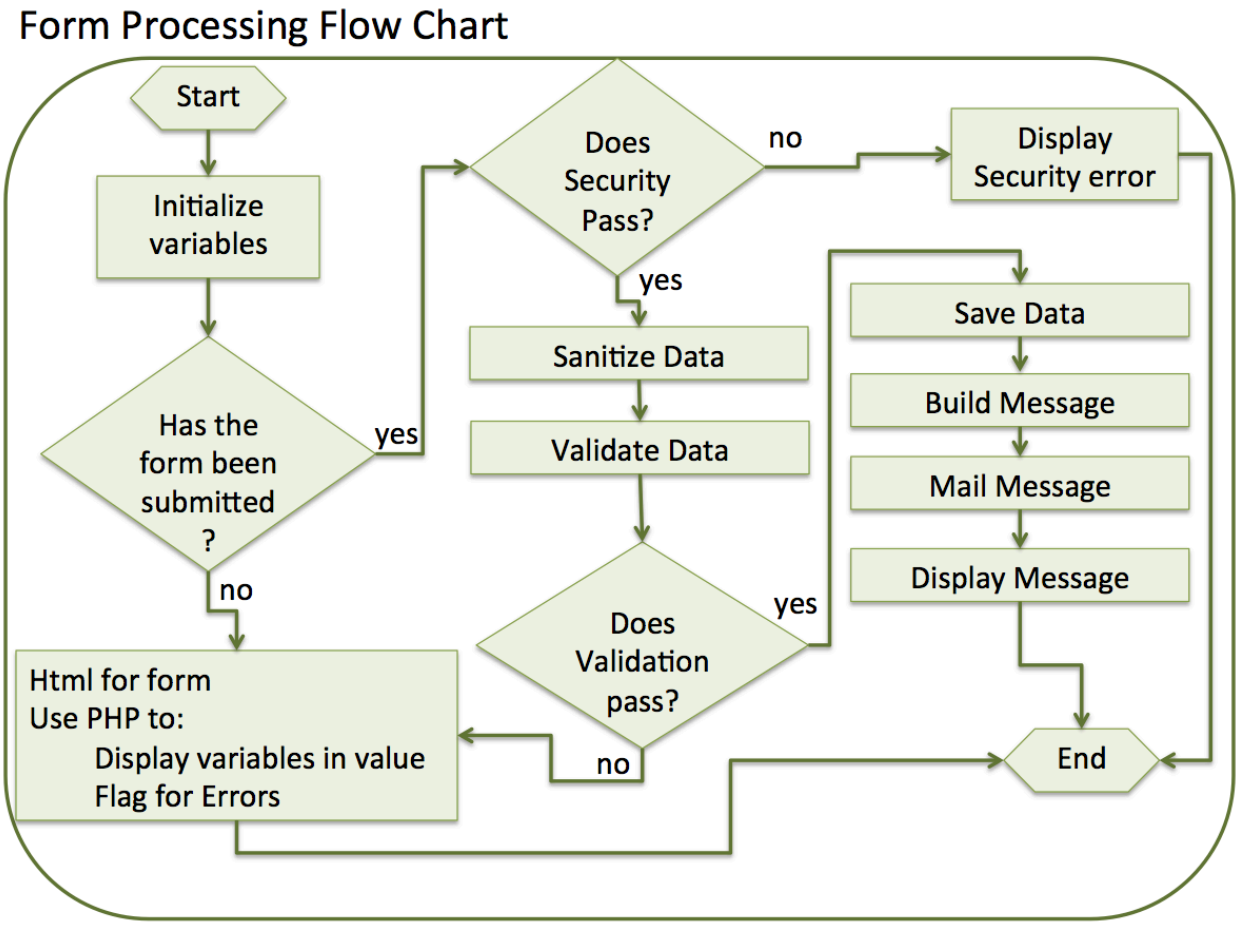
Chapter 5 Form Processing – Security Check



The above diagram represents the processing that needs to happen for every form and corresponds to the GitHub repository commits: https://github.com/Robert-Erickson/php-form-template/commits/master. Notice the diagram can be placed in three columns. The first column is for starters really just html with a small dab of php and is commits one through four.

Lets look at the diamond shape in column one. I guess the first thing you need to do is determine if the form was actually submitted. Have you noticed that when you first come to the form page the post array is empty until you hit submit? We don't want to just check if the post array is empty we want to be a little more specific (you could have more than one form on a page) by looking to see if the name of the submit button is in the post array. So at the start of section two we can use the PHP method isset [http://php.net/manual/en/function.isset.php] which checks to see if the variable is in memory:

```
if (isset($_POST["btnSubmit"])) {
```

Just be sure that you use the name you gave your submit button. The closing curly bracket } would be at the end of Section 2.

```
} // ends if form was submitted.
```

Looking at the flow diagram our next questions in column two is: Do we pass security? It is best to make a function for this check. This way we can start with a dummy check and increase the rigor of the check without changing the flow of the main code. We are going to make a small security check but as you learn more about security in other classes you can add to this function for increased security.

A good name for our function to do a security check would be securityCheck (NOTE: I used a lower case s to match my GitHub form example, I try to be consistent but I don't always get it right). The value we would need would be the URL of our web page. We have part of this from top.php but we will need to the domain without the http: part. We can do this in top.php like this:

```
$domain = "//";
$server = htmlentities($_SERVER['SERVER_NAME'], ENT_QUOTES, "UTF-8");
$domain .= $server;
```

The server name would have the url like www.uvm.edu and of course we have to use our security friend htmlentities as this is information that comes from the clients computer.

Then in our form page we can define a variable for the url like this:

```
$thisURL = $domain . $phpSelf;
```

Remember $phpSelf has already been defined in top.php. Speaking of top.php remember I said I generally put my functions in a separate file? I will put this security function in a file called security.php that I have saved in the lib (short for library) folder. Since security is so important if the file does not get included I want PHP to stop so I am going to use the require statement instead of the include statement. This works the same way as an include statement except that the require statement will produce a fatal error (E_COMPILE_ERROR) and stop the script not producing any HTML code. The include will only produce a warning (E_WARNING) and the script will continue and send HTML code though it will be missing the file include. The once part says only include it once don't bother to include a second time if it is already there. We should always do this for anything that includes functions as if you try to include a function twice it will result in a php error.

```
require_once('lib/security.php');
```

We can start our function as a dummy function to test:

```
//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//performs a few security checks
```

```
function securityCheck ($thisURL) {
     return true;
}
```

This way we can test the logic of our code to make sure that it is correct (corresponds to Github commits five and six).

I want to introduce the PHP die [http://php.net/manual/en/function.die.php] method to stop PHP from doing anything else. Normally you don't want to stop a program from running but in the case of someone trying to hack into our site sometimes it is best to just stop. It is also a way to not use an else statement as part of the if. Let's look at this code:

```
/@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
//
// SECTION: 2a Security
//
if (!securityCheck($thisURL)) {
       $msg = "<p>Sorry you cannot access this page. ";
       $msg.= "Security breach detected and reported</p>";
       die($msg);
}
```

Notice I have used an ! (exclamation point)so this if condition is read as

if not a securityCheck

or

if securityCheck equals false

```
if (securityCheck($thisURL) == false) {
```

Hmm maybe I should have changed my function name to passSecurityCheck or securityCheckPassed as it might read better. Coding can get tricky sometimes and because of that reason I just want to give you a security check function that looks like this:

```
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
// performs a simple security check
// start off thinking everything is good until a test fails
// when it is a form page check to make sure it submitted to itself

function securityCheck($myFormURL = "") {
  $debugThis = false;

  $status = true;

  if ($myFormURL != "") {
    $fromPage = htmlentities($_SERVER['HTTP_REFERER'], ENT_QUOTES, "UTF-8");
```

```
    $fromPage = preg_replace('#^https?:#', '', $fromPage);

    if ($debugThis){
        print "<p>From: " . $fromPage . " should match your Url: " ;
        print $myFormURL;
    }

    if ($fromPage != $myFormURL) {
        $status = false;
    }
  }
  return $status;
}
```

What this function does is return true if it passes or false if it does not. Inside the function it compares the url you sent to the page with the url of this page. They have to match or it fails. It is not the end all to security but this is a simple start. The nice thing is that by using a function we can make the function as complex as we are able to and it does not change the basic logic of our web page. Only the function would change.

Summary

Security is a large problem on the web and we do to everything we can to help stop the problem.

Php Htmlentities is a simple built in function we can use that will convert all applicable characters to HTML entities. This will convert JavaScript to just plain text and not execute.

Self Test Questions

1. **Simple Security**
   Create a securityCheck function of your own that reads a variable and determines whether the variable is secure or not. If the variable is set to 0, it is secure. If it is set to 1, it is not secure.
2. **Security Extended**
   Add onto your security check function from the previous problem by using the *die()* function to print a message if the value is not secure.
3. **Security Further Extended**
   Edit your securityCheck function from the previous problem so that it checks two values instead of one, and returns true if they are both equal to 0.
4. **Security With Domain**
   Write a php security check function that gets the url of the file and checks if it is empty. If its not empty, for the purpose of this exercise, the page is considered secure. Otherwise, the page is considered not secure. Print the security level on screen.
5. **Security With Domain and Die**
   Add onto your code from question 4 so that the program prints some important data on screen if the page is secure, and quits execution if its not.

Answers

1. **Simple Security**

```
<?php
$myValue = 0;

function securityCheck($theValue){
        if($theValue == 0){
                return true;
        }
        if($theValue == 1){
                return false;
        }
}

print securityCheck($myValue);
?>
```

2. **Security Extended**

```
<?php
$myValue = 0;

function securityCheck($theValue){
        if($theValue == 0){
                return true;
        }
        if($theValue == 1){
                return false;
        }
}
if(!securityCheck($myValue){
        $msg = “<p>Value not secure!</p>”;
        die($msg);
}
else {
        print “<p>Value secure.</p>”;
}
?>
```

3. **Security Further Extended**

```
<?php
$myValue = 0;
$mySecondValue = 1;
```

```
function securityCheck($theValue,$secondValue){
        if($theValue == 0 and $secondValue == 0){
                return true;
        }
        else {
                return false;
        }
}
if(!securityCheck($myValue){
        $msg = “<p>Value not secure!</p>”;
        die($msg);
}
else {
        print “<p>Value secure.</p>”;
}
?>
```

4. **Security With Domain**

```
<?php
$domain = “//”;
$server = htmlentities($_SERVER[‘SERVER_NAME’], ENT_QUOTES, “UTF-8”);
$domain .= $server;

function securityCheck($d){
        if($d != “”){
                return true;
        else {
                return false;
        }
}

if(securityCheck($domain)){
        print “<p>Secure</p>”;
} else {
        print “<p>NOT SECURE!</p>”;
}
?>
```

5. **Security With Domain and Die**

```
<?php
$domain = “//”;
$server = htmlentities($_SERVER[‘SERVER_NAME’], ENT_QUOTES, “UTF-8”);
$domain .= $server;
```

```
function securityCheck($d){
	if($d != “”){
		return true;
	else {
		return false;
	}
}

if(securityCheck($domain)){
	print “<p>Secure</p>”;
} else {
	$msg = “NOT SECURE”;
	die($msg);
}

print “<p>IMPORTANT DATA</p>”; // wont get printed if it dies!
?>
```