8636 Criptografía y Seguridad Informática

Seguridad en Arquitectura de Servicios

Agenda

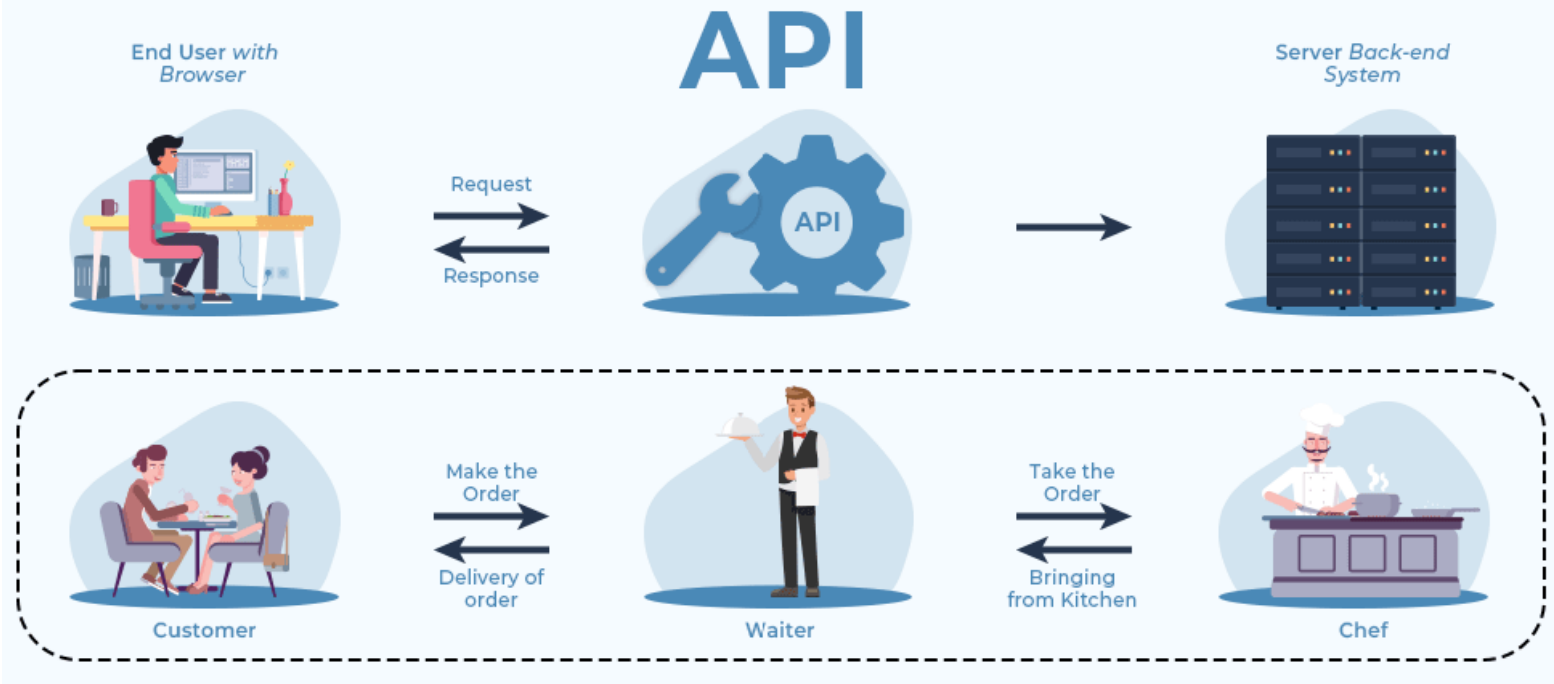
1. APIs: Protocolos y Arquitecturas
   - Rest
   - Soap
2. Protocolos de Autenticación y Autorización
   - SAML
   - OAuth y OpenID
3. OWASP top 10 APIs: Estudio de las Vulnerabilidades

Application Programming Interfaces

Las Interfaces de Programación de Aplicaciones (API) REST y SOAP facilitan la comunicación entre sistemas. Mientras SOAP se basa en XML, REST utiliza JSON y es más flexible y ampliamente adoptado en el desarrollo moderno.

Application Programming Interfaces

- Conjunto de definiciones y protocolos que se utiliza para diseñar e integrar el software de las aplicaciones.
- Permite la comunicación entre distintos sistemas sin necesidad de conocer los detalles de implementación de los mismos.



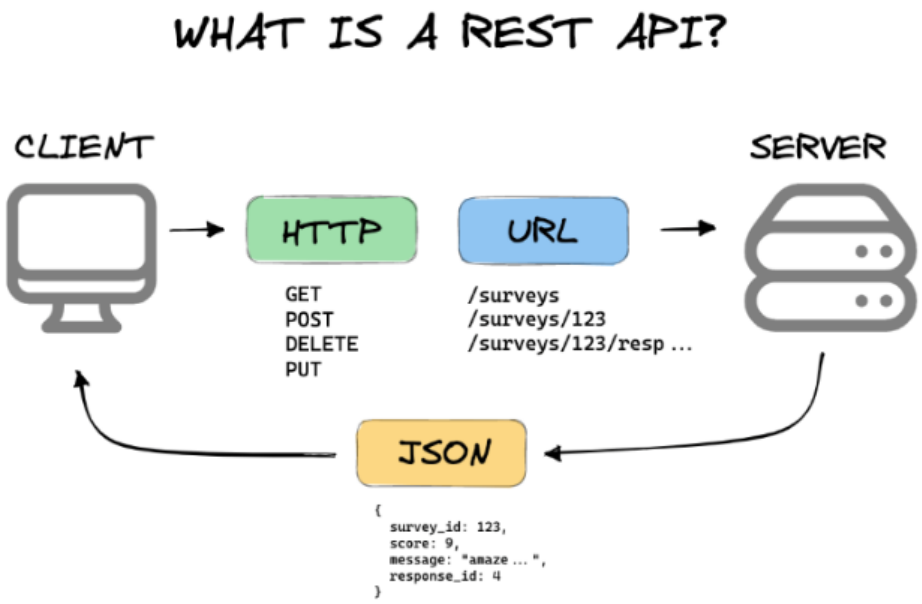
APIs: Protocolos

REST

- Es una arquitectura que se utiliza comúnmente en el desarrollo de servicios web y aplicaciones web. Siguen una colección de principios y técnicas:
- **Client-Server:** clientes y servidores plenamente identificados e independientes.
- **Stateless:** comunicaciones entre cliente y servidor sin estado.
- **Cachebale:** respuestas "cacheables" por parte del cliente.
- **Layered System:** existencia de capas intermedias de servidores para mejorar escalabilidad, rendimiento y seguridad.
- **Uniform Interface:** interfaz uniforme entre cliente y servidor, que permite su evolución y desarrollo independiente .

REST

- En REST, los recursos son la entidad central que se puede identificar con una URI.
- Las operaciones comunes que se pueden realizar en los recursos son estándares de HTTP, como GET, POST, PUT y DELETE.
- Los recursos pueden tener múltiples representaciones, que pueden ser en diferentes formatos como HTML, XML, JSON.



Formato JSON

```json
{
  "string": "Hi",
  "number": 2.5,
  "boolean": true,
  "null": null,
  "object": { "name": "Kyle", "age": 24 },
  "array": ["Hello", 5, false, null, { "key": "value", "number": 6 }],
  "arrayOfObjects": [
    { "name": "Jerry", "age": 28 },
    { "name": "Sally", "age": 26 }
  ]
}
```

APIs: Protocolos

SOAP

- Protocolo de comunicación que permite la transferencia de datos estructurados
- Puede ser utilizado sobre una variedad de protocolos , como HTTP, SMTP u otros.
- Esta basado en **XML** y define la estructura del mensajes a intercambiar.

Estructura del Mensaje

- Envelope: Establece la estructura general del mensaje SOAP y contiene los elementos esenciales que determinan cómo se organiza y procesa el mensaje.
- Header(opcional): Permite la inclusión de elementos de seguridad en el mensaje
- Body: Contiene el contenido principal del mensaje, que puede ser una solicitud, una respuesta o información relevante .

SOAP

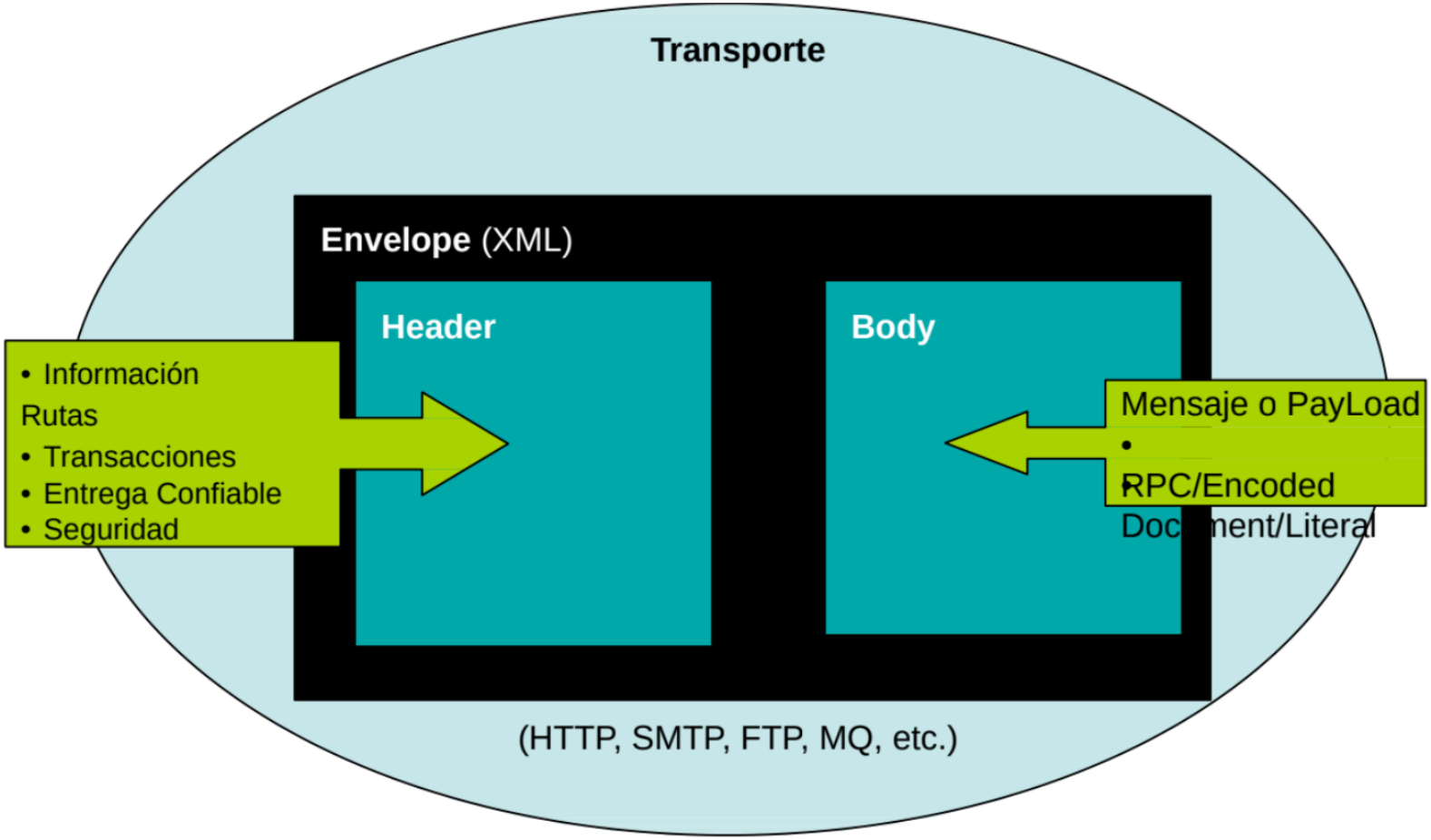
Request

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Body>
     <getProductDetails xmlns="http://warehouse.example.com/ws">
       <productId>827635</productId>
     </getProductDetails>
   </soap:Body>
</soap:Envelope>
```

Response

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Body>
     <getProductDetailsResponse xmlns="http://warehouse.example.com/ws">
       <getProductDetailsResult>
         <productName>Toptimate 3-Piece Set</productName>
         <productId>827635</productId>
         <description>3-Piece luggage set.  Black Polyester.</description>
         <price>96.50</price>
         <inStock>true</inStock>
       </getProductDetailsResult>
     </getProductDetailsResponse>
   </soap:Body>
</soap:Envelope>
```

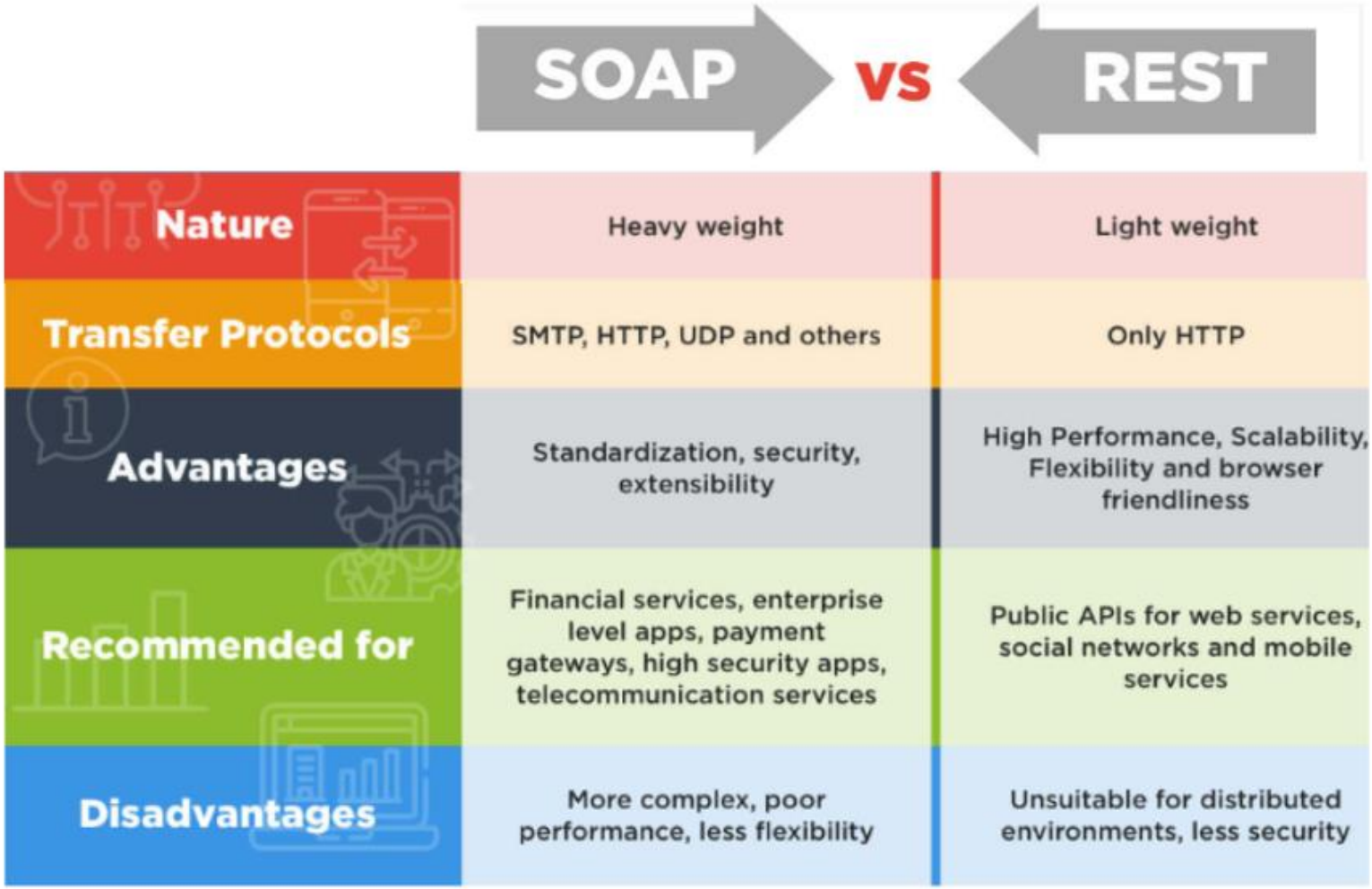
SOAP


Transporte
Envelope (XML)
Header
Body
• Información
Rutas
• Transacciones
• Entrega Confiable
• Seguridad
Mensaje o PayLoad
•
RPC/Encoded
Doc ment/Literal
(HTTP, SMTP, FTP, MQ, etc.)

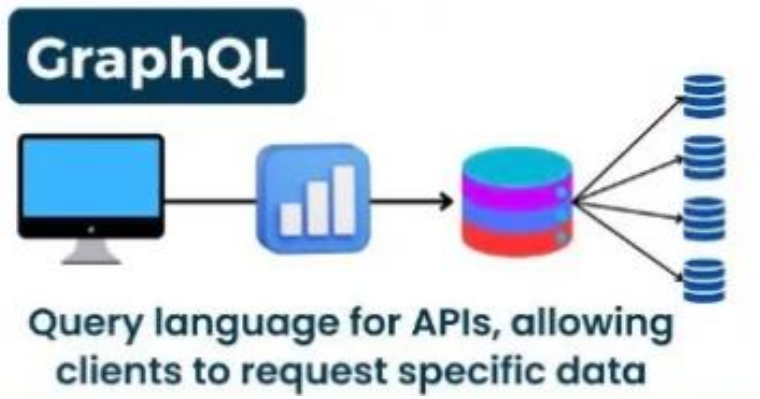
Rest VS Soap

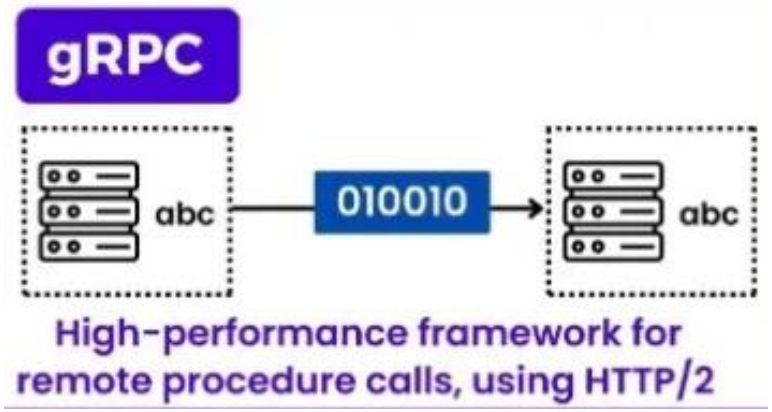
	SOAP	REST
Performance	Requires more power, resources and bandwidth	Requires fewer resources
Design	Standard protocol with predefined rules to follow	Architectural style with loose recommendation and guidelines
Caching	API calls are not cached	API calls are cached
Security	WS-Security with SSL support. Provides an in built ACID compliance	Supports SSL and HTTPS
Messaging Format	Only XML	XML, JSON, plain text YAML, HTML, and others

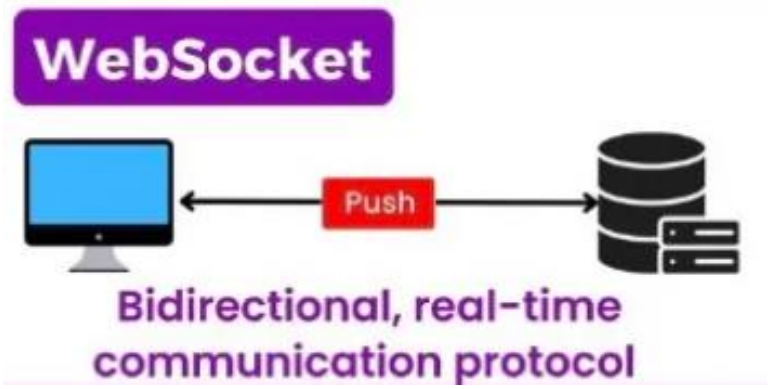
Rest VS Soap

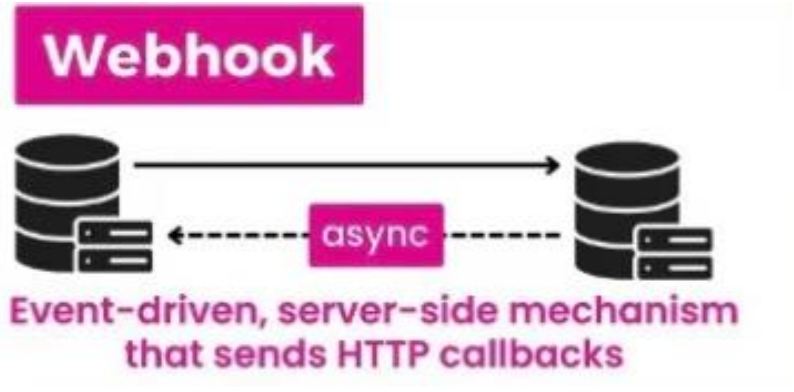

SOAP
VS
REST
Nature
Heavy weight
Light weight
Transfer Protocols
SMTP, HTTP, UDP and others
Only HTTP
Advantages
Standardization, security, extensibility
High Performance, Scalability, Flexibility and browser friendliness
Recommended for
Financial services, enterprise level apps, payment gateways, high security apps, telecommunication services
Public APIs for web services, social networks and mobile services
Disadvantages
More complex, poor performance, less flexibility
Unsuitable for distributed environments, less security

Otras Arquitecturas









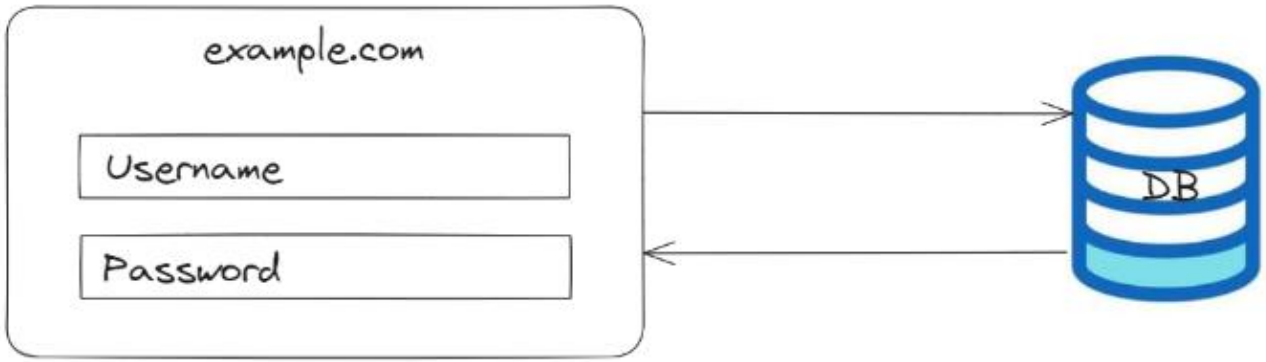
https://blog.postman.com/different-types-of-apis/

Autenticación y Autorización

La autenticación y autorización en servicios web son fundamentales. SAML permite intercambio seguro de identidad, OAuth ofrece autorización, y OpenID facilita la autenticación de usuarios de forma federada.

Autenticación y Autorización

Autenticación Basica



- Ingreso de credenciales
- Hashing de la contraseña
- Verificación de las credenciales
- Autenticación

Autenticación y Autorización

SAML

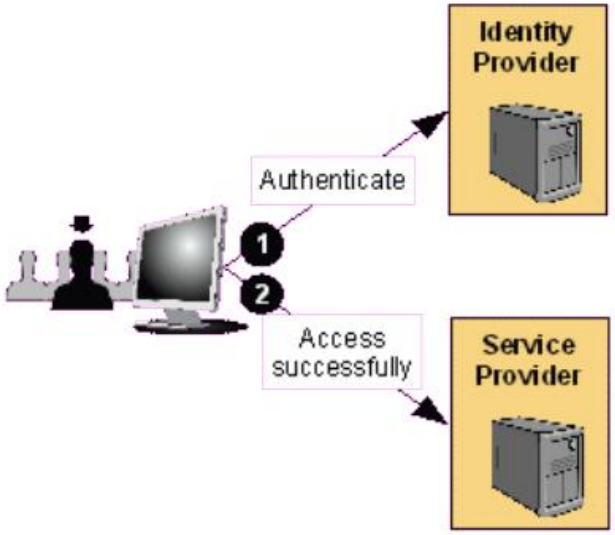
- Es un estándar basado en XML para intercambiar información de autenticación y autorización entre dos dominios de seguridad distintos.
- Es utilizado para resolver el problema de SSO en aplicaciones Web.

https://datatracker.ietf.org/doc/html/rfc7522

SAML

Participantes en SAML

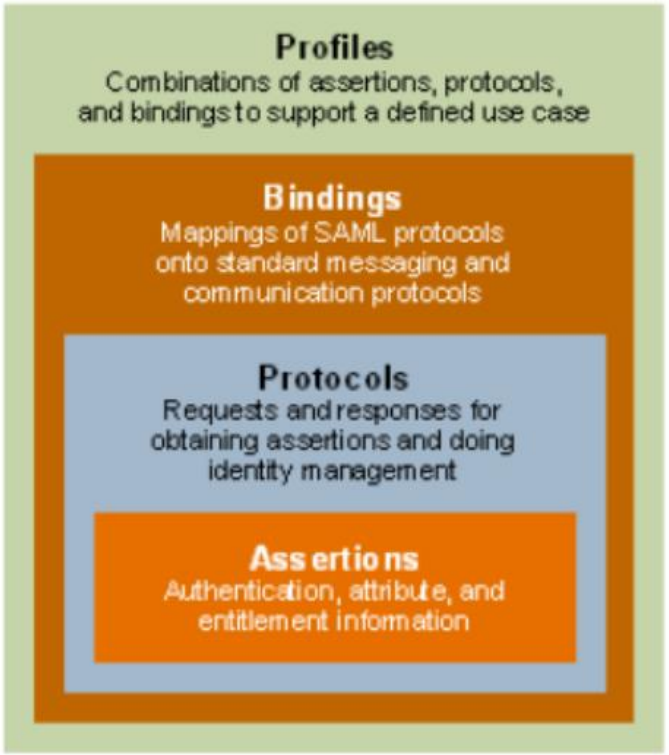
- **Identity Provider(IdP):** Es una entidad que se encarga de autenticar a los usuarios y proporcionar afirmaciones de identidad sobre ellos.
- **Service Provider (SP):** Es una entidad que ofrece servicios o recursos protegidos a los usuarios y necesita confiar en el IdP para autenticar a los usuarios



SAML

Conceptos

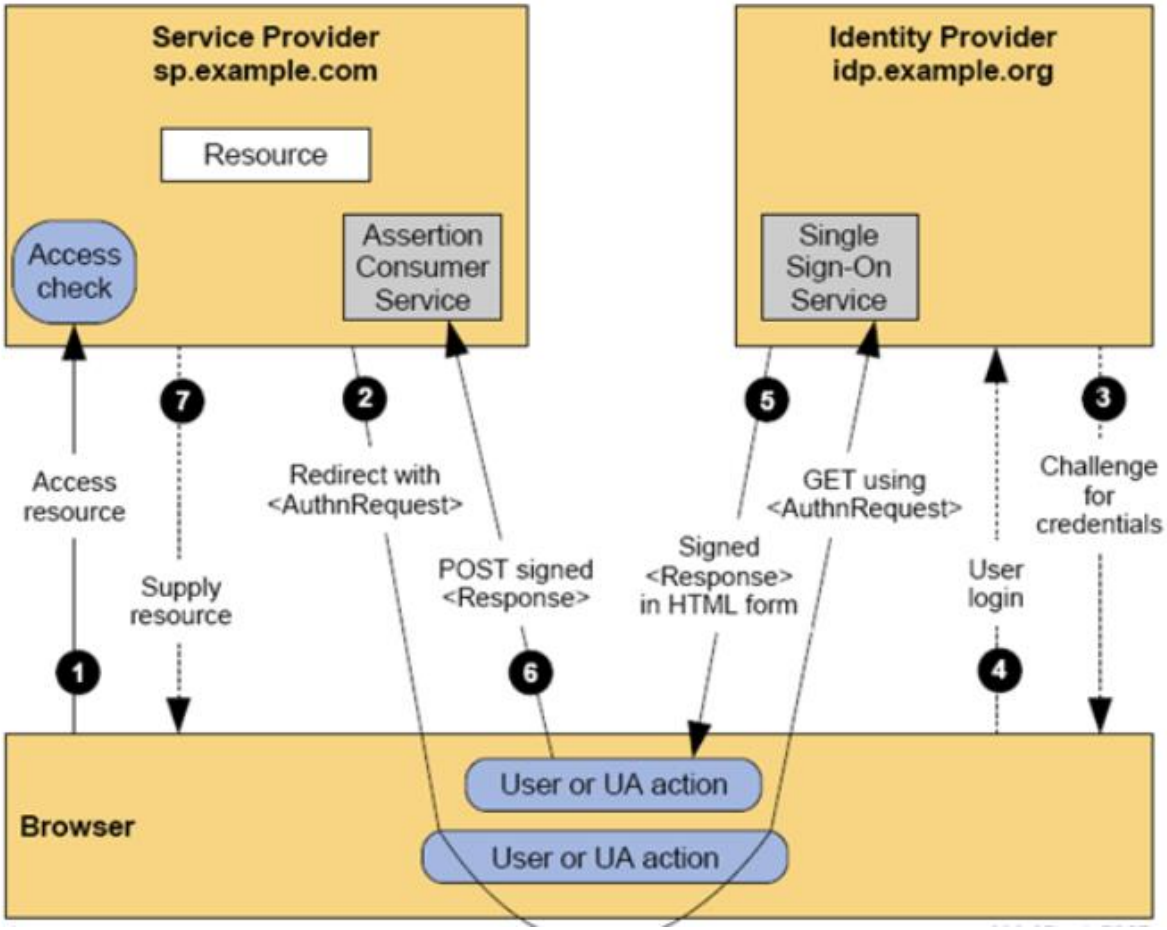
- **Assertions:** Contienen información sobre la identidad del usuario y los atributos relacionados con la autenticación y la autorización. Son firmadas digitalmente por el IdP para garantizar su integridad y autenticidad
- **Protocols:** Estándares y especificaciones que definen cómo se intercambian las assertions y los mensajes SAML entre el Identity Provider (IdP) y el Service Provider (SP).
- **Bindings:** Definen cómo se encapsulan y transmiten las assertions SAML a través de protocolos de comunicación específicos
- **Profiles:** Combinaciones de assertions, protocols y bindings.



http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-tech-overview-2.0.pdf

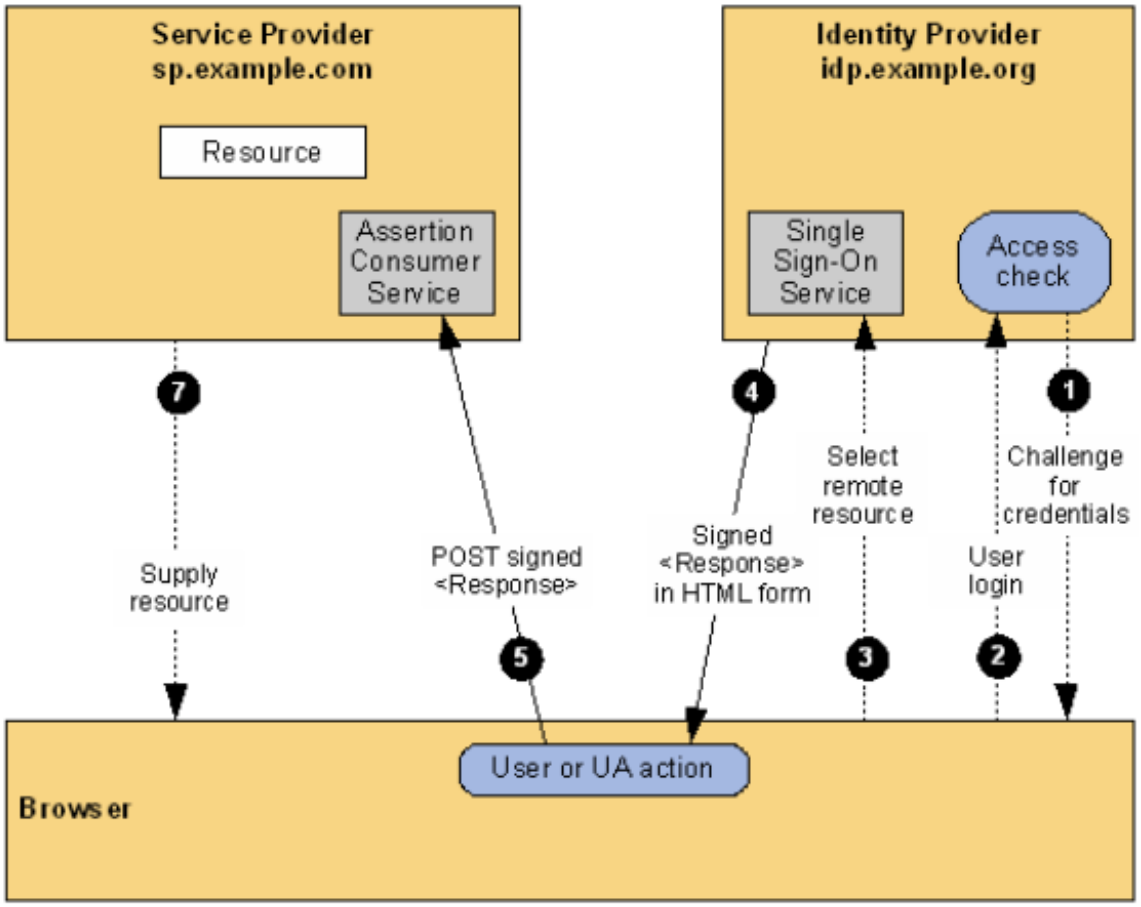
SAML

Funcionamiento: SP - INIT



SAML

Funcionamiento: IDP - INIT



Autenticación y Autorización

OAuth

- OAuth permite que una aplicación de terceros obtenga acceso limitado a un servicio HTTP, ya sea en nombre del propietario del recurso , o por sí misma.
- Es un framework de autorización, no de autenticación.

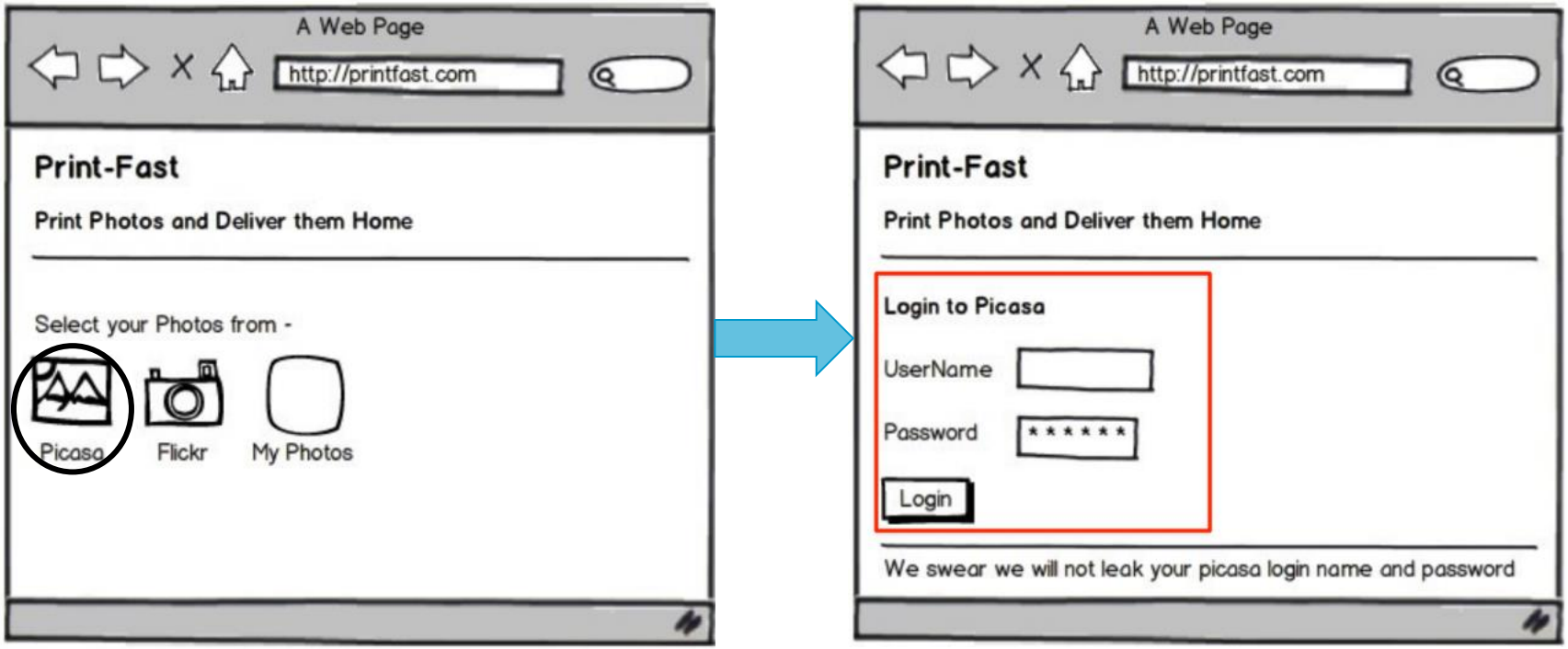
  ¿Que permisos tenes? ~~¿Quien sos?~~

- Soluciona el problema de la delegación de autorización:

  ¿Puedo dejar a un website acceder a mi información sin entregarle mi contraseña?

https://www.ietf.org/rfc/rfc6749.txt

Caso de Estudio sin OAuth



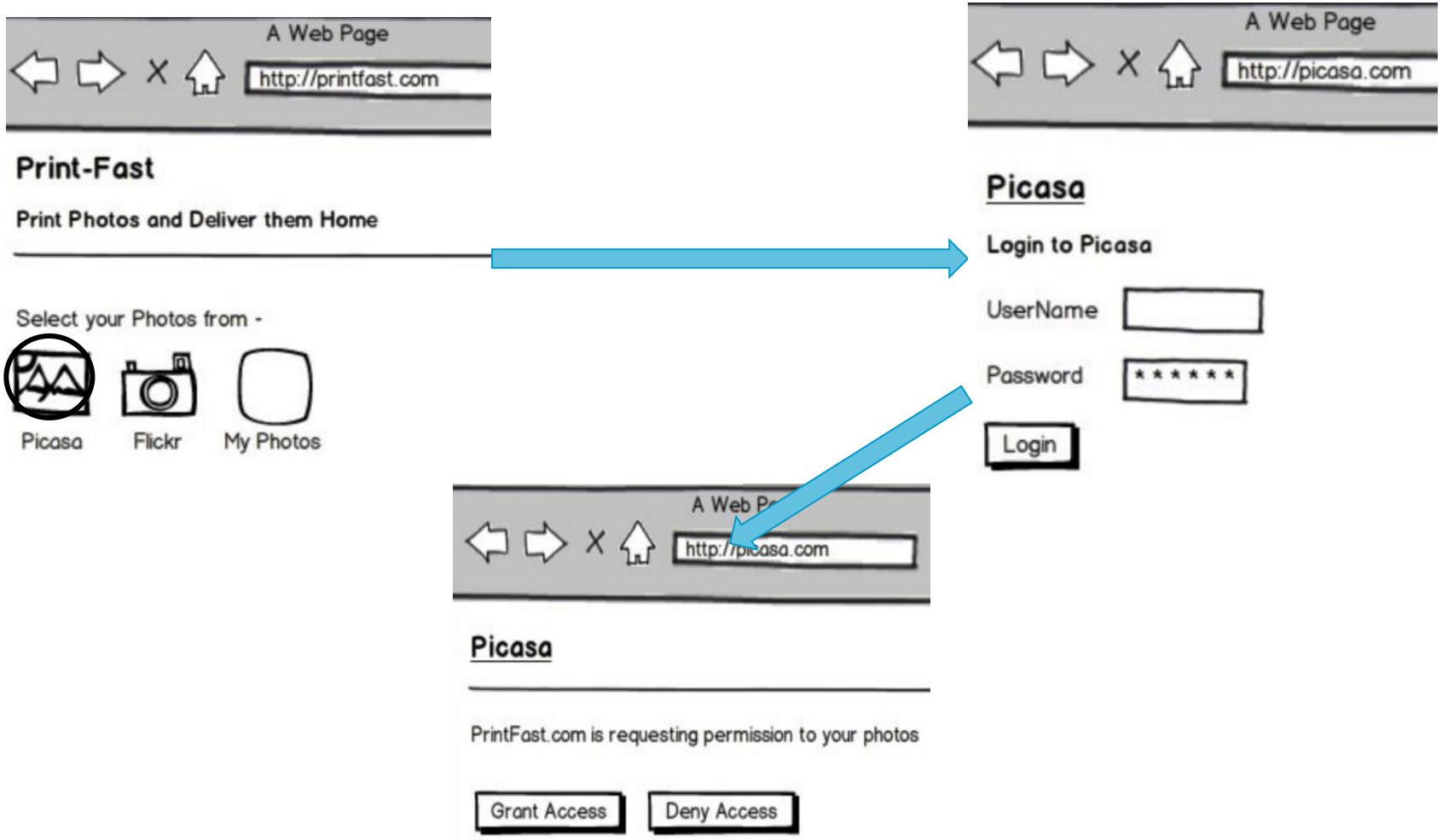
Caso de Estudio sin OAuth

- La aplicación le pidió las credenciales al usuario. Luego se hizo pasar por el usuario.

¿Por que es un problema?

- Las aplicaciones almacenan las credenciales
- Las aplicaciones tienen acceso completo a la información del usuario
- No se puede revocar el acceso a la cuenta salvo que se cambiara la password
- Si la apliación es comprometida, se comprometen los accesos a las cuentas

Caso de Estudio con OAuth

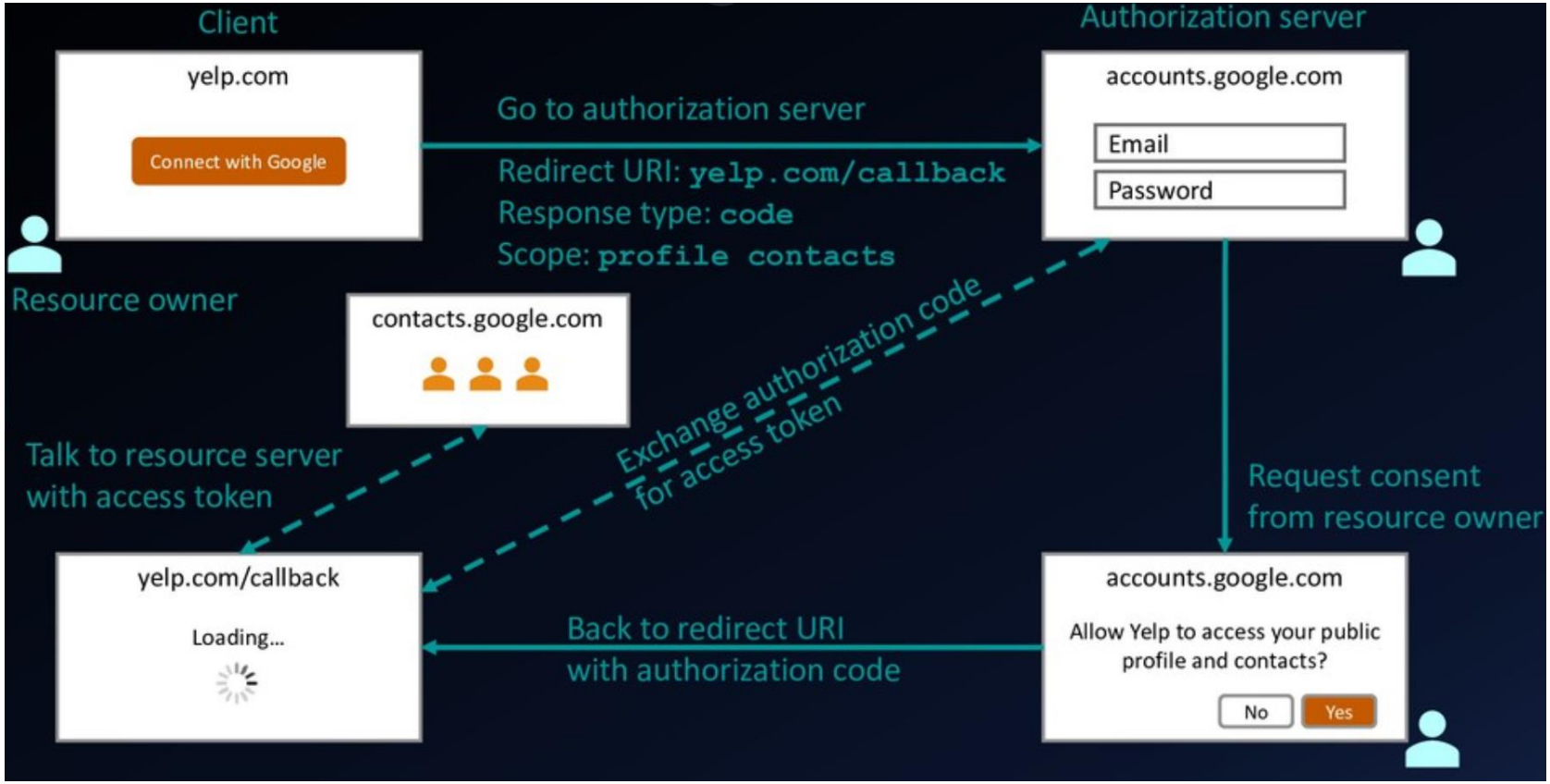


Roles en OAuth

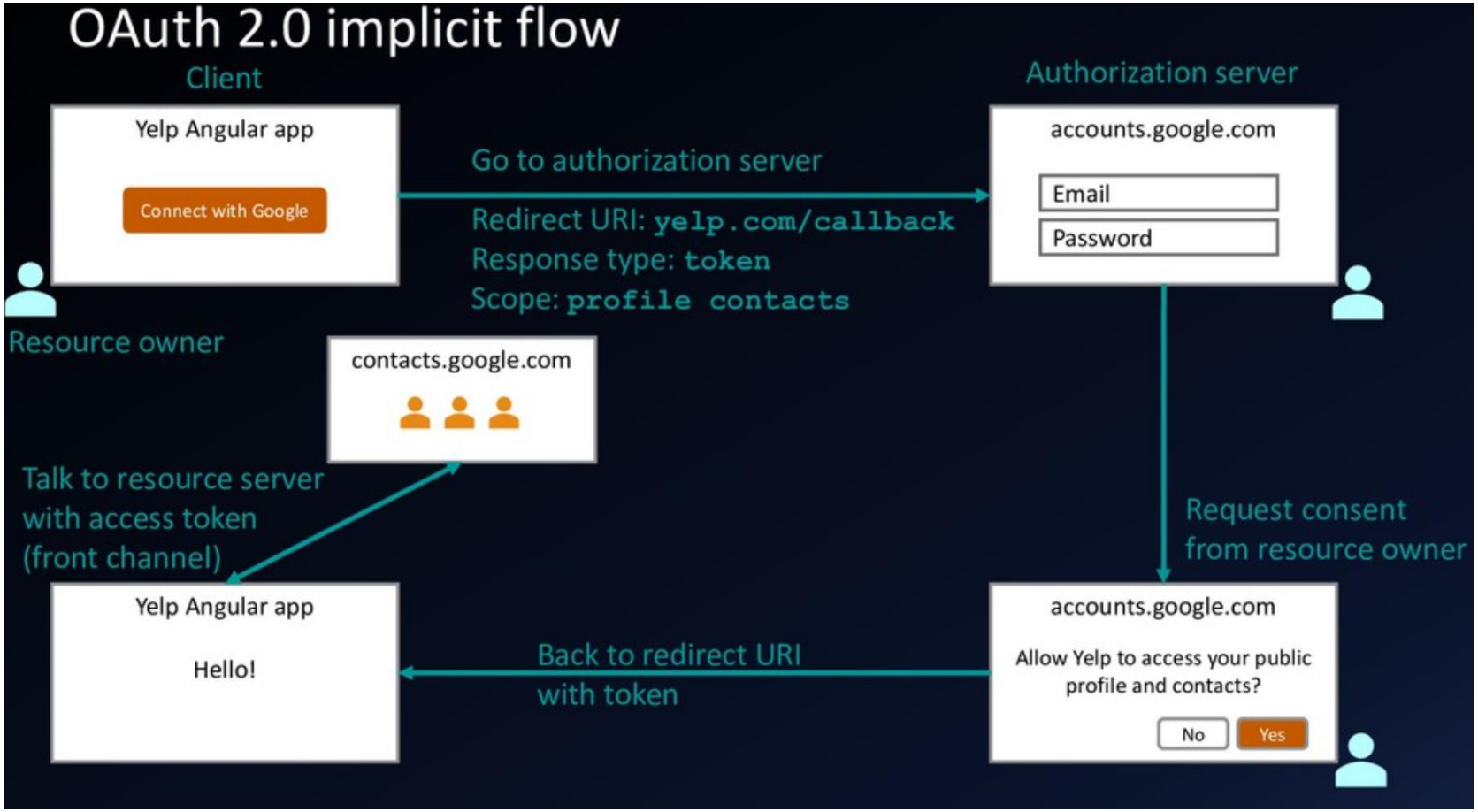
- **Resource Owner:** El dueño del recurso protegido. Cuando es una persona, se lo llama end-user. (Usuario de Picasa)
- **Resource Server:** El servidor que aloja el recurso protegido al cual queremos acceder. (Picasa)
- **Client:** La aplicación que desea acceder a los recursos protegidos del usuario. (printfast)
- **Authorization Server:** Verifica la identidad de los usuarios y emite una serie de tokens de acceso a la aplicación cliente. (Google Auth Server)

Authorization code flow

- **Objetivo:** Permitir a una aplicación de terceros tener acceso limitado a una api en nombre del usuario.



Implicit Flow



Authorization code flow

Comienza el Flujo

```
https://accounts.google.com/o/oauth2/v2/auth?
 client_id=abc123&
 redirect_uri=https://yelp.com/callback&
 scope=profile&
 response_type=code&
 state=foobar
```

Redirección

```
https://yelp.com/callback?
 error=access_denied&
 error_description=The user did not consent.
```

```
https://yelp.com/callback?
 code=oMsCeLvIaQm6bTrgtp7&
 state=foobar
```

Authorization code flow

Intercambio del código por el Token de Acceso

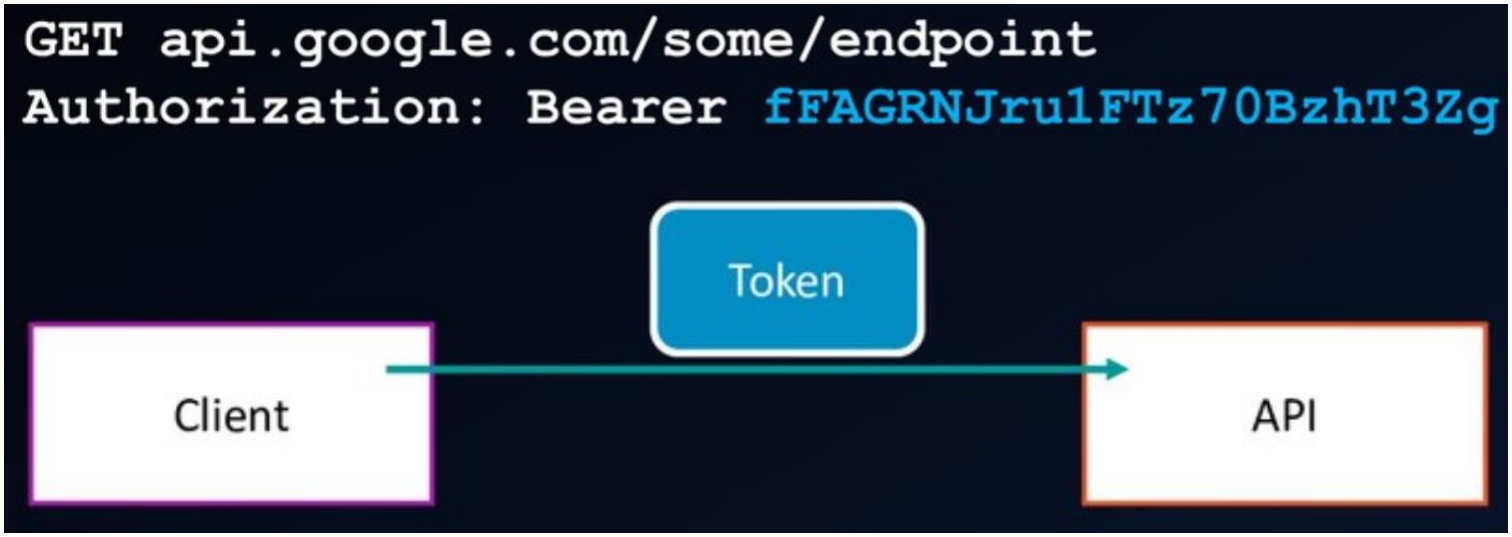
```
POST www.googleapis.com/oauth2/v4/token
Content-Type: application/x-www-form-urlencoded

code=oMsCeLvIaQm6bTrgtp7&
client_id=abc123&
client_secret=secret123&
grant_type=authorization_code
```

```
{
  "access_token": "fFAGRNJru1FTz70BzhT3Zg",
  "expires_in": 3920,
  "token_type": "Bearer",
}
```

Authorization code flow

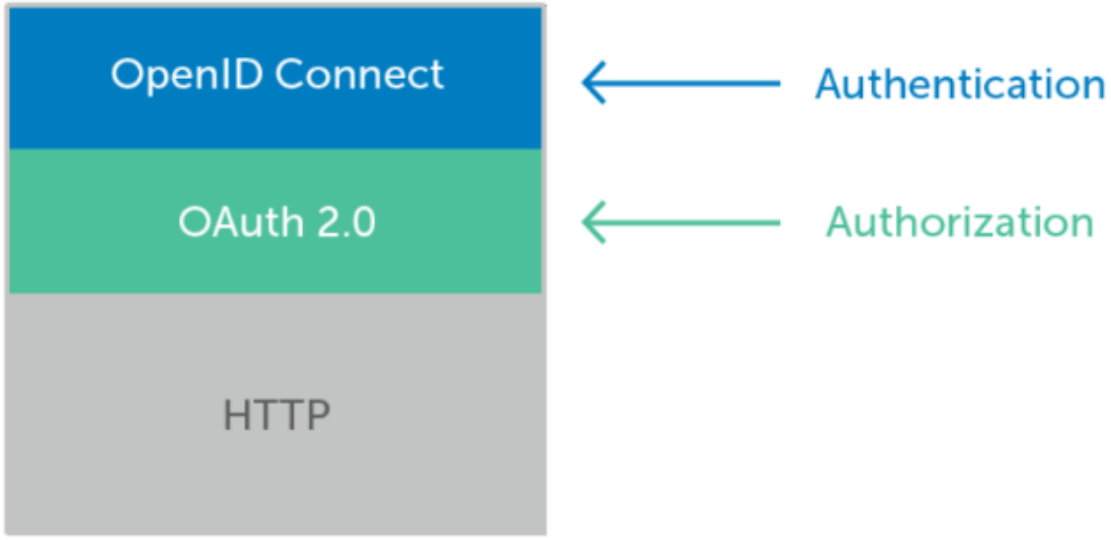
Uso del token



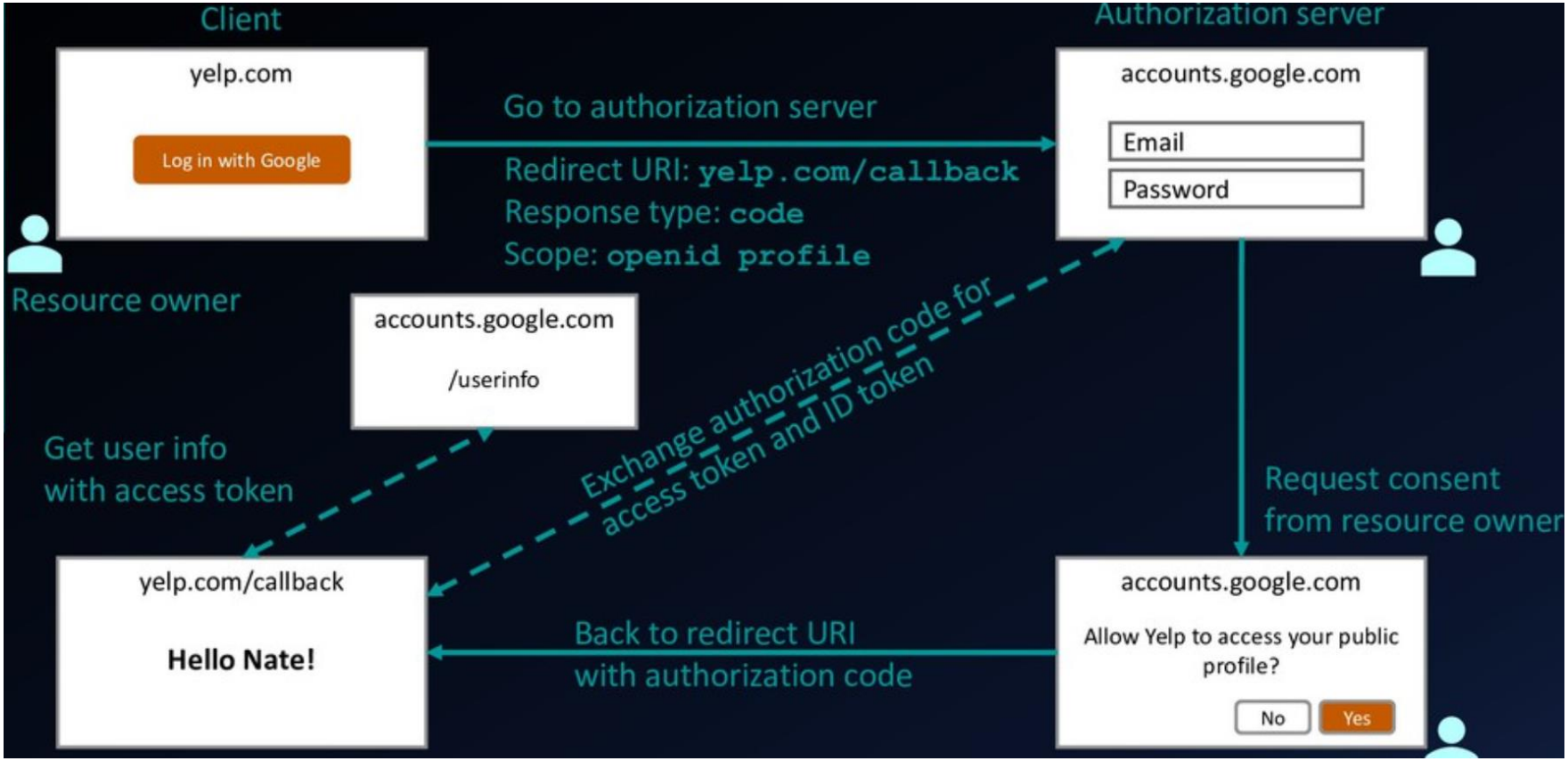
Autenticación y Autorización

OpenID Connect

- Es un protocolo de Autenticación construido sobre OAuth 2
- Permite verificar la identidad del usuario final y obtener información (consentida) del usuario.



OpenID Connect Authorization Flow



OpenID Connect Authorization Flow

Comienza el Flujo

```
https://accounts.google.com/o/oauth2/v2/auth?
 client_id=abc123&
 redirect_uri=https://yelp.com/callback&
 scope=openid profile&
 response_type=code&
 state=foobar
```

Intercambio del código por el Token de Acceso

```
POST www.googleapis.com/oauth2/v4/token
Content-Type: application/x-www-form-urlencoded

code=oMsCeLvIaQm6bTrgtp7&
client_id=abc123&
client_secret=secret123&
grant_type=authorization_code
```

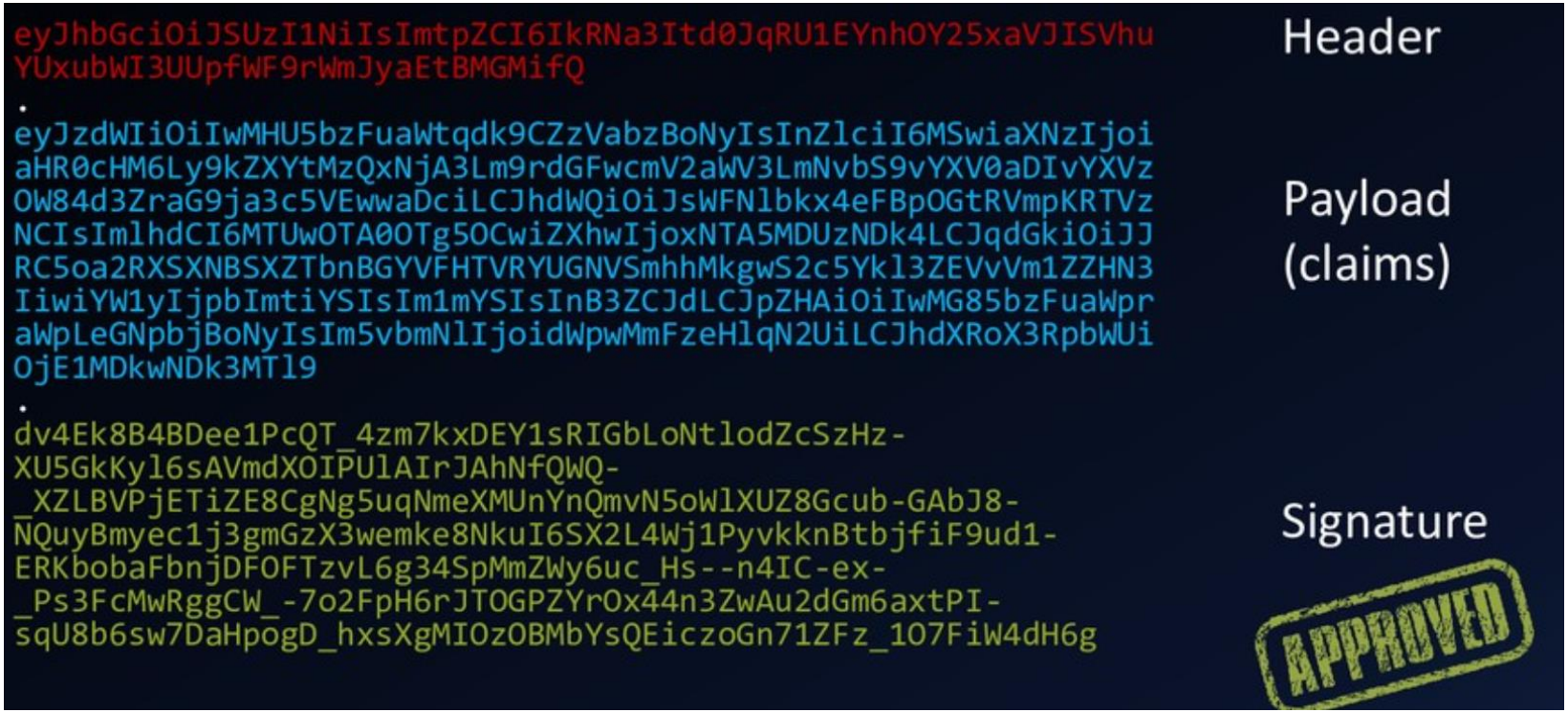
OpenID Connect Authorization Flow

Retorna Token de Acceso e Identificación

```
{
  "access_token": "fFAGRNJru1FTz70BzhT3Zg",
  "id_token": "eyJraB03ds3F..."
  "expires_in": 3920,
  "token_type": "Bearer",
}
```

OpenID Connect Authorization Flow

ID Token (JWT)



OAuth y OpenID Connect: Demo

En este laboratorio, exploraremos la implementación de dos protocolos fundamentales de seguridad y autenticación en APIs: OAuth 2.0 y OpenID Connect.

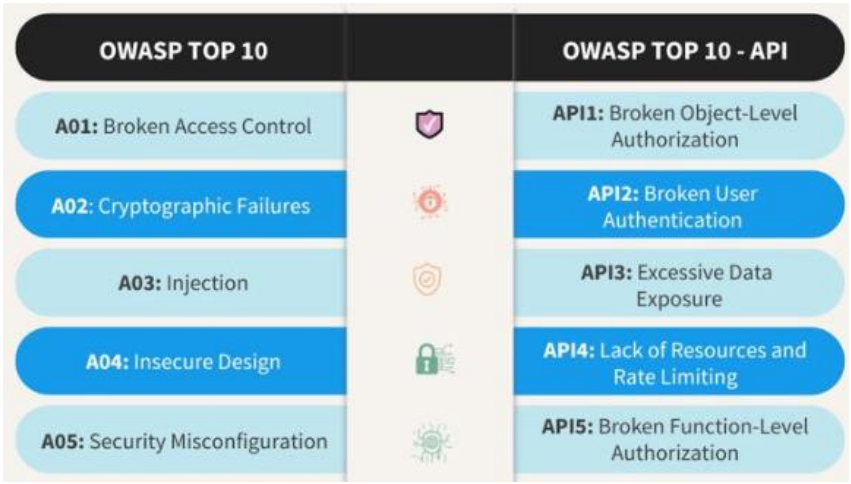
Owasp Top 10 Apis

El OWASP Top 10 para APIs es una lista que identifica las principales vulnerabilidades de seguridad en las APIs. Ofrece pautas para proteger y fortalecer sistemas API contra amenazas comunes.

OWASP TOP 10

OWASP es una comunidad colaborativa sin fines de lucro que producen artículos, metodologías y documentación sobre seguridad informática.

El aumento de la utilización de la APIs ha dado como resultado que, en 2019, OWASP cree una versión del top 10 dedicada específicamente a la seguridad de las mismas.



https://owasp.org/API-Security/editions/2019/en/0x00-header/

Vulnerabilidades

Las vulnerabilidades en servicios web REST, como inyección de código, falta de autenticación y exposición de datos sensibles, representan riesgos críticos que requieren medidas de seguridad para mitigarlos eficazmente.

Vulnerabildades: Herramienta de Analisis

Api vulnerable desarrollada en flask para el uso educativo.

Endpoints

Action	Path	Details
GET	/createdb	Creates and populates the database with dummy data
GET	/	VAmPI home
GET	/users/v1	Displays all users with basic information
GET	/users/v1/_debug	Displays all details for all users
POST	/users/v1/login	Login to VAmPI
GET	/users/v1/{username}	Displays user by username
DELETE	/users/v1/{username}	Deletes user by username (Only Admins)
GET	/books/v1	Retrieves all books
POST	/books/v1	Add new book
GET	/books/v1/{book}	Retrieves book by title along with secret

Vulnerabilidades

Objetos

Las APIs utilizan estructuras de datos conocidas como objetos. Las mismas representar a una entidad específica o un recurso dentro del sistema, por ejemplo, un usuario, una transacción, etc.

Object Level Authorization

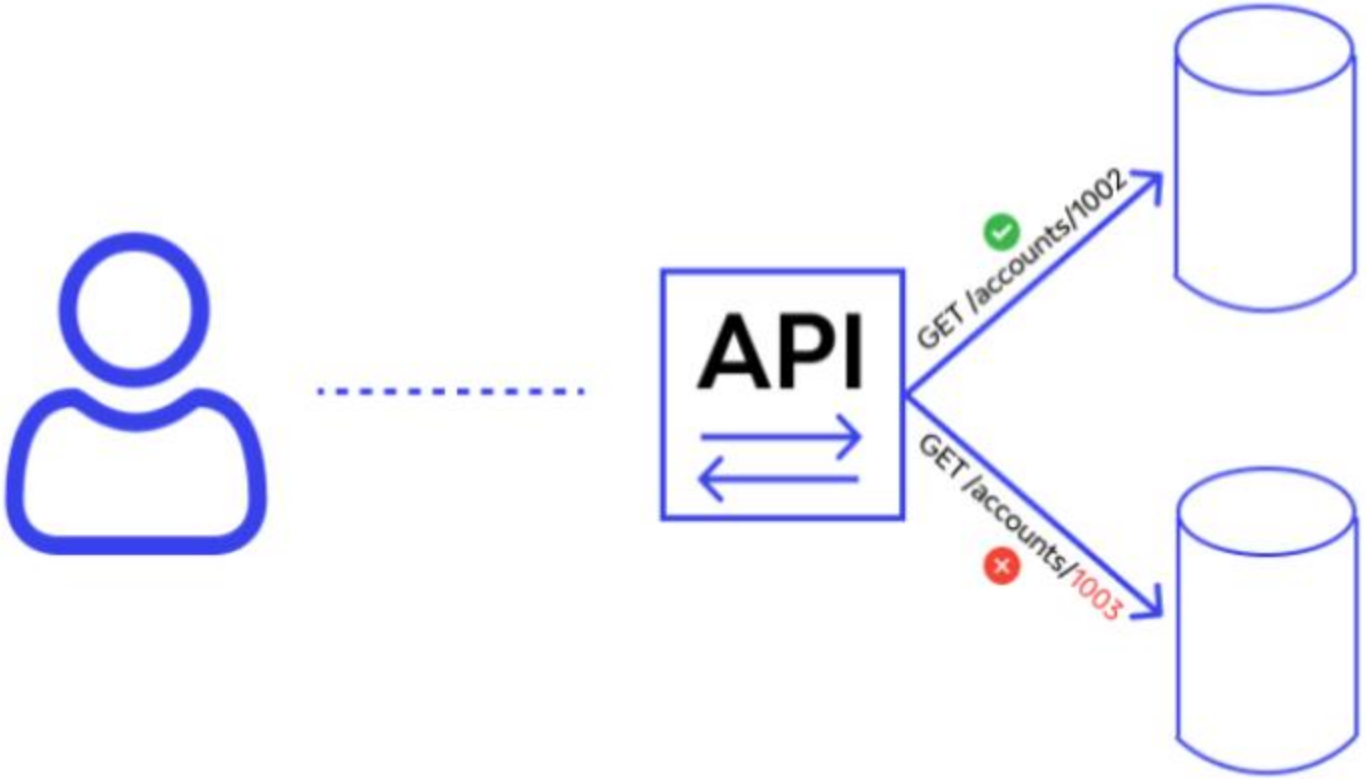
- Enfoque para controlar el acceso a nivel objeto y no a nivel de servicio o ruta de la API.
- Permite una granularidad más fina en el otorgamiento de permisos.

Vulnerabilidades

Broken Object Level Authorization

Vectores de Ataque	Debilidades de Seguridad	Impacto
Los atacantes pueden explotar los endpoints de la API que son vulnerables al manipular el ID de un objeto que se envía dentro de la solicitud. Esto puede dar lugar a un acceso no autorizado a datos sensibles.	Los mecanismos de acceso en las aplicaciones modernas son complejos. Incluso si se implementa una infraestructura adecuada, los desarrolladores podrían olvidar utilizar estas comprobaciones antes de acceder a un objeto sensible.	El acceso no autorizado puede resultar en la divulgación de datos a partes no autorizadas, pérdida de datos o manipulación de datos.

Broken Object Level Authorization



Broken Object Level Authorization: Ejemplo

(1) Login en la API Rest con un Access Token con el Usuario "name1"

```
┌──(root💀kali)-[~]
└─# curl -X POST VamAPI.com:5001/users/v1/login -H "Content-Type:application/json" -d '{"username":"name1", "password":"pass1"}'
{"auth_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE2ODU4OTU1NzIsImlhdCI6MTY4NTg5NTUxMiwic3ViIjoibmFtZTEifQ.7vZ_NF2cihFXT
y logged in.", "status": "success"}
```

(2) Listado de los libros existentes

```
┌──(root💀kali)-[~]
└─# curl VamAPI.com:5001/books/v1
{
  "Books": [
    {
      "book_title": "bookTitle16",
      "user": "name1"
    },
    {
      "book_title": "bookTitle97",
      "user": "name2"
    },
    {
      "book_title": "bookTitle22",
      "user": "admin"
    }
  ]
}
```

Broken Object Level Authorization: Ejemplo

(3) Obtengo el secret del libro que soy dueño

```
┌──(root💀kali)-[~]
└─# curl VamAPI.com:5001/books/v1/bookTitle16 -H 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE2ODU4OTYzMTksImlhdCI6M
S99SjsYguBVlLxvHYV2AgDq7xKuNm8'
{"book_title": "bookTitle16", "owner": "name1", "secret": "secret for bookTitle16"}
```

(4) Obtengo el secret de un libro que no me pertenece

```
┌──(root💀kali)-[~]
└─# curl VamAPI.com:5001/books/v1/bookTitle97 -H 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE2ODU4OTYzMTksIm
S99SjsYguBVlLxvHYV2AgDq7xKuNm8'
{"book_title": "bookTitle97", "owner": "name2", "secret": "secret for bookTitle97"}
```

(5) Me valida que el libro pertenezca al usuario

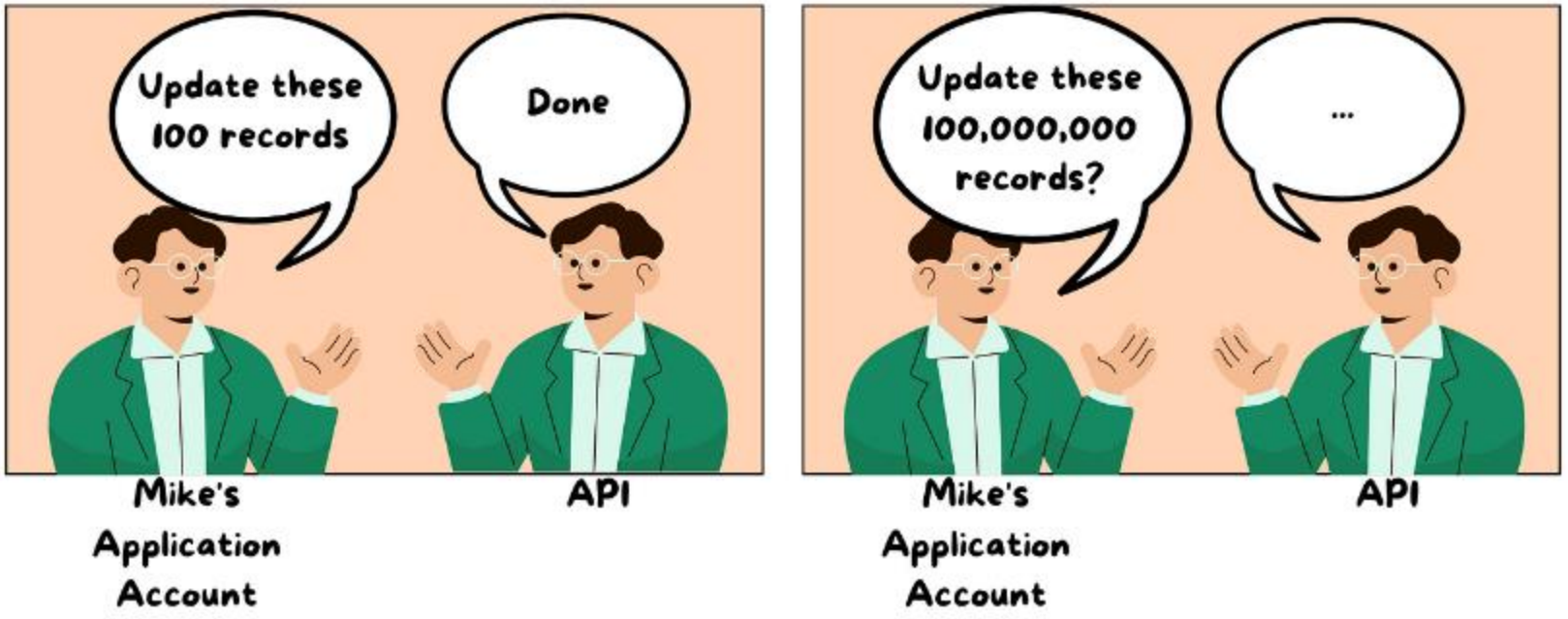
```
┌──(root💀kali)-[/home/kali/Desktop/TP/prometheus-2.18.1.linux-amd64]
└─# curl VamAPI.com:5000/books/v1/bookTitle97 -H "Authorization: Bearer ${token}"
{ "status": "fail", "message": "Book not found!"}
```

Vulnerabilidades

Unrestricted Resource Consumption

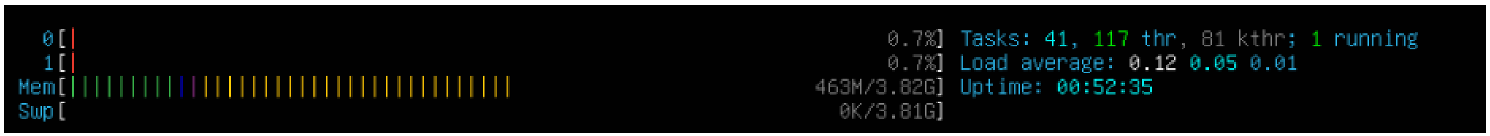
Vectores de Ataque	Debilidades de Seguridad	Impacto
La explotación requiere solicitudes simples a la API. Se pueden realizar múltiples solicitudes concurrentes desde una sola computadora local o mediante el uso de recursos de computación en la nube.	Es común encontrar APIs que no limitan las interacciones del cliente o el consumo de recursos.	La explotación puede llevar a una Denegación de Servicio (DoS) debido a la falta de recursos, pero también puede dar lugar a un aumento en los costos operativos.

Unrestricted Resource Consumption

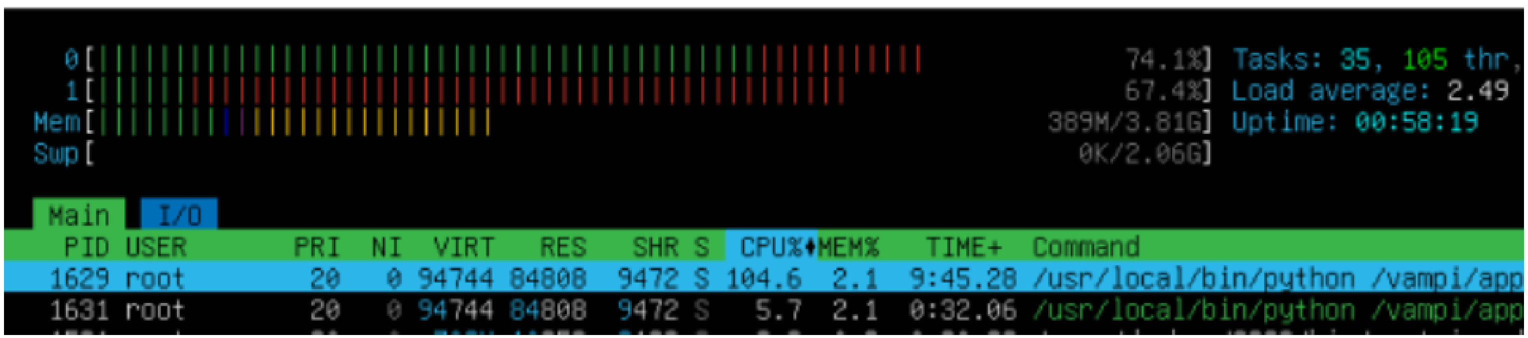


Unrestricted Resource Consumption: Ejemplo

Servidor con consumo de recursos normal



Servidor Durante la Ejecución de multiples request

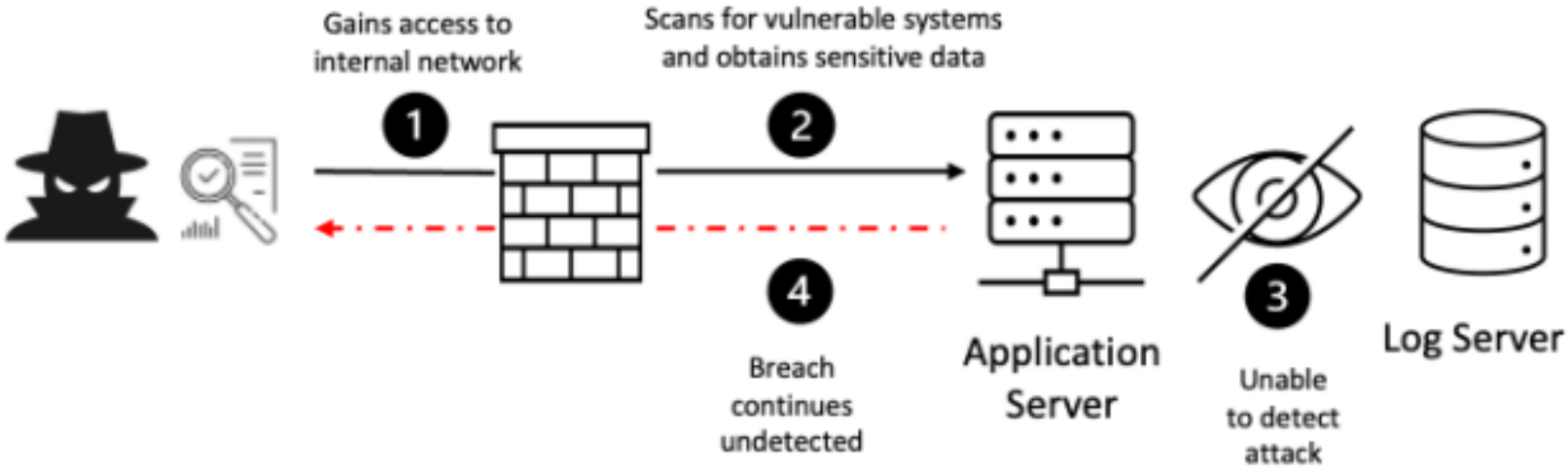


Vulnerabilidades

Insufficient Logging & Monitoring

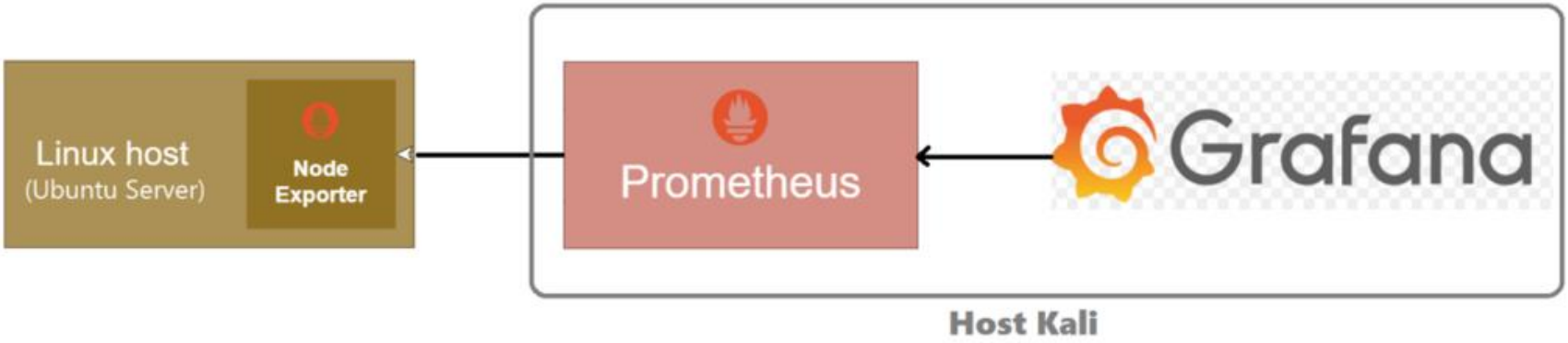
Vectores de Ataque	Debilidades de Seguridad	Impacto
Los atacantes aprovechan la falta de registros y monitoreo para abusar de sistemas sin ser notados.	Con registros y monitoreo insuficientes, es casi imposible rastrear actividades sospechosas y responder a ellas de manera oportuna.	Sin visibilidad sobre las actividades maliciosas en curso, los atacantes tienen suficiente tiempo para comprometer completamente los sistemas.

Insufficient Logging & Monitoring



Insufficient Logging & Monitoring: Ejemplo

Arquitectura



Prometheus

Targets

All Unhealthy

node_exporter (1/1 up) show less

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://192.168.0.21:9100/metrics	UP	instance="192.168.0.21:9100" job="node_exporter"	1.568s ago	29.88ms	

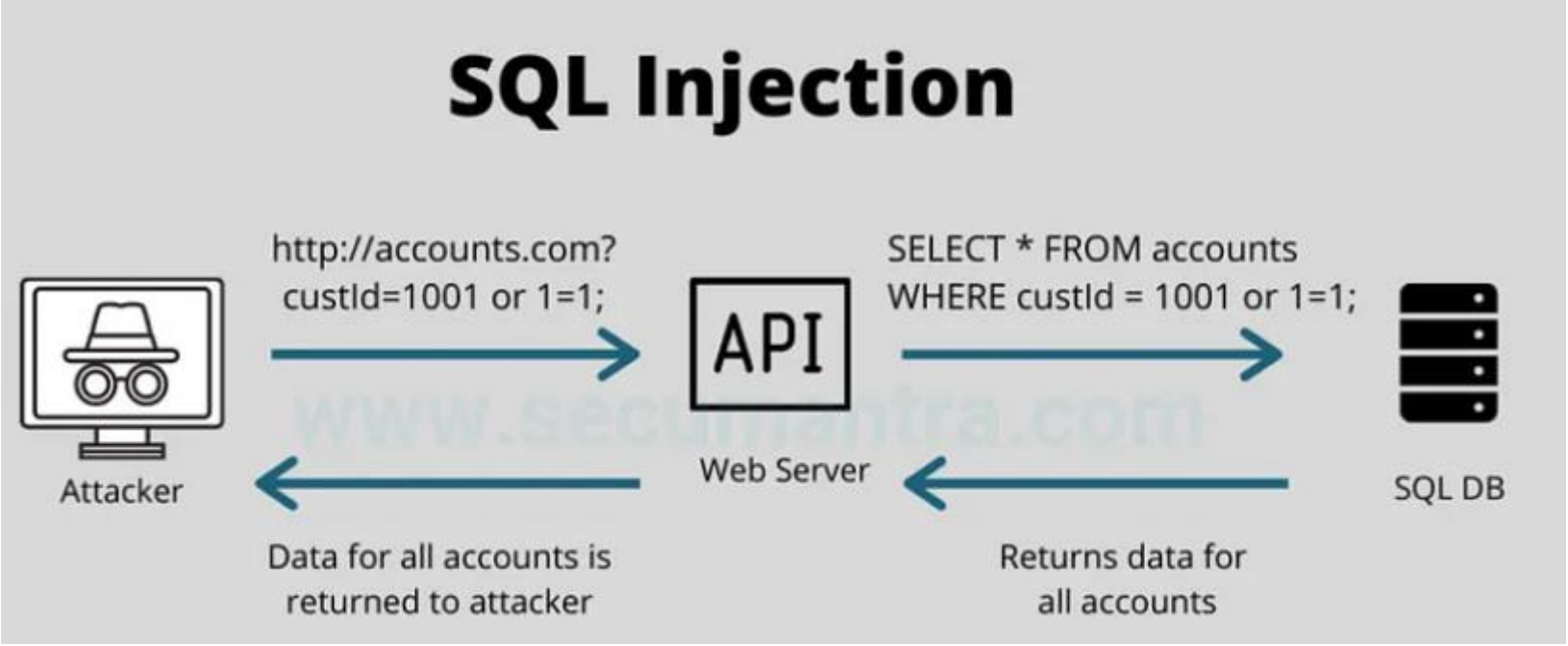
Insufficient Logging & Monitoring: Ejemplo

Vulnerabilidades

Injection

Vectores de Ataque	Debilidades de Seguridad	Impacto
Los atacantes alimentarán la API con datos maliciosos a través de cualquier vector de inyección disponible esperando que estos datos sean enviados a un intérprete.	No sanitizar los datos ingresados (entradas, parámetros, etc)	Las inyecciones pueden dar lugar a la divulgación de información y pérdida de datos. También pueden llevar a un ataque de denegación de servicio (DoS).

Injection


SQL Injection
http://accounts.com?
custId=1001 or 1=1;
SELECT * FROM accounts
WHERE custId = 1001 or 1=1;
API
Attacker
Web Server
SQL DB
Data for all accounts is
returned to attacker
Returns data for
all accounts

Injection: Ejemplo

Notamos la vulnerabilidad en la inyección

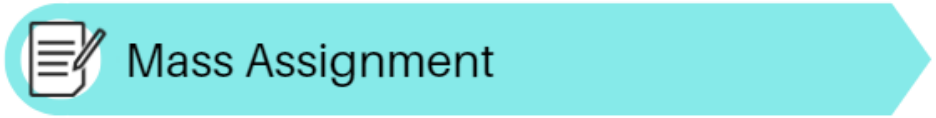
```
┌──(kali㉿kali)-[~]
└─$ curl "VamAPI.com:5001/users/v1/'"
<!doctype html>
<html lang=en>
  <head>
    <title>sqlalchemy.exc.OperationalError: (sqlite3.OperationalError) unrecognized token: "''''"
[SQL: SELECT * FROM users WHERE username = '''']
```

Usando un poco de SQL...

```
SELECT * FROM users WHERE username = '' and '1'='0' UNION select username,
   password from users where username='name2'
```

```
┌──(kali㉿kali)-[~]
└─$ curl "VamAPI.com:5001/users/v1/'%20%20and%20'1'='0'%20union%20select%20username,username,pas
sword,password,password%20from%20users%20where%20username='name2'--"
{"username": "name2", "email": "pass2"}
```

Vulnerabilidades



Vectores de Ataque	Debilidades de Seguridad	Impacto
La explotación generalmente requiere comprender la lógica.	Funciones que automáticamente enlazan la entrada del cliente en variables de código y objetos internos.	La explotación puede llevar a la escalada de privilegios, manipulación de datos, eludir mecanismos de seguridad y más.

Mass Assignment

Alice

LEGIT

POST /api/users/new

{"username":"Alice", "pass":"123456"}

Attacker

MALICIOUS

POST /api/users/new

{"username":"Alice","pass":"123456", "role":"admin"}

Mass Assignment: Ejemplo

```
┌──(kali㉿kali)-[~]
└─$ curl -X POST VamAPI.com:5001/users/v1/register -H "Content-Type:application/json" -d '{"user
name":"pepito123", "password":"pepito123","email":"pepito@gmail", "admin":true}'
{"message": "Successfully registered. Login to receive an auth token.", "status": "success"}
```

```
┌──(kali㉿kali)-[~]
└─$ curl VamAPI.com:5001/users/v1/_debug
{
  "users": [
    {
      "admin": false,
      "email": "mail1@mail.com",
      "password": "pass1",
      "username": "name1"
    },
    {
      "admin": true,
      "email": "pepito@gmail",
      "password": "pepito123",
      "username": "pepito123"
    }
  ]
}
```

Vulnerabilidades

Broken Authentication

Vectores de Ataque	Debilidades de Seguridad	Impacto
El mecanismo de autenticación es un objetivo fácil para los atacantes, ya que está expuesto a todos. Aunque se pueden requerir habilidades técnicas más avanzadas para explotar algunos problemas de autenticación.	Los conceptos erróneos de ingenieros de software y seguridad con respecto a los límites de autenticación y la complejidad inherente de la implementación hacen que los problemas de autenticación sean frecuentes.	Los atacantes pueden obtener control completo de las cuentas de otros usuarios en el sistema, leer sus datos personales y realizar acciones sensibles en su nombre.

Vulnerabilidades

Excessive Data Exposure

Vectores de Ataque	Debilidades de Seguridad	Impacto
La explotación de la Exposición Excesiva de Datos es simple y suele llevarse a cabo mediante el análisis del tráfico para examinar las respuestas de la API, buscando la exposición de datos sensibles que no deberían ser proporcionados al usuario	Las API dependen de los clientes para llevar a cabo la filtración de datos. Dado que las APIs se utilizan como fuentes de datos, a veces los desarrolladores intentan implementarlas de manera genérica sin considerar la sensibilidad de los datos expuestos.	La Exposición Excesiva de Datos comúnmente conduce a la exposición de datos sensibles.

Vulnerabilidades

Broken Function Level Authorization

Vectores de Ataque	Debilidades de Seguridad	Impacto
La explotación requiere que el atacante envíe llamadas legítimas a puntos finales de la API a los que no deberían tener acceso. Estos puntos finales podrían estar expuestos a usuarios anónimos o usuarios regulares no privilegiados.	Las comprobaciones de autorización para una recurso se gestiona a través de la configuración y, en ocasiones, a nivel de código. Implementar comprobaciones adecuadas puede ser una tarea confusa, ya que las aplicaciones modernas pueden contener muchos tipos de roles o grupos y una jerarquía de usuarios compleja.	Estas fallas permiten a los atacantes acceder a funcionalidades no autorizadas. Las funciones administrativas son objetivos clave para este tipo de ataque.

Vulnerabilidades

Security Misconfiguration

Vectores de Ataque	Debilidades de Seguridad	Impacto
Los atacantes a menudo intentarán encontrar fallas sin parchear, puntos finales comunes o archivos y directorios desprotegidos para obtener acceso no autorizado o conocimiento del sistema.	La configuración de seguridad incorrecta puede ocurrir en cualquier nivel de la pila de la API, desde el nivel de red hasta el nivel de la aplicación.	Las configuraciones de seguridad incorrectas no solo pueden exponer datos sensibles de usuarios, sino también detalles del sistema que pueden comprometer al servidor.

Vulnerabilidades

Improper Assets Management

Vectores de Ataque	Debilidades de Seguridad	Impacto
Las versiones antiguas de las API suelen no estar actualizadas y son una forma fácil de comprometer sistemas sin tener que lidiar con mecanismos de seguridad.	La documentación desactualizada dificulta la búsqueda y/o corrección de vulnerabilidades. La falta de inventario de activos y estrategias de retiro conduce a la ejecución de sistemas sin parches, lo que resulta en la filtración de datos sensibles	Los atacantes pueden obtener acceso a datos sensibles o incluso tomar el control del servidor a través de versiones antiguas y sin parches de la API que estén conectadas a la misma base de datos.