

Dynamic Application Security Testing Report

acme/app

FAILED	17 Alerts	0 High	4 Medium	10 Low	3 Informational
--------	--------------	-----------	-------------	-----------	--------------------

Target

Scanned URL	http://127.0.0.1:56557
Server	PHP's built-in web server (APP_ENV=test)
Scan	OWASP ZAP baseline scan: crawl (1 min) and check the responses
Login	None (as a visitor)
URLs with alerts	7
Fails on	Medium or higher

Run

Date	2026-10-17 14:53:03 – 14:54:18 (UTC)
Duration	1m 15s
Software	PHP 8.5.11 OWASP ZAP 2.17.0 pieceofcake2/security-dast 1.0.0+no-version-set

The baseline scan crawls the site and checks its requests and responses; it sends no attacks, so it does not find every vulnerability.

Alerts by risk

1. Medium

No.	Alert	Risk	Requests
1-1	Absence of Anti-CSRF Tokens	Medium	2
1-2	Content Security Policy (CSP) Header Not Set	Medium	5
1-3	Missing Anti-clickjacking Header	Medium	5
1-4	Sub Resource Integrity Attribute Missing	Medium	5

2. Low

No.	Alert	Risk	Requests
2-1	Application Error Disclosure	Low	1
2-2	Cookie No HttpOnly Flag	Low	2
2-3	Cookie without SameSite Attribute	Low	2
2-4	Cross-Domain JavaScript Source File Inclusion	Low	5
2-5	Cross-Origin-Embedder-Policy Header Missing or Invalid	Low	5
2-6	Cross-Origin-Opener-Policy Header Missing or Invalid	Low	5
2-7	Cross-Origin-Resource-Policy Header Missing or Invalid	Low	5
2-8	Permissions Policy Header Not Set	Low	5
2-9	Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)	Low	5
2-10	X-Content-Type-Options Header Missing	Low	5

3. Informational

No.	Alert	Risk	Requests
3-1	Authentication Request Identified	Informational	1
3-2	Non-Storable Content	Informational	1
3-3	Storable and Cacheable Content	Informational	5

1. Medium

1-1 Absence of Anti-CSRF Tokens

Medium

Alert ID	10202
Risk	Medium
Confidence	Low
CWE	CWE-352
WASC	WASC-9
Requests	2

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.

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.

Requests

GET http://127.0.0.1:56557/login	
Evidence	<form method="post" action="/login">
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, _csrfToken, data[_Token][key]] was found in the following HTML form: [Form 1: "email" "password"].

POST http://127.0.0.1:56557/login	
Evidence	<form method="post" action="/login">
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, _csrfToken, data[_Token][key]] was found in the following HTML form: [Form 1: "email" "password"].

References

- <https://www.zaproxy.org/docs/alerts/10202/>
- https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html
- <https://cwe.mitre.org/data/definitions/352.html>

1. Medium

1-2 Content Security Policy (CSP) Header Not Set

Medium

Alert ID	10038-1
Risk	Medium
Confidence	High
CWE	CWE-693
WASC	WASC-15
Requests	5

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.

Solution

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

Requests

GET	http://127.0.0.1:56557/
GET	http://127.0.0.1:56557/error
GET	http://127.0.0.1:56557/posts?page=1
GET	http://127.0.0.1:56557/robots.txt
GET	http://127.0.0.1:56557/sitemap.xml

References

- <https://www.zaproxy.org/docs/alerts/10038/>
- <https://developer.mozilla.org/en-US/docs/Web/HTTP/Guides/CSP>
- https://cheatsheetseries.owasp.org/cheatsheets/Content_Security_Policy_Cheat_Sheet.html
- <https://www.w3.org/TR/CSP/>
- <https://w3c.github.io/webappsec-csp/>
- <https://web.dev/articles/csp>
- <https://caniuse.com/#feat=contentsecuritypolicy>
- <https://content-security-policy.com/>

1. Medium

1-3 Missing Anti-clickjacking Header

Medium

Alert ID	10020-1
Risk	Medium
Confidence	Medium
CWE	CWE-1021
WASC	WASC-15
Requests	5

The response does not protect against 'ClickJacking' attacks. It should include either Content-Security-Policy with 'frame-ancestors' directive or 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.

Requests

GET	http://127.0.0.1:56557/
Parameter	x-frame-options

GET	http://127.0.0.1:56557/
Parameter	x-frame-options

GET	http://127.0.0.1:56557/posts?page=1
Parameter	x-frame-options

GET	http://127.0.0.1:56557/robots.txt
Parameter	x-frame-options

GET	http://127.0.0.1:56557/sitemap.xml
Parameter	x-frame-options

References

- <https://www.zaproxy.org/docs/alerts/10020/>
- <https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Headers/X-Frame-Options>

1. Medium

1-4 Sub Resource Integrity Attribute Missing

Medium

Alert ID	90003
Risk	Medium
Confidence	High
CWE	CWE-345
WASC	WASC-15
Requests	5

The integrity attribute is missing on a script or link tag served by an external server. The integrity tag prevents an attacker who have gained access to this server from injecting a malicious content.

Solution

Provide a valid integrity attribute to the tag.

Requests

GET	http://127.0.0.1:56557/
Evidence	<script src="http://cdn.example.com/lib.js"></script>

GET	http://127.0.0.1:56557/
Evidence	<script src="http://cdn.example.com/lib.js"></script>

GET	http://127.0.0.1:56557/posts?page=1
Evidence	<script src="http://cdn.example.com/lib.js"></script>

GET	http://127.0.0.1:56557/robots.txt
Evidence	<script src="http://cdn.example.com/lib.js"></script>

GET	http://127.0.0.1:56557/sitemap.xml
Evidence	<script src="http://cdn.example.com/lib.js"></script>

References

- <https://www.zaproxy.org/docs/alerts/90003/>
- https://developer.mozilla.org/en-US/docs/Web/Security/Defenses/Subresource_Integrity

2. Low

2-1 Application Error Disclosure

Low

Alert ID	90022
Risk	Low
Confidence	Medium
CWE	CWE-550
WASC	WASC-13
Requests	1

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.

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.

Requests

GET http://127.0.0.1:56557/error	
Evidence	HTTP/1.1 500 Internal Server Error

References

- <https://www.zaproxy.org/docs/alerts/90022/>

2. Low

2-2 Cookie No HttpOnly Flag

Low

Alert ID	10010
Risk	Low
Confidence	Medium
CWE	CWE-1004
WASC	WASC-13
Requests	2

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.

Solution

Ensure that the HttpOnly flag is set for all cookies.

Requests

GET http://127.0.0.1:56557/login	
Parameter	session
Evidence	Set-Cookie: session

POST http://127.0.0.1:56557/login	
Parameter	session
Evidence	Set-Cookie: session

References

- <https://www.zaproxy.org/docs/alerts/10010/>
- <https://owasp.org/www-community/HttpOnly>

2. Low

2-3 Cookie without SameSite Attribute

Low

Alert ID	10054-1
Risk	Low
Confidence	Medium
CWE	CWE-1275
WASC	WASC-13
Requests	2

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.

Solution

Ensure that the SameSite attribute is set to either 'lax' or ideally 'strict' for all cookies.

Requests

GET http://127.0.0.1:56557/login	
Parameter	session
Evidence	Set-Cookie: session

POST http://127.0.0.1:56557/login	
Parameter	session
Evidence	Set-Cookie: session

References

- <https://www.zaproxy.org/docs/alerts/10054/>
- <https://datatracker.ietf.org/doc/html/draft-ietf-httpbis-cookie-same-site>

2. Low

2-4 Cross-Domain JavaScript Source File Inclusion

Low

Alert ID	10017
Risk	Low
Confidence	Medium
CWE	CWE-829
WASC	WASC-15
Requests	5

The page includes one or more script files from a third-party domain.

Solution

Ensure JavaScript source files are loaded from only trusted sources, and the sources can't be controlled by end users of the application.

Requests

GET http://127.0.0.1:56557/	
Parameter	http://cdn.example.com/lib.js
Evidence	<script src="http://cdn.example.com/lib.js"></script>

GET http://127.0.0.1:56557/	
Parameter	http://cdn.example.com/lib.js
Evidence	<script src="http://cdn.example.com/lib.js"></script>

GET http://127.0.0.1:56557/posts?page=1	
Parameter	http://cdn.example.com/lib.js
Evidence	<script src="http://cdn.example.com/lib.js"></script>

GET http://127.0.0.1:56557/robots.txt	
Parameter	http://cdn.example.com/lib.js
Evidence	<script src="http://cdn.example.com/lib.js"></script>

GET http://127.0.0.1:56557/sitemap.xml	
Parameter	http://cdn.example.com/lib.js
Evidence	<script src="http://cdn.example.com/lib.js"></script>

References

- <https://www.zaproxy.org/docs/alerts/10017/>

2. Low

2-5 Cross-Origin-Embedder-Policy Header Missing or Invalid

Low

Alert ID	90004-2
Risk	Low
Confidence	Medium
CWE	CWE-693
WASC	WASC-14
Requests	5

Cross-Origin-Embedder-Policy header is a response header that prevents a document from loading any cross-origin resources that don't explicitly grant the document permission (using CORP or CORS).

Solution

Ensure that the application/web server sets the Cross-Origin-Embedder-Policy header appropriately, and that it sets the Cross-Origin-Embedder-Policy header to 'require-corp' for documents.
If possible, ensure that the end user uses a standards-compliant and modern web browser that supports the Cross-Origin-Embedder-Policy header (https://caniuse.com/mdn-http_headers_cross-origin-embedder-policy).

Requests

GET	http://127.0.0.1:56557/
Parameter	Cross-Origin-Embedder-Policy

GET	http://127.0.0.1:56557/
Parameter	Cross-Origin-Embedder-Policy

GET	http://127.0.0.1:56557/posts?page=1
Parameter	Cross-Origin-Embedder-Policy

GET	http://127.0.0.1:56557/robots.txt
Parameter	Cross-Origin-Embedder-Policy

GET	http://127.0.0.1:56557/sitemap.xml
Parameter	Cross-Origin-Embedder-Policy

References

- <https://www.zaproxy.org/docs/alerts/90004/>
- <https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Headers/Cross-Origin-Embedder-Policy>

2. Low

2-6 Cross-Origin-Opener-Policy Header Missing or Invalid

Low

Alert ID	90004-3
Risk	Low
Confidence	Medium
CWE	CWE-693
WASC	WASC-14
Requests	5

Cross-Origin-Opener-Policy header is a response header that allows a site to control if others included documents share the same browsing context. Sharing the same browsing context with untrusted documents might lead to data leak.

Solution

Ensure that the application/web server sets the Cross-Origin-Opener-Policy header appropriately, and that it sets the Cross-Origin-Opener-Policy header to 'same-origin' for documents.

'same-origin-allow-popups' is considered as less secured and should be avoided.

If possible, ensure that the end user uses a standards-compliant and modern web browser that supports the Cross-Origin-Opener-Policy header (https://caniuse.com/mdn-http_headers_cross-origin-opener-policy).

Requests

GET	http://127.0.0.1:56557/
Parameter	Cross-Origin-Opener-Policy

GET	http://127.0.0.1:56557/
Parameter	Cross-Origin-Opener-Policy

GET	http://127.0.0.1:56557/posts?page=1
Parameter	Cross-Origin-Opener-Policy

GET	http://127.0.0.1:56557/robots.txt
Parameter	Cross-Origin-Opener-Policy

GET	http://127.0.0.1:56557/sitemap.xml
Parameter	Cross-Origin-Opener-Policy

References

- <https://www.zaproxy.org/docs/alerts/90004/>
- <https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Headers/Cross-Origin-Opener-Policy>

2. Low

2-7 Cross-Origin-Resource-Policy Header Missing or Invalid

Low

Alert ID	90004-1
Risk	Low
Confidence	Medium
CWE	CWE-693
WASC	WASC-14
Requests	5

Cross-Origin-Resource-Policy header is an opt-in header designed to counter side-channels attacks like Spectre. Resource should be specifically set as shareable amongst different origins.

Solution

Ensure that the application/web server sets the Cross-Origin-Resource-Policy header appropriately, and that it sets the Cross-Origin-Resource-Policy header to 'same-origin' for all web pages.

'same-site' is considered as less secured and should be avoided.

If resources must be shared, set the header to 'cross-origin'.

If possible, ensure that the end user uses a standards-compliant and modern web browser that supports the Cross-Origin-Resource-Policy header (https://caniuse.com/mdn-http_headers_cross-origin-resource-policy).

Requests

GET	http://127.0.0.1:56557/
Parameter	Cross-Origin-Resource-Policy

GET	http://127.0.0.1:56557/
Parameter	Cross-Origin-Resource-Policy

GET	http://127.0.0.1:56557/posts?page=1
Parameter	Cross-Origin-Resource-Policy

GET	http://127.0.0.1:56557/robots.txt
Parameter	Cross-Origin-Resource-Policy

GET	http://127.0.0.1:56557/sitemap.xml
Parameter	Cross-Origin-Resource-Policy

References

- <https://www.zaproxy.org/docs/alerts/90004/>
- <https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Headers/Cross-Origin-Embedder-Policy>

2. Low

2-8 Permissions Policy Header Not Set

Low

Alert ID	10063-1
Risk	Low
Confidence	Medium
CWE	CWE-693
WASC	WASC-15
Requests	5

Permissions Policy Header is an added layer of security that helps to restrict from unauthorized access or usage of browser/client features by web resources. This policy ensures the user privacy by limiting or specifying the features of the browsers can be used by the web resources. Permissions Policy provides a set of standard HTTP headers that allow website owners to limit which features of browsers can be used by the page such as camera, microphone, location, full screen etc.

Solution

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

Requests

GET	http://127.0.0.1:56557/
GET	http://127.0.0.1:56557/
GET	http://127.0.0.1:56557/error
GET	http://127.0.0.1:56557/posts?page=1
GET	http://127.0.0.1:56557/robots.txt

References

- <https://www.zaproxy.org/docs/alerts/10063/>
- <https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Headers/Permissions-Policy>
- <https://developer.chrome.com/blog/feature-policy/>
- <https://scotthelme.co.uk/a-new-security-header-feature-policy/>
- <https://w3c.github.io/webappsec-feature-policy/>
- <https://www.smashingmagazine.com/2018/12/feature-policy/>

2. Low

2-9 Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)

Low

Alert ID	10037
Risk	Low
Confidence	Medium
CWE	CWE-497
WASC	WASC-13
Requests	5

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.

Solution

Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

Requests

GET http://127.0.0.1:56557/	
Evidence	X-Powered-By: PHP/8.4
GET http://127.0.0.1:56557/	
Evidence	X-Powered-By: PHP/8.4
GET http://127.0.0.1:56557/error	
Evidence	X-Powered-By: PHP/8.5.11
GET http://127.0.0.1:56557/posts?page=1	
Evidence	X-Powered-By: PHP/8.4
GET http://127.0.0.1:56557/robots.txt	
Evidence	X-Powered-By: PHP/8.4

References

- <https://www.zaproxy.org/docs/alerts/10037/>
- https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/01-Information_Gathering/08-Fingerprint_Web_Application_Framework
- <https://www.troyhunt.com/shhh-dont-let-your-response-headers/>

2. Low

2-10 X-Content-Type-Options Header Missing

Low

Alert ID	10021
Risk	Low
Confidence	Medium
CWE	CWE-693
WASC	WASC-15
Requests	5

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.

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.

Requests

GET http://127.0.0.1:56557/	
Parameter	x-content-type-options
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.

GET http://127.0.0.1:56557/	
Parameter	x-content-type-options
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.

GET http://127.0.0.1:56557/posts?page=1	
Parameter	x-content-type-options
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.

GET http://127.0.0.1:56557/robots.txt	
Parameter	x-content-type-options
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.

GET http://127.0.0.1:56557/sitemap.xml	
Parameter	x-content-type-options
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.

References

- <https://www.zaproxy.org/docs/alerts/10021/>
- [https://learn.microsoft.com/en-us/previous-versions/windows/internet-explorer/ie-developer/compatibility/gg622941\(v=vs.85\)](https://learn.microsoft.com/en-us/previous-versions/windows/internet-explorer/ie-developer/compatibility/gg622941(v=vs.85))
- <https://owasp.org/www-community/Security-Headers>

3. Informational

3-1 Authentication Request Identified

Informational

Alert ID	10111
Risk	Informational
Confidence	High
Requests	1

The given request has been identified as an authentication request. The 'Other Info' field contains a set of key=value lines which identify any relevant fields. If the request is in a context which has an Authentication Method set to "Auto-Detect" then this rule will change the authentication to match the request identified.

Solution

This is an informational alert rather than a vulnerability and so there is nothing to fix.

Requests

POST http://127.0.0.1:56557/login	
Parameter	email
Evidence	password
Other info	userParam=email userValue=zaproxy@example.com passwordParam=password referer=http://host.docker.internal:56557/login

References

- <https://www.zaproxy.org/docs/alerts/10111/>
- <https://www.zaproxy.org/docs/desktop/addons/authentication-helper/auth-reg-id/>

3. Informational

3-2 Non-Storable Content

Informational

Alert ID	10049-1
Risk	Informational
Confidence	Medium
CWE	CWE-524
WASC	WASC-13
Requests	1

The response contents are not storable by caching components such as proxy servers. If the response does not contain sensitive, personal or user-specific information, it may benefit from being stored and cached, to improve performance.

Solution

The content may be marked as storable by ensuring that the following conditions are satisfied:

The request method must be understood by the cache and defined as being cacheable ("GET", "HEAD", and "POST" are currently defined as cacheable)

The response status code must be understood by the cache (one of the 1XX, 2XX, 3XX, 4XX, or 5XX response classes are generally understood)

The "no-store" cache directive must not appear in the request or response header fields

For caching by "shared" caches such as "proxy" caches, the "private" response directive must not appear in the response

For caching by "shared" caches such as "proxy" caches, the "Authorization" header field must not appear in the request, unless the response explicitly allows it (using one of the "must-revalidate", "public", or "s-maxage" Cache-Control response directives)

In addition to the conditions above, at least one of the following conditions must also be satisfied by the response:

It must contain an "Expires" header field

It must contain a "max-age" response directive

For "shared" caches such as "proxy" caches, it must contain a "s-maxage" response directive

It must contain a "Cache Control Extension" that allows it to be cached

It must have a status code that is defined as cacheable by default (200, 203, 204, 206, 300, 301, 404, 405, 410, 414, 501).

Requests

GET http://127.0.0.1:56557/error	
Evidence	500

References

- <https://www.zaproxy.org/docs/alerts/10049/>
- <https://datatracker.ietf.org/doc/html/rfc7234>
- <https://datatracker.ietf.org/doc/html/rfc7231>
- <https://www.w3.org/Protocols/rfc2616/rfc2616-sec13.html>

3. Informational

3-3 Storable and Cacheable Content

Informational

Alert ID	10049-3
Risk	Informational
Confidence	Medium
CWE	CWE-524
WASC	WASC-13
Requests	5

The response contents are storable by caching components such as proxy servers, and may be retrieved directly from the cache, rather than from the origin server by the caching servers, in response to similar requests from other users. If the response data is sensitive, personal or user-specific, this may result in sensitive information being leaked. In some cases, this may even result in a user gaining complete control of the session of another user, depending on the configuration of the caching components in use in their environment. This is primarily an issue where "shared" caching servers such as "proxy" caches are configured on the local network. This configuration is typically found in corporate or educational environments, for instance.

Solution

Validate that the response does not contain sensitive, personal or user-specific information. If it does, consider the use of the following HTTP response headers, to limit, or prevent the content being stored and retrieved from the cache by another user:

Cache-Control: no-cache, no-store, must-revalidate, private

Pragma: no-cache

Expires: 0

This configuration directs both HTTP 1.0 and HTTP 1.1 compliant caching servers to not store the response, and to not retrieve the response (without validation) from the cache, in response to a similar request.

Requests

GET http://127.0.0.1:56557/	
Other info	In the absence of an explicitly specified caching lifetime directive in the response, a liberal lifetime heuristic of 1 year was assumed. This is permitted by rfc7234.

GET http://127.0.0.1:56557/	
Other info	In the absence of an explicitly specified caching lifetime directive in the response, a liberal lifetime heuristic of 1 year was assumed. This is permitted by rfc7234.

GET http://127.0.0.1:56557/posts?page=1	
Other info	In the absence of an explicitly specified caching lifetime directive in the response, a liberal lifetime heuristic of 1 year was assumed. This is permitted by rfc7234.

GET http://127.0.0.1:56557/robots.txt	
Other info	In the absence of an explicitly specified caching lifetime directive in the response, a liberal lifetime heuristic of 1 year was assumed. This is permitted by rfc7234.

GET http://127.0.0.1:56557/sitemap.xml	
Other info	In the absence of an explicitly specified caching lifetime directive in the response, a liberal lifetime heuristic of 1 year was assumed. This is permitted by rfc7234.

References

- <https://www.zaproxy.org/docs/alerts/10049/>
- <https://datatracker.ietf.org/doc/html/rfc7234>
- <https://datatracker.ietf.org/doc/html/rfc7231>
- <https://www.w3.org/Protocols/rfc2616/rfc2616-sec13.html>