Piratage

Il existe diverses façons de pirater, et ce sujet est très intéressant. En tant que hacker White Hat amateur, je trouve qu'il y a beaucoup de connaissances à acquérir dans ce domaine. Ici, je vais enregistrer quelques méthodes que j'ai explorées.

Mots de passe par défaut

Certains sites web, y compris ceux d'agences gouvernementales, utilisent encore des mots de passe par défaut. Bien que de nombreuses entreprises ou utilisateurs modifient leurs identifiants par défaut, d'autres négligent de le faire. Les utilisateurs sont souvent paresseux, et des mots de passe comme `12345678` sont encore couramment utilisés. Cela est particulièrement vrai pour les systèmes plus anciens ou spécialisés.

nmap ou Netcat

Ces outils sont utilisés pour scanner les ports d'un serveur. Portez une attention particulière aux ports couramment utilisés tels que 80, 22 et 443. Pour les instances AWS, le nom d'utilisateur par défaut est `ec2-user`. Pour les instances Azure, c'est `azure-user`. Pour les instances Google Cloud, le nom d'utilisateur par défaut est généralement `ubuntu` ou `google-cloud`. Pour les autres instances cloud, c'est généralement `root`.

Utilisation de la Console du Navigateur

La console du navigateur est utile pour inspecter des informations cachées. Parfois, des données critiques sont intégrées dans le code HTML ou JavaScript, mais ne sont pas visibles directement sur la page elle-même.

Portes dérobées

Dans la vie, les portes dérobées permettent une entrée non autorisée dans les bâtiments, souvent sans être remarquées ou surveillées, comme les parkings ou les portes latérales. De même, les systèmes peuvent avoir des portes dérobées cachées qui contournent les protocoles de sécurité normaux.

Ingénierie Sociale

Les surnoms des gens, leurs dates de naissance et leurs publications sur les réseaux sociaux peuvent révéler beaucoup d'informations personnelles. Souvent, ces détails sont utilisés pour créer des mots de passe faibles. Pour les réseaux Wi-Fi, connaître le numéro de maison de quelqu'un ou d'autres détails d'identification peut aider à deviner leur SSID ou leur mot de passe.

Injection SQL

Pour tout champ de saisie, tester avec `? 1=1` est une technique courante pour identifier les vulnérabilités et les points d'injection SQL potentiels.

API Actuator ou Health

Pour les serveurs d'API, des applications comme Spring Boot offrent un point de terminaison `/actuator` qui fournit des données sur l'état de la machine et de l'application. D'autres frameworks web disposent également de fonctionnalités similaires qui peuvent exposer des informations sensibles sur le serveur.

Surveillance du trafic

Pour comprendre comment le frontend interagit avec le backend, utilisez des applications de proxy comme Charles Proxy sur macOS pour enregistrer et analyser les journaux de requêtes. Cela peut vous donner un aperçu des chemins et des échanges de données entre les composants.

Limites et cas limites des API

Les API (Interfaces de Programmation d'Applications) sont des outils puissants pour intégrer et interagir avec des services externes. Cependant, comme tout outil, elles ont leurs limites et des cas limites qu'il est important de comprendre pour éviter des erreurs ou des comportements inattendus dans vos applications. Voici quelques points à considérer :

1. Limites de débit (Rate Limits)

La plupart des API imposent des limites de débit pour éviter une surcharge de leurs serveurs. Ces limites peuvent être basées sur le nombre de requêtes par minute, par heure ou par jour. Si vous dépassez ces limites, vous risquez de recevoir des erreurs `429 Too Many Requests` ou d'être temporairement bloqué.

Exemple :

```
{
  "error": "rate_limit_exceeded",
  "message": "You have exceeded the rate limit for this API."
}
```

2. Limites de taille des données

Certaines API ont des limites sur la taille des données que vous pouvez envoyer ou recevoir dans une seule requête. Par exemple, une API peut limiter la taille d'un fichier uploadé à 10 Mo. Si vous dépassez cette limite, la requête échouera.

Exemple :

```
{
  "error": "payload_too_large",
  "message": "The request payload exceeds the maximum allowed size."
}
```

3. Cas limites de validation des données

Les API peuvent avoir des règles strictes sur le format des données envoyées. Par exemple, un champ de date doit être au format `YYYY-MM-DD`, ou un champ numérique doit être compris entre 1 et 100. Si les données ne respectent pas ces règles, la requête échouera.

Exemple :

```
{
  "error": "invalid_input",
  "message": "The 'date' field must be in the format YYYY-MM-DD."
}
```

4. Gestion des erreurs

Les API peuvent renvoyer différents codes d'erreur en fonction de la nature du problème. Il est important de gérer ces erreurs de manière appropriée dans votre application pour éviter des plantages ou des comportements inattendus.

Exemple :

```
{
  "error": "not_found",
  "message": "The requested resource could not be found."
}
```

5. Cas limites de pagination

Pour les API qui retournent de grandes quantités de données, la pagination est souvent utilisée. Cependant, si vous ne gérez pas correctement la pagination, vous pourriez manquer des données ou surcharger l'API avec des requêtes inutiles.

Exemple :

```
{
  "data": [...],
  "next_page": "https://api.example.com/resource?page=2"
}
```

6. Limites de disponibilité

Les API ne sont pas toujours disponibles à 100%. Elles peuvent subir des temps d'arrêt pour maintenance, des problèmes de réseau, ou d'autres problèmes techniques. Il est important de prévoir des mécanismes de réessai et de gestion des erreurs dans votre application pour faire face à ces situations.

Exemple :

```
{
  "error": "service_unavailable",
  "message": "The API is currently unavailable. Please try again later."
}
```

7. Cas limites de sécurité

Les API peuvent avoir des restrictions de sécurité, comme l'exigence d'une authentification ou d'une autorisation spécifique pour accéder à certaines ressources. Si vous ne respectez pas ces règles, vous pourriez recevoir des erreurs `401 Unauthorized` ou `403 Forbidden`.

Exemple :

```
{
  "error": "unauthorized",
  "message": "You are not authorized to access this resource."
}
```

Conclusion

Comprendre les limites et les cas limites des API que vous utilisez est essentiel pour développer des applications robustes et fiables. En anticipant ces problèmes et en mettant en place des mécanismes de gestion des erreurs appropriés, vous pouvez éviter de nombreux écueils courants et offrir une meilleure expérience utilisateur.

Il est important de tester les limites et les cas limites d'une API ou d'un serveur. Une attaque par déni de service distribué (DDoS) tente de submerger les limites de requêtes. De plus, les cas limites sont des scénarios où les API pourraient permettre l'accès à des données restreintes. Tester ces cas peut aider à s'assurer que les contrôles d'accès appropriés sont en place.

Panneaux d'administration

Parfois, les panneaux d'administration ou les interfaces internes ne sont pas suffisamment protégés. Il est utile d'essayer d'accéder à des chemins comme `/admin` ou de visiter des sous-domaines tels que `admin.xx.com` pour vérifier si ces zones sont correctement sécurisées.