HTTP - Methods

Description

Les méthodes HTTP sont des verbes utilisés dans les requêtes HTTP pour indiquer l'action à effectuer sur une ressource. Les méthodes HTTP les plus courantes sont GET, POST, PUT et DELETE. Chacune de ces méthodes est utilisée dans différentes situations et peut être utilisée selon les besoins de l'application.

D'après la RFC 9110 :

- **GET** sert à obtenir une représentation de la ressource spécifiée. Par exemple, on récupérera des informations sur un utilisateur avec un GET (GET /api/v1/users/1234).
- **POST** sert à soumettre une nouvelle entité à la ressource spécifiée. Par exemple, on enregistrera un nouvel utilisateur sur une plateforme avec un formulaire soumis via un POST (POST /api/v1/users).
- **PUT** sert à remplacer les données de la ressource spécifiée avec les données de la requête. Par exemple, on mettra à jour les informations d'un utilisateur avec la méthode PUT (PUT /api/v1/users/1234)
- **DELETE** sert à supprimer la ressource spécifiée. Par exemple, on supprimera un utilisateur avec la méthode DELETE (DELETE /api/v1/users/1234)

Exemple d'une requête de type GET :

```
GET /example/page.html HTTP/1.1 <- Position de la méthode de type GET
Host: www.example.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/80.0.3987.149 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed
-exchange;v=b3;q=0.9
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Cookie: sessionID=12345; username=john.doe
```

Réponse du serveur :

```
HTTP/1.1 200 OK
Date: Thu, 31 Mar 2023 10:15:00 GMT
Content-Type: text/html; charset=UTF-8
Content-Length: 3456
Connection: keep-alive
Server: Apache/2.4.41 (Ubuntu)
X-Powered-By: PHP/7.4.3

<!DOCTYPE html>
<html>
<head>
  <title>Example Page</title>
</head>
<body>
  <h1>Welcome to the Example Page</h1>
  <p>This is an example page.</p>
</body>
</html>
```

On notera que toutes les APIs et toutes les applications ne respectent pas nécessairement la RFC.

Les services web exposent parfois involontairement des ressources qui ne devraient pas être accessibles, mais qui le sont via la modification de la méthode HTTP sur l'appel vers la ressource.

Prérequis d'exploitation

Pour exploiter cette vulnérabilité, il est nécessaire d’avoir accès à une application qui ne filtre pas correctement les méthodes HTTP autorisées.

Connaissances nécessaires

- Connaissance de base du fonctionnement du protocole HTTP ;
- Maîtrise de la notion de méthode HTTP.

Outils nécessaires

- Outils de modification et/ou d'interception de requêtes (Burp, Curl).

Flux d'exécution

Explorer

Naviguer sur l'application pour récupérer les endpoints et/ou pages de l'application pour en déterminer les méthodes HTTP utilisées pour l'accès, la modification, la création et la suppression des données.

Expérimenter

Pour chacune des requêtes identifiées, envoyer la même requête, mais en utilisant une méthode différente. L'objectif ici est de potentiellement remarquer un comportement particulier de l'application.

Exploiter

Conséquences potentielles

Une exploitation réussie de ce type de vulnérabilité peut permettre :

- Le contournement de certains mécanismes de contrôle d'accès ;
- L'accès à des ressources sensibles et/ou à un espace privé de l'application.

Contre-mesures

Les contre-mesures suivantes peuvent être mises en œuvre :

- Configurer le serveur web pour n'autoriser que les méthodes HTTP que vous souhaitez utiliser. Par exemple, autoriser uniquement les méthodes GET et POST et bloquer les autres méthodes ;
- Utiliser des contrôles d'accès basés sur les rôles et les autorisations pour limiter l'accès aux différentes ressources et fonctionnalités de votre application ;
- Utiliser un pare-feu applicatif pour bloquer les requêtes HTTP non autorisées ou suspectes.

Comment cela fonctionne

Les scénarios suivants peuvent être joués via l'exploitation de cette vulnérabilité :

- L'utilisation de la méthode delete dans une requête HTTP pour supprimer une ressource sur le serveur ;
- L'utilisation de la méthode put dans une requête HTTP pour mettre à jour une ressource sur le serveur sans avoir l'autorisation adéquate.

Exemple 1

Voici une configuration Apache potentiellement vulnérable aux modifications de méthodes HTTP :

```
<Directory /var/www/html>
    AllowOverride None
    Require all granted
</Directory>
```

Cette configuration permet aux utilisateurs de l'application de faire des requêtes HTTP avec n'importe quelle méthode. Par conséquent, quelqu'un pourrait essayer d'envoyer une requête HTTP avec une méthode telle que DELETE ou PUT pour essayer de supprimer ou de modifier des données sur le serveur serveur.

Voici une nouvelle version de la configuration Apache :

```
<Directory /var/www/html>
    AllowOverride None
    <Limit GET POST>
        Require all granted
    </Limit>
</Directory>
```

Cette configuration permet uniquement les méthodes GET et POST, ce qui signifie que toutes les autres méthodes seront bloquées. Cette limitation peut aider à protéger un serveur contre les modifications non autorisées via les méthodes HTTP.

Exemple 2

Un attaquant identifie une ressource protégée par le cookie de session sur le site “vulnerable.org”. La ressource retourne les informations d'un utilisateur si celui-ci est authentifié via l'accès en GET à /api/v1/user/{user*id}/info L'attaquant a le user*id 1234, mais souhaite obtenir les infos de l'utilisateur ayant l'id 5678 :

```
$ curl -X GET vulnerable.org/api/v1//user/5678/info -i
> < HTTP/2 401
> [...]
> Forbidden
```

En modifiant la méthode HTTP utilisée, l'attaquant accède aux données d'un autre utilisateur :

```
$ curl -X POST vulnerable.org/api/v1//user/5678/info -i
> < HTTP/2 200
> [...]
> username: victim
> [...]
```

CWEs

- CWE-650 : Trusting HTTP Permission Methods on the Server Side
    - The server contains a protection mechanism that assumes that any URI that is accessed using HTTP GET will not cause a state change to the associated resource. This might allow attackers to bypass intended access restrictions and conduct resource modification and deletion attacks, since some applications allow GET to modify state.

References

URL :

- https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods
- https://resources.infosecinstitute.com/topic/http-verb-tempering-bypassing-web-authentication-and-authorization/
- https://www.rfc-editor.org/rfc/rfc9110

Retour fiches vulnérabilités

- Cyber fiches vulnérabilités