Instituto Superior de Engenharia de Lisboa

Engenharia Informática e de Computadores

SEGURANÇA INFORMÁTICA

2ºSÉRIE

Docente: José Simão

Filipe Fé nº 42141

Inês Gomes nº 42160

José Cunha nº 43526

Índice

Exercício 1

1.1)

Estamos perante o caso em que tanto o cliente como o servidor se querem autenticar, ou seja, existe uma autenticação mútua. No contexto do protocolo TLS, os materiais criptográficos necessários que têm de ser configurados no cliente:

- Um conjunto de certificados raiz de forma a verificar se o certificado enviado pelo servidor, que contém a sua chave pública, é válido.
- O seu próprio certificado que contém a sua chave pública
- A sua chave privada para realizar uma assinatura que comprova a sua autenticidade

1.2)

No processo de handshake, após o servidor enviar para o cliente o seu certificado (que contém a sua chave pública), o cliente irá gerar um pre master secret que será cifrado utilizando a chave pública recebida, caso o certificado recebido seja válido. Esse pre master secret cifrado irá ser enviada ao servidor (desafio) de modo a verificar que este possui a sua chave privada.

O servidor obtém o pre master secret descodificando a mensagem recebida, utilizando a sua chave privada.

Após este processo, este será usado como chave no esquema simétrico de autenticação MAC , de forma a provar que, de facto, possui a chave privada.

Depois de ser estabelecida uma chave segura (pre master secret) a ser usado por ambos é garantido que o canal de comunicação é fiável.

1.3)

O record protocol é re—sponsável pela transferência de blocos de dados entre os dois intervenientes na comunicação. Através dos parâmetros negociados no handshake, é gerada uma marca de autenticidade (através do MAC) e os dados são encriptados para futuramente serem enviados ao cliente ou ao servidor.

O problema é que um atacante pode dedicar-se a tentar descobrir o padding válido usado nas mensagens encriptadas por um desses intervenientes, como é descrito em ataques baseados no de Vaudenay. Para esse efeito basta guardar as mensagens que foram trocadas entre o cliente e o servidor e mais tarde usar a técnica descrita. Desta forma consegue obter a chave usada no processo de cifra. Caso o servidor tenha essa chave do seu lado e esta seja de longa duração, neste momento é possível ao atacante decifrar as mensagens que são enviadas nesse canal de comunicação (até a chave ser alterada).

Exercício 2

2.1)

O cliente/relying party indica os recursos a que pretende ter acesso através de um pedido HTTP, cujo URL contém um query parameter ***scope*** com essa informação.

2.2)

A framework OAuth 2.0 é usada para garantir autorização no acesso a determinados recursos.

Ao fazer uso dele para autenticação, não existe um processo de asserção sobre o utilizador que se pretende autenticar, pois apenas é gerado um access token que garante acesso temporário aos recursos. Ao usar esse token para autenticação não existe a garantia de que é o mesmo utilizador responsável pelo seu pedido.

Exercício 3

3.1)

O id token (fornecido pelo Identity-Provider à aplicação web) tem o propósito de informar a aplicação web de que o cliente se autenticou.

3.2)

A aplicação cliente.

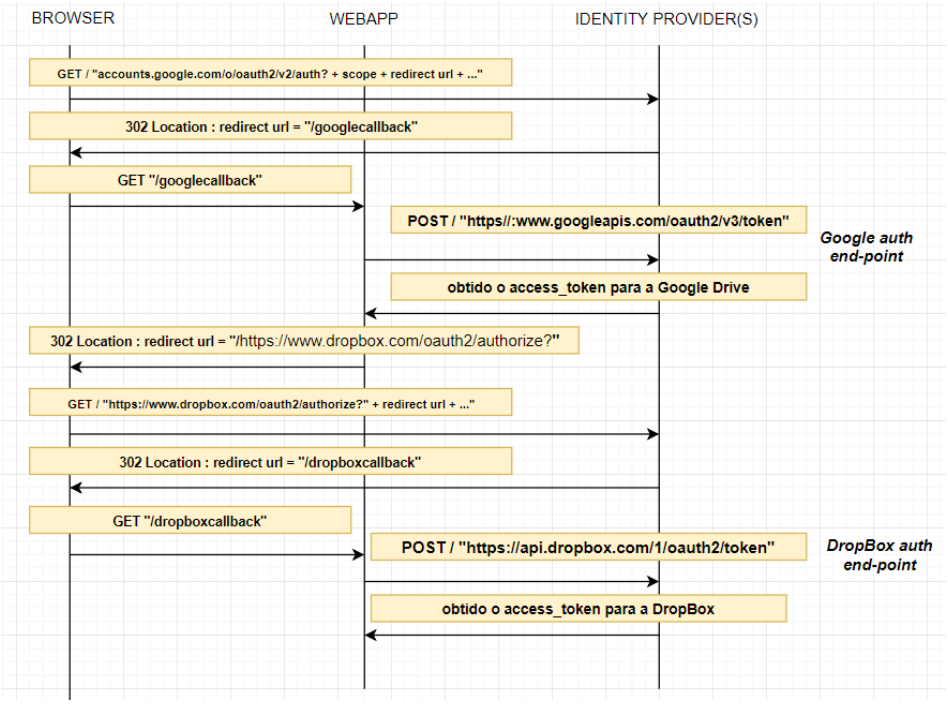
Exercício 6

O salt é um número aleatório que é adicionado à password submetida pelo utilizador e tem como objetivo tornar mais seguro o processo de encriptação.
Dado que o hash é calculado sobre a password e o salt, é muito improvável que duas passwords iguais tenham o mesmo hash (o salt é gerado aleatoriamente para cada password).
Assim, um eventual atacante que tenha acesso à base de dados onde se encontram os hash's gerados a partir das passwords e use o "ataque dicionário", terá maior dificuldade em obter uma password.
O resultado do hash da password é passado recursivamente à mesma função de hash num certo número de iterações. Tal permite tornar o processo de encriptação da password ainda mais seguro porque futuros "ataques dicionário" serão bastante ineficientes devido ao tempo que o ataque levará.

Exercício 7

Depois de termos procedido ao registo de aplicações quer na Dropbox quer na Google Drive começámos a desenhar a nossa aplicação web da seguinte forma:

- Rota home "/" que apenas tem disponível um botão onde podemos proceder ao login;
- Rota de login "/login" responsável por redirecionar para a o endpoint de autorização da Google onde é indicado no URL de autenticação os seguintes parâmetros:
  - CLIENT_ID identificando o utilizador que pretende a autorização:
  - scope onde é indicada a informação de que se quer ter acesso aos recursos, nomeadamente o e-mail e autorização para se poder ler ficheiros da Google Drive (isto indica que o pedido de autenticação e autorização aos recursos da drive foram feitos simultaneamente, tal como é indicado no scope)
  - response_type para authorization code grant
  - URL de redirect "/googlecallback" onde irá ser pedido futuramente o token para acesso aos recursos.
- Rota "/googlecallback" onde é efetuado o pedido HTTP POST para o URL https://www.googleapis.com/oauth2/v3/token com o objetivo de obter o access_token que disponibiliza acesso aos recursos previamente requisitados. Após obtenção de resposta, como redirecionados para o end point de autorização da DropBox.
- Rota "/dropboxcallback" onde é efetuado o pedido HTTP POST para o URL https://api.dropbox.com/1/oauth2/token com o objetivo de obter o access_token que disponibiliza acesso aos recursos da DropBox. Quando obtido o token de acesso redirecionamos o utilizador para a rota "/googleDriveFiles" onde são listados os seus ficheiros presentes no seu repositório do Google Drive.



Após obtenção dos token para acesso aos recursos já podemos fazer os pedidos de GET ou UPLOAD consoante a escolha do ficheiro feita pelo utilizador.

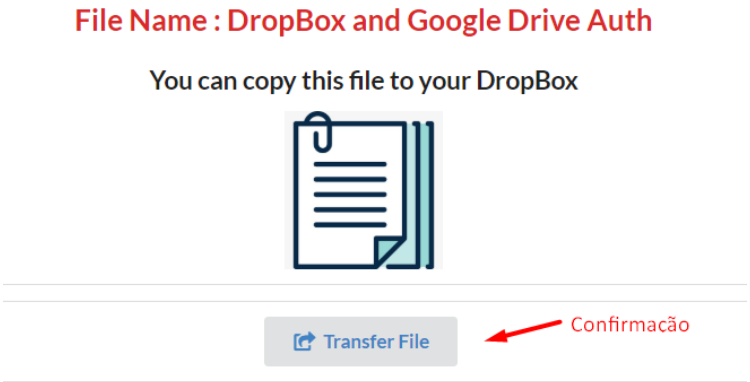
NOTA : Após obtenção dos access tokens da Google Drive e da DropBox, são criados dois Cookies onde é guardado um id que é gerado de forma aleatória para cada utilizador. Estes id's são usados como chave nos objetos "map_access_token" e "map_dropbox_access_token" de forma a relacionar cada id de um utilizador em particular com o token que obteve após autorização. Desta forma é garantido que os tokens são utilizados pelo cliente que os obteve e não por terceiros.
Caso um atacante altere a informação de um desses cookies, no processo de verificação será descoberta essa mesma alteração, visto que as marcas geradas pelo esquema simétrico MAC serão diferentes.

```
let h = crypto.createHmac('sha256', 'si-serie-2').digest(id)
// convert to base64
let hBase64 = Buffer.from(h).toString('base64');

resp.setHeader('Set-Cookie', [DRIVE_COOKIE_ID+"="+id, DRIVE_TAG_ID + "=" + hBase64]);
```

- Rota "/googleDriveFiles" onde são listados todos os ficheiros disponíveis na Google Drive do cliente. Desta forma, o cliente visualiza os ficheiros e decide qual o destinado à cópia para a sua Dropbox;

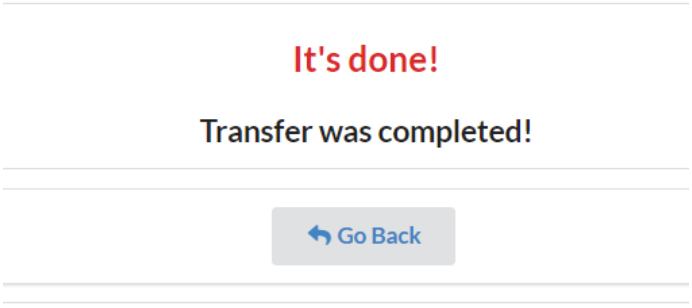
- Rota "/googleDrives/file" onde é pedido ao utilizador que confirme a transferência do ficheiro que selecionou para a DropBox; é recebido como query parameter o id e o nome do ficheiro selecionado pelo utilizador.



- Rota "/uploadFile/file" onde é feito o pedido GET https://www.googleapis.com/drive/v2/files/ + identificador do ficheiro seleciona. A resposta deste pedido permite obter os meta-dados do ficheiro, sendo um deles o URL de download. De seguida é efetuado um pedido GET para esse mesmo URL com o objetivo de obter o conteúdo em bytes do ficheiro (sendo este conteúdo guardado num ficheiro temporário na diretoria "../temp" para efeitos de debug na nossa aplicação). No fim da escrita, é feita uma leitura do conteúdo desse ficheiro temporário de modo a poder passar o resultado dessa leitura a função *uploadDropBox* que tem a responsabilidade de criar o novo ficheiro no repositório da DropBox.

- Função *uploadDropBox* onde é feito um POST para o URL https://content.dropboxapi.com/2/files/upload cujo body vai preenchido com o conteúdo do ficheiro que queremos copiar. De salientar que é necessário enviar nos headers deste pedido o *content-type* com *application/octet-stream* de modo a enviar o conteúdo do ficheiro em bytes e o header *DROPBOX_API_Arg* conforme especificado na documentação. O *access token* é também enviado.

- Se a cópia teve sucesso somos redirecionados para o URL "/updateDone" onde é permitido ao utilizador voltar para a lista de ficheiros e selecionar outro destinado a uma futura cópia.



NOTA:
Para evitar que um possível atacante altere o redirect URL da aplicação.

```
`state=${session_id}&`
```

```
if(!states[key])
    return resp.redirect(302,'/login')
```

É adicionado o query parameter state que de seguida será testado em ambos os callbacks, de modo a garantir que o redirect será feito para uma rota confiável .