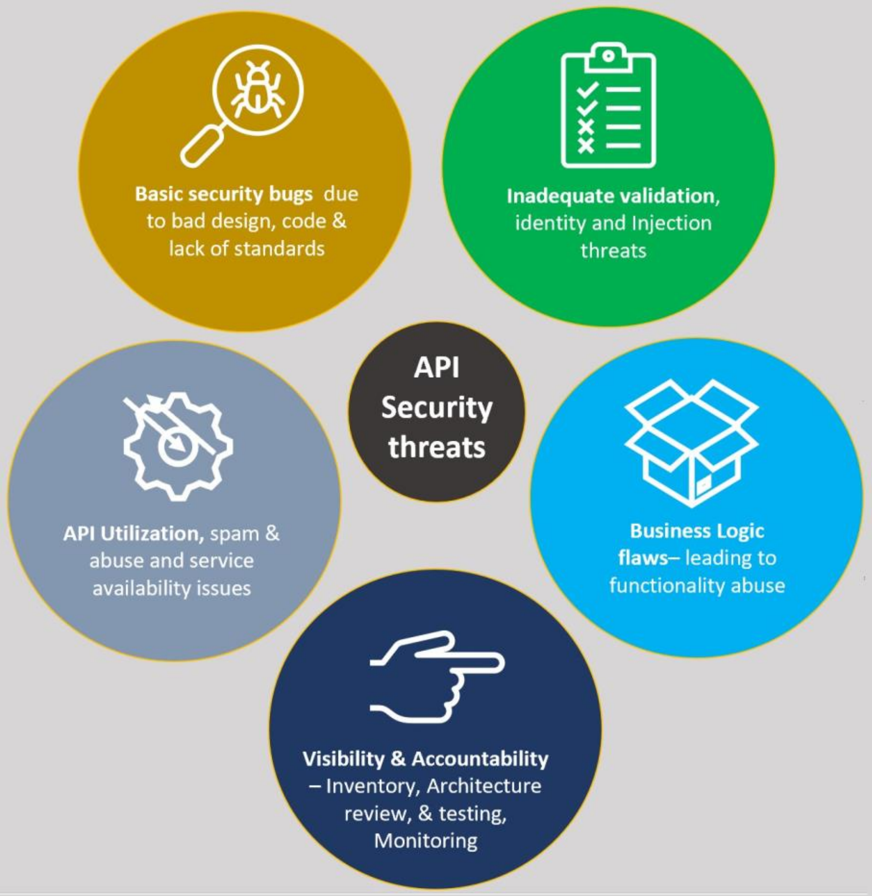
API Security threats

In many cases, API Security is an afterthought. When security's an afterthought, sometimes enterprises don't take the right precautions before exposing their APIs and services. In many cases, it's simply a lack of visibility in what's exposed, what controls are in place, and what is actually happening with the data.
Some key API Security Risks

Open API	Private API	Private API
High Risk as open to everyone on the internet	Medium Risk - open only to Partners	Lower Risk - open to Users/Apps within enterprise


Basic security bugs due to bad design, code & lack of standards
Inadequate validation, identity and Injection threats
API Security threats
API Utilization, spam & abuse and service availability issues
Business Logic flaws– leading to functionality abuse
Visibility & Accountability – Inventory, Architecture review, & testing, Monitoring

- **No Visibility & Accountability:** No accountability in maintaining a list of API's including documentation and properly deployed API versions can lead to API security threats, which leads to exploits on exposed debug endpoints, deprecated API versions, Ghost endpoints, including unknown security issues not being discovered during testing due to lack of visibility

- **Basic Security bugs due to bad design & code** : APIs that don't follow some standardized best practices and complex versioning, bad call mapping, mass assignment, lead to security risks like authorization problems, improper password storage, no restriction on size of requests, etc

- **Inadequate Validation:** No User or API or endpoint/resource validation can lead to exposure or modification of data leading to data breaches or compromised networks. (Broken Object Level Authorization leading to lateral movement, Broken Function Level Authorization, Broken User Authentication, Unconventional user input, etc)

- **Service availability threats** – As APIs become popular, they become more valuable to attackers. APIs are now a target for denial of service (DoS) attacks, bruteforcing and need to be adequately protected (restrictions on the no or size of requests, whitelisting, etc) so that API's can still be processed when the service is under load

- **Business Logic flaws leading to Functionality abuse :** Attacks like credential stuffing bots, scraper bots scraping product listing, pricing, specifications, user data, etc can lead to compromise of data if there are business logic flaws where users are overly trusted and are given overly access to API resources and displaying sensitive data where unnecessary

API Security LifeCycle

Strategy	• Define goals and requirements for each API	• Define Roles and responsibilities for Managing API's (Inventory, Design & Architecture, testing, specifications,, etc)					
Architecture & Design	• Open design (no security through obscurity)	• Design Architecture, data flows and perform threat modelling	• Design API interfaces to limit exposure	• Define roles, access for each endpoint	• Define authentication & authorization for type of resources and flow policies	• Type of authorization for each API endpoint (passthrough, API key, OAUTH 2.0)	• Define security policies for flows
Implement	• Use standard code and versioning	• Use OpenAPI standards and build Yaml / swagger files using editors like swagger editor	• Ensure Authentication for access to API endpoints	• Ensure endpoint resource explicitly authorized	• Ensure logging & compromise recording,	• API Gateway & proxy for microservices to limit aperture & access. Consider a WAF to protect API Gateway	• Use JWT for auth if large no of microservices & spread across multiple cloud environments
Review & Protect	• Perform a Risk Assessment to identify systems / data affected if APIs are compromised & implement a treatment plan	• Antibot protection and rate limiting protection to prevent Public API's from scrapper/scalping bots and DDoS	• Build AppSec test cases basis postman/swagger file and sample resource data	• Test authentication of API's by accessing API endpoints that require credentials	• Walkthrough all roles used for your API and test each role for any authorization issues	• Assessment around API OWASP top 10 like checks for input validation, injection vulnerabilities, etc	• Perform an external Pentest with a 3rd party to test your API's
Governance	• inventory with name, purpose, payload, usage, access, live & retired date, Owner - Should include test, dev, UAT & public	• Build a security calendar to periodically test API's for security bugs	• Review land monitor API logs for any misuse	• Secrets and certificate management	• API analytics - clear insights to measure, tune, and monitor your API program	• Track risks to closure & ensure Change Mgmt process before Go-Live or changes	• Secure coding guidelines & regular Trainijg to ensure minimum security bugs