Defi WebApp

Defi App

HALBORN

Defi WebApp - Defi App

Prepared by: HALBORN

Last Updated 03/06/2025

Date of Engagement by: January 20th, 2025 - February 6th, 2025

Summary

100% OF ALL REPORTED FINDINGS HAVE BEEN ADDRESSED

ALL FINDINGS	CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
11	0	0	9	2	0

TABLE OF CONTENTS

7.11 Harcoded secrets in repository

1. Introduction

`Defi App` engaged Halborn to conduct a security assessment of 4 GitHub repositories, beginning on *2025-01-20* and ending on *2025-02-06.* The security assessment was scoped to the assets provided to the Halborn team.

2. Assessment Summary

The team at Halborn was provided two and a half weeks for the engagement and assigned a full-time security engineer to verify the security of the web application. The security engineer is a penetration testing expert with advanced knowledge in web, recon, discovery & infrastructure penetration testing and blockchain protocols.

The goals for this assessment are:

- Assess the overall security of Defi App

- Improve the security of the application by testing it as white-box approach

- Identify potential security issues that could be affecting the web application

In summary, Halborn identified `9 medium` and `2 low` security issues that were addressed and acknowledged by the `Defi App team.`

Several **medium-severity** vulnerabilities related to HTTP headers, session management, secret handling and environment isolation. Lack of the correct security headers could enable attacks such as CSRF, Clickjacking and XSS. Having session tokens stored in the browser's local storage and long expiration times increase the risk of account takeover in the case of a browser vulnerability. The use of a single AWS account for all environments further amplifies the potential impact of a breach, as it could allow exploiting a development environment to access production resources.

The **low-severity** issues relate to issues that pose very low impact by themselves but combined with other vulnerabilities could have a greater impact. Storing session tokens in a browser's local storage could allow attackers exploiting other browser vulnerabilities to exfiltrate and use those credentials.

It is recommended to resolve all the security issues listed in the document to improve the security health of the application and its underlying infrastructure.

3. Test Approach And Methodology

Halborn performed a combination of manual and automated security testing to balance efficiency, timeliness, practicality, and accuracy in regard to the scope of this assessment. While manual testing is recommended to uncover flaws in logic, process and implementation; automated testing techniques help enhance coverage of the code and can quickly identify items that do not follow security best practices.

Several tests were carried out during the assessment; including, but not limited to:

- Vulnerable or outdated software or dependencies
- Access Handling
- Input Handling
- Source code audit
- Fuzzing of input parameters
- Sensitive information disclosure
- Application Logic Flaws
- Identify other potential vulnerabilities that may pose a risk to `Defi App`

4. RISK METHODOLOGY

Vulnerabilities or issues observed by Halborn are ranked based on the risk assessment methodology by measuring the LIKELIHOOD of a security incident and the IMPACT should an incident occur. This framework works for communicating the characteristics and impacts of technology vulnerabilities. The quantitative model ensures repeatable and accurate measurement while enabling users to see the underlying vulnerability characteristics that were used to generate the Risk scores. For every vulnerability, a risk level will be calculated on a scale of 5 to 1 with 5 being the highest likelihood or impact.

RISK SCALE - LIKELIHOOD

5 - Almost certain an incident will occur.
4 - High probability of an incident occurring.
3 - Potential of a security incident in the long term.
2 - Low probability of an incident occurring.
1 - Very unlikely issue will cause an incident.

RISK SCALE - IMPACT

5 - May cause devastating and unrecoverable impact or loss.
4 - May cause a significant level of impact or loss.
3 - May cause a partial impact or loss to many.
2 - May cause temporary impact or loss.
1 - May cause minimal or un-noticeable impact.

The risk level is then calculated using a sum of these two values, creating a value of 10 to 1 with 10 being the highest level of security risk.

- **10** - CRITICAL
- **9 - 8** - HIGH
- **7 - 6** - MEDIUM
- **5 - 4** - LOW
- **3 - 1** - VERY LOW AND INFORMATIONAL

5. SCOPE

FILES AND REPOSITORY

(a) Repository: defi-app-api-backend
(b) Assessed Commit ID: 9520e8e
(c) Items in scope:

Out-of-Scope:

FILES AND REPOSITORY

(a) Repository: defi-app-solana-paymaster
(b) Assessed Commit ID: 972c088
(c) Items in scope:

Out-of-Scope:

FILES AND REPOSITORY

(a) Repository: defi-app-sdk
(b) Assessed Commit ID: f084f0e
(c) Items in scope:

Out-of-Scope:

FILES AND REPOSITORY

(a) Repository: defi-app-react-app
(b) Assessed Commit ID: 3f9bc99
(c) Items in scope:

Out-of-Scope:

REMEDIATION COMMIT ID:

- ab8657f
- 418cb46

Out-of-Scope: New features/implementations after the remediation commit IDs.

6. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	9	2	0

IMPACT X LIKELIHOOD

	HAL-05 HAL-06 HAL-08			
		HAL-07 HAL-04 HAL-02 HAL-03		
		HAL-01	HAL-09	
	HAL-10 HAL-11			

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
LACK OF ENVIRONMENT ISOLATION	MEDIUM	FUTURE RELEASE - 03/06/2025
USE OF AWS IAM USERS	MEDIUM	SOLVED - 02/14/2025
SENSITIVE ACTIONS WITHOUT REAUTHENTICATION ALLOWED	MEDIUM	RISK ACCEPTED - 03/06/2025
DYNAMIC JWT TOKEN LONG-LIVED	MEDIUM	RISK ACCEPTED - 02/27/2025
JWT NOT INVALIDATED AFTER LOGOUT	MEDIUM	FUTURE RELEASE - 03/06/2025
CORS MISCONFIGURATION	MEDIUM	SOLVED - 02/02/2025
MISSING IMPORTANT SECURITY HEADERS	MEDIUM	SOLVED - 02/27/2025
LACK OF API RATE LIMITING	MEDIUM	SOLVED - 02/07/2025
UNNECESSARY FILES IN CONTAINER IMAGES	LOW	SOLVED - 02/27/2025

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
SENSITIVE INFORMATION IN BROWSER LOCAL STORAGE	LOW	FUTURE RELEASE - 03/06/2025
HARCODED SECRETS IN REPOSITORY	MEDIUM	SOLVED - 02/28/2025

7. FINDINGS & TECH DETAILS

7.1 LACK OF ENVIRONMENT ISOLATION

// MEDIUM

Description

Having different environments can be useful to ensure code changes are thoroughly tested and any issues are caught before deploying the code to live environments. These environments should be isolated to ensure that a security incident in a lower tier environment like development or staging does not affect production environments, which can contain sensitive data.

Proof of Concept

It was found that both Defi App API Backend and Defi App Solana Paymaster use the same AWS account for all SLDC environments (`dev`, `staging` and `prod`). This could allow security incidents in `dev` impacting production workloads and data. This can be seen in the Github Actions workflows used to deploy the APIs.

```
- name: Set environment variables
  id: set-env
  run: |
      if [[ "${GITHUB_REF}" == "refs/heads/staging" ]]; then
          echo "AWS_REGION=us-east-1" >> $GITHUB_ENV # set this to your preferred AWS region, e.g. us-west-1
          echo "ECR_REPOSITORY=defi-repository" >> $GITHUB_ENV # set this to your Amazon ECR repository name
          echo "ECS_SERVICE=staging-DEFI-ServiceStackstagingDEFIbackendService00621F35-dbhLfmjZXpPi" >> $GITHUB_ENV # set this to your Amazon ECS servic
          echo "ECS_CLUSTER=staging-DEFI-ClusterStackstagingECSCluster3C7C7EB2-EqQHVQnIQZ6h" >> $GITHUB_ENV # set this to your Amazon ECS cluster name
          echo "ECS_TASK_DEFINITION_ARN=arn:aws:ecs:us-east-1:058264498633:task-definition/stagingDEFITaskDefinitionStackstagingMyTaskDefinitionD501886C
          echo "CONTAINER_NAME=ApiContainer" >> $GITHUB_ENV # set this to the name of the container in the
          echo "MIGRATIONLAMBDA=runMigrationsstaging" >> $GITHUB_ENV # set this to the name of the container in the
          echo "CLEANDBLAMBDA=cleanDBstaging" >> $GITHUB_ENV # set this to the name of the container in the
          echo "UPDATEASSETSLAMBDA=updateAssetsstaging" >> $GITHUB_ENV # set this to the name of the container in the
      elif [[ "${GITHUB_REF}" == "refs/heads/master" ]]; then
          echo "AWS_REGION=us-east-1" >> $GITHUB_ENV # set this to your preferred AWS region, e.g. us-west-1
          echo "ECR_REPOSITORY=defi-repository" >> $GITHUB_ENV # set this to your Amazon ECR repository name
          echo "ECS_SERVICE=prod-DEFI-ServiceStackprodDEFIbackendServiceE86DC2BF-Awp88BXglXRM" >> $GITHUB_ENV # set this to your Amazon ECS service name
          echo "ECS_CLUSTER=prod-DEFI-ClusterStackprodECSCluster4B420692-KSJNhevfREnD" >> $GITHUB_ENV # set this to your Amazon ECS cluster name
          echo "ECS_TASK_DEFINITION_ARN=arn:aws:ecs:us-east-1:058264498633:task-definition/prodDEFITaskDefinitionStackprodMyTaskDefinitionD49931FE" >> $
          echo "CONTAINER_NAME=ApiContainer" >> $GITHUB_ENV # set this to the name of the container in the
          echo "MIGRATIONLAMBDA=runMigrationsprod" >> $GITHUB_ENV # set this to the name of the container in the
          echo "CLEANDBLAMBDA=cleanDBprod" >> $GITHUB_ENV # set this to the name of the container in the
          echo "UPDATEASSETSLAMBDA=updateAssetsprod" >> $GITHUB_ENV # set this to the name of the container in the
      else
          echo "AWS_REGION=us-east-1" >> $GITHUB_ENV # set this to your preferred AWS region, e.g. us-west-1
          echo "ECR_REPOSITORY=defi-repository" >> $GITHUB_ENV # set this to your Amazon ECR repository name
          echo "ECS_SERVICE=dev-DEFI-ServiceStackdevDEFIbackendServiceC3ED8BF1-oZsOAWwpVLea" >> $GITHUB_ENV # set this to your Amazon ECS service name
          echo "ECS_CLUSTER=dev-DEFI-ClusterStackdevECSCluster757AAB36-Z5mysyBmmZ4q" >> $GITHUB_ENV # set this to your Amazon ECS cluster name
          echo "ECS_TASK_DEFINITION_ARN=arn:aws:ecs:us-east-1:058264498633:task-definition/devDEFITaskDefinitionStackdevMyTaskDefinition63DDCA0A" >> $GI
          echo "CONTAINER_NAME=ApiContainer" >> $GITHUB_ENV # set this to the name of the container in the
          echo "MIGRATIONLAMBDA=runMigrationsdev" >> $GITHUB_ENV # set this to the name of the container in the
          echo "CLEANDBLAMBDA=cleanDBdev" >> $GITHUB_ENV # set this to the name of the container in the
          echo "UPDATEASSETSLAMBDA=updateAssetsdev" >> $GITHUB_ENV # set this to the name of the container in the
```

Score

Impact: 5

Likelihood: 2

Recommendation

It is recommended to use a distinct AWS account for each SDLC environment. AWS accounts are considered security boundaries, this can easily be used to create proper isolation between the different environments.

Remediation

FUTURE RELEASE: The **Defi App team** will be fixing this in the future, as currently their product is in Beta and their environments don't require that much isolation. Once their product is more mature, they will implement this isolation.

7.2 USE OF AWS IAM USERS

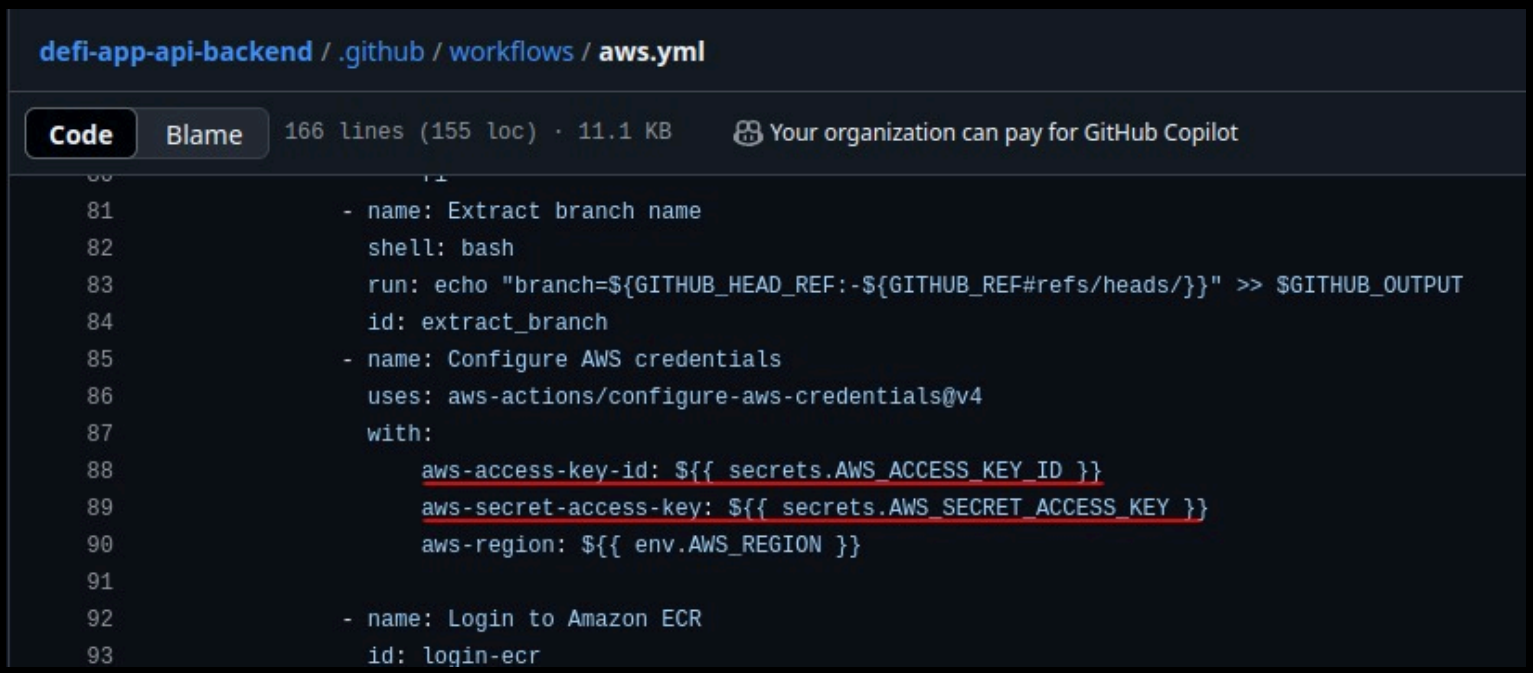
// MEDIUM

Description

AWS IAM service is used to control access to AWS environments. AWS best practices discourage the use of IAM Users, as they create long-lived credentials. Access to an IAM User is provided using a `ACCESS_KEY_ID` and a `SECRET_ACCESS_KEY`, these credentials do not expire. Therefore, if they are compromised, they can be used at any time until they are manually disabled. A better option is to use IAM Roles, as they allow the use of temporary credentials to access the environment.

Proof of Concept

It was found that Github Actions is using an AWS IAM User to deploy the APIs to ECS.



Score

Impact: 5

Likelihood: 2

Recommendation

For accessing AWS from GitHub Actions, the recommended strategy is using AWS Roles with trust policies allowing assuming the role from GitHub Actions through OIDC.

https://docs.github.com/en/actions/security-for-github-actions/security-hardening-your-deployments/configuring-openid-connect-in-amazon-web-services

Remediation

SOLVED: This issue was solved by retiring the old workflow and deleting both the secrets in GitHub and the IAM user in AWS.

7.3 SENSITIVE ACTIONS WITHOUT REAUTHENTICATION ALLOWED

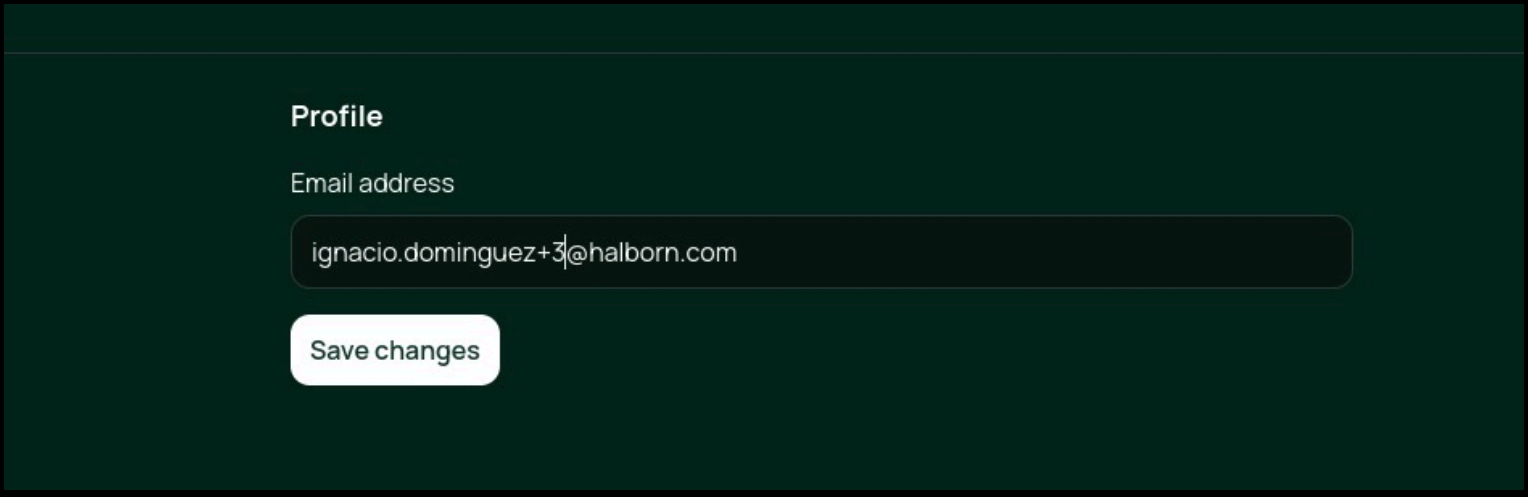
// MEDIUM

Description

Sensitive actions, such as changing email address, performing a swap or making a withdraw, can be performed without requiring re-authentication or asking for 2-factor authentication code (if configured), increasing the risk of unauthorized access and potential account compromise if a session is stolen or left unattended.

Proof of Concept

During the audit, it was discovered that sensitive actions, such as changing email address, performing a swap or making a withdraw without re-authentication or asking for 2 factor authentication code (if configured). Sensitive actions that move founds or change authentication details without an additional verification step, could lead to account compromise and unauthorized access to founds.



Score

Impact: 5

Likelihood: 2

Recommendation

To mitigate this risk, require re-authentication and 2-factor authentication for performing sensitive actions should be required, even during an active session.

Remediation

RISK ACCEPTED: Since the functionality to require re-authentication for sensitive actions is not supported out of the box by Dynamic, the **Defi App team** decided to accept the risk of this issue until Dynamic supports this feature.

7.4 DYNAMIC JWT TOKEN LONG-LIVED

// MEDIUM

Description

The Dynamic JWT token issued upon login has a week-long expiration period, which increases the risk of unauthorized access due to long-lived sessions, potentially allowing attackers to exploit stolen or compromised tokens.

Proof of Concept

When the user is authenticating, a Dynamic JWT token is issued. This token seems to have an excessively long expiration period of one week. The extended validity increases the likelihood that an attacker could exploit the token, potentially gaining unauthorized access to the user session, if this token is captured or leaked. The attacker could retain access for a week, significantly increasing the impact of an attack:


```
    {
      "email": "ignacio.dominguez@halborn.com",
      "id": "29339bcf-7ac7-4f6a-9894-411d866d7c0e",
      "public_identifier":
"ignacio.dominguez@halborn.com",
      "format": "email",
      "signInEnabled": true
    }
  ],
  "last_verified_credential_id": "29339bcf-7ac7-4f6a-
9894-411d866d7c0e",
  "first_visit": "2025-01-25T00:28:58.633Z",
  "last_visit": "2025-01-30T12:08:45.529Z",
  "new_user": false,
  "metadata": {},
  "verifiedCredentialsHashes": {
    "blockchain": "1b1e735cac10175896557b8902ca2ffd",
    "email": "a98ac7c7ac654028fc514d748217e3c6"
  },
  "iat": 1738254322,   Thu Jan 30 2025 17:25:22 GMT+0100 (Central European Standard Time)
  "exp": 1738859122   Thu Feb 06 2025 17:25:22 GMT+0100 (Central European Standard Time)
}
```


Score

Impact: 4

Likelihood: 3

Recommendation

In order to mitigate this vulnerability, reduce the JWT token expiration period to a shorter duration, such as a few hours or a day, and implement periodic re-authentication or forced token renewal.

Remediation

RISK ACCEPTED: The **Defi App team** decided to accept the risk of this issue to ensure a better user experience.

References

https://docs.dynamic.xyz/wallets/embedded-wallets/architecture-security

7.5 JWT NOT INVALIDATED AFTER LOGOUT

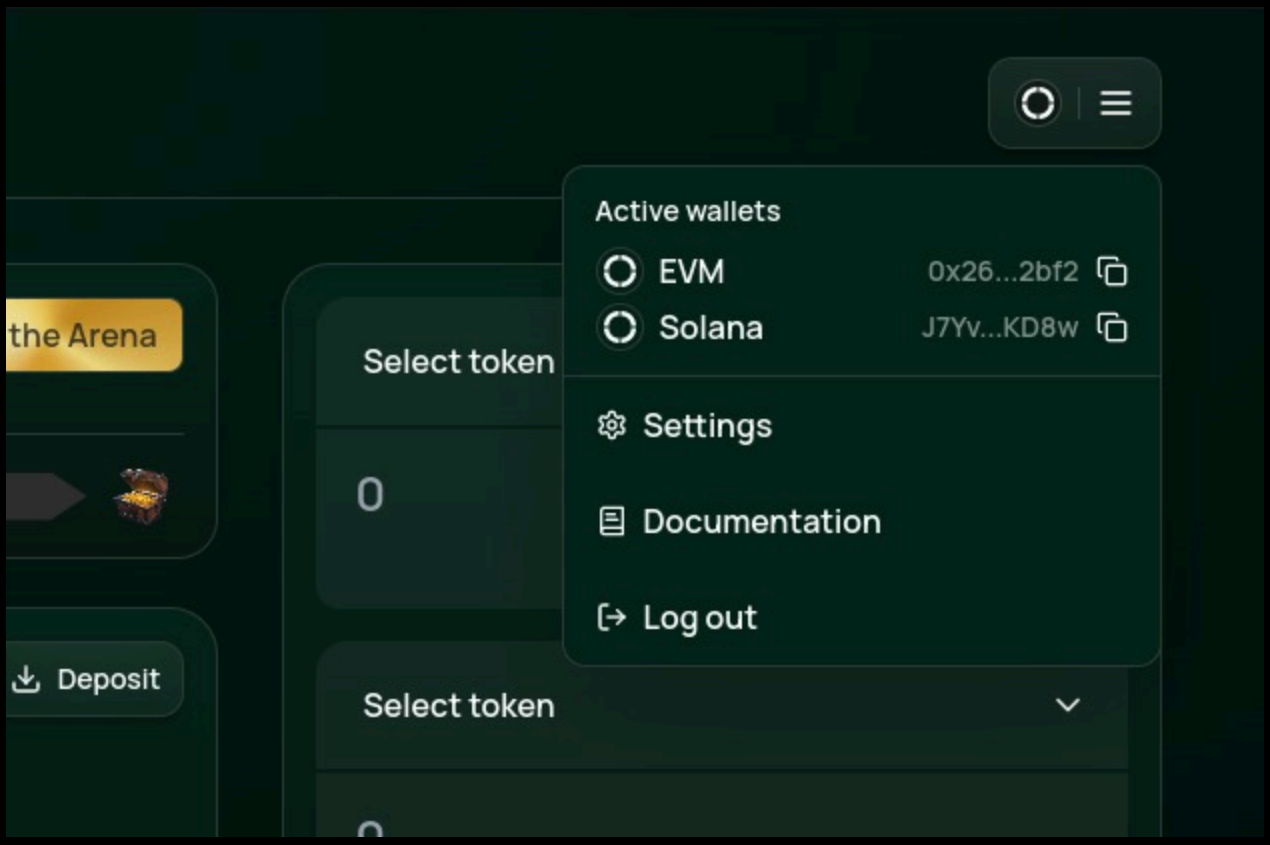
// MEDIUM

Description

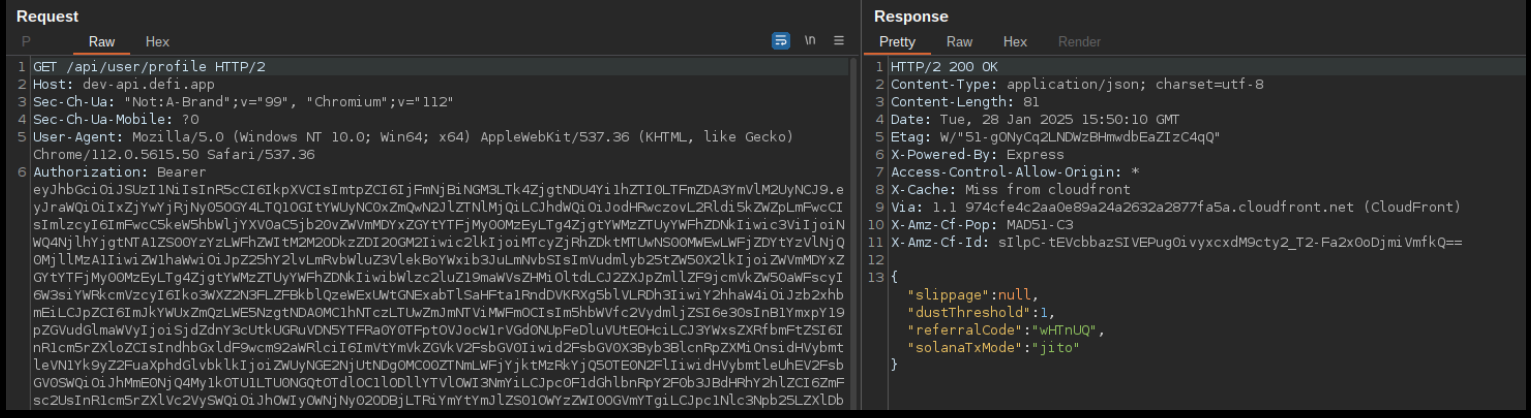
The user's `JWT` authentication token was not being invalidated after user logging out process.
This may pose a security risk for users, directly impacting on the confidentiality, integrity and availability of their data. Any malicious actor that would potentially access to the old `JWT` may be able to access and/or make modifications to the information within the Defi App API Backend application.

Proof of Concept

After the user logs out, a **POST** HTTP request to **https://app.dynamicauth.com/api/v0/sdk/eef061df-a1c3-4312-88f8-ac3e52aaad3d/revoke** is issued which deleted the cookie from the browser.



However, the JWT can still be used to retrieve private information about the user, indicating that the logout process did not revoke the session successfully from the Defi App API Backend:



Score

Impact: 4

Likelihood: 3

Recommendation

It is recommended to fully invalidate and revoke the user session after logout. In this case, as the backend uses JWTs values of revoked tokens can be stored in a cache until they expire. During token verification, this cache must be checked to ensure the token has not been revoked.

Remediation

FUTURE RELEASE: The **Defi App team** is currently implementing a remediation for this issue. The design of the solution involves using a cache to keep invalid tokens synchronized with Dynamic. This will mean user sessions will no longer be valid after logouts after a short period of time.

7.6 CORS MISCONFIGURATION

// MEDIUM

Description

The **CORS Wildcard Origin Allowed** vulnerability occurs when a web application improperly configures its Cross-Origin Resource Sharing (CORS) policy to allow any origin (*) in the `Access-Control-Allow-Origin` header.

In the affected application, this insecure configuration permitted any website to access the affected APIs, even those that should be restricted to certain trusted domains. This is particularly dangerous when the `Access-Control-Allow-Credentials: true` header is set, as it allows cookies, authentication tokens, and other sensitive credentials to be sent along with cross-origin requests.

The absence of proper origin validation leaves the application vulnerable to cross-origin attacks, such as **Cross-Site Request Forgery (CSRF)** or **Cross-Origin Resource Sharing (CORS) misconfigurations**, allowing malicious websites to exploit users' sessions and access protected data.

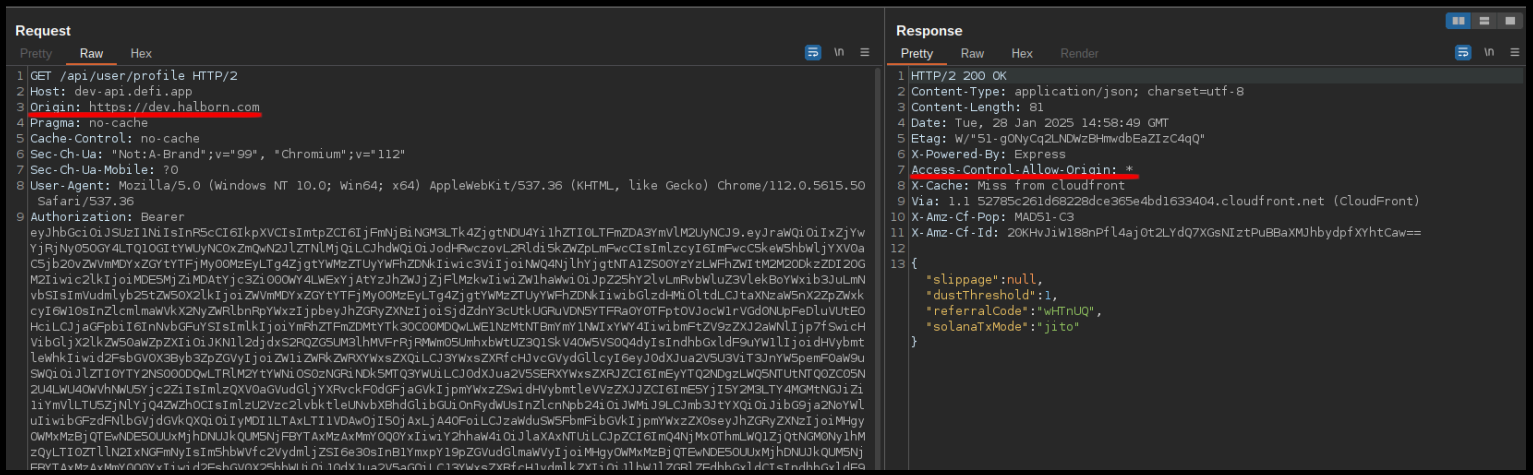
Impact

Allowing any origin (*) with credentials significantly increases the security risks:

- **Cross-Site Request Forgery (CSRF)**: Attackers can exploit the user's authenticated session to perform unauthorized actions or retrieve sensitive data from the APIs without the user's consent.
- **Unauthorized Access**: Malicious websites can make authenticated requests on behalf of users, potentially accessing sensitive data or initiating actions.
- **Session Hijacking**: The vulnerability can lead to session hijacking, where attackers can steal session tokens or other authentication credentials, compromising user accounts.
- **Data Leakage**: Sensitive information could be exposed to untrusted third-party websites, increasing the risk of data breaches.

Proof of Concept

The Defi App API Backend was found to have CORS misconfigured.



The Defi App Solana Paymaster was found to have CORS misconfigured.

Score

Impact: 4

Likelihood: 3

Recommendation

- **Strict Origin Validation**: Update the CORS policy to allow only trusted and specific origins instead of using the wildcard (`*`). This should be based on a whitelist of allowed domains.
- **Disable** `Access-Control-Allow-Credentials: true`: Remove the `Access-Control-Allow-Credentials: true` header unless absolutely necessary. If credentials are required, ensure that only trusted origins are allowed to send them.
- **Implement CSRF Protection**: Even with proper CORS configuration, implement CSRF protection mechanisms (e.g., using CSRF tokens) to ensure that cross-origin requests cannot be used to perform unauthorized actions.
- **Audit CORS Configurations**: Regularly audit CORS configurations to ensure that no unintended domains or wildcards are allowed, especially in production environments.
- **Penetration Testing**: Conduct penetration testing focusing on cross-origin attack vectors to validate that the CORS policies are correctly implemented and secure.

Remediation

SOLVED: Backend code was modified to specify the domains that can access the backend cross site. This fully remediates the issue.

Remediation Hash

https://github.com/defi-app/defi-app-api-backend/commit/ab8657ff9dcafa078df6b2717947f6a183f91dfb

7.7 MISSING IMPORTANT SECURITY HEADERS

// MEDIUM

Description

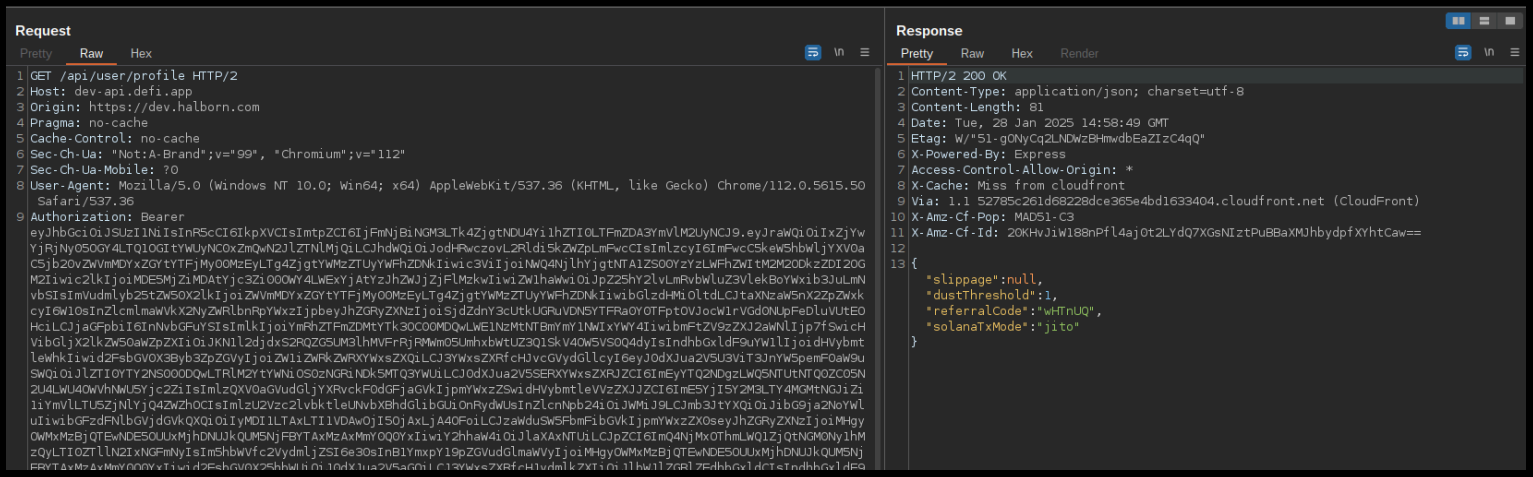
The application lacks several important HTTP security headers, which are instrumental in enhancing the security posture of web applications by instructing browsers on how to behave when handling the site's content. The absence of these headers leaves the application vulnerable to various attack vectors, including man-in-the-middle (MITM) attacks, Cross-Site Scripting (XSS), clickjacking, and MIME type sniffing.

Missing Security Headers:

- **Strict-Transport-Security**: Lacking the HTTP Strict Transport Security (HSTS) header exposes the application to MITM attacks by allowing the use of insecure HTTP connections.
- **Content-Security-Policy (CSP)**: Without a CSP header, the application is more susceptible to XSS attacks as it fails to restrict the sources of content that the browser can load.
- **X-Frame-Options**: The absence of the X-Frame-Options header increases the risk of clickjacking attacks by allowing the site to be framed by potentially malicious third-party sites.
- **X-Content-Type-Options**: Not implementing the X-Content-Type-Options header with `nosniff` allows the browser to MIME-sniff the content type, which can lead to incorrect execution of non-script MIME types as scripts.

Proof of Concept

The Defi App API Backend, Defi App Solana Paymaster and the Defi App Frontend were all found to be missing important security headers



Score

Impact: 4

Likelihood: 3

Recommendation

It is recommended to implement the missing HTTP security headers with appropriate policies. implementing these security headers will significantly enhance the security of the application by preventing various web-based attacks and ensuring safer interactions with the user's browser.

Remediation

SOLVED: The findings have been addressed in the following commits:

- https://github.com/defi-app/defi-app-react-app/commit/6295866b5bb5d361475a125a292c4ce08576bbf3
- https://github.com/defi-app/defi-app-api-backend/commit/a5abbef3d719434dc89266df135187433cd27a06
- https://github.com/defi-app/defi-app-react-app/commit/033f6ca050e9e3a809ad08b51027703e885f2f27

These add the necessary HTTP Headers to follow best practices

References

https://owasp.org/www-project-secure-headers/

7.8 LACK OF API RATE LIMITING

// MEDIUM

Description

The Defi App backend APIs did not have rate-limiting protections configured, allowing any user to overwhelm the backend server loading capacities. Furthermore, some of the API endpoints make use of third-party APIs such as CoinGecko, Solscan, Quicknode and Debank. These third-party APIs have their own rate limits depending on the purchased plan. An attacker could make enough requests to Defi App and reach the third party's rate limit, making the service unavailable for other Defi App users.
No rate-limiting was in place to control the number of simultaneous requests from a specific IP address.

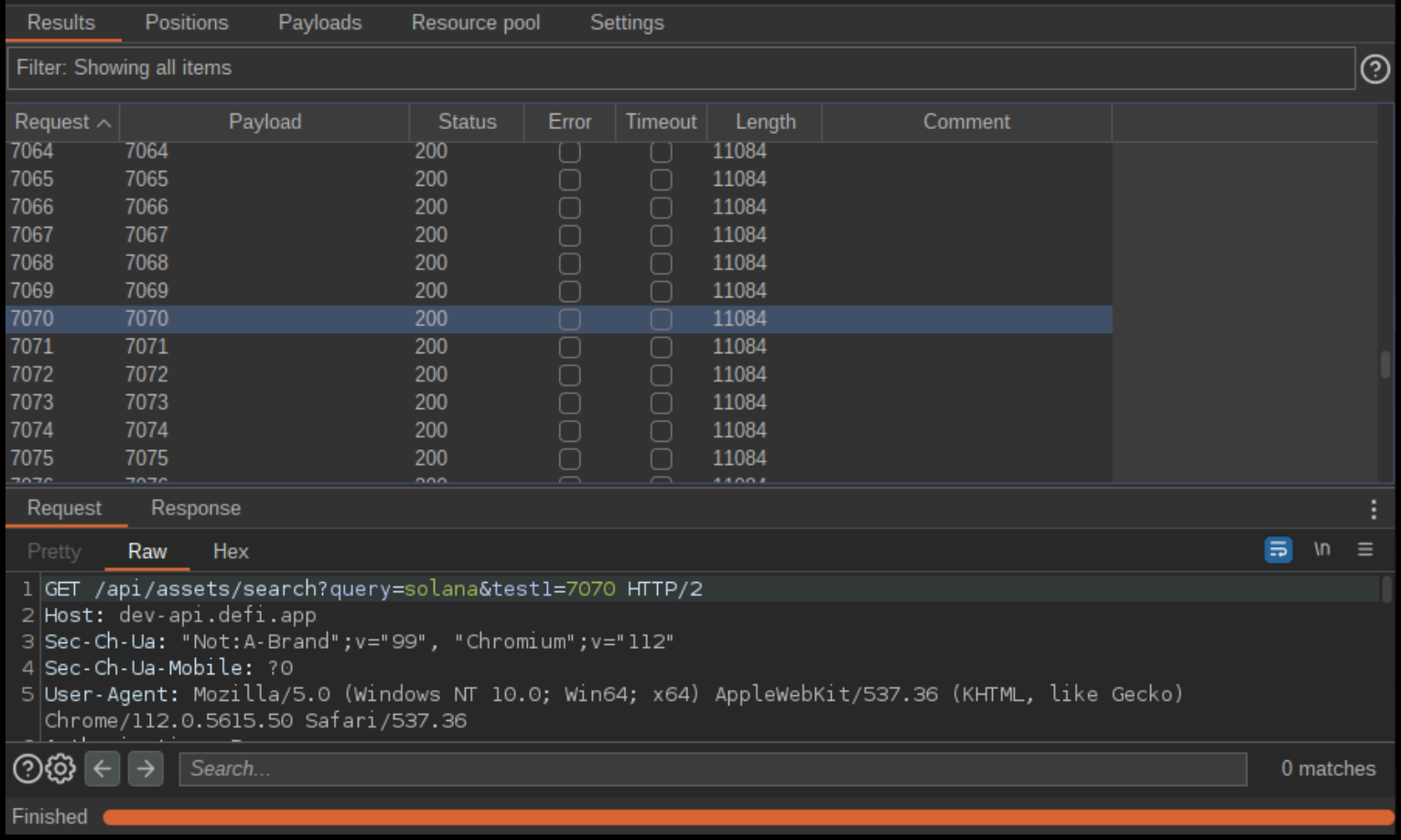
Proof of Concept

It was found that the following API endpoints make use of third party APIS:

- GET `/api/assets/gainers-losers`
- GET `/api/assets/search`
- GET `/api/assets/contracts/usd-coin/`
- PUT `/api/portfolio/cache`
- PUT `/api/portfolio/unindexed-assets-cache`

There is a mitigation in place for this attack as there is a cache to avoid requesting the same information from the third parties repeatedly in a short period of time. This mitigation can be bypassed as it uses the query string as a cache key and the query is controlled by the attacker. For example, `/api/assets/search?query=solana` and `/api/assets/search?query=solana&test1=1` make the same API request to the third party but have different cache keys. Additionally, `PUT` requests can be used to update the cache these endpoints do not check the cache and always perform the request to the third party.

It was possible to perform 10000 requests in under 3 minutes from the same IP address.



Score

Impact: 3

Likelihood: 4

Recommendation

Set up rate-limiting controls.

Remediation

SOLVED: The rate-limiting functionality was implemented in the backend, using `express-rate-limit` and leveraging Redis in order to have consistent results across replicas. The rate limiting is done based on IP address and the correct configuration has been set to avoid issues with the `X-Forwarded` HTTP header and using an ALB.

Remediation Hash

https://github.com/defi-app/defi-app-api-backend/commit/418cb46a0e3bd09f77bad99a33b866f0ee1be391

7.9 UNNECESSARY FILES IN CONTAINER IMAGES

// LOW

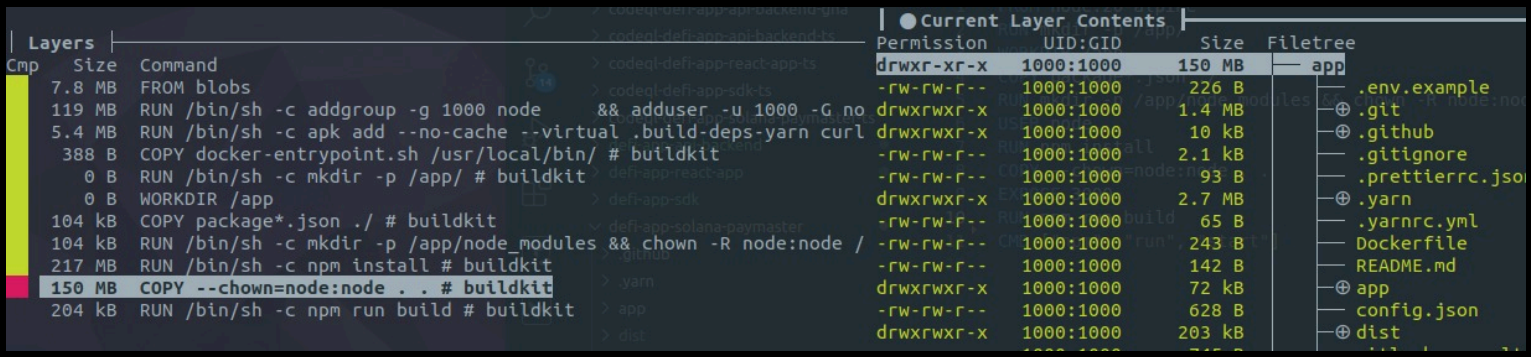
Description

It was found several container images included files that are not required for their production operation. These files could increase the attack surface or contain sensitive information. Furthermore, they increase the size of the image, which increases container startup times and storage costs.

Proof of Concept

The following container images have Typescript source files. These files are unnecessary as they get compiles into Javascript, once they have been compiled they are no longer needed in the image.

- `defi-app-api-backend/Dockerfile`
- `defi-app-solana-paymaster/Dockerfile` In this case the entire repository has been copied into the image.



Score

Impact: 2

Likelihood: 2

Recommendation

In order to resolve this issue, we advise making use of multi-stage docker files which allow compiling the Typescript files in a build stage and then copying the results into another stage, making it, so the resulting image only has the compiled files.
https://docs.docker.com/build/building/multi-stage/

Remediation

SOLVED: These issues were addressed in different repositories in the following commits:

- https://github.com/defi-app/defi-app-api-backend/commit/2745378ab1445cf629ea18e24017683f5b57e410
- https://github.com/defi-app/defi-app-solana-paymaster/commit/be04a6b2237b4b67a227c3de7efd32047796a76a

Container images are now built using multi-stage Dockerfiles in order to avoid including unnecessary files

7.10 SENSITIVE INFORMATION IN BROWSER LOCAL STORAGE

// LOW

Description

The Defi App react application stored sensitive information in the browser's local storage, namely the authentication JWT used to authenticate to both Defi API and Dynamic.
Session information stored either in the browser's local or session storage may be extracted via another attack vector, such as Cross-site Scripting (XSS). In case the vulnerability affects every user visiting a vulnerable page, this could lead to mass extraction of user's access tokens to an external server, and therefore to account takeovers.
Should another vulnerability affect the application, such as Cross-site Scripting (XSS), the browser's local/session storage would be retrieved and data exfiltrated to an external server. This could ultimately lead to account takeovers and serve as means of potential privilege escalation.

Proof of Concept

Key	Value
dynamic_authentication_token	"eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6IjFmNjBiNGM3
https://long-skilled-butterfly.solana-mainnet.quiknode.pro/8e3728ba3cf6b68f09241bb3ff49...	5eykt4UsFv8P8NJdTREpY1vzqKqZKvdpKuc147dw2N9d
dynamic_wallet_picker_search	""
-walletlink:https://www.walletlink.org:session:id	00cd626ea6a1bd0a806589bd9cd7e576
-CBWSDK:VERSION	4.0.4
acknowledgedTokens	[]
-walletlink:https://www.walletlink.org:session:secret	ee61fc9a48e0dc8fa15be4ffb69b7a682161378ce824f04d73f0d9
AMP_unsent_f462806091	[]
tradingview.chartproperties.mainSeriesProperties	{"style":1,"esdShowDividends":true,"esdShowSplits":true,"esdShow
dynamic_store	{"state":{"environmentId":"eef061df-a1c3-4312-88f8-ac3e52aaad3
tv.logger.logHighRate	false
tradingview.BarsMarksContainer.visibile	true
dynamic_primary_wallet_id	"d9ad5256-0f7c-49d5-a0f6-f6fae306ad6d"
dynamic_auth_expires_at	1739276594
dynamic_context_session_settings	"1738669481873"
-walletlink:https://www.walletlink.org:session:linked	0
dynamic_min_authentication_token	"eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6IjFmNjBiNGM3
lastLogin	2025-02-04T12:23:13.719Z
tradingview.chartproperties	{"timezone":"Etc/UTC","priceScaleSelectionStrategyName":"auto",
tv.logger.loglevel	2
tvlocalstorage.available	true
referralCode	EK/P9g
tradingview.current_theme.name	dark

Score

Impact: 2

Likelihood: 2

Recommendation

Avoid storing authentication tokens (JWTs) in localStorage or sessionStorage, as they are accessible via JavaScript and susceptible to XSS attacks.

Remediation

FUTURE RELEASE: The **Defi App team** will address this issue in a future release. It has not yet been decided what actions will be taken to mitigate this.

7.11 HARCODED SECRETS IN REPOSITORY

// MEDIUM

Description

Sharing code publicly or privately could lead to inadvertently leaving secrets in the code, making them accessible to others. Repositories are frequently cloned and forked into new projects, giving new developers access to their complete history. Any secrets within the repository's history would exist in all new repositories created from the affected one, and if stored in plain text, could lead to potential misuse of the secrets. In the case of a breach or unauthorized access to the source code, it could lead to further exploitation of infrastructure.
During the security assessment, it was identified that the repositories contained hardcoded **API Keys** for different third parties. Any unauthorized access to the source code could potentially lead to the misuse of these hardcoded secrets, compromising the security and integrity of the application and its associated infrastructure.

Proof of Concept

Identified sensitive data in the git history of repository https://github.com/defi-app/defi-app-api-backend:
1. An Ops Genie API Key was found in https://github.com/defi-app/defi-app-api-backend/commit/70c54b3fe01d14a82573198f887ba9d7d4084482#diff-a3046da0d15a27e89f2afe639b25748a7ad4d9290af3e7b1b6c1a5533c8f0a8cR12.
2. The same Ops Genie API Key was found in https://github.com/defi-app/defi-app-sdk/commit/81afe88acf9a5818f7c6db1e9f92c1e3309a7e3b#diff-a786bbea857be5ce8f109a5200a31a15a7823ec26e6aca06fd0fdcc525e6bc93R1. The API Key was tested and it was still active. It could be used to create alerts with malicious messages which could be used as a phishing vector to compromise developers that receive these alerts.

```
→ /tmp curl -X GET 'https://api.opsgenie.com/v2/alerts' --header 'Authorization:
 GenieKey 52c[redacted]09589'
{"data":[{"seen":false,"id":"0ae9c633-ed94-46fa-a401-ea107fee5e5c-1736237627163",
"tinyId":"760","alias":"0ae9c633-ed94-46fa-a401-ea107fee5e5c-1736237627163","mess
age":"Application Error","status":"open","acknowledged":false,"isSeen":false,"tag
s":[],"snoozed":false,"count":1,"lastOccurredAt":"2025-01-07T08:13:47.163Z","crea
tedAt":"2025-01-07T08:13:47.163Z","updatedAt":"2025-01-07T08:23:47.478Z","source"
:"160.242.96.122","owner":"","priority":"P1","teams":[{"id":"87b6d067-db87-4298-a
a4c-d92e117dcc30"}],"responders":[{"type":"team","id":"87b6d067-db87-4298-aa4c-d9
2e117dcc30"}],"integration":{"id":"090b7b6b-8ec5-438c-a1c1-5a7e76a09c4a","name":"
DeFi App API","type":"API"},"ownerTeamId":"87b6d067-db87-4298-aa4c-d92e117dcc30"}
```

3. A Telegram Bot API Key was found in https://github.com/defi-app/defi-app-sdk/commit/8ae74f635a493354e179cff2cdae115f4b3d05c0#diff-0ab3fa54b3c7f83ba6cee61508fbb80ed5fd7e526bed1b07c3b1c1701fcd3739R1. It was tested that the API Key is still active.

```
→ /tmp curl "https://api.telegram.org/bot7072553862:AAF[redacted]
[redacted]Kkj4/getMe"
{"ok":true,"result":{"id":7072553862,"is_bot":true,"first_name":"critical-alerts"
,"username":"defi_app_opsgenie_bot","can_join_groups":true,"can_read_all_group_me
ssages":false,"supports_inline_queries":false,"can_connect_to_business":false,"ha
s_main_web_app":false}}%
```

4. A DeBank API Key was found in https://github.com/defi-app/defi-app-sdk/commit/5e2565c8f96be79697f31540c898f8abf7f389d6#diff-fabd9a374b2f7028a40abf00c0405c60b85b63ce5e6e1c4836d426e53bb852e4R1. It was tested that the API Key is still active.

```
→ /tmp curl -X 'GET' \
  'https://pro-openapi.debank.com/v1/user/used_chain_list?id=0xcfeaead4947f0705a14ec42ac3d44129e1ef3ed5' \
  -H 'accept: application/json' -H 'AccessKey: 96748[redacted]0adcc1a7'
[{"born_at": 1717043595, "id": "op", "community_id": 10, "name": "OP", "native_token_id": "op", "logo_url": "http
s://static.debank.com/image/chain/logo_url/op/01ae734fe781c9c2ae6a4cc7e9244056.png", "wrapped_token_id": "0x42000
0000000000000000000000000000006", "is_support_pre_exec": true}, {"born_at": 1683030203, "id": "arb", "communi
```

Score

Impact: 3

Likelihood: 3

Recommendation

To address the identified risks, the following measures would be appropriate:

- Extract hardcoded secrets from the source code and the repository's commit history.
- Change and invalidate exposed secrets, replacing them with new, secure credentials.
- Employ environment variables, secret management solutions, or encryption for secure secret storage and management.

Remediation

SOLVED: The **Defi App team** has rotated or revoked the exposed secrets except the Telegram key, which is no longer under their control.

Halborn strongly recommends conducting a follow-up assessment of the project either within six months or immediately following any material changes to the codebase, whichever comes first. This approach is crucial for maintaining the project's integrity and addressing potential vulnerabilities introduced by code modifications.