We will use postman for API penetration testing.
-> just play around with postman to get familiar with.

* Basic authentication

you can authenticate your self through header request
Authorization : Basic username:password
but you will get bad request 400, because you did not encode your credentials with base64, so you have to encode them to be authorized.
after that send your request and it will login seccessfully.

use apimatic.io to transfer API ?

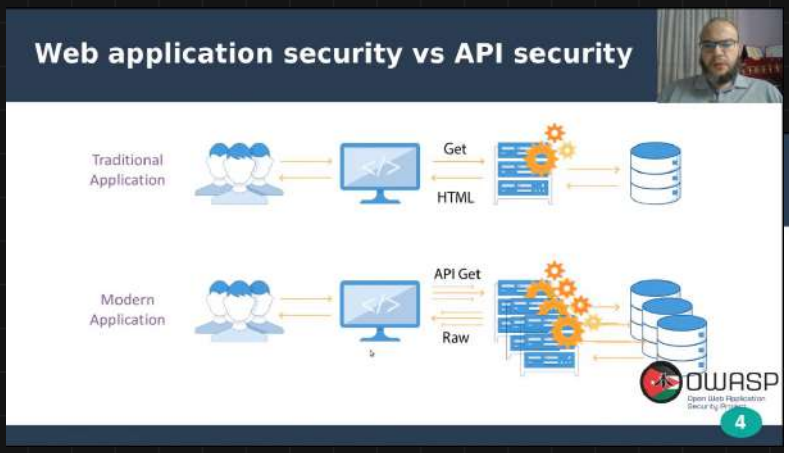
websites

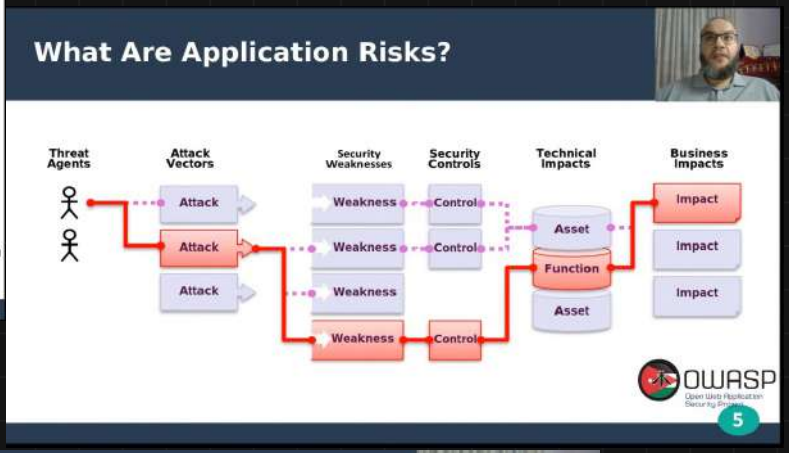
* jsonplaceholder.typicode.com/users
* postman-echo.com/basic-auth
* api.docs.imgur.com
* rest.vulnweb.com/jwt/api/users/_
* webscantest.com

How to search ?
google -> site:target.com api postman
site:.com /swagger/index.html
site:.com inurl:swagger.json
site:.com inurl:swagger-ui.json

Lab Setup

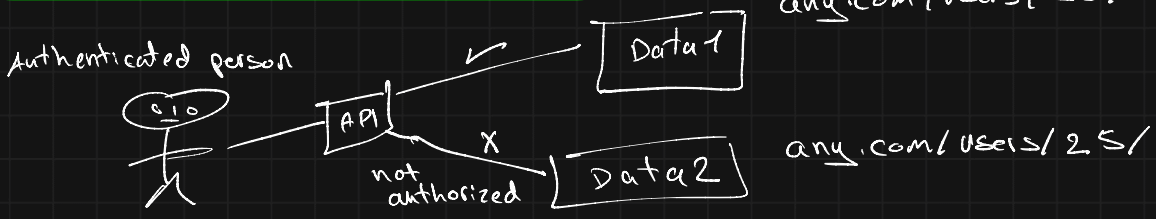
1. https://github.com/tkisason/vulnapi
2. https://github.com/snoopysecurity/dvws-node
3. https://github.com/erev0s/VAmPI/
4. https://github.com/jorritfolmer/vulnerable-api/
5. https://github.com/InsiderPhD/Generic-University





OWASP API Top 10

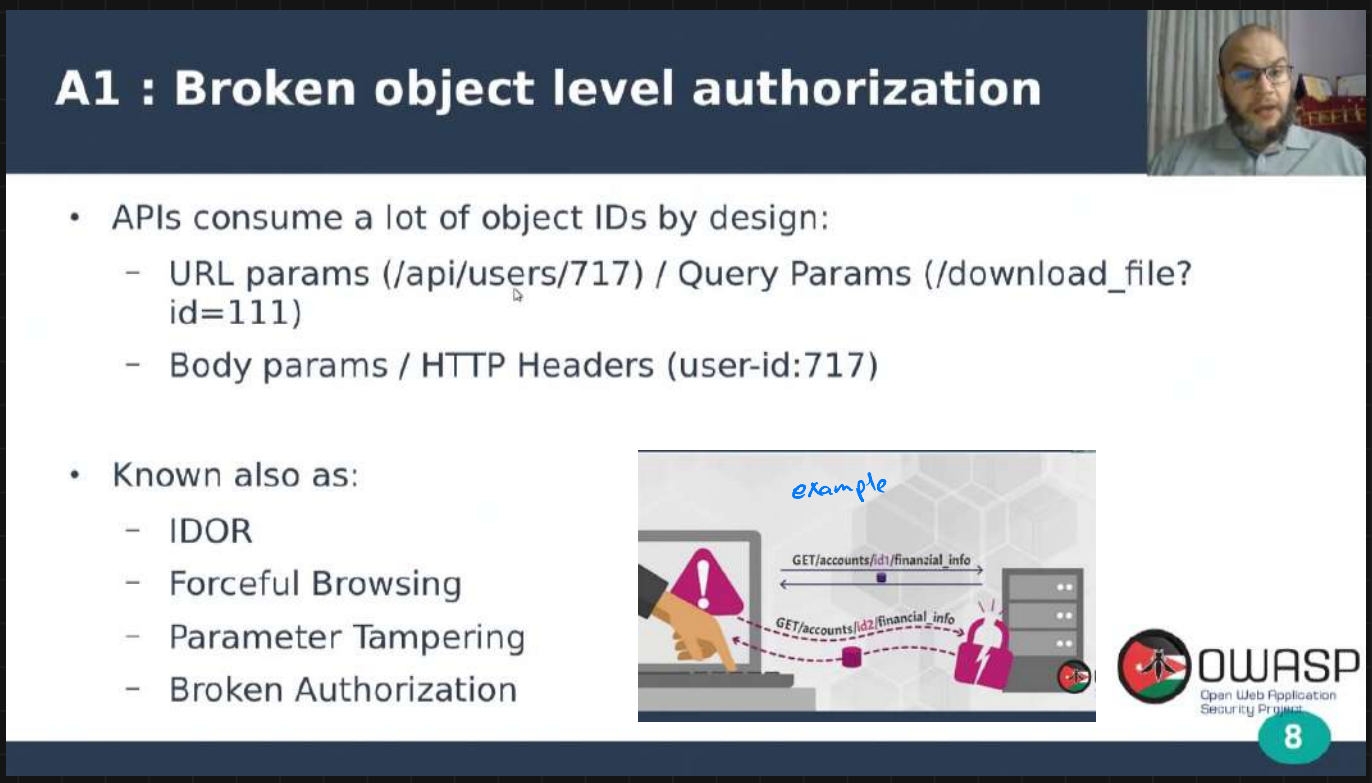
1- Broken Object level Authorization :

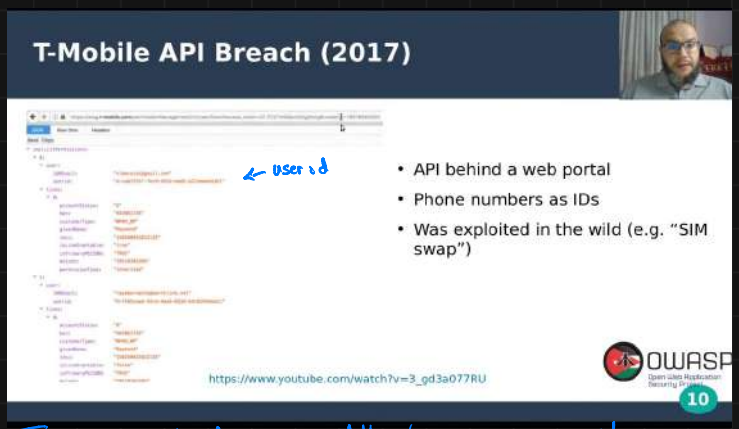


if user access something he is not allow to access, then it is vulnerability.

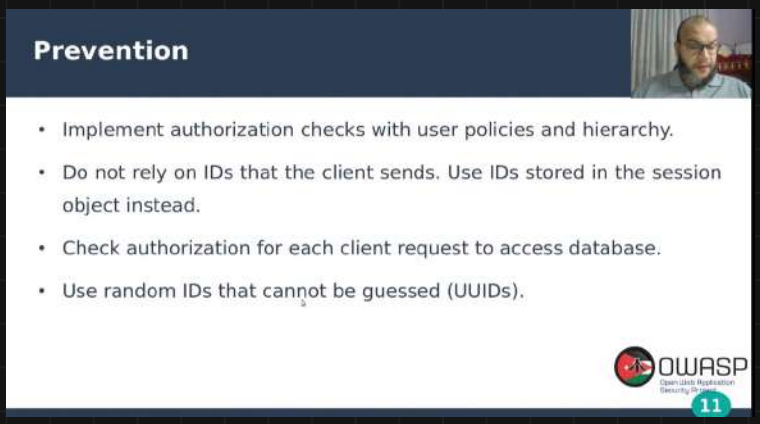
How to find ?

① create two accounts
② capture request with identifier
③ replace the authentication token account1 with account2
④ if you are able to make request, then there is broken object level authorization.





This is very dangers, Attacker can write a program (for loop to get all possible numbers from 0 ..... to 9 .....). All users has been exposed.

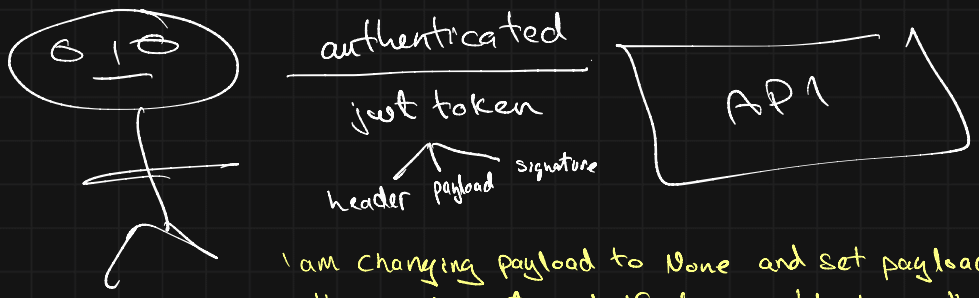


2- Broken Authentication :

→ Authentication mechanism for clients and end-user are Sometimes not properly specified and implemented.

→ Attacker can use that Falwed implementations to compromise authentication token and assume identity of another user.

→ if the system ability to identify the client is compromised the overall api security is compromised.



I am changing payload to None and set payload to another user (user id), then send jwt and if I am able to authorized, it is vulnerability.

How to find?

→ if there is no bruteforce protection, report it

→ if there is no Account local mechanism

→ weak password → /api/v/register and they accepting 1234, then there is weak password.

→ Sensitive data/token in url

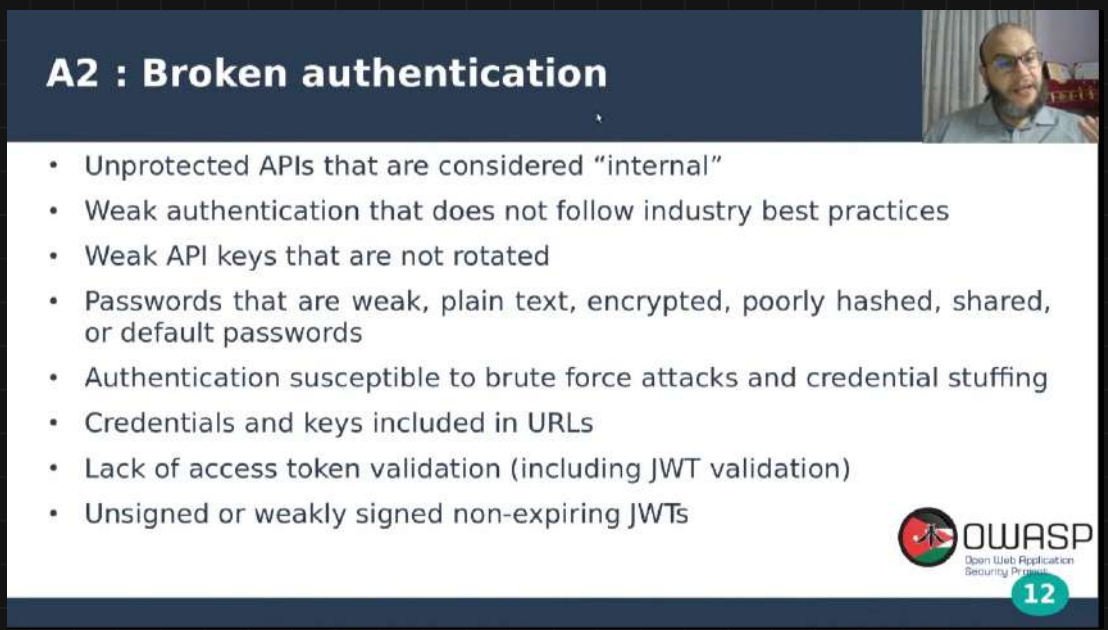
→ Does not validate token.

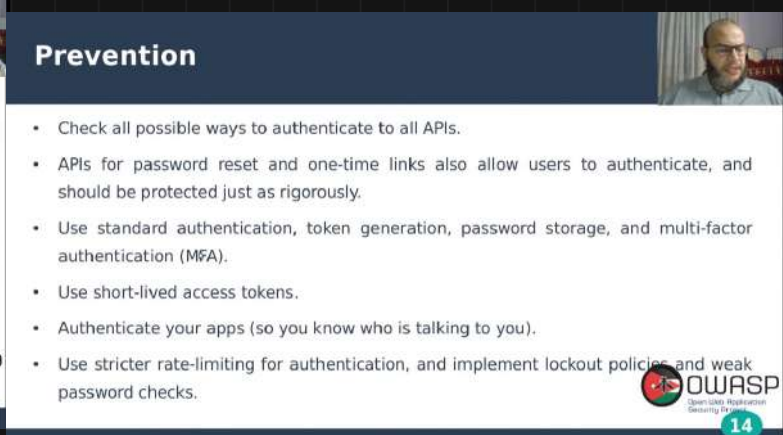
→ Accept (unsigned token or jwt non attack)

→ Use of plaintext, weak heshes possword

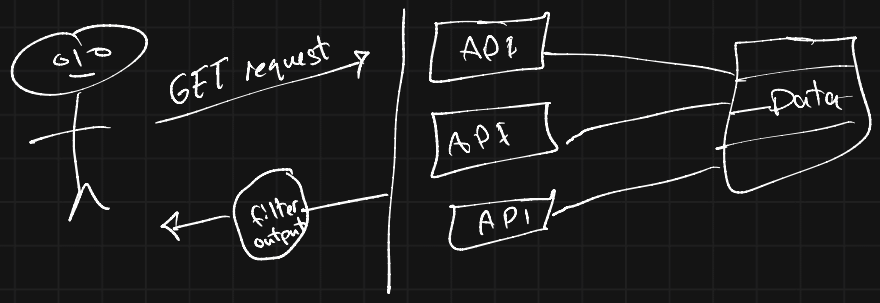
→ use of weak encryption keys / Api keys.

These checklist what I will look for: Impact: attacker can take over another accounts, sensitive data.





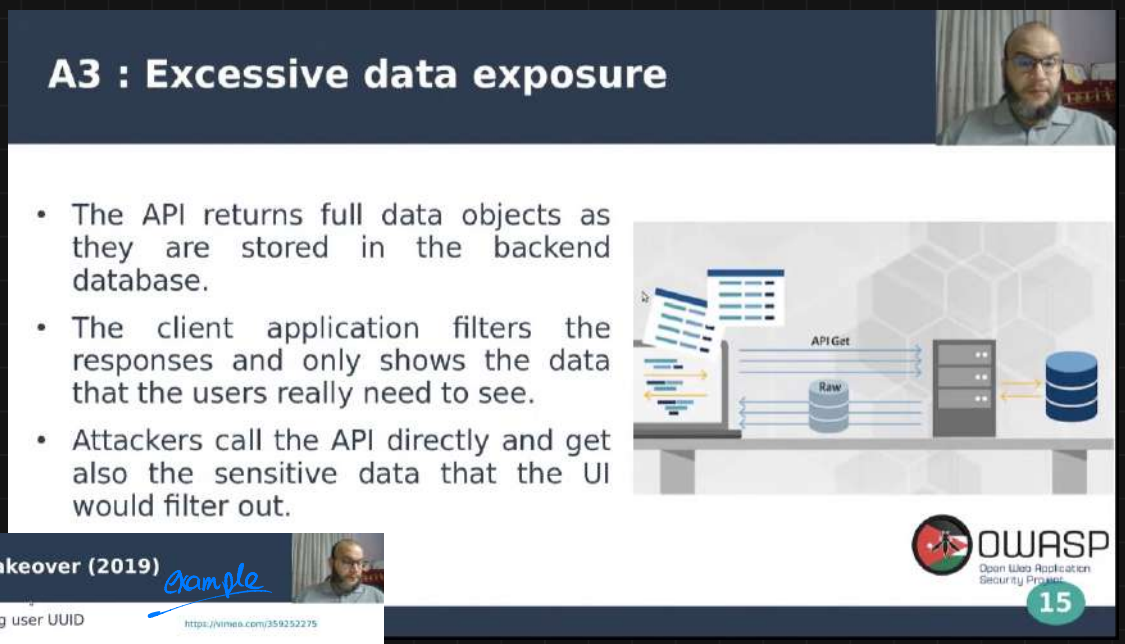
3 - Excessive Data Exposure:

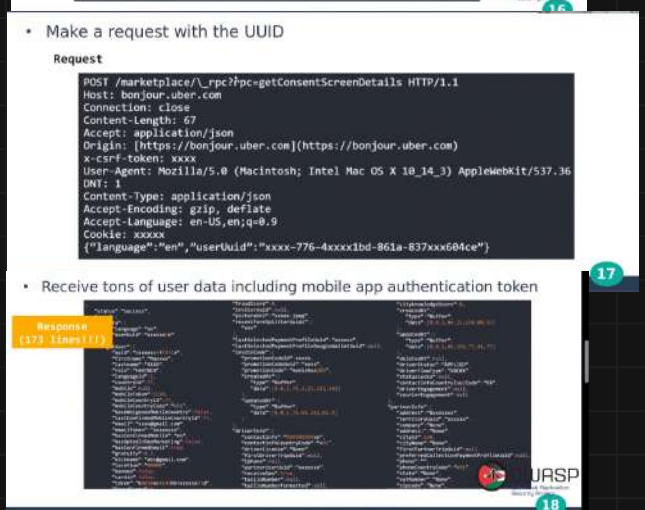


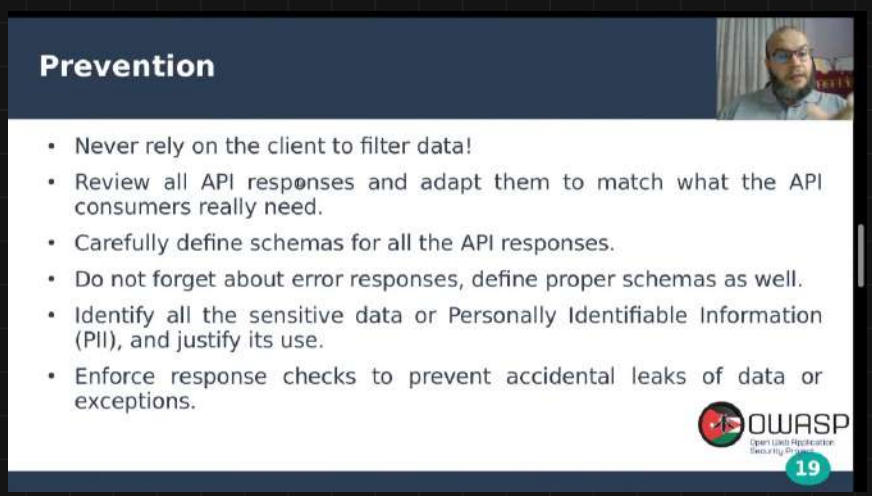
Basically api returns full data object as they are stored in background database.
The client application filter the data response and only show that data that need to see.
Attacker can call directly api and bypass the client application and he will get easily the full information. That called excissive data exposure.
The problem is that api is relying on client application to do data filtering.

How to hunt:

1. Does your api call return either sensitive Data or full data object stored in db
2. is returned content filtered on the client side before presented to the user?
   - if yes, there is a strong chance to be vulnerable.







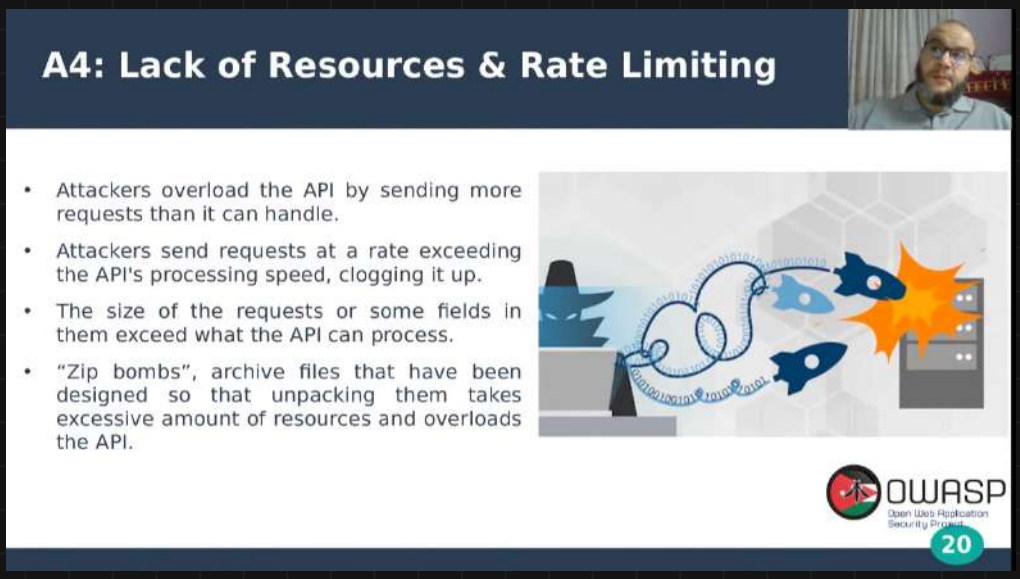
4- lack of resources and rate limiting:

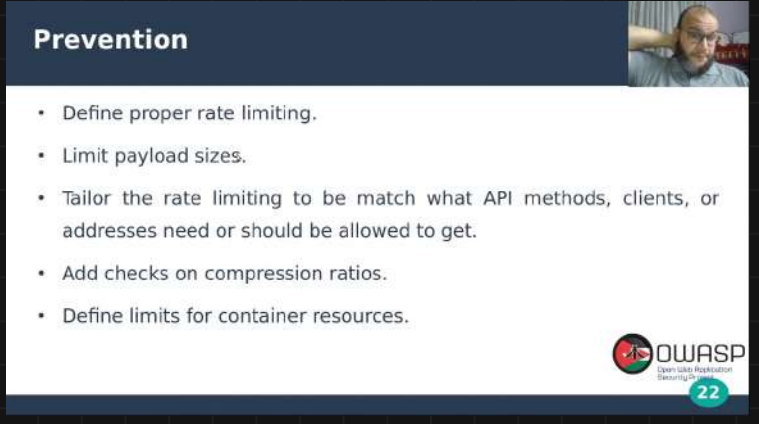
leak of resource senario:
what if attacker upload a large image by using a post request to api. /api/d/images, the available momery become exhausted / unresponsive.

Senario 2:
large number of resaurce request
we have an application that can send a list of users
/api/user?page=1&size=100
so attacker change size parameter to 20000. This will cause a performance issue on database, so api may get unresponse. --> DOS Attack / buffer overflow error.

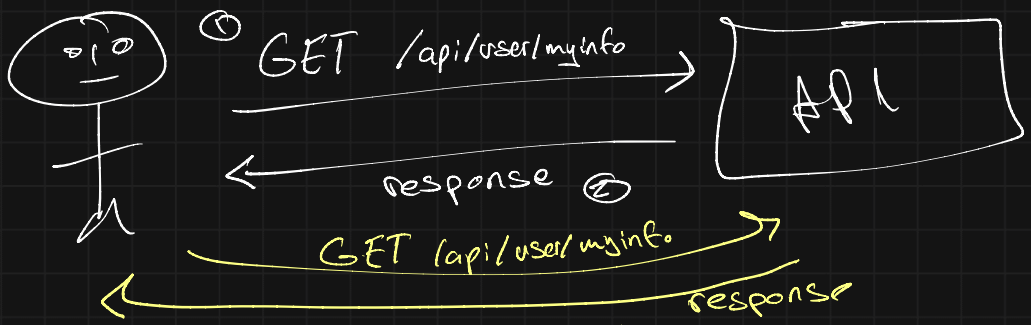
rate limite Senario (Bruteforce attack):
for authentication user + password and two factor authentication that require OTP. on OTP there is no limit request from the user, so you can send request multiple times like 100 time and more.

What is the use case ?
1. Attacker overload the api by sending more request than it can handle.
2. Attacker send request at a rate exceeding the API's processing speed, cloggit it up.
3. The size of request or some fields in them exeeced what the APIs can process
4. zip bombs archive that have been designed so that unpacking them takes excessives mount of resources and overloads the API.





5- Broken function level Authorization :



what if user request GET /api/admin/allinfo and get response then it is vulnerable . just play with the url .

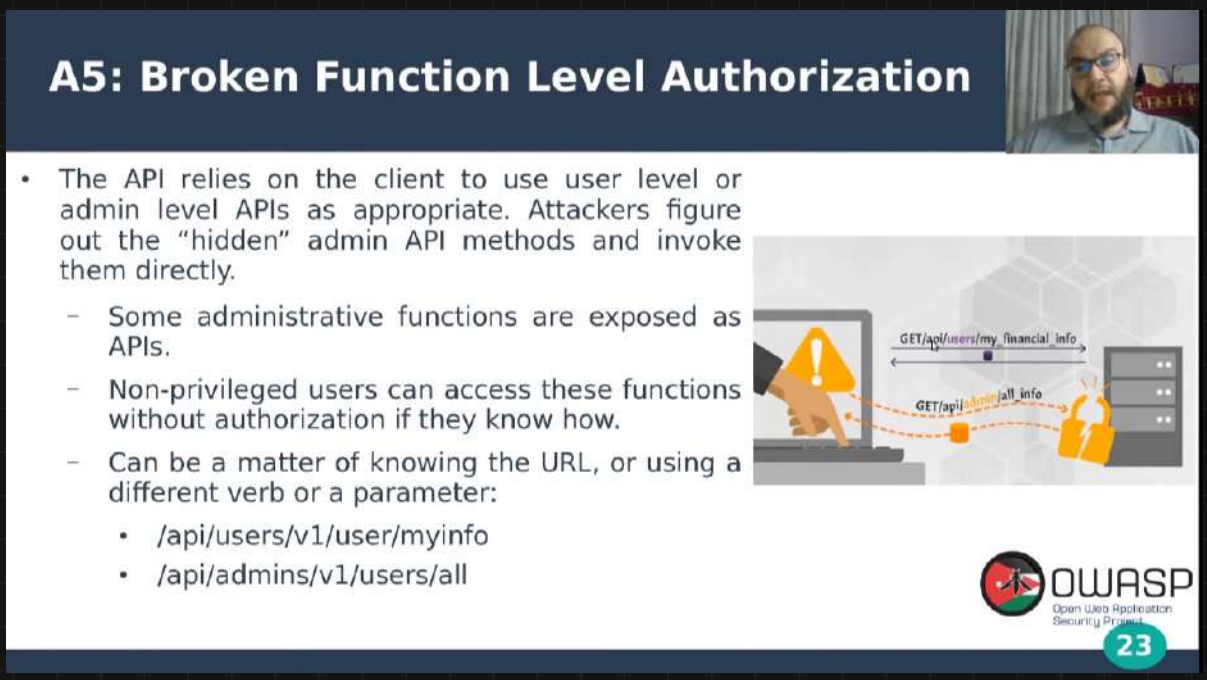
or

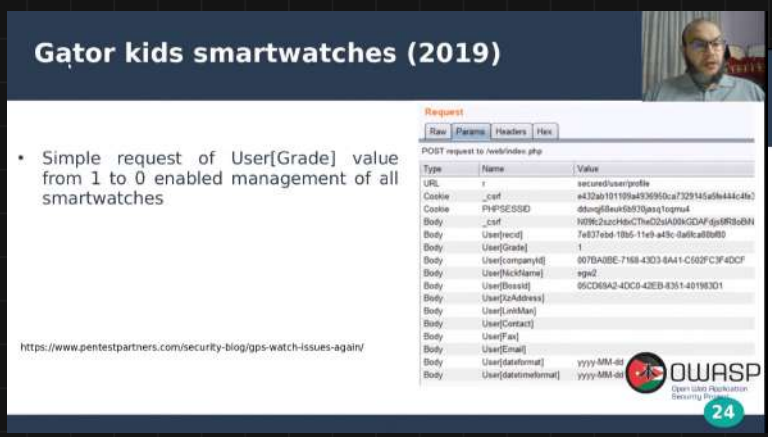
you are making post request /api/user/register and there is a parameter is_admin=False , so change it to is_admin=true and send request.

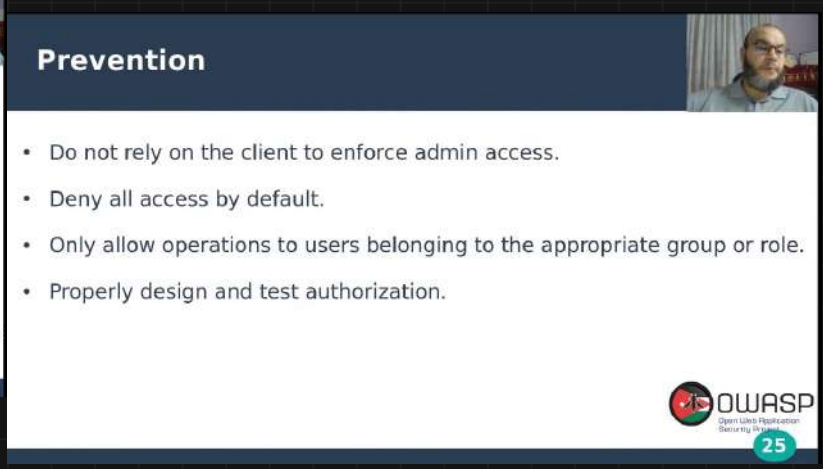
How to test ?

you need to understand deeply the authorization hierachy , so ask the following questions

1. can a regular user access administrative endpoints ?
2. can a user perform sensitive actions . (creation, modification ... by change http method for example from get to delete)
3. Can a user from group X access a function that should be exposed only to users from group Y by simply guessing the endpoint URL and parameter .







6- Mass Assignment

post /api/users

Self register form for new user. { username, password, email, comment }

{ in the background they have whitelist property and blacklist property.

We will assume the user structure:

user id
user name
:
{ user property

let's assume they have one more field

{ system property: Has_admin_Role → boolean

→ Now Capture request and add the has_admin_role=true
if request was seccessful then there is a vulnerability. (you will make requests as an admin)

→ How to know about this additional property information?
Through reading the documentation of an API or just guest it
Is try to add has_admin_role=true in request and see results.

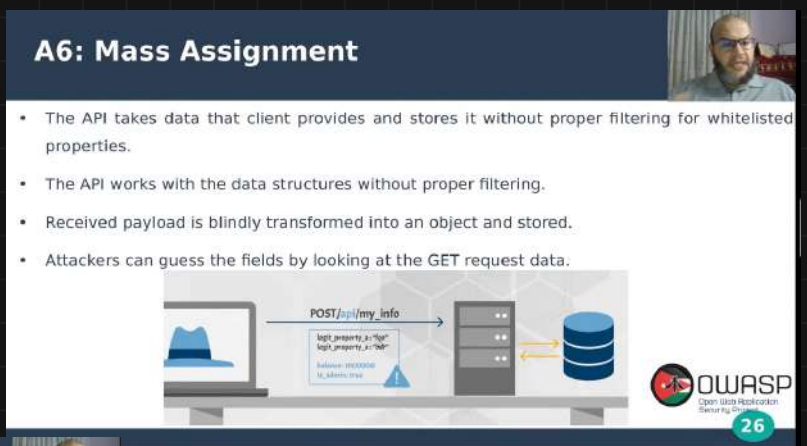
Senario: Application that allow user to update their informations.

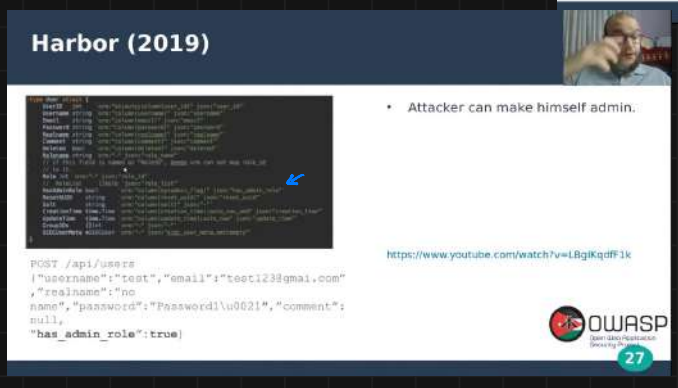
put /api/v1/users/me
{ username =
Age = }

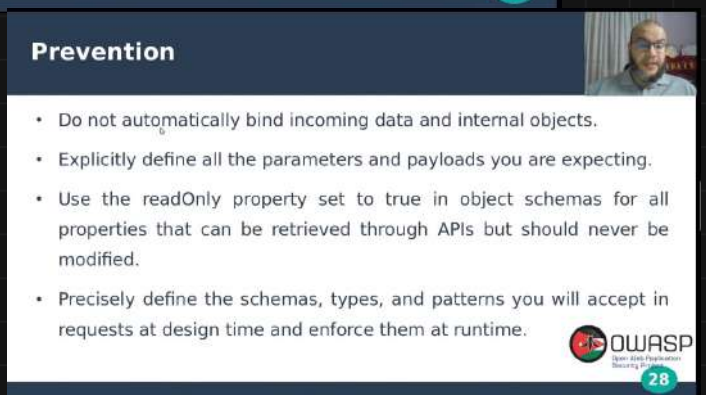
Now I am assuming that there is a credit_balance in the API
, So I will capture request and add the parameter credit_balance = 9999....

How to find?
Basically objects in modern application might contain many property.
Some of this properties will be updated directly by the client

→ permission related properties
Is user_is_admin
Is user_is_vip
→ process dependent properties
Is user_cash → This happen mostly by payment verification
→ internal properties





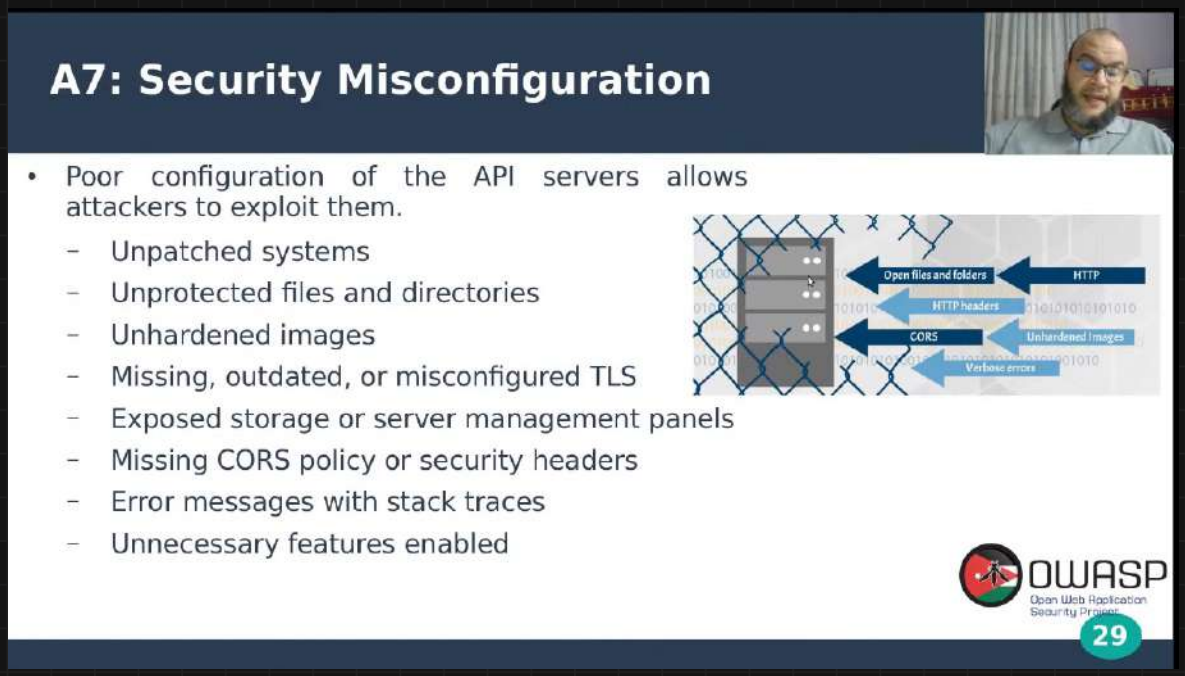


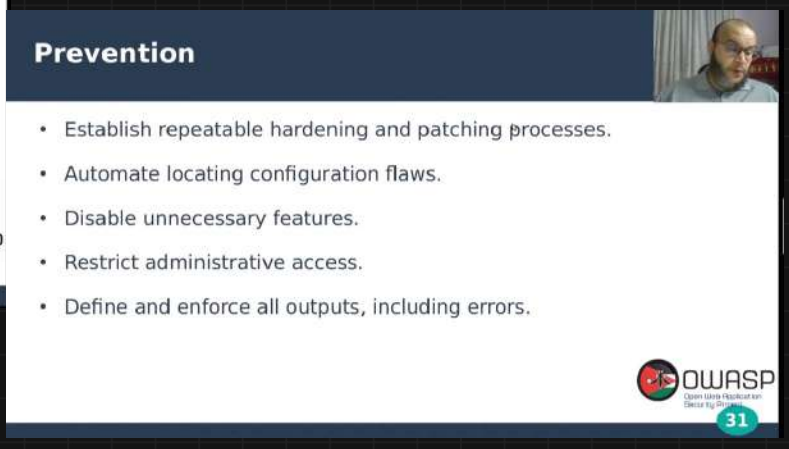
7 - Seevrity Missconfigration :

Try to look for CORS | HTTP VERB issue | Errors | unprotected files + dir's
unpatched Haus (Apache Struts) | common endpoints

Senario 1:
let say you found a bash history file contains the following command:
curl -X GET 'endpoint url' -H 'authorization token'

What to look for:
- unpatched system
- unprotected files and directories
- unhardened images
- missing outdated or missconfigured TLS
- exposed storage or server management panel
- Missing CORS policy or security headers
- Error messages with stack traces
- unnecessary featured enable





8- injection

inject query (SQL injection)

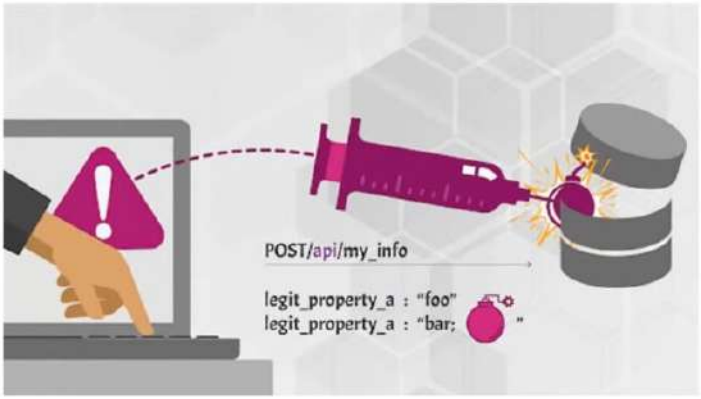
→ GET /api/users/username/info

→ POST /api/accounts/recovery
username = 'brahim'

→ /api/config/restore
appid = 123
↳ I will try to inject commands or SQL injection

A8: Injection

- Attackers construct API calls that include SQL, NoSQL, LDAP, OS, or other commands that the API or the backend behind it blindly executes.



OWASP Open Web Application Security Project

32

Samsung SmartThings Hub (2018)

- SmartThings Hub is able to communicate with cameras
- credentials API can be used to set credentials for remote server authentication
- Data is saved in SQLite database
- JSON keys with ; allowed to execute arbitrary queries

```
# using curl from inside the hub, but the same request could be sent using a SmartApp
$ sInj='",_id=0 where 1=2;insert into camera values (123,replace(substr(quote(zeroblob((10000 + 1) /
2)), 3, 10000), \\"0\\", \\"A\\"),1,1,1,1,1,1,1,1,1,1,1,1,1,1,1);--":"'
$ curl -X POST 'http://127.0.0.1:3000/credentials' -d
"{'s3':{'accessKey':'','secretKey':'','directory':'','region':'','bucket':'','sessionToken':'${sInj}'},'
videoHostUrl':'127.0.0.1/'}"
$ curl -X DELETE "http://127.0.0.1:3000/cameras/123"
```

https://talosintelligence.com/vulnerability_reports/TALOS-2018-0556/

OWASP Open Web Application Security Project

33

Prevention

- Never trust your API consumers, even if internal
- Strictly define all input data: schemas, types, string patterns -and enforce them at runtime
- Validate, filter, sanitize all incoming data
- Define, limit, and enforce API outputs to prevent data leaks

OWASP Open Web Application Security Project

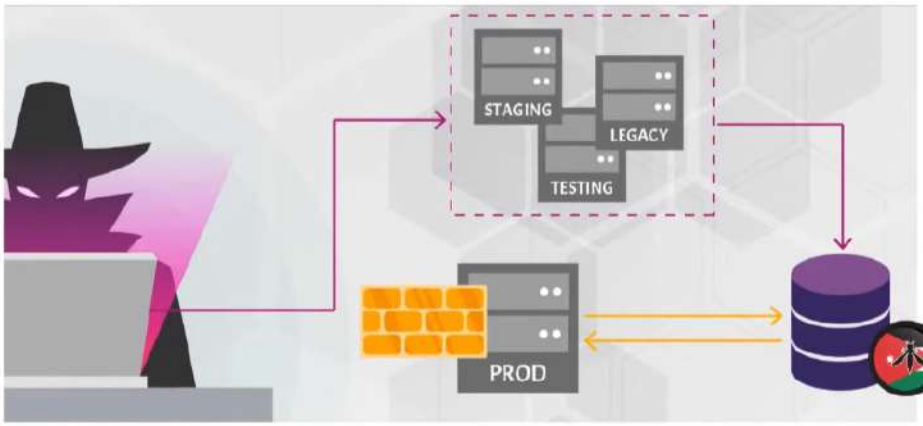
34

9- Improper assests management

- purpose of an API host is unclear and there are no explicit answers to the following;
  - Documentation is not updated.
  - No retirement plan for api version.
  - Host investory is missing or outdated.
  - old or previous api versions are running unpatched.

A9: Improper Assets Management

- Attackers find non-production versions of the API (for example, staging, testing, beta, or earlier versions) that are not as well protected as the production API, and use those to launch their attacks.

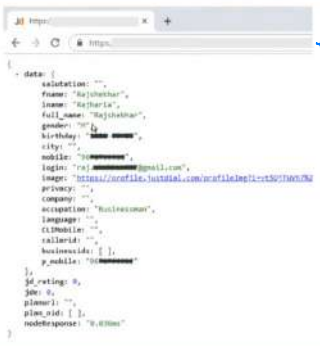


OWASP Open Web Application Security Project

35

JustDial(2019)

example 1



- India's largest local search service
- Had old unused unprotected API
- It was connected to live database
- Was leaking 100 mlnusers' data

OWASP Open Web Application Security Project

36

Tchap

example 2

- Tchap is a messaging app that was launched by the French government and released for internal use. It was hailed as a more secure replacement for Telegram and WhatsApp.
  - The sign-up API had an email address parameter that didn't validate the input format.
  - A security researcher, Elliot Alderson, submitted fs0c131y@protonmail.com@elysee.fr as the address.
  - The code simply verified that the address ended with @elysee.fr, which is a government email domain.
  - The check was successful, and Tchap sent a verification email that got delivered to fs0c131y@protonmail.com belonging to the attacker.
  - The attacker could click the confirmation link, get in, and be able to get [illegible] government chat rooms.

OWASP Open Web Application Security Project

37

Prevention

- Keep an up-to-date inventory all API hosts.
- Limit access to anything that should not be public.
- Limit access to production data, and segregate access to production and non-production data.
- Implement additional external controls, such as API firewalls.
- Properly retire old versions of APIs or backport security fixes to them.
- Implement strict authentication, redirects, CORS, and so fo

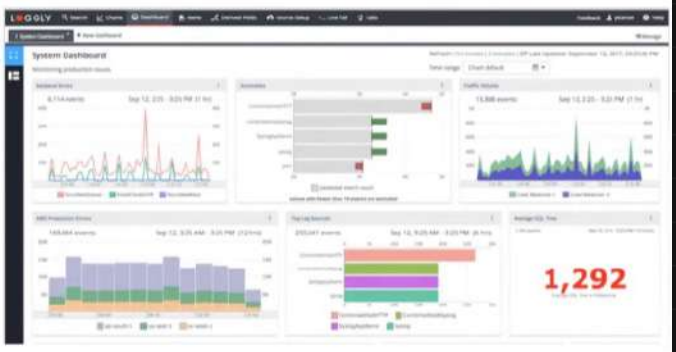
OWASP Open Web Application Security Project

38

10- Insufficient Logging & Monitoring

A10: Insufficient Logging & Monitoring

- You can't react to attacks that you don't know about.
- Logs are important for:
  - Detecting incidents
  - Understanding what happened
  - Proving who did something



OWASP Open Web Application Security Project

39

7-Elevel Japan: 7Pay

- Payment app
- Password reset allowed email change if personal details were supplied
- Attackers used API to try combinations of info for various Japanese citizens
- $510K was stolen from 900 customers
- The company only noticed after too many users complained

OWASP Open Web Application Security Project

40

if they had minotoring System, this will not happened.

Prevention

- Log failed attempts, denied access, input validation failures, any failures in security policy checks
- Ensure that logs are formatted to be consumable by other tools
- Protect logs as highly sensitive
- Include enough detail to identify attackers
- Avoid having sensitive data in logs -If you need the information for debugging purposes, redact it partially.
- Integrate with SIEMs and other dashboards, monitoring, alerting too

OWASP Open Web Application Security Project

41