Charles Olivier-Anclin
charles.olivier-anclin@uca.fr
perso.isima.fr/~chaolivi

Clermont Auvergne INP, Université Clermont Auvergne

Cours de Sécurité Web ZZ2F5

Heartbleed *: faille d'implémentation d'OpenSSL*

La cible de la démonstration

- **Ubuntu Server 12.04.4** : embarque **OpenSSL 1.0.1** (la version vulnérable)
- installation de `nginx` pour servir du TLS

```
sudo apt-get install nginx
```

- surtout pas d'upgrade pour **conserver** des services vulnérables ;)



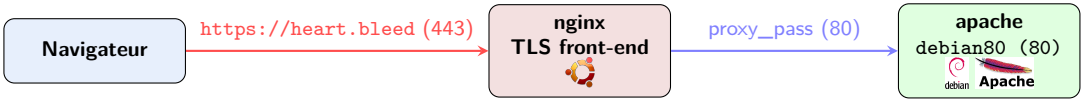
Recréer l'environnement (VirtualBox)

- le **paramétrage réseau** suit la démarche "recréer l'environnement de cours dans VirtualBox"
- machine joignable sous `heart.bleed` (172.16.76.143), backend Debian en 172.16.76.144:80

Architecture typique

nginx : un *reverse proxy*

- **nginx** route le trafic `http` (80 / 443) vers d'autres serveurs web
- dans notre cas : une **Debian** avec **apache** en backend



Avantages collatéraux

appli `node` sur port exotique, **filtrage** de requêtes, **load balancing** "pas cher", et **migrations** facile.

Heart.bleed : 80 → 443, puis proxy TLS

```
upstream debian80 { server 172.16.76.144:80; }

server {
     listen 80;
     server_name heart.bleed;
     rewrite ^ https://heart.bleed$request_uri? permanent;
}
server {
     listen 443;
     server_name heart.bleed;
     ssl_certificate     /etc/nginx/ssl/keys/heartbleed.crt;
     ssl_certificate_key /etc/nginx/ssl/keys/heartbleed.key;
     location / { proxy_pass http://debian80/; }
}
```

- le **front** parle `https` (443) avec le client
- le **backend** reçoit du `http` (80) via `proxy_pass`

Heartbleed

- découverte en **mars 2014** par Google & Codenomicon
- présente depuis **mars 2012**
  - introduite à la suite d'un *"bug fix"* par un **contributeur bénévole**, validé le 31 décembre 2011
  - visible dans **OpenSSL 1.0.1** (2012)
- **faille d'implémentation** dans OpenSSL, extension **Heartbeat**

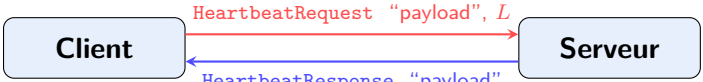


CVE-2014-0160

Un client malveillant demande au serveur "*renvoie-moi ce que je t'ai envoyé*" en mentant sur la **longueur** → le serveur recopie **plus** de mémoire qu'il ne devrait, y compris des **secrets**.

Principe de l'extension Heartbeat (RFC 6520)

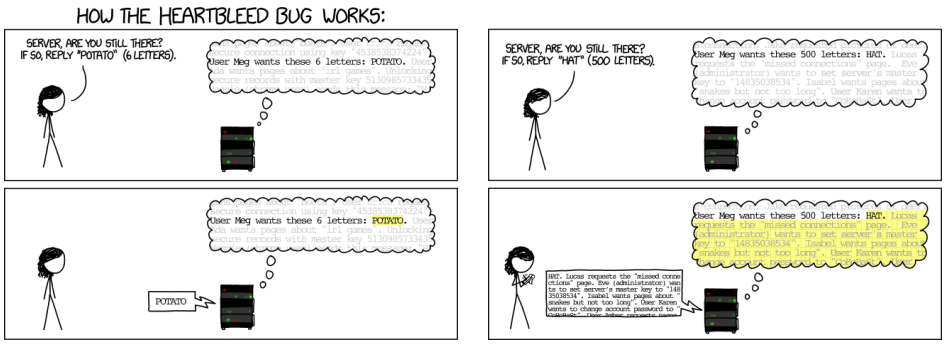
Garder la connexion vivante



- le demandeur joint un **payload** et sa **longueur** L
- le répondeur doit renvoyer **exactement** ces L octets
- but : vérifier que le pair et la connexion sont toujours **en vie**

Heartbleed : débordement de lecture

Recopie de L octets *sans* vérification



- le serveur copie L octets **sans vérifier** la longueur réelle
- $L >$ tampon reçu $\Rightarrow$ **over-read**

Exfiltration de la **mémoire** du serveur : cookies, sessions, **clés privées TLS**.

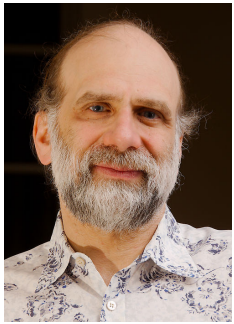
Impact d'Heartbleed

Évaluation

- **Bruce Schneier** la positionne à **11 sur une échelle de 1 à 10**
- **qui savait / qui a pu exploiter** ?
  - la **NSA** est soupçonnée par **Bloomberg** de l'avoir exploitée pendant **2 ans**

La faille est invisible

- **aucune trace** dans les logs traditionnels de l'exploitation en cours



Bruce Schneier — *security maven*

Impact : tout ce qui utilise SSL/TLS

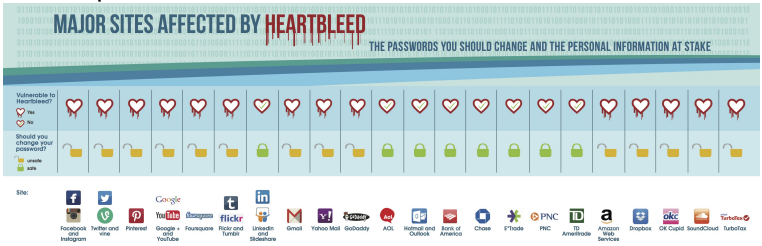
Périmètre : toute la pile TLS

- **https** : websites (nginx, apache, IIS, ...)
- **mail** : `imaps`, `pop3s`, `smtps`
- `ftps`, `vpn`
- des milliers de **services** et appliance embarqués

Concrètement

lire la **mémoire** du serveur vulnérable :

- contenu de la mémoire : cookies, sessions, identifiants, applicatif
- les **clés privées** TLS sont dans la même mémoire → **déchiffrement** du trafic chiffré



Impact : beaucoup d'équipements

Le maillon faible : l'embarqué

- **routeurs**, **box** internet / TV, NAS
- équipements d'entreprise : reverse proxies, load balancers, contrôleurs wifi, VPN boxes
- objets connectés : tout ce qui parle TLS avec OpenSSL 1.0.1

La fenêtre d'exposition dépend du **patch** (*upgrade*) de **chaque éditeur** — parfois des mois.



routeur *Heartbleed*-vulnérable

ASAP

1. **mettre à jour** OpenSSL (+ redémarrer les services)
2. **révoquer** et **changer** tous les certificats (clés privées potentiellement lues)
3. **changer tous les mots de passe**

Touche beaucoup de services web

- **HTTPS** mais aussi `smtps`, `vpn`, ...
- Windows + OpenSSH (qui embarque OpenSSL) ne sera pas épargné à la prochaine vague d'exposition

De l'introduction à la divulgation



CVE et patchs

- **CVE-2014-0160** (et CVE-2014-0346)
- la **vitesse de distribution des patchs** est cruciale : temps de **compilation** et de diffusion dans les **paquets**

Section 4 : **Détection et exploitation**

shodan & zmap



- **shodan** : `port:443 openssl 1.0.1` (Heartbleed Early Shodan Search)
- puis : `vuln:cve-2014-0160` (Heartbleed CVE Search)
- `shodan` report, et **Heartbleed Bug Health Report** (`zmap.io/heartbleed`)

Principe

Énumérer les machines **vulnérables** (version d'OpenSSL annoncée / fingerprint) puis vérifier la **CVE** en interrogeant le comportement réel.

Détection ciblée : nmap

Script `ssl-heartbleed`

```
$ nmap -p 443 --script=ssl-heartbleed heart.bleed
Starting Nmap (https://nmap.org) at 2015-10-20 12:10 EDT
Nmap scan report for heart.bleed (172.16.76.143)
Host is up (0.0014s latency).
rDNS record for 172.16.76.143: proxy
PORT    STATE SERVICE
443/tcp open  https
| ssl-heartbleed:
|   VULNERABLE:
|   The Heartbleed Bug is a serious vulnerability
|   in the popular OpenSSL cryptographic software library.
```

- le script envoie une `HeartbeatRequest` maligne et détecte s'il reçoit **plus** de données que le payload
- résultat : la machine est `VULNERABLE`

PoC d'Andrew Kennedy

- script `hb-test.py` (gist `akenn/10159084`), détaillé sur le blog d'**Andrew Kennedy** (Heartbleed)
- **aucun prérequis** côté serveur : on parle juste TLS + Heartbeat

Plateforme de test : `https://heart.bleed`

- **nginx** renvoie sur un formulaire d'**authentification** servi par **apache** en forçant `https`
- … authentifions-nous ! (les identifiants vont transiter par la mémoire TLS du serveur)

Dump de la mémoire du serveur

```
$ python2 hb-test.py heart.bleed > hb.dump \
  && cat hb.dump | tail -n +10 | head -n 40
00e0: 63 65 70 74 2D 4C 61 6E 67 75 61 67  ..Accept-Languag
00f0: 6E 2D 55 53 2C 65 6E 3B 71 3D 30 2E  e: en-US,en;q=0.
0150: 68 65 61 72 74 62 6C 65 65 64 3D 27  ie: heartbleed=
0160: 72 69 65 6E 64 73 21 27 0D 0A 43 6F  Hi friends!.use
01d0: 65 3D 6D 61 7A 65 6E 6F 76 69 26 70  rname=chaolivi&p
01e0: 6F 72 64 3D 6C 65 61 6B 65 64 70 61  assword=leakedpa
01f0: 48 95 77 37 7B E7 02 27 4D 8B 3C 2D  ss't.H.w7{..'M.<-
```

Observations

- lecture de **cookies** et des identifiants `username=chaolivi & password=leakedpass`
- aucun signe de l'attaque dans les logs

Ça marche aussi avec les **clés privées** et tous les secrets de la **mémoire TLS**.