Chapter 21

How to create secure web sites

Objectives

Applied

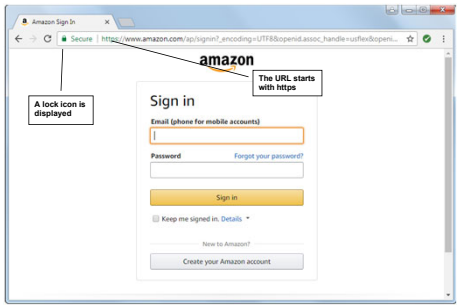
1. Use a secure connection and the Secure Sockets Layer (SSL) protocol for your web pages whenever that's needed.
2. Use form-based authentication for your web pages whenever that's needed.
3. Use PHP to encrypt and decrypt data whenever that's needed.

Objectives (continued)

Knowledge

1. Describe the use of the SSL protocol for getting a secure connection and providing for authentication, including the use of a digital secure certificate, SSL strength, and the $_SERVER array.
2. Distinguish between form-based authentication and basic authentication.
3. Describe the use of PHP for encrypting and validating passwords that are stored in a database.
4. List the four cryptography libraries presented in this chapter.
5. Describe the use of the Defuse cryptography library for encrypting the data that's stored in a database and for decrypting the data after it's retrieved from the database.

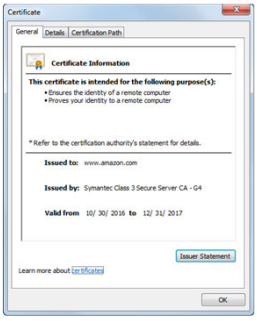
A request made with a secure connection



Key terms

- Transport Layer Security (TLS)
- Secure Sockets Layer (SSL)
- secure connection
- encryption

A digital secure certificate



Types of digital secure certificates

- Server certificate
- Client certicate

How authentication works

- *Authentication* is the process of determining whether a server or client is who and what it claims to be.
- When a browser makes an initial attempt to communicate with a server over a secure connection, the server authenticates itself by providing a *digital secure certificate*.
- If the digital secure certificate is registered with the browser, the browser won't display the certificate by default. However, the user still has the option to view the certificate.
- In some rare cases, the server may request that a client authenticate itself by presenting its own digital secure certificate.

Authorities that issue digital secure certificates

```
www.symantec.com/ssl-sem-page
www.godaddy.com/ssl
www.globalsign.com
www.startcom.org
www.comodo.com
```

SSL strengths

```
40-bit
56-bit
128-bit
256-bit
```


Key terms

- certification authority (CA)
- registration authority (RA)
- SSL strength

URLs for secure connections on a local system

Test if secure connections are configured correctly

```
https://localhost/
```

Request a secure connection

```
https://localhost/book_apps/ch21_ssl/
```

Return to a regular connection

```
http://localhost/book_apps/ch21_ssl/
```


URLs for secure connections over the Internet

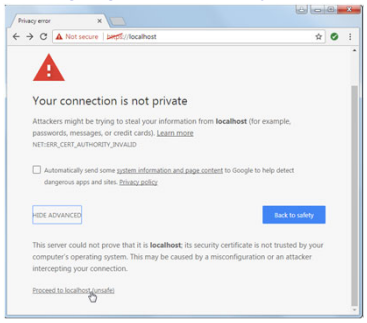
Request a secure connection

```
https://www.murach.com/
```

Return to a regular connection

```
http://www.murach.com/
```

A warning page for the security certificate



The $_SERVER array

Index	Description
`HTTPS`	Returns a non-empty value if the current request is using HTTPS.
`HTTP_HOST`	Returns the host for the current request.
`REQUEST_URI`	Returns the URI (Uniform Resource Identifier) for the current request.

A utility file that redirects to a secure connection

```
<?php
    // make sure the page uses a secure connection
    $https = filter_input(INPUT_SERVER, 'HTTPS');
    if (!$https) {
        $host = filter_input(INPUT_SERVER, 'HTTP_HOST');
        $uri = filter_input(INPUT_SERVER, 'REQUEST_URI');
        $url = 'https://' . $host . $uri;
        header("Location: " . $url);
        exit();
    }
?>
```


Form-based authentication

- Allows the developer to code a login form that gets the username and password.
- Allows the developer to only request the username and password once per session.
- By default, it doesn't encrypt the username and password before sending them to the server.

Basic authentication

- Causes the browser to display a dialog box that gets the username and password.
- Requires the browser to send the username and password for every protected page.
- By default, it doesn't encrypt the username and password before sending them to the server.

Digest authentication

- Causes the browser to display a dialog box that gets the user name and password.
- Encrypts the username and password before sending them to the server.

Two functions for working with passwords

Function	Description
`password_hash($password, $algorithm)`	Creates a new hash of the password using a strong salt and a strong one-way encryption algorithm.
`password_verify($password, $hash)`	Returns TRUE if the specified password matches the specified hash.

Two constants for setting the algorithm

Constant	Description
`PASSWORD_BCRYPT`	Uses the bcrypt algorithm to create a hash that's 60 characters long.
`PASSWORD_DEFAULT`	Uses the default algorithm of the password_hash() function. With PHP 5.5 and 7.1, the default algorithm is bcrypt.

Code that hashes a password using the default algorithm

```
$password = 's3sam3';
$hash = password_hash($password, PASSWORD_DEFAULT);
```

Code that verifies whether a password is valid

```
$valid_password = password_verify('s3sam3',
    '$2y$10$xIqN2cVy8HVuKNKUwxFQR.xRP9oRj.FF8r52spVc.XCaEFy7iLHmu');
if ($valid_password) {
    echo "Password is valid.<br>";
}
```


The admin_db.php file (continued)

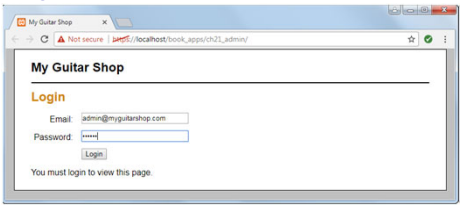
```
function is_valid_admin_login($email, $password) {
    global $db;
    $query = 'SELECT password FROM administrators
              WHERE emailAddress = :email';
    $statement = $db->prepare($query);
    $statement->bindValue(':email', $email);
    $statement->execute();
    $row = $statement->fetch();
    $statement->closeCursor();
    $hash = $row['password'];
    return password_verify($password, $hash);
}
?>
```


A script that creates a table for storing usernames and passwords

```
CREATE TABLE administrators (
  adminID           INT           NOT NULL   AUTO_INCREMENT,
  emailAddress      VARCHAR(255)  NOT NULL,
  password          VARCHAR(255)  NOT NULL,
  firstName         VARCHAR(60),
  lastName          VARCHAR(60),
  PRIMARY KEY (adminID)
);

INSERT INTO administrators (adminID, emailAddress, password) VALUES
(1, 'admin@myguitarshop.com',
    '$2y$10$lHqybsUxtrV/y6j6WfG3.utNzpVTkNCm/neRFPnaaQiBWOJVIIEiq'),
(2, 'joel@murach.com',
    '$2y$10$.imVkbsvI2XTC13bMONdUOllyhddj/IhYZBGU87nqZ1j8ebXPezre'),
(3, 'mike@murach.com',
    '$2y$10$21KIM2059gSrnAQWV.5Ciufzo9sNqONmmzIhE8qvd/IDaeQvHGlEq');
```

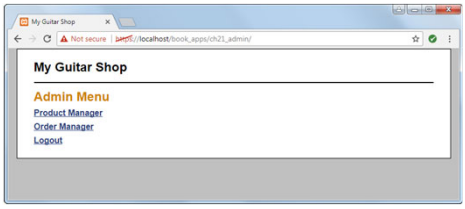

A login form



The admin_db.php file

```
<?php
function add_admin($email, $password) {
    global $db;
    $hash = password_hash($password, PASSWORD_DEFAULT);
    $query =
        'INSERT INTO administrators (emailAddress, password)
         VALUES (:email, :password)';
    $statement = $db->prepare($query);
    $statement->bindValue(':email', $email);
    $statement->bindValue(':password', $hash);
    $statement->execute();
    $statement->closeCursor();
}
```


A protected page



The controller for the protected pages

```
<?php
// Start session management and include necessary functions
session_start();
require_once('model/database.php');
require_once('model/admin_db.php');

// Get the action to perform
$action = filter_input(INPUT_POST, 'action');
if ($action == NULL) {
    $action = filter_input(INPUT_GET, 'action');
    if ($action == NULL) {
        $action = 'show_admin_menu';
    }
}

// If the user isn't logged in, force the user to login
if (!isset($_SESSION['is_valid_admin'])) {
    $action = 'login';
}
```


The controller for the protected pages (continued)

```
// Perform the specified action
switch($action) {
    case 'login':
        $email = filter_input(INPUT_POST, 'email');
        $password = filter_input(INPUT_POST, 'password');
        if (is_valid_admin_login($email, $password)) {
            $_SESSION['is_valid_admin'] = true;
            include('view/admin_menu.php');
        } else {
            $login_message = 'You must login to view this
              page.';
            include('view/login.php');
        }
        break;
```


The controller for the protected pages (continued)

```
    case 'show_admin_menu':
        include('view/admin_menu.php');
        break;
    case 'show_product_manager':
        include('view/product_manager.php');
        break;
    case 'show_order_manager':
        include('view/order_manager.php');
        break;
    case 'logout':
        $_SESSION = array();   // Clear all session data
        session_destroy();     // Clean up the session ID
        $login_message = 'You have been logged out.';
        include('view/login.php');
        break;
}
?>
```


A utility file that forces a valid admin user

```
<?php
    // make sure user is a valid administrator
    if (!isset($_SESSION['is_valid_admin'])) {
        header("Location: ." );
    }
?>
```

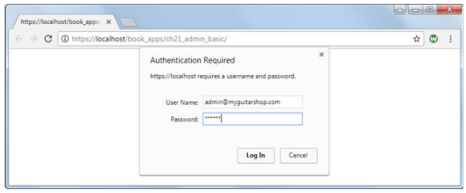
Code at the top of the login page

```
<?php
    // require a secure connection
    require_once('util/secure_conn.php');
?>
```

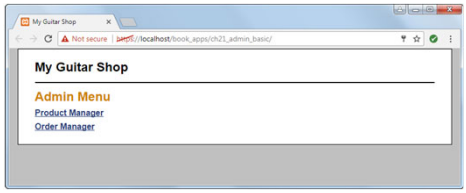
Code the top of the other protected pages

```
<?php
    // require a secure connection
    require_once('util/secure_conn.php');
    // require a valid admin user
    require_once('util/valid_admin.php');
?>
```

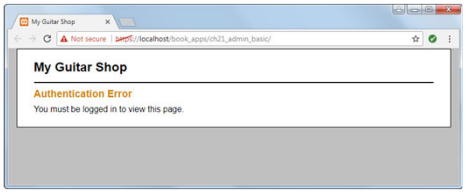

A login dialog box for basic authentication



A protected page



The unauthorized page



The $_SERVER array for basic authentication

Index	Description
PHP_AUTH_USER	The username from the authentication dialog box or a NULL value if the dialog box hasn't been displayed.
PHP_AUTH_PW	The password from the authentication dialog box or a NULL value if the dialog box hasn't been displayed.

Code that forces a valid admin user

```
<?php
require_once('model/database.php');
require_once('model/admin_db.php');

$email = '';
$password = '';
if (isset($_SERVER['PHP_AUTH_USER']) &&
isset($_SERVER['PHP_AUTH_PW'])) {
    $email = $_SERVER['PHP_AUTH_USER'];
    $password = $_SERVER['PHP_AUTH_PW'];
}

if (!is_valid_admin_login($email, $password)) {
    header('WWW-Authenticate: Basic realm="Admin"');
    header('HTTP/1.0 401 Unauthorized');
    include('unauthorized.php');
    exit();
}
?>
```


Code at the top of each protected page

```
<?php
    // require a secure connection
    require_once('util/secure_conn.php');

    // require a valid admin user
    require_once('util/valid_admin.php');
?>
```


Four cryptography libraries

- mcrypt
- Libsodium
- Defuse
- OpenSSL

The URL for the Defuse Crypto library

https://github.com/defuse/php-encryption

One way to install the Defuse cryptography library

1. Go to the URL shown above.
2. Follow the instructions there to download the defuse-crypto.phar file that contains the library. If you're serious about security, you should also follow the instructions there to verify the integrity of the defuse-crypto.phar file.
3. Copy the defuse-crypto.phar file to a logical place on your file system, such as the xampp/php/lib directory.

Some methods of the Key class

```
createNewRandomKey()
saveToAsciiSafeString()
loadFromAsciiSafeString($keyAscii)
```

Some methods of the Crypto class

```
encrypt($data, $key)
decrypt($data, $key)
```


Code that creates an encryption key and saves it to a file

```
require_once('/xampp/php/lib/defuse-crypto.phar');

use Defuse\Crypto\Key;

$key = Key::createNewRandomKey();
$keyAscii = $key->saveToAsciiSafeString();
file_put_contents('/xampp/php/defuse-key.txt', $keyAscii);
```


Code that encrypts and decrypts data

```
require_once('/xampp/php/lib/defuse-crypto.phar');

use Defuse\Crypto\Key;
use Defuse\Crypto\Crypto;
use Defuse\Crypto\Exception\WrongKeyOrModifiedCiphertextException;

// set up credit card variable
$credit_card_no = '4111111111111111';

// get encryption key
$keyAscii = file_get_contents('/xampp/php/defuse-key.txt');
$key = Key::loadFromAsciiSafeString($keyAscii);

// encrypt data
$encrypted_data = Crypto::encrypt($credit_card_no, $key);
echo 'Encrypted data: ' . $encrypted_data . '<br>';

// decrypt data
try {
    $decrypted_data = Crypto::decrypt($encrypted_data, $key);
    echo 'Decrypted data: ' . $decrypted_data . '<br>';
} catch (WrongKeyOrModifiedCiphertextException $ex) {
    echo 'Exception: ' . $ex->getMessage() . '<br>';
}
```


The Crypt class (crypt.php)

```
<?php
require_once('/xampp/php/lib/defuse-crypto.phar');

use Defuse\Crypto\Key;
use Defuse\Crypto\Crypto;
use Defuse\Crypto\Exception\WrongKeyOrModifiedCiphertextException;

class Crypt {
    private $key;

    public function __construct() {
        // make sure the following code points to a file that exists
        // and contains a valid key
        $keyAscii = file_get_contents('/xampp/php/defuse-key.txt');
        $this->key = Key::loadFromAsciiSafeString($keyAscii);
    }

    public function encrypt($data) {
        $encrypted_data = Crypto::encrypt($data, $this->key);
        return $encrypted_data;
    }
```


The Crypt class (crypt.php) (continued)

```
    public function decrypt($encrypted_data) {
        try {
            $data = Crypto::decrypt($encrypted_data, $this->key);
            return $data;
        } catch (WrongKeyOrModifiedCiphertextException $ex) {
            throw new Exception($ex->getMessage());
        }
    }
}
?>
```


Code that uses the Crypt class to encrypt and decrypt data

```
<?php
require 'crypt.php';

$credit_card_no = '4111111111111111';

// Create the Crypt object
$crypt = new Crypt();

// Use the Crypt object to encrypt the data
$encrypted_data = $crypt->encrypt($credit_card_no);
echo 'Encrypted data: ' . $encrypted_data . '<br>';

// Use the Crypt object to decrypt the data
try {
    $decrypted_data = $crypt->decrypt($encrypted_data);
    echo 'Decrypted data: ' . $decrypted_data . '<br>';
} catch (Exception $ex) {
    echo 'Exception: ' . $ex->getMessage();
}
?>
```