SiteWALL Vulnerability Assessment Report

Application Name: www.testreport.com

Date: 7th March 2025

Platform: wordPress 8.1

Web Security Score

Prepared By: PageNTRA Infosec Pvt Ltd

SiteWALL Vulnerability Assessment Report

Application : www.testreport.com

Date : 7th March 2025

Prepared by: PageNTRA Infosec Pvt Ltd

1. Executive Summary

Purpose: This report summarizes the findings from the recent vulnerability assessment conducted on our web application to identify potential security threats this month.

2. Scope of the Assessment

The scope of this assessment is strictly confined to your website and/or web application. We conduct scans on your website or web applications using professional, customized tools. These scans are performed without authentication to identify vulnerabilities and security gaps that could potentially be exploited by hackers.Should you require scans that involve authenticated sessions, which can provide deeper insights into secured areas of your website or application,please contact us to discuss and arrange for authenticated scans.

Tested Components: Web application without Authentication.

Testing Techniques: Sophisticated scanning tools and manual scanning methods.

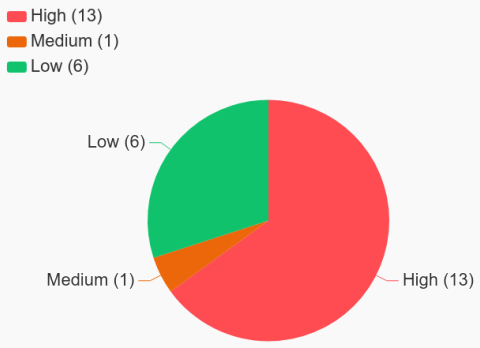
3. Key Findings

Backend Platform: wordPress 8.1

Overview: A total of [20] vulnerabilities were identified, categorized as [13] High, [1] Medium, [6] Low and [0] Info.

OWASP Top 10:

- A05: Security Misconfiguration



4. Risk Assessment

Web Security Score: F

Risk Ratings: High

Others: Missing security headers:

1) Strict-Transport-Security
When secure cookies are not used, there is an increased risk of third parties intercepting contained in these cookies. Exposing information about the server version increases the ability of attackers to exploit certain vulnerabilities. The website configuration should be changed to prevent version information from being revealed in the server header.
Click here to know, how to fix this issue

2) X-Frame-Options
Browser may display this website's content in a frames. This can lead to a clickjacking attack.
Click here to know, how to fix this issue

3) Content-Security-Policy
No valid Content Security Policy is implemented. This increases the risk of XSS & clickjacking attacks.
Click here to know, how to fix this issue

4) X-Content-Type-Options
Browser may interpret files as a different MIME type than what is specified in the Content-Type HTTP header. This can lead to a MIME confusion attacks.
Click here to know, how to fix this issue

5. Appendix and References

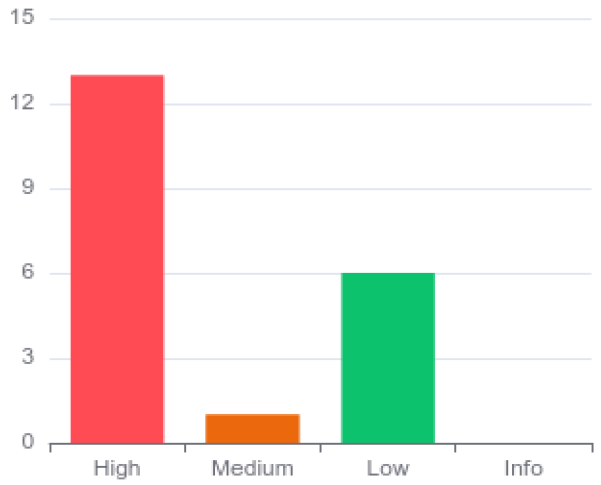
Detailed Reports: Full technical report and scan data are available in the following section.

Web Security Score: Details

Compliance and Standards: Assessment was performed considering OWASP Top 10, PCI DSS industry standard.

Vulnerability Score Card

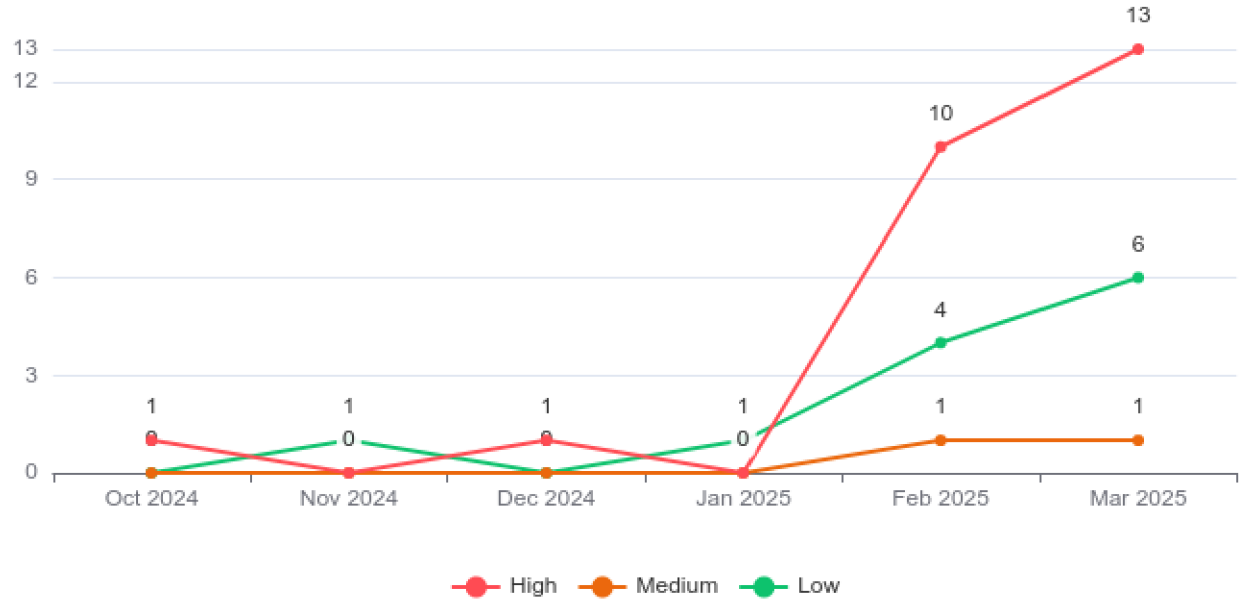
Vulnerability v/s Severity Bar Chart



6-Month Severity Table

Severity	Oct 2024	Nov 2024	Dec 2024	Jan 2025	Feb 2025	Mar 2025
High	1	0	1	0	10	13
Medium	0	0	0	0	1	1
Low	0	1	0	1	4	6
Info	0	0	0	0	0	0

Comparison of Severities Over Time



Technical Report

Summary

The table below shows the numbers of issues identified in different categories. Issues are classified according to severity as High, Medium, Low, Information. This reflects the likely impact of each issue for a typical organization. Issues are also classified according to confidence as Certain, Firm or Tentative. This reflects the inherent reliability of the technique that was used to identify the issue.

		Confidence			
		Certain	Firm	Tentative	Total
Severity	High	13	0	0	13
	Medium	0	1	0	1
	Low	3	3	0	6
	Information	0	0	0	0

Contents

1. Cross-origin resource sharing: arbitrary origin trusted

There are 13 instances of this issue:

- /wp-json/
- /wp-json/content-forms/v1/submit
- /wp-json/oembed/1.0/embed
- /wp-json/wp/v2/categories/6
- /wp-json/wp/v2/categories/7
- /wp-json/wp/v2/pages/21
- /wp-json/wp/v2/pages/26
- /wp-json/wp/v2/pages/30
- /wp-json/wp/v2/pages/32
- /wp-json/wp/v2/pages/39
- /wp-json/wp/v2/pages/41
- /wp-json/wp/v2/posts/401
- /wp-json/wp/v2/posts/404

Issue background

An HTML5 cross-origin resource sharing (CORS) policy controls whether and how content running on other domains can perform two-way interaction with the domain that publishes the policy. The policy is fine-grained and can apply access controls per-request based on the URL and other features of the request.

Trusting arbitrary origins effectively disables the same-origin policy, allowing two-way interaction by third-party web sites. Unless the response consists only of unprotected public content, this policy is likely to present a security risk.

If the site specifies the header Access-Control-Allow-Credentials: true, third-party sites may be able to carry out privileged actions and retrieve sensitive information. Even if it does not, attackers may be able to bypass any IP-based access controls by proxying through users' browsers.

Issue remediation

Rather than using a wildcard or programmatically verifying supplied origins, use a whitelist of trusted domains.

Vulnerability classifications

- CWE-942: Overly Permissive Cross-domain Whitelist

1.1. https://www.testreport.com/wp-json/

Summary

Severity:	**High**
Confidence:	**Certain**
Host:	**https://www.testreport.com**
Path:	**/wp-json/**

Issue detail

The application implements an HTML5 cross-origin resource sharing (CORS) policy for this request that allows access from any domain.

The application allowed access from the requested origin **https://kgwvghzgzzui.com**

Request

```
GET /wp-json/ HTTP/2
Host: www.testreport.com
Accept-Encoding: gzip, deflate, br
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Language: en-US;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 Safari/537.36
Connection: close
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Sec-CH-UA: "Google Chrome";v="133", "Not=A?Brand";v="8", "Chromium";v="133"
Sec-CH-UA-Platform: "Linux"
Sec-CH-UA-Mobile: ?0
Content-Length: 0
Origin: https://kgwvghzgzzui.com
```

Response

```
HTTP/2 200 OK
X-Powered-By: PHP/7.4.33
X-Robots-Tag: noindex
Link: <https://www.testreport.com/wp-json/>; rel="https://api.w.org/"
X-Content-Type-Options: nosniff
Access-Control-Expose-Headers: X-WP-Total, X-WP-TotalPages, Link
Access-Control-Allow-Headers: Authorization, X-WP-Nonce, Content-Disposition, Content-MD5, Content-Type
Allow: GET
Access-Control-Allow-Origin: https://kgwvghzgzzui.com
Access-Control-Allow-Methods: OPTIONS, GET, POST, PUT, PATCH, DELETE
Access-Control-Allow-Credentials: true
Vary: Origin,Accept-Encoding
Content-Length: 129405
Content-Type: application/json; charset=UTF-8
Date: Mon, 03 Mar 2025 18:51:16 GMT
Server: Apache

{"name":"testreport Infosystem Pvt Ltd","description":"Innovative IT Solutions to Drive Your Business Forward","url":"https:\/\/www.testreport.com","home":"https:\/\/www.testreport.com","gmt_offset":"5.5","
...[SNIP]...
```

1.2. https://www.testreport.com/wp-json/content-forms/v1/submit

Summary

Severity:	**High**
Confidence:	**Certain**
Host:	**https://www.testreport.com**
Path:	**/wp-json/content-forms/v1/submit**

Issue detail

The application implements an HTML5 cross-origin resource sharing (CORS) policy for this request that allows access from any domain.

The application allowed access from the requested origin **https://uknumjqiutcu.com**

Request

```
POST /wp-json/content-forms/v1/submit HTTP/2
Host: www.testreport.com
Accept-Encoding: gzip, deflate, br
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-US;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 Safari/537.36
Connection: close
Cache-Control: max-age=0
Origin: https://uknumjqiutcu.com
X-Requested-With: XMLHttpRequest
Referer: https://www.testreport.com/contact/
X-WP-Nonce: 89805002eb
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Sec-CH-UA: "Google Chrome";v="133", "Not=A?Brand";v="8", "Chromium";v="133"
Sec-CH-UA-Platform: "Linux"
Sec-CH-UA-Mobile: ?0
Content-Length: 229

form_type=contact&form_id=b64b0df&post_id=41&form_builder=elementor&data%5Bb64b0df%5D%5Bname%5D=VHBoIy&dat
a%5Bb64b0df%5D%5Bemail%5D=xVJDUX&data%5Bb64b0df%5D%5Bphone%5D=gnFuzJ&data%5Bb64b0df%5D%5Bmessa
...[SNIP]...
```

Response

```
HTTP/2 400 Bad Request
X-Powered-By: PHP/7.4.33
X-Robots-Tag: noindex
Link: <https://www.testreport.com/wp-json/>; rel="https://api.w.org/"
X-Content-Type-Options: nosniff
Access-Control-Expose-Headers: X-WP-Total, X-WP-TotalPages, Link
Access-Control-Allow-Headers: Authorization, X-WP-Nonce, Content-Disposition, Content-MD5, Content-Type
X-Wp-Nonce: 89805002eb
Allow: POST
Access-Control-Allow-Origin: https://uknumjqiutcu.com
Access-Control-Allow-Methods: OPTIONS, GET, POST, PUT, PATCH, DELETE
```

Access-Control-Allow-Credentials: true
Vary: Origin,Accept-Encoding
Content-Length: 44
Content-Type: application/json; charset=UTF-8
Date: Sun, 02 Mar 2025 13:06:18 GMT
Server: Apache

{"success":false,"message":"Invalid email."}

1.3. https://www.testreport.com/wp-json/oembed/1.0/embed

Summary

Severity:	**High**
Confidence:	**Certain**
Host:	**https://www.testreport.com**
Path:	**/wp-json/oembed/1.0/embed**

Issue detail

The application implements an HTML5 cross-origin resource sharing (CORS) policy for this request that allows access from any domain.

The application allowed access from the requested origin **https://plehbmphmsvt.com**

Request

GET /wp-json/oembed/1.0/embed?url=https%3A%2F%2Fwww.testreport.com%2F HTTP/2
Host: www.testreport.com
Accept-Encoding: gzip, deflate, br
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Language: en-US;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 Safari/537.36
Connection: close
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Sec-CH-UA: "Google Chrome";v="133", "Not=A?Brand";v="8", "Chromium";v="133"
Sec-CH-UA-Platform: "Linux"
Sec-CH-UA-Mobile: ?0
Content-Length: 0
Origin: https://plehbmphmsvt.com

Response

HTTP/2 200 OK
X-Powered-By: PHP/7.4.33
X-Robots-Tag: noindex
Link: <https://www.testreport.com/wp-json/>; rel="https://api.w.org/"
X-Content-Type-Options: nosniff
Access-Control-Expose-Headers: X-WP-Total, X-WP-TotalPages, Link
Access-Control-Allow-Headers: Authorization, X-WP-Nonce, Content-Disposition, Content-MD5, Content-Type

Allow: GET
Access-Control-Allow-Origin: https://plehbmphmsvt.com
Access-Control-Allow-Methods: OPTIONS, GET, POST, PUT, PATCH, DELETE
Access-Control-Allow-Credentials: true
Vary: Origin,Accept-Encoding
Content-Length: 2300
Content-Type: application/json; charset=UTF-8
Date: Mon, 03 Mar 2025 18:39:36 GMT
Server: Apache

{"version":"1.0","provider_name":"testreport Infosystem Pvt Ltd","provider_url":"https:\/\/www.testreport.com","author_name":"eduVistara","author_url":"https:\/\/www.testreport.com\/author\/admin\/","title"
...[SNIP]...

1.4. https://www.testreport.com/wp-json/wp/v2/categories/6

Summary

Severity:	**High**
Confidence:	**Certain**
Host:	**https://www.testreport.com**
Path:	**/wp-json/wp/v2/categories/6**

Issue detail

The application implements an HTML5 cross-origin resource sharing (CORS) policy for this request that allows access from any domain.

The application allowed access from the requested origin **https://krppfgbqdrks.com**

Request

GET /wp-json/wp/v2/categories/6 HTTP/2
Host: www.testreport.com
Accept-Encoding: gzip, deflate, br
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Language: en-US;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 Safari/537.36
Connection: close
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Sec-CH-UA: "Google Chrome";v="133", "Not=A?Brand";v="8", "Chromium";v="133"
Sec-CH-UA-Platform: "Linux"
Sec-CH-UA-Mobile: ?0
Content-Length: 0
Origin: https://krppfgbqdrks.com

Response

HTTP/2 200 OK
X-Powered-By: PHP/7.4.33
X-Robots-Tag: noindex
Link: <https://www.testreport.com/wp-json/>; rel="https://api.w.org/"
X-Content-Type-Options: nosniff
Access-Control-Expose-Headers: X-WP-Total, X-WP-TotalPages, Link
Access-Control-Allow-Headers: Authorization, X-WP-Nonce, Content-Disposition, Content-MD5, Content-Type
Allow: GET
Access-Control-Allow-Origin: https://krppfgbqdrks.com
Access-Control-Allow-Methods: OPTIONS, GET, POST, PUT, PATCH, DELETE
Access-Control-Allow-Credentials: true
Vary: Origin,Accept-Encoding
Content-Length: 619
Content-Type: application/json; charset=UTF-8
Date: Mon, 03 Mar 2025 18:46:25 GMT
Server: Apache

{"id":6,"count":2,"description":"","link":"https:\/\/www.testreport.com\/category\/cyber-security\/","name":"Cyber Security","slug":"cyber-security","taxonomy":"category","parent":0,"meta":[],"_links":{
...[SNIP]...

1.5. https://www.testreport.com/wp-json/wp/v2/categories/7

Summary

Severity:	**High**
Confidence:	**Certain**
Host:	**https://www.testreport.com**
Path:	**/wp-json/wp/v2/categories/7**

Issue detail

The application implements an HTML5 cross-origin resource sharing (CORS) policy for this request that allows access from any domain.

The application allowed access from the requested origin **https://ziwtwjdhmrgi.com**

Request

GET /wp-json/wp/v2/categories/7 HTTP/2
Host: www.testreport.com
Accept-Encoding: gzip, deflate, br
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Language: en-US;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 Safari/537.36
Connection: close
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Sec-CH-UA: "Google Chrome";v="133", "Not=A?Brand";v="8", "Chromium";v="133"
Sec-CH-UA-Platform: "Linux"
Sec-CH-UA-Mobile: ?0
Content-Length: 0

Origin: https://ziwtwjdhmrgi.com

Response

```
HTTP/2 200 OK
X-Powered-By: PHP/7.4.33
X-Robots-Tag: noindex
Link: <https://www.testreport.com/wp-json/>; rel="https://api.w.org/"
X-Content-Type-Options: nosniff
Access-Control-Expose-Headers: X-WP-Total, X-WP-TotalPages, Link
Access-Control-Allow-Headers: Authorization, X-WP-Nonce, Content-Disposition, Content-MD5, Content-Type
Allow: GET
Access-Control-Allow-Origin: https://ziwtwjdhmrgi.com
Access-Control-Allow-Methods: OPTIONS, GET, POST, PUT, PATCH, DELETE
Access-Control-Allow-Credentials: true
Vary: Origin,Accept-Encoding
Content-Length: 622
Content-Type: application/json; charset=UTF-8
Date: Mon, 03 Mar 2025 18:47:31 GMT
Server: Apache

{"id":7,"count":1,"description":"","link":"https:\/\/www.testreport.com\/category\/data-management\/","name":"Data
Management","slug":"data-management","taxonomy":"category","parent":0,"meta":[],"_links
...[SNIP]...
```

1.6. https://www.testreport.com/wp-json/wp/v2/pages/21

Summary

Severity:	**High**
Confidence:	**Certain**
Host:	**https://www.testreport.com**
Path:	**/wp-json/wp/v2/pages/21**

Issue detail

The application implements an HTML5 cross-origin resource sharing (CORS) policy for this request that allows access from any domain.

The application allowed access from the requested origin **https://oyzbuotxkgdr.com**

Request

```
GET /wp-json/wp/v2/pages/21 HTTP/2
Host: www.testreport.com
Accept-Encoding: gzip, deflate, br
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Language: en-US;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 Safari/537.36
Connection: close
Cache-Control: max-age=0
```

Upgrade-Insecure-Requests: 1
Sec-CH-UA: "Google Chrome";v="133", "Not=A?Brand";v="8", "Chromium";v="133"
Sec-CH-UA-Platform: "Linux"
Sec-CH-UA-Mobile: ?0
Content-Length: 0
Origin: https://oyzbuotxkgdr.com

Response

HTTP/2 200 OK
X-Powered-By: PHP/7.4.33
X-Robots-Tag: noindex
X-Content-Type-Options: nosniff
Access-Control-Expose-Headers: X-WP-Total, X-WP-TotalPages, Link
Access-Control-Allow-Headers: Authorization, X-WP-Nonce, Content-Disposition, Content-MD5, Content-Type
Link: <https://www.testreport.com/>; rel="alternate"; type=text/html
Allow: GET
Access-Control-Allow-Origin: https://oyzbuotxkgdr.com
Access-Control-Allow-Methods: OPTIONS, GET, POST, PUT, PATCH, DELETE
Access-Control-Allow-Credentials: true
Vary: Origin,Accept-Encoding
Content-Length: 23080
Content-Type: application/json; charset=UTF-8
Date: Mon, 03 Mar 2025 18:34:23 GMT
Server: Apache

{"id":21,"date":"2021-03-03T15:01:35","date_gmt":"2021-03-03T15:01:35","guid":{"rendered":"https:\/\/www.testreport.com\/?page_id=21"},"modified":"2021-12-06T10:35:48","modified_gmt":"2021-12-06T05:05:4
...[SNIP]...

1.7. https://www.testreport.com/wp-json/wp/v2/pages/26

Summary

Severity:	**High**
Confidence:	**Certain**
Host:	**https://www.testreport.com**
Path:	**/wp-json/wp/v2/pages/26**

Issue detail

The application implements an HTML5 cross-origin resource sharing (CORS) policy for this request that allows access from any domain.

The application allowed access from the requested origin **https://dpzrvklvdyai.com**

Request

GET /wp-json/wp/v2/pages/26 HTTP/2
Host: www.testreport.com
Accept-Encoding: gzip, deflate, br
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-

```
exchange;v=b3;q=0.7
Accept-Language: en-US;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 Safari/537.36
Connection: close
Cache-Control: max-age=0
Cookie: wordpress_test_cookie=WP%20Cookie%20check; PHPSESSID=c6076e3b14bbf592cbff5d29446d9b97
Upgrade-Insecure-Requests: 1
Sec-CH-UA: "Google Chrome";v="133", "Not=A?Brand";v="8", "Chromium";v="133"
Sec-CH-UA-Platform: "Linux"
Sec-CH-UA-Mobile: ?0
Content-Length: 0
Origin: https://dpzrvklvdyai.com
```

Response

```
HTTP/2 200 OK
X-Powered-By: PHP/7.4.33
X-Robots-Tag: noindex
X-Content-Type-Options: nosniff
Access-Control-Expose-Headers: X-WP-Total, X-WP-TotalPages, Link
Access-Control-Allow-Headers: Authorization, X-WP-Nonce, Content-Disposition, Content-MD5, Content-Type
Link: <https://www.testreport.com/solutions/networking-security/>; rel="alternate"; type=text/html
Allow: GET
Access-Control-Allow-Origin: https://dpzrvklvdyai.com
Access-Control-Allow-Methods: OPTIONS, GET, POST, PUT, PATCH, DELETE
Access-Control-Allow-Credentials: true
Vary: Origin,Accept-Encoding
Content-Length: 1889
Content-Type: application/json; charset=UTF-8
Date: Mon, 03 Mar 2025 18:25:20 GMT
Server: Apache

{"id":26,"date":"2021-03-03T15:54:21","date_gmt":"2021-03-03T15:54:21","guid":{"rendered":"https:\/\/www.testreport.com\/?
page_id=26"},"modified":"2021-03-03T15:56:20","modified_gmt":"2021-03-03T15:56:2
...[SNIP]...
```

1.8. https://www.testreport.com/wp-json/wp/v2/pages/30

Summary

Severity:	**High**
Confidence:	**Certain**
Host:	**https://www.testreport.com**
Path:	**/wp-json/wp/v2/pages/30**

Issue detail

The application implements an HTML5 cross-origin resource sharing (CORS) policy for this request that allows access from any domain.

The application allowed access from the requested origin **https://ggxymzdiysmq.com**

Request

```
GET /wp-json/wp/v2/pages/30 HTTP/2
Host: www.testreport.com
Accept-Encoding: gzip, deflate, br
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Language: en-US;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 Safari/537.36
Connection: close
Cache-Control: max-age=0
Cookie: wordpress_test_cookie=WP%20Cookie%20check; PHPSESSID=bd6a91c5ef1ccc58d24b7d9d080c4c39
Upgrade-Insecure-Requests: 1
Sec-CH-UA: "Google Chrome";v="133", "Not=A?Brand";v="8", "Chromium";v="133"
Sec-CH-UA-Platform: "Linux"
Sec-CH-UA-Mobile: ?0
Content-Length: 0
Origin: https://ggxymzdiysmq.com
```

Response

```
HTTP/2 200 OK
X-Powered-By: PHP/7.4.33
X-Robots-Tag: noindex
X-Content-Type-Options: nosniff
Access-Control-Expose-Headers: X-WP-Total, X-WP-TotalPages, Link
Access-Control-Allow-Headers: Authorization, X-WP-Nonce, Content-Disposition, Content-MD5, Content-Type
Link: <https://www.testreport.com/solutions/cloud/>; rel="alternate"; type=text/html
Allow: GET
Access-Control-Allow-Origin: https://ggxymzdiysmq.com
Access-Control-Allow-Methods: OPTIONS, GET, POST, PUT, PATCH, DELETE
Access-Control-Allow-Credentials: true
Vary: Origin,Accept-Encoding
Content-Length: 1840
Content-Type: application/json; charset=UTF-8
Date: Mon, 03 Mar 2025 18:25:22 GMT
Server: Apache

{"id":30,"date":"2021-03-03T15:58:21","date_gmt":"2021-03-03T15:58:21","guid":{"rendered":"https:\/\/www.testreport.com\/?page_id=30"},"modified":"2021-03-03T15:58:21","modified_gmt":"2021-03-03T15:58:2
...[SNIP]...
```

1.9. https://www.testreport.com/wp-json/wp/v2/pages/32

Summary

Severity:	**High**
Confidence:	**Certain**
Host:	**https://www.testreport.com**
Path:	**/wp-json/wp/v2/pages/32**

Issue detail

The application implements an HTML5 cross-origin resource sharing (CORS) policy for this request that allows access from any domain.

The application allowed access from the requested origin **https://hkklpqrjuxxw.com**

Request

```
GET /wp-json/wp/v2/pages/32 HTTP/2
Host: www.testreport.com
Accept-Encoding: gzip, deflate, br
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Language: en-US;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 Safari/537.36
Connection: close
Cache-Control: max-age=0
Cookie: wordpress_test_cookie=WP%20Cookie%20check; PHPSESSID=da08ad7a267d07f875842e2bff9ebd42
Upgrade-Insecure-Requests: 1
Sec-CH-UA: "Google Chrome";v="133", "Not=A?Brand";v="8", "Chromium";v="133"
Sec-CH-UA-Platform: "Linux"
Sec-CH-UA-Mobile: ?0
Content-Length: 0
Origin: https://hkklpqrjuxxw.com
```

Response

```
HTTP/2 200 OK
X-Powered-By: PHP/7.4.33
X-Robots-Tag: noindex
X-Content-Type-Options: nosniff
Access-Control-Expose-Headers: X-WP-Total, X-WP-TotalPages, Link
Access-Control-Allow-Headers: Authorization, X-WP-Nonce, Content-Disposition, Content-MD5, Content-Type
Link: <https://www.testreport.com/solutions/data-management/>; rel="alternate"; type=text/html
Allow: GET
Access-Control-Allow-Origin: https://hkklpqrjuxxw.com
Access-Control-Allow-Methods: OPTIONS, GET, POST, PUT, PATCH, DELETE
Access-Control-Allow-Credentials: true
Vary: Origin,Accept-Encoding
Content-Length: 1870
Content-Type: application/json; charset=UTF-8
Date: Mon, 03 Mar 2025 18:25:22 GMT
Server: Apache

{"id":32,"date":"2021-03-03T15:59:15","date_gmt":"2021-03-03T15:59:15","guid":{"rendered":"https:\/\/www.testreport.com\/?page_id=32"},"modified":"2021-03-03T16:00:39","modified_gmt":"2021-03-03T16:00:3
...[SNIP]...
```

1.10. https://www.testreport.com/wp-json/wp/v2/pages/39

Summary

Severity:	**High**

Confidence:	**Certain**
Host:	**https://www.testreport.com**
Path:	**/wp-json/wp/v2/pages/39**

Issue detail

The application implements an HTML5 cross-origin resource sharing (CORS) policy for this request that allows access from any domain.

The application allowed access from the requested origin **https://ylafjrirtyic.com**

Request

```
GET /wp-json/wp/v2/pages/39 HTTP/2
Host: www.testreport.com
Accept-Encoding: gzip, deflate, br
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Language: en-US;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 Safari/537.36
Connection: close
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Sec-CH-UA: "Google Chrome";v="133", "Not=A?Brand";v="8", "Chromium";v="133"
Sec-CH-UA-Platform: "Linux"
Sec-CH-UA-Mobile: ?0
Content-Length: 0
Origin: https://ylafjrirtyic.com
```

Response

```
HTTP/2 200 OK
X-Powered-By: PHP/7.4.33
X-Robots-Tag: noindex
X-Content-Type-Options: nosniff
Access-Control-Expose-Headers: X-WP-Total, X-WP-TotalPages, Link
Access-Control-Allow-Headers: Authorization, X-WP-Nonce, Content-Disposition, Content-MD5, Content-Type
Link: <https://www.testreport.com/news/>; rel="alternate"; type=text/html
Allow: GET
Access-Control-Allow-Origin: https://ylafjrirtyic.com
Access-Control-Allow-Methods: OPTIONS, GET, POST, PUT, PATCH, DELETE
Access-Control-Allow-Credentials: true
Vary: Origin,Accept-Encoding
Content-Length: 18514
Content-Type: application/json; charset=UTF-8
Date: Mon, 03 Mar 2025 18:49:42 GMT
Server: Apache

{"id":39,"date":"2021-03-03T16:02:46","date_gmt":"2021-03-03T16:02:46","guid":{"rendered":"https:\/\/www.testreport.com\/?page_id=39"},"modified":"2021-12-06T10:36:38","modified_gmt":"2021-12-06T05:06:3
...[SNIP]...
```

1.11. https://www.testreport.com/wp-json/wp/v2/pages/41

Summary

Severity:	**High**
Confidence:	**Certain**
Host:	**https://www.testreport.com**
Path:	**/wp-json/wp/v2/pages/41**

Issue detail

The application implements an HTML5 cross-origin resource sharing (CORS) policy for this request that allows access from any domain.

The application allowed access from the requested origin **https://avioszzrxuco.com**

Request

```
GET /wp-json/wp/v2/pages/41 HTTP/2
Host: www.testreport.com
Accept-Encoding: gzip, deflate, br
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Language: en-US;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 Safari/537.36
Connection: close
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Sec-CH-UA: "Google Chrome";v="133", "Not=A?Brand";v="8", "Chromium";v="133"
Sec-CH-UA-Platform: "Linux"
Sec-CH-UA-Mobile: ?0
Content-Length: 0
Origin: https://avioszzrxuco.com
```

Response

```
HTTP/2 200 OK
X-Powered-By: PHP/7.4.33
X-Robots-Tag: noindex
X-Content-Type-Options: nosniff
Access-Control-Expose-Headers: X-WP-Total, X-WP-TotalPages, Link
Access-Control-Allow-Headers: Authorization, X-WP-Nonce, Content-Disposition, Content-MD5, Content-Type
Link: <https://www.testreport.com/contact/>; rel="alternate"; type=text/html
Allow: GET
Access-Control-Allow-Origin: https://avioszzrxuco.com
Access-Control-Allow-Methods: OPTIONS, GET, POST, PUT, PATCH, DELETE
Access-Control-Allow-Credentials: true
Vary: Origin,Accept-Encoding
Content-Length: 16556
Content-Type: application/json; charset=UTF-8
Date: Mon, 03 Mar 2025 18:51:56 GMT
Server: Apache

{"id":41,"date":"2021-03-03T16:03:30","date_gmt":"2021-03-03T16:03:30","guid":{"rendered":"https:\/\/www.testreport.com\/?page_id=41"},"modified":"2022-01-13T11:51:05","modified_gmt":"2022-01-13T06:21:0
...[SNIP]...
```

1.12. https://www.testreport.com/wp-json/wp/v2/posts/401

Summary

Severity:	**High**
Confidence:	**Certain**
Host:	**https://www.testreport.com**
Path:	**/wp-json/wp/v2/posts/401**

Issue detail

The application implements an HTML5 cross-origin resource sharing (CORS) policy for this request that allows access from any domain.

The application allowed access from the requested origin **https://bkgwwauffzor.com**

Request

```
GET /wp-json/wp/v2/posts/401 HTTP/2
Host: www.testreport.com
Accept-Encoding: gzip, deflate, br
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Language: en-US;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 Safari/537.36
Connection: close
Cache-Control: max-age=0
Cookie: wordpress_test_cookie=WP%20Cookie%20check; PHPSESSID=f694603aa5742e3c2b7ecb42ca46b9cb
Upgrade-Insecure-Requests: 1
Sec-CH-UA: "Google Chrome";v="133", "Not=A?Brand";v="8", "Chromium";v="133"
Sec-CH-UA-Platform: "Linux"
Sec-CH-UA-Mobile: ?0
Content-Length: 0
Origin: https://bkgwwauffzor.com
```

Response

```
HTTP/2 200 OK
X-Powered-By: PHP/7.4.33
X-Robots-Tag: noindex
X-Content-Type-Options: nosniff
Access-Control-Expose-Headers: X-WP-Total, X-WP-TotalPages, Link
Access-Control-Allow-Headers: Authorization, X-WP-Nonce, Content-Disposition, Content-MD5, Content-Type
Link: <https://www.testreport.com/cyber-security/apple-users-targeted-by-mysterious-malware/>; rel="alternate"; type=text/html
Allow: GET
Access-Control-Allow-Origin: https://bkgwwauffzor.com
Access-Control-Allow-Methods: OPTIONS, GET, POST, PUT, PATCH, DELETE
Access-Control-Allow-Credentials: true
Vary: Origin,Accept-Encoding
Content-Length: 6374
Content-Type: application/json; charset=UTF-8
Date: Mon, 03 Mar 2025 18:24:47 GMT
```

Server: Apache

{"id":401,"date":"2021-03-06T16:15:10","date_gmt":"2021-03-06T10:45:10","guid":{"rendered":"https:\/\/www.testreport.com\/?p=401"},"modified":"2021-03-06T16:58:18","modified_gmt":"2021-03-06T11:28:18","
...[SNIP]...

1.13. https://www.testreport.com/wp-json/wp/v2/posts/404

Summary

Severity:	**High**
Confidence:	**Certain**
Host:	**https://www.testreport.com**
Path:	**/wp-json/wp/v2/posts/404**

Issue detail

The application implements an HTML5 cross-origin resource sharing (CORS) policy for this request that allows access from any domain.

The application allowed access from the requested origin **https://dmhhzxhtebck.com**

Request

```
GET /wp-json/wp/v2/posts/404 HTTP/2
Host: www.testreport.com
Accept-Encoding: gzip, deflate, br
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Language: en-US;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 Safari/537.36
Connection: close
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Sec-CH-UA: "Google Chrome";v="133", "Not=A?Brand";v="8", "Chromium";v="133"
Sec-CH-UA-Platform: "Linux"
Sec-CH-UA-Mobile: ?0
Content-Length: 0
Origin: https://dmhhzxhtebck.com
```

Response

```
HTTP/2 200 OK
X-Powered-By: PHP/7.4.33
X-Robots-Tag: noindex
X-Content-Type-Options: nosniff
Access-Control-Expose-Headers: X-WP-Total, X-WP-TotalPages, Link
Access-Control-Allow-Headers: Authorization, X-WP-Nonce, Content-Disposition, Content-MD5, Content-Type
Link: <https://www.testreport.com/data-management/what-is-data-analytics-analyzing-and-managing-data-for-decisions/>; rel="alternate"; type=text/html
Allow: GET
Access-Control-Allow-Origin: https://dmhhzxhtebck.com
```

Access-Control-Allow-Methods: OPTIONS, GET, POST, PUT, PATCH, DELETE
Access-Control-Allow-Credentials: true
Vary: Origin,Accept-Encoding
Content-Length: 5125
Content-Type: application/json; charset=UTF-8
Date: Mon, 03 Mar 2025 18:36:47 GMT
Server: Apache

{"id":404,"date":"2021-03-06T16:18:25","date_gmt":"2021-03-06T10:48:25","guid":{"rendered":"https:\/\/www.testreport.com\/?p=404"},"modified":"2021-03-06T16:56:00","modified_gmt":"2021-03-06T11:26:00","
...[SNIP]...

2. TLS cookie without secure flag set

Summary

Severity:	**Medium**
Confidence:	**Firm**
Host:	**https://www.testreport.com**
Path:	**/wp-login.php**

Issue detail

The following cookie was issued by the application and does not have the secure flag set:

- **PHPSESSID**

The cookie appears to contain a session token, which may increase the risk associated with this issue. You should review the contents of the cookie to determine its function.

Issue background

If the secure flag is set on a cookie, then browsers will not submit the cookie in any requests that use an unencrypted HTTP connection, thereby preventing the cookie from being trivially intercepted by an attacker monitoring network traffic. If the secure flag is not set, then the cookie will be transmitted in clear-text if the user visits any HTTP URLs within the cookie's scope. An attacker may be able to induce this event by feeding a user suitable links, either directly or via another web site. Even if the domain that issued the cookie does not host any content that is accessed over HTTP, an attacker may be able to use links of the form http://example.com:443/ to perform the same attack.

To exploit this vulnerability, an attacker must be suitably positioned to eavesdrop on the victim's network traffic. This scenario typically occurs when a client communicates with the server over an insecure connection such as public Wi-Fi, or a corporate or home network that is shared with a compromised computer. Common defenses such as switched networks are not sufficient to prevent this. An attacker situated in the user's ISP or the application's hosting infrastructure could also perform this attack. Note that an advanced adversary could potentially target any connection made over the Internet's core infrastructure.

Issue remediation

The secure flag should be set on all cookies that are used for transmitting sensitive data when accessing content over HTTPS. If cookies are used to transmit session tokens, then areas of the application that are accessed over HTTPS should employ their own session handling mechanism, and the session tokens used should never be transmitted over unencrypted communications.

Vulnerability classifications

- CWE-614: Sensitive Cookie in HTTPS Session Without 'Secure' Attribute

Request

```
GET /wp-login.php?redirect_to=https%3A%2F%2Fwww.testreport.com%2Fwp-admin%2F&reauth=1 HTTP/2
Host: www.testreport.com
Accept-Encoding: gzip, deflate, br
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Language: en-US;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 Safari/537.36
Connection: close
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Sec-CH-UA: "Google Chrome";v="133", "Not=A?Brand";v="8", "Chromium";v="133"
Sec-CH-UA-Platform: "Linux"
Sec-CH-UA-Mobile: ?0
```

Response

```
HTTP/2 200 OK
X-Powered-By: PHP/7.4.33
X-Frame-Options: SAMEORIGIN
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Set-Cookie: wordpress_test_cookie=WP%20Cookie%20check; path=/; secure
Set-Cookie: PHPSESSID=8fe3bb4e3cba504cd1a67d00d5e2c504; path=/
Set-Cookie: wordpress_59763a0e8e3408728317dee177fecd47=%20; expires=Sat, 02-Mar-2024 12:45:39 GMT; Max-Age=0; path=/wp-admin
Set-Cookie: wordpress_sec_59763a0e8e3408728317dee177fecd47=%20; expires=Sat, 02-Mar-2024 12:45:39 GMT; Max-Age=0; path=/wp-admin
Set-Cookie: wordpress_59763a0e8e3408728317dee177fecd47=%20; expires=Sat, 02-Mar-2024 12:45:39 GMT; Max-Age=0; path=/wp-content/plugins
Set-Cookie: wordpress_sec_59763a0e8e3408728317dee177fecd47=%20; expires=Sat, 02-Mar-2024 12:45:39 GMT; Max-Age=0; path=/wp-content/plugins
Set-Cookie: wordpress_logged_in_59763a0e8e3408728317dee177fecd47=%20; expires=Sat, 02-Mar-2024 12:45:39 GMT; Max-Age=0; path=/
Set-Cookie: wordpress_logged_in_59763a0e8e3408728317dee177fecd47=%20; expires=Sat, 02-Mar-2024 12:45:39 GMT; Max-Age=0; path=/
Set-Cookie: wp-settings-0=%20; expires=Sat, 02-Mar-2024 12:45:39 GMT; Max-Age=0; path=/
Set-Cookie: wp-settings-time-0=%20; expires=Sat, 02-Mar-2024 12:45:39 GMT; Max-Age=0; path=/
Set-Cookie: wordpress_59763a0e8e3408728317dee177fecd47=%20; expires=Sat, 02-Mar-2024 12:45:39 GMT; Max-Age=0; path=/
Set-Cookie: wordpress_59763a0e8e3408728317dee177fecd47=%20; expires=Sat, 02-Mar-2024 12:45:39 GMT; Max-Age=0; path=/
Set-Cookie: wordpress_sec_59763a0e8e3408728317dee177fecd47=%20; expires=Sat, 02-Mar-2024 12:45:39 GMT; Max-Age=0; path=/
Set-Cookie: wordpress_sec_59763a0e8e3408728317dee177fecd47=%20; expires=Sat, 02-Mar-2024 12:45:39 GMT; Max-Age=0; path=/
Set-Cookie: wordpressuser_59763a0e8e3408728317dee177fecd47=%20; expires=Sat, 02-Mar-2024 12:45:39 GMT; Max-Age=0; path=/
Set-Cookie: wordpresspass_59763a0e8e3408728317dee177fecd47=%20; expires=Sat, 02-Mar-2024 12:45:39 GMT; Max-Age=0; path=/
Set-Cookie: wordpressuser_59763a0e8e3408728317dee177fecd47=%20; expires=Sat, 02-Mar-2024 12:45:39 GMT; Max-Age=0; path=/
Set-Cookie: wordpresspass_59763a0e8e3408728317dee177fecd47=%20; expires=Sat, 02-Mar-2024 12:45:39 GMT; Max-Age=0; path=/
```

```
Set-Cookie: wp-postpass_59763a0e8e3408728317dee177fecd47=%20; expires=Sat, 02-Mar-2024 12:45:39 GMT; Max-Age=0;
path=/
Vary: Accept-Encoding
Content-Length: 7130
Content-Type: text/html; charset=UTF-8
Date: Sun, 02 Mar 2025 12:45:39 GMT
Server: Apache

<!DOCTYPE html>
  <html lang="en">
  <head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
  <title>Log In &lsaquo; testreport Infosystem Pvt Ltd &#8212; WordPress</title>
  <meta na
...[SNIP]...
```

3. Cross-origin resource sharing: unencrypted origin trusted

Summary

Severity:	**Low**
Confidence:	**Certain**
Host:	**https://www.testreport.com**
Path:	**/wp-admin/admin-ajax.php**

Issue detail

The application implements an HTML5 cross-origin resource sharing (CORS) policy for this request which trusts websites accessed using unencrypted communications.

Since the Vary: Origin header was not present in the response, reverse proxies and intermediate servers may cache it. This may enable an attacker to carry out cache poisoning attacks.

Issue background

An HTML5 cross-origin resource sharing (CORS) policy controls whether and how content running on other domains can perform two-way interaction with the domain that publishes the policy. The policy is fine-grained and can apply access controls per-request based on the URL and other features of the request.

If a site allows interaction from an origin that uses unencrypted HTTP communications, then it is vulnerable to an attacker who is in a position to view and modify a user's unencrypted network traffic. The attacker can control the responses from unencrypted origins, thereby injecting content that is able to interact with the application that publishes the policy. This means that the application is effectively extending trust to all such attackers, thereby undoing much of the benefit of using HTTPS communications.

Issue remediation

Only trust origins that use encrypted HTTPS communications.

Vulnerability classifications

- CWE-942: Overly Permissive Cross-domain Whitelist

Request

```
POST /wp-admin/admin-ajax.php HTTP/2
Host: www.testreport.com
Accept-Encoding: gzip, deflate, br
Accept: */*
Accept-Language: en-US;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 Safari/537.36
Connection: close
Cache-Control: max-age=0
Cookie: PHPSESSID=93605899a667ae301f9eb3a9ed7b569c; wordpress_test_cookie=WP%20Cookie%20check
Origin: http://www.testreport.com
X-Requested-With: XMLHttpRequest
Referer: https://www.testreport.com/wp-login.php
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Sec-CH-UA: "Google Chrome";v="133", "Not=A?Brand";v="8", "Chromium";v="133"
Sec-CH-UA-Platform: "Linux"
Sec-CH-UA-Mobile: ?0
Content-Length: 52

action=get_remaining_attempts_message&sec=2244f795b7
```

Response

```
HTTP/2 200 OK
X-Powered-By: PHP/7.4.33
Access-Control-Allow-Origin: http://www.testreport.com
Access-Control-Allow-Credentials: true
X-Robots-Tag: noindex
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Referrer-Policy: strict-origin-when-cross-origin
Pragma: no-cache
Expires: Wed, 11 Jan 1984 05:00:00 GMT
Cache-Control: no-cache, must-revalidate, max-age=0
Vary: Accept-Encoding
Content-Length: 26
Content-Type: application/json; charset=UTF-8
Date: Sun, 02 Mar 2025 13:01:42 GMT
Server: Apache

{"success":true,"data":""}
```

4. Cookie without HttpOnly flag set

Summary

Severity:	**Low**
Confidence:	**Firm**
Host:	**https://www.testreport.com**
Path:	**/wp-login.php**

Issue detail

The following cookies were issued by the application and do not have the HttpOnly flag set:

- **PHPSESSID**
- wordpress_test_cookie

The highlighted cookie appears to contain a session token, which may increase the risk associated with this issue. You should review the contents of the cookies to determine their function.

Issue background

If the HttpOnly attribute is set on a cookie, then the cookie's value cannot be read or set by client-side JavaScript. This measure makes certain client-side attacks, such as cross-site scripting, slightly harder to exploit by preventing them from trivially capturing the cookie's value via an injected script.

Issue remediation

There is usually no good reason not to set the HttpOnly flag on all cookies. Unless you specifically require legitimate client-side scripts within your application to read or set a cookie's value, you should set the HttpOnly flag by including this attribute within the relevant Set-cookie directive.

You should be aware that the restrictions imposed by the HttpOnly flag can potentially be circumvented in some circumstances, and that numerous other serious attacks can be delivered by client-side script injection, aside from simple cookie stealing.

Vulnerability classifications

- CWE-16: Configuration
- CAPEC-31: Accessing/Intercepting/Modifying HTTP Cookies

Request

```
GET /wp-login.php?redirect_to=https%3A%2F%2Fwww.testreport.com%2Fwp-admin%2F&reauth=1 HTTP/2
Host: www.testreport.com
Accept-Encoding: gzip, deflate, br
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Language: en-US;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 Safari/537.36
Connection: close
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Sec-CH-UA: "Google Chrome";v="133", "Not=A?Brand";v="8", "Chromium";v="133"
Sec-CH-UA-Platform: "Linux"
Sec-CH-UA-Mobile: ?0
```

Response

```
HTTP/2 200 OK
X-Powered-By: PHP/7.4.33
X-Frame-Options: SAMEORIGIN
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Set-Cookie: wordpress_test_cookie=WP%20Cookie%20check; path=/; secure
Set-Cookie: PHPSESSID=8fe3bb4e3cba504cd1a67d00d5e2c504; path=/
Set-Cookie: wordpress_59763a0e8e3408728317dee177fecd47=%20; expires=Sat, 02-Mar-2024 12:45:39 GMT; Max-Age=0;
```

path=/wp-admin
Set-Cookie: wordpress_sec_59763a0e8e3408728317dee177fecd47=%20; expires=Sat, 02-Mar-2024 12:45:39 GMT; Max-Age=0; path=/wp-admin
Set-Cookie: wordpress_59763a0e8e3408728317dee177fecd47=%20; expires=Sat, 02-Mar-2024 12:45:39 GMT; Max-Age=0; path=/wp-content/plugins
Set-Cookie: wordpress_sec_59763a0e8e3408728317dee177fecd47=%20; expires=Sat, 02-Mar-2024 12:45:39 GMT; Max-Age=0; path=/wp-content/plugins
Set-Cookie: wordpress_logged_in_59763a0e8e3408728317dee177fecd47=%20; expires=Sat, 02-Mar-2024 12:45:39 GMT; Max-Age=0; path=/
Set-Cookie: wordpress_logged_in_59763a0e8e3408728317dee177fecd47=%20; expires=Sat, 02-Mar-2024 12:45:39 GMT; Max-Age=0; path=/
Set-Cookie: wp-settings-0=%20; expires=Sat, 02-Mar-2024 12:45:39 GMT; Max-Age=0; path=/
Set-Cookie: wp-settings-time-0=%20; expires=Sat, 02-Mar-2024 12:45:39 GMT; Max-Age=0; path=/
Set-Cookie: wordpress_59763a0e8e3408728317dee177fecd47=%20; expires=Sat, 02-Mar-2024 12:45:39 GMT; Max-Age=0; path=/
Set-Cookie: wordpress_59763a0e8e3408728317dee177fecd47=%20; expires=Sat, 02-Mar-2024 12:45:39 GMT; Max-Age=0; path=/
Set-Cookie: wordpress_sec_59763a0e8e3408728317dee177fecd47=%20; expires=Sat, 02-Mar-2024 12:45:39 GMT; Max-Age=0; path=/
Set-Cookie: wordpress_sec_59763a0e8e3408728317dee177fecd47=%20; expires=Sat, 02-Mar-2024 12:45:39 GMT; Max-Age=0; path=/
Set-Cookie: wordpressuser_59763a0e8e3408728317dee177fecd47=%20; expires=Sat, 02-Mar-2024 12:45:39 GMT; Max-Age=0; path=/
Set-Cookie: wordpresspass_59763a0e8e3408728317dee177fecd47=%20; expires=Sat, 02-Mar-2024 12:45:39 GMT; Max-Age=0; path=/
Set-Cookie: wordpressuser_59763a0e8e3408728317dee177fecd47=%20; expires=Sat, 02-Mar-2024 12:45:39 GMT; Max-Age=0; path=/
Set-Cookie: wordpresspass_59763a0e8e3408728317dee177fecd47=%20; expires=Sat, 02-Mar-2024 12:45:39 GMT; Max-Age=0; path=/
Set-Cookie: wp-postpass_59763a0e8e3408728317dee177fecd47=%20; expires=Sat, 02-Mar-2024 12:45:39 GMT; Max-Age=0; path=/
Vary: Accept-Encoding
Content-Length: 7130
Content-Type: text/html; charset=UTF-8
Date: Sun, 02 Mar 2025 12:45:39 GMT
Server: Apache

<!DOCTYPE html>
<html lang="en">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>Log In ‹ testreport Infosystem Pvt Ltd — WordPress</title>
<meta na
...[SNIP]...

5. Client-side HTTP parameter pollution (reflected)

There are 2 instances of this issue:

- / [name of an arbitrarily supplied URL parameter]
- /page/2/ [name of an arbitrarily supplied URL parameter]

Issue background

Client-side HTTP parameter pollution (HPP) vulnerabilities arise when an application embeds user input in URLs in an unsafe manner. An attacker can use this vulnerability to construct a URL that, if visited by another application user, will modify URLs within the response by inserting additional query string parameters and sometimes overriding existing ones. This may result in links and

forms having unexpected side effects. For example, it may be possible to modify an invitation form using HPP so that the invitation is delivered to an unexpected recipient.

The security impact of this issue depends largely on the nature of the application functionality. Even if it has no direct impact on its own, an attacker may use it in conjunction with other vulnerabilities to escalate their overall severity.

Issue remediation

Ensure that user input is URL-encoded before it is embedded in a URL.

Vulnerability classifications

- CWE-233: Improper Handling of Parameters
- CWE-20: Improper Input Validation
- CAPEC-460: HTTP Parameter Pollution (HPP)

5.1. https://www.testreport.com/ [name of an arbitrarily supplied URL parameter]

Summary

Severity:	**Low**
Confidence:	**Firm**
Host:	**https://www.testreport.com**
Path:	**/**

Issue detail

The name of an arbitrarily supplied URL parameter is copied into the response within the query string of a URL.

The payload **umj&zjx=1** was submitted in the name of an arbitrarily supplied URL parameter. This input was echoed as **umj&zjx=1** within the "href" attribute of an "a" tag.

This proof-of-concept attack demonstrates that it is possible to inject arbitrary query string parameters into URLs in the application's response.

Request

```
GET /?s=&umj%26zjx%3d1=1 HTTP/2
Host: www.testreport.com
Accept-Encoding: gzip, deflate, br
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Language: en-US;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 Safari/537.36
Connection: close
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Referer: https://www.testreport.com/category/data-management/
Sec-CH-UA: "Google Chrome";v="133", "Not=A?Brand";v="8", "Chromium";v="133"
Sec-CH-UA-Platform: "Linux"
```

```
Sec-CH-UA-Mobile: ?0
Content-Length: 0
```

Response

```
HTTP/2 200 OK
X-Powered-By: PHP/7.4.33
Link: <https://www.testreport.com/wp-json/>; rel="https://api.w.org/"
Vary: Accept-Encoding
Content-Length: 46231
Content-Type: text/html; charset=UTF-8
Date: Sun, 02 Mar 2025 13:08:18 GMT
Server: Apache

<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1, minimum-scale=1">
  <link rel="profile" href="http://gmpg.org/xfn/1
...[SNIP]...
<a class="page-numbers" href="https://www.testreport.com/page/2/?s&#038;umj&#038;zjx=1%3D1">
...[SNIP]...
```

5.2. https://www.testreport.com/page/2/ [name of an arbitrarily supplied URL parameter]

Summary

Severity:	**Low**
Confidence:	**Firm**
Host:	**https://www.testreport.com**
Path:	**/page/2/**

Issue detail

The name of an arbitrarily supplied URL parameter is copied into the response within the query string of a URL.

The payload **vbb&lux=1** was submitted in the name of an arbitrarily supplied URL parameter. This input was echoed as **vbb&lux=1** within the "href" attribute of an "a" tag.

This proof-of-concept attack demonstrates that it is possible to inject arbitrary query string parameters into URLs in the application's response.

Request

```
GET /page/2/?s&vbb%26lux%3d1=1 HTTP/2
Host: www.testreport.com
Accept-Encoding: gzip, deflate, br
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-
```

```
exchange;v=b3;q=0.7
Accept-Language: en-US;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 Safari/537.36
Connection: close
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Referer: https://www.testreport.com/?s=
Sec-CH-UA: "Google Chrome";v="133", "Not=A?Brand";v="8", "Chromium";v="133"
Sec-CH-UA-Platform: "Linux"
Sec-CH-UA-Mobile: ?0
Content-Length: 0
```

Response

```
HTTP/2 200 OK
X-Powered-By: PHP/7.4.33
Link: <https://www.testreport.com/wp-json/>; rel="https://api.w.org/"
Vary: Accept-Encoding
Content-Length: 35001
Content-Type: text/html; charset=UTF-8
Date: Sun, 02 Mar 2025 13:12:35 GMT
Server: Apache

<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1, minimum-scale=1">
  <link rel="profile" href="http://gmpg.org/xfn/1
...[SNIP]...
<a class="prev page-numbers" href="https://www.testreport.com/?s&#038;vbb&#038;lux=1%3D1">
...[SNIP]...
```

6. Unencrypted communications

Summary

Severity:	**Low**
Confidence:	**Certain**
Host:	**http://www.testreport.com**
Path:	**/**

Issue description

The application allows users to connect to it over unencrypted connections. An attacker suitably positioned to view a legitimate user's network traffic could record and monitor their interactions with the application and obtain any information the user supplies. Furthermore, an attacker able to modify traffic could use the application as a platform for attacks against its users and third-party websites. Unencrypted connections have been exploited by ISPs and governments to track users, and to inject adverts and malicious JavaScript. Due to these concerns, web browser vendors are planning to visually flag unencrypted connections as hazardous.

To exploit this vulnerability, an attacker must be suitably positioned to eavesdrop on the victim's network traffic. This scenario typically occurs when a client communicates with the server over an insecure connection such as public Wi-Fi, or a corporate or home network that is shared with a compromised computer. Common defenses such as switched networks are not sufficient to prevent this. An attacker situated in the user's ISP or the application's hosting infrastructure could also perform this attack. Note that an advanced adversary could potentially target any connection made over the Internet's core infrastructure.

Please note that using a mixture of encrypted and unencrypted communications is an ineffective defense against active attackers, because they can easily remove references to encrypted resources when these references are transmitted over an unencrypted connection.

Issue remediation

Applications should use transport-level encryption (SSL/TLS) to protect all communications passing between the client and the server. The Strict-Transport-Security HTTP header should be used to ensure that clients refuse to access the server over an insecure connection.

Vulnerability classifications

- CWE-326: Inadequate Encryption Strength
- CAPEC-94: Man in the Middle Attack
- CAPEC-157: Sniffing Attacks

7. Strict transport security not enforced

Summary

Severity:	**Low**
Confidence:	**Certain**
Host:	**https://www.testreport.com**
Path:	**/**

Issue detail

This issue was found in multiple locations under the reported path.

Issue background

The application fails to prevent users from connecting to it over unencrypted connections. An attacker able to modify a legitimate user's network traffic could bypass the application's use of SSL/TLS encryption, and use the application as a platform for attacks against its users. This attack is performed by rewriting HTTPS links as HTTP, so that if a targeted user follows a link to the site from an HTTP page, their browser never attempts to use an encrypted connection. The sslstrip tool automates this process.

To exploit this vulnerability, an attacker must be suitably positioned to intercept and modify the victim's network traffic.This scenario typically occurs when a client communicates with the server over an insecure connection such as public Wi-Fi, or a corporate or home network that is shared with a compromised computer. Common defenses such as switched networks are not sufficient to prevent this. An attacker situated in the user's ISP or the application's hosting infrastructure could also perform this attack. Note that an advanced adversary could potentially target any connection made over the Internet's core infrastructure.

Issue remediation

The application should instruct web browsers to only access the application using HTTPS. To do this, enable HTTP Strict Transport Security (HSTS) by adding a response header with the name 'Strict-Transport-Security' and the value 'max-age=expireTime', where

expireTime is the time in seconds that browsers should remember that the site should only be accessed using HTTPS. Consider adding the 'includeSubDomains' flag if appropriate.

Note that because HSTS is a "trust on first use" (TOFU) protocol, a user who has never accessed the application will never have seen the HSTS header, and will therefore still be vulnerable to SSL stripping attacks. To mitigate this risk, you can optionally add the 'preload' flag to the HSTS header, and submit the domain for review by browser vendors.

Vulnerability classifications

- CWE-523: Unprotected Transport of Credentials
- CAPEC-94: Man in the Middle Attack
- CAPEC-157: Sniffing Attacks

Request

```
POST /wp-json/content-forms/v1/submit HTTP/2
Host: www.testreport.com
Accept-Encoding: gzip, deflate, br
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-US;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 Safari/537.36
Connection: close
Cache-Control: max-age=0
Origin: https://www.testreport.com
X-Requested-With: XMLHttpRequest
Referer: https://www.testreport.com/contact/
X-WP-Nonce: 89805002eb
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Sec-CH-UA: "Google Chrome";v="133", "Not=A?Brand";v="8", "Chromium";v="133"
Sec-CH-UA-Platform: "Linux"
Sec-CH-UA-Mobile: ?0
Content-Length: 229

form_type=contact&form_id=b64b0df&post_id=41&form_builder=elementor&data%5Bb64b0df%5D%5Bname%5D=VHBoIy&dat
a%5Bb64b0df%5D%5Bemail%5D=xVJDUX&data%5Bb64b0df%5D%5Bphone%5D=gnFuzJ&data%5Bb64b0df%5D%5Bmessa
...[SNIP]...
```

Response

```
HTTP/2 400 Bad Request
X-Powered-By: PHP/7.4.33
X-Robots-Tag: noindex
Link: <https://www.testreport.com/wp-json/>; rel="https://api.w.org/"
X-Content-Type-Options: nosniff
Access-Control-Expose-Headers: X-WP-Total, X-WP-TotalPages, Link
Access-Control-Allow-Headers: Authorization, X-WP-Nonce, Content-Disposition, Content-MD5, Content-Type
X-Wp-Nonce: 89805002eb
Allow: POST
Access-Control-Allow-Origin: https://www.testreport.com
Access-Control-Allow-Methods: OPTIONS, GET, POST, PUT, PATCH, DELETE
Access-Control-Allow-Credentials: true
Vary: Origin,Accept-Encoding
Content-Length: 44
Content-Type: application/json; charset=UTF-8
Date: Sun, 02 Mar 2025 12:45:30 GMT
Server: Apache

{"success":false,"message":"Invalid email."}
```