ZAP Scanning Report

Generated with ZAP on Sun 9 Apr 2023, at 19:34:40

Contents

About this report

Report parameters

Contexts

The following contexts were selected to be included:

- `http://206.189.48.173:8080`

Sites

The following sites were included:

- `http://206.189.48.173:8080`

(If no sites were selected, all sites were included by default.)

An included site must also be within one of the included contexts for its data to be included in the report.

Risk levels

Included: `High`, `Medium`, `Low`, `Informational`

Excluded: None

Confidence levels

Included: `User Confirmed`, `High`, `Medium`, `Low`

Excluded: `User Confirmed`, `High`, `Medium`, `Low`, `False Positive`

Summaries

Alert counts by risk and confidence

This table shows the number of alerts for each level of risk and confidence included in the report.

(The percentages in brackets represent the count as a percentage of the total number of alerts included in the report, rounded to one decimal place.)

		Confidence				
		User Confirmed	High	Medium	Low	Total
Risk	High	0 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)
	Medium	0 (0.0%)	1 (12.5%)	1 (12.5%)	1 (12.5%)	3 (37.5%)
	Low	0 (0.0%)	0 (0.0%)	3 (37.5%)	0 (0.0%)	3 (37.5%)
	Informational	0 (0.0%)	0 (0.0%)	2 (25.0%)	0 (0.0%)	2 (25.0%)
	Total	0 (0.0%)	1 (12.5%)	6 (75.0%)	1 (12.5%)	8 (100%)

Alert counts by site and risk

This table shows, for each site for which one or more alerts were raised, the number of alerts raised at each risk level.

Alerts with a confidence level of "False Positive" have been excluded from these counts.

(The numbers in brackets are the number of alerts raised for the site at or above that risk level.)

		Risk			
		High (= High)	Medium (>= Medium)	Low (>= Low)	Informational (>= Informational)
Site	http://206.189.48.173:8080	0 (0)	3 (3)	3 (6)	1 (7)

Alert counts by alert type

This table shows the number of alerts of each alert type, together with the alert type's risk level.

(The percentages in brackets represent each count as a percentage, rounded to one decimal place, of the total number of alerts included in this report.)

Alert type	Risk	Count
Absence of Anti-CSRF Tokens	Medium	4 (50.0%)
Content Security Policy (CSP) Header Not Set	Medium	61 (762.5%)
Missing Anti-clickjacking Header	Medium	64 (800.0%)
Cookie No HttpOnly Flag	Low	2 (25.0%)
Cookie without SameSite Attribute	Low	2 (25.0%)
X-Content-Type-Options Header Missing	Low	65 (812.5%)
Total		8

Alert type	Risk	Count
Information Disclosure - Sensitive Information in URL	Informational	87 (1,087.5%)
User Agent Fuzzer	Informational	103 (1,287.5%)
Total		8

Alerts

Risk=Medium, Confidence=High (1)

http://206.189.48.173:8080 (1)

Content Security Policy (CSP) Header Not Set (1)

▼ GET http://206.189.48.173:8080

Alert tags	▪ OWASP_2021_A05 ▪ OWASP_2017_A06
Alert description	Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks, including Cross Site Scripting (XSS) and data injection attacks. These attacks are used for everything from data theft to site defacement or distribution of malware. CSP provides a set of standard HTTP headers that allow website owners to declare approved sources of content that browsers should be allowed to load on that page — covered types are JavaScript, CSS, HTML frames, fonts, images and

	embeddable objects such as Java applets, ActiveX, audio and video files.

Request	▼ Request line and header section (207 bytes)
	
	`GET http://206.189.48.173:8080 HTTP/1.1`
	`Host: 206.189.48.173:8080`
	`User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:105.0) Gecko/20100101 Firefox/105.0`
	`Pragma: no-cache`
	`Cache-Control: no-cache`
	
	▼ Request body (0 bytes)

Response	▼ Status line and header section (119 bytes)
	
	`HTTP/1.1 200 OK`
	`Content-Type: text/html; charset=utf-8`
	`Date: Sun, 09 Apr 2023 14:27:40 GMT`
	`Content-Length: 10403`
	
	▶ Response body (10403 bytes)

Solution	Ensure that your web server, application server, load balancer, etc. is configured to set the Content-Security-Policy header.

Risk=`Medium`, Confidence=`Medium` (1)

`http://206.189.48.173:8080` (1)

Missing Anti-clickjacking Header (1)

▼ GET http://206.189.48.173:8080

Alert tags	▪ OWASP_2021_A05 ▪ WSTG-v42-CLNT-09 ▪ OWASP_2017_A06
Alert description	The response does not include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options to protect against 'ClickJacking' attacks.

Request	▼ Request line and header section (207 bytes)
	
	`GET http://206.189.48.173:8080 HTTP/1.1`
	`Host: 206.189.48.173:8080`
	`User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:105.0) Gecko/20100101 Firefox/105.0`
	`Pragma: no-cache`
	`Cache-Control: no-cache`
	
	▼ Request body (0 bytes)

Response	▼ Status line and header section (119 bytes)
	
	`HTTP/1.1 200 OK`
	`Content-Type: text/html; charset=utf-8`
	`Date: Sun, 09 Apr 2023 14:27:40 GMT`
	`Content-Length: 10403`
	
	▶ Response body (10403 bytes)

Parameter	`X-Frame-Options`
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.

Risk=`Medium`, Confidence=`Low` (1)

`http://206.189.48.173:8080` (1)

Absence of Anti-CSRF Tokens (1)

▼ `GET http://206.189.48.173:8080/login`

Alert tags	▪ OWASP_2021_A01 ▪ WSTG-v42-SESS-05 ▪ OWASP_2017_A05
Alert 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.
Other info	No known Anti-CSRF token [anticsrf, CSRFToken, __RequestVerificationToken, csrfmiddlewaretoken, authenticity_token,

	OWASP_CSRFTOKEN, anoncsrf, csrf_token, _csrf, _csrfSecret, __csrf_magic, CSRF, _token, _csrf_token] was found in the following HTML form: [Form 1: "password" "username"].

Request	▼ Request line and header section (250 bytes)
	
	`GET http://206.189.48.173:8080/login HTTP/1.1`
	`Host: 206.189.48.173:8080`
	`User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:105.0) Gecko/20100101 Firefox/105.0`
	`Pragma: no-cache`
	`Cache-Control: no-cache`
	`Referer: http://206.189.48.173:8080`
	
	▼ Request body (0 bytes)

Response	▼ Status line and header section (117 bytes)
	
	`HTTP/1.1 200 OK`
	`Content-Type: text/html; charset=utf-8`
	`Date: Sun, 09 Apr 2023 14:27:42 GMT`
	`Content-Length: 717`
	
	▼ Response body (717 bytes)
	
	`<!doctype html>`
	`<title> Welcome | MiniTwit</title>`
	`<link rel=stylesheet type=text/css href="/web/static/style.css">`
	`<div class=page>`

```
  <h1>MiniTwit</h1>
  <div class=navigation>

    <a href="/public">Public
timeline</a> |
    <a href="/register-user">Sign
up</a> |
    <a href="/login">Sign in</a>

  </div>

  <h2>Sign In</h2>


    <form action="" method=post>
        <dl>
        <dt>Username:
        <dd><input type=text
name=username size=30>
        <dt>Password:
        <dd><input type=password
name=password size=30>
        </dl>
        <div class=actions><input
type=submit value="Sign In"></div>
    </form>


  <div class=footer>
    MiniTwit &mdash; A Go Application
  </div>
</div>
```

Evidence	`<form action="" method=post>`
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.

Risk=`Low`, Confidence=`Medium` (3)

`http://206.189.48.173:8080` (3)

[Cookie No HttpOnly Flag](#) (1)

▼ `POST http://206.189.48.173:8080/login`

Alert tags	▪ [OWASP_2021_A05](#) ▪ [WSTG-v42-SESS-02](#) ▪ [OWASP_2017_A06](#)
Alert 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.

Request	▼ Request line and header section (326 bytes)
	
	`POST http://206.189.48.173:8080/login HTTP/1.1`
	`Host: 206.189.48.173:8080`
	`User-Agent: Mozilla/5.0 (Windows NT`

	`10.0; Win64; x64; rv:105.0)`
	`Gecko/20100101 Firefox/105.0`
	`Pragma: no-cache`
	`Cache-Control: no-cache`
	`Content-Type: application/x-www-form-urlencoded`
	`Referer:`
	`http://206.189.48.173:8080/login`
	`Content-Length: 25`
	
	▼ Request body (25 bytes)
	
	`username=ZAP&password=ZAP`

Response	▼ Status line and header section (313 bytes)
	
	`HTTP/1.1 302 Found`
	`Location: /`
	`Set-Cookie:`
	`mysession=MTY4MTA1MDQ4MHxEdi1CQkFFQ180SUFBUkFCRUFBQUlQLUNBQUVHYzNSeWFXNW5EQWdBQm5WelpYSkpSQVIxYVc1MEJnUUFfck4xfIBTc_uQnXlqW323XmvPb-E4kGYHW8UKNcCY7yXoda6r;`
	`Expires=Mon, 10 Apr 2023 14:28:00 GMT;`
	`Max-Age=86400`
	`Date: Sun, 09 Apr 2023 14:28:00 GMT`
	`Content-Length: 0`
	
	▼ Response body (0 bytes)

Parameter	`mysession`
Evidence	`Set-Cookie: mysession`
Solution	Ensure that the HttpOnly flag is set for all cookies.

Cookie without SameSite Attribute (1)

▼ POST http://206.189.48.173:8080/login

Alert tags	▪ OWASP_2021_A01 ▪ WSTG-v42-SESS-02 ▪ OWASP_2017_A05
Alert 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.

Request	▼ Request line and header section (326 bytes)
	
	`POST http://206.189.48.173:8080/login HTTP/1.1`
	`Host: 206.189.48.173:8080`
	`User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:105.0) Gecko/20100101 Firefox/105.0`
	`Pragma: no-cache`
	`Cache-Control: no-cache`
	`Content-Type: application/x-www-form-urlencoded`
	`Referer: http://206.189.48.173:8080/login`
	`Content-Length: 25`
	
	▼ Request body (25 bytes)
	
	`username=ZAP&password=ZAP`

Response	▼ Status line and header section (313 bytes)

```
HTTP/1.1 302 Found
Location: /
Set-Cookie:
mysession=MTY4MTA1MDQ4MHxEdi1CQkFFQ180S
UFBUkFCRUFBQUlQLUNBQUVHYzNSeWFXNW5EQWdB
Qm5WelpYSkpSQVIxYVc1MEJnUUFfck4xfIBTc_u
QnXlqW323XmvPb-E4kGYHW8UKNcCY7yXoda6r;
Expires=Mon, 10 Apr 2023 14:28:00 GMT;
Max-Age=86400
Date: Sun, 09 Apr 2023 14:28:00 GMT
Content-Length: 0
```

▼ Response body (0 bytes)

Parameter	`mysession`
Evidence	`Set-Cookie: mysession`
Solution	Ensure that the SameSite attribute is set to either 'lax' or ideally 'strict' for all cookies.

X-Content-Type-Options Header Missing (1)

▼ `GET http://206.189.48.173:8080`

Alert tags	▪ OWASP_2021_A05 ▪ OWASP_2017_A06
Alert 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.
Other info	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.

Request	▼ Request line and header section (207 bytes)
	
	`GET http://206.189.48.173:8080 HTTP/1.1`
	`Host: 206.189.48.173:8080`
	`User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:105.0) Gecko/20100101 Firefox/105.0`
	`Pragma: no-cache`
	`Cache-Control: no-cache`
	
	▼ Request body (0 bytes)

Response	▼ Status line and header section (119 bytes)
	
	`HTTP/1.1 200 OK`
	`Content-Type: text/html; charset=utf-8`
	`Date: Sun, 09 Apr 2023 14:27:42 GMT`
	`Content-Length: 10329`
	
	▶ Response body (10329 bytes)

Parameter	`X-Content-Type-Options`
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.

Risk=`Informational`**, Confidence=**`Medium` **(2)**

`http://206.189.48.173:8080` **(1)**

Information Disclosure - Sensitive Information in URL (1)

▼ `GET http://206.189.48.173:8080/user-timeline?username=Candra%20Demick`

Alert tags	▪ OWASP_2021_A01 ▪ OWASP_2017_A03
Alert description	The request appeared to contain sensitive information leaked in the URL. This can violate PCI and most organizational compliance policies. You can configure the list of strings for this check to add or remove values specific to your environment.

Other info	The URL contains potentially sensitive information. The following string was found via the pattern: user username

Request	▼ Request line and header section (283 bytes)
	
	`GET http://206.189.48.173:8080/user-timeline?username=Candra%20Demick HTTP/1.1`
	`Host: 206.189.48.173:8080`
	`User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:105.0) Gecko/20100101 Firefox/105.0`
	`Pragma: no-cache`
	`Cache-Control: no-cache`
	`Referer: http://206.189.48.173:8080`
	
	▼ Request body (0 bytes)

Response	▼ Status line and header section (119 bytes)
	
	`HTTP/1.1 200 OK`
	`Content-Type: text/html; charset=utf-8`
	`Date: Sun, 09 Apr 2023 14:27:48 GMT`
	`Content-Length: 19223`
	
	▶ Response body (19223 bytes)

Parameter	`username`
Evidence	`username`

Solution	Do not pass sensitive information in URIs.

Appendix

Alert types

This section contains additional information on the types of alerts in the report.

Absence of Anti-CSRF Tokens

Source	raised by a passive scanner (Absence of Anti-CSRF Tokens)
CWE ID	352
WASC ID	9
Reference	▪ http://projects.webappsec.org/Cross-Site-Request-Forgery ▪ http://cwe.mitre.org/data/definitions/352.html

Content Security Policy (CSP) Header Not Set

Source	raised by a passive scanner (Content Security Policy (CSP) Header Not Set)
CWE ID	693
WASC ID	15
Reference	▪ https://developer.mozilla.org/en-US/docs/Web/Security/CSP/Introducing_Content_

	Security_Policy ▪ https://cheatsheetseries.owasp.org/cheatsheets/Content_Security_Policy_Cheat_Sheet.html ▪ http://www.w3.org/TR/CSP/ ▪ http://w3c.github.io/webappsec/specs/content-security-policy/csp-specification.dev.html ▪ http://www.html5rocks.com/en/tutorials/security/content-security-policy/ ▪ http://caniuse.com/#feat=contentsecuritypolicy ▪ http://content-security-policy.com/

Missing Anti-clickjacking Header

Source	raised by a passive scanner (Anti-clickjacking Header)
CWE ID	1021
WASC ID	15
Reference	▪ https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options

Cookie No HttpOnly Flag

Source	raised by a passive scanner (Cookie No HttpOnly Flag)

CWE ID	1004
WASC ID	13
Reference	▪ https://owasp.org/www-community/HttpOnly

Cookie without SameSite Attribute

Source	raised by a passive scanner (Cookie without SameSite Attribute)
CWE ID	1275
WASC ID	13
Reference	▪ https://tools.ietf.org/html/draft-ietf-httpbis-cookie-same-site

X-Content-Type-Options Header Missing

Source	raised by a passive scanner (X-Content-Type-Options Header Missing)
CWE ID	693
WASC ID	15
Reference	▪ http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx ▪ https://owasp.org/www-community/Security_Headers

Information Disclosure - Sensitive Information in URL

Source	raised by a passive scanner (Information Disclosure - Sensitive Information in URL)

CWE ID	200
WASC ID	13

User Agent Fuzzer

Source	raised by an active scanner (plugin ID: 10104)
Reference	▪ https://owasp.org/wstg