

Instituto Superior de Engenharia de Lisboa
Licenciatura em Engenharia Informática e de Computadores
Licenciatura em Engenharia Informática, Redes e Telecomunicações
Segurança Informática
Segunda série de exercícios, Semestre de Inverno de 18/19
Entregar até 25 de novembro de 2018

FINAL 1. No contexto do protocolo TLS:

- 1.1. Qual o material criptográfico (certificados e chaves) que têm de ser configurados no cliente caso seja necessário autenticação de cliente e de servidor?
- 1.2. Qual o esquema simétrico usado no *handshake* do TLS e quais os objetivos da sua utilização?
- 1.3. Qual a característica do *record protocol* que o torna susceptível a ataques baseados no de *Vaudenay*?

2. No contexto da *framework* de autorização OAuth 2.0 :

- 2.1. Como é que o cliente/*relying party* especifica os recursos a que pretende ter acesso?
- 2.2. Quais as limitações da utilização deste protocolo para autenticação?

FINAL 3. No contexto do fluxo *authorization code* do protocolo OpenID Connect:

- 3.1. Para que serve o ID Token?
- 3.2. Qual destas duas entidades desempenha o papel de *relying party*: a aplicação cliente ou o *resource server*?

4. Considere o modelo RBAC₁ (RBAC com hierarquia de *roles*).

- 4.1. É possível existir uma sessão associada ao utilizador *u* e com o *role r* activo, sem que (*u, r*) esteja na relação *user assignment* (UA)?
- 4.2. Qual a relação entre o princípio de privilégios mínimos e o conceito de sessão na família de modelos RBAC?

FINAL 5. Realize um programa que, dado o *hostname* para um *site* com suporte HTTPS, apresenta:

- A cadeia de certificados do servidor.
- A menor data de expiração dos certificados do servidor.
- As versões dos protocolos SSL e TLS suportadas (de entre as disponíveis na plataforma Java).

6. Adicione ao programa desenvolvido na alínea 6 da primeira série de exercícios a possibilidade de usar uma **chave derivada de uma *password* para proteger e desproteger o ficheiro**. Utilize a norma PKCS#5 para derivar a chave a partir da *password*, tal como descrito na Secção 6.2 do RFC 2898 [1] e disponível na JCA [2]. O programa deve passar a utilizar a primitiva AES.

Qual o papel do *salt* e número de iterações previsto na norma PKCS#5?

7. Realize uma aplicação *web* que copia ficheiros do serviço Google Drive [3] para o serviço Dropbox [4]. A aplicação tem os seguintes requisitos:

- Só utilizadores autenticados podem realizar cópias. Os utilizadores são autenticados através do fornecedor de identidade social Google, usando o protocolo OpenID Connect [5];
- Os utilizadores autenticados podem copiar ficheiros entre os dois serviços, sendo a cópia realizado pela aplicação usando os *endpoints* */get* [6] e */upload* [7] dos respectivos serviços.

Neste exercício não pode usar os SDK da Google/Dropbox para realizar os pedidos aos serviços. Os mesmos têm de ser feitos através de pedidos HTTP construídos pela aplicação *web*.

Considere os seguintes *endpoints* Google:

- Registo de aplicações: <https://console.developers.google.com/apis/credentials>
- *Authorization endpoint*: <https://accounts.google.com/o/oauth2/v2/>
- *Token endpoint*: <https://www.googleapis.com/oauth2/v3/userinfo>

- *UserInfo endpoint*: <https://www.googleapis.com/oauth2/v3/userinfo>

e Dropbox:

- Registo de aplicações: <https://www.dropbox.com/developers/apps/create>
- *Authorization endpoint*: <https://www.dropbox.com/oauth2/authorize>
- *Token endpoint*: <https://api.dropboxapi.com/oauth2/token>
- *Resource endpoint (root)*: <https://content.dropboxapi.com/2>

Referências

- [1] <https://tools.ietf.org/html/rfc2898>
- [2] <https://docs.oracle.com/javase/7/docs/technotes/guides/security/StandardNames.html>
- [3] <https://developers.google.com/drive/api/v2/reference/>
- [4] <https://www.dropbox.com/developers/documentation/http/documentation>
- [5] <https://developers.google.com/identity/protocols/OpenIDConnect>
- [6] <https://developers.google.com/drive/api/v2/reference/files/get>
- [7] <https://www.dropbox.com/developers/documentation/http/documentation#files-upload>