BOULECHFAR Djameleddine
CORTINOVIS Enzo
GENTET Willy

Etude des autorisations web

Ce document présente les résultats des recherches que nous avons effectuées sur le but et le fonctionnement des systèmes d'autorisations web que sont les ID de session et les JWT.

I - Introduction

La majeure partie des applications en ligne permettent aux utilisateurs de créer des comptes et de s'authentifier afin qu'ils puissent effectuer certaines actions.

Selon le type d'application, il faut mettre en place des systèmes de sécurité afin de s'assurer que seules les personnes autorisées aient accès à certaines ressources. Par exemple, sur une application bancaire, il n'est pas envisageable qu'une personne puisse effectuer des actions sur le compte d'autrui.

Pour cela, un moyen simple est de demander aux utilisateurs d'entrer leur mot de passe à chaque fois qu'ils souhaitent effectuer une action, mais cela serait beaucoup trop lourd pour des applications n'ayant pas une importance aussi grande que la gestion de comptes bancaires. De plus, le protocole HTTP utilisé pour naviguer sur internet est sans état, il ne permet donc pas de stocker des informations sur la navigation, il faut donc utiliser d'autres solutions.

Différents moyens ont donc été imaginés et créés afin de permettre d'authentifier facilement les différents utilisateurs afin de vérifier leur identité et leurs autorisations tout en n'ayant pas à s'identifier à chaque action.

II - Session ID

Un moyen pour authentifier un utilisateur connecté est d'utiliser des ID de sessions : le serveur stocke pour chaque utilisateur connecté un ID qui lui permet d'être reconnu pour la session courante.

Les ID de sessions fonctionnent globalement de la manière suivante :

1. Lors de la connexion de l'utilisateur, une requête est envoyée au serveur. Le serveur va s'assurer que le mot de passe fourni par l'utilisateur est correct, et si c'est le cas, il va créer un identifiant unique, qui peut prendre la forme d'un entier, d'une chaîne de caractères, etc, et le stocker dans une base de données en le faisant correspondre à un ensemble d'informations sur l'utilisateur, par exemple, son vrai ID sur la base de données de l'application.
2. Le serveur renvoie cette ID de session au client par le biais d'un cookie sécurisé via le protocole HTTP.
3. Dès que l'utilisateur souhaite effectuer une action (modification d'une donnée, accès à une autre page, revenir sur le site après avoir navigué sur internet, ...), la requête correspondante est accompagnée de l'ID de session qui lui a été fourni par le serveur. Le serveur va alors chercher dans sa base de données l'utilisateur correspondant à cette ID de session et vérifier si la personne à l'origine de la requête est bien l'utilisateur connecté et si il a bien le droit d'accéder à la ressource demandée.

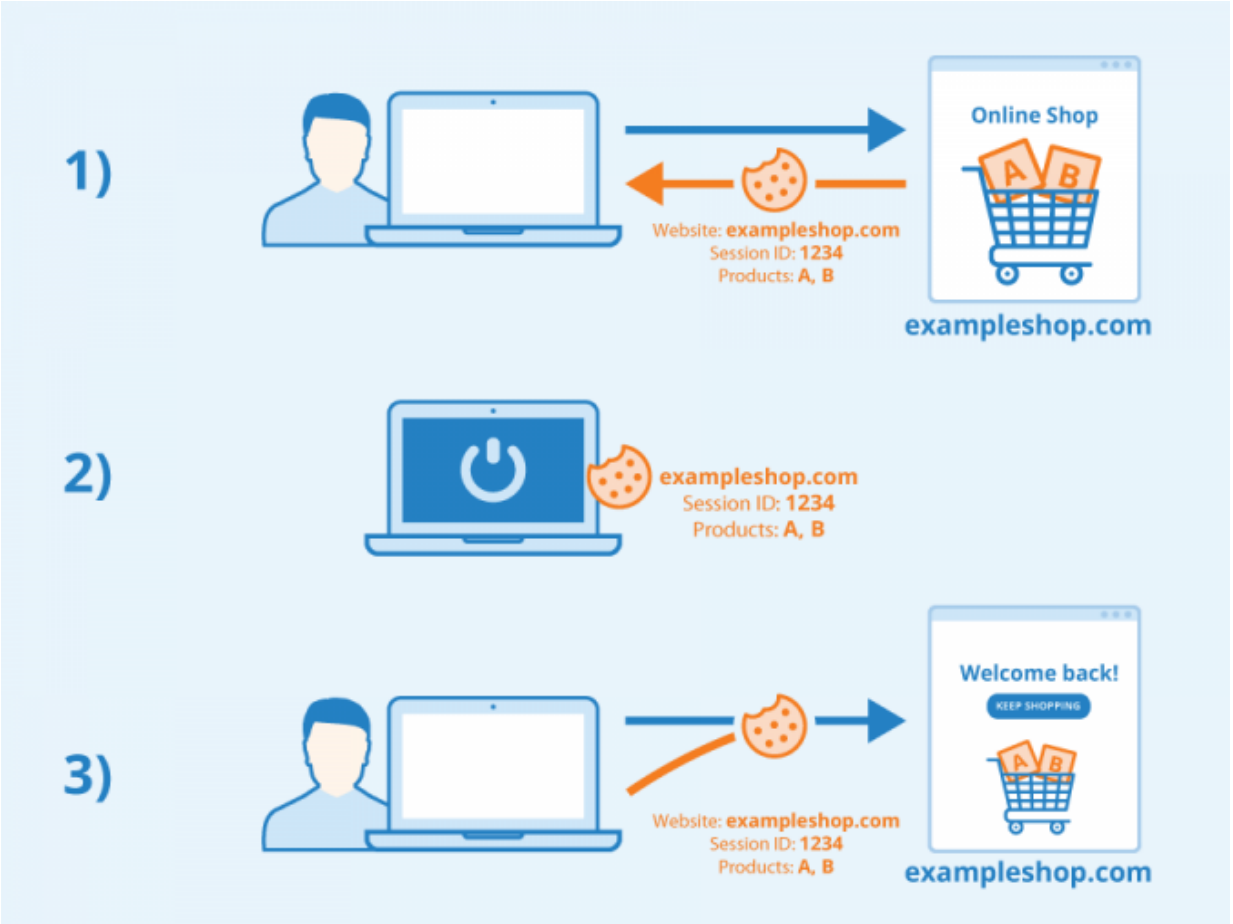


Schéma du fonctionnement global des ID de session

Remarque :
Les ID de session peuvent aussi être transmis via l'URL, ou bien dans un champ masqué, bien que cela soit beaucoup plus rare.

Ce processus est donc considéré comme "stateful" puisqu'on enregistre l'état de l'utilisateur dans une base de données.

III - JWT

Un autre moyen d'authentifier les utilisateurs est d'utiliser des JWT (JSON Web Token). Le fonctionnement est similaire aux ID de sessions mais possède quelques différences :

1. Comme pour les ID de sessions, lors de la connexion de l'utilisateur, une requête est envoyée au serveur. Après vérification du mot de passe, le serveur crée un JWT avec les informations de l'utilisateur connecté et le signe avec sa clé privée.
2. Le serveur renvoie le JWT au client qui le stocke localement.
3. Dès que l'utilisateur souhaite effectuer une action, la requête correspondante est accompagnée du JWT qui lui a été fourni par le serveur (le plus souvent dans le header "Authorization", préfixé par "Bearer "). Étant donné que le JWT a été signé par le serveur, il suffit au serveur de vérifier la signature de ce dernier avec sa clé privée. Si la signature est correcte, alors l'utilisateur est authentifié et le serveur lui renvoie une réponse à la requête.

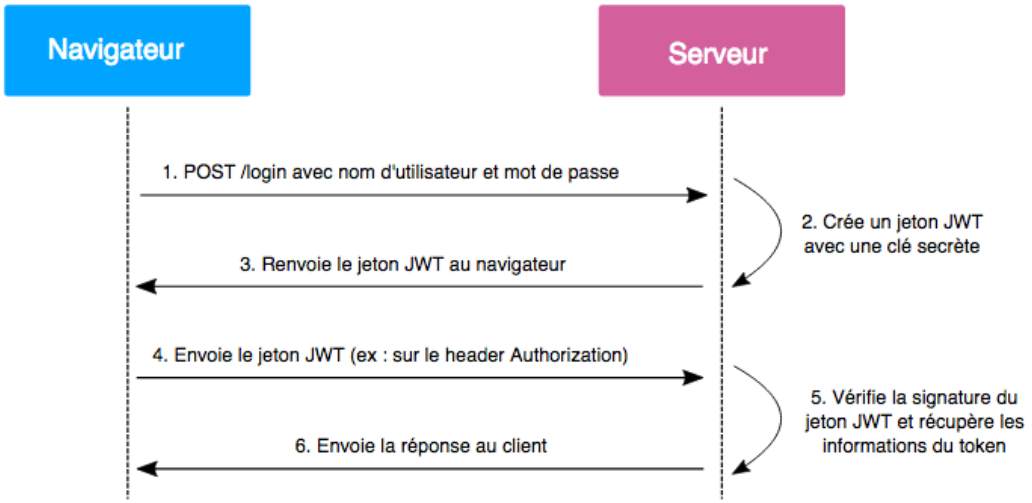
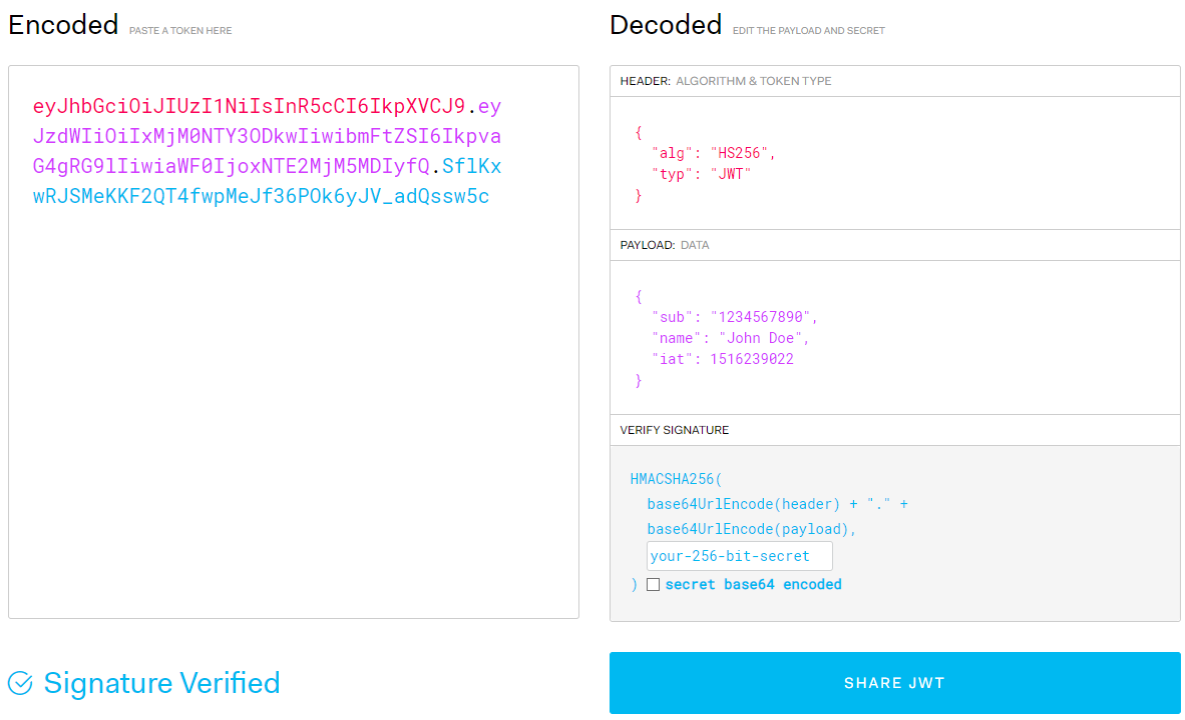


Schéma du fonctionnement global des JWT

Format d'un token :

Un JWT est composé de 3 parties distinctes, encodées en base 64 et séparées par des points. On obtient un format de la sorte : "XXXXX.XXXXXXX.XXXXXX".



Contenu d'un JWT et fonctionnement de la signature

La première section est le header. Elle contient principalement le type d'algorithme utilisé pour signer le token.

Il est possible d'utiliser des systèmes de chiffrement symétriques comme des algorithmes HMAC. Dans ce cas, on va utiliser une fonction cryptographique (par exemple SHA256) pour signer le token.

Il est également possible d'utiliser des systèmes de chiffrement asymétriques comme RSA. Dans ce cas, tout le monde pourra vérifier le token, puisque tout le monde dispose de la clé publique.

La deuxième section est le payload. Elle contient les données de JWT, toujours au format JSON. Les champs les plus courants sont :

- sub → subject : l'utilisateur concerné (souvent son ID unique).

- iat → issued at : la date à laquelle le token a été attribué.
- exp → expires : la date à laquelle le token expire.
- etc

La troisième section du token est la signature de celui-ci. Elle est obtenue en concaténant les informations du header et du payload encodés en base 64 et en les signant grâce à l'algorithme précisé dans le header. Ainsi, il est impossible pour un utilisateur de modifier les informations d'un JWT sans le rendre invalide, en effet si les informations du header ou du payload sont modifiées avant l'envoi du token au serveur, la signature calculée par le serveur ne correspondra pas à cette troisième section, et le token sera donc invalidé, et l'authentification refusée (et inversement, si la signature est modifiée, elle ne correspondra plus aux informations avec lesquelles elle a été générée).

IV - Session ID VS JWT

Comme nous avons pu le voir, les ID de session et les JWT permettent d'authentifier un utilisateur afin de lui autoriser, ou pas, l'accès à certaines ressources.

Chacune des méthodes possède ses avantages et ses inconvénients, et l'utilisation de l'une ou l'autre n'est pas un simple choix, cela dépend aussi des spécificités et des contraintes de l'application.

L'avantage majeur des JWT par rapport aux ID de sessions est que le serveur ne stocke aucune information sur l'authentification des utilisateurs. En effet, étant donné que les ID de sessions doivent être stockés sur une base de données, cela pose des problèmes de mise à l'échelle.

Premièrement, dans le cadre de l'utilisation d'ID de sessions, plus le nombre d'utilisateurs est élevé, plus la taille de la base de données l'est. De plus, de nos jours, la plupart des grosses applications possèdent plusieurs serveurs permettant de répartir la charge d'utilisateurs à travers plusieurs machines.

Avec les ID de session, si l'utilisateur doit être redirigé vers un autre serveur suite à une augmentation du trafic sur le serveur actuel, ou si l'application globale fait naviguer l'utilisateur entre différentes applications plus petites, il sera amené à se

reconnecter, à moins qu'un partage des ID de sessions entre les différents serveurs soit mis en place, ce qui est une solution peu envisageable.

Avec les JWT, ce souci n'existe pas, les différents serveurs d'une application peuvent partager la même clé privée, et ainsi vérifier la signature du JWT de l'utilisateur. Cela peut être utile dans le cadre d'une application sous forme de micro-services qui possède un frontend, et une ou plusieurs API servant de backend : chacune de ces applications peut authentifier de la même manière l'utilisateur effectuant une requête.

Cependant, le fait que les JWT soient stockés côté client n'est pas qu'un avantage, en effet, bien qu'ils soient très sécurisés, le fait de pouvoir accéder au JWT peut donner quelques informations sur la longueur des informations de ce dernier ou encore une idée de sa structure. En utilisant des ID de sessions, seul le serveur accède aux informations liées à ces ID, ce qui rend cette méthode un peu plus sécurisée.

Pour finir, bien qu'un JWT ne soit pas un objet très lourd, il est généralement bien plus conséquent qu'un ID de session, ce qui fait que la quantité de bande passante utilisée dans le cadre d'applications à grande échelle peut être beaucoup plus élevée avec des JWT, bien que cela dépende également d'autres facteurs.

V - Vulnérabilités

Bien que les méthodes d'authentification par ID de sessions ou JWT soient très sûres, il a existé, et il existe encore certaines vulnérabilités, plus ou moins complexes à mettre en place et à exploiter, nous allons passer en revue quelques-unes pour chacune des méthodes.

1. Session ID

- **Session Expiration**: Les ID de sessions peuvent rester valides pendant une période prolongée, ce qui augmente le risque de compromission. Les sessions expirées peuvent être réactivées par des attaquants pour accéder à des informations sensibles ou effectuer des actions malveillantes. C'est une vulnérabilité relativement facile à exploiter et à éviter, car il suffit de définir

une durée de vie maximale pour les identifiants de session et d'inclure des fonctionnalités de déconnexion automatique après une période d'inactivité.

- **Session Leakage**: Les identifiants de session peuvent être exposés à des tiers non autorisés si les applications ne sont pas correctement configurées. Les fuites de session peuvent se produire lorsqu'un site Web hébergeant une application mal configurée envoie des informations d'identification ou des identifiants de session sensibles à un tiers non autorisé. Cette vulnérabilité est un peu plus difficile à exploiter, mais peut être évitée en limitant les informations sensibles contenues dans les identifiants de session ainsi qu'en chiffrant les identifiants en transit, en mettant en place des mesures de sécurité supplémentaires telles que l'authentification à deux facteurs, et en surveillant les logs d'accès pour détecter toute activité suspecte.

- **Session Fixation**: Dans une attaque de fixation de session, l'attaquant force la victime à utiliser un identifiant de session compromis pour se connecter à une application, permettant ainsi à l'attaquant de prendre le contrôle de la session de l'utilisateur. Cette vulnérabilité est plus difficile à exploiter car elle nécessite une interaction directe avec la victime pour la forcer à utiliser un identifiant de session compromis. Pour minimiser ce risque, il est recommandé de mettre en place des mécanismes de renouvellement des identifiants de session à chaque connexion ou à chaque changement de privilèges.

- **Session Prédiction**: Les identifiants de session peuvent être prédits si les algorithmes de génération d'identifiants de sessions ne sont pas suffisamment aléatoires ou si les identifiants de sessions ne sont pas assez longs. Cette vulnérabilité est plus difficile à exploiter car elle nécessite une connaissance approfondie de l'algorithme de génération utilisé. Pour minimiser ce risque, il est recommandé d'utiliser des identifiants de sessions aléatoires et suffisamment longs.

- **Session Hijacking**: Dans une attaque de détournement de session, l'attaquant vole l'identifiant de session d'un utilisateur légitime pour accéder à ses données personnelles ou effectuer des actions malveillantes. Cette vulnérabilité est la plus difficile à exploiter car elle nécessite souvent une

combinaison de techniques, telles que l'ingénierie sociale, l'exploitation de vulnérabilités du navigateur ou des failles de sécurité sur le serveur.

2. JWT

- **Brute-Force des clés secrètes**: Les JWT sont souvent signés ou chiffrés à l'aide d'une clé secrète. Si cette clé est faible ou facilement devinable, elle peut être exposée à une attaque de “Brute-force”. Les attaquants peuvent alors utiliser des outils automatisés pour tester des millions de combinaisons de clés secrètes jusqu'à trouver celle qui convient. Si l'attaquant parvient à déchiffrer le JWT, il peut accéder aux informations sensibles contenues dans le JWT, comme l'identité de l'utilisateur ou des données d'autorisation.

- **Injection de données dans le JWT:** Les JWT contiennent souvent des données telles que des revendications d'authentification et d'autorisation. Si ces données peuvent être injectées par un attaquant, il peut accéder à des ressources et des privilèges qu'il ne devrait pas avoir. Les attaquants peuvent essayer d'injecter des données malveillantes dans le JWT en modifiant les revendications de l'utilisateur ou en falsifiant la signature du JWT. Pour minimiser ce risque, il est recommandé d'utiliser des bibliothèques de JWT fiables et de s'assurer que toutes les entrées de données utilisateur soient validées et échappées correctement.

- **Manipulation de la signature JWT**: Les JWT utilisent une signature pour garantir l'intégrité et l'authenticité des données, si un attaquant peut manipuler la signature, il peut modifier le contenu du JWT sans être détecté. Les attaquants peuvent utiliser une variété de techniques pour manipuler la signature, y compris la substitution de clés, la modification de la charge utile du JWT, ou encore l'utilisation de certificats douteux. Pour minimiser ce risque, il est recommandé d'utiliser des algorithmes de signature solides, comme HMAC ou RSA.

- **Exposition de données sensibles**: Les JWT peuvent contenir des données sensibles telles que des informations d'identification utilisateur, si ces données sont exposées à des tiers non autorisés, elles peuvent être utilisées pour des attaques de phishing et d'autres types d'attaques malveillantes. Les attaquants peuvent obtenir des JWT en interceptant des requêtes réseau, en

accédant aux fichiers de stockage ou en exploitant des vulnérabilités dans des bibliothèques JWT. Pour minimiser ce risque, il est recommandé de limiter les informations sensibles contenues dans le JWT, de le chiffrer et de mettre en place des mesures de sécurité supplémentaires telles que l'authentification à deux facteurs.

- **Session Fixation**: Les attaquants peuvent utiliser des JWT pour mener des attaques de fixation de session. Dans une attaque de fixation de session, l'attaquant force la victime à utiliser un JWT compromis pour se connecter à une application, permettant ainsi à l'attaquant de prendre le contrôle de la session de l'utilisateur. Pour minimiser ce risque, il est recommandé de renouveler régulièrement les sessions des utilisateurs.

VI. Conclusion

Il existe de multiples manières d'authentifier un utilisateur afin de sécuriser l'accès à certaines ressources sur les applications en ligne.

Les principales méthodes utilisées sont les ID de session et les JWT qui ont chacune leurs avantages et inconvénients et dont l'utilisation dépend majoritairement du contexte de l'application.
Bien que les méthodes d'authentification actuelles soient très sécurisées, il existe cependant des vulnérabilités.

On peut se demander si, dans le futur, des protocoles autres que HTTP seront utilisés pour la navigation web, protocoles qui pourraient inclure une gestion d'état ou encore un système génération d'identifiant unique pour tout internet, qui seraient inviolables et donc entièrement sécurisés.

Sources :

- Session vs Token Authentication in 100 Seconds
- What Is JWT and Why Should You Use JWT
- Utilité des JWT : https://auth0.com/docs/secure/tokens/json-web-tokens
- JWT VS Session ID : https://www.loginradius.com/blog/engineering/guest-post/jwt-vs-sessions/
- Est-il possible de cracker un JWT ? : https://security.stackexchange.com/questions/262106/crack-jwt-hs256-with-hashcat
- Vulnérabilités des JWT : JWT Vulnerabilities (Json Web Tokens) - HackTricks
- Vulnérabilités des session ID : https://fr.wikipedia.org/wiki/Fixation_de_session