Ministerio de
TECNOLOGÍAS
DE LA INFORMACIÓN
Y COMUNICACIÓN

CERT-PY

SEGURIDAD EN API
Un Mundo Mágico

Mg. Ing. RAÚL B. NETTO

Mes de la concienciacion en
ciberseguridad
Octubre 2022

Hola!

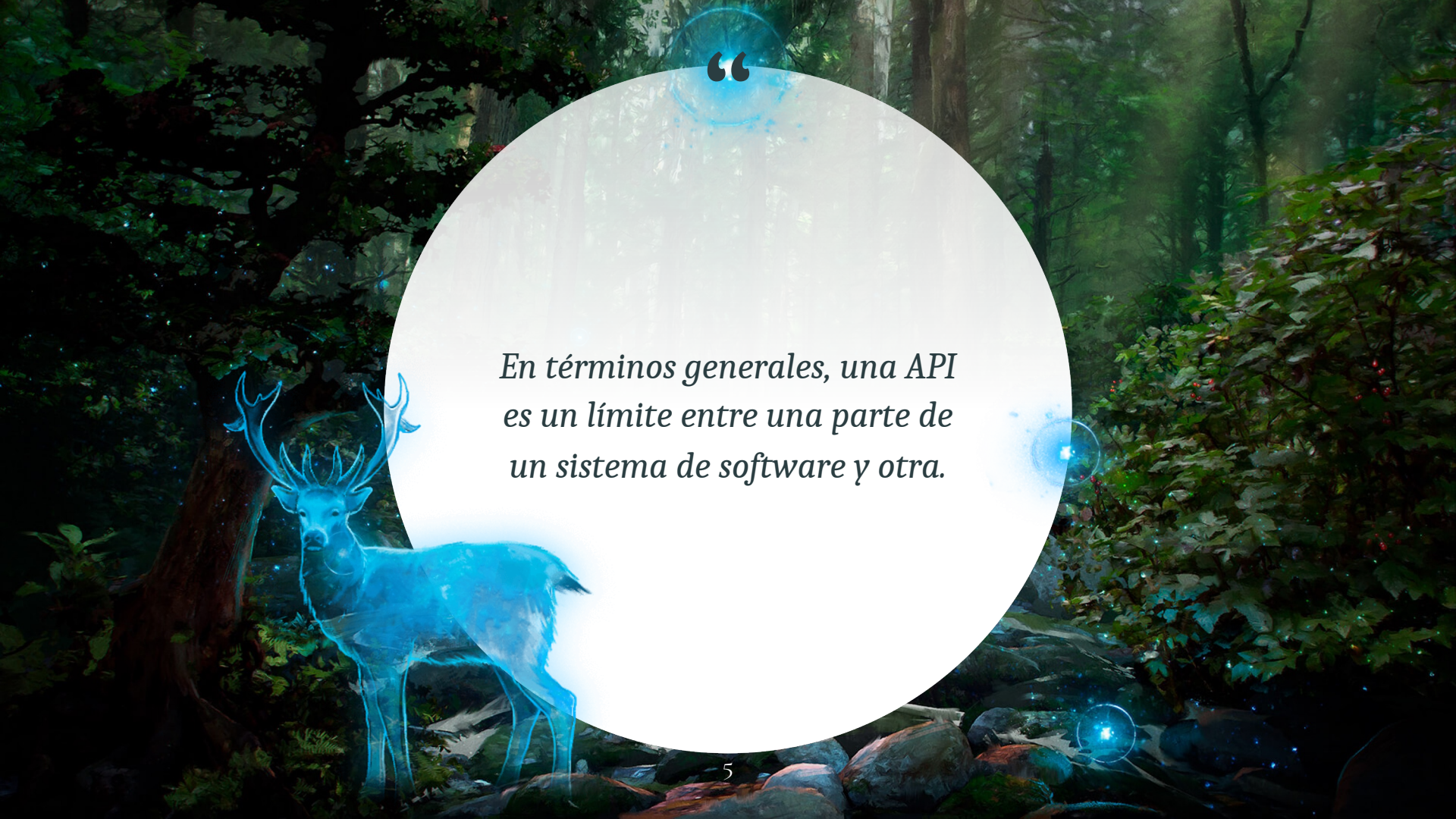
Yo soy Raúl B. Netto

Me pueden encontrar en:
En Twitter @rbnetto13
rbenitez@{mitic.gov.py, cert.gov.py}

Qué veremos ?

- Definición de API
- Mecanismos de Seguridad en API
- API REST
- OAuth 2.0 y OpenID Connect
- OWASP API Security Project
- HACKING EN VIVO!!!!

Las interfaces de programación de aplicaciones (API) están en todas partes.

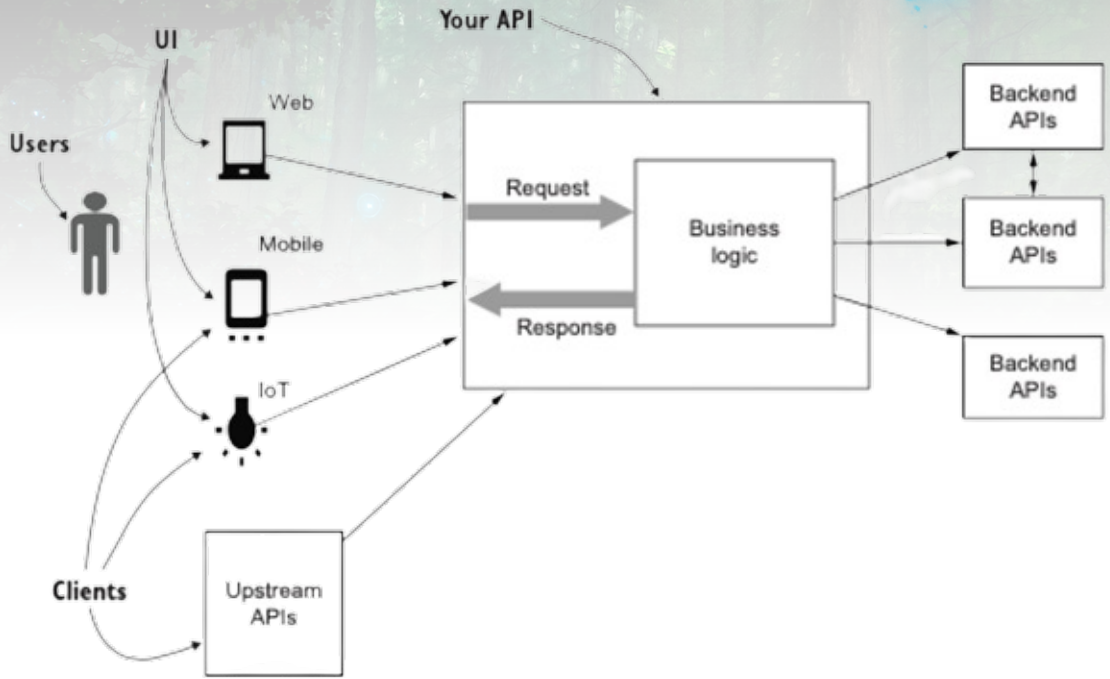


En términos generales, una API es un límite entre una parte de un sistema de software y otra.

API vs UI

Lo que distingue a una API de una interfaz de usuario es que una API está diseñada explícitamente para que otro software pueda interactuar fácilmente con ella, mientras que una interfaz de usuario está diseñada para que un usuario interactúe directamente con ella fácilmente.

Que es un API ?


Your API
UI
Web
Users
Mobile
IoT
Clients
Upstream APIs
Request
Response
Business logic
Backend APIs
Backend APIs
Backend APIs

Estilos de API I

RPC - Remote procedure call

Llamadas a procedimientos normales. Usan formatos binarios compactos para los mensajes. Se requieren stubs. Ej: grpc.io o SOAP

RMI - Remote Method Invocation

Utiliza técnicas orientadas a objetos para permitir que los clientes llamen a métodos en objetos remotos como si fueran locales. Ej: CORBA, EJBs (Java Beans)

REST - Representational State Transfer

Enfatizan los formatos de mensaje estándar y una pequeña cantidad de operaciones genéricas para reducir el acoplamiento entre un cliente y una API específica. Ej: HTTP y WWW

Estilos de API II

- Algunas API se ocupan principalmente de la consulta y el filtrado eficientes de grandes conjuntos de datos, como las bases de datos SQL o el marco GraphQL de Facebook (https://graphql.org).
- La API a menudo solo proporciona unas pocas operaciones y un lenguaje de consulta complejo le permite al cliente un control significativo sobre qué datos se devuelven.

API REST

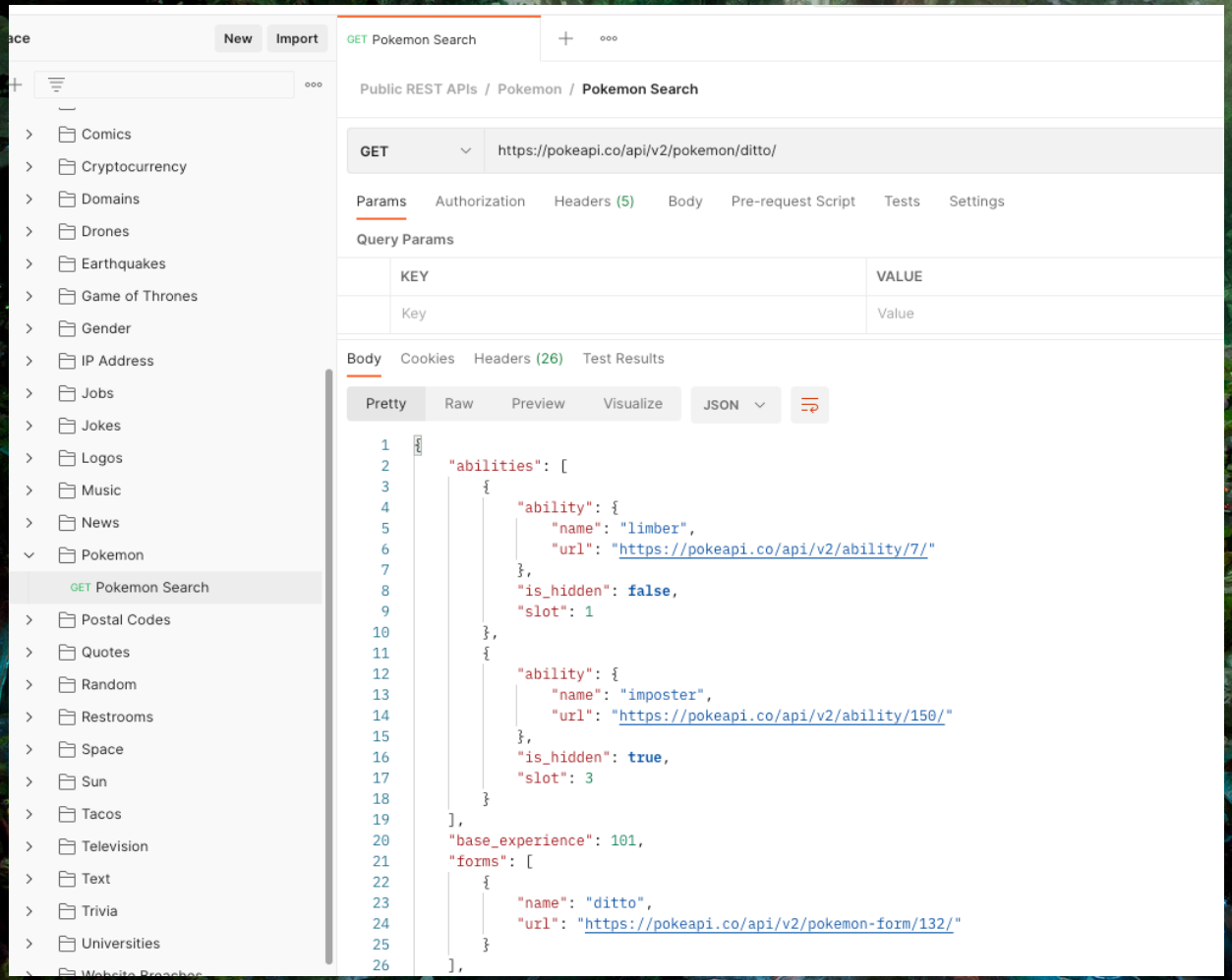
Las API REST proporcionan interfaces simples y uniformes porque se pueden usar para hacer que los datos, el contenido, los algoritmos, los medios y otros recursos digitales estén disponibles a través de URL web. Es el mas popular.

¿Cómo funcionan las API REST?

Las API REST utilizan fundamentalmente “recursos.” Un recurso puede ser cualquier información que se pueda nombrar o enumerar. Los cuatro métodos de recursos son:

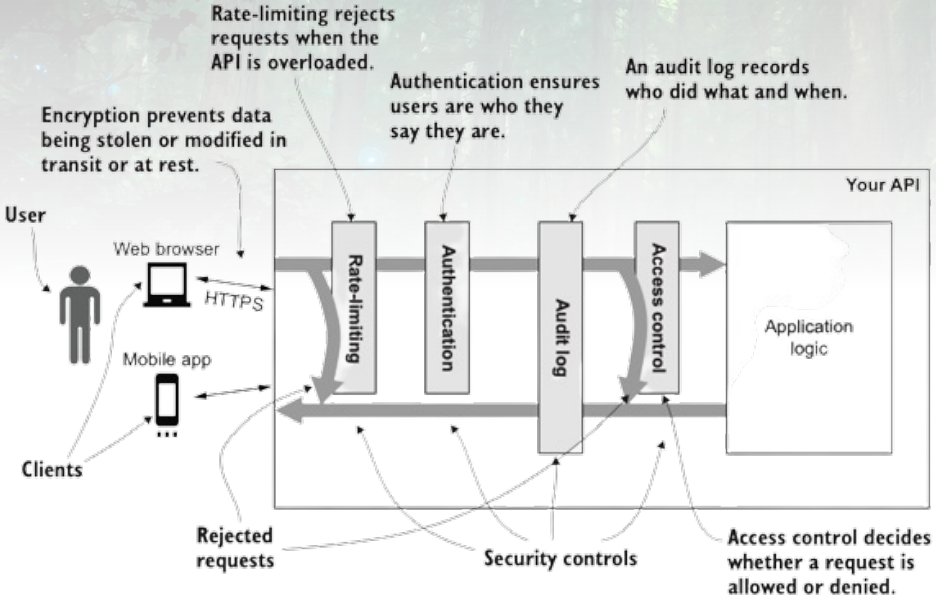
- **GET**: el servidor encuentre los datos que solicitó y se los devuelva.
- **PUT/PATCH**: el servidor actualizará una entrada en la base de datos.
- **POST**: el servidor crea una nueva entrada (o cambia de estado) en la base de datos.
- **DELETE**: el servidor elimina una entrada en la base de datos.

Mas info: https://blog.postman.com/rest-api-definition/

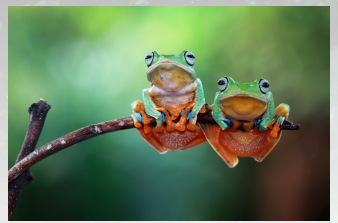


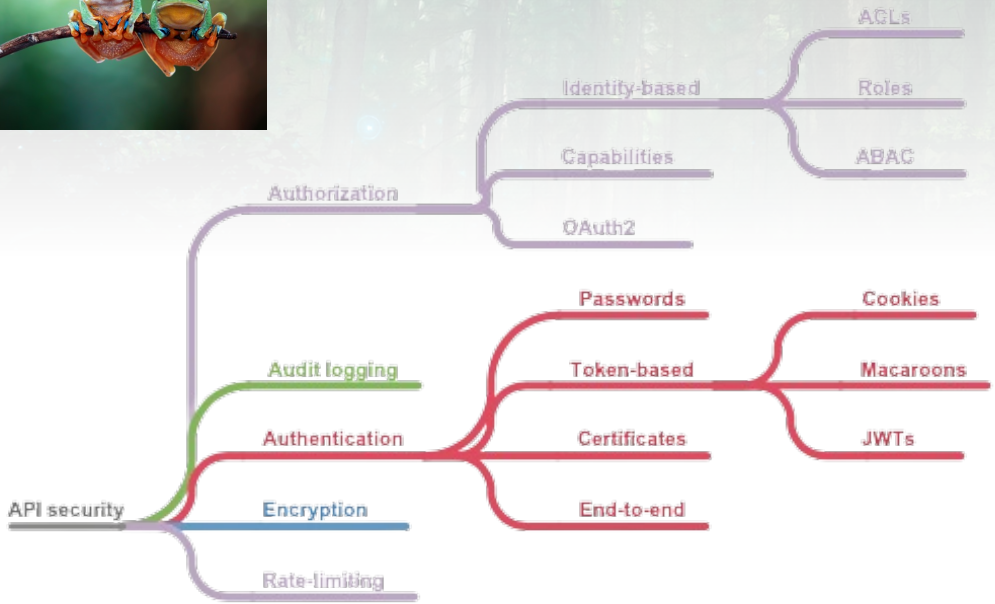
https://www.postman.com/cs-demo/workspace/public-rest-apis/collection/8854915-454a2dc7-dcbe-41cf-9bfa-da544fcd93a2?ctx=documentation

Mecanismo de Seguridad I



Mecanismo de Seguridad II





Token-Based Auth

La autenticación basada en token permite generar un token de larga duración que puede entregar al servicio de terceros en lugar de su contraseña.

Token-based Authentication

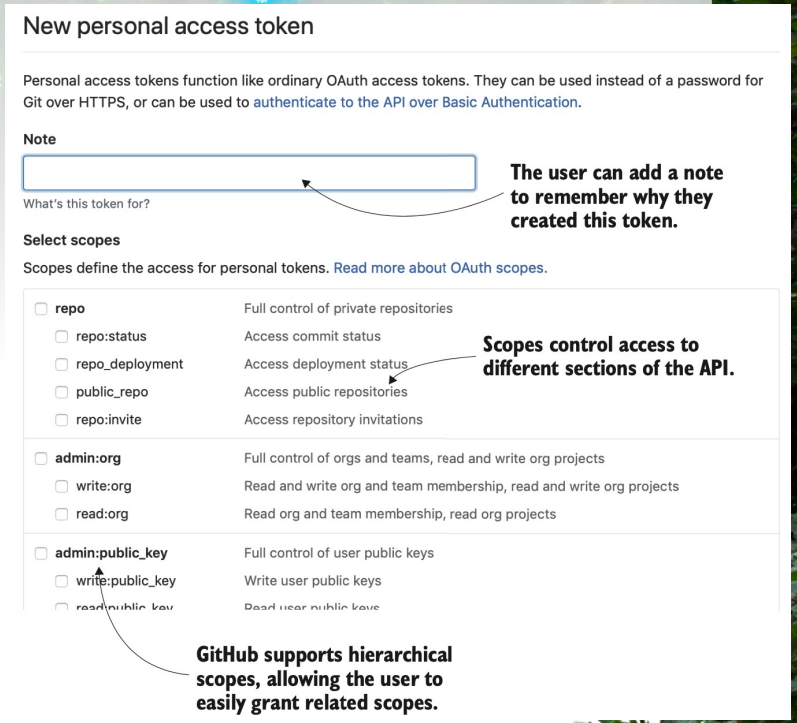
En la autenticación basada en tokens, las credenciales reales de un usuario se presentan una vez y luego se le otorga al cliente un token de corta duración. Un token suele ser una cadena breve y aleatoria que se puede usar para autenticar las llamadas a la API hasta que el token caduque.

Tokens de Alcance (Scoped tokens)

Los token permitne que se use solo dentro de un alcance (scope) bien definido.

Puede usar la etiqueta de alcance transacciones: permitir el acceso de lectura a las transacciones

y pago: crear para permitir la configuración de un nuevo pago desde una cuenta



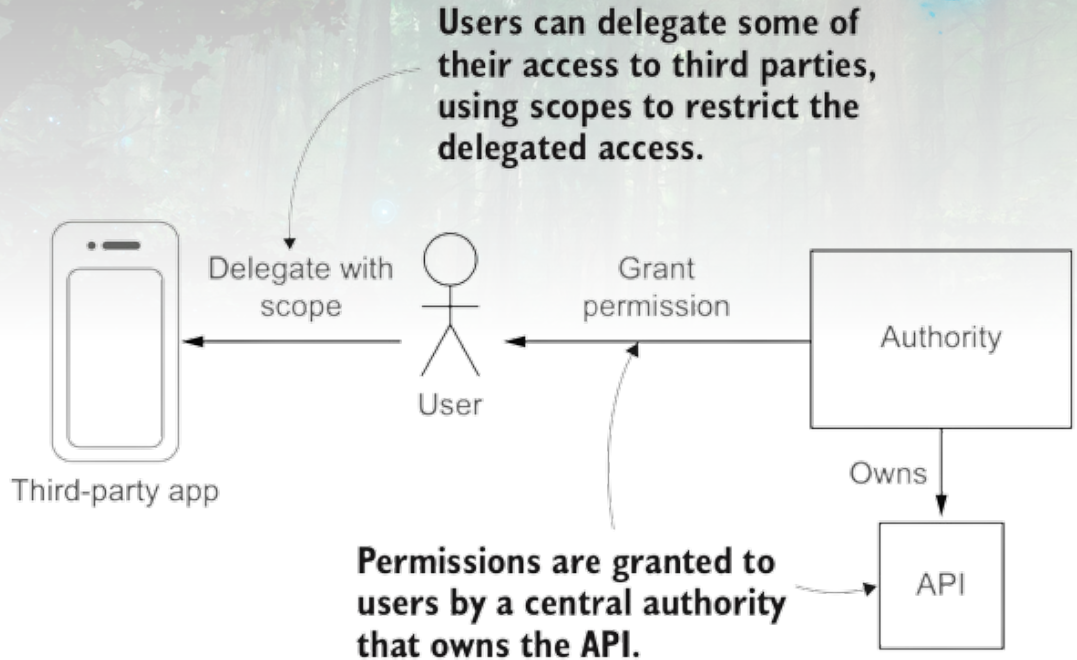
La diferencia entre alcances y permisos

La autoridad central controla por completo quién puede acceder a la API y qué se le permite hacer. Este es un ejemplo de **control de acceso obligatorio (MAC),** porque los usuarios no tienen control sobre sus propios permisos o los de otros usuarios.

Cuando un usuario delega parte de su acceso a una aplicación o servicio de un tercero, eso se conoce como **control de acceso discrecional (DAC)**, porque depende del usuario qué parte de su acceso concederá al tercero.

Mientras que los **alcances** se utilizan para la **delegación (DAC)**, los **permisos** se pueden utilizar para el **acceso obligatorio o discrecional**.

La diferencia entre alcances y permisos



JWT

Los tokens web JSON son un formato estándar para los tokens de seguridad autónomos. Un JWT consta de un conjunto de notificaciones sobre un usuario representado como un objeto JSON, junto con un encabezado que describe el formato del token. Los JWT están protegidos criptográficamente contra la manipulación y también se pueden cifrar.

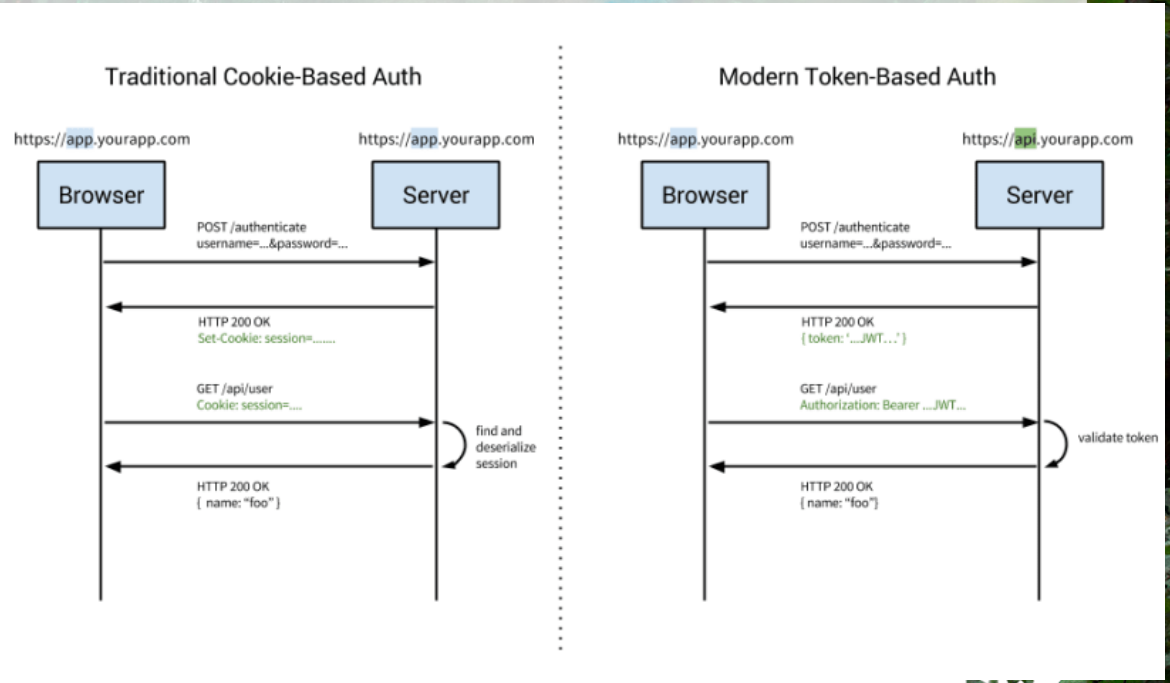
JWT

JWT proveen mas funcionalidades que los JSON Tokens:

- Un formato de encabezado estándar que contiene metadatos sobre el JWT, como qué MAC o algoritmo de cifrado se usó.
- Un conjunto de reclamos estándar que se pueden usar en el contenido JSON del JWT, con significados definidos, como exp para indicar el tiempo de vencimiento y sub para el asunto, tal como lo ha estado usando.
- Una amplia gama de algoritmos para autenticación y cifrado, así como firmas digitales y cifrado de clave pública que se tratan más adelante en este libro.

JWT vs COOKIES

- Ni los JWT ni las Cookies constituyen en sí mismos un mecanismo de autenticación.
- El primero sólo define un formato de token, y el segundo es un mecanismo de gestión de estado para las peticiones HTTP.



OAuth 2.0

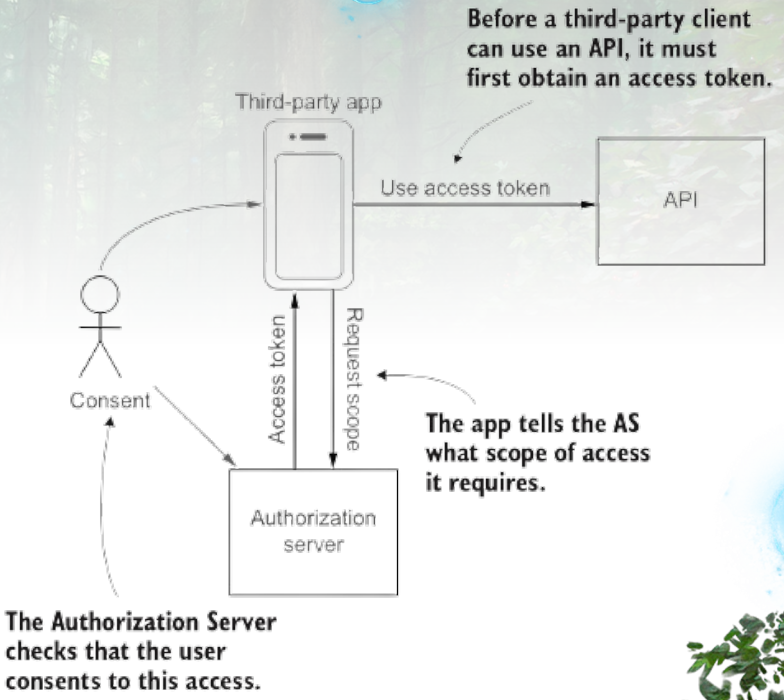
OAuth 2.0 es el protocolo estándar de la industria para la autorización.

OAuth 2.0

- Si bien permitir que sus usuarios creen manualmente tokens con alcance para aplicaciones de terceros es una mejora con respecto a compartir tokens sin alcance o credenciales de usuario, puede ser confuso y propenso a errores
- Es posible que un usuario no sepa qué ámbitos se requieren para que esa aplicación funcione y, por lo tanto, puede crear un token con muy pocos ámbitos, o tal vez delegar todos los ámbitos solo para que la aplicación funcione.
- Una mejor solución es que la aplicación solicite los ámbitos que requiere, y luego la API puede preguntar al usuario si está de acuerdo. Este es el enfoque adoptado por el protocolo de autorización delegada de OAuth2

OAuth 2.0

1. El servidor de autorización (AS) autentica al usuario y emite tokens a los clientes.
2. El usuario se conoce como el propietario del recurso (RO), porque normalmente son sus recursos a los que la aplicación de terceros intenta acceder.
3. La aplicación o el servicio de terceros se conoce como el cliente.
4. La API que aloja los recursos del usuario se conoce como servidor de recursos (RS).



OpenID

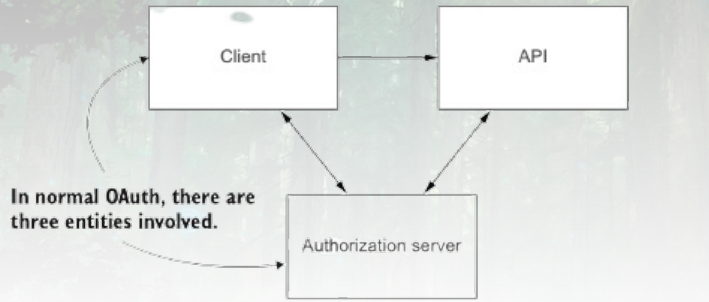
Una extensión de OAuth 2.0

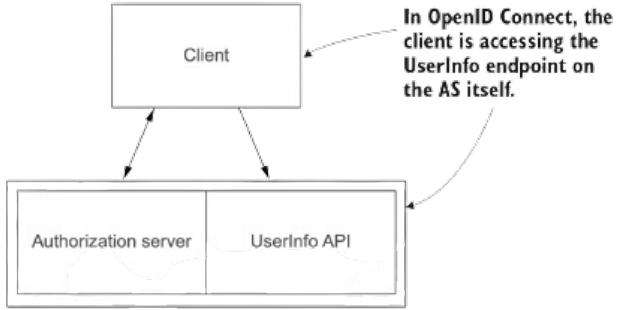
OpenID Connect (OIDC)

El conjunto de estándares OpenID Connect (OIDC) amplía OAuth2 con varias funciones:

- Una forma estándar de recuperar información de identidad sobre un usuario. El cliente puede acceder a un punto final de UserInfo para recuperar notificaciones de identidad como JSON mediante un token de acceso OAuth2 con alcances OIDC.
- Una forma para que el cliente solicite que el usuario se autentique incluso si tiene una sesión existente y solicitar que se autentique de una manera particular, como con la autenticación de dos factores (2FA). OAuth2 es principalmente un protocolo de acceso delegado, mientras que OIDC proporciona una autenticación completa.
- Extensiones para el manejo de sesión y cierre de sesión, lo que permite que los clientes sean notificados cuando un usuario cierra sesión en el AS, esto permite al usuario cerrar sesión en todos los clientes a la vez (cierre de sesión único).

OpenID Connect (OIDC)

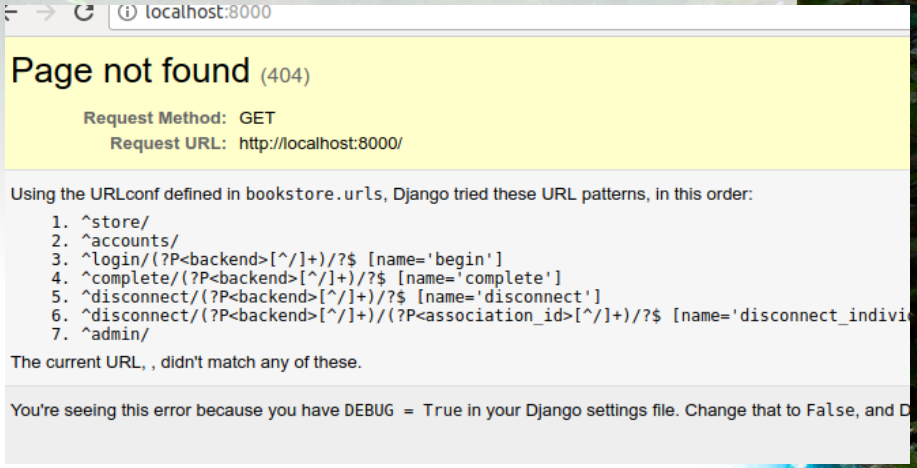




I'LL RUN A
"PENETRATION TEST"
memegenerator.net

Caso – Filtración de Datos por API I

- Recibimos un reporte de un API expuesta con lo que se estaban publicando datos de usuarios (nombre, apellido, num de telefono) sin autenticación . Miles…
- Indagando encontramos que el debug mode de django estaba activo.
- Pudiemos ver los endpoint y probarlos.
- …. Y logramos bloquear usuarios 😈 (eventualmente masivamente) ademas de otras cosas.



Caso – Filtración de Datos por API II – Qué se hizo mal ?

- Debug mode en producción !
- Era una API de un callcenter pero también proveía API para un aplicación móvil
- Las App Móvil mandaban los tokens pero no estaban siendo utilizando el backend. Cualquiera con acceso a la URL podía ver los datos y filtrarlos.
- No estaban bien definidos los API siendo que eran consumidos por dos servicios distintos, callcenter y app móvil.

Caso – Filtración de Datos por API III – Qué se hizo bién ?

- Luego de las pruebas, preparamos un informe técnico detallados de las vulnerabilidades encontradas y como solucionarlas.
- Nos contactamos con el équipo...

....Fueron muy positivos, y trataron de ayudar en todo momento a subsanar el problema 😃

- Lo solucionarón inmediatamente 😎

OWASP API Security Project

Top Ten OWASP API Security 2019

- Broken Object Level Authorization
- Broken User Authentication
- Excessive Data Exposure
- Lack of Resources & Rate Limiting
- Broken Function Level Authorization
- Mass Assignment
- Security Misconfiguration
- Injection
- Improper Assets Management
- Insufficient Logging & Monitoring

1. Falla en Autorización a nivel de objeto
2. Falla en Autenticación de usuario
3. Exposición excesiva de datos
4. Falta de recursos y rate limitng
5. Falla en la Función de Autorizacion
6. Asignación masiva
7. Configuración incorrecta de seguridad
8. Inyección
9. Gestión inadecuada de activos
10. Logs y monitoreo insuficientes

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

API1:2019 Broken object level authorization

Los atacantes sustituyen la ID de su propio recurso en la llamada a la API con una ID de un recurso que pertenece a otro usuario.

Como prevenir

- Implemente verificaciones de autorización con políticas de usuario y jerarquía.
- No confíe en las identificaciones que envía el cliente. Utilice ID almacenados en el objeto de sesión en su lugar.
- Verifique la autorización para cada solicitud de cliente para acceder a la base de datos.
- Utilice UUID aleatorios que no se puedan adivinar.

API2:2019 Broken User Authentication

La autenticación API mal implementada permite a los atacantes asumir las identidades de otros usuarios.

Como prevenir

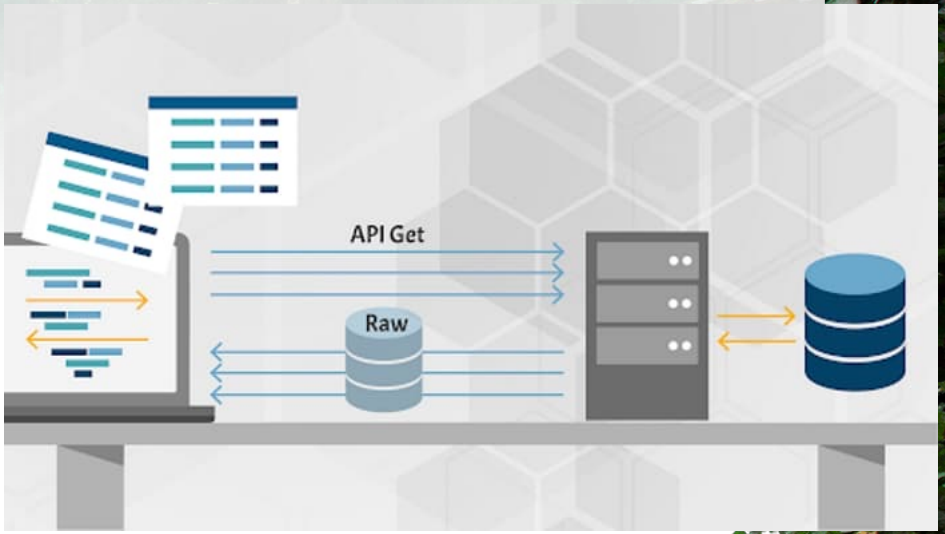
- Compruebe todas las formas posibles de autenticarse en todas las API. Incluyendo “reset” de contraseña y los enlaces únicos.
- Utilice la autenticación estándar, la generación de tokens y MFA.
- Use tokens de acceso de corta duración TOTP.
- Autentica tus aplicaciones.
- Use límites de velocidad estrictos para la autenticación e implemente verificaciones de contraseñas débiles.

API3:2019 Excessive Data Exposure

La API puede exponer muchos más datos de los que el cliente necesita legítimamente.

Como prevenir

- ¡Nunca confíe en el cliente para filtrar datos!
- Revise todas las respuestas de la API y adáptelas
- Defina cuidadosamente los esquemas para todas las respuestas de la API, incluyendo errores.
- Identificar todos los datos sensibles (PII) y justificar su uso.
- Haga cumplir las comprobaciones de respuesta para evitar fugas accidentales de datos o excepciones.



API4:2019 Lack of Resources & Rate Limiting

La API no está protegida contra una cantidad excesiva de llamadas o tamaños de carga útil o payload.

Como prevenir

- Defina la limitación de velocidad adecuada.
- Limite los tamaños de carga útil.
- Adapte la limitación de velocidad para que coincida con los métodos API, los clientes o las direcciones que necesitan o se les debe permitir obtener.
- Agregue controles en las relaciones de compresión.
- Defina límites para los recursos del contenedor.

API5:2019 Broken Function Level Authorization

La API se basa en el cliente para usar las API de nivel de usuario o nivel de administrador, según corresponda. Los atacantes descubren los métodos API de administración "ocultos" y los invocan directamente..

Como prevenir

- No confíe en el cliente para imponer el acceso de administrador.
- Denegar todos los accesos de forma predeterminada.
- Solo permita operaciones a usuarios que pertenezcan al grupo o rol apropiado.
- Diseño adecuado y autorización de prueba.

API6:2019 Mass Assignment

La API toma los datos que proporciona el cliente y los almacena sin el filtrado adecuado para las propiedades incluidas en la lista blanca. Los atacantes pueden intentar adivinar las propiedades de los objetos .

- NodeJS:

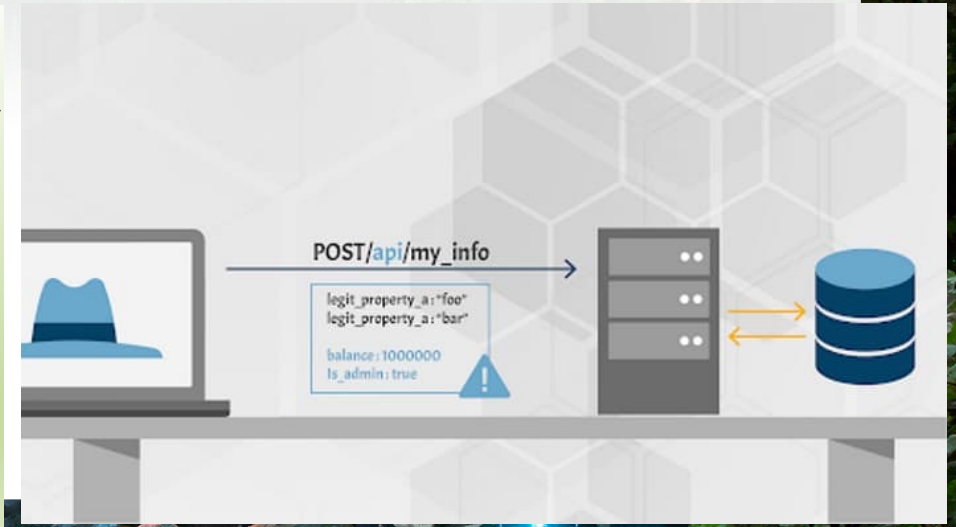
```
var user = new User(req.body);
user.save();
```

- Rails:

```
@user = User.new(params[:user])
```

Como prevenir

- No vincule automáticamente los datos entrantes y los objetos internos. Defina todos los parámetros.
- Use la propiedad readOnly=true en esquemas de objetos que se pueden leer por las API pero que nunca deben modificarse.
- Defina los esquemas, tipos y patrones que aceptará en las solicitudes en tiempo de diseño y aplíquelos en tiempo de ejecución.

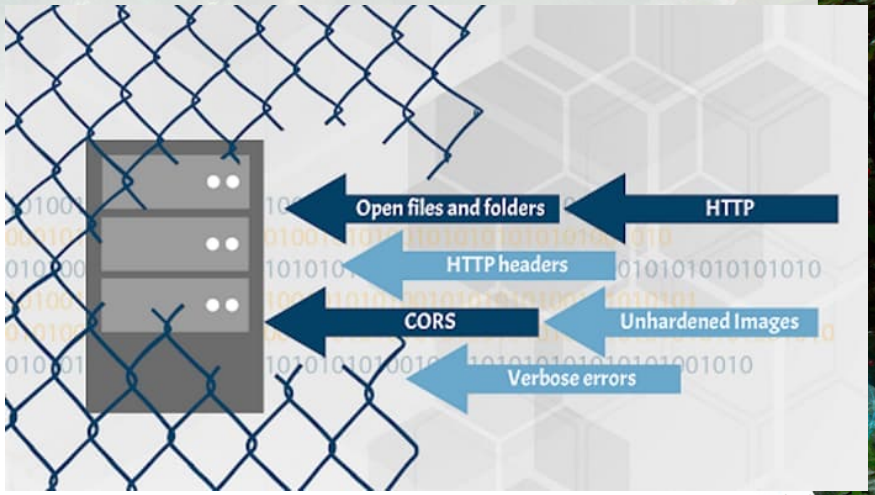


API7:2019 Security Misconfiguration

La mala configuración de los servidores API permite que los atacantes los exploten.

Como prevenir

- Establezca procesos repetibles de hardening y parcheo.
- Automatice la localización de fallas de configuración (escaneos).
- Deshabilite las funciones innecesarias.
- Restringir el acceso administrativo (Internet).
- Defina y aplique todas las salidas, incluidos los errores.

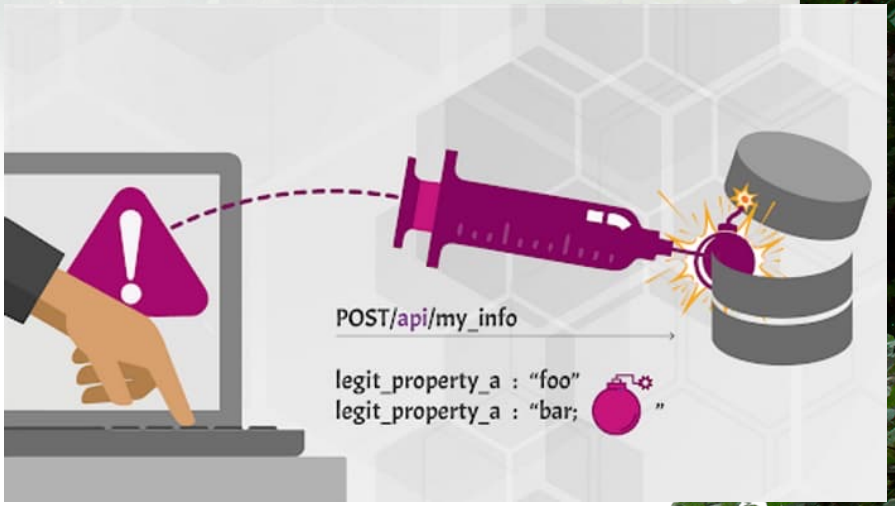


API8:2019 Injection

Los atacantes construyen llamadas API que incluyen SQL, NoSQL, LDAP, OS u otros comandos que la API o el backend detrás de ella ejecutan a ciegas.

Como prevenir

- Nunca confíes en los consumidores de tu API, incluso si son internos.
- Defina estrictamente todos los datos de entrada, como esquemas, tipos y patrones de cadenas, y aplíquelos en tiempo de ejecución.
- Valide, filtre y sanee todos los datos entrantes.
- Defina, limite y haga cumplir los resultados de la API para evitar fugas de datos.

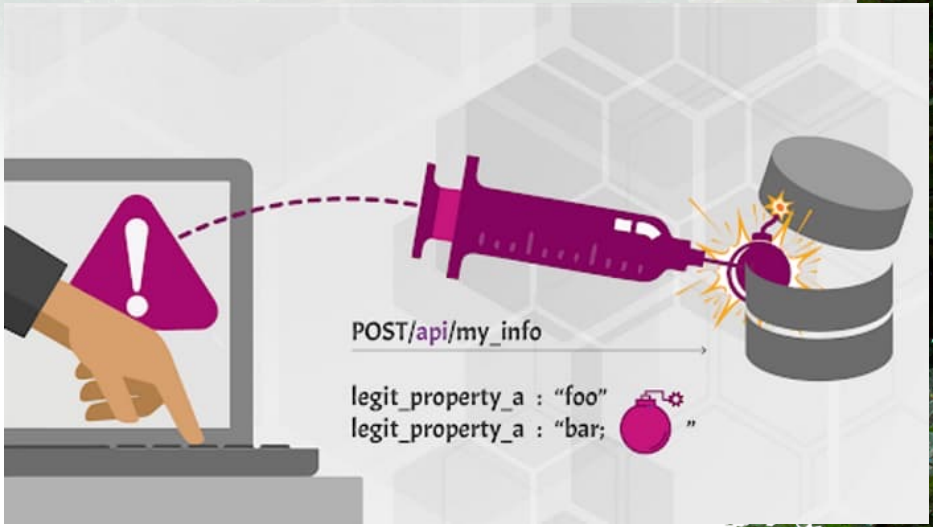


API9:2019 Improper Assets Management

Los atacantes encuentran versiones de la API que no son de producción que no están tan bien protegidas como la API de producción y las utilizan para lanzar sus ataques.

Como prevenir

- Mantenga un inventario actualizado de todos los hosts de API.
- Limite el acceso a lo que no deba ser público. Limite y segregué el acceso a los datos de producción y de no producción.
- Implemente controles externos adicionales, como firewalls API, VPN, WAF.
- Retirar las versiones antiguas de las API o sin las correcciones de seguridad.
- Implemente autenticación estricta, redireccionamientos, CORS, etc.



API10:2019 Insufficient Logging & Monitoring

La falta de registros, monitoreo y alertas adecuados permite que los ataques y los atacantes pasen desapercibidos.

Como prevenir

- Registre los intentos fallidos, el acceso denegado, las fallas de validación de entrada o cualquier falla en las verificaciones de políticas de seguridad y protejalos como información altamente confidencial.
- Asegúrese de que los registros estén formateados.
- Incluya suficientes detalles para identificar a los atacantes.
- Evite tener datos confidenciales en los registros.
- Integre con SIEM y otros herramientas de monitoreo.

DEMO

https://portswigger.net/web-security/oauth/lab-oauth-authentication-bypass-via-oauth-implicit-flow

Herramientas Útiles

- Postman
- Scripting (python, ruby, etc)
- curl
- (Pentesting) Burp Suite
- (Devs) Auth0 services

Biografía

- https://apisecurity.io/encyclopedia/content/owasp/api1-broken-object-level-authorization
- https://owasp.org/www-project-api-security/
- https://blog.postman.com/rest-api-examples/
- https://programacionymas.com/blog/jwt-vs-cookies-y-sesiones
- https://portswigger.net
- “API Security in Action ” de Neil Madden
- “Advanced API Security” de Prabath Siriwardena
- Gracias SlidesCarnival por el template de la presentación.

SCAN ME

Gracias!
Preguntas?

Me pueden encontrar en:
En Twitter @rbnetto13
rbenitez@{mitic.gov.py, cert.gov.py}