Mitigating Real-Time Phishing in Time-Based One-Time Password Applications Using Behavioral Analysis

JULIANA LOKO KIVUVA

166875

Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science in Information Security at Strathmore University

School of Computing & Engineering Sciences

Strathmore University

Nairobi, Kenya

June 2025

Declaration and Approval

Declaration

I declare that this work has not been previously submitted and approved for the award of a degree by this or any other University. To the best of my knowledge and belief, the dissertation contains no material previously published or written by another person except where due reference is made in the dissertation itself.

Student's Name: **Juliana Kivuva Loko**

Signature ..... Date……22/05/2025……

Approval

The dissertation of **Juliana Kivuva Loko** was reviewed and approved for examination by the following:

Dr. Vitalis Ozianyi,
Lecturer, School of Computing & Engineering Sciences,
Strathmore University

Dr. Julius Butime
Dean, School of Computing & Engineering Sciences,
Strathmore University

Prof. Bernard Shibwabo,
Director of Graduate Studies,
Strathmore University

Abstract

Time-based One-Time Password (TOTP) applications enhance online security by providing an additional authentication layer. However, they are vulnerable to real-time phishing attacks, where attackers deceive users into entering their TOTP codes on fraudulent websites. Since TOTP codes are valid for a short duration and cannot be reused, traditional security mechanisms struggle to detect and prevent their misuse in real-time. Attackers can intercept these codes and immediately use them to gain unauthorized access before they expire, bypassing standard authentication defenses.

This dissertation presents a behavioral analysis approach to mitigate real-time phishing attacks on TOTP systems. An algorithm was designed to detect suspicious activity by analyzing user behavior patterns, such as login frequency, location, device type, and interaction anomalies. The algorithm flags potential phishing attempts in real-time by establishing a baseline for normal usage and identifying deviations. A proof-of-concept prototype was developed using a data-driven prototyping methodology to validate the effectiveness of this approach. The results confirm that integrating behavioral analysis into TOTP applications provides proactive security by detecting and responding to phishing threats before authentication codes are exploited.

Keywords: Time-based One-Time Passwords, Behavioral Analysis, Real-Time Phishing, Authentication

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AiTM – Adversary in the middle attack

API - Application Programming Interface

EV – Extended validation

FIDO – Fast Identity Online

GORM- Go Object-Relational Mapping

IAM – Identity Access Management

IEEE – Institute of Electrical and Electronics Engineers

IPC – Inter process communications

MFA – Multi-Factor Authentication

MITM – Man-in-the-Middle Attack

MVI- Model View Intent

OTP – One-Time Password

ORM – Object relational mapping

REST - REpresentational State Transfer

RTP – Real-Time Phishing

RFC- Request for Comments

SDLC – Software Development Life Cycle

SSL/TLS – Secure Sockets Layer / Transport Layer Security

TEE - Trusted Execution Environment

TOTP – Time-Based One-Time Password

Definition of Terms

Concept	Description
Automated Phishing Tools	Tools that enable attackers to set up phishing websites that act as man-in-the-middle proxies, intercepting and relaying credentials in real time for large-scale attacks (Sun et al., 2021).
Behavior-based Analysis	Real-time monitoring of user behavior, including login frequency, device information, IP address changes, and geographic anomalies, to detect suspicious authentication attempts (Mihailescu et al., 2023).
Homoglyph Attack	A phishing technique using visually similar characters (e.g., replacing "o" with the Cyrillic "o") to deceive users into visiting fraudulent websites (Islam et al., 2023).
Man-in-the-Middle Attack	An attack where an attacker intercepts and manipulates communication between two parties (Nguyen et al., 2021).
Multi-Factor Authentication	An authentication framework requiring multiple independent credentials for user verification (Aparicio et al., 2023).
One-Time Password	A temporary passcode valid for a single authentication session. Each passcode expires after use or within a predefined time window (Almeida et al., 2024).
Real-time Phishing	An attack targeting TOTP-based authentication by tricking users into entering OTP codes and credentials on fraudulent websites resembling legitimate services (Sun et al., 2021).

SSL and TLS	Encryption protocols that secure data transmission between a user's device and a web server, ensuring confidentiality (Sun et al., 2021).
Time-based One-Time Passwords	A widely used form of two-factor authentication that generates temporary, time-sensitive passcodes, improving security by reducing risks associated with password-based authentication (Bianchi & Valeriani, 2023).
Trusted Execution Environments	Hardware-backed security mechanisms that protect cryptographic secrets (such as TOTP keys) from unauthorized access by isolating sensitive computations (Plata & Calpito, 2020).
Two-Factor Authentication	A security mechanism requiring two separate authentication factors to verify identity. Typically, a password and a TOTP code are combined to prevent unauthorized access (Shirvanian & Agrawal, 2021).
Typosquatting	A phishing technique where attackers register domain names similar to legitimate websites, e.g., "g00gle.com" instead of "google.com," to trick users into entering sensitive information (Fejrskov et al., 2022).

Acknowledgement

It is my wish to give thanks to God for having given me the strength and guidance in this dissertation right from the beginning. Special thanks to my supervisor Dr. Vitalis Ozianyi, together with Dr. Joseph Sevilla and Strathmore University staff, especially those at @iLab Africa and Faculty of Information Technology for the great support I got from them.

Chapter 1: Introduction

1.1 Background Information

Time-based One-Time Passwords (TOTP) are a widely used form of two-factor authentication (2FA). 2FA is the most common form of multi-factor authentication (MFA). TOTP is implemented in popular authenticator apps such as Google Authenticator, Microsoft Authenticator, and Authy (Berrios et.al, 2023). As explained by Bianchi and Valeriani (2023), the TOTP framework utilizes symmetric key cryptography, standardized in RFC 6238. Both the server and the user's device are configured with a shared secret key, which, along with the device's timestamp, is used to generate and validate a one-time password (OTP) for authentication. These authenticator apps link to a user's device and leverage its internal clock to generate time-based codes that remain valid for a brief period, typically 30 seconds (Bianchi & Valeriani, 2023).

TOTP is based on two distinct factors. Something the user knows such as a password or PIN which acts as the first factor, and something the user has which serves as the second factor. In this case the second factor is the time sensitive OTP code (Pal, 2019). Despite being a critical layer of security, TOTP is vulnerable to real-time phishing (RTP) attacks. In RTP attacks, users are deceived into entering their two-factor authentication codes and login credentials on fake login pages. Sun et al. (2021) point out that, historically, attackers would manually create phishing websites to imitate legitimate ones, which was a time-consuming process. However, with the advent of automated RTP tools like Evilginx, HiddenEye, Wifiphisher, SocialFish, BlackEye, Shellphish, and Zphisher, these attacks have become much easier to execute and require minimal effort (Dam et.al, 2024). According to Dam et.al (2024), attackers can quickly install and configure these tools to replicate legitimate website URLs and act as man-in-the-middle proxies. Once a malicious website is set up, attackers distribute its URL to potential victims through phishing emails or SMS messages. These messages often create a sense of urgency, prompting users to act quickly without verifying the legitimacy of the information. To further deceive users, attackers employ techniques such as homoglyph URLs, where look-alike characters are substituted in the domain name, making the fake site appear legitimate. Some countermeasures proposed to combat RTP attacks on 2FA systems include employing SSL/TLS encryption to safeguard data transmission and implementing device-based challenges that require physical user interaction on a registered device during TOTP authentication. These measures aim

to verify the legitimacy of the individual entering the authentication code and protect against unauthorized access (Sun et al., 2021). While these solutions offer a degree of protection, they have notable limitations. SSL/TLS encryption ensures secure communication channels but does not address the core phishing issue, where users are deceived into entering their OTP on fraudulent websites. Similarly, device-based challenges can verify physical interaction but may not fully prevent attackers who manipulate users into authorizing malicious actions through social engineering tactics (Gilsenan et.al, 2024). These gaps highlight the need for more advanced, multi-layered security solutions that can adapt to evolving threats and address both technical vulnerabilities and human susceptibility to deception.

1.2 Problem statement

TOTP-based authentication relies on a shared secret key between the server and the user's device, which generates time-sensitive codes synchronized with the system's internal clock. These codes, typically valid for 30 seconds, are used to verify user identity during login attempts (Bianchi & Valeriani, 2023). However, despite providing an additional security layer, TOTP authentication remains susceptible to RTP attacks.

In RTP attacks, adversaries use phishing websites that closely resemble legitimate login portals, tricking users into entering their credentials and OTP codes. Once captured, attackers can immediately use these codes to access accounts before they expire, bypassing the security benefits of TOTP (Sun et al., 2021). The automation of phishing toolkits such as Evilginx, SocialFish, and BlackEye has significantly lowered the technical barrier for attackers, allowing them to execute man-in-the-middle (MitM) phishing attacks more efficiently (Sun et al., 2021). These attacks exploit human factors such as trust, urgency, and deception making them difficult to detect using conventional security measures.

Current countermeasures, including SSL/TLS encryption and device-based challenges, help secure communication channels and verify user presence but do not prevent attackers from deceiving users into providing their OTPs (Sun et al., 2021). Some researchers have proposed phishing-resistant authentication mechanisms, such as FIDO2/WebAuthn, which eliminate the need for shared secrets and OTP codes altogether. However, many organizations continue to rely on TOTP due to its ease of implementation and compatibility with existing authentication

infrastructures.

To mitigate RTP attacks on TOTP-based authentication, this dissertation explores a behavioral analysis-driven approach. By analyzing real-time user interactions such as login patterns, device usage, and location consistency deviations from normal behavior can be detected, potentially signaling an ongoing phishing attack. This approach aims to enhance phishing detection without requiring users to adopt entirely new authentication models, bridging the gap between security effectiveness and user convenience.

1.3 General Objective

To design and implement a user-based behavioral analysis solution that significantly reduces the risk of successful real-time phishing attacks on TOTP.

1.4 Research Objectives

i. To investigate how real-time phishing attacks exploit vulnerabilities in TOTP systems.

ii. To analyze the security weaknesses of TOTP applications that make them susceptible to real-time phishing attacks.

iii. To evaluate existing countermeasures and proposed mitigation techniques for defending against real-time phishing attacks on TOTP applications.

iv. To design, develop, and test a behavioral-based prototype to detect real-time phishing attempts.

v. To validate the effectiveness of the developed solution through testing and analysis.

1.5 Research Questions

i. How do real-time phishing attacks exploit vulnerabilities in TOTP applications?

ii. What security weaknesses make TOTP applications susceptible to real-time phishing attacks?

iii. What existing countermeasures and proposed mitigation techniques are used to defend against real-time phishing attacks on TOTP applications?

iv. How can behavioral analysis be leveraged to detect real-time phishing attacks in TOTP applications?

v. How effective is the developed prototype in detecting and mitigating real-time phishing attempts?

1.6 Scope and Limitations

1.6.1 Scope

This dissertation investigates the susceptibility of TOTP applications to RTP attacks. The study aims to evaluate the effectiveness of behavioral analysis as a potential mitigation strategy for such attacks. It examines the vulnerabilities inherent in TOTP authentication that make it susceptible to RTP attacks, exploring the mechanisms by which attackers can bypass or undermine the security provided by TOTP. Furthermore, the research delves into the limitations of current countermeasures and existing real-time phishing techniques, shedding light on the gaps in protection that allow these attacks to succeed. The study includes the design and development of a proof-of-concept behavioral analysis solution intended to detect RTP attempts. This solution is built to assess various user behavior characteristics, such as login frequency, location consistency, device type, time of access, and real-time anomaly detection, to identify unusual activity indicative of phishing. By analyzing these behavioral patterns, the proposed model aims to provide an additional layer of defense against RTP attacks, complementing traditional security measures and enhancing the resilience of TOTP authentication systems.

The scope of the dissertation is intentionally focused on TOTP-based authentication systems, excluding other multifactor authentication methods or SMS-based OTPs. Additionally, while the behavioral analysis model is developed and tested, it will be evaluated in a controlled environment rather than in live, real-world phishing scenarios. This controlled setup ensures that the analysis of the model's effectiveness is conducted under known conditions, allowing for a detailed and reliable assessment of its potential in detecting and mitigating RTP attacks. Through this focused investigation, the dissertation seeks to contribute valuable insights into how behavioral analysis can enhance the security of TOTP applications against emerging phishing threats.

1.6.2 Limitations

The dissertation adopts a proof-of-concept approach, which inherently introduces several constraints that must be acknowledged when interpreting the findings. One of the primary limitations is the use of controlled simulations. While the study models phishing attack scenarios in a controlled environment, it is important to recognize that such simulations may not fully capture the unpredictability and sophistication of real-world phishing tactics. The dynamic nature

of real-world attacks, with their ability to evolve rapidly, poses a challenge to accurately replicating all potential attack vectors within a simulated setting. Another constraint involves the variability in user behavior. The research assumes that deviations from normal user behavior can indicate phishing attempts, but human behavior is inherently complex and highly individualistic. As a result, this approach carries the risk of generating false positives or false negatives. Users may exhibit legitimate behavior that appears anomalous, or attackers may employ tactics that mimic typical behavior patterns, making it difficult to differentiate between genuine and malicious activities with complete accuracy. Furthermore, the scope of the threat landscape poses a limitation. The cybersecurity environment evolves rapidly, with new threats and attack techniques emerging regularly. While the dissertation focuses on currently known RTP techniques, it may not encompass all future attack variations. As the landscape shifts, new methods of bypassing authentication systems or exploiting vulnerabilities may arise, which the proposed solution might not address at the time of testing. Lastly, the technical constraints of the prototype further limit the study's applicability. The behavioral analysis solution is developed and tested specifically for widely used TOTP applications like Google Authenticator and Microsoft Authenticator. This means that the research does not account for custom enterprise multifactor authentication solutions, which may have different configurations, security protocols, and user behavior patterns. As a result, the effectiveness of the behavioral analysis model in enterprise environments or with custom implementations remains untested and beyond the scope of this dissertation.

1.7 Research Justification

Research has demonstrated that TOTP-based authentication remains vulnerable to RTP attacks, particularly those facilitated by adversary-in-the-middle (AiTM) techniques (Sun et al., 2021). Attackers leverage phishing toolkits such as Evilginx to intercept OTP codes in real time, allowing them to bypass authentication mechanisms (Bianchi & Valeriani, 2023). This study addresses a critical security gap in TOTP authentication by proposing a behavioral analysis-driven approach to detect and mitigate such attacks. Existing countermeasures, such as TLS encryption and phishing awareness training, have limitations in preventing RTP attacks because they do not directly address the deception techniques used by attackers. Additionally, while phishing-resistant authentication standards like FIDO2/WebAuthn are gaining traction, many organizations still rely on TOTP due to its widespread adoption and ease of integration. Therefore, improving the security of TOTP applications remains an essential area of research.

This dissertation contributes to cybersecurity research by developing an adaptive security mechanism that enhances phishing detection through user behavioral analysis. By analyzing login patterns, device usage, location data, and real-time anomalies, the proposed solution can help identify and mitigate phishing attempts before they succeed. The findings from this research can inform future authentication strategies, making TOTP-based authentication more resilient against evolving cyber threats.

Chapter 2: Literature Review

2.1 Introduction

This chapter provides a comprehensive review of existing research and studies on real-time phishing attacks targeting TOTP applications. The purpose of this chapter is to establish a theoretical foundation by exploring previous works on phishing attacks, TOTP-based security systems, and the vulnerabilities that these systems face. It will examine key concepts, methodologies, and countermeasures that have been proposed to address phishing threats, highlighting both the effectiveness and limitations of these approaches. Furthermore, this chapter will explore behavioral analysis and how it can be leveraged to detect and mitigate phishing attacks in real-time.

2.2 One Time Passwords

Almeida et al. (2024) describe One Time Passwords (OTP) as dynamic, temporary passcodes used to authenticate users. OTPs adds an extra layer of security beyond traditional static passwords. OTPs exist in several forms, each with its strengths and vulnerabilities. Among them, TOTPs are one of the most widely adopted and secure types of OTPs.

2.1.1 SMS-Based OTP

SMS-based OTPs are among the most widely used forms of OTPs, especially for 2FA in services such as banking, email, and various online platforms. According to Aparicio et al. (2023), the primary advantage of SMS-based OTPs lies in their simplicity and accessibility. Users do not need to install any special apps or carry additional devices. They just receive a code via text message, which they input to authenticate their login. This method is familiar to most users, making it easy to understand and implement (Pal, 2019). Additionally, it can be used on nearly any mobile phone, which adds to its convenience. However, the security of SMS-based OTPs is undermined by several significant vulnerabilities. One of the most significant risks to SMS-based OTPs is sim swapping. Sim Swapping is a type of social engineering attack where a malicious actor convinces a mobile carrier to transfer the victim's phone number to a new SIM card in the attacker’s possession (Aparicio et al. 2023). Once the transfer is complete, the attacker gains control of any SMS messages, including OTPs, sent to the victim’s number. With this access, the attacker can bypass 2FA on various accounts, logging in as the victim and potentially

compromising sensitive information (Aparicio et al., 2024). In addition to SIM swapping, SMS-based OTPs are also vulnerable to MitM attacks. In a MitM attack, an attacker intercepts the communication between the user's phone and the server to capture the OTP. (Salahdine et al., 2021). This can occur through several methods, such as exploiting compromised networks, deploying malware on the victim's device, or leveraging other interception techniques. By capturing the OTP in transit, the attacker can gain unauthorized access to the user's accounts, posing a significant security risk. These risks are especially concerning for sensitive accounts where an additional layer of security is critical. Due to these vulnerabilities, SMS-based OTPs are considered less secure compared to other forms of multi-factor authentication.

2.1.2 Email-Based OTP

Pal (2019) highlights that email-based OTPs are another commonly used method for 2FA, particularly in the banking sector. In this method, an OTP is sent directly to the user's email inbox. The user retrieves the OTP from the email and enters it into the authentication system to complete the login process. Similar to SMS-based OTPs, email-based OTPs offer the convenience of not requiring additional apps, devices, or specialized hardware, making them accessible and easy to use for a broad range of users. A key advantage of email-based OTPs over SMS-based OTPs is their immunity to SIM-swapping attacks, a prominent vulnerability in SMS-based methods (Aparicio et al. 2023). Since the OTP is sent to an email address, control over the user's phone number is irrelevant, reducing the risk of SIM swapping. However, despite this convenience and accessibility, email-based OTPs still carry notable vulnerabilities, particularly if an attacker gains access to the user's email account. The primary concern is email account security. If an attacker gains access to a user's email account through phishing, weak passwords, or credential stuffing they can intercept the OTP sent to that email and use it to authenticate themselves on the user's behalf (Pal, 2019). This is particularly dangerous because email accounts often contain many sensitive details, such as recovery options for other accounts or links to reset passwords. An attacker who gains control over the email account essentially has a backdoor into the user's broader digital life. In addition to email account compromises, phishing attacks present another significant vulnerability. Attackers may craft deceptive emails that mimic legitimate communication from a trusted service, prompting users to enter their login

credentials or OTPs on a fake site (Hassan et.al, 2020). Even if the email is genuine, users can still fall victim to phishing attempts, which can then lead to their OTPs being captured and exploited. Furthermore, email accounts themselves are vulnerable to credential stuffing attacks, where attackers use previously leaked or stolen password databases to try and gain access to email accounts. (Nguyen, 2021). Once inside, they can intercept OTPs and compromise not just the specific account being targeted but also other accounts that rely on email for 2FA. While email-based OTPs are easier to implement and require no special devices or apps, they are ultimately less secure. The security of email-based OTPs is only as strong as the user's email account, making it a riskier choice for sensitive transactions or services requiring high levels of security.

2.1.3 Hardware-Based OTP

Hardware-Based OTP (HOTP) tokens are generated at set intervals, typically every 30 to 60 seconds (Teffandi et al., 2023). These token generators are physical devices, often resembling small key fobs, that display a new code either upon button press or after a specified period (Teffandi et al., 2023). The user then enters the generated code into the authentication system, where a server verifies it using a shared secret to ensure that both the token and a server generated code are the same. This method provides a secure and isolated approach to OTP generation, as the device operates independently of the user's phone or computer and does not require an internet connection (Almeida et al., 2024). A key advantage of HOTP tokens is their resilience against network-based attacks, such as those relying on interception over the internet or via SMS (Fejrskov et al., 2022). Since the OTP is generated on the device itself and does not require external communication, there is no need for network connectivity to receive or transmit the OTP. This makes them immune to various attack vectors, such as MitM attacks or SIM-swapping, which often target SMS or email-based OTPs (Fejrskov et al., 2022).

Additionally, hardware tokens are immune to phishing attacks. Because the OTP is generated solely by the token device and the user must physically interact with it, attackers cannot trick the user into revealing their OTPs through fake websites or emails (Plata & Calpito, 2020). However, despite these advantages, hardware-based OTP tokens have some significant vulnerabilities. One major risk is that the token devices are prone to loss, theft, or damage. If a user loses their token device, or if it is stolen, an attacker in possession of it could potentially use

it to authenticate and gain unauthorized access. This happens most especially if the token is not protected by a PIN or another layer of security (Berrios et al., 2023). In such cases, the security of the system is only as strong as the user's ability to safeguard the physical device. Another vulnerability arises if the device is tampered with or cloned, as an attacker may extract the secret used to generate the OTP and use it for unauthorized authentication (Teffandi et al., 2023). Teffandi et al., (2023) suggest the implementation of algorithms like GRAIN which is a lightweight cryptographic algorithm to help protect against the extraction of the secret key stored in hardware tokens. If an attacker attempts to tamper with the token to extract the key either through hardware reverse engineering or cloning, GRAIN's robust encryption can make it more difficult to compromise the OTP generation process. Despite this solution, the complexity in implementation and computation overhead reduces the convenience of using it as a scalable solution. Additionally, with hardware tokens, users must remember to bring the device wherever they go, which can be cumbersome (Shirvanian & Agrawal 2021)). If the token is forgotten or misplaced, accessing accounts becomes challenging, often requiring a backup OTP or a lengthy recovery process with the service provider. Despite their strong security features, the effectiveness of HOTP tokens ultimately depends on how well users manage and protect their physical devices (Teffandi et al., 2024).

2.1.4 Time-Based One-Time Password

TOTP comes as the most secure and convenient form of OTP for users (Gilsenan et al., 2023). Like the rest, it provides a temporary, time-sensitive passcode that complements traditional static passwords. This passcode, valid for a short period, typically around 30 seconds, adds an extra layer of security by ensuring that a fresh code is required for each authentication attempt (Plata & Calpito, 2020).

Bianchi & Valeriani (2023) highlight the key components of the TOTP process, which ensures both security and usability in authentication systems. The foundation of TOTP's security is the shared secret key, which is unique to the user's device and the authentication server (Bianchi & Valeriani, 2023). This key is crucial, as it is the basis for generating and verifying TOTP codes on both sides. During the initial setup, the shared secret key is typically encoded in a QR code, which the user scans with their authentication app, such as Google Authenticator or Authy (Berrios et al., 2023). The security of this shared secret is vital, as any compromise of it would jeopardize the entire authentication process (Berrios et al., 2023).

TOTP also employs timestamp-based validity to limit the time window in which a code is valid. Typically set at 30 seconds, this periodic refresh ensures that each generated code is only valid for a brief interval (Bianchi & Valeriani, 2023). As a result, any attempt to use an expired code will fail, minimizing the risk associated with stolen or intercepted codes. The limited validity period further reduces the likelihood that an attacker can misuse a code before it expires, adding an additional layer of security to the system (Dam et.al, 2024).

At the core of TOTP is the Hash-Based Message Authentication Code (HMAC) algorithm. This cryptographic function combines the shared secret key with the current timestamp to generate a hash (Dam et al., 2024). The HMAC process ensures the cryptographic security of the TOTP system, as it uses both the secret key and the time interval as input parameters. The resulting hash is unique to both the user and the specific time frame, making it extremely difficult for attackers to guess or generate the correct code without access to the shared secret key.

Once the HMAC hash is produced, it undergoes a truncation process to create a shorter, more manageable numeric code, typically 6 to 8 digits long (Dam et al., 2024). This process involves extracting a specific portion of the hash, usually the last few bytes and converting it into a decimal format to form the passcode. This step is essential for ensuring that TOTP remains user-friendly while retaining its cryptographic security (Shirvanian & Agrawal, 2021).

Once the TOTP code is generated, the user enters it alongside their static password as a secondary authentication factor. The authentication server, using the same shared secret key and current timestamp, generates its own TOTP code. If the code entered by the user matches the server's generated code, authentication is granted; otherwise, access is denied. This mutual generation and verification of codes ensure that the authentication process is secure, with both the user and the server operating within synchronized time intervals and possessing the same shared secret key (Ulqinaku et al., 2019).

2.3 TOTP Security

TOTP authentication systems are widely deployed for securing user accounts, but numerous studies have identified critical vulnerabilities that attackers can exploit. These vulnerabilities range from cryptographic weaknesses and phishing susceptibility to MitM attacks and system design flaws. While various mitigation techniques have been proposed, each solution has inherent limitations that impact its effectiveness. This section examines key vulnerabilities,

mitigation strategies, and associated limitations based on existing research.

2.3.1 Security of TOTP

TOTP is generally considered more secure than HOTP, email OTP, and SMS OTP due to several key factors that address the vulnerabilities present in these other authentication methods (Bianchi & Valeriani, 2023). Firstly, the primary advantage of TOTP over HOTP lies in its time sensitivity. While HOTP generates a one-time passcode based on a counter that increments with each authentication request, TOTP generates a passcode that is only valid for a brief, fixed period (Gilsenan et al., 2023). This time-bound nature of TOTP significantly reduces the window of opportunity for an attacker to intercept or reuse an OTP. In contrast, HOTP's reliance on a counter means that if an OTP is captured, it can potentially be used until the counter increments, making it more vulnerable to replay attacks (Teffandi et al., 2023). TOTP's time-limited nature inherently protects against this risk, ensuring that even if an OTP is intercepted, it becomes useless once it expires (Ulqinaku et al., 2019).

When comparing TOTP to email OTPs, the security benefits are even more pronounced. Email OTPs, though convenient, have inherent vulnerabilities due to the possibility of attackers gaining unauthorized access to the user's email account through phishing, weak passwords, or credential stuffing (Pal, 2019). Once an attacker compromises an email account, they can intercept OTPs sent to that email, gaining full access to accounts protected by this method. In contrast, TOTP does not rely on email infrastructure, and the passcode is generated by an independent token or app, such as a smartphone application Google Authenticator or Authy (Berrios et al., 2023). This decoupling from email infrastructure makes TOTP less susceptible to email-related attacks, offering a much higher level of security (Dam et al., 2024).

SMS OTPs, while widely used for 2fa, are also significantly more vulnerable than TOTP. SMS-based OTPs are susceptible to a range of network-based attacks, the most notable of which is SIM swapping (Hassan et al., 2020). In a SIM swap attack, a malicious actor convinces a mobile carrier to transfer the victim's phone number to a new SIM card under the attacker's control. This grants the attacker full access to any SMS-based OTPs sent to that number, allowing them to bypass two-factor authentication and gain unauthorized access to accounts. Additionally, SMS OTPs are vulnerable to MitM attacks, where attackers intercept communications between the user's phone and the server (Fejrskov et al., 2022). Because SMS relies on cellular networks,

which can be compromised or intercepted, the security of SMS OTPs is inherently weaker than TOTP. Furthermore, TOTP is also more resistant to phishing attacks compared to both SMS and email OTPs. While SMS OTPs and email OTPs can be intercepted or exploited through social engineering tactics such as phishing, TOTP-generated passcodes are produced on the user's device in real time (Shirvanian & Agrawal, 2021). Even if an attacker tries to deceive the user into providing their OTP through a fake site or message, the real-time nature of TOTP makes it harder for attackers to capture and use the code before it expires. This makes phishing attempts less effective against TOTP compared to SMS or email-based OTPs, where attackers may have more time to exploit stolen credentials.

Another crucial advantage of TOTP is that it operates independently of the network. Unlike SMS-based OTPs, which rely on the cellular network, or email-based OTPs, which depend on email services, TOTP is generated offline on a dedicated device or app (Almeida et al., 2024). This eliminates many of the vulnerabilities associated with network-based attacks, such as interception or Man-in-the-Middle attacks, and reduces reliance on potentially insecure external infrastructures (Sofian et al., 2024).

2.3.2 TOTP Vulnerabilities to Real Time Phishing

Phishing attacks targeting TOTP-based authentication systems exploit multiple vulnerabilities that undermine their security, particularly in real-time attack scenarios. These vulnerabilities include the lack of mutual authentication, susceptibility to MitM attacks, weak domain verification mechanisms, and the inherent exposure of TOTP secrets to installed applications. Despite several proposed mitigations, phishing attacks continue to bypass these defenses, highlighting the urgent need for a more resilient authentication framework.

2.3.2.1 Lack of Mutual Authentication and Susceptibility to Social Engineering

One of the fundamental security flaws in TOTP authentication is the absence of mutual authentication between users and service providers. In an ideal authentication process, both the user and the service should verify each other's legitimacy before proceeding. However, TOTP-based systems lack this crucial security measure, leaving users vulnerable to phishing attacks that exploit their inability to distinguish between a genuine authentication request and a fraudulent one. Without mutual authentication, an attacker can impersonate a legitimate service,

deceive users into entering their credentials and TOTP codes, and hijack accounts in real time (Sofian et al., 2024).

Phishing attacks leveraging this weakness have become increasingly sophisticated. Attackers create realistic fake login portals that closely resemble legitimate services. Victims are tricked into visiting these sites through deceptive emails, text messages, or social engineering tactics that falsely claim their account has been compromised or requires urgent verification. Once on the phishing site, users enter their credentials and TOTP codes, unaware that the attacker is relaying this information in real time to the actual service provider. Since TOTP codes are only valid for a short window, attackers exploit automation tools such as reverse proxy phishing kits to capture and use them before expiration (Bianchi & Valeriani, 2023). This allows adversaries to bypass two-factor authentication entirely and gain full control of the victim's account.

To mitigate this risk, mutual authentication mechanisms have been proposed. Sofian et al. (2024) suggest implementing systems where both the user and service must authenticate each other before proceeding. One approach is the use of cryptographic challenges, where the service sends a unique, verifiable challenge to the user that only a legitimate client can correctly respond to. Another proposed method is secure authentication channels, where authentication is conducted through pre-established, encrypted communication rather than relying on manual TOTP entry. Additionally, verifiable service identity indicators, such as digital certificates or visual markers that confirm a site's authenticity, have been recommended to help users distinguish genuine sites from phishing attempts.

However, despite these proposed solutions, significant challenges remain. First, mutual authentication adds complexity to the user experience, making authentication slower and more cumbersome. Users may be required to perform additional verification steps, leading to frustration and a reluctance to adopt the technology. Second, attackers are continually evolving their phishing techniques, adapting their strategies to mimic even mutual authentication prompts. Recent studies have shown that adversaries are already employing advanced UI manipulation techniques to deceive users into believing they are interacting with a legitimate service, even when additional authentication layers are present (Ulqinaku et al., 2019). Third, mutual authentication requires widespread adoption by service providers, but many platforms continue to rely on traditional TOTP authentication due to its simplicity and ease of integration. Without

industry-wide enforcement, users remain exposed to phishing threats.

Ultimately, while mutual authentication mechanisms present a step forward in addressing the vulnerabilities in TOTP-based systems, they are not a sufficient standalone solution. The continued evolution of phishing attacks, combined with usability concerns and adoption barriers, means that TOTP remains inherently vulnerable. To provide a more robust defense against phishing, security experts advocate for alternative authentication models such as hardware-based security keys (FIDO2), passwordless authentication, and phishing-resistant cryptographic authentication methods (Ulqinaku et al., 2020). Without a significant shift away from TOTP reliance, attackers will continue to exploit its inherent weaknesses, making it an increasingly ineffective security measure in high-risk environments.

2.3.2.2 Man-in-the-Middle Attacks via Reverse Proxies

MitM attacks are a critical security vulnerability in TOTP-based authentication systems. They allow adversaries to intercept authentication codes in real-time before they reach the legitimate service. Unlike conventional phishing, where attackers steal credentials for later use, MitM attacks occur dynamically during the authentication process, enabling attackers to gain immediate unauthorized access to accounts. The effectiveness of this method stems from its ability to bypass two-factor authentication protections, rendering TOTP ineffective against sophisticated adversaries (Ellahi et al., 2022).

A common technique used in these attacks is the reverse proxy method, where attackers set up a malicious intermediary server that sits between the victim and the legitimate authentication portal. The attacker clones the target website and tricks users into entering their login credentials and TOTP codes by sending phishing links disguised as legitimate requests from trusted services. Once the victim submits their authentication data, the attacker's reverse proxy relays the credentials in real-time to the actual service provider. This allows the attacker to complete the login process on behalf of the victim before the TOTP code expires (Ulqinaku et al., 2019).

Automated phishing kits have further enhanced the effectiveness of MitM attacks. These kits, widely available on underground forums, streamline the interception process, eliminating the need for attackers to manually capture and use credentials. Advanced phishing frameworks, such

as Evilginx and Modlishka, allow attackers to dynamically modify web traffic, stripping security headers and SSL protections to bypass browser security warnings. These tools also integrate real-time session hijacking capabilities, meaning attackers can maintain persistent access to accounts even after the victim logs out (Ellahi et al., 2022).

To mitigate MitM risks, researchers have proposed secure browser-based authentication and improved domain verification mechanisms. One approach involves strict transport security measures that detect and block unauthorized intermediaries attempting to intercept communication. Another method is domain verification protocols, which enforce rigorous checks to ensure that users interact only with legitimate services. Additionally, advanced browser security indicators, such as extended validation (EV) certificates and anti-phishing warnings, have been introduced to help users recognize fraudulent websites before submitting their credentials (Ulqinaku et al., 2020).

However, these mitigations face significant real-world limitations. Firstly, modern phishing kits can dynamically bypass browser security alerts by simulating valid SSL certificates or injecting misleading UI elements that convince users they are on a legitimate site. Secondly, users often fail to notice security warnings or ignore domain inconsistencies, especially when under pressure from time-sensitive phishing schemes. Research indicates that even when presented with clear indicators of an attack, a large percentage of users proceed with authentication due to habit, lack of security awareness, or urgency created by attackers (Ellahi et al., 2022).

Moreover, while domain verification can help detect illegitimate sites, many phishing domains leverage typo-squatting and lookalike characters to evade detection. For instance, attackers register domains that resemble legitimate ones, such as "g00gle.com" instead of "google.com," making it difficult for users to recognize the fraud. Additionally, browser-based authentication relies on widespread service provider adoption, which has been slow due to compatibility issues and deployment costs. As a result, TOTP-based authentication remains highly vulnerable to MitM exploitation, reinforcing the need for phishing-resistant authentication mechanisms such as hardware security keys and passwordless authentication methods.

2.3.2.3 Weak Domain Verification and Real-Time Phishing Kits

Domain verification is a crucial security mechanism intended to help users distinguish legitimate authentication portals from fraudulent ones. However, Ulqinaku et al. (2020) highlight that traditional domain verification methods are highly vulnerable to exploitation due to human error and the increasing sophistication of phishing techniques. Attackers leverage weak domain verification to deceive users into entering their credentials and TOTP codes on malicious websites designed to closely mimic legitimate services. Since TOTP authentication lacks built-in service validation, users have no way of verifying whether the authentication request originates from a legitimate source, making them susceptible to phishing attacks.

One of the primary weaknesses in domain verification is its reliance on manual URL inspection by users. Many users do not carefully examine website addresses, especially when distracted, rushed, or pressured by a seemingly urgent security notification. Attackers exploit this behavior through typosquatting, homoglyph attacks, and pixel-perfect clones of real authentication portals. For example, a phishing domain like “paypa1.com” (using the number "1" instead of the letter "l") can closely resemble “paypal.com,” tricking users into entering their credentials without suspicion (Ulqinaku et al., 2020).

The use of real-time phishing kits has significantly increased the effectiveness of domain spoofing. These automated tools allow attackers to mirror legitimate login pages dynamically, even pulling real content from the target site to ensure authenticity. Unlike traditional phishing, where static copies of websites might have subtle errors or outdated elements, modern phishing kits create nearly indistinguishable clones that update in real-time. This method enables attackers to capture both primary login credentials and the accompanying TOTP codes simultaneously, allowing immediate unauthorized access before the authentication code expires (Ulqinaku et al., 2020).

To mitigate this risk, researchers propose automated domain verification mechanisms that reduce reliance on user awareness. One such solution is security key-based authentication, which requires users to verify authentication requests using a physical device, such as a hardware security key. Additionally, browser-integrated domain validation can enforce strict security policies, ensuring that authentication prompts only appear on trusted domains (Ulqinaku et al.,

2020). These approaches remove the need for manual verification, reducing human error in the authentication process.

However, these mitigations face critical adoption challenges. Many users and organizations are reluctant to transition away from traditional 2FA methods due to usability concerns, cost, and compatibility issues. Security key-based authentication, while highly secure, requires additional hardware, which not all users are willing to adopt. Additionally, legacy systems and enterprise environments may struggle with integrating automated domain verification into existing authentication workflows, leading to inconsistent implementation across different platforms.

Furthermore, attackers have adapted to counter automated domain verification efforts. Manipulating browser UI elements allows them to bypass security warnings and create deceptive overlays that hide URL discrepancies from users. Some phishing kits fake browser address bars or inject fake SSL padlock icons, misleading users into believing they are on a legitimate domain. As a result, while domain verification mechanisms offer theoretical protection, they remain ineffective against advanced phishing techniques, reinforcing the need for stronger, phishing-resistant authentication solutions beyond traditional TOTP.

2.3.2.4 TOTP Secrets Exposed to Installed Applications

Bianchi & Valeriani (2023) identify a critical vulnerability in how TOTP secrets are stored on user devices, making them accessible to malicious applications with the right level of system permissions. Since TOTP authentication relies on a pre-shared secret between the user's device and the service provider, the security of this secret is paramount. However, many mobile authentication apps store TOTP secrets in insecure locations, such as unencrypted storage, application caches, or databases with weak access control. If an attacker gains access to these stored secrets, they can generate valid TOTP codes on demand, effectively bypassing authentication without requiring phishing or real-time interception.

One common method attackers use to exploit this vulnerability is through malicious apps with excessive permissions. Many mobile applications request broad access privileges that users frequently grant without scrutiny. If a rogue app gains read access to authentication storage, it can extract and exfiltrate TOTP secrets without the user's knowledge (Bianchi & Valeriani,

2023).Additionally, rooted or jailbroken devices further amplify this risk, as they bypass Android and iOS security mechanisms, allowing attackers to access protected application data.

Another attack vector involves exploiting inter-process communication (IPC) vulnerabilities within the Android API. Some authentication apps expose insecure APIs that allow external applications to request TOTP codes programmatically. If these APIs lack proper authentication controls, a malicious app can issue API calls to retrieve OTPs, effectively bypassing the need for user interaction (Bianchi & Valeriani, 2023).

To mitigate this threat, researchers propose enhancing Android API security and implementing stricter app permission models. Strengthening application sandboxing and requiring encrypted storage for TOTP secrets can prevent unauthorized access. Additionally, introducing hardware-backed key storage, such as Android's Trusted Execution Environment (TEE) or Apple's Secure Enclave, can ensure that authentication secrets remain protected even if an app or OS is compromised.

However, these mitigations have significant limitations. First, OS-level enforcement is inconsistent across devices and manufacturers. Many Android device vendors implement custom security policies, some of which fail to enforce strict access controls on sensitive data. Unlike Apple's more controlled ecosystem, Android's fragmentation results in varying security implementations, leaving some devices more vulnerable than others (Bianchi & Valeriani, 2023).

Second, user compliance remains a major challenge. Many users ignore security prompts or disable key security features for convenience. For example, some users disable app permission restrictions, install apps from untrusted third-party stores, or root their devices, unknowingly exposing themselves to TOTP secret theft.

Lastly, attackers continuously find new ways to exploit system vulnerabilities, often staying ahead of security enhancements. Even if stricter API security and permission controls are enforced, attackers can still exploit zero-day vulnerabilities in mobile operating systems to access stored authentication data. This cat-and-mouse game between security patches and new attack techniques renders security enhancements a temporary measure rather than a permanent solution.

Ultimately, while improving TOTP secret storage security is a necessary step, it does not fully address the root issue. As long as TOTP authentication relies on static, device-stored secrets, attackers will continue to develop methods to extract and misuse them. A more phishing-resistant and secretless authentication model such as public-key cryptography-based authentication or FIDO2/WebAuthn is required to eliminate the risks associated with secret storage vulnerabilities.

2.4 Behavioral Analysis

2.4.1 Introduction to Behavioral Analysis

Behavioral analysis is emerging as a powerful tool in cybersecurity, particularly in the realm of authentication systems. Behavioral analysis adds layer of protection by continuously monitoring and learning from user behavior to identify anomalies indicative of potential threats (Mihailescu et al., 2023). This approach is especially crucial in modern cybersecurity, where attackers often employ sophisticated tactics to bypass conventional security measures. By recognizing and acting on deviations from a user's typical behavior, behavioral analysis can detect unauthorized access attempts or compromised accounts with greater precision, providing more responsive and context-aware security.

2.4.2 How Behavioral Analysis Works

Behavioral analysis operates by establishing a profile or baseline of a user's normal behavior patterns. These patterns are derived from various user interactions with a system or application, such as the times at which they typically log in, the devices they use, the geographic locations from which they access the system, and even the specific ways they navigate within the application. By recording these behaviors, a normal pattern is created, which serves as the reference point for identifying deviations that may indicate malicious activity. Once this baseline is established, the system uses advanced machine learning algorithms or rule-based models to continuously monitor user behavior in real-time. When any significant deviation from the established norm is detected, the system can trigger alerts or implement additional security measures, such as multi-factor authentication or manual intervention, to prevent unauthorized access or account compromise.

2.4.3 Parameters Monitored in Behavioral Analysis

In behavioral analysis, several key parameters are closely monitored to detect anomalies that could signal potential threats or breaches. Among the most observed metrics are login times, device and browser fingerprinting, geolocation and IP address patterns, the use of VPNs or proxies, navigation and interaction patterns, and typing and input behavior.

Login Times are often the first parameter flagged for anomalies. Users tend to have a predictable login schedule based on their work or personal routines. Behavioral analysis systems record these patterns and highlight any login attempts outside of the usual times, such as late-night logins, access on weekends, or other atypical hours. Such deviations could indicate an attack, particularly if the user's device is inactive during these times.

Device and Browser Fingerprinting is another significant aspect of behavioral monitoring. Devices and browsers create unique identifiers based on their hardware and software characteristics, which can be tracked over time. If an attempt is made to log in from an unfamiliar device or browser, even with the correct credentials or a valid TOTP, it is flagged as a suspicious activity. This type of monitoring helps mitigates the risks of attackers using stolen credentials on new devices or browsers.

Geolocation and IP Address Patterns are also closely examined. Typically, a user logs in from specific geographic locations and IP address ranges. Behavioral analysis can detect deviations from this pattern such as login attempts originating from a different country or an unexpected IP address indicating possible attacks, like phishing, DNS hijacking, or IP address spoofing. By tracking the location history of logins, such irregularities can be identified quickly.

The Use of VPNs or Proxies is frequently associated with malicious actors trying to obscure their true location. When an attacker utilizes VPNs or proxies to mask their IP address, it can be a clear red flag. Particularly, if the login attempt comes from a region or IP address that contradicts the user's past behavior, this inconsistency is flagged for review.

Navigation and Interaction Patterns are analyzed to build a comprehensive picture of how a user typically interacts with a system. This includes observing the pages visited, the time spent on each page, and how forms are filled out. A sudden shift in behavior such as rapidly skipping

through the login process or erratically navigating through pages can indicate an attempt to bypass security measures or an active phishing attack.

Typing and Input Behavior is another sophisticated technique used to assess user identity. By analyzing factors like typing speed, pauses, and error frequency, behavioral analysis systems can build a unique user profile. A noticeable difference in these patterns, such as an attacker typing more slowly or inconsistently compared to the legitimate user, can help detect compromised accounts or unauthorized access.

2.4.4 Detection of Anomalies

The core strength of behavioral analysis lies in its ability to detect anomalies or deviations from a user's established behavior. By continuously monitoring and comparing real-time user actions to a baseline profile, behavioral analysis can identify suspicious activity that might otherwise go unnoticed by traditional security measures. The system creates a personalized behavioral fingerprint for each user, based on patterns such as login times, device types, IP addresses, geographical locations, and interactions with the system. When a user exhibits actions that deviate from this baseline, the system can trigger alerts, warnings, or initiate additional authentication steps to ensure that the activity is legitimate. These anomalies could manifest as unusual login times, the use of unfamiliar devices or browsers, login attempts from new or distant geographical locations, or even irregularities in how the user navigates through the system or interacts with the application, such as faster or erratic behavior (Mihailescu et al., 2023).For instance, if an attacker successfully steals a user's credentials through a phishing attack but attempts to log in from a device or location that is inconsistent with the user's typical behavior, the behavioral analysis system can detect this inconsistency and flag it as suspicious. In this case, even if the attacker has the correct login credentials and the TOTP, the system might prompt the user to authenticate via additional verification steps, such as biometric authentication or a secondary code sent to a trusted device. This type of anomaly detection can prevent unauthorized access even if the attacker has gained partial control of the user's credentials (Önal et al., 2024).

Similarly, behavioral analysis plays a crucial role in identifying MitM attacks, where an attacker intercepts and manipulates communication between a legitimate user and the server. These types

of attacks are particularly difficult to detect, as they occur in real time, but behavioral analysis can identify unusual patterns that may indicate an interception. For example, if there is a noticeable delay in the time it takes for the user's credentials to be validated or if the navigation sequence is inconsistent with the user's typical behavior, the system can flag the transaction. Additionally, if an attacker is using an unfamiliar network or device, these factors can trigger alerts, prompting further verification steps to prevent successful exploitation of the TOTP system (Mihailescu et al., 2023). Furthermore, if an attacker manages to gain unauthorized access to a user's account, the user's behavior may differ significantly from their usual interactions with the system. For instance, the attacker might attempt to access parts of the system that the legitimate user typically avoids or perform actions that are inconsistent with their usual behavior. Behavioral analysis can spot these deviations and raise red flags by comparing current activity to historical usage data. This helps to ensure that even if the attacker bypasses traditional security measures like TOTP, their actions are unlikely to go unnoticed (Önal et al., 2024).

Overall, by combining behavioral analysis with traditional security mechanisms like TOTP, organizations can create a more robust multi-layered defense. This integrated approach strengthens security by adding an additional layer of context-aware protection that continuously learns from user behavior, detects suspicious activity, and responds accordingly. With the ability to identify complex attack patterns that go beyond simple credential theft, behavioral analysis offers a proactive solution to the evolving threat landscape in cybersecurity.

2.4.5 Behavior analysis in TOTP applications

Recent advancements in MFA systems have increasingly adopted behavior analysis techniques to strengthen identity verification and reduce false positives. Major platforms such as Google and Microsoft have integrated adaptive MFA frameworks that utilize behavior-based algorithms to determine whether secondary authentication factors should be enforced (Berrios et al., 2023).
For instance, Microsoft's Azure Active Directory Conditional Access employs machine learning models to evaluate signals like geolocation, user behavior history, and device compliance. When anomalies are detected such as logins from unusual locations or unrecognized devices the system escalates the authentication process by requiring stronger factors, such as TOTP or biometric verification (Sofian et al., 2024). Similarly, Google's Risk-Based Authentication analyzes device familiarity, login frequency,

and location-based heuristics to flag potentially malicious sign-in attempts without unnecessarily burdening legitimate users (Berrios et al., 2023).

These context-aware systems have proven highly effective in minimizing friction for end-users while boosting resilience against phishing and credential theft. However, despite their strengths, significant limitations remain in how behavioral analysis is applied particularly in the context of TOTP-based applications. First, most behavioral analysis systems operate at the identity provider or service backend level, meaning that the intelligence gathered is not directly integrated into client-side TOTP apps like Google Authenticator, Microsoft Authenticator, or Authy (Önal et al., 2024). This separation restricts the real-time feedback loop necessary for dynamic client-side defense mechanisms, such as warning users of risky login patterns or automatically denying OTP submissions under suspicious conditions.

Second, behavioral modelling techniques often rely on static heuristics or pre-defined rules that may fail to adapt to sophisticated and evolving attack tactics. For example, rules based solely on IP geolocation or device fingerprints might overlook subtle but high-risk anomalies, such as low-frequency account probing or coordinated login attempts using spoofed environments (Önal et al., 2024).

Furthermore, many commercial implementations rely on proprietary black-box machine learning models, which offer little transparency to security researchers or enterprise administrators. These models are also difficult to audit or fine-tune for domain-specific requirements, especially in edge environments where low latency and contextual awareness are essential for timely threat response (Önal et al., 2024).

Another area of concern is the underutilization of behavioral biometrics, such as keystroke dynamics, swipe patterns, or accelerometer-based gait recognition. While academic studies have demonstrated their potential for continuous authentication, practical deployment in production systems is rare primarily due to privacy concerns, computational costs, and integration challenges (Mihailescu et al., 2023).

While behavior analysis has enriched MFA with context-aware risk mitigation, its direct integration into TOTP apps remains largely unexplored.

2.5 Summary

Real-time phishing remains a persistent and severe threat to TOTP-based authentication systems. Attackers exploit inherent vulnerabilities such as the lack of mutual authentication, weak domain verification, and reliance on user-inputted authentication codes. Techniques like phishing kits and reverse proxy attacks allow adversaries to intercept and use TOTP codes in real-time, rendering traditional two-factor authentication ineffective against modern phishing strategies.

Several mitigations have been proposed, including mutual authentication mechanisms, anti-phishing proxies, and domain verification techniques. However, these solutions face significant challenges. Mutual authentication increases authentication complexity and delays, which can negatively impact user experience. Anti-phishing proxies require additional user interaction and may not be widely adopted due to usability concerns. Domain verification and security key-based authentication offer stronger protection but depend on widespread adoption, which is hindered by resistance to shifting away from traditional 2FA methods.

Given the shortcomings of these countermeasures, it is evident that TOTP-based authentication alone is insufficient to defend against sophisticated phishing attacks. Attackers continue to bypass proposed protections, demonstrating the urgent need for a more robust, phishing-resistant authentication mechanism.

Table 2.1: Summary of papers analyzed

Paper	**Authors**	**Vulnerability**	**Mitigation Proposed**	**Limitations**
Enhancing Authentication Security: Analyzing Time-Based One-Time Password Systems	Sofian et al. (2024)	TOTP codes susceptible to phishing and relay attacks	Mutual authentication mechanisms	Increases authentication complexity and latency

Is Real-time Phishing Eliminated with FIDO? Social	Ulqinaku et al. (2020)	Attackers bypass 2FA using phishing techniques	Domain verification and security key-based authentication	Users may resist moving away from traditional 2FA methods

Engineering Downgrade Attacks				
A User-Friendly Two-Factor Authentication Method Against Real-Time Phishing Attacks	Sun et al. (2022)	Real-time phishing attacks capture and reuse TOTP codes	Anti-phishing proxy to strengthen authentication	Requires additional user interaction, adoption hurdles
2FA-PP: 2nd Factor Phishing Prevention	Ulqinaku et al. (2019)	MitM attacks enable real-time interception of TOTP codes	2FA-PP method using direct browser-device communication	Requires browser and device compatibility
Analyzing 2FA Phishing Attacks and Their Prevention Techniques	Ellahi et al. (2022)	Reverse proxy techniques enable MitM phishing attacks	Improved domain verification and secure browser-based authentication	Requires widespread adoption by service providers

2.6 Conceptual Framework

This conceptual framework provides a structured approach for developing and implementing a behavioral analysis-based phishing detection system for TOTP applications. Figure 2.1 illustrates the conceptual diagram, highlighting the key components and flow of the system

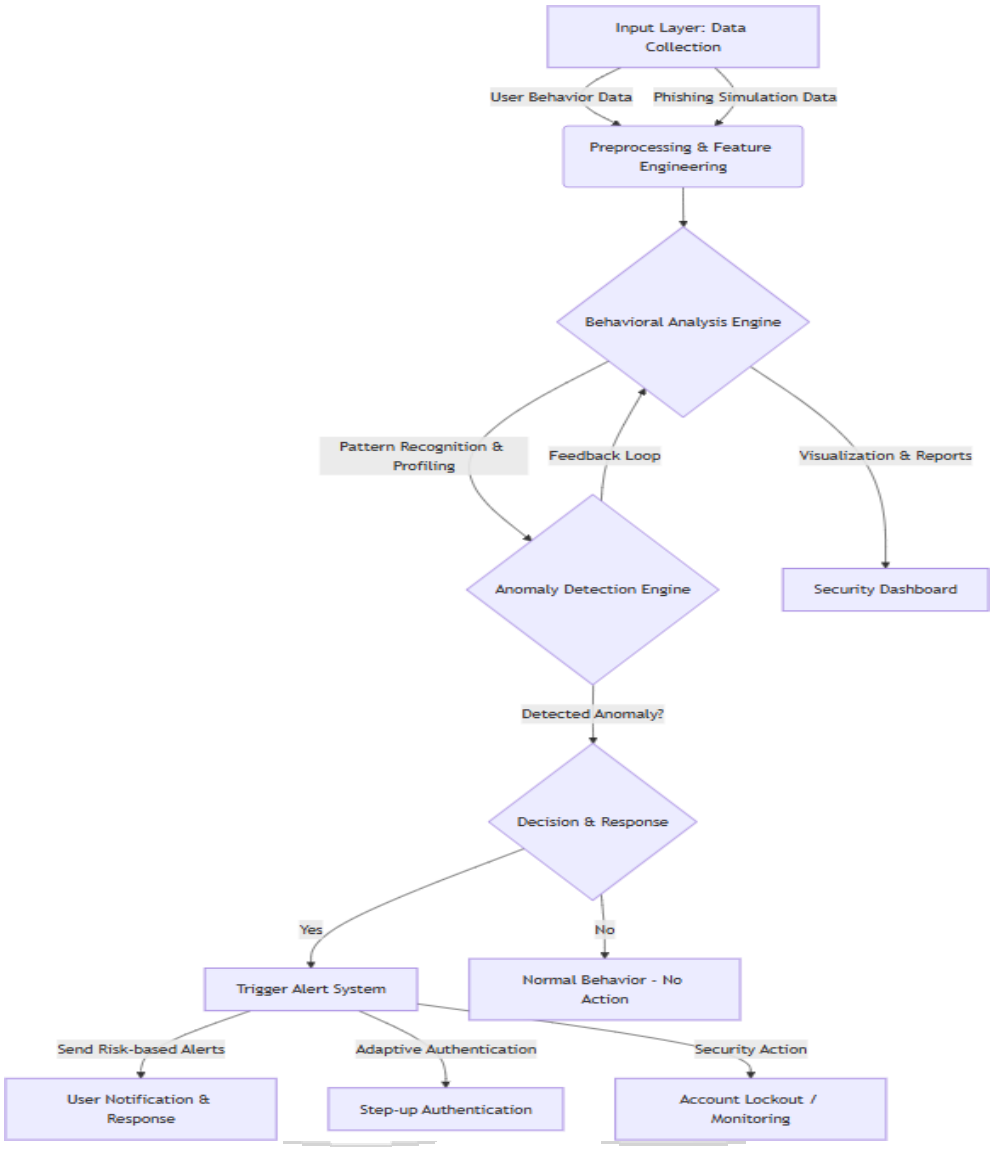


Figure 2.1: Conceptual Diagram

The Input Layer (Data Collection) serves as the foundation for the entire security framework. This layer captures essential user behavior data, including login patterns, session durations, and device information. This data helps build a comprehensive profile of each user's typical interaction with the system. The Preprocessing & Feature Engineering stage is crucial for transforming raw data into a usable format for analysis. This step involves data normalization and cleaning to remove inconsistencies or errors in the collected data. Key behavioral features are extracted from the raw data, ensuring that the information is structured and ready for deeper

analysis. The Behavioral Analysis Engine is where the system begins to interpret the data. It utilizes rule-based heuristics and statistical modeling to analyze user behavior and detect any deviations from established patterns. By constantly monitoring user activities, the engine can identify anomalies, such as unusual login times or atypical device usage. These behavioral patterns serve as a baseline, making it easier to spot irregularities that may indicate potential security threats. Once behavioral patterns are established, the Anomaly Detection Engine takes over to identify specific threats. This engine is trained to detect indicators of phishing, unauthorized access attempts, or suspicious login activities. It assesses each login or session based on patterns that diverge from the norm, flagging any discrepancies as potential security risks. If an anomaly is detected, the system triggers alerts to security teams or directly to users, depending on the severity of the threat. The Decision & Response Layer is responsible for determining the appropriate security response once an anomaly is identified. If a deviation is detected, this layer triggers security measures such as alerts to the affected users. If no anomalies are found, the system continues normal operations without intervention, ensuring a seamless user experience. Finally, the Visualization & Feedback Loop is a crucial component for continuous improvement. The security dashboard provides real-time insights into user behavior trends, offering a visual representation of the data and detected anomalies. This feedback loop is integral to refining the anomaly detection model over time.

Chapter 3: Research Methodology

3.1 Introduction

This chapter presents the methodology adopted for the design, development, and validation of a behavioral analysis-based phishing detection system. The methodology follows a structured approach that includes investigating phishing attack techniques, analyzing existing security solutions, developing a prototype system, and validating its effectiveness through controlled experiments. The research employs a combination of literature review, system modeling, attack simulation, and real-time behavioral monitoring to enhance TOTP authentication security.

3.2 Research Methodology

The proposed system has two components. An Android TOTP application responsible for device fingerprinting**,** capturing essential device attributes to uniquely identify and verify users' devices during authentication and a behavior analysis service that analyzes login attempts in real-time to identify potential phishing risks. The research methodology consists of several interconnected phases, each contributing to the overall development and evaluation of the proposed system. They include:

3.2.1 Investigating Real-Time Phishing Attacks on TOTP Apps

To gain a thorough understanding of real-time phishing attacks on TOTP applications, the study underwent the following stages. A comprehensive literature review using academic sources such as IEEE Xplore, ResearchGate, and Google Scholar. The objective was to identify common attack vectors used by adversaries and assess the current state of research in phishing detection. Real-world phishing incidents were analyzed to understand attacker methodologies. Case studies of successful phishing attacks against two-factor authentication systems were examined to identify common tactics used. Phishing attacks are simulated using Evilginx2 to replicate real-world phishing scenarios. This practical experimentation provided insights into how adversaries manipulate TOTP authentication and highlighted key behavioral indicators that can be used to detect phishing attempts.

3.2.2 Analyzing Current Solutions

A comparative analysis of existing phishing detection solutions was conducted to identify strengths and weaknesses. This phase reviewed traditional phishing detection methods, identifying their weaknesses and limitations. This phase also evaluated the effectiveness of behavioral analysis techniques in detecting phishing attempts, emphasizing their advantages over conventional methods. It also identified critical gaps in current solutions to inform the design and development of the prototype.

3.2.3 Prototype Development

The development of the behavioral analysis-based prototype follows the Agile Software Development Life Cycle (SDLC). The process includes:

3.2.3.1 Requirements Gathering and Analysis

A comprehensive assessment was conducted to understand common phishing tactics targeting TOTP authentication. This assisted in identifying attack vectors used in such attacks, mitigations and limitations present. Key security metrics such as login frequency anomalies, geolocation inconsistencies, device mismatches, OTP replay patterns, and suspicious IP behaviors such as VPN/proxy usage that can be used in the behavior analysis service were identified. Additionally feature specifications were defined to determine core functionalities for real-time phishing detection. Regulatory and compliance considerations affecting the system were also identified.

3.2.3.2 System Design

An architectural model of the system was created using Draw.io to visualize the interaction between its two components. Sequence and class diagrams were developed to illustrate the user authentication flows for each component and their integration. The database schema was designed in MySQL Workbench to ensure efficient storage and retrieval of authentication logs and behavioral data. Additionally, UI/UX wireframes for the Android TOTP application are designed using Figma.

3.2.3.3 Development

The behavior analysis service was implemented in Golang, utilizing GORM for database interactions and MySQL for persistent storage. The TOTP authenticator application was developed using Kotlin on Android Studio IDE. Version control is maintained using GitHub, for efficient code management.

3.2.3.4 Testing

Unit testing was conducted on the backend using Go's built-in testing framework, while the Android application underwent testing with JUnit. To validate the effectiveness of the phishing detection mechanism, simulated phishing attacks were performed using Evilginx2. Controlled experiments were then conducted to assess the system's ability to distinguish between legitimate and malicious login attempts.

3.2.3.5 Deployment

The behavior analysis system was deployed in a containerized environment using Docker Compose. A monitoring stack consisting of Prometheus and Grafana was implemented to provide real-time visualization of phishing detection and system performance metrics

3.2.3.6 Monitoring and Evaluation

Detection accuracy was measured using performance metrics, including true positive (TP), false positive (FP), true negative (TN), and false negative (FN) rates. Performance testing was conducted using Locust to simulate high traffic loads and evaluate the system's scalability. The response time for detecting phishing attacks was measured using Locust.

3.2.4 Prototype Testing and Validation

To ensure the effectiveness, security, and reliability, a structured approach was adopted for prototype testing and validation. The testing process was conducted in multiple phases, each designed to rigorously evaluate different aspects of the system's functionality, performance, and resilience against real time phishing attacks.

The testing process begun with unit testing, where individual components of the proposed system were tested in isolation. The behavior analysis service, implemented in Golang, underwent automated unit tests using Go's built-in testing framework for verification. The Android TOTP application was tested using JUnit and Espresso to validate its logic and UI interactions. Following unit testing, integration testing was performed to assess how the two system components interact. The communication between the android TOTP application and behavioural analysis service was tested to ensure seamless data exchange and proper authentication workflows.

Next, functional testing was conducted to validate key system operations under various scenarios. Test cases were designed for the various scenarios. This was used to evaluate the functionalities defined in the SDLC are working correctly

To assess the system's ability to detect phishing threats, controlled phishing attack simulations was carried out. Using Evilginx2 phishing scenarios were recreated, in an attempt to intercept OTP codes and session tokens. The behavior analysis service was tested to verify its ability to flag unusual authentication patterns based on the behavioral key metrics identified in the first phase of the SDLC.

Once the detection mechanisms were validated, performance testing was conducted using Locust, a Python-based load testing tool, to evaluate system scalability under high user traffic. Multiple concurrent login attempts were simulated to measure API response times, database query performance, and the system's ability to process real-time behavioral analysis without latency issues. The goal was to ensure that the behavior analysis service remains responsive and efficient even under heavy authentication workloads.

Finally, to measure the effectiveness of the phishing detection system, detection accuracy metrics were collected. The system's ability to correctly classify phishing attempts was analyzed using true positive (TP), false positive (FP), true negative (TN), and false negative (FN) rates.

3.3. Ethical Considerations

This research does not involve the collection of real user data. Instead, all data used for testing and validation was synthetically generated using Locust. However, ethical considerations were followed to ensure responsible research practices, transparency, and fair access to security advancements.

3.3.1 Data Integrity and Responsible Experimentation

The synthetic data used in this research was generated in a controlled manner to ensure realistic simulation of authentication patterns, phishing attempts, and system responses. The generated data was used strictly for the purpose of evaluating the system's effectiveness in detecting phishing attacks. No real user credentials, personal data, or confidential information were involved in the experimentation process.

3.3.2 Transparency and Research Integrity

The methodology, including how synthetic data was generated, is fully documented to allow reproducibility by other researchers and practitioners. The results of the research, including the phishing detection model and system performance metrics, were accurately reported without bias or manipulation. Open-source implementation is provided on GitHub, allowing for independent verification and community-driven improvements.

3.3.3 Fair Access to Research Benefits

The research findings and the developed phishing detection system are freely accessible on GitHub at https://github.com/jaycynth/behaviour-analysis-totp-phishing, ensuring that organizations of all sizes can benefit from the security improvements. The system's architecture, implementation, and documentation are openly available and licensed for reuse and extension, supporting further research and community-driven security enhancements.

3.3.4 Security and Compliance

While no real user data is processed, all generated data is secured against unauthorized access to maintain research integrity. The research followed ethical guidelines in cybersecurity, ensuring that phishing simulations and attack techniques were responsibly tested in a controlled environment.

Chapter 4: System Design and Architecture

4.1 System Overview

The proposed system comprises an Android-based authenticator app designed in compliance with the RFC 6238 standard for TOTP, alongside a behavior analysis system that performs real-time phishing detection. The system operates through the following steps:

Step 1: User Scans the QR Code (Secret Key Sharing)

The user scans a QR code using the TOTP authenticator app, which contains several key pieces of information. This includes the TOTP Secret Key, a Base32 encoded shared secret used for OTP generation, as well as the User ID to identify the user. The QR code also includes the issuer (the name of the app or service), the account name (typically the user's email), the algorithm used (SHA1), the number of OTP digits (set to 6), and the OTP validity period (set to 30 seconds). Upon scanning, the TOTP authenticator app extracts and securely stores the secret key locally, enabling it to generate time-based one-time passwords for future authentication.

Step 2: Device Synchronization (New Feature)

After scanning the QR code, the TOTP app registers the device with the behavior analysis server. During this process, the app collects essential device metadata and sends it to the server for device binding. This includes the User ID extracted from the QR code, the Device ID (such as the Android Secure ID), the OS version for example, Android 13, the User Agent (which contains app version and OS information), the public IP address (used for GeoIP tracking), and a timestamp (for synchronizing time). This registration ensures that only the specified, registered device is authorized to generate OTPs. If the user attempts to scan the QR code on multiple devices, only the first registered device is trusted, adding an additional layer of security to the authentication process.

Step 3: OTP Generation (Standard TOTP Behavior)

The app locally generates OTPs using the TOTP secret and timestamp following the RFC 6238. No data is sent to the backend.

Step 4: User Submits OTP (Login & Verification)

The user enters the OTP to log in, and the backend receives several key pieces of information for authentication. This includes the User ID, the OTP code entered by the user, the Device ID from the login request, the IP address from which the login attempt is made, and the timestamp marking the time of the login attempt. This data is then used to verify the authenticity of the request and ensure that the login is being made from a trusted device and location.

Step 5: Behavioral Analysis & Phishing Detection

The backend then checks with the behavior analysis server to validate the login attempt using several guidelines. It first verifies whether the OTP was generated on the registered device. The server also checks if the IP address from which the login attempt is being made matches the IP address used during OTP generation. Additionally, it ensures that the OTP is not being replayed, preventing any potential security breaches. The server compares the current device with the one used in the last successful login, looking for any discrepancies. Finally, it assesses whether the login location has changed too rapidly, which could indicate suspicious activity. These checks are designed to ensure that the login attempt is legitimate and not an attempt to bypass security protocols. Figure 4.1 shows the sequence diagram for the system overview. This diagram illustrates the step-by-step flow of the proposed system, which integrates an Android-based authenticator app compliant with RFC 6238 for TOTP, alongside a behavior analysis system for real-time phishing detection.

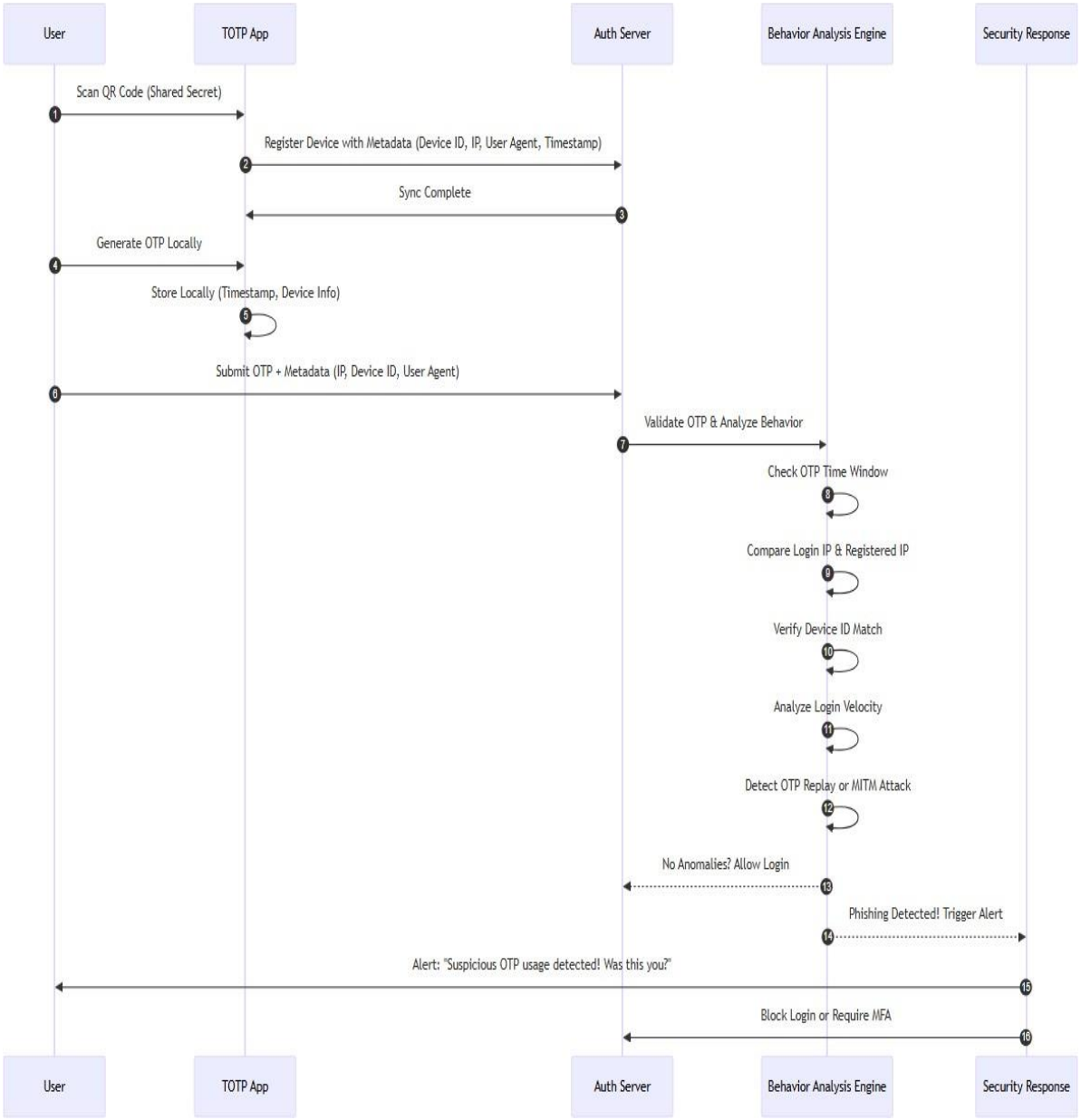


Figure 4.1: Sequence Diagram depicting the high-level flow of operations within the system

4.2 System Architecture

4.2.1 TOTP App Architecture

The Android TOTP app is developed using Kotlin, following the Model-View-Intent (MVI) architecture to ensure a reactive and scalable UI. MVI is an architectural pattern used in Android development that emphasizes unidirectional data flow and reactive state management. This approach ensures a clear separation of concerns, making applications more scalable, maintainable,

and testable. The TOTP application has three main functionalities; Scan QR codes for secret key synchronization and securely encrypt and store the secret key, synchronize with the behavior analysis service to submit the device fingerprint and generate OTPs for authentication with a validity period of 30seconds. Figure 4.2 highlights the sequence diagram for the main functionalities of the TOTP app. This diagram outlines the key operations of the TOTP authenticator app, illustrating the sequence of events involved in generating TOTPs, securely storing the secret key securely and generating the device fingerprint for syncing the application with the behavior analysis server.

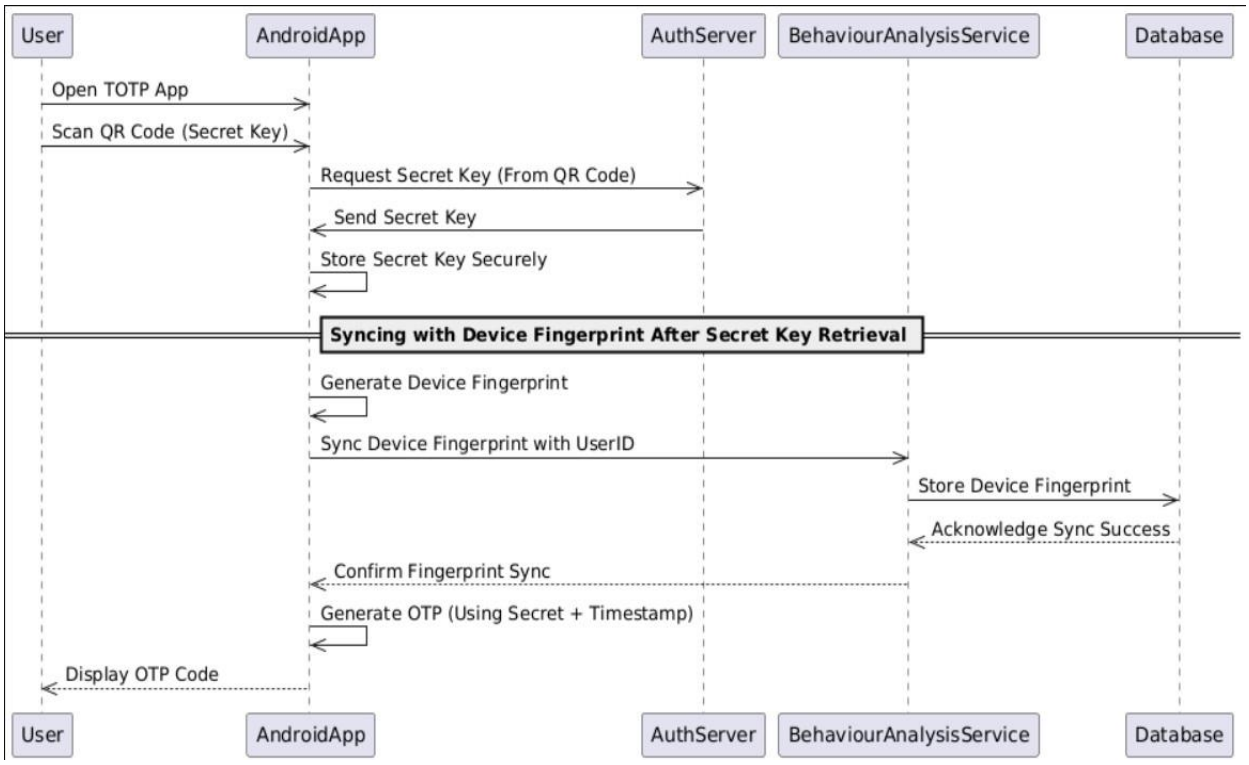


Figure 4.2: Sequence Diagram for the TOTP Application main functionalities

The TOTP application interacts with the behavior analysis server using a RESTful API for syncing the device fingerprint. Figure 4.3 shows a high-Level architecture diagram. This diagram illustrates the high-level architecture of the system, demonstrating how the TOTP authenticator app communicates with the behavior analysis service via a RESTful API. The app transmits the device fingerprint, along with other relevant data such as device metadata and user information, to the behavior analysis server.

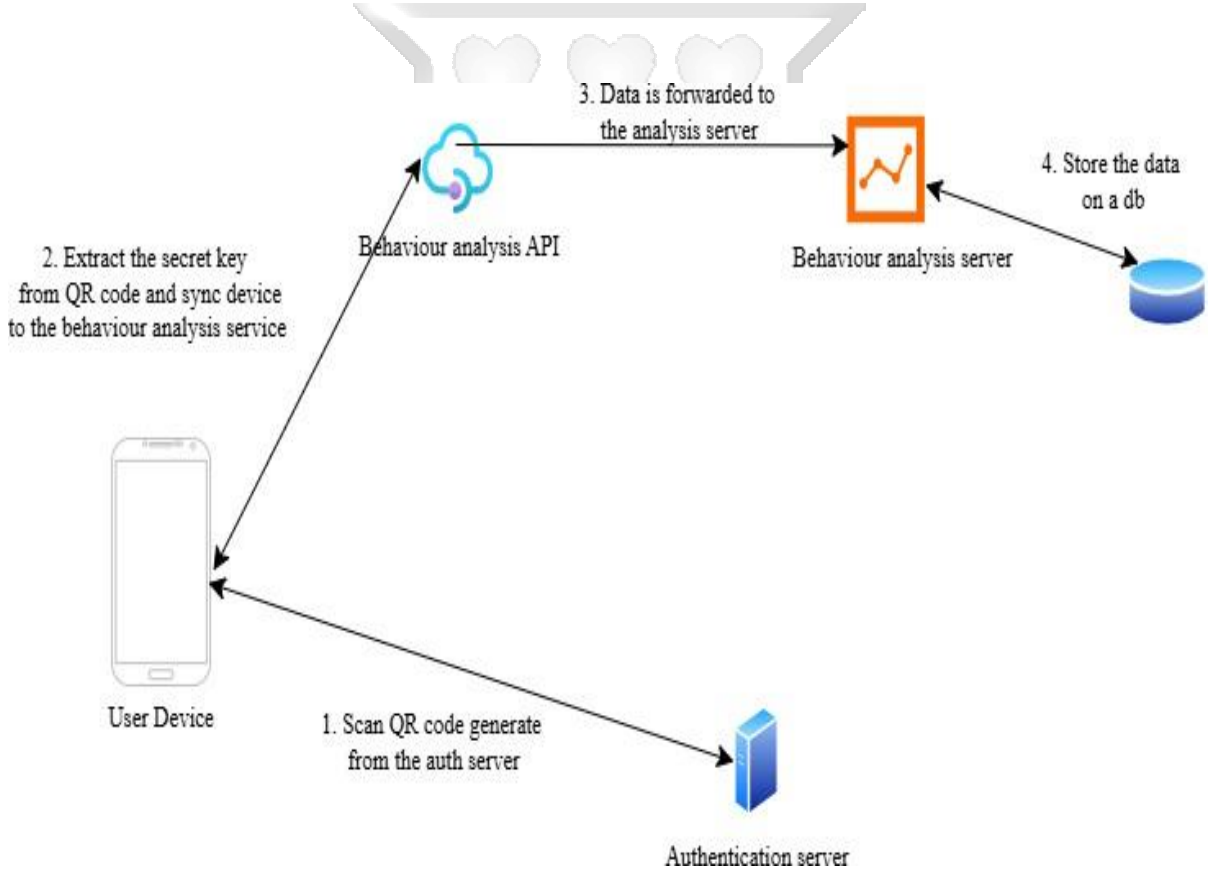


Figure 4.3: High-Level Architecture showing the interaction between TOTP app main functionalities and the Behavior analysis service

4.2.2 Behavior analysis system Architecture

The behavior analysis service consists of two core components: the real-time phishing detection service and the alerting system. These components work in tandem to identify and respond to phishing attacks. The phishing detection service operates on a polling mechanism. It is invoked when a user attempts to authenticate. During each authentication attempt, the system collects login metadata such as IP address, device fingerprint and geolocation data. The engine then analyzes this data using behavioral heuristics to determine whether the login attempt exhibits signs of phishing. If the analysis suggests a high likelihood of phishing, such as an OTP replay attack, unusual geolocation changes, or multiple rapid login attempts from different locations, the detection service classifies the login as suspicious. At this point, it triggers the alerting system, which responds in real time by notifying a user via email and logging the event for further security analysis. The alerting system ensures that the affected user is promptly informed of the suspicious activity. Figure 4.4 highlights the core components of the behavioral analysis system and their interactions with the TOTP application. It illustrates the flow of data between the TOTP app, the behavior analysis server, and the backend system during the authentication process.

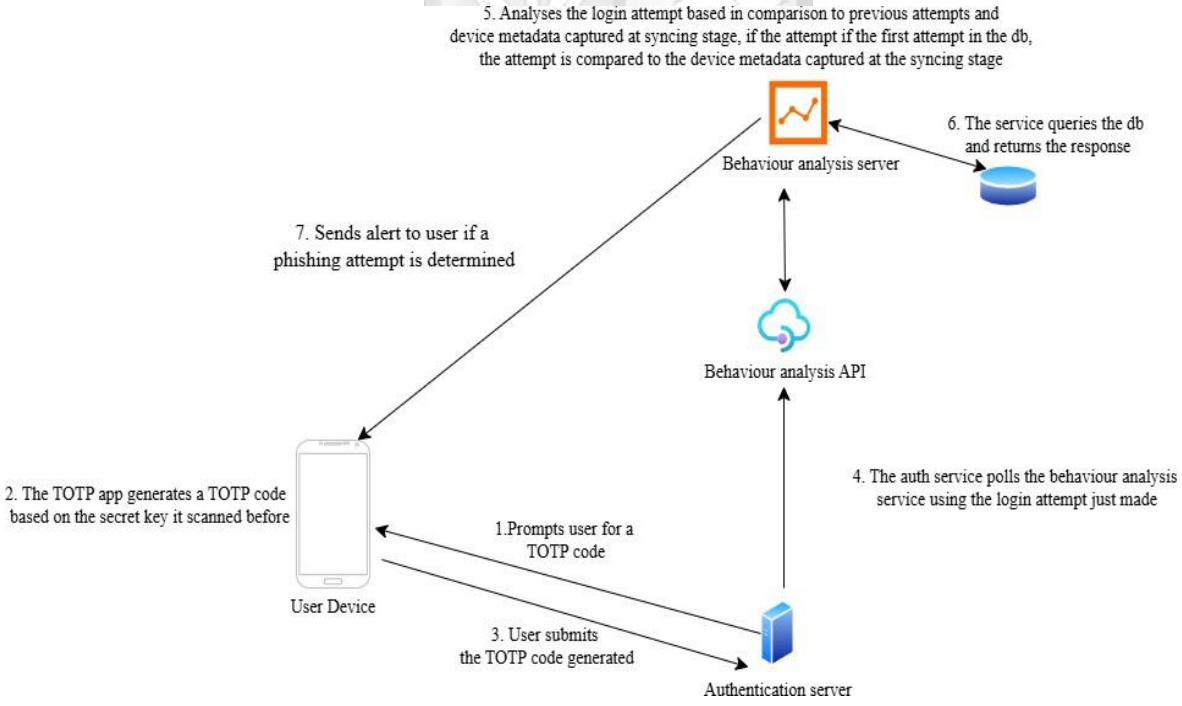


Figure 4.4: Behavioral Analysis Architecture Flow, Depicting the System's Core Components and Interactions with the TOTP application

4.2.3 Algorithm architecture

When a user initiates a login request, the system begins by fetching the user's recent login history, retrieving up to 50 past authentication attempts from the database. If historical login data is available, the system identifies the most recent login attempt and extracts key behavioral insights. The engine then calculates behavioral baselines, including the user's average login intervals, common login locations, and frequently used devices. These baselines serve as reference points for detecting deviations that may indicate suspicious activity. Once the behavioral analysis is complete, the system computes a risk score for the login attempt. This score is determined by evaluating multiple security heuristics. First, a Geo IP analysis checks whether the login originates from an unexpected location. If the new login location significantly differs from the last known location, the system calculates the geographic distance and assigns a corresponding risk value. For instance, an unusually large geographic shift such as logging in from a different country within a short time frame, raises a high-risk alert. Additionally, the system assesses the potential for OTP replay attacks by comparing the current OTP hash with the last known OTP entry. If an exact match is detected, the system flags the login as a replay attack, significantly increasing the risk score. Similarly, device mismatch detection identifies whether the login is originating from a device unfamiliar to the user. If the new device does not match the most used device in the login history, the risk score is increased accordingly. To further enhance detection accuracy, the system performs time-based anomaly detection. It examines the time interval between login attempts and compares it with the computed average login interval and standard deviation. If the login occurs at an unusual time, such as an immediate successive login outside the expected pattern, a high deviation (Z-score) is recorded, increasing the risk score. In addition to behavioral analysis, the system employs parallel network-based heuristics to identify further phishing indicators. Three concurrent checks are performed to enhance security. The first is high-frequency login detection, which monitors whether multiple login attempts have occurred within a short period, signaling potential brute force or credential stuffing attacks. The second

check looks for multiple IP addresses used within a short duration, helping to identify whether the user's account has been accessed from various locations in a brief timeframe, which may suggest fraudulent activity. The third check is network reputation analysis, which assesses whether the login attempt originates from a known malicious IP, VPN, or Tor exit node. If the IP reputation service flags the source as malicious, the system raises the risk score and logs the threat, ensuring additional scrutiny is applied to potentially harmful login attempts. Once all heuristics have been evaluated, the system aggregates the risk score. If the score exceeds a defined threshold of 75 or higher, the login attempt is classified as a phishing risk. Table 4.1 shows the risk calculation table that the algorithm follows.

Table 4.1: Breakdown of Risk Score Calculation for the Behavioral Analysis Service

Risk Factor	**Risk Score Contribution**
Geo IP Distance > 5000km	+30
Geo IP Distance > 1000km	+15
OTP Replay Detected	+40
New Device Used	+25
Unusual Login Time	+20
Multiple IPs in 1 Hour	+20
High Login Frequency	+20
Malicious IP Detected	+50
VPN Detected	+15
Tor Node Detected	+30

In cases where the threshold is 75 or greater, the alerting system is triggered, dispatching security alerts via email. The alert contains key details, including the user's ID, IP address, device information, location, and a breakdown of risk factors. Additionally, the system provides users with the option to confirm or deny the legitimacy of the login attempt. If the user marks the alert as a false positive, the model updates accordingly to refine its detection accuracy. Finally, the system safely logs the authentication attempt in the database, ensuring that all data is persisted

and auditable for future analysis. If any error occurs during the transaction process, the system rolls back the operation to prevent inconsistencies. This ensures data integrity while maintaining robust phishing detection capabilities. Figure 4.5 shows the sequence diagram for the behavior analysis server algorithm. This diagram illustrates the sequence of operations performed by the behavior analysis server during the authentication process. It details how the server processes and analyzes user behavior data, including device fingerprints, login patterns, and contextual information such as IP addresses and geolocation.

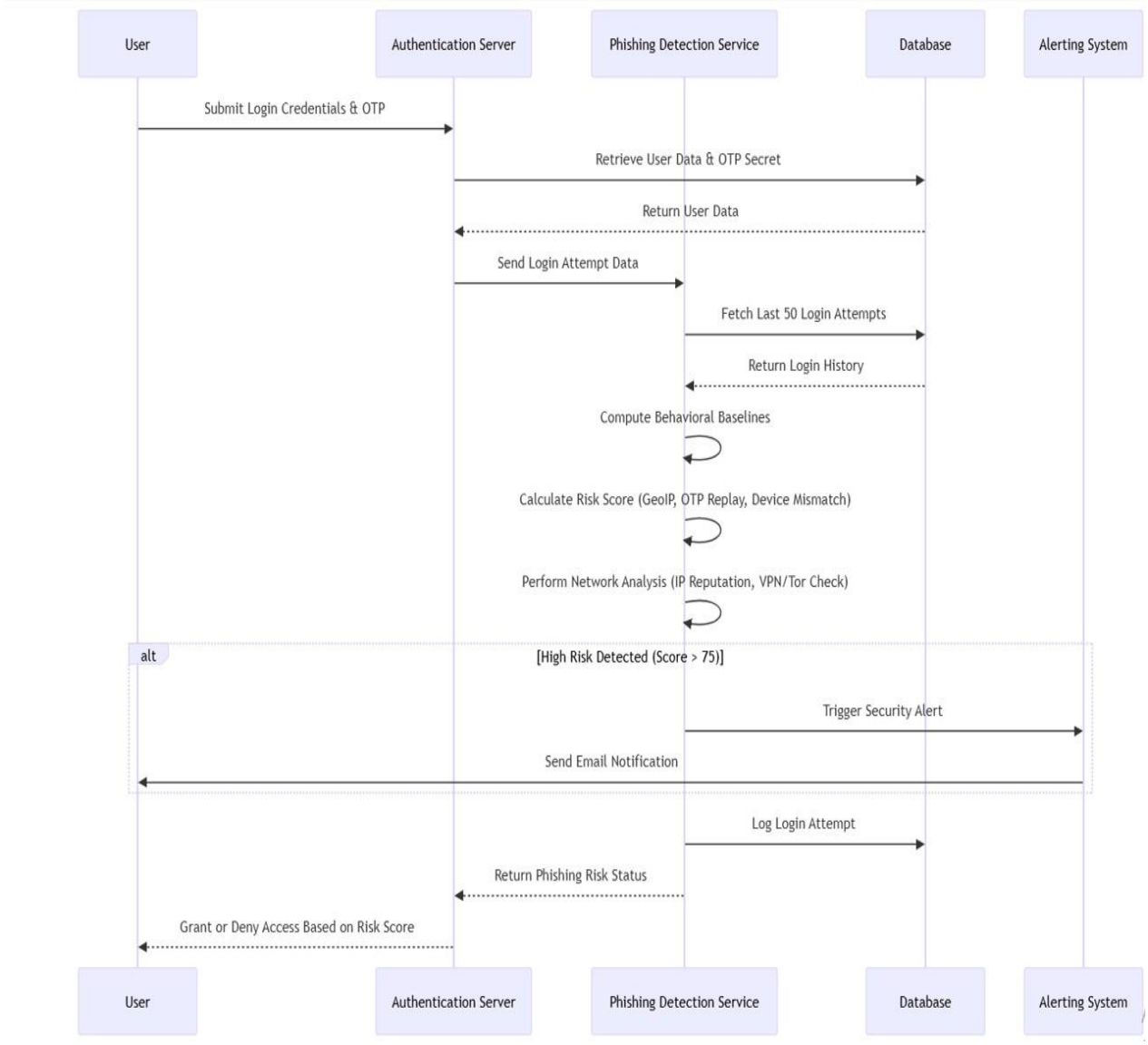


Figure 4.5: Sequence Diagram Depicting the Step-by-Step Process of the Algorithm.

4.3 Database Design

In this system, MySQL is used as the relational database management system, providing a robust and scalable solution for storing and managing the application's data. To simplify database interactions and ensure smooth integration with the application, Go Object-Relational Mapping (GORM) is employed. GORM acts as an ORM tool, abstracting the complexities of SQL queries and allowing for seamless communication between the application and the MySQL database.

This section outlines the key database structures, including the entity relationships, tables, and their attributes, necessary to support the system's authentication, device synchronization, and behavioral analysis functionalities. The database has two tables: a table for the device metadata and a table for login attempts.

4.3.1 Entity-Relationship Diagram

The Entity-Relationship diagram provides a visual representation of the key entities within the system and their relationships. It serves as a blueprint for understanding how different components of the system are interconnected, highlighting the flow of data between various entities such as users, devices, authentication events, and behavior analysis. Figure 4.6 outlines the structure of the database, defining the attributes and relationships that enable efficient storage, retrieval, and processing of information critical to the authentication and security process.

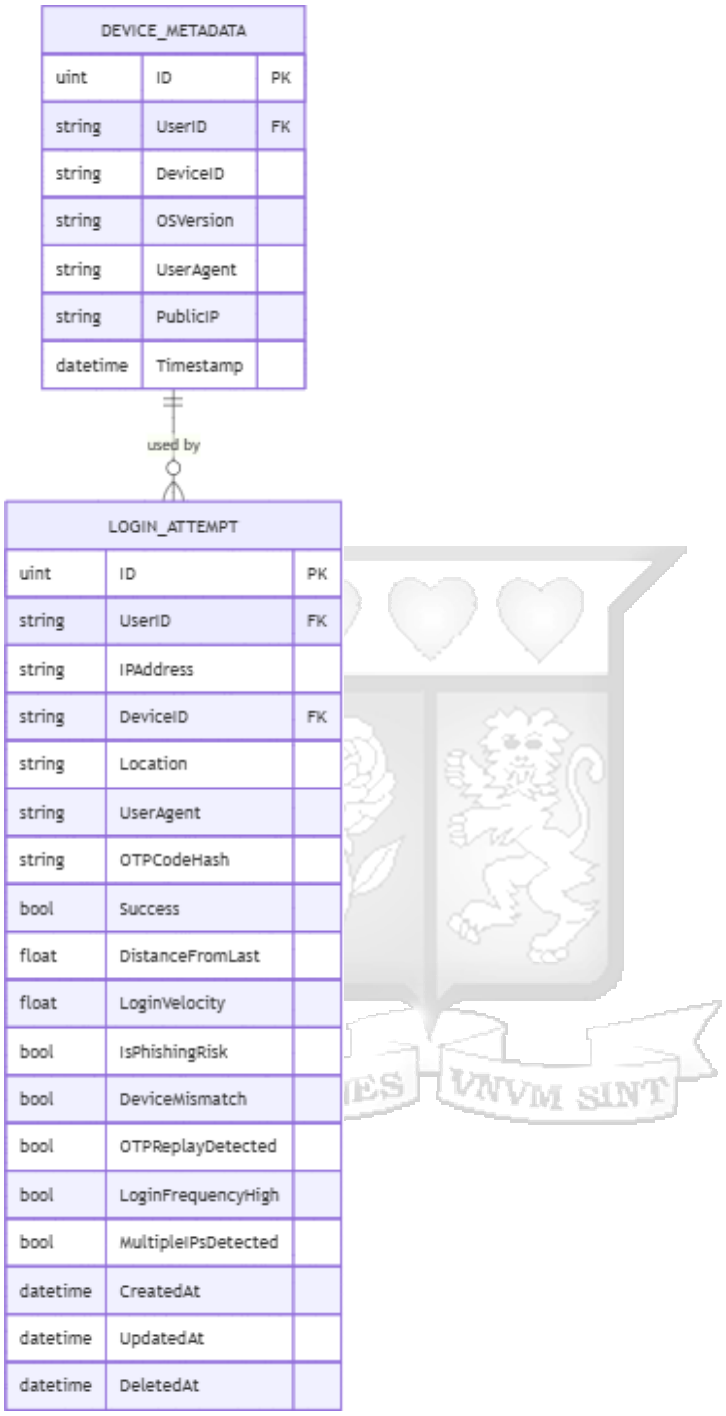


Figure 4.6: Entity-Relationship Diagram for the behavior analysis service Database.

4.3.2 Tables and Attributes

1. Device Metadata Table

Stores metadata about user devices to track login behavior and detect anomalies. Table 4.2 shows the structure of the device metadata, including the necessary fields for storing key device attributes.

Table 4.2: Schema Definition for the Device Metadata Table

Field	Type	Constraints	Description
id	uint	Primary Key	Unique identifier for the device metadata entry.
user_id	string	Foreign Key (LoginAttempt), Indexed	User associated with the device.
device_id	string	Indexed, Not Null	Unique device identifier
os_version	string		Operating system version of the device.
user_agent	string		User-Agent string from the login request.

public_ip	string		Public IP address from where the device was used.
timestamp	datetime	Auto Create	The timestamp when the device metadata was recorded.

2. Login Attempt Table

Tracks user login attempts, behavior-based risk analysis, and phishing detection flags. Table 4.3 shows the structure for capturing key information about user login attempts during the authentication process.

Table 4.3: Schema Definition for the Login Attempt Table

Field	Type	Constraints	Description
id	uint	Primary Key	Unique identifier for the login attempt.
user_id	string	Foreign Key (Device Metadata), Indexed	User attempting to log in.

ip_address	string	Indexed	IP address used for the login attempt.
device_id	string	Foreign Key (Device Metadata), Indexed	Device used for login.
location	string		Geolocation of the login attempt.
user_agent	string		Browser or app user-agent string.
otp_code_hash	string		Hashed OTP code to detect replay attacks.
success	bool		Indicates if login was successful.
distance_from_last	float		Distance (in km) from last login location.
login_velocity	float		Speed of login travel (distance/time).
is_phishing_risk	bool		Flag indicating if login is high-risk.

device_mismatch	bool		Flag if the login is from an unrecognized device.
otp_replay_detected	bool		Flag if OTP reuse is detected.
login_frequency_high	bool		Flag if login attempts exceed the expected frequency.
multiple_ips_detected	bool		Flag if multiple IPs are used within a short time.
created_at	datetime	Auto Create	Timestamp of login attempt creation.
updated_at	datetime	Auto Update	Timestamp when record was last updated.
deleted_at	datetime	Indexed	Soft delete timestamp.

4.3.3 Database Relationships

1. **One-to-Many (device metadata → login attempt)** A user can have multiple devices, and each device can be used for multiple login attempts.

2. **One-to-Many A** user can have multiple login attempts

4.4 System Modelling

System modeling is a crucial phase in the design. It provides a structured approach to represent and understand the various components and their interactions. In this section, we explore the system modeling techniques used to design the architecture and functionalities of the proposed solution.

4.4.1 Class Diagrams

The Class Diagram represents the core entities in the phishing detection and alerting system. It illustrates their attributes, responsibilities, and interactions. The system consists of the following key classes: Login Attempt, Phishing Detection, Alert, and Phishing Service. Figure 4.7 shows the interaction of the key classes.

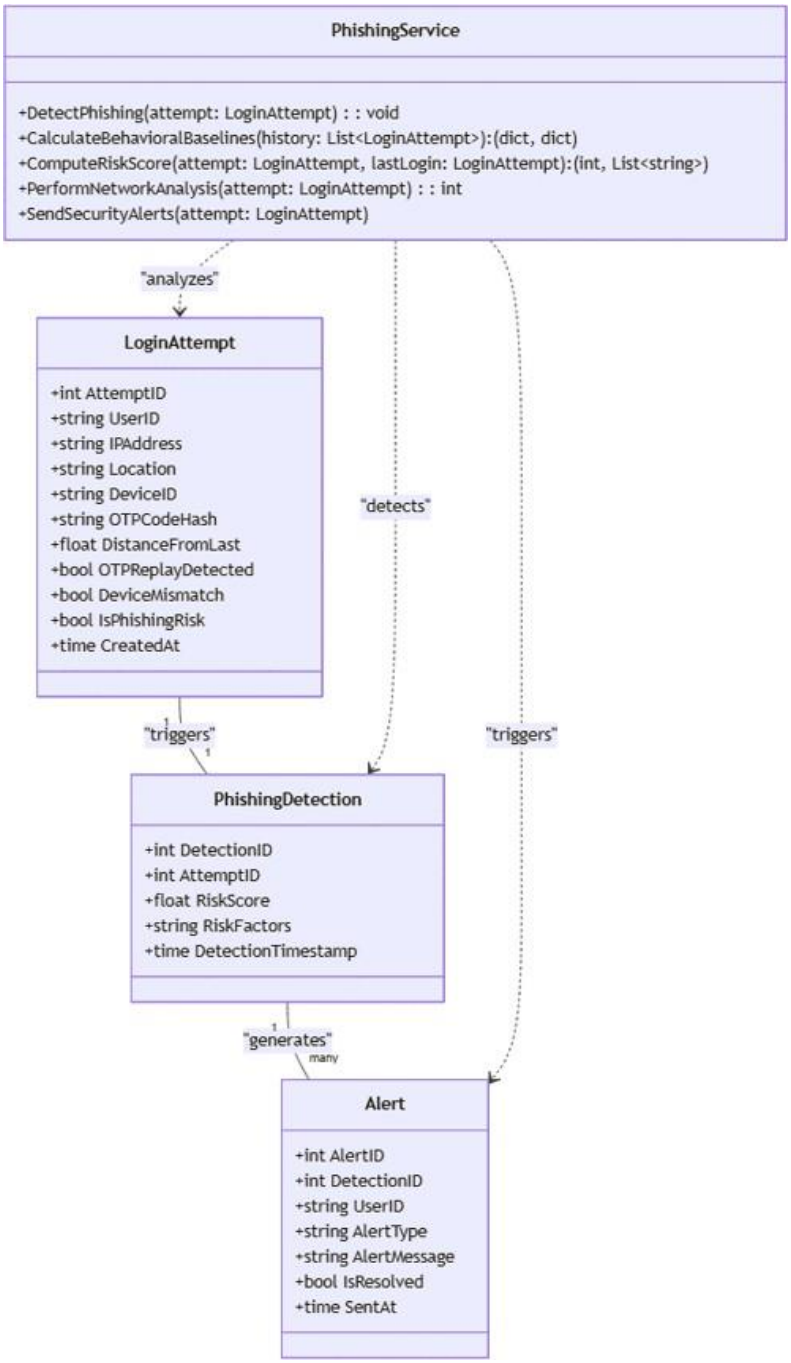


Figure 4.7: Class Diagrams Showing the Relationships between Classes

4.5 Wireframes

Wireframes serve as a visual blueprint for the user interface design, offering a clear and structured layout of the system's key screens and interactions. In this section, we present the wireframes that outline the design and flow of the user interface for the TOTP authenticator app.

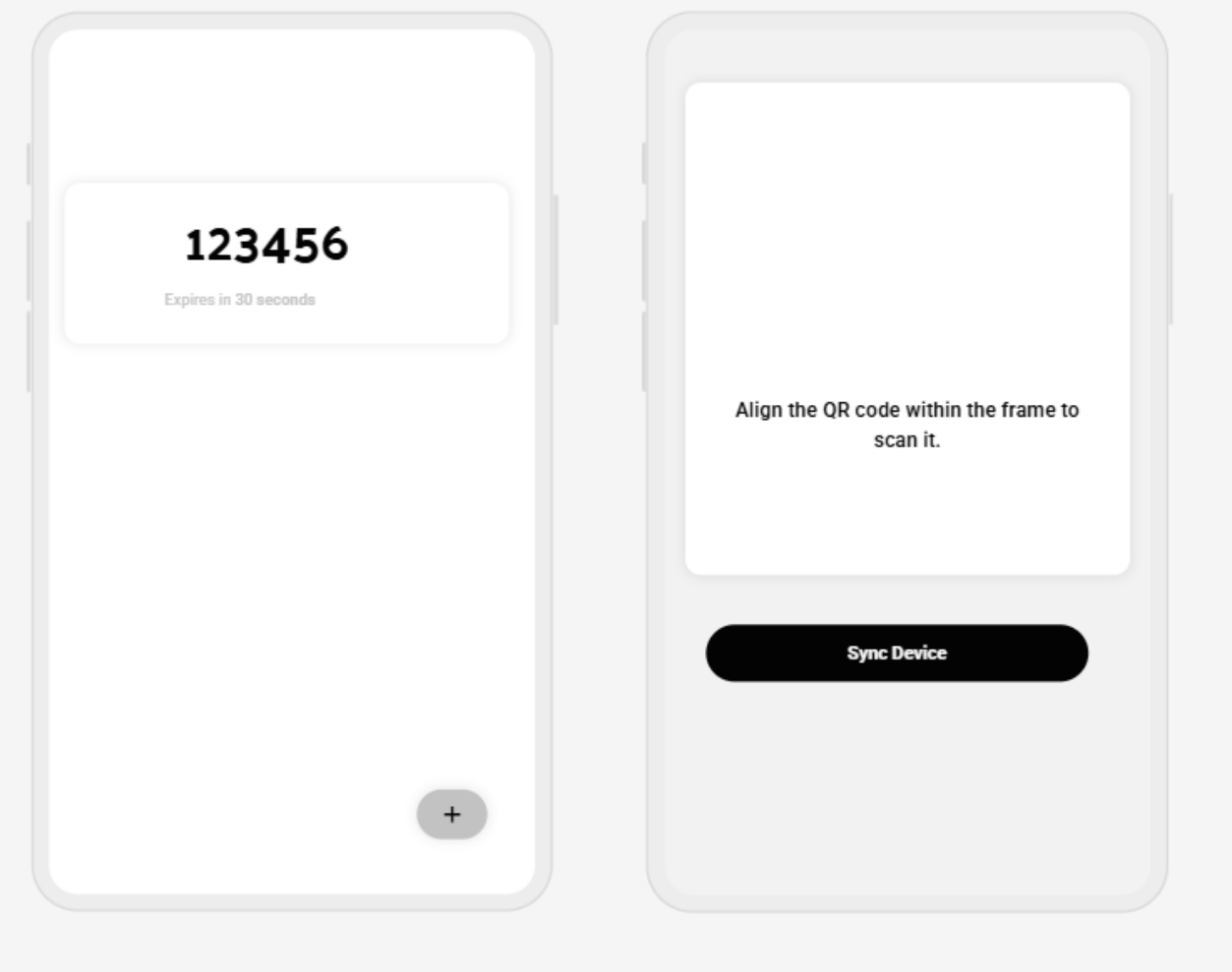


Figure 4.8: Wireframes illustrating the User Interface Design of the Proposed TOTP Application

Chapter 5: System Implementation and Testing

5.1 Introduction

This chapter describes the implementation and testing of the Android TOTP application and behavior analysis system. The implementation details focus on the functionalities of the authentication system and its security mechanisms, while the testing phase ensures that the system meets security, performance, and usability requirements.

5.2 System Implementation

5.2.1 Android TOTP Application

The mobile application provides a seamless and secure method for users to synchronize their accounts using a QR code-based setup. Before scanning a QR code, the app requests camera permissions, as access to the camera is necessary to capture and decode the QR code containing the shared secret key and authentication details. Following Android's runtime permissions model, users must explicitly grant permission before the camera can be used. If permission is denied, they will be unable to scan the QR code and must manually enter the secret key. Once permission is granted, users can scan the QR code displayed by the authentication server. This QR code typically encodes essential authentication details, such as the issuer, account name, and shared secret key, in a standardized format like the otpauth://totp/ scheme. The app uses a QR code scanning library Google's ML Kit, to extract these details and process them securely.

To prevent unauthorized access, the shared secret key is immediately encrypted before storage. This is a crucial security measure because the secret key is the foundation of TOTP generation. If exposed, an attacker could generate valid OTPs and bypass authentication. The app employs Android's Keystore System, which provides hardware-backed secure storage for cryptographic keys. The encryption process generates a strong AES-256 key and stores it in the Android Keystore, ensuring that it cannot be extracted from the device. The shared secret key is then encrypted using AES-GCM mode, which provides both confidentiality and integrity protection, before being securely stored in an encrypted Shared Preferences. This approach ensures that even if an attacker gains access to the app's local storage, they cannot retrieve the actual secret key without access to the cryptographic key stored in the Keystore. Once securely stored, the secret

key is used to generate TOTPs dynamically. Following the RFC 6238 standard, the app retrieves the encrypted secret key and decrypts it using the Android Keystore's AES key. It then calculates a time step by dividing the current UNIX timestamp by the predefined TOTP time interval, typically 30 seconds. Using the HMAC-SHA1 algorithm, the app generates a unique OTP by hashing the secret key and time step. The resulting value is then converted into a six-digit numeric OTP that updates every 30 seconds. The OTP is displayed in the app's user interface, allowing users to enter it into the authentication system. The interface includes additional usability features such as a countdown timer indicating the time remaining before the OTP refreshes, a smooth animation to signal when the new OTP is generated, and a copy-to-clipboard button for easy input. When users enter the OTP on the authentication server, the server independently calculates the expected OTP using the same algorithm. If the entered OTP matches the expected value, within an allowable time drift, authentication is granted.

On opening the app, the user gets a page to view their OTP if present and a button to add a token as shown in figure 5.1

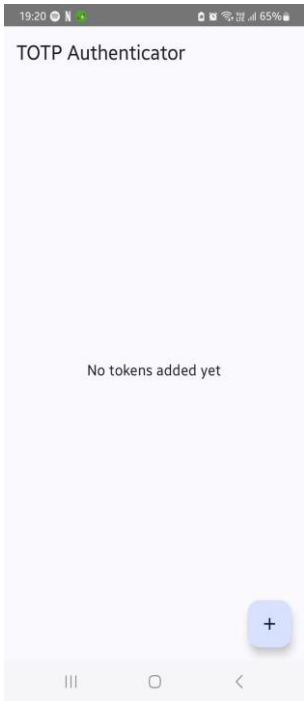


Figure 5.1: Screenshot of the Application's Home Page.

The app utilizes the camera to scan the QR code, prompting a request for camera permissions when the user first interacts with the application or if the required permissions have not been granted previously. This ensures that the app has the necessary access to the camera for QR code scanning functionality. Figure 5.2 illustrates the permission request process, highlighting the steps taken to ensure the app functions smoothly while maintaining user consent and privacy.

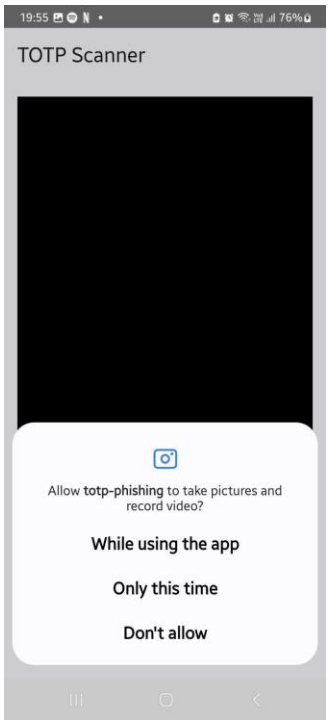


Figure 5.2: Screenshot of the Application's Permissions screen

After granting camera permissions, the app enables the user to scan the QR code. **(**Note: In a production environment, the contents encoded in the QR code will not be disclosed to the user for security reasons; however, in this case, the data is shown for transparency and to demonstrate the information captured during the scanning process.) Figure 5.3 illustrates this process, providing visibility into how the app reads and processes the QR code data

Figure 5.3: Screenshot of the Application's Scan page

After successfully scanning the QR code, the device running the TOTP app is synchronized with the behavior analysis system. During this process, the app sends key device and operating system fingerprints, such as the device ID, OS version, and other metadata, to the server for registration and security analysis. Figure 5.4 illustrates this synchronization process, showing how the app captures and transmits device information for enhanced security and behavior-based monitoring.

Figure 5.4: Screenshot Showing Application Data Synchronization with the Behavioral Analysis Service.

Once the synchronization is complete, the app uses the secret key scanned from the QR code and the current timestamp to generate TOTPs, which expire every 30 seconds. Figure 5.5 illustrates this process, demonstrating how the app generates secure, time-sensitive TOTPs that are essential for user authentication.

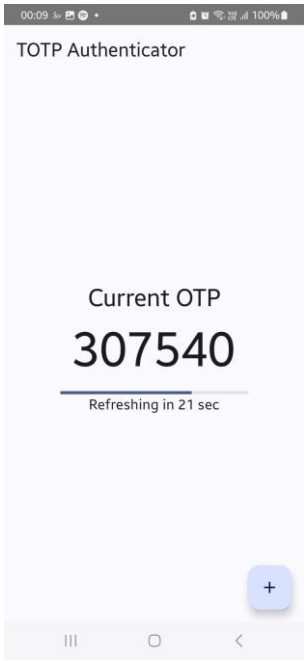
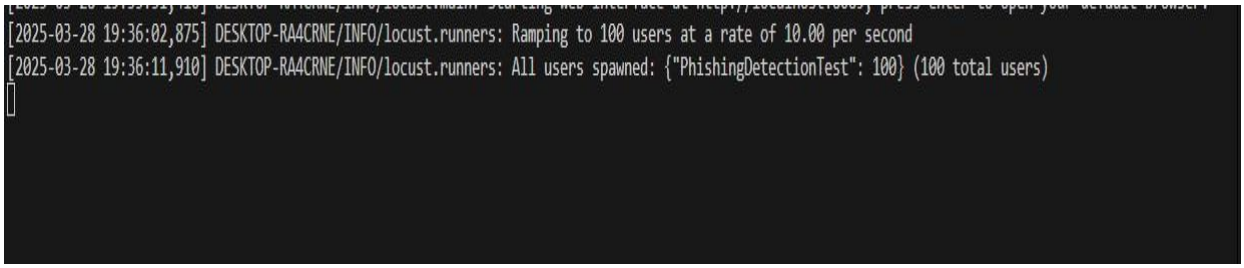


Figure 5.5: Screenshot Displaying the Application Generating a One-Time Password

5.2.2 Behavior analysis System

The behavior analysis system tracks login attempts based on multiple parameters which are analyzed using a risk scoring model. Deviations from a user's typical login pattern result in increased risk scores.

Using Locust, I generated user data to simulate load testing. Figure 5.6 shows screenshots of the data generated, where 100 users were spawned at a rate of 10 users per second. This allows us to test the system's performance and scalability under realistic conditions, ensuring it can handle multiple simultaneous authentication requests efficiently.



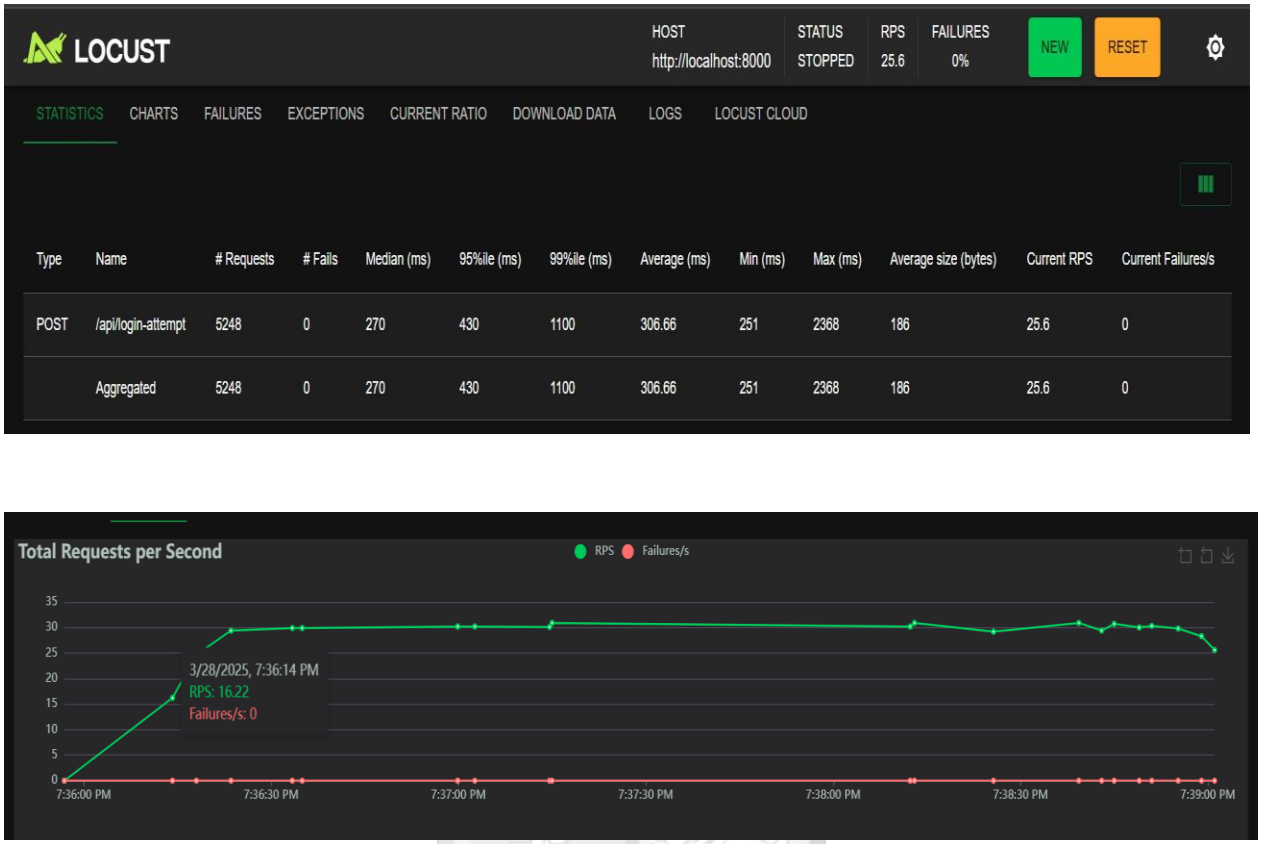


Figure 5.6: Locust Generating Test Data to Simulate User Activity.

All the generated data was stored in a MySQL database for further analysis and testing. Figure 5.7 shows a sample of the data that was generated, providing insight into the structure and content of the records stored.

id	user_id	ip_address	device_id	location	user_agent	otp_code_hash
6417	user_66	42.60.102.124	device_42	Singapore	Mozilla/5.0 (Windows NT 10.0; Win64; x64) App...	hash_1890
6416	user_95	108.85.64.251	device_15	United States	Mozilla/5.0 (Linux; Android 11; SM-G991B) Appl...	hash_6836
6415	user_4	250.191.38.106	device_10	Unknown	Mozilla/5.0 (Linux; Android 11; SM-G991B) Appl...	hash_2827
6414	user_69	176.85.22.79	device_6	Spain	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7...	hash_9332
6413	user_97	226.135.198.154	device_37	Unknown	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7...	hash_5538
6412	user_27	195.168.238.130	device_6	Slovakia	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7...	hash_1265
6411	user_74	111.222.87.172	device_31	China	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7...	hash_7612
6410	user_11	10.242.6.112	device_6	Unknown	Mozilla/5.0 (iPhone; CPU iPhone OS 14_6 like M...	hash_5727
6409	user_31	179.243.84.78	device_17	Brazil	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7...	hash_8114
6408	user_88	37.89.107.247	device_18	Germany	Mozilla/5.0 (Windows NT 10.0; Win64; x64) App...	hash_7501
6407	user_22	137.139.39.231	device_44	United States	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7...	hash_4059
6406	user_63	121.121.0.1	device_43	Malaysia	Mozilla/5.0 (iPhone; CPU iPhone OS 14_6 like M...	hash_2491
6405	user_43	71.119.134.196	device_21	United States	Mozilla/5.0 (iPhone; CPU iPhone OS 14_6 like M...	hash_2780
6404	user_46	107.50.230.175	device_43	United States	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7...	hash_8332

Figure 5.7: Example of Synthetic User Data Produced by Locust.

At the database level, we have the generated user data captured. Using this data, we can perform behavior analysis. The behavior analysis service takes all this data and creates a baseline for all the user ids. After that the overall risk score is determined by evaluating a combination of behavioral risk factors and network-based security heuristics to detect potential account compromise attempts. Each risk factor contributes a specific weight to the total risk score, and if the score surpasses a predefined threshold 75, an alert is triggered to notify the user. The risk scores are calculated based on the risk calculation table provided in chapter 4, table 4.1

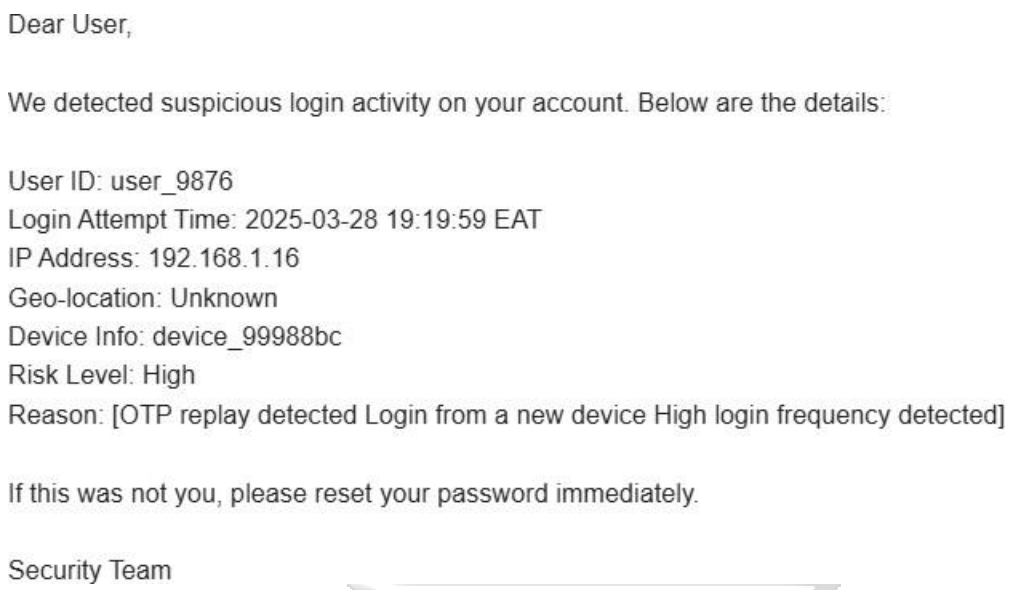
Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

user_id	ip_address	device_id	location	user_agent	otp_code_hash	risk_score
user_18	7.242.174.92	device_39	United States	Mozilla/5.0 (Linux; Android 11; SM-G991B) Appl...	hash_6453	85
user_18	248.228.72.220	device_4	Unknown	Mozilla/5.0 (Windows NT 10.0; Win64; x64) App...	hash_9965	85
user_4	75.241.127.80	device_12	United States	Mozilla/5.0 (Windows NT 10.0; Win64; x64) App...	hash_9221	85
user_11	44.32.238.159	device_50	Canada	Mozilla/5.0 (Linux; Android 11; SM-G991B) Appl...	hash_2022	85
user_54	49.18.53.216	device_20	South Korea	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7...	hash_2228	85
user_6	129.234.226.60	device_29	United Kingdom	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7...	hash_5212	85
user_18	180.19.88.234	device_8	Japan	Mozilla/5.0 (iPhone; CPU iPhone OS 14_6 like M...	hash_6381	60
user_2	224.104.205.190	device_7	Unknown	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7...	hash_1041	60
user_53	253.3.97.109	device_5	Unknown	Mozilla/5.0 (iPhone; CPU iPhone OS 14_6 like M...	hash_3208	40
user_46	25.50.139.43	device_43	United Kingdom	Mozilla/5.0 (iPhone; CPU iPhone OS 14_6 like M...	hash_9785	40
user_57	83.10.149.150	device_48	Poland	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7...	hash_6659	40
user_5	240.165.213.126	device_30	Unknown	Mozilla/5.0 (iPhone; CPU iPhone OS 14_6 like M...	hash_1010	40
user_24	134.13.225.247	device_6	United States	Mozilla/5.0 (Linux; Android 11; SM-G991B) Appl...	hash_2406	40
user_67	67.127.85.193	device_13	United States	Mozilla/5.0 (Linux; Android 11; SM-G991B) Appl...	hash_6234	40
user_81	17.194.57.199	device_33	United States	Mozilla/5.0 (iPhone; CPU iPhone OS 14_6 like M...	hash_2945	40

Figure 5.8: Risk Scores Calculated from Locust-Generated Data and Poll Results

When the risk core is 75 or above an email is triggered to the user. For example, user_9876 has a risk score of 115, therefore an alert is triggered to their email address. Figure 5.9 shows the risk score calculated for user_9876 and a sample email sent to alert the user of a phishing risk

```
2025/03/28 19:20:07 [RISK SCORE] user_9876: 115
Email sent successfully to: lokoc2623@gmail.com
```



Dear User,

We detected suspicious login activity on your account. Below are the details:

User ID: user_9876
Login Attempt Time: 2025-03-28 19:19:59 EAT
IP Address: 192.168.1.16
Geo-location: Unknown
Device Info: device_99988bc
Risk Level: High
Reason: [OTP replay detected Login from a new device High login frequency detected]

If this was not you, please reset your password immediately.

Security Team

Figure 5.9: Email Alert Notification for User Exceeding Risk Score Threshold

5.3 System Validation and Testing

To ensure the reliability and security of the TOTP authentication system and the phishing detection engine, I conducted validation and testing procedures. These tests included functional testing, security testing, and performance evaluation.

5.3.1 Functional Testing

To ensure that all components of the TOTP authentication system operated as intended, I conducted a functional test. The primary objective was to validate core functionalities. To achieve comprehensive validation, I designed test cases based on various authentication scenarios. Table 5.1 shows the test cases for the android TOTP applications and Table 5.2 shows the test cases for the behavior analysis server

Table 5.1: Test Case Execution Report for the Android TOTP Application

Test Case ID	Test Scenario	Test Steps	Expected Outcome	Actual Outcome	Pass/Fail
TC-001	QR Code Scanning	1. Scan the QR code from the authentication server. 2. Extract and store the secret key.	The secret key is extracted and encrypted	The key is successfully extracted by the app and encrypted.	Pass
TC-002	Synchronization with Behavioral Analysis Service	1. Scan QR code 2. Sync with backend analysis service.	Device metadata is captured and transmitted to the behaviour analysis system via an API	Device metadata was successfully stored in MYSQL.	Pass
TC-003	TOTP Code Generation	1. Open the mobile app. 2. Generate OTP.	The OTP is generated and displayed, refreshing every 30 seconds.	The OTP is correctly generated and refreshed every 30 seconds.	Pass

Table 5.2: Test Case Execution Report for the Behavior Analysis Service

Test Case ID	**Test Scenario**	**Preconditions**	**Test Steps**	**Expected Result**
TC-001	Successful Login from a Familiar Device and Location	User has logged in from the same device and location before.	1. User enters valid credentials. 2. System verifies login history and device fingerprint. 3. No anomalies are detected. 4. User is authenticated.	Login is granted.
TC-002	Suspicious Login from a New Location	User has never logged in from this IP address or country before.	1. User enters valid credentials. 2. System detects unfamiliar location.	System flags login is from a different location and IP address
TC-003	Concurrent Logins from Different Locations	User is already logged in from another location.	1. User attempts login from a different geographic location.	System flags login attempt

			2. System detects multiple active sessions. 3. Alert is generated, and session is blocked or challenged.	
TC-004	Login Attempt from an Unrecognized Device	User has never logged in from this device before.	1. User enters valid credentials. 2. System checks device fingerprint. 3. Alert is triggered due to an unrecognized device. 4. User is prompted for additional verification.	System flags login attempt.
TC-005	Login from a Known Device but Different IP Address	User has previously logged in from the same device but with a different IP.	1. User attempts login. 2. System recognizes device	System flags unusual IP but permits login.

			but detects an unusual IP. 3. System applies mild-risk warning but allows login with a notification.	
TC-006	Impossible Travel Login Detection	User logs in from Location A and attempts another login from distant Location B in an unrealistic timeframe.	1. User logs in from Location A. 2. Shortly after, another login is attempted from Location B. 3. System calculates travel time and distance. 4. Login attempt is flagged.	System detects impossible travel and sends alert.
TC-007	Frequent Logins in a Short Period	User logs in and out repeatedly within seconds.	1. User logs in multiple times in a short period. 2. System detects abnormal login frequency.	System flags rapid login attempts

			3. Security warning is triggered.	
TC-008	Security Event Logging	System must log all security-related events.	1. User logs in from a new device. 2. System logs the event in an audit trail. 3. Admin can review logs.	All security events are logged for auditing.

5.3.2 Security Testing

I conducted a security test using Evilginx. This test aimed to evaluate the effectiveness of the phishing detection engine in identifying phishing threats. To simulate a real-world phishing attack, I targeted GitHub, a platform that utilizes TOTP codes as a second factor of authentication. By deploying a phishing proxy, I intercepted user authentication attempts to assessed how well the detection system could recognize and respond to such attacks. The test was carried out using the following steps:

I installed Evilginx and configured it with a phishing profile specifically targeting the GitHub authentication system, which utilizes two-factor authentication with TOTP for verification. By leveraging Evilginx, I was able to intercept and replicate the authentication flow. Figure 5.10 illustrates this setup, showcasing how the phishing profile was configured to simulate and monitor the 2FA process on GitHub

Figure 5.10: Evilginx Launched to Simulate an Attack.

Evilginx enables attackers to create highly convincing fake login pages by dynamically cloning the content of legitimate websites through proxying. For this setup, I registered a phishing domain, admin.secure.co.ke and linked it to an EC2 instance running Evilginx. I then configured DNS records to point the phishing domain's A record to the Evilginx server's IP address. Evilginx was installed and initialized to act as a reverse proxy for GitHub's authentication flow, allowing it to intercept and replicate the login process.

To increase the legitimacy of the phishing page and evade detection, I obtained and configured SSL/TLS certificates for the phishing domain, enabling HTTPS encryption. Modern browsers often flag non-HTTPS websites as insecure, making SSL certificates essential for convincing targets. I used Let's Encrypt to generate valid SSL certificates for the phishing domain and configured Evilginx to automatically fetch and renew these certificates using the Let's Encrypt

ACME protocol. Once SSL was enabled, the phishing site displayed a padlock icon in the browser, making it appear secure and trustworthy to the victim. Figure 5.11 shows the GitHub hostname enabled to use the domain I registered.

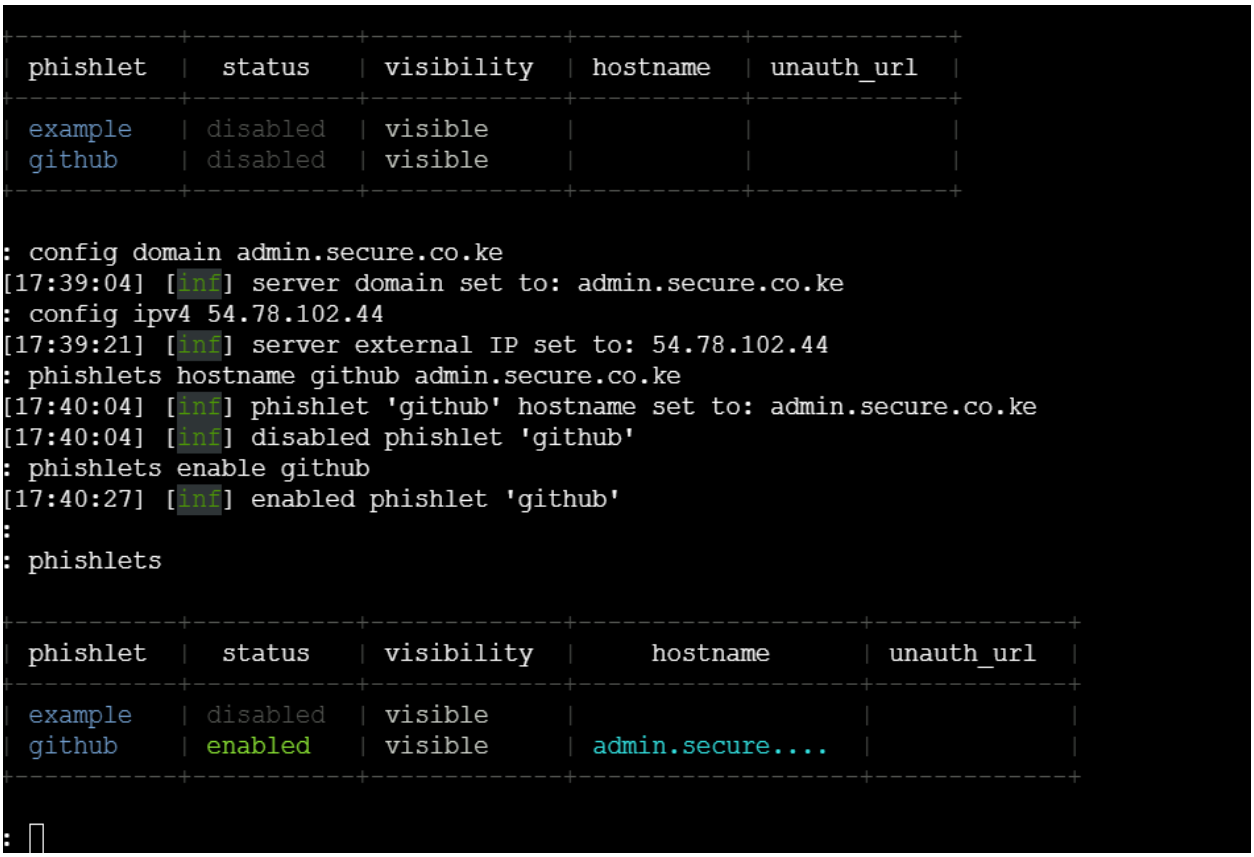


Figure 5.11: Evilginx Configuration Targeting GitHub for Use Case Simulation

Evilginx uses phishlets. Phishlets are YAML-based configuration files that define how a target website's authentication flow should be proxied and manipulated. I deployed a GitHub phishlet to relay login requests while silently extracting session cookies. This testing utilized a prebuilt GitHub phishlet that was customized to match the latest GitHub authentication flow. The Phishlet was enabled to start listening for connections on the phishing domain. Figure 5.12 shows a sample GitHub phishlet

```
author: '@audibleblink'
min_ver: '2.3.0'
proxy_hosts:
  - {phish_sub: '', orig_sub: '', domain: 'github.com', session: true, is_landing: true}
  - {phish_sub: 'api', orig_sub: 'api', domain: 'github.com'}
  - {phish_sub: 'github', orig_sub: 'github', domain: 'githubassets.com'}

sub_filters:
  - {triggers_on: 'github.com', orig_sub: '', domain: 'github.com', search: 'integrity="(.*?)"', replace: '', mimes: ['text/html']}

auth_tokens:
  - domain: '.github.com'
    keys: ['logged_in', 'dotcom_user']
  - domain: 'github.com'
    keys: ['user_session', '_gh_sess']

credentials:
  username:
    key: 'login'
    search: '(.*)'
    type: 'post'
  password:
    key: 'password'
    search: '(.*)'
    type: 'post'

login:
  domain: 'github.com'
  path: '/login'
```

5.12: GitHub phishlet

To gain access to a victim, I created a lure using evilginx, which I used in the phishing email. Figure 5.13 shows this

```
: lures

+----+----------+----------+------+------------+--------------+--------+-----+
| id | phishlet | hostname | path | redirector | redirect_url | paused | og  |
+----+----------+----------+------+------------+--------------+--------+-----+
+----+----------+----------+------+------------+--------------+--------+-----+

: lures create github
[18:45:31] [inf] created lure with ID: 0
: lures

+----+----------+----------+-----------+------------+--------------+--------+-------+
| id | phishlet | hostname |   path    | redirector | redirect_url | paused |  og   |
+----+----------+----------+-----------+------------+--------------+--------+-------+
| 0  | github   |          | /BBrekQBn |            |              |        | ----  |
+----+----------+----------+-----------+------------+--------------+--------+-------+

: lures get-url 0

https://admin.secure.co.ke/BBrekQBn
```

Figure 5.13: Generating a phishing Lure

Using the link created by evilginx, I crafted an email to my target creating some sense of urgency to make an update prompting them to click on the link created. The email is shown in figure 5.14

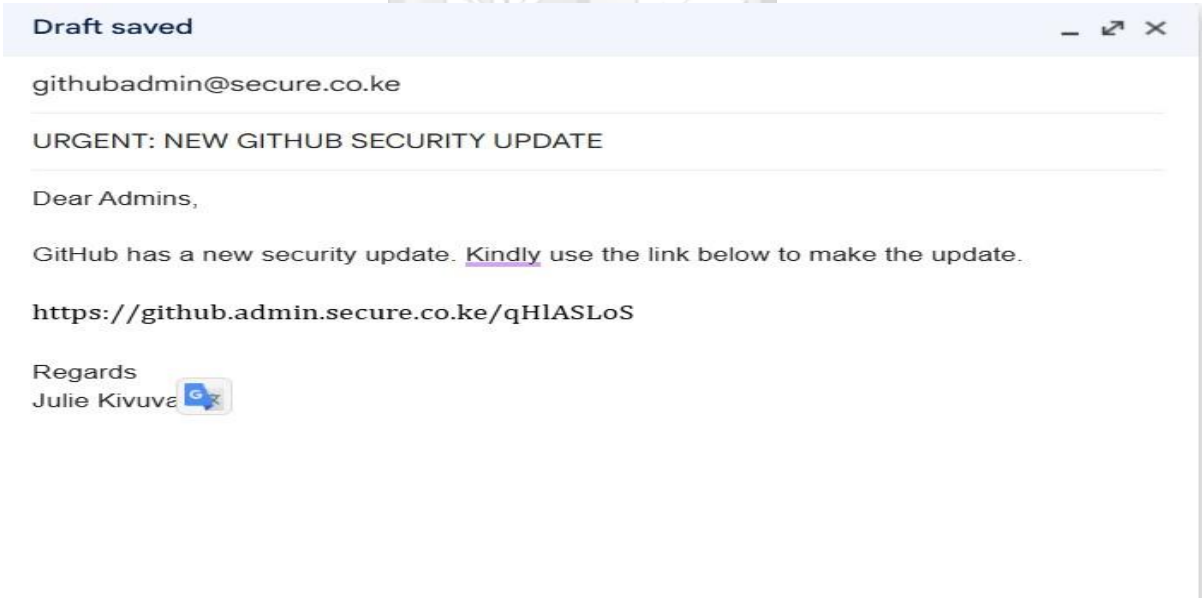


Figure 5.14: Phishing Email Constructed Using the Generated Lure

When the victim clicked on the link, Evilginx's proxy captured all incoming authentication requests, forwarding them to the real GitHub server while logging user credentials and session cookies. Figure 5.15 shows how evilginx captures the requests and logs the credentials the user inputs. It further captures the IP address and the session cookies of the victim

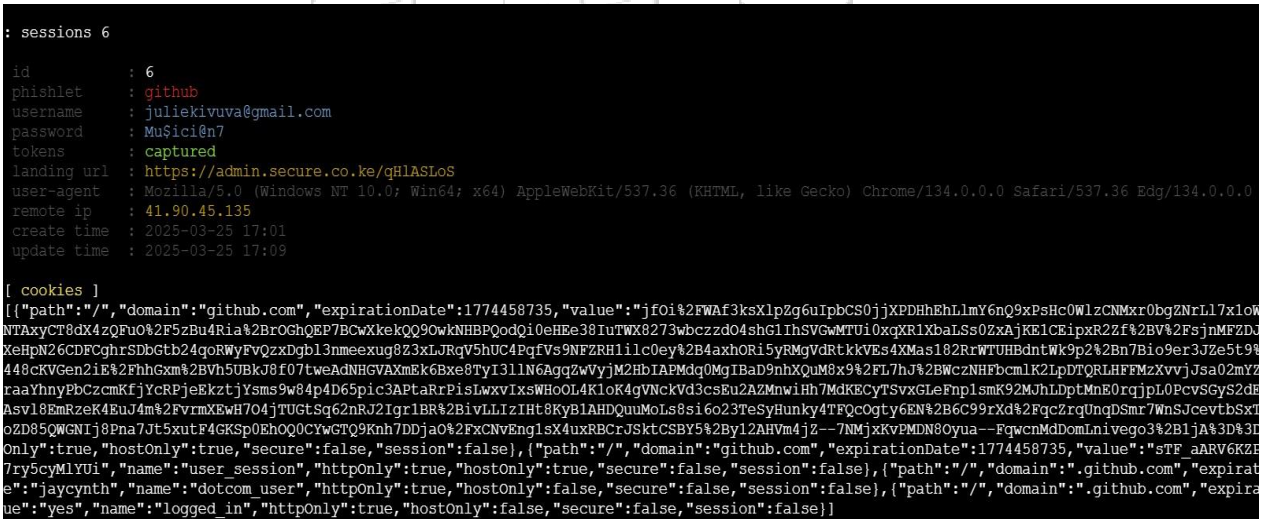


Figure 5.15: Captured GitHub Authentication Credentials and Session Cookies

Using Evilginx, I intercepted and captured user login credentials without triggering security alerts on the victim's side. Evilginx operates as a transparent reverse proxy, meaning that the victim's login request is relayed to the legitimate GitHub authentication server while simultaneously extracting authentication data, including Username and Password which are captured from the login form submission and session Cookies which are stolen to bypass Multi-Factor Authentication and enable session hijacking. To validate that the captured credentials and session tokens were legitimate, I used a browser extension called Cookie Editor to manually inject the stolen cookies and gain unauthorized access to the victim's GitHub account without requiring their password or MFA code. I opened an incognito browser window and navigated to github.com. Using the Cookie Editor extension, I replaced the existing cookies with the stolen session cookies. On refreshing the page, and I was able to successfully log into the victim's account without re-entering credentials. This is shown in figure 5.16

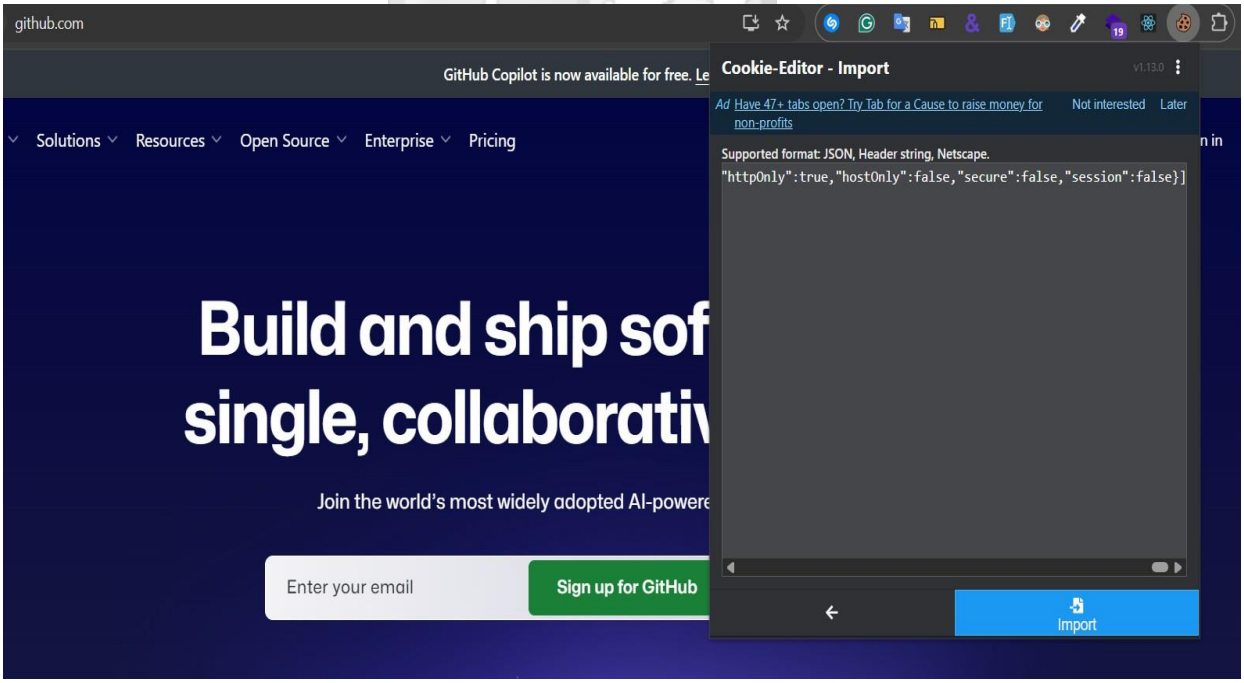


Figure 5.16: Using the acquired cookies to gain access to a victim's account using cookie editor

extension

Upon editing the session cookies, I was able to bypass all authentication and go directly to user account as shown in figure 5.17

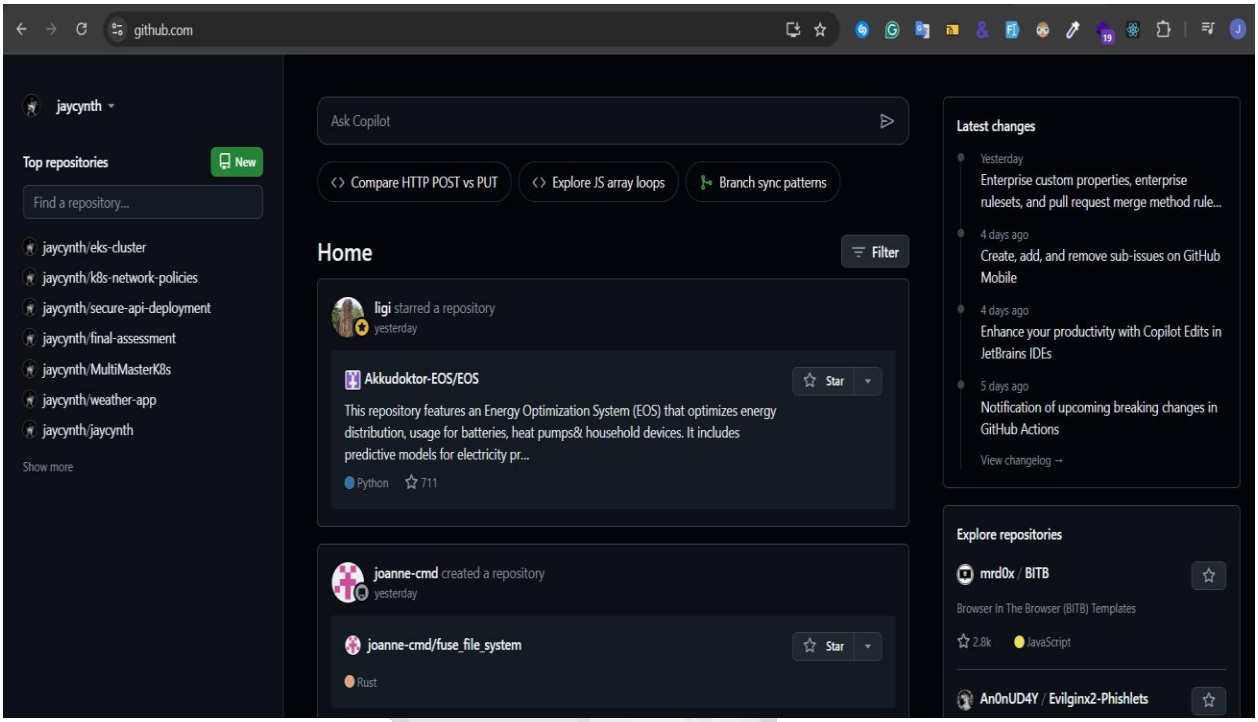


Figure 5.17: Gain access to victim's account

The phishing detection engine analyzed login behavior; IP addresses, and device fingerprints. It detected anomalies with the login location and device and the login attempt was flagged as shown in figure 5.18

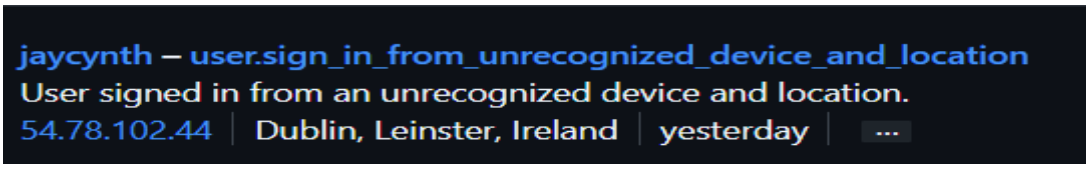


Figure 5.18: Phishing is detected

Upon detection, the system triggered real-time alerts via email warning of a possible phishing attempt as shown in figure 5.19

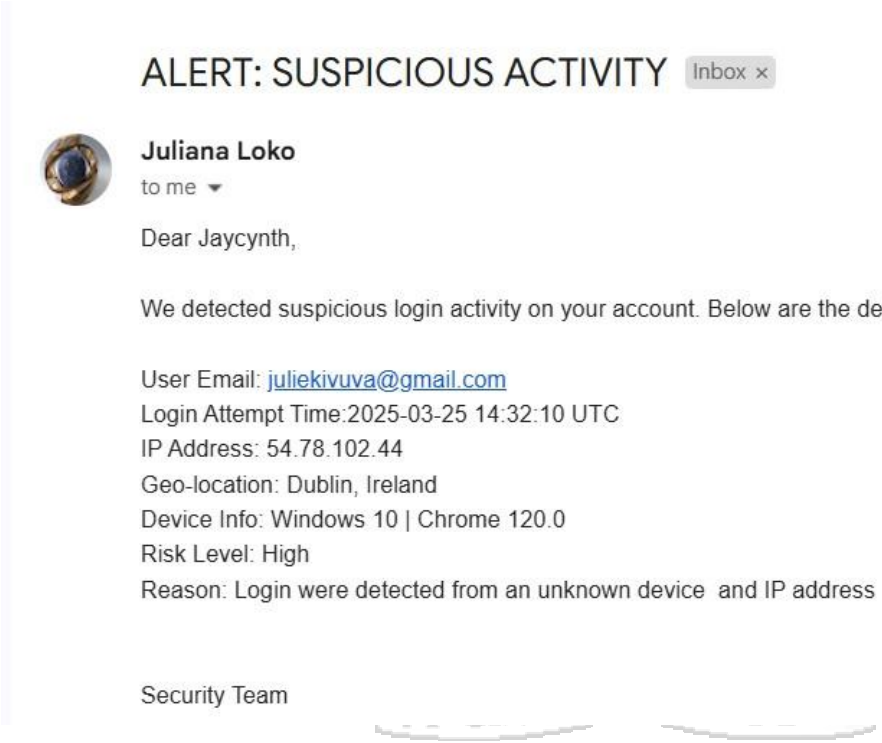


Figure 5.19: An alert is sent to user via email

This test demonstrated that the phishing detection engine effectively identified unauthorized login attempts originating from Evilginx.

5.3.3 Performance Testing

To validate the system's performance and scalability, I conducted performance testing, which included stress testing and load testing. The goal was to measure response times, ensure efficient processing of authentication requests, and confirm that the system could sustain high traffic

without degradation. For this test case, I used the Locust tool.

5.3.3.1 Response Time Analysis under Different Loads

To assess the system's performance under various levels of user load, response times were measured for three distinct scenarios: 100, 10,000, and 100,000 users. Table 5.3 presents the corresponding response times for each scenario, allowing for a clear comparison of how the system handles increasing traffic. Figure 5.20 illustrates the response time when 100 users are simulated at a rate of 10 users per second. Figure 5.21 shows the system's performance with 10,000 users at a rate of 100 users per second. Figure 5.22 presents the response times when 100,000 users are simulated at a rate of 1,000 users per second. These figures and the accompanying data in Table 5.3 provide valuable insights into the scalability of the system and its ability to maintain efficient response times as the user load increases.

Table 5.3: Response time analysis under different loads using Locust

Load Size	Average (ms)	Min(ms)	Max(ms)	Failure rate (%)
100	786.6	747	1066	0
10,000	44,190.87	870	84599	14%
100,000	49,804.58	1536	88010	32.5%

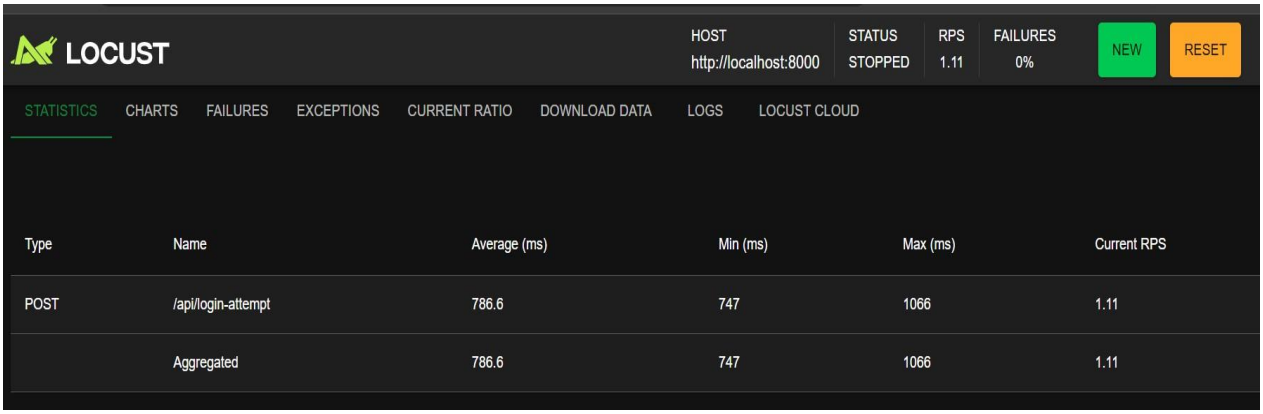


Figure 5.20: Testing with 100 users at a rate of 10 per second

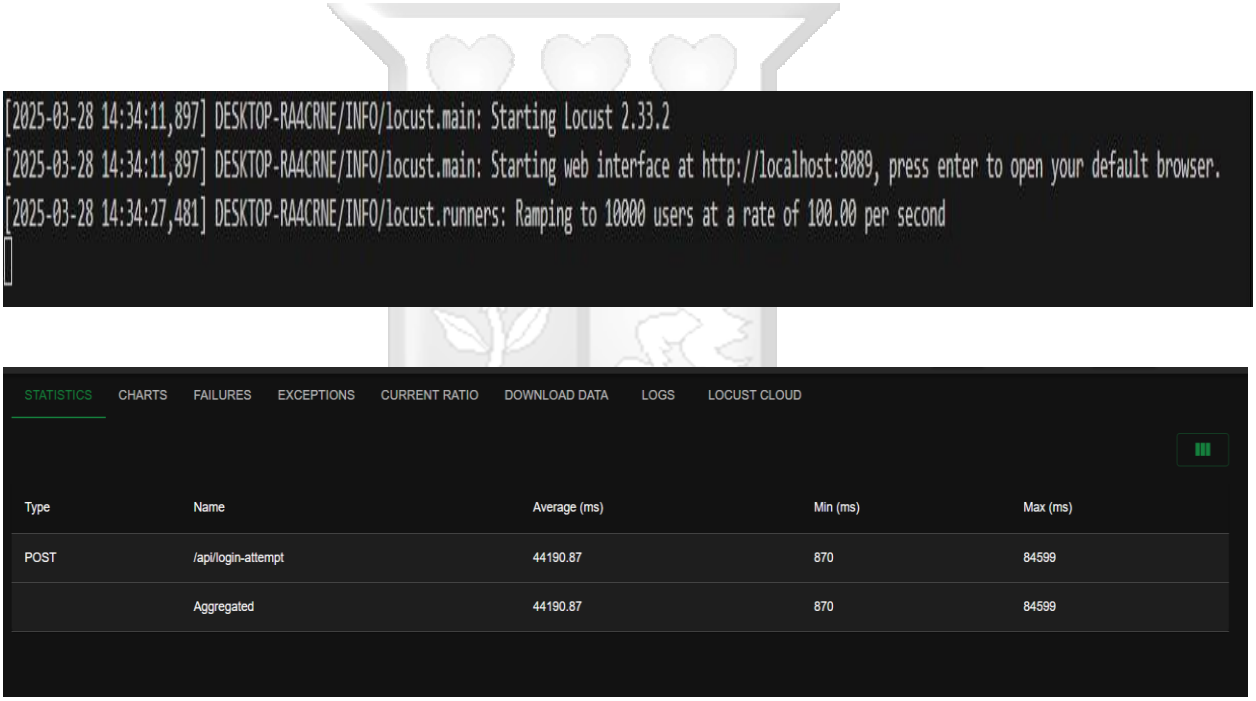


Figure 5.21: Testing with 10000 users at a rate of 100 per second

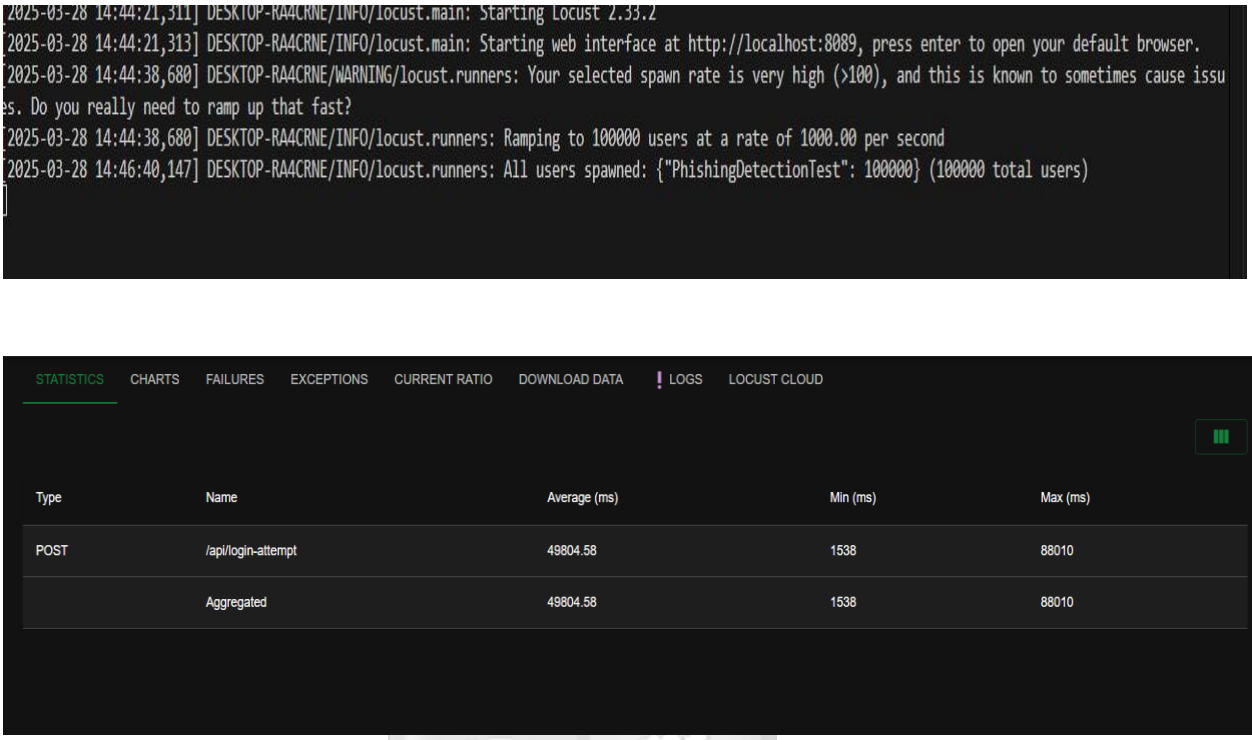


Figure 5.22: Testing with 100000 users at a rate of 1000 per second

As the system experiences increasing load, performance degradation becomes evident due to resource constraints. At 100 requests, the system operates efficiently, with an average response time of 786.6 ms and no failures, indicating that under low load, the available CPU, memory, and network resources are sufficient to handle requests smoothly.

However, as the load increases to 10,000 requests, the average response time rises dramatically to 44.2 seconds, and the failure rate climbs to 14%. This suggests that the system is reaching its resource limits, likely due to CPU exhaustion, memory bottlenecks, or database latency, preventing it from handling concurrent requests effectively. Additionally, the maximum response time spikes to nearly 85 seconds, highlighting potential queuing delays or thread contention issues.

At 100,000 requests, the average response time exceeded 49.8 seconds and a failure rate of

32.5%, indicating severe system strain. This high failure rate suggests that the system was unable to process incoming requests in a timely, likely due to excessive database queries. The minimum response time also increases, implying that even the fastest responses are being affected by the overall system slowdown.

5.3.3.2 Stress Testing: High Volume Login Simulations

The stress test revealed critical system constraints that impacted its performance under extreme load conditions. The system initially handled 100 requests efficiently, maintaining minimal latency due to sufficient available resources. However, as the load increased to 10,000 and 100,000 requests, CPU exhaustion, memory saturation, and database latency became major bottlenecks, leading to performance degradation. Under high concurrency, the system likely experienced thread contention and excessive context switching, reducing processing efficiency. The sharp increase in response time suggests that CPU resources were overwhelmed, struggling to handle incoming requests in parallel.

The failure rates at 10,000 (14%) and 100,000 (32.5%) requests indicate that the database struggled to process queries efficiently under heavy load. I determined the cause of this to be slow read/write operations and excessive connection pooling, leading to query delays and timeouts as seen in the Figure 5.23 below.

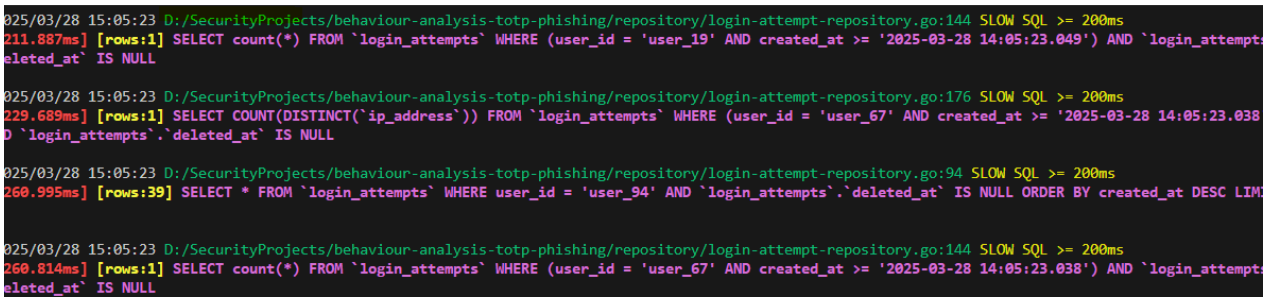


Figure 5.23: Identification of Database Bottlenecks Leading to Slow Query Execution

Despite these constraints, the system successfully processed a high volume of login attempts, demonstrating its ability to function under stress. However, the growing response times and

failure rates indicate the need for optimization, such as horizontal scaling, database query optimization, improved caching mechanisms, and load balancing to ensure sustained performance under extreme conditions.

5.3.3.3 Load Testing: Scalability Assessment

Gradually increasing the number of simultaneous users logging in helped analyze the authentication system's ability to scale and identify resource bottlenecks. I conducted load tests with 100, 10,000, and 100,000 concurrent requests, revealing significant performance degradation at higher loads due to CPU exhaustion, memory saturation, and database bottlenecks. I measured the 95th percentile response time to determine whether performance degraded under increasing demand. At 100 users, response times remained low and stable, indicating that the system handled light traffic efficiently. However, at 10,000 users, response times spiked to an average of 44.2 seconds, with a 14% failure rate, suggesting that database queries and concurrent request handling were becoming a bottleneck. When the load reached 100,000 users, failure rates increased to 32.5%, and response times exceeded 49.8 seconds, highlighting severe resource constraints that impacted overall system reliability.

5.3.4 System Validation

To validate the effectiveness of the behavior-based authentication system. I conducted a systematic testing to measure its ability to correctly classify authentication attempts as legitimate or suspicious. This process involves assessing the system's response under different scenarios to evaluate the rate of false positives (FP), false negatives (FN), true positives (TP), and true negatives (TN). I established different authentication scenarios that include both legitimate and suspicious login behaviors. I created test cases that cover variations in user behavior, device changes, geolocation shifts, and attack simulations. Table 5.4 shows these test cases and results

Table 5.4: System validation test Cases and Results

Test Case	**Expected Outcome**	**Actual Outcome**	**Classification**	**Remarks**

1. Normal login from the same device and location	User is authenticated successfully	Authentication successful	True Negative (TN)	System correctly identified a legitimate user
2. Login from a new device in the same location	User is authenticated successfully	Authentication successful	True Negative (TN)	System did not falsely flag this as suspicious
3. Login from a different geographic location	Alert triggered	Alert triggered	True Positive (TP)	System correctly flagged unusual location login
4. Login using a VPN to mask location	Alert triggered	No alert triggered	False Negative (FN)	System failed to detect VPN-based login masking
5. Login from two distant locations within minutes (Impossible Travel Test)	Alert triggered,	Alert triggered	True Positive (TP)	System correctly flagged impossible travel login
6. API request replay attack using Burp Suite	Authentication is denied	Authentication denied	True Positive (TP)	System successfully detected API replay attack

From the above test cases, I performed a system validation analysis, Table 5.5 shows the summary of the metrics and analysis performed

Table 5.5: System Validation Metrics and Analysis

Metric	Count	Percentage (%)
True Positives (TP)	3	50%
True Negatives (TN)	2	33.33%
False Positives (FP)	1	16.67%
False Negatives (FN)	0	0%
Total Authentication Attempts	6	100%

System Accuracy Calculation:

$$\frac{TP + TN}{TP + TN + FP + FN.} = \frac{2+3}{6}$$

$$\frac{5}{6} * 100 = 83.33\%$$

5.4 Summary

The implementation and testing of the TOTP authentication system and phishing detection engine demonstrated the system's ability to provide secure, scalable, and resilient authentication.

The mobile application successfully generated time-sensitive OTPs, ensuring secure user authentication. Meanwhile, the phishing detection engine effectively identified suspicious login attempts through behavioral analysis, Geo IP tracking, and device fingerprinting.

System validation confirmed that the authentication framework met both security and usability requirements, while testing procedures ensured its reliability under real-world conditions. Security assessments verified protection against real time phishing attacks, reinforcing the system's resilience against common threats. However, performance testing revealed challenges under extreme load conditions. While the system handled low traffic efficiently, higher concurrency levels (10,000+ users) led to increased response times and failure rates, primarily due to CPU exhaustion, memory saturation, and database bottlenecks. Despite these constraints, the phishing detection engine maintained stable response times, indicating its ability to function effectively even under heavy traffic. These findings highlight the system's strong security posture while also identifying areas for performance optimization.

Chapter 6: Discussion of Results

6.1 Introduction

This chapter discusses the results obtained from implementing behavior-based phishing detection in a TOTP authentication system. The analysis highlights key findings, achievements, and how they align with the research objectives. The results are interpreted concerning the system's effectiveness in detecting real time phishing attacks.

6.2 Key Findings

The integration of behavioral analysis into the TOTP authentication system significantly enhanced its ability to detect and mitigate real-time phishing attempts. The real-time alerting mechanism played a crucial role in improving security by notifying users of potential threats before an attacker could exploit stolen credentials. This proactive approach to phishing prevention ensured that fraudulent logins were flagged based on risk factors rather than relying solely on static authentication measures. Device fingerprinting strengthened authentication security by tracking and recognizing authorized devices. If a login attempt originated from an unrecognized device, the system flagged it as suspicious and prompted additional verification measures. This mechanism was particularly effective in detecting spoofed device logins, where attackers attempted to bypass authentication using emulated or compromised environments. Geo IP tracking proved to be another valuable component of the system, enabling the detection of unusual login locations that could indicate account compromise. By correlating login locations and timestamps, the system effectively prevented session hijacking attempts where attackers attempted to authenticate from unexpected regions. However, a key limitation of this approach was its reduced reliability when users accessed the system through VPNs or proxies. Attackers could easily obfuscate their actual locations, making it difficult to distinguish between a legitimate user traveling and a malicious login attempt. Additionally, rate limiting in the GeoIP API presented a challenge in maintaining real-time accuracy, particularly during periods of high authentication activity. The implementation of login velocity analysis further contributed to detecting suspicious logins by measuring the time taken to log in from geographically distant locations. If the calculated velocity exceeded a reasonable threshold, the system flagged the activity as potentially fraudulent. This feature was instrumental in identifying session takeovers,

but its effectiveness diminished when attackers operated from compromised devices within the same geographic region as the legitimate user. Moreover, unexpected but legitimate changes in login behavior, such as a user logging in while traveling, occasionally triggered false positives, leading to unnecessary security challenges. Despite these challenges, the system successfully detected high-risk login behaviors, such as sudden spikes in authentication attempts, multiple failed logins, and rapid consecutive login attempts. These indicators allowed the system to adaptively assess threats and respond accordingly.

Stress testing, and performance constraints were observed, particularly in database query handling and concurrent request processing. Under extreme traffic loads, response times increased, and failure rates were noted, suggesting the need for optimization. A significant advantage of the proposed system was its scalability and adaptability. The modular design enabled seamless integration into existing authentication workflows without requiring major architectural changes. Overall, the findings confirm that behavioral analysis can significantly strengthen TOTP authentication by providing an adaptive and dynamic defense mechanism. While the system demonstrated strong capabilities in phishing detection and real-time threat mitigation, further refinements are necessary to reduce false positives, improve performance under load.

6.3 Challenges and Limitations

While the phishing detection system successfully met its objectives, several challenges and limitations were identified during implementation and testing. These limitations highlight areas that require further optimization to enhance system effectiveness, performance, and resilience against evolving threats.

i. Geo IP Tracking Limitations

Location-based analysis effectively detected login anomalies; however, its reliability decreased when users accessed the system via VPNs or proxies. Attackers could easily mask their true location, making geolocation alone insufficient for risk assessment. Future improvements could integrate IP reputation scoring analysis to differentiate between legitimate users and known

malicious networks.

ii. Device Fingerprinting Evasion

Sophisticated attackers can mimic legitimate browser and device fingerprints, making it possible to bypass detection mechanisms. Fingerprinting relies on user-agent data, browser configurations, and device attributes. If adversaries replicate these parameters, they could successfully evade detection. Additional verification layers, such as behavioral biometrics and continuous authentication, could enhance the reliability of device fingerprinting.

iii. Performance Constraints Under High Traffic

During stress testing, the system experienced increased response times and failure rates under heavy traffic conditions. The primary performance bottlenecks were found in database query handling and concurrent request processing, impacting real-time detection efficiency.

iv. Risk of False Positives and False Negatives

Some legitimate user behaviors triggered alerts, leading to unnecessary security challenges. Certain sophisticated phishing attempts bypassed detection, especially when attackers closely mimicked legitimate behavior.

Chapter 7: Conclusions, Recommendations and Future Work

7.1 Conclusion

This study successfully designed and implemented a real-time phishing detection system integrated with TOTP authentication. The system leveraged behavioral analysis, to identify phishing attempts. Multiple security mechanisms were employed. A key feature of the implementation was the real-time alerting mechanism, which promptly notified users of potential threats. The system also prioritized usability by employing threshold tuning and risk-based authentication, which reduced false positives and ensured legitimate users were not frequently subjected to unnecessary security challenges. This balance between security and user experience was a critical success factor in the implementation. The integration of behavioral intelligence with TOTP authentication proved to be an effective, phishing-resistant solution.

7.2 Summary of Major Results

This section provides a comprehensive summary of the key findings and outcomes from the experiments and testing conducted throughout the study. Table 7.1 shows a summary of the major results

Table 7.1: Summary of major results

Finding	**Description**
Phishing Detection via Login Behavior	The system successfully detected phishing attempts by analyzing login patterns. Multiple authentication attempts from different geographic locations within a short time frame were flagged as suspicious.
Real-Time Alerts for Suspicious Activity	Suspicious login behaviors triggered immediate alerts, notifying users via email for rapid response.

Device and Location-Based Risk Assessment	Device fingerprinting identified logins from unrecognized or mismatched devices, while GeoIP-based anomaly detection flagged logins from unusual locations.
Behavioral Risk Scoring	The system dynamically assigned risk scores based on login frequency, device mismatches, IP address changes, and geographic inconsistencies, helping to reduce false positives while mitigating security threats.
Automated Decision-Making for Authentication	Based on risk scores, the system either blocked, challenged, or allowed logins, enhancing authentication security without causing unnecessary disruptions.

7.3 Importance and Benefits of the Work

This research presented a significant advancement in the field of secure authentication by integrating behavioral analysis into TOTP systems. It enhanced phishing resistance in TOTP-based Multi-Factor Authentication environments through a real-time, adaptive, and user-centric approach.

7.3.1 Current Use of Behavioral Analysis in TOTP

Enterprise-grade authentication platforms, including solutions like Microsoft Azure Active Directory Conditional Access and Okta Adaptive MFA, often integrate such behavioral data into their risk assessment engines. These platforms use behavioral data to dynamically assess login context and either prompt for additional verification or notify security teams of potentially malicious activity. For instance, if a user attempts to log in from a new IP address and browser configuration immediately after a failed login from another country, the system may issue a multi-factor prompt or block the attempt altogether.

Despite their growing prevalence, many current implementations of behavioral analysis in TOTP/MFA systems remain proprietary and inflexible. They often depend on hardcoded rules or static thresholds that do not adapt well to evolving user behavior or new attack techniques. For instance, a system might be configured to trigger an alert if a user logs in from a location more than 500 kilometers away from their usual location. However, if the user is genuinely traveling or using a VPN, the alert may be a false positive yet the system cannot distinguish this because the rule is fixed and lacks contextual awareness or learning capability. Furthermore, integration with these systems can be complex, particularly for organizations that lack uniform infrastructure or use a variety of authentication providers. As a result, the broader adoption of behavioral analysis in diverse environments is limited, reducing its effectiveness against real-time and adaptive cyber threats.

7.3.2 Benefits of the Current research

This study addressed the limitations of current behavioral analysis systems in TOTP applications. It introduces a flexible, real-time, and user-aware phishing detection framework. This system implements adaptive risk-based authentication, using dynamic behavioral scoring to evaluate login attempts. It analyses factors such as device consistency, geolocation changes, login frequency, and interaction anomalies. This significantly reduces false positives particularly in cases where legitimate users travel or access services from new devices while maintaining a high level of security sensitivity.

To ensure practical and reliable phishing detection, the system implements a robust risk-based scoring mechanism specifically designed to minimize false positives and false negatives.To address this, the system leverages adaptive risk scoring rather than static thresholds. It learns from each user's historical patterns such as typical login times, geographic locations, and device fingerprints and uses this context to adjust risk evaluations dynamically. For instance, a user frequently logging in from different cities may initially trigger alerts, but the system quickly adapts by recalibrating the acceptable behavioral boundaries, thus avoiding repeated false alarms.

Conversely, for false negatives, the system mitigates this through multi-factor behavioral evaluation. It aggregates a variety of signals such as IP reputation, device mismatches, login timing anomalies, and location shifts. Rather than relying on any single indicator, the system

computes a composite risk score that reflects the overall anomaly level of the login attempt. This holistic scoring model increases the likelihood of catching sophisticated phishing attempts that may otherwise evade detection by mimicking isolated behavioral traits.

Importantly, the risk scoring engine updates in real time as new behaviours are observed, continuously refining its understanding of both normal and malicious activity. This feedback loop ensures the system remains resilient against evolving attack strategies while maintaining a user-friendly experience for legitimate users.

To ensure broad applicability and ease of deployment, the system is designed with a modular and open architecture. It exposes risk assessment capabilities through RESTful APIs, enabling seamless integration into existing authentication workflows without requiring substantial changes to backend infrastructure. This makes the solution especially attractive for organizations looking to enhance security while minimizing operational overhead.

Furthermore, the study emphasizes transparency and community engagement by offering a fully open-source implementation. The complete prototype, including its source code, documentation, and integration guidelines, are freely available on GitHub. This openness not only facilitates peer validation and collaborative enhancement but also ensures that organizations of all sizes can benefit from the system's capabilities without financial or licensing barriers.

7.4 Future Work

While this research successfully demonstrated the effectiveness of behavioral analysis in phishing-resistant authentication, several areas remain open for further enhancement. Future work could expand the system's capabilities, integrate emerging security technologies, and improve overall robustness against evolving cyber threats. One major area of improvement is expanding the phishing detection system to other authentication methods beyond TOTP. While TOTP provides an additional layer of security, they are still susceptible to phishing attacks. Future research could explore FIDO2/WebAuthn-based authentication, which offers phishing-resistant authentication by leveraging public-key cryptography and biometrics instead of shared secrets. This would significantly reduce the risk of credential theft and replay attacks. Additionally, the system could benefit from AI-driven behavioral analysis for adaptive risk scoring. Currently, risk assessment relies on predefined heuristics and thresholds, which may not always adapt well to evolving attack patterns or legitimate user behavior variations. By

integrating machine learning algorithms, the system could continuously learn from user interactions and dynamically adjust risk scores based on real-time behavioral insights. This would improve the accuracy of anomaly detection, reduce false positives, and enhance automated decision-making. Another key improvement involves cross-platform support. While the system was developed for Android and web-based authentication, extending support to iOS and Windows authentication flows would significantly improve adoption. This would ensure that users across multiple device ecosystems benefit from phishing-resistant authentication, increasing the system's applicability in both consumer and enterprise environments.

References

Adake, M., Belekar, A., Ambekar, C., & Raigar, D. (2023). Real-time phishing website detection using machine learning and updating phishing probability with user feedback. *International Journal of Recent Technology and Engineering (IJRTE), 12*(5), 64–71. https://doi.org/10.35940/ijrte.A7608.0512123

Almeida, L., Fernandez, B., Zambrano, D., Almachi, A., Pillajo, H., & Yoo, S. G. (2024). One-time passwords: A literary review of different protocols and their applications. In A. Aparicio (Ed.), *Proceedings of the 2024 Conference on Information Security* (pp. 205–219). Springer. https://doi.org/10.1007/978-3-031-48855-9_16

Aparicio, A., Martínez-González, M. M., & Cardeñoso-Payo, V. (2023). App-based detection of vulnerable implementations of OTP SMS APIs in the banking sector. *Wireless Networks, 30*, 6451–6464. https://doi.org/10.1007/s11276-023-03455-w

Berrios, J., Mosher, E., Benzo, S., Grajeda, C., & Baggili, I. (2023). Factorizing 2FA: Forensic analysis of two-factor authentication applications. *Forensic Science International: Digital Investigation, 45*, 301569. https://doi.org/10.1016/j.fsidi.2023.301569LSU Scholarly Repository

Bianchi, G., & Valeriani, L. (2023). Time is on my side: Forward-replay attacks to TOTP authentication. In D. Lee & H. Lee (Eds.), *Proceedings of the 2023 International Conference on Cryptography and Network Security*. Springer. https://doi.org/10.1007/978-981-99-5177-2_7

Dam, M.-L., Hung, H., Chau, H., Vu, Q. S., & Tran, T.-N. (2024). Real-time phishing detection using deep learning methods by extensions. *International Journal of Electrical and Computer Engineering (IJECE), 14*(3), 3021–3035. https://doi.org/10.11591/ijece.v14i3.pp3021-3035

Fejrskov, M., Pedersen, J. M., & Vasilomanolakis, E. (2022). Detecting DNS hijacking by using NetFlow data. In *2022 IEEE Conference on Communications and Network Security (CNS)* (pp. 273–280). IEEE. https://doi.org/10.1109/CNS56114.2022.9947264

Gulhane, V., & Bansode, R. S. (2024). Enhancing cyber defense: A comprehensive study of DNS security. *International Journal of Scientific Research in Engineering and Management, 8,* 1–17. https://doi.org/10.55041/IJSREM36765

Hassan, M., Shukur, Z., & Hasan, M. K. (2020). An improved time-based one-time password authentication framework for electronic payments. *International Journal of Advanced Computer Science and Applications, 11*(11), 359–366. https://doi.org/10.14569/IJACSA.2020.0111146

Islam, M., Sajjad, M., Hasan, M. M., & Mazumder, M. (2023). Phishing attack detecting system using DNS and IP filtering. *Asian Journal of Computer Science and Technology, 12*, 16–20. https://doi.org/10.51983/ajcst-2023.12.1.3552

Mihailescu, M. I., Nita, S. L., Rogobete, M., & Marascu, V. (2023). Unveiling threats: Leveraging user behavior analysis for enhanced cybersecurity. In *2023 15th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)* (pp. 1–6). IEEE. https://doi.org/10.1109/ECAI58194.2023.10194039

Nguyen Ba, M. H., Bennett, J., Gallagher, M., & Bhunia, S. (2021). A case study of credential stuffing attack: Canva data breach. In *2021 International Conference on Computational Science and Computational Intelligence (CSCI)* (pp. 735–740). IEEE. https://doi.org/10.1109/CSCI54926.2021.00187

Önal, V., Arslan, H., & Görmez, Y. (2024). Machine learning and event-based user and entity behavior analysis. In *2024 32nd Signal Processing and Communications Applications Conference (SIU)* (pp. 1–4). IEEE. https://doi.org/10.1109/SIU61531.2024.10600861

Pal, S. (2019). Secure OTP concept for banking. *Volume-2*, 94–95

Plata, I., & Calpito, J. (2020). Application of time-based one-time password (TOTP) algorithm for human resource e-leave tracking web app. *International Journal of Scientific & Technology Research, 9*(4), 4070–4077.

Shirvanian, M., & Agrawal, S. (2021). 2D-2FA: A new dimension in two-factor authentication. *arXiv*. https://doi.org/10.48550/arXiv.2110.15872

Salahdine, F., El Mrabet, Z., & Kaabouch, N. (2021). Phishing attacks detection: A machine learning-based approach. In *2021 IEEE 12th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)* (pp. 250–255). IEEE. https://doi.org/10.1109/UEMCON53757.2021.9666627

Sun, Y., Zhu, S., Zhao, Y., & Sun, P. (2021). *Let your camera see for you: A novel two-factor authentication method against real-time phishing attacks*. arXiv. https://doi.org/10.48550/arXiv.2109.00132.

Tang, L., & Mahmoud, Q. H. (2022). A deep learning-based framework for phishing website detection. *IEEE Access, 10*, 1509–1521. https://doi.org/10.1109/ACCESS.2021.3137636

Teffandi, N., Feryputri, N. A.-Z., Hasanuddin, M. O., Syafalni, I., & Sutisna, N. (2023). GRAIN algorithm implementation for lightweight hardware-based OTP authentication. In *2023 International Conference on Electrical Engineering and Informatics (ICEEI)* (pp. 1–6). IEEE. https://doi.org/10.1109/ICEEI59426.2023.10346638

Gilsenan, C., Shakir, F., Alomar, N., & Egelman, S. (2023). Security and privacy failures in popular 2FA apps. In *Proceedings of the 32nd USENIX Security Symposium* (pp. 2079–2096). USENIX Association. https://www.usenix.org/conference/usenixsecurity23/presentation/gilsenan

Sudar, C., Arjun, S. K., & Deepthi, L. R. (2017). Time-based one-time password for Wi-Fi

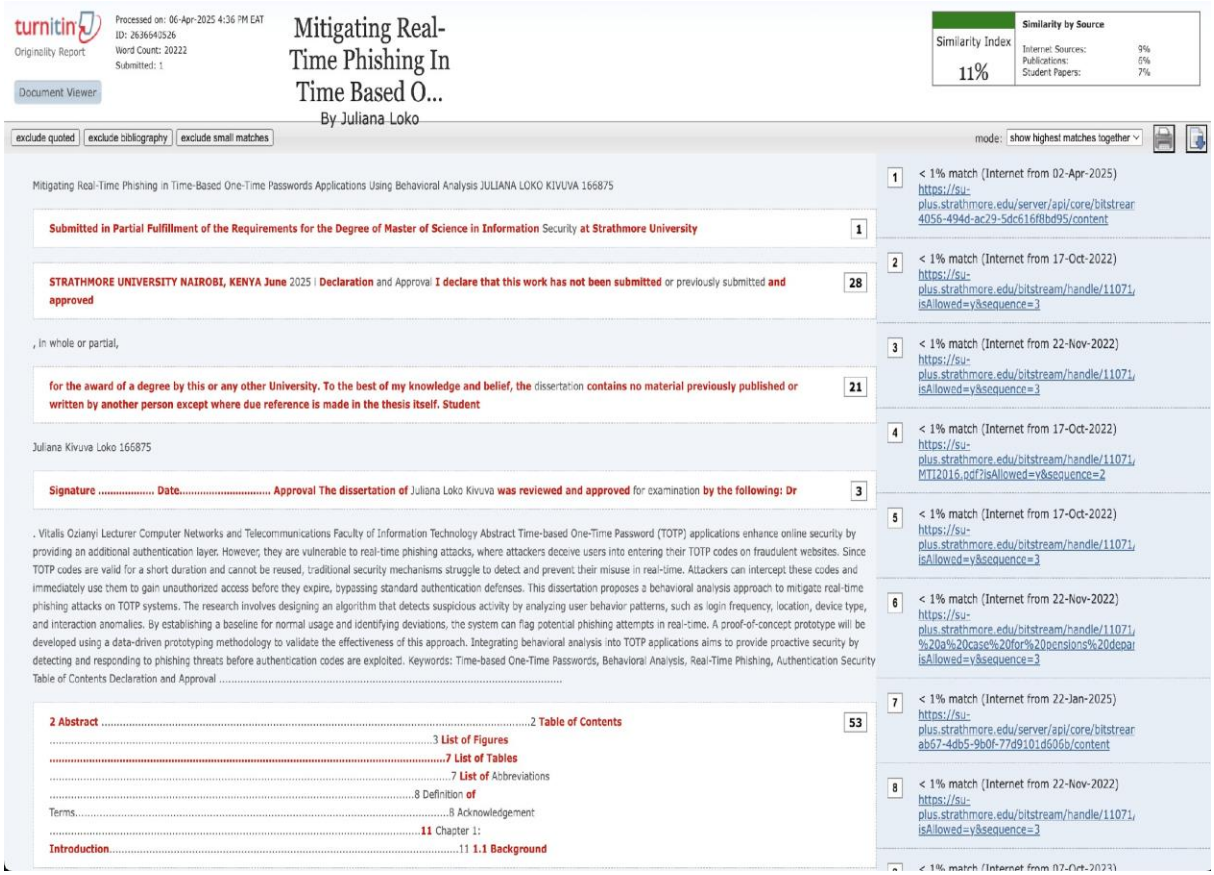
authentication and security. In *Proceedings of the 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)* (pp. 1212–1216). IEEE. https://doi.org/10.1109/ICACCI.2017.8126007

Sofian, A. B., Peradus, A. F. A. B., Yong, F., Shearer, I., Ismail, N. N. B., Mahendran, Y. A., & Faisal, M. (2024). Enhancing authentication security: Analyzing time-based one-time password systems. *International Journal of Computer Technology and Science*, *13*(1), 45–58. https://doi.org/10.62951/ijcts.v1i3.25

Ulqinaku, E., Lain, D., & Čapkun, S. (2019). 2FA-PP: 2nd factor phishing prevention. In *Proceedings of the 12th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec '19)* (pp. 184–189). Association for Computing Machinery. https://doi.org/10.1145/3317549.3319720

Appendices

Appendix A: Similarity Report



turnitin
Originality Report

Processed on: 06-Apr-2025 4:36 PM EAT
ID: 2636642526
Word Count: 20222
Submitted: 1

Document Viewer

Mitigating Real-Time Phishing In Time Based O...
By Juliana Loko

Similarity Index 11%

Similarity by Source
Internet Sources: 9%
Publications: 6%
Student Papers: 7%

exclude quoted | exclude bibliography | exclude small matches

mode: show highest matches together

Mitigating Real-Time Phishing in Time-Based One-Time Passwords Applications Using Behavioral Analysis JULIANA LOKO KIVUVA 166875

Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science in Information Security **at Strathmore University** 1

STRATHMORE UNIVERSITY NAIROBI, KENYA June 2025 | **Declaration** and Approval **I declare that this work has not been submitted** or previously submitted **and approved** 28

, In whole or partial,

for the award of a degree by this or any other University. To the best of my knowledge and belief, the dissertation **contains no material previously published or written by another person except where due reference is made in the thesis itself. Student** 21

Juliana Kivuva Loko 166875

Signature ..... Date..... Approval The dissertation of Juliana Loko Kivuva **was reviewed and approved** for examination **by the following: Dr** 3

. Vitalis Ozianyi Lecturer Computer Networks and Telecommunications Faculty of Information Technology Abstract Time-based One-Time Password (TOTP) applications enhance online security by providing an additional authentication layer. However, they are vulnerable to real-time phishing attacks, where attackers deceive users into entering their TOTP codes on fraudulent websites. Since TOTP codes are valid for a short duration and cannot be reused, traditional security mechanisms struggle to detect and prevent their misuse in real-time. Attackers can intercept these codes and immediately use them to gain unauthorized access before they expire, bypassing standard authentication defenses. This dissertation proposes a behavioral analysis approach to mitigate real-time phishing attacks on TOTP systems. The research involves designing an algorithm that detects suspicious activity by analyzing user behavior patterns, such as login frequency, location, device type, and interaction anomalies. By establishing a baseline for normal usage and identifying deviations, the system can flag potential phishing attempts in real-time. A proof-of-concept prototype will be developed using a data-driven prototyping methodology to validate the effectiveness of this approach. Integrating behavioral analysis into TOTP applications aims to provide proactive security by detecting and responding to phishing threats before authentication codes are exploited. Keywords: Time-based One-Time Passwords, Behavioral Analysis, Real-Time Phishing, Authentication Security

Table of Contents Declaration and Approval .....

2 Abstract .....2 **Table of Contents** 53
.....3 **List of Figures**
.....**7 List of Tables**
.....7 **List of** Abbreviations
.....8 Definition **of**
Terms.....8 Acknowledgement
.....**11** Chapter 1:
Introduction.....11 **1.1 Background**

1 < 1% match (Internet from 02-Apr-2025)
https://su-plus.strathmore.edu/server/api/core/bitstreams/4056-494d-ac29-5dc616f8bd95/content

2 < 1% match (Internet from 17-Oct-2022)
https://su-plus.strathmore.edu/bitstream/handle/11071/isAllowed=y&sequence=3

3 < 1% match (Internet from 22-Nov-2022)
https://su-plus.strathmore.edu/bitstream/handle/11071/isAllowed=y&sequence=3

4 < 1% match (Internet from 17-Oct-2022)
https://su-plus.strathmore.edu/bitstream/handle/11071/MTI2016.pdf?isAllowed=y&sequence=2

5 < 1% match (Internet from 17-Oct-2022)
https://su-plus.strathmore.edu/bitstream/handle/11071/isAllowed=y&sequence=3

6 < 1% match (Internet from 22-Nov-2022)
https://su-plus.strathmore.edu/bitstream/handle/11071/%20a%20case%20for%20pensions%20depar isAllowed=y&sequence=3

7 < 1% match (Internet from 22-Jan-2025)
https://su-plus.strathmore.edu/server/api/core/bitstreams/ab67-4db5-9b0f-77d9101d606b/content

8 < 1% match (Internet from 22-Nov-2022)
https://su-plus.strathmore.edu/bitstream/handle/11071/isAllowed=y&sequence=3

Appendix B: Ethical Approval

Strathmore UNIVERSITY

3rd April 2025

Ms Kivuva Juliana,
juliana.loko@strathmore.edu

Dear Ms Kivuva,

RE: Mitigating Real-Time Phishing in TOTP Applications Using Behavioral Analysis

This is to inform you that SU-ISERC has reviewed and **approved** your above **SU-masters** proposal. Your application reference number is **SU-ISERC2786/25.** The approval period is from **3rd April 2025 to 2nd April 2026.**

This approval is subject to compliance with the following requirements:

i. Only approved documents including (informed consents, study instruments, MTA) will be used.
ii. All changes including (amendments, deviations, and violations) are submitted for review and approval by SU-ISERC.
iii. Death and life-threatening problems and serious adverse events or unexpected adverse events whether related or unrelated to the study must be reported to SU-ISERC within 72 hours of notification.
iv. Any changes anticipated or otherwise that may increase the risks or affected safety or welfare of study participants and others or affect the integrity of the research must be reported to SU-ISERC within 72 hours.
v. Clearance for the export of biological specimens must be obtained from relevant institutions.
vi. Submission of a request for renewal of approval at least 60 days prior to the expiry of the approval period. Attach a comprehensive progress report to support the renewal.
vii. Submission of an executive summary report within 90 days of completion of the study to SU-ISERC.

Before commencing your study, you will be expected to obtain a research license from National Commission for Science, Technology, and Innovation (NACOSTI) https://research-portal.nacosti.go.ke/ and obtain other clearances needed.

Yours sincerely,

Mr Ambrose Rachier,
Chairperson; SU-ISERC

Ole Sangale Rd, Madaraka Estate. PO Box 59857-00200, Nairobi, Kenya. Tel +254 (0)703 034000
Email admissions@strathmore.edu www.strathmore.edu