

Project 2 | 1Stop | CS_KLEEN Security

Vikash Kumar | wiryvikash15@gmail.com

Problem Statement:

Take some website(vulnerable website). Scan using OWASP ZAP Tool (quick/automated). Set Vulnerabilities --> Make report with mitigations.

Vulnerability: Cross Site Scripting (Reflected)

URL:

GET:

`http://testphp.vulnweb.com/hpp/params.php?p=%3Cscript%3Ealert%281%29%3B%3C%2Fscript%3E&pp=12`

Description:

Cross-site Scripting (XSS) is an attack technique that involves echoing attacker-supplied code into a user's browser instance. A browser instance can be a standard web browser client, or a browser object embedded in a software product such as the browser within WinAmp, an RSS reader, or an email client. The code itself is usually written in HTML/JavaScript, but may also extend to VBScript, ActiveX, Java, Flash, or any other browser-supported technology.

When an attacker gets a user's browser to execute his/her code, the code will run within the security context (or zone) of the hosting web site. With this level of privilege, the code has the ability to read, modify and transmit any sensitive data accessible by the browser. A Cross-site Scripted user could have his/her account hijacked (cookie theft), their browser redirected to another location, or possibly shown fraudulent content delivered by the web site they are visiting. Cross-site Scripting attacks essentially compromise the trust relationship between a user and the web site. Applications utilizing browser object instances which load content from the file system may execute code under the local machine zone allowing for system compromise.

There are three types of Cross-site Scripting attacks: non-persistent, persistent and DOM-based.

Non-persistent attacks and DOM-based attacks require a user to either visit a specially crafted link laced with malicious code, or visit a malicious web page containing a web form, which when posted to the vulnerable site, will mount the attack. Using a malicious form will oftentimes take place when the vulnerable resource only accepts HTTP POST requests. In such a case, the form can be submitted automatically, without the victim's knowledge (e.g. by using JavaScript). Upon clicking on the malicious link or submitting the malicious form, the XSS payload will get echoed back and will get interpreted by the user's browser and execute. Another technique to send almost arbitrary requests (GET and POST) is by using an embedded client, such as Adobe Flash.

Persistent attacks occur when the malicious code is submitted to a web site where it's stored for a period of time. Examples of an attacker's favorite targets often include message board posts, web mail messages, and web chat software. The unsuspecting user is not required to interact with any additional site/link (e.g. an attacker site or a malicious link sent via email), just simply view the web page containing the code.

Payload:

```
http://testphp.vulnweb.com/hpp/params.php?p=%3Cscript%3Ealert%20(%22You%20are%20Hacked%20buddy!!!%20change%20your%20configuration%22)%3C%2Fscript%3E&pp=12
```

Remedies:

Preventing cross-site scripting is trivial in some cases but can be much harder depending on the complexity of the application and the ways it handles user-controllable data.

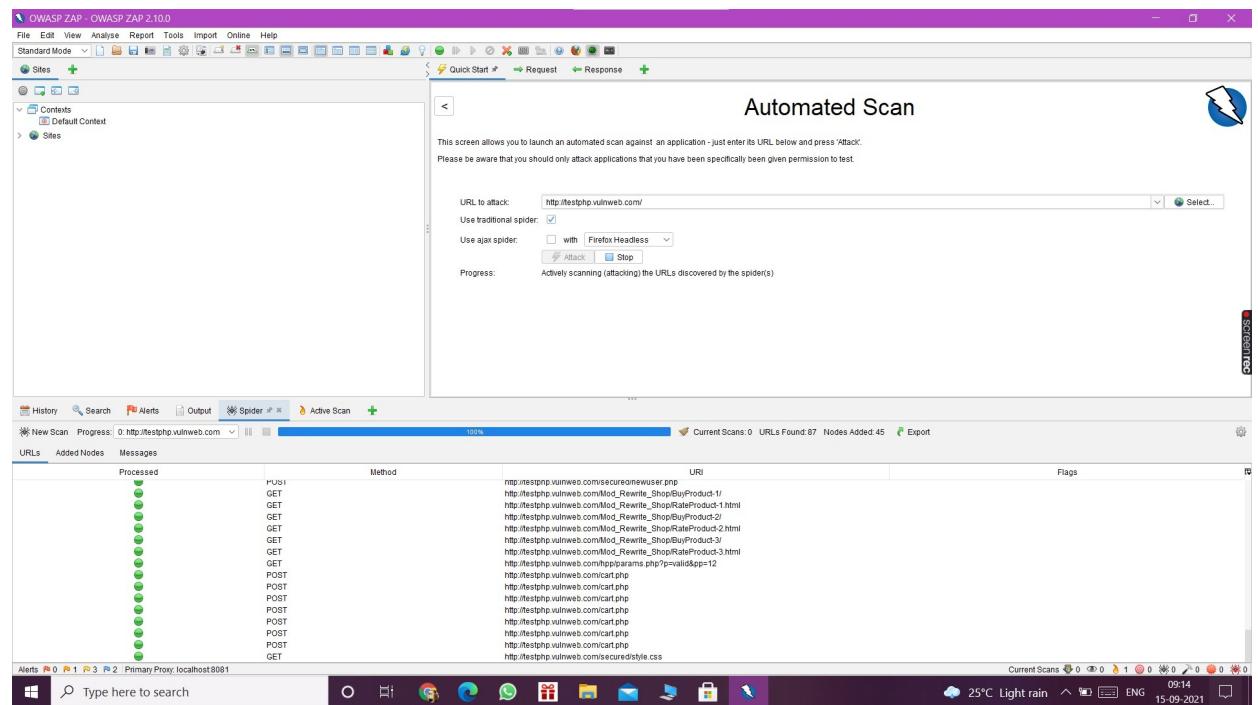
In general, effectively preventing XSS vulnerabilities is likely to involve a combination of the following measures:

- **Filter input on arrival.** At the point where user input is received, filter as strictly as possible based on what is expected or valid input.
- **Encode data on output.** At the point where user-controllable data is output in HTTP responses, encode the output to prevent it from being interpreted as active content. Depending on the output context, this might require applying combinations of HTML, URL, JavaScript, and CSS encoding.
- **Use appropriate response headers.** To prevent XSS in HTTP responses that aren't intended to contain any HTML or JavaScript, you can use the Content-Type and X-Content-Type-Options headers to ensure that browsers interpret the responses in the way you intend.
- **Content Security Policy.** As a last line of defense, you can use Content Security Policy (CSP) to reduce the severity of any XSS vulnerabilities that still occur.

Reference:

<https://portswigger.net/web-security/cross-site-scripting>

Screenshot 1:



Screenshot 2:

The screenshot shows the OWASP ZAP interface with the 'Active Scan' tab selected. The 'URL to attack' field contains <http://testphp.vulnweb.com/>. The 'Attack' button is highlighted in yellow. Below the table, the status bar shows 'Current Scans: 1 Num Requests: 434 New Alerts: 0 Export'.

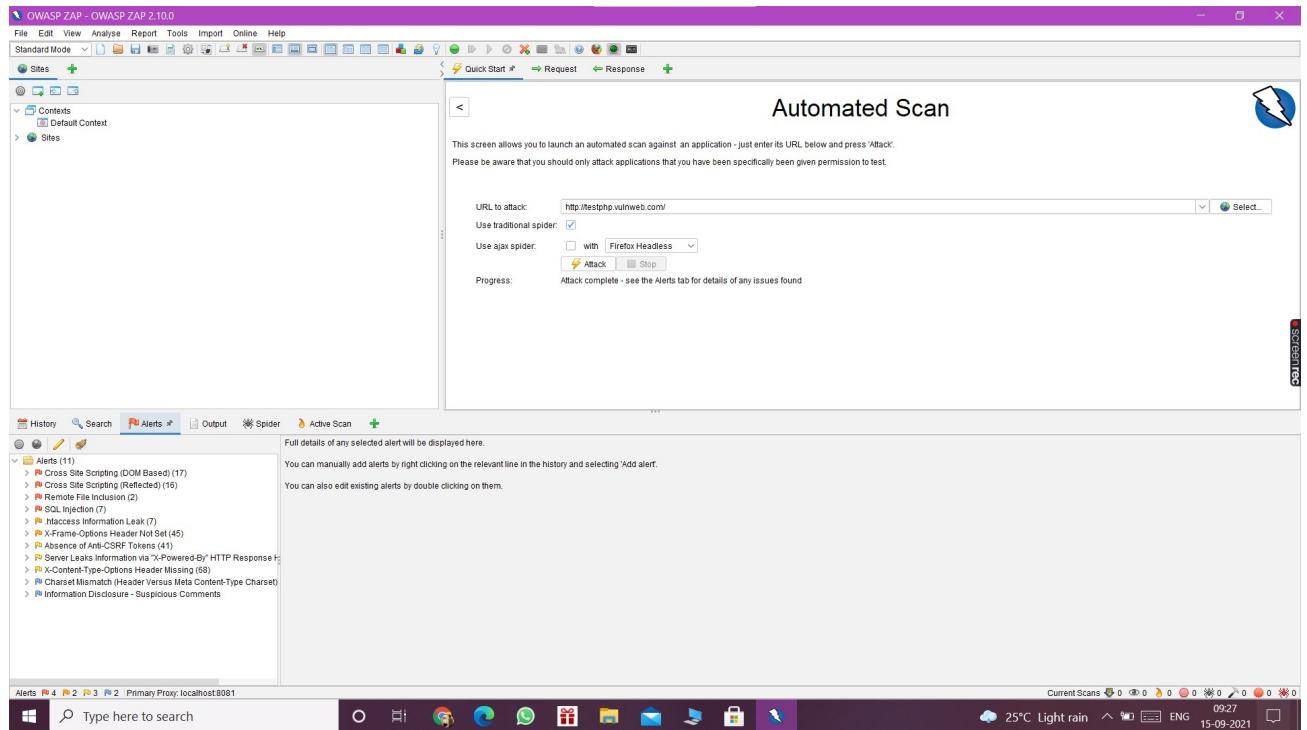
ID	Req. Timestamp	Resp. Timestamp	Method	URL	Code	Reason	RTT	Size Resp. Header	Size Resp. Body
629	15/09/21 9:12:39 AM	15/09/21 9:12:37 AM	GET	http://testphp.vulnweb.com/product.php?pic=c%34%2f-win... http://testphp.vulnweb.com/product.php?pic=%2f-%2f-%2f...	200	OK	244 ms	200 bytes	5,174 bytes
627	15/09/21 9:13:37 AM	15/09/21 9:13:37 AM	GET	http://testphp.vulnweb.com/	200	OK	219 ms	200 bytes	5,174 bytes
628	15/09/21 9:13:37 AM	15/09/21 9:13:37 AM	GET	http://testphp.vulnweb.com/index.php	200	OK	491 ms	200 bytes	5,174 bytes
629	15/09/21 9:13:37 AM	15/09/21 9:13:37 AM	GET	http://testphp.vulnweb.com/showimage.php?file=c%34%2f-wi...	200	OK	238 ms	186 bytes	200 bytes
630	15/09/21 9:13:37 AM	15/09/21 9:13:38 AM	GET	http://testphp.vulnweb.com/product.php?pic=c%34%2f-wi...	200	OK	200 ms	186 bytes	5,174 bytes
631	15/09/21 9:13:38 AM	15/09/21 9:13:38 AM	GET	http://testphp.vulnweb.com/showimage.php?file=%2f-%2f-%2f...	200	OK	218 ms	186 bytes	285 bytes
632	15/09/21 9:13:38 AM	15/09/21 9:13:38 AM	GET	http://testphp.vulnweb.com/product.php?pic=%2f-%2f-%2fpass...	200	OK	195 ms	200 bytes	5,174 bytes
633	15/09/21 9:13:38 AM	15/09/21 9:13:38 AM	GET	http://testphp.vulnweb.com/showimage.php?file=c%34%2fCV...	200	OK	203 ms	186 bytes	240 bytes
634	15/09/21 9:13:38 AM	15/09/21 9:13:38 AM	GET	http://testphp.vulnweb.com/showimage.php?file=%2f-%2f-%2f...	200	OK	359 ms	186 bytes	285 bytes
635	15/09/21 9:13:38 AM	15/09/21 9:13:38 AM	GET	http://testphp.vulnweb.com/product.php?pic=%2f-%2f-%2f...	200	OK	421 ms	200 bytes	5,174 bytes
636	15/09/21 9:13:38 AM	15/09/21 9:13:38 AM	GET	http://testphp.vulnweb.com/product.php?pic=c%34%2f-wi...	200	OK	219 ms	186 bytes	395 bytes
637	15/09/21 9:13:38 AM	15/09/21 9:13:39 AM	GET	http://testphp.vulnweb.com/product.php?pic=c%34%2f...	200	OK	234 ms	200 bytes	5,174 bytes
638	15/09/21 9:13:39 AM	15/09/21 9:13:39 AM	GET	http://testphp.vulnweb.com/showimage.php?file=%2f-%2f-%2f...	200	OK	195 ms	186 bytes	489 bytes
639	15/09/21 9:13:39 AM	15/09/21 9:13:39 AM	GET	http://testphp.vulnweb.com/product.php?pic=%2f-%2f-%2f...	200	OK	231 ms	200 bytes	5,174 bytes
640	15/09/21 9:13:39 AM	15/09/21 9:13:39 AM	GET	http://testphp.vulnweb.com/showimage.php?file=c%34%2f%	200	OK	223 ms	186 bytes	222 bytes
641	15/09/21 9:13:39 AM	15/09/21 9:13:39 AM	GET	http://testphp.vulnweb.com/product.php?pic=c%34%2f%	200	OK	216 ms	200 bytes	5,174 bytes

Screenshot 3:

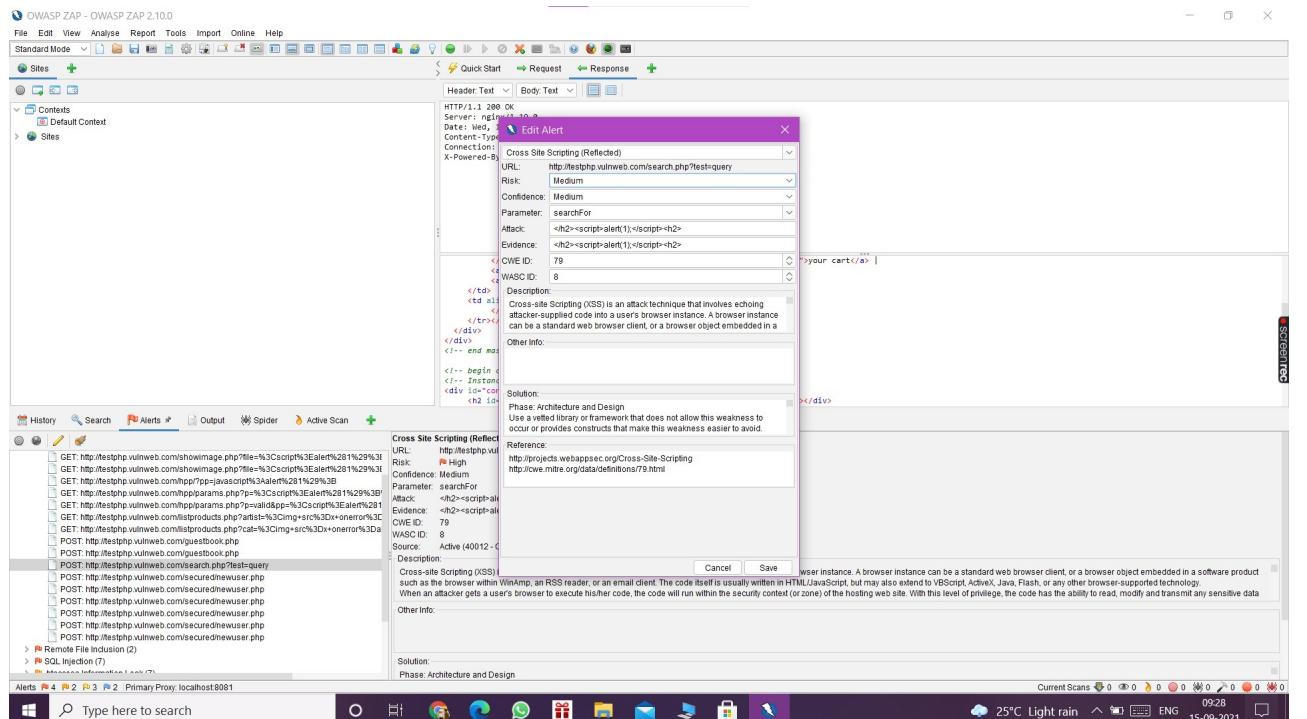
The screenshot shows the OWASP ZAP interface with the 'Active Scan' tab selected. The 'URL to attack' field contains <http://testphp.vulnweb.com/>. The 'Attack' button is no longer highlighted. Below the table, the status bar shows 'Attack complete - see the Alerts tab for details of any issues found'. The 'Current Scans: 0 Num Requests: 4726 New Alerts: 49 Export' status is also visible.

ID	Req. Timestamp	Resp. Timestamp	Method	URL	Code	Reason	RTT	Size Resp. Header	Size Resp. Body
4953	15/09/21 9:26:44 AM	15/09/21 9:26:44 AM	GET	http://testphp.vulnweb.com/index.php	301	Moved Permanently	352 ms	200 bytes	769 bytes
4954	15/09/21 9:26:45 AM	15/09/21 9:26:46 AM	GET	http://testphp.vulnweb.com/index.php	301	Moved Permanently	394 ms	200 bytes	169 bytes
4955	15/09/21 9:26:46 AM	15/09/21 9:26:46 AM	GET	http://testphp.vulnweb.com/images	301	Moved Permanently	234 ms	209 bytes	169 bytes
4956	15/09/21 9:26:47 AM	15/09/21 9:26:47 AM	GET	http://testphp.vulnweb.com/Mod_Rewrite_Shop	301	Moved Permanently	241 ms	219 bytes	169 bytes
4957	15/09/21 9:26:47 AM	15/09/21 9:26:48 AM	GET	http://testphp.vulnweb.com/Mod_Rewrite_Shop/Product-1	404	Not Found	315 ms	155 bytes	153 bytes
4958	15/09/21 9:26:48 AM	15/09/21 9:26:48 AM	GET	http://testphp.vulnweb.com/Mod_Rewrite_Shop/Product-2	404	Not Found	310 ms	155 bytes	153 bytes
4959	15/09/21 9:26:49 AM	15/09/21 9:26:49 AM	GET	http://testphp.vulnweb.com/Mod_Rewrite_Shop/Product-3	404	Not Found	316 ms	155 bytes	153 bytes
4960	15/09/21 9:26:49 AM	15/09/21 9:26:49 AM	GET	http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details	404	Not Found	320 ms	155 bytes	153 bytes
4961	15/09/21 9:26:49 AM	15/09/21 9:26:49 AM	GET	http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/color...	404	Not Found	318 ms	155 bytes	153 bytes
4962	15/09/21 9:26:49 AM	15/09/21 9:26:49 AM	GET	http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/color...	404	Not Found	317 ms	155 bytes	153 bytes
4963	15/09/21 9:26:49 AM	15/09/21 9:26:49 AM	GET	http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/color...	404	Not Found	317 ms	155 bytes	153 bytes
4964	15/09/21 9:26:50 AM	15/09/21 9:26:50 AM	GET	http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/color...	404	Not Found	319 ms	155 bytes	153 bytes
4965	15/09/21 9:26:50 AM	15/09/21 9:26:50 AM	GET	http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/color...	404	Not Found	317 ms	155 bytes	153 bytes
4966	15/09/21 9:26:50 AM	15/09/21 9:26:50 AM	GET	http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/color...	404	Not Found	317 ms	155 bytes	153 bytes
4967	15/09/21 9:26:50 AM	15/09/21 9:26:51 AM	GET	http://testphp.vulnweb.com/Mod_Rewrite_Shop/Images	301	Moved Permanently	310 ms	226 bytes	169 bytes
4968	15/09/21 9:26:52 AM	15/09/21 9:26:52 AM	GET	http://testphp.vulnweb.com/secured	301	Moved Permanently	220 ms	210 bytes	169 bytes

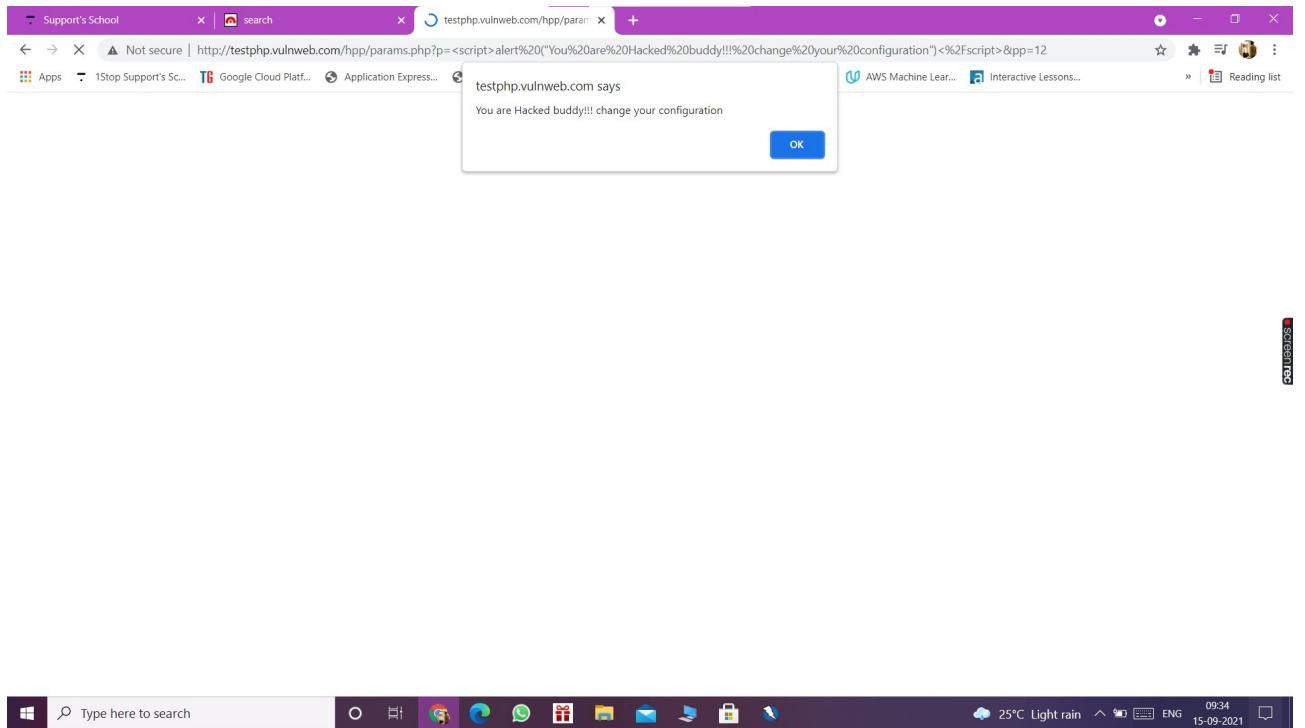
Screenshot 4:



Screenshot 5:



Screenshot 6:



Screenshot 7:

ZAP Scanning Report

Generated on Wed, 15 Sep 2021 09:35:12

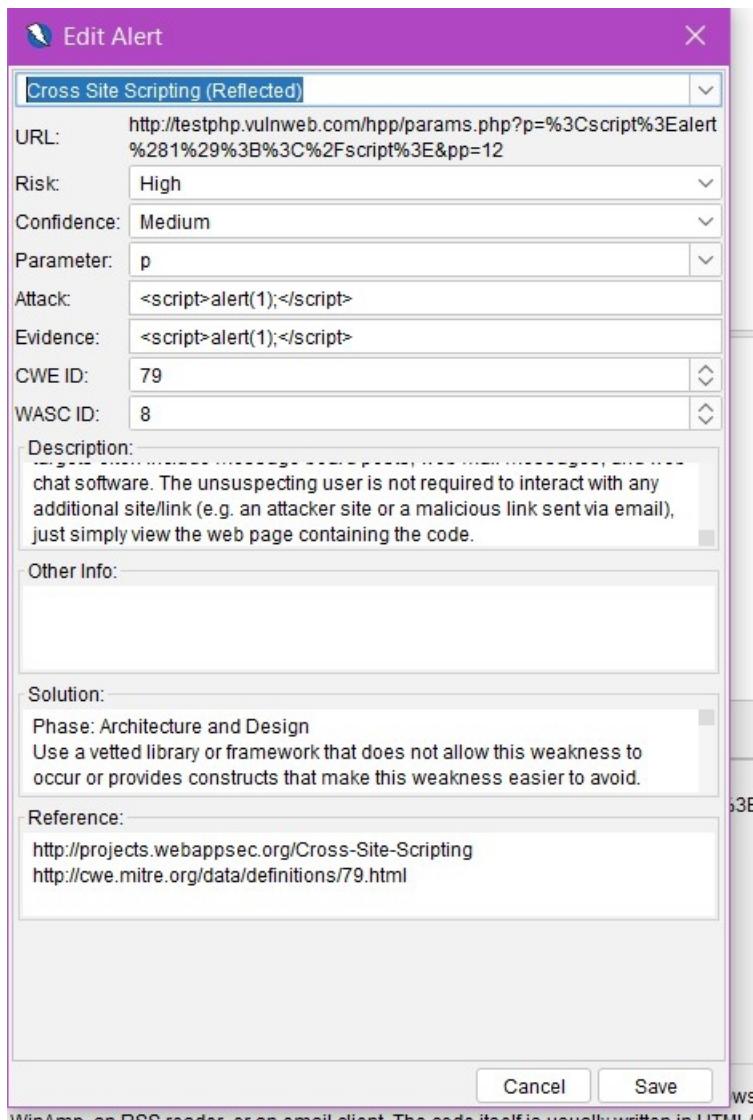
Summary of Alerts

Risk Level	Number of Alerts
High	5
Medium	3
Low	3
Informational	2

Alerts

Name	Risk Level	Number of Instances
Cross Site Scripting (DOM Based)	High	17
Cross Site Scripting (Reflected)	High	15
Remote File Inclusion	High	2
SQL Injection	High	7
.htaccess Information Leak	Medium	7
Cross Site Scripting (Reflected)	Medium	1
X-Frame-Options Header Not Set	Medium	45
Absence of Anti-CSRF Tokens	Low	41
Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)	Low	63
X-Content-Type-Options Header Missing	Low	68
Charset Mismatch (Header Versus Meta Content-Type Charset)	Informational	32
Information Disclosure - Suspicious Comments	Informational	1

Screenshot 8:



Screenshot 9:

