SCENARIO

This 's email change feature contains a race condition that enables you to associate an arbitrary email address with your account. Someone with the address carlos@ginandjuice.shop has a pending invite to be an administrator for the site, but they have not yet created an account. Therefore, any user who successfully claims this address will automatically inherit admin privileges.

# PROCEDURE

1. Open the application and inspect it properly to look for possibilities of any potential collision and we see that there exists an user login functionality.
2. Log in with the provided credentials and study the request in BurpSuite’s Repeater and make 2 copies of it for **POST /my-account/change-email request.**
3. Change the email parameter of one request to [anything@exploit-<YOUR-EXPLOIT-SERVER-ID>.exploit-server.net](mailto:anything@exploit-%3cYOUR-EXPLOIT-SERVER-ID%3e.exploit-server.net) and the email parameter of the other request to [carlos@ginandjuice.shop](mailto:carlos@ginandjuice.shop).
4. Send the requests in parallel and if we received a confirmation email in which the address in the body is carlos@ginandjuice.shop, click the confirmation link to update your address.

## PAYLOAD

**REMEDIATION**

1. **Atomic Transactions:** Ensure that the process of changing an email address, from validation to update, is atomic. Databases like SQL support atomic transactions, ensuring that the entire operation, from start to finish, is consistent and isolated from other transactions.
2. **Rate Limiting:** Implement rate limiting on sensitive endpoints, such as email changes. By limiting how often a user can request an email change within a given timeframe, you reduce the chances of a race condition.
3. **Mutex (Mutual Exclusion):** Mutexes can be used to ensure that only one operation (in this case, email change) for a user can be executed at a time. By locking the critical operation, other concurrent operations will have to wait, ensuring sequential processing.
4. **Queueing:** Instead of processing requests simultaneously, queue them so they are processed sequentially. This ensures that operations don't overlap, preventing race conditions.
5. **Token Verification:** When a user attempts to change their email, generate a unique token and send it to the new email address. Require the user to click a link in the email or input the token on the website to verify the change. This ensures that the new email address is valid and controlled by the user.
6. **Check Before Committing:** Before committing the change, check if the email address is still valid and not claimed by another user.
7. **Client-Side Rate Limiting:** While server-side checks are essential, also adding client-side restrictions (like disabling the "Submit" button for a few seconds after it's pressed) can deter casual exploit attempts. However, remember that client-side controls can be bypassed, so this should only be a supplementary measure.