Web Technologies — Week 14

Mikheil Rukhaia

International Black Sea University,
Tbilisi, Georgia

January 7, 2016

Outline

HTTP Authentication

Storing Passwords

- Since MD5 is faster hashing function, it is still widely used to store passwords in the database.
- MD5 hash can be “decrypted” using so called rainbow tables.
- The trick to protect users even if the database is compromised and the hash values are stolen is to add a “salt” (usually the username) at the end of password before applying MD5 function.
- Even if the attacker has both the salt and the MD5 hash, it is extremely unlikely that he will find another string that equates to the MD5 hash when concatenated with the salt.

Basic authentication

- Basic authentication is the oldest authentication method available in HTTP.
- Example:
  `http://username:password@example.com/`
- To force authentication, the HTTP 401 response and the type of authentication should be sent.

Basic authentication (ctd.)

- Example:

```
<?php
  if (!isset($_SERVER['PHP_AUTH_USER'])) {
    header('WWW-Authenticate: Basic');
    header('HTTP/1.0 401 Unauthorized');
    echo 'Not authenticated';
    exit;
  } else {
    echo "Hello {$_SERVER['PHP_AUTH_USER']}. ";
    echo "Password: {$_SERVER['PHP_AUTH_PW']}";
  }
?>
```

Digest authentication

- Digest authentication is a way to authenticate without sending a password to the server.
- The method relies on creating a single-direction hash of various data using a shared secret key.
- The secret key is the user password.
- Digest authentication is much better than Basic authentication.

Digest authentication (ctd.)

- The hash is built up of the following parts:
  - realm: a string to display to the user.
  - nonce: uniquely generated string on each 401 response (prevents replay attacks).
  - opaque: a string that should be passed unaltered back to the server.
  - qop: a quoted string specifying the "quality of protection" (used for backward compatibility with value `auth`)
- The calculation of the hash value is specified in the RFC standard.

Digest authentication (ctd.)

- Example:

```
<?php
  $realm = 'Restricted area';
  $users = array('admin' => 'mypass');
  if (empty($_SERVER['PHP_AUTH_DIGEST'])) {
    header('HTTP/1.1 401 Unauthorized');
    header('WWW-Authenticate: Digest realm="'
           .$realm.'",qop="auth",nonce="'
           .uniqid().'",opaque="'
           .md5($realm).'"');
    die('Not authenticated');
  }
```

Digest authentication (ctd.)

- Example:

```
$data = parse($_SERVER['PHP_AUTH_DIGEST']);
if (! $data ||
    ! isset($users[$data['username']]))
  die('Wrong Credentials!');
```

Digest authentication (ctd.)

- Example:

```
// generate the valid response
$A1 = md5($data['username'].':'.$realm
          .':'.$users[$data['username']]);
$A2 = md5($_SERVER['REQUEST_METHOD']
          .':'.$data['uri']);
$valid_response = md5($A1.':'.$data['nonce']
                      .':'.$data['nc']
                      .':'.$data['cnonce']
                      .':'.$data['qop']
                      .':'.$A2);
if ($data['response'] != $valid_response)
  die('Wrong Credentials!');
else echo "Hello {$data['username']}";
```

Digest authentication (ctd.)

- Example:

```
function parse($txt) {
  // protect against missing data
  $needed_parts = array(
    'nonce'    => 1,
    'nc'       => 1,
    'cnonce'   => 1,
    'qop'      => 1,
    'username' => 1,
    'uri'      => 1,
    response'  => 1    );
  $data = array();
  $keys = implode('|',
                  array_keys($needed_parts));
```

Digest authentication (ctd.)

- Example:

```
        preg_match_all(
          '@(' . $keys .
          ')=(?:([\'"])([^\2]+?)\2|([^\s,]+))@',
          $txt, $matches, PREG_SET_ORDER);
        foreach ($matches as $m) {
            $data[$m[1]] = $m[3] ? $m[3] : $m[4];
            unset($needed_parts[$m[1]]);
        }
        return $needed_parts ? false : $data;
    }
?>
```

PHP authentication

Introduction

- Pure PHP system typically displays a login form embedded within other content.
- Once the user authenticates, a cookie is dropped that maintains the user session.
- User cookie can be stolen either via XSS or by traffic sniffing on the network.
- It is much safer to use server-side sessions or encrypted cookies.

Cookies

- A cookie is a small piece of information that scripts can store on a client-side machine.
- Cookies are sent by an HTTP header containing data in the following format:

```
Set-Cookie: NAME=VALUE;
            [expires=DATE;]
            [path=PATH;]
            [domain=DOMAIN_NAME;]
            [secure]
```

- It is possible to `setcookie()` from PHP and retrieve later using `$_COOKIE` superglobal.

Encrypted cookies

- A secure cookie must store:
  - User ID.
  - Last known IP address of the user.
  - Last login date and time (a timestamp).
- Than there are three levels of trust:
  1. IP address matches and timestamp is recent.
  2. IP address matches but the timestamp is old.
  3. IP address does not match.

Encrypted cookies (ctd.)

- To encrypt a cookie, the most common way is to use AES 256bit encryption.

- Example:

```
$cookieData = serialize( $user );
$iv_size = mcrypt_get_iv_size(
        MCRYPT_RIJNDAEL_256, MCRYPT_MODE_CBC);
srand();
$iv = mcrypt_create_iv($iv_size, MCRYPT_RAND);
$encryptedData = mcrypt_encrypt(
        MCRYPT_RIJNDAEL_256, $secret,
        $cookieData, MCRYPT_MODE_CBC, $iv);
setcookie( 'user',
      base64_encode($encryptedData).':'.$iv );
```

Encrypted cookies (ctd.)

- Example:

```
list($encryptedData,$iv) =
  explode(':', $_COOKIE['user']);
$rawData = mycrypt_decrypt(
    MCRYPT_RIJNDAEL_256,
    $secret,
    base64_decode($encryptedData),
    MCRYPT_MODE_CBC,
    $iv
);
$user = unserialize( $rawData );
```

PHP sessions

- HTTP is a stateless protocol.
- The idea of session control is to be able to track a user during a single session on a website.
- In PHP there is `$_SESSION` superglobal available to control user session.

PHP sessions (ctd.)

- Sessions in PHP are driven by a unique session ID, a cryptographically random number.
- The session ID is generated by PHP and stored on the client side for the lifetime of a session.
- It can be passed in a cookie or through URL.
- The session ID is a key allowing to register so-called session variables at the server.
- The session ID is the only information visible at the client side.

PHP sessions (ctd.)

- PHP uses cookies by default with sessions.
- If possible, a cookie will be set to store the session ID.
- It is not mandatory to set cookies manually, it is done by PHP automatically.
- Although it is possible to pass session ID via URL, the feature is disabled by default in `php.ini` for security reasons.
- You can always embed session ID in URLs manually (it is stored in `SID` constant).

Implementing sessions

- The basic steps of using sessions are

1. Starting a session via `session_start()`
2. Registering session variables, e.g.
   ```
   $_SESSION['myvar'] = 10;
   ```
3. Using session variables, e.g.
   ```
   if (isset($_SESSION['myvar']))
   ```
4. Deregistering variables and destroying the session, e.g.
   ```
   unset($_SESSION['myvar']);
   $_SESSION = array();
   session_destroy();
   ```

Implementing sessions (ctd.)

- Example:

```
<?php                    // index.php
  session_start();
  $_SESSION['myvar'] = "Hello from session";
?>
<a href="session.php">go here</a>;

<?php                    // session.php
  session_start();
  echo "Test sessions by {$_SESSION['myvar']}";
// if myvar or entire session no longer needed
  unset($_SESSION['myvar']);
  session_destroy();
?>
```

MySQL Syntax

MySQL database

- In principle PHP can work with any database system.
- MySQL is most popular in web because:
  - it is freely available and easy to install on all major operating systems (UNIX, Windows, and Mac)
  - it is simple to use and includes some handy administration tools (e.g. phpmyadmin)
  - it is a fast, powerful system that copes well with large, complex databases.

MySQL datatypes

- `TINYINT`, `SMALLINT`, `MEDIUMINT`, `INT` and `BIGINT`, possibly `UNSIGNED`.
- `FLOAT` and `DOUBLE`.
- `BIT`.
- `DATE`, `DATETIME` (1 Jan 1000 to 31 Dec 9999), `TIMESTAMP` (1 Jan 1970 to 9 Jan 2038), `TIME`, and `YEAR` (1901 to 2155)

MySQL datatypes (ctd.)

- `CHAR(n)`, `VARCHAR(n)`, `BINARY(n)`, `VARBINARY(n)`
- `TINYTEXT`, `TEXT`, `MEDIUMTEXT`, `LONGTEXT`
- `TINYBLOB` (Binary Large Object), `BLOB`, `MEDIUMBLOB`, `LONGBLOB`
- `ENUM`, and `SET`.

Comparison operators

- = (equal to) and its NULL-safe version <=>
- != or <> (not equal to)
- <, >, <=, >=
- `AND`, `OR`, `NOT`

Commands

- `CREATE` a database, table or index
- `ALTER` the structure of a table
- `DROP` a database or table
- `SELECT` data from one or more tables
- `INSERT` data into a table
- `UPDATE` data in a table
- `DELETE` data from a table

Commands (ctd.)

- Example:

```
CREATE DATABASE mydb;
USE mydb;       -- switch to mydb
CREATE TABLE fruits (
  id     INT UNSIGNED NOT NULL AUTO_INCREMENT,
  name   VARCHAR(30)  NOT NULL UNIQUE,
  color VARCHAR(30) BINARY, # case sensitive
  zone   ENUM ('tropic', 'normal') NOT NULL,
  last   TIMESTAMP, # filled automatically
  PRIMARY KEY (id),
  INDEX (name) /* for faster search by name */
);
```

Commands (ctd.)

- Example:

```
INSERT INTO fruit ( name, color, zone )
VALUES ( 'banana', 'yellow', 'tropic' );

INSERT INTO fruit ( name, color )
VALUES ( 'ananas', 'green', 'tropic' );

INSERT INTO fruit ( name, color, zone )
VALUES ( 'apple', 'red', 'normal' );
```

Commands (ctd.)

- Example:

```
SELECT DISTINCT *
FROM fruits
WHERE name NOT LIKE '%anana_'
ORDER BY id DESC, name ASC;

SELECT color, count(name)
FROM fruits
GROUP BY color
ORDER BY count(name) DESC;
```

Commands (ctd.)

- Example:

```
SELECT DISTINCT fruits.name, users.name
FROM fruits, users
WHERE fruits.color = users.color;
```

PHP libraries

Libraries

- mysqli (MySQL improved) is specifically tied to MySQL, and provides the most complete access to MySQL from PHP.
- PDO (PHP Data Objects) is an object-oriented extension sitting between the MySQL server and the PHP engine.
- Choosing between these two extensions can be a topic of religious debate among PHP developers.
- PDO is easier and quicker to learn, but mysqli has some advanced capabilities.

Connection

- To make a connection to a database you need its DSN, username and password.
- Database Source Name (DSN) consists mainly of `host` and `dbname`, describing the database to connect to.
- If the `host` is not specified, `localhost` is assumed.
- Example

```
$dsn = "mysql:host=localhost;dbname=mydb";
  // equivalent to
$dsn = "mysql:dbname=mydb";
```

Connection (ctd.)

- Example (can you tell what are the security issues here?)

```
$dsn = "mysql:dbname=mydatabase";
$username = "root";
$password = "mypass";
$conn = new PDO( $dsn, $username, $password );
```

- To close the connection, simply assign a `null` to it, e.g. `$conn = null`.

Handling errors

- It is very hard to detect server errors without throwing exceptions when they occur.

- Example:

```
try {
  $conn = new PDO($dsn, $username, $password);
  $conn->setAttribute(PDO::ATTR_ERRMODE,
                      PDO::ERRMODE_EXCEPTION);
} catch ( PDOException $e ) {
  echo "Connection failed: ".$e->getMessage();
}
```

- NEVER display exception messages on the web page for security reasons, instead log them in a file, or email to the Webmaster.

Accessing data

- To send SQL statements to the MySQL server, use the `query` method of the PDO object.

- Example:

```
$sql = "SELECT * FROM fruit";
$rows = $conn->query( $sql );
foreach ( $rows as $row ) {
  echo "name = ".$row['name']."<br />";
  echo "color = ".$row['color']."<br />";
}
```

Prepared statements

- When receiving an input from a user, it is better to `prepare()` the statement.
- You can use `bindParam()` to bind a PHP variable with an sql statement parameter.
- The variables are evaluated when `execute()` is called.
- You can retrieve number of rows affected using `rowCount()` and the query result using `fetchAll()`.

Prepared statements (ctd.)

- Example

```
$id = 8;
$email = "user@example.com";
$sql = "UPDATE users
        SET email = :email
        WHERE id = :id";
try {
 $st = $conn->prepare( $sql );
 $st->bindValue(":id",$id,PDO::PARAM_INT);
 $st->bindValue(":email",$email,PDO::PARAM_STR);
 $st->execute();
 echo $st->rowCount()." rows affected.";
} catch ( PDOException $e ) {
  echo "Query failed: ".$e->getMessage();
}
```

Laboratory Work

Exercises

- Create a demo web site, having both kind of HTTP authentication, indicating the authentication type after login.
- Implement PHP authentication with session control (mandatory for your course projects as well).
- Test whether later site works when cookies are disabled in the browser. If not, find a solution.

Discussion?!