

## CSCI 241

---

### Security

*Department of CS & QM*  
*Winthrop University*  
 Fall 2011

Prepared by X. Wang

## Topics

---

- HTTP Authentication
- Authentication with PHP
- Form-Based PHP Authentication
- Data Encryption
- Authentication and Session-Based Applications
- Data Protection on the Web

Prepared by X. Wang

2

## Authentication

---

- Many web database applications require restrictions to control user access
- These applications need to authenticate and authorize user requests, typically by **collecting a username and password** that are checked against a list of valid users
- We can implement authentication with **HTTP** (Web server) or **PHP scripts**

Prepared by X. Wang

3

## HTTP Authentication

---

- The **HTTP protocol** provides a simple and effective support for user authentication
- A typical HTTP authentication process goes like this:
  - A browser requests a resource on a server that requires authentication
  - The server responds to the request with a 401 (Unauthorized access) response message
  - The browser recognizes the 401 response, and produces a pop-up authentication dialog box to collect a username and password

Prepared by X. Wang

4

## HTTP Authentication

---

- The user enters the username and password, and the browser then resends the original request with an extra header field that encodes the user credentials
- When receiving the request with encoded user credentials, the server will validate whether the user is authorized to access the resource. If it is authorized, the server will send the requested response back to the browser. Otherwise, the request is denied
- For the **Apache** server, it is easy to configure a resource (directory) to be protected with HTTP authentication (in the *httpd.conf* file)

Prepared by X. Wang

5

## PHP Authentication

---

- PHP scripts can also be used to implement the user authentication
- PHP scripts give us more control over the authentication than the HTTP authentication
- PHP provides a global array variable `$_SERVER` to hold some states related to the server
- Two elements `$_SERVER["PHP_AUTH_USER"]` and `$_SERVER["PHP_AUTH_PW"]` hold the **username** and **password** supplied by the user

Prepared by X. Wang

6

### PHP Authentication

- PHP can check if `$_SERVER["PHP_AUTH_USER"]` and `$_SERVER["PHP_AUTH_PW"]` are set
- If the variables are not set, the user has not been authenticated, and then PHP should send a response with the `WWW-Authenticate` header including the status code `401 Unauthorized` to the browser
- If the variables are set, the PHP script needs to check them against the credentials stored in the PHP script
- If the user's credentials match those stored in the script, the user is allowed to use the script. If not, the challenge header is sent to the browser again

Prepared by X. Wang 7

### PHP Authentication

- PHP authentication can be implemented in different ways
- The following are three basic ways
  - Hard-coded authentication: directly hard-code usernames and passwords in the script
  - File-based authentication: store multiple users with their usernames and passwords in a file
  - Database-based authentication: store multiple users with their usernames and passwords in a database table

Prepared by X. Wang 8

### PHP Authentication

❑ Hard-coded authentication

```
<?php
if (!isset($_SERVER["PHP_AUTH_USER"]) || !isset($_SERVER["PHP_AUTH_PW"]) ||
$_SERVER["PHP_AUTH_USER"] != "csci241" || $_SERVER["PHP_AUTH_PW"] != "owens04")
{
    header("WWW-Authenticate: Basic realm=\"Password Required\"");
    header("HTTP/1.1 401 Unauthorized");
    print ("You must provide the proper credentials!"); exit;
}
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head><title>Authenticated Page</title></head>
<body style="background-color: #99CCCC;">
<h1 align="center">Authenticated Page</h1>
<p>Welcome to this page.</p>
</body>
</html>
```

Prepared by X. Wang 9

### PHP Authentication

❑ Hard-coded authentication


- The previous code (`authenticatedpage1.php`) is an authenticated page using hard-coded authentication
- When this page is accessed, the PHP script will run
- The PHP script checks the credentials
- If no credentials, the PHP script first sends the header with `WWW-Authenticate`, and sets the encoding scheme to `Basic` and the realm name to `Password Required`
- The script also sends the status code `401 Unauthorized`

Prepared by X. Wang 10

### PHP Authentication

❑ Hard-coded authentication

- When receiving the special header, the browser will display the following dialog box to ask the user to enter the username and password



Prepared by X. Wang 11

### PHP Authentication

❑ Hard-coded authentication

- If the user doesn't provide the correct credentials, the header will be sent back to the browser again
- The user may have at most three attempts
- If the user clicks the `Cancel` button or submits the username and password over three attempts, the message `"You must provide the proper credentials!"` will be displayed
- If the user's credentials is correct, the HTML document in the page will be sent to the browser

Prepared by X. Wang 12

## PHP Authentication

### File-based authentication

- If a resource is authorized to multiple users with different usernames and passwords, we can store the usernames and passwords in a file
- Each user occupies one line in the file. The username and password of a user are separated with a colon
- The file is better to be stored **outside** the web server document root directory
- In the following example, the file is stored under the current directory

Prepared by X. Wang

13

## PHP Authentication

### File-based authentication

- Assume that we have three users who can access the page `authenticatedpage2.php`
- The following are three usernames and passwords that stored in the file `authenticate.txt`

```
xwang:abcd
cba:abc
cs:123
```

Prepared by X. Wang

14

## PHP Authentication

### File-based authentication

```
<?php
$authorized = false;
if (isset($_SERVER["PHP_AUTH_USER"]) && isset($_SERVER["PHP_AUTH_PW"])) {
    $authFile = file("authenticate.txt");
    foreach ($authFile as $login) {
        list($username, $password) = explode(":", $login);
        $username = trim($username);
        $password = trim($password);
        if ($_SERVER["PHP_AUTH_USER"] == $username &&
            $_SERVER["PHP_AUTH_PW"] == $password)
            { $authorized = true; break; }
    }
}
if (!$authorized) {
    header("WWW-Authenticate: Basic realm='Password Required'");
    header("HTTP/1.1 401 Unauthorized");
    print ("You must provide the proper credentials!"); exit;
}
?>.....
```

Prepared by X. Wang

15

## PHP Authentication

### File-based authentication

- In the previous script, the `file()` function is used to read the password from `authenticate.txt`.
- The `file()` function returns an array in which each element is a string to represent a line from the file
- The `explode()` function is used to split the username and password in each element, and returns another array
- The `list()` function assigns the elements in the new array to the variables listed in the `list()` function

Prepared by X. Wang

16

## PHP Authentication

### Database-based authentication

- This way is similar to the file-based authentication. The difference is that we store users' usernames and passwords in a database table
- When we check the credentials, we need to query the user account table, and check whether the username and password entered by the user are authorized
- Suppose that we have a table `users` that stores all usernames and passwords

Prepared by X. Wang

17

## PHP Authentication

### Database-based authentication

```
<?php
require 'db.php';
$authorized = false;
if (isset($_SERVER["PHP_AUTH_USER"]) && isset($_SERVER["PHP_AUTH_PW"])) {
    if (!($connection = @mysql_connect($hostname, $dbusername, $dbpassword)))
        die("Connection is failed");
    if (!(@mysql_select_db($databasename, $connection))) showerror();
    $username = $_SERVER["PHP_AUTH_USER"];
    $password = $_SERVER["PHP_AUTH_PW"];
    $query = "SELECT username, password FROM users
             WHERE username='{$username}' AND password='{$password}'";
    if (!($result = @mysql_query($query, $connection))) showerror();
    if (mysql_num_rows($result) != 0) $authorized = true;
}
if (!$authorized) {
    header("WWW-Authenticate: Basic realm='Password Required'");
    header("HTTP/1.1 401 Unauthorized");
    print ("You must provide the proper credentials!"); exit;
}
?>
```

Prepared by X. Wang

18

## Form-Based PHP Authentication

- So far we've discussed how to send the `WWW-Authenticate` header to the browser to collect the user's credentials
- We can also use the `HTML form` to collect these information from the user
- In the Session topic, we introduced how to use an HTML form to get the customer's email and password to login to an online shopping store
- We used form to collect the customer's credentials and check them with the user accounts stored in database
- We can also check the credentials with the information stored in a file

Prepared by X. Wang

19

## Form-Based PHP Authentication

- When using the `WWW-Authenticate` header to collect the user's credentials, the username and password entered by the user are stored in the `$_SERVER` array in PHP
- When using the `HTML form` to collect the user's credentials, the entered username and password are stored in the `$_POST` array (assume that the POST method is used in the form)

Prepared by X. Wang

20

## Limiting Access By IP Address

- Sometimes, we may limit access on a resource to a particular network or a particular computer
- For example, the administrative functions in an online shopping store may be restricted to a single computer or several specific computers
- To check a particular network or computer, we need to know the IP address of the client computer
- In PHP, the IP address of the computer from which a request was sent can be obtained by inspecting the variable `$_SERVER["REMOTE_ADDR"]`

Prepared by X. Wang

21

## Limiting Access By IP Address

- Besides checking the username and password, if we also check the IP address from which the authorized resource is requested, we can further limit the access to the authorized resource
- The IP address checking can also be used in the session handling
- In a session handling, the IP address can be stored as a session variable, and then it can be used to validate other user interactions in the same session to see if the access is from the same computer that creates the session

Prepared by X. Wang

22

## Data Encryption

- Directly storing the sensitive data like password and credit card as plain text in a database or a file represents a security risk because insiders, external hackers and others may gain access to the database or the file
- Therefore, a common technique to protect the sensitive data is to encrypt the sensitive data
- Some sensitive data like password is not required to be reversible after encryption
- But others like credit card is required to be reversible

Prepared by X. Wang

23

## Data Encryption

- For the non-reversible sensitive data like password, we can use a one-way encryption algorithm to encrypt the data and store the encrypted data in the database or file
- For the reversible sensitive data like credit card, we need to use a two-way encryption techniques to encrypt the data so that the encrypted data stored in database or file can be decrypted
- First, let's see how to encrypt a password for a user account and store the user information into a database

Prepared by X. Wang

24

## Data Encryption

- The process of protecting a password works as follows
  - First, a new username and password are collected from the user
  - Then the password is encrypted with a one-way encryption function and a new row is inserted into the user table (register operation) that contains the plain text username and the encrypted password
  - Later, when the user returns and logs in to the application, the password entered by the user is encrypted and used to retrieve the account information from the database
  - If the username and encrypted password supplied by the user matches the username and encrypted password stored in the database, the credentials are correct and the user passes the authentication

Prepared by X. Wang

25

## Data Encryption

- PHP provides several functions such as `crypt()`, `md5()`, `mhash()` and so on that can be used for one-way encryption of the sensitive data
- Here we only introduce the `md5()` function. Its syntax is `string md5(string message)`
- This function returns a 32-character hexadecimal number calculated from the source message using the RSA Data Security, Inc. MD5 Message-Digest Algorithm (<http://www.faqs.org/rfcs/rfc1321>)
- `md5()` is a one-way encryption function

Prepared by X. Wang

26

## Data Encryption

- For example, the following PHP code
 

```
$password = "abcd";
print "Password = " . md5($password);
```

 generates the output as
 

```
Password = e2fc714c4727ee9395f324cd2e7f331f
```
- We can see that the password "abcd" becomes a 32-character hexadecimal string after calling `md5()`

Prepared by X. Wang

27

## Data Encryption

- To store data using two-way encryption, a good set of tools is the `Mcrypt` encryption library
- PHP provides a set of functions that can use this library
- But we need to download and install the `libmcrypt` from <http://mcrypt.sourceforge.net/>
- Here we don't discuss the two-way encryption in detail
- We will use the MySQL functions `encode()` and `decode()` to encrypt the credit card number when we discuss how to create a shopping cart in the future

Prepared by X. Wang

28

## Authentication and Session-Based Applications

- In the previous topic, we presented session management as a technique for building stateful applications
- For many applications that require authentication, a session is created when a user logs in, and tracks his interactions until he logs out or the session times out
- The basic pattern of session-based authentication is to authenticate a user's credentials once, and set up a session that record this authenticated status in session variables
- Credentials are usually collected using an HTML form

Prepared by X. Wang

29

## Authentication and Session-Based Applications

- After starting a session, the credentials will not be used for other successive interactions
- Instead, the `session ID` will be used for the successive session interactions
- To protect the session ID to be hijacked, we may store IP address as a session variable
- This can reject the session interaction with the same session ID but from a different computer
- Also, to protect the sensitive data to be hacked, we need to encrypt the sensitive data

Prepared by X. Wang

30



## Authentication and Session-Based Applications

- The third page (welcome.php) is as follows

```
<?php session_start(); ?>
<!DOCTYPE html PUBLIC "-//W3C/DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head><title>Welcome</title></head>
<body>
<?php
if (!isset($_SESSION["cust_id"]) || !isset($_SESSION["ip_addr"]))
die("Invalid operation!!!<br /> Please login first");

$cust_id = $_SESSION["cust_id"];
$ip_addr = $_SESSION["ip_addr"];
if ($ip_addr != $_SERVER["REMOTE_ADDR"])
die("Invalid operation!!!<br /> Please login first");
```

Prepared by X. Wang

37

## Authentication and Session-Based Applications

```
require "db.php";
if (!($connection = @mysql_connect($hostname, $dbusername, $dbpassword)))
die("Could not connect");
if (!(@mysql_select_db($databasename, $connection))) showerror();

$query = "SELECT * FROM customers WHERE cust_id={$cust_id}";

if (!($result = @mysql_query($query, $connection))) showerror();
if (mysql_num_rows($result) == 0)
die("!!!Invalid operation!!!<br /> Please login first");
else {
print "<p>Hello, {$_SESSION["firstname"]} {$_SESSION["lastname"]}<br />";
print "Welcome to this online bookstore!<p>";
}
?>
</body>
</html>
```

Prepared by X. Wang

38

## Authentication and Session-Based Applications

- The first page (logout.php) for the logout is modified as follows

```
<?php
session_start();
if (!isset($_SESSION["cust_id"]) || !isset($_SESSION["ip_addr"]))
die("Invalid operation!!!<br /> Please login first");

$cust_id = $_SESSION["cust_id"];
$ip_addr = $_SESSION["ip_addr"];
if ($ip_addr != $_SERVER["REMOTE_ADDR"])
die("Invalid operation!!!<br /> Please login first");
session_destroy();
header("Location: logout.html");
?>
```

Prepared by X. Wang

39

## Authentication and Session-Based Applications

- The logout.html has not modified

```
<!DOCTYPE html PUBLIC "-//W3C/DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Logout from the Online Bookstore</title>
</head>
<body>
<p><b>Thank you for shopping in this online bookstore!</b></p>
</body>
</html>
```

Prepared by X. Wang

40

## Authentication and Session-Based Applications

- In order to keep the customer information stored in database being consistent, we also need to modify the customer register operation discussed in the topic – “Writing to Web Databases”
- We need to encrypt the password before the password is inserted into the *customers* table through

```
$password = md5($password1);
```
- For the register operation, we only need to modify the PHP file - *saveregister.php*

Prepared by X. Wang

41

## Data Protection on the Web

- If an application transmits sensitive data over the Web, an **encrypted connection** should be provided between the web browser and server
- The **Secure Socket Layer (SSL)** Protocol addresses three goals to provide the encryption services for the data transmission between the web browser and server
  - **Confidentiality:** The content of a message transmitted over the Internet is protected from observers
  - **Integrity:** The content of a message received are correct and have not been tampered with
  - **Authentication:** Both the sender and receiver of a message can be sure of each other's identity

Prepared by X. Wang

42

### Data Protection on the Web

---

- SSL was originally developed by Netscape
- There are two versions: [SSL v2.0](#) and [SSL v3.0](#)
- The SSL protocol is not a standard, and the Internet Engineering Task Force (IETF) has proposed the [Transport Layer Security 1.0 \(TLS\)](#) protocol as an [SSL v3.0](#) replacement (with minor changes)
- The SSL protocol operates as a layer between the HTTP protocol (browser) and the TCP/IP protocol services provided by the host

Prepared by X. Wang

43

### Data Protection on the Web

---

- When the browser sends a request to the server, the message is created as an HTTP message, and is passed to the [SSL layer to be encrypted](#) before it is passed to the TCP/IP service
- The SSL layer, configured into the web server, [decrypts the message from the TCP/IP service](#) on the server-side and then passes it to the web server
- Once SSL is installed and the web server is configured correctly, the HTTP requests and responses are automatically encrypted and decrypted
- [PHP scripting is not required for using the SSL services](#)

Prepared by X. Wang

44

### Data Protection on the Web

---

- URLs to access resources on a secure server begin with [https://](#), which means HTTP over SSL
- The default port for an SSL service is 443, not port 80 as with HTTP
- Today most web browsers and web servers can support SSL
- The detailed information about the SSL can be found at <http://wp.netscape.com/eng/ssl3/>

Prepared by X. Wang

45