

Five Deadly Sins When Using Keycloak for SSO

Abdessamad TEMMAR

First OAuth pentest

2014

Keycloak Pentest

2017

Joined the defensive side

2019

What This Talk is About ?

- Common issues when using OAuth/Keycloak
- Real-world OAuth abuse scenarios
- The future of OAuth

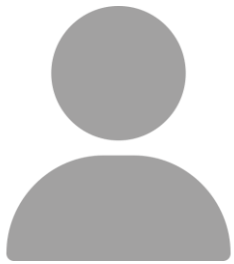
Whoami ?

- Abdessamad TEMMAR
- Application Security Engineer
- Ex-full time Pentester
- Certified CEH / CEI / OSCP
- OWASP Contributor
- Maintainer KC Academy !

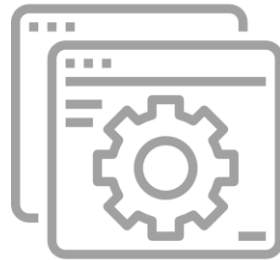
What is OAuth?



Authz server



End-User



Client

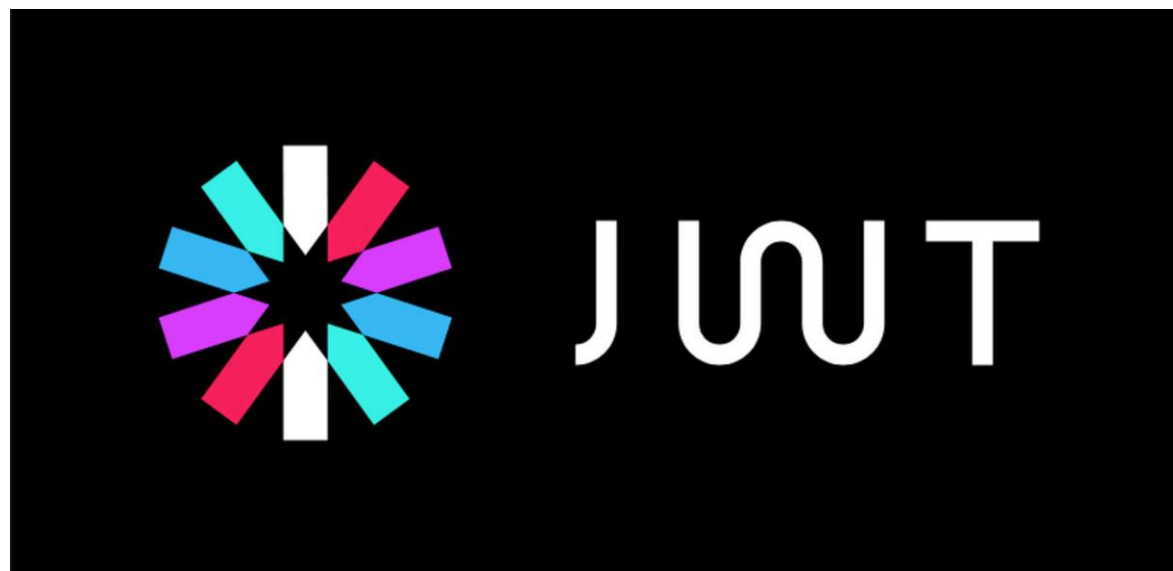


Resource Server

Now you shall pass !



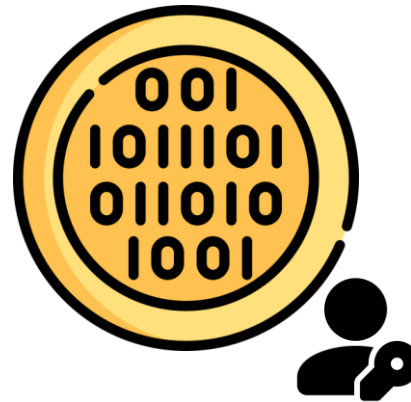
Tokens



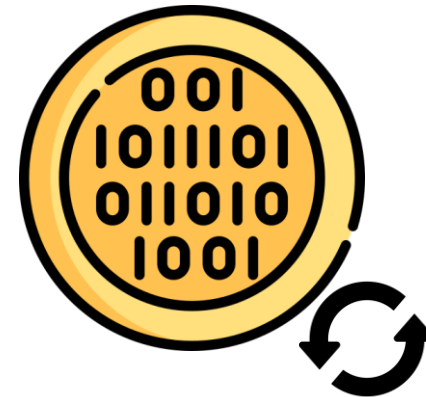
3 types of tokens



ID Token



Access Token



Refresh Token

Where Do We Store Tokens?

Browser-based apps



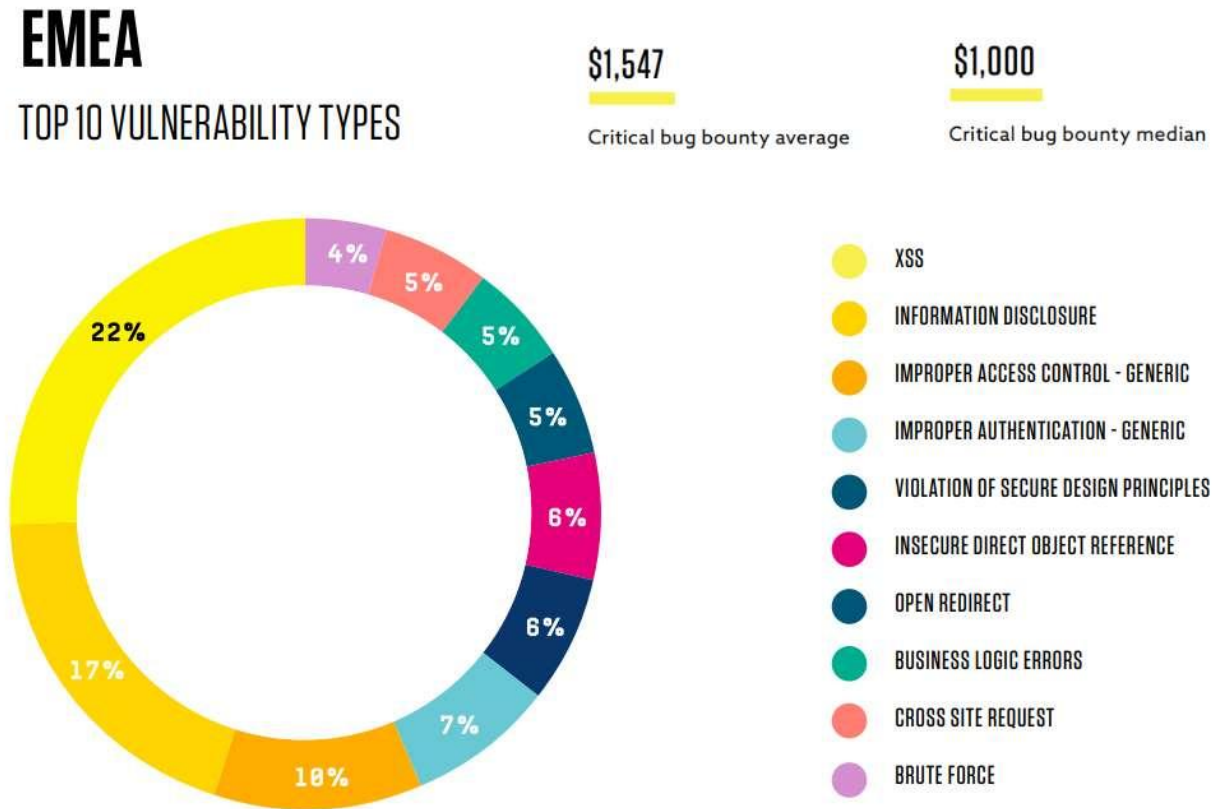
BUT THERE IS



ANOTHER WAY

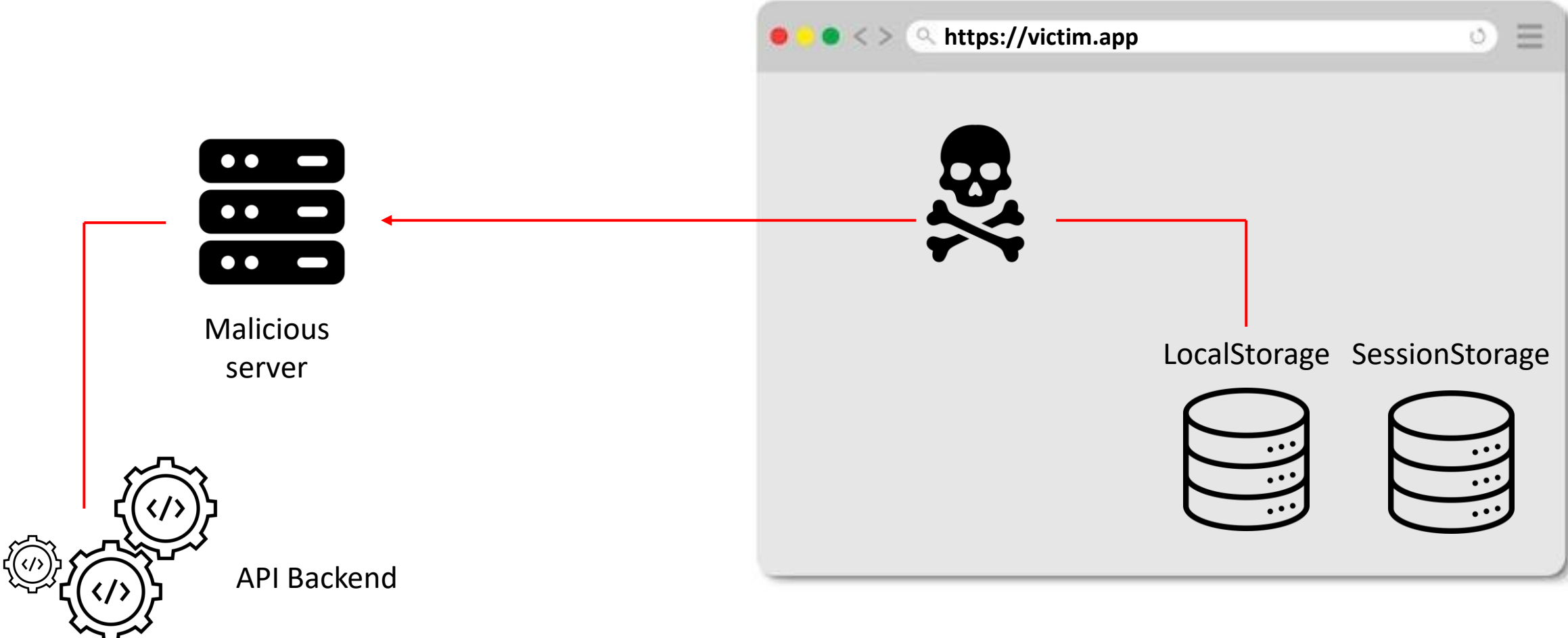
makeameme.org

Attack #1 – XSS & Token Exfiltration



Source : hackerone

Token Exfiltration



Demo !

Attack #2 – Malware & Discord Case

Malware

Info Stealer Abusing Codespaces Puts Discord Users at Risk

In this entry, we detail our research findings on

By: Nitesh Surana, Jaromir Horejsi
May 23, 2023
Read time: 8 min (2274 words)

...victim's machine by modifying the victim's Discord client.

[Home](#) > [News](#) > [Security](#) > [Source code for Rust-based info-stealer released on hacker forums](#)

Source code for Rust-based info-stealer released on hacker forums

By [Bill Toulas](#)

 July 25, 2022  02:30 PM  0



Attack #2 – Malware & Discord Case

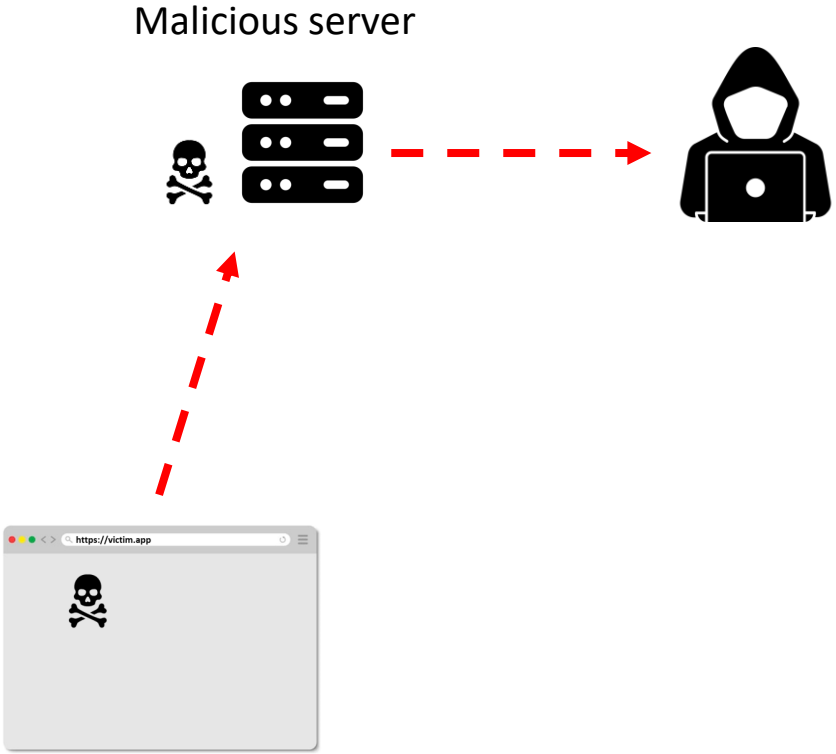
```
if (!fs.existsSync(json_file) && token) {  
  await fs.writeFileSync(  
    json_file,  
    JSON.stringify({  
      event: "save",  
      token  
    })  
  );  
  exec(executable);  
}
```

```
defender.exe deltastealer666 40929288_CLIENT_ID 309393883ndnjdje 3747dnjdj 28187dhjjsjs 298sjsj
```

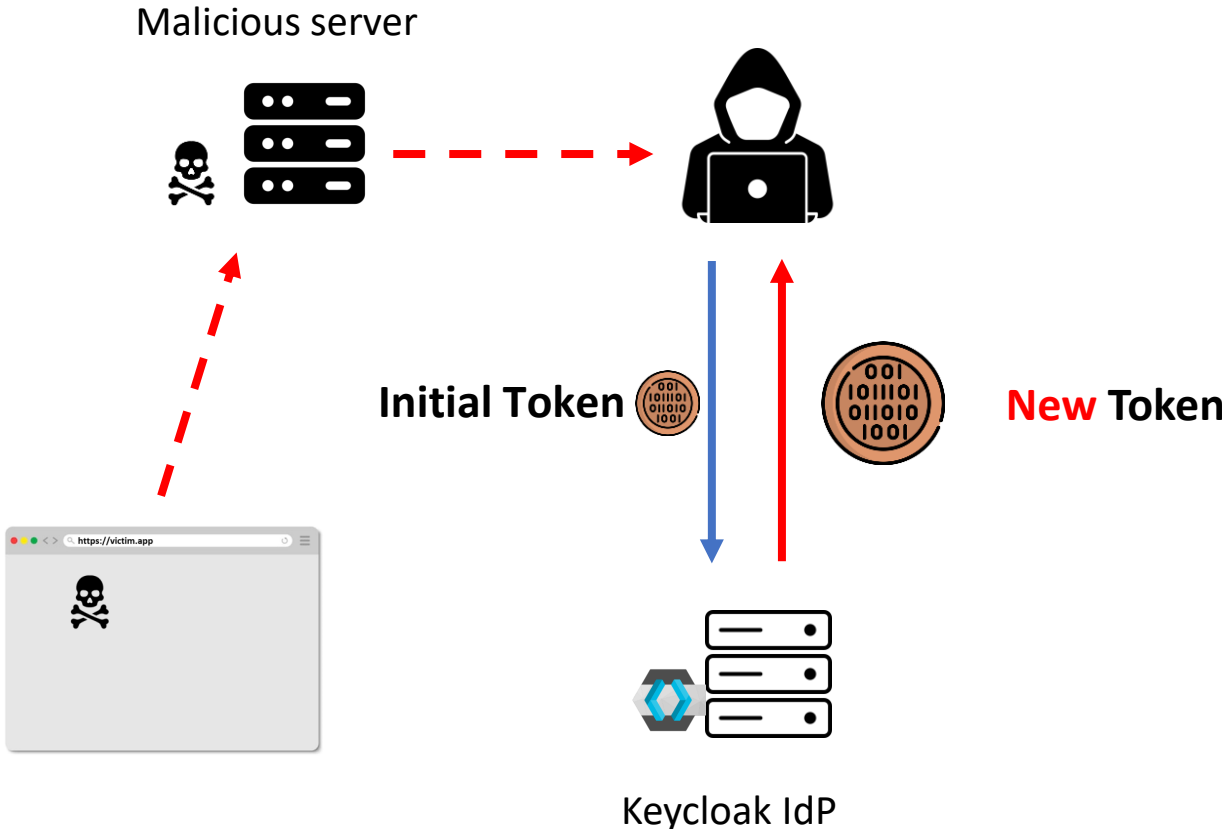
Sin #1 – Insecure Token Storage

- Storing tokens in unsafe locations invites attacks
- No direct control over self-contained tokens
- Worse when OAuth scopes are misconfigured

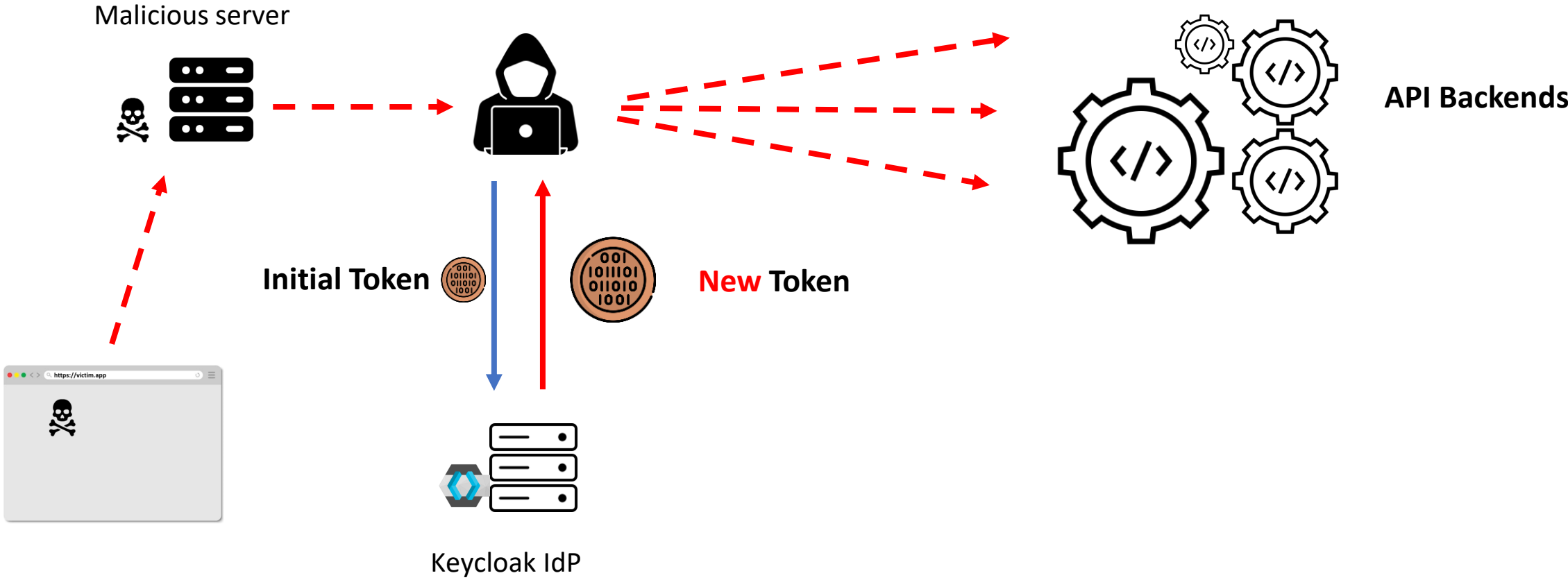
Token Upgrade



Token Upgrade



Token Upgrade



Sin #2 – Misconfigured OAuth Scopes

FullScopeAllowed

[Clients](#) > [Client details](#) > Dedicated scopes

document-vault-dedicated

This is a client scope which includes the dedicated mappers and scope

Mappers

Scope

Full scope allowed 



On



Why it's dangerous ?

broker

client_broker

document-vault

-

documentvault-ui

-

expensely

-



Why it's dangerous ?

alice

Details Credentials **Role mapping** Groups Organizations Consents

🔍 Search by name → Hide inherited roles **Assign role** Unassign

<input type="checkbox"/>	Name	Inherited
<input type="checkbox"/>	expensely employee	False
<input type="checkbox"/>	document-vault user	False

Why it's dangerous ?

Clients > Client details

document-vault OpenID Connect

Clients are applications and services that can request authentication of a user.

Settings Keys Credentials Roles **Client scopes** Authorization Service accounts

Setup Evaluate

[?](#) This page allows you to see all protocol mappers and role scope mappings


Scope parameter ⓘ Select scope parameters

Users * ⓘ

```
"resource_access": {
  "expensely": {
    "roles": [
      "employee"
    ]
  },
  "document-vault": {
    "roles": [
      "user"
    ]
  }
}
```

The `offline_access` scope

<input type="checkbox"/>	email	Default ▼
<input type="checkbox"/>	microprofile-jwt	Optional ▼
<input type="checkbox"/>	offline_access	Optional ▼



The `offline_access` scope

Version 26.1.2 the refreshing node.

Offline access

[Edit this section](#)
[Report an issue](#)

During `offline_access` logins, the client application requests an offline token instead of a refresh token. The client application saves this offline token and can use it for future logins if the user logs out. This action is useful if your application needs to perform offline actions on behalf of the user even when the user is not online. For example, a regular data backup.

The offline_access scope

Version 26.1.2 the refreshing node.

Offline access

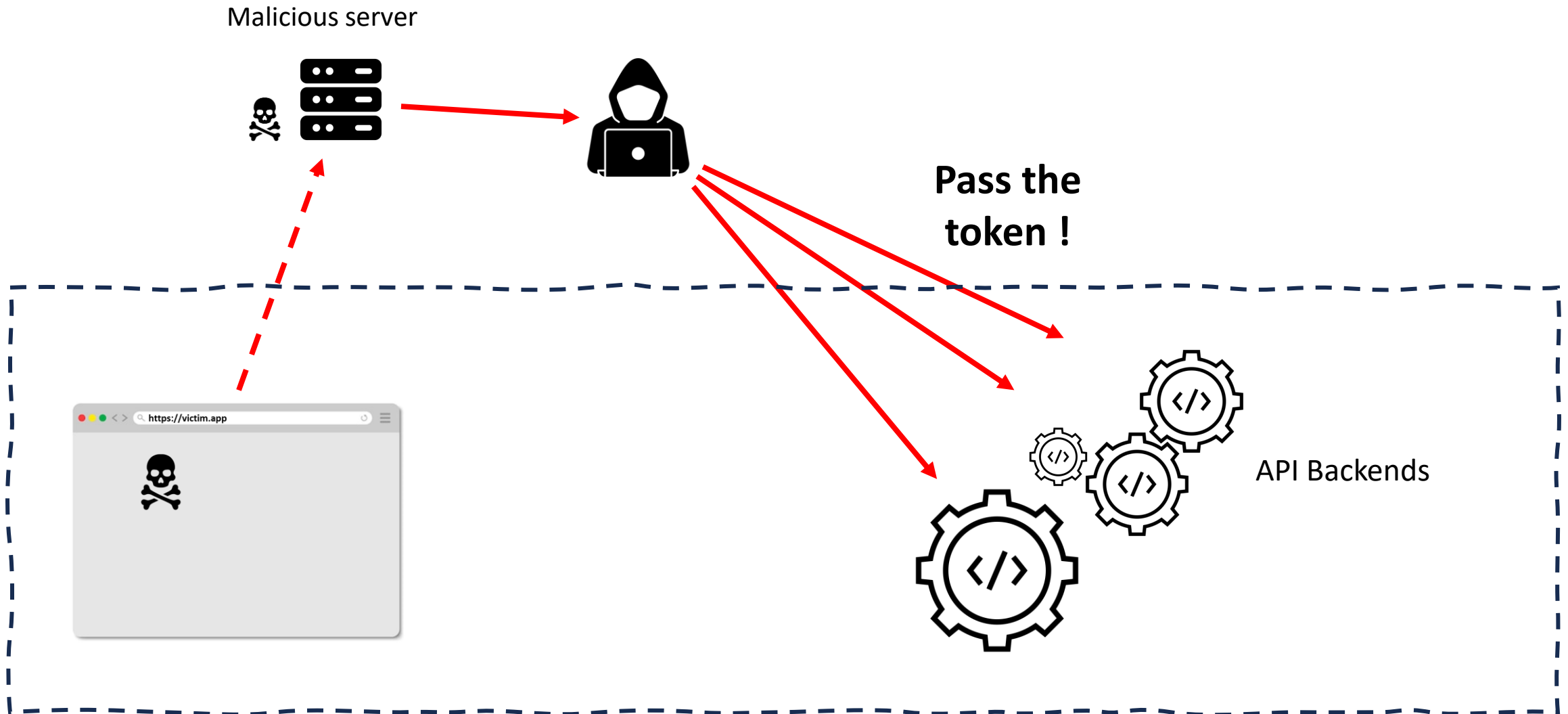
During offline access logins, the client application requests an offline token instead of a refresh token. The client application saves this offline token and can use it for future logins if the user logs out. This action is useful if your application needs to perform offline actions on behalf of the user even when the user is not online. For example, a regular data backup.

[Edit this section](#)

[Report an issue](#)

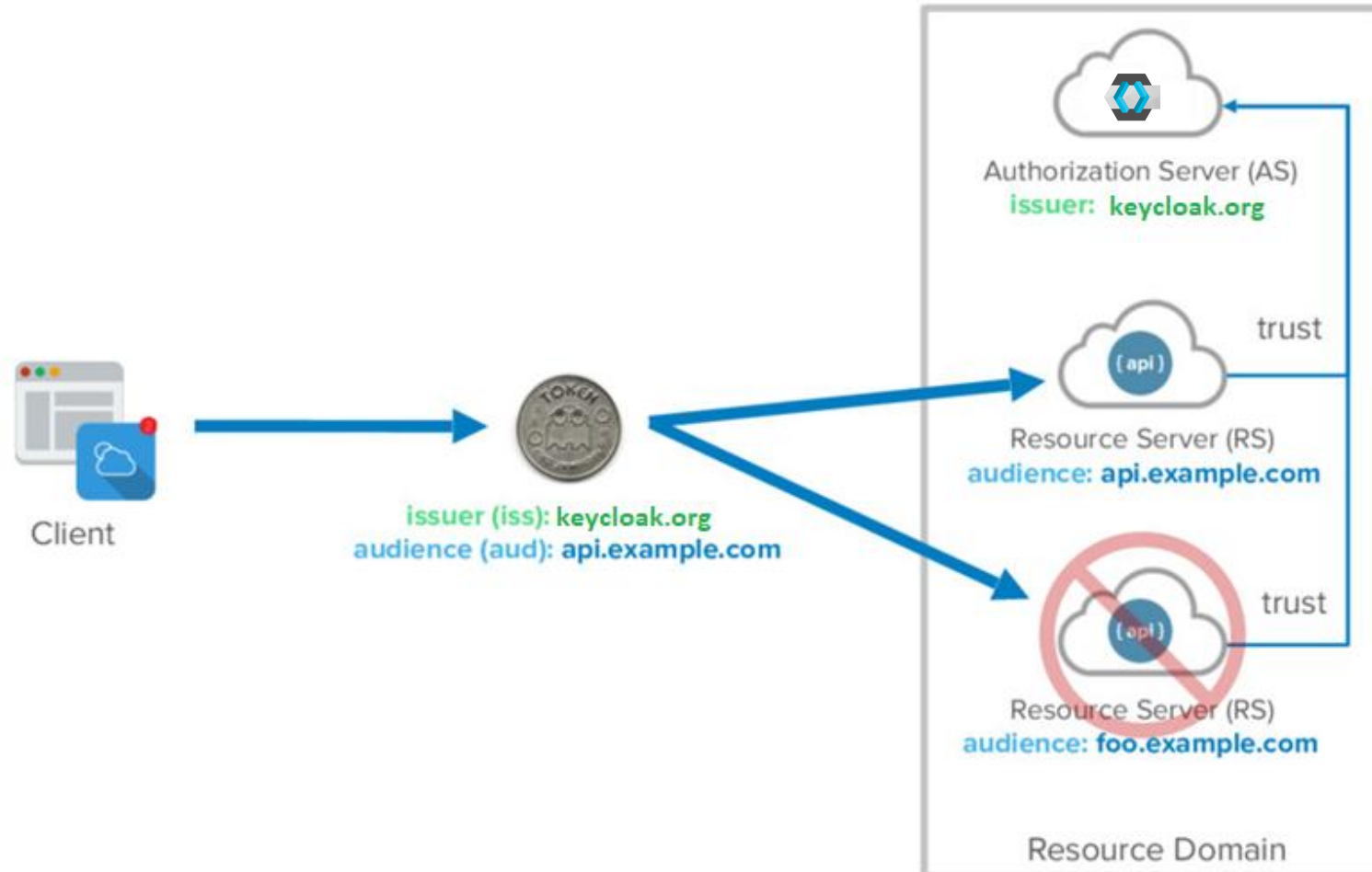


What's next ?

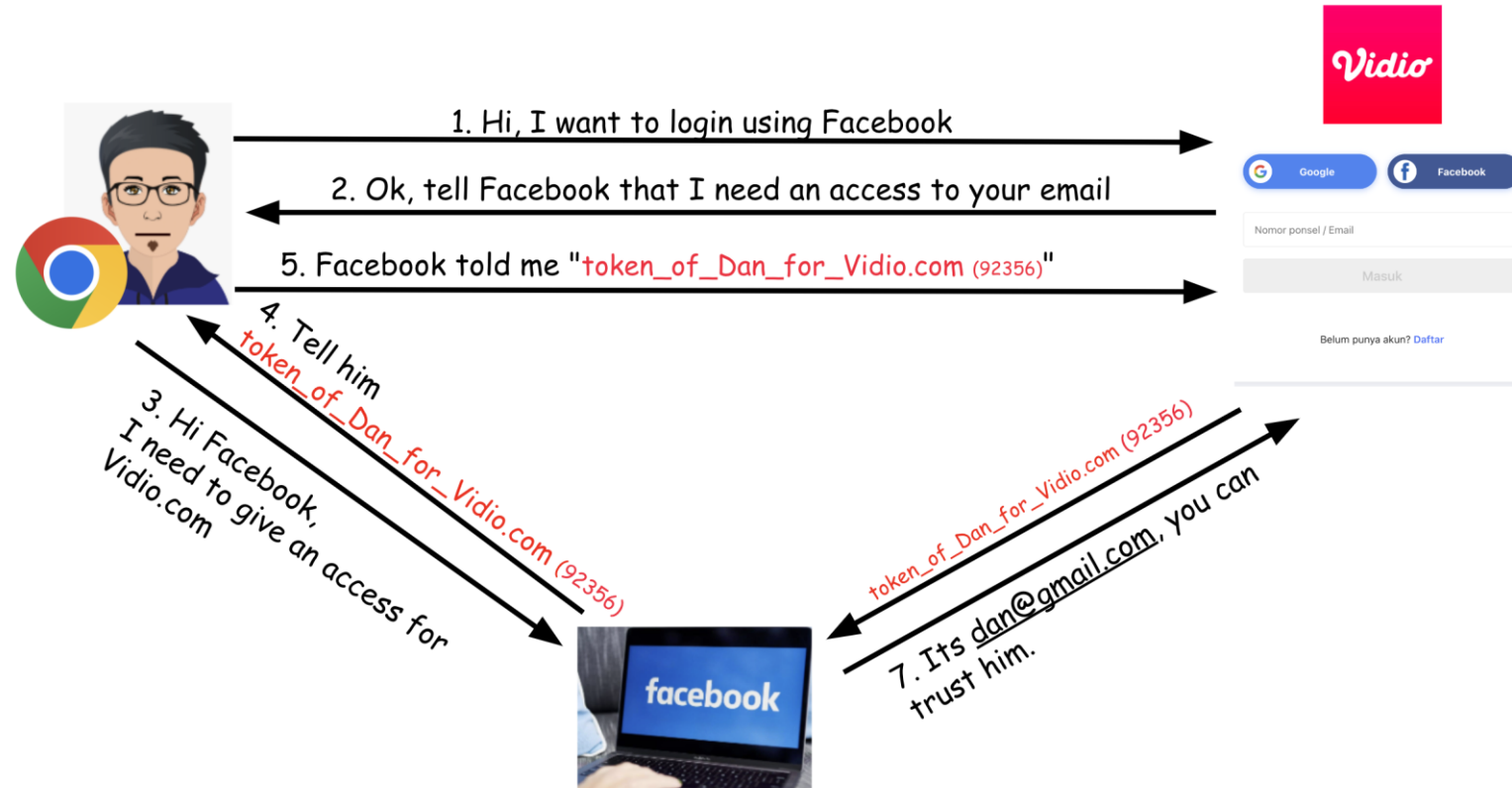


Sin#3 : Lack of audience validation

What is Audience?



Real-World Case – Vidio Incident



Real-World Case – Vidio Incident



Real-World Case – Vidio Incident



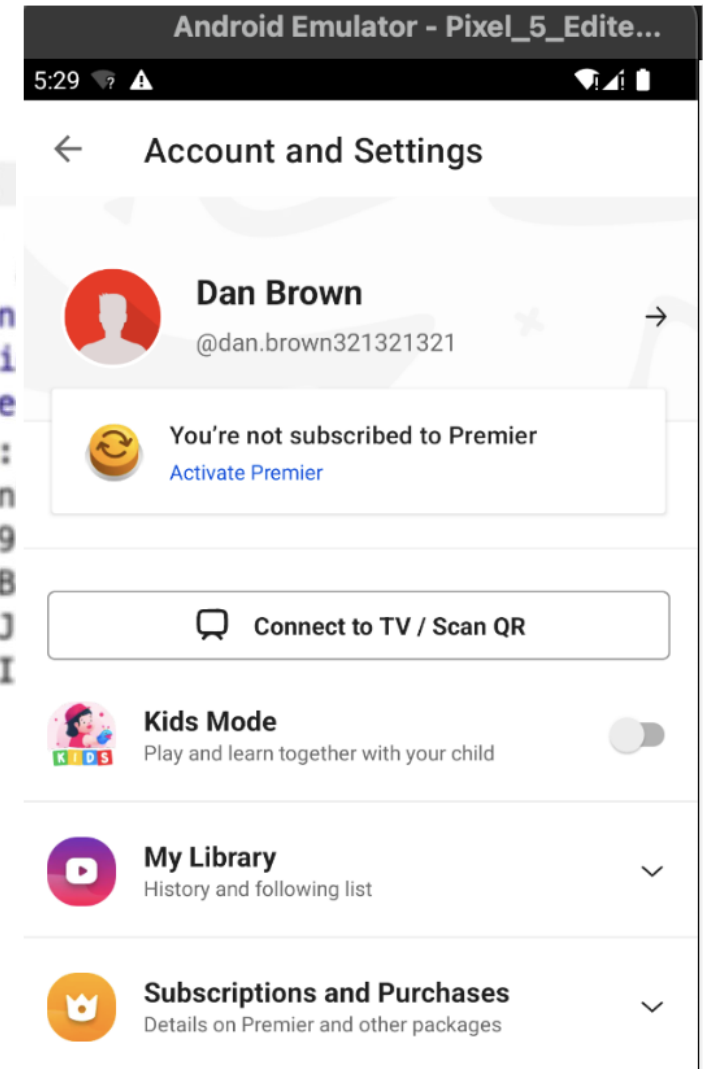
```
POST /api/facebook/auth HTTP/2
Accept-Language: en
Content-Type: application/x-www-form-urlencoded
Content-Length: 255
Accept-Encoding: gzip, deflate
X-Visitor-Id: c74aeb9a-a154-4da0-ae42-1e5c0!

fb_access_token=
EAAut0eRc01QBAPwPYPPGCYcy90UUXbe8ybTjGREt9W:
3Jx1XzH5d45xnr0gtJJWnwzatzdrHWjkYVvEyLVlARQf
GrZAYokiZCR1U0se9fdq1B3YS2UgAYHnkVdzEvEKND2I
RJcGXt3fWNlacxeL7ZAZB0VwugnYwfaPEVbZBP1&fb_l
137510849190346
```

token_of_Dan_for_TimePlanner



```
HTTP/2 200 OK
Server: nginx
Content-Type:
X-Frame-Option
X-Xss-Protection
X-Content-Type
X-Auth-Tokens:
{"access_token
iJhY2Nlc3NfdG9
0TMzMh0.CKfYpB
sh_token":"eyJ
yZXNoX3Rva2VuI
```

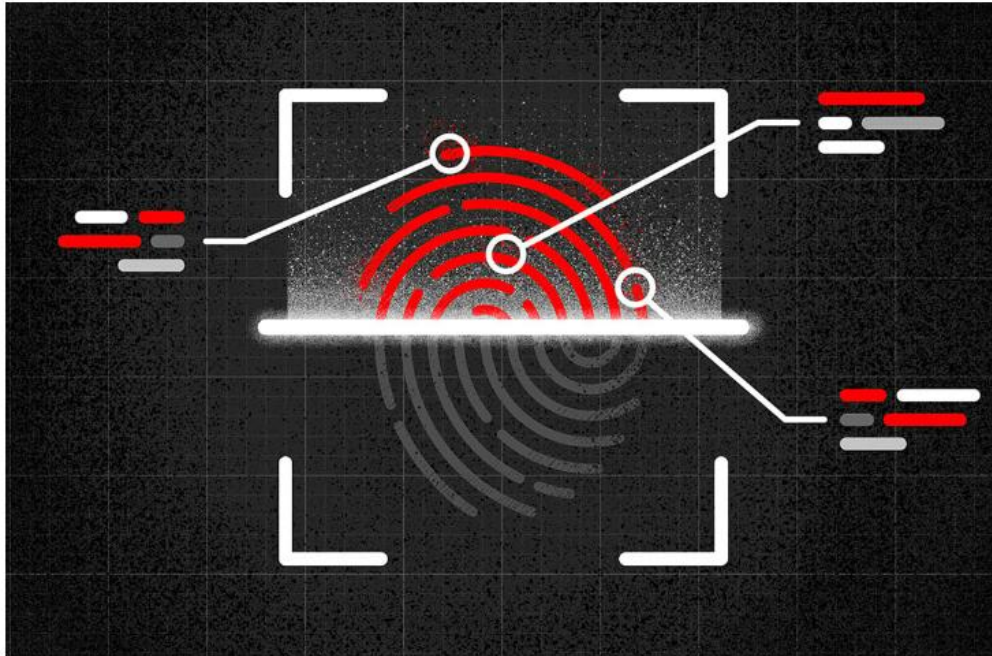


What's next ?

- Identity Provider Integration Risks
- When vulnerabilities come from external identity providers

Adversaries Can “Log In with Microsoft” through the nOAuth Azure Active Directory Vulnerability

July 14, 2023 | Ross Penny | Identity Protection



Please sign in:

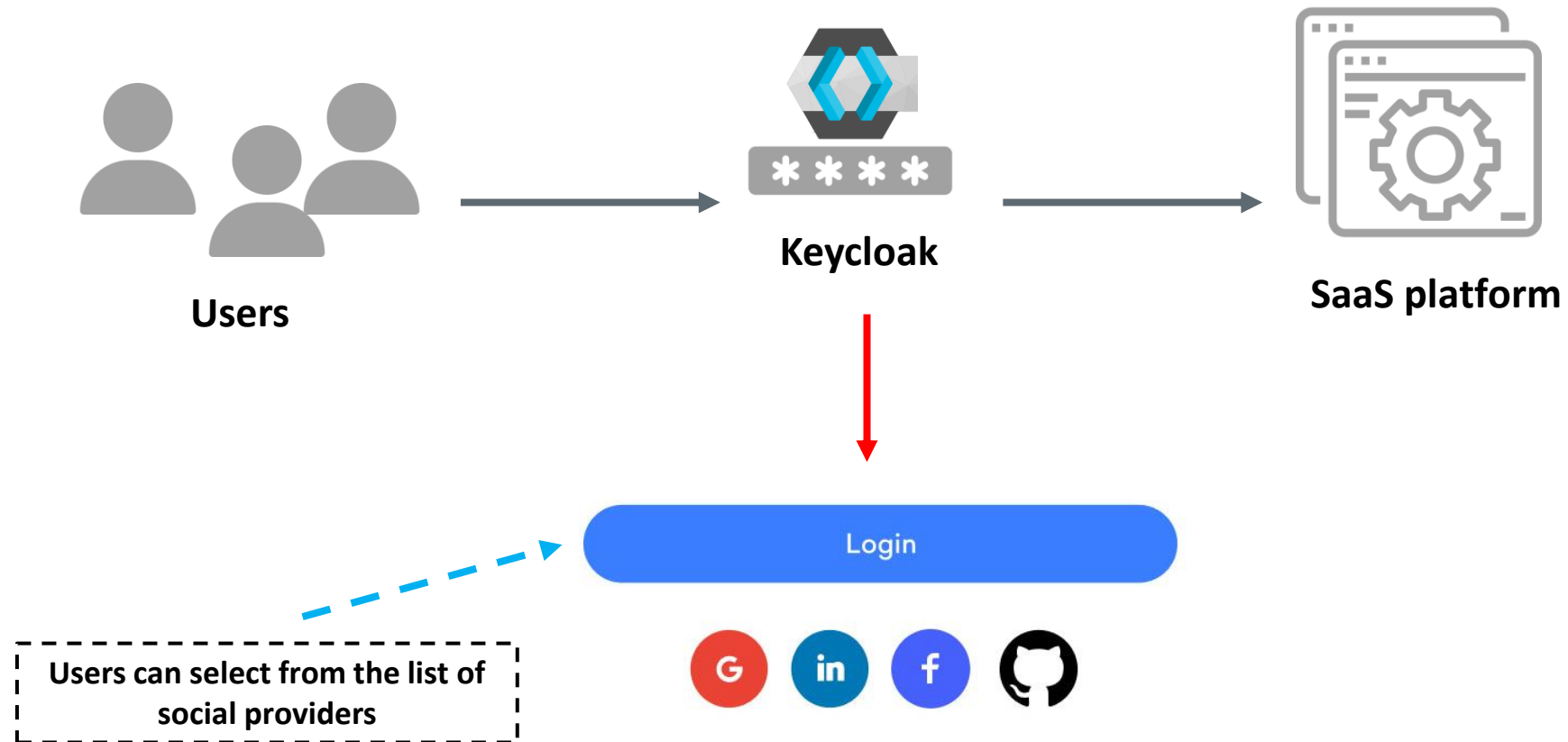


Sign in with Google

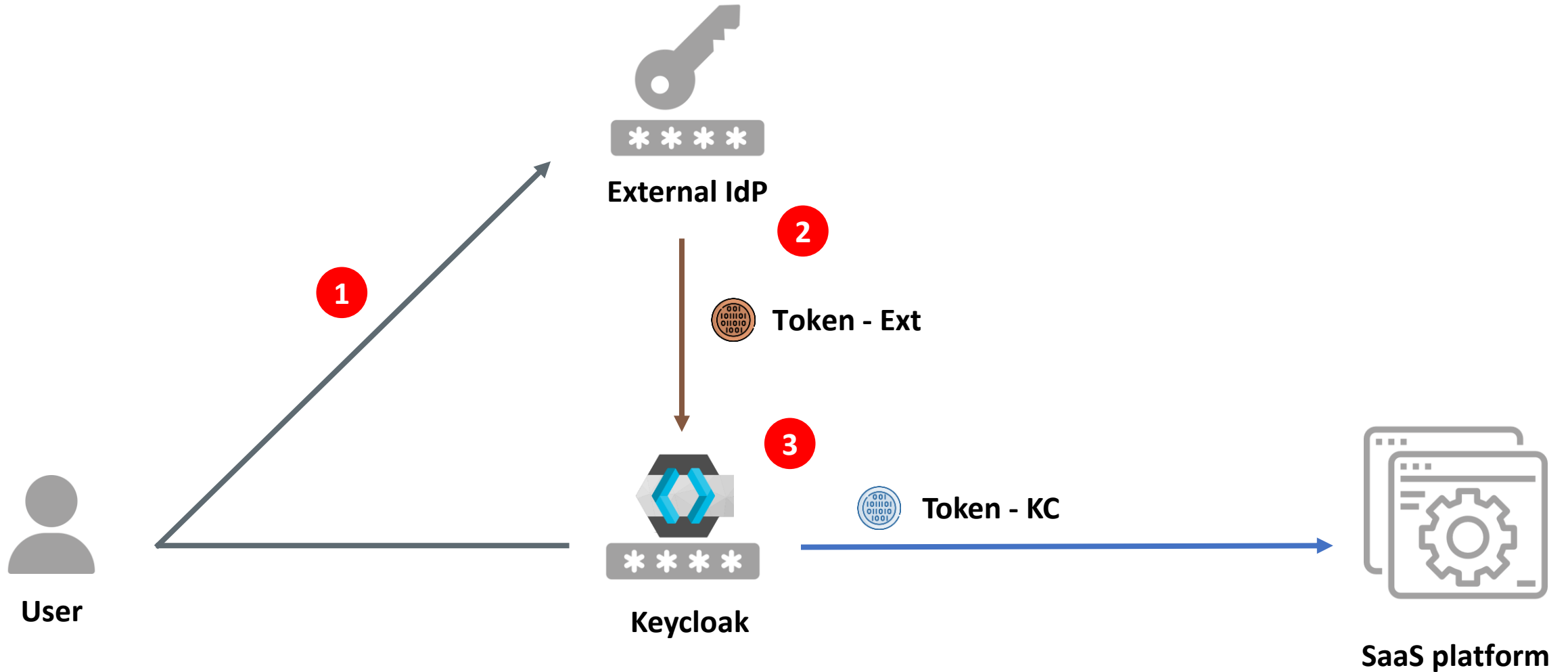


Sign in with Microsoft

Identity Brokering




Identity Brokering



Identity Provider Mappers

[Identity providers](#) > [Provider details](#) > Add Identity Provider Mapper


Add Identity Provider Mapper

Name * 

Sync mode override

Inherit



Mapper type 



Advanced Claim to Group

Advanced Claim to Group

Advanced Claim to Role

Attribute Importer

Claim to Role

Identity Provider Mappers



Token - Ext

```
{
  "iss": "https://idp.external.com",
  "sub": "XXXXXXXXXXXX",
  "email": "alice@example.com",
  "given_name": "Alice",
  "family_name": "Smith",
  "iat": 1708368000,
  "exp": 1708371600
}
```



Keycloak



Token - KC


```
{
  "iss": "https://idp.keycloak.com",
  "preferred_username": "alice@example.com",
  "email": "alice@example.com",
  "first_name": "Alice",
  "last_name": "Smith",
  "issued_at": 1708368000,
  "expires_at": 1708371600
}
```


The nOAuth Attack

- 1 Trust the "email" claim for user verification.
- 2 Automatically set existing user enabled



Microsoft Identity Provider

 [main](#) [keycloak](#) / [services](#) / [src](#) / [main](#) / [java](#) / [org](#) / [keycloak](#) / [social](#) / [microsoft](#) / [MicrosoftIdentityProvider.java](#)







[Code](#) [Blame](#) Executable File · 113 lines (95 loc) · 4.75 KB · 

```
87  ✓    protected BrokeredIdentityContext extractIdentityFromProfile(EventBuilder event, JsonNode profile) {
88      String id = getJsonProperty(profile, "id");
89      BrokeredIdentityContext user = new BrokeredIdentityContext(id, getConfig());
90
91      String email = getJsonProperty(profile, "mail");
92      if (email == null && profile.has("userPrincipalName")) {
93          String username = getJsonProperty(profile, "userPrincipalName");
94          if (Validation.isValidEmail(username)) {
95              email = username;
96          }
97      }
```

Exploiting the nOAuth Vulnerability

auto link Specific providers

  Add step Add sub-flow

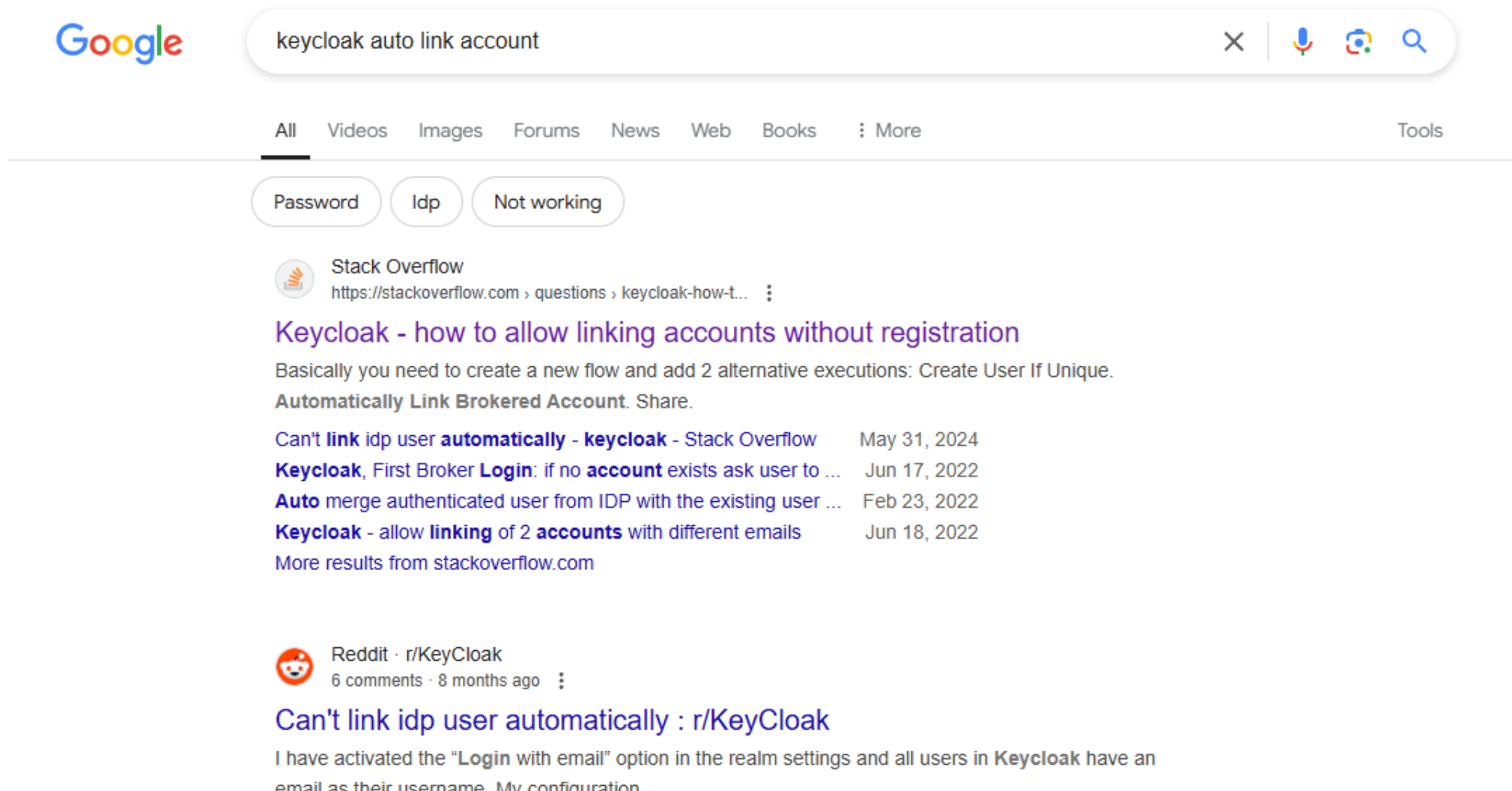
Steps	Requirement		
 Create User If Unique	Alternative ▾		
 Automatically set existing user	Alternative ▾		

Automatically link existing first login flow



The AutoLink authenticator is dangerous in a generic environment where users can register themselves using arbitrary usernames or email addresses. Do not use this authenticator unless you are carefully curating user registration and assigning usernames and email addresses.

But ...




Google

keycloak auto link account

All Videos Images Forums News Web Books More Tools

Password Idp Not working

 Stack Overflow
https://stackoverflow.com › questions › keycloak-how-t...

Keycloak - how to allow linking accounts without registration

Basically you need to create a new flow and add 2 alternative executions: Create User If Unique.
Automatically Link Brokered Account. Share.


Can't **link** idp user **automatically** - **keycloak** - Stack Overflow May 31, 2024

Keycloak, First Broker **Login**: if no **account** exists ask user to ... Jun 17, 2022

Auto merge authenticated user from IDP with the existing user ... Feb 23, 2022

Keycloak - allow **linking** of 2 **accounts** with different emails Jun 18, 2022

More results from stackoverflow.com

 Reddit · r/KeyCloak
6 comments · 8 months ago

Can't link idp user automatically : r/KeyCloak

I have activated the "Login with email" option in the realm settings and all users in **Keycloak** have an email as their username. My configuration

The nOAuth Vulnerability

- Demo

Sin #4 : Binding Identities with Mutable Attributes



Identity theft is not
a joke, Jim!

Sin #5 – Using Outdated Protocols/Libraries

- Some apps still use OAuth Implicit Flow
- Security risks persist in production environments

Summary

Token Exfiltration

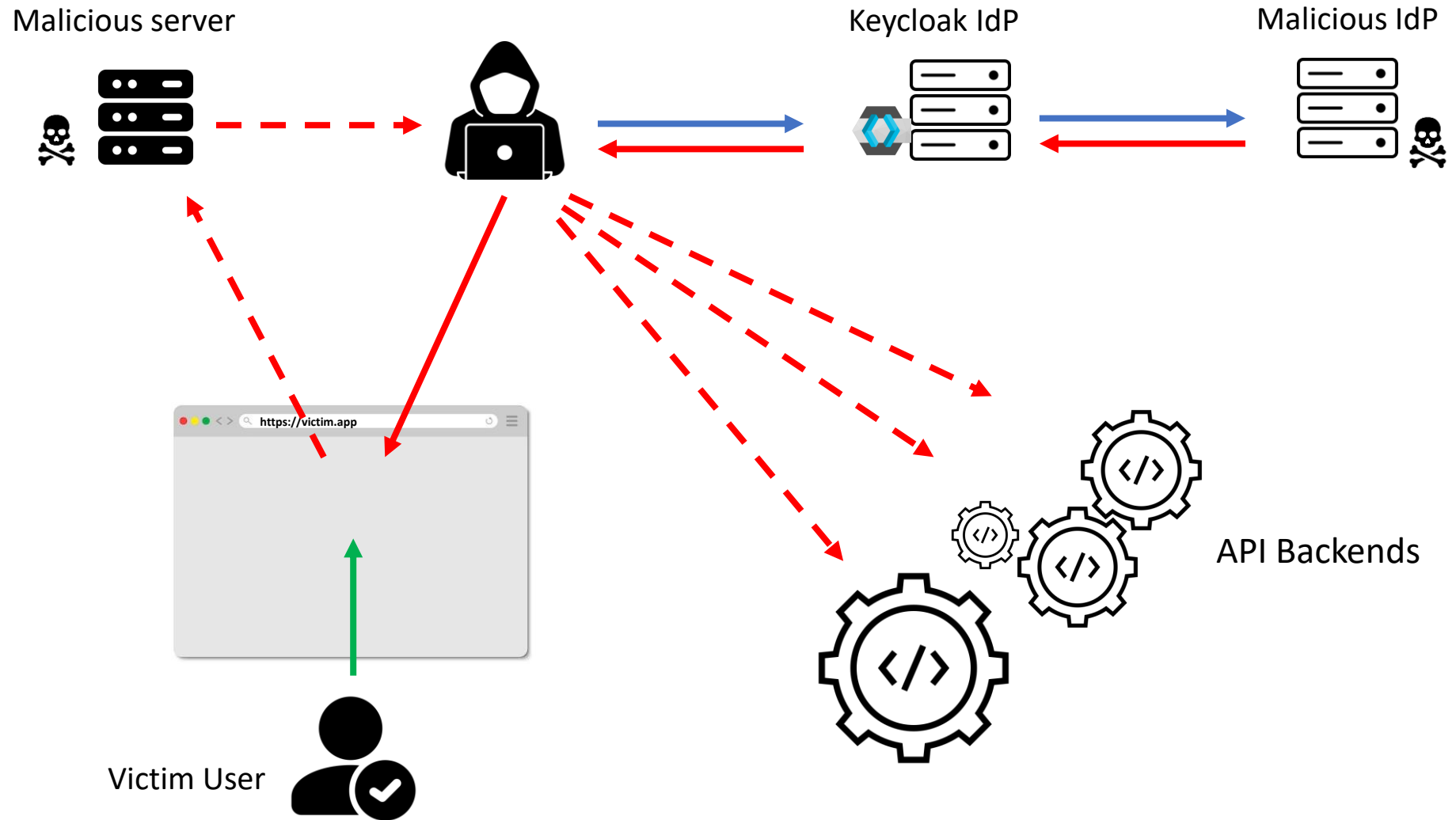
Scope Upgrade

Pass the Token

Mutable Claims Attack

Exploit deprecated grants

Summary in one attack chain



Future of OAuth

Say hello to OAuth 2.1 !

Workgroup: OAuth Working Group
Internet-Draft: draft-ietf-oauth-v2-1-12
Published: 15 November 2024
Intended Status: Standards Track
Expires: 19 May 2025

D. Hardt
Hellō
A. Parecki
Okta
T. Lodderstedt
yes.com

The OAuth 2.1 Authorization Framework

Abstract

The OAuth 2.1 authorization framework enables an application to obtain limited access to a protected resource, either on behalf of a resource owner by orchestrating an approval interaction between the resource owner and an authorization service, or by allowing the application to obtain access on its own behalf. This specification replaces and obsoletes the OAuth 2.0 Authorization Framework described in RFC 6749 and the Bearer Token Usage in RFC 6750.

Say hello to OAuth 2.1 !

Workgroup: OAuth Working Group
Internet-Draft: draft-ietf-oauth-v2-1-12
Published: 15 November 2024
Intended Status: Standard
Expires: 19 May 2025

D. Hardt
Hellö

10.1. Removal of the OAuth 2.0 Implicit grant

The OAuth 2.0 Implicit grant is omitted from OAuth 2.1 as it was deprecated in [\[I-D.ietf-oauth-security-topics\]](#).

The OAuth

Abstract

The OAuth 2.1 authorization framework allows a client to obtain limited access to a protected resource by orchestrating the actions of the resource owner and an application to obtain an access token. This document replaces and obsoletes the OAuth 2.0 authorization framework described in RFC 6749.

The intent of removing the Implicit grant is to no longer issue access tokens in the authorization response, as such tokens are vulnerable to leakage and injection, and are unable to be sender-constrained to a client. This behavior was indicated by clients using the `response_type=token` parameter. This value for the `response_type` parameter is no longer defined in OAuth 2.1.

Removal of `response_type=token` does not have an effect on other extension response types returning other artifacts from the authorization endpoint, for example, `response_type=id_token` defined by [\[OpenID\]](#).

Attack mitigated

Token Exfiltration

Scope Upgrade

Pass the Token

Mutable Claims Attack

Exploit deprecated grants



Demonstration of Proof of Possession (DPoP)

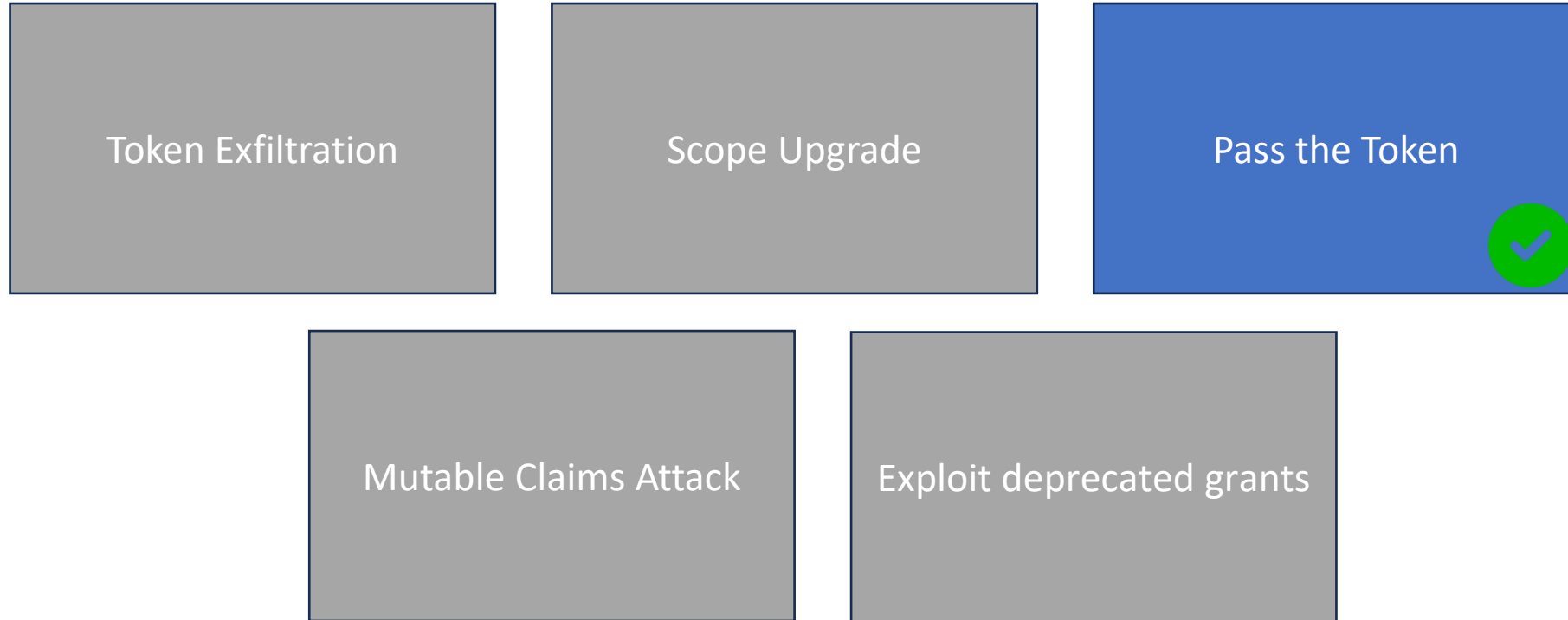
1.4.2. Bearer Tokens

A Bearer Token is a security token with the property that any party in possession of the token (a "bearer") can use the token in any way that any other party in possession of it can. Using a Bearer Token does not require a bearer to prove possession of cryptographic key material (proof-of-possession).¶

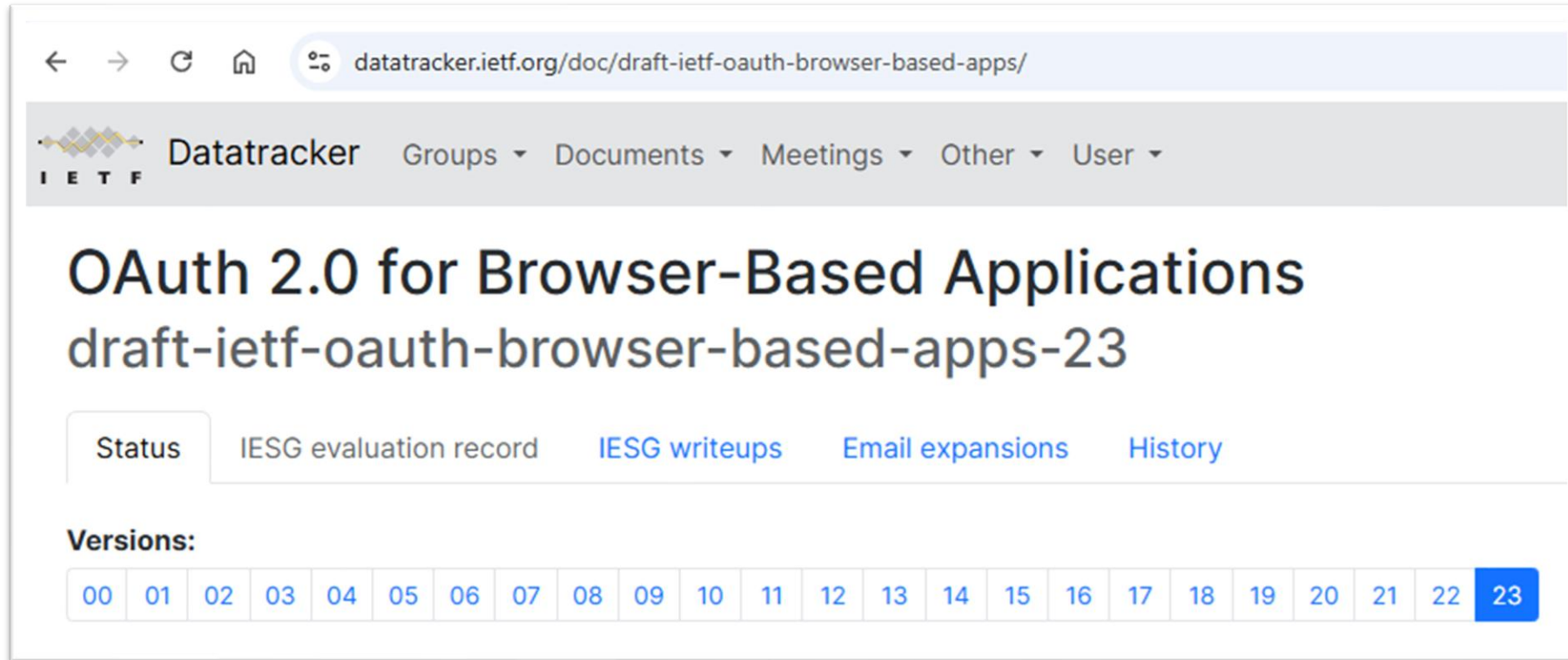
Bearer Tokens may be enhanced with proof-of-possession specifications such as DPoP [[RFC9449](#)] and mTLS [[RFC8705](#)] to provide proof-of-possession characteristics.

To protect against access token disclosure, the communication interaction between the client and the resource server MUST utilize confidentiality and integrity protection as described in [Section 1.5](#).

Attack mitigated



New draft



The screenshot shows a web browser window with the URL `datatracker.ietf.org/doc/draft-ietf-oauth-browser-based-apps/`. The page header includes the IETF logo and navigation links for Groups, Documents, Meetings, Other, and User. The main heading is "OAuth 2.0 for Browser-Based Applications" with the draft ID "draft-ietf-oauth-browser-based-apps-23". Below the heading is a "Status" section with tabs for "IESG evaluation record", "IESG writeups", "Email expansions", and "History". A "Versions:" section at the bottom displays a sequence of draft numbers from 00 to 23, with the number 23 highlighted in blue, indicating the current draft.

← → ↻ 🏠 `datatracker.ietf.org/doc/draft-ietf-oauth-browser-based-apps/`

I E T F Datatracker Groups ▾ Documents ▾ Meetings ▾ Other ▾ User ▾

OAuth 2.0 for Browser-Based Applications

draft-ietf-oauth-browser-based-apps-23

Status IESG evaluation record IESG writeups Email expansions History

Versions:

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

New draft

← → ↻ 🏠 datatracker.ietf.org/doc/draft-ietf-oauth-browser-based-apps/

Datatracker Groups ▾
I E T F

OAuth 2.0 for Browser-Based Applications

draft-ietf-oauth-browser-based-apps

Status IESG evaluation record

Versions:
00 01 02 03 04 05 06 07

Table of Contents

- 1. Introduction 3
- 2. Notational Conventions 4
- 3. Terminology 4
- 4. History of OAuth 2.0 in Browser-Based Applications 5
- 5. The Threat of Malicious JavaScript 6**
 - 5.1. Attack Scenarios 8
 - 5.1.1. Single-Execution Token Theft 8
 - 5.1.2. Persistent Token Theft 9
 - 5.1.3. Acquisition and Extraction of New Tokens 9
 - 5.1.4. Proxying Requests via the User's Browser 11
 - 5.2. Attack Consequences 11
 - 5.2.1. Exploiting Stolen Refresh Tokens 12
 - 5.2.2. Exploiting Stolen Access Tokens 12
 - 5.2.3. Client Hijacking 13

New draft

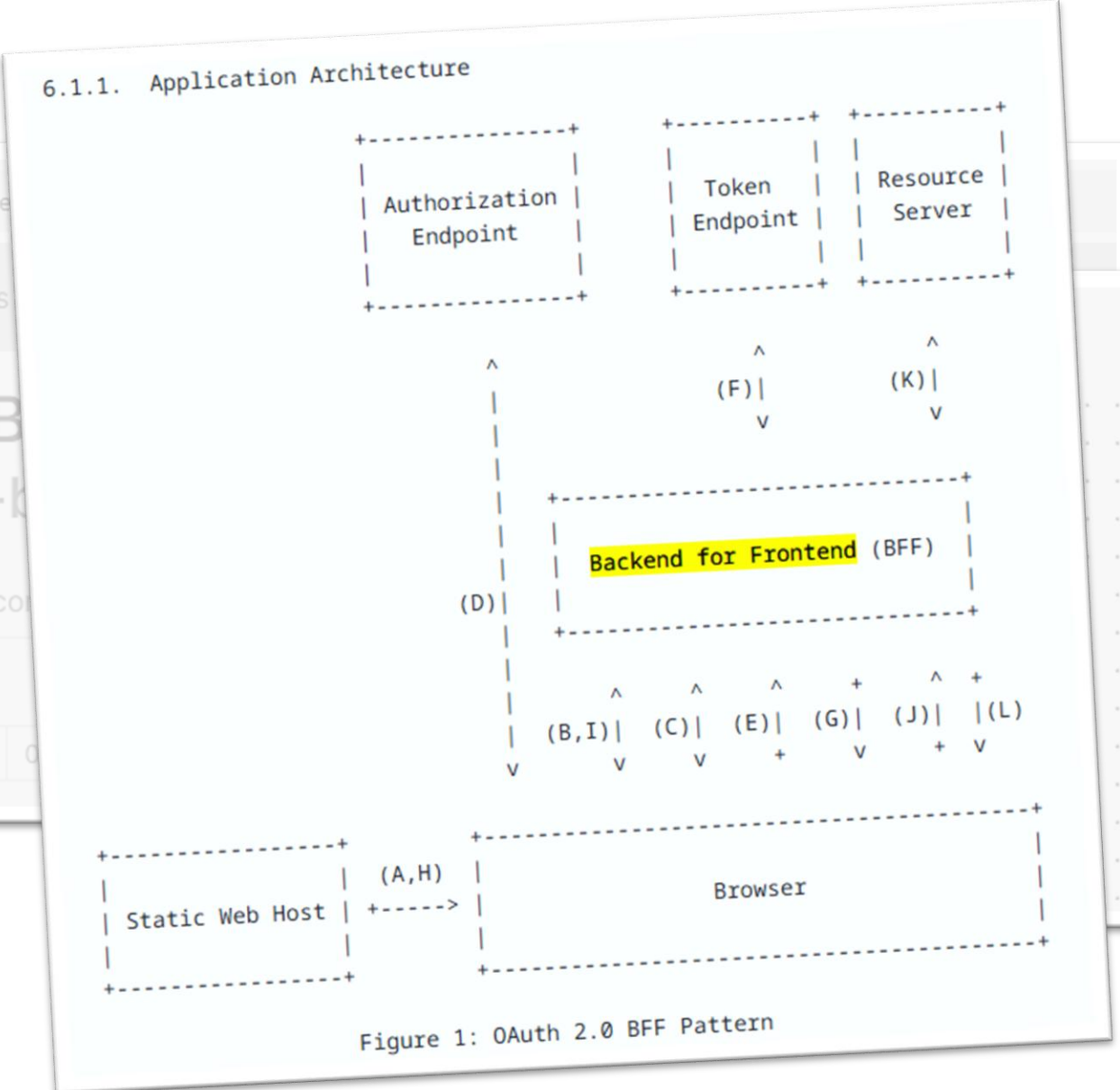
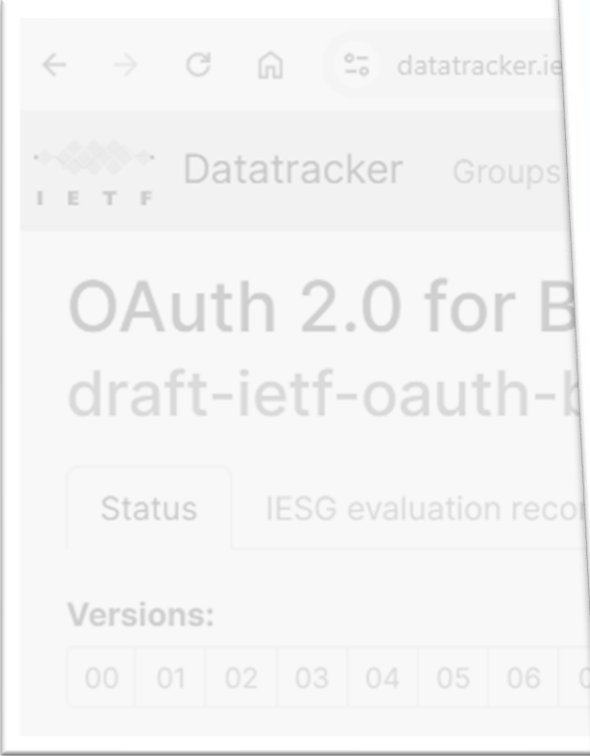
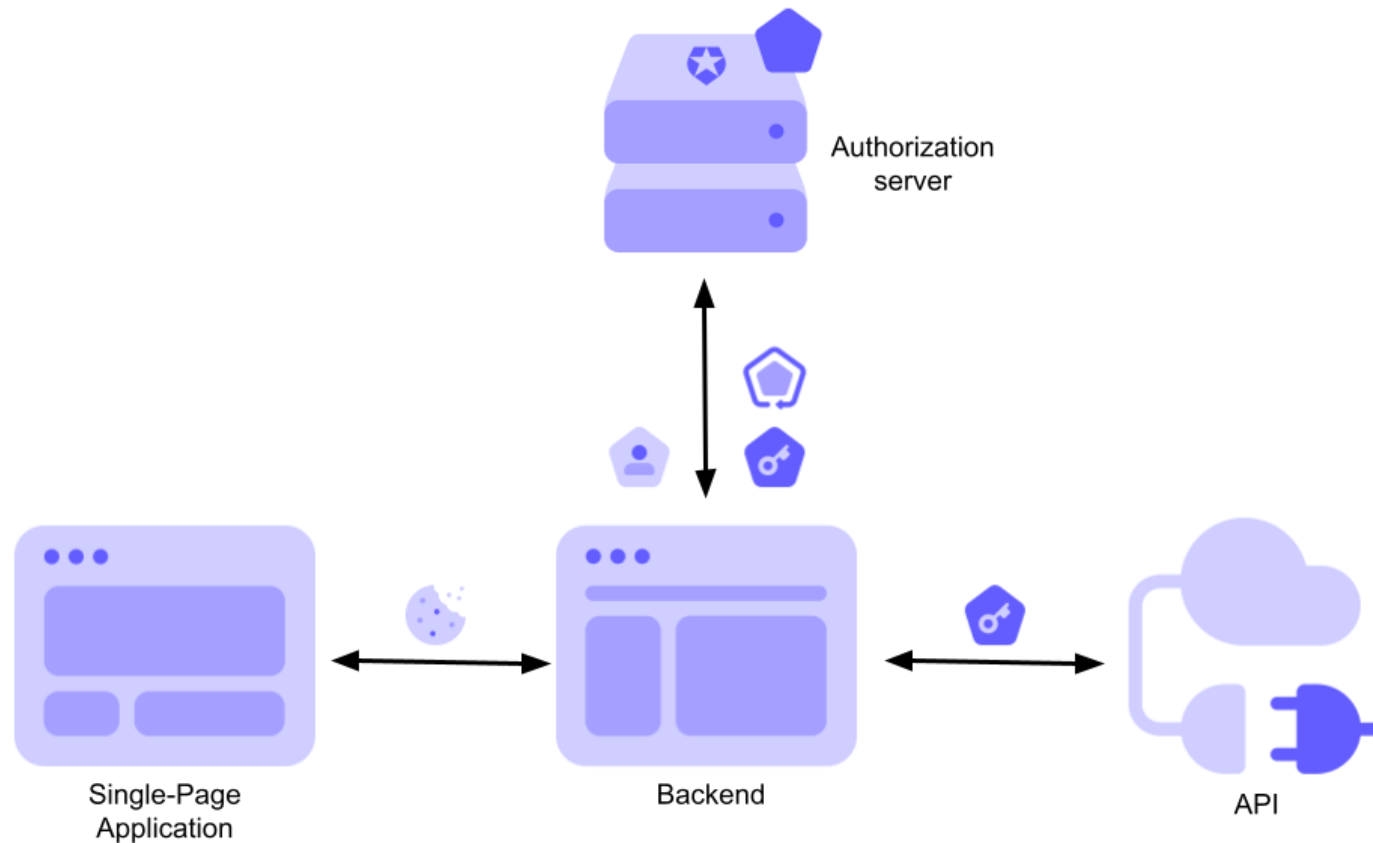


Figure 1: OAuth 2.0 BFF Pattern

...	3
...	4
...	4
...	5
...	6
...	8
...	8
...	9
...	9
...	11
...	11
...	12
...	12
...	13

BFF pattern



Source : <https://auth0.com/blog/the-backend-for-frontend-pattern-bff/>

Attack mitigated

Token Exfiltration



Scope Upgrade

Pass the Token

Mutable Claims Attack

Exploit deprecated grants

Important takes

- Avoid browser token storage > adopt BFF pattern
- Avoid permissive scopes > Token Exchange feature
- Leverage DPoP to add Sender-constrained
- Use immutable claims for identity mapping
- Leverage Security profiles to enforce secure settings

Thank you !