HTTP - POST

Description

La méthode POST du protocole HTTP est une méthode qui peut être utilisée pour envoyer des données d'une requête client à un serveur web. Elle est souvent utilisée pour envoyer des données d'un formulaire HTML à un serveur web, mais elle peut également être utilisée pour envoyer d'autres types de données.

Prérequis d'exploitation

Pour exploiter cette méthode HTTP, il est nécessaire d'avoir accès à une application qui fonctionne sur un serveur acceptant les requêtes POST.

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 qui acceptent les requêtes POST. En général, nous commençons par rechercher l'ensemble des formulaires étant donné que ces derniers utilisent très souvent la méthode POST lors de l'envoi des données à destination du serveur web.

Expérimenter

L'objectif ici est de réaliser plusieurs soumissions de formulaires avec des données différentes. Il s'agira notamment de manipuler les valeurs des différents champs du formulaire afin de créer un comportement potentiellement anormal sur le serveur. Ce comportement anormal se caractérisera souvent par une réponse HTTP différentes des autres.

Exploiter

Consulter les solutions de chaque challenge.

Conséquences potentielles

Une exploitation réussie de ce type de vulnérabilité peut permettre :

- L'accès à des ressources sensibles ;
- L'accès à des fonctionnalités sensibles.

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 ;
- 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

Le scénario suivant peut être joué via l'exploitation de cette vulnérabilité :

- Une application contient une authentification HTTP de type basic ou digest sur un répertoire (ex : /admin). Cette authentification est configurée uniquement sur la méthode GET. Un attaquant peut alors contourner cette authentification en effectuant une requête sur /admin avec la méthode POST et ainsi accéder au répertoire /admin.

Exemple 1

Voici un exemple de configuration Apache qui pourrait être vulnérable à une attaque par la méthode POST :

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

Cette configuration permet à n'importe quel utilisateur d'envoyer une requête POST au serveur Apache, sans aucune restriction. Si un attaquant parvient à envoyer du code malveillant dans une requête POST, il peut alors compromettre le serveur.

On peut corriger cette configuration en limitant la taille des données envoyées par la requête POST, en exigeant que l'utilisateur qui soumet la requête soit préalablement authentifié. On veillera également à valider les entrées utilisateur.

Exemple 2

Voici un exemple d'illustration d'un potentiel bypass d'accès à une ressource protégée via la modification du verbe HTTP GET en POST :

```
$ curl -X GET vulnerable.org/admin
> [...]
> < HTTP/2 403
> [...]

$ curl -X POST vulnerable.org/admin
> [...]
> < HTTP/2 200
> [...]
```

Exemple 3

Supposons que nous ayons un formulaire d'inscription avec les champs “Nom”, “Prénom” et “Adresse email”. Lorsque l'utilisateur soumet le formulaire, les données sont envoyées à un endpoint sur un serveur Web pour traitement.

Voici un exemple de requête POST envoyée à cet endpoint pour soumettre les données du formulaire :

```
POST /inscription HTTP/1.1
Host: www.example.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 45

nom=John&prenom=Doe&email=john.doe@example.com
```

Explication de la requête :

- La première ligne indique le type de méthode HTTP utilisée (POST) et l'URI relative de l'endpoint (/inscription) ;
- La ligne suivante indique l'hôte cible (www.example.com) ;
- La ligne suivante indique le type de contenu envoyé dans la requête (application/x-www-form-urlencoded) ;
- La ligne suivante indique la longueur du corps de la requête (45 octets) ;
- Le corps de la requête contient les données du formulaire encodées en URL. Dans cet exemple, les données envoyées sont le nom “John”, le prénom “Doe” et l'adresse email “john.doe@example.com”.

Une fois la requête POST envoyée à l'endpoint, le serveur Web peut traiter les données envoyées et les stocker dans une base de données ou effectuer d'autres opérations nécessaires.

References

URL :

- https://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html
- http://stssnsb.free.fr/telecharger/blusson/stssnir/cours/requetesHTTP.pdf
- https://old.mu.ac.in/wp-content/uploads/2017/11/WP_CBSGS_Paper-3-May-2017_Solution.pdf

Retour fiches vulnérabilités

- Cyber fiches vulnérabilités