - Risk Assessment

The Norwegian Directorate of eHealth (NDE) has made a “Code of Conduct for information security in the healthcare and care services called ‘Normen.’ This ‘Code of Conduct’ contains a fact sheet (No.7) explaining how to do risk assessment. Its formula is Probability times Consequence. The Assessment is made to comply with Norwegian laws and regulations.

Before calculating risk, it is required to set a level of risk, probability, and acceptable risk. The 'Normen´ contains examples of how these levels could be set.

The risk is calculated by multiplying Probability and Consequence. Probability is given a value from 1-4 depending on how frequent the risk can occur (see table 5.3). Consequence is also given a score from 1-4 based on outcome examples. Table 5.4 is one of many examples given in 'Normen'.[26]

Note 5.1. Example

The levels for probability, consequence and the different risk levels are just examples given by 'Normen´ and need to be reviewed and set for each use-case.

Table 5.3 Probability

Value	Probability	Frequency	Example
1	Unlikely	once each 5th year or less	Safety measurements can only be broken by intern employers with good resources and knowledge of the system.
2	Less likely	Once each year	Safety measurements can be broken by externals with good resources and knowledge of the system.
3	Possible	Once each month	Safety measures does is not fully enabled, or is not working according to its purpose.
4	Likely	Daily or more often	Safety measures can be broken without knowledge of the system.

Table 5.4 Consequence

Value	Consequence	Example
1	Insignificant	
2	Moderate	Privacy breach on a few number of people
3	Serious	Privacy breach on a numerous people
4	Critical	Full access to EHR system.

Table 5.5 Risk matrix

	Consequence			
Probability	1 Insignificant	2 Moderate	3 Serious	4 Critical
1 Unlikely	Low	Low	Medium	Medium
2 Less likely	Low	Low	Medium	Medium
3 Possible	Medium	Medium	High	High
4 Likely	Medium	Medium	High	High

5.4 Comparison of Threat Modeling methodologies

5.4.1 Microsoft's SDL

Microsoft Security Development Lifecycle (SDL) contains a method of doing Threat Modeling. This is a well documented and very much used technique for Threat Modeling. It uses STRIDE which is also developed by Microsoft. STRIDE is an acronym for different types of threats, Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, Elevation of Privileges.

Microsoft has developed a tool for carrying out the threat modeling, 'Microsoft Threat Modeling Tool.'

5.4.2 Trike

Trike is an Open Source threat methodology and tool and comes in 3 versions. Version 1.0, 1.5 and 2.0.

Version 1.0 is documented by a white paper and uses a stand-alone tool for threat modeling that is unmaintained. Version 1.5 is only partially documented, and it uses an Excel

Table 5.6 Overview of Threat Modeling methodologies techniques

Microsoft's SDL	
ADA	DFD
Risk analysis	STRIDE and DREAD
Suggested tools	Microsoft Threat Modeling Tool
Trike	
ADA	DFD and use flows
Risk analysis	Own categorization models
Suggested tools	Excel Spreadsheet templates
OCTAVE Allegro	
ADA	Information Asset Profile
Risk analysis	Threat trees and risk calculation worksheets
Suggested tools	PDF guide, worksheets and questioneers
OWASP Review Guide	
ADA	DFD
Risk analysis	STRIDE or Web Application Security Frame and DREAD
Suggested tools	None, Microsoft Threat Modeling Tool recommended
PASTA	
ADA	DFD use and abuse cases
Risk analysis	FFIEC, DREAD, STRIDE, MITRE-STIX, OWASP
Suggested tools	None

Spreadsheet as a tool. Version 2.0 is not documented at all and is a subset of version 1.5 [65, 66]

Trike is not maintained and no documentation of successful implementations where found.

5.4.3 Operationally Critical Threat, Asset, and Vulnerability Evaluation (OCTAVE) Allegro

There are different versions of OCTAVE, and the first was developed in 1999 by The Software Engineering Institute (SEI) at Carnegie Mellon University. The Octave Allegro was published in 2007. OCTAVE was developed with smaller developer teams in the health care sector in mind. [67]

The OCTAVE Allegro method consists of 8 steps further divided into activities. OCTAVE Allegro uses 'Threat Trees,' a way to categorize and identify threats. The method introduces the concept of *relative risk score* to evaluate the risk, combining the organization own prioritization to make it easier to compare risks. OCTAVE Allegro is designed to do risk-assessment with limited time, people and other resources. [67]

5.4.4 The Open Web Application Security Project (OWASP) Code Review Guide

A team consisting of volunteers write OWASP Code Review Guide. Secure Code Review helps finding threats and also the exact root cause. This Guide also contains a chapter about Application Threat Modeling. The Guide is very flexible, and it is possible for ad hoc applications to use it as well as big organizations. STRIDE or Application Security Frame (ASF) may be used as threat categorization.

In addition to the Guide, OWASP has created an open source tool for threat modeling called Threat Dragon in Pre-release.

5.4.5 Process for Attack Simulation and Threat Analysis (PASTA)$^{\text{TM}}$

Marco Morana and Tony Ucedavelez introduced PASTA in the book Risk Centric Threat Modeling in June 2015. Its goal was to introduce a broader threat model [68].

In addition to drawing a DFD, PASTA's step 6 use different techniques to model the attack. Attack Trees, Attack Vectors and Attack Paths. PASTA focuses on covering most security issues. This comes with a price. With its seven stages and 29 activities, it becomes very time-consuming to do threat modeling with PASTA.

5.5 Test Methodology Evaluation

Security testing should be done with more than black boxed penetration techniques and security tools. [69].

> "By using a risk-based approach to software security testing, testing professionals can help solve security problems while software is still in production"[69]

Methods that looks at the source code while doing secure threat modeling has the advantage of knowing where the threats are caused, thus making mitigation easier. It also makes it possible to do security testing while the software is in development and catch design problems.

To assess the security of the application, both Threat modeling and Manual Inspection will be used. Threat modeling will be used to Application Decomposition and Analysis, investigate threats and analyze risk.

Threat modeling was chosen since it can be done while the software is being developed, it can scale and be used in smaller applications, such as the prototype developed in conjunction with this thesis.

PASTA will not be used since it is too time-consuming and seems better for more complex projects. Trike lacks documentation and will therefore not be chosen.

Microsoft SDL will be used for this since it is the methodology that fits this size of application best. It is well documented, and Microsoft has also made Microsoft Threat Modeling Tool to aid the modeling. Microsoft SDL was chosen over OWASP Review guide since it has slightly better documentation.

To rank the risks, NDE's Code of Conduct for information security will be used since this complies with Norwegian laws and regulations and is developed with sensitive health data in mind.

5.6 Test environment

The Infodoc's FullFlow integration is only a prototype. It is therefore important to note that the following is expected to change if the integration should be implemented:

- AMQP queue is not from helsenorge.no, but from Microsoft Azure Service Bus. It is unclear what connection parameters helsenorge.no will use.
- helsenorge.no is on Norsk Helsenett (NHN)
- The IdentityServer is hosted from the Server. This might change.
- The API + Reciever + DB might be moved to the cloud.
- Two-factor authentication will be added.

Chapter 6

Threat Modeling

This chapter will explain how threat modeling was carried out. First assets were identified in order to make a Data Flow Diagram (DFD). This overview aided to identify the threats which were categorized with Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, Elevation of Privilege (STRIDE).

6.1 Create an architecture overview.

In order to create an architecture overview, assets needs to be identified see table 6.1). The main asset is the health data. Metadata can also be sensitive and confidential. Norwegian ID numbers are confidential in itself, but if used to gather health data about a specific condition, then an attacker can link persons to that condition.

A simple DFD was made with Microsoft Threat Modeling Tool (see figure 6.1 and table 6.1). The Processes and Data sources were first laid out, and the Data flow was connected and specified together with trust boundaries. Microsoft Threat Modeling Tool offers templates for both processes, data flows and data sources. A typical flow template could be HTTP with or without encryption.

Assets	Data sources	Processes
Health data	Skeleton Provider	REST API
Metadata	Helsenorge.no	Identity Server 4
Doctor Credentials	Temporary SQL Database	Plenario addition

Table 6.1 Assets, Data sources and Processes

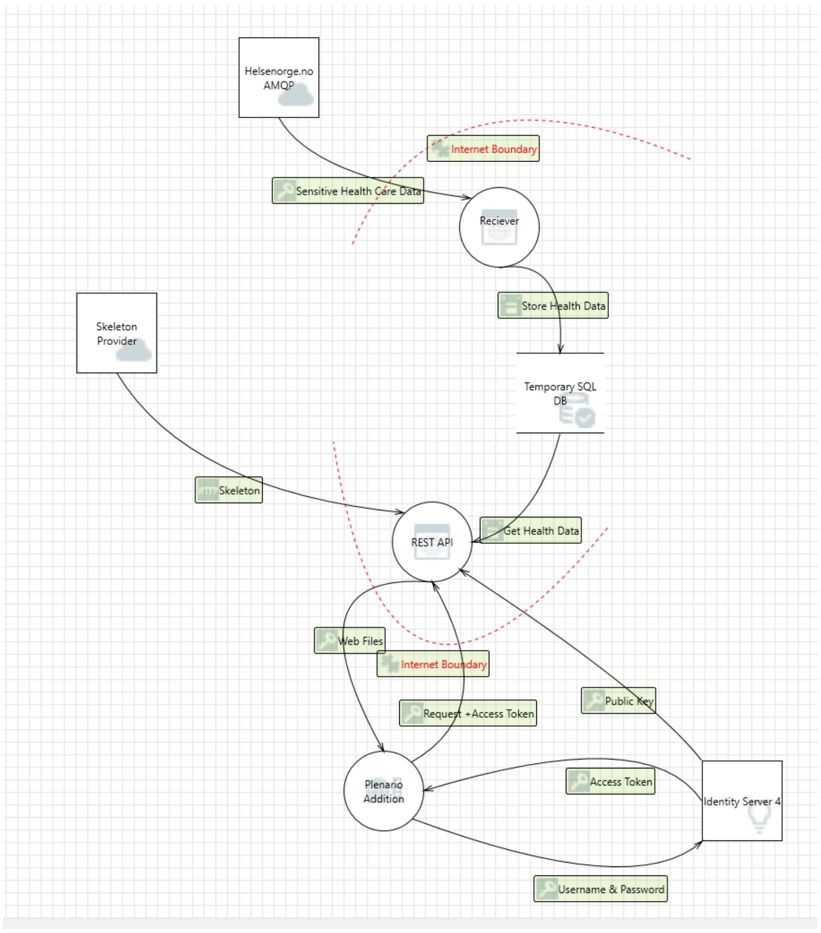


Figure 6.1 Data Flow Diagram

6.2 Identify the threats.

With the data flow diagram, the Microsoft Threat Modeling tool created a document with suggested threats (See appendix A). These threats were reviewed and taken into consideration for our list of threats. The threats will be categorized after the STRIDE model.

- Spoofing
  - Someone can spoof the Representational State Transfer (REST) Application Programming Interface (API), making General Practitioner (GP)'s get data from a false source.
  - Helsenorge.no can be spoofed, wich the reciever would store unwanted data to the Temporary SQL Server.
  - Identity Server can be spoofed.

- Tampering
  - Someone can tamper with the Data in transit
- Repudiation
  - No processes can claim that something happened which didn't, since the application lacks logging.
- Information disclosure
  - None of the data flowing is in jeopardy of being sniffed, since all sensitive data are sent with Transport Layer Security (TLS) encryption.
- Denial of Service
  - All processes connected to the Internet is prone to Denial Of Service (DoS) attacks.
- Elevation of Privilege
  - An attacker might be able to gain unauthorized access to the REST API by forging an Acces token

If harmful javascript could be executed in the Plenario Addition, leading to Information disclosure.

	Consequence			
Probability	1 Insignificant	2 Moderate	3 Serious	4 Critical
1 Unlikely	Low	Low	Medium	Medium
2 Less likely	Low	Low	Medium	Medium
3 Possible	Medium	Medium	High	High
4 Likely	Medium	Medium	High	High

Table 6.2 Risk matrix

Threat	Probability	Consequence	Risk Level(PxC)
Spoofing REST API.	1	3	Medium
Use the same token from a different client.	2	3	Medium
Steal Private key from the Identity Server	1	4	Low
DDOS Attack	2	2	Low

Table 6.3 Risk rating

Chapter 7

Penetration Testing

This chapter will list the threats that needed further investigation.

We tested the following attacks:

- Use the same token from a different client.
- Generate own token.
- Steal Private key from the Identity Server.
- Accessing data from another user.
- Known attack altering JSON Web Tokenss (JWTs).

The testing environment was set up with two computers. One running the Server side and the other running the client side of Infodoc Plenario. Postman was used to craft forged tokens, sending copies of tokens and more.

7.1 Use the same token from different client

A penetration test will be carried out with the premise that an access token is already stolen.

Using Postman to access the API with a copied token from a Client it is possible to Have the same access to the API as the user as long as the token lifetime. If trying to alter the expiration time, the signature becomes invalid.
With a stolen access token an attacker can efficiently get all the data from every patient connected to that doctor. To mitigate this problem, include JWT id [70] to make a token only possible to use a token once. In case a token gets stolen, at most one patient can be accessed from the attacker given that the access token from the doctor was also blocked.

7.2 Accessing data from another user

If Doctor A and Doctor B use the same machine, it is possible to get health data. Files are automatically erased when a user closes the program (it is deleted in the CEF deconstructor), but if a program is crashed, the files will persist.

To mitigate this, encrypt the files with Windows Data Protection Application Programming Interface (DPAPI) that uses the user login credentials. This will make it impossible for another doctor to look at the content of those files.

7.3 None - Algorithm in JWS

A JSON Web Signatures (JWS) contains three fields, JSON Object Signing and Encryption (JOSE) Header, JWS Payload and JWS Signature. The header *must* contain “Algorithm (alg)” Header parameter. This parameter tells what cryptographic algorithm is used to sign the JWS and *must* be understood and must be a JSON Web Algorithm (JWA).[71]
One of the algorithms in JWA is ‘none’, such are called an ’Unsecured JWS‘.[72].

An attacker could easily change the ‘alg’ parameter to ‘None’ and delete the signature.

It is stated in the RFC7518 standard that ‘Implementations MUST NOT accept Unsecured JWSs by default’[72]. This is also the case when it comes to IdentityServer4. So this attack is not possible.

7.4 Algorithm execution attacks

A JWT can be encoded using either Rivest–Shamir–Adleman (RSA) or Hash-based Message Authentication Code (HMAC)256 algorithm to sign the token [70]. Example where an authentication server signs a token with RSA

Listing 7.1 Creation and signing of a JWT

```
claim = {'client_id': 'client', 'scope': ['api1']}
token = jwt.encode(claim, privatekey, algorithm='RS256')
```

Listing 7.2 Secure Verification of a JWT

```
decoded = jwt.decode(token, public_key, algorithms = ['RS256','HS256'])
```

If an algorithm is not specified an attacket could encode it with HMAC but use the publicly known key from the authentication server.

Listing 7.3 Creation and signing of an forged JWT

```
public_RSA_key = get_public_rsa_key_from_auth_server(auth_addr)
attacker_claim = {'client_id': 'client', 'scope': ['api1','admin']}
token = jwt.encode(attacker_claim, public_RSA_key, algorithm='HS265')
```

The Application Programming Interface (API) will validate this with the assumption that HMAC has been used and checks it against the publicly known key which will be valid.

The problem can be solved by agreeing on and using only one algorithm and do not trust the `'alg'` field since this can also be altered by an attacker.

Listing 7.4 Secure Verification of a JWT

```
decoded = jwt.decode(token, public_key, algorithms = ['RS256'])
```

From the attacker's point of view:

- Generate a token with a payload granting access.
- Sign the token with using the HMAC256 algorithm and as a secret use the public RSA key from the authorization server.
- The API will validate the token with the public key from the authorization server.
- Since the public key is the same as the secret used to generate the HMAC256 algorithm it validates to true.

7.5 JavaScript attacks

Hostile JavaScript is inherently hard to detect[73, 74]. Assume that the attacker has infiltrated the FullFlow server, helsenorge.no or does a man in the middle attack. The attacker could then implement arbitrary JavaScript code into the HTML files. From the attackers perspective, detection of malicious JavaScript code by static analysis, can be avoided by obfuscating code.

Listing 7.5 Obfuscated code. [75]

```
<script>eval ( unescape(%64%6F%63%75%6D%65%6E%74%2E %77%72%69...
... [bytes skipped]} %3C%2F%69%66%72%61%6D%65%3E%27%29 ') ) ; </ script>
```

Listing 7.6 Deobfuscated code. [75]

```
document . write ( <iframe src="http://sedpoo.com/?338375" width=0 height=0>
</ iframe> ') ;
```

In addition to obfuscating JavaScript code, an attacker could generate JavaScript code dynamically at run-time, making it even harder to detect. [75]

The application was tested with several malicious JavaScript code. Keylogger and sending health care data was some of them. These attacks were first successful. To mitigate this, HTTP, FTP, and file protocol were disabled, making it impossible to communicate or gain information from the application. Reading and Writing to the filesystem was impossible to do since the implementation of Chromium Embedded Framework (CEF) made it impossible to read or write files.

Chapter 8

Conclusion

The goal of this thesis was to assess the risk of receiving sensitive health care data in Infodoc Plenario. A prototype with the following architecture was made, see figure 4.5.

The technologies that were used in the prototype was studied and evaluated against alternatives.

The methodology of how to best evaluate the security of the prototype was also investigated. There exist different ways of assessing security, but the method chosen was Threat modeling. Different frameworks were researched, but Microsoft's method of doing Threat modeling was best suited for this thesis.

The risk scoring example from 'Normen' was used to score the risks.

Finally, the prototype was assessed, and penetration tested in chapter 7 Penetration testing. The threats were either not exploitable or mitigated.

It is vital for the FullFlow project that it is possible to implement their data securely. This work can be used as an example of how to implement FullFlow and as documentation on the security of such an implementation.

According to Infodoc, earlier penetration tests on Plenario showed that having the server installed on-site was a high risk. Infodoc is making Plenario a Software as a Service. This means that the software will run in the cloud. A new risk is that everyone can attempt to authorize themselves. Infodoc is implementing IdentityServer 4 in order to authorize users securely. IdentityServer makes it easy to customize the authorization flow. Two-factor authentication such as BankID can be made mandatory, thus satisfying the highest degree of authentication in the 'Code of Conduct.' [26]

IdentityServer4 makes it easy for .NET applications to implement OAuth and OpenID protocols. IdentityServer makes it easy to implement authentication via an external OpenID identity providers. Big software corporation such as Google, Microsoft, and Amazon provide OpenID as well as Norwegian providers such as Norsk Helsenett and BankID.

The risk assessment and penetration testing done in the FullFlow integration can hopefully be an indication of whether or not IdentityServer 4 is safe to use in Infodoc Plenario.

8.1 Discussion of the Risk evaluation methods used

Threat modeling is a structured way of finding threats by first getting the overview over how the application work and where different threads may appear and then finding threats. It can be hard to cover all the threats, and many threats are also false positives. It is therefore beneficial to combine threat modeling with penetration testing since penetration testing can get rid of false positives as well as finding other threats not covered by threat modeling.

The ‘Code of conduct’ example of risk evaluation was to quantify probability and severity to a number on the scale 1-4 and then multiply them together. On the one hand makes it very quick to score a threat, but on the other hand, it can lead to oversimplification and ultimately give risks different scores. It would be interesting to develop a more finely grained risk assessment model for health care with more risk factors. A significant threat category in health care is losing patient data. The vulnerability could be calculated from how much data is disclosed, how many patients were affected, how sensitive the information is and to whom the data will be available too.

8.2 Further work

The work done in connection with this thesis was assessed mostly by itself. It would be interesting to do a complete threat model of whole Infodoc’s Plenario together with the Full Flow addition. A threat model should be made with ‘Infodoc SKY,’ the new cloud-based Platform for Infodoc Plenario.

In order to calculate the risk more in detail, all threats that were found in chapter 6 needs to be penetration tested.

The threat modeling was done with Microsoft’s Security Development Lifecycle (SDL) but could also be done with Operationally Critical Threat, Asset, and Vulnerability Evaluation (OCTAVE) Allegro. The two frameworks could be compared, both in time and effort and how many threats found and how many false positives they each generated.

The risk assessment and penetration testing were done on a prototype, and this only indicates the security status. Security is an ongoing process, and the FullFlow integration’s risk assessment needs to be updated continuously.

References

[1] N. Sakimura, J. Bradley, M. Jones, B. de Medeiros, and C. Mortimore, “Openid connect core 1.0 incorporating errata set 1,” *The OpenID Foundation, specification*, 2014.

[2] NIST, “Nvd - home,” https://nvd.nist.gov/, 2017, (Accessed on 09/20/2017).

[3] “Microsoft security intelligence report,” Microsoft Cooperation, Tech. Rep., 2013.

[4] “Microsoft security intelligence report,” Microsoft Cooperation, Tech. Rep., 2014.

[5] “Microsoft security intelligence report,” Microsoft Cooperation, Tech. Rep., 2015.

[6] “Microsoft security intelligence report,” Microsoft Cooperation, Tech. Rep., 2015.

[7] “Microsoft security intelligence report,” Microsoft Cooperation, Tech. Rep., 2016.

[8] “Microsoft security intelligence report,” Microsoft Cooperation, Tech. Rep., 2016.

[9] Datatilsynet, “Hva gjør dere med dataene mine? - en kartlegging av hjemmetester innen helse,” https://www.datatilsynet.no/globalassets/global/om-personvern/rapporter/gpen-sveip-rapport-2016.pdf, Datatilsynet, Tech. Rep., September 2016.

[10] A. Barth, C. Jackson, C. Reis, T. Team *et al.*, “The security architecture of the chromium browser,” *Technical report*, 2008.

[11] A. L. BANKS, L. MCGRATH, and S. N. PATEL, “Pupilscreen: Using smartphones to assess traumatic brain injury,” 2017.

[12] T. Maier, D. Kulichova, K. Schotten, R. Astrid, T. Ruzicka, C. Berking, and A. Udrea, “Accuracy of a smartphone application using fractal image analysis of pigmented moles compared to clinical diagnosis and histological result,” *Journal of the European Academy of Dermatology and Venereology*, vol. 29, no. 4, pp. 663–667, 4 2015. [Online]. Available: http:https://doi.org/10.1111/jdv.12648

[13] Sosialdepartementet, “Helseregisterloven,” https://lovdata.no/lov/2001-05-18-24/\T1\textsection27, May 2001, (Accessed on 09/28/2017).

[14] T. N. D. of eHealth, “Kjernejournal for safer healthcare - helsenorge.no,” https://helsenorge.no/kjernejournal/kjernejournal-for-safer-healthcare, 5 2017, (Accessed on 01/04/2018).

[15] T. Christensen, “Bringing the gp to the forefront of ehr development,” 2009.

[16] ——, “Allmennlegene og elektronisk pasientjournal 1988-2013,” http://legeforeningen.no/yf/Allmennlegeforeningen/Publikasjoner/Festskrift-til-Almmenlegeforeningens-75-ars-jubileum/Festskrift-til-Allmennlegeforeningens-75-arsjubileum/E-laring/Allmennlegene-og-elektronisk-pasientjournal-1988-2013/, August 2013, (Accessed on 08/26/2018).

[17] “English,” https://ehelse.no/english, 6 2017, (Accessed on 11/07/2017).

[18] “Om helsenorge.no - helsenorge.no,” https://helsenorge.no/om-helsenorge-no, 2 2017, (Accessed on 11/07/2017).

[19] N. C. for Integrated Care and Telemedicine, “Proposal of project, full flow of health data between patients and health care system,” 2016, (Accessed on 08/23/2017).

[20] N. C. for E-health Research, “Full flow of health data between patients and health care systems - ehealthresearch.no,” https://ehealthresearch.no/en/projects/fullflow, 11 2017, (Accessed on 12/04/2017).

[21] Samsung, “Children’s health of dallas uses remote patient monitoring,” https://insights.samsung.com/2016/02/10/childrens-health-of-dallas-gets-pediatric-transplant-patients-back-home-to-heal-with-remote-monitoring- 2015, (Accessed on 07/29/2018).

[22] Vivify, “Vivify health- remote care,” https://www.vivifyhealth.com/, 2018, (Accessed on 10/16/2018).

[23] J. Wang, D. C. Coleman, J. Kanter, B. Ummer, and L. Siminerio, “Connecting smartphone and wearable fitness tracker data with a nationally used electronic health record system for diabetes education to facilitate behavioral goal monitoring in diabetes care: Protocol for a pragmatic multi-site randomized trial,” *JMIR research protocols*, vol. 7, no. 4, 2018.

[24] Validic, “Validic delivers one connection to a world of mobile health data,” https://validic.com/, 2018, (Accessed on 10/18/2018).

[25] M. A. Weiss and K. Archick, “Us-eu data privacy: from safe harbor to privacy shield,” 2016.

[26] T. N. D. of eHealth, “Code of conduct for information security the healthcare and care services sector,” 2017.

[27] A. Shostack, “Experiences threat modeling at microsoft,” in *Modeling Security Workshop. Dept. of Computing, Lancaster University, UK*, 2008.

[28] ISO, “isoinbrief_2015.pdf,” https://www.iso.org/files/live/sites/isoorg/files/archive/pdf/en/isoinbrief_2015.pdf, 2016, (Accessed on 09/11/2017).

[29] R. T. Fielding and R. N. Taylor, *Architectural styles and the design of network-based software architectures.* University of California, Irvine Doctoral dissertation, 2000.

[30] “The history of rest apis,” https://blog.readme.io/the-history-of-rest-apis/, Novmber 2016, (Accessed on 09/08/2017).

[31] W. Santos, “Programmableweb api directory eclipses 17,000 as api economy continues surge | programmableweb,” https://www.programmableweb.com/news/programmableweb-api-directory-eclipses-17000-api-economy-continues-surge/research/2017/03/13, 03 2017, (Accessed on 09/15/2017).

[32] L. Masinter, T. Berners-Lee, and R. T. Fielding, “Uniform resource identifier (uri): Generic syntax,” 2005.

[33] E. Hammer, “Explaining oauth – hueniverse,” https://hueniverse.com/explaining-oauth-3735e3de27a8, 9 2007, (Accessed on 08/23/2017).

[34] D. Hardt, “The oauth 2.0 authorization framework,” Internet Requests for Comments, RFC Editor, RFC 6749, October 2012, http://www.rfc-editor.org/rfc/rfc6749.txt. [Online]. Available: http://www.rfc-editor.org/rfc/rfc6749.txt

[35] M. Jones, J. Bradley, and N. Sakimura, “Json web token (jwt),” Tech. Rep., 2015.

[36] J. C. Mogul, J. Gettys, T. Berners-Lee, and H. Frystyk, “Hypertext transfer protocol–http/1.1,” 1997.

[37] T. Bienz, R. Cohn, and C. Adobe Systems (Mountain View, *Portable document format reference manual*. Addison-Wesley Reading, MA, USA, 1993.

[38] S. Rautiainen, “A look at portable document format vulnerabilities,” *information security technical report*, vol. 14, no. 1, pp. 30–33, 2009.

[39] Knowledge@Wharton, “Driving adobe: Co-founder charles geschke on challenges, change and values - knowledge@wharton,” http://knowledge.wharton.upenn.edu/article/driving-adobe-co-founder-charles-geschke-on-challenges-change-and-values/, Sep 2008, (Accessed on 10/31/2017).

[40] I. O. for Standardization (ISO), “Iso 32000-1: 2008 document management–portable document format–part 1: Pdf 1.7.” 2008.

[41] K. Itabashi, “Portable document format malware,” *Symantec white paper*, 2011.

[42] “Chromium - the chromium projects,” https://www.chromium.org/Home, 2017, (Accessed on 09/19/2017).

[43] C. Reis, A. Barth, and C. Pizano, “Browser security: lessons from google chrome,” *Queue*, vol. 7, no. 5, p. 3, 2009.

[44] “docs - chromium/src - git at google,” https://chromium.googlesource.com/chromium/src/+/master/docs/, 2017, (Accessed on 09/26/2017).

[45] FIRST.org, “Cvss v3.0 specification document,” https://www.first.org/cvss/specification-document, 2017, (Accessed on 09/20/2017).

[46] J. Ruderman, “Same-origin policy - web security | mdn,” https://developer.mozilla.org/en-US/docs/Web/Security/Same-origin_policy, 8 2017, (Accessed on 09/22/2017).

[47] A. Barth, “Chromium blog: Security in depth: Local web pages,” https://blog.chromium.org/2008/12/security-in-depth-local-web-pages.html, 12 2008, (Accessed on 09/22/2017).

[48] ——, “The web origin concept,” Internet Requests for Comments, RFC Editor, RFC 6454, December 2011, http://www.rfc-editor.org/rfc/rfc6454.txt. [Online]. Available: http://www.rfc-editor.org/rfc/rfc6454.txt

[49] D. Ross and T. Gondrom, “Http header field x-frame-options,” Internet Requests for Comments, RFC Editor, RFC 7034, October 2013.

[50] MSDN, “Webbrowser class (system.windows.forms),” https://msdn.microsoft.com/en-us/library/system.windows.forms.webbrowser(v=vs.110).aspx, 2017, (Accessed on 09/19/2017).

[51] D. Kerr, “As it stands - electron security,” http://blog.scottlogic.com/2016/03/09/As-It-Stands-Electron-Security.html, March 2016, (Accessed on 04/18/2018).

[52] D. Crocker, “Standard for the format of arpa internet text messages,” Internet Requests for Comments, RFC Editor, STD 11, August 1982.

[53] N. Freed and N. S. Borenstein, “Multipurpose internet mail extensions (mime) part one: Format of internet message bodies,” Internet Requests for Comments, RFC Editor, RFC 2045, November 1996, http://www.rfc-editor.org/rfc/rfc2045.txt. [Online]. Available: http://www.rfc-editor.org/rfc/rfc2045.txt

[54] F. Yergeau, “Utf-8, a transformation format of iso 10646,” Internet Requests for Comments, RFC Editor, STD 63, November 2003, http://www.rfc-editor.org/rfc/rfc3629.txt. [Online]. Available: http://www.rfc-editor.org/rfc/rfc3629.txt

[55] S. Josefsson, “The base16, base32, and base64 data encodings,” Internet Requests for Comments, RFC Editor, RFC 4648, October 2006, http://www.rfc-editor.org/rfc/rfc4648.txt. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4648.txt

[56] nhn, “Norsk helsenett sf - autentisering,” https://www.nhn.no/helseid/veiledning-slik-gjoer-du-det/autentisering/, 2018, (Accessed on 10/14/2018).

[57] T. Lodderstedt, M. McGloin, and P. Hunt, “Oauth 2.0 threat model and security considerations,” Internet Requests for Comments, RFC Editor, RFC 6819, January 2013.

[58] J. Williams, “Owasp testing guide,” 2006.

[59] O. C. R. team, “Code review guide,” OWASP Foundation, Tech. Rep., july 2017.

[60] R. A. Grimes, “Threat modeling,” *Hacking the Hacker: Learn from the Experts Who Take Down Hackers*, pp. 211–215, 2017.

[61] OWASP, “Threat risk modeling,” https://www.owasp.org/index.php/Threat_Risk_Modeling, January 2017, (Accessed on 02/05/2018).

[62] B. Beyst, “Which threat modeling methodology is right for your organization?” https://www.peerlyst.com/posts/which-threat-modeling-methodology-is-right-for-your-organization-brian-beyst-mba, October 2016, (Accessed on 02/05/2018).

[63] R. Scandariato, K. Wuyts, and W. Joosen, “A descriptive study of microsoft’s threat modeling technique,” *Requirements Engineering*, vol. 20, no. 2, pp. 163–180, March 2015.

[64] D. LeBlanc, “Dreadful – david leblanc’s web log,” https://blogs.msdn.microsoft.com/david_leblanc/2007/08/14/dreadful/, July 2017, (Accessed on 02/09/2018).

[65] P. Saitta, B. Larcom, and M. Eddington, “Trike v1 methodology document,” *Draft, work in progress*, 2005.

[66] Trike, “Trike: Trike,” http://octotrike.org/, January 2012, (Accessed on 02/07/2018).

[67] R. Caralli, J. Stevens, L. Young, and W. Wilson, “Introducing octave allegro: Improving the information security risk assessment process,” Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU/SEI-2007-TR-012, 2007. [Online]. Available: http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=8419

[68] T. Ucedavelez. (2015, December) Process for attack simulation and threat analysis (pasta) risk centric threat models. Youtube. [Online]. Available: https://youtu.be/TcwPZKMVZu4

[69] B. Potter and G. McGraw, “Software security testing,” *IEEE Security Privacy*, vol. 2, no. 5, pp. 81–85, Sept 2004.

[70] M. Jones, J. Bradley, and N. Sakimura, “Json web token (jwt),” Internet Requests for Comments, RFC Editor, RFC 7519, May 2015, http://www.rfc-editor.org/rfc/rfc7519.txt. [Online]. Available: http://www.rfc-editor.org/rfc/rfc7519.txt

[71] ——, “Json web signature (jws),” Internet Requests for Comments, RFC Editor, RFC 7515, May 2015, http://www.rfc-editor.org/rfc/rfc7515.txt. [Online]. Available: http://www.rfc-editor.org/rfc/rfc7515.txt

[72] M. Jones, “Json web algorithms (jwa),” Internet Requests for Comments, RFC Editor, RFC 7518, May 2015.

[73] O. Hallaraker and G. Vigna, “Detecting malicious javascript code in mozilla,” in *10th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS’05)*, June 2005, pp. 85–94.

[74] S. Lekies, B. Stock, M. Wentzel, and M. Johns, “The unexpected dangers of dynamic javascript.” in *USENIX Security Symposium*, 2015, pp. 723–735.

[75] P. Seshagiri, A. Vazhayil, and P. Sriram, “Ama: Static code analysis of web page for the detection of malicious scripts,” *Procedia Computer Science*, vol. 93, pp. 768–773, 2016.

A Threat Modeling Report

Threat Modeling Report

Created on 14-Feb-18 2:44:31 PM

Threat Model Name:

Owner:

Reviewer:

Contributors:

Description:

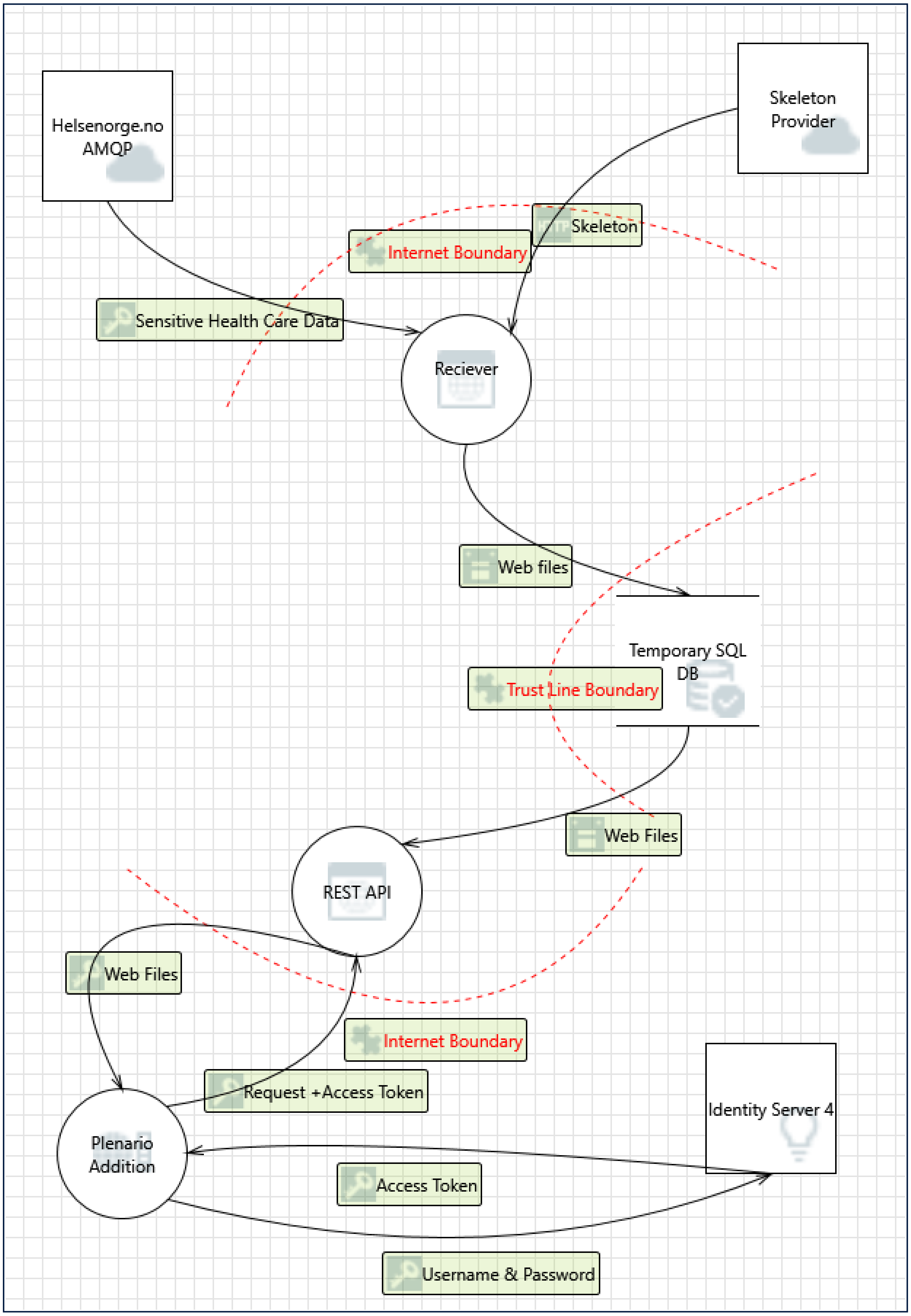
Assumptions:

External Dependencies:

Threat Model Summary:

Not Started	51
Not Applicable	0
Needs Investigation	0
Mitigation Implemented	0
Total	51
Total Migrated	0

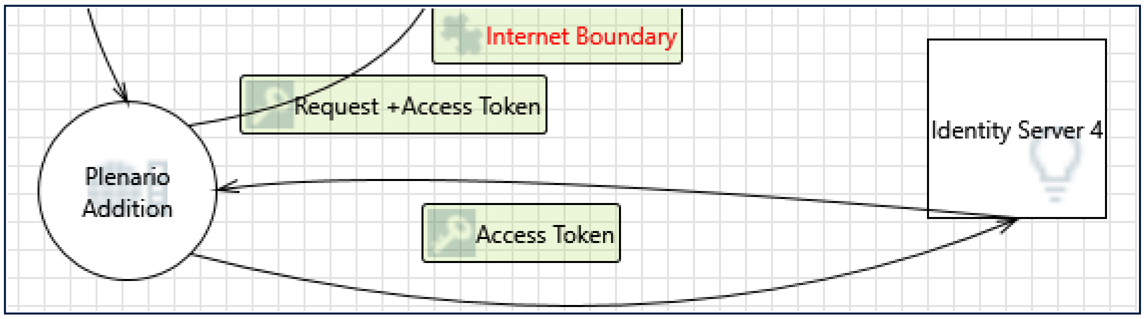
Diagram: Full Flow Integration in Plenario



Full Flow Integration in Plenario Diagram Summary:

Not Started	51
Not Applicable	0
Needs Investigation	0
Mitigation Implemented	0
Total	51
Total Migrated	0

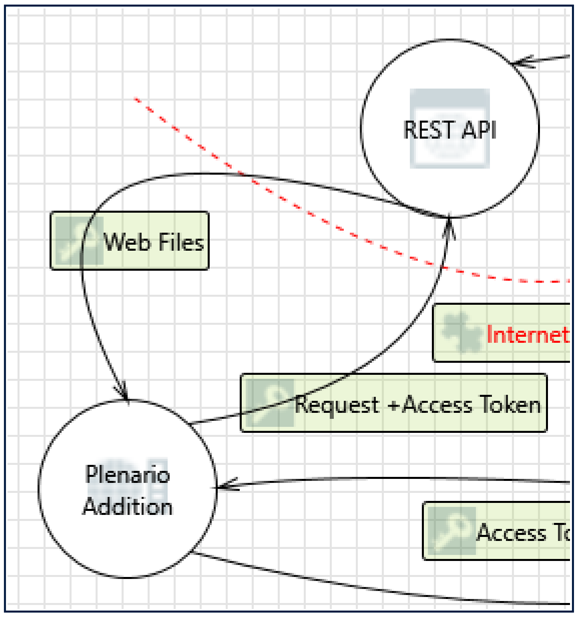
Interaction: Access Token



1. Elevation Using Impersonation [State: Not Started] [Priority: High]

Category:	Elevation Of Privilege
Description:	Plenario Addition may be able to impersonate the context of Identity Server 4 in order to gain additional privilege.
Justification:	<no mitigation provided>

Interaction: Request +Access Token



2. Elevation by Changing the Execution Flow in Web Service [State: Not Started] [Priority: High]

Category:	Elevation Of Privilege
Description:	An attacker may pass data into REST API in order to change the flow of program execution within REST API to the attacker's choosing.
Justification:	<no mitigation provided>

3. Web Service May be Subject to Elevation of Privilege Using Remote Code Execution [State: Not Started] [Priority: High]

Category:	Elevation Of Privilege
Description:	Plenario Addition may be able to remotely execute code for REST API.
Justification:	<no mitigation provided>

4. Data Flow HTTPS Is Potentially Interrupted [State: Not Started] [Priority: High]

Category:	Denial Of Service
Description:	An external agent interrupts data flowing across a trust boundary in either direction.
Justification:	<no mitigation provided>

5. Potential Process Crash or Stop for Web Service [State: Not Started] [Priority: High]

Category:	Denial Of Service
Description:	REST API crashes, halts, stops or runs slowly; in all cases violating an availability metric.
Justification:	<no mitigation provided>

6. Potential Data Repudiation by Web Service [State: Not Started] [Priority: High]

Category:	Repudiation
Description:	REST API claims that it did not receive data from a source outside the trust boundary. Consider using logging or auditing to record the source, time, and summary of the received data.
Justification:	<no mitigation provided>

7. Plenario Addition Process Memory Tampered [State: Not Started] [Priority: High]

Category:	Tampering
Description:	If Plenario Addition is given access to memory, such as shared memory or pointers, or is given the ability to control what REST API executes (for example, passing back a function pointer.), then Plenario Addition can tamper with REST API. Consider if the function could work with less access to memory, such as passing data rather than pointers. Copy in data provided, and then validate it.
Justification:	<no mitigation provided>

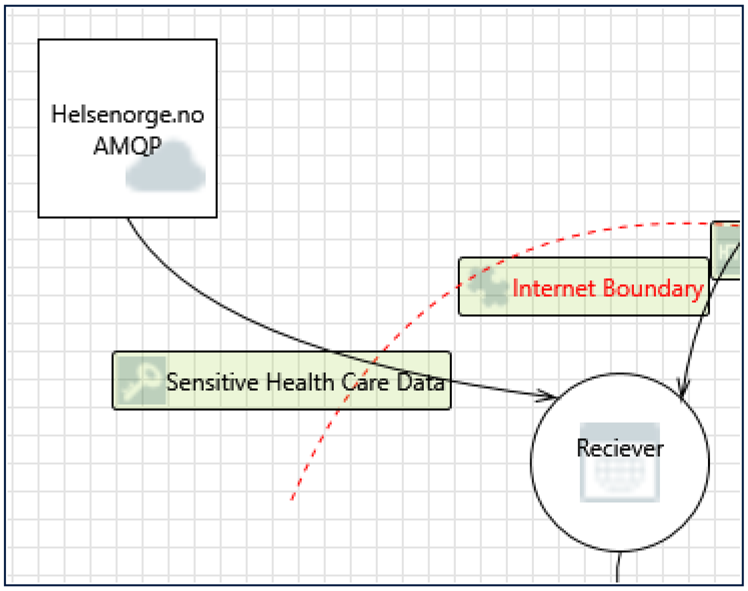
8. Elevation Using Impersonation [State: Not Started] [Priority: High]

Category: Elevation Of Privilege

Description: REST API may be able to impersonate the context of Plenario Addition in order to gain additional privilege.

Justification: <no mitigation provided>

Interaction: Sensitive Health Care Data



9. Elevation by Changing the Execution Flow in Reciever [State: Not Started] [Priority: High]

Category: Elevation Of Privilege

Description: An attacker may pass data into Reciever in order to change the flow of program execution within Reciever to the attacker's choosing.

Justification: <no mitigation provided>

10. Reciever May be Subject to Elevation of Privilege Using Remote Code Execution [State: Not Started] [Priority: High]

Category: Elevation Of Privilege

Description: Helsenorge.no AMQP may be able to remotely execute code for Reciever .

Justification: <no mitigation provided>

11. Elevation Using Impersonation [State: Not Started] [Priority: High]

Category: Elevation Of Privilege

Description: Reciever may be able to impersonate the context of Helsenorge.no AMQP in order to gain additional privilege.

Justification: <no mitigation provided>

12. Data Flow Sensitive Health Care Data Is Potentially Interrupted [State: Not Started] [Priority: High]

Category: Denial Of Service

Description: An external agent interrupts data flowing across a trust boundary in either direction.

Justification: <no mitigation provided>

13. Potential Process Crash or Stop for Reciever [State: Not Started] [Priority: High]

Category: Denial Of Service

Description: Reciever crashes, halts, stops or runs slowly; in all cases violating an availability metric.

Justification: <no mitigation provided>

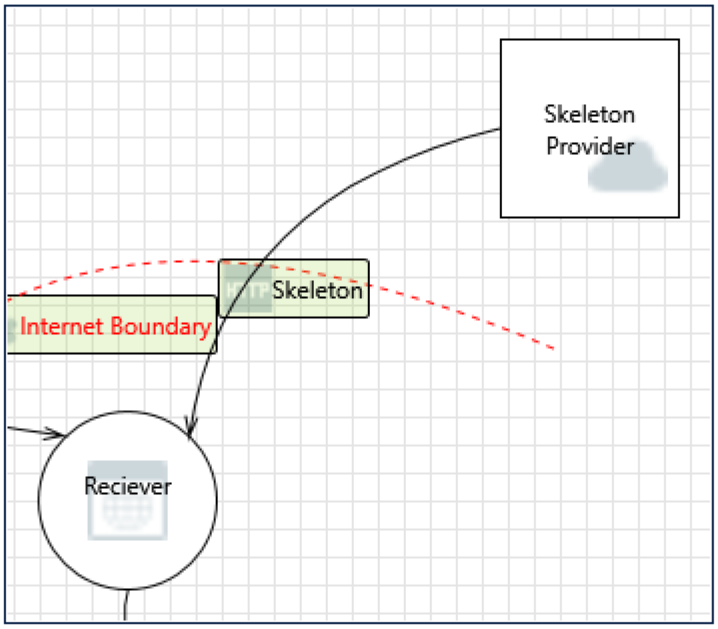
14. Potential Data Repudiation by Reciever [State: Not Started] [Priority: High]

Category: Repudiation

Description: Reciever claims that it did not receive data from a source outside the trust boundary. Consider using logging or auditing to record the source, time, and summary of the received data.

Justification: <no mitigation provided>

Interaction: Skeleton



15. Spoofing the Reciever Process [State: Not Started] [Priority: High]

Category: Spoofing

Description: Reciever may be spoofed by an attacker and this may lead to information disclosure by Skeleton Provider. Consider using a standard authentication mechanism to identify the destination process.

Justification: <no mitigation provided>

16. Spoofing the Megaservice External Entity [State: Not Started] [Priority: High]

Category: Spoofing

Description: Skeleton Provider may be spoofed by an attacker and this may lead to unauthorized access to Reciever . Consider using a standard authentication mechanism to identify the external entity.

Justification: <no mitigation provided>

17. Potential Lack of Input Validation for Reciever [State: Not Started] [Priority: High]

Category: Tampering

Description: Data flowing across Skeleton may be tampered with by an attacker. This may lead to a denial of service attack against Reciever or an elevation of privilege attack against Reciever or an information disclosure by Reciever . Failure to verify that input is as expected is a root cause of a very large number of exploitable issues. Consider all paths and the way they handle data. Verify that all input is verified for correctness using an approved list input validation approach.

Justification: <no mitigation provided>

18. Potential Data Repudiation by Reciever [State: Not Started] [Priority: High]

Category: Repudiation

Description: Reciever claims that it did not receive data from a source outside the trust boundary. Consider using logging or auditing to record the source, time, and summary of the received data.

Justification: <no mitigation provided>

19. Data Flow Sniffing [State: Not Started] [Priority: Low]

Category: Information Disclosure

Description: Data flowing across Skeleton may be sniffed by an attacker. Depending on what type of data an attacker can read, it may be used to attack other parts of the system or simply be a disclosure of information leading to compliance violations. Consider encrypting the data flow.

Justification: Skeleton is not confidential data.

20. Potential Process Crash or Stop for Reciever [State: Not Started] [Priority: Low]

Category: Denial Of Service

Description: Reciever crashes, halts, stops or runs slowly; in all cases violating an availability metric.

Justification: Can only access data from Helsenorge.no and Skeleton Provider.

21. Data Flow Skeleton Is Potentially Interrupted [State: Not Started] [Priority: High]

Category: Denial Of Service
Description: An external agent interrupts data flowing across a trust boundary in either direction.
Justification: <no mitigation provided>

22. Elevation Using Impersonation [State: Not Started] [Priority: High]

Category: Elevation Of Privilege
Description: Reciever may be able to impersonate the context of Skeleton Provider in order to gain additional privilege.
Justification: <no mitigation provided>

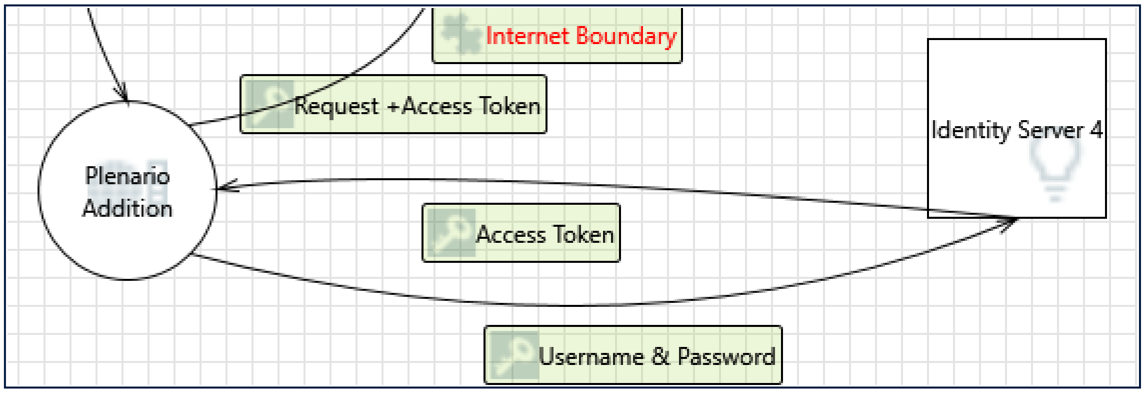
23. Reciever May be Subject to Elevation of Privilege Using Remote Code Execution [State: Not Started] [Priority: High]

Category: Elevation Of Privilege
Description: Skeleton Provider may be able to remotely execute code for Reciever .
Justification: <no mitigation provided>

24. Elevation by Changing the Execution Flow in Reciever [State: Not Started] [Priority: High]

Category: Elevation Of Privilege
Description: An attacker may pass data into Reciever in order to change the flow of program execution within Reciever to the attacker's choosing.
Justification: <no mitigation provided>

Interaction: Username & Password



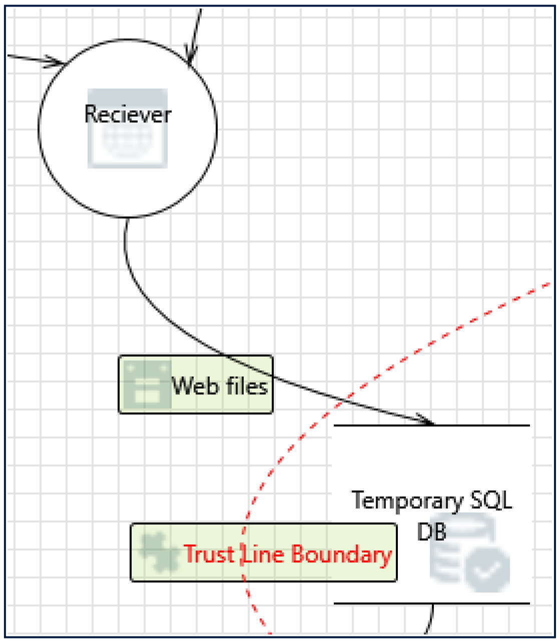
25. Weakness in SSO Authorization [State: Not Started] [Priority: High]

Category: Elevation Of Privilege

Description: Common SSO implementations such as OAUTH2 and OAUTH Wrap are vulnerable to MitM attacks.

Justification: <no mitigation provided>

Interaction: Web files



26. Spoofing of Destination Data Store SQL Database [State: Not Started] [Priority: High]

Category: Spoofing

Description: Temporary SQL DB may be spoofed by an attacker and this may lead to data being written to the attacker's target instead of Temporary SQL DB. Consider using a standard authentication mechanism to identify the destination data store.

Justification: <no mitigation provided>

27. Potential SQL Injection Vulnerability for SQL Database [State: Not Started] [Priority: High]

Category: Tampering

Description: SQL injection is an attack in which malicious code is inserted into strings that are later passed to an instance of SQL Server for parsing and execution. Any procedure that constructs SQL statements should be reviewed for injection vulnerabilities because SQL Server will execute all syntactically valid queries that it receives. Even parameterized data can be manipulated by a skilled and determined attacker.

Justification: <no mitigation provided>

28. Potential Excessive Resource Consumption for Reciever or SQL Database [State: Not Started] [Priority: High]

Category: Denial Of Service

Description: Does Reciever or Temporary SQL DB take explicit steps to control resource consumption? Resource consumption attacks can be hard to deal with, and there are times that it makes sense to let the OS do the job. Be careful that your resource requests don't deadlock, and that they do timeout.

Justification: <no mitigation provided>

29. Spoofing the Reciever Process [State: Not Started] [Priority: High]

Category: Spoofing

Description: Reciever may be spoofed by an attacker and this may lead to unauthorized access to Temporary SQL DB. Consider using a standard authentication mechanism to identify the source process.

Justification: <no mitigation provided>

30. Data Store Inaccessible [State: Not Started] [Priority: High]

Category: Denial Of Service

Description: An external agent prevents access to a data store on the other side of the trust boundary.

Justification: <no mitigation provided>

31. Data Flow Web files Is Potentially Interrupted [State: Not Started] [Priority: High]

Category: Denial Of Service

Description: An external agent interrupts data flowing across a trust boundary in either direction.

Justification: <no mitigation provided>

32. Data Flow Sniffing [State: Not Started] [Priority: High]

Category: Information Disclosure

Description: Data flowing across Web files may be sniffed by an attacker. Depending on what type of data an attacker can read, it may be used to attack other parts of the system or simply be a disclosure of information leading to compliance violations. Consider encrypting the data flow.

Justification: <no mitigation provided>

33. Data Store Denies Temporary SQL DB Potentially Writing Data [State: Not Started] [Priority: High]

Category: Repudiation

Description: Temporary SQL DB claims that it did not write data received from an entity on the other side of the trust boundary. Consider using logging or auditing to record the source, time, and summary of the received data.

Justification: <no mitigation provided>

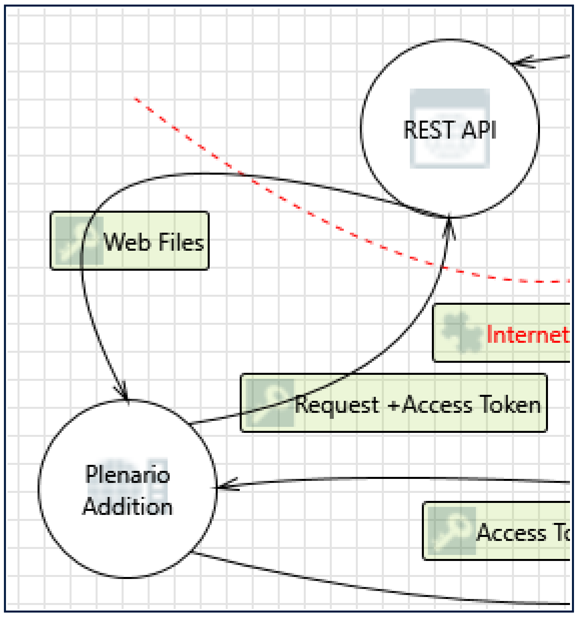
34. The Temporary SQL DB Data Store Could Be Corrupted [State: Not Started] [Priority: High]

Category: Tampering

Description: Data flowing across Web files may be tampered with by an attacker. This may lead to corruption of Temporary SQL DB. Ensure the integrity of the data flow to the data store.

Justification: <no mitigation provided>

Interaction: Web Files



35. Web Service Process Memory Tampered [State: Not Started] [Priority: High]

Category: Tampering

Description: If REST API is given access to memory, such as shared memory or pointers, or is given the ability to control what Plenario Addition executes (for example, passing back a function pointer.), then REST API can tamper with Plenario Addition. Consider if the function could work with less access to memory, such as passing data rather than pointers. Copy in data provided, and then validate it.

Justification: <no mitigation provided>

36. Elevation Using Impersonation [State: Not Started] [Priority: High]

Category: Elevation Of Privilege

Description: Plenario Addition may be able to impersonate the context of REST API in order to gain additional privilege.

Justification: <no mitigation provided>

37. Elevation by Changing the Execution Flow in Plenario Addition [State: Not Started] [Priority: High]

Category: Elevation Of Privilege

Description: An attacker may pass data into Plenario Addition in order to change the flow of program execution within Plenario Addition to the attacker's choosing.

Justification: <no mitigation provided>

38. Plenario Addition May be Subject to Elevation of Privilege Using Remote Code Execution [State: Not Started] [Priority: High]

Category: Elevation Of Privilege

Description: REST API may be able to remotely execute code for Plenario Addition.

Justification: <no mitigation provided>

39. Data Flow Web Files Is Potentially Interrupted [State: Not Started] [Priority: High]

Category: Denial Of Service

Description: An external agent interrupts data flowing across a trust boundary in either direction.

Justification: <no mitigation provided>

40. Potential Process Crash or Stop for Plenario Addition [State: Not Started] [Priority: High]

Category: Denial Of Service

Description: Plenario Addition crashes, halts, stops or runs slowly; in all cases violating an availability metric.

Justification: <no mitigation provided>

41. Potential Data Repudiation by Plenario Addition [State: Not Started] [Priority: High]

Category: Repudiation

Description: Plenario Addition claims that it did not receive data from a source outside the trust boundary. Consider using logging or auditing to record the source, time, and summary of the received data.

Justification: <no mitigation provided>

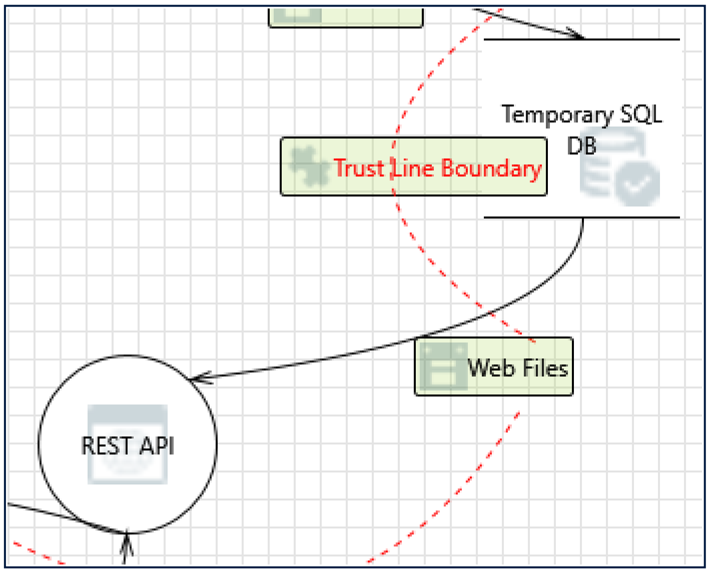
42. Weak Authentication Scheme [State: Not Started] [Priority: High]

Category: Information Disclosure

Description: Custom authentication schemes are susceptible to common weaknesses such as weak credential change management, credential equivalence, easily guessable credentials, null credentials, downgrade authentication or a weak credential change management system. Consider the impact and potential mitigations for your custom authentication scheme.

Justification: <no mitigation provided>

Interaction: Web Files



43. Spoofing of Source Data Store SQL Database [State: Not Started] [Priority: High]

Category: Spoofing

Description: Temporary SQL DB may be spoofed by an attacker and this may lead to incorrect data delivered to REST API. Consider using a standard authentication mechanism to identify the source data store.

Justification: <no mitigation provided>

44. Weak Access Control for a Resource [State: Not Started] [Priority: High]

Category: Information Disclosure

Description: Improper data protection of Temporary SQL DB can allow an attacker to read information not intended for disclosure. Review authorization settings.

Justification: <no mitigation provided>

45. REST API May be Subject to Elevation of Privilege Using Remote Code Execution [State: Not Started] [Priority: High]

Category: Elevation Of Privilege

Description: Temporary SQL DB may be able to remotely execute code for REST API.

Justification: <no mitigation provided>

46. Data Store Inaccessible [State: Not Started] [Priority: High]

Category: Denial Of Service

Description: An external agent prevents access to a data store on the other side of the trust boundary.

Justification: <no mitigation provided>

47. Data Flow Web Files Is Potentially Interrupted [State: Not Started] [Priority: High]

Category: Denial Of Service

Description: An external agent interrupts data flowing across a trust boundary in either direction.

Justification: <no mitigation provided>

48. Potential Process Crash or Stop for REST API [State: Not Started] [Priority: High]

Category: Denial Of Service

Description: REST API crashes, halts, stops or runs slowly; in all cases violating an availability metric.

Justification: <no mitigation provided>

49. Potential Data Repudiation by REST API [State: Not Started] [Priority: High]

Category: Repudiation

Description: REST API claims that it did not receive data from a source outside the trust boundary. Consider using logging or auditing to record the source, time, and summary of the received data.

Justification: <no mitigation provided>

50. Spoofing the REST API Process [State: Not Started] [Priority: High]

Category: Spoofing

Description: REST API may be spoofed by an attacker and this may lead to information disclosure by Temporary SQL DB. Consider using a standard authentication mechanism to identify the destination process.

Justification: <no mitigation provided>

51. Elevation by Changing the Execution Flow in REST API [State: Not Started] [Priority: High]

Category: Elevation Of Privilege

Description: An attacker may pass data into REST API in order to change the flow of program execution within REST API to the attacker's choosing.

Justification: <no mitigation provided>