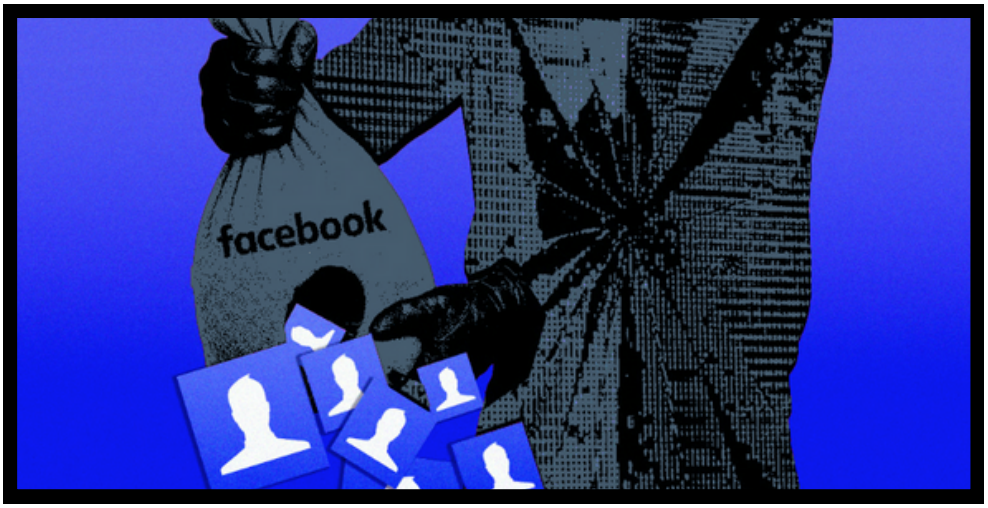
FUITE DE DONNÉES FACEBOOK

Simulation d'Attaque de Scraping et Mesures de Sécurité



PRESENTED BY

Ben Hamou Mehdi

Sommaire

Introduction : Présentation du problème

Récemment, une fuite massive de données a secoué le monde numérique : plus d'1,2 milliard de comptes Facebook auraient été exposés par le biais d'un piratage. Cette base de données contient des informations personnelles sensibles comme les noms, adresses email, numéros de téléphone, dates de naissance, et même les lieux de résidence des utilisateurs. Ce type de fuite représente une menace majeure pour la vie privée de millions de personnes à travers le monde.

Pourquoi est-ce si grave ? Parce que ces données, une fois entre de mauvaises mains, peuvent être utilisées pour des attaques ciblées telles que le phishing, l'usurpation d'identité, ou des escroqueries. Imaginez que quelqu'un utilise vos informations personnelles pour vous envoyer un message frauduleux qui semble venir d'un ami ou d'une entreprise de confiance : c'est exactement ce que permet ce type de fuite.

Cette situation soulève aussi une question importante sur la responsabilité des grandes plateformes comme Facebook (Meta) en matière de protection des données. Comment un acteur majeur du numérique, avec des moyens techniques considérables, peut-il laisser s'échapper autant d'informations personnelles ? Cela démontre qu'il ne suffit pas d'être une grande entreprise pour garantir la sécurité de ses utilisateurs. La protection des données doit être une priorité constante, proactive et transparente.

Cette fuite n'est pas un incident isolé, mais un signal d'alarme pour toute la communauté numérique. Elle rappelle que la sécurité informatique, surtout dans le traitement des données personnelles, doit être prise très au sérieux pour préserver la confiance des utilisateurs et éviter des conséquences souvent irréversibles.

Attackers' post on a data leak forum. Image by Cybernews.

Comment cette fuite a-t-elle eu lieu ?

Exploitation malveillante d'une API de Facebook

La fuite des 1,2 milliard de comptes Facebook n'est pas le fruit d'un piratage traditionnel où un serveur est compromis ou un mot de passe est deviné. Non. Ici, les attaquants ont utilisé une méthode bien plus subtile c'est **l'exploitation d'une API publique** de Facebook.

Mais d'abord, **qu'est-ce qu'une API ?**
Une API (Interface de Programmation d'Application) est une sorte de « pont » qui permet à deux logiciels ou plateformes de communiquer entre eux. Par exemple, quand vous autorisez une application à se connecter via votre compte Facebook (comme pour un jeu ou un site e-commerce), cette application utilise l'API de Facebook pour obtenir certaines informations vous concernant, comme votre nom ou votre adresse email. Mais ce pont, lorsqu'il est mal protégé, peut devenir une porte d'entrée. C'est exactement ce qui s'est passé ici.

Les attaquants ont **abusé de cette API**, probablement en automatisant des requêtes via des scripts pour extraire massivement des informations sur les utilisateurs. Ce n'est pas forcément une faille de sécurité technique comme une vulnérabilité « zero-day », mais plutôt un défaut de conception ou de limitation : **l'API permettait d'accéder à des données sensibles de manière trop large, sans contrôle strict sur la fréquence ou le volume des demandes.**

Le contexte : pourquoi les API sont devenues des cibles privilégiées ?

Aujourd'hui, presque tous les services numériques reposent sur des APIs. Elles sont essentielles pour faire fonctionner les applications modernes. Cependant, elles représentent aussi **une surface d'attaque majeure**. Pourquoi ?
Parce que :

- **Elles sont souvent exposées sur Internet**
- **Elles permettent d'accéder directement à des données**
- **Elles ne sont pas toujours protégées contre les abus (par exemple, pas de vérification forte de l'utilisateur, pas de limites de fréquence ou de volume de requêtes)**
- **Elles sont parfois mal documentées ou mal surveillées**

Les cybercriminels le savent, et ils cherchent à les exploiter pour collecter un maximum de données. Dans le cas de Facebook, l'API ciblée semble avoir été utilisée comme un **outil de scraping** à grande échelle.

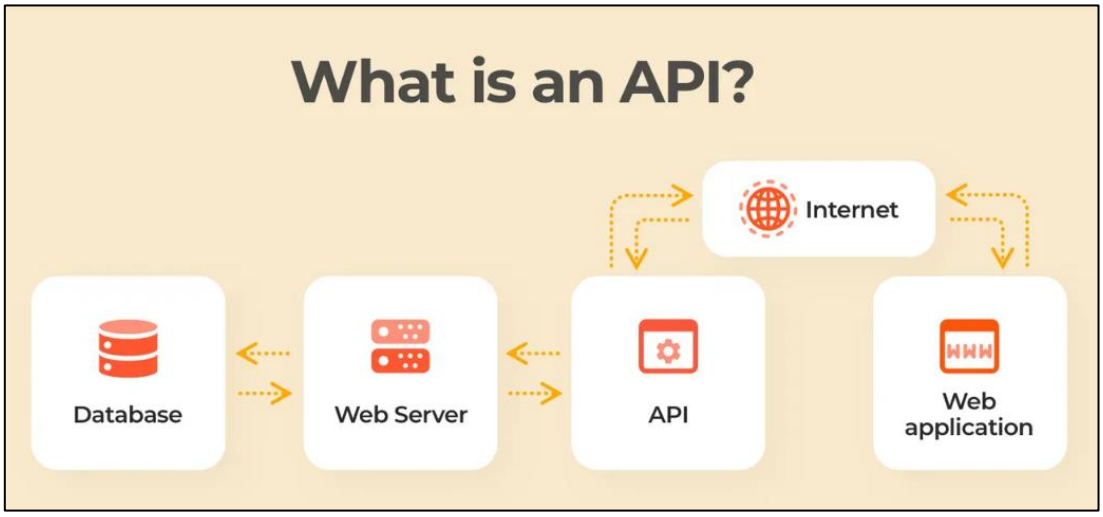
Cette méthode n'est pas nouvelle : des fuites similaires ont touché des plateformes comme LinkedIn, Twitter ou encore Instagram. Mais ce qui rend celle-ci particulièrement inquiétante, c'est **l'ampleur des données récupérées** et **le fait qu'elle provient d'un géant du web censé être un modèle en matière de sécurité.**

3. Qu’est-ce qu’une API ?

Une définition simple et accessible

Une API (ou *Application Programming Interface*, en anglais) est une **interface de programmation** qui permet à deux systèmes informatiques de communiquer entre eux. Pour faire simple, c’est un **intermédiaire logiciel** qui permet à une application d’accéder à des fonctionnalités ou des données d’un autre service — sans avoir besoin de connaître tout son fonctionnement interne.

Imaginez un restaurant : l’API, c’est un peu le serveur. Vous (le client) passez votre commande via le serveur, qui la transmet à la cuisine (le système), et vous rapporte le plat (la réponse). Vous n’avez pas besoin de savoir comment les plats sont préparés — vous interagissez seulement avec le serveur/API.



Le rôle des APIs dans les services numériques modernes

Les APIs sont **au cœur du fonctionnement du web moderne**. Elles sont utilisées partout :

- Lorsqu’un site vous permet de vous connecter avec votre compte Google, Facebook ou Apple
- Quand une application météo récupère les prévisions sur un serveur externe
- Lorsqu’un site e-commerce affiche les produits d’un autre vendeur (via une API externe)
- Ou encore lorsqu’un chatbot comme moi interagit avec une base de données pour répondre à vos questions

Facebook, Google, Twitter, Instagram, OpenAI… : tous ces géants proposent des APIs pour permettre aux développeurs de créer des applications plus intelligentes, connectées, et dynamiques.

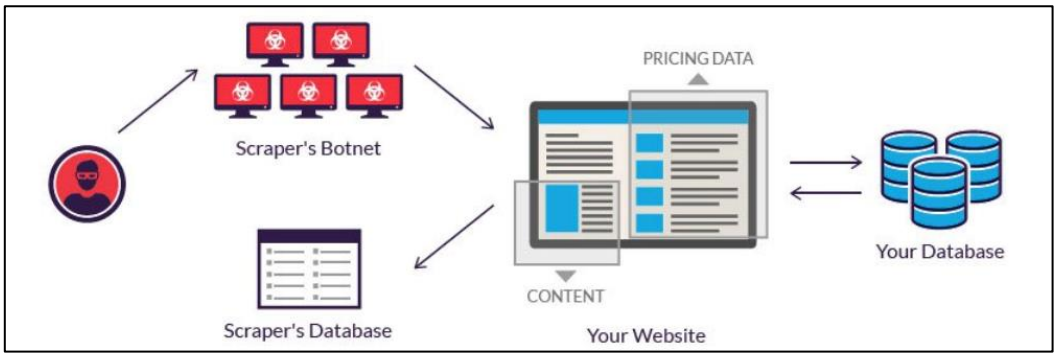
Mais comme toute technologie puissante, les APIs peuvent devenir un **point faible** si elles sont mal protégées. Une API mal conçue ou mal sécurisée peut exposer des données personnelles, comme ce fut le cas avec la fuite de 1,2 milliard de comptes Facebook.

4. Qu’est-ce que le scraping ?

Définition du scraping de données

Le **scraping** est une technique qui consiste à **collecter automatiquement des informations** sur des sites web ou des plateformes en ligne. Concrètement, un logiciel appelé *scraper* parcourt les pages d’un site, récupère des données (comme des noms, des adresses e-mail, des numéros de téléphone, etc.) et les enregistre pour une utilisation ultérieure.

C’est un peu comme un robot qui “lit” et copie tout ce qu’il trouve sur une page web, mais à une vitesse et une échelle que personne ne pourrait faire manuellement.



Utilisation légitime du scraping

Le scraping peut être **utile et légal** dans plusieurs cas, par exemple :

- Récupérer des données publiques pour effectuer des analyses de marché
- Agréger des informations de différentes sources pour un comparateur de prix
- Extraire des données pour la recherche scientifique ou pour améliorer des services

Dans ces cas, le scraping respecte souvent les conditions d’utilisation des sites, les règles de confidentialité, et ne cherche pas à nuire.

Utilisation malveillante du scraping

Mais le scraping peut aussi être utilisé **à des fins malveillantes**, notamment :

- Collecter en masse des données personnelles sans consentement, ce qui viole la vie privée des utilisateurs
- Extraire des informations sensibles ou protégées pour des attaques ciblées (phishing, usurpation d’identité)
- Saturer un service en récupérant trop de données, causant des dysfonctionnements (attaque par déni de service)

Dans le cas de la fuite Facebook, des hackers ont utilisé le scraping pour extraire illégalement un énorme volume de données via une API mal sécurisée. Ces données comprennent des informations privées comme les numéros de téléphone, les adresses e-mail, les noms, etc. Cette collecte massive pose de graves risques pour la sécurité et la confidentialité des utilisateurs.

5. Différence entre scraping et fuite de données

Scraping : collecte via exploitation d'interfaces accessibles

Le **scraping** désigne une collecte **active et automatisée** de données en exploitant des interfaces accessibles, souvent des API ou des pages web publiques. Autrement dit, le scraper « interroge » volontairement un service ou un site pour récupérer des informations, souvent à grande échelle.

Cela peut être légal ou non, selon les règles d'accès, le respect de la vie privée, et la finalité. Le scraping peut par exemple exploiter des failles dans la conception d'une API qui laisse trop d'informations disponibles sans contrôle strict.

Fuite de données : divulgation non autorisée

Une **fuite de données** (ou data breach) se produit lorsqu'un tiers obtient **sans autorisation** des informations sensibles ou privées. Cette divulgation peut résulter :

- D'une **faille de sécurité** dans un système informatique (exemple : une base de données mal protégée)
- D'une attaque ou d'un piratage direct (exemple : un hacker accède à un serveur sécurisé)
- D'une erreur humaine (exemple : un employé partage par erreur des fichiers confidentiels)

La fuite signifie que les données ont été exposées au public ou à des acteurs malveillants sans consentement, souvent en grande quantité.

Dans le cas de la récente fuite Facebook, les attaquants ont utilisé du scraping via une API mal sécurisée pour extraire un volume colossal d'informations, ce qui est à la fois une exploitation d'une interface accessible (scraping) et une fuite massive de données sensibles.

6. Comment empêcher ce type d'attaque ?

La protection des API est essentielle pour éviter que des acteurs malveillants n'exploitent des failles et ne récupèrent massivement des données personnelles. Voici les principales mesures de sécurité à mettre en place :

Limitation des requêtes (Rate Limiting)

Il s'agit de restreindre le nombre de requêtes qu'un utilisateur ou un système peut envoyer à une API sur une période donnée. Cela empêche les attaques automatisées de type scraping massif, car les robots ne pourront pas envoyer un volume trop important de demandes en peu de temps.

Authentification renforcée

Il faut s'assurer que seules des entités autorisées peuvent accéder aux données via l'API. Cela passe par des mécanismes solides d'authentification (tokens sécurisés, OAuth, clés API) et des contrôles d'accès stricts. Ainsi, les données sensibles ne seront pas exposées à tout utilisateur ou programme.

Surveillance et détection des anomalies

Une surveillance continue du trafic API permet de détecter rapidement des comportements suspects : requêtes répétitives, volumes anormaux, adresses IP inhabituelles… Ces alertes peuvent déclencher des blocages automatiques ou une enquête approfondie pour prévenir un vol de données.

Importance de la prévention proactive et de la transparence

Les entreprises doivent adopter une démarche proactive de sécurité, en testant régulièrement leurs API pour identifier les failles avant que des hackers ne les exploitent.

Explication du code et des étapes du scraping avec protection

Structure générale

Le code est une page web simple avec une interface utilisateur construite en HTML/CSS (via TailwindCSS) et un script JavaScript qui simule :

- Le scraping automatique de données via des requêtes simulées,
- Une protection qui limite le nombre de requêtes pour éviter l'abus.

Le scraping simulé

Le scraping ici est une simulation qui génère régulièrement des données utilisateur fictives, comme si une API était interrogée automatiquement plusieurs fois par seconde.

simulateScraping() : c'est la fonction principale qui simule l'envoi d'une requête de scraping.

- Elle vérifie d'abord si le scraping est actif (isScraping === true).
- Elle nettoie la liste des requêtes enregistrées dans la dernière seconde pour ne garder que celles récentes (avec cleanRequestsWindow()).
- Si la protection est activée et que le nombre de requêtes dans la dernière seconde dépasse la limite (rateLimit = 5), alors la requête est **bloquée** (on incrémente le compteur blockedCount, on affiche un message dans le journal et la requête n'est pas traitée).
- Sinon, la requête est comptabilisée (requestCount++), on génère un utilisateur aléatoire (generateRandomUser()), on affiche ses données dans le tableau, et on log l'événement.

```
function simulateScraping() {
  if (!isScraping) return;

  cleanRequestsWindow();

  if (protectionEnabled && requestsInWindow.length >= rateLimit) {
    blockedCount++;
    updateBlockedCount(blockedCount);
    updateEventLog("Requête bloquée : Limite de taux dépassée");
    return;
  }
```

- Le scraping se lance ou s'arrête via le bouton **"Lancer le Scraping" / "Arrêter le Scraping"**.
- Quand le scraping est actif, la fonction simulateScraping() est appelée toutes les 200 millisecondes (soit 5 fois par seconde), ce qui correspond à la limite imposée.

Génération des données utilisateurs

- La fonction generateRandomUser(id) crée un objet utilisateur avec des données aléatoires (nom, email, téléphone, etc.) pour simuler les informations qui pourraient être extraites d'une API réelle.
- Chaque utilisateur généré a un identifiant unique basé sur le nombre de requêtes effectuées.
- Ces données sont ensuite affichées dynamiquement dans un tableau HTML.

```
function generateRandomUser(id) {
  const names = ["Alice", "Bob", "Charlie", "David", "Eve", "Frank", "Grace", "Hannah"];
  const locations = ["Paris", "Lyon", "Marseille", "Toulouse"];
  const genders = ["Homme", "Femme"];

  const name = names[Math.floor(Math.random() * names.length)] + id;
  const email = `user${id}@example.com`;
  const username = `user_${id}`;
  const phone = `+33${Math.floor(1000000 + Math.random() * 9000000)}`;
  const location = locations[Math.floor(Math.random() * locations.length)];
  const birthday = `${1970 + Math.floor(Math.random() * 35)}-${(1 + Math.floor(Math.random()
  const gender = genders[Math.floor(Math.random() * genders.length)];
```

Protection contre le scraping excessif (Rate Limiting)

Objectif

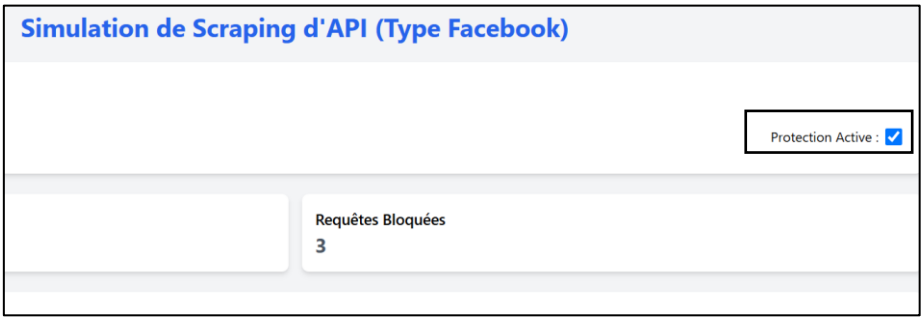
Empêcher un attaquant d'envoyer trop de requêtes en un temps très court, ce qui pourrait saturer le service et permettre la collecte massive de données.

Comment c'est fait ici ?

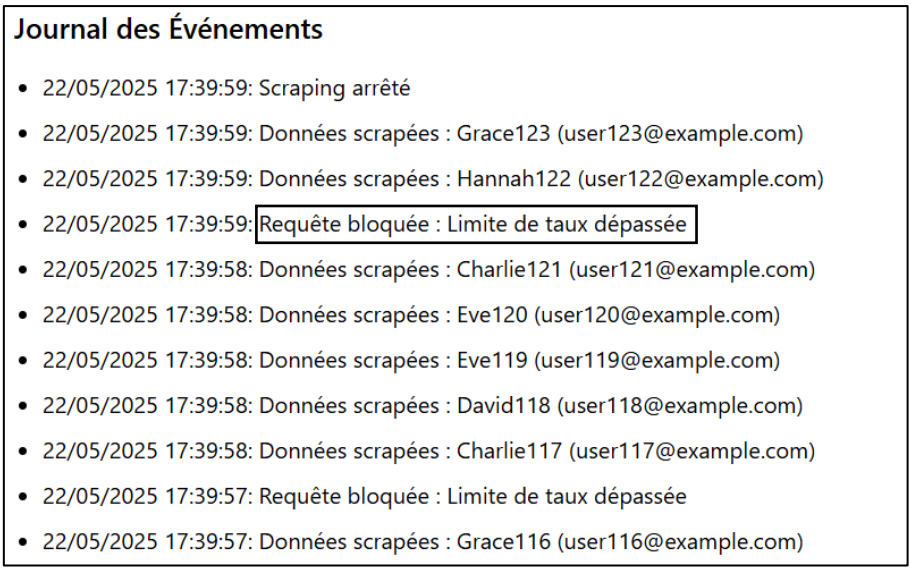
- La variable rateLimit est fixée à 5 requêtes par seconde.
- La liste requestsInWindow garde la trace de tous les timestamps (en millisecondes) des requêtes envoyées au cours de la dernière seconde.
- La fonction cleanRequestsWindow() supprime les timestamps plus vieux que 1 seconde.
- Avant de traiter une requête dans simulateScraping(), on vérifie la taille de requestsInWindow.
    - Si elle dépasse rateLimit ET que la protection est activée (protectionEnabled == true), la requête est bloquée.
- Le compteur de requêtes bloquées est mis à jour, et un message est affiché dans le journal des événements.
- Si la protection est désactivée, il n'y a aucune limitation et toutes les requêtes sont traitées.

Interaction utilisateur

- L’utilisateur peut activer/désactiver la protection avec une checkbox. Cela montre l’impact direct des mesures de sécurité sur le comportement du scraping.



- Le journal des événements affiche en temps réel ce qui se passe (scraping démarré/arrêté, requêtes bloquées, données scrapées).



- Les compteurs affichent en temps réel le nombre total de requêtes effectuées et bloquées.

Conclusion

La fuite massive de données personnelles, comme celle qui a touché plus d'un milliard de comptes Facebook, illustre parfaitement les risques encourus lorsque les APIs ne sont pas suffisamment protégées. Cette simulation a mis en lumière la vulnérabilité des interfaces accessibles face aux attaques de scraping massives, où des acteurs malveillants peuvent automatiquement collecter des données sensibles en exploitant des failles de sécurité.

À travers ce projet, nous avons compris combien il est crucial de renforcer les mécanismes de protection des APIs, notamment via la limitation du nombre de requêtes (rate limiting), une authentification renforcée et une surveillance continue des activités. Ces mesures sont indispensables pour prévenir les abus, protéger la confidentialité des utilisateurs et préserver la réputation des entreprises.

Cette vigilance est une responsabilité partagée entre les entreprises et les utilisateurs, qui doivent être conscients des risques liés à la divulgation de leurs données en ligne. En combinant compréhension théorique et simulation pratique, ce travail rappelle l'importance d'une approche proactive et innovante pour garantir la sécurité des données dans un environnement numérique en constante évolution.

Je reste convaincu que la sensibilisation et l'amélioration continue des protections sont des leviers essentiels pour relever ces défis et assurer un Internet plus sûr pour tous.