HEARTBLEED

Alexandre **GLENAT**,
Lucas **DELANIER**,
Louison **PARANT**,
Mohamed **HASSANI**

L'objectif de ce TP est de vous sensibiliser aux risques de sécurité associés à la faille Heartbleed et de vous permettre de comprendre comment elle peut être exploitée. Vous effectuerez ce TP sous forme de jeu de rôle.

Vous jouerez d'abord le rôle du client. Dans cette partie, vous simulerez, depuis une machine virtuelle, un serveur qui se fera attaquer.

Ensuite, vous changerez de rôle afin de devenir un pirate informatique. Vous attaquerez le serveur d'un autre étudiant afin de récupérer le secret.

Préparation de l'environnement

1. Télécharger **BeeBox** avec l'adresse suivante
https://sourceforge.net/projects/bwapp/files/bee-box/bee-box_v1.6.7z/download

2. Sur **Virtualbox**

a. Cliquez sur **Nouvelle**

b. **Nommez** votre VM

c. Type : **Linux**

d. Version : **Ubuntu 64 bit**

e. Pour choisir le disque dur, utilisez le **disque dur virtuel existant** et sélectionnez **bee-box.vmdk** à partir du fichier zip téléchargé, puis cliquez sur **créer**.

f. Dans les paramètres de la VM, dans **réseau** changer le mode d'accès réseau de **"NAT"** a **"Réseau privé hôte"**.

g. **Démarrer** votre machine virtuelle

Si vous avez des problèmes de curseur souris, cliquez sur **“intégration souris”** en bas à droite de la fenêtre VirtualBox

Maintenant que votre machine virtuelle est démarré, nous allons configurer le service qui simule un serveur PHP ayant une très mauvaise sécurité.

1. Depuis votre **machine virtuelle bee-box**
   a. ouvrez le site http://localhost/bWAPP (Vous pouvez aussi l’ouvrir avec l'icône **“bWAPP - Start”**)
   b. Connectez vous (**utilisateur** : bee, **mot de passe** : bug)
   c. Choisissez le bug **“Heartbleed Vulnerability”** avec un niveau de sécurité **“Low”**, cliquez sur **Hack**
   d. Il faut cliquer sur **“attack script”** afin de télécharger le script **heartbleed.py**

Vous allez a présent simuler l’utilisation normal d’un utilisateur en vous créant un identifiant de connxion.

1. Dans un terminal récupérer **l’adresse ip de la vm**
2. Ouvrez une autre page internet et rentrer **“https://<IP_addresse>:<port>”**
3. (Acceptez les messages de vulnérabilité)
4. Une fois sur le site créer un **nouvel utilisateur /!\ ATTENTION /!** ne mettez pas vos vrai mots de passe ! Vous devrez créer un compte avec un secret qui vous sera donné. Le choix du mot de passe est libre mais gardez à l’idée qu’il sera visible par l’attaquant.

Maintenant vous êtes près pour passer a la l’attaque. Prévenez un membre du staff de Heartbleed pour pouvoir démarrer.

Place a l’attaque!

Ce script est pénalement répréhensible s’il est utilisé hors de ce TP. Nous ne cautionnons pas vos actes si vous l’utilisez à mauvais escient.

En tant **qu’utilisateur**, connectez-vous avec le compte que vous venez de créer puis déconnectez-vous. Cependant, ne fermez pas la page. Cela simule une connexion comme sur l’UCA par exemple.

Maintenant vous allez Passer en **Mode Hacker** (en échangent de place avec votre voisin) Votre but est de vous connecter à son compte et aussi de récupérer son secret et l’écrire au tableau

Pour **effectuer l’attaque**, vous disposer du script **heartbleed.py**

Exécuter le script et rediriger la sortie dans un fichier texte.

le script est fait en python2 et il prend en paramètre une adresse ip ainsi que le port avec le préfixe “-p”

Ouvrez le fichier texte et trouver l’identifiant, le mot de passe ainsi que le mot secret.

Utilisez ensuite **ces information**s pour **vous connecter**.

Vous devrez récupérer le secret de l’un des autres étudiants grâce à l’exploitation de Heartbleed. Une fois tous les secrets récupérés, ils formeront une phrase.

Exercice bonus (Pour les plus futés)

```
struct {
    HeartbeatMessageType type; // 1 byte: request or the response
    uint16 payload_length;   // 2 byte: the length of the payload
    opaque payload[HeartbeatMessage.payload_length];
    opaque padding[padding_length];
} HeartbeatMessage;


/* Allocate memory for the response, size is 1 byte
 * message type, plus 2 bytes payload length, plus
 * payload, plus padding
*/
int dtls1_process_heartbleat(SSL *s){
unsigned int payload;
unsigned int padding = 16; /* Use minimum padding */

// Read from type field first
hbtype = *p++; /* After this instruction, the pointer
               * p will point to the payload_length field */

// Read from the payload_length field from the request packet
n2s(p, payload); /* Function n2s(p, payload) reads 16 bits
                  * from pointer p and store the value
                  * in the INT variable "payload". */

pl = p; // pl points to the beginning of the payload content

if (hbtype == TLS1_HB_REQUEST)
{
    unsigned char *buffer, *bp;
    int r;

    /* Allocate memory for the response, size is 1 byte
     * message type, plus 2 bytes payload length, plus
     * payload, plus padding
     */

    buffer = OPENSSL_malloc(1 + 2 + payload + padding);
    bp = buffer;
```

```
// Enter response type, length and copy payload *bp++ = TLS1_HB_RESPONSE;
 s2n(payload, bp);

 // copy payload
 memcpy(bp, pl, payload); /* pl is the pointer which
 * points to the beginning
 * of the payload content */
 bp += payload;

 // Random padding
 RAND_pseudo_bytes(bp, padding);

 // this function will copy the 3+payload+padding bytes
 // from the buffer and put them into the heartbeat response
 // packet to send back to the request client side.
 OPENSSL_free(buffer);
 r = ssl3_write_bytes(s, TLS1_RT_HEARTBEAT, buffer, 3 + payload + padding);
}
}
```

1. Expliquez la structure d'un message HeartBeat.
2. Quel est le rôle de cette fonction?
3. Identifiez et Justifiez, le(s) partie(s) de code responsable de la vulnérabilité.
4. Proposez des corrections de code pour mettre un terme aux attaques heartBleed.
5. Quelles sont les démarches/solutions que les développeurs chez OpenSSL, auraient pu mettre en place pour découvrir cette faille avant sa publication?