ZAP Scanning Report

Site: http://testphp.vulnweb.com

Generated on Sat, 2 Apr 2022 10:09:39

Summary of Alerts

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

Alerts

Name	Risk Level	Number of Instances
Cross Site Scripting (DOM Based)	High	17
Cross Site Scripting (Reflected)	High	16
SQL Injection	High	8
.htaccess Information Leak	Medium	7
Absence of Anti-CSRF Tokens	Medium	41
Missing Anti-clickjacking Header	Medium	45
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

Alert Detail

High	Cross Site Scripting (DOM Based)
	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.

Description	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.
URL	http://testphp.vulnweb.com/#jaVasCript:/*-/*`/*\`/*'/*"/**/(/* */oNcliCk=alert(5397))//%0D%0A%0d%0a//</stYle/</titLe/</teXtarEa/</scRipt/--!>\x3csVg/<sVg/oNloAd=alert(5397)//>\x3e
Method	GET
Attack	#jaVasCript:/*-/*`/*\`/*'/*"/**/(/* */oNcliCk=alert(5397))//%0D%0A%0d%0a//</stYle/</titLe/</teXtarEa/</scRipt/--!>\x3csVg/<sVg/oNloAd=alert(5397)//>\x3e
Evidence	
URL	http://testphp.vulnweb.com/artists.php#jaVasCript:/*-/*`/*\`/*'/*"/**/(/* */oNcliCk=alert(5397))//%0D%0A%0d%0a//</stYle/</titLe/</teXtarEa/</scRipt/--!>\x3csVg/<sVg/oNloAd=alert(5397)//>\x3e
Method	GET
Attack	#jaVasCript:/*-/*`/*\`/*'/*"/**/(/* */oNcliCk=alert(5397))//%0D%0A%0d%0a//</stYle/</titLe/</teXtarEa/</scRipt/--!>\x3csVg/<sVg/oNloAd=alert(5397)//>\x3e
Evidence	
URL	http://testphp.vulnweb.com/artists.php?artist=3#jaVasCript:/*-/*`/*\`/*'/*"/**/(/* */oNcliCk=alert(5397))//%0D%0A%0d%0a//</stYle/</titLe/</teXtarEa/</scRipt/--!>\x3csVg/<sVg/oNloAd=alert(5397)//>\x3e
Method	GET
Attack	#jaVasCript:/*-/*`/*\`/*'/*"/**/(/* */oNcliCk=alert(5397))//%0D%0A%0d%0a//</stYle/</titLe/</teXtarEa/</scRipt/--!>\x3csVg/<sVg/oNloAd=alert(5397)//>\x3e
Evidence	
URL	http://testphp.vulnweb.com/cart.php#jaVasCript:/*-/*`/*\`/*'/*"/**/(/* */oNcliCk=alert(5397))//%0D%0A%0d%0a//</stYle/</titLe/</teXtarEa/</scRipt/--!>\x3csVg/<sVg/oNloAd=alert(5397)//>\x3e
Method	GET
Attack	#jaVasCript:/*-/*`/*\`/*'/*"/**/(/* */oNcliCk=alert(5397))//%0D%0A%0d%0a//</stYle/</titLe/</teXtarEa/</scRipt/--!>\x3csVg/<sVg/oNloAd=alert(5397)//>\x3e
Evidence	
URL	http://testphp.vulnweb.com/categories.php#jaVasCript:/*-/*`/*\`/*'/*"/**/(/* */oNcliCk=alert(5397))//%0D%0A%0d%0a//</stYle/</titLe/</teXtarEa/</scRipt/--!>\x3csVg/<sVg/oNloAd=alert(5397)//>\x3e
Method	GET
Attack	#jaVasCript:/*-/*`/*\`/*'/*"/**/(/* */oNcliCk=alert(5397))//%0D%0A%0d%0a//</stYle/</titLe/</teXtarEa/</scRipt/--!>\x3csVg/<sVg/oNloAd=alert(5397)//>\x3e
Evidence	
URL	http://testphp.vulnweb.com/disclaimer.php#jaVasCript:/*-/*`/*\`/*'/*"/**/(/* */oNcliCk=alert(5397))//%0D%0A%0d%0a//</stYle/</titLe/</teXtarEa/</scRipt/--!>\x3csVg/<sVg

	/oNloAd=alert(5397)//>\x3e
Method	GET
Attack	#jaVasCript:/*-/*`/*\`/*'/*"/**/(/* */oNcliCk=alert(5397))//%0D%0A%0d%0a//</stYle/</titLe/</teXtarEa/</scRipt/--!>\x3csVg/<sVg/oNloAd=alert(5397)//>\x3e
Evidence	
URL	http://testphp.vulnweb.com/guestbook.php#jaVasCript:/*-/*`/*\`/*'/*"/**/(/* */oNcliCk=alert(5397))//%0D%0A%0d%0a//</stYle/</titLe/</teXtarEa/</scRipt/--!>\x3csVg/<sVg/oNloAd=alert(5397)//>\x3e
Method	GET
Attack	#jaVasCript:/*-/*`/*\`/*'/*"/**/(/* */oNcliCk=alert(5397))//%0D%0A%0d%0a//</stYle/</titLe/</teXtarEa/</scRipt/--!>\x3csVg/<sVg/oNloAd=alert(5397)//>\x3e
Evidence	
URL	http://testphp.vulnweb.com/index.php#jaVasCript:/*-/*`/*\`/*'/*"/**/(/* */oNcliCk=alert(5397))//%0D%0A%0d%0a//</stYle/</titLe/</teXtarEa/</scRipt/--!>\x3csVg/<sVg/oNloAd=alert(5397)//>\x3e
Method	GET
Attack	#jaVasCript:/*-/*`/*\`/*'/*"/**/(/* */oNcliCk=alert(5397))//%0D%0A%0d%0a//</stYle/</titLe/</teXtarEa/</scRipt/--!>\x3csVg/<sVg/oNloAd=alert(5397)//>\x3e
Evidence	
URL	http://testphp.vulnweb.com/listproducts.php?cat=4#jaVasCript:/*-/*`/*\`/*'/*"/**/(/* */oNcliCk=alert(5397))//%0D%0A%0d%0a//</stYle/</titLe/</teXtarEa/</scRipt/--!>\x3csVg/<sVg/oNloAd=alert(5397)//>\x3e
Method	GET
Attack	#jaVasCript:/*-/*`/*\`/*'/*"/**/(/* */oNcliCk=alert(5397))//%0D%0A%0d%0a//</stYle/</titLe/</teXtarEa/</scRipt/--!>\x3csVg/<sVg/oNloAd=alert(5397)//>\x3e
Evidence	
URL	http://testphp.vulnweb.com/login.php#jaVasCript:/*-/*`/*\`/*'/*"/**/(/* */oNcliCk=alert(5397))//%0D%0A%0d%0a//</stYle/</titLe/</teXtarEa/</scRipt/--!>\x3csVg/<sVg/oNloAd=alert(5397)//>\x3e
Method	GET
Attack	#jaVasCript:/*-/*`/*\`/*'/*"/**/(/* */oNcliCk=alert(5397))//%0D%0A%0d%0a//</stYle/</titLe/</teXtarEa/</scRipt/--!>\x3csVg/<sVg/oNloAd=alert(5397)//>\x3e
Evidence	
URL	http://testphp.vulnweb.com/product.php?pic=6#javascript:alert(5397)
Method	GET
Attack	#javascript:alert(5397)
Evidence	
URL	http://testphp.vulnweb.com/signup.php#jaVasCript:/*-/*`/*\`/*'/*"/**/(/* */oNcliCk=alert(5397))//%0D%0A%0d%0a//</stYle/</titLe/</teXtarEa/</scRipt/--!>\x3csVg/<sVg/oNloAd=alert(5397)//>\x3e
Method	GET
Attack	#jaVasCript:/*-/*`/*\`/*'/*"/**/(/* */oNcliCk=alert(5397))//%0D%0A%0d%0a//</stYle/</titLe/</teXtarEa/</scRipt/--!>\x3csVg/<sVg/oNloAd=alert(5397)//>\x3e
Evidence	
URL	http://testphp.vulnweb.com/userinfo.php?name=abc#<img src="random.gif" onerror=alert(5397)>
Method	GET
Attack	?name=abc#<img src="random.gif" onerror=alert(5397)>
Evidence	

URL	http://testphp.vulnweb.com?name=abc#<img src="random.gif" onerror=alert(5397)>
Method	GET
Attack	?name=abc#<img src="random.gif" onerror=alert(5397)>
Evidence	
URL	http://testphp.vulnweb.com/cart.php#jaVasCript:/*-/*`/*\`/*'/*"/**/(/* */oNcliCk=alert(5397))//%0D%0A%0d%0a//</stYle/</titLe/</teXtarEa/</scRipt/--!>\x3csVg/<sVg/oNloAd=alert(5397)//>\x3e
Method	POST
Attack	#jaVasCript:/*-/*`/*\`/*'/*"/**/(/* */oNcliCk=alert(5397))//%0D%0A%0d%0a//</stYle/</titLe/</teXtarEa/</scRipt/--!>\x3csVg/<sVg/oNloAd=alert(5397)//>\x3e
Evidence	
URL	http://testphp.vulnweb.com/guestbook.php#javascript:alert(5397)
Method	POST
Attack	#javascript:alert(5397)
Evidence	
URL	http://testphp.vulnweb.com/search.php?test=query#jaVasCript:/*-/*`/*\`/*'/*"/**/(/* */oNcliCk=alert(5397))//%0D%0A%0d%0a//</stYle/</titLe/</teXtarEa/</scRipt/--!>\x3csVg/<sVg/oNloAd=alert(5397)//>\x3e
Method	POST
Attack	#jaVasCript:/*-/*`/*\`/*'/*"/**/(/* */oNcliCk=alert(5397))//%0D%0A%0d%0a//</stYle/</titLe/</teXtarEa/</scRipt/--!>\x3csVg/<sVg/oNloAd=alert(5397)//>\x3e
Evidence	
Instances	17
	Phase: Architecture and Design Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid. Examples of libraries and frameworks that make it easier to generate properly encoded output include Microsoft's Anti-XSS library, the OWASP ESAPI Encoding module, and Apache Wicket. Phases: Implementation; Architecture and Design Understand the context in which your data will be used and the encoding that will be expected. This is especially important when transmitting data between different components, or when generating outputs that can contain multiple encodings at the same time, such as web pages or multi-part mail messages. Study all expected communication protocols and data representations to determine the required encoding strategies. For any data that will be output to another web page, especially any data that was received from external inputs, use the appropriate encoding on all non-alphanumeric characters. Consult the XSS Prevention Cheat Sheet for more details on the types of encoding and escaping that are needed. Phase: Architecture and Design For any security checks that are performed on the client side, ensure that these checks are duplicated on the server side, in order to avoid CWE-602. Attackers can bypass the client-side checks by modifying values after the checks have been performed, or by changing the client to remove the client-side checks entirely. Then, these modified values would be submitted to the server.

Solution	If available, use structured mechanisms that automatically enforce the separation between data and code. These mechanisms may be able to provide the relevant quoting, encoding, and validation automatically, instead of relying on the developer to provide this capability at every point where output is generated. Phase: Implementation For every web page that is generated, use and specify a character encoding such as ISO-8859-1 or UTF-8. When an encoding is not specified, the web browser may choose a different encoding by guessing which encoding is actually being used by the web page. This can cause the web browser to treat certain sequences as special, opening up the client to subtle XSS attacks. See CWE-116 for more mitigations related to encoding/escaping. To help mitigate XSS attacks against the user's session cookie, set the session cookie to be HttpOnly. In browsers that support the HttpOnly feature (such as more recent versions of Internet Explorer and Firefox), this attribute can prevent the user's session cookie from being accessible to malicious client-side scripts that use document.cookie. This is not a complete solution, since HttpOnly is not supported by all browsers. More importantly, XMLHTTPRequest and other powerful browser technologies provide read access to HTTP headers, including the Set-Cookie header in which the HttpOnly flag is set. Assume all input is malicious. Use an "accept known good" input validation strategy, i.e., use an allow list of acceptable inputs that strictly conform to specifications. Reject any input that does not strictly conform to specifications, or transform it into something that does. Do not rely exclusively on looking for malicious or malformed inputs (i.e., do not rely on a deny list). However, deny lists can be useful for detecting potential attacks or determining which inputs are so malformed that they should be rejected outright. When performing input validation, consider all potentially relevant properties, including length, type of input, the full range of acceptable values, missing or extra inputs, syntax, consistency across related fields, and conformance to business rules. As an example of business rule logic, "boat" may be syntactically valid because it only contains alphanumeric characters, but it is not valid if you are expecting colors such as "red" or "blue." Ensure that you perform input validation at well-defined interfaces within the application. This will help protect the application even if a component is reused or moved elsewhere.
Reference	http://projects.webappsec.org/Cross-Site-Scripting http://cwe.mitre.org/data/definitions/79.html
CWE Id	79
WASC Id	8
Plugin Id	40026

High	**Cross Site Scripting (Reflected)**
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.
URL	http://testphp.vulnweb.com/showimage.php?file=%3CscrIpt%3Ealert%281%29%3B%3C%2FscRipt%3E
Method	GET
Attack	<scrIpt>alert(1);</scRipt>
Evidence	<scrIpt>alert(1);</scRipt>
URL	http://testphp.vulnweb.com/showimage.php?file=%3CscrIpt%3Ealert%281%29%3B%3C%2FscRipt%3E&size=160
Method	GET
Attack	<scrIpt>alert(1);</scRipt>
Evidence	<scrIpt>alert(1);</scRipt>
URL	http://testphp.vulnweb.com/hpp/?pp=javascript%3Aalert%281%29%3B
Method	GET
Attack	javascript:alert(1);
Evidence	javascript:alert(1);
URL	http://testphp.vulnweb.com/hpp/params.php?p=%3CscrIpt%3Ealert%281%29%3B%3C%2FscRipt%3E&pp=12
Method	GET
Attack	<scrIpt>alert(1);</scRipt>
Evidence	<scrIpt>alert(1);</scRipt>
URL	http://testphp.vulnweb.com/hpp/params.php?p=valid&pp=%3CscrIpt%3Ealert%281%29%3B%3C%2FscRipt%3E
Method	GET
Attack	<scrIpt>alert(1);</scRipt>
Evidence	<scrIpt>alert(1);</scRipt>
URL	http://testphp.vulnweb.com/listproducts.php?artist=%3Cimg+src%3Dx+onerror%3Dprompt%28%29%3E
Method	GET
Attack	<img src=x onerror=prompt()>
Evidence	<img src=x onerror=prompt()>
URL	http://testphp.vulnweb.com/listproducts.php?cat=%3Cimg+src%3Dx+onerror%3Dprompt%28%29%3E
Method	GET
Attack	<img src=x onerror=prompt()>
Evidence	<img src=x onerror=prompt()>
URL	http://testphp.vulnweb.com/guestbook.php
Method	POST

Attack	<scrIpt>alert(1);</scRipt>
Evidence	<scrIpt>alert(1);</scRipt><strong>
URL	http://testphp.vulnweb.com/guestbook.php
Method	POST
Attack	</td><scrIpt>alert(1);</scRipt><td>
Evidence	</td><scrIpt>alert(1);</scRipt><td>
URL	http://testphp.vulnweb.com/search.php?test=query
Method	POST
Attack	<scrIpt>alert(1);</scRipt>
Evidence	<scrIpt>alert(1);</scRipt>
URL	http://testphp.vulnweb.com/secured/newuser.php
Method	POST
Attack	<scrIpt>alert(1);</scRipt>
Evidence	<scrIpt>alert(1);</scRipt>
URL	http://testphp.vulnweb.com/secured/newuser.php
Method	POST
Attack	<scrIpt>alert(1);</scRipt>
Evidence	<scrIpt>alert(1);</scRipt>
URL	http://testphp.vulnweb.com/secured/newuser.php
Method	POST
Attack	<scrIpt>alert(1);</scRipt>
Evidence	<scrIpt>alert(1);</scRipt>
URL	http://testphp.vulnweb.com/secured/newuser.php
Method	POST
Attack	<scrIpt>alert(1);</scRipt>
Evidence	<scrIpt>alert(1);</scRipt>
URL	http://testphp.vulnweb.com/secured/newuser.php
Method	POST
Attack	<scrIpt>alert(1);</scRipt>
Evidence	<scrIpt>alert(1);</scRipt>
URL	http://testphp.vulnweb.com/secured/newuser.php
Method	POST
Attack	<scrIpt>alert(1);</scRipt>
Evidence	<scrIpt>alert(1);</scRipt>
Instances	16
	Phase: Architecture and Design Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid. Examples of libraries and frameworks that make it easier to generate properly encoded output include Microsoft's Anti-XSS library, the OWASP ESAPI Encoding module, and Apache Wicket.

Solution	Phases: Implementation; Architecture and Design Understand the context in which your data will be used and the encoding that will be expected. This is especially important when transmitting data between different components, or when generating outputs that can contain multiple encodings at the same time, such as web pages or multi-part mail messages. Study all expected communication protocols and data representations to determine the required encoding strategies. For any data that will be output to another web page, especially any data that was received from external inputs, use the appropriate encoding on all non-alphanumeric characters. Consult the XSS Prevention Cheat Sheet for more details on the types of encoding and escaping that are needed. Phase: Architecture and Design For any security checks that are performed on the client side, ensure that these checks are duplicated on the server side, in order to avoid CWE-602. Attackers can bypass the client-side checks by modifying values after the checks have been performed, or by changing the client to remove the client-side checks entirely. Then, these modified values would be submitted to the server. If available, use structured mechanisms that automatically enforce the separation between data and code. These mechanisms may be able to provide the relevant quoting, encoding, and validation automatically, instead of relying on the developer to provide this capability at every point where output is generated. Phase: Implementation For every web page that is generated, use and specify a character encoding such as ISO-8859-1 or UTF-8. When an encoding is not specified, the web browser may choose a different encoding by guessing which encoding is actually being used by the web page. This can cause the web browser to treat certain sequences as special, opening up the client to subtle XSS attacks. See CWE-116 for more mitigations related to encoding/escaping. To help mitigate XSS attacks against the user's session cookie, set the session cookie to be HttpOnly. In browsers that support the HttpOnly feature (such as more recent versions of Internet Explorer and Firefox), this attribute can prevent the user's session cookie from being accessible to malicious client-side scripts that use document.cookie. This is not a complete solution, since HttpOnly is not supported by all browsers. More importantly, XMLHTTPRequest and other powerful browser technologies provide read access to HTTP headers, including the Set-Cookie header in which the HttpOnly flag is set. Assume all input is malicious. Use an "accept known good" input validation strategy, i.e., use an allow list of acceptable inputs that strictly conform to specifications. Reject any input that does not strictly conform to specifications, or transform it into something that does. Do not rely exclusively on looking for malicious or malformed inputs (i.e., do not rely on a deny list). However, deny lists can be useful for detecting potential attacks or determining which inputs are so malformed that they should be rejected outright. When performing input validation, consider all potentially relevant properties, including length, type of input, the full range of acceptable values, missing or extra inputs, syntax, consistency across related fields, and conformance to business rules. As an example of business rule logic, "boat" may be syntactically valid because it only contains alphanumeric characters, but it is not valid if you are expecting colors such as "red" or "blue." Ensure that you perform input validation at well-defined interfaces within the application. This will help protect the application even if a component is reused or moved elsewhere.
Reference	http://projects.webappsec.org/Cross-Site-Scripting http://cwe.mitre.org/data/definitions/79.html
CWE Id	79
WASC Id	8
Plugin Id	40012

High	**SQL Injection**

Description	SQL injection may be possible.
URL	http://testphp.vulnweb.com/artists.php?artist=5-2
Method	GET
Attack	5-2
Evidence	
URL	http://testphp.vulnweb.com/listproducts.php?artist=3+AND+1%3D1+--+
Method	GET
Attack	3 OR 1=1 --
Evidence	
URL	http://testphp.vulnweb.com/listproducts.php?cat=4+AND+1%3D1+--+
Method	GET
Attack	4 OR 1=1 --
Evidence	
URL	http://testphp.vulnweb.com/product.php?pic=8-2
Method	GET
Attack	8-2
Evidence	
URL	http://testphp.vulnweb.com/secured/newuser.php
Method	POST
Attack	ZAP AND 1=1 --
Evidence	
URL	http://testphp.vulnweb.com/secured/newuser.php
Method	POST
Attack	ZAP' OR '1'='1' --
Evidence	
URL	http://testphp.vulnweb.com/userinfo.php
Method	POST
Attack	ZAP AND 1=1 --
Evidence	
URL	http://testphp.vulnweb.com/userinfo.php
Method	POST
Attack	ZAP' OR '1'='1' --
Evidence	
Instances	8
	Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them.

Solution	Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply an 'allow list' of allowed characters, or a 'deny list' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application.
Reference	https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html
CWE Id	89
WASC Id	19
Plugin Id	40018

Medium	**.htaccess Information Leak**
Description	htaccess files can be used to alter the configuration of the Apache Web Server software to enable/disable additional functionality and features that the Apache Web Server software has to offer.
URL	http://testphp.vulnweb.com/Mod_Rewrite_Shop/.htaccess
Method	GET
Attack	
Evidence	HTTP/1.1 200 OK
URL	http://testphp.vulnweb.com/Mod_Rewrite_Shop/BuyProduct-1/.htaccess
Method	GET
Attack	
Evidence	HTTP/1.1 200 OK
URL	http://testphp.vulnweb.com/Mod_Rewrite_Shop/BuyProduct-2/.htaccess
Method	GET
Attack	
Evidence	HTTP/1.1 200 OK
URL	http://testphp.vulnweb.com/Mod_Rewrite_Shop/BuyProduct-3/.htaccess
Method	GET
Attack	
Evidence	HTTP/1.1 200 OK
URL	http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/color-printer/3/.htaccess
Method	GET
Attack	
Evidence	HTTP/1.1 200 OK
URL	http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/network-attached-storage-dlink/1/.htaccess
Method	GET

Attack	
Evidence	HTTP/1.1 200 OK
URL	http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/web-camera-a4tech/2/.htaccess
Method	GET
Attack	
Evidence	HTTP/1.1 200 OK
Instances	7
Solution	Ensure the .htaccess file is not accessible.
Reference	http://www.htaccess-guide.com/
CWE Id	94
WASC Id	14
Plugin Id	40032

Medium	**Absence of Anti-CSRF Tokens**
Description	No Anti-CSRF tokens were found in a HTML submission form. A cross-site request forgery is an attack that involves forcing a victim to send an HTTP request to a target destination without their knowledge or intent in order to perform an action as the victim. The underlying cause is application functionality using predictable URL /form actions in a repeatable way. The nature of the attack is that CSRF exploits the trust that a web site has for a user. By contrast, cross-site scripting (XSS) exploits the trust that a user has for a web site. Like XSS, CSRF attacks are not necessarily cross-site, but they can be. Cross-site request forgery is also known as CSRF, XSRF, one-click attack, session riding, confused deputy, and sea surf. CSRF attacks are effective in a number of situations, including: * The victim has an active session on the target site. * The victim is authenticated via HTTP auth on the target site. * The victim is on the same local network as the target site. CSRF has primarily been used to perform an action against a target site using the victim's privileges, but recent techniques have been discovered to disclose information by gaining access to the response. The risk of information disclosure is dramatically increased when the target site is vulnerable to XSS, because XSS can be used as a platform for CSRF, allowing the attack to operate within the bounds of the same-origin policy.
URL	http://testphp.vulnweb.com
Method	GET
Attack	
Evidence	<form action="search.php?test=query" method="post">
URL	http://testphp.vulnweb.com/
Method	GET
Attack	
Evidence	<form action="search.php?test=query" method="post">
URL	http://testphp.vulnweb.com/artists.php
Method	GET
Attack	
Evidence	<form action="search.php?test=query" method="post">
URL	http://testphp.vulnweb.com/artists.php?artist=1

Method	GET
Attack	
Evidence	<form action="search.php?test=query" method="post">
URL	http://testphp.vulnweb.com/artists.php?artist=2
Method	GET
Attack	
Evidence	<form action="search.php?test=query" method="post">
URL	http://testphp.vulnweb.com/artists.php?artist=3
Method	GET
Attack	
Evidence	<form action="search.php?test=query" method="post">
URL	http://testphp.vulnweb.com/cart.php
Method	GET
Attack	
Evidence	<form action="search.php?test=query" method="post">
URL	http://testphp.vulnweb.com/categories.php
Method	GET
Attack	
Evidence	<form action="search.php?test=query" method="post">
URL	http://testphp.vulnweb.com/disclaimer.php
Method	GET
Attack	
Evidence	<form action="search.php?test=query" method="post">
URL	http://testphp.vulnweb.com/guestbook.php
Method	GET
Attack	
Evidence	<form action="" method="post" name="faddentry">
URL	http://testphp.vulnweb.com/guestbook.php
Method	GET
Attack	
Evidence	<form action="search.php?test=query" method="post">
URL	http://testphp.vulnweb.com/index.php
Method	GET
Attack	
Evidence	<form action="search.php?test=query" method="post">
URL	http://testphp.vulnweb.com/listproducts.php?artist=1
Method	GET
Attack	
Evidence	<form action="search.php?test=query" method="post">
URL	http://testphp.vulnweb.com/listproducts.php?artist=2
Method	GET

Attack	
Evidence	<form action="search.php?test=query" method="post">
URL	http://testphp.vulnweb.com/listproducts.php?artist=3
Method	GET
Attack	
Evidence	<form action="search.php?test=query" method="post">
URL	http://testphp.vulnweb.com/listproducts.php?cat=1
Method	GET
Attack	
Evidence	<form action="search.php?test=query" method="post">
URL	http://testphp.vulnweb.com/listproducts.php?cat=2
Method	GET
Attack	
Evidence	<form action="search.php?test=query" method="post">
URL	http://testphp.vulnweb.com/listproducts.php?cat=3
Method	GET
Attack	
Evidence	<form action="search.php?test=query" method="post">
URL	http://testphp.vulnweb.com/listproducts.php?cat=4
Method	GET
Attack	
Evidence	<form action="search.php?test=query" method="post">
URL	http://testphp.vulnweb.com/login.php
Method	GET
Attack	
Evidence	<form name="loginform" method="post" action="userinfo.php">
URL	http://testphp.vulnweb.com/login.php
Method	GET
Attack	
Evidence	<form action="search.php?test=query" method="post">
URL	http://testphp.vulnweb.com/product.php?pic=1
Method	GET
Attack	
Evidence	<form name='f_addcart' method='POST' action='cart.php'>
URL	http://testphp.vulnweb.com/product.php?pic=1
Method	GET
Attack	
Evidence	<form action="search.php?test=query" method="post">
URL	http://testphp.vulnweb.com/product.php?pic=2
Method	GET
Attack	

Evidence	<form name='f_addcart' method='POST' action='cart.php'>
URL	http://testphp.vulnweb.com/product.php?pic=2
Method	GET
Attack	
Evidence	<form action="search.php?test=query" method="post">
URL	http://testphp.vulnweb.com/product.php?pic=3
Method	GET
Attack	
Evidence	<form name='f_addcart' method='POST' action='cart.php'>
URL	http://testphp.vulnweb.com/product.php?pic=3
Method	GET
Attack	
Evidence	<form action="search.php?test=query" method="post">
URL	http://testphp.vulnweb.com/product.php?pic=4
Method	GET
Attack	
Evidence	<form name='f_addcart' method='POST' action='cart.php'>
URL	http://testphp.vulnweb.com/product.php?pic=4
Method	GET
Attack	
Evidence	<form action="search.php?test=query" method="post">
URL	http://testphp.vulnweb.com/product.php?pic=5
Method	GET
Attack	
Evidence	<form name='f_addcart' method='POST' action='cart.php'>
URL	http://testphp.vulnweb.com/product.php?pic=5
Method	GET
Attack	
Evidence	<form action="search.php?test=query" method="post">
URL	http://testphp.vulnweb.com/product.php?pic=6
Method	GET
Attack	
Evidence	<form name='f_addcart' method='POST' action='cart.php'>
URL	http://testphp.vulnweb.com/product.php?pic=6
Method	GET
Attack	
Evidence	<form action="search.php?test=query" method="post">
URL	http://testphp.vulnweb.com/product.php?pic=7
Method	GET
Attack	
Evidence	<form name='f_addcart' method='POST' action='cart.php'>

URL	http://testphp.vulnweb.com/product.php?pic=7
Method	GET
Attack	
Evidence	<form action="search.php?test=query" method="post">
URL	http://testphp.vulnweb.com/signup.php
Method	GET
Attack	
Evidence	<form name="form1" method="post" action="/secured/newuser.php">
URL	http://testphp.vulnweb.com/signup.php
Method	GET
Attack	
Evidence	<form action="search.php?test=query" method="post">
URL	http://testphp.vulnweb.com/cart.php
Method	POST
Attack	
Evidence	<form action="search.php?test=query" method="post">
URL	http://testphp.vulnweb.com/guestbook.php
Method	POST
Attack	
Evidence	<form action="" method="post" name="faddentry">
URL	http://testphp.vulnweb.com/guestbook.php
Method	POST
Attack	
Evidence	<form action="search.php?test=query" method="post">
URL	http://testphp.vulnweb.com/search.php?test=query
Method	POST
Attack	
Evidence	<form action="search.php?test=query" method="post">
Instances	41
Solution	Phase: Architecture and Design Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid. For example, use anti-CSRF packages such as the OWASP CSRFGuard. Phase: Implementation Ensure that your application is free of cross-site scripting issues, because most CSRF defenses can be bypassed using attacker-controlled script. Phase: Architecture and Design Generate a unique nonce for each form, place the nonce into the form, and verify the nonce upon receipt of the form. Be sure that the nonce is not predictable (CWE-330). Note that this can be bypassed using XSS.

	Identify especially dangerous operations. When the user performs a dangerous operation, send a separate confirmation request to ensure that the user intended to perform that operation. Note that this can be bypassed using XSS. Use the ESAPI Session Management control. This control includes a component for CSRF. Do not use the GET method for any request that triggers a state change. Phase: Implementation Check the HTTP Referer header to see if the request originated from an expected page. This could break legitimate functionality, because users or proxies may have disabled sending the Referer for privacy reasons.
Reference	http://projects.webappsec.org/Cross-Site-Request-Forgery http://cwe.mitre.org/data/definitions/352.html
CWE Id	352
WASC Id	9
Plugin Id	10202

Medium	**Missing Anti-clickjacking Header**
Description	The response does not include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options to protect against 'ClickJacking' attacks.
URL	http://testphp.vulnweb.com
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/AJAX/index.php
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/artists.php
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/artists.php?artist=1
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/artists.php?artist=2
Method	GET
Attack	

Evidence	
URL	http://testphp.vulnweb.com/artists.php?artist=3
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/cart.php
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/categories.php
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/disclaimer.php
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/guestbook.php
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/hpp/
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/hpp/?pp=12
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/hpp/params.php?p=valid&pp=12
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/index.php
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/listproducts.php?artist=1
Method	GET
Attack	
Evidence	

URL	http://testphp.vulnweb.com/listproducts.php?artist=2
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/listproducts.php?artist=3
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/listproducts.php?cat=1
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/listproducts.php?cat=2
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/listproducts.php?cat=3
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/listproducts.php?cat=4
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/login.php
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/Mod_Rewrite_Shop/
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/Mod_Rewrite_Shop/BuyProduct-1/
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/Mod_Rewrite_Shop/BuyProduct-2/
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/Mod_Rewrite_Shop/BuyProduct-3/

Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/color-printer/3/
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/network-attached-storage-dlink/1/
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/web-camera-a4tech/2/
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/Mod_Rewrite_Shop/RateProduct-1.html
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/Mod_Rewrite_Shop/RateProduct-2.html
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/Mod_Rewrite_Shop/RateProduct-3.html
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/product.php?pic=1
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/product.php?pic=2
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/product.php?pic=3
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/product.php?pic=4
Method	GET

Attack	
Evidence	
URL	http://testphp.vulnweb.com/product.php?pic=5
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/product.php?pic=6
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/product.php?pic=7
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/signup.php
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/cart.php
Method	POST
Attack	
Evidence	
URL	http://testphp.vulnweb.com/guestbook.php
Method	POST
Attack	
Evidence	
URL	http://testphp.vulnweb.com/search.php?test=query
Method	POST
Attack	
Evidence	
URL	http://testphp.vulnweb.com/secured/newuser.php
Method	POST
Attack	
Evidence	
Instances	45
Solution	Modern Web browsers support the Content-Security-Policy and X-Frame-Options HTTP headers. Ensure one of them is set on all web pages returned by your site/app. If you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. Alternatively consider implementing Content Security Policy's "frame-ancestors" directive.
Reference	https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options

CWE Id	1021
WASC Id	15
Plugin Id	10020

Low	**Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)**
Description	The web/application server is leaking information via one or more "X-Powered-By" HTTP response headers. Access to such information may facilitate attackers identifying other frameworks/components your web application is reliant upon and the vulnerabilities such components may be subject to.
URL	http://testphp.vulnweb.com
Method	GET
Attack	
Evidence	X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
URL	http://testphp.vulnweb.com/
Method	GET
Attack	
Evidence	X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
URL	http://testphp.vulnweb.com/AJAX/index.php
Method	GET
Attack	
Evidence	X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
URL	http://testphp.vulnweb.com/artists.php
Method	GET
Attack	
Evidence	X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
URL	http://testphp.vulnweb.com/artists.php?artist=1
Method	GET
Attack	
Evidence	X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
URL	http://testphp.vulnweb.com/artists.php?artist=2
Method	GET
Attack	
Evidence	X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
URL	http://testphp.vulnweb.com/artists.php?artist=3
Method	GET
Attack	
Evidence	X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
URL	http://testphp.vulnweb.com/cart.php
Method	GET
Attack	
Evidence	X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
URL	http://testphp.vulnweb.com/categories.php
Method	GET

Attack	
Evidence	X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
URL	http://testphp.vulnweb.com/disclaimer.php
Method	GET
Attack	
Evidence	X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
URL	http://testphp.vulnweb.com/guestbook.php
Method	GET
Attack	
Evidence	X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
URL	http://testphp.vulnweb.com/hpp/
Method	GET
Attack	
Evidence	X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
URL	http://testphp.vulnweb.com/hpp/?pp=12
Method	GET
Attack	
Evidence	X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
URL	http://testphp.vulnweb.com/hpp/params.php?p=valid&pp=12
Method	GET
Attack	
Evidence	X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
URL	http://testphp.vulnweb.com/index.php
Method	GET
Attack	
Evidence	X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
URL	http://testphp.vulnweb.com/listproducts.php?artist=1
Method	GET
Attack	
Evidence	X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
URL	http://testphp.vulnweb.com/listproducts.php?artist=2
Method	GET
Attack	
Evidence	X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
URL	http://testphp.vulnweb.com/listproducts.php?artist=3
Method	GET
Attack	
Evidence	X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
URL	http://testphp.vulnweb.com/listproducts.php?cat=1
Method	GET
Attack	

Evidence	X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
URL	http://testphp.vulnweb.com/listproducts.php?cat=2
Method	GET
Attack	
Evidence	X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
URL	http://testphp.vulnweb.com/listproducts.php?cat=3
Method	GET
Attack	
Evidence	X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
URL	http://testphp.vulnweb.com/listproducts.php?cat=4
Method	GET
Attack	
Evidence	X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
URL	http://testphp.vulnweb.com/login.php
Method	GET
Attack	
Evidence	X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
URL	http://testphp.vulnweb.com/Mod_Rewrite_Shop/
Method	GET
Attack	
Evidence	X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
URL	http://testphp.vulnweb.com/Mod_Rewrite_Shop/BuyProduct-1/
Method	GET
Attack	
Evidence	X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
URL	http://testphp.vulnweb.com/Mod_Rewrite_Shop/BuyProduct-2/
Method	GET
Attack	
Evidence	X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
URL	http://testphp.vulnweb.com/Mod_Rewrite_Shop/BuyProduct-3/
Method	GET
Attack	
Evidence	X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
URL	http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/color-printer/3/
Method	GET
Attack	
Evidence	X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
URL	http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/network-attached-storage-dlink/1/
Method	GET
Attack	
Evidence	X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1

URL	http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/web-camera-a4tech/2/
Method	GET
Attack	
Evidence	X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
URL	http://testphp.vulnweb.com/Mod_Rewrite_Shop/RateProduct-1.html
Method	GET
Attack	
Evidence	X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
URL	http://testphp.vulnweb.com/Mod_Rewrite_Shop/RateProduct-2.html
Method	GET
Attack	
Evidence	X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
URL	http://testphp.vulnweb.com/Mod_Rewrite_Shop/RateProduct-3.html
Method	GET
Attack	
Evidence	X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
URL	http://testphp.vulnweb.com/privacy.php
Method	GET
Attack	
Evidence	X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
URL	http://testphp.vulnweb.com/product.php?pic=1
Method	GET
Attack	
Evidence	X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
URL	http://testphp.vulnweb.com/product.php?pic=2
Method	GET
Attack	
Evidence	X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
URL	http://testphp.vulnweb.com/product.php?pic=3
Method	GET
Attack	
Evidence	X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
URL	http://testphp.vulnweb.com/product.php?pic=4
Method	GET
Attack	
Evidence	X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
URL	http://testphp.vulnweb.com/product.php?pic=5
Method	GET
Attack	
Evidence	X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
URL	http://testphp.vulnweb.com/product.php?pic=6

Method	GET
Attack	
Evidence	X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
URL	http://testphp.vulnweb.com/product.php?pic=7
Method	GET
Attack	
Evidence	X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
URL	http://testphp.vulnweb.com/showimage.php?file='%20+%20pict.item(0).firstChild.nodeValue%20+%20'
Method	GET
Attack	
Evidence	X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
URL	http://testphp.vulnweb.com/showimage.php?file=./pictures/1.jpg
Method	GET
Attack	
Evidence	X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
URL	http://testphp.vulnweb.com/showimage.php?file=./pictures/1.jpg&size=160
Method	GET
Attack	
Evidence	X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
URL	http://testphp.vulnweb.com/showimage.php?file=./pictures/2.jpg
Method	GET
Attack	
Evidence	X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
URL	http://testphp.vulnweb.com/showimage.php?file=./pictures/2.jpg&size=160
Method	GET
Attack	
Evidence	X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
URL	http://testphp.vulnweb.com/showimage.php?file=./pictures/3.jpg
Method	GET
Attack	
Evidence	X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
URL	http://testphp.vulnweb.com/showimage.php?file=./pictures/3.jpg&size=160
Method	GET
Attack	
Evidence	X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
URL	http://testphp.vulnweb.com/showimage.php?file=./pictures/4.jpg
Method	GET
Attack	
Evidence	X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
URL	http://testphp.vulnweb.com/showimage.php?file=./pictures/4.jpg&size=160

Method	GET
Attack	
Evidence	X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
URL	http://testphp.vulnweb.com/showimage.php?file=./pictures/5.jpg
Method	GET
Attack	
Evidence	X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
URL	http://testphp.vulnweb.com/showimage.php?file=./pictures/5.jpg&size=160
Method	GET
Attack	
Evidence	X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
URL	http://testphp.vulnweb.com/showimage.php?file=./pictures/6.jpg
Method	GET
Attack	
Evidence	X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
URL	http://testphp.vulnweb.com/showimage.php?file=./pictures/6.jpg&size=160
Method	GET
Attack	
Evidence	X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
URL	http://testphp.vulnweb.com/showimage.php?file=./pictures/7.jpg
Method	GET
Attack	
Evidence	X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
URL	http://testphp.vulnweb.com/showimage.php?file=./pictures/7.jpg&size=160
Method	GET
Attack	
Evidence	X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
URL	http://testphp.vulnweb.com/signup.php
Method	GET
Attack	
Evidence	X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
URL	http://testphp.vulnweb.com/userinfo.php
Method	GET
Attack	
Evidence	X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
URL	http://testphp.vulnweb.com/cart.php
Method	POST
Attack	
Evidence	X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
URL	http://testphp.vulnweb.com/guestbook.php
Method	POST

Attack	
Evidence	X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
URL	http://testphp.vulnweb.com/search.php?test=query
Method	POST
Attack	
Evidence	X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
URL	http://testphp.vulnweb.com/secured/newuser.php
Method	POST
Attack	
Evidence	X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
URL	http://testphp.vulnweb.com/userinfo.php
Method	POST
Attack	
Evidence	X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
Instances	63
Solution	Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.
Reference	http://blogs.msdn.com/b/varunm/archive/2013/04/23/remove-unwanted-http-response-headers.aspx http://www.troyhunt.com/2012/02/shhh-dont-let-your-response-headers.html
CWE Id	200
WASC Id	13
Plugin Id	10037

Low	**X-Content-Type-Options Header Missing**
Description	The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.
URL	http://testphp.vulnweb.com
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/AJAX/index.php
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/AJAX/styles.css
Method	GET

Attack	
Evidence	
URL	http://testphp.vulnweb.com/artists.php
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/artists.php?artist=1
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/artists.php?artist=2
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/artists.php?artist=3
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/cart.php
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/categories.php
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/disclaimer.php
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/guestbook.php
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/hpp/
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/hpp/?pp=12
Method	GET
Attack	

Evidence	
URL	http://testphp.vulnweb.com/hpp/params.php?p=valid&pp=12
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/images/logo.gif
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/images/remark.gif
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/index.php
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/listproducts.php?artist=1
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/listproducts.php?artist=2
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/listproducts.php?artist=3
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/listproducts.php?cat=1
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/listproducts.php?cat=2
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/listproducts.php?cat=3
Method	GET
Attack	
Evidence	

URL	http://testphp.vulnweb.com/listproducts.php?cat=4
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/login.php
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/Mod_Rewrite_Shop/
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/Mod_Rewrite_Shop/BuyProduct-1/
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/Mod_Rewrite_Shop/BuyProduct-2/
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/Mod_Rewrite_Shop/BuyProduct-3/
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/color-printer/3/
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/network-attached-storage-dlink/1/
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/web-camera-a4tech/2/
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/Mod_Rewrite_Shop/images/1.jpg
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/Mod_Rewrite_Shop/images/2.jpg

Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/Mod_Rewrite_Shop/images/3.jpg
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/Mod_Rewrite_Shop/RateProduct-1.html
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/Mod_Rewrite_Shop/RateProduct-2.html
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/Mod_Rewrite_Shop/RateProduct-3.html
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/product.php?pic=1
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/product.php?pic=2
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/product.php?pic=3
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/product.php?pic=4
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/product.php?pic=5
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/product.php?pic=6
Method	GET

Attack	
Evidence	
URL	http://testphp.vulnweb.com/product.php?pic=7
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/secured/style.css
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/showimage.php?file='%20+%20pict.item(0).firstChild.nodeValue%20+%20'
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/showimage.php?file=./pictures/1.jpg
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/showimage.php?file=./pictures/1.jpg&size=160
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/showimage.php?file=./pictures/2.jpg
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/showimage.php?file=./pictures/2.jpg&size=160
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/showimage.php?file=./pictures/3.jpg
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/showimage.php?file=./pictures/3.jpg&size=160
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/showimage.php?file=./pictures/4.jpg
Method	GET

Attack	
Evidence	
URL	http://testphp.vulnweb.com/showimage.php?file=./pictures/4.jpg&size=160
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/showimage.php?file=./pictures/5.jpg
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/showimage.php?file=./pictures/5.jpg&size=160
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/showimage.php?file=./pictures/6.jpg
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/showimage.php?file=./pictures/6.jpg&size=160
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/showimage.php?file=./pictures/7.jpg
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/showimage.php?file=./pictures/7.jpg&size=160
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/signup.php
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/style.css
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/cart.php
Method	POST
Attack	

Evidence	
URL	http://testphp.vulnweb.com/guestbook.php
Method	POST
Attack	
Evidence	
URL	http://testphp.vulnweb.com/search.php?test=query
Method	POST
Attack	
Evidence	
URL	http://testphp.vulnweb.com/secured/newuser.php
Method	POST
Attack	
Evidence	
Instances	68
Solution	Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application /web server to not perform MIME-sniffing.
Reference	http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx https://owasp.org/www-community/Security_Headers
CWE Id	693
WASC Id	15
Plugin Id	10021

Informational	**Charset Mismatch (Header Versus Meta Content-Type Charset)**
Description	This check identifies responses where the HTTP Content-Type header declares a charset different from the charset defined by the body of the HTML or XML. When there's a charset mismatch between the HTTP header and content body Web browsers can be forced into an undesirable content-sniffing mode to determine the content's correct character set. An attacker could manipulate content on the page to be interpreted in an encoding of their choice. For example, if an attacker can control content at the beginning of the page, they could inject script using UTF-7 encoded text and manipulate some browsers into interpreting that text.
URL	http://testphp.vulnweb.com
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/AJAX/index.php
Method	GET
Attack	
Evidence	

URL	http://testphp.vulnweb.com/artists.php
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/artists.php?artist=1
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/artists.php?artist=2
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/artists.php?artist=3
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/cart.php
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/categories.php
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/disclaimer.php
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/guestbook.php
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/index.php
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/listproducts.php?artist=1
Method	GET
Attack	
Evidence	

URL	http://testphp.vulnweb.com/listproducts.php?artist=2
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/listproducts.php?artist=3
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/listproducts.php?cat=1
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/listproducts.php?cat=2
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/listproducts.php?cat=3
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/listproducts.php?cat=4
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/login.php
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/product.php?pic=1
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/product.php?pic=2
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/product.php?pic=3
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/product.php?pic=4

Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/product.php?pic=5
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/product.php?pic=6
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/product.php?pic=7
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/signup.php
Method	GET
Attack	
Evidence	
URL	http://testphp.vulnweb.com/cart.php
Method	POST
Attack	
Evidence	
URL	http://testphp.vulnweb.com/guestbook.php
Method	POST
Attack	
Evidence	
URL	http://testphp.vulnweb.com/search.php?test=query
Method	POST
Attack	
Evidence	
URL	http://testphp.vulnweb.com/secured/newuser.php
Method	POST
Attack	
Evidence	
Instances	32
Solution	Force UTF-8 for all text content in both the HTTP header and meta tags in HTML or encoding declarations in XML.
Reference	http://code.google.com/p/browsersec/wiki/Part2#Character_set_handling_and_detection
CWE Id	436
WASC Id	15
Plugin Id	90011

Informational	Information Disclosure - Suspicious Comments
Description	The response appears to contain suspicious comments which may help an attacker. Note: Matches made within script blocks or files are against the entire content not only comments.
URL	http://testphp.vulnweb.com/AJAX/index.php
Method	GET
Attack	
Evidence	where
Instances	1
Solution	Remove all comments that return information that may help an attacker and fix any underlying problems they refer to.
Reference	
CWE Id	200
WASC Id	13
Plugin Id	10027