HTTP - Headers

Description

Les Headers HTTP (HyperText Transfer Protocol) sont des en-têtes qui accompagnent les requêtes et les réponses HTTP dans le cadre de la communication entre un client (généralement un navigateur web) et un serveur. Ils permettent de transmettre des informations supplémentaires sur la requête ou la réponse, telles que l'encodage des caractères utilisés, le type de contenu, la taille du contenu, la date de dernière modification, etc.

Pré-requis d'exploitation

Pour exploiter cette vulnérabilité, il est nécessaire d'avoir accès à une application qui ne filtre pas correctement les headers HTTP autorisés.

Connaissances nécessaires

- Connaissance de base du fonctionnement du protocole HTTP ;
- Maîtrise de la notion d'en-têtes 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 puis énumérer les headers et leurs configurations afin de les analyser.

Expérimenter

Tenter d'envoyer différentes valeurs dans les headers HTTP et analyser comment l'application réagit à chaque valeur.

Exploiter

Conséquences potentielles

Une exploitation réussie de ce type de vulnérabilité peut permettre

- L'accès à des pages 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 headers HTTP que vous souhaitez utiliser. Par exemple, autoriser uniquement certains headers tels que "Content-Type" et "Accept" et bloquer les autres.
- 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 d'une valeur falsifiée dans l'en-tête "Referer" pour tenter de tromper un serveur en lui faisant croire qu'une requête provient d'un site de confiance.
- L'utilisation d'une valeur falsifiée dans l'en-tête "User-Agent" pour tenter de contourner les contrôles de sécurité basés sur le navigateur de l'utilisateur.

Exemple 1

Voici un exemple de code qui utilise un Header HTTP vulnérable :

```
app.use((req, res, next) => {
  res.header('Access-Control-Allow-Origin', '*');
  next();
});
```

Ce code ajoute un Header "Access-Control-Allow-Origin" à chaque réponse HTTP envoyée par l'application, avec une valeur de "*" qui indique que tous les domaines sont autorisés à accéder aux ressources de l'application. Cela signifie que n'importe quel site web peut accéder aux ressources de l'application via un navigateur web, ce qui peut être une vulnérabilité importante si l'application stocke des données sensibles ou critiques.

Voici un patch du code vulnérable :

```
app.use((req, res, next) => {
  res.header('Access-Control-Allow-Origin', req.headers.origin);
  next();
});
```

Pour patcher ce code vulnérable, vous pouvez remplacer la valeur "*" du Header "Access-Control-Allow-Origin" par la valeur de l'en-tête "Origin" de la requête HTTP. Cela permet de limiter l'accès aux ressources de l'application aux domaines qui en font la demande explicitement. Cela renforce la sécurité de l'application en empêchant les sites tiers de récupérer ou de modifier les données de l'application de manière non autorisée.

Exemple 2

Voici un exemple simplifié d'une requête HTTP avec les en-têtes :

```
POST /api/login HTTP/1.1
Host: www.example.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/90.0.4430.212 Safari/537.36
Content-Type: application/json
Content-Length: 35
{"username": "john_doe", "password": "pass123"}
```

- La première ligne de la requête indique la méthode HTTP utilisée (dans ce cas, POST), l'URI cible (/api/login) et la version du protocole HTTP (HTTP/1.1).
- L'en-tête "Host" spécifie le nom de domaine du serveur cible, dans cet exemple "www.example.com".
- L'en-tête "User-Agent" fournit des informations sur le navigateur ou l'application cliente utilisée pour envoyer la requête. Dans ce cas, c'est un navigateur Chrome sur Windows.
- L'en-tête "Content-Type" indique le type de contenu envoyé dans le corps de la requête. Ici, le corps est du format JSON (application/json).
- L'en-tête "Content-Length" donne la taille en octets du corps de la requête, qui est de 35 octets dans cet exemple.
- Le corps de la requête est le contenu JSON envoyé au serveur. Il contient les informations de connexion, ici un nom d'utilisateur ("john_doe") et un mot de passe ("pass123").

Voici un exemple simplifié d'une réponse HTTP avec les en-têtes :

HTTP/1.1 200 OK Date: Wed, 27 Jul 2023 12:00:00 GMT Server: Apache/2.4.41 (Ubuntu) Content-Length: 27 Content-Type: application/json {"status": "success", "token": "abc123"}

1. La première ligne de la réponse indique la version du protocole HTTP (HTTP/1.1) et le code de statut 200, ce qui signifie que la requête a été traitée avec succès.
2. Les en-têtes “Date” et “Server” fournissent des informations sur le moment où la réponse a été générée et le serveur qui l'a envoyée.
3. L'en-tête “Content-Length” indique la taille en octets du corps de la réponse, qui est de 27 octets dans cet exemple.
4. L'en-tête “Content-Type” indique le type de contenu envoyé dans le corps de la réponse. Ici, le corps est du format JSON (application/json).
5. Le corps de la réponse est le contenu JSON renvoyé par le serveur. Dans cet exemple, le serveur renvoie un objet JSON avec deux clés : “status” avec la valeur “success” pour indiquer que la connexion a réussi, et “token” avec la valeur “abc123”, qui est un jeton d'authentification à utiliser dans les futures requêtes.

References

URL :

- https://www.tutorialspoint.com/http/pdf/http_header_fields.pdf
- https://owasp.org/www-chapter-ghana/assets/slides/HTTP_Header_Security.pdf

Retour fiches vulnérabilités

- Cyber fiches vulnérabilités