Aula TP - 12/Fev/2018

Cada grupo deve colocar a resposta às perguntas dos seguintes exercícios na área do seu grupo no Github até ao final do dia 19/Fev/2018. Por cada dia de atraso será descontado 0,15 valores à nota desse trabalho.

Note que estes exercícios devem ser feitos na máquina virtual disponibilizada.

Exercícios

1. Números aleatórios/pseudoaleatórios

Experiência 1.1

Execute o seguinte comando, que gera 1024 bytes pseudoaleatórios: `openssl rand -base64 1024`

Pergunta P1.1

Teste os seguintes comandos, que vão obter 1024 bytes pseudoaleatórios do sistema e os apresentam em base64:

- `head -c 1024 /dev/random | openssl enc -base64`
- `head -c 1024 /dev/urandom | openssl enc -base64`

Que conclusões pode tirar? Em que se baseia para essas conclusões ?

Pergunta P1.2

O haveged - http://www.issihosts.com/haveged/index.html - é um daemon de entropia adaptado do algoritmo HAVEGE (*HArdware Volatile Entropy Gathering and Expansion*) - http://www.irisa.fr/caps/projects/hipsor/ -.

Instale a package haveged na máquina virtual com o seguinte comando: `sudo apt-get install haveged`.

Teste novamente os seguintes comandos, que vão obter 1024 bytes pseudoaleatórios do sistema e os apresentam em base64:

- `head -c 1024 /dev/random | openssl enc -base64`
- `head -c 1024 /dev/urandom | openssl enc -base64`

Que conclusões pode tirar? Em que se baseia para essas conclusões ?

Experiência 1.2

O exemplo utilizando o *java.security.SecureRandom* visto na aula, encontra-se na diretoria das aulas (Aula2/PseudoAleatorio), no ficheiro *RandomBytes.java*.

Analise, compile e execute este exemplo.

Pergunta P1.3

Na diretoria das aulas (Aula2/PseudoAleatorio) encontra o ficheiro *generateSecret-app.py* baseado no módulo

eVotUM.Cripto (https://gitlab.com/eVotUM/Cripto-py), já instalado na máquina virtual em /home/user/API/Cripto-py/eVotUM/Cripto.

Analise e execute esse programa de geração de segredo aleatório e indique o motivo do output apenas conter letras e dígitos (não contendo por exemplo caracteres de pontuação ou outros).

2. Partilha/Divisão de segredo (Secret Sharing/Splitting)

Experiência 2.1

O exemplo utilizando o *genSharedSecret.php* visto na aula, encontra-se na diretoria das aulas (Aula2/SecretSharing), no ficheiro com o mesmo nome.

Analise e execute este exemplo. Verifique o que acontece se tentar reconstruir o segredo (*reconstroiSecret.php*) com mais ou menos componentes do que as esperadas.

Experiência 2.2

O exemplo utilizando o *shares.pl* visto na aula, encontra-se na diretoria das aulas (Aula2/ShamirSharing), no ficheiro com o mesmo nome.

Analise e execute este exemplo. Verifique o que acontece se tentar reconstruir o segredo (*reconstruct.pl*) com mais ou menos componentes do que as esperadas.

Pergunta P2.1

Na diretoria das aulas (Aula2/ShamirSharing) encontra os ficheiros *createSharedSecret-app.py*, *recoverSecretFromComponents-app.py* e *recoverSecretFromAllComponents-app.py* baseado no módulo eVotUM.Cripto (https://gitlab.com/eVotUM/Cripto-py), já instalado na máquina virtual em /home/user/API/Cripto-py/eVotUM/Cripto.

A. Analise e execute esses programas, indicando o que teve que efectuar para dividir o segredo "Agora temos um segredo muito confidencial" em 7 partes, com quorom de 3 para reconstruir o segredo.

Note que a utilização deste programa é `python createSharedSecret-app.py number_of_shares quorum uid private-key.pem` em que:

- number_of_shares - partes em que quer dividir o segredo
- quorum - número de partes necessárias para reconstruir o segredo
- uid - identificador do segredo (de modo a garantir que quando reconstruir o segredo, está a fornecer as partes do mesmo segredo)
- private-key.pem - chave privada, já que cada parte do segredo é devolvida num objeto JWT assinado, em base 64

B. Indique também qual a diferença entre *recoverSecretFromComponents-app.py* e *recoverSecretFromAllComponents-app.py*, e em que situações poderá ser necessário utilizar *recoverSecretFromAllComponents-app.py* em vez de *recoverSecretFromComponents-app.py*.

Nota: Relembre-se que a geração do par de chaves pode ser efetuada com o comando `openssl genrsa -aes128 -out mykey.pem 1024`. O correspondente certificado pode ser gerado com o comando `openssl req -key mykey.pem -new -x509 -days 365 -out mykey.crt`

3. Authenticated Encryption

Cenário:

> A sua empresa quer colocar um serviço no mercado com as seguintes características:

- cifragem (com cifra simétrica) de segredos;
- decifragem do segredo previamente cifrado;
- o cliente pode decifrar o(s) segredo(s) que cifrou durante o tempo em que pagar a anuidade do serviço;
- de modo ao cliente saber o que cifrou, pode etiquetar o segredo.

> Para tal, a sua empresa adquiriu um hardware específico de cifra/decifra, em que a chave de cifra é automaticamente mudada todos os dias, sendo identificada por "ano.mes.dia". Esse hardware também efectua HMAC_SHA256 e tem uma API com as seguintes funções:

- cifra (segredo_plaintext), devolvendo segredo_cyphertext
- decifra (segredo_cyphertext, chave_cifra), devolvendo segredo_plaintext ou erro
- hmac (k, str), devolvendo o HMAC_SHA256 da str a autenticar com chave secreta k

Pergunta P3.1

Baseado no cenário identificado, como sugeriria à sua empresa que cifrasse e decifrasse o(s) segredo(s), de modo a garantir confidencialidade, integridade e autenticidade do segredo e da sua etiqueta? Inclua, na sua resposta, o algoritmo, que utilizando a API, pediria à área de desenvolvimento para implementar, de modo a cifrar e decifrar o segredo.

4. Algoritmos e tamanhos de chaves

O site https://webgate.ec.europa.eu/tl-browser/ disponibiliza a lista de Entidades com serviços qualificados de confiança, de acordo com o Regulamento EU 910/2014 (eIDAS) - falaremos deste Regulamento noutra aula -.

Entre esses seviços encontra-se o serviço de emissão de certificados digitais qualificados para pessoa física, designado por "QCert for ESig".

Pergunta P4.1

Cada grupo indicado abaixo deve identificar os algoritmos e tamanhos de chave utilizados nos certificados das Entidades de Certificação (EC) que emitem certificados digitais qualificados, e verificar se são os mais adequados (e se não forem, propor os que considerar mais adequados):

- Grupo 1 - Austria, para as três ECs que emitem certificados "QCert for ESig";
- Grupo 2 - Croácia, para as três ECs que emitem certificados "QCert for ESig";
- Grupo 3 - França, para as ECs "Ministère de la Justice", "Imprimerie Nationale", "Certinomis";
- Grupo 4 - Hungria, para as três ECs que emitem certificados "QCert for ESig";
- Grupo 5 - Itália, para as ECs "Banca d'Italia", "Ministero della Difesa", "Intesi Group S.p.A.";
- Grupo 6 - Lituania, para as três ECs que emitem certificados "QCert for ESig";
- Grupo 7 - Holanda, para as ECs "Ministerie van Defensie", "QuoVadis Trustlink B.V.", "KPN Corporate Market

B.V.";
- Grupo 8 - Portugal, para as ECs "Cartão de Cidadão", "ECCE", "Justica";
- Grupo 9 - Eslovénia, para as ECs "Ministry of Defence of Slovenia", "Republika Slovenija", "HALCOM D.D.";
- Grupo 10 - Bélgica, para as ECs "Zetes S.A./N.V.", "QuoVadis Trustlink BVBA", "Portima s.c.r.l. c.v.b.a.";
- Grupo 11 - Alemanha, para as ECs "Bundesagentur fuer Arbeit", "D-Trust GmbH", "Deutscher Sparkassen Verlag GmbH";
- Grupo 12 - Noruega, para as ECs "Commfides Norge AS", "Nordea Bank Norge ASA", "Statoil ASA";
- Grupo 13 - Roménia, para as ECs "CENTRUL DE CALCUL SA", "AlfaTrust Certification S.A.", "DigiSign S.A.";
- Grupo 14 - Espanha, para as ECs "Dirección General de la Policía", "Fábrica Nacional de Moneda y Timbre", "Consorci Administració Oberta de Catalunya - CAOC";
- Grupo 15 - Bulgária, para as ECs "Evrotrust Technologies JSC", "Information Services Plc.", "BORICA AD";
- Grupo 16 - República Checa, para as três ECs que emitem certificados "QCert for ESig".

Nota 1: Para Entidades de Certificação que já tenham vários certificados de EC, considere apenas o último certificado emitido.

Nota 2: Para obter o tamanho das chaves e algoritmos utilizados, deverá:

1. escolher o certificado da EC,
2. selecionar *Base 64-encoded*,
3. copiar o conteúdo do *Base 64-encoded* e gravar em ficheiro (por ex., cert.crt),
4. inserir -----BEGIN CERTIFICATE----- no inicio do ficheiro,
5. inserir -----END CERTIFICATE----- no final do ficheiro,
6. executar o seguinte comando `openssl x509 -in cert.crt -text -noout` (substitua cert.crt pelo nome que deu ao ficheiro no passo 3.)