Home Assignment 4
EITN41

A-2 *What problem in OTR is solved using the Socialist Millionaire Problem?*

Answer: Authentication, and mitigating MitM (Man-in-the-Middle) attacks. The authentication is ultimately done by a low-entropy shared secret, which is "baked in" into, and protected, in the protocol. The mitigation is achieved by utilising the Diffie-Hellman problem (given g, p, g^x and g^y it is infeasible to find g^{xy}). In order for both parties to verify each others public key fingerprint, the fingerprint is concatenated with the negotiated Diffie-Hellman secret and the shared low-entropy secret. This concatenated value is then hashed:

$$H(PK_A||PK_B||g^{x_1y_1}||"sharedsecret")$$

Because of the Diffie Hellman problem, the "shared secret" value cannot be brute forced. An aspiring MitM attacker also has to guess the secret in one try, otherwise the communicating parts will see that the protocol has failed.

A-4 *How is perfect forward secrecy solved in OTR?*

Answer: A MAC is calculated on each message, after every message the MAC key used for that message is released and a new one is calculated/agreed upon for the next message. This means that the MAC keys are 'fresh' for only one message and that any later use of it directly gives the attacker away.

A-7 *Why does not an IdP send a response to an authentication request with the HTTP GET method? What alternatives are there?*

Answer: `https://www.w3schools.com/tags/ref_httpmethods.asp`

A-9 *Name and describe four problems in SAML. You should be able to provide a more detailed description for one of them.*

Answer: TODO

A-11 *Describe and compare "discovery" in SAML and OpenID.*

Answer: TODO

A-12 *In a certain sense, there are three different types of communication in SAML and two in OpenID. Describe them and explain the difference.*

Answer: TODO

A-18 *In OAuth, explain why the access token must not be cached, and how this is achieved.*

Answer: If the token is cached,

The Authorisation Server returns an access token with a 200, the data in JSON format. The 200 has header options telling the receiver (Client) to not cache the token (and data).

`Cache-Control: no-store`, this is the *don't cache* command, if you will. `https://tools.ietf.org/html/rfc7234#section-5.2.1.5`

`Pragma: no-cache`, this is mostly used for backwards compatibility, `https://tools.ietf.org/html/rfc7234#section-5.4`. If HTTP 1.0 is used, Cache-Control is not understood, in this case Pragma specifies *don't cache*. If Cache-Control is understood, Pragma is ignored.

A-20 *What is a grant? Name and describe a few different grants.*

Answer: It is a sort of attestation, or authorisation promise. The Resource Owner grants a Client access to certain resources the Owner owns. This can be used by the Client to request an Access Token from the Authorisation Server. The process for this can be different for different types of grants. The four types / classes described in the lecture notes are:

- **Authorisation code -** the Client receives a code, which is used when they request access from Authorisation Server.
- **Implicit grant -** the Client receives a token directly from the Authorisation Server.
- **Resource Owner Password Credentials grant -** Resource Owner gives Client their own login credentials for the Authorisation Server. Using these, the Client receives a token from the server.
- **Client Credentials grant -** Here the Resource Owner is not involved. The Client directly from Authorisation Server after provding their own credentials.