Multi-Factor Graphical User Authentication for Web Applications

Hasmik Badikyan

Dissertação apresentada à Escola Superior de Tecnologia e de Gestão de Bragança para obtenção do Grau de Mestre em Sistemas de Informação.

Trabalho orientado por:

Rui Pedro Lopes

Tiago Pedrosa

Esta dissertação não inclui as críticas e sugestões feitas pelo Júri.

Bragança

2017

Multi-Factor Graphical User Authentication for Web Applications

Hasmik Badikyan

Dissertação apresentada à Escola Superior de Tecnologia e de Gestão de Bragança para obtenção do Grau de Mestre em Sistemas de Informação.

Trabalho orientado por:

Rui Pedro Lopes

Tiago Pedrosa

Esta dissertação não inclui as críticas e sugestões feitas pelo Júri.

Bragança

2017

Abstract

Nowadays everybody uses web applications and need to protect their accounts with strong authentication methods.

Following this need, this work research problems and solutions related with the authentication, specially concerning textual and graphical passwords.

The common problem among the users is the difficulty remembering a textual password that is long and random-looking. Because of the visual aspect, graphical passwords are more easy to remember.

This work proposes a recognition and recall based graphical authentication methods that can be used as a challenge to authenticate users. A security analysis is made to check the correctness of the proposed solution and how it minimizes the vulnerabilities of the authentication process.

These analysis will enable us to implement these challenges in future work as an extension to authentication, authorization and accounting services, supporting a multi-factor authentication and combining theses challenges with others already available. The idea is to extend an authentication method on Apache Shiro to provide developers with a common framework to develop secure web application with strong authentication, authorization and accounting.

Resumo

Hoje em dia, as pessoas fazem uso de aplicações web e necessitam proteger as suas contas com métodos de autenticação forte.

Considerando esta necessidade, este trabalho investiga os problemas e soluções de autenticação, especialmente relacionadas com palavras chave textuais e gráficas.

Um problema comum dos utilizadores é a dificuldade de se lembrar de palavras chave textuais que sejam longas e pareçam criadas aleatoriamente. Devido ao aspeto visual, as palavras chave gráficas são mais fáceis de recordar.

Este trabalho propõe métodos de autenticação gráfica baseados em reconhecimento e localização de pontos que podem ser utilizados como desafios de autenticação. É também efetuada uma análise de segurança aos métodos propostos por verificar a sua correção e que minimizam vulnerabilidades do processo de autenticação.

Estes resultados permitirão, no futuro, implementar desafios de autenticação adicionais como uma extensão aos serviços de autenticação, autorização e contabilização, suportando autenticação multi-fator. A ideia será estender os métodos de autenticação do Apache Shiro para permitir os programadores desenvolverem, utilizando uma framework comum, aplicações web seguras com autenticação, autorização e contabilização.

Acknowledgments

First I would like to thank my supervisors professor Rui Pedro Lopes and professor Tiago Pedrosa for providing me with the opportunity to work in this research, for their encouragement support and supervision at all levels. I am grateful to professor Gevorg Margarov for all his help and guidance from Armenia. Also specially I want to thank Erasmus+ ICM project for supporting the research collaboration between IPB and NPUA and thank you all.

Contents

Abstract **v**

Resumo **vii**

Acknowledgments **ix**

1 Introduction **1**

1.1 Goals 2

1.2 Document structure 3

2 Background **4**

2.1 Authentication, authorization and accounting 4

2.2 Web-based authentication 5

2.3 Textual passwords 8

2.4 Graphical Passwords 13

2.5 Other authentication mechanisms 17

2.5.1 OpenID 17

2.5.2 Kerberos 19

2.6 Apache Shiro 20

3 Proposal **23**

3.1 Recognition based method 26

3.1.1 Recognition based implementation 28

3.2 Recall based method 30
3.2.1 Recall based implementation 33

4 Discussion **35**

5 Conclusions **37**

List of Figures

2.1 Web-Based 6
2.2 Categorization 14
2.3 CreatePassword 16
2.4 Create Time 16
2.5 Login 16
2.6 OpenID login protocol flow. 18
2.7 ShiroBasicArchitecture 21

3.1 Challenge 24
3.2 RecognitionSample 26
3.3 Choosing Password 27
3.4 Validation 28
3.5 RecallSample 31
3.6 Recall 32
3.7 Validation 32

Chapter 1

Introduction

The Internet and World Wide Web have become important parts of most people's lives. Users regularly use various Web applications that involve personal information, including, among others, online banking, e-health services, online education, etc..

The organizations which provide these services maintain personal information on their servers, as well as a variety of other types of sensitive information crucial to the organizations' operation. These information is required to be secured and restricted to authorized individuals.

The organizations must have a policy that protects the privacy of the users' and ensure that their personal information does not fall into the hands of people for whom it is not intended.

Although there are many methods to limit access to information, they usually depend on a process of authentication and authorization. The most common authentication method is based on a user name and corresponding password. For this to be effective, it is very important that users generate and use strong passwords that are resistant to guessing and cracking. However, usually there is a trade-off between password memorability and security. Passwords that are easy to remember, as a rule, are biographical information or simple words that can be guessed or cracked by other people or computer programs. Research has shown that users, in fact, tend to create passwords that are easy to remember. For example, Leyden [1] reported that 12% of users used "password" as their password,

and the three most common types of passwords included a user's own name, date of birth, or favorite football team. Another problem with passwords is that people tend to use the same password for different accounts. If the password of one account gets hacked, then the security for all the other accounts is endanger.

The user name and password authentication method provides less security than other methods, such as biometric devices, smart cards, or token devices. However, for many websites that support personal information about users, the combination of user name and password still is the primary method of identifying and authenticating users. This method is popular because it is widely accepted by many users and it is easy to implement. It is unlikely that user name and password be replace in the near future, for reasons of convenience and practicality. Therefore, it is important to determine the methods for generating passwords that will allow you to obtain passwords that provide adequate security, but that are also memorable.

1.1 Goals

Considering the previous scenario we set the main goals of this work as follows:

- Study web application authentication solutions
- Identify the risks of textual passwords and attack vectors
- Research graphical password methods
- Propose authentication solutions based on recognition and recall based methods
- Analyze security of the proposed methods

1.2 Document structure

This document is structured in 5 chapters. The current chapter describes the main goals of this work and makes an introduction about web application authentication, their challenges and identified a problem related with weakness of password authentication.

On chapter 2, we research the authentication methods for web applications, strengths an weakness of textual and graphical passwords. We also research AAA frameworks and other common solutions.

On chapter 3, we specify and explain our graphical authentication proposals for recognition and recall based method.

On chapter 4 we discus the security of the proposed methods related with the previous identified attack vectors.

The document finishes with the conclusion chapter, where goals completion is discussed, as well as the future challenges and work.

Chapter 2

Background

In this chapter we discus the importance of web authentication as part of web application security. We identify advantages and disadvantages of textual and graphical passwords. Moreover, we analyzed some AAA frameworks and common used solutions for web applications.

2.1 Authentication, authorization and accounting

Authentication, authorization, and accounting (AAA) is a term for a framework that intelligently controls access to computer resources, audits its' usage, enforces policies and provides information necessary to bill services usage. Authentication asks the question, "Who or what are you?". Authorization asks, "What are you allowed to do?". Finally, accounting wants to know, "What did you do?". The use of AAA is considered essential to effective network management and security.

First a user needs to go trough authentication process, usually users are queried to provide a valid user name and password combination before gaining access. The AAA server compares a user's authentication credentials with other user's ones that are stored in a database. If the credentials match, the user is authenticated. If credentials are rejected, authentication fails and access is denied.

Follows the authorization phase to check if the user is allowed to perform some tasks.

After logging into a system, for instance, the user may try to issue commands. The authorization process checks if the user has the permission to issue such commands. This phase enforces the policies as it enables to define the types or properties of the activities, resources or services that the user is allowed to use. Usually, authorization take place within the context of authentication. Once you have authenticated a user, they may be authorized for different types of access or activity.

The AAA framework, also provides accounting, which measures resources usage during access. It can include the amount of system time or amount of data that the user sent and received during the session. Accounting is performed by registering session statistics and usage information and enables to control authorization, billing, trend analysis, resource utilization, and capacity planning activities.

Authentication, authorization, and accounting services are often provided by a dedicated AAA server, a program that performs these functions. A current standard by which network access servers interface with the AAA server is the Remote Authentication Dial-In User Service (RADIUS) [2].

2.2 Web-based authentication

The web-based authentication feature implements web-based authentication, which is also known as Web Authentication Proxy. The Web-based environment is a communication infrastructure, usually a network that uses specific technologies. Such as the Hypertext Transfer Protocol (HTTP), the (Extensible) Hypertext Markup Language ((X)HTML) or Uniform Resource Identifiers (URIs).

The biggest web-environment is the World Wide Web. However, web technologies are used in diverse environments, from home to enterprise networks. There are two main alternatives to authentication in such environments: authentication functions of the communication protocol (HTTP) or authentication using application specific methods and parameters, that are exchanged using forms. The following sections focus on user authentication. Server authentication will not be pursued further in this thesis. There

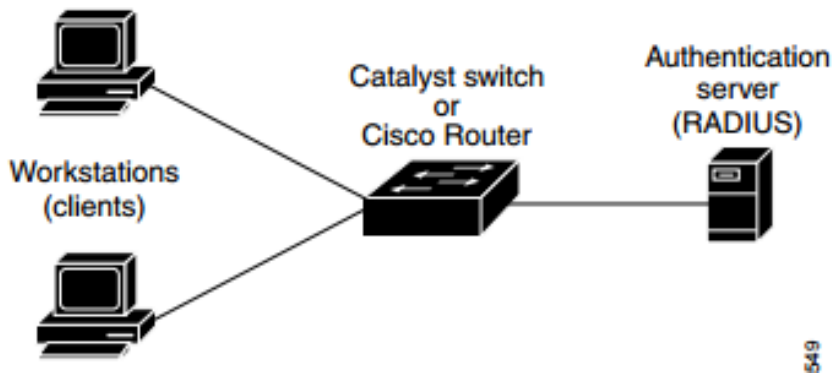

Figure 2.1: Web-based authentication device roles

are also a number of implicit methods to authenticate a server to a user. For example, the URI which a server responds to can be used to authenticate it or information provided in a SSL certificate that the server offers to secure communication. Further on this research work the focus will be on user authentication.

When a user initiates an HTTP session, the Web authentication function intercepts incoming HTTP packets from the host and sends the user's HTML login page. The user types his credentials, which the web-based authentication feature sends to the AAA server for authentication. If the authentication succeeds, web-based authentication sends a Login-Successful HTML page to the host and applies the access policies returned by the AAA server [3].

With web-based authentication, the devices in the network have specific roles as shown in 2.1

RFC 7235 specifies an authentication extension to HTTP [4], [5], [6]. This extension allows for the exchange of authentication information on the protocol level. This allows, for example, web servers that only understand HTTP to enforce a user authentication. In contrast to form-based authentication (see below) the user client has to support the authentication method since it is required to process it. The user client, for example, has to recognize a HTTP authentication request, prompt the user for his credentials and send them to the authenticator. Also, the HTTP authentication methods only arrange for a username and a password as authentication credentials. Other parameters like an

authentication provider or a control challenge, therefore, need to be guessed from the context or encoded into the username/password tokens.

There are two authentication methods specified: basic and digest access authentication. The basic authentication transfers username and password in cleartext. This allows for a direct comparison on the receiving end of the authentication with the values that are present there. If the transmitted values match the ones stored before, the authentication is successful. The transmission, especially of the password, in cleartext, however, bears a number of security risks (which make basic authentication insecure).

The HTTP digest access authentication avoids to transfer the user password in cleartext. To achieve this, the digest authentication method applies MD5 cryptographic hashing combined with nonce values to prevent crypto-analysis. Additionally, to the authentication request, the authenticator sends the client a nonce value. Both, client and server, then perform the digest authentication computations, shown in the formulas below, to calculate the digest response value. The value HA1 contains a MD5 hash of the username, the realm these credentials are valid for and the user password [7].

$$HA1 = MD5(A1) = MD5(Username : realm : password) \tag{2.1}$$

$$HA2 = MD5(A2) = MD5(method : digestUri) \tag{2.2}$$

$$request - digest = MD5(HA1 : nonce : nonceCount : clientNonce : qop : HA2) \tag{2.3}$$

The authenticator can choose to store the HA1 value in the hashed version opposite to storing the three input values in cleartext. This has the advantage that it saves the HA1 computation every time an authentication needs to be performed. Also, in case the authentication database is compromised, the attacker cannot get access to the cleartext user password. On the other hand, the realm value cannot be changed after the hash value is calculated. The second part of the digest computation is the HA2 value which hashes the HTTP request method and URI. In the last step, the HA1 and HA2 value are MD5 hashed using the nonce the authenticator provided and a number of values the

client can choose to provide itself. Those additional attributes provide further security enhancements. They include a Quality of Protection (qop) parameter that specifies which of the security enhancements are required to be used, a nonce counter that is incremented by the client, and a client generated random nonce. These enhancements are designed to protect against cryptanalysis (e.g. chosen-plaintext attack) of the digest values.

2.3 Textual passwords

Passwords are the most common form of authentication, used to control access to information, ranging from the Personal Identification Numbers (PIN) we use for automatic teller machines, credit cards, telephone calling cards, and voice mail systems to the more complex alphanumeric passwords that protect access to files, computers, and network servers. Passwords are widely used because they are simple, not expensive, and convenient mechanisms to use and implement.

At the same time, passwords are also recognized as being an extremely poor form of protection. The Computer Emergency Response Team (CERT) estimates that about 80 percent of the security incidents reported to them are related to poorly chosen passwords [8]. Password problems are very difficult to manage because a single local computer network may have hundreds or thousands of password-protected accounts and only one needs to be compromised to give an attacker an entrance to the local system or network. With today's interconnected Internet, the problems are potentially devastating on an even larger scale, a skillful intruder may break into one system and never harm it, using it instead as a platform for attacks on a population of millions of targets [9].

Passwords are a weak form of protection for many reasons. One major reason is that passwords depend on the weakest link in the computer and network security chain: namely, the human user. There is a common trend among users to choose passwords, which are simple words, can be found in any dictionary or commonly used by other users.

To make textual passwords safer we should follow several guidelines:

- Passwords should be easy to remember, and the user authentication protocol should

be executable quickly and easily by humans;

- Passwords should be secure, i.e., they should look random and should be hard to guess;
- They should be changed frequently, and should be different on different accounts of the same user;
- They should not be written down or stored in plain text.

The satisfaction of all these guidelines are virtually impossible to achieve. Consequently, users ignore them, leading to poor password practices. According to the statistics of xato.net, 30% of the users use one of the top 10,000 passwords, which makes it ridiculously easy to hack their accounts [10] (Table 2.1).

Table 2.1: Ratio of users covered by the top n password.

Number of top passwords	Covered percentage of users
3	0.9%
10	1.6%
100	4.4%
1000	13.2%
10000	30%

A further complication is that users have many passwords for computers, networks, and web sites. The large number of passwords increases interference and is likely to lead to forgetting or confusing passwords. Users typically cope with the password problem by decreasing their memory load at the expense of security.

First, users write down their passwords. Second, when they have multiple passwords, they use one password for all systems or trivial variations of a single password. As a result, users are known to ignore the recommendations on password choice. Two recent surveys have shown that users choose short, simple passwords that are easily guessable, for example, ”password”, personal names of family members, names of pets, and dictionary words. To users, the most important issue is having a password that can be remembered

reliably and that can be input quickly. They are unlikely to give priority to security over their immediate need to get on with their real work.

There are several ways in which an intruder can attack password-protected systems. The most common used are:

- Brute force attacks
- Dictionary attacks
- Replay attacks
- Phishing attacks
- Shoulder surfing

Brute Force Attacks: In this type of attack, all possible combinations of a password are tried to break the password. The brute force attack is generally applied to crack the encrypted passwords where the passwords are saved in the form of encrypted text. Early Linux systems use MD5 hashing schemes for storing the passwords. Passwords are usually stored in a file or a registry in the operating system, together with the user names. If the file is stolen by the attacker then the password can be checked. The original password is not in the file but it is encrypted in the form of a hash (MD5 or other). The encrypted password seems to be safe but, in fact, it is also vulnerable to brute force attack. For this, the attacker first converts all combinations of passwords into their MD5 hashes. In order to break the password, the attacker first extracts the MD5 hash of suspected password from the password file placed in the system. The hash is then matched with all MD5 hashes one by one. When the hashes are matched, the corresponding password is found.

Brute force attacks are very time and processor consuming, as it requires trying all the combinations and calculating the corresponding hash. For example, a user enters a password of 8 characters and all characters are lower case letters then to break the password using the brute force attack it requires $26^8 = 208827064576$ combinations. If a single computer takes 1000 passwords to check in one second then total time will be

$frac2088270645761000 = 208827064.576$ seconds which is equal to 58007.52 hours. This shows that brute force attack is effective for smaller passwords [11].

Dictionary Attack: This type of attack is relatively faster than the brute force attack. Unlike checking all possibilities, the password combinations are matched to a dictionary with the most occurring words or words of daily life usage. Many users generally write passwords related to the names of birds, familiar places, famous actors names etc. These passwords can be judged by the dictionary attack. The attacker makes the dictionary of most commonly used words that might have been be used as a password. The attacker then applies all these words to break the password. Although the dictionary attack is faster than brute force attack, it has some limitations too. It relies on a limited set of words and sometimes it is impossible to crack the password because it is not be present in the dictionary.

Replay Attacks: Also known as reflection attack, it is a way to attack the challenge response user authentication mechanism (same type of protocols by each sender and receiver side). The method for this type of attack is that the attacker first enters her name in the first login connection. To authenticate the user, the receiving device sends the challenge to the sender (in this case attacker). The attacker opens another login at the same time with its own valid user name and replies the receiving device as challenge of previous connection. The receiving side accepts the challenge and responds to it. The attacker then sends back that response through the account to be hacked and thus it gets authenticated. Then the attacker gets access to that account.

Phishing Attacks: It is an attack in which the attacker redirects the user to the fake website to get passwords/PIN codes of the user. To explain phishing, suppose a user wants to open a website, such as "www.yahoo.com". The attacker redirects the user to another website e.g. "www.yah0o.com" whose interface is similar to that of the original website to trick the user. The user then enters the login information which is retrieved by the attacker. The attacker then redirects the user to the original website and logins the user with the original website. Different phishing control filters are used nowadays but still they are not much reliable.

Shoulder surfing: Refers to a direct observation of PIN by looking over a person's shoulder or camera-based recording. The entry of a password can easily be observed in crowded place by standing next to someone [12].

There are two types of passive adversaries. The shoulder-surfing attacker is a weaker adversary whose capabilities are confined to those of a human. On the other hand, the camera-based recording attacker is a stronger adversary equipped with automatic recording devices Since PINs are so popularly used in, smartphones, automated teller machines (ATM), and Point of Sale (PoS) terminals. There is a great need for a secure PIN entry scheme that does not significantly sacrifice usability [13].

Nowadays nearly all computer systems store password in encrypted form. So, password is a key to a cryptographic system. Its length is directly proportional to its security (keys are more secure as they grow longer). However, a longer password it is still not strong password as should be (Table 2.2).

Table 2.2: Number of keys possible with various password lengths and character set constraints

Character Set	4 octet	5 octet	6 octet	7 octet	8 octet
Lowercase letters (26)	$4.6x10^5$	$1.2x10^7$	$3.1x10^8$	$8.0x10^9$	$2.1x10^{11}$
Lowercase letters/digits (36)	$1.7x10^6$	$6.0x10^7$	$2.2x10^9$	$7.8x10^{10}$	$2.8x10^{12}$
All alphanumeric characters (62)	$1.5x10^7$	$9.2x10^8$	$5.7x10^{10}$	$3.5x10^{12}$	$2.2x10^{14}$
Printable characters (95)	$8.1x10^7$	$7.7x10^9$	$7.4x10^{11}$	$7.0x10^{13}$	$6.6x10^{15}$
7-bit ASCII characters (128)	$2.7x10^8$	$3.4x10^{10}$	$4.4x10^{12}$	$5.6x10^{14}$	$7.2x10^{16}$
8-bit ASCII characters (256)	$24.3x10^9$	$1.1x10^{12}$	$2.8x10^{14}$	$7.2x10^{16}$	$1.8x10^{19}$

The time is also proportional to the length of the password (Table 2.3). Clearly, longer passwords provide better protection than shorter ones. Additionally, passwords that use a wider combination of possible bit combinations are better than ones that are highly constrained.

From this tables we can conclude that any password that people will memorize and type in on a regular basis will not be as good as a 64-bit random number. Therefore, passwords will be open to guessing attacks of one form or another [9].

A possible approach to simplify the memorability of passwords can be through the use

Table 2.3: Amount of Time to Search All Possible Keys

Character Set	4 octet	5 octet	6 octet	7 octet	8 octet
Lowercase letters (26)	0.5 sec.	12 sec.	5.2 min.	2.2 hours	2.4days
Lowercase letters/digits (36)	1.7 sec.	1 min.	36.7 min.	21.7 hours	32.4 days
All alphanumeric characters (62)	15 sec.	15 min.	15.8 hours	40.5 days	27 years
Printable characters (95)	1.4 min.	2.1 hours	8.6 days	2.2 years	209 years
7-bit ASCII characters (128)	4.5 min.	9.4 hours	50.9 days	17.8 years	2283 years
8-bit ASCII characters (256)	1.2 hours	12.7 days	8.9 years	2283 years	570,776 years

of pictures or graphical sequences, promising the same degree of security.

2.4 Graphical Passwords

Graphical passwords were originally described by Blonder [14]. In his description of the concept, an image would appear on the screen, and the user would click on a few chosen regions of it. If the correct regions were clicked in, the user would be authenticated [15].

Graphical password is an alternative to alphanumeric passwords in which users click on images to authenticate themselves rather than typing alphanumeric words. Graphical passwords are more memorable compared to the alphanumeric passwords, because it is easier to remember an image than a set of letters and numbers.

Using images instead of characters will help the user to improve the security because the size of the corpus is unlimited, in alternative to the 26 letters and 10 numbers in the case of alphanumeric password [16].

Graphical based password techniques have been proposed to solve the limitations of the conventional text based password techniques, because pictures are easier to remember than texts. It is referred as "Picture superiority effect". A literature survey of papers regarding graphical password techniques shows that the techniques can be categorized into four groups (Figure 2.2).

A. Recognition-Based Technique In this category, users will select images, icons or symbols from a collection of images. At the time of authentication, the users need to

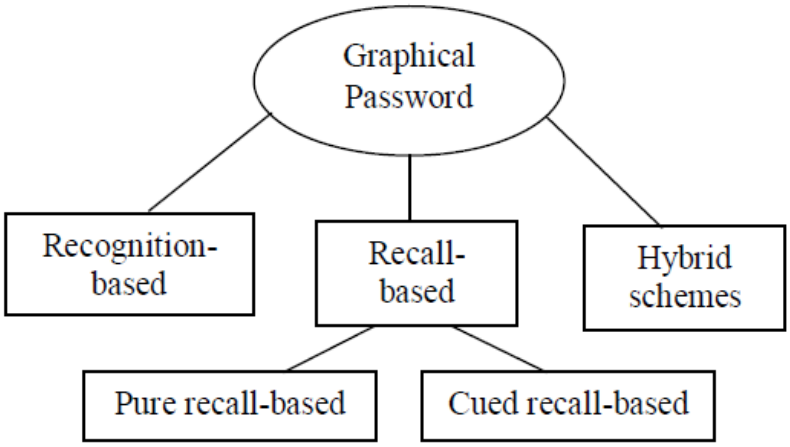

Figure 2.2: Categorization of graphical password authentication techniques.

recognize their images, symbols or icons which are selected at the time of registration among a set of images. Researches were done to find the memorability of these passwords and it shows that the users can remember their passwords even after 45 days.

B. Pure Recall-Based Technique In this category, users have to reproduce their passwords without being given any type of hints or reminder. Although this category is very easy and convenient, but it seems that users can hardly remember their passwords. Still it is more secure than the recognition based technique.

C. Cued Recall-Based Technique In this category, users are provided with the reminders or hints. Reminders help the users to reproduce their passwords or help users to reproduce the password more accurately. This is similar to the recall based schemes but it is recall with cuing.

D. Hybrid Schemes In this category, the authentication will be typically the combination of two or more schemes. These schemes are used to overcome the drawbacks of a single scheme, such as spyware, shoulder surfing and so on [17].

Recognition-based systems are also known as cognometric systems. These systems

generally require that users memorize a portfolio of images during the process of password creation and, when logged in, the users must recognize their images from decoys. An exceptional ability of humans is to recognize images previously seen, which is making recognition based algorithms popular. Various recognition based systems have been proposed using different types of images, mostly like faces, icons, everyday objects, random arts, among others [18].

Pure recall-based graphical password systems are also referred to as drawmetric systems because users recall an outline drawing on a grid that they created or selected during the registration phase. In these types of systems, users usually draw their password either on a grid or on a blank canvas. Memorability is difficult because the retrieval is done without any reminders or clues [19].

Cued-recall systems are also known as locimetric systems as it related to identifying specific locations. These systems typically require the users to remember and click on specific locations within an image. This increases the memorability as it is easier to memorize than pure recall based systems. This is a different memory task than simply recognizing an image as a whole. In these types of schemes, users are provided with an image so that they can choose points arbitrarily by clicking in the presented image as a password. For successful login, the user has to click on right click points in the correct order [16].

Hybrid schemes are the combination of two or more graphical password schemes. These schemes are introduced to overcome the limitations of a single scheme, such as hotspot problem, shoulder surfing, spyware, etc. Many single schemes on recognition-based and recall-based schemes are discussed and some of these schemes are combined to develop the hybrid schemes [16], [17].

In order to understand how users interact with this type of authentication mechanisms a study was made comparing the use of graphical and textual passwords [20]. The success rate for creating a password is higher with the text based password (Figure 2.3). In addition, the amount of time to create a password is higher for graphical passwords (Figure 2.4).

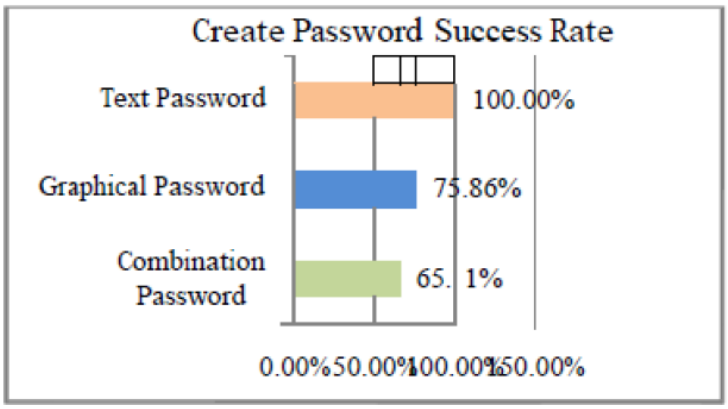

Figure 2.3: Create graphical password rate.

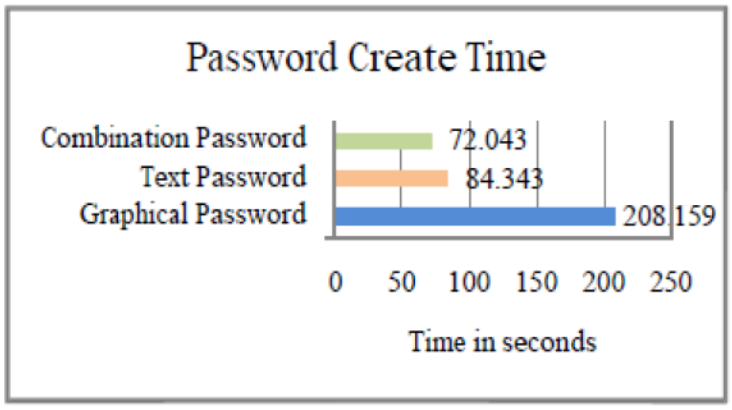

Figure 2.4: Password create time.

And the final graph shows for each system the percentage of users, who successfully logged in.

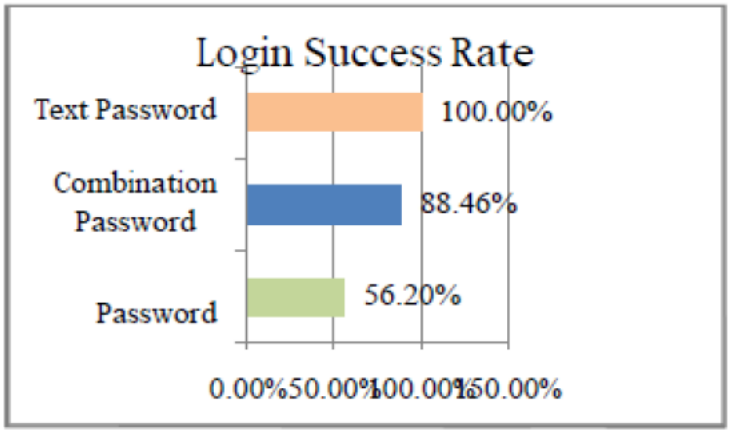

Figure 2.5: Login success rate

According to the results, the textual passwords are more easy to implement than graphical passwords. But in another to be complex to mak e them stronger to resist an attack, user tend to store the password in an insecure way. The advantages of graphical passwords are, that they are making passwords more human-friendly, increasing the level of security and dictionary attacks are infeasible. And by using it with a multi-level authentication option, security of the accounts will increase.

2.5 Other authentication mechanisms

2.5.1 OpenID

OpenID is an open framework, decentralized, a free infrastructure for a user's digital identification, which is built on the basis of Internet technologies, such as HTTP, SSL and URI. The main idea of OpenID is that a person is identified by the URI as a personal identifier that he can control. Moreover, an OpenID account can be used to log into any site that supports OpenID logins.

Development of OpenID was started by Brad Fitzpatrick of LiveJournal but is now being maintained by a community as open source software. The community gets financial and legal support by the OpenID Foundation. Because a large number of organizations, such as AOL, Microsoft, Sun and Novell, are providing OpenID support for its members, the OpenID community claims that there are over 160 million OpenID enabled URIs and nearly ten-thousand sites supporting OpenID logins. The 2.0 version of the specification will among other things support the Yadis protocol increasing its coverage even further.

OpenID defines only how the website finds and exchanges information with the identity provider responsible for the user. The process of how the identity provider actually identifies the user goes beyond the specification. This makes the framework vulnerable for phishing attacks. On the other hand, however it allows OpenID providers to implement exceptional authentication mechanisms that are more secure than password-based logins,

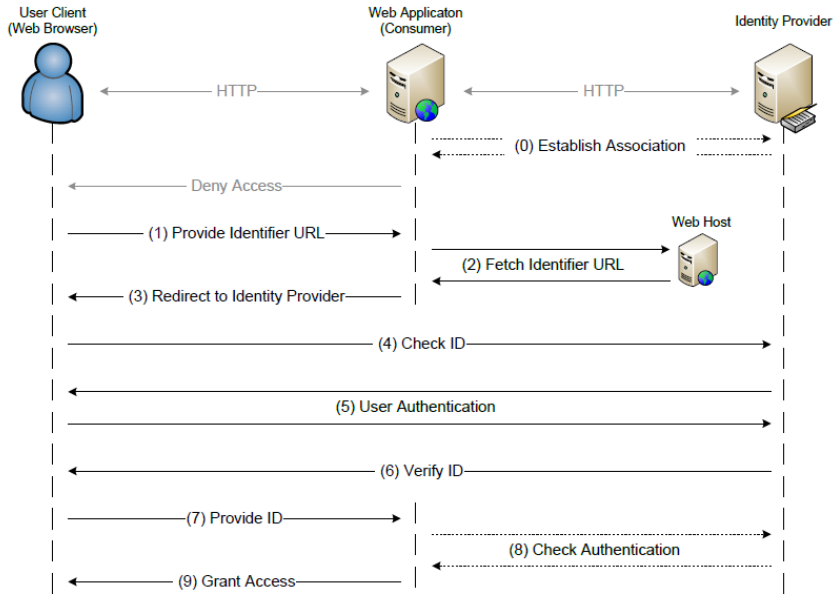

Figure 2.6: OpenID login protocol flow.

such as client certificates or even more exotic approaches like image-based authentication [21].

OpenID follows a specific flow (Figure 2.6. If a user tries to access a web page that requires him to login, the web application displays an HTML form, requesting the user to supply his OpenID identifier. The login process is started when the user submits his OpenID identifier, which is a simple URL (1). The web application, or consumer, according to the OpenID specification, retrieves the document at the indicated URL. The document is then parsed for information about the OpenID identity provider which is responsible for this URL (2). Now the web application knows which identity provider is responsible for the particular user and it responds with a HTTP redirect message to the original user request (3). The redirect message sends the user agent to the web server of the identity provider (4). The message also includes parameters that specify the web site the user should be redirected back to once the authentication is complete, the claimed identifier the user has submitted and the URL of the web application which requested the authentication. At this point the identity provider will authenticate the user using an arbitrary authentication method bidirectional with the user (5). If the authentication

is successful, the identity provider sends another HTTP redirect message to the user, delegating him back to the original web site (6). This results in a new request for the page the user wanted to access to begin with. Only this time it includes authentication information4 from the OpenID identity provider that the web application can use to verify the identity of the inquiring user (7). After the application has verified the authentication information, it allows the user access to the requested resource (9).

The verification of the user supplied authentication information can happen in two different ways. First, the web application uses a dedicated HTTP request to the identity provider to ask if the authentication message in question is valid (8). This option is called dumb mode by the OpenID specification. It has to occur in each authentication run but allows the consumer web application to be stateless in regard to the OpenID authentication process. The second option is for the web application to establish a so-called association with the OpenID provider. This is done independently from authentication runs (0) and is used to negotiate a shared secret between the identity provider and the web application. Until the shared secret expires, it is used in subsequent authentication runs to sign and verify authentication messages. This option called smart mode allows the consumer web application to directly declare authentication identifiers supplied by the user in step (7) as valid. Its drawback is that the web application is required to maintain those shared secrets individually for every identity provider it wants to use this mode with [22].

2.5.2 Kerberos

Kerberos is a network security protocol originally developed by the Massachusetts Institute of Technology (MIT). Meanwhile it is sported and promoted by the Kerberos Working Group of the Internet Engineering Task Force (IETF). The latest version of Kerberos (V5) is specified in RFC 4120 [23]. Kerberos implements an approach to network authentication. After a successful authentication against the Authentication Service, the Kerberos client receives a Ticket-Granting-Ticket. Using this ticket, it can request service specific

tickets from the Ticket-Granting Server. Tickets related to services can then be used to access a specific service. Using this approach to tickets, Kerberos supports single sign-on for all supporting services [24].

Because Kerberos must be supported by the client and the server, it is not present in normal web environments. There is some support for Kerberos as the authentication of the web server, although there are several projects to support Kerberos authentication in browsers. In September 2007, the MIT announced the launch of the Kerberos Consortium. With prominent founding sponsors such as Google, Stanford University, Sun Microsystems and the University of Michigan its goal is to advance the propagation of Kerberos. Their plans also include web authentication [22].

2.6 Apache Shiro

Apache Shiro is a powerful and easy to use security framework written on Java. It performs authentication, authorization, cryptography and session management which can be used to protect most application, like command line applications, mobile applications, largest web and enterprise applications [25].

Shiro provides the application security Application Programming Interface (API) to perform the following:

- Authentication - proving user identity, often called user 'login'.
- Authorization - access control
- Cryptography - protecting or hiding data from eavesdrop
- Session Management - per-user time-sensitive state

As was mentioned before, Apache Shiro is a framework implemented in Java language that provides both authentication and authorization in an simple API. One of the big advantages of Shiro is, that enables to implement application security without the need of coding all functionality from scratch.

The design objectives of Apache Shiro are to simplify application security, by being intuitive and easy to use. Shiro's core design models how most people think about application security.

At the highest conceptual level of architecture, Shiro has 3 basic concepts: the Subject, SecurityManager and Realms (Figure 2.7) [26].

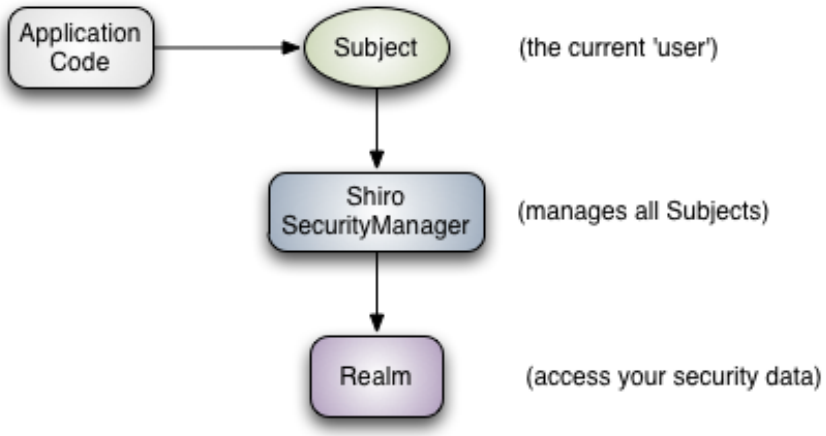

Figure 2.7: Shiro basic architecture.

The Subject is a security specific 'view' of the currently executing user. The word 'User' usually mean a human being, a Subject can be a person, but it could also represent a 3rd-party service, daemon account, cron job, or anything similar - basically anything that is currently interacting with the software. All Subject instances are tethered to (and require) a SecurityManager. When interacting with the subject, those interactions are converted to specific user interaction with the SecurityManager.

The SecurityManager is a core part of Shiro's architecture and aggregates its internal security components in a form of an graph. Usually the SecurityManger and internal graphs are configured once for the application and application developers spend almost all of their time on the Subject API.

Realms are like a 'bridge' or a 'connector' between Shiro and the application's security data. When applications need to authenticate (login) and authorize (access control) users, Shiro checks for information needed from one or more Realms that are configured for the application.

Basically a Realm is a security-specific Data Access Object (DAO): it contains connection details for data sources and retrieves required information to the Shiro as necessary. During Shiro configuration it must be specified at least one Realm to use for authentication and/or authorization. The SecurityManager requires one Realm to be configured, and supports multiple Realms configuration.

Chapter 3

Proposal

As mentioned before, graphical passwords authentication can be a solution for some of the problems discussed earlier. Two possibilities are:

- recognition-based method and
- recall-based method.

In case of the recognition-based method, the user have to recognize previously chosen or given pictures or figures. While when using recall-based methods the user has to draw a pattern or click a certain places of the screen. In addition, it is worth mentioning that dictionary attacks are unfeasible, because there are no pre-existing searchable dictionaries for graphical passwords. It will also be harder to make phishing and reply attacks.

Our proposal suggests integrating a new challenge authentication method on an AAA service that can be used by generic applications. The user sends an authentication request to the web application, which will forward it to the AAA server which will use our graphical user authentication challenge (Figure 3.1).

The AAA server will be based on the Apache Shiro framework. It is powerful and easy to use framework, which performs authentication, authorization, cryptography, and session management.

Based on the survey we can se a comparison of described categories (Table 3.1).

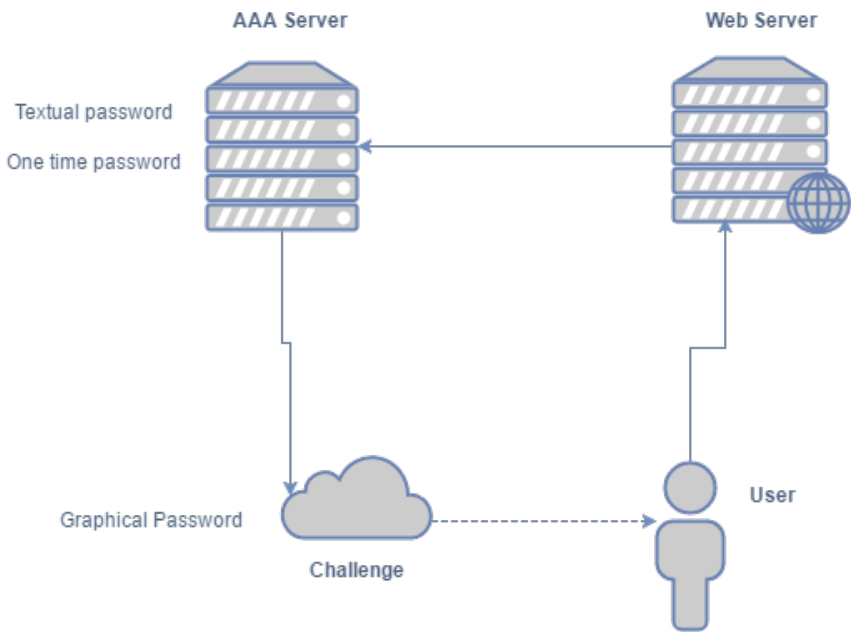

Figure 3.1: Diagram of challenge.

Table 3.1: The attack Comparison in categories of graphical passwords.

Graphical auth. methods	Dictionary Attack	Guessing Attack	Shoulder Surfing	Social Engi-neering	Reply Attack	Brute Force Attack
Recognition Based Method	No	Yes	No	No	No	Yes
Recall Based Method	No	No	Yes	Yes	No	No

Dictionary attacks on graphical authentication is unfeasible because there are no pre-existing searchable dictionaries for the graphical password methods.

Guessing attacks is impossible to do on recall based authentication method, because the possible combinations of correct sequence is equal of the factorial of the total pixels counts. For recognition based authentication method, implementing a guessing attack is more feasible, although it would require a long time.

Shoulder surfing attacks are made more difficult, because the icons usually changes their places every time. Unfortunately, we can not tell the same about recall based

method.

Social engineering is practically impossible in graphical passwords, as keyboard input is not involved, so words in dictionary cannot be used to crack the password [27]. But if attacker will have information about victim, this attack can be implemented for recall based authentication method. Because the attacker can generate for the victim his authentication image.

The brute force attack is a method that finds the password by inputting possible password combinations one by one. The number of possible password combinations of the proposed scheme is influenced by the grid size and the password length. As the grid size of the implemented prototype is 4x4, the result for minimal sequence (which is equal to 4) will be as shown in equations 3.1 and 3.2 [28].

$$A_n^k = \frac{n!}{(n-k)!} \tag{3.1}$$

$$A_{16}^4 = \frac{16!}{12!} = 43680 \tag{3.2}$$

Graphical authentication systems are a kind of knowledge-based authentication systems, since they rely on something that only the user knows. It is much easier to remember pictures and icons rather than sequences of characters, because during the evolution of humans, we developed the ability to memorize places, faces or signs. Characters only became a part of the life of an average person only in the last few centuries.

This type of systems are resistant to reply attacks. The resistance to reply attack means [29] that even if the attacker has passed the authentication message, he can not use them to authenticate it in a different time. Basically because authentication challenge generated by the server uses sequence numbers and time stamps controlled on the server side making unfeasible to generate a correct answer based on the past answers.

In addition, dictionary attacks are unfeasible, because there are no pre-existing searchable dictionaries for the graphical password methods. However, in some cases it is possible to write one as we will see in case of the Android pattern lock [16].

3.1 Recognition based method

Recognition-based methods require the user to recognize previously chosen or given pictures or figures (Figure 3.2). For example, a 4x4 matrix with the images is randomly generated with several images, divided by categories. This means that the user should remember the category, instead of the image. For example, first choose nature, than human than nature again.

Figure 3.2: Recognition based method example

The user has the possibility to choose not only the same category several times, but also the same image. The password length has no limit, so the user can choose whatever images he wants.

The algorithm works as following (Figure 3.3):

1. On the first step, the client requests, to the server, to start a registration process.

2. When the server gets the request from the user, it randomly generates a matrix with images or pictures. The pictures are stored by categories in a database.

3. The generated matrix appears on the user's screen, giving him the possibility to choose his password sequence.

4. The server stores the category sequence (password) in the database.

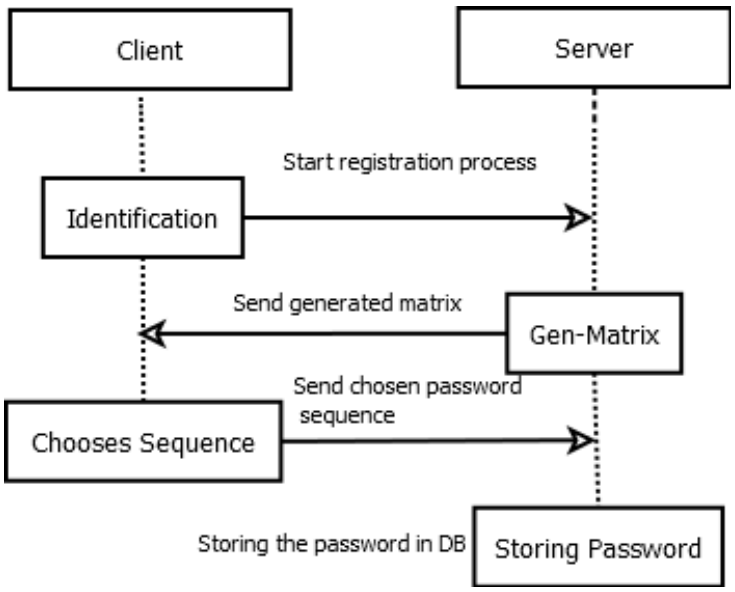

Figure 3.3: The diagram for setting password

The user validation process is described as follows (Figure 3.4):

1. On the first step, the client requests the server for a protected resource.

2. The server sends a new generated matrix with images. By generating a new matrix every time, the problem of shoulder surfing is minimized.

3. After the matrix appears, the user inputs the password by selecting the proper images.

4. In the end, the server receives the sequence from client, compares it, and validate the user.

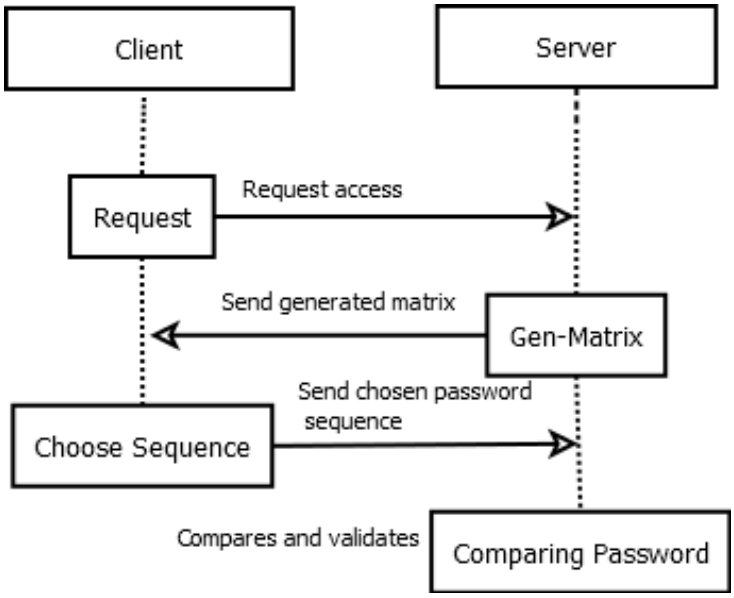

Figure 3.4: The diagram for validation

As we can see, the diagram of validation is similar with the diagram for choosing passwords. The only difference between them is that the user sends a request for validation and in the end of the all process instead of storing password, the server making a comparison between a password that is on DB and the one that was just given.

3.1.1 Recognition based implementation

The recognition based method, like many other authentication methods, requires the user to have an identifier (or username). So, on the first step, the client sends the username to the server and initiates a session on the server side (Algorithm 1).

Algorithm 1 Identification algorithm.

```
1: function IDENTIFICATION(username)        ▷ Giving ability for getting a user name
2:     sessionUser ← username STORE(sessionUser)
3: end function
```

After receiving the username of an user, the server randomly generates a 4x4 matrix with 16 images, selected among several categories stored in the database (Algorithm 2).

After the matrix appears, the user clicks on the preferred images for choosing the

Algorithm 2 Generate matrix.

```
    function GenMatrix(i, j)          ▷ The function generates 4x4 matrix with JPG
        X[i,j] = 0
2:      for ( do a = 0; a < i; a++)
            for ( do b = 0; b < j; b++)
4:              X[a, b] ← JPG
            end for
6:      end for
        return X[i,j]
    end function
```

password sequence. In the end of the process, the client sends the information to the server (Algorithm 3). Before storing the password received from the client, the server has to process it (Algorithm 4). It finally stores it (Algorithm 5).

Algorithm 3 Choosing the sequence.

```
    function Choosing_Sequence(Grid, i, j)          ▷ The function is giving ability to
    choose sequence of password Print(Grid)
        while NotFinish do
3:          Pos ← GetPos(JPG)          ▷ Getting the images which are chosen by user
            Append(Sequence,Pos)
        end while
6:  end function
```

Algorithm 4 Generate the result.

```
    function Gen_Cat(Chosen_Seq,Gen_Matrix)  ▷ Sorts the sequence by categories
        Result ← NULL
        for i = 0; i <= SizeOf(Chosen-Seq); i++ do
4:          [line, col] ← Chosen_Seq[i]
            Store_Cat ← Gen_Matrix[line, col]
            Result ← Append(Result, Store_Cat)
        end for
        return Result              ▷ The sequence of password presented with categories
8:  end function
```

On the other hand, the user validation follows a similar process. The client starts by sending a request for a protected resource, which makes the server start an authentication process. The server generates the matrix and receives the sequence input by the user.

Algorithm 5 Storing the password.

function STORE-PASS($Result, SessionUser$) ▷ Calculates hash value for each user
$Hash \leftarrow$ CALC-HASH($Result$) STORE(Hash,SessionUser)
end function

Then, it compares the sequence (Algorithm 6). The result, either success or failure, is finally returned to the client.

Algorithm 6 Comparing the password.

function COMPARE-PASS($Result, SessionUser$) ▷ This function is responsible for final validation
$Pass \leftarrow$ GEN_CAT($Result$)
$Hash \leftarrow$ CALC-HASH($Pass$)
$Stored - Pass \leftarrow$ GET-PASS-FORUSER($SessionUser$)
if Hash == Stored-Pass **then**
return Success
else
return Fail
end if
end function

3.2 Recall based method

In case of using recall-based methods, the user has to draw a pattern or click in a certain set of places of the screen. This method of authentication has a wide variety of possible implementations. In our variation of recall based authentication method, we are giving to the user an option to upload the photo for making an authentication. This allows remembering the password easier. The main operation of this authentication process is to choose points on the picture and remember the sequence. The user can choose as many points as desired (Figure 3.5).

In our challenge, it will be implemented as follows (Figure 3.6):

1. The client initiates a registration request with the server.

2. The server allows the user to upload an image to use for his password.

Figure 3.5: Recall based method example.

3. The client uploads the image for authentication.

4. After this, the server sends him an instruction to chose the sequence of points on the image.

5. The user chooses points on the image and these are forward to the server.

6. The server gets the sequence and stores it on a database.

The validation of the user words as follows (Figure 3.7):

1. On the first step, the client sends a request for a protected resource.

2. After receiving a client request, the server sends the image received from the user during registration.

3. On the third step, the user selects the sequence of points and the client sends it to the server.

4. After getting the sequence of points from client, the server checks if it is valid or not.

This method makes the password more user-friendly by giving an opportunity to user to upload a specific image, making password more easy to remember.

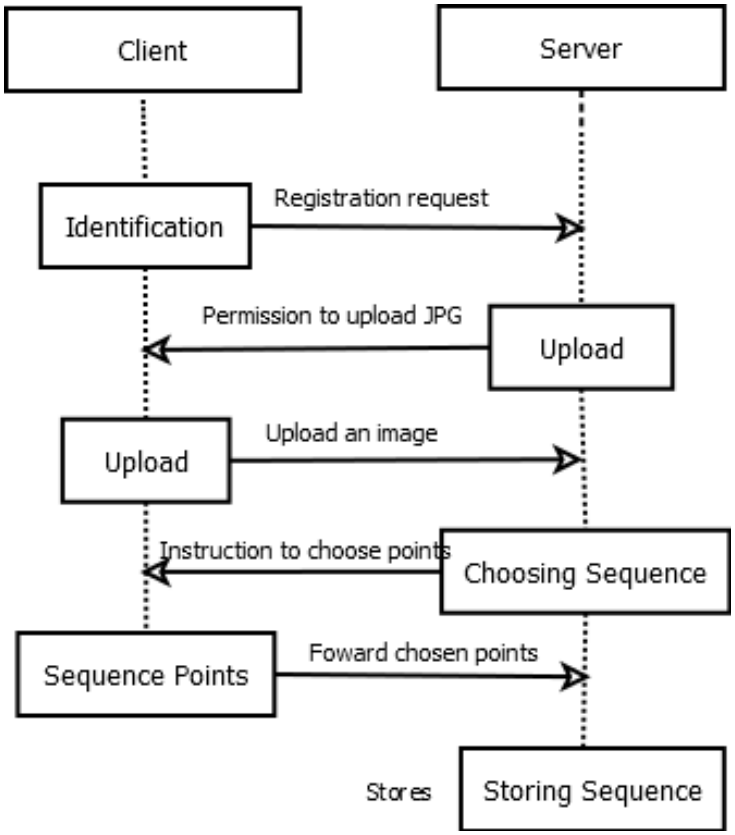

Figure 3.6: Diagram of setting password.

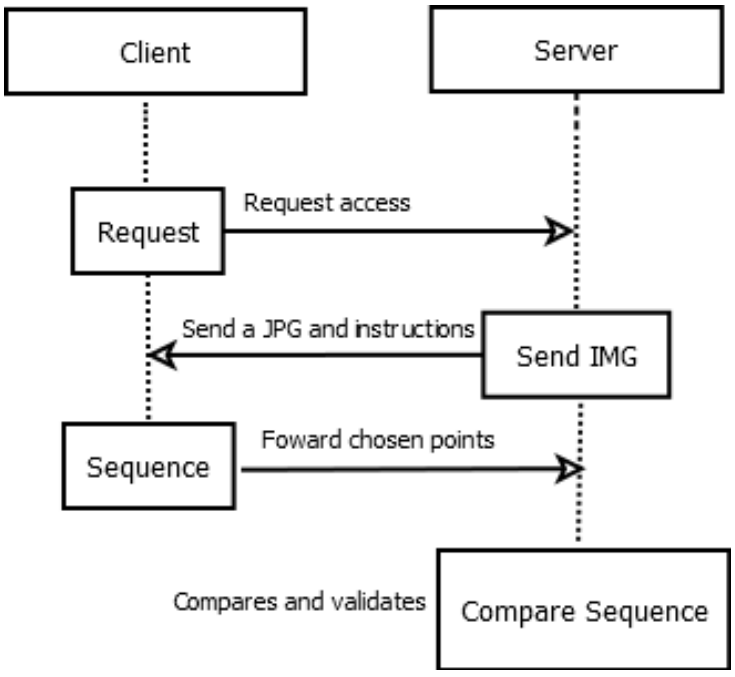

Figure 3.7: Diagram of validation

3.2.1 Recall based implementation

On the first step of recall based authentication, the client initiates a registration process, which makes the server request the username (or identification) (Algorithm 7).

Algorithm 7 Request of username.

```
1: function IDENTIFICATION(UserName)        ▷ Giving ability for getting a user name
2:     SessionUser ← UserName STORE(SessionUser)
3: end function
```

After the identification process, the server sends a request to the user and gives him permission to upload an image, which is going to be used for graphical authentication. By getting the request from the server, the user uploads his preferred image for authentication and sends it to the server (Algorithm **??**).

Algorithm 8 Storing authentication JPG

```
   function UPLOAD(Auth-JPG,SessionUser)        ▷ Giving ability to upload a JPG
       STORE(Auth-JPG,SessionUser)        ▷ Saving JPG in the DB for current user
       return Success
2: end function
```

The server receive an image from client and stores it under the client's username. Now, the server sends to the client a request to choose his password (Algorithm 9).

Algorithm 9 Choosing sequence.

```
   function CHOOSING-SEQUENCE(Auth-JPG)        ▷ The function is giving ability to
   choose sequence of password
       PRINT(Auth-JPG)
       while NotFinish do
3:         Pos ← GETPOS(JPG − point)        ▷ Gets the position of the chosen points
   APPEND(Sequence,Pos)
       end whilereturn Sequence
   end function
```

By getting an instructions from server, the client selects the sequence of his password and send it to the server that it will store (Algorithm 10).

For making a validation, the client sends the request of authentication to the server. By getting the user name of client, server sends the image of user (Algorithm 11).

Algorithm 10 Storing the password.

function STORE-PASS(Sequence,Auth-JPG,SessionUser) ▷ Calculates hash value for each user
$Hash \leftarrow$ CALC-HASH($Sequence$) STORE(Hash,Auth-JPG,SessionUser)
end function

Algorithm 11 Send the challenging image.

function SEND-IMG(SessionUser) ▷
$Auth - JPG \leftarrow$ GET-IMG($SessionUser$) ▷ The function is getting the authentication image from DB by the user name
return Auth-JPG
end function

After receiving the image, the client will input the sequence of the password and sends it to the server (Algorithm 12).

Algorithm 12 Comparing the password.

function COMPARE-PASS(Sequence,SessionUser) ▷ This function is responsible for final validation
$Hash \leftarrow$ CALC-HASH($Sequence$)
$Stored - Pass \leftarrow$ GET-PASS-FORUSER($SessionUser$)
if Hash == Stored-Pass **then**
return Success
else
return Fail
6: **end if**
end function

After receiving the password, the server compares it with the stored one and sends to the client success or fail.

Chapter 4

Discussion

In our work, for improving the security of web applications, we put focus on the authentication. The main user concerns are the risk of shoulder surfing, encryption of graphical password and entering passwords, that should be performed quickly and effortlessly. On the other hand, most of them believe that graphical passwords would provide for a more secure user experience [20].

According to the results of the research, graphical passwords are a promising way to make authentication less vulnerable. Complementing this with a multi-level authentication option, security of the accounts can even increase. The advantages of graphical passwords are more human-friendly, increases the level of security and dictionary attacks are infeasible.

By using our challenge, web developers will have opportunity to also use graphical passwords and use them as a single or multi-level authentication. The main advantage of this it that it does not put an upper limit on the password length and it is possible to use with a multi-level authentication option.

The goal of the recognition based authentication method is that it always randomly generates a new matrix with different images. Due to this, shoulder surfing attacks are unfeasible and other attacks normally used against textual password authentication are minimized because of the dynamic nature of the challenge. In time, adding new images and categories increases the possibility of combinations making the system more safer.

Beyond this, it is easier to remember the sequence of categories, than remembering all chosen images and positions.

The mapping between images and the categories that they belong is only made on server side. This will protect against some client side attacks.

The goal of recall based authentication method is that it more user-friendly. By giving an option to the user upload image that he prefers, remembering password becoming more easier and also each user can have their own personalized challenge. Next big plus of this method that it does not have limit of number of chosen points, so the attacker doesn't know the size of the sequence. The possible position of each point on the sequence is related with a pixel of the photo, which can be reused on the sequence. These two proprieties combined increases the possible combinations of a correct sequence, making a brute force attack unfeasible.

Chapter 5

Conclusions

During this work we made a research of web application authentication solutions. We identified the risks of textual passwords and the possible attack vectors. We made a research of graphical password authentication methods and made a comparison between them. Due to the research results we propose authentication solutions based on recognition and recall based methods.

Based on the research, we developed an algorithm for graphical authentication which can be used as single or multi-level authentication. As a future work we will integrate this algorithms in Apache Shiro, in order to create the extension and enabling users to use our graphical challenges.

Related with recognition based authentication method, we consider to analyze the matrix, and how many categories will increase the difficulty for attackers and make it comfortable to use.

For improving our proposed method of recall based authentication, we still need to analyze the minimal and maximal resolutions of images which user can upload, and tune the error region when validating the points, considering that should be a dynamic value related with the screen resolution and input device (mouse, finger). Moreover, to further improve the security of recall based method, we are planning to add to the identification algorithm, a validation to check if the user can request this process. The main idea is that when the user uploads the image, on registration process, an hash is calculated to the

image, which the client will sent during the identification phase. Then server will check if the photo stored has the same hash before sending the authentication challenge to the user.

Also we should test the usability of our challenge with different web developing frameworks and CMS (Content Management System) engines.

Bibliography

[1] J. Leyden, *Office workers give away passwords for a cheap pen: Security? what's that? security*, http://www.theregister.co.uk/2003/04/18/office_workers_give_away_passwords, Accessed: 2017-04-20, 2003.

[2] C. Rigney, S. Willens, A. Rubens, and W. Simpson, *Remote Authentication Dial In User Service (RADIUS)*, RFC 2865 (Draft Standard), RFC, Updated by RFCs 2868, 3575, 5080, 6929, 8044, Fremont, CA, USA: RFC Editor, Jun. 2000. DOI: 10.17487/RFC2865. [Online]. Available: https://www.rfc-editor.org/rfc/rfc2865.txt.

[3] CISCO, *Configuring web-based authentication*, https://www.cisco.com/c/en/us/td/docs/switches/lan/catalyst4500/12-2/50sg/configuration/guide/Wrapper-46SG/webauth.pdf, Accessed: 2017-04-12.

[4] R. Fielding and J. Reschke, *Hypertext Transfer Protocol (HTTP/1.1): Authentication*, RFC 7235 (Proposed Standard), RFC, Fremont, CA, USA: RFC Editor, Jun. 2014. DOI: 10.17487/RFC7235. [Online]. Available: https://www.rfc-editor.org/rfc/rfc7235.txt.

[5] R. Shekh-Yusef, D. Ahrens, and S. Bremer, *HTTP Digest Access Authentication*, RFC 7616 (Proposed Standard), RFC, Fremont, CA, USA: RFC Editor, Sep. 2015. DOI: 10.17487/RFC7616. [Online]. Available: https://www.rfc-editor.org/rfc/rfc7616.txt.

[6] J. Reschke, *The 'Basic' HTTP Authentication Scheme*, RFC 7617 (Proposed Standard), RFC, Fremont, CA, USA: RFC Editor, Sep. 2015. DOI: 10.17487/RFC7617. [Online]. Available: https://www.rfc-editor.org/rfc/rfc7617.txt.

[7] J. Franks, P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Luotonen, and L. Stewart, "Http authentication: Basic and digest access authentication", Tech. Rep., 1999.

[8] B. L. Filkins, J. Y. Kim, B. Roberts, W. Armstrong, M. A. Miller, M. L. Hultner, A. P. Castillo, J.-C. Ducom, E. J. Topol, and S. R. Steinhubl, "Privacy and security in the era of digital health: What should translational researchers know and do about it?", *American journal of translational research*, vol. 8, no. 3, p. 1560, 2016.

[9] G. Kessler, *Passwords — strengths and weaknesses*, http://www.garykessler.net/library/password.html, Jan. 1996.

[10] M. Burnett, *10,000 top passwords*, http://web.archive.org/web/20150315154609/, Jun. 2011.

[11] M. Raza, M. Iqbal, M. Sharif, and W. Haider, "A survey of password attacks and comparative analysis on methods for secure authentication", *World applied sciences journal*, vol. 19, no. 4, pp. 439–444, 2012.

[12] O. Kasat, U. Bhadade, and M. N. Trivedi, "Study and analysis of shoulder-surfing methods", 2015.

[13] T. Kwon and J. Hong, "Analysis and improvement of a pin-entry method resilient to shoulder-surfing and recording attacks", *Ieee transactions on information forensics and security*, vol. 10, no. 2, pp. 278–292, 2015.

[14] G. Blonder, *Graphical password*, US Patent 5,559,961, Sep. 1996. [Online]. Available: https://www.google.com/patents/US5559961.

[15] S. Wiedenbeck, J. Waters, J.-C. Birget, A. Brodskiy, and N. Memon, "Authentication using graphical passwords: Effects of tolerance and image choice", in *Proceedings of the 2005 symposium on usable privacy and security*, ACM, 2005, pp. 1–12.

[16] S. Ramanan and J. Bindhu, "A survey on different graphical password authentication techniques", *Volume-2, page*, no. 7, 2014.

[17] K. Renaud and E. Smith, "Jiminy: Helping users to remember their passwords", in *Annual conference of the south african institute of computer scientists and information technologists. saicsit*, 2001, pp. 73–80.

[18] H. Gao, X. Liu, S. Wang, and R. Dai, "A new graphical password scheme against spyware by using captcha.", in *Soups*, 2009.

[19] I. Jermyn, A. J. Mayer, F. Monrose, M. K. Reiter, A. D. Rubin, *et al.*, "The design and analysis of graphical passwords.", in *Usenix security*, 1999, pp. 1–14.

[20] C. Singh, L. Singh, C. Singh, and L. Singh, "Investigating the combination of text and graphical passwords for a more secure and usable experience", *International journal of network security & its applications (ijnsa)*, vol. 3, no. 2, 2011.

[21] D. Recordon and B. Fitzpatrick, "Openid authentication 1.1", *Finalized openid specification, may*, 2006.

[22] N. Neumann and X. Fu, "Diameter webauth: An aaa-based identity management framework for web applications", in *Global telecommunications conference, 2008. ieee globecom 2008. ieee*, IEEE, 2008, pp. 1–6.

[23] C. Neuman, T. Yu, S. Hartman, and K. Raeburn, *The Kerberos Network Authentication Service (V5)*, RFC 4120 (Proposed Standard), RFC, Updated by RFCs 4537, 5021, 5896, 6111, 6112, 6113, 6649, 6806, 7751, 8062, 8129, Fremont, CA, USA: RFC Editor, Jul. 2005. DOI: 10.17487/RFC4120. [Online]. Available: https://www.rfc-editor.org/rfc/rfc4120.txt.

[24] C. Neuman, S. Hartman, T. Yu, and K. Raeburn, "The kerberos network authentication service (v5)", 2005.

[25] N. Good, *Introducing apache shiro*, `https://www.ibm.com/developerworks/web/library/wa-apacheshiro/`, Sep. 2010.

[26] A. Shiro, *Apache shiro architecture*, `https://shiro.apache.org/architecture.html`, Accessed: 2017-04-12, 2008.

[27] H. K. Sarohi and F. U. Khan, "Graphical password authentication schemes: Current status and key issues", *Int. j. eng. innovative technol.(ijeit)*, vol. 10, no. 2, 2013.

[28] T. Kim, J. H. Yi, and C. Seo, "Spyware resistant smartphone user authentication scheme", *International journal of distributed sensor networks*, vol. 10, no. 3, p. 237 125, 2014.

[29] W.-C. Ku and M.-J. Tsaur, "A remote user authentication scheme using strong graphical passwords", in *Local computer networks, 2005. 30th anniversary. the ieee conference on*, IEEE, 2005, pp. 351–357.