⚠2FA BYPASS⚠

Bypassing two-factor authentication

[ ] Flawed two-factor verification logic Sometimes flawed logic in two-factor authentication means that after a user has completed the initial login step, the website doesn't adequately verify that the same user is completing the second step For example, the user logs in with their normal credentials in the first step as follows:

```
POST /login-steps/first HTTP/1.1
Host: vulnerable-website.com
...
username=carlos&password=qwerty
```

They are then assigned a cookie that relates to their account, before being taken to the second step of the login process:

```
HTTP/1.1 200 OK
Set-Cookie: account=carlos

GET /login-steps/second HTTP/1.1
Cookie: account=carlos
```

When submitting the verification code, the request uses this cookie to determine which account the user is trying to access:

```
POST /login-steps/second HTTP/1.1
Host: vulnerable-website.com
Cookie: account=carlos
...
verification-code=123456`
```

In this case, an attacker could log in using their own credentials but then change the value of the `account` cookie to any arbitrary username when submitting the verification code.

```
POST /login-steps/second HTTP/1.1
Host: vulnerable-website.com
Cookie: account=victim-user
...
verification-code=123456
```

[ ] Clickjacking on 2FA Disable Feature

```
1. Try to Iframe the page where the application allows a user to disable 2FA
2. If Iframe is successful, try to perform a  social engineering attack to manipulate  victim to
```

[ ] Response Manipulation

```
1. Check Response of the 2FA Request.
2. If you Observe "Success":false
3. Change this to "Success":true and see if it bypass the 2FA
```

[ ] Status Code Manipulation

```
1. If the Response Status Code is 4XX like  401, 402, etc.
2. Change the Response Status Code to  "200 OK" and see if it bypass the 2FA
```

[ ] 2FA Code Reusability

```
1. Request a 2FA code and use it
2. Now, Re-use the 2FA code and if it is used successfully that's an issue.
3. Also, try requesting multiple 2FA codes and see if previously requested Codes expire or not wh
4. Also, try to re-use the previously used code after long time duration say 1 day or more. That
```

[ ] CSRF on 2FA Disable Feature

```
1. Request a 2FA code and use it
2. Now, Re-use the 2FA code and if it is used successfully that's an issue.
3. Also, try requesting multiple 2FA codes and see if previously requested Codes
expire or not when a new code is requested
4. Also, try to re-use the previously used code after long time duration say 1 day or
more. That will be an potential issue as 1 day is enough duration to crack and guess
a 6-digit 2FA code
```

[ ] Backup Code Abuse

```
Apply same techniques used on 2FA such as Response/Status Code Manipulation,
Brute-force, etc. to bypass Backup Codes  and disable/reset 2FA
```

[ ] Enabling 2FA Doesn't Expire Previous Session

```
1. Login to the application in two different browsers and enable 2FA from 1st session.
2. Use 2nd session and if it is not expired, it could be an issue if there is an insufficient
session expiration issue. In this scenario if an attacker hijacks an active session before
2FA, it is possible to carry out all functions  without a need for 2FA
```

[ ] 2FA Refer Check Bypass

```
1. Directly Navigate to the page which comes after 2FA or any other authenticated
page of the application.
2. If there is no success, change the refer header to the 2FA page URL. This may fool
application to pretend as if the request came after satisfying 2FA Condition
```

[ ] 2FA Code Leakage in Response

```
1. At 2FA Code Triggering Request, such  as Send OTP functionality, capture the  Request.
2. See the Response of this request and  analyze if the 2FA Code is leaked.
```

[ ] JS File Analysis

```
1. while triggering the 2FA Code Request,
2. Analyze all the JS Files that are referred in the Response
```

3. to see if any JS file contain information that can help bypass 2FA code.

[ ] Lack of Brute-Force Protection

```
This involves all sort of issues which comes under security misconfiguration such as
lack of rate limit, no brute-force protection, etc.
1. Request 2FA code and capture this request.
2. Repeat this request for 100-200 times  and if there is no limitation set, that's a rate  limit
3. At 2FA Code Verification page, try to  brute-force for valid 2FA and see if there is  any succ
4. You can also try to initiate, requesting  OTPs at one side and brute-forcing at
another side. Somewhere the OTP will  match in middle and may give you a quick  result
```

[ ] Password Reset/Email Change - 2FA Disable

```
1. Assuming that you are able to perform email change or password reset for the
victim user or make victim user do it by any  means possible.
2. 2FA is disabled after the email is  changed or password is reset. This could
be an issue for some organizations.  However, depends on case by case basis.
```

[ ] Missing 2FA Code Integrity Validation

```
1. Request a 2FA code from Attacker Account.
2. Use this valid 2FA code in the victim 2FA  Request and see if it bypass the 2FA Protection.
```

[ ] Direct Request

```
1. Directly Navigate to the page which comes after 2FA or any other authenticated
page of the application.
2. See if this bypasses the 2FA restrictions.
3. try to change the **Referrer header** as if you came from the 2FA page.
```

[ ] Reusing token

```
1. Maybe you can reuse a previously used token inside the account to authenticate.
```

[ ] Sharing unused tokens

```
1. Check if you can get the token from your account and try to use it to bypass the 2FA in a diff
```

[ ] Leaked Token

```
1. Is the token leaked on a response from the web application?
```

[ ] Session permission

```
1. Using the same session start the flow using your account and the victim's account.
2. When reaching the 2FA point on both accounts,
3. complete the 2FA with your account but do not access the next part.
```

4. Instead of that, try to access the next step with the victim's account flow.
5. If the back-end only set a boolean inside your sessions saying that you have successfully pass

[ ] Password reset function

1. In almost all web applications the **password reset function automatically logs the user into
2. Check if a **mail** is sent with a **link** to **reset the password** and if you can **reuse**

[ ] Lack of Rate limit

Is there any limit on the number of codes that you can try, so you can just brute force it? Be ca

[ ] Flow rate limit but no rate limit

In this case, there is a flow rate limit (you have to brute force it very slowly: 1 thread and so

[ ] Re-send code and reset the limit

There is a rate limit but when you "resend the code" the same code is sent and the rate limit is

[ ] Client side rate limit bypass

{% content-ref url="rate-limit-bypass.md" %} rate-limit-bypass.md {% endcontent-ref %}

[ ] Lack of rate limit in the user's account

Sometimes you can configure the 2FA for some actions inside your account (change mail, password..

[ ] Lack of rate limit re-sending the code via SMS

You won't be able to bypass the 2FA but you will be able to waste the company's money.

[ ] Infinite OTP regeneration

If you can **generate a new OTP infinite times**, the** OTP is simple enough** (4 numbers), and y

[ ] Guessable cookie

If the "remember me" functionality uses a new cookie with a guessable code, try to guess it.

[ ] IP address

```
If the "remember me" functionality is attached to your IP address, you can try to figure out the
```

[ ] Subdomains

```
If you can find some "testing" subdomains with the login functionality, they could be using old v
```

[ ] APIs

```
If you find that the 2FA is using an API located under a /v*/ directory (like "/v3/"), this proba
```

[ ] Previous sessions

```
When the 2FA is enabled, previous sessions created should be ended. This is because when a client
```

[ ] Improper access control to backup codes

```
Backup codes are generated immediately after 2FA is enabled and are available on a single request
```

[ ] Information Disclosure

```
If you notice some confidential information appear on the 2FA page that you didn't know previousl
```

[ ] Bypass 2FA with null or 000000

[ ] Previously created sessions continue being valid after MFA activation

```
1 access the same account on https://account.grammarly.com in two devices
2 on device 'A' go to https://account.grammarly.com/security > complete all steps to activate the
Now the 2FA is activated for this account
3 back to device 'B' reload the page The session still active
```

[ ] Enable 2FA without verifying the email I able to add 2FA to my account without verifying my email

```
Attack scenario :

Attacker sign up with victim email (Email verification will be sent to victim email).
Attacker able to login without verifying email.
Attacker add 2FA.
```

[ ] Password not checked when disabling 2FA

```
PoC
1- go to your account and activate the 2FA from /settings/auth
2- after active this option click on Disabled icon beside Two-factor authentication.
3- a new window will open asking for Authentication or backup code - Password to confirm the disa
```

```
4- in the first box enter a valid Authentication or backup code and in the password filed enter a
5- the option will be disabled successful without check the validation of the password.
```

[ ] “email” MFA mode allows bypassing MFA from victim’s device when the device trust is not expired

```
Steps To Reproduce:
Note:
1-Use burp suite or another tool to intercept the requests
2-Turn on and configure your MFA
3-Login with your email and password
4-The page of MFA is going to appear
5-Enter any random number
6-when you press the button "sign in securely" intercept the request POST auth.grammarly.com/v3/a
"mode":"sms" by "mode":"email"
"secureLogin":true by "secureLogin":false

7-send the modification and check, you are in your account! It was not necessary to enter the pho
```

[ ] 2FA bypass by sending blank code

```
1- Login to Glassdoor and navigate to https://www.glassdoor.com/member/account/securitySettings_i
2- Enable 2FA
3- Logout
4- Login again and notice OTP is asked
5- Now using Burp suite intercept the POST request by sending incorrect code. [Do not forward]
6- Before forwarding the request to server, remove the code and forward
7- Turnoff Intercept and notice that your login request has been fulfilled
```