EXCELLIUM

"OAuth/OpenID Connect" system from a security point of view?

How to evaluate an

verified on a system using OAuth/OpenID Connect (OpenID Connect will be called OIDC in the rest of the post). Therefore, it assumes you are familiar with all the concepts related to OAuth/OIDC. All references to OAuth refer to OAuth 2.0. If it is not the case then you can refer to this free online course named "Introduction to OAuth 2.0 and OpenID

his post presents a collection of security-oriented validation points that should be

Connect" kindly created and provided by Dr. Philippe De Ryck or the several tutorials from ConnectId. Note that this post is mainly security-oriented feedback following a complete focused training that I have recently taken on the OAuth/OIDC topics.

The importance of OAuth & OIDC

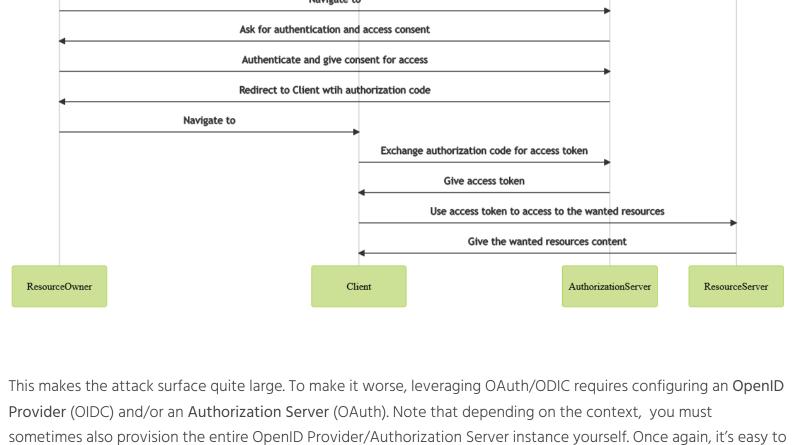
OAuth and OIDC address respectively the Authorization and Authentication aspects. Therefore, any issues in these areas can have critical consequences from a security point of view like authentication or authorization bypass for example.

One of the challenges faced is that there are several actors involved as well as different communications exchanges.

Below is a simplified example of OAuth authorization code flow:

Redirect to AS to ask for access to the wanted resources

ResourceOwner



Getting familiar with OAuth & OIDC As I was totally new to the OAuth and OIDC world, I decided to take the course named "Mastering OAuth 2.0 and

weaknesses.

evolve over time with the growth of my experience in this field.

be inaccurate for you if you know how to automate it \odot .

overview of the collection of validation points.

introduce a weakness via insecure settings.

After the lessons, I decided to create a list of all pitfalls discovered during the training. The different modules of the course are oriented for developers, but I simply converted the "attention points" into "security tests" as well as performing, in my head, a penetration test on each feature or flow presented by the instructor to identify potential attack vectors and scenarios. It is obvious that the list is not exhaustive, but it is a good foundation, and it will

assessment (code review, configuration review, penetration test, etc.) is targeting only a specific actor. Each validation point has a unique identifier in order to allow referencing it in a document, script, report etc. A table is provided to indicate if a validation point is manual or automated. The automation status is based on the technical capabilities to create code that performs the target test, without human interaction, while giving a reliable result with the same level of trust as if it was performed manually. Once again, this "automation status" can

The list of validation points was organized by actors in order to allow focusing on one actor if the scope of an

Overview of the validation points for OAuth & OIDC

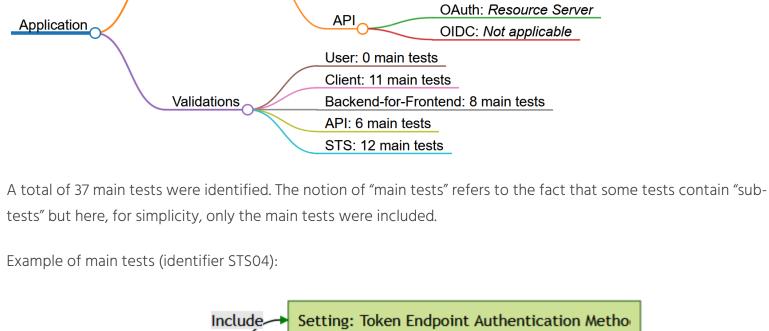
In addition to a representation using a "list" approach, a mind map was created in order to provide a high-level

Below is the overview of the number of tests identified (a validation point refers to a test): OAuth: Resource Owner

OIDC: End-User OAuth: Client Client, OIDC: Relying Party Terminology OAuth: Authorization Server

Security Token Service (STS)

OIDC: OpenID Provider



Include Setting: Allowed Web Origins

Include Setting: Allowed Origins Include Setting: Refresh Token Rotations Client settings in STS Setting: Refresh Token Expirations Include Setting: JSON Signature Algorithm Setting: Cross-Origin Verification Fallback Include` Include Setting: Allowed Grant Types The detailed version of the mind map is available on the GitHub repository of the blog post.

Please, feel free to surf here to access the following photos in case of poor quality.

list. A demo configuration is provided to allow you to reproduce the test performed.

For SPA, ensure that it uses the Authorization Code Flow with PKCE instead of the "Authorization Code" basic flow (reference CLT01)

How to apply control on the different OAuth & OIDC areas

In this section, I used a local lab based on Keycloak to show how to perform some of the validation points from the

In the demo app, when the login is used, the following request is sent to the "/auth" endpoint:

Request URL: http://localhost:8080/auth/realms/demo/protocol/openid-connect/auth?client_id=demo&redirect_uri=http%3A%2F%2Flocalhost%3A9500%2FApp.html&state=07e37

The parameter "response_type" is set to "code". However, there is no parameter named "code_challenge" so the flow used here is "Authorization Code" and not "Authorization Code with PKCE".

Risk: It is possible to start a flow with an insecure mode implying that, if the authorization code is intercepted by an attacker then, it can be used to obtain an access token before the Client uses it (an authorization code is valid once).

Allowed Grant Types (flow types enabled) and Response Modes

for a client should be limited to the needed ones (reference

As seen in the previous test, the "Authorization Code" flow is used but is the "Implicit" flow enabled?

STSO4h)

Let's try to start such a one... 5 📭 💶 > http://localhost:8080/auth/realms/demo/protocol/openid-connect/auth?client_id=demo&redirect_uri=http%3A%2F%2Flocalhost%3A9500%: App.html&state=07e37303-013d-4203-8415-faea284d65b6&response_mode=fragment&<u>response_type=token&</u>scope=openid%20profile%20api&nonce=dd598c5f-6e5: 461b-8799-2f48ad1d1db1&prompt=consent" | select-string -pattern "Sign in to your account" <h1 id="kc-page-title"> Sign in to your account

PS http "http://localhost:8080/auth/realms/demo/protocol/openid-connect/auth?client_id=demo&redirect_uri=http%3A%2F%2Flocalhost%3A9500%2 FApp.html&state=07e37303-013d-4203-8415-faea284d65b6&response_mode=fragment&response_type=token&scope=openid%20profile%20api&nonce=dd598c5f-6e53

HTTP/1.1 302 Found Referrer-Policy: no-referrer Strict-Transport-Security: max-age=31536000; includeSubDomains X-Content-Type-Options: nosniff X-XSS-Protection: 1; mode=block

⚠ Risk: It is possible to start a flow using the deprecated "Implicit" mode.

A login form is proposed so the flow is allowed. When it is not the case, the following error is received:

Ensure that the STS rejects any request specifying a scope that is not defined for the targeted API and prevent scope enumeration/discovery operation (reference STS12)

Welcome user demo!

Decode Access Token

Logout

let options = { scope: "openid profile api", keycloak.login(options);

Keycloak allows defining optional scopes:

Clients > demo

Demo 🝵

of English words:

v1.3.1

Calibration

a weak code verifier:

(-Content-Type-Options: nosn

The "plain" algorithm is accepted but "code_challenge" is rejected.

Let's try with a challenge having for value 43 x "0":

<h1 id="kc-page-title">

Now it is accepted and the flow is started.

Timeout

Follow redirects : false

: false

98c5f-6e53-461b-8799-2f48ad1d1db1&prompt=consent"

: FUZZ: raft-small-words.txt

In the web client app, the following scopes are used:

(i) localhost:9500/App.html

Console Sources

Web Client

Setup @ Evaluate @ Assigned Default Client Scopes ② Default Client Scopes ② Available Client Scopes ② profile web-origins Optional client scopes are « Remove selected Assigned Optional Client Scopes ② Optional Client Scopes @ case when they are requeste address microprofile-jwt $offline_access$ « Remove selected When a flow is started with an invalid scope, the following error is received: MP> http "http://localhost:8080/auth/realms/demo/protocol/openid-connect/auth?client_id=demo&redirect_uri=http%3A%2F%2Flocalhost%3 App.html&state=07e37303-013d-4203-8415-faea284d65b6&response_mode=fragment&response_type=code&scope=TEST_Anonce=dd598c5f-6e53-461b-8799-2f48ad1 HTTP/1.1 302 Found Content-Length: Location: http://localhost:9500/App.html#error=invalid_scope&error_description=Invalid+scopes%3A+TEST&state=07e37303-013d-4203-8415-faea284d65l Strict-Transport-Security: max-age=31536000; includeSubDomains X-Content-Type-Options: no When a scope is valid then the login form is received with an HTTP 200.

Based on this discrepancy factor, it's possible to try the following scope enumeration using FFUF with a dictionary

S la la responsable de final de la responsación de la lacada de la lacada de la response de la finaldada de la lacada de lacada de la lacada del lacada de la lacada de lacada de la lacada de lacada de la lacada de lacada de lacada de la lacada de la lacada de la lacada de la lacada de lacada delacada de lacada de lacada de lacada delacada de lacada de lacada del lacada delacada de lacada del lacada de lacada delacada delacada del lacada del lacad

2FApp.html&state=07e37303-013d-4203-8415-faea284d65b6&response_mode=fragment&response_type=code&scope=FUZZ&nonce=dd598c5f-6e53-461b-8799-2f48ad1d1d

: http://localhost:8080/auth/realms/demo/protocol/openid-connect/auth?client_id=demo&redirect_uri=http%3A%2F%2Flocalhost%3A9506

Settings Keys Roles Client Scopes Mappers Scope Revocation Sessions Offline Access Installation Offline

Matcher Response status: 200 [Status: 200, Size: 3650, Words: 840, Lines: 84]:: Errors: 0 ::
[Status: 200, Size: 3650, Words: 840, Lines: 84] :: Errors: 0 ::
[Status: 200, Size: 3650, Words: 840, Lines: 84]] :: Errors: 0 ::
[Status: 200, Size: 3650, Words: 840, Lines: 84]] :: Errors: 0 ::
[Status: 200, Size: 3650, Words: 840, Lines: 84]] :: Errors: 0 ::
[Status: 200, Size: 3650, Words: 840, Lines: 84]:02] :: Errors: 0 ::
[Status: 200, Size: 3650, Words: 840, Lines: 84]:07] :: Errors: 0 ::
[Status: 200, Size: 3650, Words: 840, Lines: 84]0:32] :: Errors: 0 :::
[43004/43004] :: Job [1/1] :: 892 req/sec :: Duration: [0:01:00] :: Errors: 0 ::: 8 scopes not present in the Client code were identified. Risk: It is potentially possible for a Client to access more resources if the User accepts the new scope requested. The STS does not support broken hashing algorithms like MD5 or SHA1 or even "plain" (reference STS00b)

S 🚧 🗫 http "http://localhost:8080/auth/realms/demo/protocol/openid-connect/auth?client_id=demo&redirect_uri=http://localhost:9500/App.html&sta Strict-Transport-Security: max-age=31536000; includeSubDomain

The code verifier must have a minimum length of 43 positions according to the RFC, so, let's try to start an

"Authorization Code with PKCE" flow with a "plain" code challenge algorithm (code_challenge = code_verifier) and

Risk: It is possible to start a flow that disables protection added by PKCE. It causes the value of the code verifier to be disclosed during the start of the flow.

🕌 ➡ http "http://localhost:8080/auth/realms/demo/protocol/openid-connect/auth?client_id=demo&redirect_uri=http://localhost:9500/App.html&sta

Anyways, these mechanisms are a true added value from a security point of view and, like any system, it is just necessary to ensure that every component in the flow uses recommended and secure settings. It's the main reason

Security Academy, provides additional insights about interesting attack vectors

Let's deliver the right solution for your business

Sign in to your account

Conclusion of my journey with OAuth and OIDC

why the checklist was created, in order to ensure during a security assessment to review a maximum of aspects. I hope that this checklist will be useful, for the defender side as well, in order to allow them to review and monitor the configuration of the involved parties. To go further on the offensive side, the training module dedicated to OAuth 2.0, from the PortSwigger Web

Did you like the article? Find even more articles written by the AppSec team here. Credits

• Dominique Righetto • Elliot Rasch • Julien Ehrhart

flows.

Uncategorized

About Us Career Opportunities Where to meet us Privacy Notice Contact Us

Meet Success

Services Eyeguard Information Security Governance Intrusion Tests - Red Team **Application Security** Network & Security Infrastructure CERT-XLM Eyetools

Training

f y in

Follow us

The Necessity of Cyber Crisis

≺ Share

Exercises The Art of Password Spraying: A

Recent Posts

Comprehensive Analysis Common client-side vulnerabilities

of web applications Categories

Consulting Cyber Blog Post

Advice

General Newsletter The Cyber Blog Times

Search ...

Uncategorized Search

Q

OpenID Connect". Indeed, OAuth and OIDC are more and more common in modern application architecture and my objective was to understand these new concepts and patterns to be able to identify, exploit and prevent security

OAuth 2.0 and OpenID Connect (OIDC) allow centralizing authorization and authentication management. On one side of the coin, it decreases the attack surface of the application by removing the need to implement some errorprone features like authentication and account management. However, on the other side, these new mechanisms are difficult to master and it is easy to introduce a weakness during the setup of the authorization/authentication

CONTACT US