SLIIT

Discover Your Future

[redacted]

IE2062 – Web Security

Lecture 6 – Introduction to API Security

SLIIT
FACULTY OF COMPUTING

Outline

- What exactly is an API ?
- How do they Work ?
- API Flexibility
- What’s the difference between all of these API’s types ?
- API Security

What is an API

[redacted]

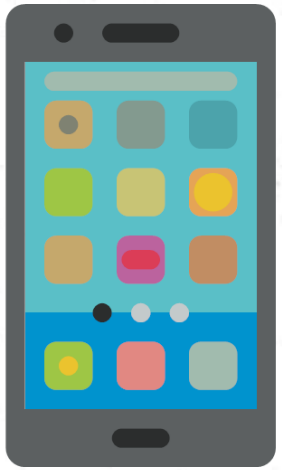
- An API is a set of definitions and protocols for building and integrating application software. API stands for application programming interface.

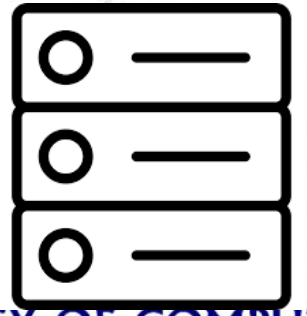
- APIs are beneficial because they allow developers to add specific functionality to an application, without having to write all of the code themselves,

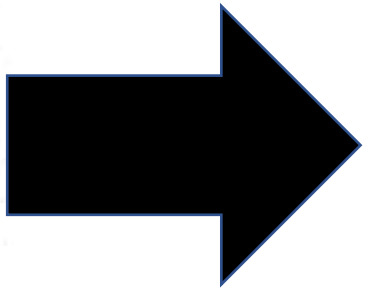
- This can simplify app development, saving time and money.

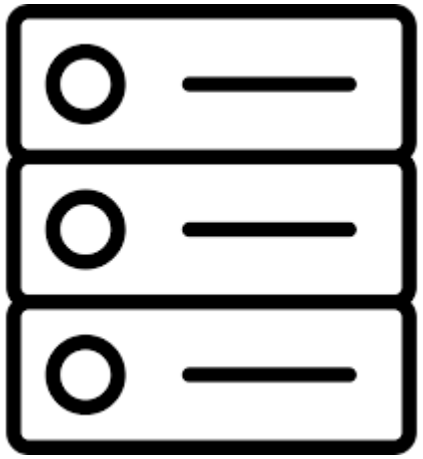
- When you're designing new tools and products—or managing existing ones—APIs give you flexibility; simplify design, administration, and use; and provide opportunities for innovation.

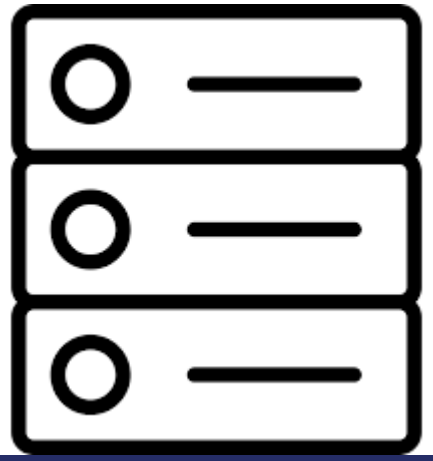
An API is an alternative to the user interface











An API is a Contract between application and service

- Consumer vs Provider

  A software application is often the “Consumer” of the API.

  when an API is offered over a network for such consumption, the service that offers the API is said to be the “Provider ” or “API Provider”.

- The APP may outsource requirements for data or functionality through API by “calling” that API

  Patient Record

  Location represented as pin on a map

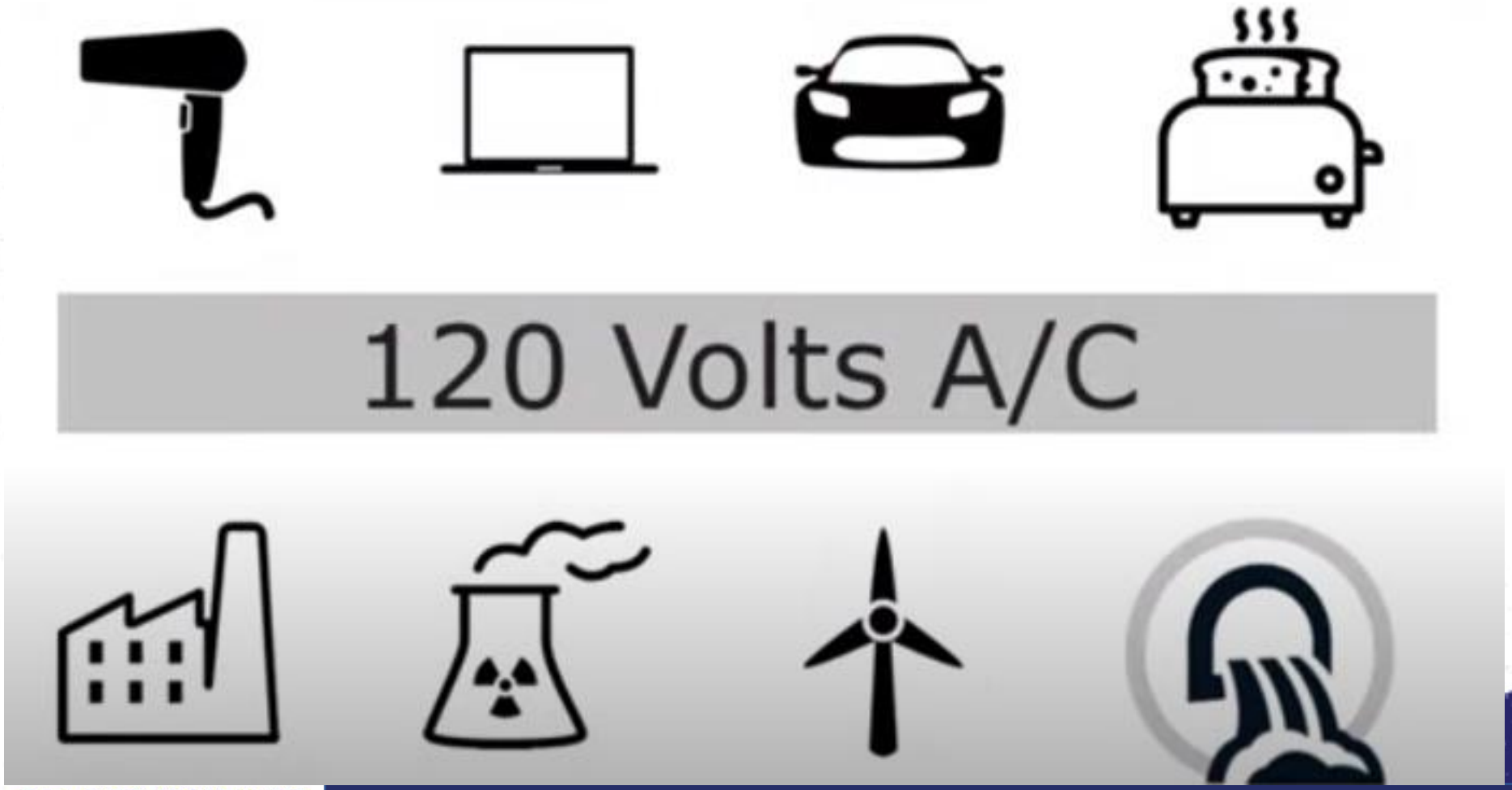
  The execution of a financial transaction

- It’s a technical contract

  Like a legal contract, it represent and understanding by all parties involved.

  The contract also represents agreed-upon standards.

Real world



API world – clients (“API Consumers”) & Servers (“API Providers”)

Different types of Consumers

- Web Apps
- Desktop Apps
- Server Apps
- Mobile Apps
- Devices (as in the internet of things)

Flexibility

Difference between all these API Types

LSUD vs SSKD

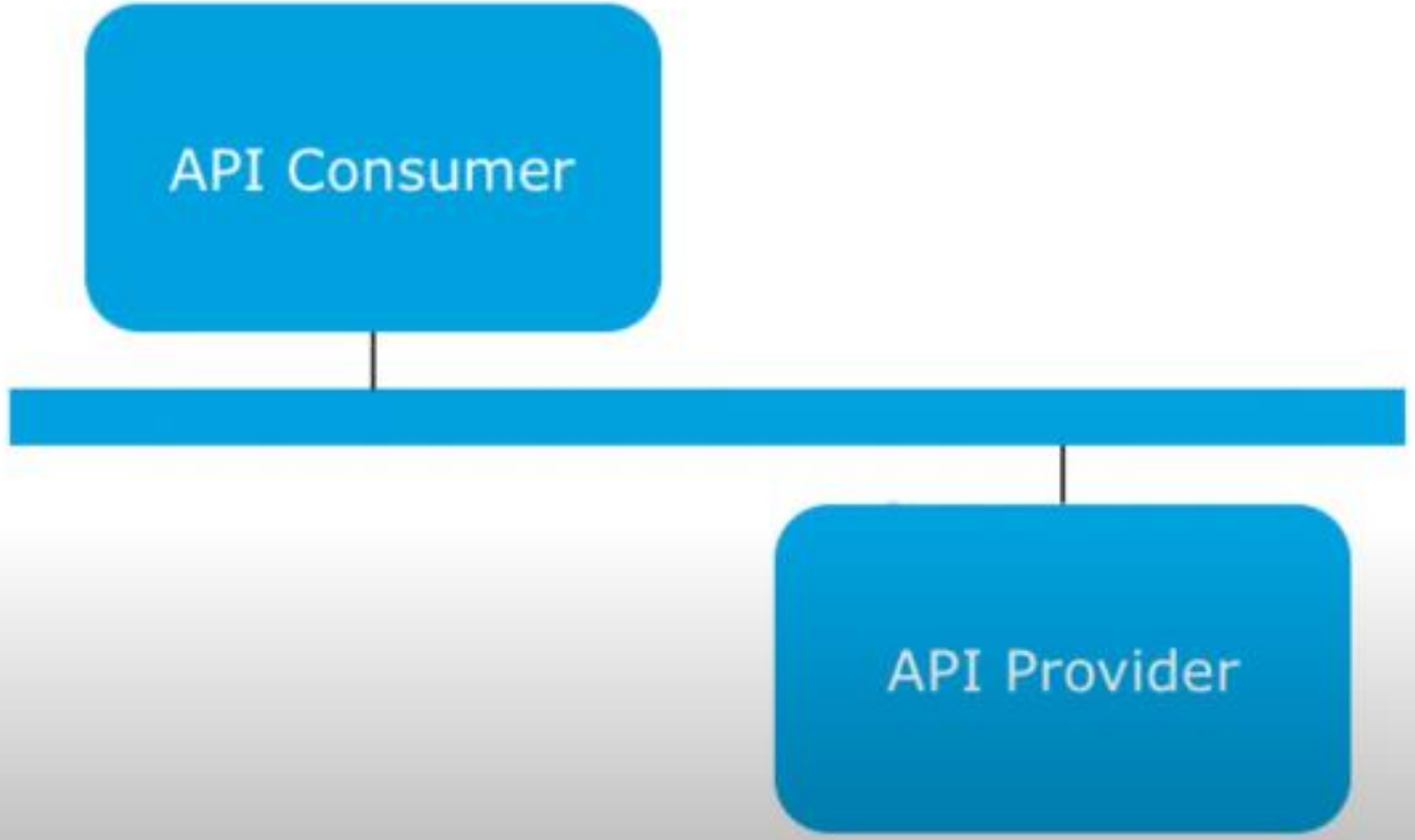
Types of APIs

- Web / Network
- Product
- Browser
- Standard
- System / Embedded

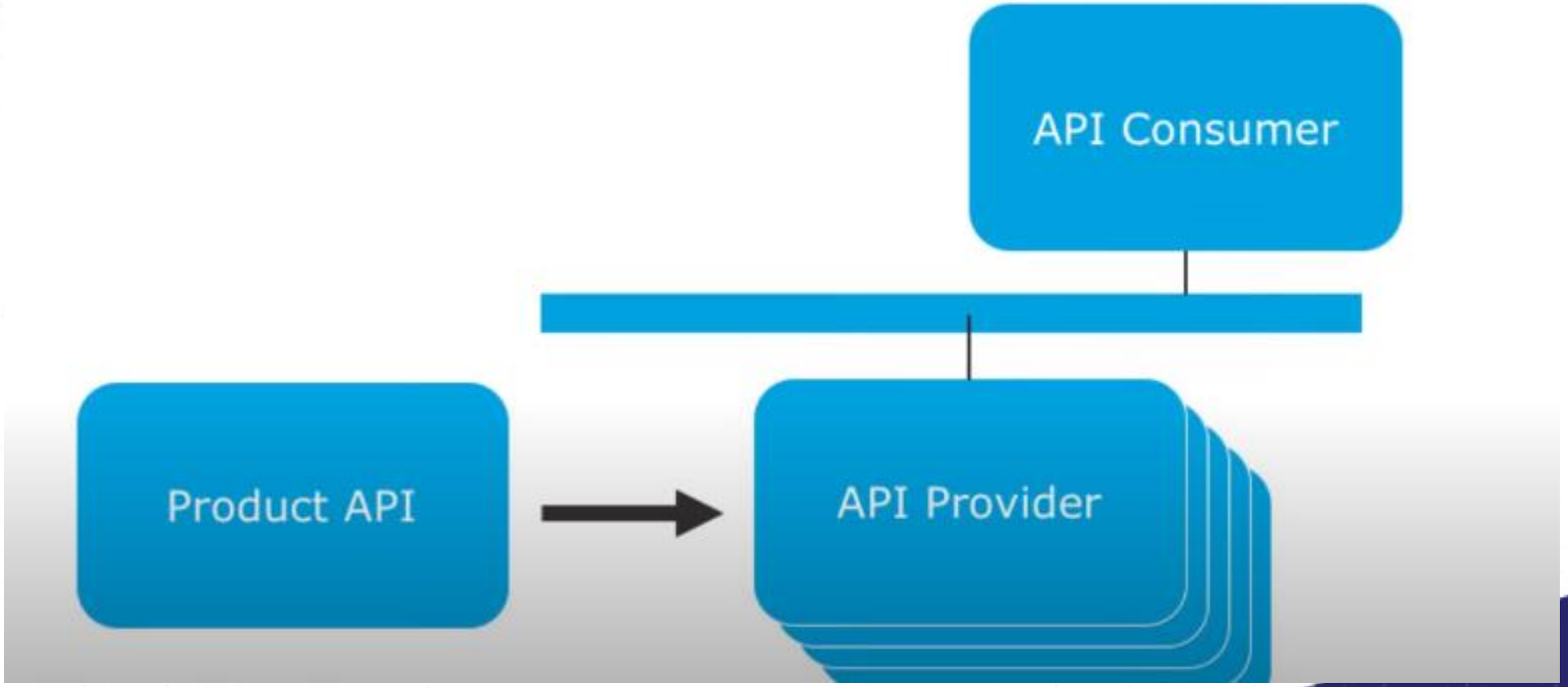
Main types of Web APIs

- ❖ Open APIs:
- ❖ Partner APIs:
- ❖ Internal APIs:
- ❖ Composite APIs:

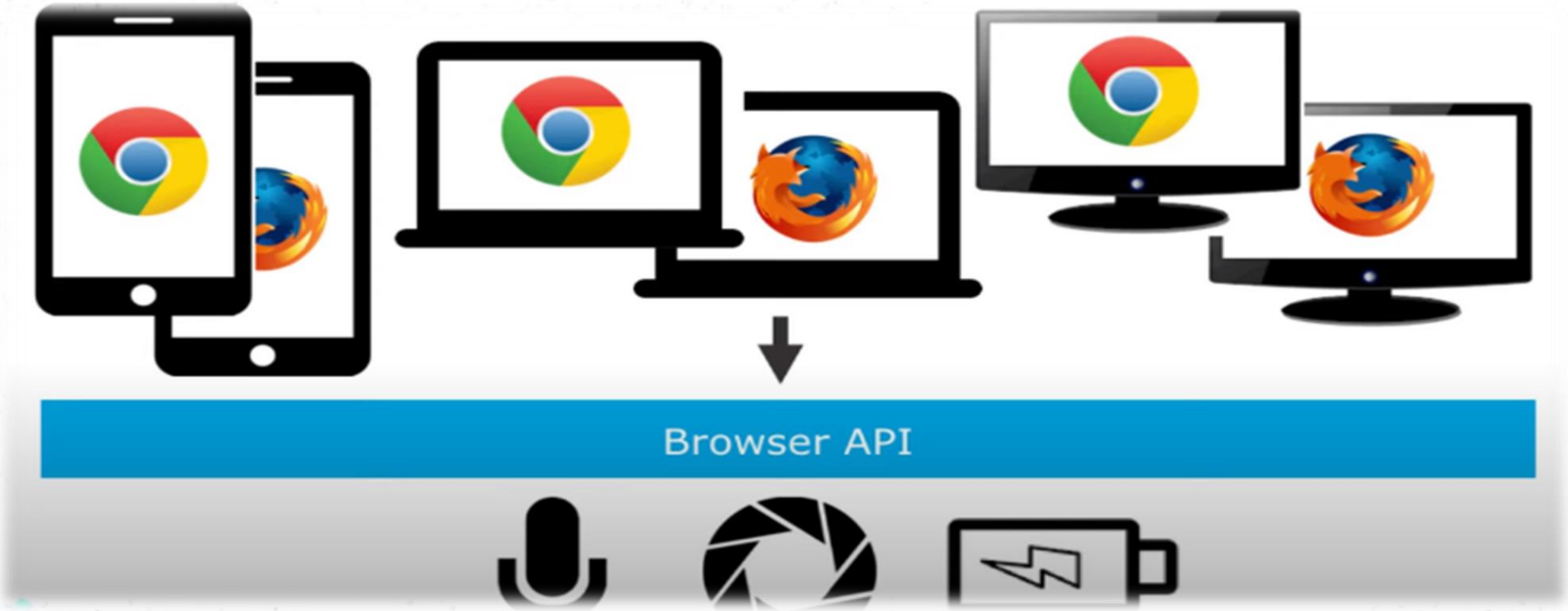
Web / Network API



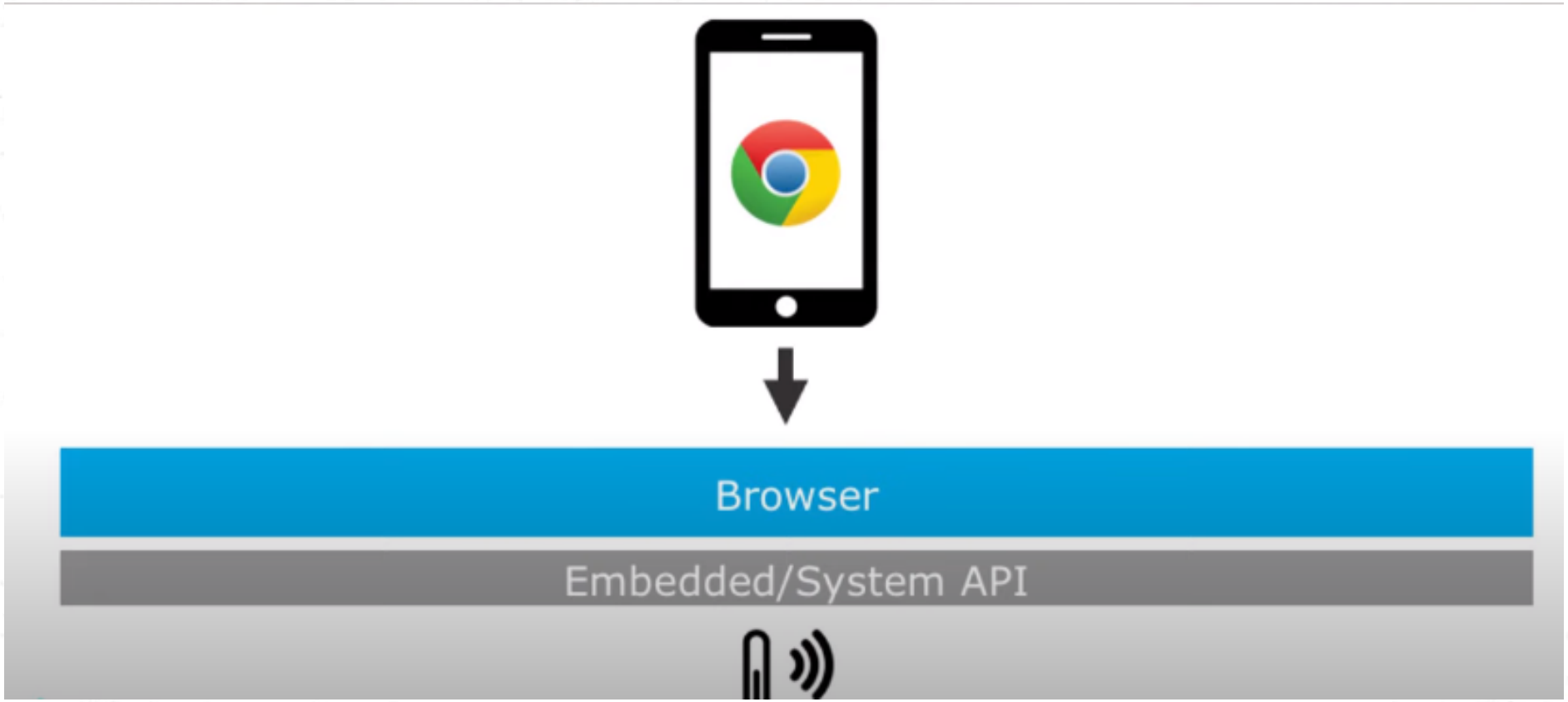
Product API



Browser API



System / Embedded API



Architectural Styles

- Typical Restful
- RPC (XML-RPC, gRPC, etc)
- Push / Streaming (Publish / Subscribe)
- GraphQL
- Browser/Native
- Indirect (eg : Evernote)

API Architectures and Protocols

[redacted]

- An API protocol defines the rules for API calls: it specifies accepted data types and commands. Different API architectures specify different protocol constraints.

REST

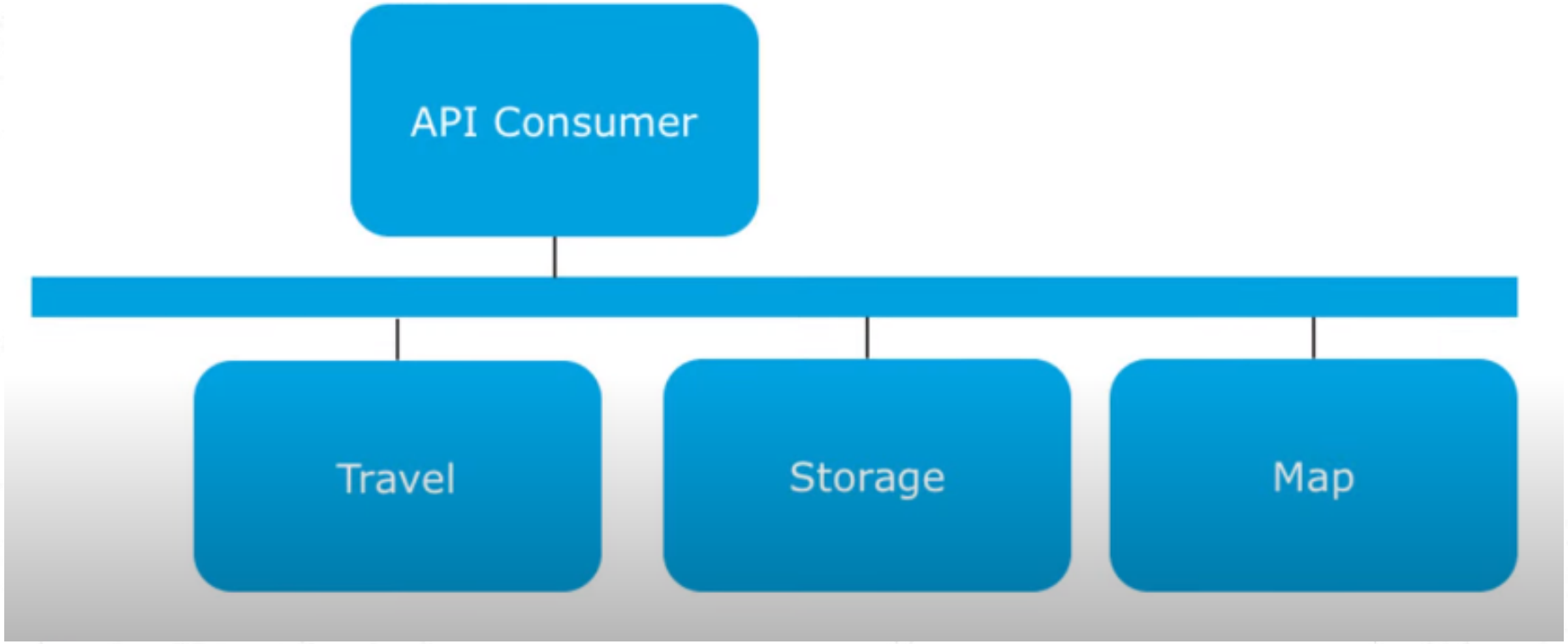
- REST (representational state transfer) is a very popular web API architecture. To be a REST API, an API must adhere to certain architectural constraints, or principles, including:
- **Client-server architecture:** the interface is separated from the backend and data storage. This allows for flexibility, and for different components to evolve independent of each other.

- **Statelessness:** no client context is stored on the server between requests.
- **Cacheability:** clients can cache responses, so a REST API response must explicitly state whether it can be cached or not.
- **Layered system:** the API will work whether it is communicating directly with a server, or through an intermediary such as a load balancer.

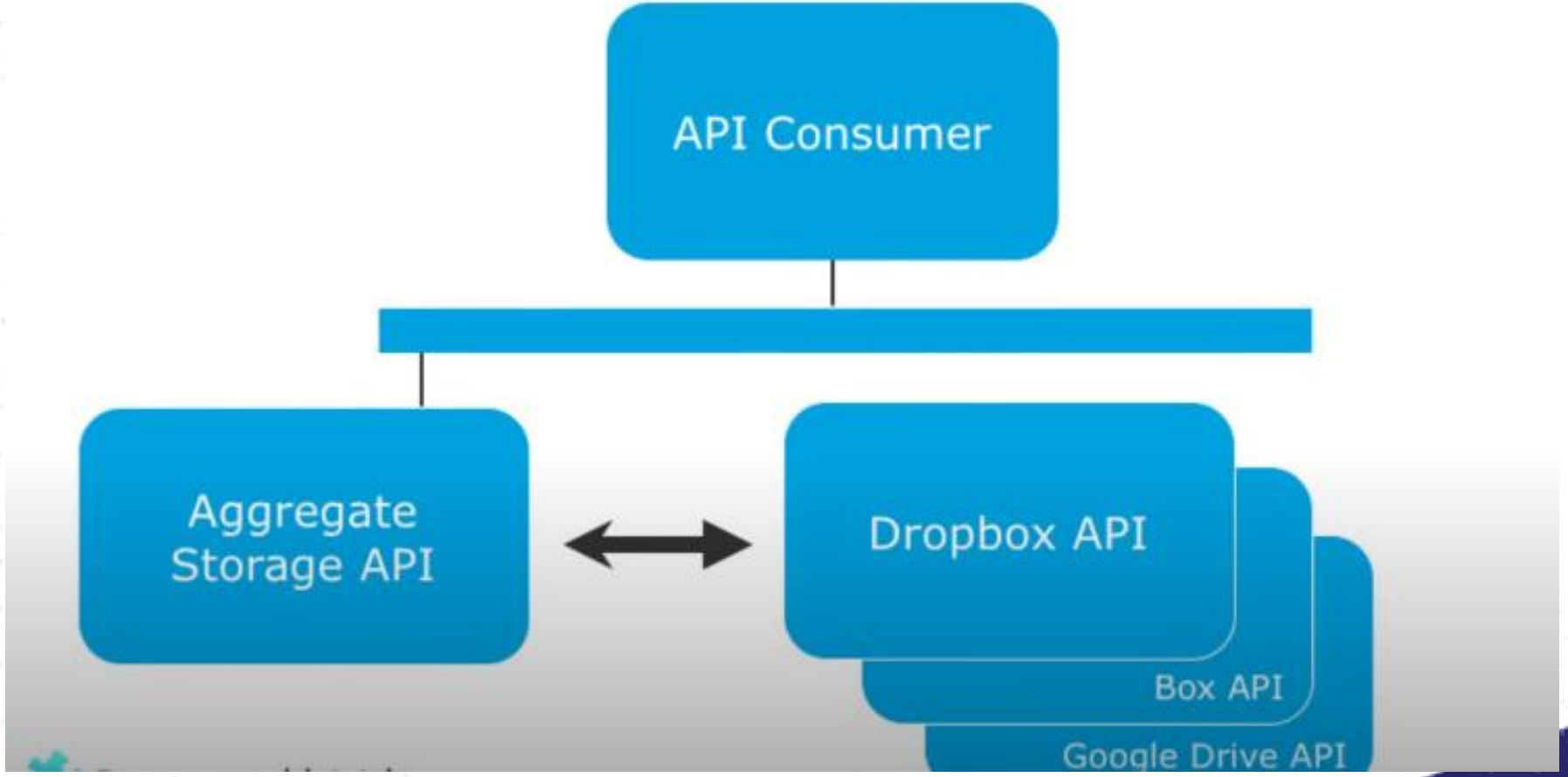
API Scopes

- Single purpose API
- Aggregate API
- Microservice API

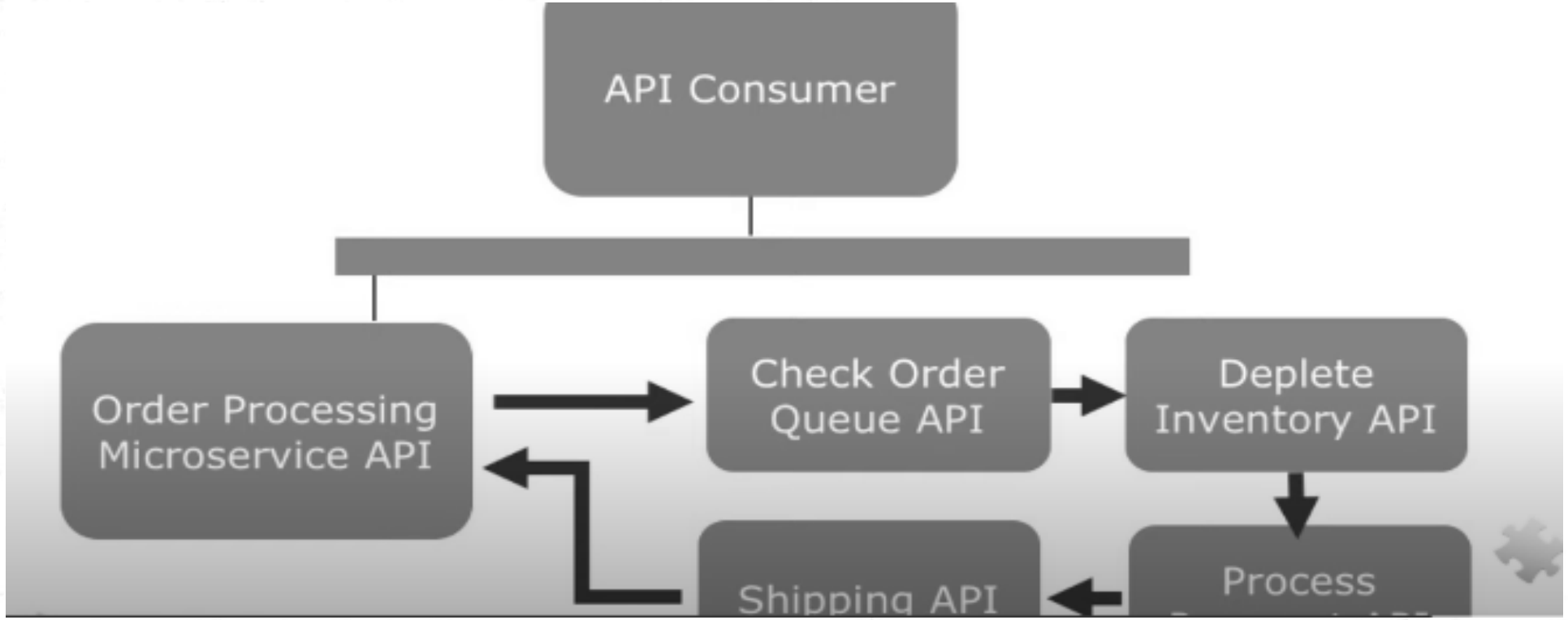
Single purpose API



Aggregate API



Microservice API



[redacted]

API Security

Overview of API Attack vectors

Common API Security Issues

- Access Controls
  - Authorization
  - Authentication
- Input Validation
- Rate Limiting
- Improperly secured endpoints
- Restricting HTTP methods
- 3rd party API abuse
- Other Application logic errors

Access Controls

- Access control schemes tend to follow a pattern.
  - Client makes a request to something that requires authentication
  - Server process auth request, check for things like
    - If an account / session exists
    - If the requested resource within access scope of the client
  - If successful, sever returns a token, session id, or other identifier to mark the session.
  - Further authenticated requests will follow the a similar pattern throughout the session.

Access Control Bugs

Common ways of testing access control bugs include :

- Enumerating potentially restricted endpoints.
- Modifying session tokens
- Reusing older session tokens
- Attempt to by pass restriction on access with IDOR
- Modifying the request with additional parameter like &admin=true

Input Validation

- Common place to test in API
  - Within the request header
  - Parameters within the URL
  - Parameters within the request
  - File uploads (PUT / DELETE requests)
  - Different request Methods

Input Validation Bugs

[redacted]

These include :

- Improper parameterization of requests within application logic
- Lack of input sanitization
- Insufficient controls for data types passed (file upload bugs, Unicode bugs)

Rate Limiting

Common ways to test rate Limiting

- Make requests in varying states of authentication.
- As an authenticated user
- As an unauthenticated user
- As a developer
- As a bot
- With a deactivated account
- With bogus credentials

❑ An API with improperly implemented rate limiting can be used to make an abnormal of requests to enumerate the application and potentially cause other issues.

Restricting HTTP Methods

- APIs are built to support a number of HTTP methods. Determining what the application supports are very important when fuzzing the API.

- Sometime the scope of specific methods is too board, leading a user to be able to PUT, DELETE, POST, etc parts of the API that it shouldn't.

3rd party API abuse

- There are few interesting attack vectors
- Request Splitting
- SSRF – Server Side Request Forgery
- Unhandled input from 3rd party

API Security Controls

- Stop Anonymous Proxy Networks
- Designate Allowed IP Ranges
- Geo Filtering
- Filter Bots from your Single Page Applications and Browser APIs
- **OAuth 2.0**

The END