OWASP ZAP Scan Report

Target: https://visitor-test.celloscope.net/

All scanned sites: https://visitor-test.celloscope.net

Javascript included from: https://visitor-test.celloscope.net

Generated on Sat, 15 Jul 2023 05:23:57

ZAP Version: 2.12.0

Summary of Alerts

Risk Level	Number of Alerts
High	1
Medium	2
Low	3
Informational	1

Alerts

Name	Risk Level	Number of Instances
Path Traversal	High	1
Content Security Policy (CSP) Header Not Set	Medium	3
Missing Anti-clickjacking Header	Medium	3
Server Leaks Version Information via "Server" HTTP Response Header Field	Low	28
Strict-Transport-Security Header Not Set	Low	1
X-Content-Type-Options Header Missing	Low	27
Re-examine Cache-control Directives	Informational	10

Alert Detail

High	Path Traversal
Description	The Path Traversal attack technique allows an attacker access to files, directories, and commands that potentially reside outside the web document root directory. An attacker may manipulate a URL in such a way that the web site will execute or reveal the contents of arbitrary files anywhere on the web server. Any device that exposes an HTTP-based interface is potentially vulnerable to Path Traversal. Most web sites restrict user access to a specific portion of the file-system, typically called the "web document root" or "CGI root" directory. These directories contain the files intended for user access and the executable necessary to drive web application functionality. To access files or execute commands anywhere on the file-system, Path Traversal attacks will utilize the ability of special-characters sequences. The most basic Path Traversal attack uses the "../" special-character sequence to alter the resource location requested in the URL. Although most popular web servers will prevent this technique from escaping the web document root, alternate encodings of the "../" sequence may help bypass the security filters. These method variations include valid and invalid Unicode-encoding ("..%u2216" or "..%c0%af") of the forward slash character, backslash characters ("..\") on Windows-based servers, URL encoded characters "%2e%2e%2f"), and double URL encoding ("..%255c") of the backslash character. Even if the web server properly restricts Path Traversal attempts in the URL path, a web application itself may still be vulnerable due to improper handling of user-supplied input. This is a common problem of web applications that use template mechanisms or load static text from files. In variations of the attack, the original

	URL parameter value is substituted with the file name of one of the web application's dynamic scripts. Consequently, the results can reveal source code because the file is interpreted as text instead of an executable script. These techniques often employ additional special characters such as the dot (".") to reveal the listing of the current working directory, or "%00" NULL characters in order to bypass rudimentary file extension checks.
URL	https://visitor-test.celloscope.net/api/v1/pdf-download-file
Method	POST
Parameter	fileName
Attack	../../../../../../../../../../../../../../../../etc/passwd
Evidence	root:x:0:0
Instances	1
Solution	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." For filenames, use stringent allow lists that limit the character set to be used. If feasible, only allow a single "." character in the filename to avoid weaknesses, and exclude directory separators such as "/". Use an allow list of allowable file extensions. Warning: if you attempt to cleanse your data, then do so that the end result is not in the form that can be dangerous. A sanitizing mechanism can remove characters such as '.' and ';' which may be required for some exploits. An attacker can try to fool the sanitizing mechanism into "cleaning" data into a dangerous form. Suppose the attacker injects a '.' inside a filename (e.g. "sensi.tiveFile") and the sanitizing mechanism removes the character resulting in the valid filename, "sensitiveFile". If the input data are now assumed to be safe, then the file may be compromised. Inputs should be decoded and canonicalized to the application's current internal representation before being validated. Make sure that your application does not decode the same input twice. Such errors could be used to bypass allow list schemes by introducing dangerous inputs after they have been checked. Use a built-in path canonicalization function (such as realpath() in C) that produces the canonical version of the pathname, which effectively removes ".." sequences and symbolic links. Run your code using the lowest privileges that are required to accomplish the necessary tasks. If possible, create isolated accounts with limited privileges that are only used for a single task. That way, a successful attack will not immediately give the attacker access to the rest of the software or its environment. For example, database applications rarely need to run as the database administrator, especially in day-to-day operations. When the set of acceptable objects, such as filenames or URLs, is limited or known, create a mapping from a set of fixed input values (such as numeric IDs) to the actual filenames or URLs, and reject all other inputs. Run your code in a "jail" or similar sandbox environment that enforces strict boundaries between the process and the operating system. This may effectively restrict which files can be accessed in a particular directory or which commands can be executed by your software. OS-level examples include the Unix chroot jail, AppArmor, and SELinux. In general, managed code may provide some protection. For example, java.io.FilePermission in the Java SecurityManager allows you to specify restrictions on file operations. This may not be a feasible solution, and it only limits the impact to the operating system; the rest of your application may still be subject to compromise.
Reference	http://projects.webappsec.org/Path-Traversal http://cwe.mitre.org/data/definitions/22.html
CWE Id	22
WASC Id	33
Plugin Id	6

Medium	**Content Security Policy (CSP) Header Not Set**
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.
URL	https://visitor-test.celloscope.net/
Method	GET

Parameter	
Attack	
Evidence	
URL	https://visitor-test.celloscope.net/robots.txt
Method	GET
Parameter	
Attack	
Evidence	
URL	https://visitor-test.celloscope.net/sitemap.xml
Method	GET
Parameter	
Attack	
Evidence	
Instances	3
Solution	Ensure that your web server, application server, load balancer, etc. is configured to set the Content-Security-Policy header.
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/
CWE Id	693
WASC Id	15
Plugin Id	10038

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	https://visitor-test.celloscope.net/
Method	GET
Parameter	X-Frame-Options
Attack	
Evidence	
URL	https://visitor-test.celloscope.net/robots.txt
Method	GET
Parameter	X-Frame-Options
Attack	
Evidence	
URL	https://visitor-test.celloscope.net/sitemap.xml
Method	GET
Parameter	X-Frame-Options
Attack	
Evidence	
Instances	3
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 Version Information via "Server" HTTP Response Header Field**
Description	The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.
URL	https://visitor-test.celloscope.net/
Method	GET
Parameter	
Attack	
Evidence	nginx/1.24.0
URL	https://visitor-test.celloscope.net/0.d7f91fe3e6563748.js
Method	GET
Parameter	
Attack	
Evidence	nginx/1.24.0
URL	https://visitor-test.celloscope.net/279.7cc6acd149e903f8.js
Method	GET
Parameter	
Attack	
Evidence	nginx/1.24.0
URL	https://visitor-test.celloscope.net/302.112504c23e744251.js
Method	GET
Parameter	
Attack	
Evidence	nginx/1.24.0
URL	https://visitor-test.celloscope.net/359.570554fb705bb5f9.js
Method	GET
Parameter	
Attack	
Evidence	nginx/1.24.0
URL	https://visitor-test.celloscope.net/637.00e17f36fe40bd1d.js
Method	GET
Parameter	
Attack	
Evidence	nginx/1.24.0
URL	https://visitor-test.celloscope.net/647.85319bb2c4c56e9b.js
Method	GET
Parameter	

Attack	
Evidence	nginx/1.24.0
URL	https://visitor-test.celloscope.net/712.7b38c2b38bc4fd2a.js
Method	GET
Parameter	
Attack	
Evidence	nginx/1.24.0
URL	https://visitor-test.celloscope.net/769.ee2738b880d16c96.js
Method	GET
Parameter	
Attack	
Evidence	nginx/1.24.0
URL	https://visitor-test.celloscope.net/api
Method	GET
Parameter	
Attack	
Evidence	nginx/1.24.0
URL	https://visitor-test.celloscope.net/api/v1/company-type/company-type-list
Method	GET
Parameter	
Attack	
Evidence	nginx/1.24.0
URL	https://visitor-test.celloscope.net/api/v1/get/person/configuration/info
Method	GET
Parameter	
Attack	
Evidence	nginx/1.24.0
URL	https://visitor-test.celloscope.net/api/v1/sector/sector-list
Method	GET
Parameter	
Attack	
Evidence	nginx/1.24.0
URL	https://visitor-test.celloscope.net/assets/i18n/bn.json
Method	GET
Parameter	
Attack	
Evidence	nginx/1.24.0
URL	https://visitor-test.celloscope.net/assets/i18n/en.json
Method	GET
Parameter	
Attack	
Evidence	nginx/1.24.0

URL	https://visitor-test.celloscope.net/main.c98044b792cbc88c.js
Method	GET
Parameter	
Attack	
Evidence	nginx/1.24.0
URL	https://visitor-test.celloscope.net/manifest.webmanifest
Method	GET
Parameter	
Attack	
Evidence	nginx/1.24.0
URL	https://visitor-test.celloscope.net/polyfills.10724c908b399f12.js
Method	GET
Parameter	
Attack	
Evidence	nginx/1.24.0
URL	https://visitor-test.celloscope.net/Roboto-Black.b4556791e2a9e005.woff2
Method	GET
Parameter	
Attack	
Evidence	nginx/1.24.0
URL	https://visitor-test.celloscope.net/Roboto-Bold.2a63183e6dff7d00.woff2
Method	GET
Parameter	
Attack	
Evidence	nginx/1.24.0
URL	https://visitor-test.celloscope.net/Roboto-Light.86fc2559ff73eac5.woff2
Method	GET
Parameter	
Attack	
Evidence	nginx/1.24.0
URL	https://visitor-test.celloscope.net/Roboto-Medium.f8693cca22ae31bc.woff2
Method	GET
Parameter	
Attack	
Evidence	nginx/1.24.0
URL	https://visitor-test.celloscope.net/Roboto-Regular.4e7449338f3a9fee.woff2
Method	GET
Parameter	
Attack	
Evidence	nginx/1.24.0
URL	https://visitor-test.celloscope.net/robots.txt
Method	GET

Parameter	
Attack	
Evidence	nginx/1.24.0
URL	https://visitor-test.celloscope.net/runtime.bbb02d81af989321.js
Method	GET
Parameter	
Attack	
Evidence	nginx/1.24.0
URL	https://visitor-test.celloscope.net/scripts.ae171163888a14c2.js
Method	GET
Parameter	
Attack	
Evidence	nginx/1.24.0
URL	https://visitor-test.celloscope.net/sitemap.xml
Method	GET
Parameter	
Attack	
Evidence	nginx/1.24.0
URL	https://visitor-test.celloscope.net/api/v1/pdf-download-file
Method	POST
Parameter	
Attack	
Evidence	nginx/1.24.0
Instances	28
Solution	Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.
Reference	http://httpd.apache.org/docs/current/mod/core.html#servertokens http://msdn.microsoft.com/en-us/library/ff648552.aspx#ht_urlscan_007 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	10036

Low	**Strict-Transport-Security Header Not Set**
Description	HTTP Strict Transport Security (HSTS) is a web security policy mechanism whereby a web server declares that complying user agents (such as a web browser) are to interact with it using only secure HTTPS connections (i.e. HTTP layered over TLS/SSL). HSTS is an IETF standards track protocol and is specified in RFC 6797.
URL	https://visitor-test.celloscope.net/api
Method	GET
Parameter	
Attack	
Evidence	
Instances	1
Solution	Ensure that your web server, application server, load balancer, etc. is configured to enforce Strict-Transport-Security.
Reference	https://cheatsheetseries.owasp.org/cheatsheets/HTTP_Strict_Transport_Security_Cheat_Sheet.html

	https://owasp.org/www-community/Security_Headers http://en.wikipedia.org/wiki/HTTP_Strict_Transport_Security http://caniuse.com/stricttransportsecurity http://tools.ietf.org/html/rfc6797
CWE Id	319
WASC Id	15
Plugin Id	10035

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	https://visitor-test.celloscope.net/
Method	GET
Parameter	X-Content-Type-Options
Attack	
Evidence	
URL	https://visitor-test.celloscope.net/0.d7f91fe3e6563748.js
Method	GET
Parameter	X-Content-Type-Options
Attack	
Evidence	
URL	https://visitor-test.celloscope.net/279.7cc6acd149e903f8.js
Method	GET
Parameter	X-Content-Type-Options
Attack	
Evidence	
URL	https://visitor-test.celloscope.net/302.112504c23e744251.js
Method	GET
Parameter	X-Content-Type-Options
Attack	
Evidence	
URL	https://visitor-test.celloscope.net/359.570554fb705bb5f9.js
Method	GET
Parameter	X-Content-Type-Options
Attack	
Evidence	
URL	https://visitor-test.celloscope.net/637.00e17f36fe40bd1d.js
Method	GET
Parameter	X-Content-Type-Options
Attack	
Evidence	
URL	https://visitor-test.celloscope.net/647.85319bb2c4c56e9b.js
Method	GET
Parameter	X-Content-Type-Options

Attack	
Evidence	
URL	https://visitor-test.celloscope.net/712.7b38c2b38bc4fd2a.js
Method	GET
Parameter	X-Content-Type-Options
Attack	
Evidence	
URL	https://visitor-test.celloscope.net/769.ee2738b880d16c96.js
Method	GET
Parameter	X-Content-Type-Options
Attack	
Evidence	
URL	https://visitor-test.celloscope.net/api/v1/company-type/company-type-list
Method	GET
Parameter	X-Content-Type-Options
Attack	
Evidence	
URL	https://visitor-test.celloscope.net/api/v1/get/person/configuration/info
Method	GET
Parameter	X-Content-Type-Options
Attack	
Evidence	
URL	https://visitor-test.celloscope.net/api/v1/sector/sector-list
Method	GET
Parameter	X-Content-Type-Options
Attack	
Evidence	
URL	https://visitor-test.celloscope.net/assets/i18n/bn.json
Method	GET
Parameter	X-Content-Type-Options
Attack	
Evidence	
URL	https://visitor-test.celloscope.net/assets/i18n/en.json
Method	GET
Parameter	X-Content-Type-Options
Attack	
Evidence	
URL	https://visitor-test.celloscope.net/main.c98044b792cbc88c.js
Method	GET
Parameter	X-Content-Type-Options
Attack	
Evidence	

URL	https://visitor-test.celloscope.net/manifest.webmanifest
Method	GET
Parameter	X-Content-Type-Options
Attack	
Evidence	
URL	https://visitor-test.celloscope.net/polyfills.10724c908b399f12.js
Method	GET
Parameter	X-Content-Type-Options
Attack	
Evidence	
URL	https://visitor-test.celloscope.net/Roboto-Black.b4556791e2a9e005.woff2
Method	GET
Parameter	X-Content-Type-Options
Attack	
Evidence	
URL	https://visitor-test.celloscope.net/Roboto-Bold.2a63183e6dff7d00.woff2
Method	GET
Parameter	X-Content-Type-Options
Attack	
Evidence	
URL	https://visitor-test.celloscope.net/Roboto-Light.86fc2559ff73eac5.woff2
Method	GET
Parameter	X-Content-Type-Options
Attack	
Evidence	
URL	https://visitor-test.celloscope.net/Roboto-Medium.f8693cca22ae31bc.woff2
Method	GET
Parameter	X-Content-Type-Options
Attack	
Evidence	
URL	https://visitor-test.celloscope.net/Roboto-Regular.4e7449338f3a9fee.woff2
Method	GET
Parameter	X-Content-Type-Options
Attack	
Evidence	
URL	https://visitor-test.celloscope.net/robots.txt
Method	GET
Parameter	X-Content-Type-Options
Attack	
Evidence	
URL	https://visitor-test.celloscope.net/runtime.bbb02d81af989321.js
Method	GET

Parameter	X-Content-Type-Options
Attack	
Evidence	
URL	https://visitor-test.celloscope.net/scripts.ae171163888a14c2.js
Method	GET
Parameter	X-Content-Type-Options
Attack	
Evidence	
URL	https://visitor-test.celloscope.net/sitemap.xml
Method	GET
Parameter	X-Content-Type-Options
Attack	
Evidence	
URL	https://visitor-test.celloscope.net/api/v1/pdf-download-file
Method	POST
Parameter	X-Content-Type-Options
Attack	
Evidence	
Instances	27
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	**Re-examine Cache-control Directives**
Description	The cache-control header has not been set properly or is missing, allowing the browser and proxies to cache content. For static assets like css, js, or image files this might be intended, however, the resources should be reviewed to ensure that no sensitive content will be cached.
URL	https://visitor-test.celloscope.net/
Method	GET
Parameter	Cache-Control
Attack	
Evidence	
URL	https://visitor-test.celloscope.net/api/v1/company-type/company-type-list
Method	GET
Parameter	Cache-Control
Attack	
Evidence	no-cache
URL	https://visitor-test.celloscope.net/api/v1/get/person/configuration/info
Method	GET

Parameter	Cache-Control
Attack	
Evidence	no-cache
URL	https://visitor-test.celloscope.net/api/v1/sector/sector-list
Method	GET
Parameter	Cache-Control
Attack	
Evidence	no-cache
URL	https://visitor-test.celloscope.net/assets/i18n/bn.json
Method	GET
Parameter	Cache-Control
Attack	
Evidence	
URL	https://visitor-test.celloscope.net/assets/i18n/en.json
Method	GET
Parameter	Cache-Control
Attack	
Evidence	
URL	https://visitor-test.celloscope.net/manifest.webmanifest
Method	GET
Parameter	Cache-Control
Attack	
Evidence	
URL	https://visitor-test.celloscope.net/robots.txt
Method	GET
Parameter	Cache-Control
Attack	
Evidence	
URL	https://visitor-test.celloscope.net/sitemap.xml
Method	GET
Parameter	Cache-Control
Attack	
Evidence	
URL	https://visitor-test.celloscope.net/api/v1/pdf-download-file
Method	POST
Parameter	Cache-Control
Attack	
Evidence	no-cache
Instances	10
Solution	For secure content, ensure the cache-control HTTP header is set with "no-cache, no-store, must-revalidate". If an asset should be cached consider setting the directives "public, max-age, immutable".
Reference	https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet.html#web-content-caching https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Cache-Control https://grayduck.mn/2021/09/13/cache-control-recommendations/

CWE Id	525
WASC Id	13
Plugin Id	10015