Bachelor Thesis

Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Penetration Testing of the eSano Platform

Moritz Schumacher
Department of Computer Science
Chair of Computer Science II (Secure Software Systems)

Prof. Dr.-Ing. Alexandra Dmitrienko
Reviewer

B. Sc. Peter Ten
Advisor

Submission
15. March 2023

www.uni-wuerzburg.de

Abstract

We performed a penetration test on the eSano eHealth platform. The platform provides internet- and mobile-based interventions for mental health and chronic disease treatment. The primary objective of the pentesting was to enhance the security of the eSano platform, which in turn ensures the protection of sensitive patient data used and stored within it. The scope of the pentesting was one of three sub-platforms of eSano, as well as the API which connects all sub-platforms with the backend database.

We employed a combination of OWASP resources and various tools to identify potential attack vectors, and subsequently prioritized them based on their impact, enabling us to concentrate our efforts on addressing the most critical vulnerabilities first.

During the penetration testing of the eSano platform, we identified several vulnerabilities. These included multiple vulnerabilities with low and medium impact, such as unused endpoints, functionalities, and missing rate limits at certain points. In addition to these lower impact vulnerabilities, we also identified several critical issues. One critical issue involved a clickjacking attack that could trick users into performing unintended actions. We could also query whether an email address was associated with a platform user, which could lead to further attacks. A misconfiguration in the CORS protocol allowed third-party websites to make privileged requests about authenticated users. Finally, we were able to bypass the file extension restrictions and perform successful XSS, SSRF, and DoS attacks on the file upload functionality. This could potentially lead to compromised user data, unauthorized access to sensitive information, and potential denial of service.

The findings of the pentesting have been shared with the developers of the eSano project, enabling them to address the identified vulnerabilities. This will make the platform more secure for data belonging to future users.

Zusammenfassung

Wir haben einen Penetrationstest auf der eSano eHealth Plattform durchgeführt. Die Plattform bietet internet- und mobilbasierte Interventionen zur Behandlung von psychischen und chronischen Erkankungen an. Das Hauptziel des Pentests war es, die Sicherheit der eSano Plattform zu verbessern, um die sensiblen Patientendaten, die darin genutzt und gespeichert werden, zu schützen. Der Umfang des Pentests umfasste eine der drei Sub-Plattformen von eSano sowie die API, die alle Sub-Plattformen mit der Backend-Datenbank verbindet.

Wir haben eine Kombination aus OWASP-Ressourcen und verschiedenen Tools eingesetzt, um potenzielle Angriffsvektoren zu identifizieren und sie anschließend auf der Grundlage ihreres Einflusses zu priorisieren. Dadurch konnten wir uns auf die Behebung der kritischsten Schwachstellen konzentrieren.

Während des Penetrationstests der eSano-Plattform haben wir mehrere Schwachstellen identifiziert. Diese Schwachstellen umfassten mehrere mit geringer und mittlerer Auswirkungen, wie z. B. ungenutzte Endpunkte und Funktionalitäten sowie fehlende Rate-Limits an bestimmten Stellen. Zusätzlich zu diesen Schwachstellen mit geringerem Einfluss, haben wir auch mehrere kritische Schwachstellen identifiziert. Eine kritische Schwachstelle betraf einen Clickjacking-Angriff, der Benutzer dazu bringen könnte, unbeabsichtigte Aktionen auszuführen. Wir konnten auch abfragen, ob eine E-Mail-Adresse mit einem Plattformbenutzer verbunden war, was zu weiteren Angriffen führen könnte. Eine Fehlkonfiguration im CORS-Protokoll erlaubte es Drittanbieter-Websites, privilegierte Anfragen über authentifizierte Benutzer zu stellen. Desweiteren konnten wir die Upload-Beschränkungen für Datei-Erweiterungen umgehen und somit erfolgreiche XSS-, SSRF- und DoS-Angriffe auf der Datei-Upload-Funktionalität durchführen. Diese können zu kompromittierten Benutzerdaten, unbefugtem Zugriff auf sensible Informationen und Denial-of-Service führen.

Die Ergebnisse des Penetrationstests wurden den Entwicklern des eSano-Projekts zur Verfügung gestellt, so dass sie die identifizierten Schwachstellen beheben können. Dadurch wird die Plattform sicherer für die Daten zukünftiger Nutzer.

Contents

1. Introduction

Smartphones, laptops, tablets, and other devices are the everyday companions of most people. Even in psychotherapy, the digitization is not stopping. Online training, apps, social networks, and video conference therapy are increasingly being used in the psychotherapy sector. A term commonly used in academia for this is Internet and mobile-based interventions (IMIs) [1]. IMIs offer the possibility of closely accompanying the affected person in daily life and thus integrating the treatment content into the daily routine in the most effective manner [1]. Studies have shown that Internet-based treatments can be effective [2]. Guided IMIs, in which patients receive treatment counseling and support, have proven particularly effective [3]. Currently, the focus of research on modern platforms for psychotherapeutic therapies has shifted from their suitability to their specific applications and methods [4]. Given that the platform collects sensitive patient information, including personal and medical data, there is a risk of this information being accessed or compromised by unauthorized individuals. Penetration testing (pentesting), a process of evaluating the security of a system by simulating an attack by a malicious actor, can help identify vulnerabilities in the platform's security infrastructure and ensure that sensitive patient information remains confidential. Therefore, it is crucial to conduct a thorough pentesting of such an electronic health (eHealth) platform to mitigate potential security risks and to protect patient privacy.

The focus of this thesis will be pentesting of the eSano eHealth platform [5]. eSano is a project of the Chair of Clinical Psychology and Psychotherapy, and the Institute of Database and Information Systems (DBIS) of the University of Ulm. It is developed in close cooperation with the Chair of Health Informatics of the University of Würzburg. eSano offers IMIs and can provide location- and time-independent support for various mental health conditions. The eSano platform consists of four components: A web-based Content management system (CMS) sub-platform , a web-based eCoach sub-platform, and a patient application sub-platform. These three sub-platforms are connected by an API, and the used data are stored in a backend database. The scope of this thesis will be limited by the analysis of the CMS sub-platform and the API that connects the CMS with the backend database. To ensure the goal of higher security on the eSano platform, we have developed a comprehensive approach that includes the following steps:

1. **Reconnaissance (information gathering).** Within the reconnaissance, we gathered information about the eSano platform, including its technology stack, architecture and functionalities. We gained a deeper understanding of the platform and identified areas of vulnerabilities, which helped us identify potential attack vectors.

2. **Definition and prioritisation of testing vectors.** Based on the information gathered during reconnaissance, we defined our testing vectors. This involved selecting specific areas of the platform to target, such as the authentication or file upload functionality. Afterwards, we prioritized our testing vectors based on the potential impact. Therefore, we utilized resources from OWASP [6][7][8][9].

3. **Testing and documentation.** The testing included documentation of each successfully exploited vulnerability, providing a proof of concept and detailing the potential impact of the vulnerability. This information was used to create a comprehensive report outlining the vulnerabilities discovered, giving recommendations for remediation, as well as providing suitable Common Weakness Enumerations (CWEs) [10] to provide more resources within the field of the found vulnerability. Finally, we shared our documentation with the developers of eSano, so they can remediate the vulnerabilities and thereby improve the security of the platform.

The results of our pentesting have shown that there are several critical vulnerabilities within the eSano platform that could potentially lead to unauthorized access or manipulation of sensitive patient data. These vulnerabilities could enable attackers to steal or modify personal and medical information.

Outline. The thesis is organized as follows: Chapter 2 provides essential background knowledge regarding security in the eHealth domain in general, as well as information about the eSano platform. In Chapter 3, we survey the related work, which includes other pentestings on eHealth platforms, as well as security standards and cyberattacks in the eHealth sector. We then describe the approach used for our pentesting in Chapter 4, and present results of the reconnaissance in Chapter 5. The vulnerabilities that were successfully exploited are listed in Chapter 6, the vulnerabilities that we were not able to exploit are listed in Chapter 7. We conclude the thesis by presenting the results and discussing possible future work in Chapter 8.

2. Background

In this chapter, we will provide the necessary background information for the thesis. First, in Section 2.1, we will present the eSano platform. Next, in Section 2.2, we will describe OWASP, which we utilize in this project to guide our security analysis. Section 2.3 will be devoted to security issues in web frameworks, and Section 2.4 focuses on security standards in web applications. In Section 2.5 we will discuss european security standards in the eHealth sector, before we finish this chapter by discussing the used tools in Section 2.6.

2.1 eSano online Trainings

As it was already mentioned in Section 1, the eSano platform consists of three sub-platforms: A web-based CMS, a web-based eCoach sub-platform, and a web-based patient sub-platform. An API connects the sub-platforms with a backend database. The aim of the project is to create a technological infrastructure for a central platform on which a wide variety of interventions can be offered. There are four roles that a user can be: An admin, an eCoach, an editor, or a patient. Depending on the role, an user can access a sub-platform. The CMS sub-platform can be used from the admin and the eCoach and serves therapists or other contributors without extensive technical knowledge to create new online lessons, which result in interventions. The eCoach sub-platform can be accessed by the eCoaches and enables all eCoaches (e.g. therapists) to perform administrative activities around the online lessons. This includes, for example, the creation and activation of new patient accounts. The patients can only access the patient sub-platform, which provides patients easy access to all content, tools, and information unlocked for them [11].

Due to limited timeframe available for the thesis, the pentesting of overall eSano platform is divided among several students. We recall that this thesis will deal with the analysis of CMS sub-platform and th API that connects the sub-platforms. Hence, in the following we briefly discuss the functionalities of these two components within the two roles (admin, editor) which can use the CMS sub-platform.

- **eSano Content-Management-System (CMS)** The CMS platform is accessible under the following Uniform Resource Locator (URL): *https://cms.esano.pentest.klips-ulm.de/*. Depending on the role one is currently acting within the CMS, different functionalities depending on the role are provided. For this reason, a rough overview of the roles and corresponding functionalities is given below.
  The admin The administrator has the ability to modify user roles and permissions.

This means that they can assign users the roles of admin, eCoach, or editor, as well as create new user accounts with the corresponding role. In addition, the administrator can resend verification emails and delete existing admin-, eCoach- or editor-accounts. The admin also has the ability to create, edit, and delete announcements for all sub-platforms (CMS, eCoach, and patient platform). An announcement is a notification, which can be displayed to selected users. They can specify the announcement type as either ”success,” ”info,” ”warning,” or ”error,” and choose on which sub-platforms they should appear as a pop-up window.
The editor Users with the role editor have the ability to manage their own groups, including creating, editing, and deleting them. They can also control access to their groups (quantity of patients) and manage interventions within those groups, such as creating, editing, deleting, and importing them. An intervention is a treatment that the eCoach offers to one or more patients via the eSano platform. It is worth noting that an intervention cannot be created without being assigned to a group. Additionally, the editor has ”read-only” access to all other groups and interventions created by other editors, but they cannot modify them.

- **API** The API is accessible under the following URL: *https://api.esano.pentest.klips-ulm.de/*. API mediates any requests from the three sub-platforms and the database. Its functionalities vary from authentication, to importing/exporting data from the database and presenting to users in different formats. RESTful interfaces are used to exchange data between different applications of the eSano platform. RESTful interfaces are a type of software interface that uses the principles of Representational State Transfer (REST) to allow communication between different systems or software components over the Internet. During transmission, the requested objects are transformed into JavaScript Object Notation JavaScript Object Notation (JSON). [12].

The eSano platform was developed using open source software. For the front end, the frameworks Angular [13], Ionic [14] and Vue.js [15] (JavaScript,TypeScript) were used. Angular [13] is being used to create the core functionality of the eSano platform, including managing data and providing dynamic user interactions. Ionic [14] is being used to develop a mobile application version of the eSano platform. It is a hybrid mobile application development framework built on top of Angular. Vue.js [15] is being used to build interactive user interfaces and handle real-time data updates. The Laravel framework [16] was used for the back-end. The backend is based on the use of REST architecture. All communication is encrypted and transmitted through Hypertext Transfer Protocol Secure (HTTPS). Kraft et al. [17] describe the functional requirements of the eSano platform. Based on these, the following security-relevant functional requirements are defined and need to be fulfilled:

1. Authentication and authorization mechanisms to allow only privileged users to access the sub-platforms. Access should only be granted to their own data or data to which they have access rights.

2. Role-based access control to allow certain users to access different areas of the sub-platform

3. There should be grouped work environments for users. So, only users which are part of the specific group should be able to work in this environment.

4. There are condition dependent parts of interventions. This means that they are only displayed under certain conditions (e.g., if a question was answered right)

5. Each lesson should be definable as being unlocked a) always, b) after completion of the previous lesson, c) at a specific date & time, or d) manually by the eCoach.

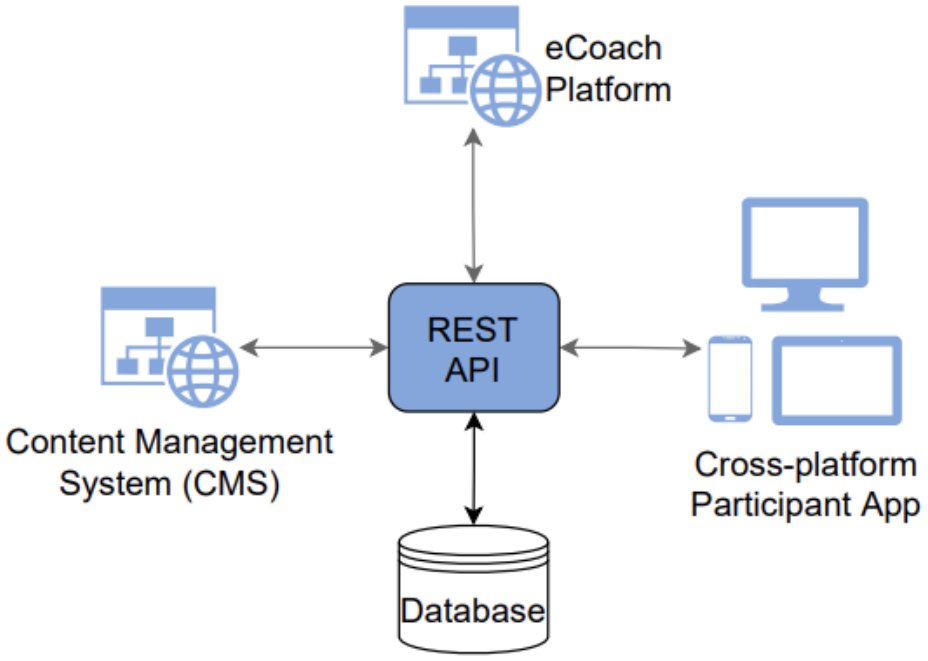

Figure 2.1: Sub-platfroms of the eSano platform in connection with the API and the back-end database [17].

This list only contains eSano security-relevant functional requirements and is, thus, definitely not complete. There are more general requirements we also consider in our analyses and discuss in Section 2.4.

2.2 Open Web Application Security Project (OWASP)

The Open Web Application Security Project is a nonprofit foundation dedicated to improving software security. It publishes different resources on software security [6]. Relevant to our work are the following: Web application security (OWASP) [7, 8] and API security [9]. These can be used extensively to guide the security analysis. The project provides a complete testing framework, not just a simple checklist or recipe for items to be covered. It can be used as a template to create own test programs or to evaluate the processes of others. The test guide describes in detail both the general test framework and the techniques required to implement the framework in practice [7].
The OWASP Top 10 [8] is a standard document on the awareness of developers and the security of web applications. It represents a broad consensus on the most critical security risks for web applications. In the latest OWASP Top 10 of 2021 [8], eight of the ten categories are the most common issues, as was established in completed security analyses. Application Security researchers are constantly researching new vulnerabilities and ways to identify and fix them. It can take years before a weakness can be tested on a scale, or in other words, testing a weakness in a way that simulates real-world conditions and environments. For this reason, OWASP uses a community survey to ask application security and development experts on the front lines what they see as essential weaknesses that the data may not show yet. The result of it are the two further categories. Together with the eight mentioned above, they build the OWASP Top 10 from 2021 [8]. According to them, the ten most common vulnerabilities in web applications are (1) Broken Access Control,

(2) Cryptographic Failures, (3) Injection, (4) Insecure Design, (5) Security Misconfiguration, (6) Vulnerable and Outdated Components, (7) Identification and Authentication Failures, (8) Software and Data Integrity Failures, (9) Security Logging and Monitoring Failures, (10) Server-Side Request Forgery.
Since the API of the eSano project will also be part of our analysis, in addition to the OWASP Top 10 for web applications [8], we will also use the latest OWASP API security Top 10, which is from 2019 [9]. These are (1) Broken Object Level Authorization, (2) Broken User Authentication, (3) Excessive Data Exposure, (4) Lack of Resources & Rate Limiting, (5) Broken Function Level Authorization, (6) Mass Assignment, (7) Security Misconfiguration, (8) Injection, (9) Improper Assets Management, (10) Insufficient Logging & Monitoring. This list, and this mentioned above, can help us to become aware of possible weaknesses and can also help us to determine the testing order of potential vulnerabilities, by first focussing on the highly ranked vulnerabilities.

2.3 Security Issues in Web Frameworks

At the beginning of web development, all applications were coded manually, which resulted in many human caused errors. Nowadays, hardly any web application is developed without using a web framework. These avoid mistakes, make developers' work easier, and create enormous time savings, since the planning of the application no longer has to be done from scratch. Furthermore, the eSano platform is developed using several web frameworks, as already mentioned in Section 2.1. In the following, we will take a look at the Laravel framework to see what it aims to achieve in terms of security and whether they can meet this goal. Laravel framework [16], used in eSano, is considered secure because it has a fully integrated security system [18]. This covers both client-side and server-side security, so the developer does not need to use other technologies. Among other things, the covered security aspects are as follows [19]:

- Salted passwords
- Password protection against Cross-Site-Request-Forgery (CSRF)
- "Bcrypt Hashing Algorithm" for password encryption
- Multiple authentication layers in the system

Despite all this, the Laravel framework [16] had vulnerabilities with respect to SQL injection, exposure of sensitive data, External entities XML (XXE), access control violation, and XSS [20].

2.4 Security Standards in Web Applications

Security is an important requirement for any application today, especially those delivered over the Web, because they are accessible over the Internet and are therefore vulnerable to a wide range of cyber threats. Web-based applications include several security requirements [21]:

- **Integrity**: It is about preventing unauthorized modification of information.
- **Confidentiality**: Keeping information private and protected from unauthorized access or disclosure.
- **Availability**: The goal is to minimize the risk of system failures.
- **Accountability**: Through control and audit systems, users can be made aware of, or even held accountable for their actions and the way they perform. In terms of security it could be a strong user authentication and access control.

- **Freshness**: Mechanisms against replay attacks [22].

In case of eHealth platforms, these general requirements could be transferred into secrity-relevant functional requirement for the eSano platform [23]:

- Ensure that only eligible users can log in the system.
- Ensure the secrecy of patient details like name or age.
- Ensure the secrecy of the patient's disease information.
- Guarantee scalability of the system.
- Ensure that the doctor's prescription is disclosed.
- Ensure that patients details are available to only authorized doctors.

To verify if these requirements are fulfilled, one can follow a number of standards or organisations for web application security such as the OWASP, the Application Security Verification Standard (ASVS) [24], the National Institute of Technologies (NIST) Special Publication (SP) 800–218 [25], the International Organization for Standardization (ISO) 27034 [26] or the Center for Internet Security (CIS) Control 16: Application Software Security [27]. Goal of them is to provide a lower bound of required security requirements that a system must meet. An example of this is Transport Layer Security (TLS) [28] for achieving confidentiality [29]. Ultimately, standards rarely require or enable the implementation of best practices in security. The main focus is rather to provide a common, often lower than possible, denominator of security for the masses [29].

2.5 European Security Staandards in eHealth

In recent years, there has been a growing interest in the use of eHealth systems across Europe. eHealth systems are designed to improve the quality and efficiency of healthcare services by using technology to manage health information and facilitate communication between healthcare providers and patients. However, the adoption of eHealth systems also raises concerns about the security and privacy of sensitive health data. In response to these concerns, various security standards and regulations have been developed to ensure the secure implementation and operation of eHealth systems in Europe.

One such standard is the European Union Agency for Cybersecurity's (ENISA) "Baseline Security Recommendations for Internet of Things (IoT) in the context of Critical Information Infrastructures" [30]. This standard provides a set of security recommendations for eHealth systems that use IoT devices. The recommendations cover a range of security topics, including access control, authentication, data protection, network security, and incident management.

Another important standard is the General Data Protection Regulation (GDPR) [31], which is a regulation in European Union (EU) law on data protection and privacy for all individuals within the EU and the European Economic Area (EEA). The GDPR has specific requirements for the processing of personal data in eHealth systems, including the need to obtain explicit consent from patients and to implement appropriate technical and organizational measures to ensure the security and confidentiality of personal data.

The European Committee for Standardization (CEN) has also developed a set of standards for eHealth systems, including EN 13606 [32], which provides a common framework for the exchange of health data between different healthcare providers and systems. The standard includes requirements for data security and confidentiality, as well as data protection and access control.

2.6 Used Tools

Besides OWASP as testing framework, we use several tools within the pentesting. In the following, we will give a quick description for a view of them.

2.6.1 Burp Suite

Burp Suite [33] is a comprehensive web application security testing platform that includes a variety of tools for performing various security testing tasks. The suite includes an intercepting proxy, scanner for identifying vulnerabilities in web applications, and an intruder tool for performing various attacks on web applications. With Burp Suite, we can perform tasks such as conducting security assessments, finding security vulnerabilities, testing for weaknesses in application logic, and testing the security of web applications by simulating attacks. Additionally, Burp Suite provides a suite of tools for performing manual security testing, making it a powerful and versatile platform for web application security testing.

2.6.2 Nmap (Network Mapper)

Nmap (Network Mapper) [34] is an open-source security tool used for network exploration, management, and security auditing. We use it to discover hosts and services on a computer network, thus creating a "map" of the network. Nmap is widely used by network administrators, security professionals, and penetration testers for various tasks such as network inventory, management, and security auditing. Nmap's flexibility and accuracy make it a valuable tool for discovering and monitoring the security of a network.

2.6.3 Postman

Postman [35] is a powerful collaboration platform for API development that provides a comprehensive and efficient solution for testing, developing, and documenting APIs. With Postman, we can create and send HTTP requests, manage environments and collections of requests, and view and analyze responses from APIs in a user-friendly format. One of the key features of Postman is the ability to create and save requests, including those with custom headers, parameters, and payloads.

2.6.4 Metasploit

Metasploit [36] is an open-source software platform for developing, testing, and executing exploits for vulnerabilities in computer systems. It provides a framework for writing, testing, and executing security exploits, allowing security researchers and penetration testers to automate the process of identifying and exploiting vulnerabilities in computer systems. We use it manly to find vulnerabilities in outdated components and to create attack payloads.

3. Related Work

The following sections examine what work related to ours already exists. First, we will discuss some security standards in the health sector in Section 3.1. In Section 3.2, we will look at a few examples for penetration tests or security analyses in the eHealth domain. In Section 3.3, we will survey cyberattacks in the health sector. Section 3.4 presents related work on managing and testing web application security. Section 3.5 gives an overview about the literature of common attacks, by discussing the OWASP Top 10 Vulnerability Risks in Web Applications [8].

3.1 Security Standards in the Health Sector

Securing patient data has always been taken seriously in eHealth. In the United States, the Health Information Portability and Accountability Act (HIPAA) was published in 1996 to ensure security of patient data, while other countries have developed similar security regulations [37]. The main elements are, for example, prevention of unauthorized access to patient data (ensuring confidentiality), preventing unauthorized modification or loss of data (integrity), and preventing compromise of the availability of data to authorized individuals (availability).

There are also critics of HIPAA. While most healthcare data centers are HIPAA certified, Patil et al. [38] say that certification is not a guarantee of patient data security, because HIPAA's focus is too much on ensuring security policies and procedures than on implementing them. The frequency with which compliance with any standards [37] is checked is the responsibility of each health facility. However, it is recommended they should be made at major software changes as a minimum [39].

Specific guidelines for the conduct of such analyses exist. In a recent guide to privacy and security of health information in the United States, the Office of the National Coordinator (ONC) for Health Information Technology (Health Information Technology (HIT)) suggested five steps that are necessary to perform a security risk analysis [37]: The first step includes (1) reviewing current health information security. After that, (2) threats and vulnerabilities are to be identified. Thirdly, the vulnerabilities and the associated risks are to be (3) evaluated according to probability and impact. Subsequent the (4) security risks are to be mitigated before finally the (5) results are to be monitored. This guide is similar to the OWASP guide [7] (cf. Section 2.2). We follow OWASP guidelines in the upcoming work, especially because in addition to a guide for the procedure, it also provides a

complete framework with tools to use and step-by-step instructions for exploiting vulnerabilities. We hope that this will increase efficiency, especially in view of the limited time available.

In Europe, healthcare organizations such as hospitals are also responsible for the security of private patient data. The security must comply with the European Data Protection Regulation (GDPR) [40]. In Germany, the Federal Ministry for Economic Affairs and Climate Action additionally offers the "Orientierungshilfe zum Gesundheitsdatenschutz" (Orientation Guide to Health Data Protection) [41], which is intended to help medical institutions to implement data protection in a practical and efficient manner.

3.2 Penetration Tests in Health Sector

Penetration testing (pentesting) deals with finding vulnerabilities in systems through simulated, authorized attacks. Documenting the procedure could help the system developers to fix these vulnerabilities.

Keven Zimmerman performed a security analysis of the UniNow application focused on private data [42]. One reason for this was the fact that many universities used the contact tracing functionality of the app during the COVID-19 pandemic. As a result, UniNow had to deal with a lot of private data. Several vulnerabilities were found through this analysis, including (1) Disclouser of multiple secret keys, (2) Sensitive data in application backups, (2) JavaScript Injection, (3) Brute-forceable appointment invitations, (4) Account takeover using device IDs, and (5) Unauthorized access to an internal endpoint.

Baumgärtner et al. [43] show that the Google/Apple Proposal (GAP) "is vulnerable to (i) profiling and possibly de-anonymizing infected persons, and (ii) relay-based wormhole attacks that basically can generate fake contacts with the potential of affecting the accuracy of an app-based contact tracing system".

Among other things, Filipp Roos thematizes performs a security analysis of the Google/Apple Exposure Notification (GAEN) (e.g., used by Corona-Warn-App (CWA)), and the TraceCORONA (TC) approach, developed from the Technical-University Darmstadt [44]. Therefore, he lists all known attacks which are applicable to contact tracing solutions and checks if GAEN or TC are vulnerable to each of them. He tested the two applications against 26 attacks from four categories: (1) Profiling with or without bluetooth LE communication, (2) Fake alert injection, (3) Time modification attacks, (4) Status reveal attacks, and (5) digital evidence. GAEN is affected at least partially to 14 of these attacks, TraceCORONA at least to 8. Furthermore, he proposed a novel Sybil attack on TC in which a user generates more than one key pair for the Diffie-Hellman key exchange and therefore simulates many devices of his own. Filipp Roos shows how in different scenarios and under given assumptions the COVID-19 contagion rate calculated by TC can be manipulated by this attack.

The eSano project has also been subjected to a security analysis in 2021. Tobias Sterns did this within his Software-Engineering Project of University of Ulm [45]. Before testing, he set up a pentest server for the eSano project. We also use it for our security analysis. We would like to emphasize the importance of security analysis in the context of this bachelor thesis, knowing that one was already carried out last year. Therefore, we list arguments that support a renewed security analysis of the eSano project:

1. The eSano project is still in the development phase. The structure of the project at the time Tobias Sterns did his security analysis was a different as it is know. It is logical to assume that through development (e.g., of new functionalities), new possible vulnerabilities could arise. We want to find these.

2. An informal security analysis is never complete. If a other person with a different subjective view on the target is performing it, it is always possible to find some not yet discovered vulnerabilities. This argument supports the importance of regular security analyses in general and thus in this bachelor thesis.

3. Tobias Sterns penetration test was limited in time by the time frame devoted to the practicum, and the methodology or prioritization of attack vectors tested isn't clear. It could have been just a coincidence, which makes it difficult to assess how well the analysis reflects the actual security level of the eSano project. Our analysis is based on the resources of OWASP [7]. By testing the most common vulnerabilities according to OWASP [8, 9], we can deduce that the likelihood of a existing non-common vulnerability, is comparatively low. With this we hope to be able to make a more conclusive statement about security of the eSano project.

3.3 Cyberattacks in Health Sector

There were many cyber attacks reported for a health sector. Kandasamy et al. [46] is investigating cyberattacks on Asian health care facilities. These have particularly increased since the switch from paper-based to Electronic Health Records (EHRs) systems. According to Kandasamy et al. [46], the most commonly used types of attacks on healthcare facilities are (1) Trojan attack, (2) Phishing attacks, (3) Ransomware, (4) Advanced Persistent Threat (APT) and (5) Malware - credential compromise. The main reasons for access are lack of proper anti-malware/anti-virus software, poor infrastructure, lack of cybersecurity awareness among the healthcare systems staff, and the absence of vulnerability and risk management processes and practices. But also technologically advanced countries such as the USA have to deal with cyberattacks such as the employee responding to a phishing email with login credentials [47], successful hacking efforts by Dark Overlord [48], and a multitude of various WannaCry ransomware encryptions [49].

This leads to an increase in data breaches. According to Healthcare Data Breach Report [50], "38 health data breaches with 500 or more records were reported to the Department of Health and Human Services' Office for Civil Rights in December 2019, an 8.57% increase from November 2019". According to Seh et al. [51], the healthcare industry is also particularly affected by data breaches compared to other industries. In the United States, 61.55% between 2005 and 2019 and 76.95% between 2015 and 2019 of the known data breaches were in the healthcare sector, which means that it has become the main victim of data breaches. Furthermore, the rate of data breaches in healthcare providers has increased even more rapidly since 2015. Seh et al. [51] also analyzed the types of attack from with which the health sector suffers. According to the authors, the healthcare service providers have been inundated with hackers since 2015, breaching the confidentiality of 90.49% health records during this time period. This shows the vulnerability of the health sector, especially in relation to hackers.

Gafni et al. [52] deals with cyberattacks during the COVID-19 pandemic. According to the World Health Organization (WHO), the number of them has increased fivefold by 2020 [53]. One of the main targets was healthcare facilities. The consequences of the attacks were so drastic that the International Red Cross Committee of more than 40 international leaders were "calling on world governments to take immediate and decisive action to prevent and stop cyberattacks that target hospitals, health care, research organizations, and international authorities providing critical care and guidance in the middle of the ongoing COVID19 pandemic" [54]. Furthermore, the European Union Agency for Cybersecurity (ENISA), as a state facility, wants to "advise cybersecurity to support hospitals and the healthcare sector against the increase of phishing campaigns and ransomware

attacks during the COVID19 crisis" [55] since hackers tried to break into the WHO in March 2020.

3.4 Testing and Managing Security of Web Applications

Yadav et al. [56] gives an overview and recommendations in security of academic institutions, e-commerce applications, database, data in general, operating systems and mobile applications.

Curphey et al. [57] focus on the different tools to test web applications. These are source code analyzers, web application (black-box) scanners, database scanners, binary analysis tools, runtime analysis tools, configuration analysis tools, proxy tools or proxies, and miscellaneous tools. In addition, they criticize that due to lack of regulations, scanner vendors "attempt to create and play on significant fear" of customers to push their selling numbers and therefore advise to test scanners on known applications before the actual usage.

Bau et al. [58] thematize automatic black-box web application vulnerability scanners, describe what vulnerabilities are tested by the scanner, how representative are the results, and how effective they are. According to them, Cross-Site Scripting, SQL Injection, other forms of Cross-Channel Scripting, and Information Disclosure are the most prevalent classes of vulnerabilities in web applications. Furthermore, they outline that such scanners are adept at detecting straightforward historical vulnerabilities and have room for improvement in more advanced vulnerabilities. This fact is also underlined by [59] by saying "the challenge faced by automated tools for web security testing is that they have to keep up with the ever-changing and evolving technologies [...]".

Huang et al. [60] constructed a mechanism to assess the security of web applications. Therefore, they (1) designed a crawler, which contains and imitates web browser functions to test web applications in real-world scenarios, followed by a (2) fault detection process to determine the most vulnerable points found through the crawler. The Web Application Vulnerability and Error Scanner (WAVES) [61] analyzes the results by using the NRE Algorithm [62] to eliminate false positives.

A different approach is taken by Jan et al. [63]. They propose a framework for asset security classification of information systems. Therefore, they collect data about the systems and assign them to high, medium, and low security levels. The results provide organizations a detailed overview about the security requirements for their assets and to prioritize security testing.

Finally, Montesino et al. [64] propose a Security Information and Event Management (SIEM) based framework to automate security controls. Using the proposed approach, organizations can automate as many security controls as possible, thus achieving greater efficiency in security management while reducing complexity in the process.

3.5 Common Vulnerabilities in Web Applications

This section provides a comprehensive overview of the 10 most common vulnerable fields in web applications, based on the OWASP top 10 list [8]. Discussing the OWASP Top 10 vulnerabilities covers the major risks in web applications because these vulnerabilities are the most commonly exploited and have the potential to cause significant harm to both the application and its users [8]. By addressing these vulnerabilities, web application developers and security professionals can significantly improve the security posture of their applications. The OWASP Top 10 is a widely recognized and accepted standard

for identifying and mitigating web application vulnerabilities, and it is regularly updated to reflect emerging threats and attack techniques [8]. In the following, various studies and approaches will be discussed for each of these fields, highlighting their significance in web application security. The coverage of this section is extensive and includes selected examples from the literature.

Access Control: According to Hassan et al. [65], in a sample of 330 websites, 30,09% were affected by a broken access control (Broken Access Control (BAC)) vulnerability. Furthermore, the frequency of BAC vulnerabilities was also divided into sectors. While 27.91% of the web applications tested in e-Commerce were affected, the percentage for the healthcare sector was 7.75%. Ouaddah et al. [66] deals with access control in the Internet of Things. Therefore, an overview of current IoT applications was created to perform an analysis of the authorization process in IoT. Akl et al. [67] show a scheme, based on cryptography, to achieve access control in hierarchies.

Cryptographic Issues: Lazar et al. [68] address the question of why the implementation of mathematical cryptographic concepts often fails. Therefore, they assign existing problems to three layers (primitive, protocol, application) where they can occur. 83% of the bugs are in applications that misuse cryptographic libraries, and only 17% of them are in the libraries themselves. The most common application-specific vulnerability is about forgetting to encrypt important data, while bugs in the cryptographic schemes themselves are mostly due to the brute-forcing of weak keys.

Injection: Injection is a big topic in web application security. Anley et al. [69] describe a SQL injection attack to the popular Microsoft Internet Information Server/Active Server Pages/SQL Server platform. Halfond et al. [70] present an approach to identify SQL injections. First, static string analysis is used to create a model that represents the set of all legitimate, expected SQL statements, which can be caused by the application. Real-time monitoring of the application now checks every SQL query that arises to see whether it violates the static model. Through this process, SQL injection attacks (SQL Injection Attacks (SQLIA)) should be recognizable. The proposed approach was able to stop realistic SQLIA on two simple examples. All legitimate SQL queries were approved and all attacks were blocked. There were no false positives or false positives cases.

Insecure Design: Schechter et al. [71] deal with the security and reliability of "secret" questions for authentication. They tested "secret" questions of 116 participants against statistical guessing. It was possible to guess 13% of the "secret" answers by providing five most popular answers for all participants. Naiakshina et al. [72] address the question why developers store passwords insecurely. According to their study, (1) developers think of functionality before security, (2) none of the participants who were not involved in the study objective (implement secure password storage) thought that the task of storing passwords needed a secure solution, despite the fact that password storage is obviously a security sensitive task. (3) No implementation of secure password storage met the security level that is possible with academic. This result also entails actions of the academic. They have to push the inclusion of new findings in current frameworks or standards. Lastly, (4) the frameworks which support high-level secure password storage leave the decision of whether to apply this to the developers as an optional feature.

Security Misconfigurations: A common field for misconfiguration is the Cross-Origin Resource Sharing (CORS). According to Chen et al. [73], 27.5% of the investigated domains that use CORS had insecure misconfigurations. Many security issues are rooted in the design and implementation of the CORS protocol itself and not in the configuration. Eshete et al. [74] audit the security of the configuration of PHP, MySQL and Apache in different operating systems (OS). The default, configuration is compared to the configuration

with the help of their proposed system SCAAMP. The system detects security configuration vulnerabilities in security critical directives and audits, fixes those and quantitatively measures the safety ratings of the environments. By default, the average safety rate of PHP, MySQL and Apache is 42.95%, 12.60% and 7.95%, respectively. With SCAAMP protection, an average safety rate of 56.96%, 50.00% and 33.33% is achieved.

Vulnerable and Outdated Components: Often attackers take advantages from known vulnerabilities of the used third-party components. Statically, scanners exist, such as the OWASP Dependency Check [75]. It scans the Project-Object-Model files of java systems for direct dependencies. By using this, Cadariu et al. [76] found that 54 out of 75 systems use at least 1 (and up to 7) vulnerable libraries. Most of the systems came from large Dutch companies. The results found were previously cleaned from false positives. However, approximately 78% of the vulnerabilities found are in indirect dependencies [77]. Therefore, Zhan et al. [77] propose the ATVHunter. The ATVHunter can effectively and efficiently find third-party libraries in the application and is oblivious to state-of-the-art obfuscation techniques.

Identification and Authentication: Authentication and identification are a big field in which attackers aim to steal or gain access to identities. In terms of identification, a common exercise is to deidentify personal data from the medical sector to preserve the privacy of, for example, research participants. Many attacks focus on restoring the identity of these individuals. This process is called reidentification. According to El Emam et al. [78], the overall success rate for all reidentification attacks was approximately 26%, and 34% for health data. The informative value of these figures is limited due to small data volumes and the associated large confidence intervals around the estimates. Only one attack scenario in the medical sector was carried out using current standards. There, the successful reidentification rate was only 0.013%. The reason for the high overall estimate is traced back to unused successful reidentification attacks. Raza et al. [79] offer authentication attacks and methods to secure this process. They give an overview of whether password-based authentication methods are resistant to brute-force attacks, dictionary attacks, shoulder surfing, replay attacks, phishing attacks, key loggers, and video recording attacks.

Data Integrity: Data integrity ensures that the data are correct and that they have not even been improperly changed in any way. Not least in the medical sector, the integrity of stored data is enormously important because diagnoses and treatments are derived from them [80]. According to Zarour et al. [80] the healthcare sector needs a more robust data integrity approach to counteract the trend of increasing data integrity violations in the health sector(cf. Section 3.1). Therefore they list new approaches for data integrity management, for example, approaches which use blockchain technology to fulfill this goal [81] [82] [83].

Security Logging and Monitoring: Failures in security logging and monitoring are often a factor in major security incidents. When security events, such as login attempts, are not adequately logged, monitored, or reported, suspicious behavior is difficult to detect, and the likelihood that an attacker can successfully exploit the application increases. Huang et al. [61] propose the WAVES which, among other techniques, uses behavior monitoring to detect malicious content before it reaches users. Marty et al. [84] present a basic structure for use-case-oriented logging. It gives an answer to the questions when to log, where to log, and exactly what to log in order to maintain forensic investigation, reporting, and correlation.

Server-Side Request Forgery: If a server does not validate the input of a server-side request properly, the system could be vulnerable to Server-Side Request Forgery (SSRF) [85]. The goal of this attack is to gain access to internal server content through existing

functions using the web application. Internally, so servers that, for example, sit behind a firewall, can also be accessed [86]. Pellegrino et al. [85] tries to detect SSRF attacks with deep learning methods, or Arora et al. [87] by using extended visual cryptography and QR code. Visual cryptography is a cryptographic technique that allows for the encryption and decryption of secret information in a way that it can be visually deciphered without the use of computers or complex algorithms. It involves the use of images or patterns to conceal information, which can only be revealed by overlapping or stacking the images in a particular way.

4. Approach

After reviewing the related work, the approach of the security analysis is explained in the following sections. To determine the procedure, we use the resources of OWASP [6] (cf. Section 2.2). These are used to structure each of the following phases. The procedures of the following phases, with the exception of the scope definition, are all defined with the help of OWASP resources. First, the scope of our tests will be defined. Thereafter, we discuss how we will gather information, in which order we will use it, and finally how we use them.

4.1 Scope

To conduct a comprehensive security analysis, it is crucial to establish a clear scope of our investigation. The scope of the analysis provides a well-defined boundary of the attack surface of the target system and sets the parameters for our assessment. Without a clearly defined scope, we risk wasting valuable time and resources analyzing irrelevant systems or third-party services.

While there may be some overlap with the analysis of other sub-platforms of eSano due to the shared architecture, the primary focus of our work is to analyze the CMS system and API. By narrowing the scope of our investigation, we can ensure that our analysis is focused, efficient, and effective. This enables us to identify potential vulnerabilities in the CMS system and API, and to develop appropriate measures to mitigate the risks. Ultimately, a clearly defined scope is essential for conducting a successful and thorough security analysis.

4.2 Reconnaissance

After the scope is defined, we will begin the reconnaissance phase. This is the process of mapping out the attack surface of a target. This will be done through manual research supported by automated tools. Gathering more information is crucial as it enhances our comprehension of how the application operates, and expands the scope of places we can examine for potential vulnerabilities. We will split this phase into gathering **(1) general information** and information about **(2) the API** and **(3) the CMS**. The main part of **general information gathering** will be to gain insight into the exact structure of the eSano project and the technologies and frameworks that are used. When gathering information about the **CMS**, the main focus will be on searching subdomains and functionalities of

the CMS. A map of the interfaces with the corresponding functionalities will be created. Publicly available information about the attack target such as permissions will also be used. Searching for open ports and identifying the services running on them to get an overview will be a goal. This phase also includes the search for information which shouldn't be publicly available. Finding such could be a successful attack. A lot of information we obtain here, will also be relevant for the **general information gathering**. This also applies in the reverse direction. For **API**, we will first perform some simple scan, or surfing the CMS sub-platform to map the available API commands the CMS uses. Next, we will do some fuzzing, for example with a generated wordlists to check for more API commands and routes that were not revealed by passive scan. One goal will be to obtain a map with all API routes with their corresponding functionalities. With the help of this, it may be possible to create situations in the API that cannot be achieved with other test methods or already used input vectors. Several unknown directories and functionalities could be discovered in this way. A check of old and not used versions of the API will also be part of the information gathering of the API.

4.3 Testing Order

After documenting the attack surface, we must discuss the order in which we will test for vulnerabilities. Due to time constraints, we cannot cover the entire attack surface. Therefore, we need to rank possible vulnerabilities. We want to proceed as follows: Since special attention is paid to the protection of private data in the context of eHealth, vulnerabilities that could reveal such data will be given the highest priority. In addition, they must be part of the OWASP Web Security Testing Guide [7]. These will then be ranked according to

1. the sensitivity or importance of the data that could potentially be accessed by an attack on the respective vulnerability, and
2. the impact of the vulnerability and how easy it is to find and exploit. Here is where the OWASP Guide [7] helps us.

The created order should help us to focus on most impactful and dangerous vulnerabilities, and thus generate the greatest benefit from the pentesting project.

4.4 Testing

The tests will be performed according to the order from the list of target vulnerabilities from the previous section. They will be guided by various OWASP resources, which we described in Chapter 2.2. For each discovered vulnerability we will create a proof-of-concept (Proof of concept (PoC)). In the information security domain, a PoC involves modeling of how an exploit works with the goal of determining the potential for compromising a computer system and presenting the optimal ways to protect against it. We will perform attacks to the extent that they can be considered as proof that they work and can cause serious harm. Damage to the system or similar things are not the focus. The pentesting of the eSano project is done on a dedicated test server to ensure that the testing process does not impact the production environment. The overriding goal of all these measures is to make it as easy as possible to reproduce the attacks, especially for the developer. For each PoC, we will estimate the impact and give some recommendations. While the giving of recommendations depends very much on the individual case, for the estimation of the impact, we want to define a more precise norm. We will map each vulnerability to a Common Weakness Enumeration (CWE) and classify the impact as "critical", "medium" or "low". For this, we strongly use the resources of OWASP [7]. When a vulnerability cannot be exploited in the time allotted for this purpose, it will be skipped to not lose much time.

5. Reconnaissance

In the reconnaissance phase, we want to gather information about the eSano platform to support targeting in the subsequent testing phase. Reconnaissance can be divided into two categories based on their interaction with the target. Active reconnaissance, where we gain information through active interaction with the eSano platform, and passive reconnaissance, where we utilize publicly available information. In the following Section 5.1, we enumerate the subdomain, afterwards we scan ports as described in Section 5.2. In Section **??**, we describe if the roles and privileges we found through manual testing, are the same as the expected from the background section. in the. In the last Section 5.4, we describe how we reverse engineer the part of the API that interacts with the CMS sub-platform.

5.1 Subdomains

The eSano project uses different subdomains to make its services available. The following list obtains all subdomains we found only by using the sub-platforms:

cms.esano.pentest.klips-ulm.de
: Under this subdomain, the cms sub-platform is addressable.

ecoach.esano.pentest.klips-ulm.de
: Under this subdomain, the eCoach sub-platform is addressable.

patient.esano.pentest.klips-ulm.de
: Under this subdomain, the patient sub-platform is addressable.

api.esano.pentest.klips-ulm.de
: Under this subdomain, the API, which all sub-platforms use, is addressable.

auth.esano.pentest.klips-ulm.de
: Under this subdomain, the authentication mechanism for all sub-platforms is addressable.

phpmyadmin.esano.pentest.klips-ulm.de
: Under this subdomain, the software to handle the administration of MySQL database (phpMyAdmin [88]) is addressable.

We also used the Burp Suite repeater [33] with a common word list [89] to brute force more subdomains depending on the already known. But we could not find any further.

5.2 Port Scanning

After listing the subdomains, we observed that all referred to the same Internet Protocol (IP) address (134.60.58.14). We use Nmap [34] to scan all Transmission Control Protocol (TCP) ports of the IP address. We concentrate on the TCP ports because most of the services that are interesting for the testing phase are running on them. Interesting in this domain means that data is sent via these ports. Ports 80 and 443 where open and used for Hypertext Transfer Protocol (HTTP) and HTTPS, respectively. Port 22 was open to run Secure Shell (SSH), more precisely OpenSSH 8.2p1.

5.3 Roles and Privileges

To assess the privileges of each role in the CMS sub-platform, we performed a series of manual tests. For each role, we logged into the CMS sub-platform using a test account with the corresponding role assigned to it. We then performed various actions in the sub-platform, such as creating new content, deleting existing content, and modifying permissions of other users.

For the admin role, we tested the user management capabilities, which included assigning roles to other users, creating and deleting user accounts, and managing announcements. We found that the admin role had the highest level of access in the CMS sub-platform, with the ability to perform nearly all actions within the sub-platform.

For the editor role, we tested the ability to create and delete groups (studies), interventions, questionnaires, and diaries. We also tested the ability to assign permissions to other users, such as adding another editor to a group and assigning different levels of access. We found that editors had a moderate level of access in the CMS sub-platform, with the ability to create and modify content, but not the ability to perform certain administrative tasks.

For the eCoach role, we tested the ability to publish interventions. We found that the ecoach role had a limited level of access in the CMS sub-platform, with the ability to publish interventions but not perform other administrative tasks.

Finally, for the patient role, we tested the ability to interact with the CMS sub-platform as a patient. We found that the patient role cannot log in to the CMS sub-platform had thus have the lowest level of access.

The results of our testing provided an actual condition assessment of the privileges in the CMS sub-platform for each role. Overall they match to the expected, described in the background Section 2.1. However, since we had no documentation of the sub-platform, we were unable to determine whether these privileges are correct. This matching must be done in a further step by the developers of the eSano platform.

5.4 Functionalities and their Interfaces

Since this analysis focuses on the API and the CMS sub-platform of the eSano project, we want to give an overview of the API endpoints used by the CMS sub-platform. Due to the fact that we do not have any documentation on the API, we have to reverse engineer it. Reverse engineering an API involves analyzing and understanding how an API works and what it does, without access to its source code or official documentation. We followed these steps to do so:

1. **Observe API traffic:** We used Burp Suite [33] to capture and inspect the API requests and responses made by a client.

2. **Identify API endpoints and request methods:** We looked for patterns in the API requests and determined the URL structure, including endpoint paths and HTTP request methods (e.g., GET, POST).

3. **Examine request and response payloads:** We studied the data format used in the API requests and responses, such as JSON or Extensible Markup Language (XML). We look for any parameters or authentication information included in the request.

4. **Test the API:** We used Postman [35] to make API calls and observe the responses. We modified request parameters and tried different request methods to get a better understanding of the API.

5. **Document the API:** We documented the findings, therefore we listed each discovered API endpoint with its corresponding functionality. Additionally, we made a statement which of the two roles, existing in the CMS ("admin", "editor"), can access this endpoint with which HTTP request method ("GET, "HEAD", "PATCH", "POST", "DELETE"). The findings are listed in Listing 10.2.

This list gives us a good overview of the structure of the API in relation to the CMS. We can use it to understand the functionalities of the CMS sub-platform and, thus, to discover and exploit possible vulnerabilities more efficiently in the testing phase. Despite all this, it is possible that the list is not complete and there are more endpoints that we have not discovered yet. This could be a potential risk, because these endpoints may contain vulnerabilities or security flaws that can be exploited.

6. Security and Privacy Issues

This chapter presents discovered vunerabilities. For each of them, we give a proof of concept, estimate the impact, provide matching CWEs, and give our recommendations to fix them. The list has a descending order; in other words, the list starts with the most critical vulnerability and ends with those for which we estimate the impact to be the lowest.

6.1 Vulnerable File Upload Functionality

The CMS sub-platform contains a file upload functionality. Using the User Interface (UI), it is possible to upload an image as a (1) "group picture," (2) "intervention picture," (3) "lesson picture," or a (4) "diary picture." Furthermore, a file could be uploaded as an element of a (5) diary or (6) lesson. In Section 6.1.1, we present the possibility of bypassing the file extension restrictions. In Section 6.1.2, we show how we can exploit a XSS, SSRF, and DoS vulnerability of the uploading functionality by uploading an Scalable Vector Graphics (SVG) file. After that, in Section 6.1.3, we consider the fact that eSano does not sanitize the uploaded data.

6.1.1 Bypass File Extension Restrictions

We want to test which file extensions are allowed to be uploaded. We can trigger an error message from eSano by uploading a file with a prohibited extension (in our case, .php). This message presents the extensions that are allowed for the upload functionality of eSano. These are: .jpe, .jpg, .png, .gif, .wav, .svg, .m4a, .mp3, .mp4, .pdf, .doc, .docx, .odt, .ppt, and .pptx. After observing which extensions are allowed, we could easily check if it is possible to upload other, not allowed ones. The extensions we tested are listed in Listing 10.1. It is not possible to upload any file type which is not explicitly allowed.

To bypass this and uploading files with a not explicitly allowed extension, we tried several methods. We manage to bypass the extension restriction by manipulating the magic bytes of the files. The magic bytes are numeric or string constant that indicates the file type. This number is in the first 512 bytes of the file. They can be changed by any hex editor. For example, we changed the first four bytes from a php file (`3C 3F 70 68`) to those of a picture (`FF D8 FF E0`) with hexedit [90]. Due to time constraints, we only tried this method for html and php files. For both, the method worked, and we could upload the files. We also tested null-byte bypass (e.g., file.php%00) to alter the intended

filter logic of the application, double extension bypass (e.g., file.php.jpeg), content-type bypass (changing the content-type of the file in the POST request), client-side validation bypass (changing the file after client-side validation), and blacklisted extension bypass (e.g., file.PhP). Except for the magic bytes manipulation, no other methods worked.

6.1.2 XSS, SSRF and DoS

After checking the allowed file extensions, we want to test whether the eSano platform implements any content validations for uploaded data. An allowed extension to upload to the eSano platform is Scalable Vector Graphics (SVG). SVG is a two-dimensional vector graphic based on XML. For test purposes, we injected different payloads into the Extended Metadata Platform (Extended Metadata Platform (XMP)) of an uploaded SVG file. XMP is an Adobe metadata XML-based schema for storing image metadata. In the following, we show three cases in which we can exploit the vulnerability of file upload functionality. We demonstrate an XSS, an SSRF, and a DoS attack. We chose these three to prove that the vulnerability exists and show how this vulnerability can be exploited.

Listing 6.1: Example Payload for XSS in .svg File.

```
1 <?xml version="1.0" standalone="no"?>
2 <!DOCTYPE svg PUBLIC "-//W3C//DTD␣SVG␣1.1//EN"
3 "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
4 <svg version="1.1" baseProfile="full"
5 xmlns="http://www.w3.org/2000/svg">
6     <polygon id="triangle" points="0,0␣0,50␣50,0"
7      fill="#009900" stroke="#004400"/>
8     <script type="text/javascript">
9         alert(document.domain);
10    </script>
11 </svg>
```

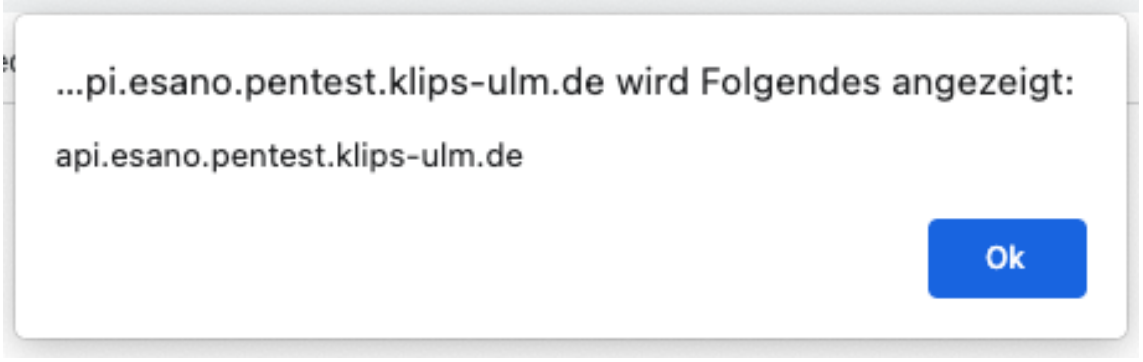

Figure 6.1: Response Message from the Application Domain to XSS Attack.

XSS means that the victim's browser executes Java Script (JS) code injected by the attacker. We uploaded and opened an SVG file that includes JS code as part of XML. Listing 6.1 shows our payload used to demonstrate that the attack is working. We inject an `alert()` method to show that the application is running the injected JS code. The response of the document domain is shown in Figure 6.1.

Since we know that the server executes the code in the SVG file, we now want to induce the server-side application to make a request to an unintended location. This is called a **SSRF** attack. The goal is to force the server to connect to internal-only services within the

sensitive infrastructure or to external systems, which could expose sensitive data such as credentials. We demonstrate the SSRF attack by requesting an image from the computer science homepage of the University of Würzburg. The payload used for this is shown in the Listing 6.2. In lines 4 and 5, we can see the destination URL to access the image. When we now access the SVG file through the UI of the platform, we see the image of the University of Würzburg. This shows that the SSRF attack works.

Listing 6.2: Example Payload for SSRF in SVG File.

```
1 <svg width="200" height="200"
2   xmlns="http://www.w3.org/2000/svg"
3   xmlns:xlink="http://www.w3.org/1999/xlink">
4   <image xlink:href="https://www.informatik.uni-wuerzburg.de/
        fileadmin/_processed_/f/8/csm_slider2_3f19ce3e27.jpg"
5   height="200" width="200"/>
6 </svg>
```

Thirdly, we want to perform a **DoS** attack on the CMS platform. For this purpose, we use an XML bomb included in the uploaded SVG file. Therefore, a number of nested XML entities are included in the payload. By opening the SVG file, the size increases rapidly because the XML entities are expanded. In our case, we use the "billion laughs attack" [91]. The file contains the lol and "lol" entities. The first entity is defined through a string and the second one through other entities. Opening this file causes a billion instances of the string "lol." Uploading this file causes a denial of service of the file upload functionality for up to ten minutes.

Listing 6.3: Example Payload for XML Bomb in SVG File (Billion Laughs Attack).

```
1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2 <!DOCTYPE svg [
3  <!ENTITY lol "lol">
4 <!ELEMENT lolz (#PCDATA)>
5 <!ENTITY lol1 "&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;
    ">
6 <!ENTITY lol2 "&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&
    lol1;&lol1;">
7 <!ENTITY lol3 "&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&
    lol2;&lol2;">
8 <!ENTITY lol4 "&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&
    lol3;&lol3;">
9 <!ENTITY lol5 "&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&
    lol4;&lol4;">
10 <!ENTITY lol6 "&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&
    lol5;&lol5;">
11 <!ENTITY lol7 "&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&
    lol6;&lol6;">
12 <!ENTITY lol8 "&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&
    lol7;&lol7;">
13 <!ENTITY lol9 "&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&
    lol8;&lol8;">
14 ]>
15 <svg>
16 <lolz>&lol9;</lolz>
17 </svg>
```

6.1.3 No File Content Sanitization of Uploads

After checking the content validation of the uploaded data, we want to test if the eSano platform sanitizes them. File content sanitization mitigates malware threats by scanning files, identifying active content, and removing active code. Finally, the file is recreated without potentially dangerous code. For our testing purpose, we used the Metasploit [36] framework to include a reverse TCP shell in the metadata of a pdf file. To verify whether the eSano platform validates and sanitizes the file, we upload it to download it again afterward. Then we could compare the input (uploaded pdf file) and the output (downloaded pdf file). After comparing them, we can conclude that there is no sanitization of uploaded files.

6.1.4 Impact

The vulnerable file upload functionality fits into the category CWE-434 (Unrestricted Upload of File with Dangerous Type) [92]. We want to classify the impact for the vulnerabilities from Section 6.1.1, 6.1.2, and 6.1.3 separately.

For the bypass of the file extension restrictions (Section 6.1.1), we consider the impact to be low, as the uploaded html or php files are not interpreted and consequently not executed by the server. It is conceivable that the eSano server interprets the files as their Multipurpose Internet Mail Extensions (MIME) type [93]. We are not able to work around this, but just the possibility of uploading files with non-permitted extensions poses a certain risk.

For the XSS, SSRF and DoS (Section 6.1.2), we consider the impact to be critical. Arbitrary code execution is possible since the uploaded files are interpreted and executed as code by the eSano server. That the vulnerability is critical can be shown with the attacks (XSS, SSRF, DoS) we have successfully performed above. XSS and SSRF can disclose sensitive patient or server data. Furthermore, DoS can cause major damage in the eHealth domain, as not using the platform can have a negative impact on patient health.

Also, for the missing file content sanitization (Section 6.1.3), we consider the impact to be critical as no sanitization of uploaded data can end in downloading dangerous files. When a patient downloads a malicious file, it may cause damage to his private computer. In our attack scenario, it could end in the remote execution of a TCP reverse shell on the patient's computer. With this, private files and unwanted actions can be executed.

6.1.5 Recommendations

We strongly recommend implementing stricter input validation for the file upload functionality. Especially because patients may download the files to work with them, the validation should include a sanitization mechanism of potentially malicious code. The attacks described in Section 6.1.2 were all performed by uploading SVG files. Therefore, we want to make a recommendation with respect to handling SVG files as input.

The easiest way to prevent attacks would be to prohibit the upload of SVG files. However, due to its ability to be scalable, open, and accessible, HTML5 specifications require web browsers to support embedding. To fulfill this requirement, Heiderich et al. [94] propose SVG purification, in which they remove all suspicious content from a given file and preserve as much content as possible. With SVGPurifier, the authors empirically improve the security of uploading SVG files on Wikipedia. We recommend including a similar approach in the eSano file upload functionality.

We also recommend editing the insecure configuration of the XML parser, which processes the uploaded files by setting an upper limit on the XML entity size and amount. With this, we could prevent the DoS of the parser and, therefore, the file upload functionality.

6.2 Misconfiguration of Cross-Origin-Resource-Sharing

CORS is an HTTP header-based mechanism. This is designed to allow domains, schemes, or ports to obtain content or use resources from foreign servers. CORS also relies on a mechanism by which browsers make a preflight request to the server hosting the cross-origin resource to verify that the server will allow the actual request. In this preflight, the browser sends headers that specify the HTTP method and headers used in the actual request.

6.2.1 CORS: Arbitrary Origin Trusted

Listing 6.4 shows the preflight request from the CMS platform to announce a GET request (line 4). The preflight server response can be seen in Listing 6.5. Line 9, "Access-Control-Allow-Credentials: true, "allows the host and the server (Access-Control-Allow-Origin) to perform GET, POST, PATCH, and DELETE operations (line 10) in the current session with the valid credentials used.

The CORS policy of the eSano project trusts arbitrary origins. In Listing 6.6, we can see a GET request from the client to *api/v1/my/profile*. The Access-Control-Allow-Origin was manipulated in an exemplary manner. This now points to the domain *https://devil.de*. Any other domain that we tried worked the same way. The server response can be seen in Listing 6.7. There we can see that *https://devil.de* as Access-Control-Allow-Origin is granted access to data about the "roles" (line 7) of the logged-in user with the current credentials (line 4). It is possible that a blacklist is implemented that blocks certain domains. Thus, it is not possible to decide with definite certainty if all domains are allowed to make cross-domain requests. All tested domains were allowed to make cross requests, so we didn't identify any restrictions.

Listing 6.4: Preflight request of client to see profiles.

```
1  OPTIONS /api/v1/my/profile HTTP/2
2  Host: api.esano.pentest.klips-ulm.de
3  Accept: */*
4  Access-Control-Request-Method: GET
5  Access-Control-Request-Headers: authorization
6  Origin: https://cms.esano.pentest.klips-ulm.de
7  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
       AppleWebKit
8  /537.36 (KHTML, like Gecko) Chrome/103.0.5060.134 Safari/537.3
       6
9  Sec-Fetch-Mode: cors
10 Sec-Fetch-Site: same-site
11 Sec-Fetch-Dest: empty
12 Referer: https://cms.esano.pentest.klips-ulm.de/
13 Accept-Encoding: gzip, deflate
14 Accept-Language: de-DE,de;q=0.9,en-US;q=0.8,en;q=0.7
```

Listing 6.5: Preflight response of eSano CMS platform for showing profiles.

```
1  HTTP/2 204 No Content
2  Server: nginx/1.19.10
3  Date: Wed, 03 Aug 2022 14:23:39 GMT
4  Cache-Control: no-cache, private
5  Access-Control-Allow-Origin:
6  https://cms.esano.pentest.klips-ulm.de
```

```
7  Vary: Origin,Access-Control-Request-Headers,
8  Access-Control-Request-Method
9  Access-Control-Allow-Credentials: true
10 Access-Control-Allow-Methods: GET, POST, PUT, PATCH, DELETE
11 Access-Control-Allow-Headers: authorization
12 Access-Control-Max-Age: 0
13 X-Frame-Options: deny
14 X-Content-Type-Options: nosniff
15 Strict-Transport-Security: max-age=31536000
```

Listing 6.6: Vulnerable GET request of client to see profiles.

```
1 GET /api/v1/my/roles HTTP/2
2 Host: api.esano.pentest.klips-ulm.de
3 ...
4 Authorization: Bearer ...
5 ...
6 Origin: https://devil.de
7 ...
```

Listing 6.7: Vulnerable response of eSano CMS platform for showing profiles.

```
1 HTTP/2 200 OK
2 ...
3 Access-Control-Allow-Origin: https://devil.de
4 Access-Control-Allow-Credentials: true
5 ...
6
7 {"data"}
```

6.2.2 Impact

The vulnerability fits into the category CWE-942 (Permissive Cross-domain Policy with Untrusted Domains) [95]. Since our case involves private patient data, we classify the impact as critical. CORS misconfiguration allows third-party websites to make privileged requests about authenticated users. Since the policy specifies the header "Access-Control-Allow-Credentials: true," third-party domains may be able to carry out privileged actions and retrieve sensitive information. Uploading malicious scripts in this way is also possible and is a serious threat. But if it is not possible to upload such scripts, it may be possible to bypass IP-based access controls by proxying through the users' browsers.

6.2.3 Recommendations

We strongly recommend not trusting every domain as Access-Control-Allow-Origin. It should not be possible to access sensitive information handled in the eSano project from there. A validation list (whitelist) should be associated with the Access-Control-Allow-Origin that identifies which specific domains can access resources. The whitelist should be straightforward to implement since it will not contain that many domains. In addition, it is worth considering not granting all HTTP request method types in the preflight (Listing 6.5 (line 10)), but only those that are absolutely necessary to implement the required functionality.

6.3 Clickjacking

During the testing of the eSano platform, we identified several pages that are vulnerable to clickjacking attacks. Clickjacking is an attack that tricks users into clicking on a button or link that performs an unintended action. This attack can be achieved by overlaying the button or link with another element, such as an invisible iframe or a disguised button. To assess the clickjacking vulnerability of eSano, we used Burp Suite clickbandit, a tool that allows us to test for clickjacking vulnerabilities. We configured the browser proxy to point to Burp Suite and intercepted a request. We then sent the request to the clickbandit and selected a clickjacking test, such as testing for framebusting or X-Frame-Options header. These would both be techniques to prevent clickjakcing. Our testing identified that clickjacking is possible on all pages of the CMS sub-platform, as we can overlay a button with an Inline Frame (iFrame) element. An iFrame is an Hypertext Markup Language (HTML) element used to embed another HTML document within the current document. It allows for the display of content from a different domain, or even a different website, within a specific area of a webpage. This could allow an attacker to perform actions on behalf of the user without their knowledge or consent.

6.3.1 Impact

This vulnerability fits into the category CWE-1021 (Improper Restriction of Rendered UI Layers or Frames) [96]. The impact of a successful clickjacking attack could be significant, particularly for an eHealth platform like eSano, which deals with sensitive medical data. The severity of the impact depends on the page that is targeted by the attack, but since pages like the one used for password management are susceptible to clickjacking, the risk for data exposure is present. It is important to note that for a successful clickjacking attack, or more generally, for all social engineering attacks, at least a human action is necessary. Additionally, it is possible that potential users of the eSano platform may not be aware of the potential risks associated with these attacks, making the probability of a successful attack not negligible. Therefore, we classify the impact of this attack as critical.

6.3.2 Recommendations

To mitigate the vulnerability of clickjacking in the eSano platform, we recommend implementing the following measures: Firstly, implementing the X-Frame-Options header can prevent the platform's content from being embedded within other websites. This header can specify which domains are allowed to be embedded on the platform, effectively blocking unauthorized domains. Secondly, implementing framebusting techniques can prevent attackers from embedding the platform's content within a malicious website. Framebusting is a technique used to prevent a website's content from being loaded within a frame on another website. This technique can be implemented using JavaScript code that detects if the website is being loaded within a frame and takes action to break out of the frame. Lastly, training end-users to recognize and avoid clickjacking attacks can help reduce the risk of a successful attack. End-users should be educated on how to identify suspicious pop-ups and unfamiliar websites and instructed to avoid clicking on them.

6.4 Difference in "Resend Verification Email" Response

To activate an account, a user must verify their email address. This first verification email is sent automatically after creating the account. When the user does not respond to this one, the user can request a further one. This functionality is available to users at *https://auth.esano.pentest.klips-ulm.de/resendVerification*. We find a difference in the server response to such an email. In this way, we can distinguish whether an email address is not used by any account in the eSano project (Listing 6.8), used by an account but is not verified (Listing 6.9), or used by an account and is already verified (Listing 6.10).

Listing 6.8: JSON response to a "Resend Verification Email" from a not existing email address.

```
HTTP/2 400 Bad Request
[...]
{
    "errors":[
        {
            "status":"400",
            "code":"4",
            "title":"No Entity Found",
            "detail":"No entity was found for this request."
        }
    ]
}
```

Listing 6.9: JSON response to a "Resend Verification Email" from an existing email address that is not verified yet.

```
HTTP/2 204 No Content
[...]
```

Listing 6.10: JSON response to a "Resend Verification Email" from an existing email address which is already verified.

```
HTTP/2 400 Bad Request
[...]
{
    "errors":[
        {
            "status":"400",
            "code":"31",
            "title":"User Already Verified.",
            "detail":"This user is already verified."
        }
    ]
}
```

6.4.1 Impact

This vulnerability fits into the category CWE-209 (Generation of Error Message Containing Sensitive Information) [97] and CWE-1230 (Exposure of Sensitive Information Through Metadata) [98]. We consider the impact critical since it reveals sensitive information about the users of the eSano platform. An attacker can query whether an email address is used by a user and thus enumerate the emails that may contain the user's name.

6.4.2 Recommendations

We generally recommend not revealing any sensitive information by error messages or server responses. In this case, the response of the server should be uniform, no matter in which context an email address is related to eSano, or if related at all..

6.5 Excessive Data Exposure from the API

It is important for an API to exclusively provide the data that is essential for the client. This implies that the client should only receive the data necessary for displaying the information in the user interface (UI). During our analysis of the eSano project's API responses, we discovered two types of data that were problematic. Firstly, some data was deemed unnecessary (1) for the client's purposes, and secondly, some data contained sensitive information that the current user should not be able to access (2). To clarify, we will now outline the specifics of unnecessary or sensitive data that we encountered in each API response.

6.5.1 Accessing Users

An admin can manage user roles. Therefore, he can view all users data like "email address", "firstname", "lastname", and "roles" ("admin", "ecoach", or "editor"). He can access these data by calling:
https://api.esano.pentest.klipsulm.de/api/v1/admin/users with a GET request. In Listing 6.11, we can see a part of the JSON response, with the definition of the "id" of the user (Line 3) and the "attributes" like "name" and "email" etc. (Line 4-21).

We can see a picture assigned to the user (Line 9). We cannot access this picture through the UI, but by calling *https://api.esano.pentest.klips-ulm.de/uploads/users/user_Id* with the corresponding "user_Id". The picture cannot be changed and is never shown in the UI. It is generated automatically when a new user is created, so its like a static "profile picture". Figure 6.2 shows the "profile picture" of the user with "user_Id" 1. Furthermore, the API endpoints in Lines 17 and 18 do not exist and thus are unnecessary.

Listing 6.11: User with "id" one in JSON representation

```
1  {
2      "type": "users",
3      "id": "1",
4      "attributes" :{
5          "name": "admin",
6          "email": "admin@local.host",
7          "firstname": "Akeem11",
8          "lastname":"Wunsch1",
9          "picture":"https:\/\/api.esano.pentest.klips-ulm.de\/
               uploads\/users\/1",
10         "is_verified":1,
11         "roles":[...]
12         },
13         "links":{...},
14         "relationships" :{
15             "codes" :{
16             "links" :{
17                 "self": "api.esano.pentest.klips-ulm.de\/\/v1
                       \/users\/1\/relationships\/codes",
18                 "related": "api.esano.pentest.klips-ulm.de\/\/
                       v1\/users\/1\/codes"
19             }
20         }
21     }
22 }
```

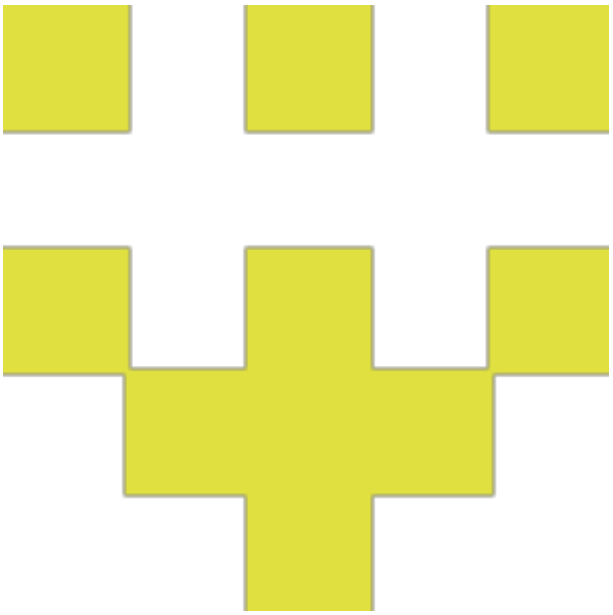

Figure 6.2: Unused profile picture of user with ”id” one.

6.5.2 Accessing Groups (Studies)

An editor can create groups. Therefore, he can add a name, a title, a description, and a picture to a group. Furthermore, he can view all other groups (groups for which he has no access). Listing 6.12 contains part of the JSON response of the API when viewing a group on which he has ”no access” by calling:
https://api.esano.pentest.klipsulm.de//api/v1/editor/studies/(studyId) with a GET request. It defines the endpoints to access ”data” (Line 2), ”links” (Line 3), and ”relationships” including ”questionnaires” (Line 5-10) ”collaborators” (Line 11-16), ”members” (Line 17-22), ”translations” (Line 23-28) and the ”metadata” (Line 31) of the group. Despite the endpoint to access the ”collaborators,” none of the above listed endpoints exist and are thus unnecessary.

Listing 6.12: Part of a Group with ”No Access” in JSON format.

```
1  {
2      "data":{...},
3          "links":{...},
4          "relationships" :{
5              "questionnaires" :{
6                  "links" :{
7                      "self": "api.esano.pentest.klips-ulm.de
                           \/\/v1\/studies\/39\/relationships\/
                           questionnaires",
8                      "related": "api.esano.pentest.klips-ulm.de
                           \/\/v1\/studies\/39\/questionnaires"
9                  }
10             },
11             "collaborators" :{
12                 "links" :{
13                     "self": "api.esano.pentest.klips-ulm.de
                           \/\/v1\/studies\/39\/relationships\/
                           collaborators",
```

```
14                      "related": "api.esano.pentest.klips-ulm.de
                           \/\/v1\/studies\/39\/collaborators"
15                  }
16              },
17              "members" :{
18                  "links" :{
19                      "self": "api.esano.pentest.klips-ulm.de
                           \/\/v1\/studies\/39\/relationships\/
                           members",
20                      "related": "api.esano.pentest.klips-ulm.de
                           \/\/v1\/studies\/39\/members"
21                  }
22              },
23              "translations" :{
24                  "links" :{
25                      "self": "api.esano.pentest.klips-ulm.de
                           \/\/v1\/studies\/39\/relationships\/
                           translations",
26                      "related": "api.esano.pentest.klips-ulm.de
                           \/\/v1\/studies\/39\/translations"
27                  }
28              }
29          }
30      },
31      "meta":{...}
32  }
```

6.5.3 Accessing Interventions

An editor can create interventions in a group. Therefore, he can add a name, a title, a description, and assign a picture for an intervention. Furthermore, he can view all other interventions (interventions from a group he has no access). Listing 6.13 shows the part of the JSON response of calling:
https://api.esano.pentest.klipsulm.de//api/v1/studies/39/interventions with a GET request. In the UI of the CMS, this would be viewing an intervention with no access. When viewing an intervention through the UI of the CMS, we cannot access the "intervention picture". But We can bypass this by combining the "id" (Line 5) and the "picture" (Line 23) in Listing 6.13 to access the picture under:
https://api.esano.pentest.klips-ulm.de/uploads/studies/57/6473a540f1d4236e12f617.jpg.
Furthermore, we have access to a "password" (Line 27). As it appears to be a hash, it is not possible to decrypt this password. However, it is unclear what this password is used for within the CMS sub-platform, or if it is even in use. If it is used to login for specific intervention in other sub-platforms, the question arises, where this password can be set or changed. Lastly, it should be noted that the API endpoints for accessing the "members" of the intervention (Lines 32, 33) do not exist and are therefore unnecessary.

Listing 6.13: Part of a Intervention with "No Access" in JSON format.

```
1  {
2      "data":[
3          {
4              "type":"interventions",
5              "id":"57",
```

```
            "attributes":{
                "default_configuration":{...},
                    "name":"Intervention1_user_2",
                    "title":"Titel Intervention1_user_2",
                    "description":"Beschreibung Intervention1
                       _user_2",
                    "text":null,
                    "consenttext":null,
                    "picture":"6473a540f1d4236e12f617.jpg",
                    "type":"intervention",
                    "intervention_type":"unaccompanied",
                    "locales":["de","en"],
                    "password":"$2y$10$gZCBiThI8qpXOfmkvfxd0.
                       WEyRZzzTXNLXUcp9uhpjzkIfm..Tmd6",
                    ...
                },
                "links":{
                    "self":"api.esano.pentest.klips-ulm.de\/\/
                       v1\/interventions\/57"
                },
                "relationships":{
                    "collaborators":{
                        "links":{
                            "self":"api.esano.pentest.klips-
                               ulm.de\/\/v1\/interventions\/57
                               \/relationships\/collaborators"
                               ,
                            "related":"api.esano.pentest.klips
                               -ulm.de\/\/v1\/interventions\/5
                               7\/collaborators"
                        }
                    },
                    "members":{
                        "links":{
                            "self":"api.esano.pentest.klips-
                               ulm.de\/\/v1\/interventions\/57
                               \/relationships\/members",
                            "related":"api.esano.pentest.klips
                               -ulm.de\/\/v1\/interventions\/5
                               7\/members"
                        }
                    }
                }
            },
        ],
        "meta":{...},
        "links":{...}
    }
}
```

6.5.4 Impact

We want to evaluate the impact of four different weaknesses that have arisen in this section: Exposure of the intervention password discussed in Section 6.5.3, the founded profile picture demonstrated in Section 6.5.1, the bypass to view ”the intervention picture” presented in Section 6.5.3, and the not existing API endpoints reported in all three sections (6.5.1 6.5.3 6.5.2)

The leaked password falls under the CWE-200 category (Exposure of Sensitive Information to an Unauthorized Actor) [99] and CWE-522 (Insufficiently Protected Credentials) [100]. If an attacker successfully cracks the password hash, they may gain unauthorized access to personal and health-related information, compromising the platform's interventions. A data breach involving a password hash can also result in regulatory penalties, reputation damage, and increased risk of future attacks. However, it is important to note that the password can only be used by a logged-in user on the eSano platform, and it is unclear for what purposes it is used. Therefore, we estimate the actual impact to be medium.

The found profile picture and the bypass to view the ”intervention picture” also fit into the category CWE-200 [99]. However, we classify the impact of these two findings as low. In the case of the ”intervention picture,” there are no reasons why this picture should not be viewable, especially because a ”group picture” could also be viewed about the UI of the CMS. For us, the ”intervention picture” and the ”group picture” are at the same level of sensitivity. The reason why the profile picture is classified as low is that it does not show sensitive content. It is generated automatically and can never be changed, so all look similar to Figure 6.2.

The impact of the not existing API endpoints is also classified as low. They do not reveal any sensitive information; it is just unnecessary that they are sent along.

6.5.5 Recommendations

For the leaked password, we generally recommend not sending it, and passwords, in general, within an API response, particularly even if it is only a hash, especially when it has no use.

For the other findings, we have no clear recommendations in therms of security, because changing them brings no security benefit. Possible recommendations would be to enable the option to view the ”intervention picture” also via the user interface of the CMS and delete the not existing API endpoints in the responses. Also, it should be discussed why the ”profile picture” functionality exists. It brings no value, since they can not be changed via the UI or directly sending a PATCH or POST request from a proxy. Not even viewing it is possible via the UI, the only possible way to do so is to access is directly with a proxy, sending a GET request. So we would recommend to delete the profile picture functionality in future releases of eSano.

6.6 No Rate Limiting

In general, rate limits are used to limit computer systems at certain points. In other words, it is intended to prevent certain actions from being executed more frequently than a parameter. If this is not implemented at important points in the system, limited resources such as Central Processing Unit (CPU), memory, and storage could be overused, and, consequently, system services can no longer be provided (denial of service). Not implementing rate limits also makes brute-force attacks possible. The rate limits could be (1) only numerical (a number is set as an upper border) or (2) numerical and temporal

(a number in a period is set as an upper border). This can be described by the classical throughput. A third method is (3) to limit the size of the uploaded object.

The following functionalities of the CMS, thus implemented API endpoints were tested on rate limits ((1), (2), or (3)), whereby the abbreviations Brute Force Attack (BFA) and DoSA are intended to indicate the type of attacks to which the respective functionality is susceptible:

i) Login attempt (BFA)

ii) Sending a password-reset e-mail to change the password (BFA)

iii) Sending verification e-mail to activate an account (DoSA)

iv) Uploading media to the ”group details” (DoSA)

v) Uploading media to the ”intervention details” (DoSA)

vi) Importing interventions as JSON file (DoSA)

vii) Changing password (BFA)

viii) Creating announcements (DoSA)

ix) Creating interventions (DoSA)

x) Creating groups (DoSA)

xi) Creating interventions (DoSA)

To automate the testing, we used Burp Suite. We tested the endpoints by performing the respective functionality 100 times in less than one minute. We considered this number realistic for the future use of the eSano platforms. To test the limit (3), we used a satellite image (54Megabyte (MB)) for media upload and a JSON file (nearly 1GB) for intervention import. In the case of the login functionality, the user was blocked for one minute after ten login attempts. After this minute, the user can make ten new login attempts, again resulting in a blocking time of one minute in which the user cannot log in. Also, after repeating this process, the possible login attempts (10) and the blocking time (1 minute) remain the same. The rest of the tests were possible without restrictions, which seems to imply that no rate limits ((1), (2), or (3)) are implemented. We also tried to manipulate the number of objects per page (e.g., users in JSON format) to return a single response. The response never changes, which leads to the acceptance that a static number of objects is set, so no manipulation is possible.

6.6.1 Impact

The vulnerability fits into the category CWE-307 (Improper Restriction of Excessive Authentication Attempts) [101] and CWE-770 (Allocation of Resources Without Limits or Throttling) [102]. The impact depends on the functionality. We can assign the missing rate limits to two risks: BFA and DoSA. The risks posed by BFA, in general, are well explored [103] and can be transferred to the risk for the eSano project. Functionalities (i), (ii), and (vii) can be assigned to be vulnerable to BFA. A statement about the risk of a successful DoS attack is more difficult. Nevertheless, if we consider requests from multiple API clients competing for resources, and the fact that the requests can easily be automated, and there is no maximum size for uploaded objects, there is a risk of a DoS of the API. The functionalities (iii), (iv), (v), (vi), (viii), (ix), (x), and (xi) are identified to be vulnerable to a DoSA. We would estimate the imppact in total to be medium.

6.6.2 Recommendations

The appropriate rate limits for an application will depend on the specific requirements and constraints of the application. In order to determine the appropriate rate limits, it is important to consider the following points:

- Monitor current usage: Gather data on the current usage of functionality to understand the typical request rate and pattern of use.
- Identify constraints: Consider the resources available on the server, such as CPU, memory, and bandwidth, and determine the maximum request rate that the server can handle without becoming overwhelmed.
- Evaluate trade-offs: Consider the trade-off between protecting the server and allowing normal use of the application. Set the rate limits high enough to allow normal use but low enough to protect the server from overuse or abuse.
- Test and adjust: Implement the rate limits and monitor the performance of the server to ensure it is functioning as expected. Make adjustments as necessary based on the results of the monitoring.

It may also be helpful to consult industry standards and best practices for rate limiting in web applications to ensure that the rate limits are set appropriately. Rate limits may need to be adjusted over time as the usage of the application changes.

Based on that, we recommend the following for the missing rate limits described in Listing 6.6: In the case of Login attempt (i), we strongly recommend increasing the blocking time with the number of failed logins and implementing an upper border (1) of login attempts, in general. Also, for Sending a password-reset e-mail to change the password (ii) and Changing password (vii), we strongly recommend implementing an upper border (1). Furthermore, there should be a maximum object size (3) for Uploading media to the "group details" (iv), Uploading media to the "intervention details" (v), and Importing interventions as JSON file (vi). For the rest of the functionalities ((iii), (viii), (ix), (x), (xi)), our recommendation would be to implement a rate limit based on throughput (2) to guarantee the accessibility of the corresponding API endpoint.

6.7 Not used Cookie

During the tests, we noticed that a cookie, "eSanoTest," is implemented. This uses the `Samesite` attribute with the value `None` and is therefore also marked as `Secure`. While testing the Cookie, it became clear that it is not used at all. It has no functionality and can be omitted. A possible reason why it is implemented could be that the developers used it for testing purposes and forgot to delete it, or they did not delete it deliberately because of not yet-implemented functionalities which need the Cookie in the future.

6.7.1 Impact

The impact of this issue is comparatively low because the Cookie is not used and does not store sensitive data directly in itself. However, it stores a link to data in the backend database. Disclosure of this via an unused cookie creates an avoidable attack vector.

6.7.2 Recommendations

We advise deleting the Cookie for the eSano project. If it is necessary for future functionalities or parts (e.g., the iOS or Android app), it can be introduced again.

7. Other Tested Attack Vectors

After providing discussions about found vulnerabilities in the previous chapter, this chapter focuses on testing vectors we didn't manage to exploit. For these, we give an explanation what we tried to exploit the vector. In the end of the chapter we additionally provide a stress test of the blacklisting mechanism of the login mechanism to identify potential for a DoSA (Section 7.5).

7.1 Testing the Security of JWT Authentication

The eSano platform uses JWT to securely authenticate and authorize users, allowing them to access the system's features and services. JWT is a widely adopted standard for securely transmitting data as a JSON object between two parties. We attempted to change the algorithm used in the JWT from Hash-based Message Authentication Code (HMAC) to Rivest–Shamir–Adleman (RSA) to see if it would result in any vulnerabilities or weaknesses. However, this change did not lead to any significant discoveries.

Next, we considered changing the algorithm to the "Null-Algorithm," which does not perform any cryptographic operations and is often used as a default value. However, we were unable to do so due to the presence of a validation key. This key is crucial for verifying the authenticity and integrity of messages, and changing to the Null-Algorithm would have compromised the system's security. Therefore, we could not proceed with this change.

We also considered changing to the "Null-Algorithm," which is a type of algorithm that does not perform any cryptographic operations and is often used as a default value. However, we were unable to do so due to a validation key. A validation key is a type of cryptographic key that is used to verify the integrity and authenticity of a message or data. In this case, the validation key was likely necessary to authenticate and verify the message, and changing to the "Null-Algorithm" would have compromised the system's security. Therefore, we were unable to proceed with this change.

In our testing, we used a tool called "gojwtcrack" to brute force the JWT. Specifically, we attempted to crack the signature portion of the JWT, which is generated using a secret key known only to the server. By brute forcing the signature, we hoped to generate a valid signature for a modified JWT, which would then be accepted by the server as a legitimate authentication token. The tool "gojwtcrack" works by generating a list of candidate secret keys, and using each key to verify the signature of the JWT. The tool does this by trying each key until a matching signature is found, at which point it outputs the corresponding

key. We used a dictionary attack to generate the candidate keys, which involved trying a list of common and likely passwords and phrases.

However, despite our efforts, we were not able to successfully brute force the JWT. This could be due to the strength of the secret key used by the server, or other security measures implemented to protect the JWT. Regardless, we were not able to find a way to exploit the JWT using brute force techniques.

7.2 Bypassing Authorization Scheme

We divided our testing into two levels: Testing the function level authorization scheme, and user level authorization scheme. A function level authorization scheme involves controlling access to specific functions or actions within a software application based on the user's authorization level. A user level authorization scheme involves controlling access to the entire software application based on the user's authorization level. Therefore, we split the tests into vertical and horizontal ones. Vertical Privilege Escalation refers to the act of gaining access to higher levels of permissions or privileges than what is authorized for a user. This can be achieved by exploiting vulnerabilities in the application or system. Horizontal Privilege Escalation, on the other hand, refers to the act of gaining access to the same level of permissions or privileges as another user, but with a different identity or role. This is often accomplished by taking advantage of insufficient access control mechanisms in the application or system.

1. **Vertical Privilege Escalation**
   We have checked whether vertical privilege escalation is possible. In essence, we verified the feasibility of accessing every function or executing every request in the application with a user holding a distinct identity and role. We tested (1) access to administrative functions while logged in as an editor and (2) editor functions while logged in as admin.

2. **Horizontal Privilege Escalation**
   We've conducted an assessment to determine the potential for horizontal privilege escalation. Specifically, we've analyzed every function and request executed by the application to determine whether it is possible for a user with a different identity but the same role and differing privileges to access it. In both cases, we found no mistake in the authorization scheme.

We have conducted extensive testing to identify any potential vulnerabilities in our authorization scheme and have not found any errors. However, to further enhance its security, we conducted tests to identify non-standard headers that could bypass the authorization mechanism, including "X-Original-URL" and "X-Rewrite-URL". These headers can override the target URL and proxy requests, potentially compromising the authorization scheme. Additionally, we tested for headers that can reveal the client's IP address, including "X-Forwarded-For", "X-Remote-IP", "X-Originating-IP", "X-Remote-Addr", and "X-Client-IP", which could be used to determine if certain administrative rights are available only within the server's local network. To prevent unauthorized access, the authorization scheme must verify the original and target URLs to ensure they are authorized. Moreover, the aforementioned headers can be spoofed, and therefore, the authorization scheme needs to be designed to verify the client's IP address using more secure methods, such as client certificates or other authentication mechanisms. We were unable to identify any non-standard headers that could bypass the authorization mechanism during our testing. We also did not find any headers that could reveal the client's IP address.

7.3 Vulnerable and Outdated Components

The eSano project is based on several open source components. For these components, there are known vulnerabilities with documented attack instructions. We tested the components for these vulnerabilities using the metasploit framework [36]. We were unable to find any outdated or vulnerable components.

Furthermore we want to discuss the found vulnerabilities from the pentesting performed from Tobias Sterns [45] in 2021 and if they still exist:

1. Tobias Sterns found a misconfiguration in the CORS protocol. This misconfiguration still exists and is discriped in Section 6.2.
2. The second vulnerability Tobias Sterns found, was that no protection against DoSA was implemented. After our testing we could say that the level of protection against DoSA is still to low. There are still several vectors where such attacks can be executed. Examples would be the file upload functionality described in Section 6.1.2, the functionalities vulnerable to DoSA described in Section 6.6, or the authentication mechanism in the following Section 7.5.
3. The last vulnerability Tobias Sterns mentioned in his report was ”that the web application server is leaking information via one or more “X-Powered-By” HTTP response headers”. After our testing, we could say that the application server of the eSano project does not reveal information by using the “X-Powered-By” HTTP response header any more. Nevertheless, the server still leaking information through the ”server” HTTP response header. With the command ”server_tokens off”, this could be disabled for security purpose.

7.4 SQL Injection in Input Fields

SQL is a popular database language that is used to build data structures in relational databases and to edit and query data sets based on them. In the case of SQL Injection (SQLi), the goal is to exploit vulnerabilities in the source code of the software by injecting SQL commands into the program through input masks. We tested the CMS for SQLi vulnerabilities at every point for user input. For this, we tested whether there is a different response from the input fields if we inject SQL commands instead of ”normal input”. The ”normal input” depends on the type of input field. Therefore, we used SQLi payloads from SecLists [89] and automated testing using Burp Suite [33]. We could not find any conspicuous deviations in the answers, and so the tested input fields are secure against SQLi under the used payloads.

7.5 Stress Tests to Identify Potential for DoSA

While testing the platform we noticed that the eSano platform implements a blacklisting mechanism for the login tokens. The platform instance we test is hosted on a test server with few registered users, so the blacklisting mechanism works well and the login and logout time is sufficiently fast. Since the platform will later be used by many more users, the scalability of the implemented functions to many more users plays a major role. To test this behavior we automate the login and logout process by using Apache JMeter [104]. JMeter performs the following procedure automatically: It (1) logs into the CMS sub-platform, (2) views the profile of the logged in user, and (3) logs out of the CMS sub-platform. After each iteration of this procedure, the login token is blacklisted and a new token must be created for the next login. We performed this procedure 100.000 times and tracked the elapsed time (the time from just before sending the request to just after the last response has been

received [105]) for (1), (2), and (3) separately. To make sure the measurements are valid and don't suffer from, e.g., bad Internet connection, we have carried out the experiment a second time from a different location and with a different computer. The first execution is called Test 1, the second Test2. To illustrate the results we compute a moving average. More precise, for each value p_n out of all measured times $T_{mov} = \{p_n \mid 1 \leq n \leq 99.000\}$, we compute:

$$SMA_k = \frac{p_n + p_{n+1} + p_{n+2} + ... + p_{n+k-1} + p_{n+k}}{k}, \ with \ k = 1000.$$

The Figures 7.1, 7.2, and 7.3 show the results of the measurements for (1),(2), (3) respectively. Table 7.5 shows the average elapsed time for each execution (Test 1, Test2) divided in four quarter (T_1-T_4). In the following we discuss the observations of the measurements:

1. The measured elapsed time drops rapidly at the beginning and end of each plot. This can be attributed to the ramp-up period of JMeter. The ramp-up period tells JMeter how long to take to "ramp-up" to the full number of threads chosen. In our test scenario we ramp up in one second to 100 threads. In other words, during the first second, every 10 millisecond a new thread starts and after the first second all 100 threads are working. The removal of the threads at the end follows the same principle. As a consequence, the measured elapsed time increases and decreases strongly at the beginning and at the end, because of the increasing (from 1 to 100) and decreasing (from 100 to 1) number of parallel threads running.

2. The elapsed time to login (Figure 7.1) increases with the amount of login requests. In Test 1, the average time increases from 1107.35ms in the first 25.000 logins, to 1302.24ms in the last 25.000 logins (Table 7.1). This means an increase of 17,60%. In Test 2, the average time increases from 1132.60ms in the first 25.000 logins, to 1290.61ms in the last 25.000 logins (Table 7.1). This means an increase of 13,95%.

3. The elapsed time to view the profile (Figure 7.2) remains relatively constant. In Test 1, the average time decreases from 763.04ms in the first 25.000 logins, to 728.26ms in the last 25.000 logins (Table 7.1). This means an decrease of 4,56%. In Test 2, the average time decreases from 746.39ms in the first 25.000 logins, to 726.50ms in the last 25.000 logins (Table 7.1). This means an decrease of 2,66%.

4. The elapsed time to logout (Figure 7.3) increases with the amount of logout requests. In Test 1, the average time increases from 821.61ms in the first 25.000 logouts, to 954.84ms in the last 25.000 logouts (Table 7.1). This means an increase of 16,22%. In Test 2, the average time increases from 795.28ms in the first 25.000 logouts, to 982.09ms in the last 25.000 logouts (Table 7.1). This means an increase of 23,49%.

Average elapsed time in ms				
Test 1 (1), Test 2 (2)	T_1 (first quarter)	T_2 (second quarter)	T_3 (third quarter)	T_4 (fourth quarter)
Login(1)	1107.35	1208.45	1297.64	1302.24
Login(2)	1132.60	1213.55	1315.31	1290.61
Viewing Profile(1)	763.04	747.42	740.89	728.26
Viewing Profile(2)	746.39	747.95	739.05	726.50
Logout(1)	812.61	888.19	952.15	954.84
Logout(2)	795.28	906.06	996.01	982.09

Table 7.1: $T_1 = \{p_n \mid 1 \leq 25.000\}$, $T_2 = \{p_n \mid 25.001 \leq n \leq 50.000\}$, $T_3 = \{p_n \mid 50.001 \leq n \leq 75.000\}$, $T_4 = \{p_n \mid 75.001 \leq n \leq 100.000\}$

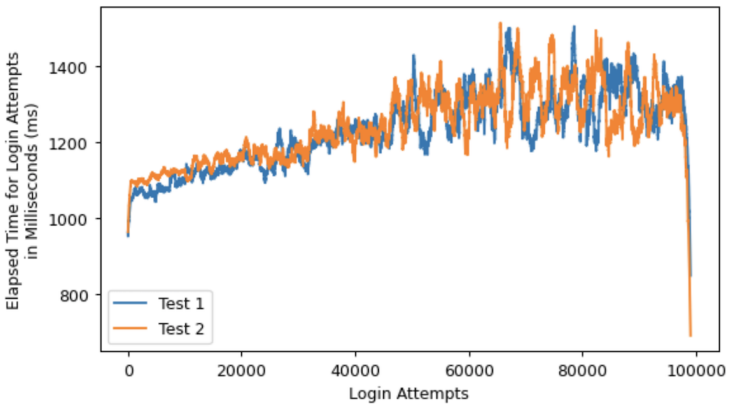

Figure 7.1: Measured times for login to the CMS sub-platform (1).

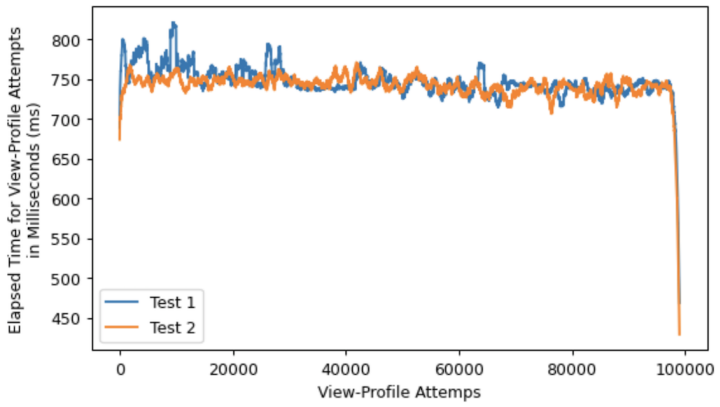

Figure 7.2: Measured times for viewing the profile in the CMS sub-platform (2).

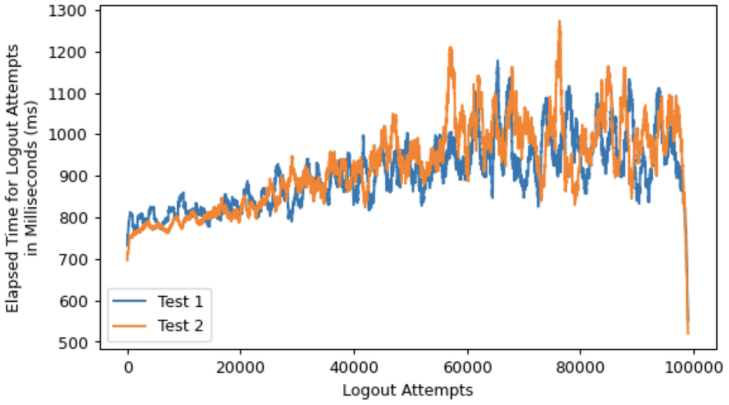

Figure 7.3: Measured times for logout to the CMS sub-platform (3).

Our analysis of the login and logout mechanisms of eSano revealed that the time it takes to log in or out increases with the number of attempts, while the time for performing other actions within the website remains constant. The implemented token blacklist can contribute to slower login and logout times as the website needs to check the blacklist for each attempted login or logout. The blacklist could be an effective method to prevent malicious activity, but it can also result in slower response times for legitimate users, which can lead to frustration and decreased user satisfaction. Additionally, slow login times can create opportunities for attackers to launch DoSA, which can result in website downtime and lost revenue. Since the goal of the eSano platform is to be used by many more users, we recommend investigating the reason for the observed behavior within a code review. The results also show, that there are no security measures such as rate-limiting or CAPTCHA challenges [106] implemented. These are designed to prevent BFA and protect against unauthorized access. Would these be implemented it wouldn't be possible to automate the login and logout like we did in the testing and use automation for launching DoSA.

8. Conclusion

We conducted a comprehensive security analysis of the eSano eHealth platform in four stages, namely defining the scope, performing reconnaissance, constructing the testing order, and finally executing the tests. Our main area of focus was the security of sensitive user data. The subsequent sections detail our findings and propose recommendations for future work.

8.1 Results

During a penetration testing of the eSano platform, multiple vulnerabilities were identified with varying degrees of severity. The low and medium impact issues included unused endpoints, functionalities, and missing rate limits at certain points, which may not pose an immediate threat but can potentially be exploited in the future. However, the critical vulnerabilities found are more concerning and require immediate action.

One of the critical issues discovered during the penetration testing is a clickjacking attack. This type of attack can trick users into performing unintended actions by hiding malicious content behind legitimate content. This means that a user could accidentally perform an action that they did not intend to, such as changing his password. So Clickjacking attacks can be used to compromise user data or steal personal information. This vulnerability must be addressed immediately. Another critical vulnerability discovered in the eSano platform was the ability to query if an email address was related to a platform user. This vulnerability could potentially lead to further attacks by an attacker who already has access to some user information. For instance, the attacker could use this information to perform a phishing attack or a brute force attack on the platform login page. A misconfiguration in the CORS protocol was also discovered during the penetration testing. This vulnerability allowed third-party websites to make privileged requests about authenticated users, leading to potential data leakage and unauthorized access to sensitive information. Furthermore, we were able to bypass the file extension restrictions of the eSano platform's file upload functionality, resulting in successful XSS, SSRF, and DoS attacks. This could result in compromised user data, unauthorized access to sensitive information, and potential denial of service.

Finally, we described all unsuccessful attack attempts, including bypassing the JWT authentication, evading the authorization scheme, and SQLi. It is important to note that

failed attempts do not necessarily mean that the identified vulnerabilities cannot be exploited by attackers. The fact that the attempts failed only indicates that the functionality was secure in the context of our testing.

In summary, the critical vulnerabilities found during the penetration testing of the eSano platform pose a significant threat to the security and privacy of users' data. It is of utmost importance that these vulnerabilities are remediated without delay to prevent possible breaches of confidential information and harm to the platform's reputation.

8.2 Future Work

Our penetration testing of the eSano eHealth platform has identified several potential security issues that need to be addressed. First and foremost, we recommend that the identified vulnerabilities be addressed in a timely manner to ensure the safety and privacy of patients' personal health information.

In addition, we recommend that the platform be subjected to regular security assessments and updates to ensure that any new vulnerabilities are identified and addressed promptly. Given the evolving threat landscape and the constant emergence of new attack vectors, we recommend that eSano establishes a comprehensive and ongoing security testing program, at least within every update, to ensure that the platform remains secure over time.

Furthermore, our login and logout stress tests identified performance issues that need to be investigated further. By analyzing the results of these tests, eSano can identify the root causes of these performance issues and take steps to optimize the platform's login and logout processes. This will not only improve the user experience for patients and healthcare providers, but also enhance the security of the platform by reducing the risk of authentication-related vulnerabilities.

List of Figures

Listings

List of Tables

9. Acronyms

ENISA European Union Agency for Cybersecurity's

EU European Union

CEN European Committee for Standardization

iFrame inline frame

EEA European Economic Area

GDPR General Data Protection Regulation

IoT Internt of Things

IMIs Internet and mobile-based interventions

CMS Content management system

API Application programming interface

DBIS Institute of Database and Information Systems

eHealth electronic health

OWASP Open Web Application Security Project

PoC Proof of concept

URL Uniform Resource Locator

HTTPS Hypertext Transfer Protocol Secure

REST Representational State Transfer

JSON JavaScript Object Notation

CSRF Cross-Site-Request-Forgery

XXE External entities XML

XSS Cross-Site Scripting

SQL Structured Query Language

ASVS Application Security Verification Standard

NIST National Institute of Technologies

ISO International Organization for Standardization

CIS Center for Internet Security

TLS Transport Layer Security

HIPAA Health Information Portability and Accountability Act

ONC Office of the National Coordinator

HIT Health Information Technology

pentesting Penetration testing

GAEN Google/Apple Exposure Notification

TC TraceCORONA

EHRs Electronic Health Records

APT Advanced Persistent Threat

BAC Broken Access Control

IoT Internet of Things

GAP Google/Apple Proposal

JS Java Script

SQLIA SQL Injection Attacks

SIEM Security Information and Event Management

Nmap Network Mapper

CWE Common Weakness Enumeration

CWA Corona-Warn-App

HTTP Hypertext Transfer Protocol

IP Internet Protocol

SSH Secure Shell

TCP Transmission Control Protocol

UI User Interface

SSRF Server-Side Request Forgery

CORS Cross-Origin-Resource-Sharing

DoS Denial of Service

SVG Scalable Vector Graphics

XML Extensible Markup Language

XMP Extended Metadata Platform

MIME Multipurpose Internet Mail Extensions

BFA Brute Force Attack

CPU Central Processing Unit

DoSA Denial of Service Attack

GB Gigabyte

MB Megabyte

JWT JSON Web Token

HMAC Hash-based Message Authentication Code

RSA Rivest–Shamir–Adleman

SQLi SQL Injection

HTML Hypertext Markup Language

iFrame Inline Frame

10. Appendix

10.1 Tested File Extensions

- .txt (text file)
- .pdf (Portable Document Format)
- .doc, .docx (Microsoft Word document)
- .xls, .xlsx (Microsoft Excel spreadsheet)
- .ppt, .pptx (Microsoft PowerPoint presentation)
- .jpg, .jpeg (JPEG image)
- .png (Portable Network Graphics)
- .gif (Graphics Interchange Format)
- .bmp (Bitmap image)
- .tif, .tiff (Tagged Image File Format)
- .zip (compressed archive file)
- .rar (compressed archive file)
- .tar (compressed archive file)
- .mp4, .avi, .wmv (video file)
- .mp3, .wav (audio file)
- .js (JavaScript file)
- .html, .htm (HTML file)
- .php (PHP script file)
- .asp, .aspx (Active Server Pages)
- .jsp (Java Server Pages)
- .py (Python script file)
- .rb (Ruby script file)
- .exe (Executable file)
- .msi (Windows Installer package)
- .dll (Dynamic Link Library file)
- .jar (Java Archive file)
- .sh (Shell script file)
- .bat (Batch file)
- .cmd (Windows Command file)
- .ps1 (PowerShell script file)
- .vbs (Visual Basic script file)
- .reg (Windows Registry file)
- .svg (Scalable Vector Graphics)
- .eps (Encapsulated PostScript)
- .ai (Adobe Illustrator)
- .sketch (Sketch file)
- .indd (Adobe InDesign)
- .cdr (CorelDRAW file)
- .dxf (AutoCAD Drawing Interchange Format)
- .dwg (AutoCAD Drawing Database)
- .max (3D Studio Max)
- .obj (Wavefront OBJ)
- .fbx (Autodesk FBX)
- .blend (Blender 3D)
- .stl (STereoLithography file)

10.2 API Endpoints

https://api.esano.pentest.klips-ulm.de/api/v1/auth/login This endpoint provides the login functionality. With a POST request and the corresponding login credentials in JSON format, editors and admins can login to the CMS sub-platforms and get their access token.

https://api.esano.pentest.klips-ulm.de/api/v1/auth/logout This endpoint provides the logout functionality. With a DELETE request, editors and admins can delete the actual session and blacklist the used access token.

https://api.esano.pentest.klips-ulm.de/api/v1/auth/refresh This endpoint provides the token-refresh functionality. For security purposes, access tokens are valid for a limited period of time. Once they expire, client applications can use a refresh method to "refresh" the access token. In other words, it's a credential artifact that allows a client application to get new access tokens without having to ask the user to log in again. Besides expiring the tokens, we could also invoke the mechanism by manipulating the access token. The mechanism uses a POST request.

https://api.esano.pentest.klips-ulm.de/api/v1/auth/refresh/assign/register

https://api.esano.pentest.klips-ulm.de/api/v1/auth/password/reset/instructions This endpoint provides the functionality to reset the password. An admin or an editor can send a POST request with their email.

https://api.esano.pentest.klips-ulm.de/api/v1/my/roles An admin or an editor can view their personal roles. Therefore, they can perform GET or HEAD requests.

https://api.esano.pentest.klips-ulm.de/api/v1/my/profile An admin or an editor can view their personal user information ("email", "roles", "username", "firstname", "lastname") with a GET or HEAD request. Additionally, they can edit their "username", "firstname", or "lastname" with a PATCH request. The DELETE request is also enabled, but we did not find specific functionality for this at this endpoint.

https://api.esano.pentest.klips-ulm.de/api/v1/my/profile/password An admin or an editor can edit their password. Therefore, they can perform a PATCH request.

https://api.esano.pentest.klips-ulm.de/api/v1/announcements An admin or an editor can view the announcements for the eSano sub-platforms. They can perform GET or HEAD requests.

https://api.esano.pentest.klips-ulm.de/api/v1/admin/announcements An admin can edit, delete, and create the announcements. Therefore, he can perform PATCH, DELETE, and POST requests.

https://api.esano.pentest.klips-ulm.de/api/v1/admin/roles An admin can edit the roles of the other users ("ecoach", "editor", "admin"). Therefore, he can perform PATCH requests.

https://api.esano.pentest.klips-ulm.de/api/v1/admin/users An admin can view and delete the other users. Therefore, he can perform GET, HEAD, and DELETE requests.

https://api.esano.pentest.klips-ulm.de/api/v1/admin/auth/assign/register An admin can register new users. Therefore, he can perform a POST request.

https://api.esano.pentest.klips-ulm.de/api/v1/admin/delete/requests An admin can view the "account-delete requests". Therefore, he can perform GET or HEAD requests.

https://api.esano.pentest.klips-ulm.de/api/v1/editor/interventions An editor can view interventions for which he is the owner or collaborator and create new interventions. Therefore, he can perform GET, HEAD, and POST requests.

https://api.esano.pentest.klips-ulm.de/api/v1/editor/questionnaires An editor can view, delete, and edit questionnaires for which he is the owner or collaborator. Therefore, he can perform GET, HEAD, DELETE, and PATCH requests.

https://api.esano.pentest.klips-ulm.de/api/v1/editor/interventions/intervention_id/questionnaires

An editor can view and create questionnaires within the intervention specified in the endpoint path ("intervention_id"). Therefore, he can perform GET or HEAD, and POST requests.

https://api.esano.pentest.klips-ulm.de/api/v1/editor/interventions/noncollaborating An editor can view the "other interventions" (the interventions not with him as collaborator or owner). Therefore, he can perform GET or HEAD requests. PATCH and DELETE requests are enabled for this endpoint, but it is not possible to delete or edit any of the interventions.

https://api.esano.pentest.klips-ulm.de/api/v1/editor/questionnaires/questionnaire_id/in/workshop/copy

An editor can copy questionnaires within a group (study). This means that the source intervention from which the questionnaire is copied is in the same group (study) as the target intervention in which the questionnaire will be copied. Therefore, he has to be the owner or collaborator of the intervention and the group (study). This functionality can be used by performing a POST request with a type description of the copied entity (here: "questionnaires"), the id of the intervention ("intervention_id") into which the questionnaire is to be copied, and the position ("position") at which the questionnaire should appear in it.

https://api.esano.pentest.klips-ulm.de/api/v1/editor/questionnaires/questionnaire_id/to/workgroup/copy

An editor can copy questionnaires for which he is the owner or collaborator. Therefore, he can perform a POST request with a type description of the copied entity (here: "questionnaires"), the id of the intervention ("intervention_id") into which the questionnaire is to be copied, and the position ("position") at which the questionnaire should appear in it. This endpoint is used if the group (study) of the source intervention is not the same as the group (study) of the target intervention.

https://api.esano.pentest.klips-ulm.de/api/v1/editor/questionnaires/positions An editor can edit the positions of questionnaires within their intervention. Therefore, he can perform a PATCH request. GET or HEAD requests are also enabled, but we found no specific functionality for them at this endpoint.

https://api.esano.pentest.klips-ulm.de/api/v1/editor/studies An editor can view the groups (studies) for which he is the owner or collaborator. Therefore, they can perform a GET or a HEAD request.

https://api.esano.pentest.klips-ulm.de/api/v1/editor/studies An editor can view all groups (studies) with a GET or HEAD request. If he is the owner of the study that can be specified in the endpoint path, he can delete the study with a DELETE request.

https://api.esano.pentest.klips-ulm.de/api/v1/editor/studies/study_id/copy An editor can copy a group (study) for which he is the owner or collaborator. Therefore, he can perform a POST request with a type description of the copied entity (here: "studies") and the id of the group ("study_id") which should be copied.

https://api.esano.pentest.klips-ulm.de/api/v1/editor/studies/noncollaborating An editor can view the other groups (studies) (the groups not with him as collaborator).

Therefore, he can perform a GET or a HEAD request. PATCH and DELETE requests are enabled for this endpoint, but it is not possible to delete or edit any of these groups (studies).

https://api.esano.pentest.klips-ulm.de/api/v1/editor/interventions/intervention_id/to/workgroup/copy

An editor can copy interventions for which he is the owner or collaborator. Therefore, he can perform a POST request with a type description of the copied entity (here: "interventions") and the id of the group ("study_id") into which the intervention should be copied.

https://api.esano.pentest.klips-ulm.de/api/v1/editor/users An editor can view all other editors. Therefore, he can perform GET or HEAD requests.

https://api.esano.pentest.klips-ulm.de/api/v1/editor/diaries An editor can view and create diaries. Therefore, he can perform GET, HEAD, and POST requests. He can edit and delete diaries for which he is the owner or collaborator. To do that, he has to specify the diary in the endpoint path and perform a PATCH or DELETE request.

https://api.esano.pentest.klips-ulm.de/api/v1/studies/stdy_id/collaborators An editor can view the collaborators of the study with the specific "study_id" defined in the endpoint path. Therefore, he can perform GET and PATCH requests.

https://api.esano.pentest.klips-ulm.de/api/v1/studies An editor can view all groups (studies) and create new ones. Therefore, he can perform GET, HEAD, and POST requests. If he is the owner of the study that can be specified in the endpoint path, the PATCH request is enabled but does not bring any changes if we use id.

https://api.esano.pentest.klips-ulm.de/api/v1/studies/id/interventions An editor can view all interventions of the group (study) specified in the endpoint path. Therefore, he can perform GET or HEAD requests.

https://api.esano.pentest.klips-ulm.de/api/v1/studies/id/media An editor can view, upload, and delete media of a group. Therefore, he has to be the owner or collaborator of it. He can perform GET, HEAD, POST, and DELETE requests.

https://api.esano.pentest.klips-ulm.de/api/v1/studies/id/relationships/collaborators An editor can edit the user permission ("owner", "editing rights", "copy right", "no access") for the studies which he is the owner. Therefore, he can perform POST requests to grant a user permission, or DELETE requests to revoke permission.

https://api.esano.pentest.klips-ulm.de/api/v1/diaries/study/id An editor can view all diaries of a group (study). Therefore, he can perform GET or HEAD requests.

https://api.esano.pentest.klips-ulm.de/api/v1/questionnaires/questionnaire_id/elements

An editor can create new or replace existing elements of a questionnaire with POST requests. Therefore, he has to be the owner or collaborator of the questionnaire specified in the endpoint path ("questionnaire_id").

https://api.esano.pentest.klips-ulm.de/api/v1/editor/questionnaires/questionnaire_id/elements
An editor can view the elements of the questionnaire specified in the endpoint path ("questionnaire_id"). Therefore, he can perform GET or HEAD requests.

https://api.esano.pentest.klips-ulm.de/favicon.ico This endpoint provides the favicon icon.

https://api.esano.pentest.klips-ulm.de/uploads/studies/study$_id/$ At this endpoint, the uploaded media of the studies is stored.

https:/api.esano.pentest.klips-ulm.de/uploads/users/ At this endpoint, a "profile picture" for each user is stored.

GET/api/v1/interventions/intervention_id/translation/status An editor can view the translation status of the interventions, soecified in the endpoint path. Therefore, he can perform GET or HEAD requests.

https://api.esano.pentest.klips-ulm.de/api/v1/misc/locales This endpoint provides the language locales, the eSano sub-platforms provide.

Bibliography

[1] D. D. Ebert, T. Van Daele, T. Nordgreen, M. Karekla, A. Compare, C. Zarbo, A. Brugnera, S. Øverland, G. Trebbi, K. L. Jensen, *et al.*, "Internet-and mobile-based psychological interventions: applications, efficacy, and potential for improving mental health," *European Psychologist*, 2018.

[2] A. Barak, L. Hen, M. Boniel-Nissim, and N. Shapira, "A comprehensive review and a meta-analysis of the effectiveness of internet-based psychotherapeutic interventions," *Journal of Technology in Human services*, vol. 26, no. 2-4, pp. 109–160, 2008.

[3] G. Andersson, P. Cuijpers, P. Carlbring, H. Riper, and E. Hedman, "Guided internet-based vs. face-to-face cognitive behavior therapy for psychiatric and somatic disorders: a systematic review and meta-analysis," *World psychiatry*, vol. 13, no. 3, pp. 288–295, 2014.

[4] P. Cuijpers, A. Kleiboer, E. Karyotaki, and H. Riper, "Internet and mobile interventions for depression: Opportunities and challenges," *Depression and anxiety*, vol. 34, no. 7, pp. 596–602, 2017.

[5] "eSano - e-behavioral & e-mental-health." `https://esano.klips-ulm.de/de/`. (Accessed on 06/20/2022).

[6] "OWASP foundation | open source foundation for application security." `https://owasp.org/`. (Accessed on 05/15/2022).

[7] "OWASP web security testing guide version 4.2." `https://rnpg.ir/Documents/WSTG-V4.2.pdf`. (Accessed on 05/15/2022).

[8] "OWASP top 10." `https://owasp.org/Top10/`. (Accessed on 05/15/2022).

[9] "OWASP api security top 10." `file:///Users/moritzschumacher/Downloads/owasp-api-security-top-10.pdf`. (Accessed on 05/24/2022).

[10] "CWE - common weakness enumeration." `https://cwe.mitre.org/`. (Accessed on 03/12/2023).

[11] "It-plattform." `https://esano.klips-ulm.de/de/it-plattform/`. (Accessed on 05/15/2022).

[12] "Rfc 7159: The javascript object notation (json) data interchange format." `https://www.rfc-editor.org/rfc/rfc7159`. (Accessed on 07/06/2022).

[13] "Angular." `https://angular.io/`. (Accessed on 05/15/2022).

[14] "Cross-platform mobile app development: Ionic framework." `https://ionicframework.com/`. (Accessed on 05/15/2022).

[15] "Vue.js - the progressive javascript framework | vue.js." `https://vuejs.org/`. (Accessed on 05/15/2022).

[16] "Laravel - the php framework for web artisans." https://laravel.com/. (Accessed on 05/15/2022).

[17] R. Kraft, A. R. Idrees, L. Stenzel, T. Nguyen, M. Reichert, R. Pryss, and H. Baumeister, "eSano–an ehealth platform for internet-and mobile-based interventions," in *2021 43rd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, pp. 1997–2002, IEEE, 2021.

[18] M. A. Tariq, F. Arif, and I. Ullah, "Security analysis of laravel framework," *International Journal of Computer Science and Security (IJCSS)*, vol. 12, February 2018.

[19] M. K. Bagwan and P. S. Ghule, "A modern review on laravel-php framework," *IRE Journals*, vol. 2, no. 12, pp. 1–3, 2019.

[20] I. Vanderlei, J. Araujo, R. Rocha, G. Silva, F. Pacheco, and J. Dantas, "Analysis of laravel framework security techniques against web application attacks," in *2021 16th Iberian Conference on Information Systems and Technologies (CISTI)*, pp. 1–7, IEEE, 2021.

[21] "Security for web services: Standards and research issues journal article | igi global." https://www.igi-global.com/gateway/article/full-text-pdf/37388&riu=true. (Accessed on 05/17/2022).

[22] P. Syverson, "A taxonomy of replay attacks [cryptographic protocols]," in *Proceedings The Computer Security Foundations Workshop VII*, pp. 187–191, IEEE, 1994.

[23] P. Salini and S. Kanmani, "Elicitation of security requirements for e-health system by applying model oriented security requirements engineering (mosre) framework," in *Proceedings of the Second International Conference on Computational Science, Engineering and Information Technology*, pp. 126–131, 2012.

[24] "Microsoft word - OWASP application security verification standard 4.0-en.docx." https://owasp.org/www-pdf-archive/OWASP_Application_Security_Verification_Standard_4.0-en.pdf. (Accessed on 09/14/2022).

[25] "Secure software development framework (ssdf) version 1.1: Recommendations for mitigating the risk of software vulnerabilities." https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-218.pdf. (Accessed on 09/14/2022).

[26] "Iso/iec 27034-5:2017(en), information technology — security techniques — application security — part 5: Protocols and application security controls data structure." https://www.iso.org/obp/ui/#iso:std:iso-iec:27034:-5:ed-1:v1:en. (Accessed on 09/14/2022).

[27] "Cis critical security control 16: Application software security." https://www.cisecurity.org/controls/application-software-security. (Accessed on 09/14/2022).

[28] S. Turner, "Transport layer security," *IEEE Internet Computing*, vol. 18, no. 6, pp. 60–63, 2014.

[29] J. Viega and J. Epstein, "Why applying standards to web services is not enough," *IEEE Security & Privacy*, vol. 4, no. 4, pp. 25–31, 2006.

[30] E. U. A. for Cybersecurity, "Baseline security recommendations for iot in the context of critical information infrastructures." https://www.enisa.europa.eu/publications/baseline-security-recommendations-for-iot, 2019.

[31] E. Parliament and Council, "General data protection regulation." https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32016R0679, 2016.

[32] E. C. for Standardization, "En 13606: Health informatics - electronic health record communication." https://www.cen.eu/work/areas/ict/eHealth/Pages/EN13606.aspx, 2008.

[33] "Burp suite - application security testing software - portswigger." https://portswigger.net/burp. (Accessed on 11/21/2022).

[34] "Nmap: the network mapper - free security scanner." https://nmap.org/. (Accessed on 06/22/2022).

[35] "Postman api platform | sign up for free." https://www.postman.com/. (Accessed on 02/05/2023).

[36] "Metasploit | penetration testing software, pen testing security | metasploit." https://www.metasploit.com/. (Accessed on 10/09/2022).

[37] A. Act, "Health insurance portability and accountability act of 1996," *Public law*, vol. 104, p. 191, 1996.

[38] H. K. Patil and R. Seshadri, "Big data security and privacy issues in healthcare," in *2014 IEEE international congress on big data*, pp. 762–765, IEEE, 2014.

[39] I. C. Cucoranu, A. V. Parwani, A. J. West, G. Romero-Lauro, K. Nauman, A. B. Carter, U. J. Balis, M. J. Tuthill, and L. Pantanowitz, "Privacy and security of patient data in the pathology laboratory," *Journal of pathology informatics*, vol. 4, no. 1, p. 4, 2013.

[40] "Datenschutz in der eu | eu-kommission." https://ec.europa.eu/info/law/law-topic/data-protection/data-protection-eu_de. (Accessed on 06/21/2022).

[41] "Orientierungshilfe zum gesundheitsdatenschutz." https://www.bmwk.de/Redaktion/DE/Downloads/M-O/orientierungshilfe-gesundheitsdatenschutz.pdf?__blob=publicationFile&v=16. (Accessed on 06/21/2022).

[42] K. Zimmermann, "Security analysis of uninow app," 2021.

[43] L. Baumgärtner, A. Dmitrienko, B. Freisleben, A. Gruler, J. Höchst, J. Kühlberg, M. Mezini, R. Mitev, M. Miettinen, A. Muhamedagic, *et al.*, "Mind the gap: Security & privacy risks of contact tracing apps," in *2020 IEEE 19th international conference on trust, security and privacy in computing and communications (TrustCom)*, pp. 458–467, IEEE, 2020.

[44] F. Roos, "Security and privacy aspects of digital contact tracing," master's thesis, University of Würzburg, 2021.

[45] T. Sterns, "University of ulm software-engineering project eSano," 2021.

[46] K. Kandasamy, S. Srinivas, K. Achuthan, and V. P. Rangan, "Digital healthcare-cyberattacks in asian organizations: An analysis of vulnerabilities, risks, nist perspectives, and recommendations," *IEEE Access*, vol. 10, pp. 12345–12364, 2022.

[47] "Phishing attack on uc davis health breaches data on 15,000 patients | healthcare it news." https://www.healthcareitnews.com/news/phishing-attack-uc-davis-health-breaches-data-15000-patients. (Accessed on 07/31/2022).

[48] "Cybercrime jumps more than 50% in 2019, new threats emerge from covid-19 pandemic - cna." https://www.channelnewsasia.com/singapore/cybercrime-jumps-more-than-50-2019-new-threats-covid-19-csa-662381. (Accessed on 07/31/2022).

[49] "Nsa uncovers ties between north korea and wannacry attacks | healthcare it news." https://www.healthcareitnews.com/news/nsa-uncovers-ties-between-north-korea-and-wannacry-attacks. (Accessed on 07/31/2022).

[50] "December 2019 healthcare data breach report." https://www.hipaajournal.com/december-2019-healthcare-data-breach-report/. (Accessed on 08/02/2022).

[51] A. H. Seh, M. Zarour, M. Alenezi, A. K. Sarkar, A. Agrawal, R. Kumar, and R. Ahmad Khan, "Healthcare data breaches: Insights and implications," *Healthcare*, vol. 8, no. 2, 2020.

[52] R. Gafni and T. Pavel, "Cyberattacks against the health-care sectors during the covid-19 pandemic," *Information & Computer Security*, 2021.

[53] "Who reports fivefold increase in cyber attacks, urges vigilance." https://www.who.int/news/item/23-04-2020-who-reports-fivefold-increase-in-cyber-attacks-urges-vigilance. (Accessed on 07/31/2022).

[54] "Governments must stop cyber attacks on health care | icrc." https://www.icrc.org/en/document/governments-work-together-stop-cyber-attacks-health-care. (Accessed on 07/31/2022).

[55] "Cybersecurity in the healthcare sector during covid-19 pandemic — enisa." https://www.enisa.europa.eu/news/enisa-news/cybersecurity-in-the-healthcare-sector-during-covid-19-pandemic. (Accessed on 07/31/2022).

[56] D. Yadav, D. Gupta, D. Singh, D. Kumar, and U. Sharma, "Vulnerabilities and security of web applications," in *2018 4th International Conference on Computing Communication and Automation (ICCCA)*, pp. 1–5, IEEE, 2018.

[57] M. Curphey and R. Arawo, "Web application security assessment tools," *IEEE Security & Privacy*, vol. 4, no. 4, pp. 32–41, 2006.

[58] J. Bau, E. Bursztein, D. Gupta, and J. Mitchell, "State of the art: Automated black-box web application vulnerability testing," in *2010 IEEE symposium on security and privacy*, pp. 332–345, IEEE, 2010.

[59] A. Jaiswal, G. Raj, and D. Singh, "Security testing of web applications: Issues and challenges," *International Journal of Computer Applications*, vol. 88, 01 2014.

[60] Y.-W. Huang, C.-H. Tsai, T.-P. Lin, S.-K. Huang, D. Lee, and S.-Y. Kuo, "A testing framework for web application security assessment," *Computer Networks*, vol. 48, no. 5, pp. 739–761, 2005.

[61] Y.-W. Huang, S.-K. Huang, T.-P. Lin, and C.-H. Tsai, "Web application security assessment by fault injection and behavior monitoring," in *Proceedings of the 12th international conference on World Wide Web*, pp. 148–159, 2003.

[62] H. Wang, "A new algorithm for euclidean shortest paths in the plane," in *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pp. 975–988, 2021.

[63] S. Jan, O. B. Tauqeer, F. Q. Khan, G. Tsaramirsis, A. Ahmad, I. Ahmad, I. Maqsood, and N. Ullah, "A framework for systematic classification of assets for security testing," *Comput. Mater. Contin.*, vol. 66, no. 1, pp. 631–645, 2021.

[64] R. Montesino, S. Fenz, and W. Baluja, "Siem-based framework for security controls automation," *Information Management & Computer Security*, 2012.

[65] M. Hassan, M. Ali, T. Bhuiyan, M. Sharif, and S. Biswas, "Quantitative assessment on broken access control vulnerability in web applications," in *International Conference on Cyber Security and Computer Science 2018*, 2018.

[66] A. Ouaddah, H. Mousannif, A. Abou Elkalam, and A. A. Ouahman, "Access control in the internet of things: Big challenges and new opportunities," *Computer Networks*, vol. 112, pp. 237–262, 2017.

[67] S. G. Akl and P. D. Taylor, "Cryptographic solution to a problem of access control in a hierarchy," *ACM Transactions on Computer Systems (TOCS)*, vol. 1, no. 3, pp. 239–248, 1983.

[68] D. Lazar, H. Chen, X. Wang, and N. Zeldovich, "Why does cryptographic software fail? a case study and open problems," in *Proceedings of 5th Asia-Pacific Workshop on Systems*, pp. 1–7, 2014.

[69] C. Anley, "Advanced sql injection in sql server applications," 2002.

[70] W. G. Halfond and A. Orso, "Combining static analysis and runtime monitoring to counter sql-injection attacks," in *Proceedings of the third international workshop on Dynamic analysis*, pp. 1–7, 2005.

[71] S. Schechter, A. B. Brush, and S. Egelman, "It's no secret. measuring the security and reliability of authentication via "secret" questions," in *2009 30th IEEE symposium on security and privacy*, pp. 375–390, IEEE, 2009.

[72] A. Naiakshina, A. Danilova, C. Tiefenau, M. Herzog, S. Dechand, and M. Smith, "Why do developers get password storage wrong? a qualitative usability study," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 311–328, 2017.

[73] J. Chen, J. Jiang, H. Duan, T. Wan, S. Chen, V. Paxson, and M. Yang, "We still {Don't} have secure {Cross-Domain} requests: an empirical study of {CORS}," in *27th USENIX Security Symposium (USENIX Security 18)*, pp. 1079–1093, 2018.

[74] B. Eshete, A. Villafiorita, and K. Weldemariam, "Early detection of security misconfiguration vulnerabilities in web applications," in *2011 Sixth International Conference on Availability, Reliability and Security*, pp. 169–174, IEEE, 2011.

[75] "OWASP dependency-check | OWASP foundation." `https://owasp.org/www-project-dependency-check/`. (Accessed on 09/06/2022).

[76] M. Cadariu, E. Bouwers, J. Visser, and A. van Deursen, "Tracking known security vulnerabilities in proprietary software systems," in *2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, pp. 516–519, IEEE, 2015.

[77] X. Zhan, L. Fan, S. Chen, F. We, T. Liu, X. Luo, and Y. Liu, "Atvhunter: Reliable version detection of third-party libraries for vulnerability identification in android applications," in *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, pp. 1695–1707, IEEE, 2021.

[78] K. El Emam, E. Jonker, L. Arbuckle, and B. Malin, "A systematic review of re-identification attacks on health data," *PloS one*, vol. 6, no. 12, p. e28071, 2011.

[79] M. Raza, M. Iqbal, M. Sharif, and W. Haider, "A survey of password attacks and comparative analysis on methods for secure authentication," *World Applied Sciences Journal*, vol. 19, no. 4, pp. 439–444, 2012.

[80] M. Zarour, M. Alenezi, M. T. J. Ansari, A. K. Pandey, M. Ahmad, A. Agrawal, R. Kumar, and R. A. Khan, "Ensuring data integrity of healthcare information in the era of digital health," *Healthcare Technology Letters*, vol. 8, no. 3, pp. 66–77, 2021.

[81] N. J. Vickers, "Animal communication: when i'm calling you, will you answer too?," *Current biology*, vol. 27, no. 14, pp. R713–R715, 2017.

[82] S. Bowman, "Impact of electronic health record systems on information integrity: quality and safety implications," *Perspectives in health information management*, vol. 10, no. Fall, 2013.

[83] M. Zarour, M. T. J. Ansari, M. Alenezi, A. K. Sarkar, M. Faizan, A. Agrawal, R. Kumar, and R. A. Khan, "Evaluating the impact of blockchain models for secure and trustworthy electronic healthcare records," *IEEE Access*, vol. 8, pp. 157959–157973, 2020.

[84] R. Marty, "Cloud application logging for forensics," in *proceedings of the 2011 ACM Symposium on Applied Computing*, pp. 178–184, 2011.

[85] G. Pellegrino, O. Catakoglu, D. Balzarotti, and C. Rossow, "Uses and abuses of server-side requests," in *International Symposium on Research in Attacks, Intrusions, and Defenses*, pp. 393–414, Springer, 2016.

[86] B. Jabiyev, O. Mirzaei, A. Kharraz, and E. Kirda, "Preventing server-side request forgery attacks," in *Proceedings of the 36th Annual ACM Symposium on Applied Computing*, pp. 1626–1635, 2021.

[87] N. Arora, P. Singh, S. Sahu, V. K. Keshari, and M. Vinoth Kumar, "Preventing ssrf (server-side request forgery) and csrf (cross-site request forgery) using extended visual cryptography and qr code," in *Proceedings of Second International Conference on Smart Energy and Communication*, pp. 215–227, Springer Singapore, 2021.

[88] "phpmyadmin." `https://www.phpmyadmin.net/`. (Accessed on 01/10/2023).

[89] "Github - danielmiessler/seclists: Seclists is the security tester's companion. it's a collection of multiple types of lists used during security assessments, collected in one place. list types include usernames, passwords, urls, sensitive data patterns, fuzzing payloads, web shells, and many more.." `https://github.com/danielmiessler/SecLists`. (Accessed on 09/22/2022).

[90] "Hexedit(1)." `http://rigaux.org/hexedit.html`. (Accessed on 10/11/2022).

[91] C. Späth, C. Mainka, V. Mladenov, and J. Schwenk, "{SoK}:{XML} parser vulnerabilities," in *10th USENIX Workshop on Offensive Technologies (WOOT 16)*, 2016.

[92] "CWE - CWE-434: Unrestricted upload of file with dangerous type (4.9)." `https://cwe.mitre.org/data/definitions/434.html`. (Accessed on 11/16/2022).

[93] E. Levinson, "The mime multipart/related content-type," tech. rep., 1998.

[94] "Crouching tiger - hidden payload: security risks of scalable vectors graphics." https://dl.acm.org/doi/pdf/10.1145/2046707.2046735?casa_token=AHd5UQ64U8sAAAAA:YLhB6Rgm2kTYrMsugKwk_HDsAEAojKeGWWsSfvPoQqOvRSYzaPcubEGF1wikUmbV0ReI7Y4lA1Kszg. (Accessed on 11/20/2022).

[95] "CWE - CWE-942: Permissive cross-domain policy with untrusted domains (4.9)." https://cwe.mitre.org/data/definitions/942.html. (Accessed on 12/08/2022).

[96] "CWE - CWE-1021: Improper restriction of rendered ui layers or frames (4.10)." https://cwe.mitre.org/data/definitions/1021.html. (Accessed on 02/25/2023).

[97] "CWE - CWE-209: Generation of error message containing sensitive information (4.9)." https://cwe.mitre.org/data/definitions/209.html. (Accessed on 12/01/2022).

[98] "CWE - CWE-1230: Exposure of sensitive information through metadata (4.9)." https://cwe.mitre.org/data/definitions/1230.html. (Accessed on 12/01/2022).

[99] "CWE - CWE-200: Exposure of sensitive information to an unauthorized actor (4.9)." https://cwe.mitre.org/data/definitions/200.html. (Accessed on 11/30/2022).

[100] "CWE - CWE-522: Insufficiently protected credentials (4.9)." https://cwe.mitre.org/data/definitions/522.html. (Accessed on 11/30/2022).

[101] "CWE - CWE-307: Improper restriction of excessive authentication attempts (4.9)." https://cwe.mitre.org/data/definitions/307.html. (Accessed on 11/21/2022).

[102] "CWE - CWE-770: Allocation of resources without limits or throttling (4.9)." https://cwe.mitre.org/data/definitions/770.html. (Accessed on 11/21/2022).

[103] L. R. Knudsen and M. J. B. Robshaw, *Brute Force Attacks*, pp. 95–108. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011.

[104] "Apache jmeter - apache jmeter™." https://jmeter.apache.org/. (Accessed on 01/08/2023).

[105] "Apache jmeter - user's manual: Glossary." https://jmeter.apache.org/usermanual/glossary.html. (Accessed on 01/09/2023).

[106] J. Yan and A. S. El Ahmad, "Usability of captchas or usability issues in captcha design," in *Proceedings of the 4th symposium on Usable privacy and security*, pp. 44–52, 2008.

I declare that I have developed and written the enclosed thesis completely by myself, and have not used sources or means without declaration in the text.

Würzburg, 15. March 2023

.....

(Moritz Schumacher)