ALL YOUR APIS ARE MINE - SECURE THEM NOW

Andreas Falk
@andifalk

About Me

Andreas Falk

NOVATEC

Managing Consultant / Novatec Consulting

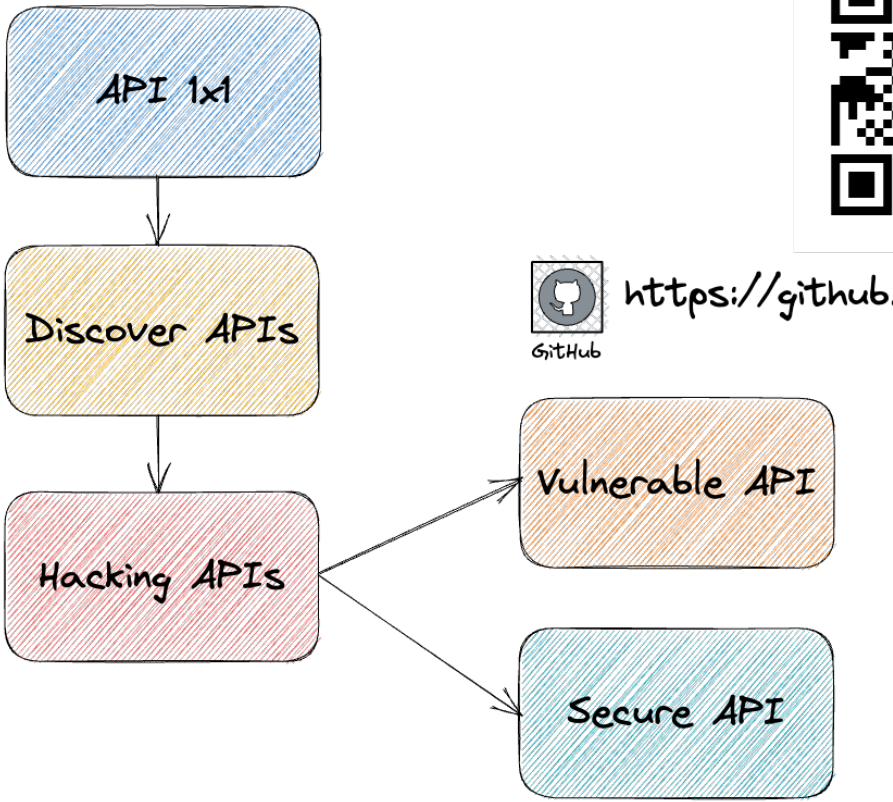
andreas.falk@novatec-gmbh.de

@andifalk

OWASP Open Web Application Security Project

OpenID

Hitchhikers Guide for this Talk



GitHub https://github.com/andifalk/api-security

Spring Boot
Spring Security
Rest API
GraphQL
Spring Authz Server
JWT

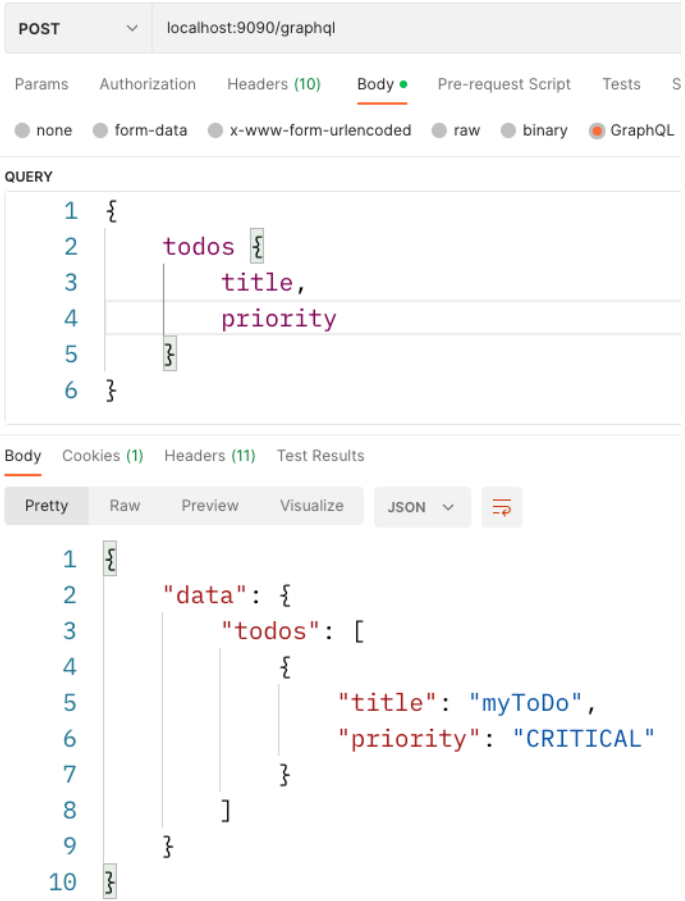
APIs 1x1

Web APIs: REST (Representational State Transfer)

- Principles
  - Resources
  - Representation of Resources
  - HTTP Verbs
  - Hypermedia Controls
- Documentation of REST APIs
  - OpenAPI & Swagger UI
  - RESTful API Modeling Language (RAML)
  - Spring REST Docs & AsciiDoctor

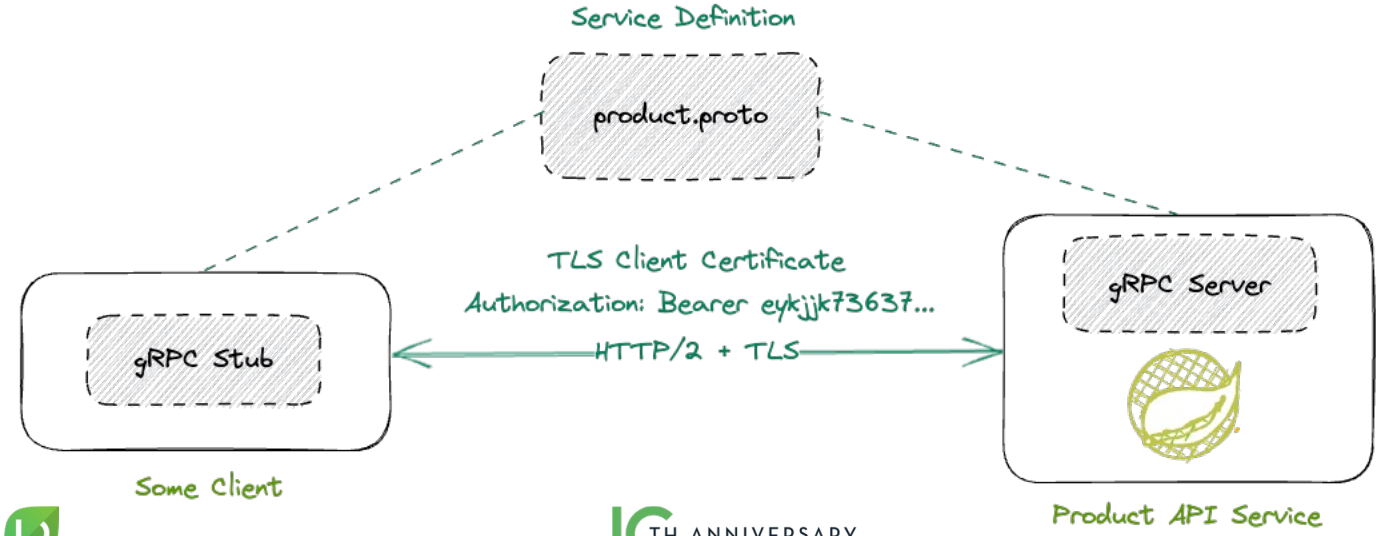
Web APIs: GraphQL

- Query language for APIs
- Runtime for fulfilling those queries
- Allows clients to define the structure of the data
- Standard specified by https://spec.graphql.org
- Spring GraphQL
  https://spring.io/projects/spring-graphql



Web APIs: gRPC

- High performance, open-source Remote Procedure Calls (RPC) framework (initially created by Google)
- Uses Protocol Buffers to encode data



Discovering & Analyzing APIs

API Discovery: Visibility & Protection of APIs

Public APIs:
No restrictions on access, requires an authentication key or token

Partner APIs:
Require specific permissions or licenses to access

Private APIs:
Used internally only, no public access

API Discovery & Analysis

- Search Engines & Tools
  - Google (Dorks)
  - Shodan
  - Programmable Web
  - Postman Explorer
  - OWASP Amass
- Analyse Exposed Information
  - GitHub Repositories
  - API Documentation
    - OpenAPI / Swagger
    - GraphiQL

API Discovery & Analysis

- OpenAPI /Swagger (REST / gRPC)
  - *springdoc.api-docs.enabled=false*
  - *springdoc.swagger-ui.enabled=false*
  - Force Authentication/Authorization for OpenAPI endpoints
- GraphQL
  - *spring.graphql.graphiql.enabled=false*
  - *spring.graphql.schema.printer.enabled=false*
  - Force Authentication/Authorization for */graphql* and */graphiql* endpoints

Hacking APIs

Attacking APIs:
A Web API from a Security Perspective

METHOD

GET, POST, PUT, PATCH, DELETE

PROTOCOL & HOSTNAME

http(s)://test.com

PATH

api/v2/users/{id}
/graphql

HEADERS

Authorization: Bearer eyljkhjghjgkjk…

PAYLOAD (DATA)

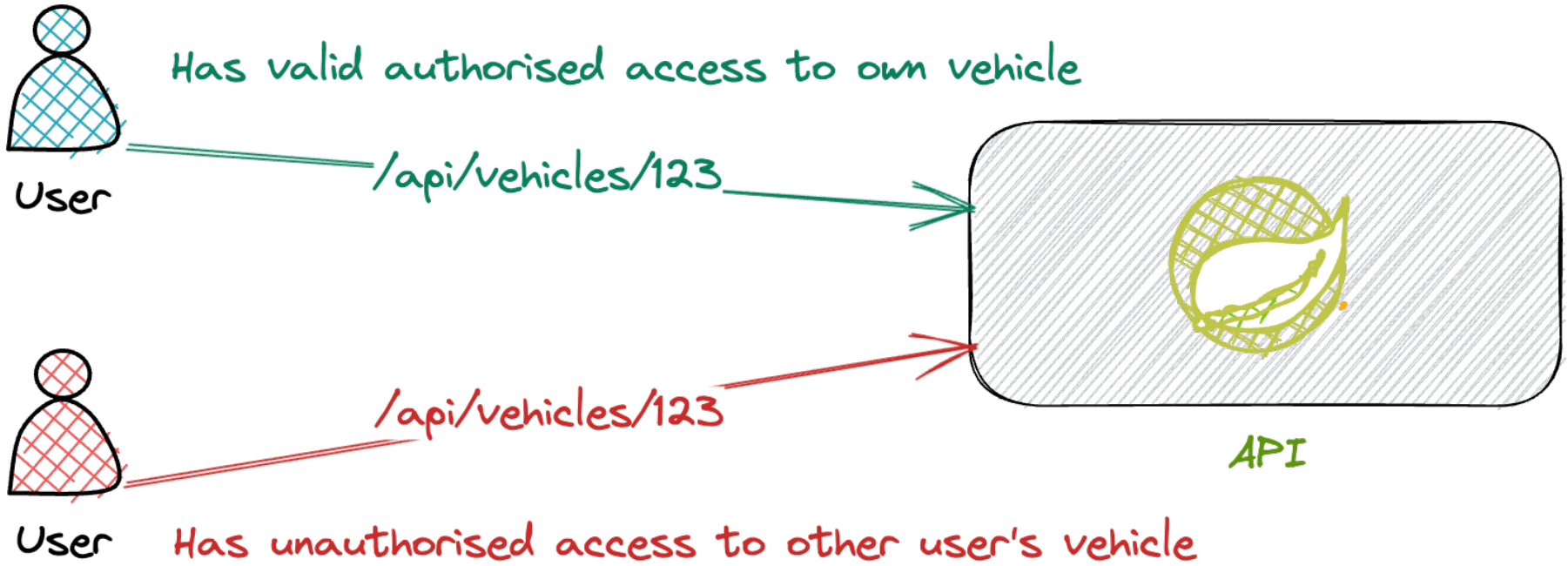
JSON / XML / BINARY

OWASP API Security Top 10 - 2023 (RC1)

API1:	Broken Object Level Authorization
API2:	Broken Authentication
API3:	Broken Object Property Level Authorization
API4:	Unrestricted Resource Consumption
API5:	Broken Function Level Authorization
API6:	Server-Side Request Forgery
API7:	Security Misconfiguration
API8:	Lack of Protection from Automated Threats
API9:	Improper Inventory Management
API10:	Unsafe Consumption of APIs

https://owasp.org/www-project-api-security

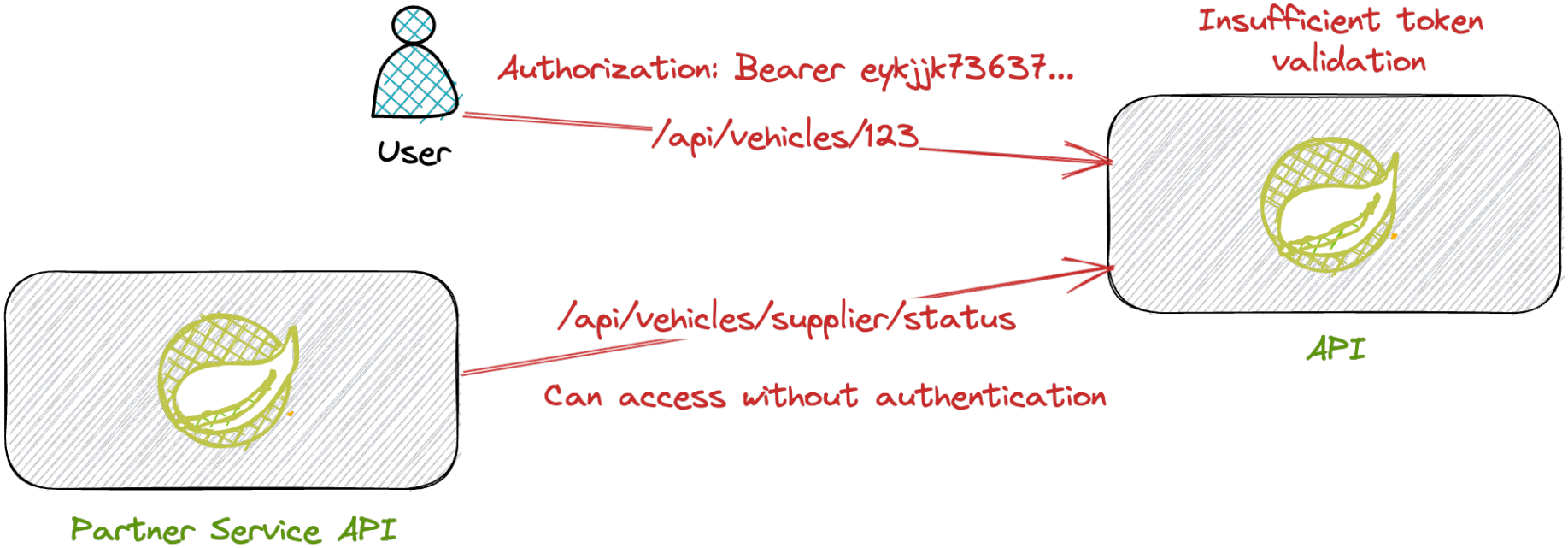
API1:Broken Object Level Authorization



API1:Broken Object Level Authorization

- How To Prevent
  - Implement proper authorization controls on objects level *(@PreAuthorize/@PostAuthorize* on objects or ACLs)
  - Prefer the use of random and unpredictable java.util.UUID values for IDs
  - Write automated tests to validate your authorization model

API2:Broken Authentication



API2:Broken Authentication JWT Issues

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "kid": "1c92250f-4842-4398-84e8-840b87faade4",
  "typ": "jwt",
  "alg": "RS256"
}
```

https://github.com/ticarpi/jwt_tool

PAYLOAD: DATA

```
{
  "sub": "c52bf7db-db55-4f89-ac53-82b40e8c57c2",
  "website": "https://example.com",
  "zoneinfo": "Europe/Berlin",
  "email_verified": true,
  "profile": "https://example.com/bwayne",
  "roles": [
    "USER"
  ],
  "iss": "http://localhost:9000",
  "preferred_username": "bwayne",
  "given_name": "Bruce",
  "locale": "de-DE",
  "aud": "demo-client-pkce",
  "nbf": 1684359424,
  "updated_at": "1970-01-01T00:00:00Z",
  "scope": [
    "openid",
    "profile",
    "email"
  ],
  "name": "Bruce Wayne",
  "nickname": "bwayne",
  "exp": 1684360324,
```

API2:Broken Authentication

- How To Prevent
  - Don't reinvent the wheel in authentication or token generation
    → Use Standards (OAuth 2.1 / OpenID Connect with JWT)
  - Use multi-factor authentication
  - Implement anti-brute force mechanisms
  - Learn how authentication mechanisms really work
- Token Authentication Best Practices
  - Use Authorization Code Grant + Proof Key for Code Exchange (PKCE) to retrieve Tokens (OAuth 2.1 / OpenID Connect)
  - Always validate JWT first before any usage (Signature, Expiry, Audience)

API3:Broken Object Property Level Authorization

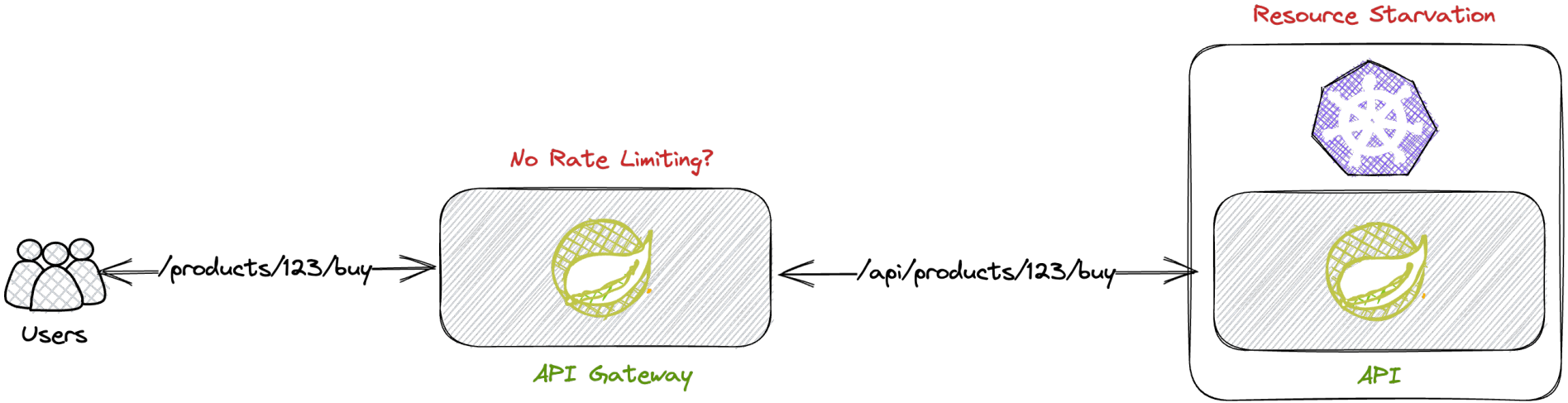

```
[
    {
        "id": "GrCApBovHfUBaGjfco8rGn",
        "title": "My new vehicle",
        "content": "Got a new car",
        "author": {
            "nickname": "Robot",
          ✗ "email": "robot001@example.com",
          ✗ "vehicleid": "189b4b7b-7b45-4c81",
        },
        "authorid": 3,
    },
    ...
]
```

API3:Broken Object Property Level Authorization

- How To Prevent
  - Restrict object data exposed in API endpoint
    (Value objects on API level or use @JsonIgnore/@JsonFilter)
  - Always make sure users should have access to exposed object’s data
  - Validate all changes to object's properties
  - Follow YAGNI* on business requirements for exposed data

*) You Aren't Gonna Need It
(https://martinfowler.com/bliki/Yagni.html)

API4:Unrestricted Resource Consumption



API4: Unrestricted Resource Consumption

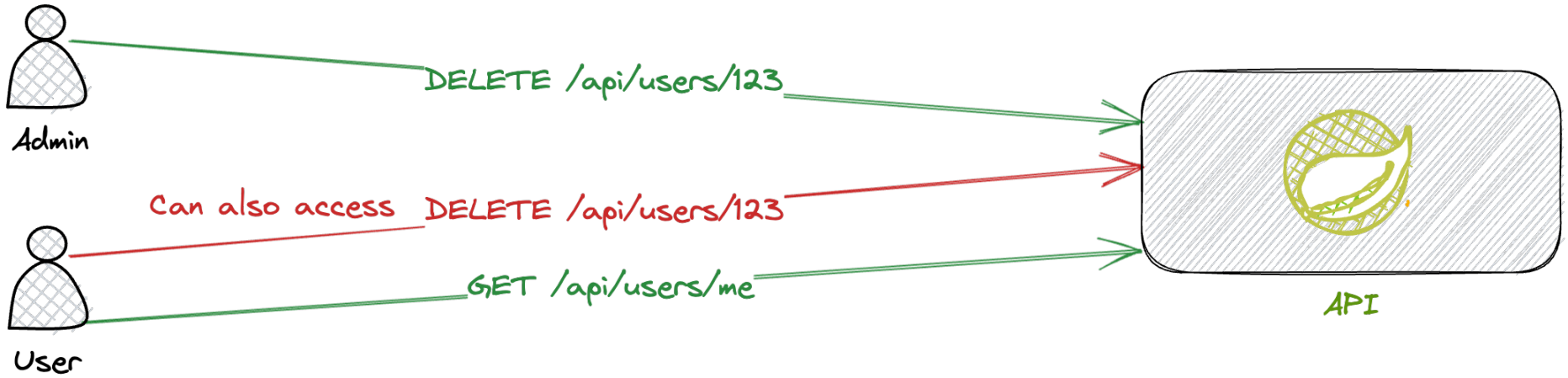
- How To Prevent
  - Implement Rate limiting (based on the business & protection needs) (*RedisRateLimiter* of Spring Cloud Gateway)
  - Restrict GraphQL query batching & query depth
  - Limit application resources (Memory, CPU, Processes, …) for **Java RT** and **Containers** and **Kubernetes** workloads
  - Enforce maximum size of all input parameters & payloads and file uploads
  - Set maximum limits on cloud service consumptions (i.e. autoscaling)

GraphQL Query Batching

POST /graphql

```
[
  {"query": "mutation {activateCard(token: \"abcdef\", card: \"0000\")
{authToken}}"},
  {"query": "mutation {activateCard(token: \"abcdef\", card: \"0001\")
{authToken}}"},
  ...
  {"query": "mutation {activateCard(token: \"abcdef\", card: \"9999\")
{authToken}}"}
}
```

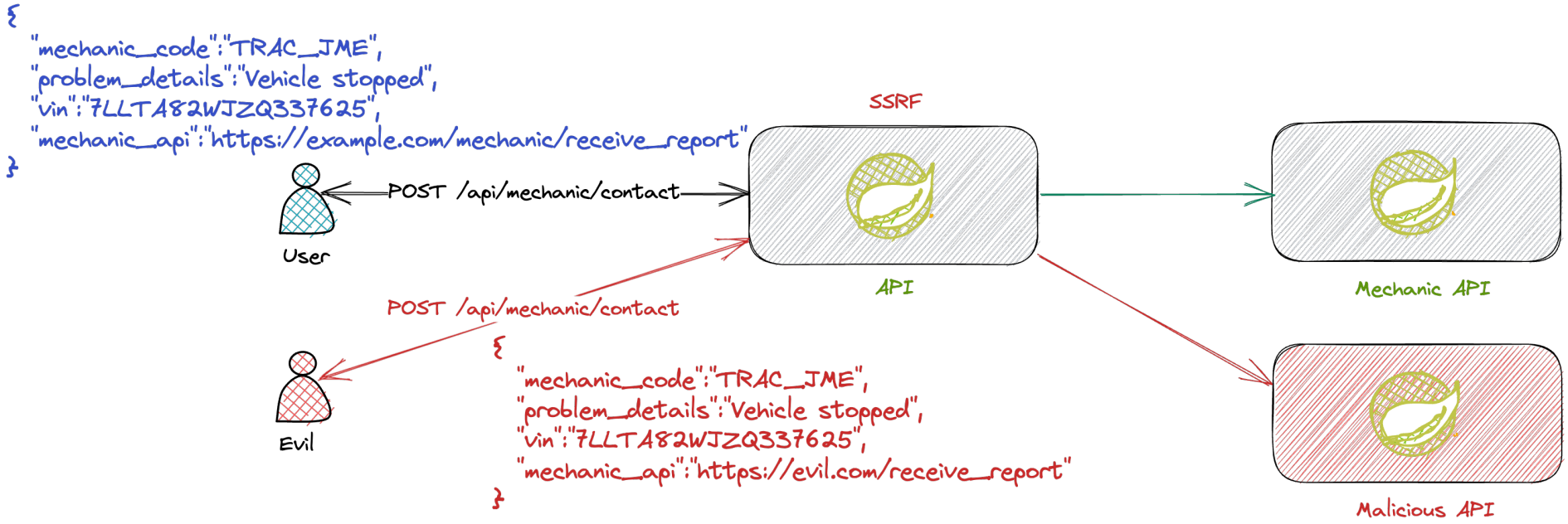
API5:Broken Function Level Authorization



API5:Broken Function Level Authorization

- How To Prevent
  - Make sure all API endpoints check on specific user roles
    *.requestMatchers("/admin/**").hasRole("ADMIN"))*
    @PreAuthorize("hasRole('ADMIN')")
  - The enforcement mechanism(s) should deny all access by default
    .anyRequest().denyAll() / .anyRequest().hasRole("USER")
  - Validates required roles with automated tests for all your API endpoints

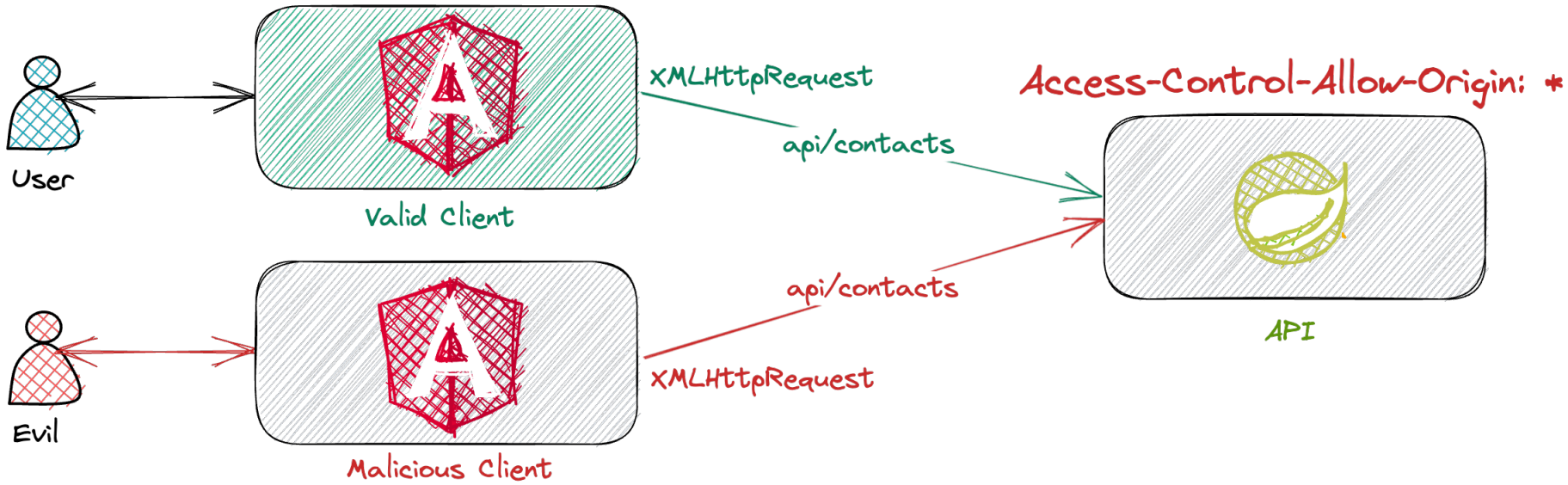
API6: Server Side Request Forgery (SSRF)



API6: Server Side Request Forgery (SSRF)

- How To Prevent
  - Use whitelist of
    - Remote Hosts
    - URL schemes and ports
    - Accepted media types for a given functionality
  - Disable HTTP redirections
  - Validate and sanitize all client-supplied input data (i.e. *jakarta.validation.constraints*)

API7: Security Misconfiguration



API7: Security Misconfiguration

- How To Prevent
  - Ensure that all API communications happen over TLS (internal and public APIs)
  - Be specific about which HTTP verbs each API can be accessed
  - Implement a proper CORS policy (No wildcards)
  - Disable including stack traces in errors (*server.error.include-stacktrace=never*)
  - Continuously review and update configurations (K8s Yaml, cloud services)

ManagedChannel channel = Grpc.newChannelBuilder("localhost:50051",
InsecureChannelCredentials*.create()).build();*

API7: Security Misconfiguration: GraphQL

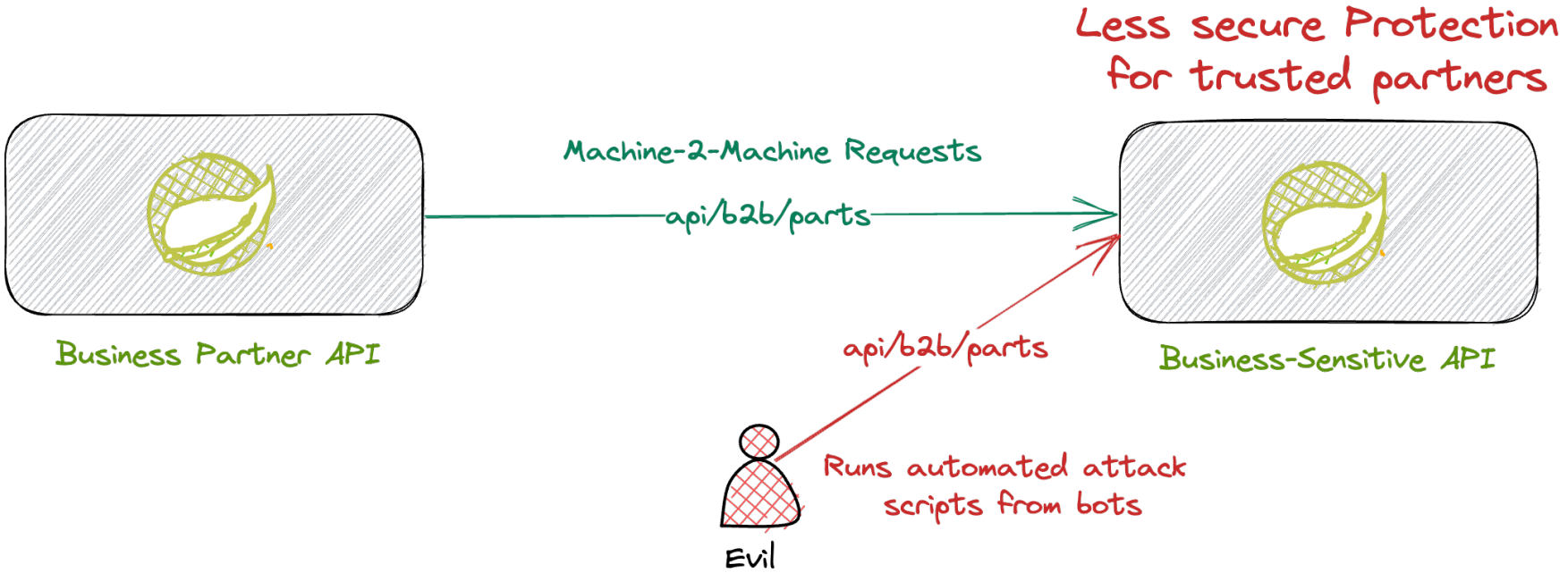
Security Considerations

graphql-java provides the following features which should be taken into consideration:

Field Suggestions	Query Depth Limit	Query Cost Analysis	Automatic Persisted Queries	Introspection	Debug Mode	Batch Requests
✅ Enabled by Default	⚠️ Disabled by Default	⚠️ Disabled by Default	❌ No Support	✅ Enabled by Default	❌ No Support	⚠️ Disabled by Default

https://github.com/nicholasaleks/graphql-threat-matrix

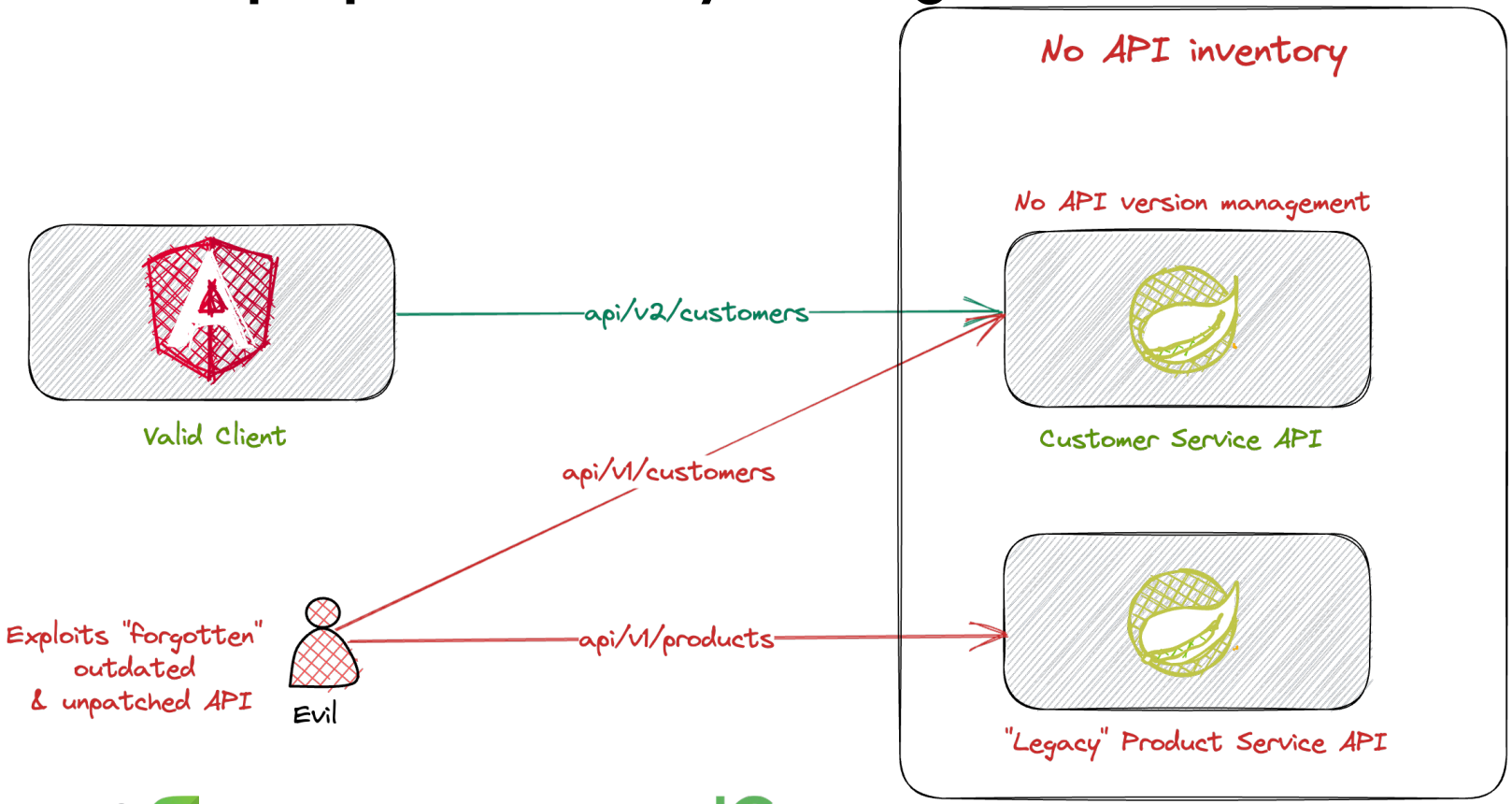
API8: Lack of Protection from Automated Threats



API8: Lack of Protection from Automated Threats

- How To Prevent
  - Identify critical business flows that might harm you if they are excessively used
  - Choose suitable protection mechanisms
    - Denying service to unexpected client devices
    - Analyze the user flow to detect non-human patterns (API Runtime Security)
  - Secure and limit access to APIs that are consumed directly by machines

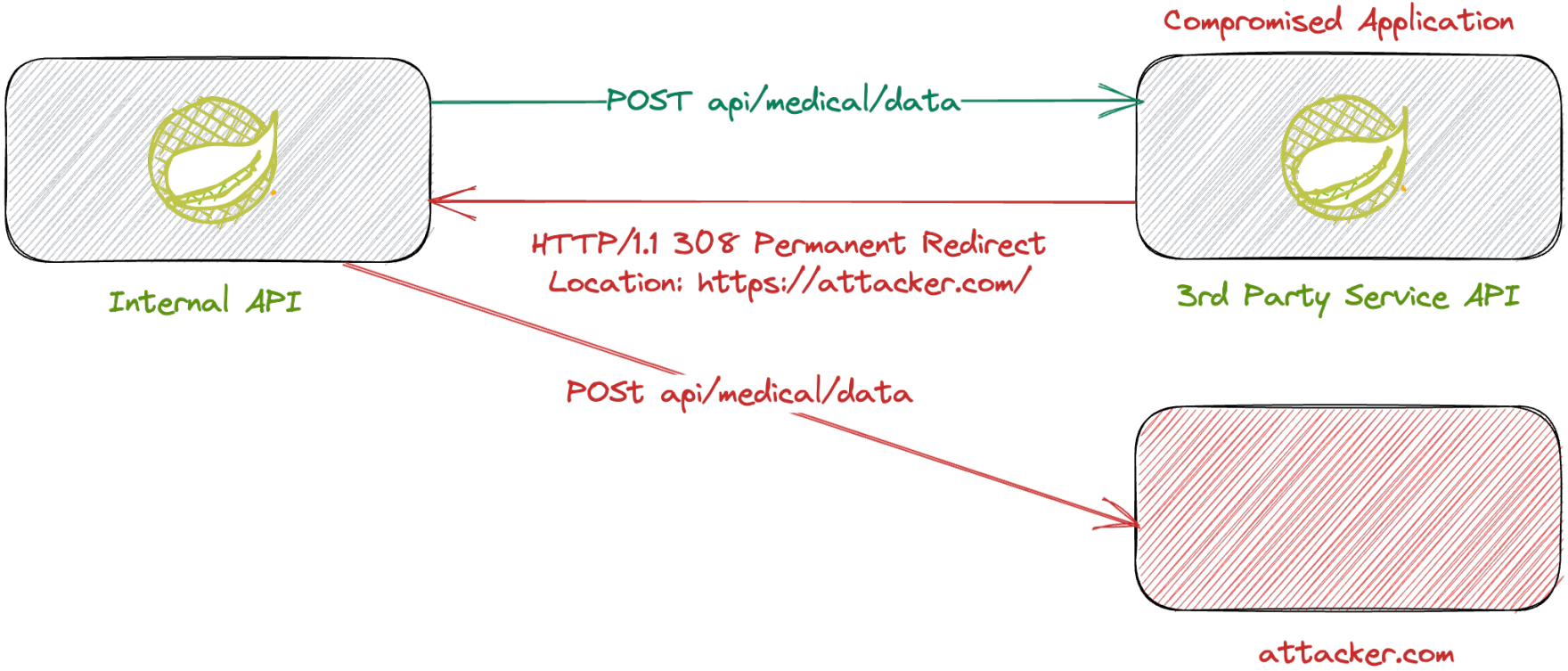
API9: Improper Inventory Management



API9: Improper Inventory Management

- How To Prevent
  - Inventory of all API services (versions, environments, processed data, authentication/authorization mechanism, consumers, targets, …)
  - Generate documentation automatically
    - OpenAPI 3 / Swagger
    - Spring Rest Docs
    - GraphiQL
  - API documentation for private APIs should only be available to those authorized to use the API
    (Or just disable on production, e.g. *spring.graphql.graphiql.enabled=false*)

API10: Unsafe Consumption of APIs



API10: Unsafe Consumption of APIs

- How To Prevent
  - Assess external 3rd party API security level and risks
  - Ensure all external API interactions happen over TLS
  - Always validate and properly sanitize data received from integrated APIs
  - Whitelist for Redirect URIs of external APIs, do not blindly follow redirects

Summary / Best Practices

- API Security Governance
  - Inventory of every API (version) & data flow
  - Classification of processed data
- API Runtime Security
  - Monitor & Log all API traffic
  - Detect & respond to abnormal API traffic
- Secure API Development
  - Checklists (OWASP, …) & Code Reviews
- API Security Testing
  - Static & Dynamic Application Security Testing

THANK YOU!
Q&A

Andreas Falk
@andifalk

https://github.com/andifalk/api-security

References

API1:Broken Object Level Authorization

- References
  - https://cheatsheetseries.owasp.org/cheatsheets/Authorization_Cheat_Sheet.html
  - https://docs.spring.io/spring-security/reference/servlet/authorization/index.html
  - https://docs.spring.io/spring-security/reference/servlet/test/method.html

API2:Broken Authentication

- References
  - https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html
  - OAuth 2.1 (https://oauth.net/2.1)
  - OpenID Connect (https://openid.net/specs/openid-connect-core-1_0.html)
  - JWT Best Current Practices (https://www.rfc-editor.org/rfc/rfc8725.html)
  - https://docs.spring.io/spring-security/reference/servlet/oauth2/resource-server/index.html

API3:Broken Object Property Level Authorization

- References
  - https://cheatsheetseries.owasp.org/cheatsheets/Mass_Assignment_Cheat_Sheet.html
  - https://www.baeldung.com/jackson-ignore-properties-on-serialization
  - https://docs.spring.io/spring-security/reference/servlet/authorization/index.html

API4:Unrestricted Resource Consumption

- References
  - https://cheatsheetseries.owasp.org/cheatsheets/Denial_of_Service_Cheat_Sheet.html
  - https://docs.spring.io/spring-cloud-gateway/docs/current/reference/html
  - https://learn.microsoft.com/en-us/azure/developer/java/containers/overview
  - https://docs.docker.com/config/containers/resource_constraints
  - https://kubernetes.io/docs/tasks/configure-pod-container

API5:Broken Function Level Authorization

- References
  - https://cheatsheetseries.owasp.org/cheatsheets/Authorization_Cheat_Sheet.html
  - https://docs.spring.io/spring-security/reference/servlet/authorization/authorize-http-requests.html
  - https://docs.spring.io/spring-security/reference/servlet/authorization/method-security.html

API6: Server Side Request Forgery (SSRF)

- References
  - https://cheatsheetseries.owasp.org/cheatsheets/Server_Side_Request_Forgery_Prevention_Cheat_Sheet.html
  - https://www.baeldung.com/httpclient-stop-follow-redirect
  - https://docs.spring.io/spring-framework/reference/integration/rest-clients.html
  - https://docs.spring.io/spring-framework/reference/web/webflux-webclient.html

API7: Security Misconfiguration

- References
  - https://cheatsheetseries.owasp.org/cheatsheets/Error_Handling_Cheat_Sheet.html
  - https://github.com/andifalk/client-certificate-demo
  - https://docs.spring.io/spring-framework/reference/web/webmvc-cors.html
  - https://securityheaders.com

API8: Lack of Protection from Automated Threats

- References
  - https://owasp.org/www-project-automated-threats-to-web-applications

API9: Improper Inventory Management

- References
  - https://springdoc.org
  - https://docs.spring.io/spring-restdocs/docs/current/reference/htmlsingle

API10: Unsafe Consumption of APIs

- References
  - https://cheatsheetseries.owasp.org/cheatsheets/Unvalidated_Redirects_and_Forwards_Cheat_Sheet.html
  - https://cheatsheetseries.owasp.org/cheatsheets/Input_Validation_Cheat_Sheet.html