ZAP Scanning Report

Site: https://ehr-online.co.uk

Generated on Thu, 21 Jul 2022 13:57:50

Summary of Alerts

Risk Level	Number of Alerts
High	1
Medium	1
Low	7
Informational	1
False Positives:	0

Alerts

Name	Risk Level	Number of Instances
Path Traversal	High	1
Vulnerable JS Library	Medium	1
Absence of Anti-CSRF Tokens	Low	1
Cookie No HttpOnly Flag	Low	1
Cookie Without Secure Flag	Low	1
Cookie without SameSite Attribute	Low	1
Incomplete or No Cache-control Header Set	Low	3
Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)	Low	3
Timestamp Disclosure - Unix	Low	4
Information Disclosure - Suspicious Comments	Informational	3

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://ehr-online.co.uk/interface/login/login.php?site=%2Flogin.php
Method	GET
Attack	/login.php
Evidence	
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	**Vulnerable JS Library**
Description	The identified library jquery, version 1.4.3 is vulnerable.
URL	https://ehr-online.co.uk/library/js/jquery-1.4.3.min.js
Method	GET
Attack	
Evidence	jquery-1.4.3.min.js
Instances	1
Solution	Please upgrade to the latest version of jquery.
Reference	https://nvd.nist.gov/vuln/detail/CVE-2012-6708 https://github.com/jquery/jquery/issues/2432 http://research.insecurelabs.org/jquery/test/ https://bugs.jquery.com/ticket/9521 http://blog.jquery.com/2016/01/08/jquery-2-2-and-1-12-released/ http://bugs.jquery.com/ticket/11290 https://blog.jquery.com/2019/04/10/jquery-3-4-0-released/ https://nvd.nist.gov/vuln/detail/CVE-2019-11358 https://nvd.nist.gov/vuln/detail/CVE-2015-9251 https://github.com/jquery/jquery/commit/753d591aea698e57d6db58c9f722cd0808619b1b https://blog.jquery.com/2020/04/10/jquery-3-5-0-released/ https://nvd.nist.gov/vuln/detail/CVE-2011-4969
CWE Id	829
WASC Id	
Plugin Id	10003

Low	**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	https://ehr-online.co.uk/interface/login/login.php?site=default
Method	GET
Attack	
Evidence	<form method="POST" action="../main/main_screen.php?auth=login&site=default" target="_top" name="login_form" onsubmit="return imsubmitted();">
Instances	1
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.
Reference	http://projects.webappsec.org/Cross-Site-Request-Forgery http://cwe.mitre.org/data/definitions/352.html
CWE Id	352
WASC Id	9
Plugin Id	10202

Low	**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	https://ehr-online.co.uk/interface/login/login.php?site=default
Method	GET
Attack	
Evidence	Set-Cookie: LibreHealthEHR
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
Plugin Id	10010

Low	**Cookie Without Secure Flag**
Description	A cookie has been set without the secure flag, which means that the cookie can be accessed via unencrypted connections.
URL	https://ehr-online.co.uk/interface/login/login.php?site=default
Method	GET
Attack	
Evidence	Set-Cookie: LibreHealthEHR
Instances	1
Solution	Whenever a cookie contains sensitive information or is a session token, then it should always be passed using an encrypted channel. Ensure that the secure flag is set for cookies containing such sensitive information.
Reference	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
CWE Id	614
WASC Id	13
Plugin Id	10011

Low	**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	https://ehr-online.co.uk/interface/login/login.php?site=default
Method	GET
Attack	
Evidence	Set-Cookie: LibreHealthEHR
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
Plugin Id	10054

Low	**Incomplete or No Cache-control Header Set**
Description	The cache-control header has not been set properly or is missing, allowing the browser and proxies to cache content.

URL	https://ehr-online.co.uk/acknowledge_license_cert.html
Method	GET
Attack	
Evidence	max-age=3600, must-revalidate
URL	https://ehr-online.co.uk/mpl_license.txt
Method	GET
Attack	
Evidence	max-age=3600, public, must-revalidate
URL	https://ehr-online.co.uk/robots.txt
Method	GET
Attack	
Evidence	max-age=3600, public, must-revalidate
Instances	3
Solution	Whenever possible ensure the cache-control HTTP header is set with no-cache, no-store, must-revalidate.
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
CWE Id	525
WASC Id	13
Plugin Id	10015

Low	**Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)**
Description	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.
URL	https://ehr-online.co.uk/interface/login/login.php?site=default
Method	GET
Attack	
Evidence	X-Powered-By: PHP/7.4.30
URL	https://ehr-online.co.uk/interface/login_screen.php?error=1&site
Method	GET
Attack	
Evidence	X-Powered-By: PHP/7.4.30
URL	https://ehr-online.co.uk/interface/main/main_screen.php?auth=login&site=default
Method	POST
Attack	
Evidence	X-Powered-By: PHP/7.4.30
Instances	3
Solution	Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.
Reference	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	10037

Low	Timestamp Disclosure - Unix
Description	A timestamp was disclosed by the application/web server - Unix
URL	https://ehr-online.co.uk/library/css/bootstrap-3-2-0.min.css
Method	GET
Attack	
Evidence	33333333
URL	https://ehr-online.co.uk/library/css/bootstrap-3-2-0.min.css
Method	GET
Attack	
Evidence	42857143
URL	https://ehr-online.co.uk/library/css/bootstrap-3-2-0.min.css
Method	GET
Attack	
Evidence	66666667
URL	https://ehr-online.co.uk/library/css/bootstrap-3-2-0.min.css
Method	GET
Attack	
Evidence	80000000
Instances	4
Solution	Manually confirm that the timestamp data is not sensitive, and that the data cannot be aggregated to disclose exploitable patterns.
Reference	http://projects.webappsec.org/w/page/13246936/Information%20Leakage
CWE Id	200
WASC Id	13
Plugin Id	10096

Informational	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	https://ehr-online.co.uk/library/js/jquery-1.4.3.min.js
Method	GET
Attack	
Evidence	db
URL	https://ehr-online.co.uk/library/js/jquery-1.4.3.min.js
Method	GET
Attack	
Evidence	select
URL	https://ehr-online.co.uk/library/js/jquery-1.4.3.min.js
Method	GET
Attack	

Evidence	username
Instances	3
Solution	Remove all comments that return information that may help an attacker and fix any underlying problems they refer to.
Reference	
CWE Id	200
WASC Id	13
Plugin Id	10027