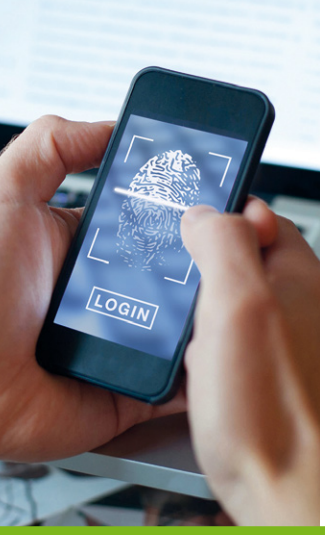

OPENID CONNECT : PRÉSENTATION DU PROTOCOLE ET ÉTUDE DE L'ATTAQUE BROKEN END-USER AUTHENTICATION

Rémi CASSAM CHENAÏ – remi.cassam@protonmail.com
Officier de la Marine nationale - Diplômé ESSI
Olivier LEVILLAIN – olivier.levillain@ssi.gouv.fr
ANSSI - Responsable du centre de formation

mots-clés : OPENID CONNECT / FÉDÉRATION D'IDENTITÉ

L'emploi quotidien de nombreux services sur le Web rend l'utilisation de méthodes d'authentification unifiées très utile. La fédération d'identité avec OpenID Connect est une manière de mettre en œuvre cette authentification unique. Cependant, ce jeu à trois acteurs (utilisateur, fournisseur d'identité, fournisseur de service) ne fonctionne que si tout le monde a la même vision de la situation !

1 La fédération d'identité dans le monde Web

Les applications web sont aujourd'hui devenues omniprésentes dans nos vies. Que ce soit dans la sphère personnelle ou dans un contexte professionnel, nous accédons à de nombreuses applications chaque jour, depuis nos ordinateurs, nos tablettes, nos téléphones ou encore nos télévisions connectées.

1.1 Authentification classique par mot de passe

L'une des problématiques soulevées par la multitude des services disponibles est certainement la multiplication des comptes utilisateurs. Pour utiliser ces applications, il est en effet généralement nécessaire de s'authentifier. Or, la solution la plus courante pour justifier de son identité reste encore aujourd'hui l'authentification par mot de passe. Dans ce cas de figure, l'utilisateur s'identifie avec un identifiant (nom, adresse électronique, pseudonyme…) puis prouve son identité en fournissant le mot de passe associé pour se connecter.

Le schéma par mot de passe est simple et facile à déployer. Il présente néanmoins plusieurs inconvénients de taille. Tout d'abord, il faut bien choisir son mot de passe pour qu'il ne soit pas prédictible facilement (rappel : ni le nom de votre chien, ni votre date de naissance, ni *azerty12* ne sont de bons mots de passe).

Cependant, cela n'est pas suffisant, car il arrive que les fournisseurs de service ne protègent pas correctement votre mot de passe et que ce précieux sésame se retrouve dans la nature. Pour éviter qu'une telle compromission affecte l'ensemble de votre activité numérique, il est donc nécessaire de ne pas réutiliser le même secret sur des services différents, ce qui vous amène à devoir définir (et retenir) des dizaines de mots de passe.

Enfin, la nécessité de s'authentifier individuellement pour chacune de ses applications web est fondamentalement pénible. Or, derrière ce problème d'ergonomie, il faut comprendre que l'utilisateur va rapidement se lasser d'oublier ou de se tromper fréquemment pour finalement réutiliser le même mot de passe (ou le même schéma de mots de passe).

Une manière classique de s'en sortir est d'utiliser un coffre-fort de mots de passe : un outil qui génère, conserve et protège ces précieux sésames, et que l'on déverrouille à partir d'un mot de passe maître. Il devient en effet alors possible de gérer l'authentification de ses différentes identités numériques de manière plus ou moins automatique, en n'ayant qu'un seul mot de passe à retenir. Cette méthode atteint tout de même ses limites quand l'utilisateur change fréquemment d'équipements, et que ceux-ci n'acceptent pas de tels outils (la télévision connectée par exemple).

1.2 L'authentification unique

Heureusement, une autre méthode est disponible pour l'authentification des utilisateurs auprès des services auxquels on souhaite se connecter : l'authentification

unique (SSO pour *Single Sign On* en anglais). Dans le monde bureautique classique de l'entreprise, un moyen répandu d'implémenter ce concept est Kerberos. L'utilisateur s'authentifie la première fois auprès d'un serveur central (KDC, *Key Distribution Center*). À chaque fois qu'il souhaite ensuite accéder à un service (le serveur de messagerie, une imprimante, etc.), il retourne voir le KDC pour obtenir un ticket à présenter au fournisseur du service.

Dans le monde web, ce modèle très centralisé ne peut généralement pas fonctionner et n'est d'ailleurs pas bien adapté. C'est pourquoi un autre modèle d'authentification unique est généralement rencontré : la fédération d'identité.

Si ce n'est pas déjà fait, l'utilisateur commence par se créer un compte auprès d'un fournisseur d'identité (Facebook, Google...). Cette étape consiste la plupart du temps à choisir un identifiant et un mot de passe, mais aussi à renseigner d'autres informations personnelles telles que sa date de naissance.

Ensuite, lorsque l'utilisateur souhaite utiliser un service sur Internet (musique en *streaming*, presse en ligne, réseau social professionnel, etc.), il doit s'enregistrer avec un compte. Cependant, au lieu d'en créer un ad hoc, spécifique au service souhaité, il va lui indiquer qu'il souhaite se connecter via son fournisseur d'identité.

L'utilisateur sera alors redirigé vers celui-ci pour s'authentifier. Une fois cette étape réalisée, les informations d'identité (son identifiant au minimum) seront transmises par le fournisseur d'identité au fournisseur de service auquel souhaite accéder l'utilisateur.

Le modèle contient donc trois types d'acteurs pour lesquels il existe généralement plusieurs instances (voir figure 1) : les utilisateurs (humains ou systèmes automatisés), les fournisseurs d'identité (FI) et les fournisseurs de service (FS). Les principaux acteurs du Web favorisent aujourd'hui l'utilisation de ce modèle qui connaît une forte croissance depuis quelques années.

L'avantage de ce schéma est que l'utilisateur n'a plus qu'à gérer qu'un seul compte, auprès de son fournisseur d'identité. Il peut donc choisir un mot de passe robuste, et en théorie mieux maîtriser les informations personnelles qu'il accepte de transmettre aux fournisseurs de services. Enfin, les services auxquels l'utilisateur accède n'ont pas accès aux accréditations de ce dernier, ce qui limite a priori les conséquences d'une éventuelle compromission.

Le lecteur attentif remarquera néanmoins que dans ce schéma, la perte du mot de passe devient catastrophique, puisque le fournisseur d'identité se trouve être un acteur critique du dispositif. Cependant, il devient également possible de renforcer la sécurité de l'authentification dans ce cas : choisir un mot de passe très robuste, activer l'authentification à double facteur, utiliser des jetons de sécurité U2F. Enfin, en cas de compromission du compte, il suffit de contacter un opérateur pour reprendre la main sur l'ensemble des services.

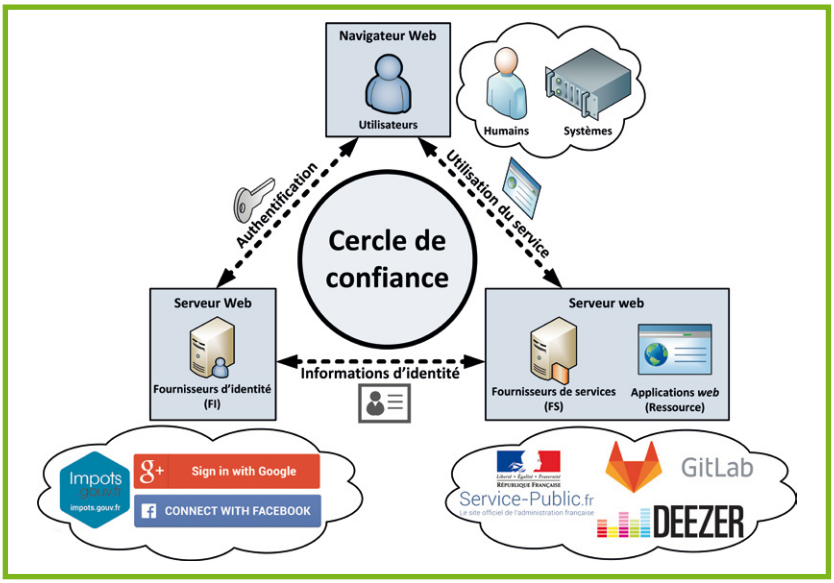

Fig. 1 : Modèle de gestion d'identité fédérée : certains fournisseurs de services peuvent déléguer l'identification et l'authentification de leurs utilisateurs à d'autres fournisseurs d'identité.

2 Standards existants

Pour mettre en place une telle fédération d'identité, il existe essentiellement deux protocoles : SAML et OpenID Connect.

2.1 SAML

SAML (*Security Assertion Markup Language*) est un protocole soutenu par le consortium mondial à but non lucratif OASIS. Il est très populaire dans les fédérations d'identité universitaires, comme celui opéré par RENATER en France. Les spécifications de la version 2.0 **[SAML]** datent de 2005, mais ont fait l'objet d'amendements réguliers depuis.

SAML s'appuie sur le format XML pour transmettre les informations d'identité, aussi appelées assertions. Ces assertions peuvent être chiffrées ou signées à l'aide de XML Encryption et de XML-DSig. Il existe différentes manières pour communiquer les assertions SAML entre les fournisseurs d'identité et de services : en passant par l'utilisateur (*HTTP Redirect Binding* ou *HTTP POST Binding*), ou en utilisant un canal de communication dédié entre le FI et le FS (*HTTP Artifact Binding*, qui utilise le protocole SOAP). En pratique, c'est la première méthode (*HTTP Redirect Binding*), qui repose sur une simple requête GET, qui est la plus employée.

Bien que SAML soit aujourd'hui un standard éprouvé, documenté et encore très utilisé, il souffre de plusieurs limitations (gestion du consentement de l'utilisateur hors spécification, déconnexion unique difficile à gérer, etc.) et est techniquement complexe à mettre en œuvre. De plus, avec l'arrivée de protocoles plus récents, SAML dispose désormais d'une communauté un peu moins active.

2.2 OpenID Connect

OpenID Connect, publié en 2014 **[OIDC]**, est une évolution du standard OpenID et s'appuie sur le protocole OAuth 2.0 **[OAUTH]** en y ajoutant une couche d'authentification.

OpenID Connect repose sur le standard JSON Web Token (JWT) pour échanger des informations d'identité ou toute autre information pertinente pour le FS. De manière similaire aux assertions SAML, ces jetons peuvent être signés et chiffrés.

Il existe plusieurs manières pour le protocole de fonctionner (appelées « cinématiques »). La méthode dite du code d'autorisation (*Authorization Code Mode*) est aujourd'hui la plus répandue et reste généralement très simple à mettre œuvre.

L'avantage de cette cinématique est qu'elle réduit les risques de compromission. En effet, bien que certaines informations passent par l'intermédiaire de l'utilisateur, le jeton d'identité ne transite en revanche pas par celui-ci. Ces informations sensibles sont directement échangées entre le FI et le FS, ce qui permet de restreindre les possibilités d'un utilisateur (ou d'un attaquant ayant compromis celui-ci) qui chercherait à modifier ce jeton. C'est pour cela que cette méthode est recommandée, notamment par l'OWASP, dès qu'une communication entre les fournisseurs (*back channel*) est réalisable..

Bien qu'OpenID Connect soit un standard relativement jeune, il est porté par la fondation OpenID, soutenue par de nombreuses entreprises influentes comme Google, Microsoft ou Oracle.

Si SAML semble adapté à des fédérations d'identité dans des environnements plus cadrés, OpenID Connect a vocation à rendre possible la fédération d'identité dans un environnement très ouvert. Certaines fonctionnalités optionnelles du protocole témoignent en particulier de cette philosophie : la détermination à la volée du FI associé à l'utilisateur ou l'enregistrement dynamique d'un FS.

Nous avons donc choisi de nous intéresser plus en détail à OpenID Connect.

3 Description d'OpenID Connect

Comme indiqué précédemment, il existe plusieurs cinématiques OpenID Connect. Nous décrivons ici uniquement la méthode du code d'autorisation (*Authorization Code Mode*), la plus répandue et la plus recommandée.

3.1 Terminologie du protocole

Avant de présenter le fonctionnement technique du protocole, voici quelques définitions propres à OIDC :

- *OpenID Provider* (OP) : il s'agit du fournisseur d'identité (FI).
- *Relying Party* (RP) : c'est le fournisseur de services (FS).
- *End-User* : ce terme désigne l'utilisateur.

Pour faciliter la compréhension de la suite de l'article, les termes génériques fournisseur d'identité (FI), fournisseur de service (FS) et utilisateur seront préférés aux appellations spécifiques du protocole.

- *ID Token* : jeton d'identité au format *JSON Web Token* (JWT) signé et/ou chiffré permettant au FI de certifier l'authentification de l'utilisateur au FS.
- *Access Token* : jeton d'autorisation permettant au FS d'accéder à une ressource de l'utilisateur protégée et généralement hébergée par le FI. Ces ressources sont la plupart du temps des informations d'identité (attributs) supplémentaires concernant l'utilisateur authentifié. Le consentement de l'utilisateur avant toute transmission de ces informations est obligatoire.

3.2 Détermination du fournisseur d'identité

La première étape consiste pour le FS à déterminer le FI associé à l'utilisateur. Il existe plusieurs façons d'y arriver.

Une première méthode consiste à associer tous les utilisateurs au même FI. C'est évidemment la solution la plus simple, mais elle ne permet pas au sens strict la mise en œuvre de la fédération d'identité.

Il est également possible de proposer une page statique contenant la liste des FI reconnus par le FS. C'est alors l'utilisateur lui-même qui choisit le FI auprès de qui il est enregistré. On parle alors d'*association statique*.

Enfin, la dernière manière, appelée « *Issuer discovery* » dans le protocole, consiste à associer dynamiquement un FI à l'utilisateur. Pour cela, le FS utilise les éléments d'identité qu'il a en sa possession (typiquement, une adresse électronique), et en déduit le FI correspondant.

La seconde option (l'association statique) est celle qui est le plus souvent retenue, comme le montre la figure 2.

Dans tous les cas, le FS va récupérer certaines métadonnées propres au FI. Les *endpoints* que l'on peut citer en particulier sont les URI déclarées par le FI et qui permettent par exemple :

- la bonne redirection de l'utilisateur pour son authentification (`authorization_endpoint`) ;
- la récupération par le FS des données personnelles de l'utilisateur authentifié (`token_endpoint`) ;
- l'enrôlement dynamique du FS auprès du FI (`registration_endpoint`).

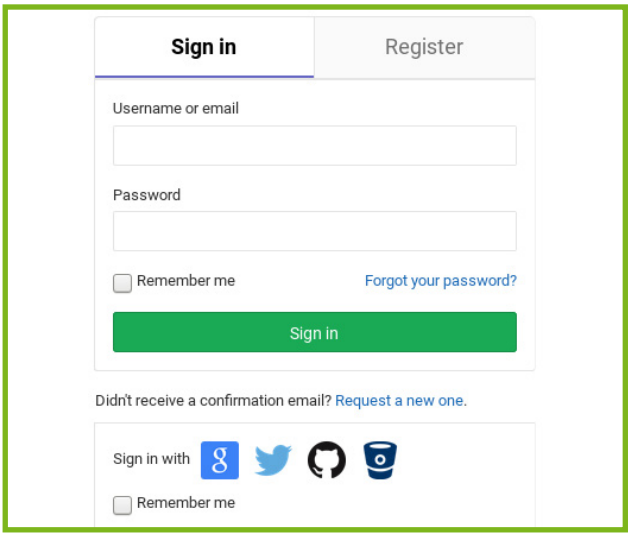

Fig. 2 : Exemple de formulaire mettant en œuvre l'association statique du FI (GitLab propose de s'authentifier avec un compte GitLab, ou via OIDC en choisissant l'un des FI listés en bas du formulaire).

3.3 Enrôlement du fournisseur de service

La seconde étape consiste si ce n'est déjà fait à faire enrôler le FS auprès du FI. Cette phase peut encore une fois s'effectuer de manière statique ou dynamique. Dans le cas statique, les administrateurs du FI et du FS renseignent les informations en amont de l'ouverture du service aux utilisateurs. Dans le cas dynamique (*Dynamic Client Registration*), les informations sont échangées à la volée, de façon automatique, entre le FI et le FS, lorsqu'un utilisateur renseigne un FI pour la première fois auprès du FS.

Quelle que soit l'option choisie, l'établissement de ce lien, assimilable à un contrat, donne lieu à l'échange des informations suivantes :

- **`client_id`** : identifiant unique du FS auprès du FI. Il s'agit d'une valeur publique, généralement générée aléatoirement par le FI ;
- **`client_secret`** : cette valeur est optionnelle, mais est généralement utilisée pour authentifier le FS lorsqu'il récupère un code d'autorisation auprès du FI. Elle doit rester secrète et n'être connue que par les deux acteurs ;
- **`redirect_uri`** : URL appartenant au domaine du FS utilisée par le FI après l'authentification de l'utilisateur pour renvoyer la réponse d'autorisation au FS (via l'utilisateur).

3.4 Cinématique du « code d'autorisation »

Une fois le couple FS / FI établi lors de la phase d'enrôlement, la phase d'authentification de l'utilisateur peut commencer. La figure 3 fait apparaître la cinématique

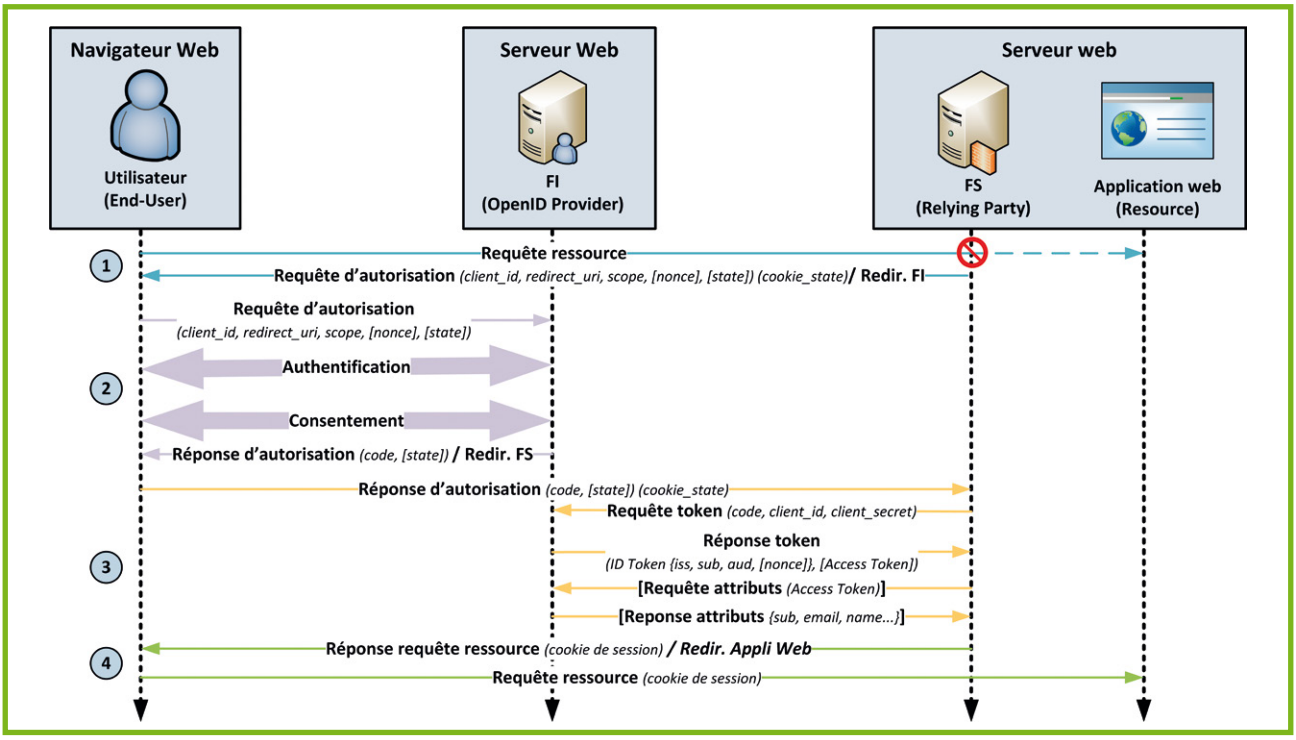

Fig. 3 : Cinématique du code d'autorisation.

avec les principaux paramètres échangés (ceux considérés dans le standard comme non obligatoires - seulement recommandés ou optionnels - sont encadrés entre crochets). Afin de comprendre le processus mis en place par OIDC et l'attaque présentée au paragraphe suivant, détaillons leur signification.

- **client_id**, **client_secret**, **redirect_uri** : paramètres explicités précédemment et identiques à ceux échangés lors de la phase d'enrôlement ;
- **scope** : liste les ressources de l'utilisateur protégées par le FI auxquelles souhaite accéder le FS. Il s'agit généralement d'informations d'identité supplémentaires tels que l'adresse électronique ;
- **nonce** : valeur opaque générée par le FS lors de la requête d'autorisation. Le FI la reçoit via l'utilisateur lors du renvoi de cette requête et l'intègre ensuite à l'*ID Token*. Le FS se charge de comparer la valeur stockée dans la session de l'utilisateur à la valeur reçue dans l'*ID Token* (mécanisme anti-rejeu) ;
- **state** : valeur opaque générée par le FS lors de la requête d'autorisation. Bien que cela ne soit pas imposé, mais seulement mentionné par le protocole, ce paramètre est généralement accompagné d'un cookie (appelé dans la suite de cet article **cookie_state**) contenant une autre valeur opaque. Ces deux valeurs sont ensuite renvoyées in fine par le navigateur web de l'utilisateur avec le code d'autorisation. À travers la vérification de la cohérence du **state** et du **cookie_state**, le FS s'assure ainsi qu'il s'agit bien du même utilisateur entre la requête initiale et la réponse d'autorisation (mécanisme anti *cross-site request forgery*) ;
- **cookie_session** : cookie permettant de faire le lien entre l'utilisateur authentifié et sa session sur le FS.

Bien que ce mécanisme ne fasse pas à proprement parlé partie du protocole OIDC, il est aujourd'hui largement utilisé. Le navigateur de l'utilisateur le reçoit classiquement une fois que le FS a validé les informations d'identité récupérées auprès du FI et qu'il a effectué son contrôle d'accès.

Étape 1. L'utilisateur non authentifié essaye d'accéder à une ressource protégée. Sa requête est interceptée par le FS qui prépare une requête d'autorisation. Celle-ci est renvoyée à l'utilisateur qui la présente à son tour à son fournisseur d'identité via l'**authorization_endpoint**.

Étape 2. Le FI vérifie la requête d'autorisation puis invite l'utilisateur à s'authentifier. Ce dernier est ensuite sollicité pour autoriser la transmission des informations demandées par le FS dans le paramètre **scope**. Son consentement acquis, le FI transmet au FS via le navigateur de l'utilisateur une réponse d'autorisation contenant notamment le code d'autorisation.

Étape 3. Le FS vérifie la réponse d'autorisation fournie par l'utilisateur puis en extrait le code d'autorisation. Le FS transmet alors ce code via une requête de token au FI. Ce message, authentifié à l'aide du *client_ID* et du *client_secret*, est envoyé sur le **token_endpoint**. Le FI authentifie le FS, vérifie le code d'autorisation et envoie au FS le jeton d'identité (*ID Token*) correspondant à l'utilisateur authentifié. Un *Access Token* peut également être fourni à ce stade pour récupérer des informations d'identité supplémentaires. Cette démarche s'effectuera le cas échéant par le FS via une requête d'attributs.

Étape 4. Le FS vérifie l'*ID Token* et retrouve l'identité de l'utilisateur pour effectuer son contrôle d'accès. Si l'utilisateur a le droit d'accéder au service désiré, le FS lui délivre la ressource demandée initialement.

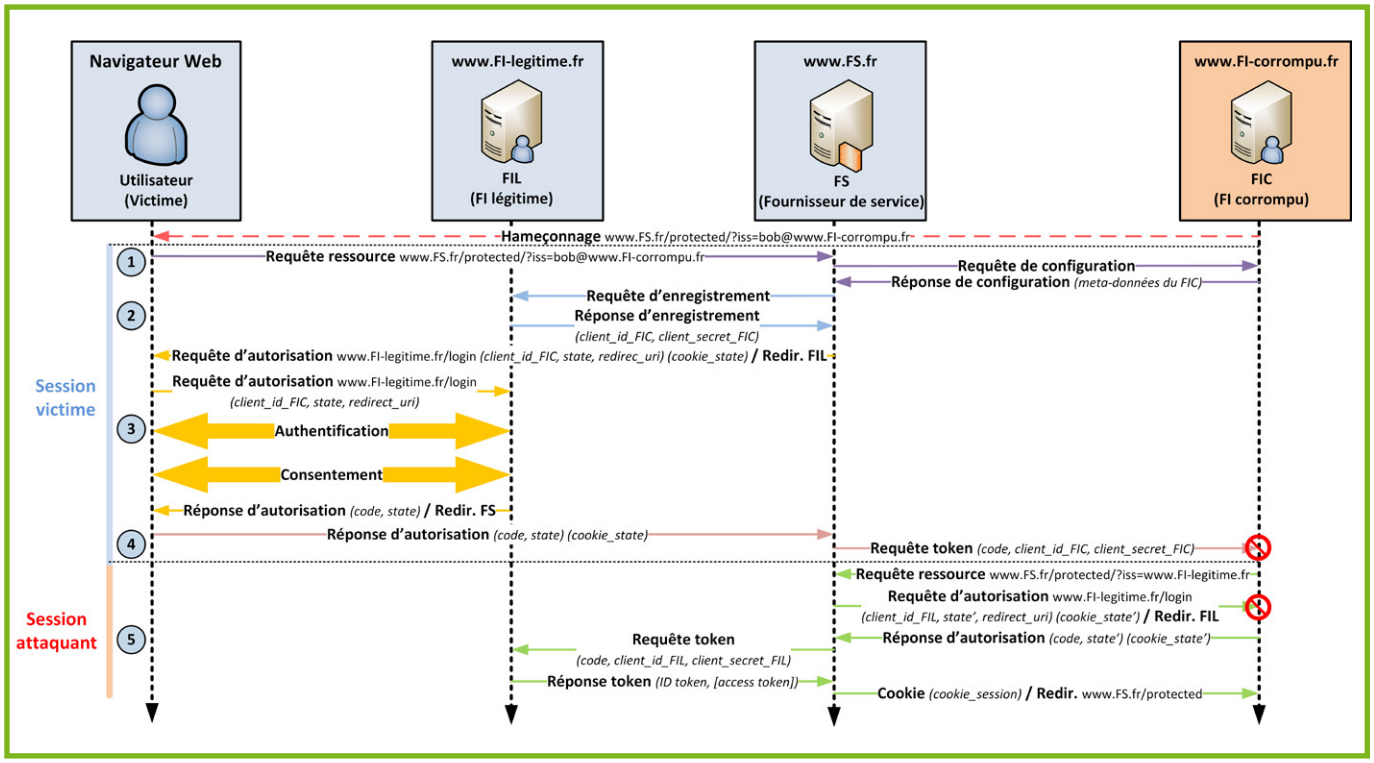

Fig. 4 : Déroulement de l'attaque Broken End-User Authentication.

4 L'attaque Broken End-User Authentication

En 2016, une équipe de chercheurs de l'université de la Ruhr à Bochum a publié une analyse sur la sécurité du protocole OIDC **[BEA]**. Cette étude a en particulier mis au jour une attaque intitulée *Broken End-User Authentication*.

Cette attaque est intéressante, car elle montre que dans un standard complexe comme OpenID Connect, le diable se cache dans les détails.

4.1 Hypothèse et modèle de l'attaquant

Avant toute chose, on suppose que l'attaquant évolue dans le contexte suivant :

- TLS est systématiquement utilisé lors des échanges entre les différents acteurs et l'attaquant ne peut intercepter ou modifier ces communications ;
- les logiciels utilisés (système d'exploitation, navigateurs, serveurs web, etc.) ne présentent pas de faille exploitable ;
- les secrets utilisés dans OIDC sont générés correctement : l'attaquant n'a pas les moyens d'effectuer en temps raisonnable une attaque générique pour retrouver ces valeurs ;
- le champ `state` (et le cookie associé) est utilisé par le FS dans sa requête d'autorisation, mais pas le champ `nonce` qui reste optionnel dans les spécifications.

On s'intéresse à une victime (un utilisateur final), enregistré auprès d'un fournisseur d'identité légitime (FIL). Il utilise régulièrement un fournisseur de services donné, FS, auprès duquel il s'authentifie en utilisant son identité du FIL.

L'objectif de l'attaquant est d'accéder aux ressources appartenant à la victime et stockées par le FS.

L'attaquant peut mettre en place un fournisseur d'identité corrompu (FIC) et initier des communications vers les acteurs légitimes. De plus, on suppose que l'attaquant peut réaliser des campagnes d'hameçonnage auprès de sa victime pour l'inviter à suivre un lien corrompu.

On suppose enfin que les mécanismes de découvertes de FI (*Issuer discovery*) et d'enrôlement automatique (*Dynamic Client Registration*) sont activées chez les acteurs légitimes.

4.2 Déroulement de l'attaque

Étape 1 (association de la victime avec le FI corrompu). L'attaquant envoie par hameçonnage un lien pointant vers le FS. Il contient dans les paramètres URL une information d'identité (une adresse mail par exemple) associée au domaine de l'attaquant. Lorsque l'utilisateur

suit le lien, le FS légitime reçoit sa requête et utilise l'information d'identité reçue pour associer, via le service *Issuer discovery* , l'utilisateur au FI corrompu. Un dialogue propre au service *Issuer discovery* s'installe ensuite entre le FS et le serveur de l'attaquant pour récupérer les métadonnées. Le FI corrompu lui transmet les informations suivantes (liste non exhaustive), mêlant des informations relatives au FI légitime et d'autres concernant le FI corrompu :

```
{
  "issuer" : "https://www.FI-corrompu.fr",
  "registration_endpoint" : "https://www.FI-legitime.fr/register" ,
  "authorization_endpoint" : "https ://www.FI-legitime.fr/login" ,
  "token_endpoint" : "https://www.FI-corrompu.fr/token"
}
```

Étape 2 (enrôlement du FI corrompu). Bien que le FS soit déjà enregistré auprès du FI légitime, il se base sur le champ **`issuer`** des métadonnées pour identifier le FI. En l'occurrence, il ne s'est jamais enrôlé auprès du FI corrompu. Le FS lance donc une procédure d'enregistrement à travers son service *Dynamic Client Registration* à l'URL indiquée dans le champ **`registration_endpoint`** (https://www.FI-legitime.fr/register). Le FI légitime lui envoie un identifiant (**`client_id_FIC`**) et un secret (**`client_secret_FIC`**) dont l'attaquant n'a pas connaissance.

Étape 3 (authentification de la victime). Une fois enregistré, le FS génère sa requête d'autorisation et renvoie la victime s'authentifier à l'adresse indiquée dans le champ **`authorization_endpoint`** (https://www.FI-legitime.fr/login). Les phases d'authentification et de consentement qui s'en suivent s'effectuent sans difficulté puisque la victime fait confiance au FI légitime qui gère son identité. L'utilisateur reçoit ensuite un code d'autorisation qu'il renvoie au FS.

Étape 4 (vol du code d'autorisation). Afin de recevoir l'*ID Token* lié à l'utilisateur, le FS envoie ce code d'autorisation, ainsi que le **`client_id_FIC`** et le **`client_secret_FIC`** à l'adresse indiquée dans le champ **`token_endpoint`** des métadonnées (https://www.FI-corrompu.fr/token). À ce stade, l'attaquant reçoit donc trois informations : le **`client_id_FIC`**, le **`client_secret_FIC`** ainsi que le code d'autorisation. Ces deux dernières informations sont particulièrement sensibles, car elles ne sont pas supposées être révélées à une tierce personne.

Étape 5 (usurpation d'identité). L'attaquant initie sa propre session avec le FS en lui envoyant une requête spécifiant qu'il est associé au FI légitime. Le FS est déjà enrôlé auprès du FI légitime et a déjà reçu ses métadonnées. Il génère donc directement une requête d'autorisation qu'il envoie à l'attaquant. Celui-ci ignore la redirection vers le FI légitime et renvoie au FS une réponse d'autorisation contenant le code d'autorisation précédemment récupéré. Lorsque le FS envoie au FI légitime une requête token contenant ce code, il reçoit en retour l'*ID Token* associé à la victime. À ce moment, l'attaquant a usurpé l'identité de la victime auprès du FS et peut donc accéder à ses ressources.

4.3 Analyse des causes

4.3.1 Absence de mécanisme anti-rejeu

L'usurpation d'identité est notamment rendue possible par la réutilisation par l'attaquant du code d'autorisation de la victime précédemment récupéré par l'intermédiaire du FS. L'utilisation dans la requête d'autorisation du paramètre **`nonce`** généré par le FS permet à celui-ci de corréler l'*ID Token* reçu en échange du code d'autorisation avec la session de l'utilisateur. Dans notre scénario, l'usurpation d'identité n'aurait donc pas pu être réalisée dans ces conditions puisque les **`nonces`** des deux sessions auraient été différents.

4.3.2 Contrôle de la phase d'enrôlement

La cause initiale provient d'une manipulation des métadonnées envoyées par le FIC au FS. Certains *endpoints* font référence au FIL et d'autres au FIL, ce qui engendre l'enregistrement du FS auprès du FIL alors que l'utilisateur est associé au FIC... Il aurait été utile de vérifier que les métadonnées envoyées par le FI lors de son enrôlement, et notamment les *endpoints*, correspondent bien au domaine du FI déclaré, ce qui aurait contré l'attaque. Cette vérification est évidemment plus compliquée à faire lorsque l'enregistrement peut être fait de manière dynamique.

4.3.3 Absence de garanties d'intégrité et de confidentialité sur certains échanges

Une fois son FIC enrôlé, deux étapes permettent à l'attaquant de réaliser son attaque si l'on met de côté la campagne d'hameçonnage :

- la première consiste à récupérer lors de l'étape 4 le code d'autorisation de la victime généré par le FIL. Garantir la confidentialité de ce code dans les échanges entre les acteurs concernés (soit du point de vue du FIL : lui-même et le FS) n'aurait pas permis à l'attaquant d'accéder à cette information capitale ;
- la seconde consiste pour l'attaquant à fabriquer à l'étape 5 une réponse d'autorisation ad hoc contenant le code d'autorisation de la victime récupéré à l'étape précédente. Garantir l'intégrité de cette réponse d'autorisation, normalement générée par le FI, aurait permis d'éviter une telle manipulation.

Malheureusement, le protocole ne permet pas en l'état actuel de garantir l'intégrité et la confidentialité sur ces deux échanges, bien que le FI et le FS disposent d'un secret partagé pour réaliser ces deux opérations.

Au-delà de l'attaque présentée dans cette section, d'autres scénarios possibles ont été décrits pour mettre

en cause les propriétés de sécurité d'OpenID Connect. Ils reposent sur des hypothèses différentes et suivent des cheminements différents. Le lecteur intéressé pourra notamment consulter l'attaque IdP Confusion **[SOK]**.

5 Contremesures et recommandations

5.1 Cela va sans dire...

Tout d'abord, les recommandations classiques liées à la mise en œuvre de ce type de mécanismes sur le Web doivent être appliquées.

En premier lieu, l'ensemble des communications doit être protégé par TLS.

Il est également essentiel de s'assurer de la bonne entropie de l'ensemble des secrets générés : ceux-ci doivent être tirés aléatoirement, et être d'une taille suffisamment longue. Cette recommandation s'applique évidemment au code d'autorisation, mais également aux champs `state`, `nonce` et au `client_secret`.

Afin de pouvoir détecter et déverminer les problèmes éventuels, il faut enfin journaliser les messages échangés. Cela permet dans certains cas de détecter des anomalies, au moins a posteriori. Ici, le FS reçoit coup sur coup deux codes d'autorisation identiques, issus de FI distincts, ce qui ne devrait a priori pas arriver.

5.2 Durcissement d'OpenID Connect

Il faut d'abord utiliser tous les mécanismes mis à notre disposition dans la spécification pour garantir que la vision des différents acteurs soit cohérente. Il est donc indispensable d'utiliser les champ `nonce`, `state` et de mettre en place un `cookie_state` correspondant.

Le protocole prévoit également une phase d'authentification du FS auprès du FI dans la requête token. Au lieu d'envoyer le mot de passe `client_secret` directement, une méthode d'authentification plus sécurisée doit être privilégiée. Cela peut passer soit par l'utilisation de clefs asymétriques, soit par l'utilisation du mécanisme symétrique `client_secret_jwt` qui repose sur HMAC et qui permet de ne pas divulguer le `client_secret`..

Au-delà de la protection du canal de communications, il est également utile de protéger les objets transportés, les jetons JWT. Pour cela, il est fortement recommandé de signer et de chiffrer les jetons JWT. Comme indiqué dans la section 4.3, certains paramètres ne peuvent pas être incorporés dans un JWT comme le code d'autorisation présent dans la requête token et dans la réponse d'autorisation.

En revanche, la spécification permet d'utiliser les JWT pour protéger les paramètres de la requête d'autorisation, notamment les aléas comme `state` ou `nonce`. Il faut donc imposer que ces éléments ne soient transmis que chiffrés et signés, uniquement au sein de jetons JWT. Ainsi, on protège ces éléments sensibles et on prive l'attaquant de leur connaissance, ce qui peut compliquer une usurpation d'identité.

5.3 Un mot sur l'enregistrement à la volée

Enfin, dans l'attaque décrite ci-dessus, une des causes est l'autorisation pour les FS de s'enregistrer de manière automatique auprès d'un FI. En pratique, l'enrôlement est une opération importante qui correspond à un contrat entre les deux fournisseurs. Il semble donc important que les informations utilisées / recueillies lors de cette étape soient vérifiées scrupuleusement, idéalement via un contrôle humain. Mais sans aller jusque-là, l'utilisation par un FI donné du même `registration_endpoint` qu'un autre FI, comme c'est le cas ici, devrait au moins lever une alerte !

De manière plus générale, il faut se méfier des modes automatiques de découverte et d'enregistrement de fournisseurs.

Conclusion

Le concept d'authentification unique est très utile dans le monde du Web. Combiné avec d'autres outils (coffres forts de mots de passe, authentification à deux facteurs), la fédération d'identité peut aider à améliorer de manière notable la sécurité des communications et des accès. OpenID Connect est aujourd'hui la solution la plus en vogue, en particulier dans des contextes de déploiement largement ouverts.

Cependant, si ce standard présente de bonnes propriétés, le fonctionnement du protocole est assez complexe, et peut mettre en jeu un nombre important d'acteurs, pas forcément tous de confiance. Il est donc important d'appliquer de bonnes pratiques à son déploiement OpenID Connect et d'utiliser si possible une implémentation certifiée par la fondation OpenID. L'objectif est d'assurer que l'ensemble des acteurs légitimes (utilisateur, FI et FS) ait une vision cohérente de la situation, et ce malgré l'utilisation d'échanges bilatéraux. ■

Remerciements

Les auteurs tiennent à remercier Vincent Lancino qui a bien voulu relire cet article.

Retrouvez toutes les références de cet article sur le blog de MISC : https://www.miscmag.com/ .