ORBITDESK / WEB APPLICATION

Security assessment report

orbitdesk.example

This is an educational example of a fictional report, not a real scan result. All data, requests and identified issues are invented, no real client data is used.

Critical	High	Medium	Privacy
01	02	01	02

The privacy category partially coincides with risk levels — these are the same issues, not separate errors.

01 / WHAT YOUR TEAM NEEDS TO KNOW

The biggest risk is the possibility of accessing private conversations without authorization. It is also necessary to fix the mechanisms for granting credits and managing access to attachments. The cause of the first three issues is insufficient permission checking on the server or excessive trust in client-supplied data.

Product and technology

A fictional SaaS service for teamwork: chats, file sharing and paid credits for accessing additional features.

React TypeScript Supabase PostgreSQL

The fictional scenarios include the following roles: a guest, user A and user B. All chats, files and balances are fictional.

Findings by priority

Critical

CMA-01 · Supabase RPC · chat_previews

Last messages from private chats are accessible without login

> **Business impact**
>
> A guest can view previews of private conversations. The last message may include confidential information such as agreements or contact details.

Behavior in the fictional scenario

In the fictional scenario, the chat page requires login, but chat_previews provides the last message of a test chat to a guest without a user session.

Causes of the risk

The application function permits calls without a user session and does not verify if the user is part of the chat. Page protection does not cover the individual request.

Fictional observation

```
FICTIONAL REQUEST
POST https://api.orbitdesk.example/rest/v1/rpc/chat_previews
User session: absent

SIMULATED RESPONSE · HTTP 200
{ "chat": "fixture-chat-A",
  "last_message": "Private fixture message A" }

Expected: access denied.
```

Required changes

Prohibit anonymous function calls. Verify the session and user participation for each returned chat.

Re-evaluate EXECUTE permissions and data access rules. Implement explicit object ownership verification for functions with enhanced privileges.

Conduct tests for guests, chat participants, and other users using synthetic data.

How to verify the fix

A guest is denied access. User A sees only their test chat. User B does not see its preview.

High

CMA-02 · Billing API · credit_grant

Credits are granted based on a browser flag without payment.

> **Business impact**
>
> A user can obtain paid resources without purchasing them, leading to direct service costs and undermining the reliability of account balances.

Behavior in the fictional scenario

In the fictional model, the server accepts the paid: true parameter from a test client and increases the balance from 0 to 500, but there is no confirmed payment.

Causes of the risk

Credit grants depend on a client-supplied parameter. The server does not verify a confirmed payment, its amount or the order owner.

Fictional observation

```
FICTIONAL MODEL
User: fixture-user-A
Balance before: 0
Client parameter: paid = true
Confirmed payment: absent
Balance after: 500

Expected: balance stays at 0.
```

Required changes

- Grant credits only after the server verifies the payment status and the authenticity of the payment provider's notification.
- The server should determine the amount and recipient based on the order. Changing the client flag should not affect the balance.
- Ensure idempotency for credit fulfillment: repeated notifications for the same payment should not result in multiple credit grants.

How to verify the fix

The client-supplied paid: true flag does not affect the balance. A confirmed test payment grants credits only once, and repeating the event does not grant additional credits.

High

CMA-03 · Storage · private attachments

A private attachment is accessible to a user from another chat

> **Business impact**
>
> Knowing the link allows accessing the file outside the permitted conversation context. Message protection does not guarantee attachment protection.

Behavior in the fictional scenario

In the fictional scenario, fixture-attachment-A is linked to chat A. User B receives an HTTP 200 response for the file link, despite not being part of chat A.

Causes of the risk

File access is independent of user permissions. A complex file name does not substitute access control.

Fictional observation

```
FICTIONAL ACCESS MODEL
Object: fixture-attachment-A
Owner: test chat A
Request: user B
Member of chat A: no
Simulated response: HTTP 200

Expected: HTTP 403 or 404.
```

Required changes

- Verify chat membership before providing a file or temporary link.
- Implement private storage and set access rules for objects, not just messages.
- Avoid storing private attachments in shared caches accessible after user switching.

How to verify the fix

The test file owner gets access, while users from other chats and guests are denied.

Medium

CMA-04 · Browser · credential storage

The password is stored in browser storage

> **Business impact**
>
> Execution of third-party JavaScript code or access to the browser profile may expose a stored password. However, the presence of the entry does not necessarily mean that access has already been obtained.

Behavior in the fictional scenario

In the fictional scenario, the saved_password field remains in localStorage after authorization, but the interface restoration does not require the password.

Causes of the risk

The form saves entered credentials along with user preferences.

Fictional observation

```
FICTIONAL OBSERVATION
Store: localStorage
Key: saved_password
Value: [fixture value withheld]
Entry persists after login completes.
```

Required changes

Eliminate password storage in localStorage and sessionStorage, and remove existing entries.

Utilize the session mechanism provided by the authorization system. Keep interface settings separate from credentials.

How to verify the fix

After authorization and logout, the password should not be present in browser storage.

Remediation plan

First	CMA-01	Restrict anonymous access to chat previews and verify participant permissions
Before launch	CMA-02 · CMA-03	Link credits to confirmed payments and secure private attachments
Next	CMA-04	Delete stored passwords and verify session storage mechanism.

The fix is complete if permitted scenarios function, forbidden ones are blocked, and changes are tested with synthetic data.