Universidade da Beira Interior
Faculdade de Engenharia
Departamento de Informática

Segurança Informática

Guia para Aula Laboratorial 10

Licenciatura em Engenharia Informática
Licenciatura em Informática Web
Licenciatura em Tecnologias e Sistemas da Informação

Computer Security

Guide for Laboratory Class 10

Degree in Computer Science and Engineering
Degree in Web Informatics
Degree in Information Technologies and Systems

Sumário

Instalação de um servidor HyperText Transfer Protocol (HTTP) com tecnologia de pré-processamento de hipertexto (Hypertext Preprocessor (PHP)). Demonstração de como ativar o protocolo *Secure Sockets Layer* (SSL) na aplicação a desenvolver, para segurança das comunicações. Criação de um formulário simples para registo numa aplicação Web e estudo da forma segura de armazenamento de palavras-passe em bases de dados.

Summary

Instalation of an HyperText Transfer Protocol (HTTP) server with the Hypertext PreProcessing (PHP) technology. Demonstration of how to activate and configure the Secure Sockets Layer *(SSL) protocol to secure the communications of the developed web application. Implementation of a simple form for registration in the web application and study how the passwords should be securely stored in databases.*

Pré-requisitos:

Algumas tarefas enunciadas em baixo devem ser executadas num sistema com um servidor de páginas *Hypertext Markup Language* (HTML), robustecido com a tecnologia PHP e com uma *shell* SQLite3. Pressupõe-se o acesso a um sistema com estas condições preenchidas ou com a possibilidade de instalar o *software* necessário.

1 Instalação do Servidor HTTP, do PHP e do MySQL

Instalation of the HTTP Server, PHP and MySQL

Para criar uma aplicação *web* baseada em páginas HTTP que permita, entre outras funcionalidades, servir de intermediário entre o utilizador final e uma base de dados, é necessário primeiro instalar um servidor de ficheiros, e habilitá-lo com a tecnologia de suporte a páginas dinâmicas. Um servidor de páginas HTTP bastante popular actualmente e com código fonte aberto é o Apache (`http://www.apache.org/`). Uma das tecnologias de suporte a páginas dinâmicas com mais provas dadas é conhecida por *Hypertext Preprocessor* (PHP) (`http://www.php.net/`), e um dos sistemas de gestão de bases de dados relacional que melhor se integra com os sistemas antes apontados é o MySQL (`http://www.mysql.com/`). Existem implementações e pacotes de instalação para os vários sistemas operativos de utilização mais comum. É inclusivé possível encontrar combinações pré-configuradas de todas as tecnologias referidas em pacotes únicos, cuja designação acaba tipicamente com as letras AMPP.

Q1.: O que significam as várias letras do acrónimo antes referido?

- ☐ Apache HTTP Server
- ☐ Internet Information Services
- ☐ PHP
- ☐ Perl
- ☐ MySQL
- ☐ SQLite
- ☐ Postgresql
- ☐ Python

Uma dessas combinações / pacotes é designada por XAMPP (`http://www.apachefriends.org/en/xampp.html`), em que 'X' abrevia a palavra *cross*, o que significa que é um pacote disponível para as várias plataformas (sistemas operativos). Seguindo a mesma ordem de ideias, deduz-se que o pacote LAMP é especialmente vocacionados para sistemas Linux, o WAMP para sistemas Windows, e o MAMP para MAC.

Tarefa 1 *Task 1*

Caso não tenha alguma das tecnologias mencionadas antes disponíveis na sua máquina, a sua pri-

meira tarefa consiste na sua instalação.

Sistema Operativo Microsoft Windows

No sistema operativo Windows, a instalação segue os trâmites normais associados à instalação de uma aplicação: (i) *download* do pacote de instalação; (ii) execução do mesmo; e (iii), ajuste de alguns parâmetros via caixas de diálogo do assistente de instalação. Os vários serviços têm de ser ativados após instalação para que o o sistema funcione. A aplicação disponibiliza normalmente um painel de controlo que permite colocar os serviços a correr facilmente, embora o utilizador possa não ficar com a ideia exata das configurações que disponibilizam o seu sistema em rede, e que devem ser sujeitas a auditoria no desenvolvimento de aplicações seguras e robustas.

Sistema Operativo Fedora

No sistema operativo Fedora, a instalação do servidor Apache, do MySQL, do PHP e dos pacotes de interligação destas tecnologias passa pela emissão de uma sequência de comandos parecida com a que é mostrada a seguir:

```
$ sudo dnf -y install httpd mariadb-server
php php-common php-mysql php-pdo php-gd
php-mbstring
```

(palavra-passe do administrador de sistemas)

Nota: depois de instalar os vários pacotes, é necessário colocar os serviços a correr e verificar que a *firewall* permite ligações *Transmission Control Protocol* (TCP) na porta 80 (HTTP) e na porta 3306 (MySQL) – esta última pode não ser necessária para ligações locais. Os dois comandos seguintes devem ser suficientes para conseguir o efeito de permitir (de forma efémera) tráfego HTTP e HTTPS para a máquina que está a utilizar na *firewall* do Fedora:

```
$ sudo firewall-cmd --permanent
--add-service=http
```

```
$ sudo firewall-cmd --permanent
--add-service=https
```

Para colocar os serviços (`httpd` e `mysqld`) a correr, emita os seguintes comandos no terminal:

```
$ sudo systemctl start httpd
```

```
$ sudo systemctl start mariadb
```

No final, pode verificar o estado dos serviços com instruções semelhantes a:

```
$ sudo systemctl status httpd
```

```
$ sudo systemctl status mariadb
```

No final, pode configurar a instalação do MySQL (ou, neste caso, do sistema compatível conhecido como mariadb) invocando o *script* seguinte:

```
$ mysql_secure_installation
```

Não se esqueça de ajustar uma palavra-passe para o *root* de que se recorde adiante e teste o sistema tentando aceder via *shell* com:

```
$ mysql
```

Note que, em Fedora, as páginas servidas pelo Apache estão normalmente em `/var/www/html/`. Evolua para essa pasta com `$ cd /var/www/html/`.

Sistema Operativo Ubuntu

Em Ubuntu, o mesmo pode ser conseguido usando comandos semelhantes aos seguintes (todos os comandos requerem direitos de administração):

```
$ sudo apt-get install apache2
```

```
$ sudo apt-get install mysql-client
mysql-server php5 php5-mysql
```

Não deve esquecer que os serviços têm de ser colocados em execução. Tal tarefa pode ser levada a cabo com os dois comandos seguintes:

```
$ sudo /etc/init.d/apache2 restart
```

```
$ sudo /etc/init.d/mysqld restart
```

Nota importante: depois de colocar o servidor HTTP em funcionamento, deve ser possível ver uma página de teste no endereço `http://localhost/` usando o *browser*. Deve consultar a documentação do servidor HTTP para saber em que diretoria deve colocar os ficheiros a serem servidos por ele.

2 Ativar o HTTPS

Enabling HTTPS

O robustecimento do servidor de páginas HTTP Apache com o *Transport Layer Security* (TLS) requer que se produza um certificado digital e um par de chaves pública e privada (normalmente RSA). O certificado digital interliga o nome do servidor com a sua chave pública. Quando um *browser* faz um pedido ao servidor, este envia-lhe o certificado com a sua chave pública. O *browser* gera uma chave de sessão simétrica, envia-a ao servidor cifrada com a chave pública, e passa a usá-la para cifrar todas

as comunicações com endereços desse servidor começados por HTTP**S**.

Para sites oficiais, o certificado X.509 de chave pública é normalmente gerado por uma entidade de confiança (uma empresa) para o site em questão. Este serviço é pago ou incluido no pacote de alojamento do site. Caso o certificado seja assinado por uma entidade de confiança, cujos certificados provenham de uma raiz de confiança já instalada no *browser*, este não dá qualquer erro quando acede ao site, e até informa que é de confiança. Caso o certificado seja gerado localmente, como de resto exemplificado neste guia, o *browser* dá um erro associado à confiança da segurança fornecida, já que não sabe se o certificado que recebe é feito por alguém de confiança, ou por alguém que, tal como os alunos destas aulas, sabem fazer o seu próprio certificado.

Existe atualmente uma inicativa que permite a criação de certificados digitais de forma gratuita. **Q2.: Qual o nome dessa iniciativa?**
☐ *Let's go to school.*
☐ *Let's go to jail.*
☐ *Let's be secure.*
☐ *Let's encrypt.*

Q3.: Quais são os maiores patrocinadores dessa iniciativa (selecione todos os que se aplicam)?
☐ Mozilla ☐ Google ☐ Cisco
☐ Facebook ☐ Internet Society ☐ Akamai
☐ HP ☐ Microsoft

Sistema Operativo Microsoft Windows e XAMPP

Caso esteja a usar o XAMPP, a ativação do SSL é relativamente simples. Este pacote trás já uma ferramenta de conveniência, que automatiza o processo de geração de chaves e do certificado, bem como a configuração do servidor HTTP. De qualquer forma, **pensa-se que é do seu interesse estudar a instalação passo-a-passo incluída para o sistema operativo Linux, contida na secção seguinte.** Para configurar o SSL no XAMPP, siga os passos seguintes:

1. Abra um terminal (*Start* → *Run*, escreva `cmd` e carregue em OK);
2. Navegue até à diretoria do Apache dentro da diretoria do XAMPP (e.g., `cd C:\XAMMP\APACHE`);
3. Execute o comando `makecert`.
4. Siga as instruções e preencha os campos que achar necessários corretamente.
5. No final, convém reiniciar o servidor HTTP.

Sistema Operativo (Arch) Linux

Nota: o guia incluido a seguir foi testado no sistema operativo *Arch Linux*, embora possa ser generalizado para outros sistemas operativos Linux, como o Fedora. Os comandos para Ubuntu estão numa secção subsequente.

A instalação dos vários serviços e tecnologias no *Arch Linux* pode ser feita imitindo o seguinte comando no terminal:

```
$ pacman -S apache php php-apache mariadb
```

O ficheiro de configuração do Apache (`httpd.conf`) no *Arch Linux* está na diretoria `/etc/httpd/conf/`. Em princípio, é necessário editar este ficheiro (precisa de direitos de administração), descomentar a linha

```
LoadModule mpm_prefork_module
modules/mod_mpm_prefork.so
```

e comentar a linha

```
LoadModule mpm_event_module
modules/mod_mpm_event.so
```

antes de colocar o servidor a correr com:

```
$ systemctl start httpd
```

Repare que o serviço que colocamos a correr tem um *d* no final, indicativo da palavra *daemon*. *Daemon* é usada para designar um processo que corre em segundo plano. Depois deste comando, deve ser possível aceder à página `http://localhost` num *browser*.

Por defeito, as páginas disponibilizadas pelo Apache estão na pasta `/srv/http` e, a menos de uma ou outra nuance, nas pastas `/public_html` de cada um dos utilizadores do sistema. A ativação do SSL pode ser conseguida pela execução ordenada dos 4 passos seguintes numa consola:

1. `$ cd /etc/httpd/conf`
   Este comando coloca o *path* atual na diretoria de configuração do Apache, onde serão colocados o certificado e a chave.

2. `$ openssl req -new -x509 -extensions v3_ca -keyout server.key -nodes -out server.crt -days 1825`
   Recorre-se ao OpenSSL para gerar um par de chaves RSA (opção `-keyout`) para o ficheiro `server.key` não cifrado (`-nodes`) **e** um certificado auto-assinado (opção `-X509`) para o ficheiro `server.crt`, com a validade de 1825 dias. Não se esqueça de colocar o nome do URL no *Common Name*. Neste caso será `localhost`.

3. Deverá ser necessário mudar as permissões dos ficheiros com o certificado e com as chaves. Os comandos serão semelhantes a: `$ chmod 400 server.key` e `$ chmod 444 server.crt`

4. Abre-se o ficheiro `/etc/httpd/conf/httpd.conf` para edição e descomentam-se as três linhas seguintes:

```
LoadModule ssl_module
modules/mod_ssl.so
LoadModule socache_shmcb_module
modules/mod_socache_shmcb.so
Include conf/extra/httpd-ssl.conf
```

5. Finalmente, reinicia-se o servidor HTTP com `$ systemctl restart httpd`.

No final destes passos, deve ser possível aceder à página `https://localhost` (repare no **s** incluido no endereço).

Sistema Operativo Ubuntu

O sistema operativo Ubuntu oferece uma série de scripts para inicialização do SSL no apache. Como tal, deve bastar inserir ordenadamente os seguintes comandos no terminal (requerem privilégios de administração):

```
$ sudo make-ssl-cert
generate-default-snakeoil
```

`$ sudo a2enmod ssl`

`$ sudo a2ensite default-ssl`

3 Rascunho da Aplicação

Draft of the Web Application

O objetivo desta parte do guia é estabelecer a ponte com a parte em que se implementa alguma lógica aplicacional para conseguir registar nomes de utilizador e palavra-passe numa base de dados a partir dos dados introduzidos num formulário.

Tarefa 2 *Task 2*

Na raíz dos ficheiros que são servidos pelo apache, crie um um ficheiro `index.php` com o código:

```
<?php
 phpinfo();
 ?php>
```

Verifique que está a funcionar acedendo a `http://localhost`.

Tarefa 3 *Task 3*

Crie uma página (ficheiro, prtanto) chamada `register.php`. Coloque apenas o código HTML necessário para obter os seguintes elementos e funcionalidade:

- O título da página deve ser `Registration form. Enter username and password.`
- A página deve ter um formulário **centrado**, que evolui para a página `insert.php` ao se pressionar o botão de submeter. Deve usar o método POST para transmissão dos valores no formulário;
- O formulário deve ter duas caixas de texto (uma chamada `username` e outra chamada `password`) e dois botões. Um dos botões deve ter o texto `submit` e o outro deve ter o texto `reset`;
- As caixas de texto devem ter valores predefinidos: `Insert User` para a primeira e `********` para a segunda. A caixa para palavras-passe não deve mostrar o que está a ser introduzido;
- O botão de `reset` deve reiniciar as duas caixas de texto referidas antes;
- O botão de `submit` deve fazer a página evoluir para `register.php` (i.e., a sua definição deve ser semelhante a `<input type="submit">`).

Depois de a criar, teste a página.

Tarefa 4 *Task 4*

Depois de se certificar que está a funcionar corretamente, copie o ficheiro `register.php` para

`insert.php`. **Q4.: Consegue reconhecer o melhor comando para o conseguir?**

☐ `$ mv register.php insert.php`
☐ `$ cp register.php insert.php`
☐ `$ ln register.php insert.php`

Altere este ficheiro de modo a que a nova página assim criada mostre os valores introduzidos pelo utilizador no formulário e certifique-se de que funciona. Depure as vezes que forem necessárias antes de avançar para a próxima secção.

4 Armazenamento Seguro de Palavras-Passe numa Base de Dados

Secure Storage of Passwords in a Database

As aplicações construidas sobre bases de dados de hoje em dia contêm normalmente um mecanismo de controlo de acesso aos dados baseado numa combinação *nome de utilizador* / *palavra-passe*. Para que a aplicação possa validar o *login*, essa combinação ou uma representação equivalente tem de estar guardada algures no sistema implementado, potencialmente na própria base de dados a que a aplicação acede. Caso esta informação não seja manuseada com a devida precaução, pode ser a origem de alguns problemas de segurança. Por exemplo, se a palavra-passe for guardada na própria base de dados em texto limpo, qualquer utilizador com acesso direto ao sistema e permissões de leitura na tabela respetiva, ou no ficheiro que a guarda, pode ver todas as palavras-passe guardadas no sistema.

Tarefa 5 *Task 5*

Aceda ao sistema de gestão de bases de dados MySQL (`$ mysql -u root -p`) e crie a base de dados `WebApp`. **Q5.: Qual o comando que pode usar para conseguir este efeito?**

☐ `mysql > SELECT WebApp;`
☐ `mysql > CREATE WebApp;`
☐ `mysql > CREATE DATABASE WebApp;`
☐ `mysql > MAKE DB WebApp;`

No final, não se esqueça de evoluir para o contexto da base de dados criada antes com um comando semelhante a:

```
mysql > USE WebApp;
```

Tarefa 6 *Task 6*

Crie a tabela `User` com os seguintes campos:

- `username` (chave primária do tipo `VARCHAR(30)`);
- `salt` (do tipo `CHAR(32)`) e;
- `rep` (do tipo `CHAR(64)`).

Q6.: A instrução SQL seguinte produz o resultado pedido antes?

```
CREATE TABLE User(username VARCHAR(30),salt
    CHAR(32),rep CHAR(64));
```

☐ Sim, produz! ☐ Não, caramba!

Q7.: Faz sentido usar o tipo `VARCHAR` para o primeiro atributo e `CHAR` para os outros dois?

☐ Não faz sentido nenhum e o docente fez de propósito para que não fizesse para eu ter de pensar nisso.
☐ Faz todo o sentido e, por acaso, o docente até pensou bem no que estava a fazer.

Q8.: Para que serve o `salt` no contexto desta tabela e da autenticação de utilizadores?

☐ O valor do `salt` é um valor aleatório e potencialmente único para cada utilizador do sistema, que é concatenado com a palavra-passe, e que aumenta fortemente a resistência a ataques de dicionário ou de pré-processamento de valores de *hash*.
☐ O `salt` serve para temperar a salada, que está insossa.

Já agora, insira (`INSERT`) uma linha fictícia na tabela e verifique, no final que, de facto, foi inserida a informação.

Tarefa 7 *Task 7*

Na página `insert.php`, insira o código que achar necessário para produzir um número aleatório com 32 caracteres hexadecimais e guarde o resultado na variável `$salt`[1]. Escreva o resultado no HTML da página.

Tarefa 8 *Task 8*

[1] Se calhar, pode usar uma combinação de `rand()` com `md5()`, mas o ideal seria ler diretamente de `/dev/random`...

Como se pode ver pela definição da tabela, o primeiro campo leva um nome de utilizador, o segundo deve levar um valor aleatório (ou pseudo-aleatório) que nem precisa ser conhecido para o utilizador, e o terceiro deve levar o valor de *hash* (e.g., o *output* do algoritmo SHA256) do valor do *salt* **concatenado** com a palavra-passe. **Esta tarefa** consiste na implementação do código necessário à produção da representação (mais ou menos) segura da palavra-passe em PHP. A ideia, para já, é mostrar essa representação segura na própria página `insert.php`. Considere usar a seguinte função PHP:

```
string hash (string $algo, string $data)
```

Tarefa 9 *Task 9*

Desenrasque as instruções PHP e SQL necessárias à inserção do `$username`, do `$salt` e da `$rep` (esta última criada na tarefa anterior) na base de dados `WebApp`, nomeadamente na tabela `User`. Provavelmente vai precisar das seguintes funções PHP para fazer a ligação com a base de dados MySQL:

- A função seguinte tenta estabelecer uma ligação à base de dados, guardando o descritor em `$c` no caso de ser bem sucedida. No contexto desta aula, `host` deve ser `localhost`, `user` deve ser `root` (embora esta configuração **não** seja considerada boa prática), enquanto que a `pass` deve conter a palavra passe do utilizador especificado para o MySQL.

```
$c = mysql_connect("host","user","pass");
```

- Função para selecionar a base de dados correta gerida pelo MySQL. No contexto desta aula, `"my_db"` deve ser `"WebApp"`.

```
mysql_select_db("my_db", $c);
```

- Função que submete a instrução SQL constante em `$query` na base de dados antes selecionada.

```
$result=mysql_query("$query");
```

- Função que termina a ligação à base de dados.

```
mysql_close($c);
```

No final, teste a aplicação e certifique-se de que os dados estão efetivamente a ser introduzidos na base de dados (i.e., aceda diretamente ao `mariadb` e verifique, através de um `SELECT`, que os dados estão na tabela `User` da base de dados `WebApp`).

Tarefa 10 *Task 10*

Finalmente, implemente uma página com um formulário de autenticação (*login*) compatível com o esquema definido para guardar as representações das palavra-passe. Para fazer introdução de dados e validação irá precisar de dois ficheiros `php` (embora pudesse fazer tudo num só). Batize-os de `login.php` e `validate.php`. A página `validate.php` deve enviar o utilizador de novo para a página de *login* caso as credenciais fornecidas estejam erradas. Caso contrário, deve mostrar a mensagem: `Tótil, bacano(a)!`.

Teste exaustivamente a sua aplicação.