ZAP Scanning Report

Site: http://127.0.0.1:8080

Generated on Tue, 23 Jul 2024 17:58:29

ZAP Version: 2.15.0

ZAP is supported by the Crash Override Open Source Fellowship

Summary of Alerts

Risk Level	Number of Alerts
High	4
Medium	4
Low	6
Informational	4

Alerts

Name	Risk Level	Number of Instances
Cross Site Scripting (Reflected)	High	7
Path Traversal	High	3
SQL Injection	High	3
SQL Injection - SQLite	High	4
Absence of Anti-CSRF Tokens	Medium	26
Content Security Policy (CSP) Header Not Set	Medium	21
Missing Anti-clickjacking Header	Medium	18
Vulnerable JS Library	Medium	2
Big Redirect Detected (Potential Sensitive Information Leak)	Low	1
Cookie No HttpOnly Flag	Low	1
Cookie without SameSite Attribute	Low	1
Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)	Low	21
Server Leaks Version Information via "Server" HTTP Response Header Field	Low	32
X-Content-Type-Options Header Missing	Low	26
Authentication Request Identified	Informational	2
Information Disclosure - Suspicious Comments	Informational	5
Session Management Response Identified	Informational	6
User Controllable HTML Element Attribute (Potential XSS)	Informational	29

Alert Detail

High	Cross Site Scripting (Reflected)
Description	Cross-site Scripting (XSS) is an attack technique that involves echoing attacker-supplied code into a user's browser instance. A browser instance can be a standard web browser client, or a browser object embedded in a software product such as the browser within WinAmp, an RSS reader, or an email client. The code itself is usually written in HTML /JavaScript, but may also extend to VBScript, ActiveX, Java, Flash, or any other browser-supported technology. When an attacker gets a user's browser to execute his/her code, the code will run within the security context (or zone) of the hosting web site. With this level of privilege, the code has the ability to read, modify and transmit any sensitive data accessible by the browser. A Cross-site Scripted user could have his/her account hijacked (cookie theft), their browser redirected to another location, or possibly shown fraudulent content delivered by the web site they are visiting. Cross-site Scripting attacks essentially compromise the trust relationship between a user and the web site. Applications utilizing browser object instances which load content from the file system may execute code under the local machine zone allowing for system compromise. There are three types of Cross-site Scripting attacks: non-persistent, persistent and DOM-based. Non-persistent attacks and DOM-based attacks require a user to either visit a specially crafted link laced with malicious code, or visit a malicious web page containing a web form, which when posted to the vulnerable site, will mount the attack. Using a malicious form will oftentimes take place when the vulnerable resource only accepts HTTP POST requests. In such a case, the form can be submitted automatically, without the victim's knowledge (e.g. by using JavaScript). Upon clicking on the malicious link or submitting the malicious form, the XSS payload will get echoed back and will get interpreted by the user's browser and execute. Another technique to send almost arbitrary requests (GET and POST) is by using an embedded client, such as Adobe Flash. Persistent attacks occur when the malicious code is submitted to a web site where it's stored for a period of time. Examples of an attacker's favorite targets often include message board posts, web mail messages, and web chat software. The unsuspecting user is not required to interact with any additional site/link (e.g. an attacker site or a malicious link sent via email), just simply view the web page containing the code.
URL	http://127.0.0.1:8080/?file=%3C%2Fh1%3E%3CscrIpt%3Ealert%281%29%3B%3C%2FscRipt%3E%3Ch1%3E&page=documentation&path=.%2F
Method	GET
Attack	</h1><scrIpt>alert(1);</scRipt><h1>
Evidence	</h1><scrIpt>alert(1);</scRipt><h1>
Other Info	
URL	http://127.0.0.1:8080/?file=ATTRIBUTION.md&page=%22%3E%00%3CscrIpt%3Ealert%281%29%3B%3C%2FscRipt%3E&path=.%2F
Method	GET
Attack	">\x0000<scrIpt>alert(1);</scRipt>
Evidence	">\x0000<scrIpt>alert(1);</scRipt>
Other Info	
URL	http://127.0.0.1:8080/?page=%22%3E%3CscrIpt%3Ealert%281%29%3B%3C%2FscRipt%3E
Method	GET
Attack	"><scrIpt>alert(1);</scRipt>
Evidence	"><scrIpt>alert(1);</scRipt>
Other Info	

URL	http://127.0.0.1:8080/?page=%22%3E%3CscrIpt%3Ealert%281%29%3B%3C%2FscRipt%3E&path=.%2FHELP%2F
Method	GET
Attack	"><scrIpt>alert(1);</scRipt>
Evidence	"><scrIpt>alert(1);</scRipt>
Other Info	
URL	http://127.0.0.1:8080/?page=%22%3E%3CscrIpt%3Ealert%281%29%3B%3C%2FscRipt%3E
Method	POST
Attack	"><scrIpt>alert(1);</scRipt>
Evidence	"><scrIpt>alert(1);</scRipt>
Other Info	
URL	http://127.0.0.1:8080/?page=contact
Method	POST
Attack	'"\x0000<scrIpt>alert(1);</scRipt>
Evidence	'"\x0000<scrIpt>alert(1);</scRipt>
Other Info	
URL	http://127.0.0.1:8080/?page=register
Method	POST
Attack	"><img src=x onerror=prompt()>
Evidence	"><img src=x onerror=prompt()>
Other Info	
Instances	7
	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. Examples of libraries and frameworks that make it easier to generate properly encoded output include Microsoft's Anti-XSS library, the OWASP ESAPI Encoding module, and Apache Wicket. Phases: Implementation; Architecture and Design Understand the context in which your data will be used and the encoding that will be expected. This is especially important when transmitting data between different components, or when generating outputs that can contain multiple encodings at the same time, such as web pages or multi-part mail messages. Study all expected communication protocols and data representations to determine the required encoding strategies. For any data that will be output to another web page, especially any data that was received from external inputs, use the appropriate encoding on all non-alphanumeric characters. Consult the XSS Prevention Cheat Sheet for more details on the types of encoding and escaping that are needed. Phase: Architecture and Design For any security checks that are performed on the client side, ensure that these checks are duplicated on the server side, in order to avoid CWE-602. Attackers can bypass the client-

Solution	side checks by modifying values after the checks have been performed, or by changing the client to remove the client-side checks entirely. Then, these modified values would be submitted to the server. If available, use structured mechanisms that automatically enforce the separation between data and code. These mechanisms may be able to provide the relevant quoting, encoding, and validation automatically, instead of relying on the developer to provide this capability at every point where output is generated. Phase: Implementation For every web page that is generated, use and specify a character encoding such as ISO-8859-1 or UTF-8. When an encoding is not specified, the web browser may choose a different encoding by guessing which encoding is actually being used by the web page. This can cause the web browser to treat certain sequences as special, opening up the client to subtle XSS attacks. See CWE-116 for more mitigations related to encoding/escaping. To help mitigate XSS attacks against the user's session cookie, set the session cookie to be HttpOnly. In browsers that support the HttpOnly feature (such as more recent versions of Internet Explorer and Firefox), this attribute can prevent the user's session cookie from being accessible to malicious client-side scripts that use document.cookie. This is not a complete solution, since HttpOnly is not supported by all browsers. More importantly, XMLHTTPRequest and other powerful browser technologies provide read access to HTTP headers, including the Set-Cookie header in which the HttpOnly flag is set. 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." Ensure that you perform input validation at well-defined interfaces within the application. This will help protect the application even if a component is reused or moved elsewhere.
Reference	https://owasp.org/www-community/attacks/xss/ https://cwe.mitre.org/data/definitions/79.html
CWE Id	79
WASC Id	8
Plugin Id	40012

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	http://127.0.0.1:8080/download.php?file=..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2Fetc%2Fpasswd
Method	GET
Attack	../../../../../../../../../../../../../../../../etc/passwd
Evidence	root:x:0:0
Other Info	
URL	http://127.0.0.1:8080/?file=ATTRIBUTION.md&page=documentation&path=..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F
Method	GET
Attack	../../../../../../../../../../../../../../../../
Evidence	etc
Other Info	
URL	http://127.0.0.1:8080/?page=documentation&path=..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F
Method	GET
Attack	../../../../../../../../../../../../../../../../
Evidence	etc
Other Info	
Instances	3
	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 '.'

Solution	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	https://owasp.org/www-community/attacks/Path_Traversal https://cwe.mitre.org/data/definitions/22.html
CWE Id	22
WASC Id	33
Plugin Id	6

High	SQL Injection
Description	SQL injection may be possible.
URL	http://127.0.0.1:8080/?file=ATTRIBUTION.md&page=documentation+AND+1%3D1+--+&path=.%2F
Method	GET
Attack	documentation AND 1=1 --
Evidence	
Other Info	The page results were successfully manipulated using the boolean conditions [documentation AND 1=1 --] and [documentation AND 1=2 --] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was returned for the original parameter. The vulnerability was detected by successfully restricting the data originally returned, by manipulating the parameter
URL	http://127.0.0.1:8080/?page=installation
Method	POST
Attack	AND 1=1 --
Evidence	
Other	The page results were successfully manipulated using the boolean conditions [AND 1=1 --] and [AND 1=2 --] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was returned for the original parameter.

Info	The vulnerability was detected by successfully restricting the data originally returned, by manipulating the parameter
URL	http://127.0.0.1:8080/?page=register
Method	POST
Attack	ZAP' OR '1'='1' --
Evidence	
Other Info	The page results were successfully manipulated using the boolean conditions [ZAP' AND '1'='1' --] and [ZAP' OR '1'='1' --] The parameter value being modified was stripped from the HTML output for the purposes of the comparison Data was NOT returned for the original parameter. The vulnerability was detected by successfully retrieving more data than originally returned, by manipulating the parameter
Instances	3
Solution	Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply an 'allow list' of allowed characters, or a 'deny list' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application.
Reference	https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html
CWE Id	89
WASC Id	19
Plugin Id	40018

High	**SQL Injection - SQLite**
Description	SQL injection may be possible.
URL	http://127.0.0.1:8080/?file=ATTRIBUTION.md&page=documentation&path=./
Method	GET
Attack	case randomblob(10000000) when not null then 1 else 1 end
Evidence	The query time is controllable using parameter value [case randomblob(10000000) when not null then 1 else 1 end], which caused the request to take [11] milliseconds, parameter value [case randomblob(100000000) when not null then 1 else 1 end], which caused the request to take [81] milliseconds, when the original unmodified query with value [ATTRIBUTION.md] took [325] milliseconds.
	The query time is controllable using parameter value [case randomblob(10000000) when

Other Info	not null then 1 else 1 end], which caused the request to take [11] milliseconds, parameter value [case randomblob(100000000) when not null then 1 else 1 end], which caused the request to take [81] milliseconds, when the original unmodified query with value [ATTRIBUTION.md] took [325] milliseconds.
URL	http://127.0.0.1:8080/?page=documentation&path=./HELP/
Method	GET
Attack	case randomblob(100000000) when not null then 1 else 1 end
Evidence	The query time is controllable using parameter value [case randomblob(100000000) when not null then 1 else 1 end], which caused the request to take [471] milliseconds, parameter value [case randomblob(1000000000) when not null then 1 else 1 end], which caused the request to take [3,336] milliseconds, when the original unmodified query with value [documentation] took [626] milliseconds.
Other Info	The query time is controllable using parameter value [case randomblob(100000000) when not null then 1 else 1 end], which caused the request to take [471] milliseconds, parameter value [case randomblob(1000000000) when not null then 1 else 1 end], which caused the request to take [3,336] milliseconds, when the original unmodified query with value [documentation] took [626] milliseconds.
URL	http://127.0.0.1:8080/?page=installation
Method	POST
Attack	case randomblob(10000000) when not null then 1 else 1 end
Evidence	The query time is controllable using parameter value [case randomblob(10000000) when not null then 1 else 1 end], which caused the request to take [787] milliseconds, parameter value [case randomblob(100000000) when not null then 1 else 1 end], which caused the request to take [1,799] milliseconds, when the original unmodified query with value [] took [595] milliseconds.
Other Info	The query time is controllable using parameter value [case randomblob(10000000) when not null then 1 else 1 end], which caused the request to take [787] milliseconds, parameter value [case randomblob(100000000) when not null then 1 else 1 end], which caused the request to take [1,799] milliseconds, when the original unmodified query with value [] took [595] milliseconds.
URL	http://127.0.0.1:8080/?page=register
Method	POST
Attack	case randomblob(100000) when not null then 1 else 1 end
Evidence	The query time is controllable using parameter value [case randomblob(100000) when not null then 1 else 1 end], which caused the request to take [812] milliseconds, parameter value [case randomblob(1000000) when not null then 1 else 1 end], which caused the request to take [865] milliseconds, when the original unmodified query with value [register] took [572] milliseconds.
Other Info	The query time is controllable using parameter value [case randomblob(100000) when not null then 1 else 1 end], which caused the request to take [812] milliseconds, parameter value [case randomblob(1000000) when not null then 1 else 1 end], which caused the request to take [865] milliseconds, when the original unmodified query with value [register] took [572] milliseconds.
Instances	4
	Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them.

Solution	Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply an 'allow list' of allowed characters, or a 'deny list' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application.
Reference	https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html
CWE Id	89
WASC Id	19
Plugin Id	40024

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://127.0.0.1:8080
Method	GET
Attack	
Evidence	<form class="navbar-form navbar-right" role="form" method="post" action="?page=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] was found in the following HTML form: [Form 1: "email" "password"].
URL	http://127.0.0.1:8080/
Method	GET
Attack	
Evidence	<form class="navbar-form navbar-right" role="form" method="post" action="?page=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] was found in the following HTML form: [Form 1: "email" "password"].
URL	http://127.0.0.1:8080/?file=ATTRIBUTION.md&page=documentation&path=./
Method	GET
Attack	
Evidence	<form class="navbar-form navbar-right" role="form" method="post" action="?page=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] was found in the following HTML form: [Form 1: "email" "password"].
URL	http://127.0.0.1:8080/?file=README.md&page=documentation&path=./
Method	GET
Attack	
Evidence	<form method="post" action="?page=installation" class="">
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: ""].
URL	http://127.0.0.1:8080/?file=README.md&page=documentation&path=./
Method	GET
Attack	
Evidence	<form class="navbar-form navbar-right" role="form" method="post" action="?page=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] was found in the following HTML form: [Form 1: "email" "password"].
URL	http://127.0.0.1:8080/?file=RESOURCES.md&page=documentation&path=./HELP/
Method	GET
Attack	
Evidence	<form class="navbar-form navbar-right" role="form" method="post" action="?page=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] was found in the following HTML form: [Form 1: "email" "password"].
URL	http://127.0.0.1:8080/?page=contact
Method	GET
Attack	
Evidence	<form class="navbar-form navbar-right" role="form" method="post" action="?page=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] was found in the following HTML form: [Form 1: "email" "password"].
URL	http://127.0.0.1:8080/?page=contact
Method	GET
Attack	
Evidence	<form action="?page=contact" method="post">

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 2: "email" "name" "recipients[]"].
URL	http://127.0.0.1:8080/?page=documentation
Method	GET
Attack	
Evidence	<form class="navbar-form navbar-right" role="form" method="post" action="?page=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] was found in the following HTML form: [Form 1: "email" "password"].
URL	http://127.0.0.1:8080/?page=documentation&path=./
Method	GET
Attack	
Evidence	<form method="post" action="?page=installation" class="">
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: ""].
URL	http://127.0.0.1:8080/?page=documentation&path=./
Method	GET
Attack	
Evidence	<form class="navbar-form navbar-right" role="form" method="post" action="?page=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] was found in the following HTML form: [Form 1: "email" "password"].
URL	http://127.0.0.1:8080/?page=documentation&path=./HELP/
Method	GET
Attack	
Evidence	<form method="post" action="?page=installation" class="">
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: ""].
URL	http://127.0.0.1:8080/?page=documentation&path=./HELP/
Method	GET
Attack	
Evidence	<form class="navbar-form navbar-right" role="form" method="post" action="?page=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] was found in the following HTML form: [Form 1: "email" "password"].
URL	http://127.0.0.1:8080/?page=downloads
Method	GET
Attack	
Evidence	<form class="navbar-form navbar-right" role="form" method="post" action="?page=login">
	No known Anti-CSRF token [anticsrf, CSRFToken, __RequestVerificationToken,

Other Info	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: "email" "password"].
URL	http://127.0.0.1:8080/?page=installation
Method	GET
Attack	
Evidence	<form class="navbar-form navbar-right" role="form" method="post" action="?page=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] was found in the following HTML form: [Form 1: "email" "password"].
URL	http://127.0.0.1:8080/?page=installation
Method	GET
Attack	
Evidence	<form method="post" action="?page=installation" class=" diwa-reset">
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 2: ""].
URL	http://127.0.0.1:8080/?page=login
Method	GET
Attack	
Evidence	<form class="navbar-form navbar-right" role="form" method="post" action="?page=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] was found in the following HTML form: [Form 1: "email" "password"].
URL	http://127.0.0.1:8080/?page=messagesent
Method	GET
Attack	
Evidence	<form class="navbar-form navbar-right" role="form" method="post" action="?page=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] was found in the following HTML form: [Form 1: "email" "password"].
URL	http://127.0.0.1:8080/?page=register
Method	GET
Attack	
Evidence	<form class="navbar-form navbar-right" role="form" method="post" action="?page=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] was found in the following HTML form: [Form 1: "email" "password"].
URL	http://127.0.0.1:8080/?page=register
Method	GET
Attack	
Evidence	<form method="post" action="?page=register">
Other	No known Anti-CSRF token [anticsrf, CSRFToken, __RequestVerificationToken, csrfmiddlewaretoken, authenticity_token, OWASP_CSRFTOKEN, anoncsrf, csrf_token,

Info	_csrf, _csrfSecret, __csrf_magic, CSRF, _token, _csrf_token] was found in the following HTML form: [Form 2: "email" "invitation-code" "password" "password-repeat" "username"].
URL	http://127.0.0.1:8080/?page=secret-xu2d7a
Method	GET
Attack	
Evidence	<form class="navbar-form navbar-right" role="form" method="post" action="?page=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] was found in the following HTML form: [Form 1: "email" "password"].
URL	http://127.0.0.1:8080/?page=contact
Method	POST
Attack	
Evidence	<form class="navbar-form navbar-right" role="form" method="post" action="?page=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] was found in the following HTML form: [Form 1: "email" "password"].
URL	http://127.0.0.1:8080/?page=installation
Method	POST
Attack	
Evidence	<form class="navbar-form navbar-right" role="form" method="post" action="?page=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] was found in the following HTML form: [Form 1: "email" "password"].
URL	http://127.0.0.1:8080/?page=login
Method	POST
Attack	
Evidence	<form class="navbar-form navbar-right" role="form" method="post" action="?page=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] was found in the following HTML form: [Form 1: "email" "password"].
URL	http://127.0.0.1:8080/?page=register
Method	POST
Attack	
Evidence	<form class="navbar-form navbar-right" role="form" method="post" action="?page=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] was found in the following HTML form: [Form 1: "email" "password"].
URL	http://127.0.0.1:8080/?page=register
Method	POST
Attack	
Evidence	<form method="post" action="?page=register">
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 2: "email" "invitation-code" "password" "password-repeat" "username"].
Instances	26
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	https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html https://cwe.mitre.org/data/definitions/352.html
CWE Id	352
WASC Id	9
Plugin Id	10202

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	http://127.0.0.1:8080
Method	GET
Attack	
Evidence	
Other Info	
URL	http://127.0.0.1:8080/

Method	GET
Attack	
Evidence	
Other Info	
URL	http://127.0.0.1:8080/?file=ATTRIBUTION.md&page=documentation&path=./
Method	GET
Attack	
Evidence	
Other Info	
URL	http://127.0.0.1:8080/?file=README.md&page=documentation&path=./
Method	GET
Attack	
Evidence	
Other Info	
URL	http://127.0.0.1:8080/?file=RESOURCES.md&page=documentation&path=./HELP/
Method	GET
Attack	
Evidence	
Other Info	
URL	http://127.0.0.1:8080/?page=contact
Method	GET
Attack	
Evidence	
Other Info	
URL	http://127.0.0.1:8080/?page=documentation
Method	GET
Attack	
Evidence	
Other Info	
URL	http://127.0.0.1:8080/?page=documentation&path=./
Method	GET
Attack	
Evidence	
Other Info	
URL	http://127.0.0.1:8080/?page=documentation&path=./HELP/
Method	GET

Attack	
Evidence	
Other Info	
URL	http://127.0.0.1:8080/?page=downloads
Method	GET
Attack	
Evidence	
Other Info	
URL	http://127.0.0.1:8080/?page=installation
Method	GET
Attack	
Evidence	
Other Info	
URL	http://127.0.0.1:8080/?page=login
Method	GET
Attack	
Evidence	
Other Info	
URL	http://127.0.0.1:8080/?page=messagesent
Method	GET
Attack	
Evidence	
Other Info	
URL	http://127.0.0.1:8080/?page=register
Method	GET
Attack	
Evidence	
Other Info	
URL	http://127.0.0.1:8080/?page=secret-xu2d7a
Method	GET
Attack	
Evidence	
Other Info	
URL	http://127.0.0.1:8080/css/
Method	GET
Attack	
Evidence	

Other Info	
URL	http://127.0.0.1:8080/js/
Method	GET
Attack	
Evidence	
Other Info	
URL	http://127.0.0.1:8080/sitemap.xml
Method	GET
Attack	
Evidence	
Other Info	
URL	http://127.0.0.1:8080/?page=installation
Method	POST
Attack	
Evidence	
Other Info	
URL	http://127.0.0.1:8080/?page=login
Method	POST
Attack	
Evidence	
Other Info	
URL	http://127.0.0.1:8080/?page=register
Method	POST
Attack	
Evidence	
Other Info	
Instances	21
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 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/
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	http://127.0.0.1:8080
Method	GET
Attack	
Evidence	
Other Info	
URL	http://127.0.0.1:8080/
Method	GET
Attack	
Evidence	
Other Info	
URL	http://127.0.0.1:8080/?file=ATTRIBUTION.md&page=documentation&path=./
Method	GET
Attack	
Evidence	
Other Info	
URL	http://127.0.0.1:8080/?file=README.md&page=documentation&path=./
Method	GET
Attack	
Evidence	
Other Info	
URL	http://127.0.0.1:8080/?file=RESOURCES.md&page=documentation&path=./HELP/
Method	GET
Attack	
Evidence	
Other Info	
URL	http://127.0.0.1:8080/?page=contact
Method	GET
Attack	
Evidence	
Other Info	
URL	http://127.0.0.1:8080/?page=documentation
Method	GET
Attack	
Evidence	

Other Info	
URL	http://127.0.0.1:8080/?page=documentation&path=./
Method	GET
Attack	
Evidence	
Other Info	
URL	http://127.0.0.1:8080/?page=documentation&path=./HELP/
Method	GET
Attack	
Evidence	
Other Info	
URL	http://127.0.0.1:8080/?page=downloads
Method	GET
Attack	
Evidence	
Other Info	
URL	http://127.0.0.1:8080/?page=installation
Method	GET
Attack	
Evidence	
Other Info	
URL	http://127.0.0.1:8080/?page=login
Method	GET
Attack	
Evidence	
Other Info	
URL	http://127.0.0.1:8080/?page=messagesent
Method	GET
Attack	
Evidence	
Other Info	
URL	http://127.0.0.1:8080/?page=register
Method	GET
Attack	
Evidence	
Other Info	

URL	http://127.0.0.1:8080/?page=secret-xu2d7a
Method	GET
Attack	
Evidence	
Other Info	
URL	http://127.0.0.1:8080/?page=installation
Method	POST
Attack	
Evidence	
Other Info	
URL	http://127.0.0.1:8080/?page=login
Method	POST
Attack	
Evidence	
Other Info	
URL	http://127.0.0.1:8080/?page=register
Method	POST
Attack	
Evidence	
Other Info	
Instances	18
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

Medium	**Vulnerable JS Library**
Description	The identified library bootstrap, version 3.3.7 is vulnerable.
URL	http://127.0.0.1:8080/js/bootstrap.min.js
Method	GET
Attack	
Evidence	* Bootstrap v3.3.7
Other Info	CVE-2018-14041 CVE-2019-8331 CVE-2018-20677 CVE-2018-20676 CVE-2018-14042 CVE-2016-10735
URL	http://127.0.0.1:8080/js/jquery.min.js

Method	GET
Attack	
Evidence	/*! jQuery v3.2.1
Other Info	CVE-2020-11023 CVE-2020-11022 CVE-2019-11358
Instances	2
Solution	Please upgrade to the latest version of bootstrap.
Reference	https://github.com/twbs/bootstrap/issues/28236 https://github.com/advisories/GHSA-pj7m-g53m-7638 https://github.com/twbs/bootstrap/issues/20184 https://github.com/advisories/GHSA-ph58-4vrj-w6hr https://github.com/twbs/bootstrap/issues/20631 https://github.com/advisories/GHSA-4p24-vmcr-4gqj https://github.com/advisories/GHSA-9v3m-8fp8-mj99 https://nvd.nist.gov/vuln/detail/CVE-2018-20676
CWE Id	829
WASC Id	
Plugin Id	10003

Low	**Big Redirect Detected (Potential Sensitive Information Leak)**
Description	The server has responded with a redirect that seems to provide a large response. This may indicate that although the server sent a redirect it also responded with body content (which may include sensitive details, PII, etc.).
URL	http://127.0.0.1:8080/?page=contact
Method	POST
Attack	
Evidence	
Other Info	Location header URI length: 17 [?page=messagesent]. Predicted response size: 317. Response Body Length: 2,084.
Instances	1
Solution	Ensure that no sensitive information is leaked via redirect responses. Redirect responses should have almost no content.
Reference	
CWE Id	201
WASC Id	13
Plugin Id	10044

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	http://127.0.0.1:8080
Method	GET
Attack	
Evidence	Set-Cookie: PHPSESSID
Other Info	
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 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://127.0.0.1:8080
Method	GET
Attack	
Evidence	Set-Cookie: PHPSESSID
Other Info	
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	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	http://127.0.0.1:8080
Method	GET
Attack	
Evidence	X-Powered-By: PHP/7.4.33
Other Info	
URL	http://127.0.0.1:8080/
Method	GET
Attack	
Evidence	X-Powered-By: PHP/7.4.33
Other Info	
URL	http://127.0.0.1:8080/?file=ATTRIBUTION.md&page=documentation&path=./
Method	GET
Attack	
Evidence	X-Powered-By: PHP/7.4.33
Other Info	
URL	http://127.0.0.1:8080/?file=README.md&page=documentation&path=./

Method	GET
Attack	
Evidence	X-Powered-By: PHP/7.4.33
Other Info	
URL	http://127.0.0.1:8080/?file=RESOURCES.md&page=documentation&path=./HELP/
Method	GET
Attack	
Evidence	X-Powered-By: PHP/7.4.33
Other Info	
URL	http://127.0.0.1:8080/?page=contact
Method	GET
Attack	
Evidence	X-Powered-By: PHP/7.4.33
Other Info	
URL	http://127.0.0.1:8080/?page=documentation
Method	GET
Attack	
Evidence	X-Powered-By: PHP/7.4.33
Other Info	
URL	http://127.0.0.1:8080/?page=documentation&path=./
Method	GET
Attack	
Evidence	X-Powered-By: PHP/7.4.33
Other Info	
URL	http://127.0.0.1:8080/?page=documentation&path=./HELP/
Method	GET
Attack	
Evidence	X-Powered-By: PHP/7.4.33
Other Info	
URL	http://127.0.0.1:8080/?page=downloads
Method	GET
Attack	
Evidence	X-Powered-By: PHP/7.4.33
Other Info	
URL	http://127.0.0.1:8080/?page=installation
Method	GET
Attack	

Evidence	X-Powered-By: PHP/7.4.33
Other Info	
URL	http://127.0.0.1:8080/?page=login
Method	GET
Attack	
Evidence	X-Powered-By: PHP/7.4.33
Other Info	
URL	http://127.0.0.1:8080/?page=messagesent
Method	GET
Attack	
Evidence	X-Powered-By: PHP/7.4.33
Other Info	
URL	http://127.0.0.1:8080/?page=register
Method	GET
Attack	
Evidence	X-Powered-By: PHP/7.4.33
Other Info	
URL	http://127.0.0.1:8080/?page=secret-xu2d7a
Method	GET
Attack	
Evidence	X-Powered-By: PHP/7.4.33
Other Info	
URL	http://127.0.0.1:8080/download.php?file=lorem-ipsum-1000-words.txt
Method	GET
Attack	
Evidence	X-Powered-By: PHP/7.4.33
Other Info	
URL	http://127.0.0.1:8080/download.php?file=lorem-ipsum-10000-words.txt
Method	GET
Attack	
Evidence	X-Powered-By: PHP/7.4.33
Other Info	
URL	http://127.0.0.1:8080/?page=contact
Method	POST
Attack	
Evidence	X-Powered-By: PHP/7.4.33

Other Info	
URL	http://127.0.0.1:8080/?page=installation
Method	POST
Attack	
Evidence	X-Powered-By: PHP/7.4.33
Other Info	
URL	http://127.0.0.1:8080/?page=login
Method	POST
Attack	
Evidence	X-Powered-By: PHP/7.4.33
Other Info	
URL	http://127.0.0.1:8080/?page=register
Method	POST
Attack	
Evidence	X-Powered-By: PHP/7.4.33
Other Info	
Instances	21
Solution	Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.
Reference	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/2012/02/shhh-dont-let-your-response-headers.html
CWE Id	200
WASC Id	13
Plugin Id	10037

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	http://127.0.0.1:8080
Method	GET
Attack	
Evidence	Apache/2.4.54 (Debian)
Other Info	
URL	http://127.0.0.1:8080/
Method	GET
Attack	
Evidence	Apache/2.4.54 (Debian)
Other Info	

URL	http://127.0.0.1:8080/?file=ATTRIBUTION.md&page=documentation&path=./
Method	GET
Attack	
Evidence	Apache/2.4.54 (Debian)
Other Info	
URL	http://127.0.0.1:8080/?file=README.md&page=documentation&path=./
Method	GET
Attack	
Evidence	Apache/2.4.54 (Debian)
Other Info	
URL	http://127.0.0.1:8080/?file=RESOURCES.md&page=documentation&path=./HELP/
Method	GET
Attack	
Evidence	Apache/2.4.54 (Debian)
Other Info	
URL	http://127.0.0.1:8080/?page=contact
Method	GET
Attack	
Evidence	Apache/2.4.54 (Debian)
Other Info	
URL	http://127.0.0.1:8080/?page=documentation
Method	GET
Attack	
Evidence	Apache/2.4.54 (Debian)
Other Info	
URL	http://127.0.0.1:8080/?page=documentation&path=./
Method	GET
Attack	
Evidence	Apache/2.4.54 (Debian)
Other Info	
URL	http://127.0.0.1:8080/?page=documentation&path=./HELP/
Method	GET
Attack	
Evidence	Apache/2.4.54 (Debian)
Other Info	
URL	http://127.0.0.1:8080/?page=downloads
Method	GET

Attack	
Evidence	Apache/2.4.54 (Debian)
Other Info	
URL	http://127.0.0.1:8080/?page=installation
Method	GET
Attack	
Evidence	Apache/2.4.54 (Debian)
Other Info	
URL	http://127.0.0.1:8080/?page=login
Method	GET
Attack	
Evidence	Apache/2.4.54 (Debian)
Other Info	
URL	http://127.0.0.1:8080/?page=messagesent
Method	GET
Attack	
Evidence	Apache/2.4.54 (Debian)
Other Info	
URL	http://127.0.0.1:8080/?page=register
Method	GET
Attack	
Evidence	Apache/2.4.54 (Debian)
Other Info	
URL	http://127.0.0.1:8080/?page=secret-xu2d7a
Method	GET
Attack	
Evidence	Apache/2.4.54 (Debian)
Other Info	
URL	http://127.0.0.1:8080/css
Method	GET
Attack	
Evidence	Apache/2.4.54 (Debian)
Other Info	
URL	http://127.0.0.1:8080/css/
Method	GET
Attack	

Evidence	Apache/2.4.54 (Debian)
Other Info	
URL	http://127.0.0.1:8080/css/bootstrap.min.css
Method	GET
Attack	
Evidence	Apache/2.4.54 (Debian)
Other Info	
URL	http://127.0.0.1:8080/css/font-awesome.min.css
Method	GET
Attack	
Evidence	Apache/2.4.54 (Debian)
Other Info	
URL	http://127.0.0.1:8080/download.php?file=lorem-ipsum-1000-words.txt
Method	GET
Attack	
Evidence	Apache/2.4.54 (Debian)
Other Info	
URL	http://127.0.0.1:8080/download.php?file=lorem-ipsum-10000-words.txt
Method	GET
Attack	
Evidence	Apache/2.4.54 (Debian)
Other Info	
URL	http://127.0.0.1:8080/js
Method	GET
Attack	
Evidence	Apache/2.4.54 (Debian)
Other Info	
URL	http://127.0.0.1:8080/js/
Method	GET
Attack	
Evidence	Apache/2.4.54 (Debian)
Other Info	
URL	http://127.0.0.1:8080/js/bootstrap.min.js
Method	GET
Attack	
Evidence	Apache/2.4.54 (Debian)
Other	

Info	
URL	http://127.0.0.1:8080/js/jquery.min.js
Method	GET
Attack	
Evidence	Apache/2.4.54 (Debian)
Other Info	
URL	http://127.0.0.1:8080/js/main.js
Method	GET
Attack	
Evidence	Apache/2.4.54 (Debian)
Other Info	
URL	http://127.0.0.1:8080/robots.txt
Method	GET
Attack	
Evidence	Apache/2.4.54 (Debian)
Other Info	
URL	http://127.0.0.1:8080/sitemap.xml
Method	GET
Attack	
Evidence	Apache/2.4.54 (Debian)
Other Info	
URL	http://127.0.0.1:8080/?page=contact
Method	POST
Attack	
Evidence	Apache/2.4.54 (Debian)
Other Info	
URL	http://127.0.0.1:8080/?page=installation
Method	POST
Attack	
Evidence	Apache/2.4.54 (Debian)
Other Info	
URL	http://127.0.0.1:8080/?page=login
Method	POST
Attack	
Evidence	Apache/2.4.54 (Debian)
Other Info	
URL	http://127.0.0.1:8080/?page=register

Method	POST
Attack	
Evidence	Apache/2.4.54 (Debian)
Other Info	
Instances	32
Solution	Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.
Reference	https://httpd.apache.org/docs/current/mod/core.html#servertokens https://learn.microsoft.com/en-us/previous-versions/msp-n-p/ff648552(v=pandp.10) https://www.troyhunt.com/shhh-dont-let-your-response-headers/
CWE Id	200
WASC Id	13
Plugin Id	10036

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	http://127.0.0.1:8080
Method	GET
Attack	
Evidence	
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.
URL	http://127.0.0.1:8080/
Method	GET
Attack	
Evidence	
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.
URL	http://127.0.0.1:8080/?file=ATTRIBUTION.md&page=documentation&path=./
Method	GET
Attack	
Evidence	
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.
URL	http://127.0.0.1:8080/?file=README.md&page=documentation&path=./
Method	GET
Attack	

Evidence	
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.
URL	http://127.0.0.1:8080/?file=RESOURCES.md&page=documentation&path=./HELP/
Method	GET
Attack	
Evidence	
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.
URL	http://127.0.0.1:8080/?page=contact
Method	GET
Attack	
Evidence	
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.
URL	http://127.0.0.1:8080/?page=documentation
Method	GET
Attack	
Evidence	
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.
URL	http://127.0.0.1:8080/?page=documentation&path=./
Method	GET
Attack	
Evidence	
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.
URL	http://127.0.0.1:8080/?page=documentation&path=./HELP/
Method	GET
Attack	
Evidence	
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.
URL	http://127.0.0.1:8080/?page=downloads
Method	GET
Attack	
Evidence	

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.
URL	http://127.0.0.1:8080/?page=installation
Method	GET
Attack	
Evidence	
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.
URL	http://127.0.0.1:8080/?page=login
Method	GET
Attack	
Evidence	
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.
URL	http://127.0.0.1:8080/?page=messagesent
Method	GET
Attack	
Evidence	
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.
URL	http://127.0.0.1:8080/?page=register
Method	GET
Attack	
Evidence	
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.
URL	http://127.0.0.1:8080/?page=secret-xu2d7a
Method	GET
Attack	
Evidence	
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.
URL	http://127.0.0.1:8080/css/bootstrap.min.css
Method	GET
Attack	
Evidence	

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.
URL	http://127.0.0.1:8080/css/font-awesome.min.css
Method	GET
Attack	
Evidence	
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.
URL	http://127.0.0.1:8080/download.php?file=lorem-ipsum-1000-words.txt
Method	GET
Attack	
Evidence	
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.
URL	http://127.0.0.1:8080/download.php?file=lorem-ipsum-10000-words.txt
Method	GET
Attack	
Evidence	
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.
URL	http://127.0.0.1:8080/js/bootstrap.min.js
Method	GET
Attack	
Evidence	
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.
URL	http://127.0.0.1:8080/js/jquery.min.js
Method	GET
Attack	
Evidence	
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.
URL	http://127.0.0.1:8080/js/main.js
Method	GET
Attack	
Evidence	
	This issue still applies to error type pages (401, 403, 500, etc.) as those pages are often still

Other Info	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.
URL	http://127.0.0.1:8080/robots.txt
Method	GET
Attack	
Evidence	
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.
URL	http://127.0.0.1:8080/?page=installation
Method	POST
Attack	
Evidence	
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.
URL	http://127.0.0.1:8080/?page=login
Method	POST
Attack	
Evidence	
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.
URL	http://127.0.0.1:8080/?page=register
Method	POST
Attack	
Evidence	
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.
Instances	26
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	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
CWE Id	693
WASC Id	15
Plugin Id	10021

Informational	**Authentication Request Identified**

Description	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.
URL	http://127.0.0.1:8080/?page=login
Method	POST
Attack	
Evidence	password
Other Info	userParam=email userValue=zaproxy@example.com passwordParam=password referer=http://127.0.0.1:8080
URL	http://127.0.0.1:8080/?page=login
Method	POST
Attack	
Evidence	password
Other Info	userParam=email userValue=zaproxy@example.com passwordParam=password referer=http://127.0.0.1:8080/
Instances	2
Solution	This is an informational alert rather than a vulnerability and so there is nothing to fix.
Reference	https://www.zaproxy.org/docs/desktop/addons/authentication-helper/auth-req-id/
CWE Id	
WASC Id	
Plugin Id	10111

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	http://127.0.0.1:8080/js/jquery.min.js
Method	GET
Attack	
Evidence	db
Other Info	The following pattern was used: \bDB\b and was detected 2 times, the first in the element starting with: " a.removeEventListener("load",S),r.ready()}"complete"===d.readyState||" loading"!==d.readyState&&!d.documentElement.doScroll?a", see evidence field for the suspicious comment/snippet.
URL	http://127.0.0.1:8080/js/jquery.min.js
Method	GET
Attack	
Evidence	select
Other Info	The following pattern was used: \bSELECT\b and was detected in the element starting with: "!function(a,b){"use strict";"object"==typeof module&&"object"==typeof module.exports?module.exports=a.document?b(a,!0):function(", see evidence field for the suspicious comment/snippet.
URL	http://127.0.0.1:8080/js/main.js
Method	GET
Attack	
Evidence	admin
Other	The following pattern was used: \bADMIN\b and was detected 2 times, the first in the

Info	element starting with: " $('.select-admin').prop('checked', true);", see evidence field for the suspicious comment/snippet.
URL	http://127.0.0.1:8080/js/main.js
Method	GET
Attack	
Evidence	select
Other Info	The following pattern was used: \bSELECT\b and was detected in the element starting with: " $('.select-all-admins').click(function () {", see evidence field for the suspicious comment /snippet.
URL	http://127.0.0.1:8080/js/main.js
Method	GET
Attack	
Evidence	user
Other Info	The following pattern was used: \bUSER\b and was detected 2 times, the first in the element starting with: " $('.remove-user').click(function () {", see evidence field for the suspicious comment/snippet.
Instances	5
Solution	Remove all comments that return information that may help an attacker and fix any underlying problems they refer to.
Reference	
CWE Id	[200](200)
WASC Id	13
Plugin Id	[10027](10027)

Informational	**Session Management Response Identified**
Description	The given response has been identified as containing a session management token. The 'Other Info' field contains a set of header tokens that can be used in the Header Based Session Management Method. If the request is in a context which has a Session Management Method set to "Auto-Detect" then this rule will change the session management to use the tokens identified.
URL	http://127.0.0.1:8080
Method	GET
Attack	
Evidence	4770a94265f84995ce5a293b6549668b
Other Info	cookie:PHPSESSID
URL	http://127.0.0.1:8080
Method	GET
Attack	
Evidence	503e3a88fce45aeba51112adf465fab0
Other Info	cookie:PHPSESSID
URL	http://127.0.0.1:8080
Method	GET
Attack	
Evidence	b9f903e70530e72ca91e7f7beb80ea53
Other	

Info	cookie:PHPSESSID
URL	http://127.0.0.1:8080
Method	GET
Attack	
Evidence	e99b87ad0a60a825ce6ee0f964c69406
Other Info	cookie:PHPSESSID
URL	http://127.0.0.1:8080/?page=messagesent
Method	GET
Attack	
Evidence	4770a94265f84995ce5a293b6549668b
Other Info	cookie:PHPSESSID
URL	http://127.0.0.1:8080/?page=register
Method	POST
Attack	
Evidence	b9f903e70530e72ca91e7f7beb80ea53
Other Info	cookie:PHPSESSID
Instances	6
Solution	This is an informational alert rather than a vulnerability and so there is nothing to fix.
Reference	https://www.zaproxy.org/docs/desktop/addons/authentication-helper/session-mgmt-id
CWE Id	
WASC Id	
Plugin Id	10112

Informational	**User Controllable HTML Element Attribute (Potential XSS)**
Description	This check looks at user-supplied input in query string parameters and POST data to identify where certain HTML attribute values might be controlled. This provides hot-spot detection for XSS (cross-site scripting) that will require further review by a security analyst to determine exploitability.
URL	http://127.0.0.1:8080/?file=ATTRIBUTION.md&page=documentation&path=./
Method	GET
Attack	
Evidence	
Other Info	User-controlled HTML attribute values were found. Try injecting special characters to see if XSS might be possible. The page at the following URL: http://127.0.0.1:8080/?file=ATTRIBUTION.md&page=documentation&path=./ appears to include user input in: a(n) [body] tag [class] attribute The user input found was: page=documentation The user-controlled value was: documentation
URL	http://127.0.0.1:8080/?file=ATTRIBUTION.md&page=documentation&path=./
Method	GET
Attack	
Evidence	
Other Info	User-controlled HTML attribute values were found. Try injecting special characters to see if XSS might be possible. The page at the following URL: http://127.0.0.1:8080/?file=ATTRIBUTION.md&page=documentation&path=./ appears to include user input in: a(n)

	[a] tag [href] attribute The user input found was: path=./ The user-controlled value was: ./
URL	http://127.0.0.1:8080/?file=README.md&page=documentation&path=./
Method	GET
Attack	
Evidence	
Other Info	User-controlled HTML attribute values were found. Try injecting special characters to see if XSS might be possible. The page at the following URL: http://127.0.0.1:8080/?file=README.md&page=documentation&path=./ appears to include user input in: a(n) [body] tag [class] attribute The user input found was: page=documentation The user-controlled value was: documentation
URL	http://127.0.0.1:8080/?file=README.md&page=documentation&path=./
Method	GET
Attack	
Evidence	
Other Info	User-controlled HTML attribute values were found. Try injecting special characters to see if XSS might be possible. The page at the following URL: http://127.0.0.1:8080/?file=README.md&page=documentation&path=./ appears to include user input in: a(n) [a] tag [href] attribute The user input found was: path=./ The user-controlled value was: ./
URL	http://127.0.0.1:8080/?file=RESOURCES.md&page=documentation&path=./HELP/
Method	GET
Attack	
Evidence	
Other Info	User-controlled HTML attribute values were found. Try injecting special characters to see if XSS might be possible. The page at the following URL: http://127.0.0.1:8080/?file=RESOURCES.md&page=documentation&path=./HELP/ appears to include user input in: a(n) [body] tag [class] attribute The user input found was: page=documentation The user-controlled value was: documentation
URL	http://127.0.0.1:8080/?page=contact
Method	GET
Attack	
Evidence	
Other Info	User-controlled HTML attribute values were found. Try injecting special characters to see if XSS might be possible. The page at the following URL: http://127.0.0.1:8080/?page=contact appears to include user input in: a(n) [body] tag [class] attribute The user input found was: page=contact The user-controlled value was: contact
URL	http://127.0.0.1:8080/?page=documentation
Method	GET
Attack	
Evidence	
Other Info	User-controlled HTML attribute values were found. Try injecting special characters to see if XSS might be possible. The page at the following URL: http://127.0.0.1:8080/?page=documentation appears to include user input in: a(n) [body] tag [class] attribute The user input found was: page=documentation The user-controlled value was: documentation
URL	http://127.0.0.1:8080/?page=documentation&path=./
Method	GET
Attack	
Evidence	
	User-controlled HTML attribute values were found. Try injecting special characters to see if XSS might be possible. The page at the following URL: http://127.0.0.1:8080/?

Other Info	page=documentation&path=./ appears to include user input in: a(n) [body] tag [class] attribute The user input found was: page=documentation The user-controlled value was: documentation
URL	http://127.0.0.1:8080/?page=documentation&path=./
Method	GET
Attack	
Evidence	
Other Info	User-controlled HTML attribute values were found. Try injecting special characters to see if XSS might be possible. The page at the following URL: http://127.0.0.1:8080/?page=documentation&path=./ appears to include user input in: a(n) [a] tag [href] attribute The user input found was: path=./ The user-controlled value was: ./
URL	http://127.0.0.1:8080/?page=documentation&path=./HELP/
Method	GET
Attack	
Evidence	
Other Info	User-controlled HTML attribute values were found. Try injecting special characters to see if XSS might be possible. The page at the following URL: http://127.0.0.1:8080/?page=documentation&path=./HELP/ appears to include user input in: a(n) [body] tag [class] attribute The user input found was: page=documentation The user-controlled value was: documentation
URL	http://127.0.0.1:8080/?page=downloads
Method	GET
Attack	
Evidence	
Other Info	User-controlled HTML attribute values were found. Try injecting special characters to see if XSS might be possible. The page at the following URL: http://127.0.0.1:8080/?page=downloads appears to include user input in: a(n) [body] tag [class] attribute The user input found was: page=downloads The user-controlled value was: downloads
URL	http://127.0.0.1:8080/?page=installation
Method	GET
Attack	
Evidence	
Other Info	User-controlled HTML attribute values were found. Try injecting special characters to see if XSS might be possible. The page at the following URL: http://127.0.0.1:8080/?page=installation appears to include user input in: a(n) [body] tag [class] attribute The user input found was: page=installation The user-controlled value was: installation
URL	http://127.0.0.1:8080/?page=login
Method	GET
Attack	
Evidence	
Other Info	User-controlled HTML attribute values were found. Try injecting special characters to see if XSS might be possible. The page at the following URL: http://127.0.0.1:8080/?page=login appears to include user input in: a(n) [body] tag [class] attribute The user input found was: page=login The user-controlled value was: login
URL	http://127.0.0.1:8080/?page=messagesent
Method	GET
Attack	
Evidence	
	User-controlled HTML attribute values were found. Try injecting special characters to see if

Other Info	XSS might be possible. The page at the following URL: http://127.0.0.1:8080/?page=messagesent appears to include user input in: a(n) [body] tag [class] attribute The user input found was: page=messagesent The user-controlled value was: messagesent
URL	http://127.0.0.1:8080/?page=register
Method	GET
Attack	
Evidence	
Other Info	User-controlled HTML attribute values were found. Try injecting special characters to see if XSS might be possible. The page at the following URL: http://127.0.0.1:8080/?page=register appears to include user input in: a(n) [body] tag [class] attribute The user input found was: page=register The user-controlled value was: register
URL	http://127.0.0.1:8080/?page=secret-xu2d7a
Method	GET
Attack	
Evidence	
Other Info	User-controlled HTML attribute values were found. Try injecting special characters to see if XSS might be possible. The page at the following URL: http://127.0.0.1:8080/?page=secret-xu2d7a appears to include user input in: a(n) [body] tag [class] attribute The user input found was: page=secret-xu2d7a The user-controlled value was: secret-xu2d7a
URL	http://127.0.0.1:8080/?page=installation
Method	POST
Attack	
Evidence	
Other Info	User-controlled HTML attribute values were found. Try injecting special characters to see if XSS might be possible. The page at the following URL: http://127.0.0.1:8080/?page=installation appears to include user input in: a(n) [body] tag [class] attribute The user input found was: page=installation The user-controlled value was: installation
URL	http://127.0.0.1:8080/?page=login
Method	POST
Attack	
Evidence	
Other Info	User-controlled HTML attribute values were found. Try injecting special characters to see if XSS might be possible. The page at the following URL: http://127.0.0.1:8080/?page=login appears to include user input in: a(n) [body] tag [class] attribute The user input found was: page=login The user-controlled value was: login
URL	http://127.0.0.1:8080/?page=register
Method	POST
Attack	
Evidence	
Other Info	User-controlled HTML attribute values were found. Try injecting special characters to see if XSS might be possible. The page at the following URL: http://127.0.0.1:8080/?page=register appears to include user input in: a(n) [option] tag [value] attribute The user input found was: country=Albania The user-controlled value was: albania
URL	http://127.0.0.1:8080/?page=register
Method	POST
Attack	
Evidence	
	User-controlled HTML attribute values were found. Try injecting special characters to see if XSS might be possible. The page at the following URL: http://127.0.0.1:8080/?

Other Info	page=register appears to include user input in: a(n) [input] tag [value] attribute The user input found was: email=zaproxy@example.com The user-controlled value was: zaproxy@example.com
URL	http://127.0.0.1:8080/?page=register
Method	POST
Attack	
Evidence	
Other Info	User-controlled HTML attribute values were found. Try injecting special characters to see if XSS might be possible. The page at the following URL: http://127.0.0.1:8080/?page=register appears to include user input in: a(n) [input] tag [value] attribute The user input found was: invitation-code=ZAP The user-controlled value was: zap
URL	http://127.0.0.1:8080/?page=register
Method	POST
Attack	
Evidence	
Other Info	User-controlled HTML attribute values were found. Try injecting special characters to see if XSS might be possible. The page at the following URL: http://127.0.0.1:8080/?page=register appears to include user input in: a(n) [input] tag [value] attribute The user input found was: invitation-code=ZAP The user-controlled value was: zaproxy@example.com
URL	http://127.0.0.1:8080/?page=register
Method	POST
Attack	
Evidence	
Other Info	User-controlled HTML attribute values were found. Try injecting special characters to see if XSS might be possible. The page at the following URL: http://127.0.0.1:8080/?page=register appears to include user input in: a(n) [body] tag [class] attribute The user input found was: page=register The user-controlled value was: register
URL	http://127.0.0.1:8080/?page=register
Method	POST
Attack	
Evidence	
Other Info	User-controlled HTML attribute values were found. Try injecting special characters to see if XSS might be possible. The page at the following URL: http://127.0.0.1:8080/?page=register appears to include user input in: a(n) [input] tag [value] attribute The user input found was: password=ZAP The user-controlled value was: zap
URL	http://127.0.0.1:8080/?page=register
Method	POST
Attack	
Evidence	
Other Info	User-controlled HTML attribute values were found. Try injecting special characters to see if XSS might be possible. The page at the following URL: http://127.0.0.1:8080/?page=register appears to include user input in: a(n) [input] tag [value] attribute The user input found was: password=ZAP The user-controlled value was: zaproxy@example.com
URL	http://127.0.0.1:8080/?page=register
Method	POST
Attack	
Evidence	
	User-controlled HTML attribute values were found. Try injecting special characters to see if

Other Info	XSS might be possible. The page at the following URL: http://127.0.0.1:8080/?page=register appears to include user input in: a(n) [input] tag [value] attribute The user input found was: password-repeat=ZAP The user-controlled value was: zap
URL	http://127.0.0.1:8080/?page=register
Method	POST
Attack	
Evidence	
Other Info	User-controlled HTML attribute values were found. Try injecting special characters to see if XSS might be possible. The page at the following URL: http://127.0.0.1:8080/?page=register appears to include user input in: a(n) [input] tag [value] attribute The user input found was: password-repeat=ZAP The user-controlled value was: zaproxy@example.com
URL	http://127.0.0.1:8080/?page=register
Method	POST
Attack	
Evidence	
Other Info	User-controlled HTML attribute values were found. Try injecting special characters to see if XSS might be possible. The page at the following URL: http://127.0.0.1:8080/?page=register appears to include user input in: a(n) [input] tag [value] attribute The user input found was: username=ZAP The user-controlled value was: zap
URL	http://127.0.0.1:8080/?page=register
Method	POST
Attack	
Evidence	
Other Info	User-controlled HTML attribute values were found. Try injecting special characters to see if XSS might be possible. The page at the following URL: http://127.0.0.1:8080/?page=register appears to include user input in: a(n) [input] tag [value] attribute The user input found was: username=ZAP The user-controlled value was: zaproxy@example.com
Instances	29
Solution	Validate all input and sanitize output it before writing to any HTML attributes.
Reference	https://cheatsheetseries.owasp.org/cheatsheets/Input_Validation_Cheat_Sheet.html
CWE Id	20
WASC Id	20
Plugin Id	10031