

TD Feuille 6 — Failles de sécurité

Ce TD propose des extraits de code comportant des failles. Certaines sont explicites, d'autres appellent plutôt à la malice d'un attaquant. N'hésitez pas à regarder durant la séance la documentation des différents langages et fonctions pour trouver des indices et répondre aux questions. Seul le premier exercice porte sur les vulnérabilités spécifiques au C, mais nous aurons l'occasion d'explorer davantage celles-ci en TP. Les autres exercices portent sur des vulnérabilités qu'on retrouve en fait dans de nombreux langages (y compris en C).

Exercice 1 (Sémantique de C et vulnérabilités aux comportements non définis (UB)). On rappelle le lien sur le site du cert : <https://wiki.sei.cmu.edu/confluence/> Et la liste des comportements non définis en C : <https://wiki.sei.cmu.edu/confluence/display/c/CC.+Undefined+Behavior>

On considère la fonction C suivante qui alloue un buffer de taille maximale capacity, en initialisant ses size premiers indices à la valeur v. Elle retourne NULL en cas d'erreur (ou d'absence de place mémoire pour la capacity demandée).

```
1 double *bufInit(int capacity, int size, double v) {
2   if (size > capacity) return NULL;
3   double *dst = (double *) malloc(sizeof(double)*capacity);
4   if (dst == NULL) return NULL;
5   for (unsigned int i=0; i<size; i++) { dst[i]=v; }
6   return dst;
7 }
```

▸ **Question 1.** Expliquer en quoi ce programme a des vulnérabilités que par exemple un attaquant en "dénier de service" pourrait exploiter - en interagissant avec le logiciel de manière à le faire planter. Lister tous les comportements indéfinis possibles.

▸ **Question 2.** Comment pourrait-on localiser ces vulnérabilités avec le plugin EVA de FRAMA-C?

▸ **Question 3.** On déclare maintenant size et capacity en unsigned int. Est-ce que ça résout les problèmes? Justifier votre réponse.

▸ **Question 4.** Écrire une variante non vulnérable de ce programme dans laquelle capacity et size sont du type size_t (type entier non signé de la taille des adresses mémoires qui est le type de l'argument du malloc). Le but est ainsi d'avoir une fonction qui s'adapte aux tailles d'adresse de la machine cible. On pourra consulter : <https://wiki.sei.cmu.edu/confluence/display/c/INT30-C.+Ensure+that+unsigned+integer+operations+do+not+wrap>

Est-ce qu'EVA de FRAMA-C peut valider ce code?

Exercice 2 (un classique : l'injection de commande à exécuter). Soit le programme PHP suivant qui vise à lister le contenu du répertoire d'un utilisateur :

```
1 $userName = $_POST["user"];
2 $command = 'ls -l /home/' . $userName;
3 system($command);
```

- **Question 1.** Expliquer les faiblesses de ce programme et les corriger.

Exercice 3 (un autre classique : l'injection SQL et ses déclinaisons). On considère un site permettant à un utilisateur de se connecter à son compte. Sur la page web, une connexion en PHP à une base de données permet d'authentifier l'utilisateur.

```

1 // on considere $login et $motdepasse recuperes dans le formulaire de connexion
2 $connexion = mysqli_connect($SQLServerName, $sqllogin, $sqlpass, $sqldbname);
3 // connexion a la BD sql
4 $requete = "select name from users where
5 username = '". $login. "'
6 AND password = '". $motdepasse. "'";
7 // requete a la base de donnee
8 $result = mysqli_query($connexion, $requete);
9 // on propose la page de compte de l'utilisateur

```

- **Question 1.** Analyser le code et expliquer les failles potentielles. Proposer une correction.

- **Question 2.** On s'intéresse au formulaire en lui-même

```
<input type="text" name="nom" value="du texte">
```

Quel problème peut on exploiter dans ce formulaire ?

- **Question 3.** Peut-on le résoudre en échappant les <, > ?
- **Question 4.** Si on échappe des mots clés, résolvons nous le problème ?

Exercice 4 (Détournement de champ par un attaquant malin). Nous nous intéressons à la fonction mail de PHP pour les versions d'avant 5.4.42 et 5.5.27. La fonction se présente comme suit

```

1 mail(
2     string $to,
3     string $subject,
4     string $message,
5     array|string $additional_headers = [],
6     string $additional_params = ""
7 )

```

Et on peut envoyer un email en utilisant un serveur SMTP comme suit

```

1 <?php
2 $message = "aaa";
3 mail('mail@mail.org', 'Sujet', $message, $headers);
4 ?>

```

- **Question 1.** Le champ additional_headers n'échappe ni les sauts de lignes, ni le contenu contrairement aux autres champs. Expliquer les problèmes que cela peut poser et comment les corriger.

Exercice 5 (Moindre privilège). On s'intéresse ici à des programmes permettant de créer des répertoires. On utilisera la fonction `mkdir` spécifiée de la manière suivante : `os.mkdir(path[, mode])` : Create a directory named `path` with numeric `mode`. The default mode is `0777` (octal). If the directory already exists, `OSError` is raised.

▸ **Question 1.** Lister les points délicats liés à ce type d'opérations.

▸ **Question 2.** Soit le programme Python suivant. Expliquer ce qu'il fait. La règle de sécurité qui préconise de donner des droits uniquement dans la partie du code où c'est nécessaire peut ici être potentiellement violée.

```

1 def makeNewUserDir(username):
2     if invalidUsername(username):
3         #avoid CWE-22 and CWE-78
4         print('Usernames cannot contain invalid characters')
5         return False
6     try:
7         raisePrivileges()
8         os.mkdir('/home/' + username)
9         lowerPrivileges()
10    except OSError:
11        print('Unable to create new user directory for user:' + username)
12        return False
13    return True

```

▸ **Question 3.** Proposer une correction.

▸ **Question 4.** Soit le programme php suivant. Expliquer quel est le problème potentiel de ce programme et le corriger.

```

1 function createUserDir($username){
2     $path = '/home/' . $username;
3     if(!mkdir($path)){ return false;}
4     if(!chown($path,$username)){rmdir($path); return false;}
5     return true;
6 }

```

Exercice 6 (Encore une injection de commande). Soit le programme Javascript `curl.js` suivant qui va récupérer une liste de pages web depuis un fichier.

```

1 // [...]
2 const exec = util.promisify(require('child_process').exec);
3
4 async function main() {
5     const input = ... // on y met le contenu du fichier en argument
6
7     for await (const line of input) { // on itere sur les lignes lues
8         if (line.trim().length > 0) {
9             const { stdout, stderr } = await exec('curl ${line}');
10            console.log({ stdout, stderr });

```

```
11     }  
12   }  
13 }  
14  
15 main()
```

On utilise ainsi : `node curl.js < liste_url.txt` où

```
1 $ cat liste_url.txt  
2 https://www-verimag.imag.fr/  
3 https://www.univ-grenoble-alpes.fr/  
4 https://www.grenoble-inp.fr/
```

- **Question 1.** Proposez une entrée utilisateur malveillante et expliquez la vulnérabilité.
- **Question 2.** Proposez une correction pour ce programme.