****

|  |
| --- |
| Hugo Jesus - A29243 – ADEETC - LEIC  12-12-2020 |

|  |
| --- |
| ISEL – Instituto Superior de Engenharia de Lisboa |
| Trabalho 2 |
| Segurança Informática 2020/2021 (Sem. Inv.) |

**1. Considere o protocolo TLS e as infraestruturas de chave pública:**

**1.1. Quais os mecanismos que o sub-protocolo handshake tem para evitar ataques de replay (reposição de mensagens de handshakes anteriores)?**

**1.2. De que forma um ataque a uma autoridade de certificação torna possível um ataque de man-in-the-middle?**

1.1) O mecanismo usado pelo sub-protocolo handshake para evitar ataques de replay, utiliza *nounces* (valores aleatórios) diferentes para cada handshake. Este mecanismo permite que a mensagem de fecho de handshake (Finished message) seja sempre diferente para cada handshake, e que seja gerada de forma não determinística.

1.2) O ataque a uma autoridade de certificação torna possível um ataque man-in-the-middle utilizando um certificado obtido de forma ilícita, mas assinado por uma autoridade de certificação reconhecida. Se, de alguma forma um atacante conseguir, obter um certificado ilícito para um site, este poderá fazer-se passar por esse mesmo site, sem que os clientes consigam perceber, uma vez que o certificado está assinado por uma autoridade de certificação reconhecida pela vítima, e dessa forma poderão obter dados privados das vítimas.

**2. No contexto da autenticação baseada em passwords, um ataque de dicionário à interface de autenticação pode ser dificultado usando um salt com mais bits na geração da informação de validação?**

A utilização de mais bits irá dificultar um ataque de dicionário, uma vez que irá aumentar a complexidade computacional das passwords possíveis, e consequentemente o conjunto de possibilidades possíveis. Um conjunto de bits maior de salt aumenta também a necessidade do atacante ter uma tabela mais robusta de hipóteses de valores para efetuar a autenticação, ao mesmo tempo que reduz a hipótese dessas tabelas existirem de forma acessível.

3. **Considere uma aplicação web que guarda no browser cookies contendo o par (u, H(u)), sendo u o identificador de um utilizador e H uma função de hash. Assuma que a construção do cookie é conhecida. A comunicação entre browser e aplicação é feita sobre HTTPS.**

**3.1. Como poderia um atacante fazer-se passar por outro utilizador para o qual sabe o seu identificador (u)?**

**3.2. Que alterações propõe para evitar o ataque anterior?**

3.1) Sabendo a construção do cookie e sabendo o identificador u, o atacante poderia tentar fazer-se passar pelo utilizador testando outputs de funções de hash de u contra o servidor.

3.2) O cookie não deverá ter informação sensível de uma forma relativamente acessível, e neste caso o identificador do utilizador apenas está protegido por uma função de hash. A função de hash H deveria ser feita com base em outras variáveis para aumentar a proteção, como por exemplo utilizar H sobre u e um identificador de sessão única longo e aleatório, e ser esse o resultado = H(u + sessionId). O cookie deverá também ter uma duração curta para que seja computacionalmente impossível replicar o resultado de H no tempo da sessão.

**4. Considere a norma OAuth 2.0 e OpenID Connect no fluxo authorization code grant:**

**4.1. O valor indicado no scope é escolhido pelo cliente ou pelo dono de recursos?**

**4.2. Em que situações o cliente e o servidor de autorização comunicam indiretamente através do browser do dono de recursos?**

**4.3. Qual a diferença entre o access\_token e o id\_token?**

4.1) O valor indicado no scope é escolhido pelo cliente (aplicação usada pelo dono dos recursos) e confirmada a usa utilização pelo dono dos recursos.

4.2) Numa fase inicial, existe uma comunicação indireta entre o Client e o Authorization Server (AS), quando não existe uma autenticação por parte do Resource Owner (RO) junto do AS. O processo é feito partindo do Client que necessita aceder aos Protected Resources (PR) e reencaminha o RO para um endpoint do AS onde este fará a sua autenticação. Após uma autenticação efetuada com sucesso é enviado um redirect URI para o RO com o respectivo Authorization Code (AC), posteriormente passando ao Cliente o URI e o AC para que este possa finalmente solicitar um Access Token (AT) para acesso ao recurso que necessita. Toda a comunicação após o fornecimento do AC ao Client passa a ser efetuada de forma direta entre o Client e o AS.

4.3) O IdToken é um token que o Authorization Server (AS) retorna com informação sobre a autenticação do Resource Owner e que são criados para que outras aplicações possam ter acesso a essas informações, enquanto um AccessToken é criado como uma autorização para acesso a um determinado recurso de forma a que apenas o servidor que aloja esse recurso possa interpretar o seu conteúdo.

**5. Pretende-se configurar e testar um servidor web com HTTPS, com e sem autenticação de cliente, usando o browser e uma aplicação Java. Considere o certificado e chave privada do servidor www.secure-server.edu em anexo, o qual foi emitido pela CA1-int da primeira série.**

**a) Configure e teste o servidor usando o cliente browser, com e sem autenticação de cliente. Tenha por base o ficheiro do servidor em anexo (server.js);**

**b) Realize a aplicação cliente Java;**

**c) Teste o servidor usando o cliente Java, com e sem autenticação de cliente.**

a) Passos executados:

1. Conversão dos certificados “secure-server.pfx” e “secure-server.cer” para as suas versões PEM “secure-server.pem” e “secure-server-cert.pem” via linha de comandos OpenSSL (instalar OpenSLL-Win64). Executar o mesmo para o ficheiro “CA1.cer” e “CA1-int.cer” do ficheiro zip fornecido no primeiro trabalho.

Comando usado: “**openssl x509 -inform der -in certificate.cer -outform pem -out certificate.pem”**

2. Alterar o código para ler os ficheiros PEM convertidos no ficheiro server.js. O objeto “options” deverá manter-se com as opções “requestCert” e “rejectUnauthorized” a “false”.

3. Alterar ficheiro hosts do Windows (Windows\System32\Drivers\etc\hosts) com a introdução de uma linha: 127.0.0.1 [www.secure-server.edu](http://www.secure-server.edu)

4. Importar os certificados “CA1.cer” e “CA1-int.cer” no browser (Chrome > Definições > Segurança > Gerir Certificados) para que possam ser considerados confiáveis (reiniciar o PC após este passo).

5. Por omissão a configuração do server.js é na porta 4433, e após colocar o servidor a correr com o comando “node server.js”, abrir uma janela do browser com o site <https://www.secure-server.edu:4433>, se tudo estiver corretamente configurado deverá abrir a página sem indicação de certificado inválido.

6. Neste momento o servidor funciona com certificados válidos, que nós como cliente reconhecemos, próximo passo identificarmo-nos como clientes válidos quando o servidor solicitar.

7. Retirar de comentário a linha que está nesse estado no código de criação do servidor. Mudar o objeto “options” e colocar “true” nas opções anteriormente assinaladas a false.

8. Importar os ficheiros “pfx” de cliente (ex. “Alice\_1.pfx”) no browser. O processo é similar ao de importação dos ficheiros de certificados, mas desta vez com ficheiros “pfx”. A password destes ficheiros é “changeit”. Estes certificados deverão estar nos “pessoais” após importação.

9. Colocar o servidor a correr, e ao chamar o site no browser novamente, o browser deverá apresentar uma janela para escolher o certificado pessoal a enviar.

10. Se tudo estiver corretamente configurado, temos uma ligação com ambos os intervenientes a autenticarem-se. A página só é aberta se o certificado for válido.

**6. Realize uma aplicação web para adicionar eventos ao calendário Google do utilizador, a partir de datas de milestones presentes num dos repositórios GitHub a que o utilizador tenha acesso.**

**Requisitos funcionais:**

**• Após autenticação do utilizador na aplicação, são apresentados os milestones do GitHub para um determinado projecto no qual o utilizador tem acesso;**

**• Os utilizadores autenticados podem adicionar eventos ao seu calendário Google a partir das datas presentes em milestones GitHub.**

**Requisitos não funcionais:**

**• A aplicação só pode ser usada por utilizadores autenticados, exceto a rota de autenticação. Os utilizadores são autenticados através do fornecedor de identidade social Google, usando o protocolo OpenID Connect;**

**• O estado de autenticação entre o browser e a aplicação web é mantido através de cookies;**

**• A aplicação tem de estar preparada para ser usada em simultâneo por utilizadores diferentes;**

**• Os dados guardados para cada utilizador podem estar apenas em memória;**

**• É valorizado o acesso a repositórios privados do Github.**