ZAP Scanning Report

Summary of Alerts

Generated on Sun, 4 Jul 2021 21:49:49

Risk Level	Number of Alerts
High	0
Medium	3
Low	7
Informational	2

Alerts

Name	Risk Level	Number of Instances
Application Error Disclosure	Medium	1
Session ID in URL Rewrite	Medium	1
X-Frame-Options Header Not Set	Medium	20
Absence of Anti-CSRF Tokens	Low	7
Application Error Disclosure	Low	3
Cookie No HttpOnly Flag	Low	1
Cookie without SameSite Attribute	Low	1
Information Disclosure - Debug Error Messages	Low	2
Referer Exposes Session ID	Low	1
X-Content-Type-Options Header Missing	Low	21
Information Disclosure - Suspicious Comments	Informational	2
Loosely Scoped Cookie	Informational	2

Alert Detail

Medium (Medium)	Application Error Disclosure
Description	This page contains an error/warning message that may disclose sensitive information like the location of the file that produced the unhandled exception. This information can be used to launch further attacks against the web application. The alert could be a false positive if the error message is found inside a documentation page.
URL	http://localhost:8080/JavaVulnerableLab/install.jsp
Method	GET
Evidence	JDBC Driver
Instances	1
Solution	Review the source code of this page. Implement custom error pages. Consider implementing a mechanism to provide a unique error reference/identifier to the client (browser) while logging the details on the server side and not exposing them to the user.
Reference	
CWE Id	200
WASC Id	13
Source ID	3

Medium (High)	Session ID in URL Rewrite
Description	URL rewrite is used to track user session ID. The session ID may be disclosed via cross-site referer header. In addition, the session ID might be stored in browser history or server logs.
URL	http://localhost:8080/JavaVulnerableLab/vulnerability/baasm/URLRewriting.jsp;jsessionid=AA6CBD8410BCB4F8EAED719B94E69099
Method	GET
Evidence	jsessionid=AA6CBD8410BCB4F8EAED719B94E69099
Instances	1
Solution	For secure content, put session ID in a cookie. To be even more secure consider using a combination of cookie and URL rewrite.
Reference	http://seclists.org/lists/webappsec/2002/Oct-Dec/0111.html
CWE Id	200
WASC Id	13
Source ID	3

Medium (Medium)	X-Frame-Options Header Not Set
Description	X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.
URL	http://localhost:8080/JavaVulnerableLab/changeCardDetails.jsp
Method	GET
Parameter	X-Frame-Options
URL	http://localhost:8080/JavaVulnerableLab/vulnerability/securitymisconfig/pages.jsp?id=1
Method	GET
Parameter	X-Frame-Options
URL	http://localhost:8080/JavaVulnerableLab/vulnerability/baasm/URLRewriting.jsp;jsessionid=AA6CBD8410BCB4F8EAED719B94E69099
Method	GET
Parameter	X-Frame-Options
URL	http://localhost:8080/JavaVulnerableLab/
Method	GET
Parameter	X-Frame-Options
URL	http://localhost:8080/JavaVulnerableLab/login.jsp
Method	GET
Parameter	X-Frame-Options
URL	http://localhost:8080/JavaVulnerableLab/install.jsp
Method	GET
Parameter	X-Frame-Options
URL	http://localhost:8080/JavaVulnerableLab/vulnerability/Injection/orm.jsp?id=1
Method	GET
Parameter	X-Frame-Options
URL	http://localhost:8080/JavaVulnerableLab/vulnerability/xss/xss4.jsp
Method	GET
Parameter	X-Frame-Options
URL	http://localhost:8080/JavaVulnerableLab/vulnerability/xss/flash/exss.jsp
Method	GET
Parameter	X-Frame-Options
URL	http://localhost:8080/JavaVulnerableLab/myprofile.jsp?id
Method	GET
Parameter	X-Frame-Options
URL	http://localhost:8080/JavaVulnerableLab/vulnerability/sde/hash.jsp
Method	GET
Parameter	X-Frame-Options
URL	http://localhost:8080/JavaVulnerableLab/vulnerability/idor/change-email.jsp
Method	GET
Parameter	X-Frame-Options
URL	http://localhost:8080/JavaVulnerableLab/vulnerability/sqli/download.jsp
Method	GET
Parameter	X-Frame-Options
URL	http://localhost:8080/JavaVulnerableLab/vulnerability/idor/download.jsp
Method	GET
Parameter	X-Frame-Options
URL	http://localhost:8080/JavaVulnerableLab/vulnerability/Injection/xxe.jsp
Method	GET
Parameter	X-Frame-Options
URL	http://localhost:8080/JavaVulnerableLab/vulnerability/xss/search.jsp
Method	GET
Parameter	X-Frame-Options
URL	http://localhost:8080/JavaVulnerableLab/vulnerability/Injection/xpath_login.jsp
Method	GET
Parameter	X-Frame-Options
URL	http://localhost:8080/JavaVulnerableLab/vulnerability/baasm/SiteTitle.jsp
Method	GET
Parameter	X-Frame-Options
URL	http://localhost:8080/JavaVulnerableLab/ForgotPassword.jsp
Method	GET
Parameter	X-Frame-Options
URL	http://localhost:8080/JavaVulnerableLab/vulnerability/sqli/union2.jsp
Method	GET
Parameter	X-Frame-Options
Instances	20
Solution	Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (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
Source ID	3

Low (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://localhost:8080/JavaVulnerableLab/vulnerability/Injection/xpath_login.jsp
Method	GET
Evidence	<form action="/JavaVulnerableLab/XPathQuery.do" method="post">
URL	http://localhost:8080/JavaVulnerableLab/install.jsp
Method	GET
Evidence	<form action="Install" method="POST">
URL	http://localhost:8080/JavaVulnerableLab/vulnerability/xss/xss4.jsp
Method	GET
Evidence	<form action="xss4.jsp" method="get">
URL	http://localhost:8080/JavaVulnerableLab/login.jsp
Method	GET
Evidence	<form action="LoginValidator" method="post">
URL	http://localhost:8080/JavaVulnerableLab/vulnerability/xss/search.jsp
Method	GET
Evidence	<form id="form1" name="form1" method="GET" action="search.jsp">
URL	http://localhost:8080/JavaVulnerableLab/ForgotPassword.jsp
Method	GET
Evidence	<form action="ForgotPassword.jsp" method="post">
URL	http://localhost:8080/JavaVulnerableLab/vulnerability/Injection/xslt.jsp?style=1.xsl
Method	GET
Evidence	<form >
Instances	7
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.
Other information	No known Anti-CSRF token [anticsrf, CSRFToken, __RequestVerificationToken, csrfmiddlewaretoken, authenticity_token, OWASP_CSRFTOKEN, anoncsrf, csrf_token, _csrf, _csrfSecret, __csrf_magic, CSRF] was found in the following HTML form: [Form 1: "Login" "password" "username"].
Reference	http://projects.webappsec.org/Cross-Site-Request-Forgery http://cwe.mitre.org/data/definitions/352.html
CWE Id	352
WASC Id	9
Source ID	3

Low (Medium)	Application Error Disclosure
Description	This page contains an error/warning message that may disclose sensitive information like the location of the file that produced the unhandled exception. This information can be used to launch further attacks against the web application. The alert could be a false positive if the error message is found inside a documentation page.
URL	http://localhost:8080/JavaVulnerableLab/vulnerability/forum.jsp
Method	GET
Evidence	HTTP/1.1 500
URL	http://localhost:8080/JavaVulnerableLab/vulnerability/Injection/xslt.jsp?style=1.xsl
Method	GET
Evidence	HTTP/1.1 500
URL	http://localhost:8080/JavaVulnerableLab/vulnerability/forumposts.jsp?postid=1
Method	GET
Evidence	HTTP/1.1 500
Instances	3
Solution	Review the source code of this page. Implement custom error pages. Consider implementing a mechanism to provide a unique error reference/identifier to the client (browser) while logging the details on the server side and not exposing them to the user.
Reference	
CWE Id	200
WASC Id	13
Source ID	3

Low (Medium)	Cookie No HttpOnly Flag
Description	A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.
URL	http://localhost:8080/JavaVulnerableLab/install.jsp
Method	GET
Parameter	JSESSIONID
Evidence	Set-Cookie: JSESSIONID
Instances	1
Solution	Ensure that the HttpOnly flag is set for all cookies.
Reference	https://owasp.org/www-community/HttpOnly
CWE Id	1004
WASC Id	13
Source ID	3

Low (Medium)	Cookie without SameSite Attribute
Description	A cookie has been set without the SameSite attribute, which means that the cookie can be sent as a result of a 'cross-site' request. The SameSite attribute is an effective counter measure to cross-site request forgery, cross-site script inclusion, and timing attacks.
URL	http://localhost:8080/JavaVulnerableLab/install.jsp
Method	GET
Parameter	JSESSIONID
Evidence	Set-Cookie: JSESSIONID
Instances	1
Solution	Ensure that the SameSite attribute is set to either 'lax' or ideally 'strict' for all cookies.
Reference	https://tools.ietf.org/html/draft-ietf-httpbis-cookie-same-site
CWE Id	1275
WASC Id	13
Source ID	3

Low (Medium)	Information Disclosure - Debug Error Messages
Description	The response appeared to contain common error messages returned by platforms such as ASP.NET, and Web-servers such as IIS and Apache. You can configure the list of common debug messages.
URL	http://localhost:8080/JavaVulnerableLab/vulnerability/forumposts.jsp?postid=1
Method	GET
Evidence	Internal Server Error
URL	http://localhost:8080/JavaVulnerableLab/vulnerability/forum.jsp
Method	GET
Evidence	Internal Server Error
Instances	2
Solution	Disable debugging messages before pushing to production.
Reference	
CWE Id	200
WASC Id	13
Source ID	3

Low (Medium)	Referer Exposes Session ID
Description	A hyperlink pointing to another host name was found. As session ID URL rewrite is used, it may be disclosed in referer header to external hosts.
URL	http://localhost:8080/JavaVulnerableLab/vulnerability/baasm/URLRewriting.jsp;jsessionid=AA6CBD8410BCB4F8EAED719B94E69099
Method	GET
Evidence	www.cysecurity.org
Instances	1
Solution	This is a risk if the session ID is sensitive and the hyperlink refers to an external or third party host. For secure content, put session ID in secured session cookie.
Reference	http://seclists.org/lists/webappsec/2002/Oct-Dec/0111.html
CWE Id	200
WASC Id	13
Source ID	3

Low (Medium)	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://localhost:8080/JavaVulnerableLab/install.jsp
Method	GET
Parameter	X-Content-Type-Options
URL	http://localhost:8080/JavaVulnerableLab/vulnerability/securitymisconfig/pages.jsp?id=1
Method	GET
Parameter	X-Content-Type-Options
URL	http://localhost:8080/JavaVulnerableLab/
Method	GET
Parameter	X-Content-Type-Options
URL	http://localhost:8080/JavaVulnerableLab/vulnerability/xss/xss4.jsp
Method	GET
Parameter	X-Content-Type-Options
URL	http://localhost:8080/JavaVulnerableLab/vulnerability/idor/change-email.jsp
Method	GET
Parameter	X-Content-Type-Options
URL	http://localhost:8080/JavaVulnerableLab/vulnerability/Injection/xxe.jsp
Method	GET
Parameter	X-Content-Type-Options
URL	http://localhost:8080/JavaVulnerableLab/login.jsp
Method	GET
Parameter	X-Content-Type-Options
URL	http://localhost:8080/JavaVulnerableLab/vulnerability/sde/hash.jsp
Method	GET
Parameter	X-Content-Type-Options
URL	http://localhost:8080/JavaVulnerableLab/myprofile.jsp?id
Method	GET
Parameter	X-Content-Type-Options
URL	http://localhost:8080/JavaVulnerableLab/changeCardDetails.jsp
Method	GET
Parameter	X-Content-Type-Options
URL	http://localhost:8080/JavaVulnerableLab/vulnerability/Injection/orm.jsp?id=1
Method	GET
Parameter	X-Content-Type-Options
URL	http://localhost:8080/JavaVulnerableLab/vulnerability/idor/download.jsp
Method	GET
Parameter	X-Content-Type-Options
URL	http://localhost:8080/JavaVulnerableLab/vulnerability/sqli/download.jsp
Method	GET
Parameter	X-Content-Type-Options
URL	http://localhost:8080/JavaVulnerableLab/vulnerability/baasm/URLRewriting.jsp;jsessionid=AA6CBD8410BCB4F8EAED719B94E69099
Method	GET
Parameter	X-Content-Type-Options
URL	http://localhost:8080/JavaVulnerableLab/vulnerability/baasm/SiteTitle.jsp
Method	GET
Parameter	X-Content-Type-Options
URL	http://localhost:8080/JavaVulnerableLab/vulnerability/xss/flash/xss1.swf?vuln=/JavaVulnerableLab
Method	GET
Parameter	X-Content-Type-Options
URL	http://localhost:8080/JavaVulnerableLab/vulnerability/sqli/union2.jsp
Method	GET
Parameter	X-Content-Type-Options
URL	http://localhost:8080/JavaVulnerableLab/ForgotPassword.jsp
Method	GET
Parameter	X-Content-Type-Options
URL	http://localhost:8080/JavaVulnerableLab/vulnerability/xss/flash/exss.jsp
Method	GET
Parameter	X-Content-Type-Options
URL	http://localhost:8080/JavaVulnerableLab/vulnerability/xss/search.jsp
Method	GET
Parameter	X-Content-Type-Options
Instances	21
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.
Other information	This issue still applies to error type pages (401, 403, 500, etc.) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At "High" threshold this scan rule will not alert on client or server error responses.
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
Source ID	3

Informational (Low)	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://localhost:8080/JavaVulnerableLab/ForgotPassword.jsp
Method	GET
Evidence	username
URL	http://localhost:8080/JavaVulnerableLab/vulnerability/Injection/xxe.jsp
Method	GET
Evidence	username
Instances	2
Solution	Remove all comments that return information that may help an attacker and fix any underlying problems they refer to.
Other information	The following pattern was used: \bUSERNAME\b and was detected in the element starting with: "<script type="text/javascript"> $(document).ready(function(){ $("#username").change(function", see evidence field for the suspicious comment/snippet.
Reference	
CWE Id	200
WASC Id	13
Source ID	3

Informational (Low)	Loosely Scoped Cookie
Description	Cookies can be scoped by domain or path. This check is only concerned with domain scope.The domain scope applied to a cookie determines which domains can access it. For example, a cookie can be scoped strictly to a subdomain e.g. www.nottrusted.com, or loosely scoped to a parent domain e.g. nottrusted.com. In the latter case, any subdomain of nottrusted.com can access the cookie. Loosely scoped cookies are common in mega-applications like google.com and live.com. Cookies set from a subdomain like app.foo.bar are transmitted only to that domain by the browser. However, cookies scoped to a parent-level domain may be transmitted to the parent, or any subdomain of the parent.
URL	http://localhost:8080/JavaVulnerableLab/install.jsp
Method	GET
URL	http://localhost:8080/JavaVulnerableLab/install.jsp
Method	GET
Instances	2
Solution	Always scope cookies to a FQDN (Fully Qualified Domain Name).
Other information	The origin domain used for comparison was: localhost JSESSIONID=4F30C4C6844C7D9E6F927505335DCE33
Reference	https://tools.ietf.org/html/rfc6265#section-4.1 https://owasp.org/www-project-web-security-testing-guide/v41/4-Web_Application_Security_Testing/06-Session_Management_Testing/02-Testing_for_Cookies_Attributes.html http://code.google.com/p/browsersec/wiki/Part2#Same-origin_policy_for_cookies
CWE Id	565
WASC Id	15
Source ID	3