Ease.space

Software application Security Whitepaper

2017

1. Introduction
2. Securing personal accounts' passwords
3. Securing your Ease.space Master Password
4. Securing team accounts' passwords
5. Securing communications between client and servers
6. Preventing attacks
7. Diagrams of subscription & authentication flow

1. Introduction

We use Ease.space to store passwords safely and to be able to use them easily anywhere in a secure way. Companies use Ease.space to store, share, and use passwords safely at the team level.

The purpose of this paper is to explain you how our security system works, at the personal and company level, to:

- grant you the highest level of security
- integrate the best and latest security practices
- anonymize your data

2. Securing personal accounts' passwords

A personal account password is a password of any account you store on Ease.space. For example, it can be the password of your LinkedIn account.

a. Step 1: securing first with asymmetric encryption

Info: to understand how we secure your personal account passwords, you need to understand how asymmetric encryption works. Asymmetric encryption is a system with two secret keys. The first one is a public key, it is used to encrypt a password. The second one is a private key, it is used to decipher a password.

The technology we use on this part is called R.S.A. Many well-known security protocols like SSH, OpenPGP, S/MIME, and SSL/TLS rely on RSA for encryption and digital signature functions. It is also used in browsers for example, which need to establish secure connections over an insecure network (like the Internet) or validate a digital signature.

Ease.space uses a public key to encrypt your **Personal Account Passwords** then you use your private key to decipher a password and login to a website. We call the private key: The **User Account Private Key**.

On Ease.space, you have as many **User Account Private Keys** as you have passwords (apps). Thus, securing the way the **User Account Private Keys** are stored and who can access them is important.

b. Step 2: securing a second time with symmetric encryption

Info: all User Account Private Keys is stored encrypted using a different encryption method called: symmetric encryption. Symmetric encryption has one single key that can encrypt and decipher the data. We call this new key that secures User Account Private Keys: The User Master Key.

The technology used to generate the **User Master Key** is **AES-256**. AES-256's quality in proven worldwide. It was originally developed by the National Institute of Standards and Technology (a unit of the US Government).

In 2003, the US Government announced that AES could be used to secure classified information. The **NSA** chose AES as one of the cryptographic algorithms to be used by its Information Assurance Directorate to protect national security systems. Thanks to that, and the fact that it is fully open to public scrutiny and comment, AES is widely used in the private sector and by many security protocols such as Secure Sockets Layer (SSL)/Transport Layer Security (TLS). AES-256 can be found in most modern applications and devices that need encryption functionality.

There is one **User Master Key** for each Ease.space user. That one key deciphers each **User Account Private Keys**. Thus, securing how the **User Master Key** is stored and who can access it is important.

c. Step 3: securing a third time with symmetric encryption and a salt

The (AES generated) **User Master Key** is also stored encrypted on our servers, so that even Ease.space cannot have access to it.

When you type your **Ease.space Master Password** in the morning to login to your space, we use this **password and a salt** (a random chain characters that we add to your password) to create the key to decipher the **User Master Key**. Adding a salt increases the complexity of deciphering, and makes the password stronger.

The technology of step 3 works as explained below:

- We generate a 20 bytes' salt.
- The User Master Password is used, with the salt, to generate the AES-256 bit key.
- Then, the data is (de)ciphered using this AES-256 bit key

d. Using information to connect you

Once the user's data has been deciphered, it is only loaded into our server's **RAM**. For security purposes, the deciphered data is not stored on the Ease.space database itself.

In addition to that, once the **User Master Key** is deciphered, the variable "password+salt" is automatically deleted. The **User Master Key** is also automatically deleted when you log out from Ease.space. Ease.space automatically logs out all users at least every day.

Thanks to **(1)** the **User Master Key** which is unique to each user, **(2)** the communication system between a user's computer and Ease.space servers: a user's data (passwords and other information) is stored only in the **user's session** that is unique to each connected user.

3. Securing your Ease.space Master Password

Now we understand the importance of the **Ease.space Master Password**. This part will be dedicated to explaining how we secure it.

To understand the security level of your Ease.space Master Password, you need to understand another security method called "Hash".

a. Securing Ease.space Master Password storage with "Hash"

Info: encrypting and deciphering a password is done thanks to a key (symmetric encryption) or two keys (asymmetric encryption). Hashing a password works only one way and does not create any key: it transforms a password in something unreadable called a hash.

This method enables Ease.space to store your password without knowing it, and without having a way (a key) to know it.

If we don't know your password; how do we know that you are the one logging in every morning?

b. Enabling anonymous connection

Method: When you create your Ease.space account, you enter your Ease.space Master Password for the first time, we hash it and store the hash in our data base. When you come back the next day, you enter your Ease.space Master Password again to log in. We hash it again and compare the two hash (the one from when you created your account and the one from when you logged in). If the two hash are the same, we can know that it is you.

To increase the security, we add a salt to your **Ease.space Master Password** before hashing it. The salt method was explained during step 3 of this paper.

The result is: you can log in, we know it is you, and we don't know your **Ease.space Master Password**.

c. Using internationally recognized technology

The Hash technology used for securing your Ease.space Master Password is called Bcrypt.
This hashing function has a high number of iterations (+/- 60'000).

Unlike other hashing functions susceptible to rainbow table and brute-force attacks, Bcrypt is very slow. It is an adaptive function, meaning its hash function can be made more expensive and thus slower (to hack) as computing power increases.

Bcrypt uses the Eksblowfish algorithm to hash passwords. The key schedule phase of Eksblowfish ensures that any subsequent state depends on both salt and key (your Ease.space Master Password), and no state can be precomputed without the knowledge of both. Because of this, nobody can retrieve your "plain text" Ease.space Master Password without already knowing the salt and the number of iterations and the key (your password).

4. Securing team accounts' passwords

Passwords of team accounts are needed by multiple people at different times. Therefore, we have created a different encryption system for team passwords that enable different people from the same team to decipher (and then use) the same password while keeping the security level at its highest.

For a personal password, we had:

- User Account Passwords deciphered by User Account Private Keys deciphered by User Master Key deciphered by User Master Password + Salt

For a team password, we have:

- User Account Passwords deciphered by User Account Private Keys deciphered by **Team Master Key** deciphered by User Master Key deciphered by User Master Password + Salt

The **Team Master Key** is unique to a team and is shared among its members. It enables the deciphering of passwords shared among team members by allowing a team member to use anonymously other members' **User Account Private Keys** to decipher passwords.

In addition to that, we created 2 keys called **User Private Key** and **User Public Key**. There are unique for each user. Those two keys are used to send and receive the **Team Master Key** during the process of inviting a new member in the team.

Technically, the secured steps to invite a user in a team and share him/her the account passwords he/she needs, are as follows:

- A team administrator sends an **invitation** to an existing user (or through email for users not registered on Ease.space yet).
- Once the invitation has been **accepted** and the account has been **created**: the team user has a User Master Password, a User Master Key, a User Private Key, and User Public Key.

- The administrator will send the **Team Master Key** to the user by ciphering it with the **User Public Key** of the new team user.
- The user then deciphers the **Team Master Key** with his **User Private Key** and ciphers it again with his **User Master Key** (so that the access to team passwords depends on the user's password).
- The new team user can now use team passwords.

We maintain the security level of team passwords with two aspects:

1. After registration, the **Team Master Key** can only be deciphered by a team user's **User Master Key** (Remember, the User Master Key is only accessible by your password).
2. Access to teams are through **invitations** only with an additional **validation** of the team admin.

The technology used to generate the **Team Master Key** is **AES-256** and the technology behind the User Public Key and the User Private Key is R.S.A.

5. Securing communications between client and servers

a. Why HTTPS

All communications between Ease.space users (the client) and Ease.space servers are secured with **HTTPS** using **SSL/TLS** connections (explained bellow). We chose the **Let's Encrypt certificate**.

Info: There are two main reasons why HTTPS exists: encryption and identification. Encryption: all data transfers that occur between users' computers and our servers must be encrypted to prevent anyone from catching sensitive information on the way (your passwords for example). Identification: making sure that the right computer is speaking to the right server (and that the server can be trusted), and vice versa.

b. How does HTTPS works

- Encrypting data transfers

During the data transfer, HTTPS's process uses asymmetrical encryption and symmetrical encryption and a hashing method.

(1) The client (you) and the server need to agree on the **best encryption method** they both can use at that moment (regarding computing power for example). So, the client send the methods he will use for the key (ex: **RSA**), the cypher (ex: **AES**), and the hash (ex: **SHA**). In addition to that, the client sends a **random number** and the **SSL** version/protocol (ex: 3.3, TLS).

(2) The server sends his certificate to the client. Among other info, the certificate contains the **Server Public Key**.

(3) The client uses the **Server Public Key** to encrypt the random number, and then sends it to the server.

(4) The server deciphers this number with his **Server Private Key**, and both sides use this number to generate a **symmetric key**

(5) The symmetric key is used to cipher and decipher data on both side.

- Verifying identification

First the Ease.space server must get a **HTTPS certificate** from a **Certificate Authority**. To get it, Ease.space had to give various official information to **Let's Encrypt** (who the server belongs to, how long the certificate is valid for, various serial numbers, etc.).

All this info is condensed into 1 string with a hash function. This string is then encrypted using a **Certificate Authority Private Key**. So, anyone having the right **Certificate Authority Public Key** can verify the certificate. Then the certificate is installed onto Ease.space web servers.

A browser already has the certificates from Certificate Authorities from all around the world, installed in it. That enables the browser to check the authenticity of any certificates it receives. In other words, the browser already has the **Certificate Authority Public Key** of any certificate available.

c. Using Let's Encrypt

Let's Encrypt is a service provided by the **Internet Security Research Group (ISRG)** supported and recommended by the **Mozilla Foundation, OVH, Akamai, and Cisco Systems**. It is a certificate authority that provides **Transport Layer Security (TLS)** encryption that creates, validates, signs, installs, and renews the certificates (explained above) for secure websites.

6. Preventing attacks

a. Preventing Brute force and rainbow table attacks

Due to **Bcrypt**, the **salt** method, and given that Ease.space **enforces reasonably strong password requirements**, the hash of the **Ease.space Master Password** is not contained in a dictionary and an attack would be impractical.

The user remains responsible for protecting his own computer from non-authorized access, and for making sure he is not installing potentially infected software.

In that matter, we advise our users to keep their devices up to date and safe by **activating automatic updates** on their operating system, as well as **install & update an antivirus** (we recommend **Kaspersky, Bitdefender or Avira**).

b. Preventing password recovery attacks

As we don't save any version of your **Ease.space Master Password** thanks to the hash method, we can't recover it.

However, we still have a **"password lost" process** (for practical reasons) which gives you back the access to your Ease.space account. But during that process, all your personal passwords **will be lost** and you will have to re-enter each personal password again.

The password recovery system is different for the team accounts. Thanks to the **Team Master Key**, we can give the right to a **Team Administrator** to recover the team passwords for a team member.

Through our interface, you will have to contact an administrator to ask for your access to the team accounts.

Ease.space itself doesn't have any way to access the **Team Master Key**, because it is ciphered with team members' **User Master Keys**. Therefore, to recover your team accounts, you need the Team Administrator (which has access to the **Team Master Key**) to share it to you again.

c. Preventing XSS Attacks

Info: Cross-Site Scripting (XSS) attacks are a type of injection, in which malicious scripts are injected into otherwise trusted websites. XSS attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end user. Flaws that allow these attacks are quite widespread and occur anywhere a web application uses input from a user within the output it generates, without validating or encoding it.

To prevent XSS Attacks, Ease.space first validates that all input forms the user uses only accept the appropriate data type (e.g., **dates can only be a date format**), do not contain scriptable HTML tags, and are stripped of any potential tags, before rendering. In addition, all HTML outputs are encoded to ensure that the browser does not process any unwanted scripts.

d. Preventing SQL Injection

Info: SQL injection is an attack used to either gain access to a database or retrieve information directly from a database. It occurs when an application program: accepts arbitrary SQL from an untrusted source (the "end user"), blindly adds it to the application's SQL, and executes it.

Ease.space avoids this at-all-times by always using bind variables within its SQL queries. A bind variable is a placeholder in a SQL command for a value that is supplied at runtime by the application. Using parameters (or parameter markers) to hold values is more secure than concatenating the values into a string that is then executed as part of a query.

e. Applying same-origin policy to prevent malicious scripts

The same-origin policy is an important concept in web application security. Under this policy, only scripts loaded from the same URL can share the same data.

An origin is defined as a combination of URL scheme, hostname, and port number. This policy prevents maliciously injected scripts to access sensitive data through the Document Object Model (DOM).

e. Security level in a browser

Running security tools within a browser environment brings its own problems. But, over the past decade, browsers have made numerous improvements in their security and in the isolation of their processes.

Sandboxing within the browser provides the first line of defense. Ease.space also Structures its in-browser code to expose only what needs to be exposed.

JavaScript, the language used within the browser, offers us limited ability to clear data from memory. Sensitive information, that we would like the user's computer to forget, may remain in memory longer than useful. **We advise our users to regularly clean their cache, cookies and**
history to keep their browser safe.

Despite our best efforts there is a limit to any security architecture. If a hacker has a full access to your device, he still can make damaging actions regardless if you use Ease.space or not. He could modify your device source code, install malware (such as a Keylogger that saves keystrokes), or edit your Ease.space extension and collect sensitive data.
Therefore, securing your hardware is important for your security level.

f. Solving the SHA1 issue inside AES-256

We use SHA1 as the Pseudo Random function of our Key Derivation to generate a suitable 32 bytes AES Derivated Key from the **User Master Password**.

There is a well-known issue in SHA1 that allows possible attackers to find **collisions**. Collisions are two different preimages (the plain text) that will give the same hash (the

unreadable version of the text).

This vulnerability in SHA1 does not affect the security of Ease.space. In fact, such an attack requires knowing both the preimage and the computed hash. With Ease.space and AES, The result of the SHA1 hash is only used as a value for encrypting the User master Key and is not stored anywhere at any time. It is just used as a computing value.

7. Diagrams of subscription & authentication flow

The initial registration for a user follows the flow described in Figure 1

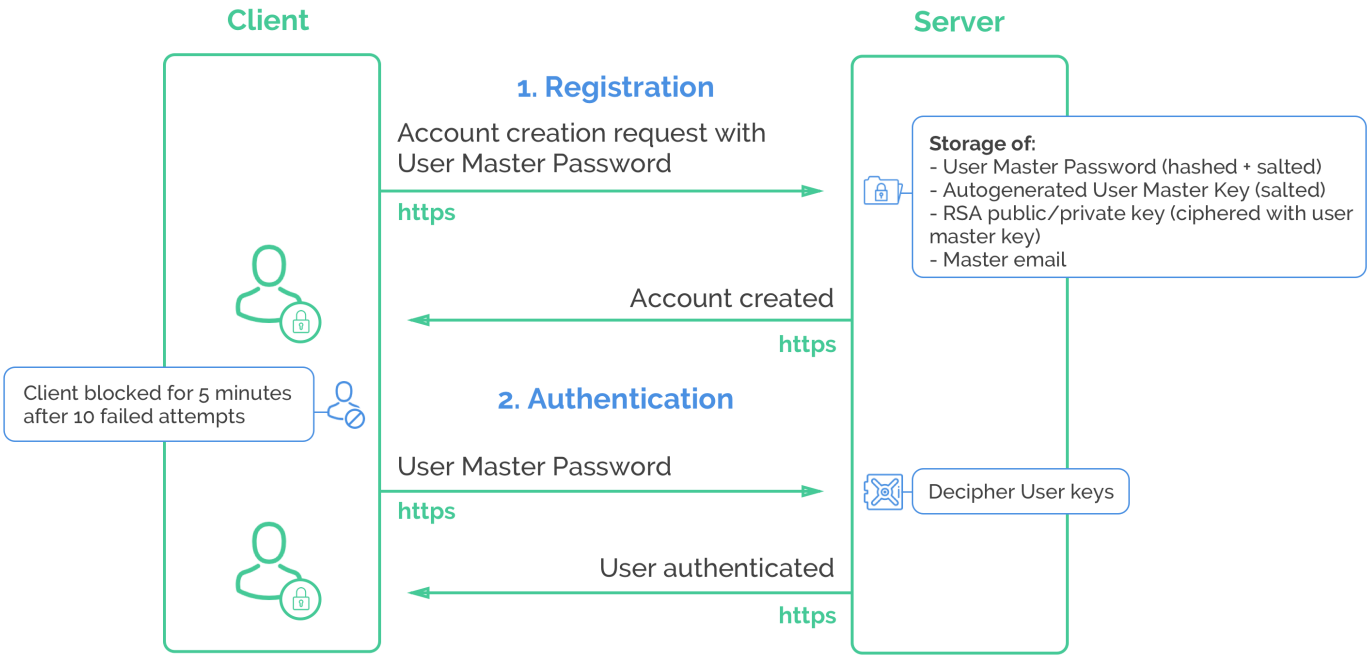

Figure 1: Authentication flow during registration

The subscription of a new team member follows the flow described in Figure 2

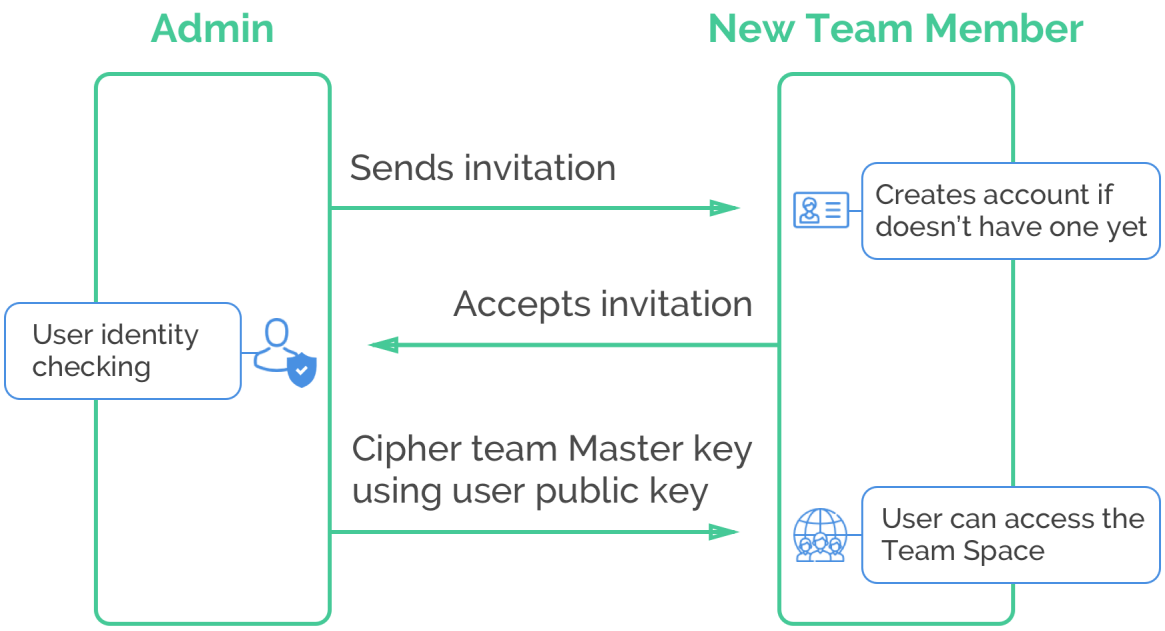

Figure 2: new Team Member subscription flow

Thank you for taking the time to read this paper.

Feel free to contact us for more information.

The Ease.space team
https://ease.space
contact@ease.space
+33 6 26 97 88 97