Questões - Trabalho 2

**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?**

RSA🡪 dominante de key exchange

Diffie-Hellman 🡪 este key Exchange é usado para reduzir o risco de ser comprometido

**1.2. Qual o esquema simétrico usado no handshake do TLS e quais os objetivos da sua utilização?**

Protocolo de Handshake do TLS trata da criação de conexão segura e manuseamento, principalmente para assegurar os parâmetros do record protocol.

O esquema simétrico de encriptação (Symmetric encryption scheme), é usado para suportar os esquemas de bloco e de stream. São utilizados para transferir dados com a chave simétrica durante o key-exchange

Handshake Protocol

– Handles the secure connection creation and management, namely the secure establishment of the record protocol cryptographic parameters

**1.3. Qual a característica do record protocol que o torna suscetível a ataques baseados no de Vaudenay?**

Record protocol tem a característica de authenticate-then-encrypt que o torna vulnerável aos ataques baseados no de Vaudenay. O TLS utiliza o padding como maneira de chegar ao bloco correto ou incorreto fazendo downgrading. O servidor responde dizendo se o padding está correcto ou errado. Permitindo vários ataques poodle, podendo fazer um ataque a cada downgrade que acontece.

<https://www.us-cert.gov/ncas/alerts/TA14-290A>

Record Protocol

– Handles data fragmentation, compression, confidentiality and message authentication

– Requires a reliable transport protocol

<https://moxie.org/blog/the-cryptographic-doom-principle/>

<https://crypto.stackexchange.com/questions/202/should-we-mac-then-encrypt-or-encrypt-then-mac>

\*\*Fragment, Compress, Authenticate (MAC) then Encrypt • Two independent connection directions – Separate keys, IVs and sequence number (client write and server write)

The MAC protects the : sequence number, packet type, version and the payload

<https://technet.microsoft.com/pt-pt/library/cc783349(v=ws.10).aspx>

**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?**

Os clientes têm de se registar no servidor de autorização, sendo-lhes atribuído um client\_id. Para alguns é também atribuído um client\_secret, usado pelo cliente no servidor de autorização

Relying party guarda o client\_id e cliente\_secret.

O cliente/ relying party especifica os recursos a que pretende ter acesso indicando um access token. Para obtenção de um access token há quarto Grant flows: cliente credentials, resource owner password, authorization code and implicit.

**2.2. Quais as limitações da utilização deste protocolo para autenticação?**

**A utilização do protocolo para autenticação**

In the traditional client-server authentication model, the client

requests an access-restricted resource (protected resource) on the

server by authenticating with the server using the resource owner's

credentials. In order to provide third-party applications access to

restricted resources, the resource owner shares its credentials with

the third party. This creates several problems and limitations:

o Third-party applications are required to store the resource

owner's credentials for future use, typically a password in

clear-text.

o Servers are required to support password authentication, despite

the security weaknesses inherent in passwords.

o Third-party applications gain overly broad access to the resource

owner's protected resources, leaving resource owners without any

ability to restrict duration or access to a limited subset of

resources.

o Resource owners cannot revoke access to an individual third party

without revoking access to all third parties, and must do so by

changing the third party's password.

o Compromise of any third-party application results in compromise of

the end-user's password and all of the data protected by that

password.

OAuth addresses these issues by introducing an authorization layer

and separating the role of the client from that of the resource

owner. In OAuth, the client requests access to resources controlled

by the resource owner and hosted by the resource server, and is

issued a different set of credentials than those of the resource

owner.

Instead of using the resource owner's credentials to access protected

resources, the client obtains an access token -- a string denoting a

specific scope, lifetime, and other access attributes. Access tokens

are issued to third-party clients by an authorization server with the

approval of the resource owner. The client uses the access token to

access the protected resources hosted by the resource server.

For example, an end-user (resource owner) can grant a printing

service (client) access to her protected photos stored at a photo-

sharing service (resource server), without sharing her username and

password with the printing service. Instead, she authenticates

directly with a server trusted by the photo-sharing service

(authorization server), which issues the printing service delegation-

specific credentials (access token).

This specification is designed for use with HTTP ([[RFC2616](https://tools.ietf.org/html/rfc2616)]). The

use of OAuth over any protocol other than HTTP is out of scope.

The OAuth 1.0 protocol ([[RFC5849](https://tools.ietf.org/html/rfc5849)]), published as an informational

document, was the result of a small ad hoc community effort. This

Standards Track specification builds on the OAuth 1.0 deployment

experience, as well as additional use cases and extensibility

requirements gathered from the wider IETF community. The OAuth 2.0

protocol is not backward compatible with OAuth 1.0. The two versions

may co-exist on the network, and implementations may choose to

support both. However, it is the intention of this specification

that new implementations support OAuth 2.0 as specified in this

document and that OAuth 1.0 is used only to support existing

deployments. The OAuth 2.0 protocol shares very few implementation

details with the OAuth 1.0 protocol. Implementers familiar with

OAuth 1.0 should approach this document without any assumptions as to

its structure and details.

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

**3.1. Para que serve o ID Token?**

OpenID Connect é uma camada de identidade simples criada com base no protocolo de OAuth 2.0. O ID Token é um token de segurança que contem os pedidos de autenticação de um utilizador através de um servidor de autenticação. Verifica a identidade do utilizador e fornece informações de perfil básicas sobre o utilizador.

O ID *token* guarda as informações sobre um utilizador autentificado e serve para transferir estas informações de forma segura do *identity provider* para o *relying party*.

**3.2. Qual destas duas entidades desempenha o papel de relying party: a aplicação cliente ou o resource server?**

O resource server desempenha o papel de relying party, sendo o *relying party* um servidor que estamos a aceder na qual irá direcionar para um servidor autentificador.

A aplicação cliente desempenha o papel de relying party, o OpenID Connect retorna o resultado da autenticaçao realizada pelo servidor ao cliente, de uma maneira segura de forma que o cliente possa confiar.

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* ativo, sem que (u; r) esteja na

relação *user assignment* (UA)?

**Hierarchical RBAC** means that the roles are organised into a hierarchical structure (DAG or Tree), where 'higher' roles inherit all permissions from connected 'lower' roles. Hierarchical RBAC is further subdivided in General- and Restricted Hierarchical RBAC.

SIM???

NAO???

4.2. Qual a relação entre o principio de privilegios minimos e o conceito de sessao na familia de modelos RBAC?