INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

SEGURANÇA INFORMÁTICA

LICENCIATURA ENGENHARIA INFORMÁTICA E COMPUTADORES

FASE DE EXERCÍCIOS
Fase 1

Autores:
43552 - Samuel Costa
43320 - André Mendes

Docente:
José Simão

25 de Novembro de 2019

Conteúdo

Introdução

O trabalho realizado para este fase pretende que os temas desenvolvidos durante as aulas sejam postos em prática. Para esta fase os exercícios focaram essencialmente a segunda parte da matéria.

- Autenticação baseadas em passwords
- Gestão de Identidade em Aplicações Web
- Modelos de Controlo de Acesso

Neste trabalho prático pretendemos responder aos vários exercícios propostos e implementar uma demonstração dos pontos referidos.

1 Exercício 1

1.1

Um esquema de assinatura as chaves usadas são assimétricas o que permite a que autentica a mensagem usa a sua chave privada para o efeito e o recetor usa a chave publica do emissor para a autenticar essa mesma mensagem. Um esquema Mac como a chave é simétrica numa fase inicial através de um canal seguro teriam que ser trocadas para que desta forma ambos os intervenientes tenham a mesma chave para verificar a autenticidade da mensagem.

1.2

As recomendações do texto abordam fragilidades de segurança no caso de um atacante conseguir obter a chave privada de um servidor durante a sessão estabelecida entre o cliente e o servidor. No caso de um atacante obter a chave privada de um servidor passa a poder obter todas as mensagens dirigidas a ele, mas com a nova diretiva obriga a que a cada sessão durante a fase de *handshake* sejam estabelecidas novas chaves o que faz com que a chave do atacante deixe ter utilidade.

2 Exercício 2

O algoritmo simétrico para cifrar a *password* de um utilizador é criado através da função de *hash* com um valor aleatório unico a cada utilizador prevenindo assim que ataques de dicionario, ou seja, *passwords* iguais passam a ter valores de *hash* diferentes tornando a sua chave unica para cada utilizador. Para realizar a decifra realiza se o mesmo processo à *password* introduzida pelo utilizador, usa se o mesmo valor de *salt* e realiza se a comparação devolvendo um verdadeiro ou falso.

3 Exercício 3

A criação de um *cookie* passa por vaias fases até ser guardado no lado do cliente para garantir a autenticidade durante a sua sessão sem que este tenha que validar as suas credencias constantemente. O processo passa pelos seguintes passos:

1. O utilizador garante a sua autenticidade através de credenciais introduzidas no lado do servidor ou através de um redirecionamento para um fornecedor de identidade.
2. O utilizador é então autenticado recebendo um código.
3. É novamente redirecionado para o servidor sendo que este troca o código por um *id_ token*.
4. O servidor realiza a operação de *setCookie* por forma a que o utilizador nas proximas chamadas envie o *cookie* para que este não tenha a necessidade de se autenticar.
5. O servidor autenticou o *cookie* e de todas as vezes realizar uma operação de autenticação para o confirmar.

4 Exercício 4

4.1

O parâmetro *scope* refere-se ao access token a ser gerado e enumera o acesso que a aplicação cliente pretende ter.

4.2

O client id e o client secret servem para o relying party se autenticar junto do fornecedor de identidade, e, como tal, é usado num pedido POST originado no relying party para o fornecedor de identidade. Portanto, nunca devem passar pelo browser.

4.3

Indirectamente sim. No entanto, depois de um utilizador se autenticar junto do servidor de autorização, é gerada uma resposta com código 302 (redirect), para o browser com o callback da aplicação cliente no header location e com CODE. o browser gera um pedido get para o callback da aplicação cliente, que recebe o CODE, e o envia para o token endpoint do servidor de autorização junto com o client id e o client secret. Em resposta, o servidor de autorização envia o ID Token, que, de acordo com o parâmetro scope, inclui a informação de identidade a que utilizador deu acesso à aplicação cliente.

5 Exercício 5

O ID Token é obtido quando o relying party acede ao token endpoint, onde troca CODE pelo ID Token. É um objecto estruturado, assinado e cifrado e serve ao relying party para garantir que o utilizador se autenticou junto do itentity provider, contendo informação de perfil, que pode ser usada para personalizar a experiência do utilizador. Este também pode ser usado pelo relying party para associar um utilizador ao seu autenticador, sendo uma má prática publicá-lo directamente no autenticador a entregar ao utilizador.

6 Exercício 6

i) Primeiro, efectuou-se a autenticação do servidor, usando o browser como aplicação cliente:

1. Foi gerado o ficheiro PEM com chave privada do servidor 'secure-server-private.pem' através do comando: openssl pkcs12 -in secure-server.pfx -out secure-server-private.pem -nodes
2. Foi gerado o ficheiro PEM com certificado do servidor 'secure-server.pem' através do comando : openssl x509 -inform der -in secure-server.cer -out secure-server.pem
3. Foi adicionado ao ficheiro `C:\Windows\System32\drivers\etc\hosts` a linha '127.0.0.1 www.secure-server.edu'
4. Foram comentadas na aplicação servidora as linhas

```
ca: fs.readFileSync('CA1.pem'),
requestCert: true,
rejectUnauthorized: true
```

5. Acedeu-se a `https:\www.secure-server.edu:4433`

Em seguida, efectuou-se, ainda usando o browser como aplicação cliente, a autenticação de cliente da seguinte forma:

1. Gerou-se o ficheiro com o certificado PEM da CA root através do comando openssl x509 -inform der -in CA1.cer -out CA1.pem
2. Instalaram-se os quatro certificados-folha com o Assistente para importar certificados do sistema operativo.
3. Acedeu-se a `https:\www.secure-server.edu:4433` e obteve-se a resposta 200 OK, com a string 'Secure Hello World with node.js'.

ii) Foi desenvolvida a aplicação cliente Java constante de HttpsClient.java. Foram gerados os KeyStores (.jks) com a chave privada do cliente (client.jks) e com os certificados das raízes de confiança (trustedroots.jks). Para gerar o primeiro ficheiro, executaram-se os comandos:

1. keytool -importkeystore -srckeystore Alice_1.pfx -srcstoretype pkcs12 -destkeystore client.jks -deststoretype JKS
2. keytool -changealias -alias "1" -destalias "Alice_1" -keystore "client.jks"

Para gerar o segundo ficheiro, executaram-se os comandos:

1. keytool -importcert -file "CA1.cer" -keystore trustedroots.jks -alias "CA1"
2. keytool -importcert -file "CA1-int.cer" -keystore trustedroots.jks -alias "CA1-int"

iii) Foi testada a aplicação desenvolvida, com e sem autenticação de cliente. Para desabilitar autenticação de cliente foram comentadas na aplicação cliente as linhas

```
ca: fs.readFileSync('CA1.pem'),
requestCert: true,
rejectUnauthorized: true
```

Na aplicação cliente, foi desabilitado o carregamento do KeyStore com o certificado do cliente.