Intrusions logicielles

ISI

2 - Intrusions

Résumé du document

Ce document traite des principales attaques logicielles, dont Heartbleed et Log4Shell, ainsi que des vulnérabilités courantes comme les débordements de mémoire tampon. Il explore les méthodes d'exploitation, les conséquences possibles, et les stratégies de protection, telles que la randomisation des adresses mémoire et l'utilisation de bibliothèques sécurisées.

Table des matières

1. Attaques logicielles connues 2

1.1. Heartbleed 2

1.2. Log4Shell / Log4j 2

2. Memory overflow 4

2.1. Buffer overflow 4

2.1.1. Exemple de code vulnerable 4

2.2. Shellcode 5

3. Manipulation de la mémoire 6

3.1. Protection contre la manipulation 6

1. Attaques logicielles connues

1.1. Heartbleed

- 66% des sites Web touchés !
  - Apache (52% des sites Web actifs au 04.2014)
  - nginx (14% des sites Web actifs au 04.2014)

Exploitation d'une vulnérabilité dans la librairie OpenSSL. Cette librairire était utilisée dans de nombreaux serveurs web.

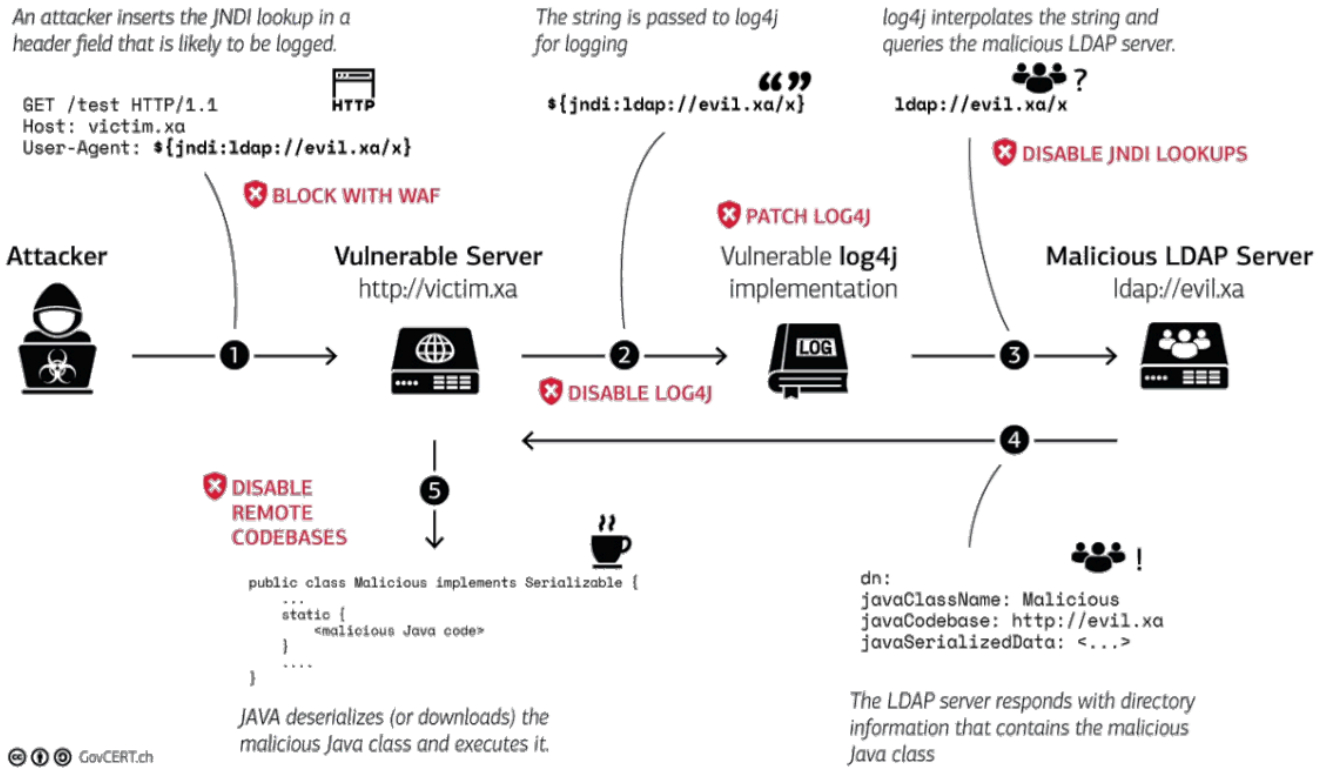
- Sites Web sociaux
- Sites Web professionnel (commercial ou HEIG-VD)
- Sites Web commerciaux
- Sites Web de hobby, forums,
- Sites Web distribuant des logiciels,
- Sites Web du gouvernement

Ce n'est pas une vulnéralibité dans le protocole SSL/TLS, mais dans l'implémentation de la librairie OpenSSL (donc d'une erreur de programation).

- Sans utiliser d'information privilégiée ou d'accès, ils ont réussi à voler :
  - des clés cryptographique,
  - des clés privées liés aux certificats X.509,
  - des noms d'utilisateurs,
  - des mots de passe,
  - des messages instantanés,
  - des emails,
  - des documents et des communications sensibles.

1.2. Log4Shell / Log4j

- Bibliothèque de journalisation (logs) pour Java
- « La » principale librairie utilisée dans les applications en Java
- Utilisée dans des milliers de programmes courants



- La charge utile peut être placée :

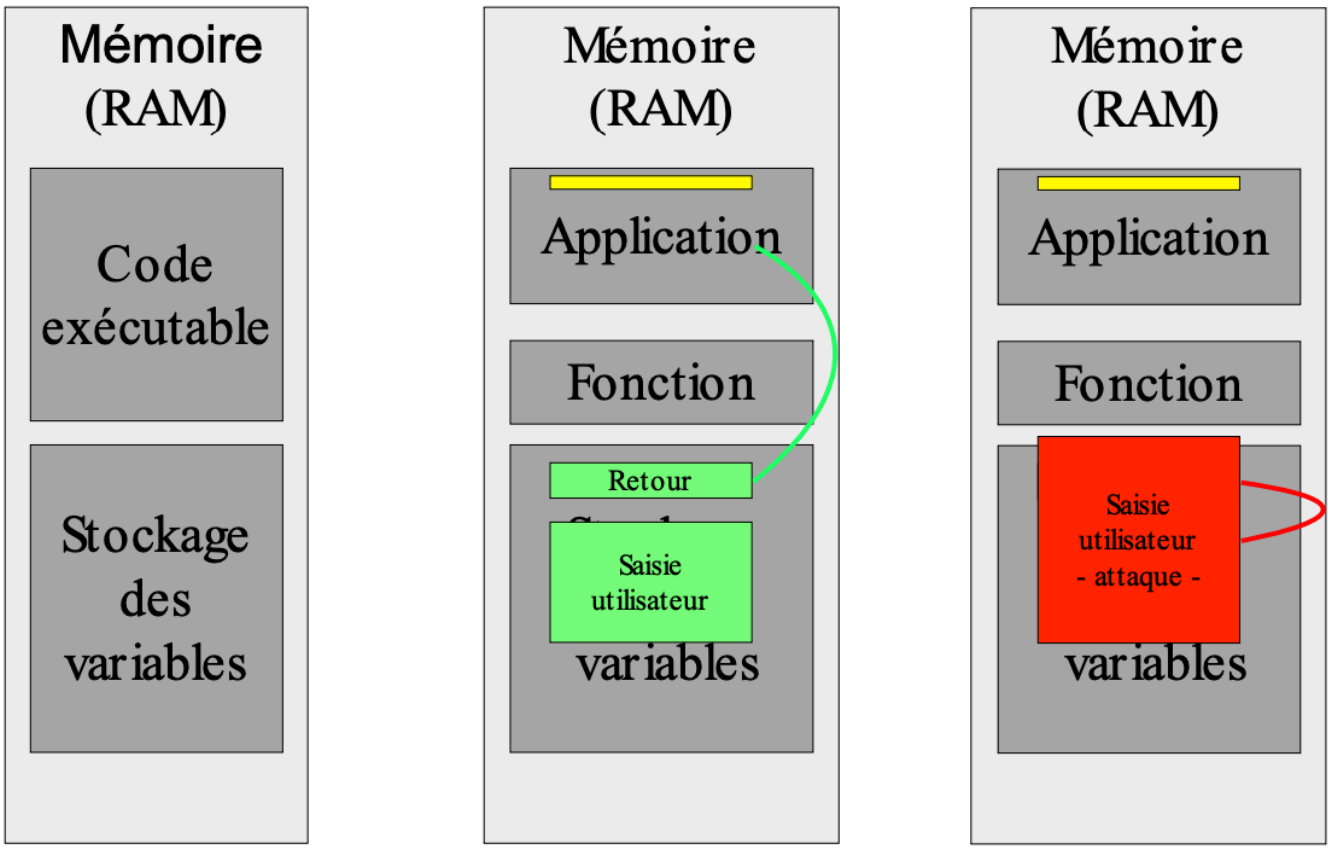
- Champs Web : en-tête, identifiant, mot de passe, etc.
- Fichiers robot.txt ou security.txt (serveur de l’attaquant)
- Enregistrement DNS “TXT” (de l’attaquant)
- Champs d’emails : en-têtes, adresse source, etc.
- Champs des certificats SSL/TLS
- Métadonnées de fichiers : images, PDF, Word, Excels
- Noms du réseau Wifi (iPhone, Instagram envoie le BSSID à l’éditeur) ou de son appareil Bluetooth
- Et d’autres

2. Memory overflow

- Ecriture/lecture/exécution sans autorisation
  - typiquement en exploitant un « buffer overflow »
- Interprétation des entiers
  - typiquement un grand nombre devient négatif, ou vice versa
- Interprétation des chaînes de caractères
  - typiquement en supprimant la fin d'une chaîne de caractères

2.1. Buffer overflow

- Principe
  - un espace est réservé pour stocker une entrée contrôlable par celui qui
  - le pirate arrive à passer une donnée qui fait écrire le programme au-delà de l'espace réservé
- En français
  - dépassement de tampon
  - débordement de tampon



2.1.1. Exemple de code vulnerable

```
void myFunction(char *str) {
  char BufferB[16];
  strcpy(bufferB, str);
}
void main() {
  char bufferA[256];
  myFunction(bufferA);
}
```

- Pas de contrôle si bufferB est assez grand
- Taille `bufferA` : 256
- Taille `bufferB` : 16
- Copier de bufferA dans bufferB (256o dans 16o)
  - Ecrasement de la mémoire

2.2. Shellcode

- Un « shellcode » est une suite d'instructions destinées à être injectées, puis exécutées par un programme exploité.
- Un « shellcode » peut notamment être utilisé pour exploiter un « buffer overflow ».
- Le plus important lors de la création
  - aussi petit que possible
  - exécutable (dans le context d'exécution du processus injecté, en particulier peu importe la localization en mémoire)
- Peut permettre toutes sortes de choses :
  - L'ouverture d'un accès au système cible
  - Faire naître un /bin/sh ou cmd.exe
  - Changer des droits de fichiers
  - Ajouter un utilisateur
  - Ouvrir un port
  - et beaucoup d'autres…

3. Manipulation de la mémoire

3.1. Protection contre la manipulation

- Stack/heap non-exécutable
- Utilisation de canaris
- Randomisation des adresses mémoire (ASLR)
- Librairies sécurisées
  - Libsafe
  - utiliser strncpy au lieu de strcpy
  - utiliser snprintf au lieu de sprintf
  - utiliser fgets(stdin, str, 10) au lieu de gets(str)
  - utiliser scanf(“%10s”, str) au lieu de scanf(“%s”,str)
  - et bien d’autres…
- Autres contre-mesures possibles