ZAP Scanning Report

Summary of Alerts

Risk Level	Number of Alerts
High	0
Medium	1
Low	3
Informational	0

Alerts

Name	Risk Level	Number of Instances
X-Frame-Options Header Not Set	Medium	7
Absence of Anti-CSRF Tokens	Low	7
Cookie without SameSite Attribute	Low	8
X-Content-Type-Options Header Missing	Low	8

Alert Detail

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://127.0.0.1:5000/login
Method	POST
Parameter	X-Frame-Options
URL	http://127.0.0.1:5000/sign-up
Method	POST
Parameter	X-Frame-Options
URL	http://127.0.0.1:5000/login?next=%2F
Method	GET
Parameter	X-Frame-Options
URL	http://127.0.0.1:5000/sign-up
Method	GET
Parameter	X-Frame-Options
URL	http://127.0.0.1:5000/login?next=%2F
Method	POST
Parameter	X-Frame-Options
URL	http://127.0.0.1:5000/login

Method	GET
Parameter	X-Frame-Options
URL	http://127.0.0.1:5000/
Method	GET
Parameter	X-Frame-Options
Instances	7
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://127.0.0.1:5000/sign-up
Method	GET
Evidence	<form method="POST">
URL	http://127.0.0.1:5000/login
Method	GET
Evidence	<form method="POST">
URL	http://127.0.0.1:5000/
Method	GET
Evidence	<form method="POST">
URL	http://127.0.0.1:5000/sign-up
Method	POST

Evidence	<form method="POST">
URL	http://127.0.0.1:5000/login?next=%2F
Method	POST
Evidence	<form method="POST">
URL	http://127.0.0.1:5000/login?next=%2F
Method	GET
Evidence	<form method="POST">
URL	http://127.0.0.1:5000/login
Method	POST
Evidence	<form method="POST">
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: "email" "firstName" "password1" "password2"].
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)	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://127.0.0.1:5000/
Method	GET
Parameter	session
Evidence	Set-Cookie: session
URL	http://127.0.0.1:5000/sign-up
Method	POST
Parameter	session
Evidence	Set-Cookie: session
URL	http://127.0.0.1:5000
Method	GET
Parameter	session
Evidence	Set-Cookie: session
URL	http://127.0.0.1:5000/login?next=%2F
Method	POST
Parameter	session
Evidence	Set-Cookie: session
URL	http://127.0.0.1:5000/login
Method	GET
Parameter	session
Evidence	Set-Cookie: session
URL	http://127.0.0.1:5000/sign-up
Method	GET
Parameter	session
Evidence	Set-Cookie: session
URL	http://127.0.0.1:5000/login
Method	POST
Parameter	session
Evidence	Set-Cookie: session
URL	http://127.0.0.1:5000/login?next=%2F
Method	GET
Parameter	session
Evidence	Set-Cookie: session
Instances	8
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)	**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://127.0.0.1:5000/login
Method	GET
Parameter	X-Content-Type-Options
URL	http://127.0.0.1:5000/sign-up
Method	GET
Parameter	X-Content-Type-Options
URL	http://127.0.0.1:5000/login?next=%2F
Method	POST
Parameter	X-Content-Type-Options
URL	http://127.0.0.1:5000/static/index.js
Method	GET
Parameter	X-Content-Type-Options
URL	http://127.0.0.1:5000/login
Method	POST
Parameter	X-Content-Type-Options
URL	http://127.0.0.1:5000/login?next=%2F
Method	GET
Parameter	X-Content-Type-Options
URL	http://127.0.0.1:5000/sign-up
Method	POST
Parameter	X-Content-Type-Options
URL	http://127.0.0.1:5000/
Method	GET
Parameter	X-Content-Type-Options
Instances	8
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