CYBER SECURITY

SECURE MEDICAL SYSTEM

DUE TO 11/6

Alessandro Spinosi
Guillaume Kerckhofs
Samain Clément
Randi Dochot
Alix Declerck
Professor Romain Absil

Abstract. The goal of this project is to implement a secure client / server system handling patient's medical records, such as the ones available in hospitals.

We install un server using and existing tech combo (from Gerardo Fernandez) which is composed by docker to run the server, Nginx the server itself, Symfony 6.2 Boilerplate for the applicative framework and we change the database to mariadb

We create a client using npm run dev from the Node Package Manager

Contents

1. Project description

Installations procedures can be find here :
`https://github.com/Randi-Dcht/project_security_web`

Our project handle a server (backend) and a client (frontend) architecture. They are 3 roles : admin, patient and doctor. The admin role provide links between patient and doctors. Patients and doctors access files in database which are encrypted. We also implement some features :

- User registration, authentification and revocation
- Adding / deleting a doctor
- Consulting a medical record
- Uploading, editing and deleting files from the server

Framework versions.

- The `server` is developped using symfony (version 6.2), php (version 8.2) and maria DB (version 10.11)
- The `client` is developped using node.js (version 20.2.0), npm (version 9.6.6), nvm (0.35.3)
- m.a.j pour mariadb (vulnerabilities), on sait où on se situe entre 10.11.0 et 10.11.4 ?

1.2. Academdic hypothesis

In general the trust of a webbsite [1] is based on *domain* name. It's the information that will be asked by an authority like let's encrypt [2] and we don't have any domain name, this is an academic project. Therefore we will suppose that we grant to certification authority a domain name which is trusted and we use a X.509 [3] self-signed certificate on our server.

A similar situation occur for the client which is iddentified by *email*, in some situation we want to be sure that the user is not an impostor (if Alice send a message to Bob we want to be sure it's not Oscar who might have slolen some information from Alice). We may consider that all parties have given one time the iddentity card and and confirmed their email addresss.

1.3. Strong authentification scheme

Symfony [4] is implemented using *authentification* (firewall) and *authorisation* (access control).

Many authentification broken scenarios [5] don't concern us because we don't use any passwords to connect but the mail.

- m.a.j est-ce qu'on peu avoir que Oscar a volé le mail de Bob et accède tranquilou à ses fichiers, est-ce qu'on keep l'adresse IP dans les logs ? Est-ce qu'on envoie un sms au patient pour lui dire qu'il a un rendez-vous ?

About session timeout ...

- m.a.j sessions : on fait du statefull ou du stateless ?
- m.a.j check : OWASP guidelines

2. Check-list

2.1. Confidentiality

Non repudiation by certificate authority.

During creation, *patient* and *doctor* create their own public and private keys. The server activate a *certificate request* which is shaped in the Section 1.2 but we also use symfony [4] component that can text the user on mobile phone.

End-to-end encryption.

All data transmission are protected by end-to-end encryption. We use assymetric keys. When a *doctor* or a *patient* connect to the server we check by mail and phone if it's the right person. Then we send information for the meeting encrypted (using the public key). Then the *patient* and the *doctor* have to decrypt using they own private key which is then not stored on the server. Then *system administrator doesn't have any access to sensible data of some arbitrary user.*

Data storage.

The client encrypt all files and passwords using a symetric key which ensure a secure storage and mean that the system administrator doesn't have access to it neather.

2.2. Integrity

We store the datas in database using a symetric keys. Then patient encrypt the symetric key using the public key of the doctor which decrypt the coded message using his private key.

2.3. Relying secrecy

- m.a.j todo

Do my security features rely on secrecy, beyond credentials?

2.4. Injection vulnerability

Am I vulnerable to injection?

- URL, SQL, Javascript and dedicated parser injections

2.5. Remanence attack vulnerability

- m.a.j todo

Am I vulnerable to data remanence attacks?

2.6. Replay attack vulnerability

- m.a.j todo

Am I vulnerable to replay attacks?

2.7. Fraudulent request forgery vulnerability

- m.a.j todo

Am I vulnerable to fraudulent request forgery?

2.8. Monitoring

The server generate *logs* to let the administrator monitor all users activities and detect, analyse malicious events eventually.

- m.a.j Do I simply reject invalid entries, or do I analyse them?
- m.a.j Can logs be falsified?

2.9. Component security knowledge

We checkt on website as `https://www.acunetix.com`, `https://snyk.io/fr/`, `http://www.cvedetails.com` for node.js [6], symphony [7] and mariadb [8] and it appear that the named vulnerabilities were present until older version that the one we use regarding Section 1.1.1

2.10. System

- m.a.j todo

Is my system updated?

2.11. Access control

- m.a.j todo

Is my access control broken (cf. OWASP 10) [5]? Do I use indirect references to resource or functions?

2.12. General security features

- m.a.j todo

Are my general security features misconfigured (cf. OWASP 10) [5]? Also, note that you will unlinkely graduate should you fail to

,.

achieve end-to-end encryption (if relevant)..

The end to end encryption is handled

References

[1] "Scamdoc." [Online]. Available: https://fr.scamdoc.com/

[2] "Let's encrypt." [Online]. Available: https://letsencrypt.org/fr/

[3] "X.509 certificate." [Online]. Available: https://www.openssl.org/docs/man3.0/man7/x509.html

[4] "Symfony." [Online]. Available: https://symfony.com/

[5] "Owasp guidelines." [Online]. Available: https://owasp.org/www-project-secure-coding-practices-quick-reference-guide/

[6] "Sensio labs for node.js." [Online]. Available: https://www.cvedetails.com/vulnerability-list/vendor_id-12113/Nodejs.html

[7] "Sensio labs for symfony." [Online]. Available: https://www.cvedetails.com/vulnerability-list/vendor_id-11981/product_id-22402/Sensiolabs-Symfony.html

[8] "Sensio labs for maria db." [Online]. Available: https://www.cvedetails.com/vulnerability-list/vendor_id-12010/Mariadb.html

UMONS, Belgium
Email address: alessandro.spinosi@student.umons.ac.be

UMONS, Belgium
Email address: guillaume.kerckhofs@student.umons.ac.be

UMONS, Belgium
Email address: samain.clement@student.umons.ac.be

UMONS, Belgium
Email address: randi.dochot@student.umons.ac.be
URL: www.dochot.be

UMONS, Belgium
Email address: alix.Declerck@student.umons.ac.be

UMONS, Belgium
Email address: romain.absil@umons.ac.be