**OWASP API Security Top 10**

[**API1:2019 – Broken object level authorization** 2](#_Toc79550865)

[**Các tình huống tấn công ví dụ** 2](#_Toc79550866)

[**Làm thế nào để ngăn chặn** 3](#_Toc79550867)

[**API2:2019 – Broken Authentication** 4](#_Toc79550868)

[**Các tình huống tấn công ví dụ** 4](#_Toc79550869)

[**Tình huống 1** 4](#_Toc79550870)

[**Tình huống 2** 4](#_Toc79550871)

[**Làm thế nào để ngăn chặn** 5](#_Toc79550872)

[**API3:2019 – Excessive data exposure** 6](#_Toc79550873)

[**Các tình huống tấn công ví dụ** 7](#_Toc79550874)

[**Làm thế nào để ngăn chặn** 7](#_Toc79550875)

[**API4:2019 – Lack of resources and rate limiting** 8](#_Toc79550876)

[**Tình huống tấn công ví dụ** 8](#_Toc79550877)

[**Tình huống 1** 8](#_Toc79550878)

[**Tình huống 2** 9](#_Toc79550879)

[**Làm thế nào để ngăn chặn** 10](#_Toc79550880)

[**API5:2019 – Broken function level authorization** 10](#_Toc79550881)

[**Tình huống tấn công ví dụ** 10](#_Toc79550882)

[**Làm thế nào để ngăn chặn** 11](#_Toc79550883)

[**API6:2019 – Mass assignment** 12](#_Toc79550884)

[**Tình huống tấn công ví dụ** 12](#_Toc79550885)

[**Làm thế nào để ngăn chặn** 13](#_Toc79550886)

[**API7:2019 – Security Misconfiguration** 14](#_Toc79550887)

[**Tình huống tấn công ví dụ** 14](#_Toc79550888)

[**Tình huống 1** 14](#_Toc79550889)

[**Tình huống 2** 14](#_Toc79550890)

[**Tình huống 3** 14](#_Toc79550891)

[**Làm thế nào để ngăn chặn** 15](#_Toc79550892)

[**API8:2019 – Injection** 16](#_Toc79550893)

[**Tình huống tấn công ví dụ** 16](#_Toc79550894)

[**Làm thế nào để ngăn chặn** 17](#_Toc79550895)

[**API9:2019 – Improper Assets Management** 18](#_Toc79550896)

[**Tình huống tấn công ví dụ** 18](#_Toc79550897)

[**Làm thế nào để ngăn chặn** 19](#_Toc79550898)

[**API10:2019 – Insufficient Logging & Monitoring** 20](#_Toc79550899)

[**Tình huống tấn công ví dụ** 20](#_Toc79550900)

[**Tình huống 1** 20](#_Toc79550901)

[**Tình huống 2** 21](#_Toc79550902)

[**Làm thế nào để ngăn chặn** 22](#_Toc79550903)

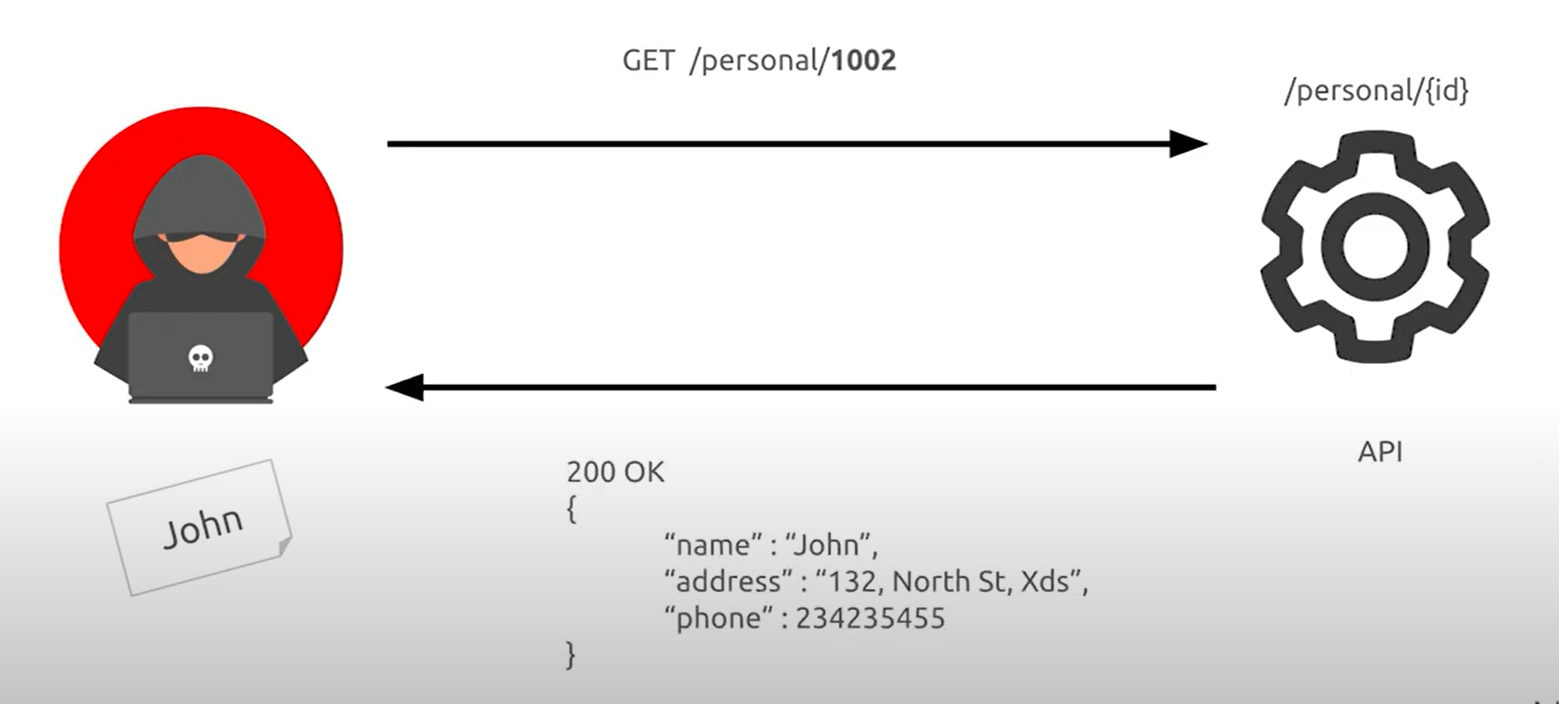
# **API1:2019 – Broken object level authorization**

Các API có xu hướng để lộ các parameter xử lý các đối tượng, tạo ra vấn đề tấn công kiểm soát truy cập. Kiểm tra quyền hạn đối với các đối tượng trong mọi chức năng truy cập nguồn dữ liệu khi nhận đầu vào từ người dùng.

## **Các tình huống tấn công ví dụ**

Người dùng John có id là 1002.

Người dùng John truy cập đến api GET /personal/1002 sẽ trả về thông tin của người dùng.

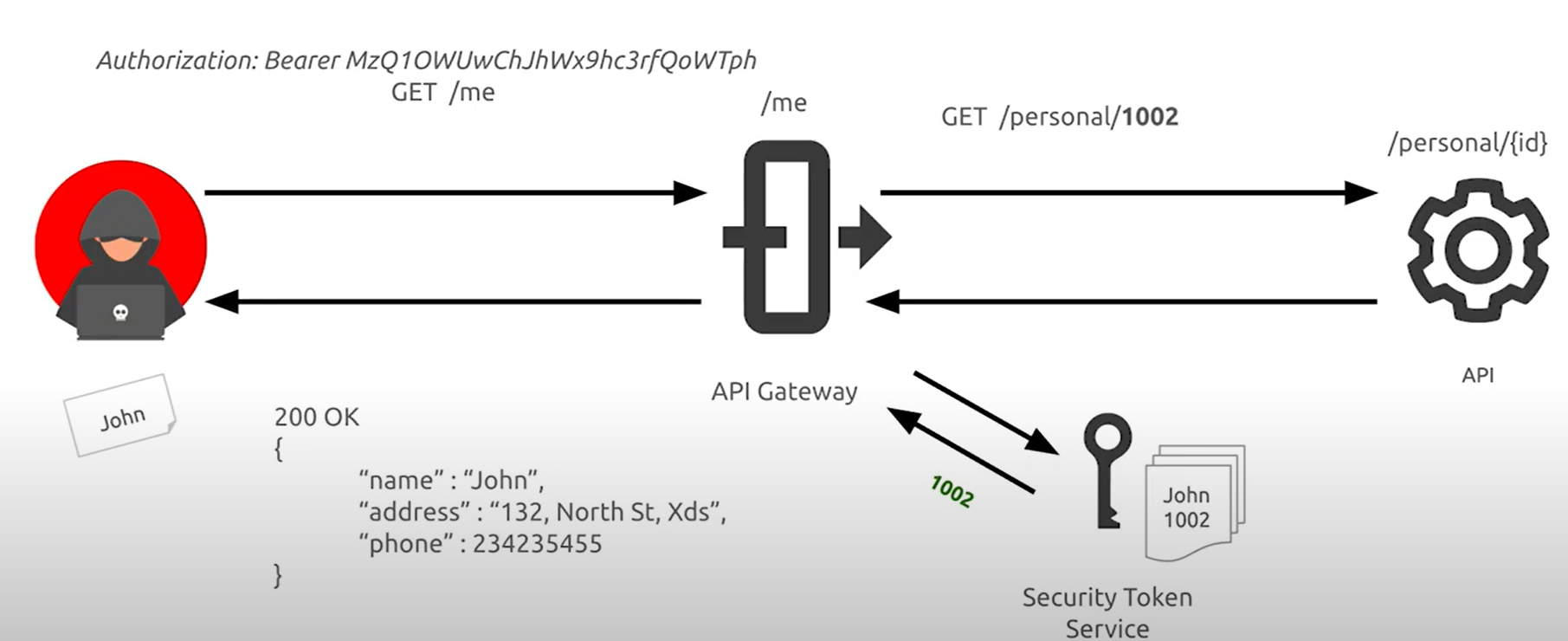


Người dùng John tiếp tục thay đổi id 1002 thành 1001 để truy vấn đến api GET /personal/1001 và thu được thông tin của người dùng có id là 1001 là Mary.

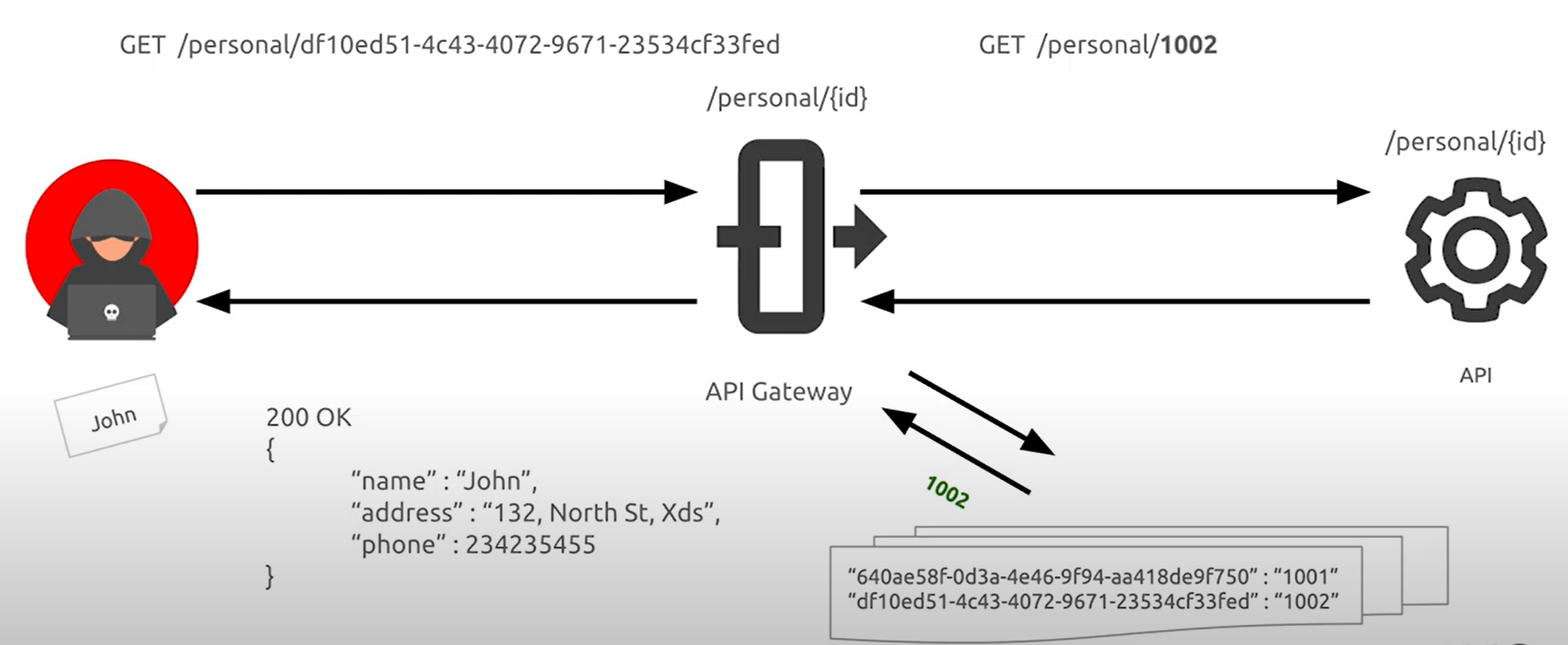


## **Làm thế nào để ngăn chặn**

* Thực hiện cơ chế phân quyền phù hợp dựa trên chính sách người dùng và phân cấp hệ thống.
* Viết các test case kiểm tra để đánh giá cơ chế phân quyền. Không triển khai project có điểm yếu không vượt qua các test case.
* Sử dụng cơ chế phân quyền để kiểm tra xem người dùng đã đăng nhập có quyền truy cập để thực hiện hành động được yêu cầu hay không.



* Sử dụng những giá trị ngẫu nhiên và không thể đoán trước để làm id cho các bản ghi dữ liệu databases.



# **API2:2019 – Broken Authentication**

Cơ chế xác thực thường được triển khai không chính xác, cho phép những kẻ tấn công xâm phạm mã thông báo xác thực hoặc khai thác các lỗ hổng triển khai để giả định danh tính của người dùng khác tạm thời hoặc vĩnh viễn. Làm giảm khả năng xác định khách hàng / người dùng của hệ thống, ảnh hưởng đến bảo mật API nói chung.

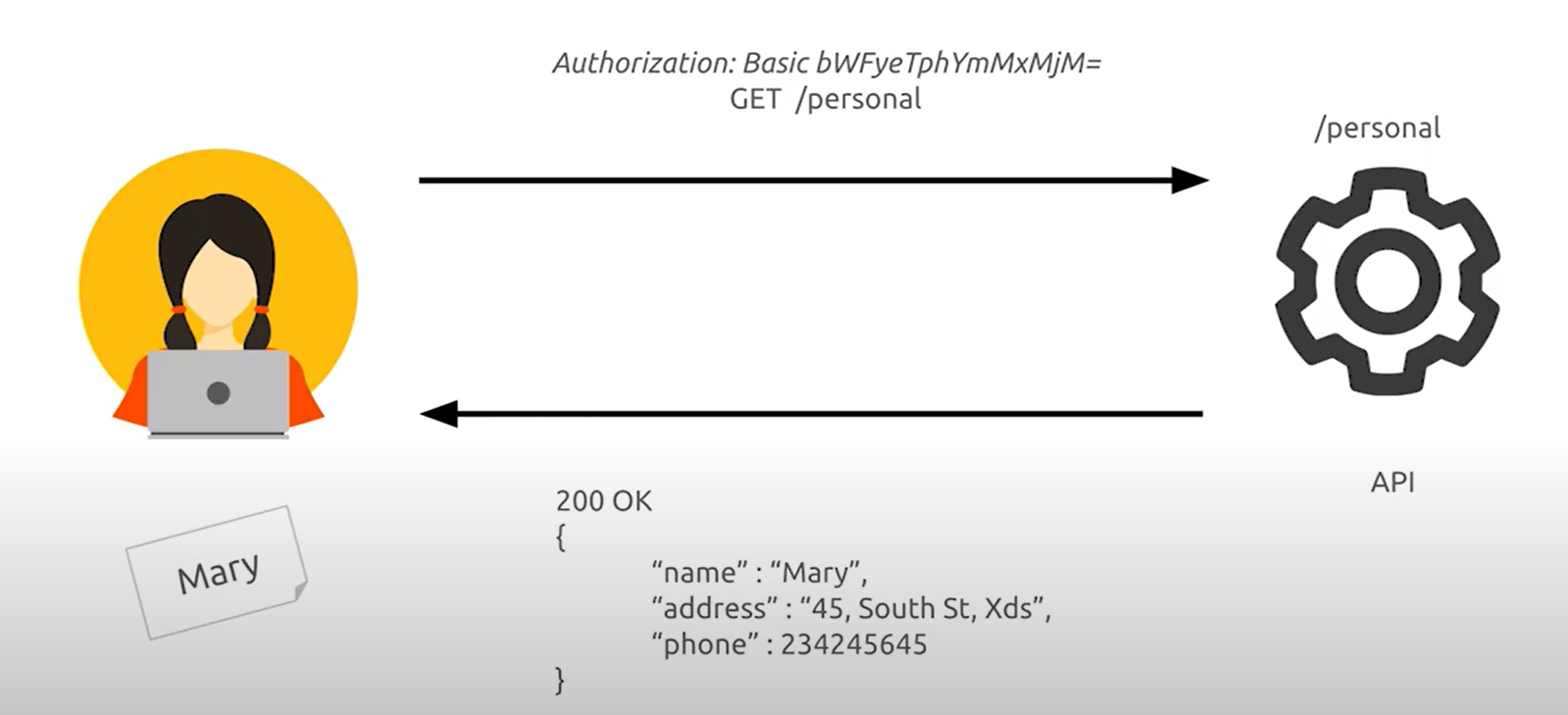
## **Các tình huống tấn công ví dụ**

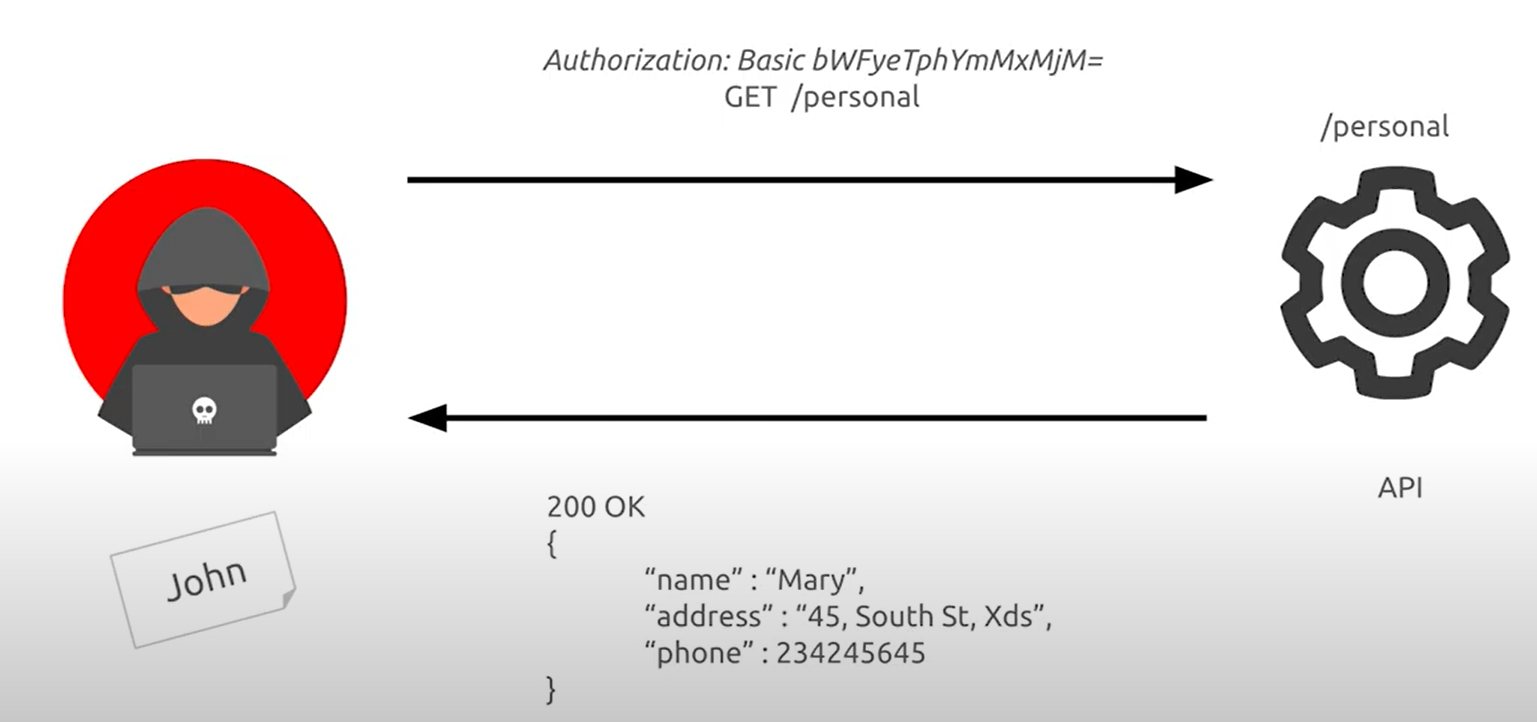
## **Tình huống 1**

Credential stuffing (sử dụng danh sách tên người dùng / mật khẩu đã biết), là một cuộc tấn công phổ biến. Nếu ứng dụng không triển khai các biện pháp bảo vệ tự động hoặc phát hiện tự động credential stuffing, ứng dụng có thể được sử dụng như một công cụ brute force mật khẩu để xác định xem thông tin đăng nhập có hợp lệ hay không.

## **Tình huống 2**

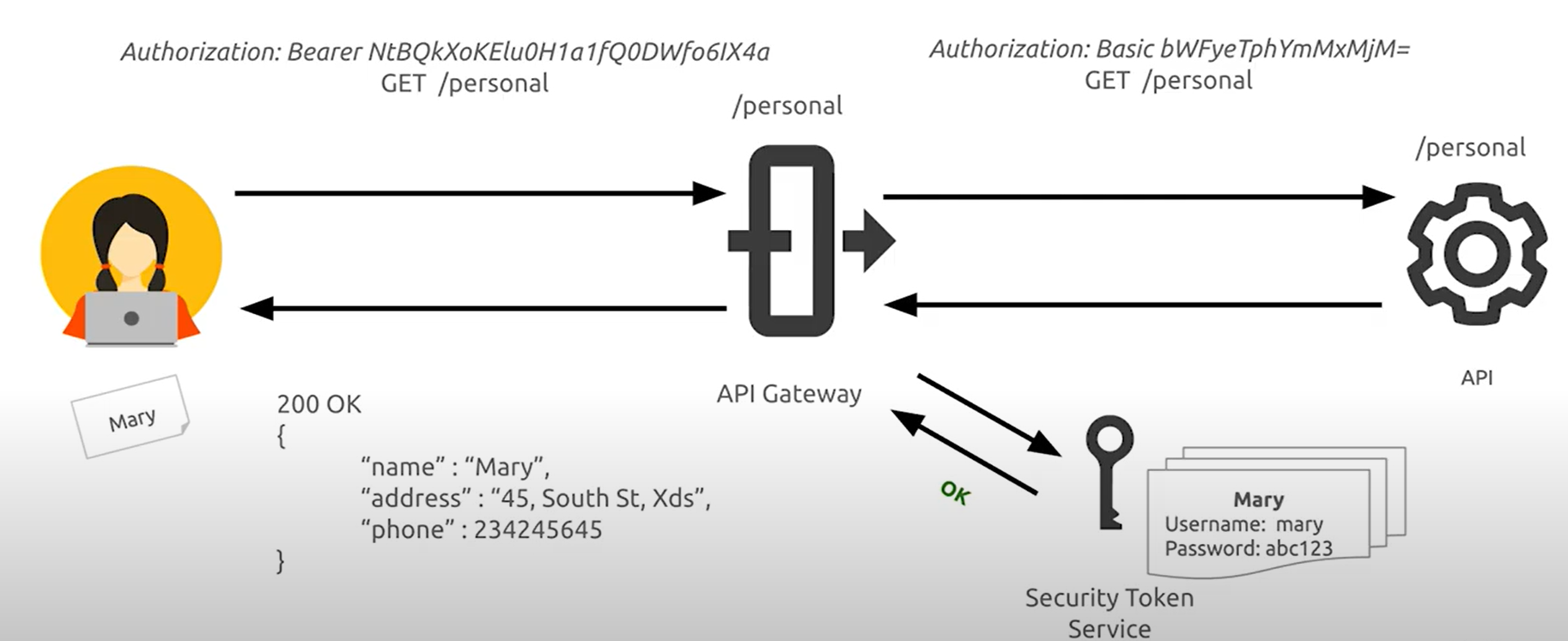
Kẻ tấn công bắt đầu quy trình khôi phục mật khẩu bằng cách đưa ra yêu cầu POST tới /api/system/verify-code và bằng cách cung cấp tên người dùng trong phần thân yêu cầu. Tiếp theo, một mã thông báo SMS có 6 chữ số sẽ được gửi đến điện thoại của nạn nhân. Vì API không triển khai chính sách giới hạn tốc độ, nên kẻ tấn công có thể kiểm tra tất cả các kết hợp có thể có bằng cách sử dụng tập lệnh đa luồng, dựa vào điểm cuối /api/system/verify-Code/{smsToken} để phát hiện ra mã thông báo phù hợp trong vòng vài phút..

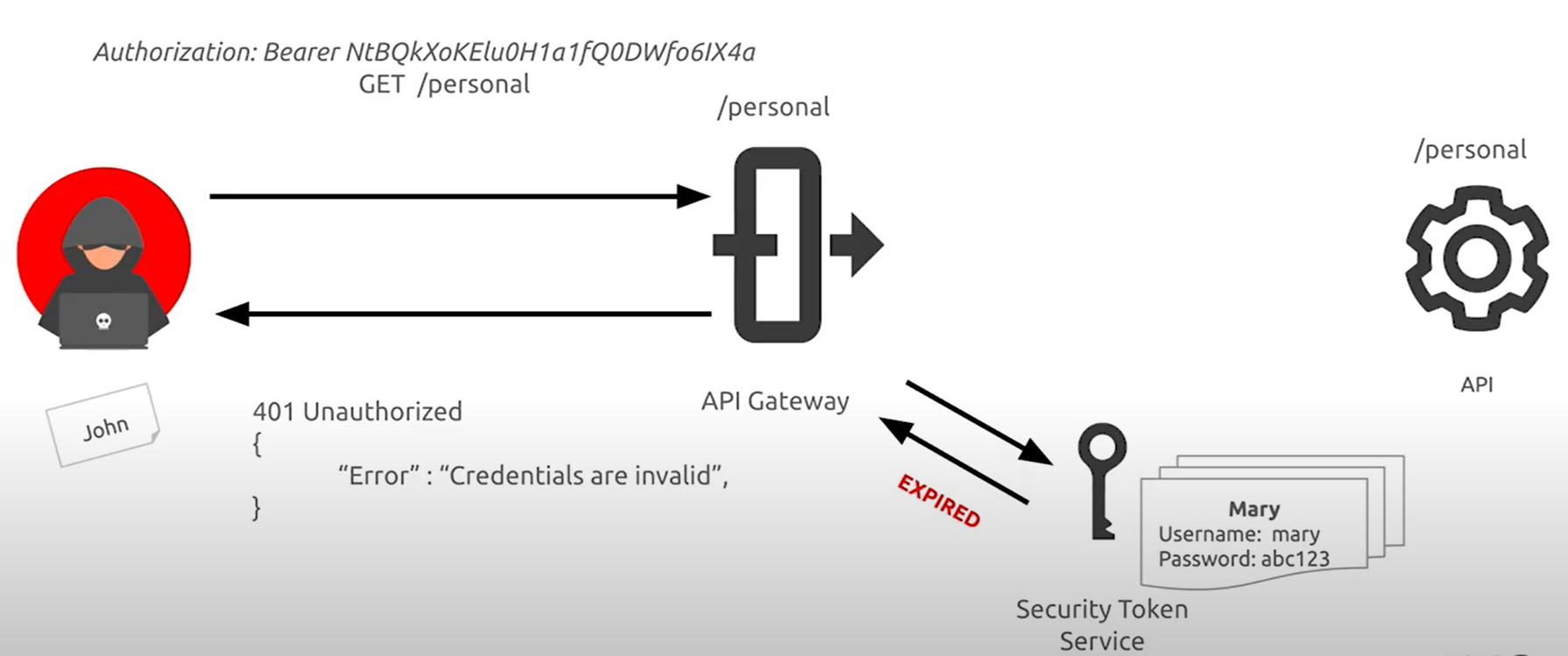




## **Làm thế nào để ngăn chặn**

* Đảm bảo rằng bạn biết tất cả các quy trình có thể để xác thực với API (di động / web / liên kết triển khai xác thực bằng một cú nhấp chuột / v.v.)
* Hỏi kỹ sư của bạn về luồng mà bạn đã bỏ lỡ.
* Đọc về cơ chế xác thực của bạn. Đảm bảo rằng bạn hiểu chúng được sử dụng như thế nào và những gì. OAuth không phải là xác thực và cũng không phải là API keys.
* Credential recovery/forget password endpoints phải được coi là login endpoints trong điều khoản của brute force, rate limiting, và lockout protections.
* Sử dụng [OWASP Authentication Cheatsheet](https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html).
* Nếu có thể, hãy triển khai xác thực đa yếu tố.
* Triển khai các cơ chế chống brute force để giảm thiểu Credential stuffing attack, Dictionary Attack và các cuộc tấn công brute force vào các endpoint xác thực của bạn. Cơ chế này phải chặt chẽ hơn cơ chế giới hạn tỷ lệ thông thường trên API của bạn.
* Thực hiện cơ chế khóa tài khoản / captcha để ngăn chặn hành vi brute force attack đối với những người dùng cụ thể. Thực hiện kiểm tra mật khẩu yếu.
* Không nên sử dụng các API keys để xác thực người dùng mà để xác thực [client app/ project authentication](https://cloud.google.com/endpoints/docs/openapi/when-why-api-key).





# **API3:2019 – Excessive data exposure**

Mong muốn triển khai chung, các nhà phát triển có xu hướng hiển thị tất cả các thuộc tính đối tượng mà không xem xét độ nhạy cá nhân của chúng, dựa vào khách hàng để thực hiện lọc dữ liệu trước khi hiển thị cho người dùng.

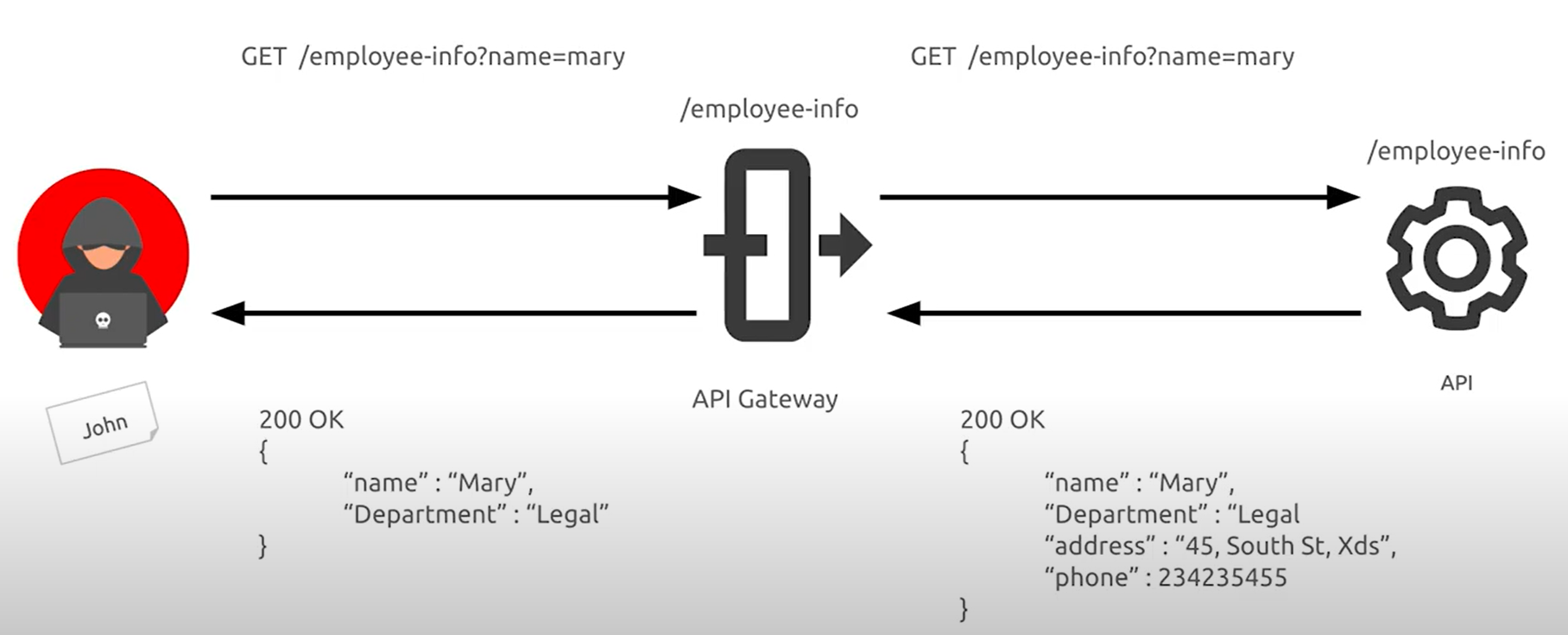
## **Các tình huống tấn công ví dụ**

Người dùng thực hiện gọi đến api để xem thông tin nhân viên GET /employee-info?name=mary và hệ thống trả về kết quả bao gồm những thông tin nhạy cảm khác như thông tin số điện thoại, địa chỉ nhà riêng, số chứng minh thư nhân dân,v.v…



## **Làm thế nào để ngăn chặn**

* Không bao giờ dựa vào phía khách hàng để lọc dữ liệu nhạy cảm.
* Xem lại các response từ API để đảm bảo rằng chúng chỉ chứa dữ liệu hợp pháp.
* Tránh sử dụng các phương thức chung chung như to\_json () và to\_string (). Thay vào đó, hãy chọn những thuộc tính cụ thể mà bạn thực sự muốn trả lại
* Phân loại thông tin nhạy cảm và nhận dạng cá nhân (PII) mà ứng dụng của bạn lưu trữ và làm việc, xem xét tất cả các lệnh gọi API trả về thông tin đó để xem liệu các response này có gây ra vấn đề bảo mật hay không.
* Triển khai cơ chế xác thực response như một lớp bảo mật bổ sung. Là một phần của cơ chế này, xác định và thực thi dữ liệu được trả về bởi tất cả các phương thức API, bao gồm cả lỗi.



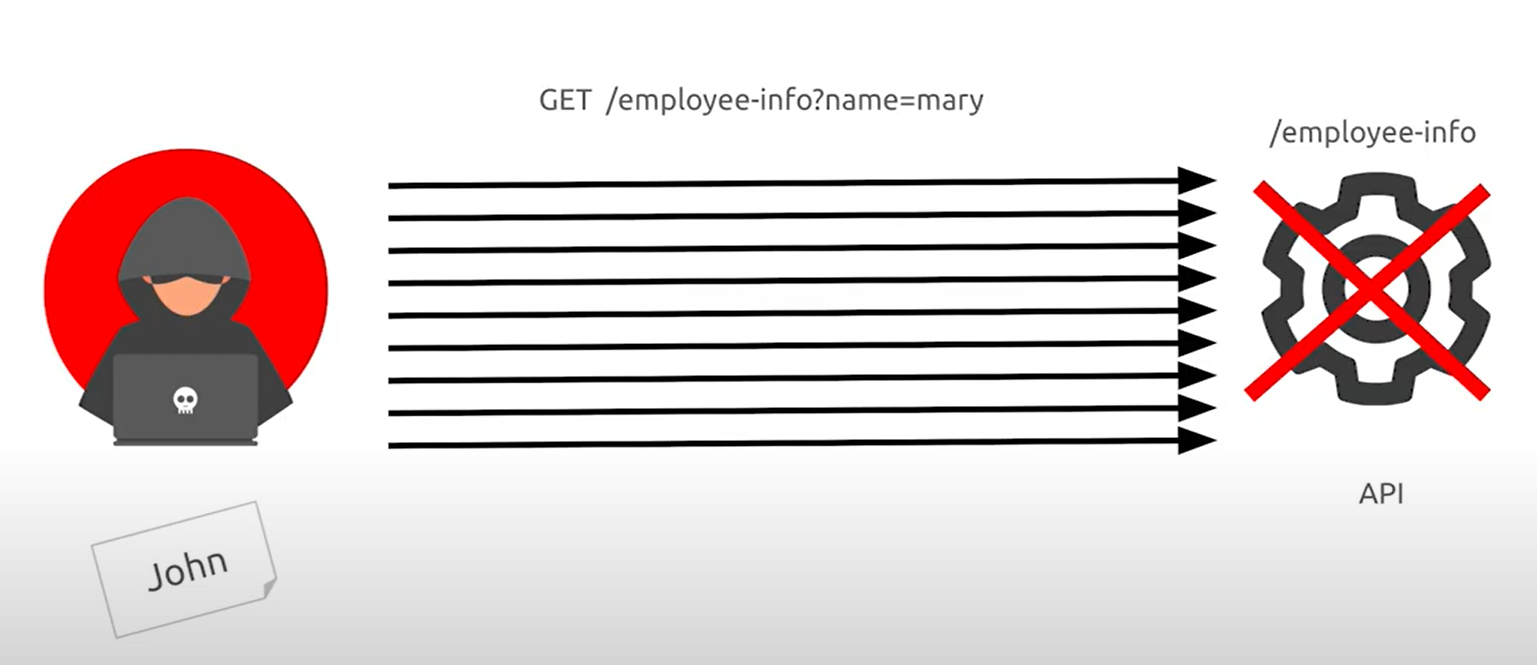
# **API4:2019 – Lack of resources and rate limiting**

Thông thường, các API không áp đặt bất kỳ hạn chế nào về kích thước hoặc số lượng tài nguyên có thể được khách hàng / người dùng request. Điều này không chỉ có thể ảnh hưởng đến hiệu suất máy chủ API, dẫn đến Từ chối dịch vụ (DoS), mà còn mở ra cánh cửa cho các lỗi xác thực như brute force.

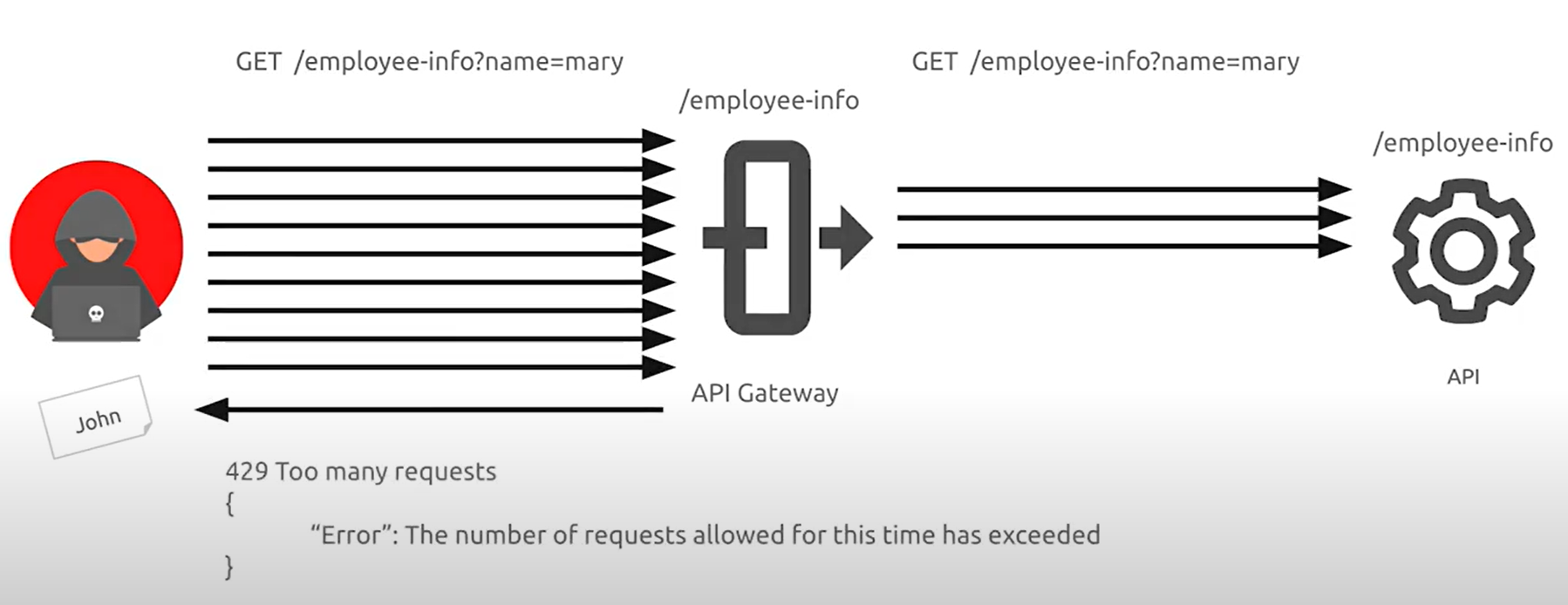
## **Tình huống tấn công ví dụ**

### **Tình huống 1**

Attacker gửi nhiều threads request dẫn đến tấn công từ chối dịch vụ (DoS).

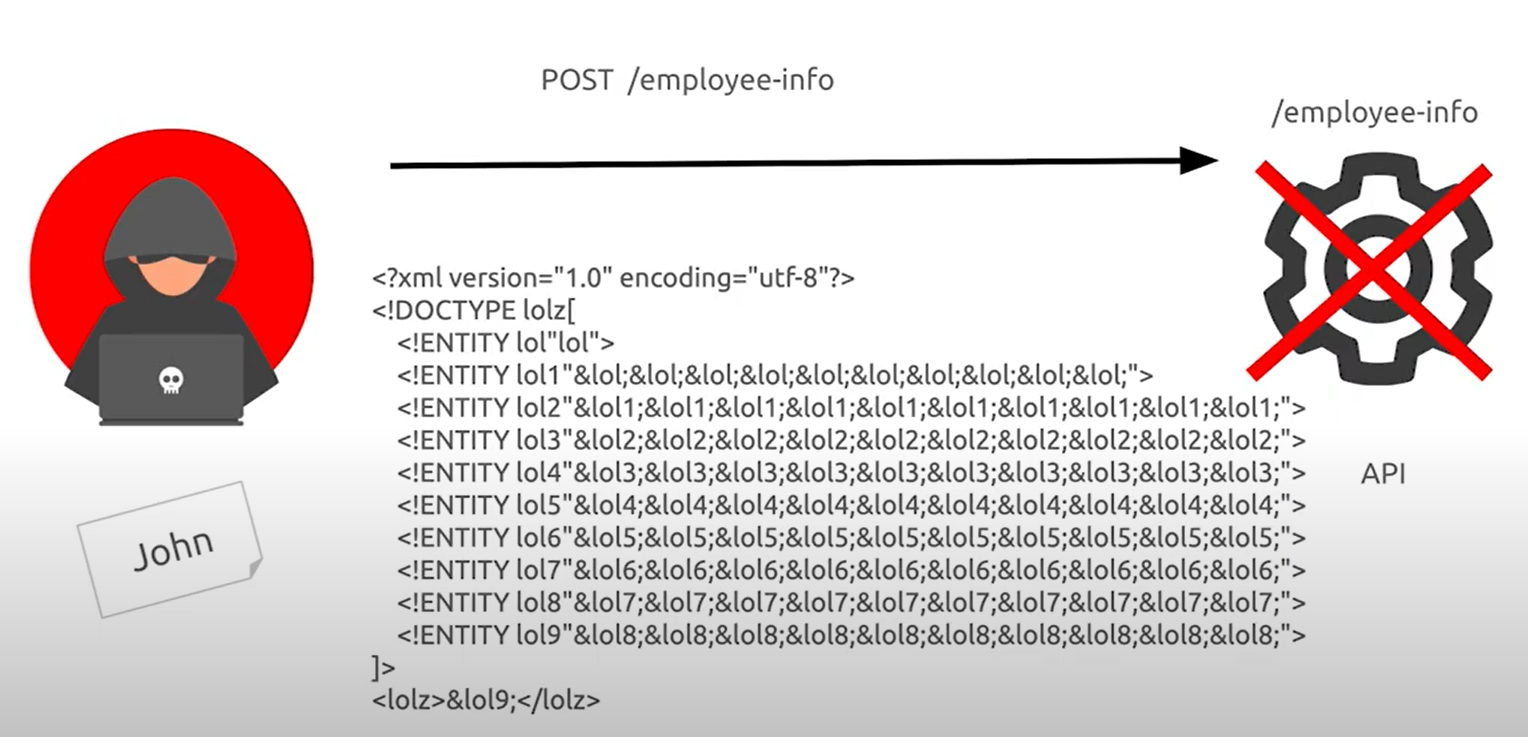


**Khắc phục:**

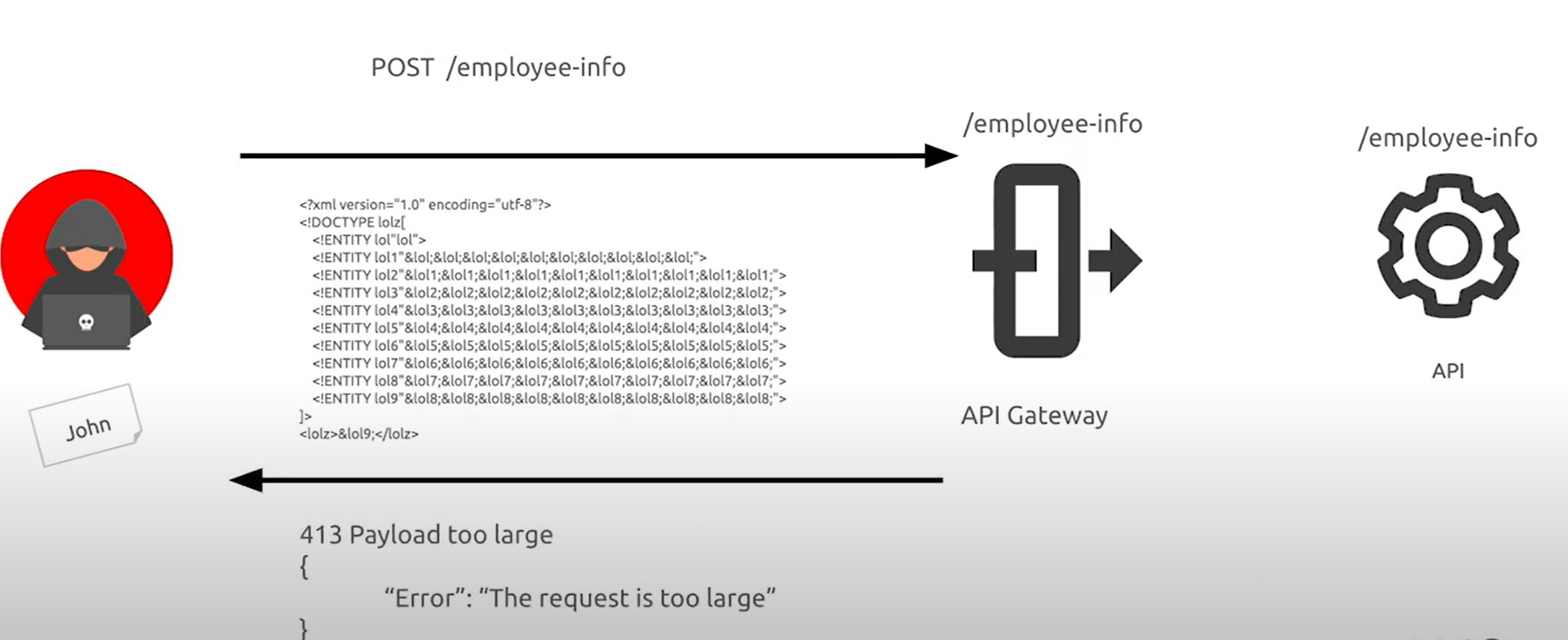


### **Tình huống 2**

Attacker gửi request lên với data quá lớn dẫn đến hệ thống xử lý kém hiệu suất dẫn đến tấn công từ chối dịch vụ (DoS).



**Khắc phục:**



## **Làm thế nào để ngăn chặn**

* Docker giúp dễ dàng giới hạn [memory](https://docs.docker.com/config/containers/resource_constraints/#memory), [CPU](https://docs.docker.com/config/containers/resource_constraints/#cpu), [number of restarts](https://docs.docker.com/engine/reference/commandline/run/#restart-policies---restart), [file descriptors, and processes](https://docs.docker.com/engine/reference/commandline/run/#set-ulimits-in-container---ulimit).
* Thực hiện giới hạn về tần suất khách hàng có thể gọi API trong một khung thời gian xác định.
* Thông báo cho khách hàng khi vượt quá giới hạn bằng cách cung cấp số giới hạn và thời gian đặt lại giới hạn.
* Thêm xác thực phía máy chủ thích hợp cho chuỗi truy vấn và tham số nội dung yêu cầu, cụ thể là tham số kiểm soát số lượng bản ghi được trả về trong response.
* Xác định và thực thi kích thước tối đa của dữ liệu trên tất cả các tham số và trọng tải đến, chẳng hạn như độ dài tối đa cho chuỗi và số phần tử tối đa trong mảng.

# **API5:2019 – Broken function level authorization**

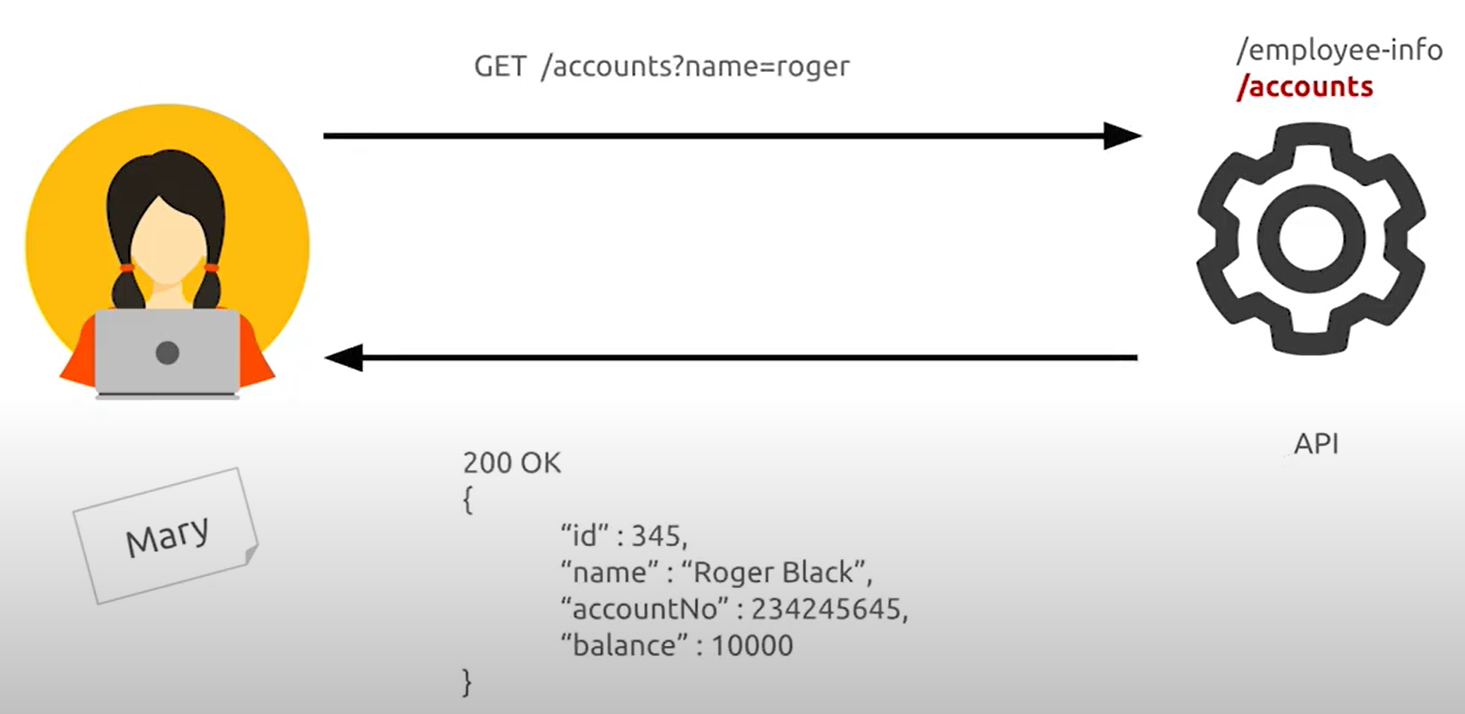
Các chính sách kiểm soát truy cập phức tạp với các phân cấp, nhóm và vai trò khác nhau và sự tách biệt không rõ ràng giữa chức năng quản trị và chức năng thường, có xu hướng dẫn đến sai sót về ủy quyền. Bằng cách khai thác những vấn đề này, những kẻ tấn công có quyền truy cập vào tài nguyên và / hoặc chức năng quản trị của người dùng khác.

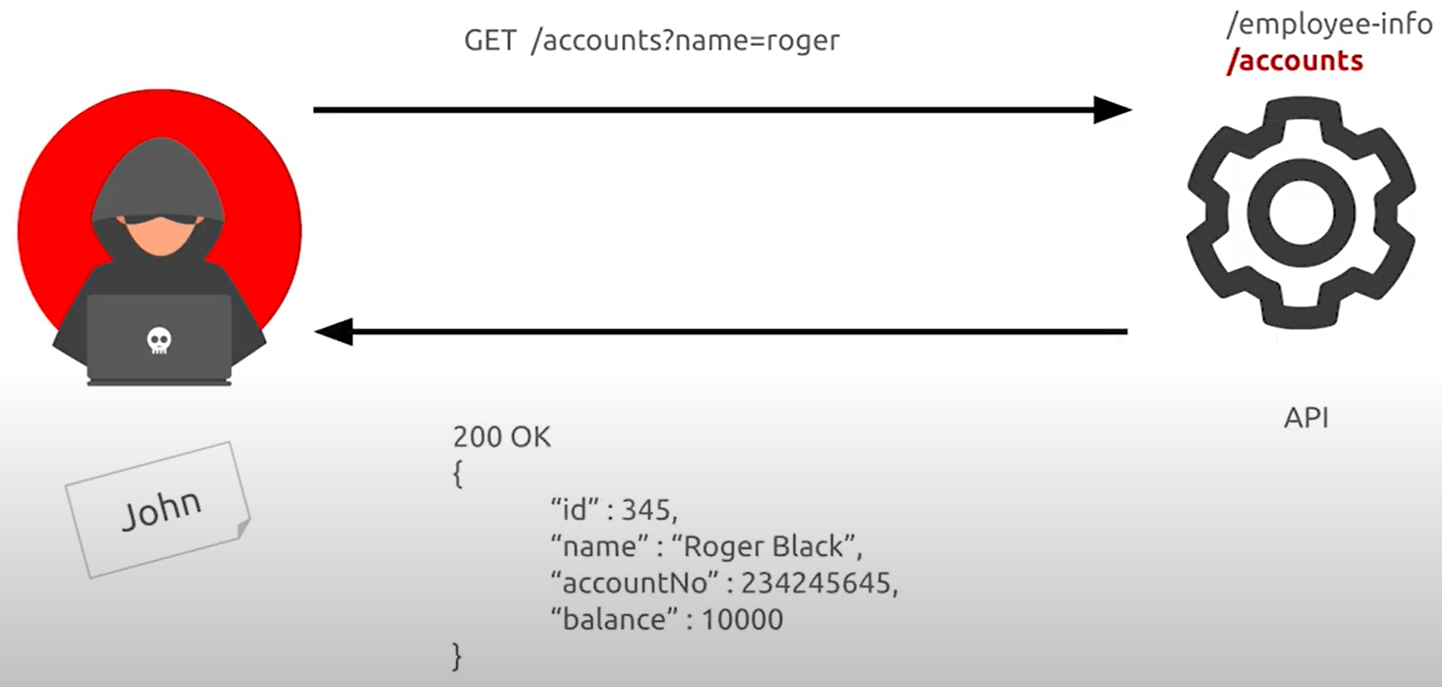
## **Tình huống tấn công ví dụ**

API xem thông tin account chỉ có group Finance mới có thể truy vấn.

Nhân viên Mary ở group Finance truy vấn đến api GET /accounts?name=roger để xem thông tin nhân viên tên Roger Black.

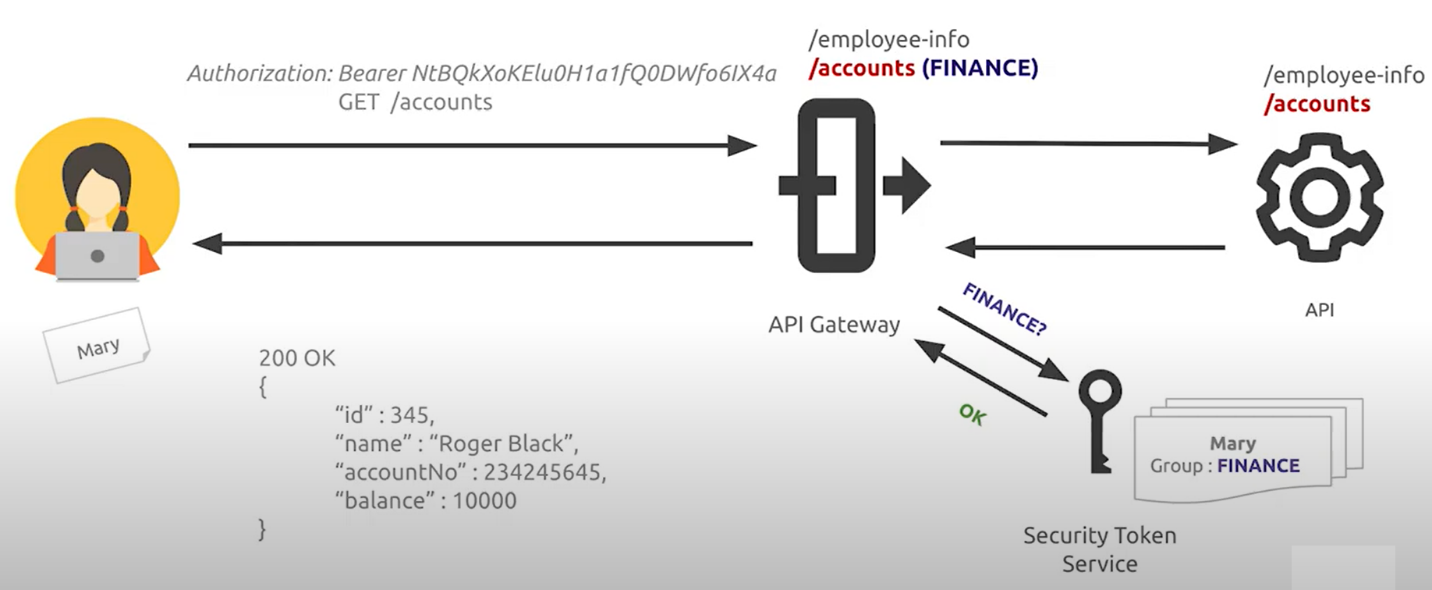
Nhân viên John ở group IT truy vấn đến api GET /accounts?name=roger để xem thông tin nhân viên tên Roger Black và api vẫn response về thông tin nhân viên.

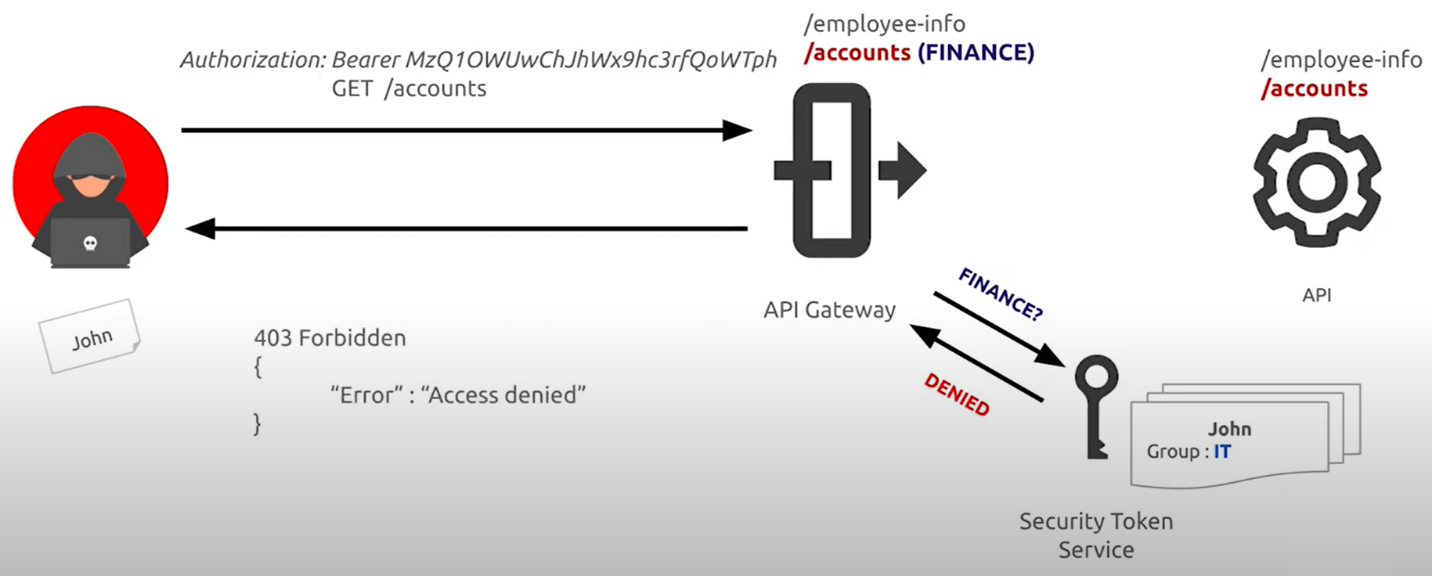




## **Làm thế nào để ngăn chặn**

* Các cơ chế thực thi nên từ chối tất cả quyền truy cập theo mặc định, yêu cầu cấp quyền rõ ràng cho các vai trò cụ thể để truy cập vào mọi chức năng.
* Xem xét các endpoint API của bạn để chống lại các lỗi phân quyền cấp chức năng, đồng thời ghi nhớ logic nghiệp vụ của ứng dụng và hệ thống phân cấp nhóm.
* Đảm bảo rằng tất cả administrative controllers của bạn kế thừa từ administrative abstract controller thực hiện kiểm tra ủy quyền dựa trên group/role.
* Đảm bảo rằng các chức năng quản trị bên trong controller thực hiện kiểm tra ủy quyền dựa trên group và role.





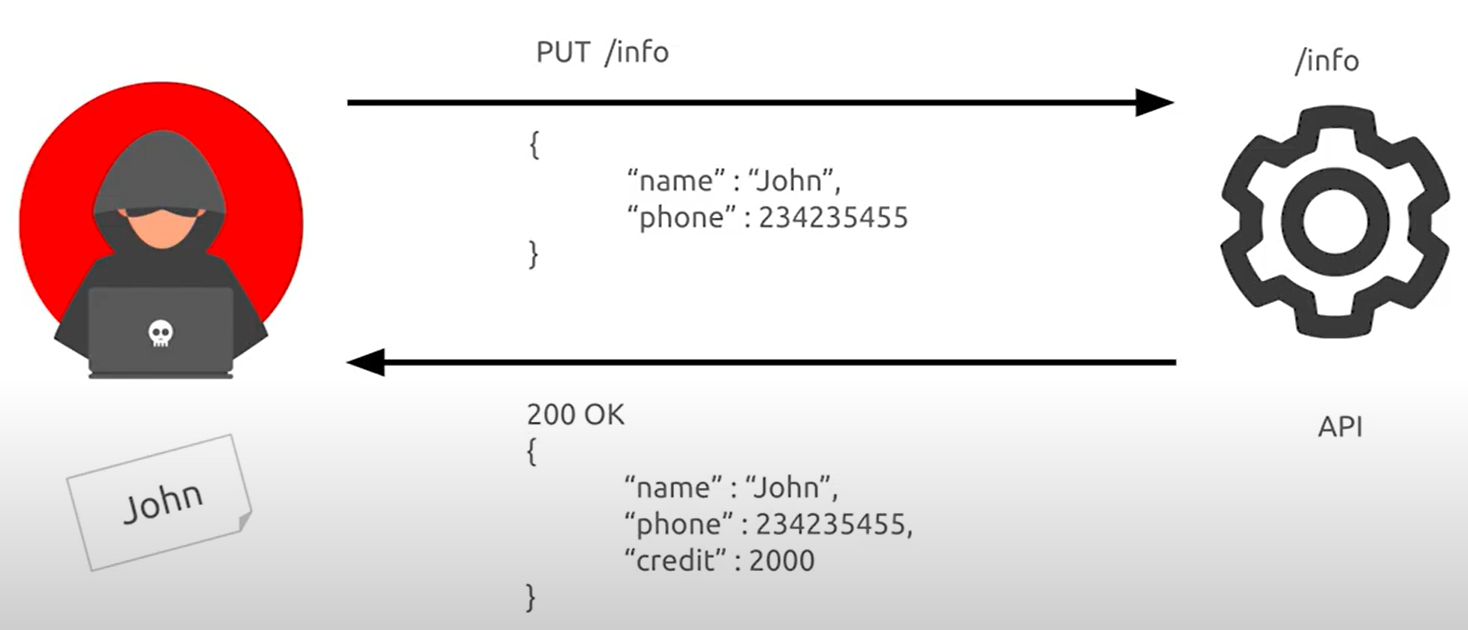
# **API6:2019 – Mass assignment**

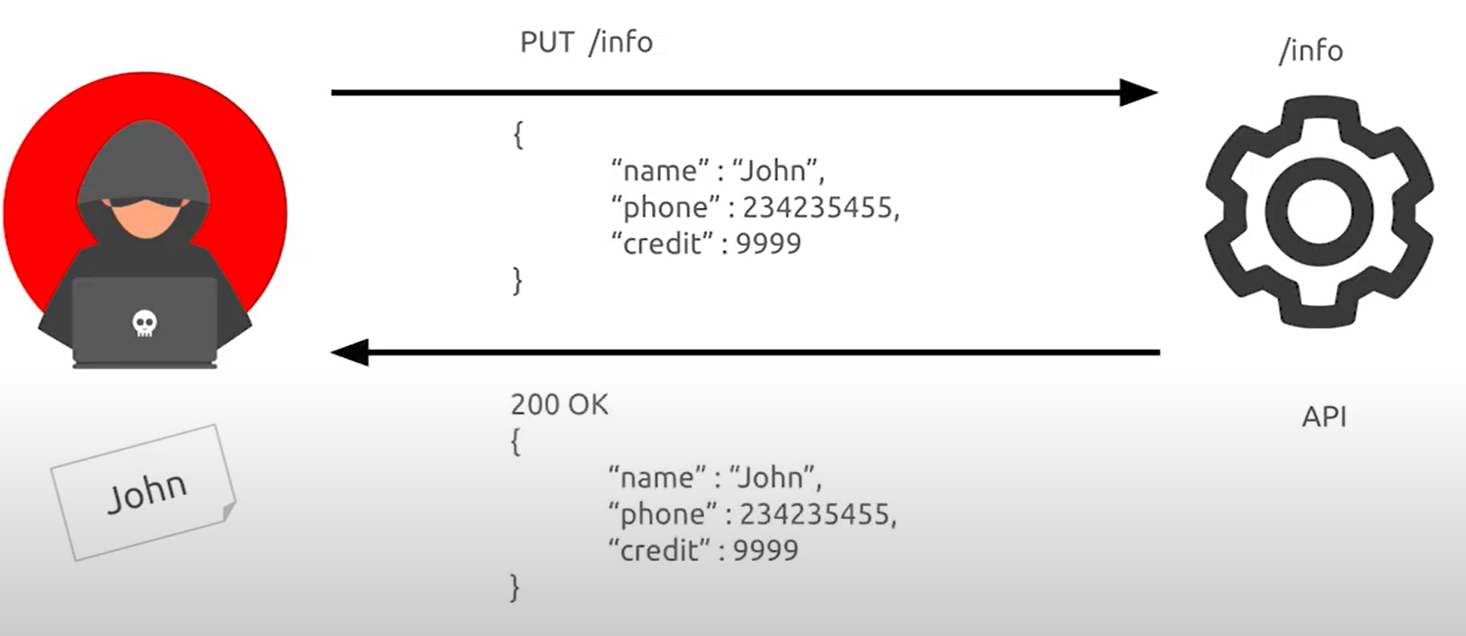
Việc ràng buộc dữ liệu do khách hàng cung cấp (ví dụ: JSON) với các models dữ liệu mà không có tính năng lọc thuộc tính thích hợp dựa trên danh sách cho phép, thường dẫn đến Mass Assignment. Việc đoán các thuộc tính đối tượng, khám phá các endpoints API khác, đọc tài liệu hoặc cung cấp các thuộc tính đối tượng bổ sung trong request payload, cho phép kẻ tấn công sửa đổi các thuộc tính đối tượng mà chúng không được phép.

## **Tình huống tấn công ví dụ**

Người dùng chỉ được phép update name và phone, api response trả về có thêm các thông tin nhạy cảm khác như credit.

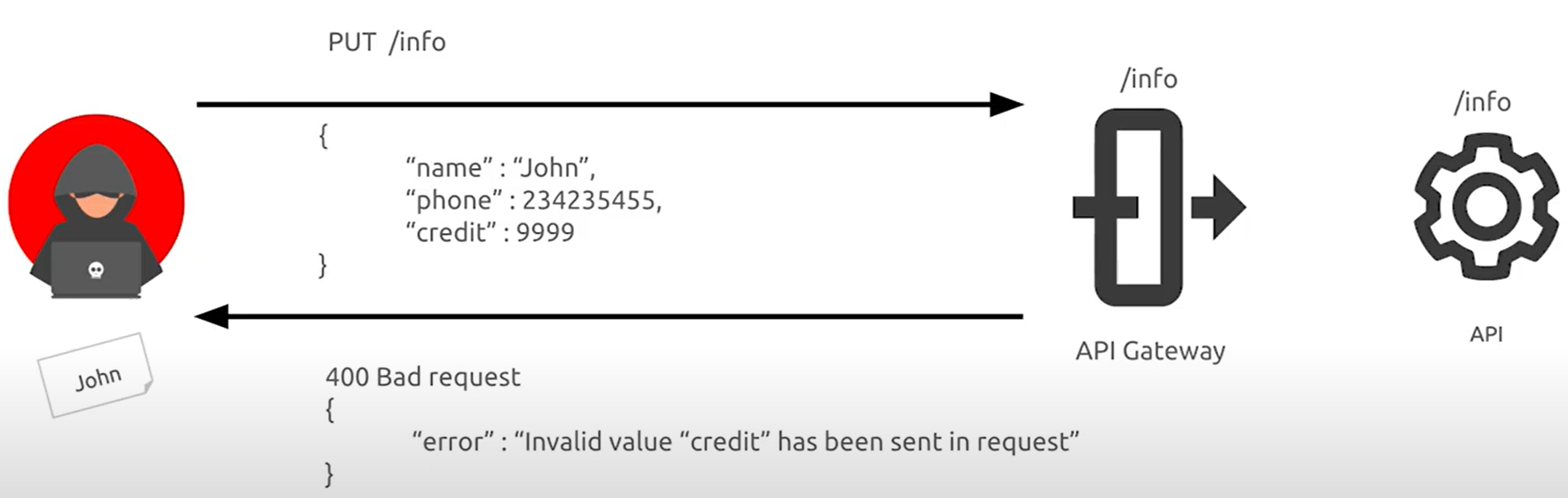
Người dùng thực hiện update cả endpoint credit, do api không có bộ filter lọc dữ liệu cho phép được sửa nên người dùng thực hiện update dữ liệu trái phép thành công.





## **Làm thế nào để ngăn chặn**

* Nếu có thể, hãy tránh sử dụng các hàm tự động liên kết đầu vào của khách hàng vào các biến mã hoặc đối tượng nội bộ..
* Tạo whitelist chỉ liệt kê các thuộc tính cần được cập nhật bởi client.
* Sử dụng các tính năng tích hợp để đưa các thuộc tính vào blacklist mà client không nên truy cập.
* Nếu có thể, hãy xác định rõ ràng và thực thi các enforce schemas cho các dữ liệu payload đầu vào.



# **API7:2019 – Security Misconfiguration**

Cấu hình bảo mật sai thường là kết quả của cấu hình mặc định không an toàn, cấu hình không đầy đủ hoặc đặc biệt, open cloud storage, HTTP headers cấu hình sai, các HTTP method không cần thiết, dính lỗi Cross-Origin resource sharing (CORS) và thông báo lỗi dài dòng chứa thông tin nhạy cảm.

## **Tình huống tấn công ví dụ**

### **Tình huống 1**

Kẻ tấn công tìm thấy tệp .bash\_history trong thư mục gốc của máy chủ, chứa các lệnh được nhóm DevOps sử dụng để truy cập API:

|  |
| --- |
| $ curl -X GET 'https://api.server/endpoint/' -H 'authorization: Basic Zm9vOmJhcg==' |

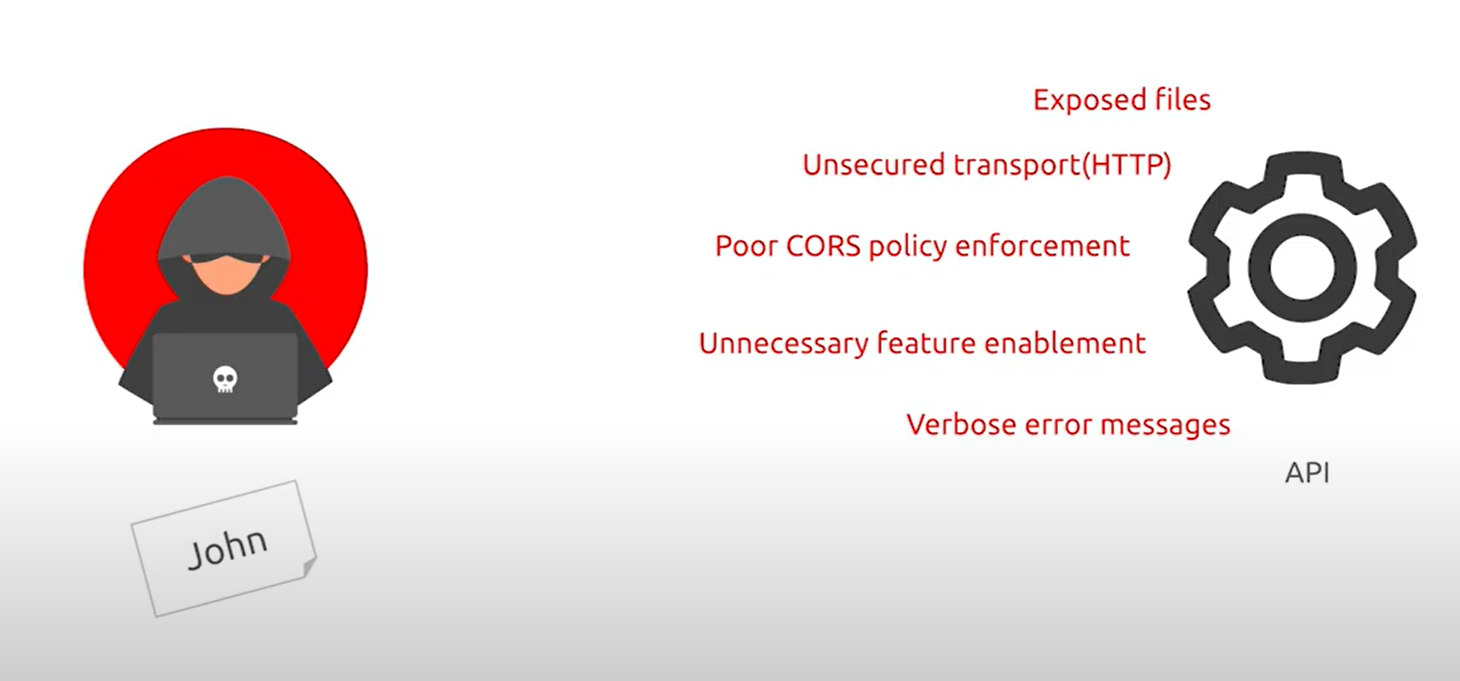
Kẻ tấn công cũng có thể tìm thấy các endpoint mới trên API chỉ được sử dụng bởi nhóm DevOps và không được ghi lại..

### **Tình huống 2**

Để nhắm một mục tiêu dịch vụ cụ thể, kẻ tấn công sử dụng một công cụ tìm kiếm phổ biến để tìm kiếm các máy chủ có thể truy cập trực tiếp từ Internet. Kẻ tấn công đã tìm thấy một máy chủ chạy hệ quản trị cơ sở dữ liệu phổ biến, đang lắng nghe trên cổng mặc định. Máy chủ đang sử dụng cấu hình mặc định, đã tắt xác thực theo mặc định và kẻ tấn công đã giành được quyền truy cập vào hàng triệu bản ghi với PII, personal preferences và authentication data.

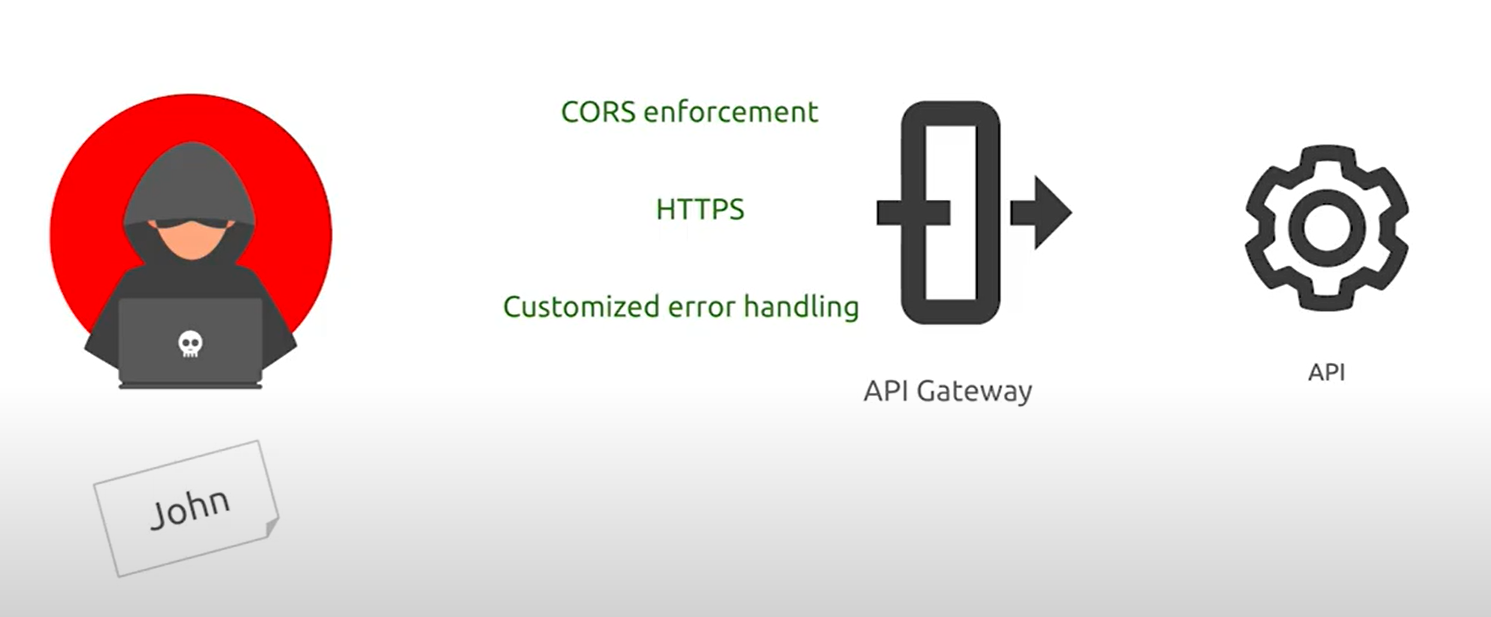
### **Tình huống 3**

Kiểm tra lưu lượng của một ứng dụng di động, kẻ tấn công phát hiện ra rằng không phải tất cả lưu lượng HTTP đều được thực hiện trên một giao thức an toàn (ví dụ: TLS). Kẻ tấn công nhận thấy điều này là đúng, cụ thể là đối với việc download các hình ảnh hồ sơ. Vì tương tác của người dùng là binary, mặc dù thực tế là lưu lượng truy cập API được thực hiện trên một giao thức an toàn, kẻ tấn công tìm thấy một mẫu về kích thước response API, mà hắn sử dụng để theo dõi tùy chọn của người dùng đối với nội dung được hiển thị (ví dụ: hình ảnh hồ sơ).



## **Làm thế nào để ngăn chặn**

* Để ngăn chặn các thông báo exception và thông tin có giá trị khác bị gửi lại cho những kẻ tấn công, nếu có, hãy xác định và thực thi tất cả API response payload schemas bao gồm các response error.
* Đảm bảo API chỉ có thể được truy cập bởi các method HTTP được chỉ định. Tất cả các method HTTP khác nên bị vô hiệu hóa (ví dụ: HEAD).
* Các API mong đợi được truy cập từ các ứng dụng khách dựa trên trình duyệt (ví dụ: giao diện người dùng WebApp) nên triển khai Cross-Origin Resource Sharing (CORS) policy thích hợp.

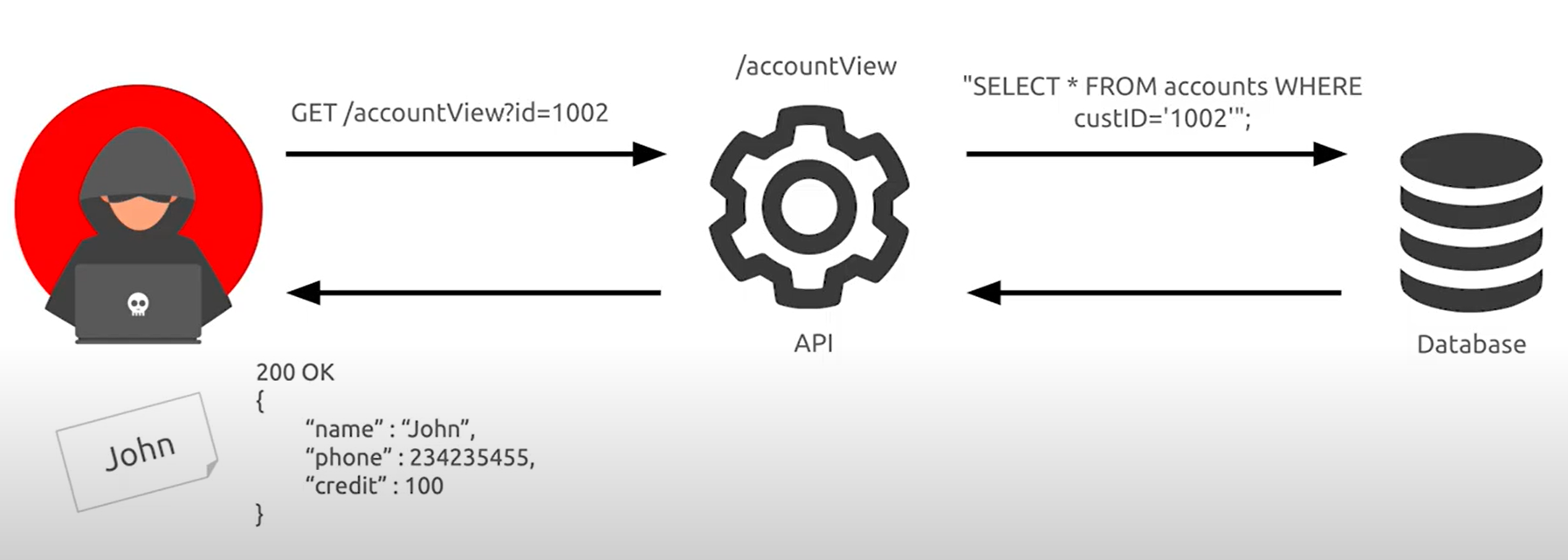


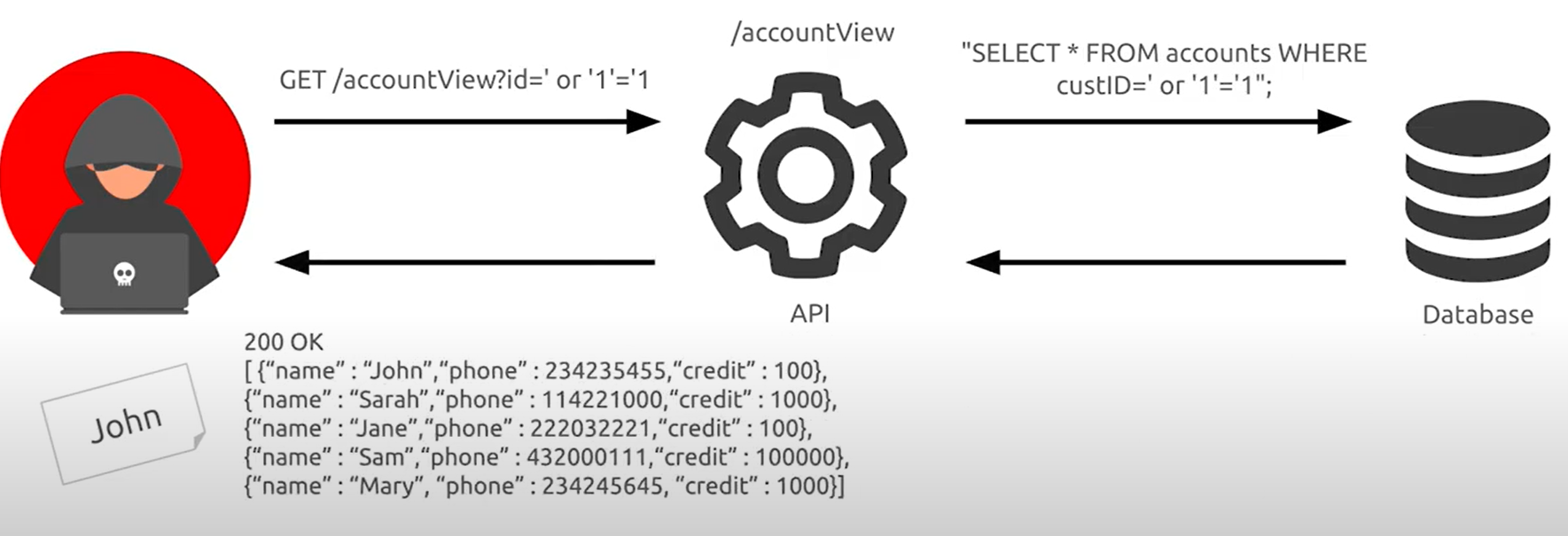
# **API8:2019 – Injection**

Các lỗi injection, chẳng hạn như SQL, NoSQL, Command Injection, v.v., xảy ra khi dữ liệu không đáng tin cậy được gửi đến trình thông dịch như một phần của lệnh hoặc truy vấn. Dữ liệu độc hại của kẻ tấn công có thể đánh lừa trình thông dịch thực hiện các lệnh không mong muốn hoặc truy cập dữ liệu mà không có sự cho phép thích hợp.

## **Tình huống tấn công ví dụ**

Attacker tấn công SQL injection và entry point id của API GET /accountView?id

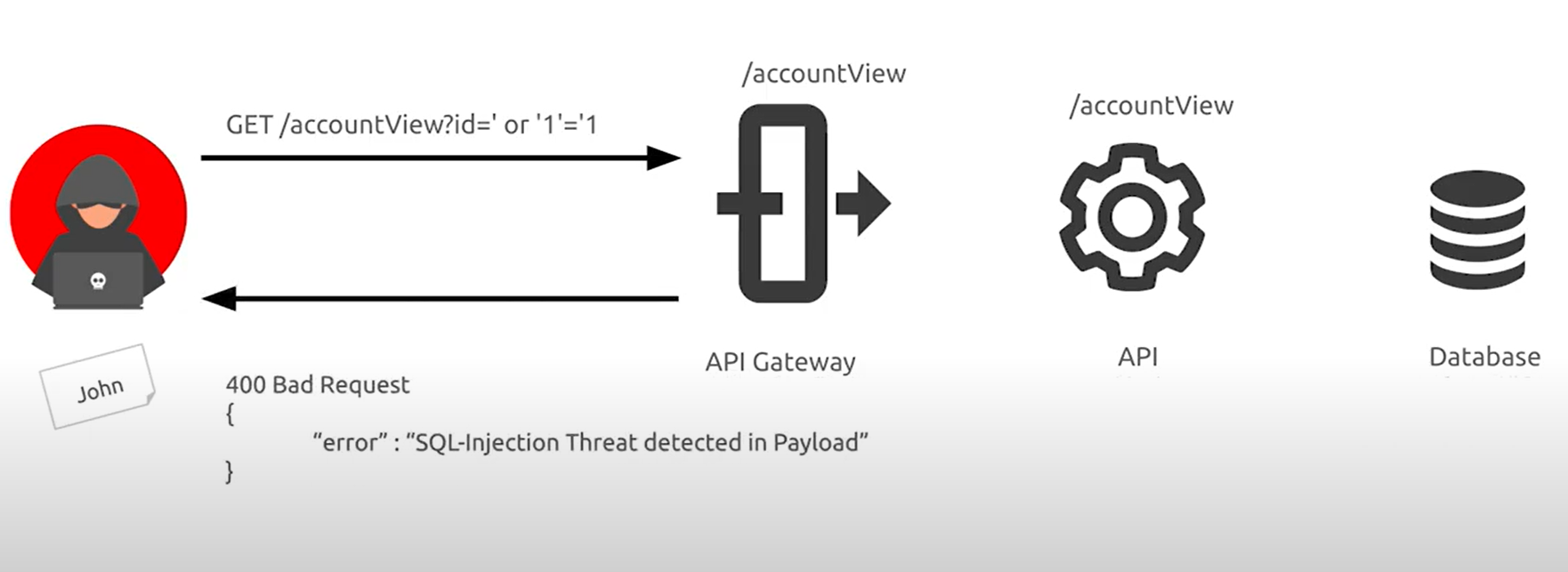




## **Làm thế nào để ngăn chặn**

Việc ngăn chặn việc tấn công injection yêu cầu giữ dữ liệu tách biệt với các lệnh và truy vấn SQL.

* Thực hiện xác thực dữ liệu bằng cách sử dụng một thư viện duy nhất, đáng tin cậy và được duy trì tích cực.
* Xác thực, lọc tất cả dữ liệu do khách hàng cung cấp hoặc dữ liệu khác đến từ các hệ thống tích hợp.
* Các ký tự đặc biệt phải được escaped bằng cách sử dụng syntax cụ thể cho trình thông dịch.
* Luôn giới hạn số lượng records trả về để tránh bị lộ hàng loạt trong trường hợp bị injection.
* Xác thực dữ liệu đến bằng cách sử dụng đủ bộ lọc để chỉ cho phép các giá trị hợp lệ cho mỗi tham số đầu vào.
* Xác định kiểu dữ liệu và các mẫu nghiêm ngặt phải được tuân thủ cho tất cả các tham số String.

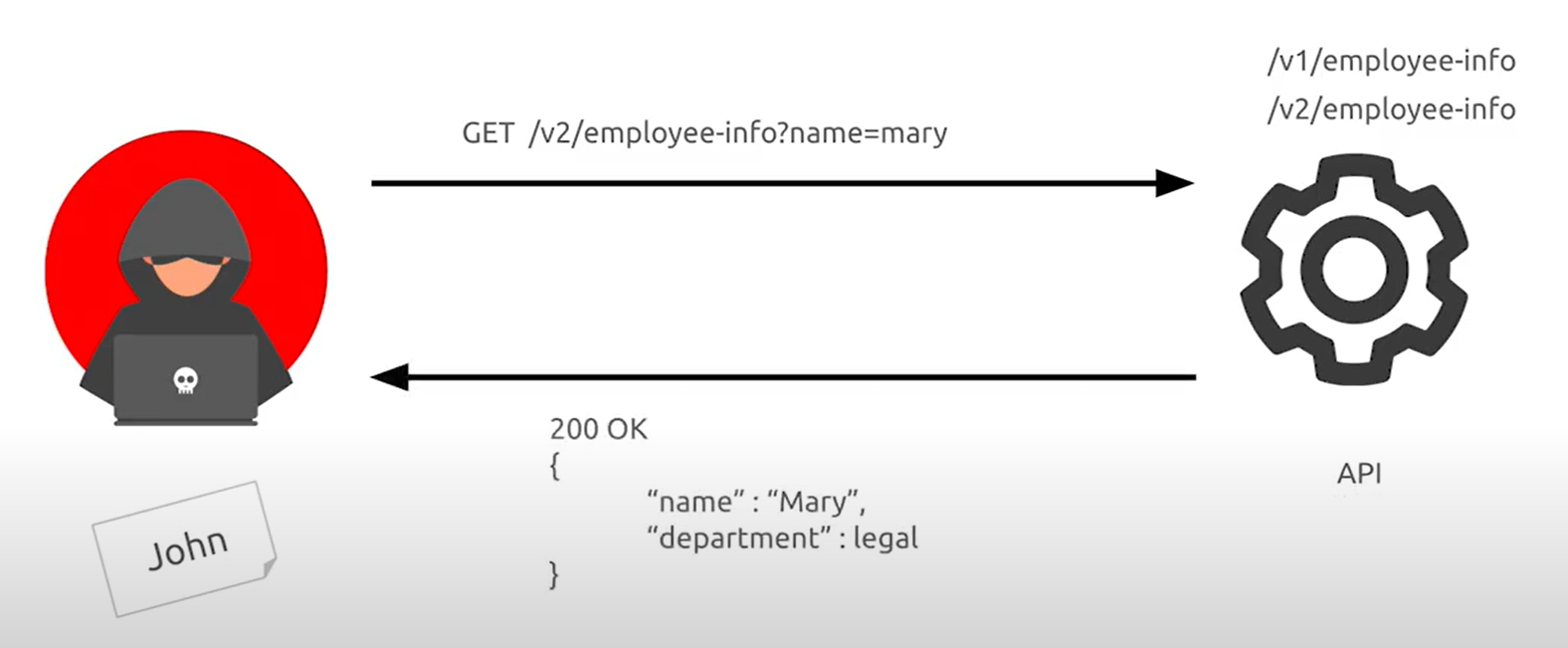


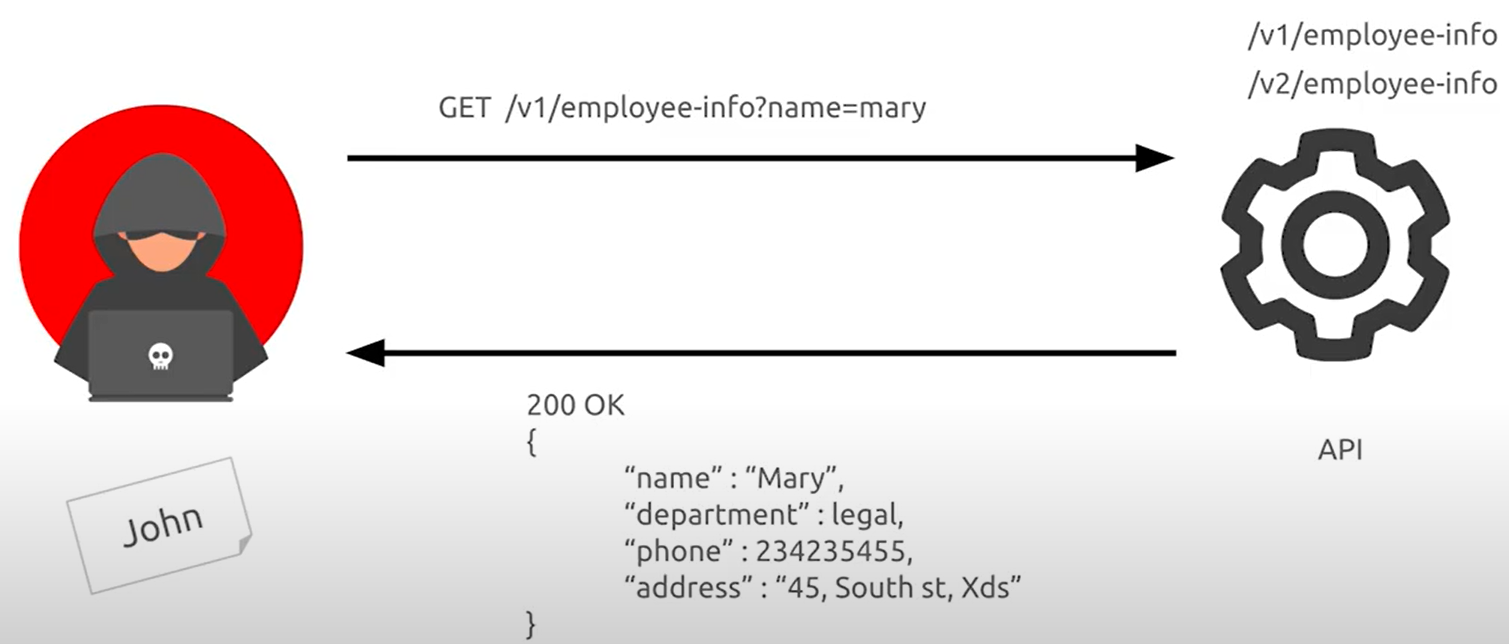
# **API9:2019 – Improper Assets Management**

Các API có xu hướng hiển thị nhiều endpoint hơn các web application truyền thống, làm document phù hợp và document quan trọng được update. Máy chủ phù hợp và phiên bản API đã triển khai cũng đóng một vai trò quan trọng để giảm thiểu các vấn đề như các phiên bản API không dùng nữa và các endpoint debug bị lộ.

## **Tình huống tấn công ví dụ**

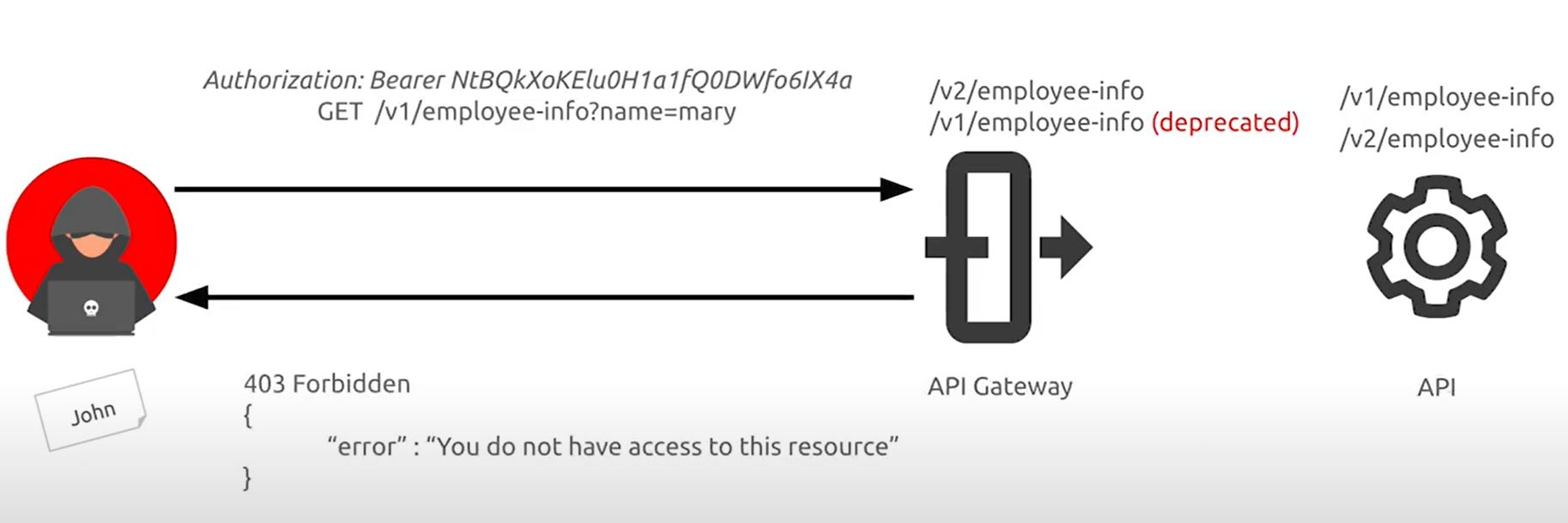
Sau khi thiết kế lại các ứng dụng của họ, một dịch vụ tìm kiếm cục bộ đã để lại một phiên bản API cũ (api.someservice.com/v1) đang chạy, không được bảo vệ và có quyền truy cập vào cơ sở dữ liệu người dùng. Trong khi nhắm mục tiêu vào một trong những ứng dụng được phát hành mới nhất, kẻ tấn công đã tìm thấy địa chỉ API (api.someservice.com/v2). Việc thay thế v2 bằng v1 trong URL đã cho phép kẻ tấn công truy cập vào API cũ, không được bảo vệ, làm lộ thông tin nhận dạng cá nhân (PII) của hơn 100 triệu người dùng.

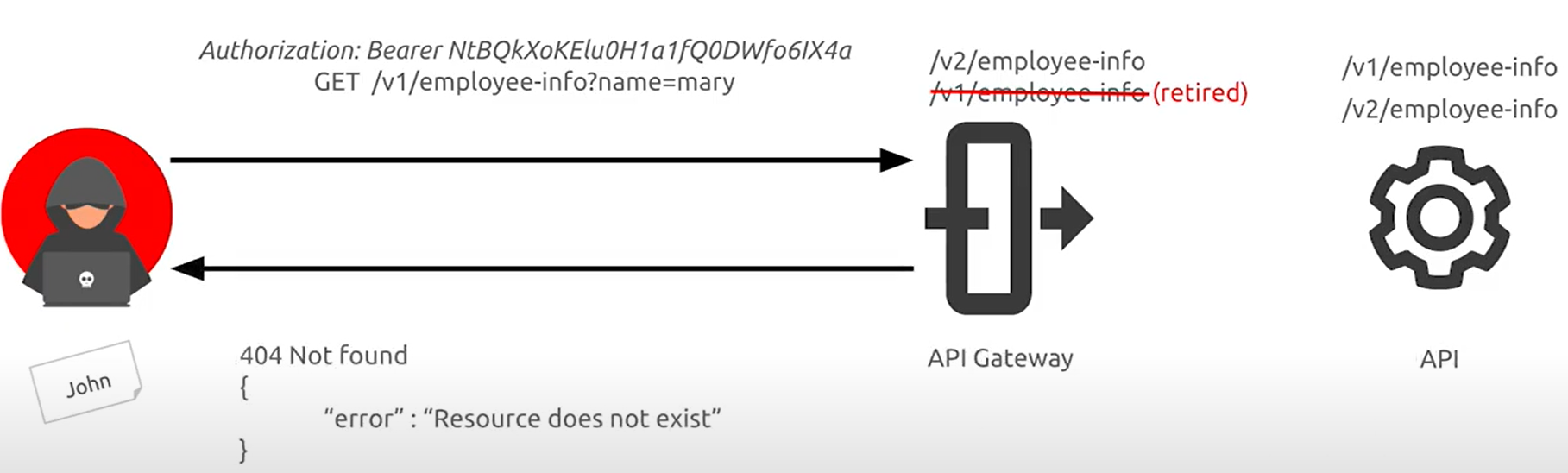




## **Làm thế nào để ngăn chặn**

* Kiểm kê tất cả các máy chủ API và ghi lại các khía cạnh quan trọng của từng máy chủ, tập trung vào môi trường API (ví dụ: production, staging, test, development), những người có quyền truy cập mạng vào máy chủ (ví dụ: công khai, nội bộ, đối tác) và phiên bản API.
* Ghi lại tất cả các khía cạnh API của bạn như xác thực, lỗi, chuyển hướng, giới hạn tốc độ, chính sách cross-origin resource sharing (CORS) và các endpoint, bao gồm các thông số, request và response của chúng.
* Tạo tài liệu tự động bằng cách áp dụng các tiêu chuẩn mở. Bao gồm tài liệu bản build trong đường dẫn CI/CD của bạn.
* Cung cấp tài liệu API cho những người được phép sử dụng API.
* Sử dụng các biện pháp bảo vệ bên ngoài như tường lửa bảo mật API cho tất cả các phiên bản tiếp xúc của API của bạn, không chỉ cho phiên bản sản xuất hiện tại.
* Tránh sử dụng dữ liệu product với các API triển khai không phải product. Nếu điều này là không thể tránh khỏi, các thiết bị đầu cuối này phải được xử lý bảo mật giống như product.
* Khi các phiên bản API mới hơn bao gồm các cải tiến bảo mật, hãy thực hiện phân tích rủi ro để đưa ra quyết định về các hành động giảm thiểu cần thiết cho phiên bản cũ hơn: ví dụ: liệu có thể backport các cải tiến mà không phá vỡ khả năng tương thích của API hay bạn cần sử dụng phiên bản cũ hơn nhanh chóng và buộc tất cả khách hàng chuyển sang phiên bản mới nhất.





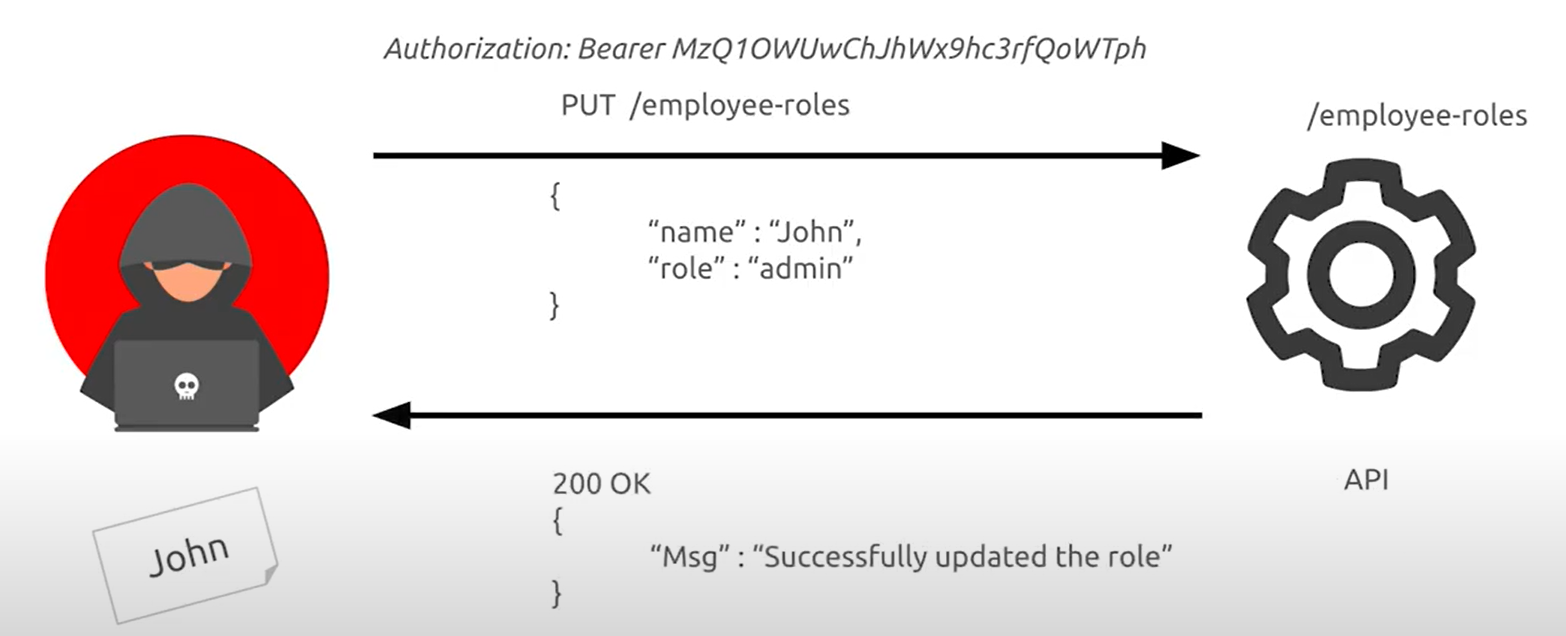
# **API10:2019 – Insufficient Logging & Monitoring**

Ghi nhật ký và giám sát không đầy đủ, cùng với việc tích hợp thiếu hoặc khắc phục sự cố không hiệu quả, cho phép attacker tấn công hệ thống sâu hơn nữa, duy trì tấn công lâu dài, trích xuất hoặc phá hủy dữ liệu. Hầu hết các nghiên cứu về tấn công cho thấy thời gian để phát hiện tấn công là hơn 200 ngày, thường được phát hiện bởi các chuyên gia bên ngoài nhóm lập trình hơn là các quy trình hoặc giám sát nội bộ.

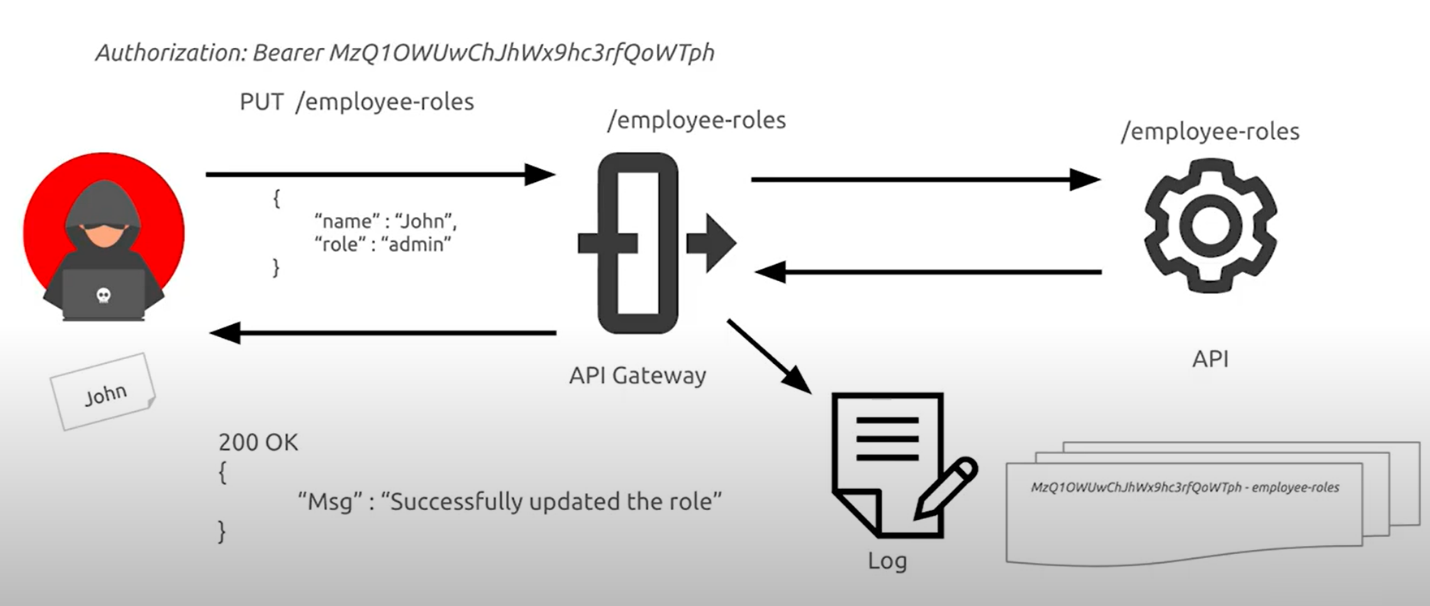
## **Tình huống tấn công ví dụ**

### **Tình huống 1**

Các key truy cập của một API quản trị đã bị rò rỉ trên một kho lưu trữ công khai. Chủ sở hữu kho lưu trữ đã được thông báo qua email về khả năng bị rò rỉ, nhưng đã mất hơn 48 giờ để xử lý sự cố và việc lộ key truy cập có thể đã cho phép truy cập vào dữ liệu nhạy cảm. Do ghi log không đầy đủ, công ty không thể đánh giá dữ liệu nào đã được truy cập bởi các phần tử độc hại.

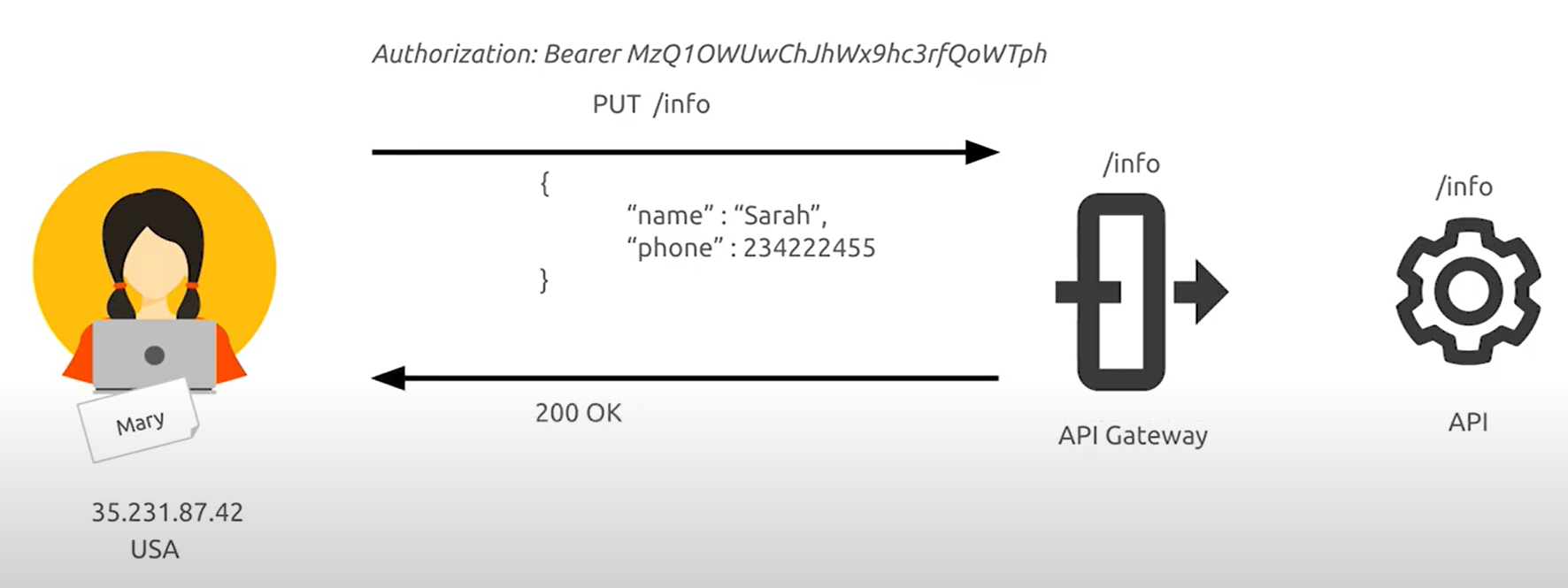


**Giải pháp:**

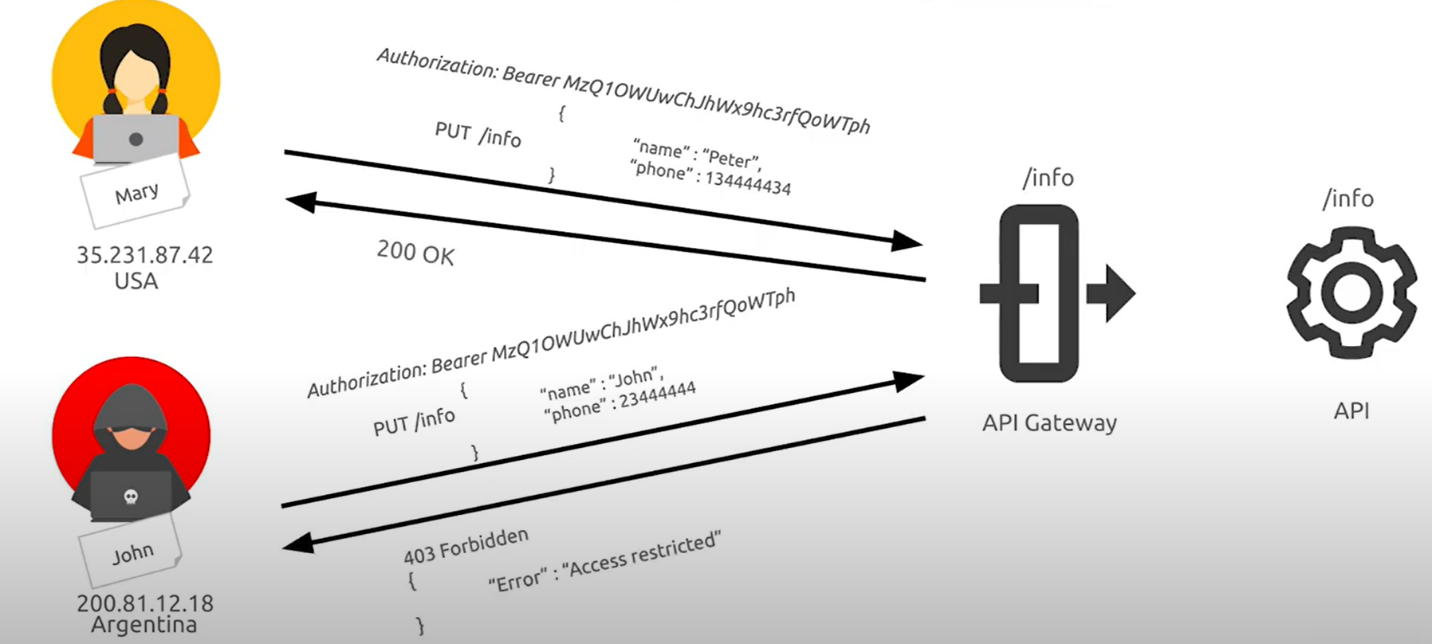


### **Tình huống 2**

Một nền tảng chia sẻ video đã bị tấn công bởi một cuộc tấn công credential stuffing "quy mô lớn". Mặc dù đăng nhập không thành công được ghi lại, không có cảnh báo nào được kích hoạt trong khoảng thời gian của cuộc tấn công. Để phản ứng với các khiếu nại của người dùng, log API đã được phân tích và phát hiện cuộc tấn công. Công ty đã phải thông báo rộng rãi yêu cầu người dùng đặt lại mật khẩu và báo cáo sự việc cho.



**Giải pháp:**



## **Làm thế nào để ngăn chặn**

* Ghi lại tất cả các lần xác thực không thành công, quyền truy cập bị từ chối và lỗi xác thực đầu vào.
* Log phải được viết bằng định dạng phù hợp với giải pháp quản lý log và phải bao gồm đủ chi tiết để xác định hành vi độc hại.
* Log phải được xử lý như dữ liệu nhạy cảm và tính toàn vẹn của chúng phải được đảm bảo khi lưu trữ và di chuyển.
* Cấu hình hệ thống giám sát để giám sát liên tục cơ sở hạ tầng, mạng và hoạt động của API.
* Sử dụng hệ thống Quản lý sự kiện và Thông tin Bảo mật (Security Information and Event Management - SIEM) để tổng hợp và quản lý log từ tất cả các thành phần của API stack và máy chủ.
* Cấu hình các trang tổng quan và cảnh báo tùy chỉnh, cho phép các hoạt động đáng ngờ được phát hiện và phản hồi sớm hơn.