Basic security

Risk factors - Exploitability

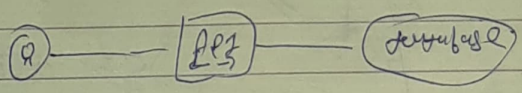
Attack Vectors
Exploitability
tools
user interaction
Repeatability of env
access of account

Security Weakness
weakness Prevalence
lack of awareness
complicated concepts
immature tools
lack of Time

Detectability
Tools
false positive

Impacts
Tech impact
CIA
confidentiality
Integrity
Availability.

Broocken authentication
- Insecure direct object reference (IDOR)
- Auth failure
- object
  - group of values.



api action
get update
create delete

weakness Prevalence
- Very Common
- Linked api (It is access for object)

Lack of auth (code doesent written)
Human eroroon (code eroroon)

Detectability
- Difficult tools (just tools won't help brain powers also matters)
- If you know what you looking for (Easy for human)

How you can find Vunrability if it happen and you know it
- find the Id in (URL, Body, Headers)
- If Id dosent appibe then it show be name?

Response Interpret
- authorized
- not authorized
- not found

Technical Impact
get / create / update / delete
get message
create message
update message
delete message

Automated Attack
- Burp suite (Brut force with random Id)
- You can give Id manually or automatically
/message/12345
/message/{ID} to 99999

=> Defense 1
Predictable IDs
Do not try to give Id in numbers
" " " " file name with dates we can find date easily.
Predictable Patterns.

Unpredictable IDs
GUIDs
- ea26AD1-CK1-2P222-7C2A
Hard to guess - not sequential

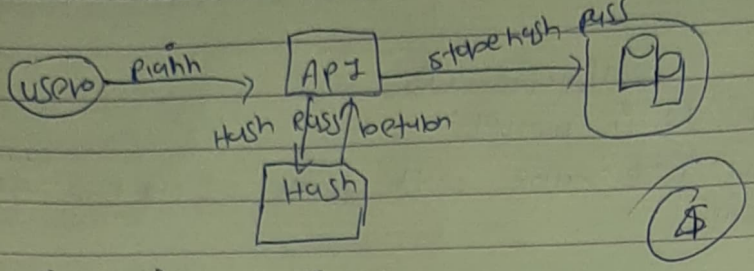
=> Defense 2
Check authorization using code.
users api with Id
Don't trust user
Confirm object access check
Automated testing
check the functionality work or not
notify when its broken

(4)

=> Insecure Password storage
- should be secret
  - Hidden on client screen
  - Encrypted in transit
  - securely stored
- Storing Pass Badly
  - Direct (plain text)
  - Staff can see it
  - External attackers
    - Attkr can see the data base

Defance Hashing
- Hash the pass (no one can know the pass)
- can't be reversed
- we dont know the secret (more then 1 Algo so we don't know which one use to incry

Not encryption
- encryption can be reversed
- we know the pass (this cant be happen)



⇒ Credential stuffing
- Attcro use the faulty API to retrive all store credential then copy and use for persional.
- then they can use the data based to Login for your acc.

⇒ Defence - Cogential stuffing.
- Additional information
  - Somthing you know like pass
  - Somthing you have like device
  - Somthing you are like fingerprint
- Defeat automation
  - Captcha

④

⇒ Common JWT Failures {Json} web token
⇒ JWT
- Result of successful login
- Temp credentials.
- Short expiry

```
{
  "typ" : "JWT"          ← Head
  "alg" : "HS256"        ← Hashing algo used
}
{
  "iss" : "globo",       ← Payload
  "exp" : 1577885820,    ← Expiration time
  "name" : "GavinJL".
  "admin" : false        ← Check for admin time
} 03f32983b86f7d9 - ← Signature
```

=> JWT Expiry
- Exp time (exp)
- Seconds since January 1st 1970
- is expiry checked?
- JWT should be short live.

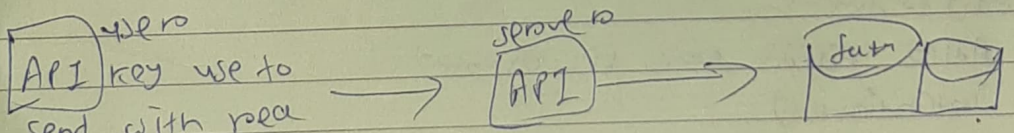
=> JWT signature
- created by API
- Header + payload created by server with hash
  - Server created so only server has hash code (secret)

Algorithm (ALG)
- "alg" : "none"
- Token is valid
- check algo

Defence - JWT
- must validate exp time so if token (temp pass is stolen after some time that agent won't work or do fork)
- Algo is valided algo none is use less
- JWT sig must re-generated every time when API get JWT if JWT signature is invalid the that contain is not trustable.
-

④

API keys



- but if we're sending the request every time the some one can steal it
- every req is encrypted so that is minimal risk

⇒ Atk

- key exposure
- weak storage (Hard coded in source)
- Hard coded in config. storage

Exposed in client application

Risk or exposed

- Analytics API is acceptable
- Payment 3rd party.

Private API

Not in source-code

Not in client application

(5)

Excessive Data Exposure

- Data is invisible to user
- The data can be seen
- Browser dev tools
- Slightly more effort on mobile

All you have to do is look!

⇒ Prevalence

- Very common
- Assumption no one will look for the developer
- Tooling enables rapid development so this would be easy to develop
- Added fields would be exposed.
- Done intentionally
- might be useful in feature
- save in development time.

⇒ Detectability

- automated tools Don't know which field is visible or not. Programmers easily understand.

Technical Impact

- Exposed data
- Account takeover
- Laws and regulation
- PII
  - Identity theft
  - Fraud