Heartbleed

YIN Yan, ZHANG Lei

Introduction

Bref Histoire

Heartbleed est une vulnérabilité logicielle présente dans la bibliothèque de cryptographie open source OpenSSL à partir de mars 2012, qui permet à un buffer over-read de lire le mémoire d'un serveur ou d'un client pour récupérer[1].

Ce bug est dit Heartbleed due à la réalisation de l'Heartbeat en Openssl. L'extension Hearbeat pout le protocole TLS (Transport Layer Security) est proposée comme un standard en février 2012 par RFC 6520[2]. Il fournit un moyen de tester et de keep-alive (maintenir en vie) de communications sécurisées sans avoir à renégocier la connexion à chaque fois.

En 2011, l'un des auteurs du RFC, Robin Segglemann a implémenté l'extension Heartbeat pour OpenSSL[3]. Avec le bogue dans ce code, la version 1.0.1 d'OpenSSL a lancé le 14 mars 2012, avec le support d'Heartbeat est actif par défaut, ce qui provoque les versions affectées d'être vulnérable.

Ce boque est decouvert par Google et Codenomicon à avril 1 2014, environ 2 ans après le lancement de la première version affectent[4].

Le Protocole SSL

SSL = Secure Sockets Layer, le prédécesseur de TLS (Transport Layer Security),
C'est un protocole de la couche de transport pour crypter et décrypter les paquets de données.

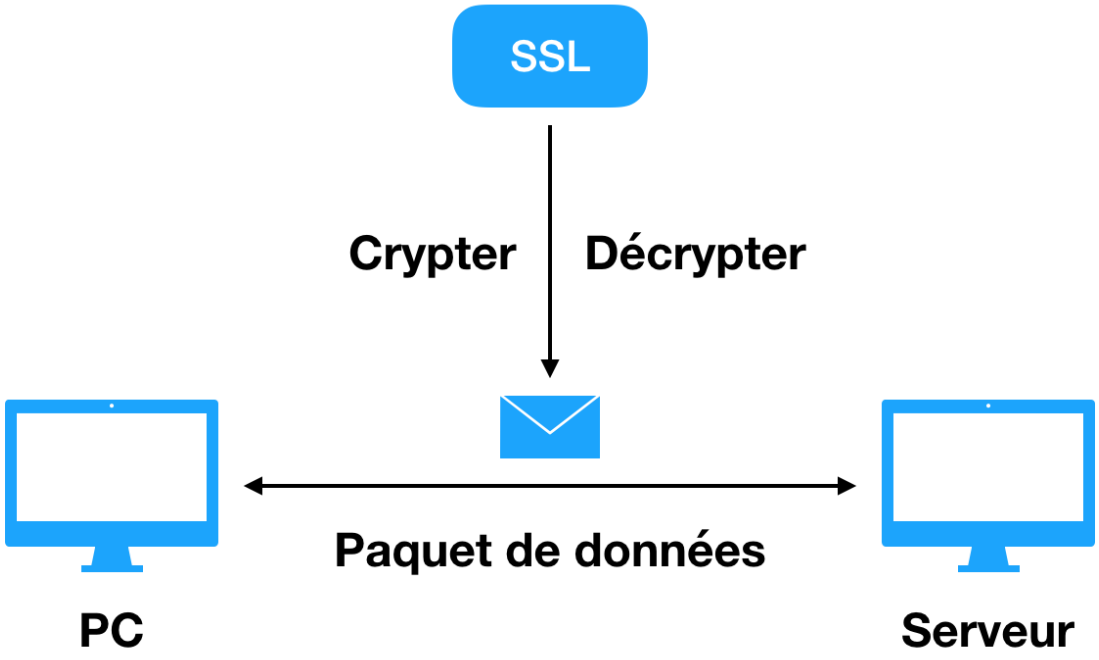

Schéma 1: Rôle de SSL

Par exemple, pour le web, quand le protocole est “http”, les paquets transférés entre le PC et le serveur ne sont pas cryptés, et quand le protocole est “https”, ca vaut dire que tous les paquets transférés sont cryptées.

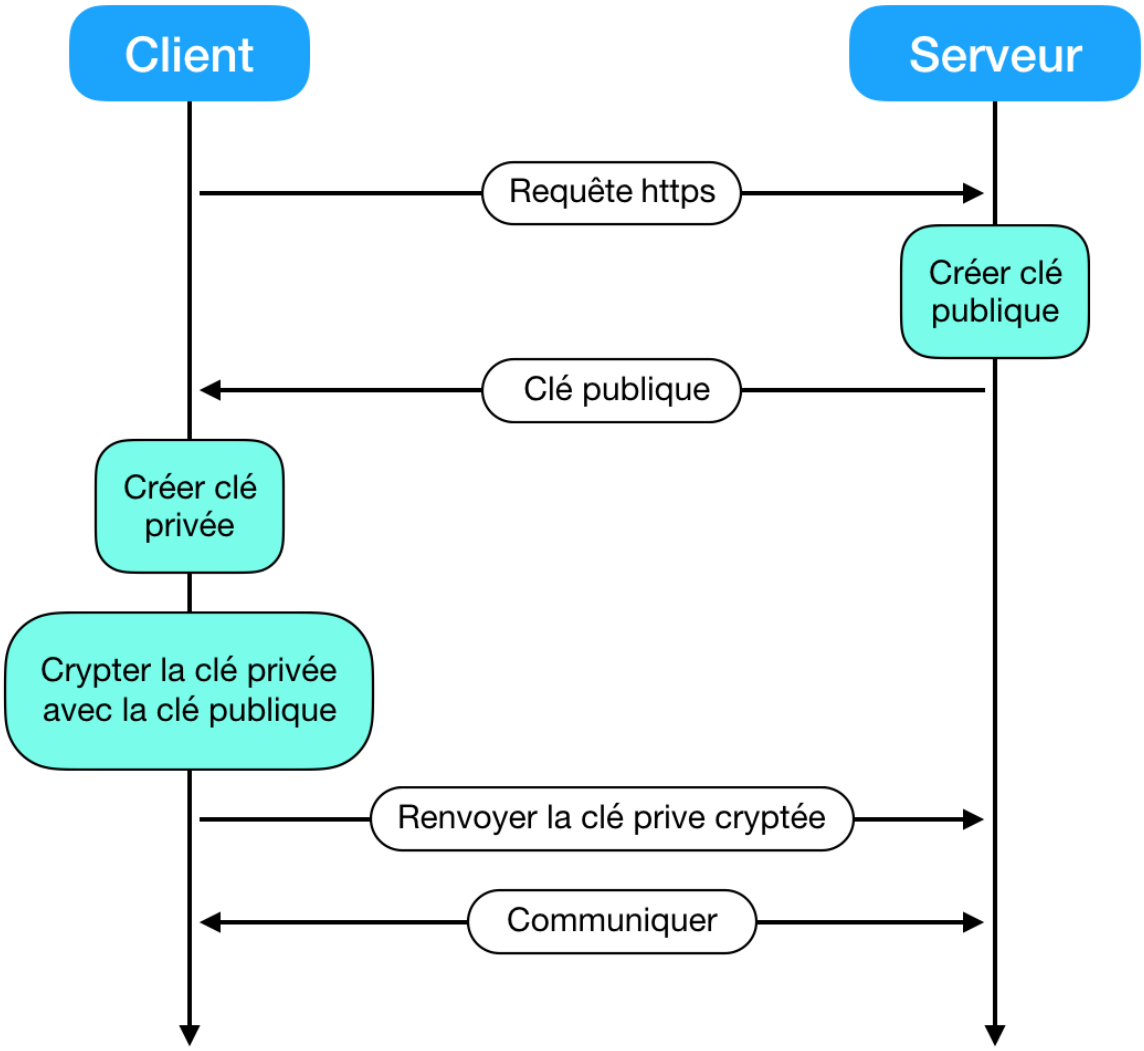

Schéma 2: Processus de création de communication

Le Mécanisme Heartbeat

La réception et l'envoi de données sur le réseau sont réalisés par le Socket du système d'exploitation. C’est-à-dire qu’on doit créer une connection du socket pour envoyer ou recevoir des données et on ne peut pas envoyer ou recevoir des données quand le socket est déconnecté.

Pour maintenir la connexion entre le client et le serveur en vie, le client peut périodiquement envoyer un paquet vers le serveur pour lui teindre informer que le client est encore disponible. Ce paquet peut contenir n'importe quel message sauf les informations confidentielles. Par exemple, le client envoie un signal automatiquement chaque 30 seconds, si le serveur reçoit ce signal, il renvoie une réponse correspondante.

On appelle ce type de mécanisme HeartBeat, Parce que c'est comme notre battement de coeur, très régulier, quand le battement de coeur s'arrête, la communication ne pourra pas continuer.

Lorsque le serveur garde souvent multiple connexion avec de multiples clients, la communication Heartbeat n'est pas cryptée comme les données normalement transferes par SSL.

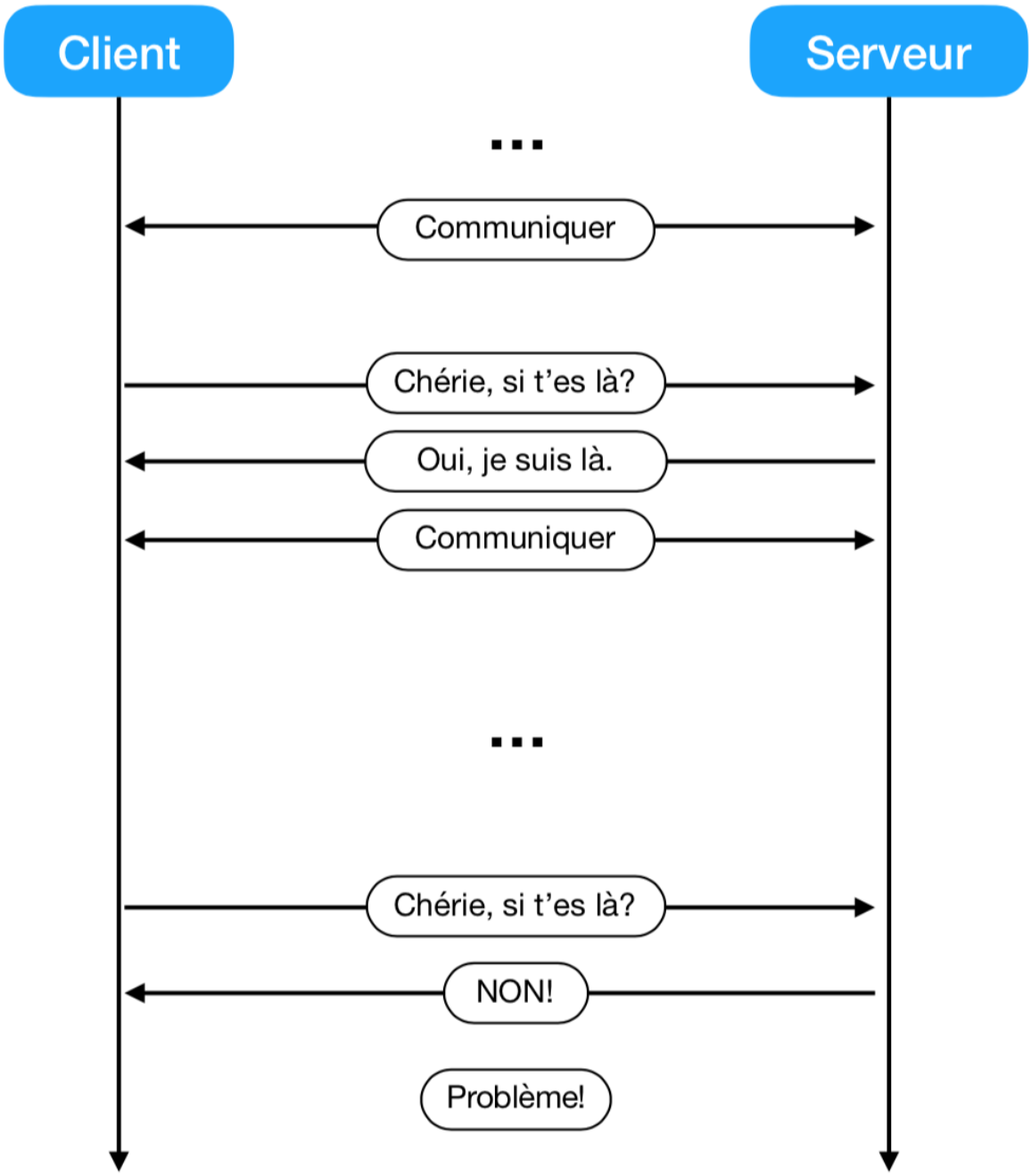

Schéma 3: Le mécanisme HeartBeat

Analyse de bogue

La Réalisation de Heartbeat en OpenSSL

```
typedef struct ssl3_record_st
{
    /*r */  int type; /* type of record */
    /*rw*/  unsigned int length; /* How many bytes available */
    /*r */  unsigned int off; /* read/write offset into 'buf' */
    /*rw*/  unsigned char *data; /* pointer to the record data */
    /*rw*/  unsigned char *input; /* where the decode bytes are */
    /*r */  unsigned char *comp; /* only used with decompression - malloc()ed */
    /*r */  unsigned long epoch; /* epoch number, needed by DTLS1 */
    /*r */  unsigned char seq_num[8]; /* sequence number, needed by DTLS1 */
} SSL3_RECORD;
```

Schéma 4: La structure de données de HeartBeat en OpenSSL [5]

C'est-à-dire que le client envoie chaque fois un message de type en dessus, il y a deux valeurs plus importantes pour cette vulnérabilité, le "length" et le "data". On peut simplifier le processus tel que chaque fois le client envoie le "data" et la longueur du "data" au serveur, quand le serveur reçoit le "data", il retourne le même "data" comme la réponse.

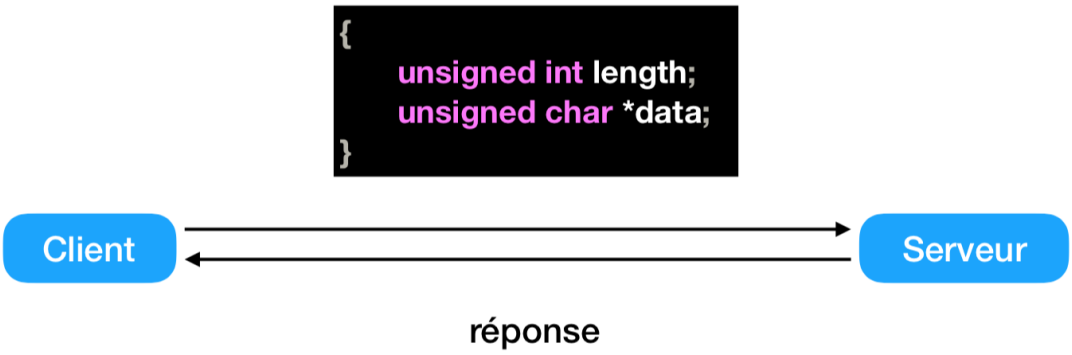

Schéma 5: Schéma simplifié du processus HeartBeat

Le problème de OpenSSL est que le serveur ne vérifie pas le vrai longueur du "data" reçu, par exemple, si le "data" qu'on l'envoie est une chaîne de caractère de 30 bits, et on met le "length" = 128, alors, quand le serveur reçoit le message, il peut trouver la valeur de "length" est 128, il crée donc une réponse de 128 bits, il copie le "data" dans les premiers 30 bits, et les bits restants(de 31 à 128) pourrait contenir des données dans l'espace du mémoire du serveur.

memcpy

Pour fabriquer la réponse, Openssl utilise la fonction `memcpy` sans penser aux longueurs de la donnée a copié.

```
void * memcpy( void * destination, const void * source, size_t size );
```

Cette fonction permet de copier un bloc de mémoire spécifié par le paramètre source, et dont la taille est spécifiée via le paramètre size, dans un nouvel emplacement désigné par le paramètre destination. [6]

Exemple d'Attaque

Quand la longueur passe dépasse la vraie longueur de source, pour fabriquer une réponse de la longueur demandée, le programme continue à copier les données dans la mémoire.

En conséquence, une mauvaise requête peut obtenir un morceau de données dans la mémoire du serveur jusqu'à 64 KB, il peut contient n'importe quelle information, par exemple, les identifiants des utilisateurs, les mots de passe, les cookies, etc.

Schéma 6: Un exemple d'attaque [7]

Réparation de bogue

Un patch d'OpenSSL vers la version 1.0.1g corrigeant le problème a été écrit par Adam Langley et Bodo Moeller.[8]

Il est conseillé de passer dans cette version 1.0.1g. Pour ceux qui n'ont pas la possibilité d'upgrader immédiatement, il faut recompiler OpenSSL avec l'option OPENSSL_NO_HEARTBEATS.

Ce patch rajoute le bound check manquant qui est à l'origine du buffer-overflow ayant permis d'exploiter cette faille.

```
1479  1489                 * payload, plus padding
1480  1490                 */
1481        -              buffer = OPENSSL_malloc(1 + 2 + payload + padding);
      1491  +              buffer = OPENSSL_malloc(write_length);
1482  1492                 bp = buffer;
1483  1493
```

Schéma 7: Le patch

References

1 http://heartbleed.com/ The Heartbleed Bug

2 https://tools.ietf.org/html/rfc6520 Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) Heartbeat Extension

3 https://rt.openssl.org/Ticket/Display.html?user=guest&pass=guest&id=2658 #2658: [PATCH] Add TLS/DTLS Heartbeats

4 https://awe.com/mark/blog/20140409.html OpenSSL Heartbleed Timeline

5 http://blog.existentialize.com/diagnosis-of-the-openssl-heartbleed-bug.html existential type crisis : Diagnosis of the OpenSSL Heartbleed Bug

6 http://koor.fr/C/cstring/memcpy.wp Fonction *memcpy*

7 https://www.nes.fr/fr/securitylab/openssl-heartbleed-vulnerability-cve-2014-0160/ OPENSSL HEARTBLEED VULNERABILITY (CVE-2014-0160)

8 https://github.com/openssl/openssl/commit/96db9023b881d7cd9f379b0c154650d6c108e9a3 Add heartbeat extension bounds check.