Aula 3 TP – 18/Fev/2019

Grupo 3

1. *Assinaturas cegas (Blind signatures) baseadas no Elliptic Curve Discrete Logarithm Problem (ECDLP)*

- ***Experiência 1.1***

```
root@CSI:~/Desktop/Trabalho 2# openssl ecparam -name prime256v1 -genkey -noout -out key.pem
root@CSI:~/Desktop/Trabalho 2# openssl req -key key.pem -new -x509 -days 365 -out key.crt
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:PT
State or Province Name (full name) [Some-State]:Minho
Locality Name (eg, city) []:Braga
Organization Name (eg, company) [Internet Widgits Pty Ltd]:ES
Organizational Unit Name (eg, section) []:Grupo3
Common Name (e.g. server FQDN or YOUR name) []:Grupo3
Email Address []:
```

- ***Experiência 1.2***

  - *Inicialização*

```
root@CSI:~/Desktop/Trabalho 2# python initSigner-app.py
Output
Init components: ec3e8293d2f8408c7b0668cabaa8863955dd9efdef73b7dd9f
pRDashComponents: ec3e8293d2f8408c7b0668cabaa8863955dd9efdef73b7dd9
```

  - *Ofuscação*

```
root@CSI:~/Desktop/Trabalho 2# python generateBlindData-app.py
Input
Data: Vamos assinar esta mensagem sem o Bob a ver
pRDash components: ec3e8293d2f8408c7b0668cabaa8863955dd9efdef73b7dd9f8e02fa122d56e0.
Output
Blind message: 6dfd1ad529604a5137e96d4fec0b9bae4905e6b2ab1f02dc9cd88a90a56dcc45
Blind components: fefae8d0d62de33aecc0f849eb991cb2082a2213ce01b7ed96c523e355d5a928.c
pRComponents: caf14770242d7b1f676d8b5dc3a04c7e92458d7d50cd4cb0adb5d13f4c8483cf.abf2c
```

- ***Assinatura***

```
root@CSI:~/Desktop/Trabalho 2# python generateBlindSignature-app.py key.pem
Input
Passphrase: segredo
Blind message: 6dfd1ad529604a5137e96d4fec0b9bae4905e6b2ab1f02dc9cd88a90a56dcc45
Init components: ec3e8293d2f8408c7b0668cabaa8863955dd9efdef73b7dd9f8e02fa122d56e0.bc
Output
Blind signature: 4028b8b4974d15c5214e61f8b89594a55f7fb15d41e631b2a89b55100ee23998b5a
```

- ***Desofuscação***

```
root@CSI:~/Desktop/Trabalho 2# python unblindSignature-app.py
Input
Blind signature: 4028b8b4974d15c5214e61f8b89594a55f7fb15d41e631b2a89b55100ee23998b5a7
Blind components: fefae8d0d62de33aecc0f849eb991cb2082a2213ce01b7ed96c523e355d5a928.ca
pRDash components:  ec3e8293d2f8408c7b0668cabaa8863955dd9efdef73b7dd9f8e02fa122d56e0.
Output
Signature: ae053c9e51de21489555d5d3a419d2dd8258134bf2dde1a4638a1f79b664938
```

- ***Verificação***

```
root@CSI:~/Desktop/Trabalho 2# python verifySigature-app.py key.crt
Input
Original data: Vamos assinar esta mensagem sem o Bob a ver
Signature: ae053c9e51de21489555d5d3a419d2dd8258134bf2dde1a4638a1f79b664938
Blind components: fefae8d0d62de33aecc0f849eb991cb2082a2213ce01b7ed96c523e355d5a928.caf
pR components: caf14770242d7b1f676d8b5dc3a04c7e92458d7d50cd4cb0adb5d13f4c8483cf.abf2cd
Output
Valid signature
```

● ***Pergunta 1.1***

Como foi pedido no enunciado, alteramos o código fornecido na experiência 1.2, de forma a simplificar o input e output. Encontra-se abaixo, o resultado obtido nos diferentes códigos:

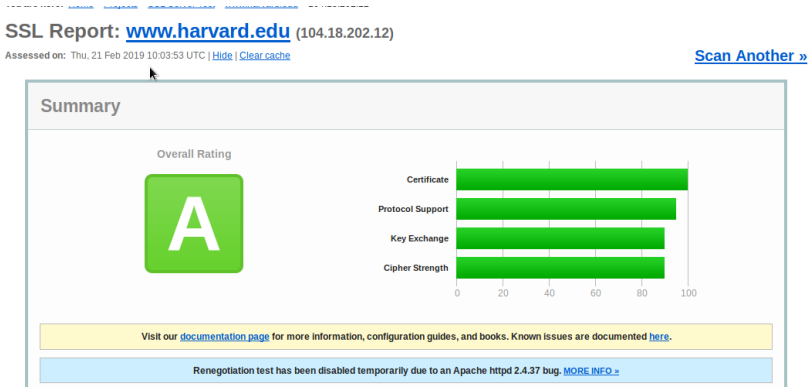
```
user@CSI:~/Desktop$ python initSigner-app.py -init
user@CSI:~/Desktop$ python initSigner-app.py
Output
pRDashComponents: 5cec07de7fd980003c68a8a6e5eaa84bdfbca978b073b7aac96e5ec36d1719
65.2096f914480657bc974a7bda6d89f52e2a35d0c99923738a36c11a0daedb3170
user@CSI:~/Desktop$ python generateBlindData-app.py -msg 'Vamos assinar esta men
sagem sem o Bob a ver' -RDash '5cec07de7fd980003c68a8a6e5eaa84bdfbca978b073b7aac
96e5ec36d171965.2096f914480657bc974a7bda6d89f52e2a35d0c99923738a36c11a0daedb3170
'
Output
Blind message: 4d853506e5263e7aeb69a26e5c123b79e3f0ccc5d07eda0c6164a2c1f1a667c1
user@CSI:~/Desktop$ python generateBlindSignature-app.py -key key.pem -bmsg '4d8
53506e5263e7aeb69a26e5c123b79e3f0ccc5d07eda0c6164a2c1f1a667c1'
Input
Passphrase: segredo
Output
Blind signature: e82ba7a491a6f91a051a13913a3ca7f97acdabfd0dd107dcafad26567145934
4e236b9183919a5e6b003663303b6c09822aebf4764247786093726ebccb58c6
user@CSI:~/Desktop$ python unblindSignature-app.py -s 'e82ba7a491a6f91a051a13913
a3ca7f97acdabfd0dd107dcafad265671459344e236b9183919a5e6b003663303b6c09822aebf476
4247786093726ebccb58c6' -RDash '5cec07de7fd980003c68a8a6e5eaa84bdfbca978b073b7aa
c96e5ec36d171965.2096f914480657bc974a7bda6d89f52e2a35d0c99923738a36c11a0daedb317
0'
Output
Signature: 3dc7d6477c34942af1ff94c6668d3eb3a349dc965168732899e001c003680580
user@CSI:~/Desktop$ python verifySigature-app.py -cert key.crt -msg 'Vamos assin
ar esta mensagem sem o Bob a ver' -sDash '3dc7d6477c34942af1ff94c6668d3eb3a349dc
965168732899e001c003680580' -f ficheiro_do_requerente
Output
Valid signature
```

2. *Protocolo SSL/TLS*

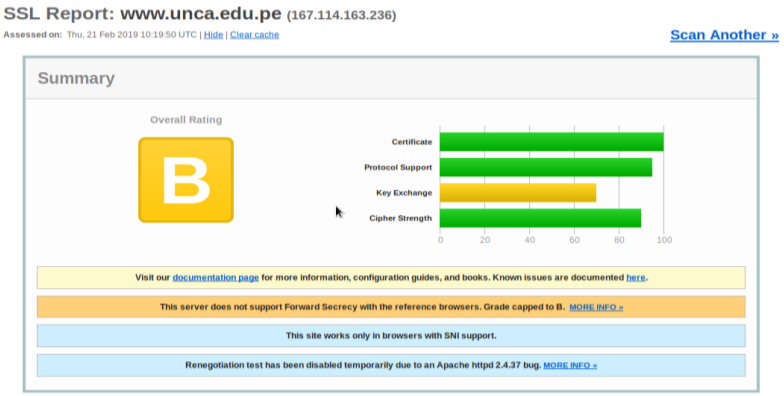
- ***Pergunta 2.1***

1) Os três sites de Universidades não Europeias bem como os seus resultados do *SSL Server test*, são os seguintes.

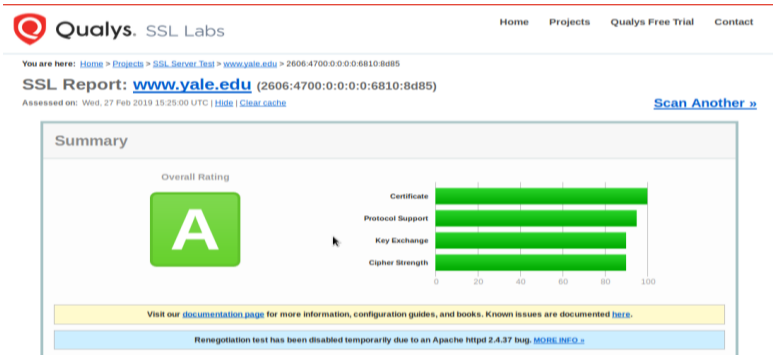
a) Universidade de Harvard, em Cambridge nos EUA



b) Universidade Nacional Ciro Alegria, Peru



c) *Universidade de Yale, em New Haven, nos EUA*



2) A Universidade com pior rating é a do Peru, com o nível B. O problema da segurança está na troca de chaves RSA, pois não fornece o sigilo de encaminhamento, ou seja, não existe uma comunicação segura porque os acordos de chaves de longo prazo não comprometem as chaves de sessão anteriores.

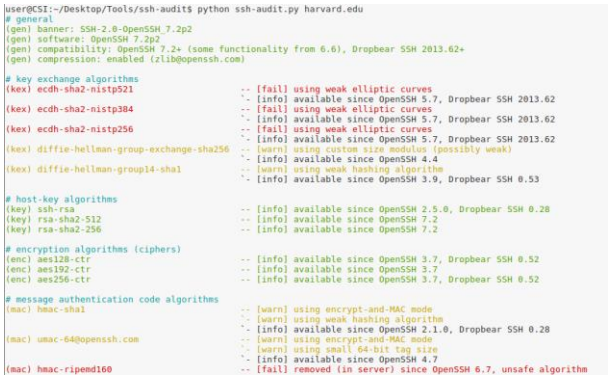
3) A Autorização de Autoridade de Certificação (CAA) é uma proposta para melhorar as emissões de certificados, isto é, esta CAA cria um mecanismo de DNS que permite que os proprietários de nomes de domínios façam uma lista de CAs nas quais confiam para emitir certificados para os seus nomes de host. Com isto, pretende-se que uma CA antes de emitir um certificado tem de verificar se está no registo DNS de permissões, caso não esteja recusa a emissão.

3. *Protocolo SSH*

- ***Pergunta 3.1***

Depois de instalado o *ssh-audit* na conta do utilizador *user* na máquina virtual, foram escolhidas duas universidades não europeias, sendo elas:

a) *Universidade de Harvard, em Cambridge nos EUA*



Analisando o resultado do *ssh-audit* vemos que o software e versão utilizada pelos servidores ssh é o SSH2.0 - *OpenSSH 7.2p2*.

b) Universidade de Yale, em New Haven, nos EUA

Recorrendo aos resultados apresentados o software e versão utilizados pelos servidores ssh são *SSH-2.0 – OpenSSH 5.6.*

A nível de vulnerabilidades de software, recorremos ao site *CVE details*, inserindo o nome do produto e a versão a pesquisar. Analisando cada uma das versões acima referidas a que apresenta maior número de vulnerabilidades é a versão OpenSSH 5.6. Contudo a que patenteia a vulnerabilidade mais grave, tendo um nível score de 7.8 (de acordo com o *CVSS score* identificado no CVE details) corresponde à versão OpenSSH 7.2. Esta vulnerabilidade *CVE-2016-6515* tem como objetivo provocar um ataque de DoS (Denial of Service), ou seja, esta versão poderia não limitar os comprimentos de senha para autenticação dos utilizadores, o que permitia que os atacantes remotos causassem uma negação de serviço.