API Scanner Report

✔ http://rest.testsparker.com/jwt

Summary

Overall risk level:

High

Risk ratings:

Scan information:

Start time:	2023-03-21 12:20:21 UTC+02
Finish time:	2023-03-21 12:34:26 UTC+02
Scan duration:	14 min, 5 sec
Tests performed:	44/44
Scan status:	Finished

Findings

SQL Injection

CONFIRMED

URL	Method	Parameters	Evidence	Replay Attack
http://rest.testsparker.com/jwt/api/users/{username}	GET	Headers: Authorization=Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiIsImtpZCI6InNlY3JldC50eHQifQ.eyJ1c2VyIjoidGVzdCJ9.jqBFzyBB68KWiOvEJhcaDgMY0Gea-t0KNnf-fR2Ioyc	Injecting the value `'` in the URL path generated the following error(s) in the response: `SQL errorPDOException: SQLSTATE[42000]: Syntax error or access violation: 1064 You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near ''test_username''' at line 1 in /var/www/src/routes/users.php:67`	
http://rest.testsparker.com/jwt/api/users/'	GET	Headers: Authorization=Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiIsImtpZCI6InNlY3JldC50eHQifQ.eyJ1c2VyIjoidGVzdCJ9.jqBFzyBB68KWiOvEJhcaDgMY0Gea-t0KNnf-fR2Ioyc	Injecting the value `'` in the URL path generated the following error(s) in the response: `SQL errorPDOException: SQLSTATE[42000]: Syntax error or access violation: 1064 You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near ''''' at line 1 in /var/www/src/routes/users.php:67`	

Details

Risk description:
We found that the web application is vulnerable to SQL Injection attacks.
SQL Injection is a vulnerability caused by improper input sanitization and allows an attacker to inject arbitrary SQL commands and execute them directly on the database.

The risk exists that an attacker gains unauthorized access to the information from the database of the application. He could extract information such as: application usernames, passwords, client information and other application specific data.

Recommendation:
We recommend implementing a validation mechanism for all the data received from the users.
The best way to protect against SQL Injection is to use prepared statements for every SQL query performed on the database.
Otherwise, the user input can also be sanitized using dedicated methods such as: mysqli_real_escape_string.

References:

https://owasp.org/www-community/attacks/SQL_Injection
https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html

Classification:
CWE : CWE-89
OWASP Top 10 - 2013 : A1 - Injection
OWASP Top 10 - 2017 : A1 - Injection

OS Command Injection

CONFIRMED

URL	Method	Parameters	Evidence	Replay Attack
http://rest.testsparker.com/jwt/api/users/{username}	PUT	Body: email=example_email@example.com;ps #';ps #";ps # first_name=first_name last_name=last_name password=Secure123456$ username=us3rn4me2bed373rm1n3d Headers:...	Injected the `ps` command in the **email body parameter** and found the expected command output in the response	

Details

Risk description:
We found that the target web application can be manipulated into running operating system commands on the host machine.
The risk exists that an attacker uses the application to run OS commands with the privileges of the vulnerable application. This could lead (but not limited) to Remote Code Execution, Denial of Service, Sensitive Information Disclosure, Sensitive Information Deletion.

Recommendation:
There are multiple ways to mitigate this attack:
- avoid calling OS commands directly (use built-in library functions) - escape values added to OS commands specific to each OS
- implement parametrization in conjunction with Input Validation (segregate data by command; implement Positive or whitelist input validation; White list Regular Expression)
In order to provide Defense in Depth, we also recommend to allocate the lowest privileges to web applications.

References:
https://owasp.org/www-community/attacks/Command_Injection
https://cheatsheetseries.owasp.org/cheatsheets/OS_Command_Injection_Defense_Cheat_Sheet.html

Classification:
CWE : CWE-78
OWASP Top 10 - 2013 : A1 - Injection
OWASP Top 10 - 2017 : A1 - Injection

Local File Inclusion

CONFIRMED

URL	Method	Parameters	Evidence	Replay Attack

http://rest.testsparker.com/jwt/api/comments/{commentId}	PUT	Body:
		comment=/proc/1/sched
		post_id=1123123
		user_id=1123123
		Headers:
		Authorization=Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiIsImtpZCI6InNlY3JldC50eHQifQ.eyJ1c2VyIjoidGVzdCJ9.jqBFzyBB68KWiOvEJhcaDgMY0Gea-t0KNnf-fR2loyc
		accept=application/json
		content-type=application/json
			We found a Local File Inclusion vulnerability in the **comment body parameter**.We managed to read the contents of two files.
			First, we tested for the vulnerability by injecting the payload: `/proc/cpuinfo`. We extracted the data:
			`processor : 0`
			`vendor_id : GenuineIntel`
			`cpu family : 6`
			`model : 79`
			`model name : Intel`
			`stepping : 1`
			`processor : 1`
			`vendor_id : GenuineIntel`
			`cpu family : 6`
			`model : 79`
			
			Additionally, we validated the vulnerability by injecting the payload: `/proc/1/sched`. The extracted data was:
			`se.exec_start : 37893075.567327`
			`se.vruntime : 128724.039940`
			`se.sum_exec_runtime : 1499.092015`
			`se.nr_migrations : 2`
			`nr_switches : 38706`
			`nr_voluntary_switches : 38487`
			`nr_involuntary_switches : 219`
			`se.load.weight : 1048576`
			`se.avg.load_sum : 49`
			`se.avg.util_sum : 21504`	

Details

Risk description:

Local File Inclusion (also known as LFI) is the process of including files, that are already locally present on the server, by exploiting the vulnerable inclusion procedures implemented in the application.
For instance, by supplying a payload like:"?file=../../../etc/passwd" to the target URL, an attacker could be able to display the contents of /etc/passwd file (stored locally).
The risk exists that an attacker can manipulate the affected parameter in order to load and sometimes execute any locally stored file. This could lead to reading arbitrary files, code execution, Cross-Site Scripting, denial of service, sensitive information disclosure.

Recommendation:

The most effective solution to eliminating file inclusion vulnerabilities is to avoid passing raw user-submitted input to any filesystem API. If this is not possible, the application can maintain a white list of files that may be included by the page, and then check to see if the user input matches against any of the entries in the white list. Any request containing an invalid identifier has to be rejected. In this way, there is no attack surface for malicious users to manipulate the path.

References:

https://owasp.org/www-project-web-security-testing-guide/stable/4-Web_Application_Security_Testing/07-Input_Validation_Testing/11.1-Testing_for_Local_File_Inclusion

Classification:

CWE : CWE-22
OWASP Top 10 - 2013 : A1 - Injection
OWASP Top 10 - 2017 : A1 - Injection

Vulnerabilities found for server-side software

UNCONFIRMED

Risk Level	CVSS	CVE	Summary	Exploit	Affected software
	9.8	CVE-2023-25690	Some mod_proxy configurations on Apache HTTP Server versions 2.4.0 through 2.4.55 allow a HTTP Request Smuggling attack. Configurations are affected when mod_proxy is enabled along with some form of RewriteRule or ProxyPassMatch in which a non-specific pattern matches some portion of the user-supplied request-target (URL) data and is then re-inserted into the proxied request-target using variable substitution. For example, something like: RewriteEngine on RewriteRule "^/here/(.*)" "http://example.com:8080/elsewhere?$1"; [P] ProxyPassReverse /here/ http://example.com:8080/ Request splitting/smuggling could result in bypass of access controls in the proxy server, proxying unintended URLs to existing origin servers, and cache poisoning. Users are recommended to update to at least version 2.4.56 of Apache HTTP Server.	N/A	http_server 2.4.25
	9	CVE-2022-36760	Inconsistent Interpretation of HTTP Requests ('HTTP Request Smuggling') vulnerability in mod_proxy_ajp of Apache HTTP Server allows an attacker to smuggle requests to the AJP server it forwards requests to. This issue affects Apache HTTP Server Apache HTTP Server 2.4 version 2.4.54 and prior versions.	N/A	http_server 2.4.25
	7.8	CVE-2019-9517	Some HTTP/2 implementations are vulnerable to unconstrained interal data buffering, potentially leading to a denial of service. The attacker opens the HTTP/2 window so the peer can send without constraint; however, they leave the TCP window closed so the peer cannot actually write (many of) the bytes on the wire. The attacker then sends a stream of requests for a large response object. Depending on how the servers queue the responses, this can consume excess memory, CPU, or both.	N/A	http_server 2.4.25
	7.5	CVE-2017-3167	In Apache httpd 2.2.x before 2.2.33 and 2.4.x before 2.4.26, use of the ap_get_basic_auth_pw() by third-party modules outside of the authentication phase may lead to authentication requirements being bypassed.	N/A	http_server 2.4.25
	7.5	CVE-2017-3169	In Apache httpd 2.2.x before 2.2.33 and 2.4.x before 2.4.26, mod_ssl may dereference a NULL pointer when third-party modules call ap_hook_process_connection() during an HTTP request to an HTTPS port.	N/A	http_server 2.4.25
	7.5	CVE-2019-11043	In PHP versions 7.1.x below 7.1.33, 7.2.x below 7.2.24 and 7.3.x below 7.3.11 in certain configurations of FPM setup it is possible to cause FPM module to write past allocated buffers into the space reserved for FCGI protocol data, thus opening the possibility of remote code execution.	N/A	php 7.1.26
	7.5	CVE-2019-13224	A use-after-free in onig_new_deluxe() in regext.c in Oniguruma 6.9.2 allows attackers to potentially cause information disclosure, denial of service, or possibly code execution by providing a crafted regular expression. The attacker provides a pair of a regex pattern and a string, with a multi-byte encoding that gets handled by onig_new_deluxe(). Oniguruma issues often affect Ruby, as well as common optional libraries for PHP and Rust.	N/A	php 7.1.26
	7.5	CVE-2017-8923	The zend_string_extend function in Zend/zend_string.h in PHP through 7.1.5 does not prevent changes to string objects that result in a negative length, which allows remote attackers to cause a denial of service (application crash) or possibly have unspecified other impact by leveraging a script's use of .= with a long string.	N/A	php 7.1.26
	7.5	CVE-2019-9641	An issue was discovered in the EXIF component in PHP before 7.1.27, 7.2.x before 7.2.16, and 7.3.x before 7.3.3. There is an uninitialized read in exif_process_IFD_in_TIFF.	N/A	php 7.1.26
	6.8	CVE-2019-9675	** DISPUTED ** An issue was discovered in PHP 7.x before 7.1.27 and 7.3.x before 7.3.3. phar_tar_writeheaders_int in ext/phar/tar.c has a buffer overflow via a long link value. NOTE: The vendor indicates that the link value is used only when an archive contains a symlink, which currently cannot happen: "This issue allows theoretical compromise of security, but a practical attack is usually impossible."	N/A	php 7.1.26

Details

Risk description:
These vulnerabilities expose the affected applications to the risk of unauthorized access to confidential data and possibly to denial of service attacks. An attacker could search for an appropriate exploit (or create one himself) for any of these vulnerabilities and use it to attack the system.

Recommendation:
We recommend you to upgrade the affected software to the latest version in order to eliminate the risk of these vulnerabilities.

Classification:
CWE : CWE-1026
OWASP Top 10 - 2013 : A9 - Using Components with Known Vulnerabilities
OWASP Top 10 - 2017 : A9 - Using Components with Known Vulnerabilities

Communication is not secure

CONFIRMED

URL	Evidence
http://rest.testsparker.com/jwt	Communication is made over unsecure, unencrypted HTTP.

Details

Risk description:
The communication between the web browser and the server is done using the HTTP protocol, which transmits data unencrypted over the network. Thus, an attacker who manages to intercept the communication at the network level is able to read and modify the data transmitted (including passwords, secret tokens, credit card information and other sensitive data).

Recommendation:
We recommend you to reconfigure the web server to use HTTPS - which encrypts the communication between the web browser and the server.

Classification:
CWE : CWE-311
OWASP Top 10 - 2013 : A6 - Sensitive Data Exposure
OWASP Top 10 - 2017 : A3 - Sensitive Data Exposure

Internal Server Error Found

CONFIRMED

URL	Method	Parameters	Evidence
http://rest.testsparker.com/jwt/api/users	POST	Body: email=example_email@example.com first_name=first_name last_name=last_name password=Secure123456$ username=us3rn4me2bed373rm1n3d Headers: Authorization=Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiIsImtpZCI6InNIY3JldC50eHQifQ.eyJ1c2VyIjoidGVzdCJ9.jqBFzyBB68KWiOvEJhcaDgMY0Gea-t0KNnf-fR2Ioyc...	Response has an internal server error status code: 500

Details

Risk description:
The website does not handle or incorrectly handles an exceptional condition. An attacker may use the contents of error messages to help launch another, more focused attack. For example, an attempt to exploit a path traversal weakness (CWE-22) might yield the full pathname of the installed application.

Recommendation:
Ensure that error messages only contain minimal details that are useful to the intended audience, and nobody else. The messages need to strike the balance between being too cryptic and not being cryptic enough. They should not necessarily reveal the methods that were used to determine the error. Such detailed information can be used to refine the original attack to increase the chances of success. If errors must be tracked in some detail, capture them in log messages - but consider what could occur if the log messages can be viewed by attackers. Avoid recording highly sensitive information such as passwords in any form. Avoid inconsistent messaging that might accidentally tip off an attacker about internal state, such as whether a username is valid or not.

Classification:
CWE : CWE-209
OWASP Top 10 - 2013 : A5 - Security Misconfiguration
OWASP Top 10 - 2017 : A6 - Security Misconfiguration

Server software and technology found

UNCONFIRMED

Software / Version	Category
Debian	Operating systems
Apache 2.4.25	Web servers
PHP 7.1.26	Programming languages

Details

Risk description:
An attacker could use this information to mount specific attacks against the identified software type and version.

Recommendation:
We recommend you to eliminate the information which permits the identification of software platform, technology, server and operating system: HTTP server headers, HTML meta information, etc.

References:
https://owasp.org/www-project-web-security-testing-guide/stable/4-Web_Application_Security_Testing/01-Information_Gathering/02-Fingerprint_Web_Server.html

Classification:
OWASP Top 10 - 2013 : A5 - Security Misconfiguration
OWASP Top 10 - 2017 : A6 - Security Misconfiguration

Error message containing sensitive information

UNCONFIRMED

URL	Method	Parameters	Evidence
http://rest.testsparker.com/jwt/api/users/'	GET	Headers: Authorization=Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiIsImtpZCI6InNl Y3JldC50eHQifQ.eyJ1c2VyIjoidGVzdCJ9.jqBFzyBB6 8KWiOvEJhcaDgMY0Gea-t0KNnf-fR2loyc	Identified possible information disclosure message in the source page, specifically: Error message You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version...(truncated)

Details

Risk description:
The website does not handle or incorrectly handles an exceptional condition. An attacker may use the contents of error messages to help launch another, more focused attack. For example, an attempt to exploit a path traversal weakness (CWE-22) might yield the full pathname of the installed application.

Recommendation:
It is recommended treating all exceptions of the application flow. Ensure that error messages only contain minimal details.

Classification:
CWE : CWE-209
OWASP Top 10 - 2013 : A5 - Security Misconfiguration
OWASP Top 10 - 2017 : A6 - Security Misconfiguration

Security.txt file is missing

CONFIRMED

URL
Missing: http://rest.testsparker.com/.well-known/security.txt

Details

Risk description:
We have detected that the server is missing the security.txt file. There is no particular risk in not creating a valid Security.txt file for your server. However, this file is important because it offers a designated channel for reporting vulnerabilities and security issues.

Recommendation:
We recommend you to implement the security.txt file according to the standard, in order to allow researchers or users report any security

issues they find, improving the defensive mechanisms of your server.

References:
https://securitytxt.org/

Classification:
OWASP Top 10 - 2013 : A5 - Security Misconfiguration
OWASP Top 10 - 2017 : A6 - Security Misconfiguration

Spider results

URL	Method	Parameters
http://rest.testsparker.com/jwt/api/comments/{commentId}	PUT	Body: comment=comment post_id=1123123 user_id=1123123 Headers: Authorization=Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiIsImtpZCI6InNlY3JldC50eHQifQ.eyJ1c2VyIjoidGVzdCJ9.jqBFzyBB68KWiOvEJhcaDgMY0Gea-t0KNnf-fR2loyc accept=application/json content-type=application/json
http://rest.testsparker.com/jwt/api/comments/{commentId}	GET	Headers: Authorization=Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiIsImtpZCI6InNlY3JldC50eHQifQ.eyJ1c2VyIjoidGVzdCJ9.jqBFzyBB68KWiOvEJhcaDgMY0Gea-t0KNnf-fR2loyc
http://rest.testsparker.com/jwt/api/comments/{commentId}	DELETE	Query: callback=callback Body: comment=comment post_id=1123123 user_id=1123123 Headers: Authorization=Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiIsImtpZCI6InNlY3JldC50eHQifQ.eyJ1c2VyIjoidGVzdCJ9.jqBFzyBB68KWiOvEJhcaDgMY0Gea-t0KNnf-fR2loyc
http://rest.testsparker.com/jwt/api/comments	POST	Body: comment=comment post_id=1123123 user_id=1123123 Headers: Authorization=Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiIsImtpZCI6InNlY3JldC50eHQifQ.eyJ1c2VyIjoidGVzdCJ9.jqBFzyBB68KWiOvEJhcaDgMY0Gea-t0KNnf-fR2loyc accept=application/json content-type=application/json
http://rest.testsparker.com/jwt/api/comments	GET	Headers: Authorization=Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiIsImtpZCI6InNlY3JldC50eHQifQ.eyJ1c2VyIjoidGVzdCJ9.jqBFzyBB68KWiOvEJhcaDgMY0Gea-t0KNnf-fR2loyc
http://rest.testsparker.com/jwt/api/posts/{commentId}/comments	PUT	Body: content=content title=title user_id=1123123 Headers: Authorization=Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiIsImtpZCI6InNlY3JldC50eHQifQ.eyJ1c2VyIjoidGVzdCJ9.jqBFzyBB68KWiOvEJhcaDgMY0Gea-t0KNnf-fR2loyc accept=application/json content-type=application/json

http://rest.testsparker.com/jwt/api/posts/{commentId}/comments	GET	Headers: Authorization=Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiIsImtpZCI6InNlY3JldC50eHQifQ.eyJ1c2VyIjoidGVzdCJ9.jqBFzyBB68KWiOvEJhcaDgMY0Gea-t0KNnf-fR2loyc
http://rest.testsparker.com/jwt/api/posts/{commentId}/comments	DELETE	Query: callback=callback Body: content=content title=title user_id=1123123 Headers: Authorization=Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiIsImtpZCI6InNlY3JldC50eHQifQ.eyJ1c2VyIjoidGVzdCJ9.jqBFzyBB68KWiOvEJhcaDgMY0Gea-t0KNnf-fR2loyc
http://rest.testsparker.com/jwt/api/posts/{postId}	GET	Headers: Authorization=Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiIsImtpZCI6InNlY3JldC50eHQifQ.eyJ1c2VyIjoidGVzdCJ9.jqBFzyBB68KWiOvEJhcaDgMY0Gea-t0KNnf-fR2loyc
http://rest.testsparker.com/jwt/api/posts	POST	Body: content=content title=title user_id=1123123 Headers: Authorization=Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiIsImtpZCI6InNlY3JldC50eHQifQ.eyJ1c2VyIjoidGVzdCJ9.jqBFzyBB68KWiOvEJhcaDgMY0Gea-t0KNnf-fR2loyc accept=application/json content-type=application/json
http://rest.testsparker.com/jwt/api/posts	GET	Headers: Authorization=Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiIsImtpZCI6InNlY3JldC50eHQifQ.eyJ1c2VyIjoidGVzdCJ9.jqBFzyBB68KWiOvEJhcaDgMY0Gea-t0KNnf-fR2loyc
http://rest.testsparker.com/jwt/api/users/{username}/posts	GET	Headers: Authorization=Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiIsImtpZCI6InNlY3JldC50eHQifQ.eyJ1c2VyIjoidGVzdCJ9.jqBFzyBB68KWiOvEJhcaDgMY0Gea-t0KNnf-fR2loyc
http://rest.testsparker.com/jwt/api/users/{username}	PUT	Body: email=example_email@example.com first_name=first_name last_name=last_name password=Secure123456$ username=us3rn4me2bed373rm1n3d Headers: Authorization=Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiIsImtpZCI6InNlY3JldC50eHQifQ.eyJ1c2VyIjoidGVzdCJ9.jqBFzyBB68KWiOvEJhcaDgMY0Gea-t0KNnf-fR2loyc...
http://rest.testsparker.com/jwt/api/users/{username}	GET	Headers: Authorization=Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiIsImtpZCI6InNlY3JldC50eHQifQ.eyJ1c2VyIjoidGVzdCJ9.jqBFzyBB68KWiOvEJhcaDgMY0Gea-t0KNnf-fR2loyc
http://rest.testsparker.com/jwt/api/users/{username}	DELETE	Query: callback=callback Body: email=example_email@example.com first_name=first_name last_name=last_name password=Secure123456$ username=us3rn4me2bed373rm1n3d Headers:...

http://rest.testsparker.com/jwt/api/users	POST	Body: email=example_email@example.com first_name=first_name last_name=last_name password=Secure123456$ username=us3rn4me2bed373rm1n3d Headers: Authorization=Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiIsImtpZCI6InNlY3JldC50eHQifQ.eyJ1c2VyIjoidGVzdCJ9.jqBFzyBB68KWiOvEJhcaDgMY0Gea-t0KNnf-fR2loyc...
http://rest.testsparker.com/jwt/api/users	GET	Headers: Authorization=Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiIsImtpZCI6InNlY3JldC50eHQifQ.eyJ1c2VyIjoidGVzdCJ9.jqBFzyBB68KWiOvEJhcaDgMY0Gea-t0KNnf-fR2loyc
http://rest.testsparker.com/jwt/api	GET	Headers: Authorization=Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiIsImtpZCI6InNlY3JldC50eHQifQ.eyJ1c2VyIjoidGVzdCJ9.jqBFzyBB68KWiOvEJhcaDgMY0Gea-t0KNnf-fR2loyc
http://rest.testsparker.com/jwt	GET	Headers: User-Agent=Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.0.0 Safari/537.36

Input Reflected in Response

UNCONFIRMED

URL	Method	Parameters	Evidence	Replay Attack
http://rest.testsparker.com/jwt/api/comments/{commentId}	DELETE	Query: callback=callback'"--> </noscript></title> </textarea></style> </template> </noembed></script> <svg/*/onload=alert(document.domain)//> Body: comment=comment post_id=1123123 user_id=1123123 Headers:...	Injected the payload `'"--></noscript></title></textarea></style></template></noembed></script><svg/*/onload=document.body.append`${541890-54189}`//>` in the **callback query parameter** and it was found reflected in the response	
http://rest.testsparker.com/jwt/api/users/{username}	DELETE	Query: callback=callback'"--> </noscript></title> </textarea></style> </template> </noembed></script> <svg/*/onload=alert(document.domain)//> Body: email=example_email@example.com first_name=first_name last_name=last_name password=Secure123456$ username=us3rn4me2bed373rm1n3d Headers:...	Injected the payload `'"--></noscript></title></textarea></style></template></noembed></script><svg/*/onload=document.body.append`${470640-47064}`//>` in the **callback query parameter** and it was found reflected in the response	

Details

Risk description:

The web application reflects the input it receives in the body of the HTTP Response. This does not represent a vulnerability by itself, but it is the first step in common client-side vulnerabilities like Cross-Site Scripting.

Recommendation:
It is recommended that a tester inspects this issue manually to find out if it can be escalated to higher-risk vulnerabilities.

References:
https://owasp.org/www-community/attacks/xss
https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html

Classification:
CWE : CWE-20

Website is accessible.

Nothing was found for outdated JavaScript libraries.

Nothing was found for CORS misconfiguration.

Nothing was found for use of untrusted certificates.

Nothing was found for enabled HTTP debug methods.

Nothing was found for directory listing.

Nothing was found for passwords submitted unencrypted.

Nothing was found for Cross-Site Scripting.

Nothing was found for debug messages.

Nothing was found for code comments.

Nothing was found for missing HTTP header - Strict-Transport-Security.

Nothing was found for missing HTTP header - Content Security Policy.

Nothing was found for missing HTTP header - X-Frame-Options.

Nothing was found for missing HTTP header - X-XSS-Protection.

Nothing was found for missing HTTP header - X-Content-Type-Options.

Nothing was found for missing HTTP header - Referrer.

Nothing was found for domain too loose set for cookies.

Nothing was found for mixed content between HTTP and HTTPS.

Nothing was found for cross domain file inclusion.

Nothing was found for HttpOnly flag of cookie.

Nothing was found for Secure flag of cookie.

Nothing was found for login interfaces.

Nothing was found for secure password submission.

Nothing was found for sensitive data.

Nothing was found for Server Side Request Forgery.

Nothing was found for Open Redirect.

Nothing was found for PHP Code Injection.

Nothing was found for JavaScript Code Injection.

Nothing was found for Ruby Code Injection.

Nothing was found for Python Code Injection.

Nothing was found for Perl Code Injection.

Nothing was found for Remote Code Execution through Log4j.

Nothing was found for Remote Code Execution through VIEWSTATE.

Scan coverage information

List of tests performed (44/44)

- Checking for website accessibility...
- Checking for secure communication...
- Spidering target...
- Checking for website technologies...
- Checking for vulnerabilities of server-side software...
- Checking for absence of the security.txt file...
- Checking for outdated JavaScript libraries...
- Checking for CORS misconfiguration...
- Checking for use of untrusted certificates...
- Checking for enabled HTTP debug methods...
- Checking for error messages...
- Checking for SQL Injection...
- Checking for internal error code...
- Checking for OS Command Injection...
- Checking for Cross-Site Scripting...
- Checking for Local File Inclusion...
- Checking for directory listing...
- Checking for passwords submitted unencrypted...
- Checking for Cross-Site Scripting...
- Checking for debug messages...
- Checking for code comments...
- Checking for missing HTTP header - Strict-Transport-Security...
- Checking for missing HTTP header - Content Security Policy...
- Checking for missing HTTP header - X-Frame-Options...
- Checking for missing HTTP header - X-XSS-Protection...
- Checking for missing HTTP header - X-Content-Type-Options...
- Checking for missing HTTP header - Referrer...
- Checking for domain too loose set for cookies...
- Checking for mixed content between HTTP and HTTPS...
- Checking for cross domain file inclusion...
- Checking for HttpOnly flag of cookie...
- Checking for Secure flag of cookie...
- Checking for login interfaces...
- Checking for secure password submission...
- Checking for sensitive data...
- Checking for Server Side Request Forgery...
- Checking for Open Redirect...
- Checking for PHP Code Injection...
- Checking for JavaScript Code Injection...
- Checking for Ruby Code Injection...
- Checking for Python Code Injection...
- Checking for Perl Code Injection...
- Checking for Remote Code Execution through Log4j...
- Checking for Remote Code Execution through VIEWSTATE...

Scan parameters

Target:	http://rest.testsparker.com/jwt
Scan Type:	REST
Spec URL:	http://rest.testsparker.com/files/openapi-swagger_jwt.yaml

Scan stats

Unique Injection Points Detected:	19
URLs spidered:	1
Total number of HTTP requests:	10937
Average time until a response was received:	149ms