Final Report: 2D-2FA Software Implementation

Zane Globus-O'Harra, Doug Ure

9 June 2023

Abstract

TODO TODO TODO TODO TODO TODO TODO TODO TODO

Contents

1 Introduction

TODO: need to go through and reword/edit this section

Two-factor authentication (2FA) is the contemporary approach to authorizing a user. The first authentication factor is the user's password, with the second factor being an additional piece of information that only the user could know, often a PIN or a push notification sent to the user over a

secure channel to a trusted device. However, both PIN-2FA and push-based 2FA have some issues, which are addressed in the paper "2D-2FA: A New Dimension In Two-Factor Authentication" by Maliheh Shirvanian and Shashank Agrawal [1].

Some attacks to these two common 2FA methods include shoulder surfing, short PINs, and neglectful user approvals. The authors present a new approach to 2FA, which they have coined "2D-2FA." In this new approach, when a user logs in with their username and password, a unique identifier is displayed to them. The user then inputs this same identifier on their device. A one-time PIN is generated on the device, and transferred automatically to the server, along with the identifier. The identifier is used in the PIN's computation, so that the PIN is bound to a specific session.

The user's device and the server agree on a secret key during a one-time registration process, which is also used in the PIN computation. Once the PIN is transferred to the server, the server authenticates the session associated with the identifier by verifying the PIN, thereby taking two dimensions into account (the PIN and the identifier).

1.1 Keywords

To avoid further confusion, we will address several important keywords that are important to the design and implementation of the 2D-2FA system.

2D-2FA This is the name of the system that is outlined in [1]. It uses no third party software, and relies only on standardized encryption and hashing algorithms, as well as standardized network protocols. The two dimensions that this system uses to authenticate the user is the identifier and the PIN.

Identifier This is a random value that is generated by the server and presented to the user via the client interface. The user needs to enter the identifier onto their device, where it is used during the PIN generation. The server then uses the identifier that it originally generated to verify the PIN. In [1], they recommend using a pattern or a QR code as the identifier for ease of use. In our implementation, we used a 6-digit number as the identifier to increase the ease of implementation.

PIN The PIN is generated using the time slice and the identifier. It is sent to

User words words words

Client words words words

Server words words words

Device words words words

1.2 Motivations

We are motivated to work on this project for several key reasons. Firstly, the subject of this project has real-world relevance. 2FA is becoming increasingly more common to authenticate users, and as it becomes more prevalent, attackers will focus more of their efforts on finding ways to break through its layers of security. Our project will help us learn about ways to further increase the security of 2FA by using additional information along with the user's credentials and the server-provided "identifier."

This is also a learning opportunity for us. Neither of us are very familiar with security, and it is something that we are very interested in learning about. By completing this project, we will develop valuable technical skills and increase our knowledge base, as well as preparing us for future projects and industry roles.

Lastly, this project could have a real impact on end-users. 2FA enhances users' trust and confidence in online systems by making their personal information and online interactions more secure. This project has the potential to contribute to a larger goal of make a more secure digital environment.

1.3 Objectives

Our objectives for this project are to create a working implementation of the 2D-2FA system in software, focusing mainly on the authentication phase, as described in section 3.2 of [1]. As previously mentioned in our progress report, we have written software to implement the functionality of the server and the device in this authentication scheme. We also added a simple web interface for the client and the device, so that the user can use their browser to use our implementation.

In terms of specific deliverables, these were outlined in our midterm report, but are repeated here for posterity.

- A working implementation of 2D-2FA.
  - Programs for the server and the device, as described in the 2D-2FA paper.
  - This implementation will work across multiple devices.
  - This implementation will work for multiple users.
- Test cases for our code.
- Documentation.
  - Installation instructions.
  - Usage instructions.
  - A design diagram.
  - Well-commented code.

2 Related Work

In this section, we will look at some traditional 2FA implementations, chiefly hardware token-based authentication and single sign-on software token-based authentication.

2.1 Hardware Token Authentication

2.2 Single Sign-On Software Token Authentication

3 2D-2FA

3.1 Design

The 2D-2FA system consists of two phases, the registration phase and the authentication phase. In our project, we assumed that the registration phase had already been completed between the involved

parties, but we will go over that phase here because it would be required for the implementation of a full 2D-2FA system, rather than a toy example or proof of concept implementation.

Registraion Phase During the registration phase, the parties involved in the 2D-2FA protocol need to establish communication channels between them, and share secret keys between the server and the device.

First, a user would register their username and password with the server. The server would pick a secret key for that user (generating it from their password using HMAC). The secret then needs to be transferred to the user's device (e.g., cell phone), which is typically done by manually entering it into the device, or by scanning a QR code generated by the server. This is so that the device will be able to generate PINs later on in the 2D-2FA process.

The server stores a hash of the user's password and secret key, and the device stores only the secret key related to the user's account stored on the server. The client (e.g., a web browser on a laptop that the user uses to log in to the server) does not store any information during the registration phase.

Authentication Phase The authentication phase involves the interaction of the parties to authenticate the user to the server in a secure fashion.

First, the user attempts to log in to the server from the client. The user's login information (username and password) are used as the first authentication factor. Next, the server displays a unique identifier to the user through the client (the identifier can be implemented in a variety of ways, either via QR code, a fourgram pattern as described in section 6.1 of [1], and so on).

On the user's device, the user selects the server that they are logging in to, and enters the identifier that was displayed to them on the client. The device generates a PIN using the identifier, the current time, and the user's secret key. The PIN and the identifier are sent to the server.

When the server receives the PIN and the identifier, it authenticates the user's session associated with the identifier, generating several PINs to find one that matches with the time the PIN was generated on the user's device. During this phase, the server keeps a temporary record of all active sessions and identifiers.

3.2 Implementation

3.3 Security

TODO TODO TODO TODO TODO TODO TODO TODO TODO TODO

There are several possible avenues of attack to the 2D-2FA system that are proposed in [1]. In this section, we will go over each attack vector and analyze how the system protects against these attacks, as well as explaining how our implementation prevents these attacks.

Client Compromise In this attack, an attacker would have compromised the client that the user uses to log into the server, and thus gain access to the user's password. Assuming the device is not also compromised, the user's account is still secure.

This attack is prevented because once an attacker logs in using the user's password, they will receive an identifier generated by the server. But, without access to the user's secret key, the attacker has no means of computing a PIN that corresponds to the identifier, and thus no means of providing a valid PIN to the server. Additionally, without access to the user's device, the attacker has no way to generate a valid PIN without guessing the user's secret key, which would not be feasible.

In our implementation, TODO

Device Compromise In this attack, an attacker would have compromised the user's device, and as such they have full control over the device. In this attack vector, we assume that the attacker has access to the secret information stored on the device. This allows the attacker to generate any PIN, for any identifier.

This attack is prevented because for the attacker to gain access to the user's account, the attacker still needs to know the user's password to log in on the client. If the user has chosen a secure password that can not easily be guessed, the attacker must guess the user's password to gain access to their account, which would not be feasible.

In our implementation, TODO

Channel Compromise In this attack, an attacker has control over the channels connecting the parties (the client, device, and server) in the system. Specifically, there are three main channels in the 2D-2FA system. The channel between the client and the server is secure, the channel between the client and the device is through the user, and the channel between the device and the server is a regular channel. In this attack vector, the attacker could either listen to the channels (eavesdrop), or they could modify or block the network traffic.

This attack is prevented because, like with the client compromise, the attacker does not know the user's password.

TODO TODO TODO TODO

In our implementation, TODO

User Negligence In this attack, a user is simply negligent, and inadvertantly grants access to an attacker.

This attack is prevented because, a user would only accidentally approve an attacker's session when the user enters the identifier that is displayed to the attacker, rather than the identifier that is displayed to the user. Depending on the type of identifier used, the possibility of this occurring is very low.

In our implementation, we used a 6-digit identifier, which means that if an attacker has logged in and is presented with an identifier, the user has a one-in-a-million chance (0.0001%) of approving that attacker's session.

Attacks on Third Parties In other MFA systems, third party entities are introduced, such as a MFA service provider. The system owner would need to trust these service providers, as well as study the security of the services that they provide to be comfortable with the security of their services. Attackers can target these third parties, as well as the channels connecting the third party services to the main system, which increases the complexity of the security analysis of the system.

Because 2D-2FA does not use any third party systems, this decreases the complexity of the security analysis, and thus reduces the surface of attack into the system. Additionally, 2D-2FA only uses well-established and well-studied technologies (HMAC, SSL/TLS, random number generation, etc.), further reducing attack vectors.

In our implementation, we use the well-established technologies mentioned in and used by [1]. The only variation is that we use TCP connections and encrypt the messages that we send over those connections instead of using TLS.

4 Results

5 Analysis

6 Conclusion

6.1 Complications

6.2 Lessons

The important lessons that we have learned include project planning, collaboration skills, and iterative development. While we had prior experience in all of these areas from previous projects, this project helped ingrain these principles into how we worked, altogether adding to a better workflow and increased productivity.

For the project planning, we had thoroughly read through the implementation section in [1]. From this, we broke down each element of the implementation into smaller chunks that were easier to tackle and implement. This allowed us to develop one module at a time, and ensure that module was functioning in the desired way before continuing to the next module.

In terms of collaboration, we have weekly meetings where we discuss what we have accomplished in the past week, and what we plan on completing for the next week. During the week, we update each other with our progress, as well as asking questions or seeing if we have suggestions for each other.

With regard to iterative development, this goes hand in hand with our project planning. Because we have broken down the problem into bite-sized chunks, we can iteratively implement these small portions, easily adding features and functionality to them as we progress, and iteratively changing them or modifying them when we encounter the need to do so.

6.3 Recommendations

Possibly add to this with typing proof [2]

References

[1] M. Shirvanian and S. Agrawal, "2d-2fa: A new dimension in two-factor authentication," 2021.

[2] X. Liu, Y. Li, and R. H. Deng, "Typing-proof: Usable, secure and low-cost two-factor authentication based on keystroke timings," in *Proceedings of the 34th Annual Computer Security Applications Conference*, ACSAC '18, (New York, NY, USA), pp. 53–65, Association for Computing Machinery, 2018.