Web Technologies — Week 13

Mikheil Rukhaia

International Black Sea University,
Tbilisi, Georgia

December 31, 2015

Outline

1. RESTful Web Services
2. Web Application Security
3. Digital Certificates
4. Laboratory Work

RESTful Web Services

REST

- REST (REpresentational State Transfer) is a software architecture describing a client-server communication.
- REST is based on two main concepts: resources (subjects) and verbs.
- A client can request that an action be executed on a resource and then “rest” before another request.
- The client does not maintain a constant connection to server, but it can store data to send with each request (cookies).
- In other words: each request to server must be self-descriptive, containing enough context to be processed.

REST (ctd.)

- An application following the REST model is often called RESTful.
- The most common RESTful service is HTTP (with verbs GET and POST).
- Although a RESTful service does not have to be based on HTTP, PHP and HTTP do make a natural pair for it.

Verbs

- GET: used solely for retrieving data; it should never modify any of the data on the server.
- HEAD: used to retrieve just the header information but none of the content.
- POST: used to edit and create new resources.
- PUT: only used to edit a resource (opposite of GET).
- DELETE: used to remove a resource.

cURL

- cURL is a command-line tool that is useful for making HTTP/S requests.
- In fact it supports many protocols, like FTP/S, Telnet, SSH, etc.
- See `http://curl.haxx.se` for the detailed manual.
- Example:

```
> curl www.ibsu.edu.ge
> curl --head www.ibsu.edu.ge
> curl -O www.ibsu.edu.ge/index.php
> curl -X POST -d "submit=ok" localhost
```

libcurl

- libcurl is a library that is used by cURL.
- PHP provides functionality to use libcurl from its scripts.
- Connection to libcurl is one of the major advantages of PHP to create RESTful applications.
- libcurl provides many functions, but the most important ones are: `curl_init()`, `curl_setopt()`, `curl_exec()`, and `curl_close()`.

libcurl (ctd.)

- Example: using GET verb.

```
<?php
  $ch = curl_init();
  curl_setopt($ch,CURLOPT_URL,"www.ibsu.edu.ge")
  curl_setopt($ch,CURLOPT_HEADER,false);
  curl_setopt($ch,CURLOPT_RETURNTRANSFER,true);
  $data = curl_exec($ch);
  curl_close($ch);
  echo $data;
?>
```

libcurl (ctd.)

- Example: using POST verb.

```
<?php
  $postData = array("name"=>"value");
  $ch = curl_init();
  curl_setopt($ch,CURLOPT_URL,"www.ibsu.edu.ge")
  curl_setopt($ch,CURLOPT_HEADER,false);
  curl_setopt($ch,CURLOPT_POST,true);
  curl_setopt($ch,CURLOPT_POSTFIELDS,$postData);
  curl_setopt($ch,CURLOPT_RETURNTRANSFER,true);
  $data = curl_exec($ch);
  curl_close($ch);
  echo $data;
?>
```

libcurl (ctd.)

- Example: using HEAD verb.

```
<?php
  $ch = curl_init();
  curl_setopt($ch,CURLOPT_URL,"www.ibsu.edu.ge")
  curl_setopt($ch,CURLOPT_HEADER,true);
  curl_setopt($ch,CURLOPT_RETURNTRANSFER,true);
  curl_setopt($ch,CURLOPT_CUSTOMREQUEST,"HEAD");
  $data = curl_exec($ch);
  curl_close($ch);
  echo nl2br($data);
?>
```

Web Application Security

Strategies

- Web application security is one of the major headaches of web programmers.
- There are too many computers in the Internet and not all of their users are “trusted”.
- The hardest task is to find balance between security and business needs.
- Security is not a feature, it is a constant part of the design, that should be maintained even after the deployment of the project.

Balancing Security and Usability

- The task: reduce the chance that cracker programs guess a user password.
- Solution: make four different login screens, where you require different passwords and require that all of them are changed at least once a month (no old passwords allowed).
- Consequence: your application would be secure, but nobody would bother to use it!

Main threats to face

- Access or modification of sensitive data (to keep passwords, credit card numbers and the like safe).
- Loss or destruction of data (a server, or a hard disk is dead).
- Denial of Service (DoS and DDoS attacks).
- Malicious code injection (Cross Site Scripting (XSS) and Cross Site Request Forgery (CSRF) attacks).
- Compromised server (invaders gain a root access).

Who can attack

- Crackers (not hackers, most hackers are honest programmers).
- Infected machines.
- Disgruntled employees.
- Hardware thieves (e.g. a cleaning lady unplugging a cable and the like).
- Ourselves and the code we write (unpleasant news).

Securing the code

- Filter all user input, that comes from external sources.
- Double-check expected values, even for simple types.
- Example: if you have radio buttons for gender, make sure that proper value is passed:

```
<?php
  switch ($_POST['gender']) {
    case 'Male':
    case 'Female':
      // do something
      break;
    default:
      echo "Invalid gender!";
      break;
  } ?>
```

Securing the code (ctd.)

- Example: typecast simple types

```
$age = (int) $_POST['age'];
if ($age < 18 && $age > 65)
  echo "Invalid age!";
```

- Example: check dates and more complex structures

```
$ddmmyyyy = split($_POST['birth_date'], '/');
if (count($ddmmyyyy) != 3 ||
      !checkdate(
         $ddmmyyyy[0],
         $ddmmyyyy[1],
         $ddmmyyyy[2]
       )
    )
  echo "Invalid date!";
```

Securing the code (ctd.)

- To prevent XSS attacks, escape all output.
- Use `htmlspecialchars()` or `htmlentities()` functions to escape the output, coming from outer sources.
- To prevent SQL injection, filter and escape all strings sent to database using functions like `mysql_escape_string()`, etc.
- Never include database connection information directly in the files accessible from the Internet.
- Never put include or other files, which should not be accessed directly from the user in the root directory.

Securing the code (ctd.)

- Example: if your root folder is `/home/www/html`, put include files in `/home/www/lib`.

```
<?php
  // this is /home/www/lib/dbconnect.php
  $db_server = 'localhost';
  $db_user_name = 'example';
  $db_password = 'secret';
  $db_name = 'somedb';
?>
<?php
  // this is /home/www/html/index.php
  include('../lib/dbconnect.php);
  $conn = @new mysqli($db_server,$db_user_name,
                    $db_password,$db_name);
    // etc
?>
```

Securing the code (ctd.)

- PHP has feature to execute programs on server via `exec` or back quotes (‘).
- Never allow user input to be executed in this way.
- At the very least, use `escapeshellcmd()` function and possibly more filtering.
- Example:

```
<?php
  // generate a pdf for a user
  `pdflatex /var/www/temp/test.tex`;
?>
```

Securing the code (ctd.)

- Preventing CSRF attacks is more complex.
- Example: facebook login form was (maybe it still is) vulnerable to CSRF attacks for a long time.
- The rules are following:
  1. Always use POST method (in case of search form and the like GET is also ok).
  2. Make your site safe from XSS attacks.
  3. Use anti-CSRF keys in your forms.

Securing the code (ctd.)

- Example:

```
<?php
  $key = sha1(microtime());
  $_SESSION[’csrf’] = $key;
  if(!isset($_SESSION[’csrf’]) ||
     $_SESSION[’csrf’] !== $_POST[’csrf’])
        die "CSRF attack";
?>

<form action="this.php" method="post">
  <input type="hidden" name="csrf"
         value="<?php echo $key; ?>" />
  <!-- Some other form fields -->
</form>
```

Web server security

- Look for a hosting that:
    - Gives entire directory tree, not only a document root.
    - Has a good support (online documentation, up-to-date software, `php.ini` information, etc.)
    - Offers a trial period, money-back guarantees and the like.

Database server security

- Create database users with minimum permissions and only add permissions later if they absolutely need it.
- Make sure that an error occurs when connecting to database:
  - without specifying a username and password.
  - as root with an incorrect password or without specifying a password.
  - as a user and accessing a table for which the user should not have permission.
  - as a user and accessing system databases or permissions tables.

Protecting the network

- If you need to set up your own server, you should:
  - Install and run only necessary software.
  - Install a firewall and configure it properly (open only necessary ports like 80 and 443)
  - Use Demilitarized Zone (DMZ)
  - Physically secure the server.

Disaster planning

- Make data back-up daily and take it off-site to another facility.
- Create server configuration recovery scripts and store them off-site as well.
- Store source code in multiple (secure) locations.
- Make arrangements with hardware provider to get new hardware immediately if necessary.
- Prohibit all members of your team traveling together.

Digital Certificates

Cryptography basics

- Cryptography is nearly 4,000 years old but came of age in World War II.
- It consists of two parts: encryption and decryption.
- Encryption algorithm is a mathematical process to transform information into a seemingly random string of data.
- Decryption algorithm is a reverse process.
- Hash function is a non-reversible encryption algorithm.
- Passwords usually are stored using hash functions.

Hash functions

- The most common hash functions are MD5 and SHA-1.
- Although MD5 is faster than SHA-1, the latter is more secure.
- In 2004 it was shown that MD5 is not collision resistant.
- In 2008 researchers faked MD5 based SSL certificate validity.
- Since then MD5 is officially announced as cryptographically broken.

Hash functions (ctd.)

- In 2005 it was shown that theoretically SHA-1 is also collision resistant.
- So far no practical example of such collision is published.
- In 2013-2014 major companies such as Microsoft, Google and Mozilla announced that they will not accept SHA-1 based SSL certificates from 2017.
- Alternative is SHA-2 family (especially SHA-512), derived from SHA-1.
- There is also ongoing standardization process of SHA-3, a new hash function NOT derived from SHA-2.

Encryption algorithms

- Encryption algorithms are divided into two parts: the ones using a secret key and the others using public and private keys.
- Data Encryption Standard (DES), developed by IBM in the 1970s, is widely used secret key algorithm.
- Rivest Cipher (RC2, RC4, RC5, RC6) and Advanced Encryption Standard (AES) are other more secure examples using secret key algorithm.
- Obvious flaw: a secure way is needed to deliver the secret key to recipient.

Encryption algorithms (ctd.)

- To fill the gap, an encryption algorithm based on public and private keys was published in 1978.
- The public key is used to encrypt messages and the private key to decrypt them.
- The most common public key algorithm is RSA, developed by Rivest, Shamir, and Adelman at MIT.
- A public key system is used to transmit the key for a secret key system that will be used for the remainder of communication.
- Such hybrid systems are used because secret key systems are much faster than public key systems.

Digital Signatures

- Digital signatures reverse the role of public and private keys.
- This means that sender can encrypt a message with private key and anyone having a public key can read it.
- Digital signatures are really useful, because the recipient can be sure that the message has not been modified, and the sender can not repudiate, or deny sending, the message.
- Problem: public key encryption is very slow and inefficient for large messages.

Digital Signatures (ctd.)

- Solution:
  - A sender generates a hash value (typically via SHA-1) of the message and encrypts only the hash value (creates a signature).
  - Then message is sent together with its signature via normal unsecure way.
  - The receiver decrypts the signature and creates the hash value of the message in the same way as sender did.
  - If the hash values match, message was not altered.

Secure Web Servers

- Secure Web Servers are called those servers, that allow Secure Socket Layer (SSL) or Transport Layer Security (TLS) connections.
- SSL/TLS connections are based on digital certificates, that combines a public key and organization's (or individual's) details in a signed digital format.
- Problem: anybody can generate and sign a certificate claiming to be anybody he likes.
- Solution: trusted third parties, called certifying authorities (CAs).

Certifying authorities

- CAs issue a certificate after they have seen proof of the persons or companys identity.
- The exact process to get a certificate varies between CAs.
- The best known CAs are VeriSign and its child company Thawte.
- The cheapest ones are Network Solutions and GoDaddy.

Certifying authorities (ctd.)

- The certificate does not guarantee that you are dealing with somebody reputable, but if you are ripped off, you have a good chance of having a real physical address and somebody to sue.
- It is a network of trust: if you trust CA, you can then choose to trust the people they choose to trust and then trust the people the certified party chooses to trust.

Laboratory Work

Exercises

- Create (or use existing) student registration form and apply all the security rules given on slides to its handler.
- Test your security using `curl`.

Discussion?!