Hochschule Wismar
University of Applied Sciences
Technology, Business and Design
Fakultät für Ingenieurwissenschaften, Bereich EuI

Master's Thesis

Conception and Assessment of a System for Usage of Biometric Sensors in Smartphones to Authenticate against Remote Servers

Handed in on: August 31st, 2019

by: Daniel Köhler
born on 12.05.1996
in Berlin

Reviewer: Prof. Dr.-Ing. Matthias Kreuseler
Secondary Reviewer: Dr. Feng Cheng

ACKNOWLEDGEMENT

I would first like to thank my thesis advisor Prof. Dr.-Ing Matthias Kreuseler of the University of Wismar. The contact to Prof. Kreuseler was always available whenever I needed it. Even though I put a lot of pressure upon him he always seeked to provide me with educated feedback and input. While he constantly aimed to steer my advancements into the right direction, he always ensured that it this paper stays my own work.

I would further like to thank Erik Klieme from the Hasso-Plattner-Institute. Being an expert in the area of authentication technologies, his feedback always proved valuable. Due to his passion for the topic and my work I could always count on his insights, thoughts and ideas.

I would also like to acknowledge Alexander Mühle, who helped me tremendously providing me with infrastructure to set up my own implementations outlined in this thesis.

Furthermore I would like to thank Prof. Dr. Christoph Meinel, Dr. Feng Cheng and the other researchers at HPI for welcoming me open-minded and warm-hearted and allowing me to focus my time working as a research associate upon finishing this work.

Finally, I must express my very profound gratitude to my parents and to my girlfriend for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them.

Thank you.

Daniel Köhler

Aufgabenstellung

Das Ziel der Masterthesis ist die Konzeption und Analyse eines Systems zur Nutzung von biometrischen Sensoren in Smartphones um Authentifizierungen an externen Systemen durchzuführen. Um dieses Ziel umsetzen zu können sollen zugrundeliegende Thematiken aufbereitet und erklärt werden. Weiterhin wird eine Analyse der Funktionsweise und der Anforderungen an das zu entwickelnde System durchgeführt. Hierbei soll auf die Interessen möglicher Stakeholder, vor allem jedoch auf sicherheitsrelevante Thematiken eingegangen werden. Darauffolgend soll ein Prototyp eines Systems zusammengestellt werden, welcher Anwendungsfälle des Systems erfüllen kann. Anhand dieses Prototyps soll eine Überprüfung auf die Nutzbarkeit des Systems erfolgen. Weiterhin sollen die unterschiedlichen Komponenten und Lösungsmöglichkeiten für die verschiedenen Herausforderungen analysiert werden. Hierbei sollen insbesondere bei mehreren Möglichkeiten der Umsetzung Vor- und Nachteile erörtert und analysiert werden. Dies soll ein Kernpunkt der Thesis werden und mit möglichen Erkenntnissen beispielsweise im Vergleich zu bereits existierenden Systemen neue Ideen zur Verbesserung der Sicherheit bringen.

Abstract

The aim of this work is to conceptualize and assess a system which uses biometric sensors from smartphones to authenticate against remote systems. To achieve this target, initially, underlying technologies are introduced and explained. Further, an analysis of the functioning of other competitor systems is supposed to be performed. This assessment is to be performed especially with a focus on security-relevant aspects. Based on the evaluated criteria, a prototype is to be set-up and using that prototype the usability of the system shall be assessed. Further, an analysis of the different components shortcomings and possible enhancements or advancements shall be performed. Based on the results, first ideas for possible future scenarios to enable systems to move past the current limitations should be sparked.

Contents

1 Introduction

1.1 Motivation

People enhance their lives every day using current technologies. Smartphones are getting increasingly intelligent and constantly aiming to solve our daily problems more efficiently. One of the problems users face daily is how they secure and protect our belongings. Both - the physical ones as well as virtual ones such as images, data, and passwords. With the influence of the internet getting stronger every day, people tend to double the number of internet user accounts we use every five years. [1] With every new account a user creates, he [1] has the burden of choice: reuse an old password or create a new one which needs to be remembered.

A solution for the password problem could be biometric authentication. For years, biometric security systems have already found an application in the field of company security. Be it a biometric access control, special privileged accesses or the startup of a computer. However for several years, biometric security systems have barely found applications in the end-user's daily lives. This is on one side due to skepticism from the users. On the other side, it has been due to the lack of affordability of the required sensors. Luckily, technology has rapidly evolved over the last years. It was only back in 2013 when Apple first introduced a fingerprint sensor in the iPhone 5s. Since then biometric sensors in mobile devices have scaled up and scaled out. Almost every new phone that is for sale today has some kind of biometric sensor ranging from fingerprint to face to iris sensors. [2] This development enables technology companies to rethink the possibilities of biometric authentication systems for everyone.

Unfortunately, some ecosystems cannot use biometric sensors to authenticate users directly. One example of those might be an online store such as Amazon. To be able to authenticate against those systems without the hassle of passwords, companies

[1] For the sake of improved fluency and improved legibility, this work omits the additional formulation of text passages in the female form. The exclusively male formulation shall be interpreted as genderless.

like e.g. Samsung, Apple, or Google have come up with password safes on the phones that are biometrically secured and are automatically used when browsing to the specific website. from your mobile phone. However, one problem that those systems face is that the passwords have to be stored somewhere on the phone. This leads to those systems still being prone to the same attacks and vulnerabilities as most common password security systems.

1.2 State of the Art & Problems

Social networks, industry, and the expanding use of technology in every aspect of our lives lead to authentication systems everywhere. Currently, there are plenty of different authentication systems familiar to everyone. Be it passwords for social networks, Personal Identification Number (PIN)s to withdraw money, transaction numbers (TANs) for online banking, One-Time-Passcodes (OTP) to secure online accounts, physical membership cards to allow entrance to the gym or even our voice to identify ourselves when calling a relative. Those forms of authentication are possible ways to verify who we are or that we belong to a certain group.

To perform user authentication in Information Technology (IT) systems, there are many possibilities which have been developed in the last years. Those range from remembering something (*Knowledge*) over keeping something with you (*Possession*) to being something (*Biometrics).* The main difficulty we face today is implementing those systems in a way that they are *secure*. While *secure* on one hand concerns the security of the accessed system, but further the security of the user's identity. As research from the last years has shown that *Knowledge*-based or *Possession*-based security mechanisms have their disadvantages, companies and applications try to shift towards the usage of *Biometric* authentication systems. The primary problem of biometric systems is that once an individual steals a user's biometric information, he has the ability to impersonate him. While this is possible with other systems as well, the consequences of stolen biometric information can be more damaging. Everybody is used to changing their password, but how would one change his fingerprint?

1.3 Objective of this Work

limitations arise, when one wants to use biometric authentication for a system which you can not access directly. Such as a Web Application. This work researches in

the question of how a biometric authentication for any online service - e.g. Amazon - would look.

This work aims to analyze what research or applications already exist for that specific use-case. As previously explained, compromised biometric information could have a lasting consequences for people. Therefore, security considerations when using biometric systems for remote authentication shall be assessed and evaluated. Different possibilities or applications to perform the task at hand then need to be analyzed and compared. Finally, suggestions for a secure remote biometric authentication system are to be proposed.

1.4 Structure of this Work

This work starts off by introducing the reader into the underlying technologies and areas of interest. Such are basic understandings in the area of authentication methods (*Section 2.2*) as well as a brief excursion into the theories around Zero-Knowledge-Proofs. With this work's focus on biometric authentication, possible attacks on biometric systems are outlined as well as state-of-the-art technologies and theories to secure biometric information are presented. In *Section 2.3*, this thesis analyzes Google's Android Operating System (OS) for mobile devices and highlights it's security features.

During *Section 3*, possible applications for remote authentications, currently present on the market, are highlighted and explained. *Section 3.1.2* focusses on the authentication framework *FIDO2*. Using the basic understanding from the other applications, *Section 3.2* proposes security requirenments for biometric authentication systems. *Section 3.3* features an exemplary implementation of one remote biometric authentication system.

Section 4 introduces a framework for the *Comparative Evaluation of Authentication Schemes*. The framework has originally been proposed in the work of *Bonneau et al.* in 2012. The presented framework is analyzed and assessed, finishing up by proposing extensions and adaptations to the original framework in *Section 4.2*. The aim of those adaptations is, to be able to perform a better assessment with a focus on remote biometric authentication systems. Further, they shall make up for the advancements in technology and enable better usability and easier understanding for current applications.

Section 5.1 applies the presented and adapted framework to the originally introduced applications from the market. In-depth security assessments are performed for the presented applications. *Section 5.2* presents the results of the assessment and analyzes them to help the reader understand the most important aspects. Finally, *Section 5.3* explaines an exemplary attack against the previously installed system using the according Android application.

Section 6 provides a conclusion of the work. This chapter highlights the most important aspects of this work. Further, brief ideas for possible solutions to the shortcomings of the presented applications are highlighted. Finally, *Section 7* presents more ideas on how to presume research in the area of remote biometric authentication systems.

2 Basic Technologies and Concepts

The upcoming chapters provide basic knowledge and understanding of the areas to be investigated in this work. To start off, this paper describes the origin, development, and perspective of passwords.

2.1 A brief History of Passwords

Nowadays, passwords are mainly used for security aspects. This is being reflected as people often give advice such as *Make sure to secure your account with a strong password*. Hence, one might imagine passwords as a digital key which is used to lock or unlock something. To understand the original use of passwords, the origin of passwords is explored.

Tracing down the origins quickly leads to the name of *Fernando Corbató*. Mr. Corbató has been the head of the Compatible Time-Sharing System (CTSS) Program at the Massachusetts Institute of Technology (MIT) from 1960 onward. During that time, a *computer* could only be used by one user after the other. During the nights, the machines would often not be used. Therefore, the CTSS program was supposed to identify possibilities to utilize to computers computing powers in the best possible way. This ultimately lead to the problem that a computer system would need to be created which could be used by multiple users at once. However, creating such a system meant that multiple users would be accessing the same system at the same time. Everybody with his own files and programs. Corbató explained that "Putting a password on for each individual user as a lock seemed like a very straightforward solution". [3] Corbató hence decided to use the passwords to identify a user and thus secure his private documents. However, there would have been alternatives at hand. To understand the differentiations, the following sections provide an introduction into the different areas of authentication methods.

2.2 Authentication Methods

During mankind's history, various forms and methods of authentication have evolved. Some systems dating back up to and farther than the Romans having used *passwords* to differentiate friend from foe. The upcoming sections provide an understanding of the basic authentication concepts that are still common and applied in modern IT systems. The top-level differentiation between different methods of authentications is usually performed by the different *factors* that are used for the authentication. The main three factors of authentication are: knowledge, object (property) and biometrics. [4]

2.2.1 Knowledge-Based Authentication

Knowledge-based authentication describes the process of verifying one's identity by proving to have the knowledge of *something*. This knowledge can be of any kind. The most common ones are passwords, PINs and/or knowledge for security questions such as "*What was the name of your first pet*". The main characteristics of a proper knowledge-based authentication (KBA) factor are its obscurity and secrecy. For the upcoming paragraphs, this work is going to use the term *password* referring to all KBA factors.

The flaws of those however lie inside the design. Just as a bystander may listen to a secret shared in a conversation, the password gets less and less secret while it is being used. Therefore, researchers have come up with possibilities to enable a more secure way of communication using passwords.

Zero-Knowledge Proofs

In cryptography, a common request is to prove knowledge of *something*. This might be a PIN, passphrase, password or any other piece of information or secret. Once being asked to prove the knowledge of that information, presenting the information to the requesting party would be the straightforward option to do so. However, this bears the danger of malicious parties being able to capture the information and thus compromising the secrecy of the information. To be able to do so, in 1989, Shafi Goldwasser, Silvio Micali and Charles Rackoff proposed a first **zero-knowledge proof (ZKP)** system. In their paper, they propose a system which allows one party to test the other party's knowledge of certain secret by challenging them to

a multitude of identical tests. The special aspect is that during the challenges, the *secret* itself is not being transmitted.

A ZKP is always assessed between two parties. Commonly those parties are referred to as *Peggy* who is trying to *prove* her knowledge of a secret to *Victor* who is trying to *verify* Peggy's knowledge. Once *Peggy* initiates a ZKP with *Victor*, she will be challenged by him. If *Peggy* knows the secret, she will be able to provide the correct answer to *Victor*. However, if she does not know the secret, she will fail to provide the answer with a probability p. If *Peggy* has passed the first challenge, *Victor* will challenge her again. As previously, if *Peggy* is aware of the secret, she will pass the test. however, if she is trying to cheat *Victor*, she will fail with the same probability p. This challenge-and-response testing is repeated for n times. After n challenges, the probability of a cheating *Peggy* having successfully passed all tests (S) will be $S = (1 - p)^n$. Even if the probability of a cheating *Peggy* failing is only 10% per challenge, her chance for passing the test after 10 challenges will be $S = (1 - 0.1)^{10} = 0.35 = 35\%$. In realistic scenarios, the probability of a cheating *Peggy* failing a single challenge will usually be *50%* (c.f. e.g *Discrete Log of a given Value* [5] or *Hamilton Cycle for a Large Graph* [6]). Thus, even for small iterations such as $n = 10$, the probability of a cheating *Peggy* being able to pass the test shrinks to $S = (1 - 0.5)^{10} = 0.001 = 0.1\%$. [7]

Per definition, a proper ZKP adheres to the following properties [8]:

- **Completeness** If the protocol is executed correctly, an honest *Peggy* will always be able to convince an honest *Victor*.
- **Soundness** If the protocol is executed correctly, no cheating *Peggy* can convince *Victor* besides with a minimal probability.
- **Zero-Knowledge** In a properly executed protocol, no party learns anything besides about the honesty of the *prover*.

Passwords

However, it has been shown by research over the last half-century that there are and will always be flaws in password security. [9] Further, a password needs to be remembered. If there is a multitude of *secret* (i.e. complex) passwords which need to be remembered, users will get frustrated and will start doing either of the following options: [10]

- change passwords less often
- reuse old passwords or use passwords for multiple services
- use simple passwords to be capable of remembering them
- write them down

All of the above options lead to a single result: insecurity in the KBA mechanism. To remove the need of remembering something from the user, further authentication methods such as the object-based authentication (OBA) have been created.

2.2.2 Object-Based Authentication

OBA mechanisms use a physical device/item to verify the identity of a person. Hence they are characterized by physical possession. The most common example of OBA from everyday life is the use of a key to open a door. In this case, the key is the physical object which verifies the user by the fact that the user possesses it. Therefore, the main flaw of the authentication factor of possession is to be found in its design again. Possession equals authentication. Therefore if a key was lost, whoever found it afterward simply needs to find the correct lock. [4]

The next step in advancing security mechanisms is to enhance the *key* (or any other possessed object) to make it impossible to loose it. To achieve this, a key-like aspect has to be found that is always connected to the person it belongs to.

2.2.3 Biometric Authentication

Today, terms such as fingerprint or face-recognition are widely used and aren't questioned. However, this has not always been like it is. Earliest examples of biometric features have been found as early as 200 Years before Christ in China. In ancient records, details about handprints being used as evidence have been found. First theories about identifying people from their friction ridge skin (today's fingerprints) has been identified from the 14th century in Persia. [11]

The term *Biometric* derives from the greek words *Bios* (eng: *Life*) and *Metron* (eng: *Measure*). Therefore, biometric authentitcation is a technique for the authentication and identification of persons according to specific body features. [12]

The fingerprint as a matter of identification has been widely used over the last centuries. The first fingerprint repository for that cause was created in 1904, called the

National Bureau of Criminal Identification. Advancements in the field of encoding fingerprints were invented in 1914 by the Denmark Police. Being able to encode the fingerprints enabled transferral of the information for identification through electronic communications. [13]

Today's biometric systems consist of four main modules. [14]

Enrollment Unit This module is used to initially enroll new users to the biometric database. To register the users, their biometric data is scanned by a biometric reader. Using the individual biometric characteristics, a digital representation - the so-called biometric template - is being created and stored.

Feature Extraction Unit This unit is being used to extract the biometric features from the physical or digital representation of the biometric characteristic. The process of extracting features happens during enrollment to store the compact representation of the biometric identifier - the template - to a database. The Feature Extraction is additionally used in the process of authenticating or verifying a user.

Matching Unit The Matching Unit compares the extracted features from a live input to the features stored in the biometric template in the database. If the system is performing a verification, the Matching Unit will calculate a *Match Value*. The Match Value describes the probability of the current data being identical to the stored template and thus the probability of correctly identifying the user.

Decision Maker This module inherits the authority of accepting or rejecting an issued user authentication or verification based on the Matching Score. Depending on the rigidity of the implemented security mechanism and on the security needs of the application/system, a threshold will be defined. Depending on that threshold in comparison to the Matching Score, the Decision Maker will accept or reject the user.

Attacks on Biometric Security Systems

During the process of a Biometric Authentication, there are multiple points of attack that would lead to a compromise of the system's security. (c.f. *Figure 2.1*) All of

the following manipulations would either authorize a fraud or deny the service to a legitimate user. [14]

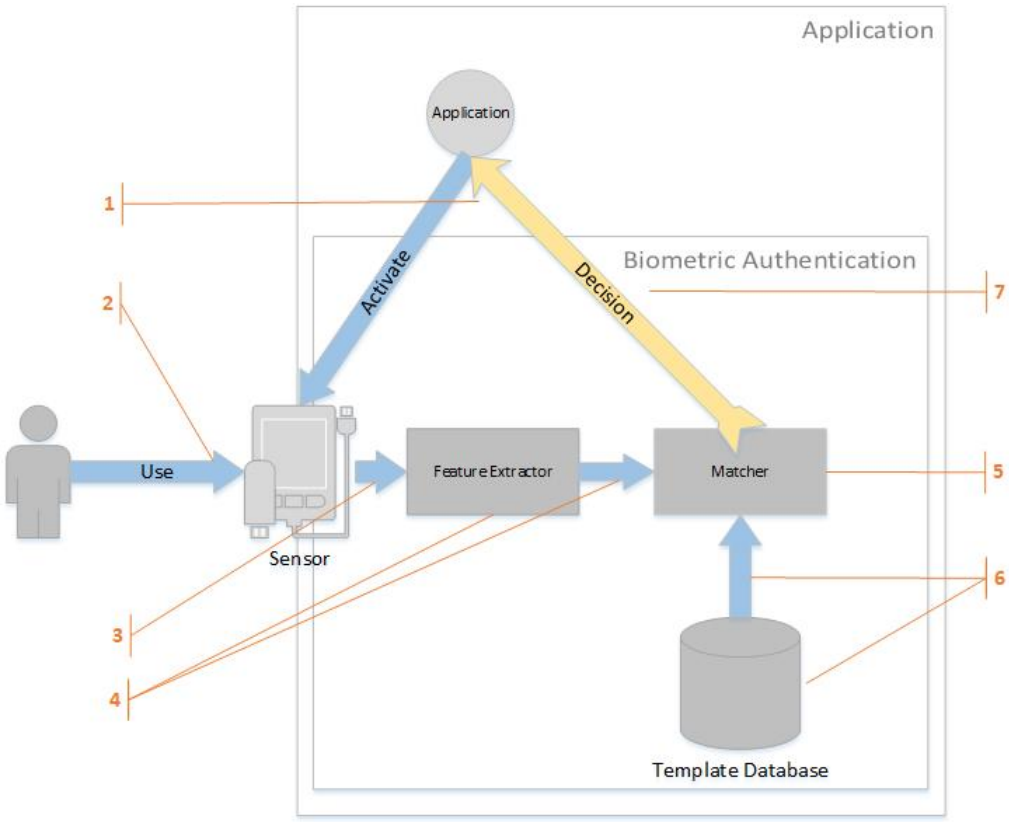


Figure 2.1: Common Attack Factors on Biometric Systems (adapted from [15])

Attack on the Secure System (1) A first possibility for an attack against biometric systems might lie outside of the system itself but rather in the integration of the system. If the system is not secured against e.g. tampering from the outside, that system is prone to attacks of all other sorts rather then just the ones describes in the following paragraphs. One example might be that the execution flow of a certain program could be directed around the biometric system, not using it at all.

Attack on the Sensor (2) One basic attack against Biometric Authentication Systems is the use of a spoofed biometric treat. Examples of such would be a faked fingerprint or a faked iris printed on a contact lens which is presented to the respective sensor. To defend against those attacks, liveness-detection techniques are

used. Those might be e.g. measuring the iris' reaction upon light influences to ensure a live person rather than an image is presented to the sensor.

Attack on the Biometric Signals (3) Similarly to attack scenario one (1), one form of an attack could aim to manipulate the initial biometric images. At this stage in the authentication mechanism, a possible attack would be to resubmit previously-stored or -retrieved biometric signals and replay them to the system bypassing the sensor.

Manipulation of the Extracted Features (4) This form of attack can be achieved by either manipulating the Feature Extractor to e.g. only extract a manually specified set of features rather than the system-specified set. Further though, the transferred feature sets could be manipulated during their transfer to the Matcher.

Overriding the Matcher (5) The Matcher could be manipulated to generate a pre-defined result independent of the actual result that the input data would achieve.

Manipulation of the (stored) Template (6) An attacker might manipulate templates either in their stored form in the database or while those are being transferred to the Matcher for matching them against the retrieved templates during the authentication process.

Manipulation of the Decision (7) Once the Matcher has determined a decision, this decision needs to be sent to the secure system which is going to be accessed. During the transfer of this information, the result might be intercepted and manipulated.

One downside of biometric authentication is the inability to change the biometrics if they have been corrupted as one would do with a password. [4] Therefore, as there are limited amounts of different biometrics being available to an individual, the templates created from those biometrics should be thoroughly secured e.g. by applying non-revertible transformations.

Transmission and Storage Security of Biometric Information

As biometric systems and the connected equipment have become less expensive over the last years, more and more companies set up their own biometric systems. However, often the biometric templates in those systems are stored "*only*" encrypted. As discussed previously, the main downside of biometric systems is the potential loss of the biometric data. As the information cannot be easily replaced, revoked or reset, it has to be especially secured. Thus, biometric applications should be designed with the security feature as the main focus. Even better would be designing secure biometric applications that - even when compromised - do not allow an attacker to forge the original biometric template from the retrieved data. [16]

Biometric data, due to its characteristics is harder to deal with in terms of securing it than e.g. passwords. For passwords, the standard procedure would be to use a *Cryptographic Hash function* (One-Way-Function) to derive a string of characters (*hash*) from the password which ensures that it is not possible to recreate the original password from that *hash*. To face the task of securing biometric data, there have been several approaches to overcome having a template stored in a way in which the original data might be retrieved from it. These approaches are mainly dividable into three categories [16]:

- **Robust Hash Functions** One main criterion for cryptographic hash functions on passwords is that with any change of the original password, the change in the retrieved or computed hash should be as big as possible. e.g. there should be no possible way to tell whether two different hashes have been derived from similar passwords. However, the difficulty in the field of biometric systems is the noise of the data. There will never be 100% identical biometric samples to be used during authentication. Therefore biometric systems usually work using thresholds of confidence of a match between the samples. However, if biometric data would be stored using a *standard* cryptographic hash function, there would be - by definition and requirement - no possibility to tell whether two samples are similar. For biometric systems, the requirement would hence be to create a *Robust Hash Function* where small changes in a sample would result in a similar hash value. (Some examples of such research can be found in [17] [18] [19])
- **Similarity-Preserving Transformations** are used to modify biometric templates for a specific use. Often, primarily the visualization and representation of the template changes. One basic example would be the transformation

from the image of a fingerprint to the abstraction of minutiae vectors in a matrix. Most of the biometric transformations are build in such a way that they would keep the similarities in between separate templates. As pointed out previously, this is needed to enable matching of templates later on. However, in most cases, the transformations are not *One-Way Functions*. I.e. the original template can be derived from the transformed representation in many cases. Thus the transformation would not be a proper way to ensure the security of the original template. Therefore research in the field of *similarity-preserving, hard-to-invert transformations* has been conducted. (Various work in this field can be found to be publicly available. Some examples of which are: [20] [21])

- **Sketches** tackle the problem of *noisy* biometric data from another perspective. The assumption in this field of research is biometric data can be *cleaned up* and one may be able to retrieve a set of data which will always be identical. To be able to do this, one uses so-called *sketches* [22]. Sketches are publicly available and are initially generated by deriving them from the original biometric sample. As soon as another biometric input is to be analyzed, the publicly available sketch can be used to clean noise from the input biometric sample. However, the main difficulty is just as above that the original biometric template should not be retrievable from the publicly available sketch. [23]

2.2.4 Multi-Factor Authentication

To enhance security systems, researchers previously began to develop systems that support so-called Multi-Factor Authentication (MFA). These systems allow using more than one authentication mechanism for a single authentication procedure. In most cases, the MFA provides the user with (limited) freedom of choosing their preferred authentication procedures.

Firstly and most prominently this combines multiple layers of authentication. Thus it remedies a multitude of drawbacks of the standalone authentication mechanisms. This process especially remedies the danger of loosing a secret or an object as in KBA or OBA While an MFA mechanism does not ensure the secret is not lost, it ensures that somebody who has compromised a password or stolen a physical token can not directly access the system but has to go through another authentication factor which he most probably does not have at that point in time. Therefore the

system is far more secure against user errors such as stolen passwords or lost physical tokens.

The most-common biometric sensors are to be found in smartphones. Mobile devices with added security features such as biometric sensors have proven to be very popular over the years. Just as well as the manufacturers have extended functionalities of the devices, they have added extended and advanced security systems inside the devices. Some of these are separate execution environments, improved access control systems and support for PIN, password, or biometrical logins (c.f. *Section 2.3*).

However, while MFA can use biometric authentication to provide the second layer of security for any other method of authentication, improved security of the biometric templates is not the most prominent aim of multi-factor systems. [24] A common integration of a multi-factor authentication is the use of a mobile device. As can be imagined, enabling mobile devices to perform those security features can be a taunting task. To understand the involved platform, the following sections highlight the functionalities and security features of the mobile OS Android.

2.3 Android OS Security

Android Smartphone OS is the name of one of the most-common OSs on the market for mobile devices. Recent Studies estimate the market share of Android OS to range in the area of roughly 75 % (c.f. [25] [26]). Android has been developed by the Open Handset Alliance led by Google since 2008. The open-source operating system Android has experienced impressive growth in popularity since its release.

One of the key elements of the Operating System's success is the amount of provided applications (*apps*) that are downloadable via Google's Play Store. The *Play Store* currently hosts more than 2.5 Million (3rd Party) apps. [27] In contrast to Apple's *App Store*, apps hosted on Google Play are not assessed and verified manually. Google rather uses a dynamic emulation environment called *Bouncer* to protect against malicious apps. The emulation environment checks the application's code for malicious pieces such as downloads from external websites or installation of other external tools. However, it can not automatically detect and analyze flaws like vulnerabilities. Therefore, while Google is able to automatically limit the number of malicious apps, there will always be apps with security issues and flaws being published. [28]

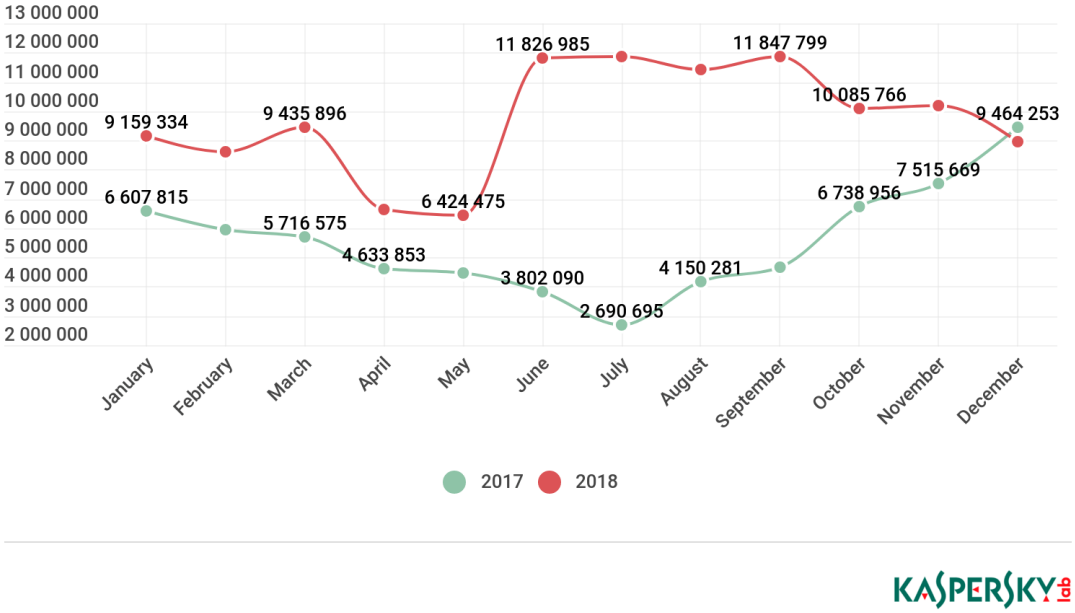


Figure 2.2: Comparison of Attacks against Mobile Devices in 2017 and 2018 [29]

The popularity of Android as a mobile OS has further drawn criminal's interests. Over the last years, security researchers and providers have found noticeable increases in attacks against mobile devices. (c.f. *Figure 2.2: Comparison of Attacks against Mobile Devices in 2017 and 2018 [29]*). The techniques, attackers are able to leverage on mobile systems are almost as endless as on classic computers. Using flaws and vulnerabilities in the system or other applications, attackers might be able to receive control over the underlying OS allowing them to receive access to personal data or manipulate features such as settings on the phone. However, to fully understand how Android OS might be exploited, the underlying function and security mechanisms are highlighted.

Android System Structure Overview The bottom layer of the Android OS developed under the Android Open Source Project (AOSP) is a Linux Kernel modified to especially provide support for the often limited resources of embedded environments. This approach bases Android on a robust driver model with existing memory- and process-management, networking support, and further basic functionalities. Atop the Linux Kernel, Android uses native libraries which further support high-performance third-party libraries. Android applications are developed in Java and C/C++. Up to Android 5.0 *Lollipop* the code for the Android applications

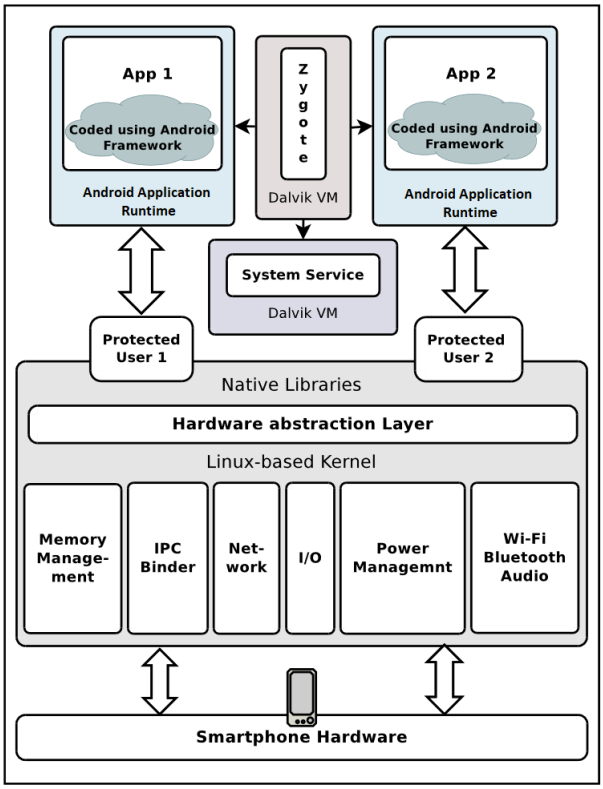


Figure 2.3: Android System Architecture Overview (adapted from [28])

was translated into Dalvik bytecode using the Dalvik runtime and was executed in a Dalvik virtual machine (VM). From Android 5.0 onward, the application code is translated into binary code using the Android Application Runtime (ART) and then run inside a Dalvik VM (c.f. Software-Stack as seen in *Figure C.1: Android Software Stack [30]*). The main difference between the former compilation using the Dalvik runtime and the current ART is the execution speed. Using Dalvik runtime, the code is translated *Just in Time*. Which refers to the process of translating the code required for execution only at the moment in which it is to be executed. This had the main advantage that less memory space is required to save the compiled code. However, with current technologies, memory space is not as much of an issue as it had been 12 years ago. Hence ART has been invented to perform *Ahead of Time* compilation. This refers to the method of compiling all code at once when installing the application. As all code has been compiled and translated already, the execution of a program happens faster. However, the *Ahead of Time* compilation leads to a slightly increased installation time and a bigger memory footprint. [31]

Once the code has been compiled, it is run in the Dalvik VM. The Dalvik VM works as a sandboxing environment for the application. I.e. each application runs

as an instance of a separate protected user on the underlying Linux system (c.f. *Figure 2.3*). Therefore each application is assigned a separate space in memory and storage on the device to make sure it does not interfere with other applications. [30]

Android Security Enforcement

Application Sandboxing Android utilizes the Discretionary Access Control feature from Linux at the kernel level. This is achieved by assigning a Unique Identifier (UID) to each application. By implementing this feature, Android ensures that an application only runs in its separate sandbox and thus cannot interfere with other apps or system services. Further, Android implements so-called *Paranoid Network Security* which runs network and communication features such as Wi-Fi, internet access, Bluetooth or telephone services in different groups. This allows a fine-grained assignment of permissions for a specific network service only. Further, aside from the UIDs, an application might be assigned one or more Group Identifier (GID). GIDs manage an application's group assignments. Based on those group assignments, interactions with other applications are controlled.

Public Key Infrastructure (PKI) Certificates A further aspect of Android's Sandboxing environment are PKI certificates. Each app needs to contain a PKI certificate signed by the developers. Signing applications creates the point of trust between Google and Android developers. By providing PKI certificates, developers can be assured that their applications are provided to the users unmodified. Hence users and Google know that the developers are responsible for their apps' behavior. Further, the signing is used to place an application inside a sandbox on the device. When installing an application, Android checks whether the certificate of the new app matches the certificate of any other app that has previously been installed. If there is a match, the new app gets assigned the same UID as the old app to place them inside the same sandbox and allow interaction in-between the apps. [28]

Permissions at Framework Level In the Android Application Framework, the permission system has been implemented as the very basis. Permissions are needed for interactions of an app with anything that is outside of its sandbox (I.e. it's UID- and GID-assignment). The standard for permissions in Android is that out of the box an application has no permissions at all. Permissions need to be explicitly

assessed and assigned by the developer in the application's *AndroidManifest.xml* file. Android divides permissions into four different levels. [28]

- **Normal** Permissions with minimal risk for device, system, and user (data) are part of this group. These permissions - if specified in the manifest - are automatically provided to the application during its installation.
- **Dangerous** This categorization contains permissions which might allow access to the user's data or important functionalities of the device such as telephony or network connections. These permissions have to be manually granted by the user before or during the installation process. (Usually, a user will be prompted with a pop-up window in which he can enable the permissions.)
- **Signature** These permissions are strictly limited. They are only provided to the application if the application has the same signature certificate as the application that offers the service for which the permission is needed. An example would be a standard Android System functionality which's usage is only allowed for official Google applications. In that case, the Google application is signed with the same certificate as the service that offers the functionality.
- **SignatureOrSystem** This group of permissions slightly expands the *Signature* group. Additionally, to what is described above, these permissions might be accessed by apps signed with the same certificate as the Android system image. This would allow for official Google apps to use third-party generated permissions (that no other third party is supposed to be allowed to use).

The Android Application Framework requests all permissions to be coarse-grained. This refers especially to the fact that there is no implicit assignment of permissions. I.e. if an app is assigned with permissions to *WRITE_CONTACTS*, the permission to *READ_CONTACTS* has to be given separately. [32]

Secure System Partition One main task to ensure the security of a system is to ensure the system partitions are safe from tampering through third parties. For Android, the system software such as the Linux / Android kernel, system libraries, runtime, framework and system applications are stored on a separate system partition. This partition is read-only to ensure no unauthorized access or modification. Further, some parts of the file system and memory are specially secured with privileges to prevent modification or tampering when the device is connected to a PC via USB.

Application's AndroidManifest.xml The previously mentioned *AndroidManifest.xml* is a core component of every application that is installed on a mobile device. The *AndroidManifest.xml* contains basic information about the application such as the package name, application name and its components. Further, it contains the previously mentioned information about the applications permissions. Finally, the application manifest serves as core component for the application's certificate security. The previously mentioned certificates that are used to e.g. allocate the application's sandbox are applied to the *AndroidManifest.xml*. During the evolution of Android's security mechanisms, the usage of the AndroidManifest.xml and corresponding certificates has changed. One example of such is that originally unsigned Android applications could be installed onto a mobile device. Further, originally the Android Manifest was the only component of the application secured via the developers certificates. This would ensure no tampering with e.g. the permissions or the core components such as the activities of an application. However, as the code for the components itself has not been secured with the certificate, the code could be modified without invalidating the signed manifest. The current version of Android's security principles - *Android Package (APK) Signature Scheme v2* - provides a whole-file signature. This strengthens integrity guarantees as now even minimal manipulations to the application's code invalidate the signature. [33]

Further Security Enhancements

- Google encourages users to only use software provided via the official **Play Store**. This is due to the fact that before making a software able to download, Google verifies the normal behavior of an app by using Bouncer, which is a dynamic sandbox analysis environment. This verification provides a reasonably effective security mechanism. [34]
- Since Android 4.3 *Jelly Bean*, **Mandatory Access Control (MAC)** has been introduced over Discretionary Access Control (DAC). In MAC the owner of the resource rather then the system decides who can access it. Thus, even in cases in which the system's *root* [1] may be compromised, MAC would limit the potential of malicious activities. [28]
- **Google Play Protect** is the name of a service that scans all apps installed on a device for *Potentially Harmful Applications*, thus bringing a further layer

[1] In Linux- / Unix-based OSs such as Ubuntu, AndroidOS or MacOS, the *root* user is the system user which has the highest privileges and (potentially) access to every file, service and function of the system.

of security to applications which might have not been installed via the official Play Store.

- Google further pursues constant **Security Research** and advancements. To achieve this, Google founded the *Android Security Rewards Program* which offers some of the highest-priced rewards in the industry (more than $3 million in total reward program payouts in 2018). [35]
- **Trusty** is a secure OS for Android that provides a Trusted Execution Environment (TEE) for Android. Trusty provides Android with a separate Operating System that is both - physically and virtually - isolated from the Android OS. The Trusty isolation enables Android systems to be capable of having an area that is secure from malicious Android applications or vulnerabilities discovered in Android OS. [36]

3 Application Analysis

Previously, this work explained underlying technologies and possible security features - especially for biometric authentication systems. However most of the presented content has been derived from research efforts. This chapter identifies applications that use biometric methods for remote authentication. Thereby a common understanding about the technologies used outside of reserach efforts is provided in the following chapter.

3.1 Market Analysis

Market research on improvements in biometric authentication systems is used to gain insights into which purposes of such software are currently served. However, a proper analysis should be performed for *Software Products* (both, Open Source and Commercial) and just as well for *Frameworks and Standards* which should be adhered to. To identify proper solutions, web research has been conducted identifying prominent actors in that field. The solutions identified are *Microsoft Authenticator, ThumbSignIn, BioID, Gluu,* and the framework *FIDO2*.

3.1.1 Commercial Software

When considering commercial attempts to authentication, one has to differentiate between applications or hardware that secure direct authentication attempts and software or hardware which enables remote biometric authentication. As direct biometric authentication systems are no new trend anymore and used almost everywhere, this work will prominently consider those systems that provide features for remote authentication.

Microsoft Authenticator is one such system. [37] The application provides the possibility of using a phone to sign in to any Microsoft application. In detail, a username is entered at the service that the user aims to access. This triggers an

authentication request to the user's phone. The user has to unlock his phone by either using his passcode/PIN or by using the preferred biometric authentication method of his choice such as FaceID or fingerprint authentication. Once the phone has been unlocked, the user will receive a notification asking him to press a button and hence approve the authentication attempt. As per Microsoft's definition, this process should be counted as a proper Two-Factor Authentication (2FA) as the phone as well as the knowledge (PIN) or the biometric treat is needed to enable the authentication. [38]

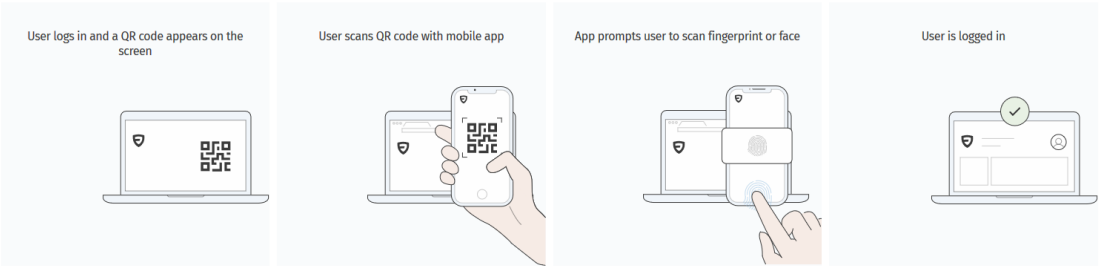


Figure 3.1: ThumbSignIn Biometric Remote Authentication Process Overview [39]

ThumbSignIn [39] is one other company that offers Software as a Service (SaaS) solutions in the field of authentication. Those range from classic 2FA solutions to modern, password-less and biometric login systems. *ThumbSignIn* provides solutions for almost every use case, ranging from remote biometric authentication for enterprise apps up to identification for call-centers or Internet of Things (IoT)-devices. However, in contrast to Microsoft's Authenticator, ThumbSignIn uses a slightly different action flow to remotely perform biometric authentication (c.f. *Figure 3.1*). Rather than connecting a mobile device to a user account in advance and sending PUSH-Notifications, the user proactively has to scan a code which contains the information needed for the login. The application then takes corresponding action and asks the user to authenticate as previously defined and then authenticates him against the server. This provides the advantage of easier and quicker implementation into any other service. However, at the same time this somewhats disrupts the user's workflow as the phone has to be picked up, pointed at the display and only then the authentication can be performed. [40]

BioID is a german company offering authentication solutions which can either be bought as Biometrics-as-a-Service or as on-premise solutions. [41]. BioID follows some high-priority privacy and security design principles:

- **Anonymous Face Recognition** During an authentication process, biometric data is stored as a biometric template with no personally-identifying information to the user.
- **Security By Design** As a German company, BioID is obliged to adhere to the EU General Data Protection Regulation (GDPR). Therefore the software is built in-house and hosted in European data centers.
- **Liveness Detection** BioID has understood that the most important aspect in securing unmonitored biometric authentication systems is a proper liveness detection. Therefore BioID has developed a multitude of patents in the field of liveness detection and constantly strives to enhance their product.

BioID themselves see use cases in almost all industries. Ranging from activation of telephony SIM cards over financial services up to verification for online examinations. For their online services, BioID offers the BioID Web Service Application Programming Interface (API) which can be directly integrated into any existing application infrastructure. Further, BioID provides a dedicated mobile application to perform the authentication.

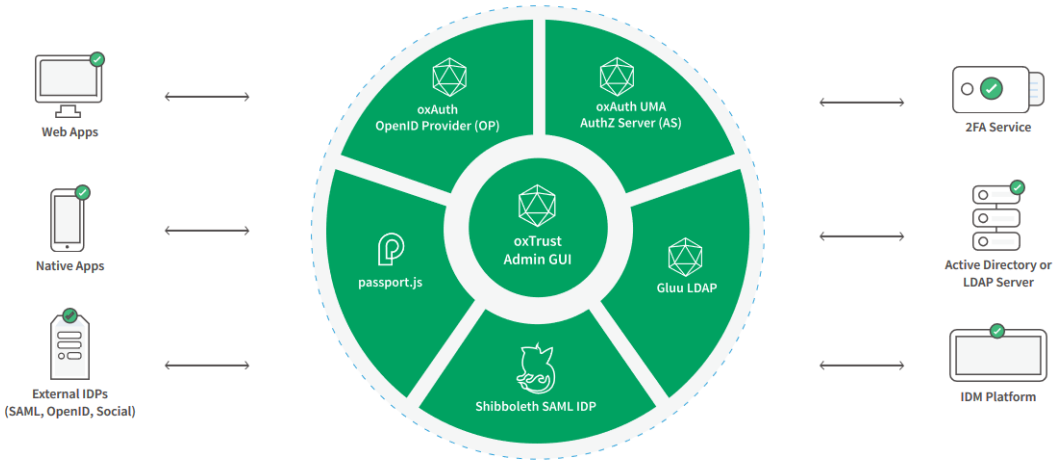


Figure 3.2: Gluu IAM Solution Overview [42]

Gluu offers an Open-Source IAM solution. Their solution provides all kinds of services to operate enterprise-grade architectures (c.f. *Figure 3.2*) . The core functionality of the Gluu landscape is the *Gluu Server* which, in its form as identity provider (IdP), is the heart of an IAM solution. IdPs are used to store and manage

the users and authentication processes in a shared environment. They implement services and features such as Single-Sign-On or provide support for 2FA applications. With its core architecture and functionalities being Open-Source, the Gluu Server can be easily modified to fit every architecture.

3.1.2 Frameworks and Standards

> "In today's digital world, our customers expect to do their banking the way they want, when they want and in a consistent, reliable, clear and easy way. ING's strategy is built on providing our customers access anytime, anywhere in a clear and easy way. Yet, security is paramount and cannot be compromised. We support FIDO authentication and its advancement globally, because it's an open, scalable and interoperable framework that can support the current and future authentication needs of our customers and the industry." [43] Rob Bening, Chief Information Security Officer, ING Bank.

Fast Identity Online - FIDO

The Fast Identity Online (FIDO) Alliance has currently published three sets of specifications. All of which are aiming at enabling a simpler and stronger authentication for the web. The main functions of the FIDO Alliance are: [44]

- ***Develop* technical specifications** that define an open set of mechanisms for scalable, interoperable and secure authentication without passwords
- ***Operate* industry certification programs** to ensure successful adaption of the standards around the globe.
- ***Submit* specifications** for formal standardization to recognized standards development organizations

Since its announcement in 2013, the FIDO Alliance has increased in size and thus enlarged its momentum in the industry (c.f. *Figure 3.3*). Today, the FIDO Alliance consists of a multitude of industry-leading companies which act together with the goal to enable systems, users, and people to a simpler method of authentication. Members are major tech companies such as *Amazon*, *American Express*, *Facebook*, *Google*, and *Microsoft* (c.f. *Figure 3.4*).

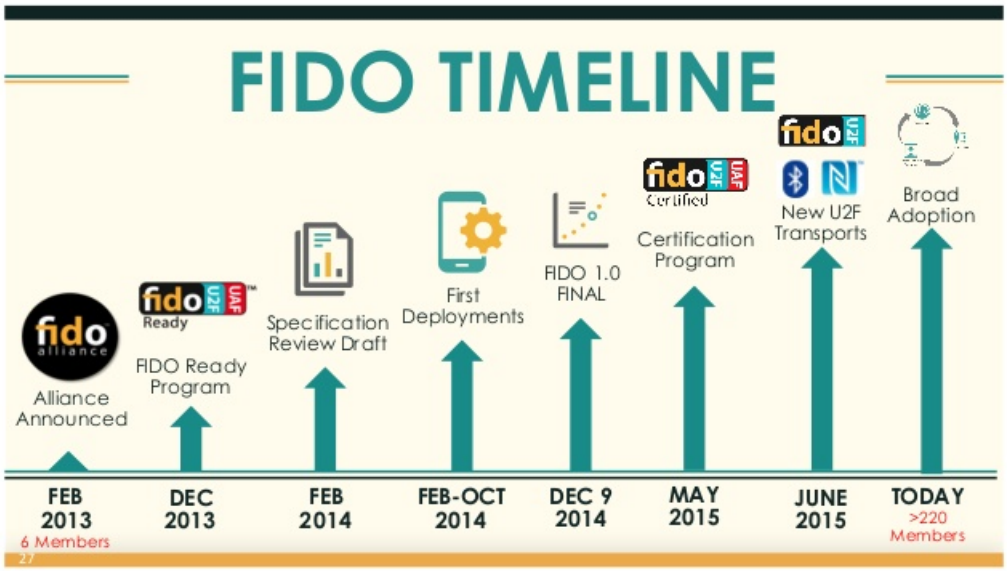


Figure 3.3: FIDO Alliance Status and Evolution [45]

The FIDO Alliance currently has published three sets of specifications, the most important of which being the FIDO2 specification. [44]

The FIDO2 Specification

FIDO2 is the most prominent set of specifications from the FIDO Alliance. *FIDO2* provides possibilities to leverage common devices to be able to perform easy and standardized authentication procedures with online services in both: mobile and desktop environments. To achieve this, the *FIDO2* specification leverages the World Wide Web Consortium (W3C)'s **Web Authentication** specification and combines it with the FIDO Alliance's corresponding **Client-to-Authenticator Protocol (CTAP)**.

The **Web Authentication** Specification enables online applications and services to perform FIDO authentication by using a standard web API that can be used by the related platform infrastructure such as browsers and backend servers. Web Authentication (WebAuthn) is a collaborative effort based on specifications initially proposed to the W3C by the FIDO Alliance in 2015 and was then reviewed, iterated

Figure 3.4: FIDO Alliance Board Members [46]

and finalized by both, FIDO and W3C communities. The official web standard of WebAuthn has been designated in March 2019. Currently, WebAuthn is supported by default in *Windows 10*, *Android Platforms*, *Google Chrome*, *Mozilla Firefox*, and *Microsoft Edge*. [44]

The *WebAuthn* API allows servers to register and login users by using public-key cryptography instead of a password. When a user registers with the service, the user application creates a cryptographic key pair consisting of a private and a public key. This key pair is considered a *credential*. The user further sends the public key to the server and stores the private key securely on the device. During a login attempt, the server sends a challenge (a random string in this case, also called a *nonce*) to the user. The user encrypts that string with his private key and sends it to the server. If the server is able to decrypt the challenge using the user's public key, the server can be sure that the private key of the user has been correct and thus will authenticate him.

CTAP is the complementary specification to WebAuthn. While WebAuthn is providing the possibility of the server to connect to the computer via the browser's APIs, CTAP is providing the possibility for the browser to connect to external devices such as USB-Sticks or mobile phones for authentication.

Within the FIDO Alliance, FIDO2 is seen as the evolution of both predecessor standards: **UAF** and **U2F**. [47]

Universal Second Factor (U2F) has been the first specification proposed by the FIDO Alliance. With this specification, the Alliance supported the usage of a second factor for authentication. This increased the security of current password infrastructures as it enhanced a single-factor authentication to a MFA system (as explained in *Section 2.2.4: Multi-Factor Authentication*). The specification provides details on the integration of the second factors (tokens, smartphones, biometrics). With the release of FIDO2, U2F has been relabeled to *CTAP*.

Universal Authentication Framework (UAF) is the framework for password-less authentication that has been proposed by the FIDO Alliance. For this framework, the user would carry a device with an enabled FIDO UAF stack. During registration, the user would assign a local authentication mechanism out of those allowed by the system. Examples of such might be swiping a finger or looking into a camera. Upon authentication, the user only has to repeat the local authentication and will then be authenticated.

3.2 Security Requirement Engineering

As the analysis of competitor products showed, remote authentication using biometric principles and methods requires extensive engineering of appropriate security principles. The important requirements shall be collected to be able to assess the compared systems and solutions against those requirements.

- **Server Matching** During biometric identification processes, the matching and thus the decision whether a user is allowed to access a certain resource or not should always be performed by the partner with the higher level of trust. Most times, this will refer to the party that is providing the service which demands the authentication. E.g. If a user uses a remote, possibly third-party fingerprint sensor to authenticate against a highly secure database, it should be the authentication system's backend (on the database side) which has the sovereignty to approve or deny an authentication attempt. Otherwise, corrupted sensors or devices might make for attractive attack vectors. Therefore when considering the use-case of biometric authentication using a smartphone to log in to a remote application, it should be the application's sovereignty to run the *Matching Unit* and to generate a decision on the authenticity of the user. If this would happen to not be possible due to whichever reasons, the enrolled devices should be carefully selected. Potentially allowing a device

with a weak security to perform the decision making process would directly influence the security of the system as a whole.

- **No Original Biometrics** Previously, it has been pointed out that the main danger from using biometric systems is the potential loss of a person's biometric data. Therefore the main goal should - at any point in time - be the security of the user's biometric data. Hence, as soon as biometric samples are retrieved from the sensors, they should be directly transformed into a secure representation of the biometric template. The chosen process to enable this security mechanism should ensure the original biometric data is under no circumstances retrievable from the secure template. Some possibilities to achieve this goal have been mentioned in *Section 2.2.3: Transmission and Storage Security of Biometric Information*.

- **Revocable Templates** As every system could fail, the possibility should not be left untouched. In such case a secure version of the system would need to be set up again. To enable the same users to use the same system securely again, the biometric data should not only be stored in secure templates but the templates should be revocable as well. This enables a user to reregister a new revocable template in the case of compromise of the former template data and a the login with the old templates could be deactivated.

- **Biometric Data Avoidance** In addition to storing the biometric data in secure templates rather than as raw biometric data, the systems should further aim to minimize the amount of data which is being transferred via the network and may be prone to e.g. Man-in-the-Middle attacks or network sniffing. Therefore, biometric systems should aim to adapt e.g. ZKP principles for biometric authentication. [48]. This will minimize the amount of information that is being prone to potential digital eavesdropping attacks.

Features	Explanation
Server-Matching	The matching and scoring of the biometric templates should be performed on the server as the server is in the responsibility of securing the requested resource.
No Storage	Original biometric information should neither be stored on the server nor on the Client.
Revocable Templates	Revocable templates enable a further usage of the same biometric treats in case of a compromise of previously used templates.
Biometric Data Avoidance	Principles such as ZKPs should be implemented to evade as much potential data compromise as possible.

Table 3.1: Main Security Criteria for Remote Biometric Authentication to ensure the Security of Biometric Information

Aside from those specific requirements, there are also other requirements which are being considered *Best Practices* in software engineering and which by today have been adopted by almost any system and will thus be assessed in more detail. Some of those might be aspects such as secure (encrypted) communication, proper certificate architectures and provisioning, as well as extensive application testing for security vulnerabilities.

3.3 Exemplary Implementation

For the sake of a better understanding, the upcoming paragraphs provide an exemplary implementation of a remote biometric authentication system using smartphones. To be able to investigate and assess the exemplary implementation (further also referred to as *Prototype*), the chosen solution to implement shall be of an Open-Source architecture. Due to its modularity, a *Gluu Server* is implemented.

3.3.1 Underlying Information

The server is set up on a virtual machine running on a desktop computer. The desktop computer resides in the Hasso-Plattner-Institut (HPI)'s (wireless) network with a range of *172.17.0.0/14* with an IP-Address of **172.17.9.4**. On the computer, a virtual network with the Internet-Protocol version *IPv4* and an address range of

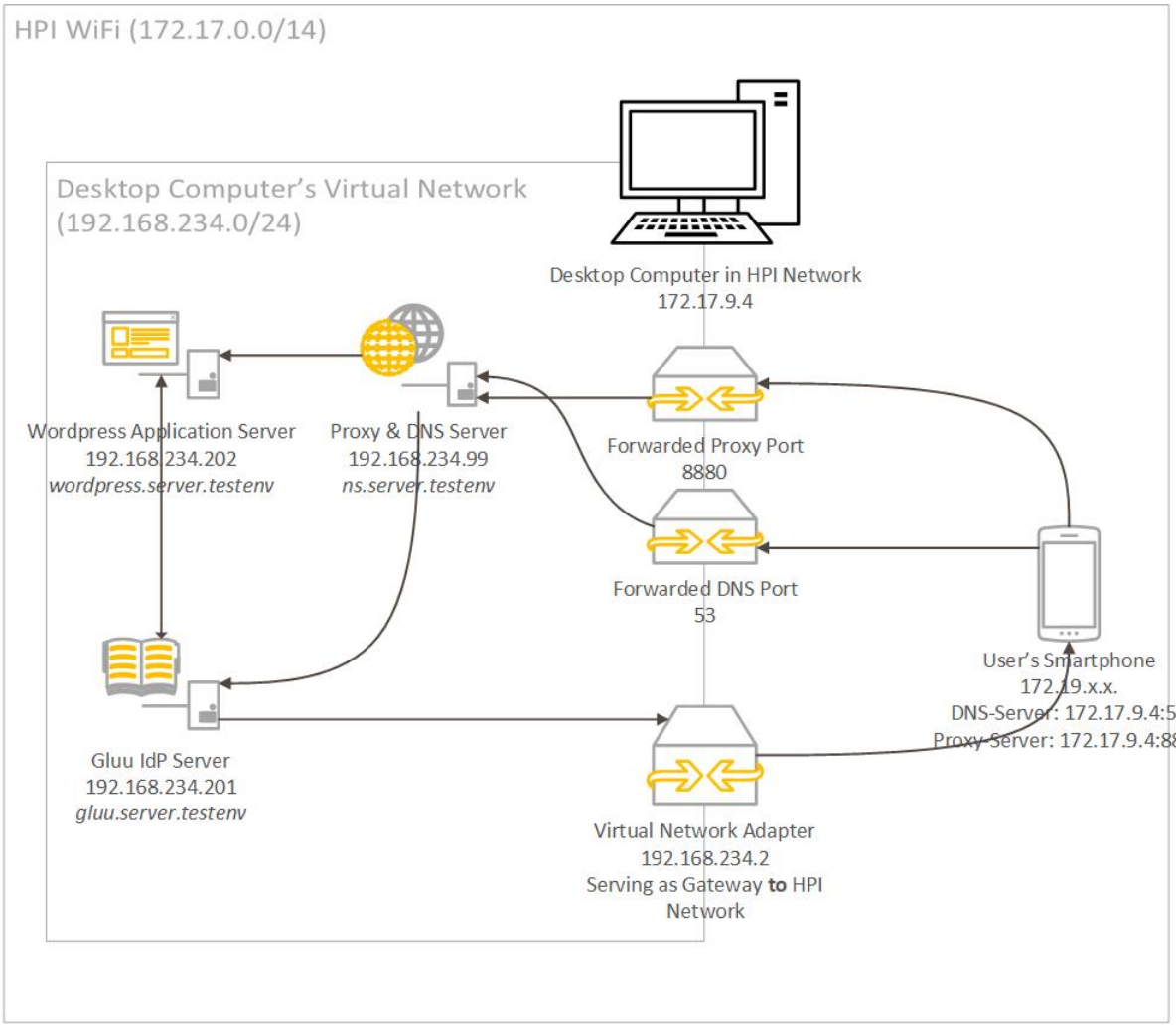


Figure 3.5: Communication & Network Overview of the Virtual Network, the Host System and a Mobile Device

129.168.234.0/24 is created. This network is connected to the HPI network via a virtual network adapter on the host computer. The virtual network offers a Domain Name System (DNS) and a proxy server which will be crucial for the communication of the mobile device to the Gluu Server. The mobile device connects to both, the proxy as well as the DNS server. Therefore all requests from the mobile device are routed directly into the virtual network via the host computer's port forwarding to the proxy server. (c.f. *Figure 3.5*)

3.3.2 Installation & Setup of the Gluu Server

Gluu Server is installed on a *Ubuntu 18.04 Server* instance. Ubuntu is an open source OS that can run on almost any system and is based on Linux kernels. [49] The Ubuntu server is already configured with an IP-Address of *192.168.234.201* and a DNS-Name of *gluu.server.testenv*. To be able to install a Gluu Server on a host system, small adjustments such as increasing the *file descriptor* limit have to be applied. This setting specifies the amount of files that the system is allowed to open in parallel. Formerly the limit has been used to ensure that a system does not run into problems with its hardware consumption such as memory or processor usage. However nowadays, these aspect are hardly influenced by the amount of open files at all. [50] Anyway, the limit needs to be increased to ensure that no errors are caused. Just as well, the server needs a Full Qualified Domain Name and a fixed IP-Address which has in this case been assigned already.

The installation itself is straight-forward. Gluu provides an repository which can be added to the Linux repository list. Once the repository is added, the GNU Privacy Guard (GPG) key for the Gluu repository is be added to the system enabling integrity checks and verification of the downloaded packages. Finally, the Linux package manager can be used to install the application. [51]

```
# echo "deb https://repo.gluu.org/ubuntu/ xenial main" > /etc/apt/sources.list.d/gluu-repo.list
# curl https://repo.gluu.org/ubuntu/gluu-apt.key | apt-key add -
# apt-get update
# apt-get install gluu-server-3.1.6.sp1
```

Once the Gluu Server has been installed using Ubuntu's package manager, the server needs to be initialized. To do so, the script *[..]/community-edition-setup/setup.py* from the Gluu installation folder is run. During the setup script, the underlying information for issuing a *X.509 Certificate* is entered. *X.509 Certificates* are digital certificates that use the International Telecommunication Union's PKI to generate, maintain, and verify certificates. (c.f. *Figure 3.6*)

Now that the Gluu Server has successfully been installed, the Uniform Resource Locator (URL) *https://gluu.server.testenv* can be accessed from the desktop computer to finalize the setup of the Gluu Server. Once authenticated with the administrator credentials, the administrative user interface is presented (c.f. *Figure C.2: Gluu Administrator Interface*). From the administrative interface, the configuration of the server can be performed and assessed. Some of the options are: configuring

```
root@localhost:~# service gluu-server-3.1.6 login
gluu-server-3.1.6 is running...
logging in...
root@localhost:~# /install/community-edition-setup/setup.py

Installing Gluu Server...
Detected OS  :  ubuntu
Detected init:  systemd
Detected Apache:  2.4

Installing Gluu Server...

For more info see:
  /install/community-edition-setup/setup.log
  /install/community-edition-setup/setup_error.log


** All clear text passwords contained in /install/community-edition-setup/setup.properties.last.

Do you acknowledge that use of the Gluu Server is under the MIT license? [N|y] : y
Enter IP Address : 192.168.234.201
Enter hostname : gluu.server.testenv
Enter your city or locality : Berlin
Enter your state or province two letter code : BE
Enter two letter Country Code : GE
Enter Organization Name : HPI
Enter email address for support at your organization : daniel[redacted].de
Enter maximum RAM for applications in MB [3072] :
Optional: enter password for oxTrust and LDAP superuser [R[redacted]B] :
Install oxAuth OAuth2 Authorization Server? [Yes] : Yes
Install oxTrust Admin UI? [Yes] : Yes
Install LDAP Server? [Yes] : Yes
Install Apache HTTPD Server [Yes] : Yes

hostname                                        gluu.server.testenv
orgName                                                     HPI
os                                                       ubuntu
city                                                     Berlin
state                                                        BE
countryCode                                                  GE
support email                                   daniel[redacted].de
Applications max ram                                       3072
Admin Pass                                              R[redacted]B
Install oxAuth                                            True
Install oxTrust                                           True
Install LDAP                                              True
Install Apache 2 web server                               True
```

Figure 3.6: Gluu Server Setup & Issuing of the *X.509* Certificate (*redacted*)

the authentication methods, setting up and managing relying party (RP)s, managing users, or inspecting logs. RPs refer to those services that use the identity provider (IdP) (our Gluu Server) as authentication service. During the current step of the setup, the aim is to enable user registration. This allows a user to browse to a specific link and register themself rather than the need for an administrator to manually create an user in a database. Depending on the configuration of the *user_registration* script in the administrator interface, a user might be enabled by default or might need to be manually verified and enabled. (c.f. *Figure C.3: Gluu Custom Script for User Registration Enabled*). Gluu offers a mobile application for biometric authentication called *Super Gluu*. To enable the password-less authen-

tication with that application, the *super_gluu* script in the *Person Authentication* section is modified and enabled. (c.f. *Figure C.4: Enable Super Gluu Authentication Script*). Assessing the script's options, it offers a parameter *authentication_mode* which is set to *two_step* by default (c.f. *Figure C.5: Super Gluu Authentication Script Options*).

```
66      self.oneStep = StringHelper.equalsIgnoreCase(authentication_mode, "one_step
        ")
        self.twoStep = StringHelper.equalsIgnoreCase(authentication_mode, "two_step
        ")
68
        if not (self.oneStep or self.twoStep):
70        print "Super-Gluu. Initialization. Valid authentication_mode values are
        one_step and two_step"
          return False
```

Listing 3.1: Excerpt from *super_gluu_authentication_script.py* showing the Assessment of the *authentication_mode* Variable

A quick analysis of the underlying source code (c.f. *Listing 3.1*) shows that setting the *authentication_mode* variable to *one_step* sets up the system for passwordless biometric authentication. Once the IdP is configured, a RP needs to be set up.

3.3.3 Installation & Setup of a Wordpress Application as the RP

Similarly to the Gluu Server, the Wordpress Application is to set up on an Ubuntu server version 18.04. The server has already been configured with a hostname of *wordpress* (DNS-Name: *wordpress.server.testenv*) and an IP-Address of 192.168.234.202. Wordpress is an open source content management system which allows to create blogs, websites or apps. [52] To be able to deploy a Wordpress application, a running web server as well as an underlying database is needed. Therefore there are multiple possible software stacks that can be used as a basis to run a Wordpress application. This work uses the *LAMP* stack (Linux, Apache, MySQL and PHP) as a basis for the Wordpress application. In addition to a MySQL database though, a Structured Query Language (SQL)-Client is needed. To communicate with the database, a MariaDB Client is installed. [53] (c.f. *Figure 3.7*) Finally, a Secure Sockets Layer (SSL) certificate is added to the Apache web server. This allows an encrypted connection between the client and the web server.

Once the underlying and required software stack is installed, a database and correspondind user for Wordpress are created. The Wordpress application is then downloaded and added to the the web server's directory (Apache in this case). Once

```
daniel@wordpress:~$ service apache2 status | grep Active
   Active: active (running) since Tue 2019-08-20 04:25:35 PDT; 2h 41min ago
daniel@wordpress:~$ php -version
PHP 7.1.31-1+ubuntu18.04.1+deb.sury.org+1 (cli) (built: Aug  7 2019 10:23:12) ( NTS )
Copyright (c) 1997-2018 The PHP Group
Zend Engine v3.1.0, Copyright (c) 1998-2018 Zend Technologies
    with Zend OPcache v7.1.31-1+ubuntu18.04.1+deb.sury.org+1, Copyright (c) 1999-2018, by Zend Technologies
daniel@wordpress:~$ service mariadb status | grep Active
   Active: active (running) since Tue 2019-08-20 04:25:36 PDT; 2h 41min ago
daniel@wordpress:~$ service mysql status | grep Active
   Active: active (running) since Tue 2019-08-20 04:25:36 PDT; 2h 41min ago
```

Figure 3.7: Analysis & Confirmation of the installed *LAMP* stack for the Wordpress Application

the credentials and database information are added to the *wp_config.php* file, the Wordpress application can be accessed using the URL *https://wordpress.server.testenv/index.php*. From there, the installation and initialization can be triggered.

Once Wordpress has been successfully installed, the URL *https://wordpress.server.testenv/wp_admin* is used to login to the admin interface. From there, settings such as appearance, users and authentication can be customized. Further, plugins can be added to the Wordpress instance. For this work, a plugin which can be used for the authentication with the Gluu server is needed. Gluu server provides support for *OpenID Connect (OIDC)* authentication mechanisms. OIDC is an authentication layer atop the authentication framework *OAuth 2.0* which is managed by the *OpenID Foundation*. [54] A search for *OpenID Connect* Wordpress plugin returns various results, one of which is a plugin by Gluu. (c.f. *Figure C.6: Wordpress Plugin Search Results for OpenID Connect featuring a Gluu Plugin*). Even though all other plugins supporting the OpenID Connect standard would work as well, the Gluu plugin is used.

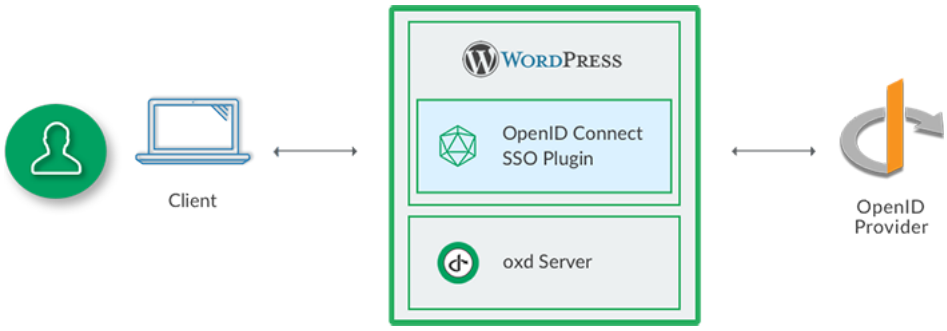


Figure 3.8: Access Flow & Overview using the Gluu OXD-Server [55]

The OpenID Connect plugin by Gluu relies on the Gluu oxd-server. The oxd-server enables web applications to use simple and static APIs for the authentication against

OAuth 2.0 IdPs. To work as a mediator service, oxd-server exposes APIs that can be accessed by the web service. Oxd-server then forwards the authentication requests to the connected identity provider (IdP) or authorization server. (c.f. *Figure 3.8*) [55]

Once the plugin is installed, several options can be configured. Amongst those the URL of the IdP, *https://gluu.server.testenv*. Further, the *Login* and *Logout* URLs are configured. Additionally, the port on which the oxd-server listens for API calls is specified. Finally, the *Enrollment and Access Management* section specifies settings for the authenticated users. Here, the option to *Automatically register any user with an account in the OpenID Provider* is selected. Therefore, every user with an account at the IdP is being issued a user account for the Wordpress instance and able to authenticate. Those users are automatically assigned with the *Administrator* role for this work. (c.f. *Figure C.7: Wordpress OpenID Connect Plugin Configuration*).

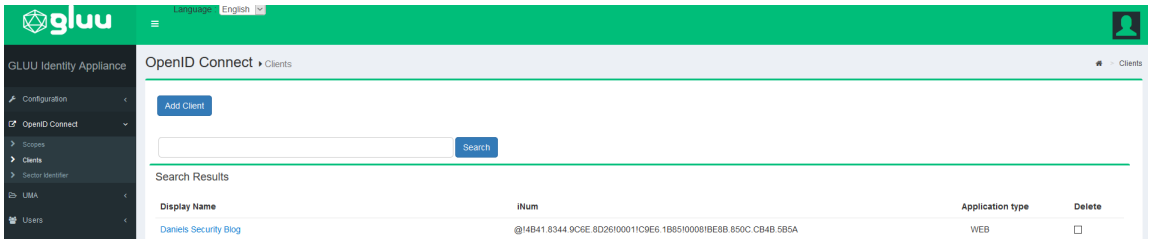


Figure 3.9: Gluu Server Overview over Registered Relying Parties (RP)

Upon confirming the setup within the Wordpress instance, the Wordpress application - respectively its oxd-server as mediator service - is registered as Relying Party (RP) within the Gluu IdP server. This can be confirmed by investigating the registered clients for OpenID Connect Authentication. (c.f. *Figure 3.9*). Finally, inspecting the detail information on the registered client, Gluu provides information about the properties that are provided to the client during an authentication process. The choice of which information shall be transmitted can be influenced by different factors. At hand is a very limited amount of properties being only *openid, email* and *permission* to allow a basic authentication (c.f. *Figure C.9: Gluu Server OpenID Wordpress Client Details*). If a RP were to require e.g. the date of birth for an assessment of categories the user has access to, that property would need to be added within the scopes of the authentication process as well. Finally, the authentication system for administrative logins to the Wordpress application is ready to use.

3.3.4 Authentication Process

First, a new user is registered at the Gluu server using the URL *https://gluu.server.testenv/identity/register*. The user is presented with a form to insert his information into. In this case, the required information consists of *username, name, E-Mail address, and a password*. The exemplary user being created is called *mastertest* (c.f. *Figure C.10: Gluu Server Mastertest User Registration*).

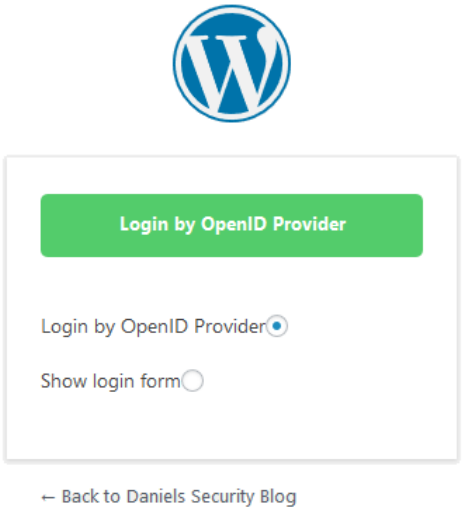


Figure 3.10: Wordpress Login Screen

During an authentication attempt, the user is provided with the login page from Wordpress. (c.f. *Figure 3.10*) The user is offered the option to login via *OpenID Provider*. In a production instance where the authentication process should be as seamless as possible, the option *Bypass the local WordPress login page and send users straight to the OP for authentication* from the OpenID Connect Configuration (c.f. C.8)would be selected. Once the user selects the option to authenticate via OpenID Provider, he is - during his first authentication - required to register his device. (c.f. *Figure C.11: Gluu Server First-Time Device Registration*)

To be able to use the Super Gluu authentication mechanism, the user has to install the Super Gluu app first. A download code for the respective app store is provided in the online interface. During the initial start of the application, the user is requested to confirm the biometric treat which he uses to secure his phone. Such might be the fingerprint on Android devices or the FaceID on iOS devices. Every time the application gets used in the future, the user is asked to confirm his identity via the biometric authentication mechanism.

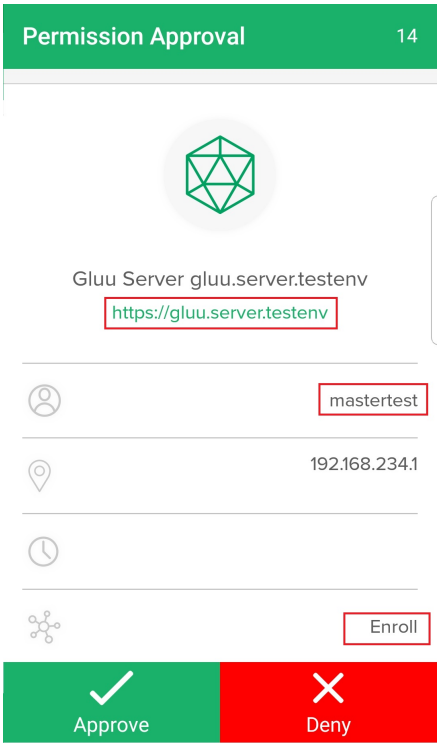


Figure 3.11: Android Super Gluu Application Approve Registration

With the installed *Super Gluu* application on either iOS or Android, the user scans the Quick Response (QR)-Code presented by the web application. Once the code is scanned, the Super Gluu application requires the user to confirm the enrollment of the current device for the specified username at the server. All this information is retrieved from the QR-Code. (c.f. *Figure 3.11*) Once the enrollment is approved on the mobile device, the authentication is approved (c.f. *Figure C.13: Android Super Gluu Enrollment Successful Confirmation*) and the user is logged in in his web browser and presented with the Wordpress administrative panel. (c.f. *Figure C.14: Wordpress User master.test Authenticated*)

Once a user has enrolled his device, for all subsequent authentication attempts, the process is slightly different. Once the user chooses to authenticate with the IdP, a notification is sent to his mobile device. (c.f. *Figure 3.12*) Upon selecting the notification, the Super Gluu application is started asking the user to confirm his identity using the fingerprint authentication. (c.f. *Figure 3.13*) This calls the Android fingerprint APIs to compare the used fingerprint against the one that is stored in *Trusty*, Android's Trusted Execution Environment. (c.f. *Section 2.3: Further Security Enhancements*). Once the user has successfully authenticated using his fingerprint, the Super Gluu application opens the notification and prompts the user

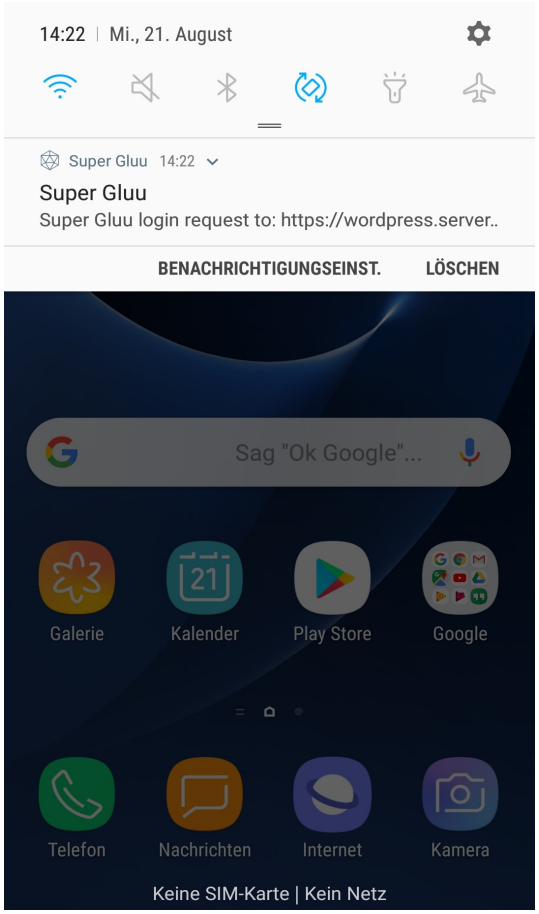


Figure 3.12: Android Notification from Super Gluu Application for Authentication

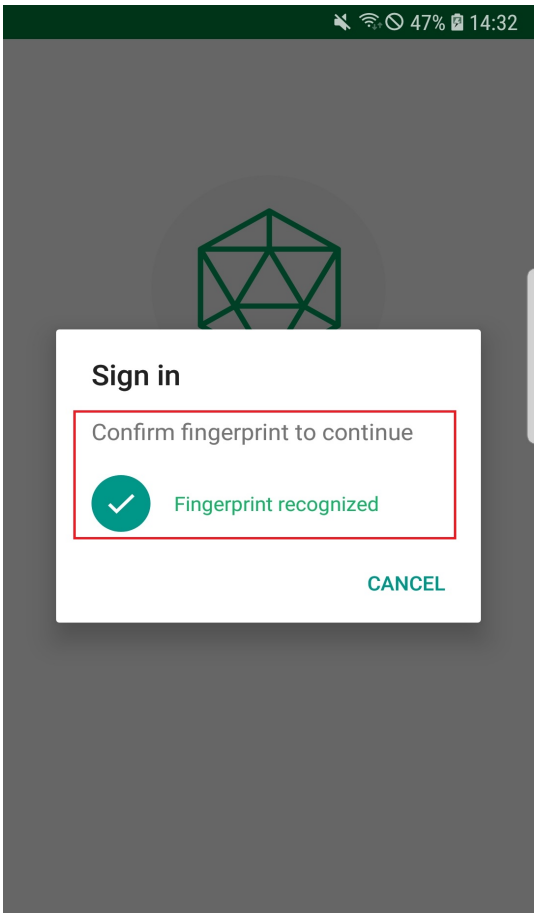


Figure 3.13: Android Super Gluu Application Login Requesting Fingerprint as Security Mechanism

to approve the login similarly to how the original device registration was approved. (c.f. *Figure C.15: Android Super Gluu Authentication Approval*) As soon as the authentication request has been approved, the user is provided with a confirmation message (c.f. *Figure C.16: Android Super Gluu Authentication Confirmation*) and authenticated at the web application.

4 Assessing, Adapting and Applying a Framework for the Structured Security Analysis

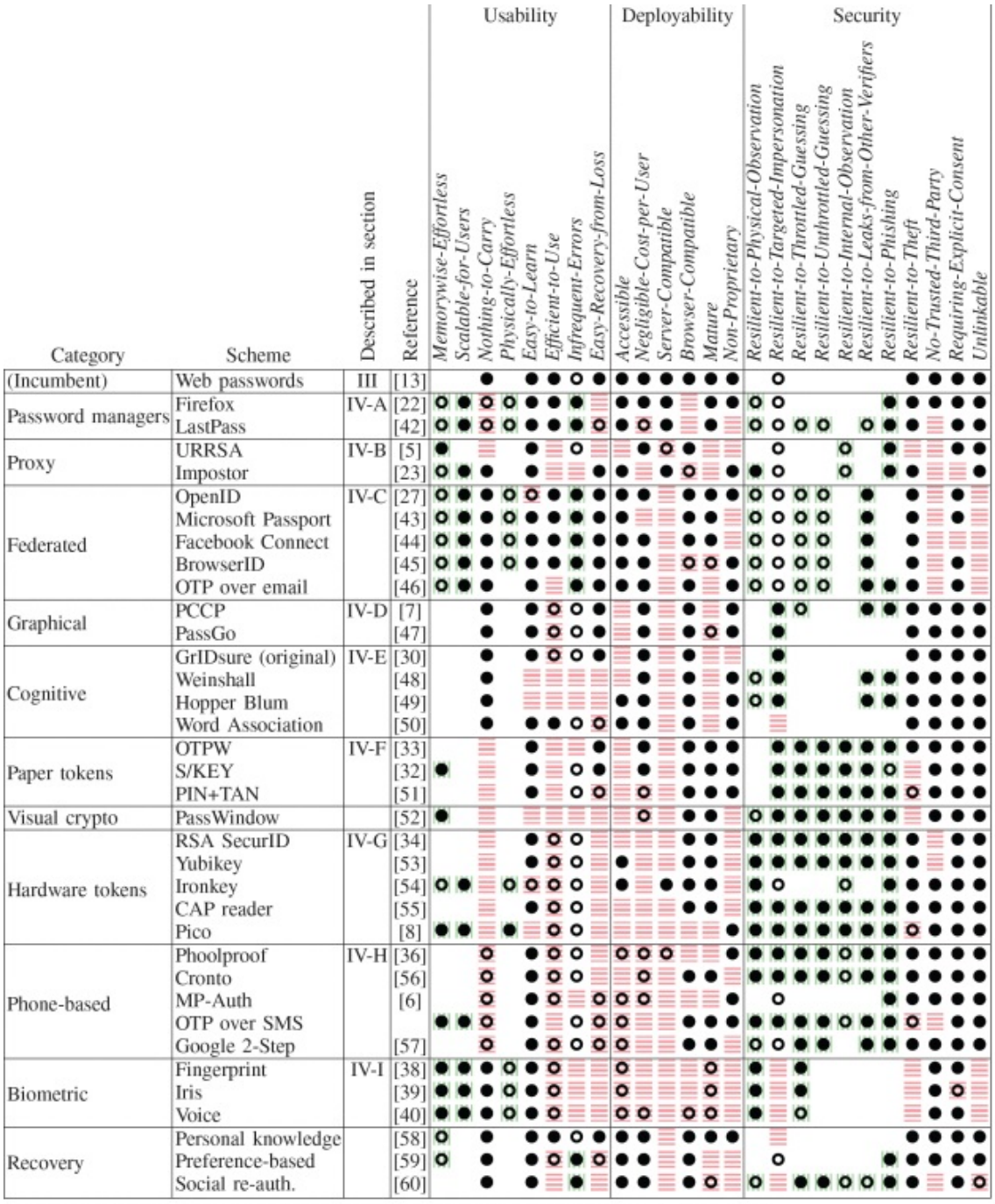
As recent as six years ago, *Bonneau et al.* [56] proposed a framework to assess authentication mechanisms. In their paper and the corresponding technical report *"The Quest to Replace Passwords"* they propose a framework to assess authentication mechanisms. The underlying research performed by *Joseph Bonneau, Cornac Herley, Paul van Oorschot* and *Frank Stajano*† assessed two decades of proposals to replace text passwords on the web. By doing so, they have been able to identify more than 20 different benefits that an authentication method might provide over the use of passwords. Those can be grouped into three different categories: *Usability*, *Deployability*, and *Security*. The authors during that time evaluated a multitude of alternative authentication methods that have been available on the open market at that time - ranging from password management software over hardware tokens up to biometrics (The result of that research and assessment is to be found in *Figure 4.1: Assessment of Authentication Mechanisms from Bonneau et al. [56]*). [57]

During the discussion of their findings, *Bonneau, Herley, Oorschot, and Stajano* agreed that at that point in time no system could be traded for another one providing only benefits for the overall solution. Specifically they assessed the fact, that the benefits in which passwords excel - namely, *Nothing-to-Carry, Efficient-to-Use*, and *Easy-Recovery-from-Loss* - are exactly where stronger authentication methods need improvement.

One important aspect to keep in mind is the fact that the original assessment has been performed almost a decade ago. During recent years, prominent technologies and systems have changed broadly. One example of such are smartphones. During the time in which the original assessment has been performed, pocket-computers of the size, kind, and capabilities as today's smartphones could hardly be imagined. Focusing on investigating the biometrical assessment in the original work shows that one main disadvantage of biometric systems has been the missing dispersal of of biometric sensors. Setting up a biometric authentication system would lead

to massive issues and costs as sensors would have to be bought and distributed. Further, carrying these sensors around would have proved to be a big effort thus negatively influencing the accessibility of the system.

To investigate and understand those aspects in detail, the original framework is provided, analyzed and explained in the following sections.



Category	Scheme	Described in section	Reference
(Incumbent)	Web passwords	III	[13]
Password managers	Firefox	IV-A	[22]
	LastPass		[42]
Proxy	URRSA	IV-B	[5]
	Impostor		[23]
Federated	OpenID	IV-C	[27]
	Microsoft Passport		[43]
	Facebook Connect		[44]
	BrowserID		[45]
	OTP over email		[46]
Graphical	PCCP	IV-D	[7]
	PassGo		[47]
Cognitive	GrIDsure (original)	IV-E	[30]
	Weinshall		[48]
	Hopper Blum		[49]
	Word Association		[50]
Paper tokens	OTPW	IV-F	[33]
	S/KEY		[32]
	PIN+TAN		[51]
Visual crypto	PassWindow		[52]
Hardware tokens	RSA SecurID	IV-G	[34]
	Yubikey		[53]
	Ironkey		[54]
	CAP reader		[55]
	Pico		[8]
Phone-based	Phoolproof	IV-H	[36]
	Cronto		[56]
	MP-Auth		[6]
	OTP over SMS		
	Google 2-Step		[57]
Biometric	Fingerprint	IV-I	[38]
	Iris		[39]
	Voice		[40]
Recovery	Personal knowledge		[58]
	Preference-based		[59]
	Social re-auth.		[60]

●= offers the benefit; ○= almost offers the benefit; *no circle* = does not offer the benefit.
|||= better than passwords; ≡= worse than passwords; *no background pattern* = no change.
We group related schemes into categories. For space reasons, in the present paper we describe at most one representative scheme per category; the companion technical report [1] discusses all schemes listed.

Figure 4.1: Assessment of Authentication Mechanisms from *Bonneau et al.* [56]

4.1 Assessment and Analysis of the original Framework

In order to understand the details of the assessment that has been performed in the work of *Bonneau et al.*, their rating has to be understood. In the framework, benefits are assessed. There is no detailed quantification of a certain factor. The systems are mainly rated as either *Offering* or *Not Offering* the specific benefit. In cases where a scheme almost offers a benefit, this is indicated by using a *Quasi-* prefix. Further, there are cases where certain benefits are not applicable to a specific authentication system or method. Instead of introducing a *Not Applicable* evaluation into the system, the authors have decided to rate the schemes as *Offering* the benefit. This is done on the basic assumption that a *Not Offering* would identify that a benefit and the corresponding aspect/mechanism had the possibility to fail. The decision of specifying the criterium as *Offering* therefore identifies the fact that there is no possibility for something to fail in regard to the specific criterion. As previously mentioned, the original work assessed 25 criteria in the fields of *Usability* (8), *Security* (11), and *Deployability*(6). The extensive list of the original criteria can be found in *Section B.0.1: Assessment Criteria from Bonneau, Herley, Oorschot and Stajano [56].*

During the decisions on the schema and categories, the original authors decided to neglect some assessments for future work. Examples of this are the missing assessment of resistance against some specific attack vectors such as (active) Man-in-the-Middle, Relay, and/or Session-Hijacking attacks. Further, an assessment of user affection of a certain scheme was omitted. I.e. even if a certain authentication method might provide plenty of improvements and benefits over other authentication schemes, it might be difficult to use in a certain aspect that has not been evaluated within the other benefits such that a successful user adoption would not be achievable at all.

Another aspect that the original framework does not offer is a nuanced rating. The original authors considered adding a *"Fatal"* rating (*Offering, Quasi-Offering, Not-Offering, Fatal*) to the framework to indicate extremely poor performance in a certain field that should lead to eliminating that scheme from all serious considerations.

To enable an in-depth assessment under today's standards, some adaptions have to be made to the original framework.

4.2 Adapting the Framework

In terms of the adaption of the framework, this work proposes the following:

Differentiation As the usage of some kind of a *Fatal* rating was proposed for future assessments, this would make for 4 possible different results per benefit. Further, the interpretation of some of the ratings for a specific criterion might prove difficult. In most related works, the interpretation needed to be explained separately in advance. To enable an easier interpretation of the results, the rating that is going to be used moving forward is to be found on a scale from [- | ∘ | + | ++ | +++]. Using this scale, a [-] will represent the *Fatal* assessment. For the amount of [+] it depends on the maturity in which a specific criterion is matched. This better adheres with the naming of the criteria as *benefits* where now a [∘] is referred to as *neutral*, i.e. not offering a benefit from this specific criterion.

Comparison to Passwords In the original work, the author's aim was to enable companies to choose other solutions over passwords. Hence the comparison was performed using the maturity of a certain benefit over the maturity of passwords as a reference. However, in the last years, many companies have already moved past passwords or extended passwords to use password-safes or similar systems. Hence performing a comparison to passwords in the current era of security mechanisms will not prove to be viable for most companies and people. Hence this work aims to perform a neutral analysis of each criterion for each system on its own.

User Affection Some of the originally proposed benefits such as *U5 - Easy-to-Learn* or *U8 - Infrequent-Errors* (or most of the other *Usability* benefits)provide the possibility to theoretically assess User affection. However, in the meantime, there has been other research that points out the importance of proper assessments of user affection. One such study, which has performed a user affection analysis based on the systems and technologies proposed in the original work by *Bonneau et al.*, has been performed by *Ruoti, Roberts*, and *Seamons* in their paper *Authentication Melee: A Usability Analysis of Seven Web Authentication Systems* [58]. This research will be used as a baseline for the analysis of user acceptance.

Focus As this work focusses on the use of biometric systems, the assessment will be performed mainly for the biometric systems assessed and presented earlier.

4.2.1 Expanding the Benefits

The previously presented aspects just as well as the security factors that have been presented in *Section 3.2: Security Requirement Engineering*, require the list of benefits to be expanded. This work hence proposes the use of the following additional benefits for the analysis of web authenticating biometric security systems:

Usability Benefits

U9 *User-Affection*: The use of the system appeals to the user. *Ruoti et al.* [58] have investigated that even though a system might be easy to use, it does not necessarily fancy the user's expectations.

U10 *Portability*: The authentication using the system should be easily applicable when using other systems to access the specific resource. (c.f. *Alaca et al.*[59])

U11 *Range-of-Use*: This benefit defines the possibilities to use the analyzed system for multiple solutions. E.g. password managers would be rated best here as they can be used for any other application. The less the possible applications to use the authentication mechanism, the worse their rating. This criterion is somewhat similar to *Portability* and the original *Server-Compatible*. However, *Portability* describes the ease of using the same system to access the same resource from another device. *Server-Compatible* describes the efforts that the developers need to pursue to implement the solution. This benefit of *Range-of-Use* describes the amounts of different systems/applications the user can authenticate against with a single solution.

Deployability Benefits

D1 *Accessible*: Originally this criterion was used to specify that users who were able to use passwords are able to use the system as well. However, as this work does not propose comparing to passwords anymore, this criterion will be used to rate the accessibility. E.g. the system would be rewarded with [+++] if both - illiterate people as well as amputees - are able to use the system.

Security Benefits

S9 *No-Trusted-Third-Party*: This benefit was originally proposed to ensure that a scheme does not rely on any trusted third parties by design or implementation. This might have been for example a remote Identity Provider. However, for this work, this benefit will be expanded upon the fact that - as the owner of an authentication system - there should be no need to rely on another party to perform an assessment (e.g. the biometric *Matching*) that provides access to your infrastructure. (c.f. *Section 3.2: Security Requirement Engineering*) Put concretely, one aspect of this criterion is the sovereignity of authentication to always be with the server side of an remote system.

S12 *Revokable-Biometrics*: As evaluated in previous chapters of this work, the security of the original biometric data should be the highest priority in a biometric authentication system. Therefore the usage of *revokable biometrics* is highly suggested.

S13 *Biometric-Data-Secrecy*: Due to the previously highlighted issues once biometric identifiers might be compromised, a policy of *biometric data avoidance* is proposed. The implementation might vary from system to system and might have to be closely investigated to enable a proper assessment.

5 Security Analysis of Biometric Authentication Systems

5.1 Analysis of other Competitor Products

Previously in this work, an overview of other available products has been provided (c.f. *Section 3.1.1: Commercial Software*). Further, a framework to analyze authentication systems has been assessed and expanded (c.f. *Section 4*). This chapter focuses on the security analysis of the specific applications. While investigating the mechanics of a given product or system, the corresponding system is being assessed based on the **Benefits** from the proposed framework. The results of the assessment can be found in *Table 5.1: Comparative Evaluation of the Assessed Authentication Systems*.

To perform an initial, in-depth analysis of a well-known system, the next subchapter performs an exemplary assessment of the Microsoft Authenticator application. This application is based on the FIDO Framework and is thus used as a baseline for the evaluation of the Framework itself in the following section. Following up, the FIDO-Based applications *ThumbSignIn* as well as the non-FIDO application *BioID* are assessed. Finally, the implemented system, *Gluu* is investigated and assessed before the results are presented.

5.1.1 Microsoft Authenticator

For Microsoft's Authenticator, the application relies on the use of FIDO2 to authenticate the users once the device has been unlocked. The application uses the authentication method of choice from the user to unlock his smartphone. If that method of choice is a biometric login, the authentication will be performed on the basis of biometrics as well. However, it is prone to the same vulnerabilities that the choice of OS and authentication method brings with it. I.e. if the user uses an insecure PIN for his phone, his account will be similarly insecure as the phone. [60] Therefore, the *Memorywise-Effortless* can only be assigned as a benefit to some

extent - depending on the implementation there is only one secret to be remembered for multiple accounts. The scheme is hence considered *Scalable-for-Users*. As the authentication is based on the smartphone, the user obviously has to carry around said device. Most people carry their smartphone around at any time, either way, therefore the benefit is rated with [++]. Adding to this, the scheme is rated with [++] for the benefit of *Physically-Effortless*. Due to the fact that there is no knowledge or skill required besides unlocking your smartphone, the benefit of *Easy-to-Learn* is granted as well. The benefit of *Efficient-to-Use* is rated with [++] as there is the need for extra interaction which takes little, but some time. As the benefit of *Infrequent-Errors* (low False-Reject-Rate) can not always be ensured for biometric systems - especially in regard to higher-security systems - this benefit will only be rated with [○]. The recovery is rated with [++] as it is possible directly from the phone. Though, the process appears to be somewhat lacking in terms of its documentation. Deriving from *Ruoti et al.'s* [58] work, the *User-Affection* of a smartphone-using (biometric) authentication system is rated [+++]. The benefit of *Portability* is present as well. The authentication can be performed in the same way every time - independent of the system from which the user requested the authentication.

In detail, Microsoft's Authenticator works very similarly to the FIDO2 authentication process. During the *Registration Phase*, the authenticating device (e.g. smartphone) creates a set of cryptographic keys. The public key is sent to the identity system (e.g. Azure Active Directory) and is stored in the user's profile at the identity provider. The private key is securely locked on the device (e.g. smartphone) secured by the authentication factor of choice (e.g. biometrics or PIN). During an *authentication attempt*, the device sends an authentication request to the identity system. The identity system requests verification and thus sends an authentication prompt to the user's device (e.g. smartphone). The authentication prompt contains a cryptographic *nonce*, i.e. a random or pseudo-random arbitrary number that will just be used once in a cryptographic communication. This prevents the possibility of using captured traffic for later replay attacks. The device receives the authentication prompt and requests the user to authenticate (e.g. PIN or biometrics). The principle is thus rated with a [+] for the benefit of *Accessible*. The local authentication unlocks the locally stored and secured private key that has initially been created. The provided *nonce* is being encrypted with the private key of the user and send back to the identity system. The identity system will then decrypt the *nonce* using the public key of the user that has been stored in the database and thus verify the user. Hence the system is rated with [+++] in terms of the benefit

of *Resilient-to-Phishing*. Similarly, as no biometric information is transferred, the system is rated [+++] for the benefit of *Unlinkable*.

This system offers the advantage that all biometric authentication is performed locally rather than performing a remote biometric authentication. [61] In regard to the benefit of *Revokable-Biometrics*, Microsoft Authenticator does not appear to be pursuing special actions to secure the biometric templates (∘). However, as the biometric authentication is only performed on the device itself, the scheme is rated [+++] for the *Biometric-Data-Secrecy* benefit. Regarding the benefit *Resilient-to-Physical-Observation* the system will be granted a [∘] as this highly depends on how the mobile devices are secured. Similarly, the *Resilient-to-Targeted-impersonation* would be granted [∘] if it was not for the need to be in possession of the phone as well. The authentication mechanism is therefore granted a [+] for that benefit.

With the available technologies, the system is granted a [++] for the benefit of *Resilient-to-Throttled-Guessing*. Whereas due to the limited constraints of mobile phone PINs (often only 10^4 - 10^6 combinations) the schema will be rated only [∘] for *Unthrottled-Guessing*. As the scheme is using standard devices and does not work based on MFA principles, it is rated [∘] for the benefit *Resilient-to-Internal-Observation*. Due to the usage of the cryptographic keys being stored only on the device itself, the principle is rated [++] for *Resilient-to-Leaks-from-Other-Verifiers* (this assumes that the user is not using the same PIN for multiple services). As the scheme uses the smartphone for authentication which is not usable without the corresponding PIN or biometrics, the scheme is rated [++] in regard to *Resilient-to-Theft*. Operating within the Microsoft ecosystem, the system could be rated as non-dependent on a third party in regard to the original definition of the benefit. However, the system relies on the mobile phone to correctly allow access to the user. Hence, the system's benefit of *No-Trusted-Third-Party* is only rated with [∘].

With the background information on Microsoft Authenticator, information, the scheme will be granted the following *Deployability Benefits*. As the (biometric) assessment is performed on the devices and only an encrypted nonce is received by the server, the principle might be slightly more cost-intensive (due to the extra challenge in the authentication process as opposed to e.g. a password login), but this *cost* is neglected due to current technology standards and security benefits ([+++]). As just presented, password scheme authentication mechanisms on the server-side would need adjustment ([∘]). The authentication being moved to the smartphone does not affect the browser, thus the scheme is considered *Browser-Compatible*. Due to the popularity and research performed around the FIDO2 specification, the sys-

tem is considered to be *Mature*. However, the system is obviously marketed by Microsoft and hence *Proprietary*. Similarly, there is obviously a specific price point, most probably included in the underlying products(*Negligible-Cost-per-User* [+]. Further, the application only works in the Microsoft application landscape, or if at least Microsoft's authentication products (e.g. Azure Active Directory) are used. Hence the *Range-of-Use* is rated with [+].

To conclude, the main attack vector against Microsoft's Authenticator is the attack on the mobile device itself. Depending on the device, the security thresholds of the device's internal authentication might differentiate. Further, security features and quality of e.g. the biometric sensor might differentiate between different devices. *Liveness Detection* is one example of a security mechanism for biometric sensors which older devices might be lacking. It describes the possibility for a sensor to identify whether the provided biometric sample is derived from a live subject or e.g. only a copied version of a fingerprint. Without the possibility to ensure security standards for each device using the Authenticator, the system will only be as secure as the least secure device.

5.1.2 FIDO2

The Microsoft Authenticator application is based on the principles from the FIDO2 specification. Therefore most of the assessed benefits are the same as presented previously. However, some of the benefits are dependent on the type of implementation of the FIDO2 specifications. Therefore the following paragraphs assess those criteria that might differentiate in-between different implementations of FIDO2. The result of this assessment can be found in *Table 5.1*, where benefits that are not defined by the specification are represented as an empty field highlighting that those are the areas in which an implementation of the specification might provide benefits.

Memorywise-Effortless*, *Nothing-to-Carry*, *Resilient-to-Physical- Observation* and *Resilient-to-Targeted-Impersonation The general FIDO2 standard allows the usage of an authentication method of choice. This might be e.g. a PIN, a biometric treat, a FIDO keyfob, or USB-Stick. Obviously, for those services that offer the FIDO2 authentication without any specification on which authentication scheme to use, the apparent benefits can not be evaluated as those depend on the choice of the user. One example would be that a biometric scheme is *Memorywise-Effortless* and *Resilient-to-Physical-Observation* but **not** *Resilient-to-Targeted-Impersonation*.

Portability One of the basic principles behind FIDO authentication is that each device a user uses to login to a service is registered separately. I.e. if a web application offers FIDO authentication, an user can register a compatible USB-Stick on his computer to trigger the authentication. However if the user wants to use the FIDO sign in from another device such as his mobile phone, he will need to register the phone separately. Hence depending on the choice of the implementing system, registration might be needed as a one-time step for each device. This would obviously need to be considered when assessing *Portability*.

Accessible Similarly to the first criteria, the *Accessibility* depends on the implementation. If a system e.g. requires the user to use a passphrase for a company-issued mobile phone and that phone is used as a login factor for the web application, this might prove to be a barrier for e.g. illiterate people. Therefore the freedom of choice for the user has to be assessed for every system implementation on it's own.

Browser-Compatible With Microsoft's Authenticator, the user triggered an authentication request which gets processed by Microsoft's backend and ends up as a popup notification from the Authenticator application on the smartphone. Contrasting, the FIDO2 specifications require a FIDO-compatible browser. Normally during FIDO2 authentication attempts, the APIs of the WebAuthn and CTAP are used. Therefore only browsers implementing the specific APIs are capable of performing FIDO2 authentication. Currently, the APIs are implemented by *Windows 10* and *Android* platforms as well as *Chrome, Firefox*, and *Edge* browsers. [44]

Proprietary With today's possibility of open knowledge and communication via the world wide web, security products are improved by a multitude of experts reviewing products, code, and implementations. However, this approach works best for Open-Source products. Whenever a product or application uses proprietary code, protocols or applications this means that there is less possibility for public researchers to investigate the applications for potential vulnerabilities and issues. As the Microsoft Authenticator application shows, a product might use and implement the Open-Source FIDO2 specification but still contain proprietary content in the application. Therefore this criterion has to be assessed separately.

Resilient-to-Throttled-Guessing **and** ***Resilient-to-Unthrottled-Guessing*** Based on the implementation of the specification and the underlying system, the different applications and architectures might provide a different amount of resiliency against the specified attacks. There might be systems which are set-up to delete themselves after x amount of false logins and are thus resilient to both types of

attacks. However, there may be systems which do not offer any kind of protection against guessing attacks.

Resilient-to-Theft Obviously, an USB-Stick or FIDO keyfob might be more prone to theft (or to losing it) than a smartphone or a (desktop) computer is. Hence this criterion depends on the implementation of the system and the choices that are provided to the user.

Revokable-Biometrics The FIDO specification does not declare any specific information on how a biometric authentication on the user's device shall be handled. The usage or the lack of *Revokable-Biometrics* therefore highly depends on the device and implementation.

Range-of-Use Even though the FIDO2 specifications have been published previously, they are not yet adopted as widely as they should be. However, even if they were, the specification can be implemented in different ways that allow or do not allow the communication or usage with specific products only or all products supporting the standard. Hence, this benefit is rated on a per-product-basis.

Besides the previously outlined benefits, most of the benefits of the FIDO2 specification apply to every system implementing the protocols and specifications. One example of those might be the benefit of *Resilient-to-Phishing*. FIDO2 applications perform an origin check of the requests for authentication and only communicate via encrypted *nonces*. Hence a phishing attempt - might it be a redirection to a fake login website or sniffing communication content - will not be possible viable due to the specification itself.

5.1.3 ThumbSignIn

ThumbSignIn provides FIDO2-based authentication products. As ThumbSignIn relies on FIDO2 standards similarly to Microsoft Authenticator, the authentication and registration process is very similar. However, ThumbSignIn emphasizes the need for ensured security of the biometric templates. Therefore, ThumbSignIn applications make use of biometric templates. The biometric data itself is only available temporarily during the process of authentication or registration. The data is then being transferred into a template which can not be reverse-engineered to expose the original credentials. Further, ThumbSignIn relies on the usage of iOS *Face-ID/ Touch-ID* which is taking advantage of a secure enclave on the system to allow biometric keys to still be secure even if the operating system got corrupted. [62]

ThumbSignIn is rated [+++] for the benefit of *Revocable-Biometrics* as - due to the usage of a template - one could revoke such template and create a new template for the user in case of compromise of a smartphone and the secure enclave (as this is the only case in which the biometric template might get compromised). Further, ThumbSignIn (in a biometric configuration) unlocks the cryptographic keys by using biometric authentication rather than knowledge-based authentication as e.g. a PIN. Therefore, it is rated [+++] for the criterion of *Memorywise-Effortless*. In regard to the benefit of *Nothing-to-Carry*, the application is rated with [++] as a user obviously has to carry his smartphone. ThumbSignIn advertises the usage of their system with their application for any possible authentication scenario. Hence the system is rated [+++] for portability.

In terms of *Accessibility*, ThumbSignIn is only limited for people without smartphones or without fingerprints([++]). As ThumbSignIn is based on its own backend to redirect authentications to a mobile phone, the browser which is accessing the specific resource does not limit the possibilities. ThumbSignIn is a *Proprietary* authentication system but can be used for free ([+]). However, still, ThumbSignIn can only be used for services which offer their specific signin possibility. The *Range-of-Use* is hence be rated with [+]

The authentication application does not rely on the use of a password at all. Depending on the configuration the users might be needed to authenticate using biometric factors or via e.g. scanning a QR-Code with a registered device. All of the options are considered to be *Resilient-to-Physical-Observation*. However - once an impostor has gotten hold of a registered device - they are prone to *Targeted-Impersonation* attacks ([+]]). As no PINs or passwords are used, the system is considered *Resilient-to-Guessing*. Even when the mobile device is stolen, the system still required the user to authenticate to be able to use that phone ([++]). ThumbSignIn's biometric configuration uses e.g. *iOS FaceID/ TouchID* and hence relies on Apple's algorithms for the biometric matching which makes for a rating of [○] for the benefit of *No-Trusted-Third-Party*.

5.1.4 BioID

BioID features multiple design choices to underline their slogan *"privacy by design"*. The main aspects that form the pillars for BioID's security are: [63]

- **Irreversible, Revocable Biometric Templates** During the registration and signin process, BioID creates a template from the image data. As previ-

ously explained (c.f. *Section 2.2.3: Transmission and Storage Security of Biometric Information*), by transferring the picture data into a secure biometric template, reverse-engineering the original data becomes impossible. Further, if a template of a user becomes compromised, the template can be revoked and a new template for the user can be issued. I.e. a login with the old template is not possible anymore, but the service provider can re-enroll the user generating a new anonymous identifier and a new template.

- **Anonymous Face Recognition** To ensure the highest possible security, BioID stores the biometric templates anonymously. No other personal information such as a name or an email address is connected to a biometric template. Instead, BioID issues a randomly generated anonymous identifier to the user during registration.
- **Secure and Tailored Data Storage** Underlying the GDPR, BioID offers its service with carefully selected cloud providers only. Data centers are located close to the client to keep data e.g. within Germany for a German company. Further, BioID offers possibilities of a private hosted BioID Web Service that could be located as an on-premise solution directly in a customer's data center. Finally, BioID highlights the fact that highly-secured data centers are the better choice for storing secure data than a client's device which may be far more vulnerable to attacks.
- **High-Security Levels** BioID has issued multiple patents on anti-spoofing and liveness detection with pictures. With those patents, they have been able to create a standalone patented *photo fake defender*. Further, BioID's Web Service is performing texture-based detection of photo and video replays, both physical and digital. Finally, the service implements a challenge-response mechanism to perform motion-based fraud prevention.

Many of the previously assessed criteria frot Microsoft's Authenticator or Thumb-SignIn similarly apply to BioID's system as it is being assessed under the assumption that the mobile application is used. For example - as with the others - the scheme is *Memorywise-Effortless* as well as *Scalable-for-Users*. However during the authentication process e.g. when reacting to the challenge-response of the fraud prevention principles, the scheme is less *Physically-Effortless* and less *Efficient-to-Use*.

Based upon the findings from *Ruoti et al.* in terms of *User-Affection*, the scheme is only rated with [++] (compared to [+++] for the other systems) because the authentication is more of a hassle and might possibly prove to be difficult in some

situations. [58] If an application supports the BioID authentication, the authentication can be done in the same way from either a desktop webcam or a mobile phone, the portability is hence rated [+++].

In terms of benefits such as *Range-of-Use*, *Accessibility*, *Negligible-Cost-per-User*, *Server-Compatible*, and *Browser-Compatible* it achieves the same scores as the other systems. Even though BioID has developed many patents, its *Maturity* is rated behind the of FIDO2 to some extent. This is mainly caused by the amount of work from different companies and different experts that has been used to publish the FIDO2 specifications.

In regard to security benefits, the scheme is resilient against all type of attacks on knowledge. However, it might be prone to *Targeted-Impersonation* attacks. Due to the amount of work and mechanisms put in place to secure against impersonation attacks [64] [65], the scheme is rated [++] for that benefit. In contrast to the FIDO2 authentication mechanism, BioID appears to not have implemented sophisticated security mechanisms to prevent *Phishing*. In contrast to the other assessed principles, the assessment of the biometric data is performed by the server rather than on the remote device. Therefore attacks on the matcher are far more complicated and hence the *Trust-to-Third-Parties* is significantly lower than with the other systems ([+++]).

One major disadvantage of the system appears to be its transfer of the biometric data. As far as this work could investigate, the biometric data (pictures) is sent over an encrypted connection to the server. Only there, the data is being transformed into the irreversible biometric templates. Therefore, if an attacker was to sniff on that connection and was to eventually crack the encryption (or even perform Man-In-The-Middle attacks), he could get hold of the original biometric data. To protect the user, BioID transfers the data anonymously, without a connection to the user. This means that even if an attacker was able to compromise pictures, he would not directly have an understanding of whom the pictures belong to. Therefore the benefit of *Biometric-Data-Secrecy* is rated with [+].

5.1.5 Gluu

Gluu Server has been implemented as an exemplary remote biometric authentication system using a smartphone. Gluu's authentication mechanism is based on FIDO principles: The authentication server sends a cryptographic nonce which gets upon user confirmation encrypted with a private key stored on the device and send

back to the server. Therefore benefits such as *Physically-Effortless*, *Efficient-to-Use* or *Resilient-against-Phishing* are assessed as superimposed by the framework definition.

The Super Gluu mobile application with fingerprint authentication offers a *Memorywise-Effortless* authentication process ([+++]). However, similarly as with the other systems, the user has to carry his mobile device to perform the authentication (*Nothing-to-Carry*[++]). Similarly to the other assessed applications, Super Gluu is used independently of which system triggered the login (*Portability* [+++]). The Gluu Server can be used as an IdP for any service offering *OAuth2.0* Authentication which is more or less a standard in authentication systems and thus widely used. Therefore the Server rates [+++] in terms of *Range-of-Use*.

The rating of *Accessibility* is applied equally to the other systems as they all rely upon the use of smartphones as authenticating devices. Further, using the browser only to trigger the login process, it is rated [+++] in terms of *Browser-Compatibility*. One contrast is to be found in Gluu as open source product which obviously does not require any payment to install, test or expand the system (*Negligible-Cost-per-User* [+++]). Additionally, as per definition of Gluu Server being open source, it is rated [+++] in terms of *Non-Proprietary*.

In terms of security benefits, the implemented Gluu biometric authentication solution is rated [+++] in regard to *Physical-Observation* but only [+] for *Targeted Impersonation* (the attacker has to gain control of the device first). As no PINs or passwords are used at all, the system is considered *Resilient-to-Guessing* [+++]. When the device is stolen, the fingerprint of the user is still required making it resilient to theft ([++]). Super Gluu uses the iOS / Android alorithm's for biometric matching and is hence rated [○] for the benefit of *No-Trusted-Third-Party*. Finally, *Revocable-Biometrics* do not appear to be focused on in the standard iOS / Android authentication mechanisms ([○]).

5.2 Framework Assessment Results

The presented table (5.1) shows a good overview of the different analyzed systems, their similarities, and possible differentiation. As the *Benefits* from the framework are describing positive aspects, the *Benefits* which have received exectional results do not need further investigation. However, where the systems fail might be possible

	Benefits	*Analyzed Systems*				
		FIDO2	*MS Authenticator*	*ThumbSignIn*	*BioID*	*Gluu*
Usability	*Memorywise-Effortless*		++	+++	+++	+++
	Scalable-for-Users	+++	+++	+++	+++	+++
	Nothing-to-Carry		++	++	++	++
	Physically-Effortless	++	++	++	+	++
	Easy-to-Learn	++	++	++	++	++
	Efficient-to-Use	++	++	++	+	++
	Infrequent-Errors	○	○	○	○	○
	Easy-Recovery-from-Loss	++	++	++	+++	++
	User-Affection	+++	+++	+++	++	++
	Portability		+++	+++	+++	+++
	Range-of-Use		+	+	+	+++
Deployability	*Accessible*		+	++	++	++
	Negligible-Cost-per-User		+	++	+	+++
	Server-Compatible	○	○	○	○	○
	Browser-Compatible		+++	+++	+++	+++
	Mature	+++	+++	+++	++	+++
	Non-Proprietary		○	+	○	+++
Security	*Resilient-to-Physical-Observation*		○	+++	+++	+++
	Resilient-to-Targeted-Impersonation		+	+	++	+
	Resilient-to-Throttled-Guessing		++	+++	+++	+++
	Resilient-to-Unthrottled-Guessing		○	+++	+++	+++
	Resilient-to-Internal-Observation	○	○	○	○	○
	Resilient-to-Leaks-from-Other-Verifiers	++	++	++	+	++
	Resilient-to-Phishing	+++	+++	+++	○	+++
	Resilient-to-Theft		++	++	+++	++
	No-Trusted-Third-Party		○	○	+++	○
	Requiring-Explicit-Consent	+++	+++	+++	+++	+++
	Unlinkable	+++	+++	+++	+++	+++
	Revokable-Biometrics		○	+++	+++	○
	Biometric-Data-Secrecy	+++	+++	+++	+	+++

Table 5.1: Comparative Evaluation of the Assessed Authentication Systems

options to enhance the current technologies. An analysis of the results shows the following :

Infrequent-Errors Due to the nature of biometric systems, they are prone to errors. Especially the decision on a threshold for the *Matcher* or the *Decision-Making module* will make up for different results in this category. Especially due to intra-human variations, biometric systems will always only be able to perform based on a matching score. It is very unlikely that there will be a biometric system in the future which can match a person to a template as e.g. a password is matched to its hash value. Therefore there currently is no possibility to enhance the benefit of *Infrequent-Errors*. (Even if the consideration might be to reduce the threshold for the matching, this would mean there are fewer errors for legitimate users, but the *False-Match-Rate*, i.e. the cases in which an impostor would be allowed access would rise as well putting the security of the entire system at chance.)

Range-of-Use With the FIDO2 specifications, we are advancing in the right direction. However, there is still a long way to go to enhance the specifications for biometric authentication systems. Very often, one authentication system only works with its own mobile application. For multiple plattforms from multiple providers a user needs multiple authentication applications on his mobile device. Even though all the applications are built on the same specifications, an inter-application authentication process is still not possible.

Resilient-to-Internal-Observation Recalling from the original framework description, systems are not considered resilient against *Internal-Observation* attacks when the capturing of data from inside the device might lead to a compromise of the systems security. All kinds of smartphone-based biometric authentication systems are possibly prone to malware that modifies, alterers, and/or compromises raw biometric data as soon as it is captured by the sensor. Further, even if the biometric data would be transformed into a template before being compromised, whenever it is supposed to be used to authenticate, the basic device's functionalities have to be used. Under the assumption that every part of the device can be infected with malware, there currently is no way to secure against those kinds of attacks without implementing further complexity in the system (e.g. an additional One-Time-Passcode system).

Revokable-Biometrics A common practice should nowadays be the implementation of systems that perform the transformation of the biometric data into non-reversible biometric templates. This process provides two main advantages: Even

if a template became compromised, an attacker would not be able to retrieve the original data from the template and would thus be unable to use the data to perform *Targeted-Impersonation* attacks. Further, once the template becomes compromised, the template could be revoked and would thus be unusable for login to the same system anymore. The user could be issued a new template which ensures he can still authenticate without the possibilities of an attacker authenticating with the compromised template.

Biometric-Data-Secrecy A common misunderstanding today concerns the security of *secured* (i.e. encrypted) data. The common understanding is that encrypted data is safe. Which is only correct to some extent. Today's encryption algorithms rely on the fact that it is not possible to invert one-way-functions such as the discrete logarithms. However, this assumption is only correct for current computers and a limited timespan. Even a cryptographic algorithm which would need more than 1000 years to be cracked today might be easily crackable in just 20 years, as according to the *Moore's Law* computation power doubles every 18 Months. [66] Further, researchers still assume that once the development of large-scale quantum computers is enabled and boosted, current cryptographic protocols will be easily breached by the new form of computers. Therefore cryptographic algorithms are of good usage for passwords which are very likely to be changed within the next 20 years. However, when considering the properties of biometric data, the fact that those stay the same for our entire lives becomes important. I.e. the fingerprint that *YOU* have today will still be identical to your fingerprint in 20, 40 or 80 years. Therefore if an attacker was to capture a *secure* transmission of a user's biometric data today and decrypt it in 20 years, it is very likely that the attacker would have valid biometric data to log in to any system that is secured via those biometric treats. Therefore a proper policy for the avoidance of biometric data should be approached. This is especially applicable for transfers of biometric data through electronic networks. An aim should be to process biometric data where it gets created and only send it through a network in seldom cases (e.g. during a registration process in which the server obviously needs to receive the sample once). FIDO authentication principles implement this strategy in a very good way: biometrics are only used locally to unlock a store for cryptographic keys which are then sent to a *verifier*. Contrasting this practice, e.g. BioID sends the biometric data over the network during every authentication attempt. This aspect should be taken into close consideration when creating architectures of (remote) biometric authentication systems.

No-Trusted-Third-Party Many mobile authentication systems use the authenti-

cation methods provided by the device. That may be the Face/TouchID of Apple, fingerprint sensors of Samsung or any other authentication method. This is the proposed way by FIDO2: The user unlocks his phone using the local biometric authentication which then unlocks the secret credentials and uses those to authenticate against the server. However, by using those systems, the remote application needs to trust the user's device. Even in a case in which the mobile device might not be compromised, the security requirements for the biometric authentication system might be severely lower than the security requirement for the web application. One such factor could be e.g. a fingerprint sensor which does not perform a proper *liveness* detection and is hence easily fooled. Further, the thresholds for the authentication which have been specified might provide a high *False-Match-Rate*. I.e. many impostors would be granted access. To overcome the drawbacks of performing the biometric authentication on the mobile phone directly, the system architecture should rather consider the server's backend to perform the matching and decision-making process. That way, it is ensured that the remote system has the sovereignty of allowing or denying access according to principles which are defined in the backend. As explained previously, BioID performs such assessment where images are sent to the server and analyzed (fraud detection, matching, decision making...) in the system's backend. Due to the previously outlined proplems, drawbacks, difficulties, and vulnerabilities superimposed onto the system by using smartphones in the position as decision maker, no remote application should put the burden of decision making upon a mobile device. To underline the vulnerability of smartphones, the previously implemented exemplary system is examed closely with a malicious intend.

5.3 Breaking Mobile Phone Security using the Example of Super Gluu on Android

As previously discussed, relying on the mobile phone as a secure authenticator proves a risk. To assess the likelyhood of malicious usage of the mobile phones, in the upcoming sections a security analysis of the Android OS Super Gluu application for password-less biometric authentication is performed.

5.3.1 Preliminary Thoughts

The scenario in which this might be of interest is in a case in which a malicious entity (further referred to as *hacker* or *attacker*) has gotten control over a victim's (*user's*) mobile phone. This could be because the phone was lost , stolen or simply accessed in a moment of distraction. On his phone, the victim has set up biometric, password-less authentication for the Super Gluu application to authenticate his administrative access to his Wordpress application.

Once the attacker has gotten control over the physical device itself, he has to overcome a the lock screen. However, research and findings over the last couple of years has shown that there are plenty techniques and methods to do so. Some of which are using the *emergency call* feature to override the lock screen, booting into safe mode, or flashing other toolkits to delete the respective android files.[67][68] The key takeaway from the multitude of research is that the mobile phones security measures are always prone to be cracked. Hence, an access to the phone is always possible. [69][70][71][72][73][74][75] Assuming that a lock screen is no obstacle, the focus of this work is on cracking the secure application on the phone itself.

5.3.2 Overriding the Super Gluu Fingerprint Verification

To be able to analyze the application, it is of big use to test out the application and understand how it works. On the victim's device, trying to open the application fails as it is secured with the user's fingerprint. Therefore, the application is installed on another device for testing purposes. Quickly it becomes apparent that the fingerprint confirmation secures the access to the application itself. Closer research on the functioning of the application shows that the application implements FIDO-like standards. Apparently, the authentication request which is transmitted to the phone is being answered and signed using a cryptographic key pair which is located on the phone. Per the FIDO2 specification, the keys should be secured to require an additional action to unlock them. This is in the case of the phone done by pressing the *Approve* button (c.f. *Figure C.15: Android Super Gluu Authentication Approval*). Access to that button is secured via the biometric authentication when starting up the application.

Once the initial investigation has been finished, several possible attack-vectors have been identified:

- **Intercept** the incoming FIDO authentication request and try to approve it by accessing the credentials with root access on the phone.
- **Spoof** the fingerprint sensor (as described in previously mentioned researches [71][72][73])
- **Manipulate the System** to return a successful fingerprint verification for the attacker (c.f. attack vectors (2-6) from *Figure 2.1: Common Attack Factors on Biometric Systems (adapted from [15])* as mentioned in *Section 2.2.3: Attacks on Biometric Security Systems*)
- **Override** the fingerprint confirmation for access to the application (c.f. attack vector 1 of *Figure 2.1: Common Attack Factors on Biometric Systems (adapted from [15])*)

Once these possible attack-vectors have been outlined, the last one seems to be the most interesting one as findings from that assessment might be easily transferred to other Android applications with (biometric) security enforcement. Therefore the next step is to try to reverse-engineer the Super Gluu Android application. In Android, the applications are provided via Google's Play Store. Once an application is selected to be installed, the *.apk* file is downloaded and installed. The APK format is used to convert, package, and distribute android applications. Once the application is installed, it's package (*.apk)* file can be exported and transferred to any external system. This allows for analysis of the file on a computer.

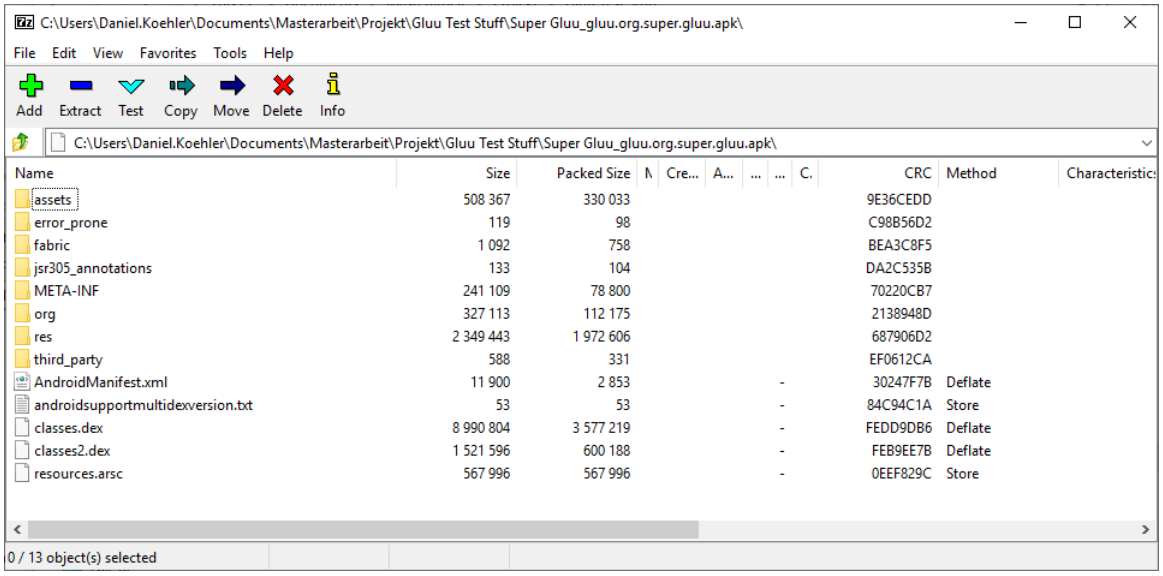


Figure 5.1: Inspecting the Super Gluu *.apk* archive

Inspecting the APK archive shows multiple folders and files. Some of which are the *AndroidManifest.xml* and *classes[2].dex* (c.f. *Figure 5.1*). As previously mentioned, the *AdnroidManifest.xml* contains configuration information of the application such as e.g. its permissions, name, and settings. The manifest further plays a key role in the security of an application. (c.f. *Section 2.3: Application's AndroidManifest.xml*) Recalling from *Section 2.3: Android OS Security*, for Android applications, the Java source code is transformed into bytecode-like formats using the DalvikVM. Those bytecode representations are found in the **.dex* files. However, inspecting those files with a standard text editor will not provide any useful information.

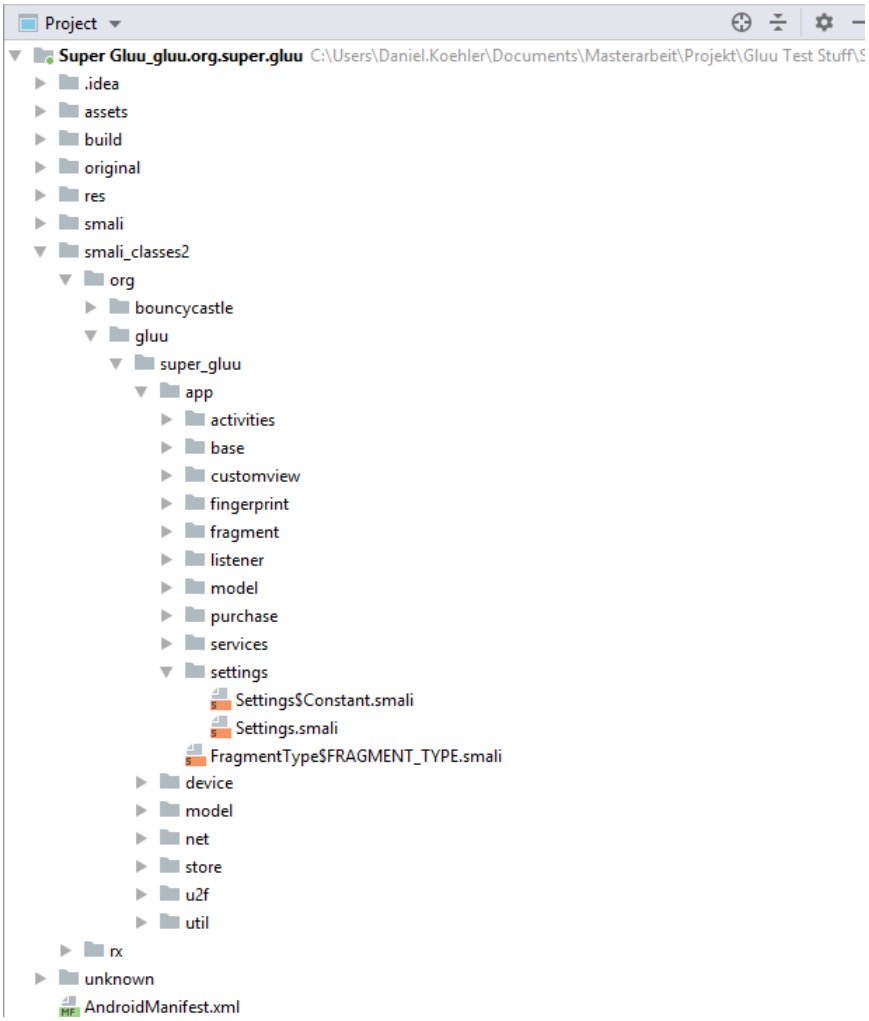


Figure 5.2: Inspecting the Decompiled Super Gluu Smali Project Files

To enable a proper investigation of the source code, the application needs to be decompiled. Decompilation refers to the process of trying to rebuild the original code that has been used to create bytecode by analyzing the bytecode and transforming

it back into its original state. However, this process is tedious and especially for larger software projects, proper decompilation will prove to be hardly possible in many cases. For Android, there are two methods of decompilation. One is the *dex2jar* [76] tool which can be used to disassemble the *.dex* files into their original *.class* Java representations. However, working with *dex2jar* often proved to be very tedious as the retrieved *.class* representation of the code is malstructured and unformatted. Further, rebuilding the application from the decompiled *.class* files is impossible. The tool could hence be used for code analysis but lacks functionalities when it comes to enabling modification and the rebuild of the application. Another software available is *apktool* [77]. Instead of attempting to retrieve the original *.class* representations of the application, *apktool* transforms the bytecode into a human-readable bytecode representation named smali. [78]

Now that the code has been decompiled, it can be inspected in its project-like structure. Within the project representation, one folder and the contained files sparked an especial interest: *settings.smali*. (c.f. *Figure 5.2*) One of the previously identified possible attack vectors is to override the biometric verification which secures the access to the application. One these is that it is likely that the usage of the biometric sensor and possibly the storage of biometric templates are configured in the settings. Therefore an in-depth analysis of the *settings.smali* source code is conducted.

```
.method public static isAuthEnabled(Landroid/content/Context;)Z
364   .locals 1
      .line 21
366   invoke-static {p0}, Lorg/gluu/super_gluu/app/settings/Settings;->
        getFingerprintEnabled(Landroid/content/Context;)Ljava/lang/Boolean;
      move-result-object v0
368   invoke-virtual {v0}, Ljava/lang/Boolean;->booleanValue()Z
      move-result v0
370   if-nez v0, :cond_1
      invoke-static {p0}, Lorg/gluu/super_gluu/app/settings/Settings;->
        getPinCodeEnabled(Landroid/content/Context;)Ljava/lang/Boolean;
372   move-result-object v0
      invoke-virtual {v0}, Ljava/lang/Boolean;->booleanValue()Z
374   move-result v0
      if-nez v0, :cond_1
376   invoke-static {p0}, Lorg/gluu/super_gluu/app/settings/Settings;->isAppLocked(
        Landroid/content/Context;)Ljava/lang/Boolean;
      move-result-object p0
378   invoke-virtual {p0}, Ljava/lang/Boolean;->booleanValue()Z
      move-result p0
380   if-eqz p0, :cond_0
      goto :goto_0
382   :cond_0
      const/4 p0, 0x0
384   goto :goto_1
      :cond_1
386   :goto_0
```

```
    const/4 p0, 0x1
388 :goto_1
    return p0
390 .end method
```

Listing 5.1: Excerpt from *settings.smali* showing the Method *isAuthEnabled*

```
.method public static isAuthPending(Landroid/content/Context;)Z
392   .locals 3
    const-string v0, "oxPushSettings"
394   const/4 v1, 0x0
    .line 203
396   invoke-virtual {p0, v0, v1}, Landroid/content/Context;->getSharedPreferences(
        Ljava/lang/String;I)Landroid/content/SharedPreferences;
    move-result-object p0
398   const-string v0, "0xRequestData"
    const/4 v2, 0x0
400   .line 204
    invoke-interface {p0, v0, v2}, Landroid/content/SharedPreferences;->getString(
        Ljava/lang/String;Ljava/lang/String;)Ljava/lang/String;
402   move-result-object p0
    if-eqz p0, :cond_0
404   const/4 v1, 0x1
    :cond_0
406   return v1
.end method
```

Listing 5.2: Excerpt from *settings.smali* showing the Method *isAuthPending*

Upon inspecting the code, the two methods, *isAuthEnabled* and *isAuthPending* appear interesting. Upon closer inspection, the *isAuthEnabled* method is called to investigate if and which authentication process should be used. (c.f. *Listing 5.1*) The other method checks whether an authentication process is still pending. (c.f. *Listing 5.2*) Closer inspection of the code shows that both methods evaluate various variables, read system & application configuration, and call other methods. However, the final result that is going to be returned - *v1* respectively *p0* - is defined in the conditional statements in the final lines. Currently, the return arguments are defined in the conditional blocks to either *const/4 p0, 0x0* (which is the equivalent to Java's *boolean p0 = false;*) or *const/4 p0, 0x1* (which is the equivalent to Java's *boolean p0 = true;*). Judging by the funtion's names, an attacker trying to overcome the authentication of the application would want the variables to return a boolean **false** for both cases. This would be interpreted as: *There is no authorization needed, no authentication is pending, and none is enabled.* In this situation, the app should unlock without requiring the biometric verification. To enable this, the lines in which the return variables are set to the boolean *true* are edited to the smali code *const/4 p0[v1],* ***0x0*** representing **false**. That way - independent of what the remainder of the function is evaluating, in any case a boolean **false** is returned.

(c.f. *Listing A.2: Excerpt from settings.smali showing the altered isAuthEnabled and isAuthPending Methods*) That way an attacker should be able to login without needing to confirm the (user's) fingerprint. Finally, the app needs to be installed to the Android device.

5.3.3 Installing the Modified Application.

Now that the files have been altered, an Android package (*.pkg*) needs to be rebuild from the folder structure. To do so, the same tool as for the decompilation - *apktool* - is used. The commandline tool is called using

 # apktool b -r "Super Gluu Modified_gluu.org.super.gluu"
-o "super_gluu_modified.org.super.gluu" -p "/tmp/apktoolFrameworks"

In detail, this does the following:

- **b** is used to **b**uild an *.apk* file
- **-r** is used to specify which **r**esources to use. During the decompilation, the framework resources haven been saved seperately in a **p**ath specified with the **-p** switch and have not been decompiled. Now, during the rebuild process, the modified sources are build together with the original **r**esource files. This process is less error-prone as the original framework files are not altered.
- **"Super Gluu Modified_gluu.org.super.gluu"** specifies the input folder name of the project
- **-o** specifies the **o**utput name to be *super_gluu_modified.org.super.gluu*

Once the process finishes execution, the created *.apk* can be found under the specified name. The next step is installing the package on the device. To enable better debugging and error understanding, the Android-Debug-Bridge (ADB) is used to evaluate errors, view device information, and manage installation or deletion of applications on the phone. ADB is a command-line tool which is used to handle communication with a device. This contains aspects such as pushing or pulling data but just as well enabling a (remote) command-line execution environment on the phone. [79] To allow a computer use a mobile device with ADB, the functionality *Universal Serial Bus (USB)-Debugging* from the Android developer settings has to be enabled first. [80]

However, before an installation can be triggered, few other things need to be considered: If an installation would be attempted in the current stage, the following

problem would arise: Currently there is the **original** application installed. Neither the version nor name of the application have been modified. The Android installation would hence fail as it is not considered an update to the current application. There are two possible solutions at hand to overcome this problem.

First, the *AndroidManifest.xml* could be modified to change the version of the application. This however bears the difficulty that the *AndroidManifest.xml* is secured with the digital certificate and signature of the original developers. (c.f. *Section 2.3: Application's AndroidManifest.xml*) If the manifest would be edited and signed with another signature, Android's security enforcement would install the app into another sandbox. (c.f. *Section 2.3: Further Security Enhancements*) Thus rendering the app useless as the stored login certificates of the victim can not be accessed from inside another sandbox.

```
C:\Users\Daniel.Koehler\Documents\Masterarbeit>adb uninstall -k gluu.org.super.gluu
The -k option uninstalls the application while retaining the data/cache.
At the moment, there is no way to remove the remaining data.
You will have to reinstall the application with the same signature, and fully uninstall it.
If you truly wish to continue, execute 'adb shell cmd package uninstall -k'.

C:\Users\Daniel.Koehler\Documents\Masterarbeit>adb shell cmd package uninstall -k gluu.org.super.gluu
Success
```

Figure 5.3: Inspecting the ADB *uninstall* ***-k*** option

Second, one these it that it might be possible to uninstall the original application and reinstall the modified version without modifying the application's name or version. This process could install the app in the same sandbox as the original application. However, when uninstalling an application, Android usually cleans up all of the application's data. In that case our reinstalled application would be missing the victim's security certificates as well. The solution for this problem is ADB's *uninstall* ***-k*** switch. According to the documentation, this switch shall be used for development purposes only as it will keep the applications data and only uninstall the *.apk* package. (c.f. *Figure 5.3*)

```
C:\Users\Daniel.Koehler\Documents\Masterarbeit>adb install "Super Gluu Modified_gluu.org.super.gluu.apk"
Performing Streamed Install
adb: failed to install Super Gluu Modified_gluu.org.super.gluu.apk: Failure [INSTALL_PARSE_FAILED_NO_CERTIFICATES:
Failed to collect certificates from /data/app/vmdl1730176413.tmp/base.apk: Attempt to get length of null array]
```

Figure 5.4: ADB Installation Failed due to *NO_CERTIFICATES*

Once the original application has been uninstalled, the modified version can be installed. However, the installation fails and the error that is thrown is caused by the fact that the modified *.apk* that has been created does not contain any certificates.

(c..f *Figure 5.4*) Recalling from *Section 2.3: Android OS Security*, Android installs applications in a sandbox environment depending on the certificate that has been used to sign the application. While in early versions of Android an installation was able to be performed without providing certificates, current versions prohibit installing unsigned applications due to possible security risks. The application has therefore to be signed to allow it to be installed. By definition of a secure certificate though, there is no possibility to sign the application with the original certificate. The application can hence be only signed with our own certificate but will - by doing so - not be placed in the same sandbox as the original application. One solution to that problem might thus be to overcome the Android signature checking.

5.3.4 Disabling Signature Checking

The *.apk* file can be signed using the Java executable *SignApk.jar* which works similarly to Google's original *apksigner* [81]. The Java application uses a certificate (*.pem*) as well as a keyfile (*.pk8)* to sign an *.apk* package. To be able to install this signed application into the same sandbox as the original Super Gluu application, the Android Signature checking needs to be disabled.

By default, the Android system, users, applications, and processes do only have limited privileges. Similarly to other Unix OSs, there is a *root* user available on the system but that user cannot be accessed by any application or process. The signature checking specifications are system-wide defined arguments and a standard user is not able to access or modify those settings. Therefore, a special procedure is applied that frees the operating system of the Android restrictions and enables the usage of a *root* user. This process is called *rooting* a phone. In most cases it works by uploading, flashing and installing custom software to the phone during the boot up process.

The mobile phone under assessment for this work is a *Samsung Galaxy S7 Edge* with a device identifier of *SM-G935F*. For this phone, the software to flash the custom firmware is called *Odin*. [82] The custom firmware chosen is called *Chainfire (CF) Auto Root*. [83] On the device at hand, the Odin downloader is accessed by holding the Volume-Down, Home and Power buttons while the phone is turned off. (c.f. *Figure 5.5*) Once connected to a computer, the CF firmware can be flashed onto the device. After a reboot, the device can be accessed and a new application has been added - the *SuperUser*. This application manages access to the newly unlocked root account. (c.f. *Figure 5.6*)

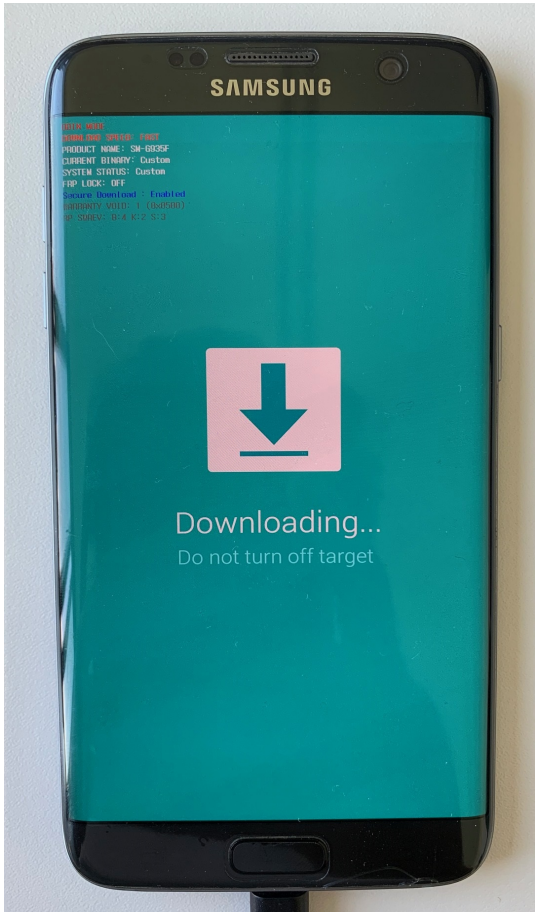


Figure 5.5: Samsung Mobile Phone in Odin Downloader Menu

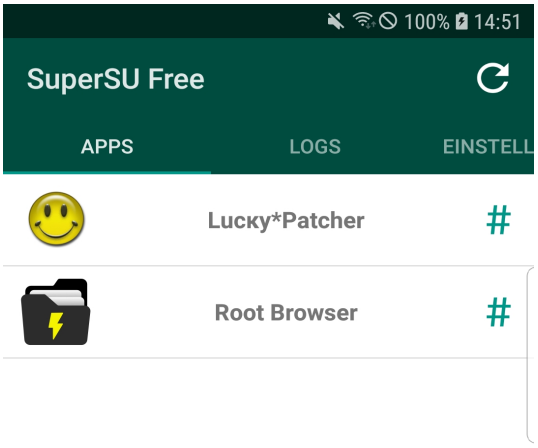


Figure 5.6: Android *SuperUser* Application

Finally, the setting which specifies the information for the signature checking is not to be found in the standard settings. It is rather interlaced into the original android system packages. Theoretically one would need to extract the system packages, decompile them and recompile them like it was performed earlier for the Super Gluu application. However, luckily there are tools which automate this process. One of which is called *LuckyPatcher* [84].

LuckyPatcher allows to easily modify (*patch*) several system settings or application's settings. LuckyPatcher can further be used to patch games allowing direct access to purchasable in-game content. Further, it allows to bypass the various advertisements that normally accompany free apps. From inside the LuckyPatcher menu, the options *Toolbox* and *Patch to Android* can be used to be provided with a menu which allows disabling the Android system's signature checking. Once the patches have been selected, the mobile device restarts a couple of times. A final check of the patches in LuckyPatcher will show all of them applied. (c.f. *Figure 5.7*)

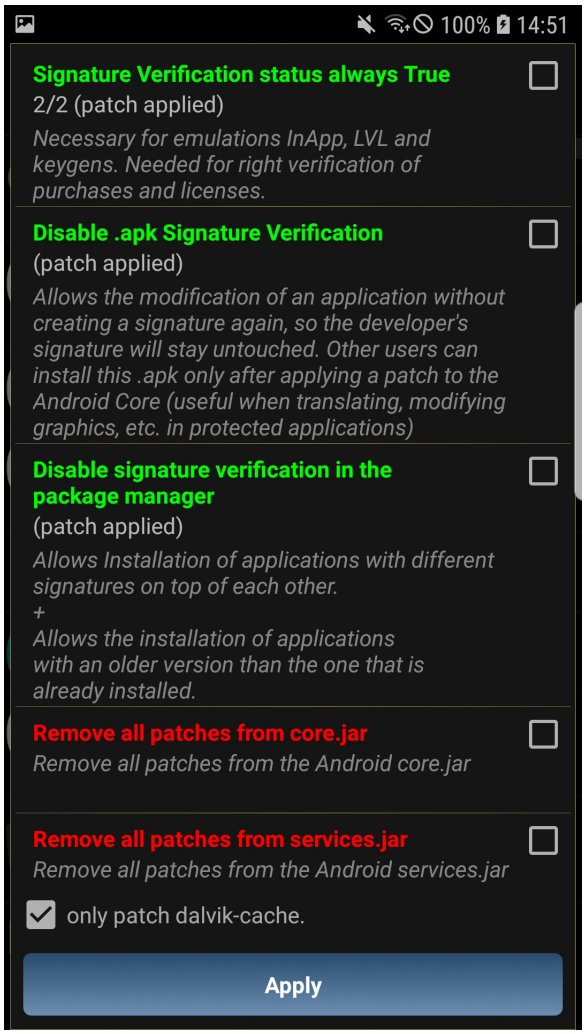


Figure 5.7: Android *LuckyPatcher* Signature Disabling

5.3.5 Putting Everything Together

```
C:\Users\Daniel.Koehler\Documents\Masterarbeit>adb install signed_modified_gluu.org.super.gluu.apk
Performing Streamed Install
Success
```

Figure 5.8: ADB Install of the Signed and Modified Application Package

Now that the Android Signature checking has been disabled, the self-signed *.apk* that has been created earlier can be installed on the device. This is done with the help of ADB again. Shortly after issuing the command, the function returns a *success* notification indicating that the app has been installed. (c.f. *Figure 5.8*) The Android system does not allocate sandboxes according to an application's signature anymore but rather according to it's name. Hence, the newly installed application has been places in the same sandbox as the the original application resided in beforehand. The new application can thus access original files such as the cryptographic keys for the authentication process.

Upon opening the application, the functions that have previously been modified are executed. Instead of returning *true* boolean values for specifications such as *isAuthEnabled*, the modified application is returning *false* values. Therefore, using the application is now possible without the need for a fingerprint verification. Upon opening up the application, a popup asking to *"Add a PIN Code"* is shown. (c.f. *Figure 5.9*) This is due to the fact that the application is under the impression that there is no security measure set up. The popup can be easily ignored by pressing the *No* button. This opens the home screen of the Super Gluu application. Upon trying to authenticate with the victims account now, the wordpress application issues a notification to the victim's phone as previously depicted. (c.f. *Section 3.3.4: Authentication Process*: *Figure 3.12: Android Notification from Super Gluu Application for Authentication*) Upon opening the notification, the former login screen opens, asking the attacker to set up a PIN code. Once ignored, the application provides us with the confirmation screen to perform the authentication. (c.f. *Figure 5.10*) Once approved, the attacker is provided with the standard confirmation screen in the application and is authenticated with the web application. The Wordpress admin panel greets the attacker without knowledge of a password, PIN, or possession of the victim's fingerprint. Simply because of the trust that the Identity Provider (Gluu Server) puts upon the mobile device.

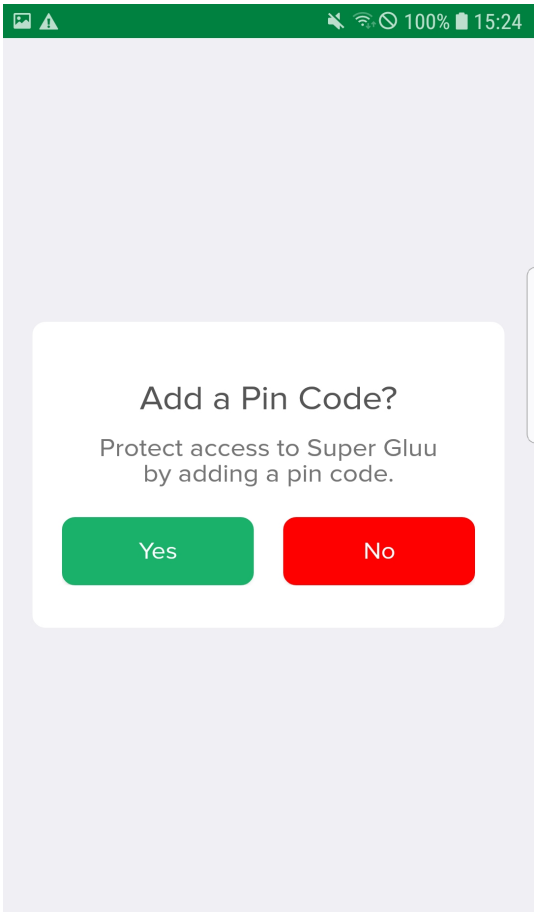


Figure 5.9: Super Gluu Application proposing to add a PIN Code

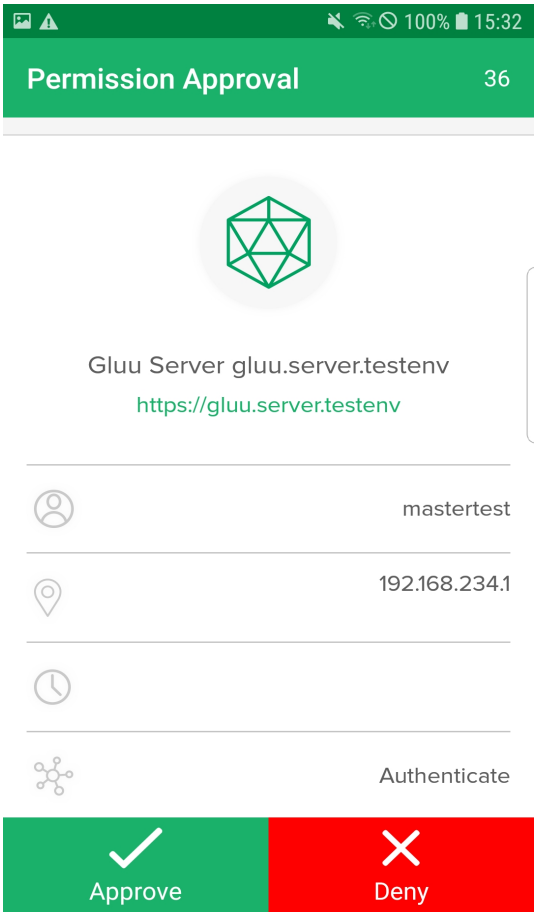


Figure 5.10: Super Gluu Application requesting Approval of the Authentication

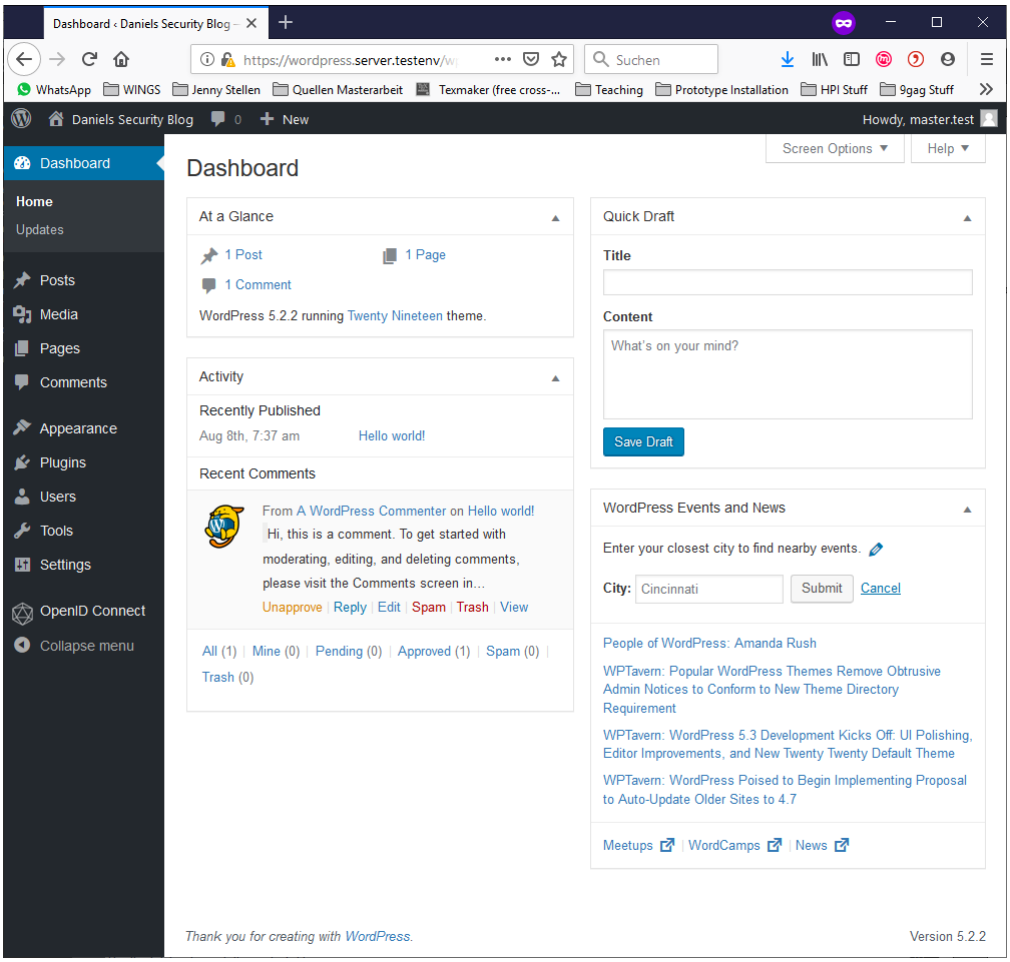


Figure 5.11: Wordpress Admin Panel Logged in by overriding the Fingerprint Authentication

6 Conclusion

Back during the invention of the password by *Fernando Corbató*, not only knowledge-based authentication schemes were available. However, at that time, using knowledge-based authentication has offered several advantages over methods such as possession-based or biometric authentication. Some of those could be attributed to it's ease of implementation and efficiency. Further, the drawbacks of passwords are much less-faceted and single-dimensional. Once a password is stolen, a new one is issued. People could be observed while typing it or the password could be guessed but increasing the password length made such attacks way harder.

Biometric authentication has only started to boom in recent years as the popularity of smartphones helped the spread of the sensors. For the user's, the main aim is ease of use. To make authentication safer while keeping it extremely easy to use, researchers have over the last couple of years started to develop more and more advanced security systems. Due to the variance in the architecture or functioning of the different authentication systems, they can hardly be compared. A structured approach is needed. One such is the framework proposed by *Bonneau et al.* in their paper *"The Quest to Replace Passwords"* [56].

This work analyzes the provided framework for assessing authentication systems and adjusts it to modern standard as well as extends it with specific criteria for the assessment of remote biometric authentication systems. Judging from the results of the analysis of four different applications and systems - commercial, free, and open source - as well as a framework to specify those authentication mechanisms, some fundamental observations can be investigated and differentiated.

Some of the main takeaways from the assessment are the fact that biometric authentication systems are prone to many common attacks such as guessing, brute force, phishing or physical observation. However, the analyzed systems share some drawbacks. Most of the systems do not implement any principle of *revokable biometrics* or at least apply *robust hash functions* or *similarity-preserving transformations* to the biometric data. This feature does not necessarily enhance the security of the authentication system itself but it enhances the security of a user's biometric data.

Given the fact that a user most probably has to live for another couple decades with the same fingerprint, preserving the security and secrecy of that data should be underlying topmost efforts. Further and similarly important, in the aim to standardize the authentication process, the FIDO2 specification has not taken into account that their proposed system superimposes the responsibility of performing the biometric authentication to the remote devices. The remote device at hand for most users is with high probability a standard smartphone without additional security or encryption options.

The theory of using the mobile device to create a secure two-factor authentication system is comprehensible but lacks thorough assessment. Theoretically, the possession of the smartphone proves to be a first factor while the e.g. fingerprint is a second factor (biometric). However, as this work shows, the fingerprint sensors and security mechanisms implemented in smartphones can easily be fooled to make the biometric verification *quasi-non-existent*. Therefore, the *quasi-secure* two-factor authentication using a smartphone becomes as weak as a key which solely has to be stolen to be retrieving control over the victim's account and personality. This is underlined by the attack against a victim's password-free remote biometric authentication with a web server using his mobile phone. Upon receiving control about the phone, the security measures installed inside Android OS could easily be overcome and the biometric authentication broken, providing the attacker with access to the victim's account.

Judging from the conducted research, provided examples, and performed analysis, the main criteria that modern biometric authentication systems should feature in terms of security aspects are **sovereignity of the authentication** as well as **biometric data secrecy**. The sovereignity to decide which user gets authenticated should always be the responsibility of the server and not of a roaming device. Further, the secrecy and security of biometric data should never be put at stake for the sake of easier implementation or lower system requirements. It should always be the top priority of a system. Always remember, biometric data does not change. It is our responsibility to secure the data *today* in a way that will still be considered secure in *decades*.

7 Outlook

Having assessed that the current frameworks and applications do not provide a satisfactory amount of security for both - the service provider as well as the user, research and development is needed to advance on our *Quest to Replace Passwords*.

Concretely, this work proved that the currently available principles are lacking safety mechanisms of one sort or the other. Be it the lack of secrecy of the biometric data or too much trust which is put upon third parties or mobile devices.

Assuming the authentication shall be performed on the server side, the main difficulty proves to be only the client (e.g. a smartphone) having access to the user's fingerprint. Obviously, at least for some initial registration of a new user with the system, the original fingerprint (or a Revocable, Non-Revertible Transformation of it) needs to be transferred to the server. However that process could possibly be secured by transferring the data on e.g. a USB-Stick or via Near-Field-Communication. Therefore the initial registration process provides a possible attack vector. However, this process only happens once.

In a perfect world, an authentication attempt is performed multiple times per day. This refers to the process of confirming and comparing the biometric treat from the user against the originally specified template on the server. If that exact process was being performed in a highly secure way, this would enhance biometric authentication tremendously. Obviously the most straight-forward solution would be to create a biometric authentication system which accesses the biometric sensor, reads the user's data and immediately transforms it into it's *Revocable, Non-Revertible Template* inside the device's *Trusted Execution Environment* and transfer that in encrypted form to the server to perform the matching and decision making process.

However, with quantum computing around the corner, no one can precisely determine whether a secure transmission of data will still be secure in a post-quantum world. Possibly, the new methods of quantum computing allow decryption of previously properly encrypted transmissions. Just as well, a quantum computer might

be able to break a *Non-Revertible Template*. In that case, quantum computing processes might be able to retrieve the original data from a formerly secure template. How long will it take for technologies to advance to this level? Nobody can predict with certainty, but our fingerprints will still be the same in 60 years.

What if it were possible to not transfer the biometric data via a network at all? Previously explained *zero-knowledge proofs* which are already used to perform assessments on the knowledge of a certain secret without transmission of the secret itself. Can such a principle be applied to biometric templates? [48] This work hence proposes research in the area of potential ZKP-like principles for biometric templates, allowing a biometric data secrecy by design.

Bibliography

[1] Le Bras, T.: *Online Overload – It's Worse Than You Thought.* https://blog.dashlane.com/infographic-online-overload-its-worse-than-you-thought/. Version: 2015

[2] Aman: *What smartphone finger print sensor came first?* https://www.quora.com/What-smartphone-finger-print-sensor-came-first. Version: 2019, Accessed: 25th July 2019

[3] McMillan, R.: *The World's First Computer Password? It Was Useless Too.* https://www.wired.com/2012/01/computer-password/. Version: 2012

[4] O'Gorman, L.: Comparing Passwords, Tokens, and Biometrics for User Authentication. In: *Proceedings of the IEEE* (2003)

[5] Chaum, D. ; Evertse, J. ; Graaf, J. van d.: An Improved Protocol for Demonstrating Possession of Discrete Logarithms and Some Generalizations. Advances in Cryptology. In: *Lecture Notes in Computer Science, EuroCrypt '87* (1987)

[6] Blum, M.: How to Prove a Theorem So No One Else Can Claim It. In: *Congresso Internacional De Matemáticos: Proceedings* (1986)

[7] Goldwasser, S. ; Micali, S. ; Rackoff, C.: The Knowledge Complexity of Interactive Proof Systems. In: *Society for Industrial and Applied Mathematics* (1989)

[8] Goldreich, O. ; Oren, Y.: Definition and Properties of Zero-Knowledge Proof Systems. In: *Journal of Cryptology* (1994)

[9] Morris, R. ; Thompson, K.: Password Security:A Case History. In: *Communications of the ACM* (1979)

[10] Jaeger, D. et a.: Analysis of Publicly Leaked Credentials and the Long Story of Password (Re-)use. In: *PASSWORDS'16* (16)

[11] German, E.: *The History of Fingerprints*. http://onin.com/fp/fphistory.html. Version: 2019

[12] Kreuseler, M.: *Angewandte biometrische Systeme*. 2018

[13] Jørgensen, Hakon: *Distant identification*. Arnold Busck, Copenhagen, 1922

[14] Ambalakat, P.: Security of Biometric Authentication Systems. In: *21st Computer Science Seminar* (2005)

[15] Ratha, K. ; Connell, J ; Bolle, M.: An Analysis of Minutiae Matching Strength. In: *Proc. Third Int'l. Conf. Audio- and Video-Based Biometric Person Authentication* (2011)

[16] Sutcu, Y. ; Li, Q. ; Menon, N.: Secure Biometric Templates from Fingerprint-Face Features. In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2007)

[17] Connie, T. et a.: Palmhashing: a novel approach for cancelable biometrics. In: *Information Processing Letters* (2005)

[18] Kevenaar, T. et a.: Face recognition with renewable and privacy preserving binary templates. In: *Fourth IEEE Workshop on Automatic Identification Advanced Technologies* (2005)

[19] Tehod, A. ; Ngo, D. ; Goh, A.: Personalised cryptographic key generation based on facehashing. In: *Computers and Security* (2004)

[20] Ratha, N. ; Connell, J. ; Bolle, R.: Cancelable Biometrics: A Case Study in Fingerprints. In: *18th International Conference on Pattern Recognition* (2006)

[21] Le, T. et a.: Protecting Biometric Features by Periodic Function-Based Transformation and Fuzzy Vault. In: *Transactions on Large-Scale Data- and Knowledge-Centered Systems XVI* (2014)

[22] Dodis, Y. ; Reyzin, L. ; Smith, A.: Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In: *Eurocrypt, Volume 3027 of LNCS* (2002)

[23] Li, Q. ; Setcu, Y. ; Menon, N.: Secure Sketch for Biometric Templates. In: *Advances in Cryptology – ASIACRYPT 2006* (2006)

[24] Yang, G. et a.: Two-factor mutual authentication based on smart cards and passwords. In: *Journal of Computer and System Sciences* (2008)

[25] IDC: *OS Data Overview*. https://www.idc.com/promo/smartphone-market-share/os. Version: 2019

[26] Marketshare, Net: *Operating System Market Share*. https://www.netmarketshare.com/operating-system-market-share.aspx? Version: 2019

[27] Brain, App: *Number of Android apps on Google Play*. https://www.appbrain.com/stats/number-of-android-apps. Version: 2019

[28] Faruki, P. et a.: Android Security: A Survey of Issues, Malware Penetration, and Defenses. In: *IEEE Communications Surveys and Tutorials* (2014)

[29] Chebyshev, V.: *Mobile malware evolution 2018*. https://securelist.com/mobile-malware-evolution-2018/89689/. Version: 2019

[30] AOSP: *Secure an Android Device*. https://source.android.com/security/. Version: 2019

[31] Vitas, M.: *ART vs Dalvik - introducing the new Android runtime in KitKat*. https://infinum.co/the-capsized-eight/art-vs-dalvik-introducing-the-new-android-runtime-in-kit-kat. Version: 2013

[32] Sanz, B. et a.: PUMA: Permission Usage to Detect Malware in Android. In: *International Joint Conference CISIS12-ICEUTE12-SOCO12 Special Sessions* (2012)

[33] Android: *Android Application Signing*. https://source.android.com/security/apksigning. Version: 2019, Accessed: 26th August 2019

[34] Oberheide, J. ; Miller, C.: *Dissecting the Android Bouncer*. https://jon.oberheide.org/blog/2012/06/21/dissecting-the-android-bouncer/. Version: 2012

[35] Google: *Android Security & Privacy - 2018 Year in Review*. https://source.android.com/security/reports/Google_Android_Security_2018_Report_Final.pdf. Version: 2019

[36] AOSP: *Trusty TEE*. https://source.android.com/security/trusty. Version: 2019

[37] Microsoft: *Microsoft Authenticator*. https://www.microsoft.com/de-de/account/authenticator. Version: 2019, Accessed: 08th July 2019

[38] Simons, A.: *No password, phone sign in for Microsoft accounts!* https://techcommunity.microsoft.com/t5/Azure-Active-Directory-Identity/No-password-phone-sign-in-for-Microsoft-accounts/ba-p/245254. Version: 2018, Accessed: 08th July 2019

[39] ThumbSignIn: *Any Authentication, Any Channel!* https://thumbsignin.com/. Version: 2019, Accessed: 10th July 2019

[40] Lakshman, V.: *The future of biometric authentication lies beyond mobile apps.* https://www.biometricupdate.com/201812/the-future-of-biometric-authentication-lies-beyond-mobile-apps. Version: 2018, Accessed: 10th July 2019

[41] BioID: *BioID - Be recognized.* https://www.bioid.com/. Version: 2019, Accessed: 23rd July 2019

[42] Gluu: *Gluu Homepage.* https://www.gluu.org/. Version: 2019, Accessed: 20th August 2019

[43] Alliance, FIDO: *Board Member Perspectives.* https://fidoalliance.org/members/board-member-perspectives/. Version: 2019, Accessed: 15th July 2019

[44] Alliance, FIDO: *Alliance Overview.* https://fidoalliance.org/overview/. Version: 2019, Accessed: 15th July 2019

[45] Alliance, FIDO: *FIDO Alliance Today: Status and News.* https://www.slideshare.net/FIDOAlliance/fido-alliance-today-status-and-news. Version: 2019, Accessed: 15th July 2019

[46] Alliance, FIDO: *The Value of FIDO Alliance Membership.* https://www.slideshare.net/FIDOAlliance/the-value-of-fido-alliance-membership. Version: 2019, Accessed: 15th July 2019

[47] Chong, Jerrod: *10 Things You've Been Wondering About FIDO2, WebAuthn, and a Passwordless World.* https://www.yubico.com/2018/08/10-things-youve-been-wondering-about-fido2-webauthn-and-a-passwordless-world/. Version: 2019, Accessed: 16th July 2019

[48] Kikuchi, H. et a.: Privacy-preserving similarity evaluation and application to remote biometrics authentication. In: *Soft Comuting, Volume 14 Issue 5* (2009)

[49] Ubuntu: *Ubuntu Homepage*. https://ubuntu.com/. Version: 2019, Accessed: 20th August 2019

[50] Nagy, B.: *Raising the Maximum Number of File Descriptors*. https://underyx.me/articles/raising-the-maximum-number-of-file-descriptors. Version: 2015, Accessed: 20th August 2019

[51] Gluu: *Gluu Server Installation Instructions*. https://gluu.org/docs/ce/3.1.6/installation-guide/install/. Version: 2019, Accessed: 20th August 2019

[52] Wordpress: *Wordpress Homepage*. https://de.wordpress.org/. Version: 2019, Accessed: 20th August 2019

[53] Boucheron, B.: *How To Install WordPress with LAMP on Ubuntu 18.04*. https://www.digitalocean.com/community/tutorials/how-to-install-wordpress-with-lamp-on-ubuntu-18-04. Version: 2018, Accessed: 20th August 2019

[54] Foundation, OpenID: *Welcome to OpenID Connect*. https://openid.net/connect/. Version: 2019, Accessed: 21st August 2019

[55] Gluu: *oxd 3.1.2 Documentation*. https://gluu.org/docs/oxd/3.1.2/. Version: 2019, Accessed: 21st August 2019

[56] Bonneau, J. et a.: The Quest to Replace Passwords: A Framework for Comparative Evaluation of Web Authentication Schemes. In: *2012 IEEE Symposium on Security and Privacy* (2012)

[57] Bonneau, J. et a.: The quest to replace passwords: a framework for comparative evaluation of Web authentication schemes / University of Cambridge, Computer Laboratory. Version: März 2012. https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-817.pdf. 2012 (UCAM-CL-TR-817). – Forschungsbericht

[58] Ruoti, S. ; Roberts, B. ; Seamons, K.: Authentication Melee: A Usability Analysis of Seven Web Authentication Systems. In: *Proceedings of the 24th International Conference on World Wide Web* (2015)

[59] Alaca, F. ; Abdou, A. ; Van Oorschot, P.: Comparative Analysis and Framework Evaluating Mimicry-Resistant and Invisible Web Authentication Schemes. In: *IEEE Transactions on Dependable and Secure Computing* (2019)

[60] FIDO-Alliance: *Microsoft's Path to Passwordless*. https://de.slideshare.net/FIDOAlliance/microsofts-path-to-passwordless-fido-authentication-for-windows-azure-active-directory. Version: 2018, Accessed: 08th July 2019

[61] Microsoft: *Password-less Protection*. https://query.prod.cms.rt.microsoft.com/cms/api/am/binary/RE2KEup. Version: 2018, Accessed: 09th July 2019

[62] ThumbSignIn: *Why your biometric data should never get hacked*. https://thumbsignin.com/blog/why-your-biometric-data-should-never-get-hacked/. Version: 2019, Accessed: 10th July 2019

[63] BioID: *BioID - Securing privacy by design*. https://www.bioid.com/securing-privacy-by-design/. Version: 2019, Accessed: 23rd July 2019

[64] Frischholz, R. ; Strohm, P.: *Method for discriminating between a real face and a two-dimensional image of the face in a biometric detection process*. 2012

[65] AG, BioID: *Method For Discriminating Between A Real Face And A Two-Dimensional Image Of the Face In A Biometric Detection Process*. 2012

[66] Moore, G.: Cramming more components onto integrated circuits. In: *Intel Electronics, Volume 38* (1965)

[67] Shariar, K.: *How To Bypass Android Pattern Lock, Password/PIN or lock screen security Without Wiping Data*. https://getandroidstuff.com/how-to-bypass-android-lock-screen/. Version: 2019, Accessed: 22nd August 2019

[68] McLaughlin, M.: *How to Bypass Android Lock Screen Using Emergency Call*. https://www.lifewire.com/bypass-android-lock-screen-using-emergency-call-4178718. Version: 2019, Accessed: 22nd August 2019

[69] Brandom, R.: *Your phone's biggest vulnerability is your fingerprint*. https://www.theverge.com/2016/5/2/11540962/iphone-samsung-fingerprint-duplicate-hack-security. Version: 2016, Accessed: 22nd August 2019

[70] Moore, M.: *Now even your smartphone's fingerprint sensor can be HACKED*. https://www.express.co.uk/life-style/science-technology/791055/smartphone-fingerprint-scanner-hacked-criminals-scan-data. Version: 2017, Accessed: 22nd August 2019

[71] Zorabedian, J.: *Your smartphone fingerprint reader could be hacked using paper and ink.* https://nakedsecurity.sophos.com/2016/03/08/your-smartphone-fingerprint-reader-could-be-hacked-using-paper-and-ink/. Version: 2016, Accessed: 22nd August 2019

[72] frank: *Chaos Computer Club breaks Apple TouchID.* https://www.ccc.de/en/updates/2013/ccc-breaks-apple-touchid. Version: 2013, Accessed: 22nd August 2019

[73] Cao, K. ; Jain, A.: Hacking Mobile Phones using 2D Printed Fingerprints / Michigan State University. Version: Februar 2016. https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-817.pdf. 2016 (MSU-CSE-16-2). – Forschungsbericht

[74] oli: *Sicherheits-GAU bei Nokia: So einfach lässt sich dein Handy von jedem entsperren.* https://www.watson.de/digital/smartphone/586622213-sicherheits-gau-bei-android-so-einfach-laesst-sich-dein-nokia-handy-von-jedem-entsperren. Version: 2013, Accessed: 22th August 2019

[75] Whittaker, Z.: *Hackers can remotely steal fingerprints from Android phones.* https://www.zdnet.com/article/hackers-can-remotely-steal-fingerprints-from-android-phones/. Version: 2015, Accessed: 22nd August 2019

[76] Panxiaobo: *dex2jar Package Description.* https://tools.kali.org/reverse-engineering/dex2jar. Version: 2019, Accessed: 22nd August 2019

[77] Brut.alll: *Apktool Homepage.* https://github.com/iBotPeaches/Apktool. Version: 2019, Accessed: 22nd August 2019

[78] Gruver, B.: *About Smali.* https://github.com/JesusFreke/smali/wiki. Version: 2019, Accessed: 22nd August 2019

[79] Android: *Android Debug Bridge (adb).* https://developer.android.com/studio/command-line/adb. Version: 2019, Accessed: 23rd August 2019

[80] Ruhenstroth, M.: *USB-Debugging aktivieren (Android).* https://mobilsicher.de/schritt-fuer-schritt/usb-debugging-aktivieren. Version: 2018, Accessed: 23rd August 2019

[81] Android: *Android apksigner.* https://developer.android.com/studio/command-line/apksigner. Version: 2019, Accessed: 23rd August 2019

[82] Samsung: *Samsung Odin*. https://samsungodin.com/. Version: 2019, Accessed: 23rd August 2019

[83] Chainfire: *Chainfire Auto Root*. https://autoroot.chainfire.eu/. Version: 2019, Accessed: 23rd August 2019

[84] ChelpuS: *Lucky Patcher Official Website*. https://www.luckypatchers.com/. Version: 2019, Accessed: 23rd August 2019

[85] Jain, A. ; Ross, A. ; Nadakumar, K.: *Introduction to Biometrics*. Sopringer, 2011

[86] Krupp, A. ; Rathgeb, C. ; Busch, C.: Social acceptance of biometric technologies in Germany: A survey. In: *2013 International Conference of the BIOSIG Special Interest Group (BIOSIG)* (2013)

[87] Bhargav-Spantzel, A. et a.: Privacy Preserving Multi-Factor Authentication with Biometrics. In: *CERIAS Tech Report 2007-17* (2007)

[88] Enck, W. ; Ongtang, M. ; McDaniel, P.: Understanding Android Security. In: *IEEE Security & Privacy* (2009)

[89] Van Vleck, T.: *Fernando J ("Corby") Corbato*. https://amturing.acm.org/award_winners/corbato_1009471.cfm. Version: 1990

List of Figures

Listings

List of Tables

Acronyms

2FA Two-Factor Authentication.

ADB Android-Debug-Bridge.

AOSP Android Open Source Project.

API Application Programming Interface.

APK Android Package.

ART Android Application Runtime.

CTAP Client-to-Authenticator Protocol.

CTSS Compatible Time-Sharing System.

DAC Discretionary Access Control.

DNS Domain Name System.

FIDO Fast Identity Online.

GDPR EU General Data Protection Regulation.

GID Group Identifier.

GPG GNU Privacy Guard.

HPI Hasso-Plattner-Institut.

IAM Identity- and Access Management.

IdP identity provider.

IoT Internet of Things.

IT Information Technology.

KBA knowledge-based authentication.

MAC Mandatory Access Control.

MFA Multi-Factor Authentication.

MIT Massachusetts Institute of Technology.

OBA object-based authentication.

OIDC OpenID Connect.

OS Operating System.

OTP One-Time-Passcodes.

PIN Personal Identification Number.

PKI Public Key Infrastructure.

QR Quick Response.

RP relying party.

SaaS Software as a Service.

SQL Structured Query Language.

SSL Secure Sockets Layer.

TEE Trusted Execution Environment.

U2F Universal Second Factor.

UAF Universal Authentication Framework.

UID Unique Identifier.

URL Uniform Resource Locator.

USB Universal Serial Bus.

VM virtual machine.

W3C World Wide Web Consortium.

WebAuthn Web Authentication.

ZKP zero-knowledge proof.

A Listings of Source Code

```
    def authenticate(self, configurationAttributes, requestParameters, step):
134   [...]
      if step == 1:
136     print "Super-Gluu. Authenticate for step 1"
        user_name = credentials.getUsername()
138     if self.oneStep:
          session_device_status = self.getSessionDeviceStatus(session_attributes,
      user_name)
140       if session_device_status == None:
            return False
142       u2f_device_id = session_device_status['device_id']
          validation_result = self.validateSessionDeviceStatus(client_redirect_uri,
      session_device_status)
144       if validation_result:
            print "Super-Gluu. Authenticate for step 1. User successfully
      authenticated with u2f_device '%s'" % u2f_device_id
146       else:
            return False
148       if not session_device_status['one_step']:
            print "Super-Gluu. Authenticate for step 1. u2f_device '%s' is not one
      step device" % u2f_device_id
150         return False
          # There are two steps only in enrollment mode
152       if session_device_status['enroll']:
            return validation_result
154       identity.setWorkingParameter("super_gluu_count_login_steps", 1)
          user_inum = session_device_status['user_inum']
156       u2f_device = deviceRegistrationService.findUserDeviceRegistration(user_inum
      , u2f_device_id, "oxId")
          if u2f_device == None:
158         print "Super-Gluu. Authenticate for step 1. Failed to load u2f_device '%s
      '" % u2f_device_id
            return False
160       logged_in = authenticationService.authenticate(user_name)
          if not logged_in:
162         print "Super-Gluu. Authenticate for step 1. Failed to authenticate user
      '%s'" % user_name
            return False
164       print "Super-Gluu. Authenticate for step 1. User '%s' successfully
      authenticated with u2f_device '%s'" % (user_name, u2f_device_id)
          return True
166     elif self.twoStep:
          authenticated_user = self.processBasicAuthentication(credentials)
168       if authenticated_user == None:
            return False
170       if (self.use_super_gluu_group):
```

```
      print "Super-Gluu. Authenticate for step 1. Checking if user belong to
super_gluu group"
172       is_member_super_gluu_group = self.isUserMemberOfGroup(authenticated_user,
 self.audit_attribute, self.super_gluu_group)
      if (is_member_super_gluu_group):
174         print "Super-Gluu. Authenticate for step 1. User '%s' member of
super_gluu group" % authenticated_user.getUserId()
          super_gluu_count_login_steps = 2
176       else:
          if self.use_audit_group:
178           self.processAuditGroup(authenticated_user, self.audit_attribute, self
.audit_group)
          super_gluu_count_login_steps = 1
180       identity.setWorkingParameter("super_gluu_count_login_steps",
super_gluu_count_login_steps)
      if super_gluu_count_login_steps == 1:
182         return True
    auth_method = 'authenticate'
184     enrollment_mode = ServerUtil.getFirstValue(requestParameters, "loginForm:
registerButton")
    if StringHelper.isNotEmpty(enrollment_mode):
186       auth_method = 'enroll'
    if auth_method == 'authenticate':
188       user_inum = userService.getUserInum(authenticated_user)
      u2f_devices_list = deviceRegistrationService.findUserDeviceRegistrations(
user_inum, client_redirect_uri, "oxId")
190       if u2f_devices_list.size() == 0:
          auth_method = 'enroll'
192         print "Super-Gluu. Authenticate for step 1. There is no U2F '%s' user
devices associated with application '%s'. Changing auth_method to '%s'" % (
user_name, client_redirect_uri, auth_method)
    print "Super-Gluu. Authenticate for step 1. auth_method: '%s'" %
auth_method
194     identity.setWorkingParameter("super_gluu_auth_method", auth_method)
    return True
196   return False
elif step == 2:
198   print "Super-Gluu. Authenticate for step 2"
  user = authenticationService.getAuthenticatedUser()
200   if (user == None):
      print "Super-Gluu. Authenticate for step 2. Failed to determine user name"
202       return False
  user_name = user.getUserId()
204   session_attributes = identity.getSessionId().getSessionAttributes()
  session_device_status = self.getSessionDeviceStatus(session_attributes,
user_name)
206   if session_device_status == None:
      return False
208   u2f_device_id = session_device_status['device_id']
  # There are two steps only in enrollment mode
210   if self.oneStep and session_device_status['enroll']:
      authenticated_user = self.processBasicAuthentication(credentials)
212       if authenticated_user == None:
        return False
214     user_inum = userService.getUserInum(authenticated_user)
      attach_result = deviceRegistrationService.attachUserDeviceRegistration(
```

```
    user_inum, u2f_device_id)
216         print "Super-Gluu. Authenticate for step 2. Result after attaching
    u2f_device '%s' to user '%s': '%s'" % (u2f_device_id, user_name, attach_result)
        return attach_result
218       elif self.twoStep:
        if user_name == None:
220           print "Super-Gluu. Authenticate for step 2. Failed to determine user name
    "
          return False
222       validation_result = self.validateSessionDeviceStatus(client_redirect_uri,
    session_device_status, user_name)
        if validation_result:
224           print "Super-Gluu. Authenticate for step 2. User '%s' successfully
    authenticated with u2f_device '%s'" % (user_name, u2f_device_id)
        else:
226           return False
        super_gluu_request = json.loads(session_device_status['super_gluu_request'
    ])
228       auth_method = super_gluu_request['method']
        if auth_method in ['enroll', 'authenticate']:
230           if validation_result and self.use_audit_group:
            user = authenticationService.getAuthenticatedUser()
232           self.processAuditGroup(user, self.audit_attribute, self.audit_group)
          return validation_result
234       print "Super-Gluu. Authenticate for step 2. U2F auth_method is invalid"
      return False
236   else:
      return False
```

Listing A.1: Excerpt from *super_gluu_authentication_script.py* showing the *authenticate* method using the *authentication_mode* (*one_step/two_step*) Variables

```
,
.method public static isAuthEnabled(Landroid/content/Context;)Z
364   .locals 1
    .line 21
366   invoke-static {p0}, Lorg/gluu/super_gluu/app/settings/Settings;->
      getFingerprintEnabled(Landroid/content/Context;)Ljava/lang/Boolean;
    move-result-object v0
368   invoke-virtual {v0}, Ljava/lang/Boolean;->booleanValue()Z
    move-result v0
370   if-nez v0, :cond_1
    invoke-static {p0}, Lorg/gluu/super_gluu/app/settings/Settings;->
      getPinCodeEnabled(Landroid/content/Context;)Ljava/lang/Boolean;
372   move-result-object v0
    invoke-virtual {v0}, Ljava/lang/Boolean;->booleanValue()Z
374   move-result v0
    if-nez v0, :cond_1
376   invoke-static {p0}, Lorg/gluu/super_gluu/app/settings/Settings;->isAppLocked(
      Landroid/content/Context;)Ljava/lang/Boolean;
    move-result-object p0
378   invoke-virtual {p0}, Ljava/lang/Boolean;->booleanValue()Z
    move-result p0
380   if-eqz p0, :cond_0
    goto :goto_0
```

```
382     :cond_0
        const/4 p0, 0x0
384     goto :goto_1
        :cond_1
386     :goto_0
        const/4 p0, 0x0
388     :goto_1
        return p0
390 .end method
    .method public static isAuthPending(Landroid/content/Context;)Z
392     .locals 3
        const-string v0, "oxPushSettings"
394     const/4 v1, 0x0
        .line 203
396     invoke-virtual {p0, v0, v1}, Landroid/content/Context;->getSharedPreferences(
          Ljava/lang/String;I)Landroid/content/SharedPreferences;
        move-result-object p0
398     const-string v0, "OxRequestData"
        const/4 v2, 0x0
400     .line 204
        invoke-interface {p0, v0, v2}, Landroid/content/SharedPreferences;->getString(
          Ljava/lang/String;Ljava/lang/String;)Ljava/lang/String;
402     move-result-object p0
        if-eqz p0, :cond_0
404     const/4 v1, 0x0
        :cond_0
406     return v1
    .end method
```

Listing A.2: Excerpt from *settings.smali* showing the altered *isAuthEnabled* and *isAuthPending* Methods

B Further Material

B.0.1 Assessment Criteria from *Bonneau, Herley, Oorschot and Stajano* [56]

Usability benefits

U1 *Memorywise-Effortless*: Users of the scheme do not have to remember any secrets at all. We grant a *Quasi-Memorywise-Effortless* if users have to remember one secret for everything (as opposed to one per verifier).

U2 *Scalable-for-Users*: Using the scheme for hundreds of accounts does not increase the burden on the user. As the mnemonic suggests, we mean "scalable" only from the user's perspective, looking at the cognitive load, not from a system deployment perspective, looking at allocation of technical resources.

U3 *Nothing-to-Carry*: Users do not need to carry an additional physical object (electronic device, mechanical key, piece of paper) to use the scheme. Quasi-Nothing-to-Carry is awarded if the object is one that they'd carry everywhere all the time anyway, such as their mobile phone, but not if it's their computer (including tablets).

U4 *Physically-Effortless*: The authentication process does not require physical (as opposed to cognitive) user effort beyond, say, pressing a button. Schemes that don't offer this benefit include those that require typing, scribbling or performing a set of motions. We grant Quasi-Physically- Effortless if the user's effort is limited to speaking, on the basis that even illiterate people find that natural to do.

U5 *Easy-to-Learn* : Users who don't know the scheme can figure it out and learn it without too much trouble, and then easily recall how to use it.

U6 *Efficient-to-Use* : The time the user must spend for each authentication is acceptably short. The time required for setting up a new association with a verifier, although possibly longer than that for authentication, is also reasonable.

U7 *Infrequent-Errors* : The task that users must per- form to log in usually succeeds when performed by a legitimate and honest user. In other words, the scheme isn't so hard to use or unreliable that genuine users are routinely rejected.

U8 *Easy-Recovery-from-Loss* : A user can conveniently regain the ability to authenticate if the token is lost or the credentials forgotten. This combines usability aspects such as: low latency before restored ability; low user inconvenience in recovery (e.g., no requirement for physically standing in line); and assurance that recovery will be possible, for example via built-in backups or secondary recovery schemes. If recovery requires some form of re-enrollment, this benefit rates its convenience.

Deployability Benefits

D1 *Accessible*: Users who can use passwords are not prevented from using the scheme by disabilities or other physical (not cognitive) conditions.

D2 *Negligible-Cost-per-User*: The total cost per user of the scheme, adding up the costs at both the prover's end (any devices required) and the verifier's end (any share of the equipment and software required), is negligible. The scheme is plausible for startups with no per-user revenue.

D3 *Server-Compatible*: At the verifier's end, the scheme is compatible with text-based passwords. Providers don't have to change their existing authentication setup to support the scheme.

D4 *Browser-Compatible*: Users don't have to change their client to support the scheme and can expect the scheme to work when using other machines with an up-to-date, standards-compliant webbrowser and no additional software. In 2012, this would mean an HTML5-compliant browser with JavaScript enabled. Schemes fail to provide this benefit if they require the installation of plugins or any kind of software whose installation requires administrative rights. Schemes offer Quasi browser-Compatible if they rely on non-standard but very common plugins, e.g., Flash.

D5 *Mature*: The scheme has been implemented and deployed on a large scale for actual authentication purposes beyond research. Indicators to consider for granting the full benefit may also include whether the scheme has undergone user testing, whether the standards community has published related documents, whether open-source projects implementing the scheme exist, whether anyone otherthan the implementers has adopted the scheme, the amount of literature on the scheme and so forth.

D6 *Non-Proprietary*: Anyone can implement or use the scheme for any purpose without having to pay royalties to anyone else. The relevant techniques are generally known, published openly and not protected by patents or trade secrets.

Security Benefits

S1 *Resilient-to-Physical-Observation*: An attacker cannot impersonate a user after observing them authenticate one or more times. We grant Quasi-Resilient-to-Physical-Observation if the scheme could be broken only by repeating the observation more than, say, 10–20 times. Attacks include shoulder surfing, filming the keyboard, recording keystroke sounds, or thermal imaging of keypad.

S2 *Resilient-to-Targeted-Impersonation*: It is not possible for an acquaintance (or skilled investigator) to impersonate a specific user by exploiting knowledge of personal details (birth date, names of relatives etc.). Personal knowledge questions are the canonical scheme that fails on this point.

S3 *Resilient-to-Throttled-Guessing*: An attacker whose rate of guessing is constrained by the verifier cannot successfully guess the secrets of a significant fraction of users. The verifier-imposed constraint might be enforced by an online server, a tamper-resistant chip or any other mechanism capable of throttling repeated requests. To give a quantitative example, we might grant this benefit if an attacker constrained to, say, 10 guesses per account per day, could compromise at most 1% of accounts in a year. Lack of this benefit is meant to penalize schemes in which it is frequent for user-chosen secrets to be selected from a small and well-known subset (low min-entropy [14]).

S4 *Resilient-to-Unthrottled-Guessing*: An attacker whose rate of guessing is constrained only by available computing resources cannot successfully guess the secrets of a significant fraction of users. We might for example grant this benefit if an attacker capable of attempting up to 240 or even 264 guesses per account could still only reach fewer than 1% of accounts. Lack of this benefit is meant to penalize schemes where the space of credentials is not large enough to withstand brute force search (including dictionary attacks, rainbow tables and related brute force methods smarter than raw exhaustive search, if credentials are user-chosen secrets).

S5 *Resilient-to-Internal-Observation*: An attacker cannot impersonate a user by intercepting the user's input from inside the user's device (e.g. by key-logging malware) or eavesdropping on the clear-text communication between prover and verifier (we assume that the attacker can also defeat TLS if it is used, perhaps through the CA). As with Resilient-to-Physical-Observation above,we grant Quasi-Resilient-to-Internal-Observation if the scheme could be broken only by intercepting input or eavesdropping cleartext more than, say, 10–20 times. This penalizes schemes that are not replay-resistant, whether because they send a static response or because their dynamic response countermeasure can be cracked with a few observations. This benefit assumes that general-purpose devices like software-updatable personal computers and mobile phones may contain malware, but that hardware devices dedicated exclusively to the scheme can be made malware-free. We grant Quasi-Resilient-to-Internal-Observation to two-factor schemes where both factors must be malware-infected for the attack to work. If infecting only one factor breaks the scheme, we don't grant the benefit.

S6 *Resilient-to-Leaks-from-Other-Verifiers*: Nothing that a verifier could possibly leak can help an attacker impersonate the user to another verifier. This penalizes schemes where insider fraud at one provider, or a successful attack on one back-end, endangers the user's accounts at other sites.

S7 *Resilient-to-Phishing*: An attacker who simulates a valid verifier (including by DNS manipulation)cannot collect credentials that can later be used to impersonate the user to the actual verifier. This penalizes schemes allowing phishers to get victims to authenticate to lookalike sites and later use the harvested credentials against the genuine sites.It is not meant to penalize schemes vulnerable to more sophisticated real-time man-in-the-middle or relay attacks, in which the attackers have one connection to the victim prover (pretending to be the verifier) and simultaneously another connection to the victim verifier (pretending to be the prover).

S8 *Resilient-to-Theft*: If the scheme uses a physical object for authentication, the object cannot be used for authentication by another person who gains possession of it. We still grant Quasi-Resilient-to-Theft if the protection is achieved with the modest strength of a PIN, even if attempts are not rate-controlled, because the attack doesn't easily scale to many victims.

S9 *No-Trusted-Third-Party*: The scheme does not rely on a trusted third party (other than the prover and the verifier) who could, upon being attacked or otherwise becoming untrustworthy, compromise the prover's security or privacy.

S10 *Requiring-Explicit-Consent*: The authentication process cannot be started without the explicit consent of the user. This is both a security and a privacy feature (a rogue wireless RFID-based credit card reader embedded in a sofa might charge a card without user knowledge or consent).

S11 *Unlinkable*: Colluding verifiers cannot determine, from the authenticator alone, whether the same user is authenticating to both. This is a privacy feature. To rate this benefit we disregard linkability introduced by other mechanisms (same user ID, same IP address, etc).

C Further Figures

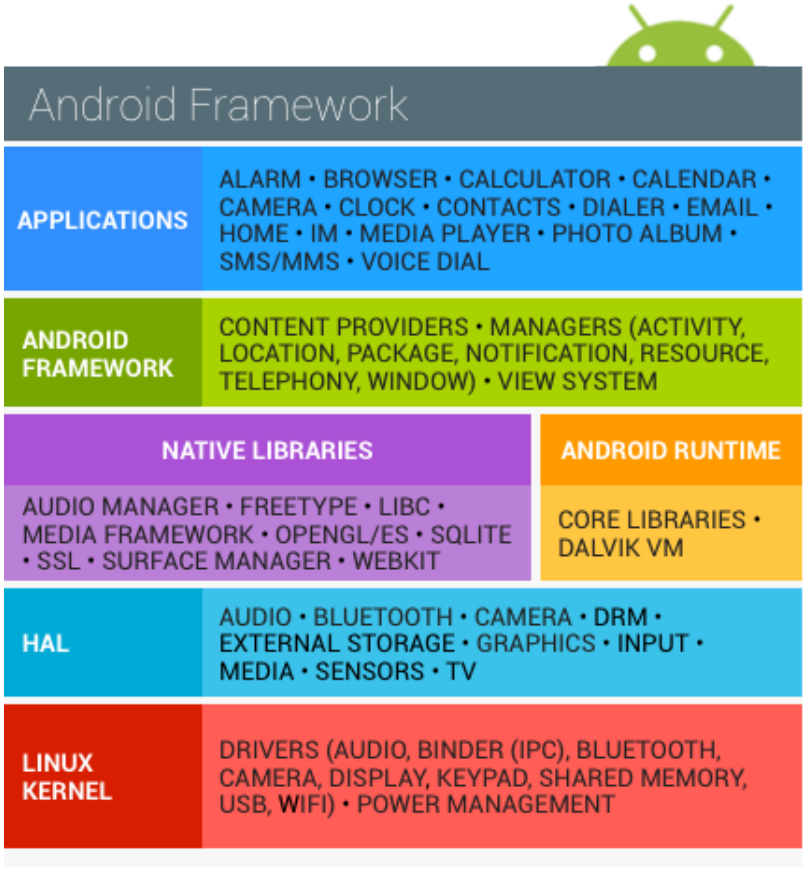


Figure C.1: Android Software Stack [30]

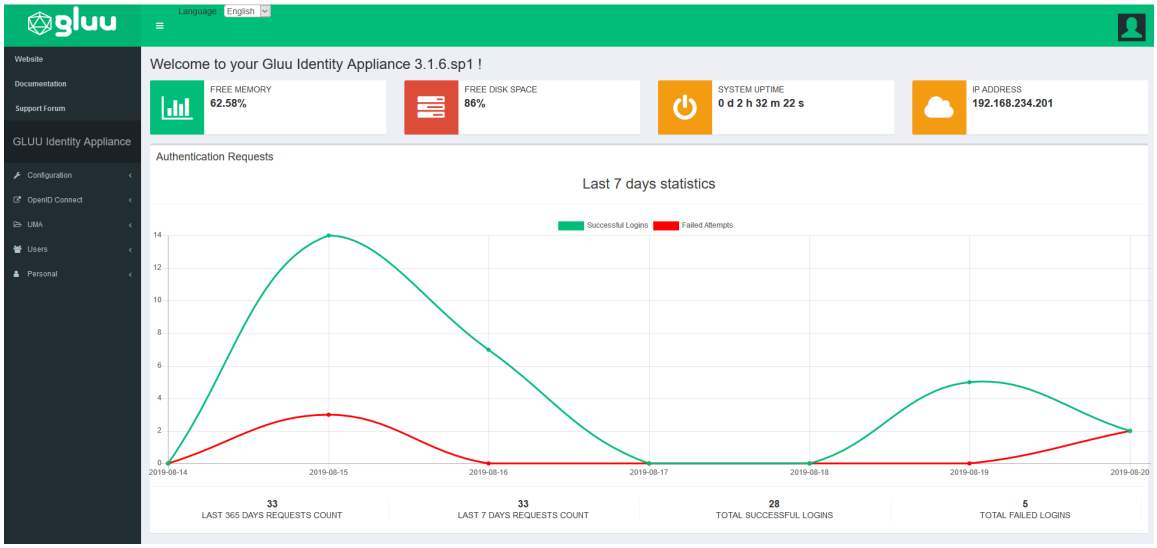


Figure C.2: Gluu Administrator Interface

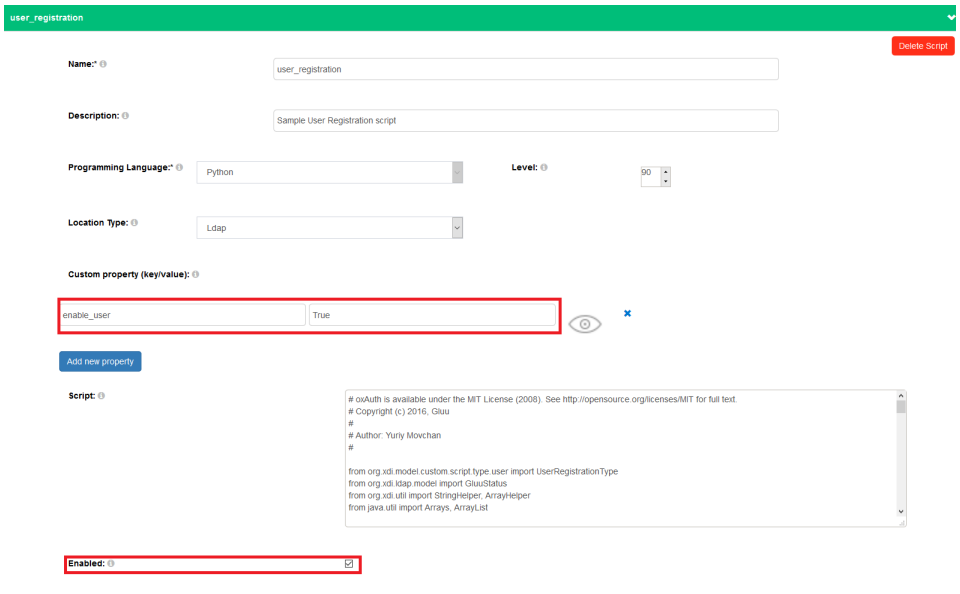


Figure C.3: Gluu Custom Script for User Registration Enabled

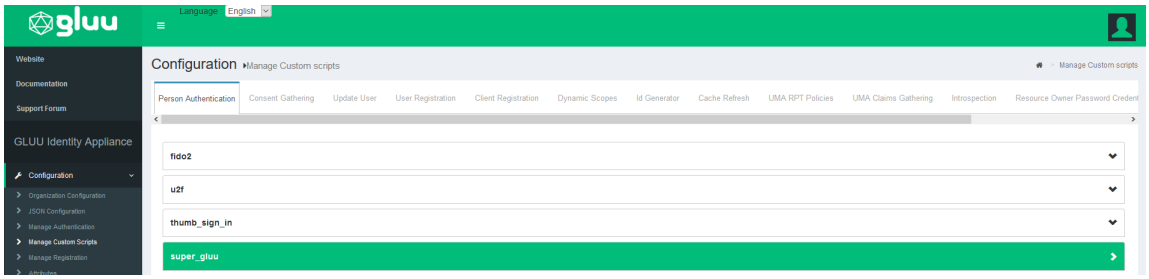


Figure C.4: Enable Super Gluu Authentication Script

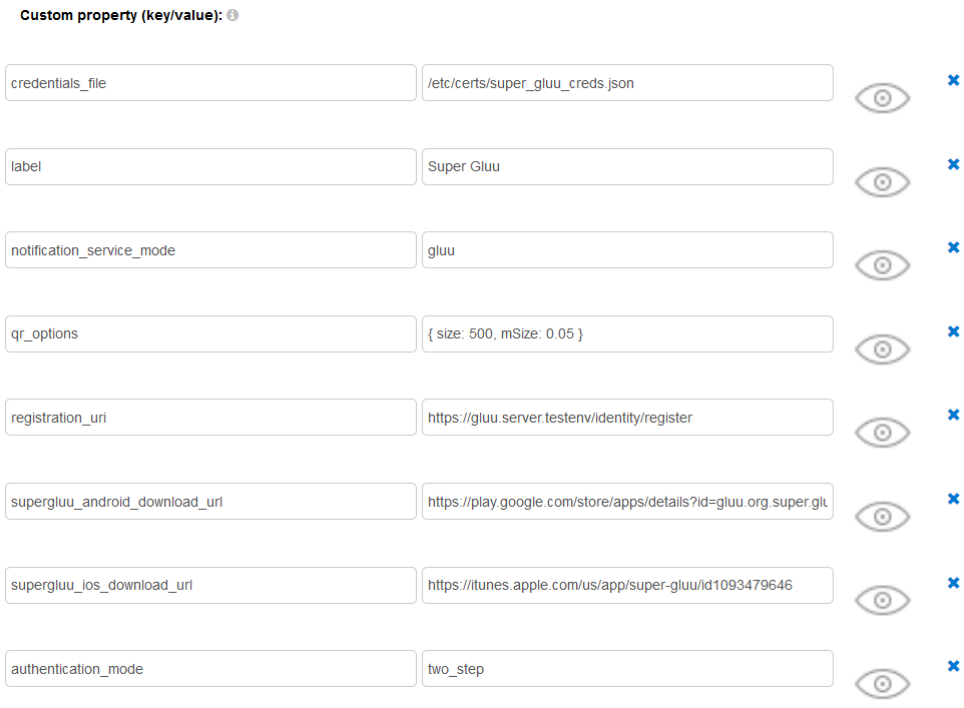


Figure C.5: Super Gluu Authentication Script Options

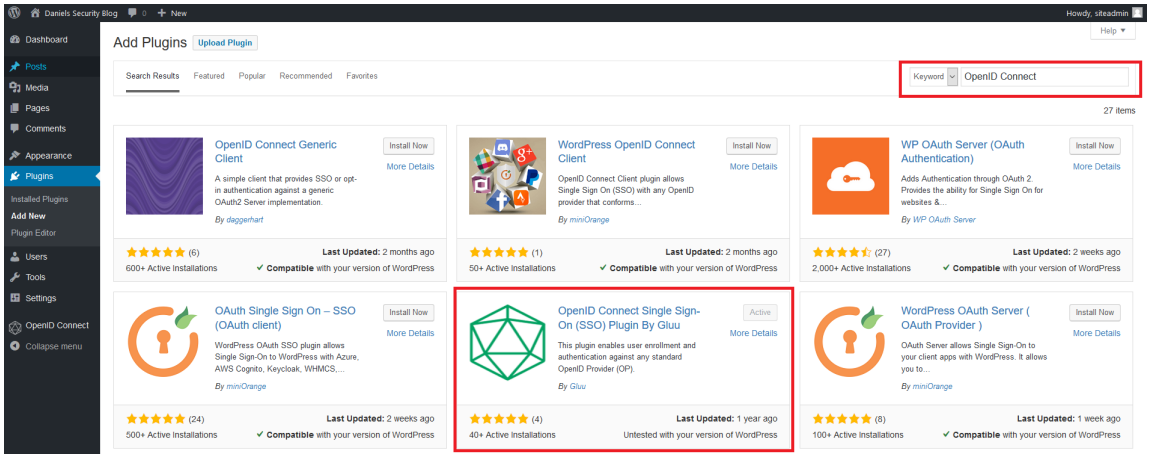


Figure C.6: Wordpress Plugin Search Results for *OpenID Connect* featuring a Gluu Plugin

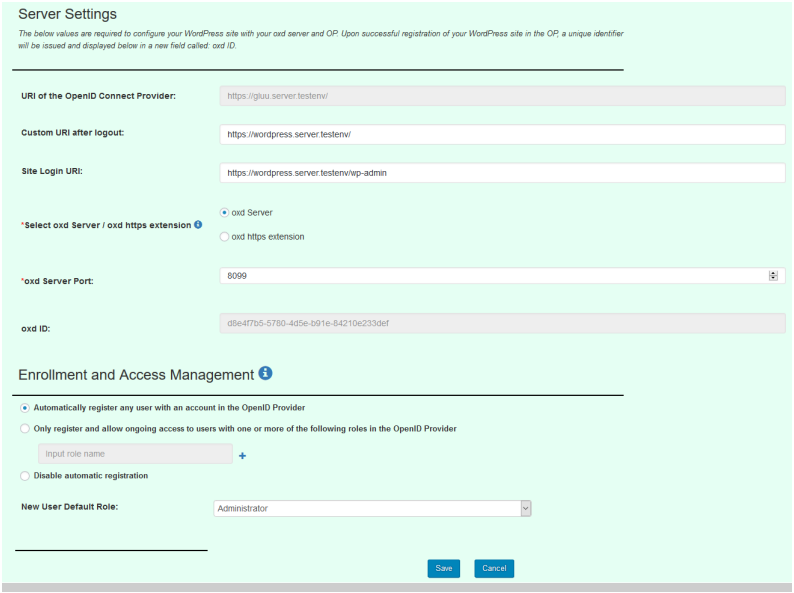


Figure C.7: Wordpress OpenID Connect Plugin Configuration

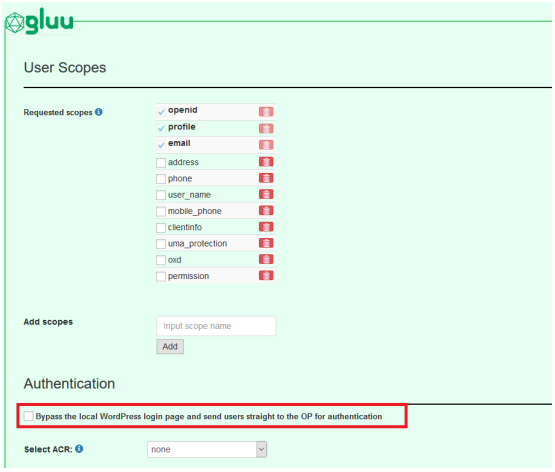


Figure C.8: Wordpress OpenID Connect Plugin Advanced Settings

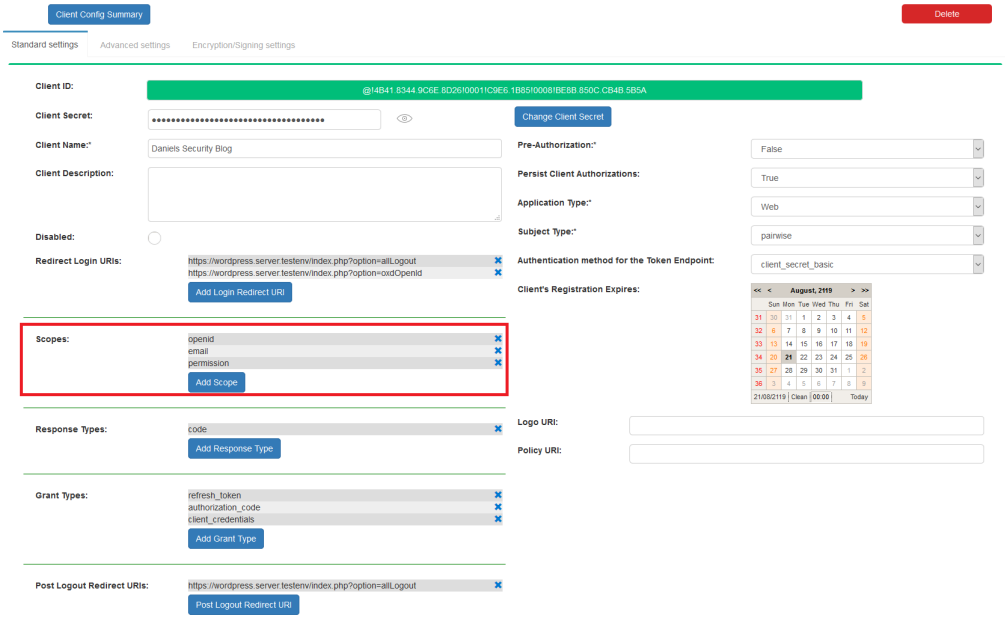


Figure C.9: Gluu Server OpenID Wordpress Client Details

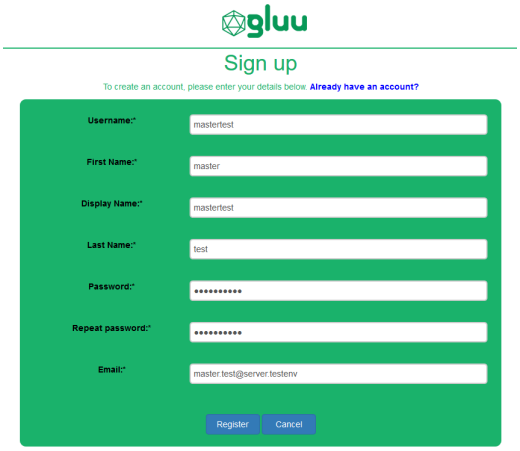


Figure C.10: Gluu Server *Mastertest* User Registration

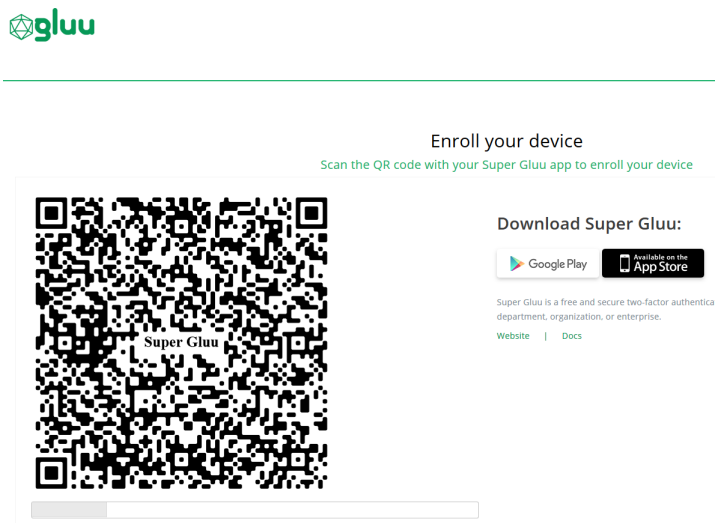


Figure C.11: Gluu Server First-Time Device Registration

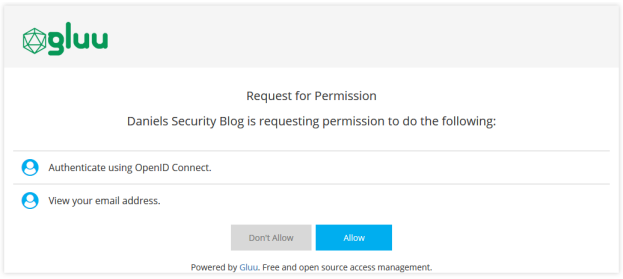


Figure C.12: Gluu Server First-Time Permission Approval

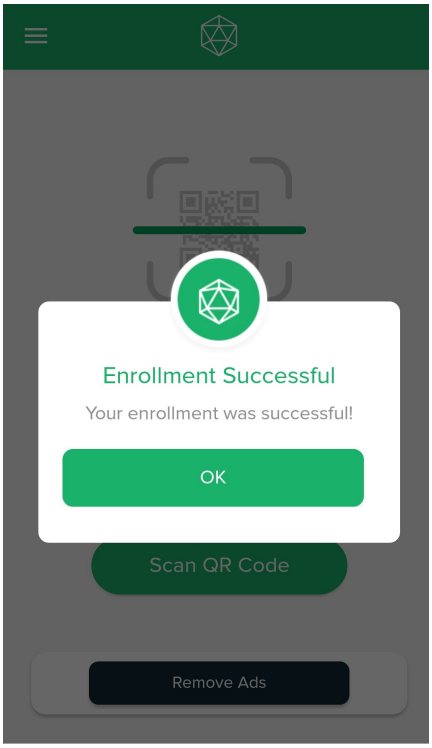


Figure C.13: Android Super Gluu Enrollment Successful Confirmation

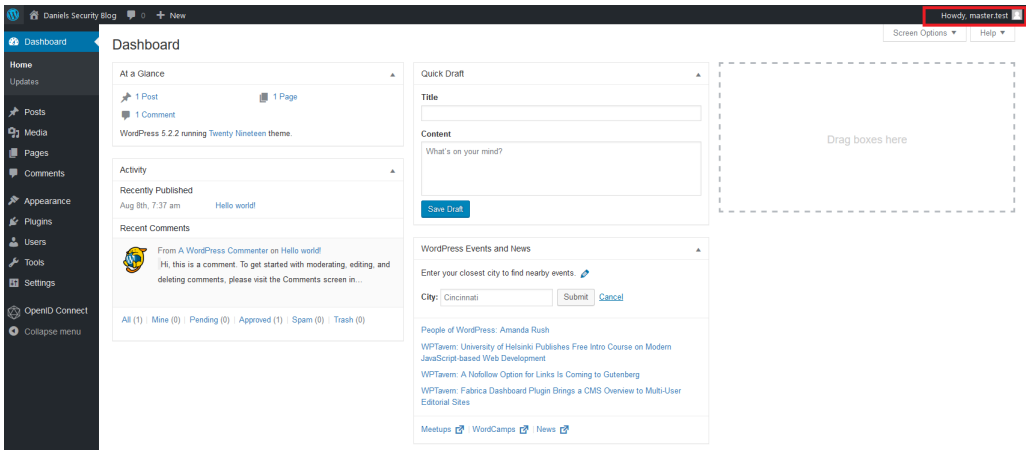


Figure C.14: Wordpress User *master.test* Authenticated

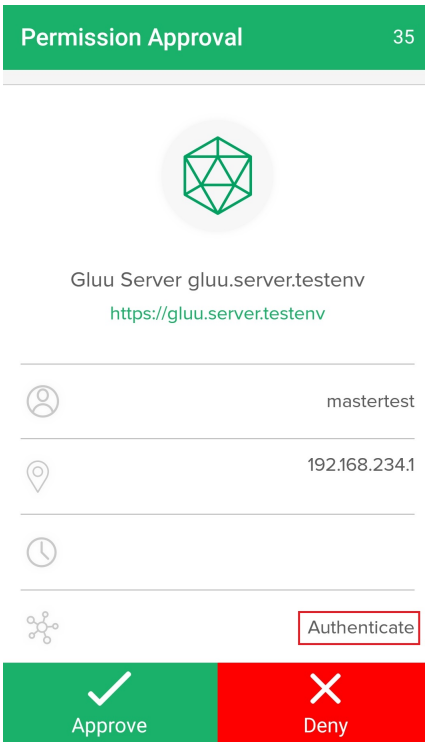


Figure C.15: Android Super Gluu Authentication Approval

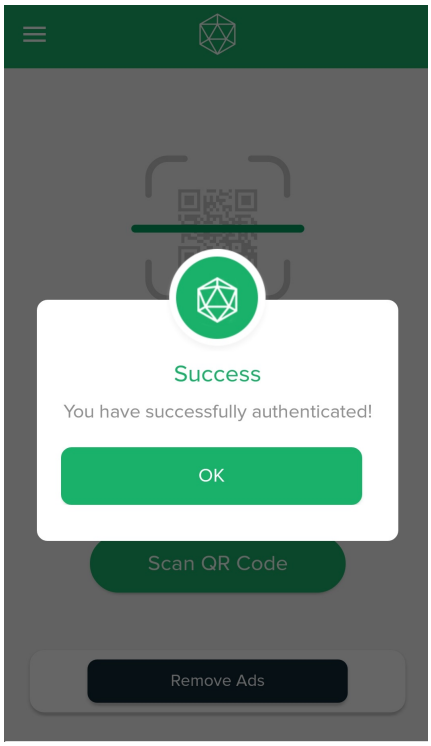


Figure C.16: Android Super Gluu Authentication Confirmation

Declaration of Academic Integrity

Hereby, I declare that I have composed the presented paper independently on my own and without any other resources than the ones indicated. All thoughts taken directly or indirectly from external sources are properly denoted as such.

This paper has neither been previously submitted to another authority nor has it been published yet

Place, Date Signature

Theses

Master's Thesis
Conception and Assessment of a System for Usage of Biometric Sensors in Smartphones to Authenticate against Remote Servers

Handed in on: August 30, 2019

by: Daniel Köhler
born on 12.05.1996
in Berlin

Reviewer: Prof. Dr.-Ing. Matthias Kreuseler
Secondary Reviewer: Dr. Feng Cheng

1. Systems for remote biometric authentication are becoming more and more popular.
2. To be able to assess different systems, formalized criteria are needed.
3. Frameworks for remote biometric authentication are becoming more popular.
4. The FIDO2 framework proposes a shift of the trust in a biometric authenticaiton to the remote devices in multi-factor authentication.
5. Mobile devices that can physically be accessed will always be prone to manipulation.
6. A password-less remote biometric authentication system using solely a mobile device such as a smartphone as an authenticator can therefore not be considered secure.
7. The two-factor thought of owning the smartphone plus owning the fingerprint turns out invalid if the fingerprint sensor can be manipulated.
8. Relaying trust towards in biometric authentication towards smartphones is hence not considered a secure approach.
9. Security in biometric authentication is needed to be implemented and enforced on a server rather than on a roaming device.