API Pentesting Mindmap {{GraphQL Attacking}}

- ① Endpoint discovery
  - Common endpoints
    - /graphql
    - /graphiql
    - /graph
    - /explorer
    - etc
  - Common GraphQL IDEs
    - GraphQL Playground
    - GraphiQL
    - etc

Some times it being left in the wild, so an attacker can get benefit from them, it makes the query implementation easier

Both of them can be found at SecLists GraphQl wordlist

- ② Getting started with queries
  - ① Enumeration
    - Basic enumeration — Enumerate schema fields, types and names
    - Deep enumeration — Enumerate the whole database schema
  - ② Analyzing schema
    - Query syntax
      - Data types
      - edges
      - nodes
      - etc
    - Query parameters / Arguments
      - Arguments structure
      - Number of arguments
    - etc

Both of them got payloads in payloadAllTheThings

- ③ Getting started with mutations
  - ① API's documentation
    - Organization's github
    - API documentation in the application
    - Open source project with public documentation
    - Local documentation
    - etc
  - ② Schema analysis
    - Extract mutations according to the type in introspection system
    - Enumerating the whole schema then hunting the mutations
    - etc
  - ③ Application's behavior analysis
    - Modifying item
    - Adding item
    - Item deletion
    - etc

If you was able to control mutations then you might be able to modify / control data in server-side

- ④ Mutations exploit
  - Exploit mutations for:
    - Adding files / attachments
    - Deleting resource
    - Modifying resource
    - Users managements
    - etc
  - Possible mutations functions
    - Profiling systems
    - Logging systems
    - Business systems — e.g: items cart
    - etc
- ⑤ Queries exploit
  - Data retrieval
    - PII data
    - Financial data
    - Internal data
    - etc
- ⑥ Authentication attacks
  - Multiple brute forcing — Brute force more than single credentials through one query due to the nested queries
  - Sessions / Cookies Retrievals — If the targets stores sessions / cookies in the application's database, attacker can retrieve them and inject through request to gain access in user's account
  - Restrictions bypasses — e.g: Account locking bypassing if the application for example got a mutation called: unlock() and it takes argument for the id, simply the attacker can unlock it's account like this: unlock(id: 1) after that the attacker will be able to login again
  - etc
- ⑦ Business logic attacks
  - Mutations
    - Modifying item's price
    - Modifying item's currency
    - Bypassing checkouts — e.g: Adding items to purchased section, then confirm the orders and finally exploited without executing the checkout function
    - Adding / Removing items from other users accounts

Presented with XMind