Authentication Gone Bad - Penetration Testing Techniques for Chaining Vulnerabilities

By Foothold Security

www.footholdsecurity.com

Whoami

- More about me:
  - Co-Founder of Foothold Security
  - CEHv10, OSWE, ISTQB Security Tester
  - ISTQB Security trainer, University Lecturer
  - OWASP Member, BACEH Founding Member

Why this topic?

- During a penetration test the authentication flow of the customer was broken for less than an hour

- What was the impact?
- Complete account takeover
- Administrative account takeover
- For the next 2 hours the team was able to break into the servers due to a vulnerability in the administrative section

- Such vulnerabilities are far too common

- People just try to reinvent the wheel

Some basic terminology

- Vulnerability = Security bug, that could cause serious harm
- Exploit = Method or piece of code that takes advantage of vulnerabilities
- Payload = a piece of malicious code that is designed to execute a specific action on a target system. This code is typically delivered to the target system through a vulnerability or security flaw
- Vulnerability chain = Sequence of multiple exploits used to bypass a system's security measures. Usually a sophisticated attack.
- Authentication = The process of confirming the identity of a user

Vulnerability chain

- Step 1: User enumeration
- Step 2: Reset a user's password
- Step 3: A weird way to brute force user's password
- Step 4: Bypass Multi Factor Authentication by exploiting vulnerable API

Quick word on authentication

- What is authentication?
- The process of confirming the identity of a user
- Usually by Username/Email and Password

- What is Multi Factor Authentication?
- User needs to present two or more pieces of evidence to an authentication mechanism

What issues could break auth?

- User enumeration
- Password reset functionality
- Weak password policies
- Brute forcing passwords using different methods

How can this be solved?

- User enumeration – Return generic messages for wrong user/password combination and password reset
- Password reset functionality – Follow security guidelines when implementing
- Weak password policies - Follow security guidelines when implementing
- Brute forcing passwords using different methods – Rate limiting, enforce strong passwords

So…

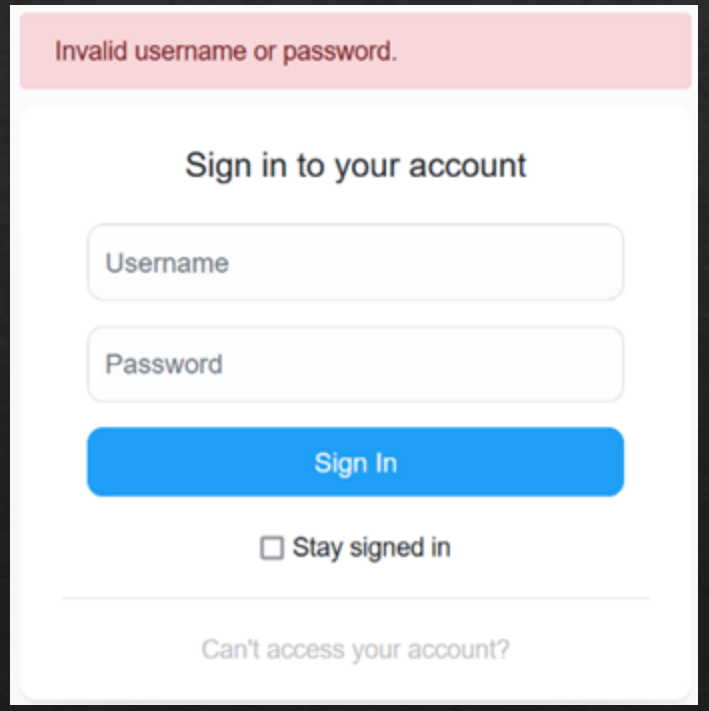
User enumeration

- 1. Just, browsing the app and interacting with other users could give us emails and usernames

- 2. Front end:
- Login with wrong password -> Generic message
- Login with non existent username -> Generic message
- Forgot password -> Use generic message

- 3. APIs:
- Same as front end. But still should be checked!

- There is a catch!



The catch: User enumeration

- Front end:
  - Login using email instead of username -> Generic message
  - Login using username instead of email -> Same generic message
  - Forgot password using username and email -> Use generic message
- APIs:
  - Same techniques as with front end tests
  - Adding another username/password to the request?

Enumeration: common mistakes

- Front end:
  - FE requests for username, but also accepts email -> email goes through different checks
- APIs:
  - API Returns different response code or response body for invalid user than the one for wrong password

The API response was key

Request

Pretty Raw Hex

```
1 POST /forgot/password HTTP/2
2 Host: [redacted]
3 Cookie: XSRF-TOKEN=daa4f2bb-5ce2-4177-93ec-02bf49c6165d
4 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:124.0)
  Gecko/20100101 Firefox/124.0
5 Accept: application/json
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate, br
8 Content-Type: application/json
9 X-Xsrf-Token: daa4f2bb-5ce2-4177-93ec-02bf49c6165d
0 Content-Length: 37
1 Origin: [redacted]
2 Referer: [redacted]
3 Sec-Fetch-Dest: empty
4 Sec-Fetch-Mode: cors
5 Sec-Fetch-Site: same-origin
6 Te: trailers
7
8 {
    "email":"mhekimov[redacted]"
  }
```

Response

Pretty Raw Hex Render

```
1 HTTP/2 400 Bad Request
2 Date: Mon, 25 Mar 2024 12:26:02 GMT
3 Content-Type: application/json
4 Vary:
  origin,access-control-request-method,access-control-request-headers,accept-e
  ncoding
5 Strict-Transport-Security: max-age=15724800; includeSubDomains
6
7 {
    "errors":[
      {
        "scope":"global",
        "message":"No user found with email <mhekimov[redacted]>"
      }
    ]
  }
```

Password reset functionality

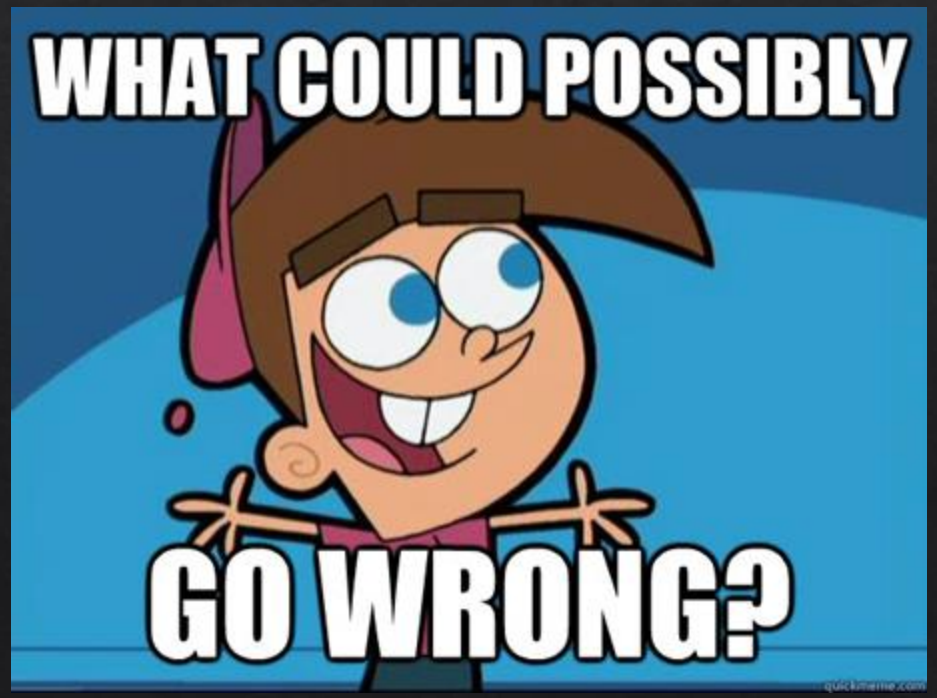
- Front end:
  - Use valid/invalid username -> Generic message
  - Use valid/invalid email -> Same generic message
- APIs:
  - Same techniques as with front end tests
  - Adding another username/password to the request?

Common password reset

- Common password reset scenarios
- A recovery email is sent containing URL token similar to: forgotpassword?ID=01234567890ABCDEF
- Using different factors to recover password e.g. app notification or SMS

Reinventing the wheel

- Change the user's password and send it via email
- The new password consists of 8 symbols (a-z, A-Z, 0-9) followed by a special symbol
- Also unblocks the user if lockout blocked the account



Hello **PenTestOrg1U01**, here is your new account password

pzGatbK4%

Please do not reply to this email.

Login brute force protection

- When logging in with incorrect password after 5 attempts the user gets blocked -> this is a danger zone as it could lead to Denial of service for many users if not implemented according to OWASP's guidelines
- Captcha really does the job most of the time
- Many other solutions
  - GeoBlock
  - Suspicious IP block
  - WAF, etc.

Reinventing the wheel

- Three issues!!!
  - Remember the password reset? -> Yeah, it was resetting the rate limiter
  - After failed 5 login attempts using the username, the user gets blocked for 5 minutes -> Such lockout mechanism was not implemented if you were using the user's email
  - API call was not protected by Captcha or WAF

So far so good

- Now we have a cool attack chain
  - We can enumerate the users
  - We can change their passwords to relatively easy to guess password
  - We can escape the lockout mechanism
- But the users are protected by MFA

Lets have a look at MFA

- The users need to setup MFA using their email
  - They would receive an email with a code when successfully log in
- How can we break this?
  - So far no industry standards were followed
  - What will happen if a legitimate user with active MFA changes password?
  - But this should be implemented by industry standard, right?

Wrong?

- When changing email (second factor), you should get a validation code!!!
  - However in many cases this is not happening -> email change was also changing the second factor so the MFA code was sent to the new mail

- How can we break this?

API Protection

- When logging in and before passing the MFA, the user usually gets Authorization header
- However in many cases the header could be used to make authenticated API calls WITHOUT passing the MFA code

It did terribly wrong

Request

Pretty Raw Hex

```
1  PATCH /user/info/ HTTP/2
2  Host: [redacted]
3  Cookie: XSRF-TOKEN=b37ac152-f112-4af8-bce5-2244506b10bb; JSESSIONID=
   9B0F175FD7DA04B4538FF8994F157419
4  User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:124.0)
   Gecko/20100101 Firefox/124.0
5  Accept: application/json
6  Accept-Language: en-US,en;q=0.5
7  Accept-Encoding: gzip, deflate, br
8  Content-Type: application/json
9  Content-Length: 70
10 Referer: [redacted]
11 Orgid: 2c9400238e5670bf018e74d067d90111
12 Origin: [redacted]
13 Sec-Fetch-Dest: empty
14 Sec-Fetch-Mode: cors
15 Sec-Fetch-Site: same-origin
16 Authorization: Bearer [redacted]
17 Te: trailers
18
19 {
     "name":"Milcho",
     "jobTitle":"Sec",
     "email":"mhekimov [redacted]
   }
```

Response

Pretty Raw Hex Render

```
1  HTTP/2 200 OK
2  Date: Thu, 28 Mar 2024 13:46:23 GMT
3  Content-Length: 0
4  Vary: Origin
5  Vary: Access-Control-Request-Method
6  Vary: Access-Control-Request-Headers
7  X-Content-Type-Options: nosniff
8  X-Xss-Protection: 1; mode=block
9  Cache-Control: no-cache, no-store, max-age=0, must-revalidate
10 Pragma: no-cache
11 Expires: 0
12 Strict-Transport-Security: max-age=15724800; includeSubDomains
13 X-Frame-Options: DENY
14 X-Xss-Protection: 1; mode=block
15 X-Content-Type-Options: nosniff
16 Cache-Control: no-store, no-cache, must-revalidate, proxy-revalidate,
   max-age=0
17 Expect-Ct: max-age=3600, enforce,
   report-uri= [redacted]
18 Expect-Staple: max-age=3600;
   report-uri= [redacted] includeSubDomains;
   preload
19 Report-To: [redacted]
20
21
22
```

Now we have MFA bypass

- So far so good, now we have the following chain
  - Step 1: User enumeration – using forgot password API
  - Step 2: Reset a user's password – setting a new password using Forgot password again
  - Step 3: A weird way to brute force user's password – the new password is weak and predictable, we can also bypass the lockout mechanism
  - Step 4: Bypass Multi Factor Authentication – changing the email without having MFA verified

Takeaways

- Standard testing techniques most of the times will not catch such vulnerabilities
- When it comes to security testing, never trust or assume, always verify
- Think of timing for setting tokens or roles in the app, is it too soon?
- Stay curious and never stop learning