



Universidade do Minho
Escola de Engenharia

CertApp

Projeto de Desenvolvimento

Mestrado Integrado em Engenharia Informática

Engenharia de Segurança

Luís Vila
(a84439)

Hugo Matias
(a85370)

July 25, 2021

Contents

1	Introdução	5
2	Segurança do código	5
2.1	Estudo sobre o <i>third party software</i> utilizado	5
2.1.1	Vulnerabilidades públicas	6
2.1.2	Mitigar vulnerabilidades conhecidas	7
2.2	Análise estática da segurança do código	7
2.3	Utilização de <i>standards</i> criptográficos	8
2.4	Validação de input	8
3	SAMM	9
4	ASVS	9
4.1	Authentication Verification Requirements	9
4.2	Access Control Verification Requirements	10
4.3	Validation, Sanitization and Encoding Verification Requirements	10
4.4	Data Protection Verification Requirements	10
4.5	API and Web Service Verification Requirements	11
5	Análise PIA	11
6	Testar o código-fonte	11
6.1	Documentação do código	11
6.2	<i>OpenSSL</i>	11
6.3	<i>NodeJS</i>	14
6.4	Código-fonte	14
7	Funcionalidades da aplicação	14
7.1	Utilizador	14
7.1.1	Registo de um utilizador	14
7.1.2	Autenticação de um utilizador	15
7.1.3	Emissão de certificados	15
7.1.4	Verificação do estado de um certificado (<i>OCSP</i>)	16
7.1.5	<i>Certificate Revocation List (CRL)</i>	16
7.1.6	Lista de certificados	17
7.1.7	Emissão do <i>timestamp</i>	17
7.1.8	Verificação do <i>timestamp</i>	17
7.2	Administrador	17
7.2.1	Aceitação de pedido de certificado	17
7.2.2	Revogação de certificados	17
7.2.3	Aceitação de pedido de <i>timestamp</i>	18

8	Conclusão	18
9	Bibliografia	18
10	Anexos	18
10.1	Documentação código	18
10.2	<i>Webapp</i>	23
10.3	PIA	27

List of Figures

1	Vulnerabilidade <i>OpenSSL</i> (1)	6
2	Vulnerabilidade <i>OpenSSL</i> (2)	6
3	Versão <i>OpenSSL</i> utilizada.	7
4	<i>Output</i> da ferramenta <i>Semgrep</i>	7
5	Software Assurance Maturity Model	9
6	Ficheiro de configuração para emissão de um certificado. . . .	16
7	Página inicial de registo e <i>login</i>	23
8	Página de funcionalidades relacionadas com certificados, disponíveis aos utilizadores.	23
9	Funcionalidade que mostra os certificados de um utilizador .	24
10	Funcionalidade que apresenta a informação de um certificado.	24
11	Funcionalidade de verificação do estado de um certificado com recurso a um servidor <i>OCSP</i>	24
12	Funcionalidade de criação de um <i>certificate request</i>	25
13	Funcionalidade de verificação da revogação de um certificado recorrendo à <i>CRL</i>	25
14	Funcionalidade de verificação do <i>timestamp</i> de um certificado	26
15	Página de administrador com as respectivas funcionalidades que este poderá efectuar.	26
16	Visão geral do plano de ação.	27
17	Cartografia dos riscos potenciais da aplicação.	28
18	Visão geral sobre os riscos da aplicação.	29

1 Introdução

O projecto desenvolvido visa a implementação de uma plataforma para a emissão de certificados, *CRL*, *timestamps*, da disponibilização do serviço *OCSP* e o desenvolvimento de *interface web* para os utilizadores, que lhes providencie as funcionalidades necessárias para a sua autenticação, emissão de certificados, acesso aos seus certificados e a validação dos seus certificados e *timestamps*. O desenvolvimento das funcionalidades acima referidas foi feito com recurso à linha de comando *OpenSSL*. O *OpenSSL* apresenta ficheiros de configuração alteráveis que permitem a personalização da emissão de certificados, *CRL*, *timestamps*, bem como a utilização do serviço *OCSP*.

2 Segurança do código

No desenvolvimento da aplicação, teve-se em consideração práticas de desenvolvimento de código seguro e programação defensiva. Para este efeito, utilizou-se como orientação o *Microsoft Security Development Lifecycle*, seguindo-se práticas de desenvolvimento seguro como utilização de *standards* criptográficos, utilização de ferramentas aprovadas, entre outros, práticas estas que se listam nas seguintes sub-secções.

2.1 Estudo sobre o *third party software* utilizado

Como foi mencionado anteriormente, para a realização deste projeto, utilizou-se o *software open source OpenSSL*. Por norma, este tipo de *software* é seguro, mesmo devido à sua característica de ser *open source*, pois caso ocorra uma vulnerabilidade, um maior número de pessoas trabalhará no sentido de a resolver. De qualquer das formas, vulnerabilidades podem ocorrer, e sendo estas públicas, é importante não utilizar *software* com vulnerabilidades conhecidas (o que pode ocorrer ao utilizar-se o *software* numa versão já ultrapassada). Tendo isto em conta, foram pesquisadas vulnerabilidades conhecidas sobre este.

2.1.1 Vulnerabilidades públicas

Para inferir sobre possíveis vulnerabilidades do *OpenSSL*, pesquisou-se em <https://cve.mitre.org/> sobre este, tendo-se encontrado algumas vulnerabilidades conhecidas, das quais se listam algumas das mais recentes:

- CVE-2021-3450

🚩 CVE-2021-3450 Detail

Current Description

The X509_V_FLAG_X509_STRICT flag enables additional security checks of the certificates present in a certificate chain. It is not set by default. Starting from OpenSSL version 1.1.1h a check to disallow certificates in the chain that have explicitly encoded elliptic curve parameters was added as an additional strict check. An error in the implementation of this check meant that the result of a previous check to confirm that certificates in the chain are valid CA certificates was overwritten. This effectively bypasses the check that non-CA certificates must not be able to issue other certificates. If a "purpose" has been configured then there is a subsequent opportunity for checks that the certificate is a valid CA. All of the named "purpose" values implemented in libcrypto perform this check. Therefore, where a purpose is set the certificate chain will still be rejected even when the strict flag has been used. A purpose is set by default in libssl client and server certificate verification routines, but it can be overridden or removed by an application. In order to be affected, an application must explicitly set the X509_V_FLAG_X509_STRICT verification flag and either not set a purpose for the certificate verification or, in the case of TLS client or server applications, override the default purpose. OpenSSL versions 1.1.1h and newer are affected by this issue. Users of these versions should upgrade to OpenSSL 1.1.1k. OpenSSL 1.0.2 is not impacted by this issue. Fixed in OpenSSL 1.1.1k (Affected 1.1.1h-1.1.1j).

[+View Analysis Description](#)

Severity CVSS Version 3.x CVSS Version 2.0

CVSS 3.x Severity and Metrics:

 **NIST:** NVD **Base Score:** 7.4 HIGH **Vector:** CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:N

Figure 1: Vulnerabilidade *OpenSSL* (1)

- CVE-2021-3449

🚩 CVE-2021-3449 Detail

Current Description

An OpenSSL TLS server may crash if sent a maliciously crafted renegotiation ClientHello message from a client. If a TLSv1.2 renegotiation ClientHello omits the signature_algorithms extension (where it was present in the initial ClientHello), but includes a signature_algorithms_cert extension then a NULL pointer dereference will result, leading to a crash and a denial of service attack. A server is only vulnerable if it has TLSv1.2 and renegotiation enabled (which is the default configuration). OpenSSL TLS clients are not impacted by this issue. All OpenSSL 1.1.1 versions are affected by this issue. Users of these versions should upgrade to OpenSSL 1.1.1k. OpenSSL 1.0.2 is not impacted by this issue. Fixed in OpenSSL 1.1.1k (Affected 1.1.1-1.1.1j).

[+View Analysis Description](#)

Severity CVSS Version 3.x CVSS Version 2.0

CVSS 3.x Severity and Metrics:


 **NIST:** NVD **Base Score:** 5.9 MEDIUM **Vector:** CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:N/A:H

Figure 2: Vulnerabilidade *OpenSSL* (2)

2.1.2 Mitigar vulnerabilidades conhecidas

Atentando às descrições das vulnerabilidades supramencionadas, percebe-se que estas vulnerabilidades se encontram em determinadas versões do *software*, não incluindo a versão mais recente, sendo que se recomenda o *upgrade* para essa, nomeadamente a versão *1.1.1k*.

Desta forma, foi então utilizada a versão mais recente e sem vulnerabilidades conhecidas do *OpenSSL*.

```
[vilz@vilz ~]$ openssl version
OpenSSL 1.1.1k  25 Mar 2021
```

Figure 3: Versão *OpenSSL* utilizada.

2.2 Análise estática da segurança do código

Para testar a segurança do código desenvolvido, utilizou-se uma ferramenta de testagem de código estático, mais concretamente a ferramenta *semgrep*. Escolheu-se esta ferramenta por ser uma ferramenta de utilização gratuita, *open source* e que suporta *javascript*, linguagem em que este projeto foi desenvolvido.

A ferramenta foi utilizada na diretoria em que se encontra o projeto, indicando o seu output que não foram encontrados erros de implementação no projeto.

```
[vilz@vilz PD_EE]$ semgrep --config p/ci
A new version of Semgrep is available. Please see https://github.com/returntocorp/semgrep#upgrading for more information.
using config from https://semgrep.dev/p/ci. Visit https://semgrep.dev/registry to see all public rules.
downloading config...
running 135 rules...
100%|██████████████████████████████████████████████████████████████████████████████|135/135
ran 135 rules on 1018 files: 0 findings
[vilz@vilz PD_EE]$
```

Figure 4: *Output* da ferramenta *Semgrep*

2.3 Utilização de *standards* criptográficos

Para o funcionamento da aplicação, é necessário guardar informação sensível sobre os seus utilizadores, nomeadamente nomes de utilizador e palavras-chave.

Assim, surgiu a necessidade de proteger as informações guardadas, para o caso de estas serem tornadas públicas por via de algum potencial ataque.

Por isso, implementou-se a cifragem destas palavras-passe, através de uma função de *hash*, protegendo-se a informação que a aplicação guarda para operar.

2.4 Validação de input

A aplicação funciona com *input* por parte do utilizador. Assim surgiu a necessidade de proteger a aplicação com metodologias de programação defensiva que permitam evitar potenciais ataques devido a esta particularidade.

Desta forma, implementou-se validação de *input* para cada *input* que fosse necessário por parte do utilizador.

Para a validação dos *inputs* criou-se uma *whitelist* de caracteres permitidos, sendo que qualquer *input* que integrasse caracteres que não constassem desta lista não permite ao utilizador executar a ação em que estes foram requeridos.

Desta lista fazem parte apenas caracteres alfanuméricos e também alguns caracteres como *underscores*, sendo assim deixados de parte caracteres que podem potencialmente ser perigosos, como por exemplo *plicas*, que podem ser utilizadas em ataques a uma base de dados.

A *whitelist* utilizada foi a criada da seguinte forma:

```
const whitelist = ('1234567890'+'abcdefghijklmnopqrstuvxyz'+  
'abcdefghijklmnopqrstuvxyz'.toUpperCase()+').split("");
```


3 SAMM

Pergunta P2.1

Current Maturity Score					
Functions	Security Practices	Current	Maturity		
			1	2	3
Governance	Strategy & Metrics	0,95	0,30	0,30	0,35
Governance	Policy & Compliance	0,70	0,00	0,35	0,35
Governance	Education & Guidance	0,65	0,20	0,25	0,20

Pergunta P2.2

O nível de maturidade que se pretende alcançar nas práticas de segurança é 3.

Pergunta P2.3

Business Functions	Security Practices	Current	1	2	3
Governance	Strategy & Metrics	3,00	1,00	1,00	1,00

Governance		Phase 1 Projection		Phase 2 Projection	
Strategy & Metrics		Answer	Rating	Answer	Rating
SM1	Is there a software security assurance program in place? Are development staff aware of future plans for the assurance program? Do the business stakeholders understand your organization's risk profile?	Yes, it's a number of years Yes, a small percentage Yes, at least half of them	1,25	Yes, it's a number of years Yes, the majority of them Yes, at least half of them	2,17
SM2	Are many of your applications and resources categorized by risk? Are risk ratings used to tailor the required assurance activities? Does the organization know about what's required based on risk ratings?	Yes, at least half of them Yes, at least half of them Yes, at least half of them		Yes, the majority of them Yes, the majority of them Yes, the majority of them	
SM3	Is per-project data for the cost of assurance activities collected? Does your organization regularly compare your security spend with that of other organizations?	Yes, a small percentage Yes, we do it every few		Yes, at least half of them Yes, we do it every few	

Governance		Phase 3 Projection		Phase 4 Projection	
Strategy & Metrics		Answer	Rating	Answer	Rating
SM1	Is there a software security assurance program in place? Are development staff aware of future plans for the assurance program? Do the business stakeholders understand your organization's risk profile?	Yes, it's a pretty mature Yes, the majority of them Yes, at least half of them	2,58	Yes, it's a pretty mature Yes, the majority of them Yes, the majority of them	3,00
SM2	Are many of your applications and resources categorized by risk? Are risk ratings used to tailor the required assurance activities? Does the organization know about what's required based on risk ratings?	Yes, the majority of them Yes, the majority of them Yes, the majority of them		Yes, the majority of them Yes, the majority of them Yes, the majority of them	
SM3	Is per-project data for the cost of assurance activities collected? Does your organization regularly compare your security spend with that of other organizations?	Yes, the majority of them Yes, we do it every few		Yes, the majority of them Yes, we do it at least	

Figure 5: Software Assurance Maturity Model

4 ASVS

Nesta secção serão apresentados os requerimentos de segurança verificados no projecto desenvolvido, de acordo com a lista de requerimentos de segurança *ASVS* (*Application Security Verification Standard*). Para a análise e implementação destes requerimentos, assumiu-se uma *ASVS* de nível 1.

4.1 Authentication Verification Requirements

Autenticação é o acto de estabelecer, ou confirmar, algo ou alguém como autêntico e que reivindicações feitas por um indivíduo ou sobre um dispositivo são corretas, resistentes a imitação, e previnem a recuperação e interceptação de password. Tendo isto em conta foram verificados os seguintes requerimentos relativamente a este campo na aplicação desenvolvida:

- Verificar que a password de um utilizador têm no mínimo 12 caracteres;
- Verificar que passwords não podem ter mais de 128 caracteres;

- Verificar que "dicas" de password ou autenticação baseada em conhecimento (questões secretas) não se encontram presentes.

4.2 Access Control Verification Requirements

Autorização é o conceito de permitir acesso a recursos, apenas a quem têm permissão de os usar. Foram verificados os seguintes requerimentos relacionados com este campo no projecto:

- Verificar que a aplicação impõe regras de controlo de acesso, especialmente se controlo de acesso do lado do cliente está presente e há a possibilidade de ser contornado;
- Verificar que os utilizadores apenas têm acesso a funções, ficheiros de dados, serviços e outros recursos para os quais estes possuem autorização específica;
- Verificar que os controlos de acesso falham de forma segura, inclusive quando uma excepção ocorre;
- Verificar que todos os atributos de dados e utilizadores não podem ser manipulados por end-users, a menos que estes tenham autorização para tal.

4.3 Validation, Sanitization and Encoding Verification Requirements

Validação de input vindo do cliente ou do ambiente antes de o utilizar directamente.

- Verificar que a aplicação têm defesas contra ataques de poluição HTTP;
- Verificar que existe protecção contra ataques de parâmetros inseguros.

4.4 Data Protection Verification Requirements

Aplicações devem assumir que os dispositivos dos utilizadores estão comprometidos de alguma forma. A aplicação é assim responsável por assegurar que os dados armazenados nos dispositivos não são acedidos, ou alterados com facilidade.

- Verificar que versões antigas de SSL e TLS estão desactivadas;
- Verificar que todos os dados sensíveis criados e processados pela aplicação foram identificados, e assegurar que há uma política para lidar com este tipo de dados.

4.5 API and Web Service Verification Requirements

Requerimentos relativos a aplicações que usem API de serviços de confiança.

- Verificar que acesso à administração e gestão de funções é limitada a administradores limitados.

5 Análise PIA

Foi efetuada uma análise sobre o impacto da utilização de dados pessoais dos utilizadores da aplicação para o funcionamento desta.

Estas informações encontram-se no ficheiro *pia.pdf*, assim como as imagens resultantes desta análise podem ser também encontradas no repositório do projeto, assim como nos **Anexos**.

6 Testar o código-fonte

Para a utilização da aplicação, são necessárias a instalação e configuração do *OpenSSL*, assim como a instalação do *NodeJS*.

6.1 Documentação do código

O código desenvolvido encontra-se maioritariamente no ficheiro *index.js*, na diretoria **App/routes**. Todo esse código se encontra documentado no ficheiro, estando essa documentação também incluída em **Anexos**.

6.2 *OpenSSL*

Para atingir o objectivo deste projecto foi necessário efetuar alterações no ficheiro de configuração *OpenSSL*, bem como a criação de directorias de modo a obter ordem no armazenamento dos ficheiros criados.

As alterações feitas ao ficheiro de configuração *OpenSSL* foram as seguintes:

```
[ CA_default ]

dir          = /root/ca          # Where everything is kept
certs        = $dir/certs        # Where the issued certs are kept
crl_dir      = $dir/crl          # Where the issued crl are kept
database     = $dir/index.txt    # database index file.
#unique_subject = no             # Set to 'no' to allow creation of
                                # several certs with same subject.
new_certs_dir = $dir/newcerts    # default place for new certs.
```

```

certificate    = $dir/cacert.pem      # The CA certificate
serial         = $dir/serial          # The current serial number
crlnumber      = $dir/crl/pulp_crl_number # the current crl number
                # must be commented out to leave a V1 CRL
crl            = $dir/crl/crl.pem      # The current CRL
private_key    = $dir/private/cakey.pem # The private key

x509_extensions = usr_cert            # The extensions to add to the cert
[ usr_cert ]
authorityInfoAccess = OCSP;URI:http://127.0.1.1:8080/
# These extensions are added when 'ca' signs a request.
[ v3_OCSP ]

basicConstraints = CA:FALSE
keyUsage = nonRepudiation, digitalSignature, keyEncipherment
extendedKeyUsage = OCSPSigning

[ crl_ext ]

# CRL extensions.
# Only issuerAltName and authorityKeyIdentifier make any sense in a CRL.

# issuerAltName=issuer:copy
authorityKeyIdentifier=keyid:always,issuer:always
[ tsa_config1 ]

# These are used by the TSA reply generation only.
dir          = /root/ca/timestamp      # TSA root directory
serial       = $dir/tsa_serial         # The current serial number (mandatory)
crypto_device = builtin               # OpenSSL engine to use for signing
signer_cert  = $dir/tsa.crt           # The TSA signing certificate
                # (optional)
certs        = $dir/tsa-chain.pem      # Certificate chain to include in reply
                # (optional)
signer_key   = $dir/tsa.key            # The TSA private key (optional)
signer_digest = sha256                # Signing digest to use. (Optional)
default_policy = tsa_policy1          # Policy if request did not specify it
                # (optional)
other_policies = tsa_policy2, tsa_policy3 # acceptable policies (optional)
digests       = sha1, sha256, sha384, sha512 # Acceptable message digests (mandatory)
accuracy      = secs:1, millisecs:500, microsecs:100 # (optional)
clock_precision_digits = 0            # number of digits after dot. (optional)
ordering      = yes                  # Is ordering defined for timestamps?
                # (optional, default: no)

```

```

tsa_name          = yes      # Must the TSA name be included in the reply?
                        # (optional, default: no)
ess_cert_id_chain = yes      # Must the ESS cert id chain be included?
                        # (optional, default: no)
ess_cert_id_alg   = sha256    # algorithm to compute certificate
                        # identifier (optional, default: sha1)
                        #
                        #
[ tsa_ext ]

authorityKeyIdentifier = keyid:always
basicConstraints = critical,CA:false
extendedKeyUsage = critical,timeStamping
keyUsage = critical,nonRepudiation
subjectKeyIdentifier = hash

```

Foi necessário criar uma directoria para a *certificate authority*, e dentro desta foram criadas directorias para os certificados, chaves privadas, *certificate requests*, *timestamps* e *OCSP*. O conteúdo de cada uma destas directorias é o seguinte:

- */root/ca/*: Esta é a directoria principal, aqui encontra-se o certificado da *certificate authority*, assim como o resto das directorias necessárias para o funcionamento da aplicação;
- */root/ca/requests/*: Aqui encontram-se todos os *certificate requests* de todos os utilizadores;
- */root/ca/certs/*: Directoria que contém todos os certificados dos utilizadores;
- */root/ca/private/*: Directoria que contém todas as chaves privadas dos certificados;
- */root/ca/OCSP/*: A directoria *OCSP* contém o certificado *OCSP*, e a chave privada, necessários para o funcionamento do servidor *OCSP*;
- */root/ca/timestamp/*: Esta directoria têm o certificado para *timestamping* e a respectiva chave privada, de modo a ser possível aplicar esta funcionalidade aos certificados;
- */root/ca/crl/*: Nesta directoria encontra-se a *CRL* e os ficheiros necessários para a criação desta lista.

6.3 *NodeJS*

Para a instalação do *NodeJS* apenas é necessário o *download* e instalação dos pacotes *npm* e *nodejs*.

6.4 Código-fonte

A testagem do código fonte, após as configurações supramencionadas, apenas necessita da execução do comando:

```
$ npm start
```

O servidor da aplicação será iniciado e a aplicação poderá ser acessada em qualquer *browser* em localhost:3000.

Para utilização do serviço *OCSP* a partir da aplicação, é necessário que o servidor deste serviço esteja a operar, e este pode ser iniciado com o seguinte comando:

```
$ openssl ocsp -index /root/ca/index.txt -port 8080 -rsigner \  
/root/ca/ocsp/ocspSigning.crt -rkey /root/ca/ocsp/ocspSigning.key -CA\  
/root/ca/cacert.pem -text -out log.txt &
```

Para garantir o bom funcionamento da *CRL*, é necessário regenerar a lista depois da revogação de um certificado:

```
$ openssl ca -gencrl -keyfile /root/ca/private/cakey.pem  
-cert /root/ca/cacert.pem -out /root/ca/crl/crl.pem  
  
$ cat /root/ca/cacert.pem /root/ca/crl/crl.pem > /root/ca/crl/test.pem
```

7 Funcionalidades da aplicação

7.1 Utilizador

7.1.1 Registo de um utilizador

Para que se possa utilizar a aplicação, é necessário que se crie um utilizador, providenciando um nome de utilizador e uma palavra-passe.

Estes *inputs* serão validados, e garantir-se-á que o novo nome de utilizador não existe já na aplicação.

Verificando-se estas condições, o novo utilizador é registado, podendo a partir deste momento autenticar-se e começar a utilizar a aplicação.

7.1.2 Autenticação de um utilizador

Tendo um utilizador criado, uma pessoa pode autenticar-se na aplicação, sendo que para tal efeito deve fornecer os seus dados de nome de utilizador, assim como a sua palavra-passe. Estas informações serão validadas (tanto passam por um processo de validação de *input* como de verificação de que os dados existem na aplicação e estão corretos).

Verificando-se estas condições, o utilizador é autenticado e pode começar a utilizar a aplicação.

7.1.3 Emissão de certificados

Uma das principais funcionalidades desta aplicação é a emissão de certificados. Assim recorrendo a linha de comandos *OpenSSL* é construído um *certificate request*, que por sua vez será validado por um administrador.

Para a criação do *request*, o utilizador deve preencher um formulário com informações que serão posteriormente utilizadas para a criação do certificado. Estas informações são: o nome que pretende dar ao certificado; a sigla do país em que se encontra; o nome da sua organização; um *common name* (ex.: *server FQDN* ou nome do utilizador);

Com estas informações será gerado um ficheiro de configuração (como demonstra a seguinte imagem) que será então utilizado na criação do *request* de um novo certificado.

```

PD_EE > App > ≡ serv.cnf
1  FQDN = umniho
2  ORGNAME = minho
3  ALTNames = DNS:$FQDNS
4  [ req ]
5  default_bits = 2048
6  default_md = sha256
7  prompt = no
8  encrypt_key = no
9  distinguished_name = dn
10 req_extensions = req_ext
11 [ dn ]
12 C = PT
13 O = $ORGNAME
14 CN = $FQDN
15 [ req_ext ]
16 subjectAltName = $ALTNames
17

```

Figure 6: Ficheiro de configuração para emissão de um certificado.

O novo *request* criado ficará em *stand by* até que seja aceite por um administrador.

Quando aceite, será gerado o certificado correspondente ao *request*.

7.1.4 Verificação do estado de um certificado (*OCSP*)

A verificação do estado de um certificado é feita com recurso a um servidor *OCSP*.

Inicialmente é inicializado um servidor *OCSP* que irá permitir a um utilizador verificar o estado de um dos seus certificados. Para a verificação do estado de um certificado é necessário apresentar como input o nome do certificado, é verificado se este certificado pertence ao utilizador actual, caso pertença, o resultado da verificação do estado por parte do servidor *OCSP* é guardada num ficheiro, que é apresentado posteriormente ao utilizador no *website*.

7.1.5 *Certificate Revocation List (CRL)*

A verificação dos certificados revogados é feita com recurso à construção de uma *Certificate Revocation List*. Nesta lista constam todos os certificados

revogados de um utilizador. A funcionalidade de verificação da revogação de um certificado recebe como input o nome do certificado. Verifica-se de seguida se este certificado pertence ao utilizador que fez o pedido de verificação, caso pertença, é verificado se o certificado pertence à lista de certificados revogados (*CRL*).

7.1.6 Lista de certificados

Um utilizador pode escolher ver a lista dos seus certificados (*certificate requests* aceites pelo administrador). Estes certificados são apresentados no *website*, ao utilizador por nome.

7.1.7 Emissão do *timestamp*

A emissão de um *timestamp* sobre um certificado necessita do certificado para o qual se pretende aplicar o *timestamp* e o nome deste ultimo. É verificado se o certificado em questão corresponde realmente ao utilizador que pretende emitir o *timestamp*. O pedido de *timestamp* é feito, e é posteriormente aceite pelo administrador.

7.1.8 Verificação do *timestamp*

A verificação de um *timestamp* é feita por um utilizador, é recebido como *input* o nome do *timestamp* a verificar. É verificado se este *timestamp* pertence realmente ao utilizador que faz o pedido de verificação, e , de seguida, o resultado da verificação é guardado num ficheiro, que é depois apresentada ao utilizador no *website*.

7.2 Administrador

7.2.1 Aceitação de pedido de certificado

Para a aceitação de um certificado é necessário como *input*, o nome do pedido de certificado a aceitar. O nome é por sua vez validado, e caso passe o teste de validação, é aceite o pedido em questão e criado um novo certificado com o mesmo nome que o pedido.

O utilizador que pediu o certificado pode consultá-lo a partir deste momento.

7.2.2 Revogação de certificados

Para a revogação de um certificado é necessário como input, o nome do certificado a revogar. O nome é por sua vez validado, e caso passe o teste de validação, é revogado o certificado em questão.

A verificação da revogação de um certificado é feita com recurso à funcionalidade **Verificação do estado de um certificado**.

7.2.3 Aceitação de pedido de *timestamp*

Para a aceitação de um *timestamp* é necessário como *input*, o nome do pedido de *timestamp* a aceitar. O nome é por sua vez validado, e caso passe o teste de validação, é aceite o pedido em questão e criado um novo *timestamp* com o mesmo nome que o pedido.

O utilizador que pediu o *timestamp* pode consultá-lo a partir deste momento.

8 Conclusão

O desenvolvimento deste projecto demonstrou ser moderadamente complexo, pois apesar das funcionalidades a implementar serem objectivas, e a ferramenta *OpenSSL* utilizada de fácil compreensão, as alterações necessárias às configurações desta ultima demonstraram-se um pouco confusas e meticolosas. Porém, a documentação clara da linha de comandos *OpenSSL* permitiu imediatamente identificar como haveriam de ser realizadas as funcionalidades requeridas neste projecto.

9 Bibliografia

- <https://semgrep.dev/>
- <https://www.openssl.org/>
- <https://nodejs.org/>
- <https://www.microsoft.com/en-us/securityengineering/sdl/practices>
- <https://owasp.org/>

10 Anexos

10.1 Documentação código

```
[breaklines]
/*****\
 * VARIÁVEIS GLOBAIS *
\*****/

var express = require('express');
```

```

var router = express.Router();

const saltRounds = 10;
const bcrypt = require('bcrypt');
const fs = require('fs');
const alert = require('alert');
const axios = require('axios');
const exec = require('child_process').exec;
const { SSL_OP_EPHEMERAL_RSA } = require('constants');
const writeline = require('prompt');

var logged_user = ''; // variável onde se guarda o nome de utilizador que inicia sessão

// Lista de caracteres aceites para validação de input
const whitelist = ('1234567890'+'abcdefghijklmnopqrstuvwxyz'+'abcdefghijklmnopqrstuvwxyz');

/*****\
 * VALIDAÇÃO DE INPUT *
*****/

// Função que servirá para validação de input, verificando se algum caracter do input é válido
function validate_input(input){

/*****\
 * FUNCIONALIDADES APLICAÇÃO *
*****/

/* GET home page.
 * Carrega o HTML da página inicial ao lançar a aplicação
 */
router.get('/', function(req, res, next) {...}

/* Registo de um utilizador
 * Recebido input com o novo username e com a password para registo.
 * Os inputs são validados.
 * Verifica-se se o nome de utilizador já foi utilizado.
 * É registado o novo utilizador.
 */
router.post('/', async (req, res, next) => {...}

/* Validação do tamanho das passwords
 */

```

```

async function val(pass){...}

/* Login na aplicação.
 * Recebe o username e password como input.
 * Valida os inputs.
 * Verifica se o username existe e se a password inserida é a correta.
 * Sendo o login bem-sucedido é carregada a página inicial da aplicação.
 */
router.post('/login', async (req, res, next) => {...}

/* Auxiliar ao login.
 * Verifica se um nome de utilizador e respetiva password se encontram no ficheiro c
 */
async function login_aux(username, password, data) {...}

/* Ação de clicar no botão de novo certificado.
 * Carrega a página onde se preenchem as informações para pedido de novo certificado
 */
router.post('/newcertificate', async (req, res, next) => {...}

/* Emissão de novo pedido de certificado.
 * São recebidas por input as informações necessárias para emissão de um novo certifi
 * Os inputs são validados.
 * Cria-se um ficheiro de configuração que especifica com que informações será criada
 * Cria-se um certificate request com recurso a esse ficheiro e a um comando do Open
 */
router.post('/newcertificateemit', async (req, res, next) => {...}

/* Auxiliar à emissão de um certificate request.
 * Função que cria o ficheiro de configuração para emissão de um pedido de novo cert
 */
async function create_config(country, orgname, fqdn){...}

/* Verificação do estado de um certificado por OCSP.
 * Recebe como input qual o certificado sobre o qual ocorrerá a verificação.
 * O input é validado.
 * Verifica-se que o certificado pertence ao utilizador que faz o pedido de verificação
 * A verificação é efetuada através de um comando do OpenSSL.

```

```

    * O seu resultado é guardado num ficheiro que é posteriormente aberto para o utilizador.
    */
    router.post('/ocsp', async (req, res, next) => {...}

/* Verificação do estado de um certificado por CRL.
 * Recebe como input qual o certificado sobre o qual ocorrerá a verificação.
 * O input é validado.
 * Verifica-se que o certificado pertence ao utilizador que faz o pedido de verificação.
 * A verificação é efetuada através de um comando do OpenSSL.
 * O seu resultado é guardado num ficheiro que é posteriormente aberto para o utilizador.
 */
router.post('/crl', async (req, res, next) => {...}

/* Pedido de timestamp sobre um certificado.
 * Recebe-se como input qual o certificado sobre o qual se quer o timestamp e qual o tempo.
 * Os inputs são validados.
 * Verifica-se que o certificado pertence ao utilizador que pede o timestamp.
 * É criado um pedido de timestamp através de um comando de OpenSSL.
 */
router.post('/timestamp', async (req, res, next) => {...}

/* Verificar timestamp.
 * Recebe como input qual o timestamp sobre o qual ocorrerá a verificação.
 * O input é validado.
 * Verifica-se que o timestamp pertence ao utilizador que faz o pedido de verificação.
 * A verificação é efetuada através de um comando do OpenSSL.
 * O seu resultado é guardado num ficheiro cuja informação é lida e passada ao utilizador.
 */
router.post('/timestampcheck', async (req, res, next) => {...}

/* Função auxiliar que verifica se um timestamp pertence ao utilizador ligado à aplicação.
 */
async function check_user_timestamp(timestamp){...}

/* Função auxiliar que verifica se um certificado pertence ao utilizador ligado à aplicação.
 */
async function check_user_cert(certificate){...}

```

```

/* Mostra a um utilizador quais são os seus certificados (pedidos e já aceites).
 */
router.post('/mycertificates', async (req, res, next) => {...}

/* Função auxiliar que verifica quais os certificados do utilizador a usar a aplicação
 */
async function my_certificates(){...}

/* Ver um certificado
 * Utilizador indica qual o certificado que pretende ver.
 * input é validado e verifica-se que o certificado pertence ao utilizador que pediu
 * É mostrado o conteúdo do certificado ao utilizador.
 */
router.post('/viewcertificate', async (req, res, next) => {...}

/*****\
 * ADMIN *
*****/

/* Aceitar um certificate request.
 * É dado como input o nome do request.
 * Valida-se o input.
 * Request é aceite e novo certificado criado com o mesmo nome por comando OpenSSL.
 */
router.post('/acceptcertificate', async (req, res, next) => {...}

/* Revocar um certificado.
 * É dado como input o nome do certificado.
 * Valida-se o input.
 * Certificado é revocado por comando OpenSSL.
 */
router.post('/revokecertificate', async (req, res, next) => {...}

/* Aceitar um timestamp request.
 * É dado como input o nome do request.
 * Valida-se o input.
 * Request é aceite e novo timestamp criado com o mesmo nome por comando OpenSSL.
 */
router.post('/accepttimestamp', async (req, res, next) => {...}

```

```

/* Logout da aplicação
 */
router.post('/logout', async (req, res, next) => {...}

```

10.2 Webapp

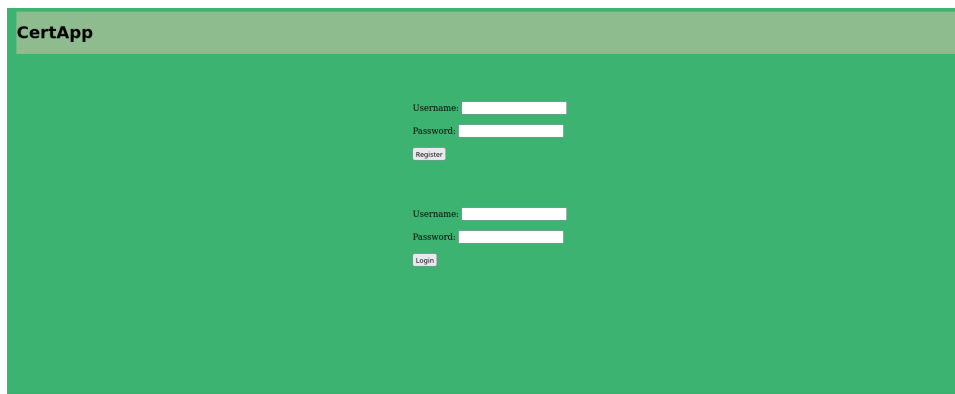


Figure 7: Página inicial de registo e *login*.



Figure 8: Página de funcionalidades relacionadas com certificados, disponíveis aos utilizadores.

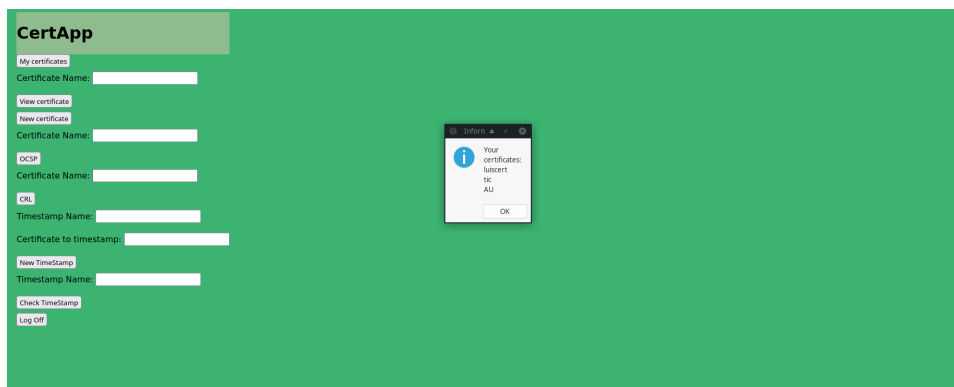


Figure 9: Funcionalidade que mostra os certificados de um utilizador

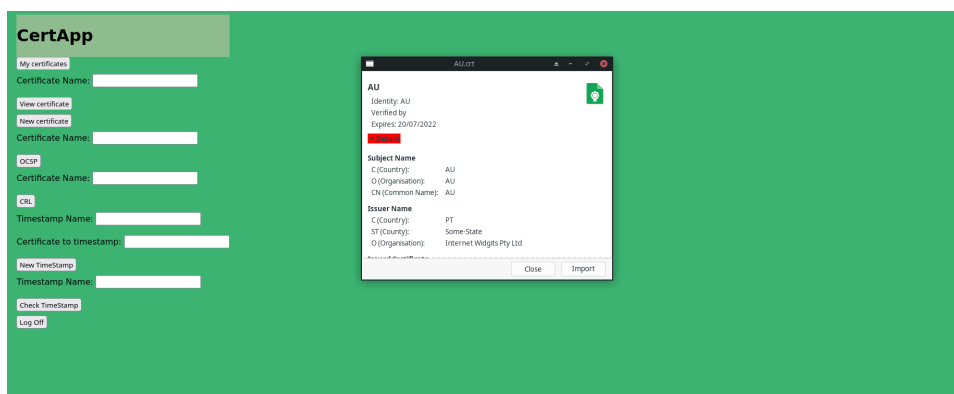


Figure 10: Funcionalidade que apresenta a informação de um certificado.



Figure 11: Funcionalidade de verificação do estado de um certificado com recurso a um servidor *OCSP*.


```
OCSP Response Data:
OCSP Response Status: successful (0x0)
Response Type: Basic OCSP Response
Version: 1 (0x0)
Responder ID: C = AU, ST = Some-State, O = Internet Wldgits Pty Ltd, CN = ca.project.local, emailAddress = ppp
Produced At: Jul 25 11:12:52 2021 GMT
Responses:
Certificate ID:
Hash Algorithm: sha1
Issuer Name Hash: 3d186477266f41f77c94069e9c633530d4858756
Issuer Key Hash: 4896795a76394ea2c058126ce7525cac08e92778
Serial Number: 1235
Cert Status: good
This Update: Jul 25 11:12:52 2021 GMT
Response Extensions:
OCSP Nonces:
0418b9f441c68f08028f23848995254784
Signature Algorithm: sha256withRSAEncryption
ab:7e:43:7a:1c:1b:f9:2d:4a:a5:8b:eb:b3:30:d1:4c:d3:a2:
24:25:06:3a:e5:08:2a:a5:25:3a:a9:ab:9a:de:44:81:7f:1b:
89:60:ca:50:ec:9c:d5:87:17:87:4a:79:ac:80:7e:42:14:5a:
d5:21:4d:ca:cd:69:1f:7c:e9:19:da:55:34:f9:51:68:8a:ec:
57:5a:17:70:35:7c:c1:70:7c:de:58:d2:f4:2a:60:f0:ee:19:
87:2b:71:6d:a8:af:e4:83:6c:a8:85:a9:cc:83:93:a6:98:a8:
43:6a:ab:d1:8b:f0:45:91:f2:5a:94:a9:52:9a:57:68:18:30:
d9:e0:3d:4b:58:ca:6c:2b:3a:c3:ee:8a:45:8b:3d:74:e9:a5:
ea:a0:3d:3a:5c:f0:b4:55:67:85:6f:58:53:58:b7:e2:27:70:
9c:19:c2:74:93:6d:1d:5a:0f:08:39:c5:9f:f2:fb:f2:98:de:
6a:eb:38:f5:50:95:82:c9:49:9c:9d:43:54:f0:6f:70:f8:f0:
ed:1f:ac:f5:7c:7f:c1:6d:5d:e4:37:86:a6:79:0f:6d:ed:4a:
6f:2d:59:17:35:1b:1b:6b:0a:1b:04:29:ac:60:fb:cf:c0:61:44:
d4:9c:e8:09:88:d2:86:7d:56:2f:3f:fc:ca:c3:48:61:b7:83:
cf:34:1d:9d
Certificate:
Data:
Version: 3 (0x2)
Serial Number: 4081 (0x1205)
Signature Algorithm: sha256withRSAEncryption
Issuer: C=PT, ST=Some-State, O=Internet Wldgits Pty Ltd
Validity
Not Before: Jul 10 15:10:18 2021 GMT
Not After : Jul 10 15:10:18 2022 GMT
Subject: C=AU, ST=Some-State, O=Internet Wldgits Pty Ltd, CN=ca.project.local/emailAddress=ppp
Subject Public Key Info:
Public Key Algorithm: rsaEncryption
```

Figure 12: Funcionalidade de criação de um *certificate request*.

```
/root/ca/certs/AU.crt: OK
```

Figure 13: Funcionalidade de verificação da revogação de um certificado recorrendo à *CRL*

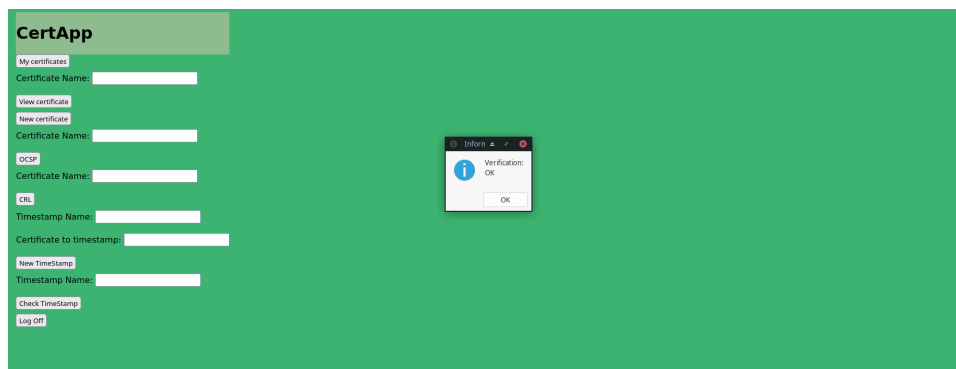


Figure 14: Funcionalidade de verificação do *timestamp* de um certificado



Figure 15: Página de administrador com as respectivas funcionalidades que este poderá efectuar.

10.3 PIA

Visão geral

Princípios fundamentais

Objetivos
Base legal
Dados adequados
Precisão de dados
Duração dos dados
Informação para os titulares dos dados
Obtenção do consentimento
Direito de acesso e portabilidade de dados
Direito à retificação e apagamento
Direito à restrição e à oposição
Subcontratação
Transferências

Medidas improváveis

Medidas aceitáveis

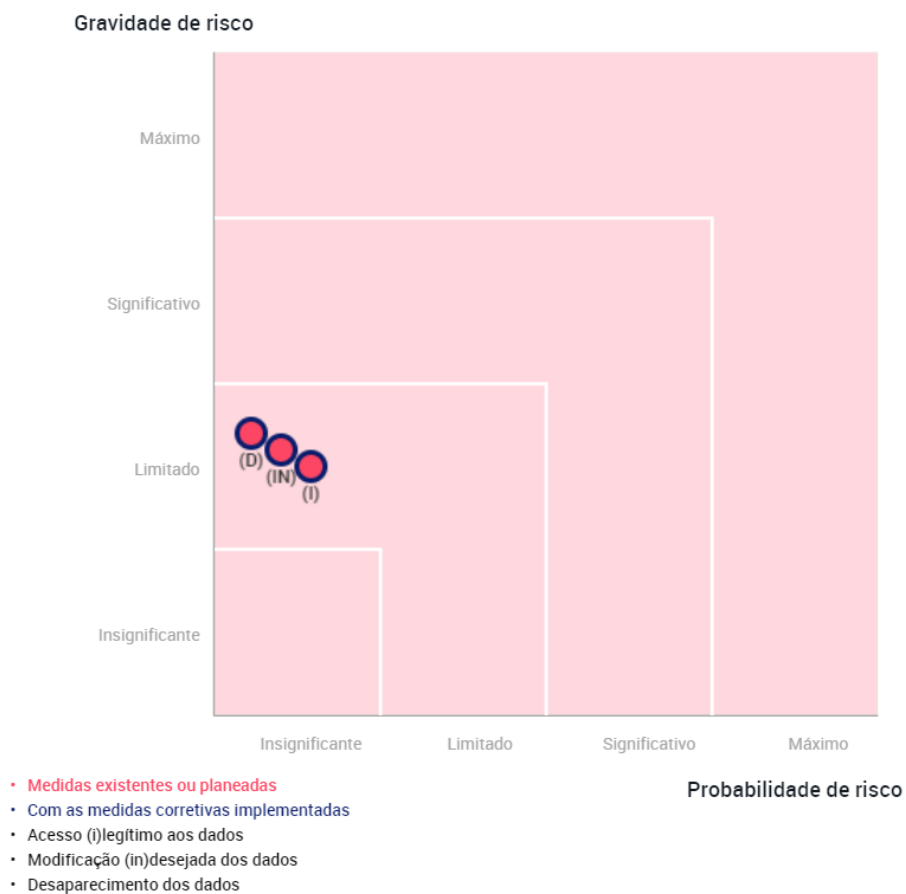
Medidas existentes ou planeadas

Cifragem
Controlo de acesso lógico
de dados pessoais
Relações com terceiros

Riscos

Acesso ilegítimo de dados
Modificação indesejada de dados
Desaparecimento de dados

Figure 16: Visão geral do plano de ação.



09/07/2021

Figure 17: Cartografia dos riscos potenciais da aplicação.

Impactos potenciais

Podiam ser tornados públicos
Podiam ser eliminados ou a

Ameaças

SQL Injection
Spoofing
Tampering
DoS

Fontes

Hackers
Negligência dos utilizadores
Falha no software de terceiros

Medidas

Cifragem
Controlo de acesso lógico
de dados pessoais
Relações com terceiros

Acesso ilegítimo dos dados

Gravidade : Limitado

Probabilidade : Insignificante

Modificação indesejada dos dados

Gravidade : Limitado

Probabilidade : Insignificante

Desaparecimento de dados

Gravidade : Limitado

Probabilidade : Insignificante

Figure 18: Visão geral sobre os riscos da aplicação.