Api test case	
API1:2023 - Broken Object Level Authorization	
1)change object parameter in get request or in body ,perform Brute Force	1)Use random IDs that cannot be guessed (UUIDs) (Universal Unique Identifier). 2)Check authorization for each client request to access database.
API2:2023 - Broken Authentication	
1)Observe if new token is generated after every login	Authenticate all APIs
2)Remove the token hit the request and observe the response	Provide brute force, rate limiting and lockout protections
3)Change the token of two different user	Set short expiration time for Access Tokens Implement multi-factor authentication Don't use API keys for user authentication. Use them only for client app/project authentication
Expired time of token	
Brute foricing on credenital to check if rate limtting is properly appiled ornot	
Week password acceptance	
Default Credientials	
API3:2023 - Broken Object Property Level Authorization(mass assignment + Excessive Data Exposure)	
Check Every Api Call and observe if Any Excessive data is expose than required	Make sure that API provides only relevant data in responses Classify sensitive and personally identifiable information (PII)
Mass Assignment: Capture the request of two different request of two user having different accesses eg "User" and "Admin" observe request and response find the property which diffening the role of user try to change that property and observe the response	Don't bind incoming data and internal objects properties/ variables blindly Set 'ReadOnly' to properties that are sensitive 'ReadOnly' properties can be read by client using GET method but can't be updated/modified using PUT method
	Whitelist properties that should be updated by the client. Blacklist properties that should not be accessed by client
API4:2023 - Unrestricted Resource Consumption	
Perform Brute forcing eg 429 to many request Send input more then fixed sizes Send nested XML or JSON Send a request with 1000+ levels of JSON or XML nesting.Useful against parsers vulnerable to Billion Laughs or Entity Expansion attacks.	Define proper rate limiting Limit payload sizes Enforce maximum size of data on all incoming parameters and payloads Add proper server-side validation for query string and request body parameters
API5:2023 - Broken Function Level Authorization	
Change the HTTP method from GET to POST D	All access should be deny by default Explicit access grants to specific roles should be required
Login with normal user and try to hit the endpoin of admin user or Use auth token of normal user to admin endpoint.	Authorization checks should be implemented based on the user's group/role
Try changing the endpoints from account to accouns,user to /users,/account_all,/user_all,/ admin,/administrator etc.	Do not rely on the client to enforce admin access
API6:2023 Unrestricted Access to Sensitive Business Flows	

Identify the api endpoint which requied to run business try to find vulnerability in that api eg create multiple account , Automated Purchasing / Ticketing Abuse,	To mitigate this vulnerability we first need to identify the sensitive functionalities of the application and check if it harms the business if they are excessively used. And provide protection to them.
	Rate limiting,Device fingerprinting / IP address tracking,Human detection: using captcha
API7:2023 Server Side Request Forgery	
Check api body contain any url ,Try to Acesses localhost internal ip	Disable HTTP redirections.
Check if the server supports unsupported URI schemes.eg "url": "file:///etc/passwd"	Validate and sanitize all client-supplied input data.
Out-of-Band SSRF (OOB) Detection: If not able to see immedidate responseUsing Burp Collaborator check if is there any DND in Burp Collaborator	
API8:2023 Security Misconfiguration	
A Cross-Origin Resource Sharing (CORS) policy is missing or improperly set	
Check if error messages include stack traces, or expose other sensitive information	
Unnecessary HTTP Methods Enabled	
Missing input validation	
Improper SSL/TLS Configuration: Misconfigured SSL/TLS settings, such as using weak cipher suites, expired certificates etc.	
API9:2023 Improper Inventory Management	
API10:2023 Unsafe Consumption of APIs	When evaluating service providers, assess their API security posture.
File Upload	Ensure all API interactions happen over a secure communication channel (TLS).
upload file with large size,double extention,	Always validate and properly sanitize data received from integrated APIs before using it.
Change the path of file upload to check if internal file are accessible	
Other Pentest Scenarios:	
Authenticated activities without authentication (same as web)	
Convert **Content-Type: applicaton/Json** to **text/HTML** and add XSS payloads	
CRLF Injection: 1) Add **%0d%0aTest** in the request and if application is vulnerable to CRLF, Test word is reflected in the response e.g.: GET /Vapi/api5/getuser.php**%0d%0aTest** HTTP/1.1	
Checking for content type XML/JSON supports?: 1) Convert body parameter to XML/JSON and observe is you are getting response accordingly	
Buffer Overflow: 1) Test for a very big input.e.g.: 5-6 lines of random text in username field etc.	
Fuzzed Input: 1) Give simple mathematicle expression in the input filed e.g.:username: 4+5+6(it shouldn't get executed)	

Host header injection: 1) Change the host and observe the **Location** field in response. 2)Add **X-Forwarded-Host**: demo.testfire.net below the main host	
Directory traversal : 1) /../../ in API endpoint or go one folder back and observe the reponse.	
Testing for Directory Listing: 1) Modify endpoint by removing all the sub folder names and observe the response.	
Injection: 1) Injection: Perform brute force for SQL, XSS payloads and other injections on API qury parameters	
Missing Input Validation: 1) Input XSS scripts into fields like username, mobile number, and email ID. These fields should reject special characters such as <>?+()*&. 2) Ensure that input fields only accept data corresponding to their types. For instance, mobile number fields should only accept numerical characters, while username fields should not accept numbers.	
API functionality specific testing: 1) Accoring to API calls functionlity perfomr additional test scenarios on api. E.g: OTP testing: test all OTP scenarios. File upload: test all file uplaod scenarios.	
File Download 1)Directory Traversal-egGET /download?id=../../etc/passwd 2)Manipulated Content-Type Header eg-GET /download with Accept: text/html or Content-Type: application/javascript	