Aide-mémoire d'authentification

Introduction

L'authentification (AuthN) est le processus de vérification qu'une personne, une entité ou un site Web est bien celui ou celle qu'il prétend être, en déterminant la validité d'un ou plusieurs authentificateurs (comme les mots de passe, les empreintes digitales ou les jetons de sécurité) utilisés pour étayer cette affirmation.

L'identité numérique est la représentation unique d'une personne effectuant une transaction en ligne. Elle est toujours unique dans le contexte d'un service numérique, mais ne permet pas nécessairement de la relier à une personne physique.

La vérification d'identité permet de s'assurer qu'une personne est bien celle qu'elle prétend être. Ce concept, lié à la connaissance du client (KYC), vise à associer une identité numérique à une personne physique.

La gestion de session est un processus par lequel un serveur conserve l'état d'une entité interagissant avec lui. Ceci est nécessaire pour que le serveur puisse se souvenir comment réagir aux requêtes suivantes au cours d'une transaction. Les sessions sont gérées sur le serveur par un identifiant de session qui peut être échangé entre le client et le serveur lors de la transmission et de la réception de requêtes. Les sessions doivent être uniques pour chaque utilisateur et très difficiles à prédire informatiquement. Aide -mémoire sur la gestion de session contient des conseils supplémentaires sur les meilleures pratiques dans ce domaine.

Directives générales d'authentification

Identifiants utilisateur

La fonction principale d'un identifiant utilisateur est d'identifier de manière unique un utilisateur au sein d'un système. Idéalement, les identifiants utilisateur devraient être générés aléatoirement afin d'éviter la création d'identifiants prévisibles ou séquentiels, ce qui pourrait présenter un risque pour la sécurité, notamment dans les systèmes où les identifiants utilisateur pourraient être exposés ou déduits de sources externes.

Noms d'utilisateur

Les noms d'utilisateur sont des identifiants faciles à mémoriser, choisis par l'utilisateur et utilisés pour s'identifier lors de la connexion à un système ou un service. Les termes « identifiant utilisateur » et « nom d'utilisateur » peuvent être utilisés indifféremment si le nom d'utilisateur choisi sert également d'identifiant unique au sein du système.

Les utilisateurs devraient pouvoir utiliser leur adresse courriel comme nom d'utilisateur, à condition que celle-ci soit vérifiée lors de l'inscription. Ils devraient également avoir la possibilité de choisir un autre nom d'utilisateur. Pour plus d'informations sur la validation des adresses courriel, veuillez consulter la discussion par courriel relative à la validation des entrées .

Solution d'authentification et comptes sensibles

- N'autorisez PAS la connexion avec des comptes sensibles (c'est-à-dire des comptes pouvant être utilisés en interne au sein de la solution, comme pour accéder à un serveur, un middleware ou une base de données) à une interface utilisateur frontale.
- N'utilisez PAS la même solution d'authentification (par exemple, IDP/AD) que celle utilisée en interne pour l'accès non sécurisé (par exemple, accès public/DMZ).

Mettre en place des contrôles de force de mot de passe appropriés

Un aspect essentiel de l'authentification par mot de passe est la robustesse de ce dernier. Une politique de mots de passe « forts » rend difficile, voire impossible, de deviner le mot de passe, que ce soit manuellement ou automatiquement. Les caractéristiques suivantes définissent un mot de passe fort :

- Longueur du mot de passe
  - L'application devrait imposer une longueur minimale pour les mots de passe.
    - Si l'authentification multifacteur est activée, les mots de passe de moins de 8 caractères sont considérés comme faibles (NIST SP800-63B).
    - Si l'authentification multifacteur n'est pas activée, les mots de passe de moins de 15 caractères sont considérés comme faibles (NIST SP800-63B).
  - La longueur maximale du mot de passe doit être d'au moins 64 caractères pour permettre les phrases de passe (NIST SP800-63B). Notez que certaines implémentations d'algorithmes de hachage peuvent provoquer un déni de service lié aux mots de passe longs.
- Ne tronquez pas silencieusement vos mots de passe. Aide-mémoire sur le stockage des mots de passe fournit des indications supplémentaires sur la manière de gérer les mots de passe dont la longueur dépasse la longueur maximale.
- Autoriser l'utilisation de tous les caractères, y compris Unicode et les espaces. Aucune règle de composition de mot de passe ne doit limiter les types de caractères autorisés. L'utilisation de majuscules, de minuscules, de chiffres ou de caractères spéciaux ne doit pas être requise.
- Assurez-vous de la rotation des identifiants en cas de fuite de mot de passe, lors de l'identification d'une compromission ou en cas de changement de technologie d'authentification. Évitez d'exiger des changements de mot de passe périodiques ; encouragez plutôt les utilisateurs à choisir des mots de passe robustes et à activer l'authentification multifacteurs (MFA). Selon les directives du NIST, les vérificateurs ne devraient pas imposer de changements de mot de passe arbitraires (par exemple, périodiquement).
- Intégrez un indicateur de force du mot de passe pour aider les utilisateurs à créer un mot de passe plus complexe.
  - Bibliothèque zxcvbn-ts peut être utilisé à cette fin.

    - D'autres implémentations linguistiques de zxcvbn sont listées ici ; Toutefois, vérifiez l'âge et la maturité de chaque exemple avant utilisation.

- Bloquer les mots de passe courants et ceux déjà compromis
    - Mots de passe compromis Ce service permet de vérifier les mots de passe par rapport aux mots de passe déjà compromis. Plus d'informations sur l'API sont disponibles ici.
    - Vous pouvez également télécharger Pwned Passwords. base de données utilisant ce mécanisme l'héberger vous-même.
    - D'autres listes de mots de passe populaires sont disponibles, mais leur mise à jour n'est pas garantie :
        - Listes de mots de passe diverses hébergé par SecLists de Daniel Miessler.
        - Copie statique des 100 000 mots de passe les plus courants de « Have I Been Pwned », hébergée par NCSC dans le texte et JSON format.

Pour plus d'informations, consultez

- Exigences de sécurité des mots de passe ASVS v5.0
- L'évolution des mots de passe : Guide d'authentification pour l'ère moderne

Mettre en œuvre un mécanisme sécurisé de récupération de mot de passe

Il est courant qu'une application dispose d'un mécanisme permettant à un utilisateur d'accéder à son compte en cas d'oubli de son mot de passe. Veuillez consulter la section « Mot de passe oublié ».
Aide-mémoire pour plus de détails sur cette fonctionnalité.

Stockez vos mots de passe en toute sécurité.

Il est essentiel pour une application de stocker un mot de passe en utilisant la technique cryptographique appropriée.
Veuillez consulter le guide de stockage des mots de passe. pour plus de détails sur cette fonctionnalité.

Comparer les hachages de mots de passe à l'aide de fonctions sécurisées

Dans la mesure du possible, le mot de passe fourni par l'utilisateur doit être comparé au hachage du mot de passe stocké à l'aide d'une fonction de comparaison de mots de passe sécurisée fournie par le langage ou le framework, telle que password_verify(). fonction en PHP. Lorsque cela n'est pas possible, assurez-vous que la fonction de comparaison :

- Comporte une longueur d'entrée maximale, afin de se protéger contre les attaques par déni de service avec des entrées très longues.
- Définit explicitement le type des deux variables, afin de se protéger contre les attaques par confusion de type telles que les Magic Hashes en PHP.

- Retourne en temps constant, afin de se protéger contre les attaques temporelles.

Fonction de changement de mot de passe

Lors du développement d'une fonctionnalité de changement de mot de passe, assurez-vous de prendre en compte :

- L'utilisateur est authentifié avec une session active.
- Vérification du mot de passe actuel. Ceci afin de garantir que c'est bien l'utilisateur légitime qui modifie le mot de passe. Prenons l'exemple d'un utilisateur qui se connecte à un ordinateur public et oublie de se déconnecter. Une autre personne pourrait alors utiliser cette session. Si nous ne vérifions pas le mot de passe actuel, cette personne pourrait le modifier.

Ne transmettez les mots de passe que via TLS ou un autre protocole de transport robuste.

Voir : Aide-mémoire sur la sécurité de la couche transport

La page de connexion et toutes les pages d'authentification suivantes doivent être accessibles exclusivement via TLS ou un autre protocole de sécurité robuste. L'absence de protocole TLS ou d'un autre protocole de sécurité robuste pour la page de connexion permet à un attaquant de modifier le formulaire de connexion, ce qui peut entraîner la publication des identifiants de l'utilisateur vers une adresse arbitraire. L'absence de protocole TLS ou d'un autre protocole de sécurité robuste pour les pages d'authentification après la connexion permet à un attaquant de visualiser l'identifiant de session non chiffré et de compromettre la session authentifiée de l'utilisateur.

Exiger une réauthentification pour les fonctionnalités sensibles

Afin d'atténuer les risques de CSRF et de détournement de session, il est important d'exiger les identifiants actuels d'un compte avant de modifier des informations sensibles telles que le mot de passe ou l'adresse électronique de l'utilisateur, ou avant d'effectuer des transactions sensibles, comme la livraison d'un achat à une nouvelle adresse. Sans cette mesure, un attaquant pourrait exécuter des transactions sensibles via une attaque CSRF ou XSS sans connaître les identifiants actuels de l'utilisateur. De plus, un attaquant pourrait obtenir un accès physique temporaire au navigateur de l'utilisateur.

voler leur identifiant de session pour prendre le contrôle de la session de l'utilisateur.

Réauthentification après des événements à risque

Aperçu : La réauthentification est essentielle lorsqu'un compte a fait l'objet d'une activité à risque, comme une récupération de compte, une réinitialisation de mot de passe ou des comportements suspects. Cette section explique quand et comment déclencher une réauthentification afin de protéger les utilisateurs et d'empêcher tout accès non autorisé.
Pour plus de détails, consultez la section « Exiger une réauthentification pour les fonctionnalités sensibles » .

Quand déclencher la réauthentification

- Activité suspecte sur un compte : en cas de schémas de connexion inhabituels, de changements d'adresse IP ou d'inscriptions d'appareils.
- Récupération de compte après la réinitialisation du mot de passe ou la modification des informations sensibles du compte
- Actions critiques Pour les actions à haut risque comme la modification des informations de paiement ou l'ajout de nouveaux appareils de confiance

Mécanismes de réauthentification

- Authentification adaptative : utilisez des modèles d'authentification basés sur les risques qui s'adaptent au comportement et au contexte de l'utilisateur.
- L'authentification multifacteurs (MFA) exige une couche de vérification supplémentaire pour les actions ou événements sensibles.
- La vérification par défi invite les utilisateurs à confirmer leur identité au moyen d'une question de sécurité ou d'une méthode secondaire.

Recommandations de mise en œuvre

- Réduisez les frictions pour l'utilisateur. Veillez à ce que la réauthentification ne perturbe pas inutilement l'expérience utilisateur.
- Décisions contextuelles : Les décisions de réauthentification sont prises en fonction du contexte (par exemple, la géolocalisation, le type d'appareil, les habitudes de connexion antérieures).
- Gestion sécurisée des sessions : invalidez les sessions après la réauthentification et faites tourner les jetons (voir le guide de gestion des sessions OWASP) .

Références

- Aide-mémoire de gestion de session OWASP
- OWASP ASVS – 2.2.2 : Exigences de réauthentification
- NIST 800-63B : Lignes directrices relatives à l'identité numérique – Niveaux d'assurance d'authentification

Envisagez une authentification forte des transactions

Certaines applications doivent utiliser une authentification à deux facteurs pour vérifier si un utilisateur est autorisé à effectuer des opérations sensibles. Pour plus d'informations, consultez le guide pratique sur l'autorisation des transactions.

Authentification du client TLS

L'authentification TLS du client, également appelée authentification TLS bidirectionnelle, consiste en l'envoi, par le navigateur et le serveur, de leurs certificats TLS respectifs lors de la négociation TLS. De même qu'il est possible de vérifier l'authenticité d'un serveur à l'aide de son certificat et en interrogeant une autorité de certification (AC) reconnue, le serveur peut authentifier l'utilisateur en recevant un certificat du client et en le validant auprès d'une AC tierce ou de sa propre AC. Pour ce faire, le serveur doit fournir à l'utilisateur un certificat généré spécifiquement pour lui.

On attribue des valeurs au sujet afin de déterminer quel utilisateur le certificat doit valider. L'utilisateur installe le certificat sur un navigateur et l'utilise ensuite pour… site web.

Cette approche est appropriée lorsque :

- Il est acceptable (voire préférable) que l'utilisateur n'ait accès au site web que depuis un seul ordinateur/navigateur.
- L'utilisateur n'est pas facilement effrayé par le processus d'installation des certificats TLS sur son navigateur, sinon quelqu'un, probablement du service d'assistance informatique, s'en chargera pour lui.
- Le site web requiert une étape de sécurité supplémentaire.
- C'est également une bonne solution lorsque le site web est destiné à l'intranet d'une entreprise ou d'une organisation.

Il est généralement déconseillé d'utiliser cette méthode pour les sites web largement accessibles au public et fréquentés par un utilisateur lambda. Par exemple, son implémentation sur un site comme Facebook serait inappropriée. Bien que cette technique puisse éviter à l'utilisateur de saisir un mot de passe (le protégeant ainsi contre le vol de données par un enregistreur de frappe classique), il est toujours recommandé d'utiliser à la fois un mot de passe et l'authentification TLS côté client.

De plus, si le client se trouve derrière un proxy d'entreprise qui effectue le déchiffrement SSL/TLS, cela interrompra l'authentification du certificat à moins que le site ne soit autorisé sur le proxy.

Pour plus d'informations, consultez : Négociation TLS authentifiée par le client

Messages d'authentification et d'erreur

Des messages d'erreur mal implémentés dans le cadre de l'authentification peuvent être utilisés pour l'énumération des identifiants et mots de passe. Une application doit répondre (en HTTP et en HTML) de manière générique.

Réponses d'authentification

Quel que soit le mécanisme d'authentification utilisé (connexion, réinitialisation ou récupération du mot de passe), une application doit renvoyer un message d'erreur générique, que :

- L'identifiant ou le mot de passe de l'utilisateur est incorrect.
- Ce compte n'existe pas.
- Le compte est verrouillé ou désactivé.

La fonctionnalité d'inscription de compte doit également être prise en compte, et la même approche, basée sur un message d'erreur générique, peut être appliquée dans le cas où l'utilisateur existe.

L'objectif est d'empêcher la création d'un facteur de divergence. permettant à un attaquant de lancer une action d'énumération des utilisateurs contre l'application.

Il est intéressant de noter que la logique métier elle-même peut introduire un facteur de disparité lié au temps de traitement. En effet, selon l'implémentation, ce temps peut varier considérablement selon le cas (succès ou échec), permettant ainsi à un attaquant de mener une attaque temporelle. (delta de quelques secondes par exemple).

Exemple d'utilisation de pseudo-code pour une fonctionnalité de connexion :

- Première implémentation utilisant l'approche de « sortie rapide »

```
SI L'UTILISATEUR_EXISTE(nom_utilisateur) ALORS
        mot_de_passe_hachage=HACHE(mot_de_passe)
        IS_VALID=LOOKUP_CREDENTIALS_IN_STORE(nom_utilisateur, mot_de_passe_hash)
        SI IS_VALID N'EST PAS VALIDE
                ALORS RETOURNER Error("Nom d'utilisateur ou mot de passe invalide !")
        FIN SI
SINON
      RETOURNER Erreur("Nom d'utilisateur ou mot de passe invalide !")
FINSI
```

Il est clair que si l'utilisateur n'existe pas, l'application générera directement une erreur.
Autrement, si l'utilisateur existe mais que le mot de passe est incorrect, il est évident que le traitement sera plus long avant que l'application ne génère une erreur. Par conséquent, le temps de réponse sera différent pour une même erreur, ce qui permettra à un attaquant de distinguer un nom d'utilisateur incorrect d'un mot de passe incorrect.

- Deuxième implémentation sans recourir à l'approche de « sortie rapide » :

```
mot_de_passe_hachage=HACHE(mot_de_passe)
IS_VALID=LOOKUP_CREDENTIALS_IN_STORE(nom_utilisateur, mot_de_passe_hash)
SI IS_VALID N'EST PAS VALIDE ALORS
      RETOUR Erreur("Nom d'utilisateur ou mot de passe invalide !")
FINSI
```

Ce code suivra le même processus quels que soient l'utilisateur ou le mot de passe, permettant à l'application de répondre dans un temps approximativement identique.

Le problème lié à l'affichage d'un message d'erreur générique relève de l'expérience utilisateur (UX). Un utilisateur légitime pourrait se sentir perdu face à un tel message, ce qui rendrait l'utilisation de l'application difficile et pourrait, après plusieurs tentatives infructueuses, l'abandonner en raison de sa complexité. La décision d'afficher ou non un message d'erreur générique peut être prise en fonction de la criticité de l'application et des données qu'elle traite. Par exemple, pour les applications critiques, l'équipe peut décider qu'en cas de panne, l'utilisateur sera systématiquement redirigé vers la page d'assistance et qu'un message d'erreur générique lui sera affiché.

Concernant l'énumération des utilisateurs elle-même, la protection contre les attaques par force brute est également efficace car elle empêche un attaquant d'appliquer l'énumération à grande échelle. Utilisation de CAPTCHA peut s'appliquer à une fonctionnalité pour laquelle un message d'erreur générique ne peut être renvoyé car l'expérience utilisateur doit être préservée.

EXEMPLES DE RÉPONSES INCORRECTES ET CORRECTES

Se connecter

Exemples de réponses incorrectes :

- "Connexion pour l'utilisateur foo : mot de passe invalide."
- "Échec de la connexion, identifiant utilisateur invalide."
- "Échec de la connexion ; compte désactivé."
- "Échec de la connexion ; cet utilisateur n'est pas actif."

Exemple de réponse correcte :

- "Échec de la connexion ; identifiant ou mot de passe invalide."

Récupération de mot de passe

Exemples de réponses incorrectes :

- «Nous venons de vous envoyer un lien de réinitialisation de mot de passe.»
- «Cette adresse e-mail n'existe pas dans notre base de données.»

Exemple de réponse correcte :

- « Si cette adresse électronique figure dans notre base de données, nous vous enverrons un courriel pour réinitialiser votre mot de passe. »

Création de compte

Exemples de réponses incorrectes :

- "Cet identifiant utilisateur est déjà utilisé."
- "Bienvenue ! Votre inscription a bien été prise en compte."

Exemple de réponse correcte :

- « Un lien pour activer votre compte a été envoyé par courriel à l'adresse fournie. »

CODES D'ERREUR ET URL

L'application peut renvoyer un code d'erreur HTTP différent En fonction de la réponse à la tentative d'authentification, le système peut renvoyer un code 200 en cas de succès et un code 403 en cas d'échec. Même si une page d'erreur générique s'affiche, le code de réponse HTTP peut varier.
qui peuvent divulguer des informations sur la validité du compte.

La divulgation des erreurs peut également servir de facteur de divergence ; consultez le guide de gestion des erreurs. concernant la gestion globale des différentes erreurs dans une application.

Se protéger contre les attaques automatisées

Il existe différents types d'attaques automatisées que les attaquants peuvent utiliser pour tenter de compromettre les comptes utilisateurs. Les plus courantes sont énumérées ci-dessous :

Type d'attaque	Description
Force brute	Tester plusieurs mots de passe provenant d'un dictionnaire ou d'une autre source sur un seul compte.
Attestation Rembourrage	Tester les paires nom d'utilisateur/mot de passe obtenues suite à la violation d'un autre compte. site.
Mot de passe Pulvérisation	Tester un seul mot de passe faible face à un grand nombre de mots de passe différents comptes.

Différents mécanismes de protection peuvent être mis en œuvre pour se prémunir contre ces attaques. Dans de nombreux cas, ces défenses n'offrent pas une protection totale, mais leur combinaison, selon une approche de défense en profondeur, permet d'atteindre un niveau de protection satisfaisant.

Les sections suivantes porteront principalement sur la prévention des attaques par force brute, bien que ces mesures de contrôle puissent également être efficaces contre d'autres types d'attaques. Pour plus d'informations sur la protection contre le bourrage d'identifiants et l'attaque par pulvérisation de mots de passe, consultez le guide pratique sur le bourrage d'identifiants.

Authentification multifacteurs

L'authentification multifacteurs (MFA) est de loin la meilleure défense contre la majorité des attaques liées aux mots de passe, y compris les attaques par force brute, une analyse de Microsoft suggérant qu'elle aurait empêché 99,9 % des compromissions de comptes. Par conséquent, son application devrait être mise en œuvre partout où cela est possible ; toutefois, selon le public cible de l'application, il peut s'avérer difficile, voire impossible, d'imposer l'utilisation de l'authentification multifacteur.

Aide-mémoire sur l' authentification multifactorielle contient des conseils supplémentaires sur la mise en œuvre de l'authentification multifacteur.

Limitation du débit de connexion

La limitation des tentatives de connexion est un protocole utilisé pour empêcher un attaquant de tenter trop souvent de deviner un mot de passe par des moyens interactifs normaux ; il comprend les contrôles suivants :

- Nombre maximal de tentatives.

VERROUILLAGE DU COMPTE

La protection la plus courante contre ces attaques consiste à mettre en œuvre le verrouillage des comptes, qui empêche toute nouvelle tentative de connexion pendant une période donnée après un certain nombre de tentatives infructueuses.

Le compteur d'échecs de connexion devrait être associé au compte lui-même, et non à la source.
adresse IP, afin d'empêcher un attaquant de tenter de se connecter à partir d'un grand nombre d'adresses IP.
différentes adresses IP. Plusieurs facteurs doivent être pris en compte lors de la sélection d'une adresse IP.
mettre en œuvre une politique de verrouillage des comptes afin de trouver un équilibre entre sécurité et facilité
d'utilisation :

- Le nombre de tentatives infructueuses avant le verrouillage du compte (seuil de verrouillage).
- La période dans laquelle ces tentatives doivent avoir lieu (fenêtre d'observation).
- Durée du blocage du compte (durée du blocage).

Plutôt que d'implémenter une durée de verrouillage fixe (par exemple, dix minutes), certaines applications utilisent un verrouillage exponentiel, où la durée de verrouillage commence par une période très courte (par exemple, une seconde), mais double après chaque tentative de connexion infructueuse.

- Durée du délai après chaque blocage de compte (max 2-3, après ce blocage de compte permanent).

Lors de la conception d'un système de verrouillage de compte, il est essentiel de veiller à ce qu'il ne soit pas utilisé pour provoquer un déni de service en bloquant les comptes d'autres utilisateurs. Une solution possible consiste à autoriser l'utilisation de la fonction « mot de passe oublié » pour se connecter, même si…
Le compte est bloqué.

CAPTCHA

L'utilisation d'un CAPTCHA efficace peut contribuer à prévenir les tentatives de connexion automatisées aux comptes. Cependant, de nombreuses implémentations de CAPTCHA présentent des failles qui permettent de les contourner par des techniques automatisées ou de sous-traiter cette tâche à des services spécialisés. De ce fait, l'utilisation d'un CAPTCHA doit être envisagée comme un contrôle de sécurité en profondeur visant à rendre les attaques par force brute plus longues et plus coûteuses, plutôt que comme une mesure préventive.

Il serait peut-être plus convivial de n'exiger la résolution d'un CAPTCHA qu'après un petit nombre de tentatives de connexion infructueuses, plutôt que de l'exiger dès la toute première connexion.

Questions de sécurité et mots-clés à retenir

L'ajout d'une question de sécurité ou d'un mot-clé peut également contribuer à la protection contre les attaques automatisées, notamment lorsque l'utilisateur est invité à saisir un certain nombre de caractères choisis aléatoirement parmi ceux du mot. Il convient de noter que cela ne constitue pas une authentification multifacteurs.

L'authentification, car les deux facteurs sont identiques (un élément que vous connaissez), est essentielle. De plus, les questions de sécurité sont souvent faibles et leurs réponses prévisibles ; il est donc crucial de les choisir avec soin. Consultez le guide pratique « Choisir et utiliser des questions de sécurité ». contient des indications supplémentaires à ce sujet.

Journalisation et surveillance

Activer la journalisation et la surveillance des fonctions d'authentification pour détecter les attaques/défaillances en temps réel.

- S'assurer que toutes les défaillances sont consignées et analysées.
- S'assurer que tous les échecs de connexion sont consignés et analysés.
- S'assurer que tous les blocages de compte sont consignés et examinés

Utilisation de protocoles d'authentification ne nécessitant aucun mot de passe

Bien que l'authentification par nom d'utilisateur, mot de passe et authentification multifacteurs soit généralement considérée comme sécurisée, elle n'est pas toujours optimale, voire même sûre. C'est le cas, par exemple, des applications tierces qui souhaitent se connecter à l'application web depuis un appareil mobile, un autre site web, un ordinateur ou dans d'autres situations.

Dans ce cas, il est fortement déconseillé d'autoriser une application tierce à stocker les identifiants et mots de passe, car cela expose davantage les données à des attaques, vous laissant ainsi sans contrôle. Pour ce cas de figure et d'autres, plusieurs protocoles d'authentification permettent de protéger les données de vos utilisateurs contre les attaquants.

OAuth 2.0 et 2.1

OAuth est un cadre d'autorisation pour l'accès délégué aux API. Voir aussi : Astuce OAuth 2.0 Feuille.

> Remarque : OAuth 2.1 est un projet de norme du groupe de travail de l'IETF qui consolide OAuth 2.0 et les meilleures pratiques largement adoptées et vise à remplacer les RFC 6749/6750 ; les conseils de ce guide s'appliquent à la fois à OAuth 2.0 et à OAuth 2.1. Références : draft-ietf-oauth-v2-1-13 oauth.net/2.1

OpenID Connect (OIDC)

OpenID Connect 1.0 (OIDC) est une couche d'identité qui s'appuie sur OAuth. Elle définit comment un client (partie de confiance) vérifie l'identité de l'utilisateur final à l'aide d'un jeton d'identité (un JWT signé) et comment obtenir les revendications de l'utilisateur de manière interopérable. Utilisez OIDC pour l'authentification/SSO ; utilisez OAuth pour l'autorisation.
aux API.

Guide de mise en œuvre de l'OIDC

- Valider les jetons d'identification sur la partie de confiance : émetteur (`iss`), audience (`aud`), signature (par fournisseur JWK), expiration (`exp`).
- Privilégiez les bibliothèques/SDK bien maintenus et les points de terminaison de découverte de fournisseurs/JWKS.

- Utilisez le point de terminaison UserInfo lorsque des revendications supplémentaires au-delà du jeton d'identification sont requises.

> Pour éviter toute confusion : OpenID 2.0 (« OpenID ») était un protocole d'authentification distinct et ancien, désormais obsolète et remplacé par OpenID Connect. Les nouveaux systèmes ne doivent pas implémenter OpenID 2.0.
> Références : Fondation OpenID — bibliothèques OpenID 2.0 obsolètes. Migration d'OpenID 2.0 vers OIDC

SAML

Le langage SAML (Security Assertion Markup Language) est souvent considéré comme un concurrent d'OpenID. La version 2.0 est la plus recommandée car elle offre un large éventail de fonctionnalités et une sécurité renforcée. À l'instar d'OpenID, SAML utilise des fournisseurs d'identité, mais contrairement à OpenID, il est basé sur XML et offre une plus grande flexibilité. SAML repose sur des redirections de navigateur qui transmettent des données XML. De plus, l'authentification SAML n'est pas uniquement initiée par un fournisseur de services ; elle peut également l'être par le fournisseur d'identité. Cela permet à l'utilisateur de naviguer sur différents portails tout en restant authentifié automatiquement, rendant ainsi le processus transparent.

Bien qu'OpenID domine le marché grand public, SAML est souvent privilégié pour les applications d'entreprise, car peu de fournisseurs d'identité OpenID répondent aux exigences de niveau entreprise (leur méthode de validation de l'identité de l'utilisateur ne respectant pas les normes élevées requises pour l'identité en entreprise). L'utilisation de SAML est plus fréquente au sein des sites intranet, parfois même avec un serveur intranet comme fournisseur d'identité.

Ces dernières années, des applications comme SAP ERP et SharePoint (SharePoint utilisant Active Directory Federation Services 2.0) ont opté pour l'authentification SAML 2.0 comme méthode souvent privilégiée pour les implémentations d'authentification unique lorsque la fédération d'entreprise est requise pour les services Web et les applications Web.

Voir aussi : Aide-mémoire sur la sécurité SAML

FIDO

L'alliance FIDO (Fast Identity Online) a créé deux protocoles pour faciliter l'authentification en ligne : le protocole UAF (Universal Authentication Framework) et le protocole U2F (Universal Second Factor). UAF privilégie l'authentification sans mot de passe, tandis qu'U2F permet d'ajouter un second facteur d'authentification à un système d'authentification par mot de passe existant. Ces deux protocoles reposent sur un modèle de cryptographie à clé publique avec échange de questions-réponses.

UAF exploite les technologies de sécurité existantes sur les appareils pour l'authentification, notamment les capteurs d'empreintes digitales, les caméras (reconnaissance faciale), les microphones (reconnaissance vocale), les environnements d'exécution de confiance (TEE), les éléments sécurisés (SE), etc. Le protocole est conçu pour intégrer ces fonctionnalités dans un cadre d'authentification commun. UAF est compatible avec les applications natives et les applications web.

U2F renforce l'authentification par mot de passe grâce à un jeton matériel (généralement une clé USB) qui stocke des clés d'authentification cryptographiques et les utilise pour la signature. L'utilisateur peut utiliser ce même jeton comme second facteur d'authentification pour plusieurs applications. U2F est compatible avec les applications web et offre une protection contre le phishing en utilisant l'URL du site web pour retrouver la clé d'authentification stockée.

Gestionnaires de mots de passe

Les gestionnaires de mots de passe sont des programmes, des extensions de navigateur ou des services web qui automatisent la gestion d'un grand nombre d'identifiants différents. La plupart des gestionnaires de mots de passe offrent des fonctionnalités permettant aux utilisateurs de les utiliser facilement sur les sites web, soit : (a) en collant les mots de passe dans le formulaire de connexion, soit (b) en simulant la saisie manuelle par l'utilisateur.

Les applications Web ne doivent pas compliquer inutilement la tâche des gestionnaires de mots de passe et doivent respecter les recommandations suivantes :

- Utilisez des formulaires HTML standard pour la saisie du nom d'utilisateur et du mot de passe avec un type approprié. attributs.
- Évitez les pages de connexion basées sur des plugins (tels que Flash ou Silverlight).
- Mettez en œuvre une longueur maximale raisonnable pour les mots de passe, d'au moins 64 caractères, comme indiqué dans la section « Mettre en œuvre des contrôles appropriés de la force des mots de passe ».
- Autoriser l'utilisation de tout caractère imprimable dans les mots de passe.
- Autoriser les utilisateurs à coller des informations dans les champs nom d'utilisateur, mot de passe et MFA.
- Permettre aux utilisateurs de naviguer entre les champs nom d'utilisateur et mot de passe en appuyant simplement sur la touche Tab .

Modification de l'adresse e-mail enregistrée d'un utilisateur

Les adresses électroniques des utilisateurs changent souvent. La procédure suivante est recommandée pour gérer ces situations au sein d'un système :

Remarque : Le processus est moins rigoureux avec l'authentification multifactorielle. car une preuve d'identité est plus fiable qu'un simple mot de passe.

Procédure recommandée si l'utilisateur dispose d'une authentification multifactorielle Activé

1. Vérifiez la validité du cookie/jeton d'authentification de l'utilisateur. S'il n'est pas valide, affichez un formulaire de connexion. écran.
2. Décrivez la procédure de modification de l'adresse électronique enregistrée de l'utilisateur.

3. Demandez à l'utilisateur de soumettre une nouvelle adresse électronique, en vous assurant qu'elle est conforme au système. règles.
4. Demander l'utilisation de l'authentification multifactorielle pour la vérification d'identité.
5. Enregistrez la nouvelle adresse e-mail proposée comme une modification en attente.
6. Créer et stocker deux nonces à durée de vie limitée pour (a) la notification des administrateurs système, et (b) confirmation de l'utilisateur.
7. Envoyez deux courriels contenant des liens incluant ces nonces :
    - Un courriel de notification uniquement sera envoyé à l'adresse actuelle pour informer l'utilisateur du changement imminent et lui fournir un lien pour signaler toute activité inattendue.
    - Un courriel de confirmation sera envoyé à la nouvelle adresse proposée, invitant l'utilisateur à confirmer le changement et fournissant un lien en cas d'imprévu.
8. Traitez les réponses provenant des liens en conséquence.

Procédure recommandée si l'utilisateur ne dispose pas de l'authentification multifactorielle

Authentification activée

1. Vérifiez la validité du cookie/jeton d'authentification de l'utilisateur. S'il n'est pas valide, affichez un formulaire de connexion. écran.
2. Décrivez la procédure de modification de l'adresse électronique enregistrée de l'utilisateur.
3. Demandez à l'utilisateur de soumettre une nouvelle adresse électronique, en vous assurant qu'elle est conforme au système. règles.
4. Demander le mot de passe actuel de l'utilisateur pour vérification d'identité.
5. Enregistrez la nouvelle adresse e-mail proposée comme une modification en attente.
6. Créer et stocker trois nonces temporaires pour la notification des administrateurs système et des utilisateurs confirmation, et une étape supplémentaire pour la sécurité du mot de passe.
7. Envoyez deux courriels contenant des liens vers ces nonces :
    - Un courriel de confirmation sera envoyé à l'adresse actuelle, demandant à l'utilisateur de confirmer la modification et fournissant un lien en cas d'imprévu.
    - Un courriel de confirmation distinct sera envoyé à la nouvelle adresse proposée, demandant à l'utilisateur de confirmer le changement et fournissant un lien en cas d'imprévu.
8. Traitez les réponses provenant des liens en conséquence.

Remarques sur les processus ci-dessus

- Il convient de noter que Google adopte une approche différente pour les comptes protégés uniquement par un mot de passe : l'adresse électronique actuelle reçoit alors un courriel de notification uniquement. Cette méthode comporte des risques et requiert la vigilance de l'utilisateur.
- Une formation régulière à la prévention des attaques d'ingénierie sociale est essentielle. Les administrateurs système et le personnel du support technique doivent être formés à suivre la procédure prescrite, à reconnaître les attaques d'ingénierie sociale et à y répondre. Consultez le document de la CISA intitulé « Éviter les attaques d'ingénierie sociale et de phishing ». pour vous guider.

Authentification adaptative ou basée sur les risques

Une caractéristique des applications plus avancées est la possibilité d'exiger différentes étapes d'authentification en fonction de divers attributs environnementaux et contextuels (y compris, mais sans s'y limiter, la sensibilité des données pour lesquelles l'accès est demandé, l'heure de la journée, l'emplacement de l'utilisateur, l'adresse IP ou l'empreinte digitale de l'appareil).

Par exemple, une application peut exiger une authentification multifacteur (MFA) lors de la première connexion depuis un appareil donné, mais pas lors des connexions suivantes depuis ce même appareil. À l'inverse, une solution d'authentification unique (SSO) peut authentifier l'utilisateur et lui permettre de rester connecté pendant une journée, mais exiger une réauthentification s'il tente d'accéder à sa page de profil.

Une autre option consiste à adopter une approche inverse : une application autorise un accès à faible risque grâce à un simple identifiant de l'appareil (par exemple, une empreinte numérique spécifique du mobile, un cookie persistant et l'empreinte numérique du navigateur à partir de l'adresse IP précédente), puis exige progressivement une authentification plus forte pour les opérations plus sensibles. Par exemple, un utilisateur pourrait être autorisé à consulter son solde bancaire actuel, mais pas son numéro de compte ni aucune autre information personnelle. S'il souhaite consulter ses transactions, l'application lui demande une authentification de base ; pour effectuer des virements, l'authentification multifacteur (MFA) est requise.

Les questions à prendre en compte lors de la mise en œuvre d'un tel mécanisme sont notamment les suivantes :

- Les politiques mises en place sont-elles conformes aux politiques de l'entreprise et notamment aux réglementations en vigueur ?
- Quels attributs de l'utilisateur ou de l'appareil (adresse IP, géolocalisation, empreinte digitale de l'appareil, heure de la journée, données biométriques comportementales, etc.) allons-nous surveiller au début de la session ?
- Quels signaux doivent être actualisés pendant une session active, et à quelle fréquence ? cadence?
- Comment allons-nous garantir la précision de chaque signal et gérer les données manquantes ou peu fiables ?
- Quel modèle de notation (pondération, seuils, apprentissage automatique, basé sur des règles, hybride) convertira les signaux bruts en un niveau de risque ?
- Où le modèle sera-t-il exécuté (périphérique, passerelle API, service central) et quel est notre budget de latence ?

- Quelle action correspond à chaque niveau de risque (autoriser, CAPTCHA, authentification multifacteur renforcée, bloquer, révoquer la session) ?
- Quels messages et codes d'erreur destinés à l'utilisateur accompagneront chaque action ?
- À quelles couches de code ou de plateforme exactes allons-nous invoquer le moteur de risque (contrôleur de connexion, middleware, passerelle API, maillage de services) ?
- Comment propager les décisions de manière cohérente entre les clients web, mobiles et API ?
- Comment modifier, prolonger ou révoquer les jetons/cookies lors d'un contrôle des risques en cours de session escalade ?
- Comment synchroniser l'état sur plusieurs appareils ou onglets de navigateur simultanés ?
- Quels mécanismes de surveillance et d'alerte seront mis en place pour les activités potentiellement suspectes, et notamment comment l'utilisateur sera informé ?