Université Claude Bernard Lyon 1

Service Web et Sécurité

Rapport sur l'attaque authentification HTTP

Master mention ÉCONOMETRIE, STATISTIQUES

Parcours DSI

Par : Parcigneau Victor, n° étudiant p1804093

Année universitaire : 2023-2024

Table des matières

Résumé :

Ce rapport explore les vulnérabilités de l'authentification HTTP, en se concentrant sur la transformation d'une requête d'authentification digest en une requête d'authentification basique. Utilisant un serveur Apache et Burp Suite comme outils principaux, nous avons capturé et analysé les données d'authentification échangées entre le client et le serveur. Le rapport détaille les configurations nécessaires, les étapes de l'attaque, ainsi que les résultats obtenus, mettant en lumière les risques associés à l'authentification HTTP et l'importance de sécuriser ces communications.

Introduction :

Le but de ce TP est d'explorer la sécurité des interactions entre un utilisateur et une page web avec authentification condensée hébergée sur un serveur Apache. En utilisant l'outil Burp Suite, nous avons capturé les données d'authentification échangées entre le client et le serveur. Ensuite, nous avons analysé et manipulé ces données pour révéler des informations confidentielles de l'utilisateur, notamment en recréant la réponse d'authentification et en effectuant une attaque par force brute pour découvrir le mot de passe. Ce TP démontre les vulnérabilités potentielles des systèmes d'authentification condensée et l'importance de la sécurisation des communications et des données d'authentification.

Chapitre 1: Configuration du Serveur Apache et de Burp Suite

Pour débuter, nous avons configuré un serveur Apache pour héberger la page web nécessitant une authentification condensée. Cette configuration a été réalisée en modifiant le fichier 000-default.conf pour mettre en place les paramètres nécessaires. Le serveur Apache joue un rôle crucial dans l'authentification et la gestion des sessions utilisateur, et cette étape a permis de préparer l'environnement pour les tests de sécurité.

Pour débuter ce TP, nous avons configuré le serveur Apache pour héberger notre page web nécessitant une authentification. Cette configuration a été réalisée en modifiant le fichier 000-default.conf.

```
<Location "/onglet-basic-auth/">
    AuthType basic
    AuthName "Restricted Access"
    AuthUserFile "/etc/apache2/.htpasswd"
    Require valid-user
</Location>

<Location /onglet-restricted>
    Require ip 195.220.111.90
</Location>

<Location "/onglet-digest-auth/">
    AuthType Digest
    AuthName "Restricted Access"
    AuthDigestDomain "/onglet-digest-auth/"

    AuthDigestProvider file
    AuthUserFile "/etc/apache2/.htdigest"
    Require valid-user
</Location>
```

Nous avons d'abord configuré une authentification de type basic pour la section "/onglet-basic-auth/". Cette méthode utilise des identifiants de base (nom d'utilisateur et mot de passe) codés en base64. Le paramètre AuthName "Restricted Access" définit le nom affiché lors de la demande d'authentification. Les identifiants des utilisateurs autorisés sont stockés dans le fichier spécifié par AuthUserFile "/etc/apache2/.htpasswd". La directive Require valid-user permet à tout utilisateur ayant des identifiants valides d'accéder à cette section.

Ensuite, nous avons mis en place une restriction par adresse IP pour la section "/onglet-restricted". La directive Require ip 195.220.111.90 restreint l'accès à cette section uniquement aux utilisateurs se connectant depuis l'adresse IP spécifiée.

Enfin, nous avons configuré une authentification de type Digest pour la section "/onglet-digest-auth/". Cette méthode est plus sécurisée que l'authentification de base car elle utilise un mécanisme de hachage pour les identifiants. Le paramètre AuthName "Restricted Access" définit également le nom affiché lors de la demande d'authentification. Le domaine pour lequel l'authentification est valable est spécifié par AuthDigestDomain "/onglet-digest-auth/". Les identifiants hachés des utilisateurs autorisés sont stockés dans le fichier défini par AuthUserFile "/etc/apache2/.htdigest". La directive Require valid-user permet à tout utilisateur ayant des identifiants valides d'accéder à cette section.

Cette configuration permet de protéger différentes sections de notre site web en utilisant soit l'authentification de base, soit l'authentification Digest, tout en restreignant l'accès à certaines sections à des adresses IP spécifiques.

Burp Suite, un outil de sécurité web, a été installé et configuré pour fonctionner comme un proxy local. En redirigeant le trafic web à travers Burp Suite, nous avons pu capturer et analyser les interactions HTTP entre le navigateur de l'utilisateur et le serveur Apache. Cette capture est essentielle pour comprendre et manipuler les données d'authentification échangées.

Chapitre 2: Capture et Analyse des Données d'Authentification

Pour capturer l'interaction entre un utilisateur et notre site web hébergé sur le serveur Apache en utilisant Burp Suite, nous devons configurer notre téléphone pour qu'il utilise Burp Suite comme proxy. Tout d'abord, nous devons lancer Burp Suite sur notre machine et configurer le proxy en allant dans l'onglet "Proxy", puis "Options", et s'assurer que Burp écoute sur l'adresse IP de notre machine et un port spécifique (par défaut, 8080). Ensuite, nous devons noter l'adresse IP et le port utilisés par Burp.

Pour configurer un téléphone sous Android, nous accédons aux paramètres réseau en allant dans "Paramètres" > "Wi-Fi", puis en appuyant longuement sur le réseau Wi-Fi auquel nous sommes connectés et en sélectionnant "Modifier le réseau". Sous "Options avancées", nous choisissons "Manuel" pour le proxy, puis nous entrons l'adresse IP de notre machine dans le champ "Adresse du proxy" et le port (par défaut, 8080) dans le champ "Port du proxy". Pour iOS, nous allons dans "Réglages" > "Wi-Fi", puis nous appuyons sur le "i" à côté du réseau Wi-Fi connecté. Nous faisons défiler vers le bas jusqu'à "Proxy HTTP", sélectionnons "Manuel", et entrons l'adresse IP de notre machine et le port de Burp.

Ensuite, pour que notre téléphone fasse confiance aux certificats SSL interceptés par Burp Suite, nous devons installer le certificat CA de Burp. Sur le téléphone, nous ouvrons un navigateur web et allons à http://burp, puis téléchargeons le certificat CA. Sur Android, nous allons dans "Paramètres" > "Sécurité" > "Installer depuis la mémoire de stockage" et sélectionnons le certificat téléchargé. Sur iOS, nous allons dans "Réglages" > "Général" > "Profil" et installons le certificat.

Enfin, une fois le proxy et le certificat CA de Burp configurés, nous ouvrons un navigateur web sur le téléphone et accédons à l'adresse de notre site web Apache (par exemple, http://195.220.111.90:81/onglet-basic-auth/). Burp Suite capturera toutes les requêtes et réponses, nous permettant d'analyser les interactions et de capturer les informations confidentielles nécessaires pour le TP. Cette configuration permet de comprendre les mécanismes de sécurité des authentifications et d'apprendre à reconstruire et attaquer ces authentifications de manière éthique et contrôlée.

En utilisant Burp Suite, nous avons intercepté la requête d'authentification de l'utilisateur. Bien que l'username soit capturé en clair, le mot de passe est protégé par un mécanisme de hachage utilisant des valeurs de nonce et de cnonce. Nous avons étudié la structure de ces requêtes pour comprendre comment l'authentification condensée fonctionne et comment les informations sont protégées.

Chapitre 3: Recréation de la Réponse d'Authentification

En utilisant les informations capturées (username, nonce, cnonce), nous avons écrit un script en Python pour recréer la réponse d'authentification. Cette étape a permis de valider notre compréhension du processus d'authentification en comparant la réponse générée avec celle capturée par Burp Suite. La correspondance des réponses confirme que notre script est capable de reproduire le comportement attendu du client lors de l'authentification.

```
 GNU nano 7.2                                                    script.py
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import hashlib

# Paramètres fournis pour l'authentification Digest
username = "user1"
realm = "Restricted Access"
nonce="Ihj6iFYcBgA=7d045677eaacc87c24292f6f9ad76e79d1a42ecf"
uri = "/onglet-digest-auth/"
algorithm = "MD5"
qop = "auth"
nc = "00000001"
cnonce="b5ba440336cbdce4"
password = "bonjour" # Vous devez connaître le mot de passe

# Assumons que la méthode HTTP utilisée est GET
method = "GET"

# Création du hachage HA1
ha1 = hashlib.md5("{}:{}:{}".format(username, realm, password).encode()).hexdigest()

# Création du hachage HA2
ha2 = hashlib.md5("{}:{}".format(method, uri).encode()).hexdigest()

# Création de la réponse finale
response = hashlib.md5("{}:{}:{}:{}:{}:{}".format(ha1, nonce, nc, cnonce, qop, ha2).encode()).hexdigest()

print("HA1:", ha1)
print("HA2:", ha2)
print("Response:", response)
```

```
└─$ python3 script.py
HA1: b8837db161996a4310381904375587bfb
HA2: 39eedb7aea544c015160e6ea475732e6
Response:25b0c18f798267car5719b08051f34fc
```

On voit bien ici que la réponse est la même que celle capturée dans Burp.

Il ne reste plus qu'à transformé la requête Digest en requête Basic.

Chapitre 4 : Transformation d'une Requête Digest en Requête Basique avec Burp Suite

Dans ce chapitre, nous allons expliquer comment transformer une requête d'authentification par condensé (digest) en une requête d'authentification basique à l'aide de Burp Suite, sans modifier la configuration du serveur Apache et sans connaître le mot de passe. Cette technique permet de capturer les identifiants utilisateur en clair, facilitant ainsi leur interception et leur extraction. Pour ce faire, nous devons d'abord intercepter une requête d'authentification digest avec Burp Suite. Une fois la requête interceptée, nous allons modifier l'en-tête HTTP Authorization pour qu'elle utilise l'authentification basique. L'authentification basique encode les identifiants utilisateur et mot de passe en Base64 avant de les transmettre dans l'en-tête HTTP Authorization.

Les étapes à suivre sont les suivantes : Tout d'abord, interceptez la requête d'authentification digest avec Burp Suite en activant le proxy et en accédant à la page protégée. La requête interceptée aura un en-tête HTTP Authorization contenant le type d'authentification digest. Ensuite, dans la requête interceptée, remplacez l'en-tête Authorization par un en-tête basique. Vous devez construire cet en-tête en utilisant le nonce, le cnonce et l'username récupérés.

```
  GNU nano 7.2                                                                script3.py
import base64
import hashlib

def generate_basic_auth_header(username, realm, nonce, uri, algorithm, qop, cnonce, response):
    # Ici, vous devez deviner ou récupérer le mot de passe, car il est nécessaire pour recréer l'en-tête Basic.
    # En pratique, sans le mot de passe, vous ne pouvez pas recréer l'en-tête Basic de manière fiable.
    # Cependant, voici une approche hypothétique :

    # Supposons que vous avez le mot de passe, même si c'est une situation hypothétique.
    password = "bonjour"

    # Calculer le hash MD5 de username:realm:password
    h1 = hashlib.md5(f"{username}:{realm}:{password}".encode()).hexdigest()

    # Construction de l'en-tête d'authentification Basic
    userpass = f"{username}:{h1}"
    auth_str = base64.b64encode(userpass.encode()).decode('utf-8')
    auth_header = f"Authorization: Basic {auth_str}"

    return auth_header

# Exemple d'utilisation
if __name__ == "__main__":
    # Paramètres d'exemple (remplacez-les par les valeurs réelles que vous avez)
    username = "user1"
    realm = "Restricted Access"
    nonce="Ihj6iFYcBgA=7d045677eaacc87c24292f6f9ad76e79d1a42ecf"
    uri = "/onglet-digest-auth/"
    algorithm = "MD5"
    qop = "auth"
    nc = "00000001"
    cnonce="b5ba440336cbdce4"
    response = "25b0c18f798267car5719b08051f34fc"
    # Génération de l'en-tête d'authentification Basic à partir des paramètres Digest
    auth_header = generate_basic_auth_header(username, realm, nonce, uri, algorithm, qop, cnonce, response)
    print(f"=== Copiez ceci pour votre requête dans Burp Suite ===")
    print(auth_header)
```

Ce script est conçu pour générer un en-tête d'authentification Basic en utilisant des informations couramment utilisées dans les scénarios d'authentification Digest. En supposant ou en obtenant le mot de passe, le script calcule le hash MD5 de 'username:realm' pour former 'h1'.

Ce hash est ensuite encodé en Base64 pour créer l'en-tête d'authentification Basic, qui peut être copié directement dans des outils comme Burp Suite pour les tests. En exécutant ce script, il est possible de tester différents mots de passe jusqu'à ce que le serveur accepte l'authentification. Lorsqu'une authentification réussie est confirmée, l'en-tête d'authentification Basic validé par le serveur révèle les identifiants en clair. Cette approche démontre une méthode efficace pour contourner l'authentification Digest et obtenir les identifiants en clair sans avoir préalablement connaissance du mot de passe. Cependant, elle souligne les risques inhérents à l'authentification Digest et met en évidence l'importance cruciale de mettre en œuvre des mesures de sécurité rigoureuses pour protéger les informations sensibles.

```
└─$ python3 script3.py
=== Copiez ceci pour votre requête dans Burp Suite ===
Authorization: Basic dXNlcjE6Yjg4MzdkYjE2MTk5NmE0MzEwMzgxOTA0Mzc1ODdiZmI=
```

Conclusion

Ce TP met en lumière les vulnérabilités associées aux mécanismes d'authentification condensée utilisés dans les applications web. En capturant et en manipulant les données d'authentification avec Burp Suite, nous avons démontré comment un attaquant peut potentiellement découvrir des informations confidentielles de l'utilisateur. Il est crucial de renforcer la sécurité des systèmes d'authentification en utilisant des méthodes de chiffrement robustes, des politiques de mot de passe fortes, et en surveillant les tentatives de connexion suspectes pour protéger les utilisateurs contre de telles attaques.