Web Hosting & Security 2 - Nov 9

Defending Against Cross Site Scripting (XSS)

Recall XSS: when malicious code is injected through a form on the website
Prevent this by validating inputs & sanitizing variables (input encoding).

Filtering User Inputs

```
<!DOCTYPE html>
<html lang="en">
<head> <meta charset="utf-8" /> <title>My Basic Form</title> </head> <body> <h1
style="text-align:center;">Newsletter Sign-Up</h1>
<form name="MyForm" method="Post" action="thankyou.html">
  <div style="text-align:center;">
    <input type="text" name="fullname" placeholder="Your name">
    <input type="text" name="email" placeholder="Your email">
    <input type="submit" name="submit" value="Submit"
onClick="javascript:return validateInputs();">
  </div>
</form>

<script> function validateInputs() {
var SubmitForm; var FormErrors;      //declaration of variables
SubmitForm = true; //Initially set SubmitForm to true.

//Retrieve variables to be validated and sanitized
//Assume they are dangerous for now
var fullname = new String(document.MyForm.fullname.value);
var email = new String(document.MyForm.email.value);

//Check that the user inputs are not blank
if ( fullname.length<1 || email.length<1 ) {
  FormErrors = "All fields are mandatory. Please complete the form.";
  SubmitForm = false;
} else {
//Set up a filter for the pattern of an email
  var filter =
/^([\w-]+(?:\.[\w-]+)*)@((?:[\w-]+\.)*\w[\w-]{0,66})\.([a-z]{2,6}(?:\.[a-z]{2})
?)$/i;

//Use test() method to check user email against the filter
if (!filter.test(email)) {
  FormErrors = "Your form contains invalid field entries. Please correct your
form before submitting";
```

```
  SubmitForm = false; } }

if (SubmitForm == false) { //The form cannot be submitted.
  alert(FormErrors);
  return false;
} else {
  //SANITIZE user inputs by allowing only [a-z 0-9 _ - . @]
  //strip forbidden characters
fullname = fullname.replace(/[^a-z0-9\s\-]/gim,"");
fullname = fullname.trim();
email = email.replace(/[^a-z0-9_@.-]/gim,"");
email = email.trim();

//ready to submit
document.MyForm.submit(); } }


</script> </body> </html>
```

Learn more about referencing characters: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Regular_Expressions/Character_Classe s

test() method: https://www.w3schools.com/jsref/jsref_regexp_test.asp

Defending Against Directory Traversal

Don’t open the door

Avoid passing any user supplied input to the file system in the first place.

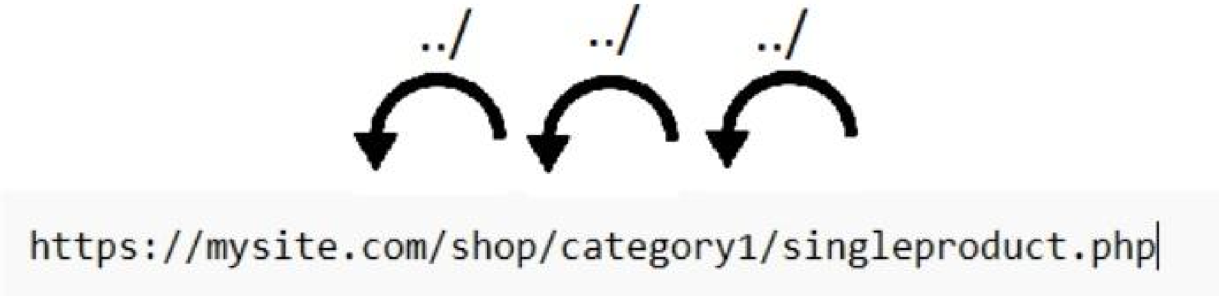
BASE_DIRECTORY

Establish the base directory in your code.

Sanitize User Input Variables

Same as with defending against XSS

Use Relative Paths



https://mysite.com/shop/category1/singleproduct.php

This function is from Preston to sanitize the file name and remove any ../
This looks good to me. Thanks Preston.

```
function sanitizeFilename(filename) {
  let sanitized = filename.replace(/\.\.\//g, '');
  return sanitized;
}
```

The OWASP Foundation

The Open Web Application Security Project® (OWASP) is the world's largest non-profit foundation concerned with software security.

Top 10 Security Risks

https://owasp.org/www-project-top-ten/

WordPress Security

When new versions that include security patches are released, update WordPress theme/plugin versions to ensure that your website is not exposed to preventable attacks

Wordfence

The Wordfence dashboard will give you an overview of your website security, and it is on this page that you will see any security notifications or vulnerabilities that may have been identified in a recent scan.

Websites do slow down while they are being scanned. To avoid having a negative impact on user experience, security scans are best performed during low traffic periods. Regular security scans allow you to identify and patch the vulnerabilities so that your website is not exposed to attack.

Git Reviews

Version control allows teams to review code before it is live, and to restore previous versions if something goes wrong with the live version. In the context of performing WordPress version and plugin security updates, Git's version control allows you to undo and analyze any updates that would cause a display or compatibility problem.

Best Practices

- Commit often. Every time you "commit", you are capturing a snapshot of the code base. This will be a version that you could return to if needed.
- When you commit you can and should leave a detailed log message explaining why you are making this commit and what is contained within it.
- Make sure to git pull (fetch) the global copy so you are working with the most recent version of the code.
- Use the staging area to collect/review a group of edits before writing them to a commit.