Universidade de Aveiro

departamento de electrónica, telecomunicações e informática

47232- Segurança

IEDCS

Identity Enabled Distribution Control System

8240 - mestrado integrado em engenharia de computadores e telemática

António Rafael da Costa Ferreira
NMec: 67405 | P4G1

Rodrigo Lopes da Cunha
NMec: 67800 | P4G1

Docente: João Paulo Silva Barraca

Novembro de 2015
2015-2016

Conteúdos

1 Introdução 2
2 Processos apresentados na 1ª apresentação 3
2.1 Processo de cifra do conteúdo do livro 3
2.2 Processo de decifra do conteúdo do livro 4
3 Alterações efetuadas em relação à 1ª apresentação 5
3.1 Processo de cifra do conteúdo do livro 5
3.2 Cifra simétrica e modo 7
3.3 Visualização de um livro 7
3.4 Processo de decifra do conteúdo do livro 8
4 Tecnologias usadas 9
5 Casos de utilização 10
5.1 Registo de um utilizador 10
5.2 Registo de um device 10
5.3 Aquisição de um livro 11
5.4 Restrições na visualização de um livro 11
6 Instalação 11
7 HTTPS 12
8 Problemas identificados na primeira entrega 12
8.1 Soluções 12

1 Introdução

O trabalho proposto para o projeto da unidade curricular de Segurança é um IEDCS: Identity Enabled Distribution Control System. Para o efeito foi necessário implementar uma Ebook Webstore, um WebService e um Player de reprodução dos Ebooks em formato de texto.

O objetivo deste sistema é garantir a máxima e possível segurança do serviço, utilizando os conhecimentos adquiridos na unidade curricular de Segurança. Para isso são necessários vários processos como por exemplo, a utilização de certificados HTTPS, a cifragem de todo o material existente, geração de chaves e registo de utilizadores.

O relatório reflete todos os passos e decisões tomadas na criação do sistema, assim como uma análise ao que foi mostrado na primeira apresentação e decisões que se tomaram depois desta, tecnologias utilizadas, descrição dos vários processos existentes e conclusão.

2 Processos apresentados na 1ª apresentação

2.1 Processo de cifra do conteúdo do livro

Na primeira apresentação foram concebidos dois processos que foram abandonados após a sua apresentação. O primeiro processo apresentado foi o "Book content cipher process". Este pretendia fazer uso de cifras simétricas e assimétricas de forma a ser possível obter uma File Key que seria usada para cifrar o conteúdo do livro. Idealmente o conteúdo do livro seria cifrado com a chave privada do Player, depois cifrado com a chave pública do Device e finalmente seria feito sobre esses resultados uma cifra com o User Key e a File Key. O processo de cifra com o User Key e a File Key seriam repetidos n vezes que seria obtido através do random2. Foi pensado também usar duas funções para derivar as keys, uma f1 que seria usada para derivar a User Key e uma f2 que seria usado para derivar a File Key.

A ideia de usar uma função de derivação da File Key e da User key é boa, e usando um número de iterações também. Isto permite que quem esteja a ouvir na rede tenha mais dificuldades em obter uma chave usando aquilo que conseguiu ouvir, teria de conhecer o processo usado pelos intervenientes para conseguir obter as chaves. Digamos que também não seria difícil perceber o processo usado pelos dois intervenientes pois basta ao atacante usar o serviço e obter um Player e usando ferramentas próprias analisar o código.

A causa de termos abandonado este processo para cifrar o conteúdo do livro foi:

- É computacionalmente exigente cifrar com chaves assimétricas conteúdos grandes.
- Normalmente para evitar este forço computacional faz-se uma assinatura sobre o resultado de uma função de síntese desse mesmo conteúdo do livro.
- As chaves simétricas são muito mais eficientes a nível computacional.
- As funções para derivar as chaves foram abandonadas em detrimento da PBKDF2 que efetua já as iterações idealizadas.
- Não é boa ideia usar as passwords dos utilizadores para cifrar o conteúdo do mesmo pois caso o utilizador mude a palavra pass teremos de derivar as chaves outra vez.

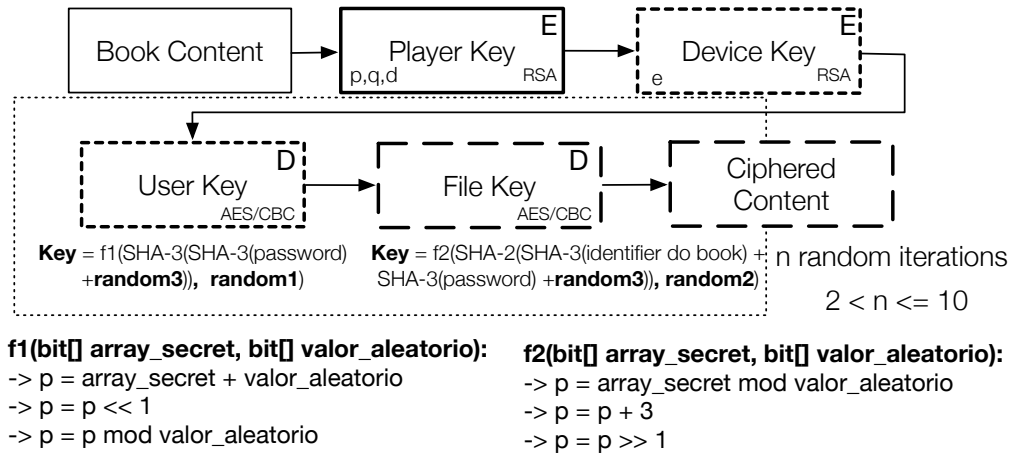
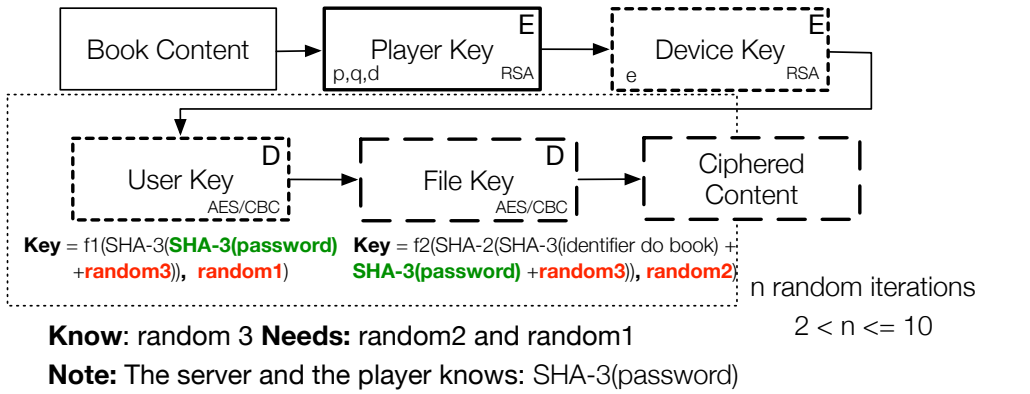


Figura 1:
Processo de cifra usado na 1º apresentação

2.2 Processo de decifra do conteúdo do livro

O processo de decifra foi desenhado para reverter o processo anunciado em cima. Sendo por isso nada mais do que fazer as operações inversas das realizadas no processo de cifra. As razões para que se tenha abandonado este processo de decifra são as mesmas enunciadas no processo de cifra.

O processo de "challenge" consistia no pedido do *random1* e do *random2* ao servidor, mas, contudo isso não é suficiente para que o processo seja seguro, pois basta ao atacante conhecer o processo de cifra e decifra e interceptar os pedidos ao servidor do *random1* e do *random2*. Para o Player também seria fácil decifrar sem necessitar da cooperação do servidor, pois, bastava que coopera-se uma primeira vez.



f1(bit[] array_secret, bit[] valor_aleatorio):
-> p = array_secret + valor_aleatorio
-> p = p << 1
-> p = p mod valor_aleatorio

f2(bit[] array_secret, bit[] valor_aleatorio):
-> p = array_secret mod valor_aleatorio
-> p = p + 3
-> p = p >> 1

Figura 2:
Processo de decifra usado na 1º apresentação

3 Alterações efetuadas em relação à 1ª apresentação

No seguimento da primeira apresentação procedeu-se a muitas alterações no modo como o processo de cifra e decifra do conteúdo de um livro seria efetuado.

3.1 Processo de cifra do conteúdo do livro

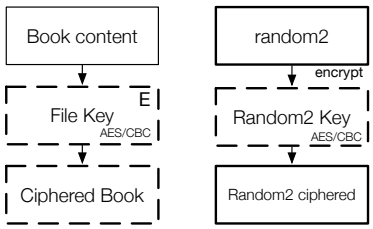


Figura 3:
Processo de cifra de um livro

Neste novo processo, o servidor cifra um valor aleatório, designado por Random2, com uma chave Random2 Key, calculada através da seguinte

forma:

Random2 Key = PBKDF2(username + "jnpc" + book.identifier, book.identifier, 1000)

Esta key é partilhada pelo servidor e pelo cliente, pois ambos conhecem a forma de como ela é calculada, não sendo preciso enviar a mesma. Apenas o conteúdo de Random2 cifrado é enviado para o cliente. O valor Random2 foi criado, para através dele tanto o servidor como o cliente obterem a File Key que será utilizada para cifrar o conteúdo de um livro e que não poderá ser guardada do lado do cliente e poderá ser reutilizada para um mesmo utilizador noutro Player ou noutro Device.

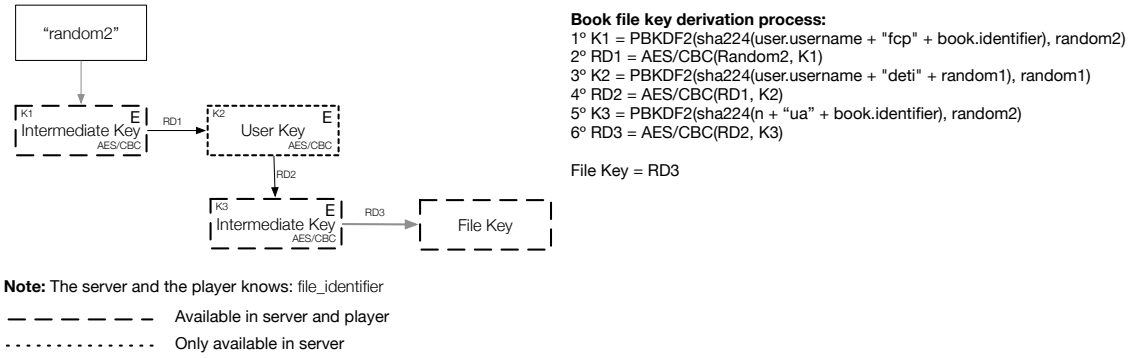


Figura 4:
Processo de geração da File Key

Para derivar a File Key, como poderemos ver na figura 4, o Random2 é cifrado com uma chave K1, que é calculada através do SHA-224 do username, de um salt ("fcp") e do identificador do livro. Este SHA é a nossa password e o Random2 o nosso slat aplicando-se assim a função PBKDF2 (recomendado na apresentação), com o valor por defeito de 1000 iterações, obtendo-se assim a chave pretendida, K1. O cálculo desta chave K1 é possível de ser efetuado tanto do lado do servidor como do cliente.

Após isto, o resultado obtido da cifra de Random2 com a chave K1, RD1, vai ser cifrado com uma nova chave K2, sendo esta apenas e exclusivamente calculada pelo servidor. Esta é calculada praticamente como a K1, sendo diferente por no SHA-224 utilizar um outro salt ("deti") e um *random1*, em vez de *random2*, apenas disponível no servidor, em vez de usar o identificador do livro. Aplica-se igualmente a função PBKDF2, com o SHA como password e o *random1* como salt, de forma a obter K2 para se poder cifrar o valor de RD1.

Numa última fase deste processo de geração da File Key, é calculada ainda uma terceira chave, K3, com o mesmo processo das outras chaves K1 e K2, sendo nesta última utilizado no calculo do SHA um valor n, que corresponde a:

n = (soma de todos os digitos de random2) % 10

Juntamente com o valor n, utiliza-se o identificador do livro e um salt ("ua"). Através deste SHA e do random2 que actuam respectivamente como password e salt, aplica-se uma função PBKDF2 para obter K3.

Por último através da cifra de RD2 (valor obtido da cifra de *random1* com a chave K2) com a chave K3 obtida, obtém-se a File Key, que será utilizada para cifrar o conteúdo do livro e desta forma este conteúdo cifrado será enviado para o cliente.

3.2 Cifra simétrica e modo

Todo o processo de cifra e decifra de um livro e de um random2, foi efetuado através da cifra simétrica por blocos AES com o modo de cifra CBC.

Foi escolhida a cifra simétrica AES, pois cifra por blocos e garante maior rapidez tanto no processo de cifra como de decifra.

O modo de cifra CBC foi selecionado devido à existência de confusão, introduzida por um IV no inicio do processo de cifra e decifra. Cada bloco cifrado funciona como realimentação para o bloco seguinte, exceto no inicial, onde é utilizado o IV. Isto permite ter acesso aleatório uniforme apenas na decifra, que é o pretendido para o nosso método de paginação, sendo então possível decifrar o bloco que desejamos, sem termos de processar os anteriores.

3.3 Visualização de um livro

Na primeira vez que um utilizador solicita a visualização de um livro é gerado o primeiro Random2 e Random1 que vai ser único para determinado utilizador e livro que tenha adquirido o livro. Estes valores aleatório são guardados na base de dados cifrados com uma chave que depende do utilizador e do livro.

Todos os Initialization Vectors são também guardados na base de dados com o mesmo processo.

Os Random2, Random1 e Initialization Vectors passam todos por um processo de cifra antes de serem guardados na base de dados. Este processo consiste em:

- Gerar um *identifier*, através de um SHA-224 composto por um alias (nome pelo qual queremos identificar o conteúdo, por exemplo "random2"), username e identificador do livro.
- Gerar um valor que será a nossa password na função PBKDF2, através de um SHA-224, composto pelo username, email e último nome.
- Depois de gerados os dois valores acima descritos, recorre-se a uma função PBKDF2, onde o salt é o *identifier*. Esta função terá um número de 1000 iterações por defeito.
- Posto isto, o conteúdo que desejamos guardar na base de dados é cifrado com a chave gerada acima, ficando assim salvaguardado com alguma segurança.

3.4 Processo de decifra do conteúdo do livro

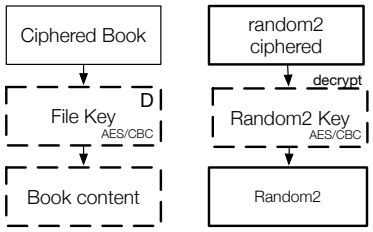


Figura 5:
Processo de decifra de um livro

Tal como no processo de cifra de um livro, a Random2 Key é calculada, desta feita no lado do cliente para este poder decifrar o valor de random2 que recebe cifrado, obtendo assim o valor original que vai ser utilizado para o cálculo da File Key usando também um processo de challenge.

O processo de geração da File Key no lado do servidor e do cliente é feito da mesma maneira, apenas mudando, que quando o Player está a gerar a File Key, precisa de colocar um challenge ao servidor (Figura 7), pois ele é o único capaz de calcular o valor de RD2, obtido através da cifra de RD1 (cifra de Random2 com K1) com uma chave K2 calculada com um random1 que só o servidor conhece (Figura 5).

Posto isto, o servidor retorna, o valor que calculou, para o cliente, para que este prossiga o processo de geração da File Key.

Após este processo de geração da File Key o Player decifra o conteúdo cifrado do livro de forma a obter o conteúdo original do mesmo, podendo agora apresentá-lo ao utilizador.

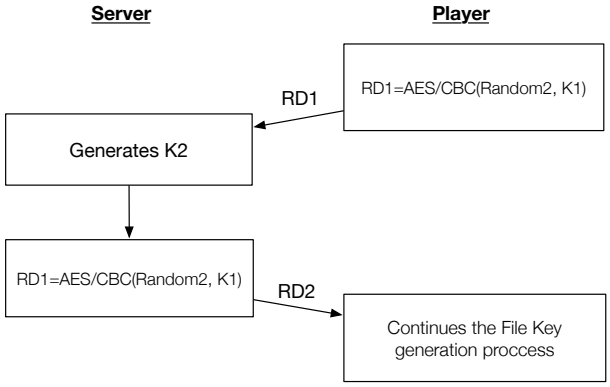


Figura 6:
Challenge entre Player e Servidor na geração da File Key

4 Tecnologias usadas

As tecnologias usadas no desenvolvimento do IEDCS Server, foram: Django + Python e AngularJS + JS. O Django dá a possibilidade de usar protecção contra CSRF usando um token e um middleware que obriga que o cliente envie este token sempre que faz um pedido para a aplicação.

O Angular.JS já tem suporte para enviar o CSRF token, sempre que recebe um *Set-Cookie CSRFToken = XXX* na resposta do servidor, irá gravar esse csrf token num cookie no browser do utilizador e irá enviar sempre que for feito um pedido.

Já o nosso Player foi feito usando Java e JavaFX. No java tivemos de usar uma biblioteca para fazer os pedidos HTTP e HTTPS exterior, neste caso usámos a org.apache.http. Esta biblioteca não tem suporte para uso de um CSRF de forma correta, por isso, tivemos de implementar uma classe (Requests.java) que gerisse uma ligação HTTP e guardasse o seu estado de sessão. Nesta classe enviamos um X-CSRFToken no Header do pedido.

No servidor, também temos a proteção contra SQL Injection e XSS ataques pois fazemos uso dos métodos disponibilizados pelo Django para manipulação de modelos da base de dados. Já o AngularJS não permite que seja injetado código JS nem HTML sem intenção do programador da aplicação.

Como usamos Python e Java já temos protecção contra buffer overflows e mesmo assim tivemos cuidado para não fazer implementações que comprometessem o estado tanto do servidor como do player, como por exemplo, uma saída inesperada devido a um erro.

5 Casos de utilização

5.1 Registo de um utilizador

Um registo de um utilizador é feito na página web disponibilizada pelo IEDCS, não é feito nenhuma síntese quando se envia desde o Javascript para o servidor, sendo este um dos problemas identificados por nós. No entanto, ao gravar na base de dados a password é gravada com PBKDF2 e SHA256 o que torna computacionalmente trabalhoso obter as passwords dos nossos utilizadores caso a base de dados seja exposta. Ao fazer login, tanto no Player como na interface Web também poderia ser aplicado um processo de síntese para caso o HTTPS seja comprometido as passwords não sejam enviadas em claro pela rede.

Não existe nenhuma criação da chamada UserKey, esta no entanto, é gerada para cada livro por cada utilizador quando ele solicita pela primeira vez a leitura do mesmo. Optou-se por fazer um UserKey por Livro para dificultar ainda mais a tarefa a quem tente comprometer o nosso sistema de IEDCS.

	id	password	last_login	email	username	first_name	last_name	created_at	updated_at
	...	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	pbkdf2_sha256$15000$aoPM4j5hSUZG$iTF...	2015-11-12 21:39:38.927...	mail@rafa...	rafaelferreira	Rafael	Ferreira	2015-11-07 23:17:21.586...	2015-11-07 23:17:21.587...

Figura 7:
Password guardada com PBKDF2 e SHA256

5.2 Registo de um device

Quando um player é utilizado pela primeira vez num determinado device, no momento do login de um utilizador, o Device é registado na base de dados com os seguintes parâmetros:

- Identificador único
- Modelo de CPU
- Modelo do PC
- IP
- País
- Hora
- Host name

- Chave pública do Device

No Player é usado a biblioteca sigar para obter os dados do device. O País é obtido no servidor através do IP recebido pelo mesmo. A cada utilizador também são associados os vários devices por onde já se conectou ao servidor. Caso no momento do login, o device já exista na base de dados, este é actualizado com os novos dados recebidos, tal como a hora, IP, entre outros já especificados.

5.3 Aquisição de um livro

No momento de aquisição do livro não há nenhuma cifra, apenas um registo na base de dados que o utilizador adquiriu um livro.

5.4 Restrições na visualização de um livro

Para a visualização de um livro criaram-se dois tipos de restrições, por país e por hora de utilização.

Isto permite que os conteúdos não possam ser acedidos a qualquer momento e em qualquer lado caso existam restrições nesse livro.

6 Instalação

Para uso e instalação do nosso sistema, criou-se uma pasta /installation que tem um ficheiro *Vagrantfile* que permite criar um servidor ubuntu trusty 64bits que cria uma pasta partilhada neste em /var/www que irá servir para lançar o servidor IEDCS com recurso ao Apache2. O ficheiro *server-bootstrap.sh* servirá para instalar todas as bibliotecas necessárias, todos os pacotes do python, a configuração do Apache2 para o nosso site que está no ficheiro *iedcs.rafaelferreira.pt.conf* e *default-ssl.conf* e também para instalar os certificados necessários para o HTTPS nas pastas usadas para esse efeito.

O ficheiro *iedcs.rafaelferreira.pt.conf* serve apenas para redirecionar todos os pedidos HTTP para HTTPS. O ficheiro *default-ssl.conf* tem todas as configurações necessárias para o uso de HTTPS e para usar o modWSGI do Apache2 para o Django.

Criou-se um package de settings para o Django para ter definições de development e outras de servidor. Usando o hostname do servidor sabemos em que modo deve estar, sendo assim nesse servidor iniciado com Vagrant o servidor estará em modo produção, ou seja, não irá lançar output de debug para o utilizador.

O *install_cert.sh* é usado para criar uma keystore que será usada pelo Java para armazenar os certificados em que ele confia ao usar HTTPS.

Para usar o servidor, além de ter o Vagrant intalado, é apenas necessário fazer: vagrant up

A password caso seja pedida é sempre vagrant e para aceder à máquina é apenas necessário fazer vagrant ssh.

O site pode ser acessível em: `https://iedcs.rafaelferreira.pt/`

7 HTTPS

Para criar um certificado criou-se uma chave privada RSA com:
openssl req -new -newkey rsa:2048 -nodes -keyout server.key -out server.csr
output: /installation/ssl/generating.txt

O certificado foi depois emitido usando a Free SSL.

Como foi criado um domínio que aponta para o endereço *192.168.33.10* que é o endereço local do servidor IEDCS inicializado pelo Vagrant.

8 Problemas identificados na primeira entrega

Nesta primeira entrega, identificaram-se alguns problemas, nomeadamente no corrompimento do HTTPS, através da troca da keystore onde são guardados os certificados confiáveis, podendo assim injetar todos os certificados que se deseja de forma a que seja possível interceptar as comunicações entre o servidor e o Player. Para isso sabemos que o utilizador pode descompilar o código do Player, analisar o nosso processo de cifra e o código. Com isto ele poderá descobrir que usamos um ficheiro cacerts.keystore para o Java confiar apenas no nosso certificado, pode então modificar essa keystore e o cliente ao fazer download pode ser-lhe dada uma versão corrompida, mas sem estar corrompida no código, apenas no cacerts.keystore. Se estiver presente outro certificado que o atacante domine, ele pode interceptar as comunicações HTTPS e aceder às comunicações que estão a ser trocadas entre a vítima que está a usar o Player e o nosso Servidor e assim, pode, obter o Random2 e derivar o n (este é enviado inicialmente num Header), o book identifier e o rd2 que é enviado pelo servidor no pedido de challenge ao cliente. O salt "ua" é possível de ser obtido descompilando o código. Após isto, ele pode cifrar com AES e obter a file key. Com isto, consegue decifrar o nosso livro cifrado.

Outra possibilidade seria um atacante/cliente que usasse ferramentas de análise de memória para obter o livro cifrado, e todas as outras variáveis que estão expostas em Runtime.

8.1 Soluções

- Assinar o Player com uma chave assimétrica assim como o ficheiro cacerts.keystore.
- Autenticando ambas as entidades no momento em que é negociada o HTTPS entre o cliente e o servidor (2º parte do trabalho).

- Sessão para a file key usada ser sempre diferente tendo em conta a sessão, proibindo a reutilização da file key em vários devices e players.
- Cifra por página, ou seja, a paginação ser realizada no lado do servidor.