TP vulnérabilité CRIME sur HTTPS

Clément Fréville, Baptiste Baverel
Bastien Ollier, Clément Laporte

Introduction

Ce sujet a été testé à l'IUT (OpenSSL 1) et sur une machine personnelle (OpenSSL 3). Il ne fonctionne que sous GNU/Linux.

Paquets nécessaires sous Debian : `build-essentials zlib1g-dev libssl-dev openssl`

Préparation

Cloner le dépôt Git à l'IUT ou sur votre machine.

Pour rappel, à l'IUT, le proxy se désactive avec unset `http_proxy`.

```
git clone https://codefirst.iut.uca.fr/git/clement.freville2/http-crime
cd http-crime
```

Compiler le projet avec Make, et générer un certificat auto-signé à l'aide d'OpenSSL.

```
make
openssl req -x509 -newkey rsa:4096 -keyout key.pem -out cert.pem -sha256 -days 365 -
nodes
```

Observation

- Exécuter le serveur HTTPS avec `./server 8080`. Dans un autre terminal, afficher le contenu de la page d'accueil avec `curl -k https://localhost:8080`.
- Tester si la compression TLS est active avec OpenSSL. Pourquoi est-elle désactivée ?

```
openssl s_client -connect 127.0.0.1:8080 -comp < /dev/null
```

- Nous allons recompiler la bibliothèque OpenSSL avec le support de la bibliothèque *zlib*. Comparer avec comment la plupart des distributions GNU/Linux fournissent OpenSSL. Donner une raison pour laquelle l'option `no-zlib` est précisée par ces distributions.

```
cd /tmp
# Selon la version majeure d'OpenSSL (openssl version)
curl https://www.openssl.org/source/openssl-3.1.3.tar.gz | tar xzf -
curl https://www.openssl.org/source/openssl-1.1.1w.tar.gz | tar xzf -
cd openssl-*
./config zlib
make -j$(nproc)
```

- La variable d'environnement `LD_PRELOAD`[1] permet de charger une bibliothèque compilée avant toutes les autres. Cela nous permet dans notre cas de remplacer la version d'OpenSSL du système par notre version. Appeler l'exécutable `server` et `client` avec les fichiers `.so` issus de la compilation d'OpenSSL.
- Observer le comportement du client. Quels arguments prend-il en compte ? Que connaît-on sur la requête HTTPS ?

[1] `man 8 ld.so`

Attaque

Profiter des vulnérabilités de CRIME pour récupérer le *flag* en jeu. Il s'agit d'un cookie nommé `flag` envoyé par le client vers le serveur.

Pour cela, le code Python suivant peut vous être utile. On rappelle le principe de l'attaque CRIME : lorsqu'un attaquant peut contrôler une partie des données, il peut progressivement récupérer le cookie en modifiant les parties et en observant comment la taille totale de la requête change pendant la compression.

Une séquence de caractères répétée sera compressée. Par conséquence, si une partie de l'URL correspond à une partie du cookie, alors la taille de la requête sera plus petite. Il suffit alors de tester toutes les possibilités pour chaque caractère. On continue tant qu'on peut ajouter un caractère à la requête qui n'augmente pas sa taille.

```
from string import ascii_lowercase, ascii_uppercase, digits
from subprocess import check_output

alphabet = ascii_lowercase + ascii_uppercase + digits
env = {
    "LD_PRELOAD": "..." # TODO
}

def request(url: str) -> int:
    """Executes the client binary with the following URL, and checks its output."""
    out = check_output(["./client", "127.0.0.1", "8080", url],
env=env).decode('utf-8')
    return int(out.split(' ')[1])
```