École Polytechnique de Montréal

Département de Génie Informatique et Génie Logiciel

INF4420a - Sécurité Informatique

Travail Pratique 2

Automne 2022

Table des matières

1 Directives

Tous les travaux devront être remis avant 23h55 le jour de la remise sur le site Moodle du cours. À moins que cela ne soit explicitement demandé dans le sujet, vous ne devez remettre qu'un fichier PDF nommé selon le format TPX-matricule1-matricule2.pdf. Vous pouvez inclure des annexes dans votre rapport si vous jugez que cela améliore la lisibilité (code source, ...)

- Voir la date de remise du rapport de ce laboratoire dans le plan du cours.
- Le travail devra être fait par équipe de deux. Toute exception (travail individuel, équipe de trois) devra être approuvée au préalable par le professeur.
- L'orthographe et la forme seront prises en compte pour chaque question.
- Indiquez toutes vos sources d'information, qu'elles soient humaines ou documentaires.

NOTE : POUR TOUTES LES QUESTIONS, VOUS DEVEZ MONTRER COMMENT VOUS AVEZ OBTENU LES RÉPONSES, INCLUANT DANS VOTRE RAPPORT LES CAPTURES D'ÉCRAN MONTRANT LES COMMANDES UTILISÉES ET LEUR SORTIE.

2 Scénario

Vous êtes un pirate informatique et vous avez réussi à pénétrer dans un local de Omnisoft, une compagnie lavalloise qui se spécialise dans le domaine du multimédia interactif. Ceci est une prise de taille, car si vous mettez la main sur le code source de leur tout dernier jeu, vous pourriez gagner gros en vendant des copies pirates avant la date de lancement. Le local contient une machine sans aucune protection physique.

3 Question 1 - Accès physique = Game Over [/1]

Dans cet exercice vous allez voir certaines techniques permettant d'obtenir les droits administrateur sur une machine à laquelle vous avez accès physiquement. La machine est disponible dans votre repertoire /home/INF4420a/A2022/TP2.

3.1 Phase de reconnaissance

1. [/0.2] Démarrer la machine virtuelle (VM) et essayer de vous connecter à une session. Que constatez-vous ?
2. [/0.2] Redémarrez la VM et au démarrage appuyez sur F2 pour rentrer dans le BIOS. Que se passe-t-il ?
3. Appuyez sur Echap pour continuer le boot de la machine. L'écran de GRUB présente les différentes options de boot pour la machine. Dans notre cas, il n'y a qu'une seule ligne, qui correspond au système Ubuntu. Habituellement il est possible d'éditer la ligne de commande correspondante en appuyant sur la touche e.
4. [/0.3] Est-ce possible dans notre cas ? Sinon, pourquoi ?

3.2 Réalisation de l'attaque

1. Authentifiez-vous et accédez à GRUB
   utilisateur : Poly ; mot de passe : BigPassword
2. [/0.3] A l'écran de GRUB appuyez sur e pour éditer la commande. Sélectionnez la ligne commençant par :

   ```
   linux   /boot/vlinuz-generic root=UUID=f0861cc3-3d4d-44ed-8ebc-5e04f857c41a ro ...
   ```

   supprimez la suite de ligne à partir de ro et remplacez la par :

   ```
   rw init=/bin/bash
   ```

   puis appuyez sur Ctrl+x. Votre système se lance sur un fenêtre avec un shell root confirmer que le root a les accès en lecture et en écriture sur le système de fichier.

   ```
   # mount | grep -w /
   ```

   Puis utilisez la commande passwd pour réinitialiser le mot de passe de root. Redémarrez la machine et ouvrez une session avec l'utilisateur root.

4 Question 2 - Exploitation des vulnérabilité [/2.5]

Dans cet exercice, vous aurez la possibilité de vous familiariser avec le Framework Metasploit[2] d'exploitation de vulnérabilité.
Démarrez la machine inf4420a

4.1 Phase de reconnaissance

1. Avec le compte root que vous avez acquis précédemment, affichez l'adresse IP de la machine inf4420a.
2. Sur votre machine Kali, assignez une adresse IP pour que les machines (kali et inf4420a) soient dans le même sous-réseau.
3. Avec la commande ping envoyez deux paquets seulement pour vérifier la connectivité.
4. [/0.3] À quoi sert Nmap ?
5. [/0.3] Utilisez nmap[1] pour scanner la machine inf4420a. Vous avez à identifier les services et les système d'exploitation. Expliquez les options que vous avez utilisées lors de votre scan.

4.2 Réalisation de l'attaque

1. [/0.2] Connectez-vous sur le service ftp en mode anonyme, listez les fichier disponible et récupérez le fichier secret.txt.
2. [/0.3] Comment empêcher la communication de manière anonyme ? Donnez votre réponse en fonction du scénario actuel.
3. [/0.4] Pourquoi le protocole ftp n'est pas pas un bon moyen pour un accès à distance et quelle serait une alternative plus sûr ?
4. [/0.2] Avec les informations recueillies dans la question de nmap précédente, identifiez le programme vulnérable et sa version.

5. Lancez metasploit avec la commande msfconsole
6. [/0.1] Utilisez l'exploit /exploit/ftp/vsftpd_234_backdoor avec

```
#  use /exploit/unix/ftp/vsftpd_234_backdoor
```

7. Affichez les options de l'exploit avec la commande options
8. [/0.2] Quels sont le(s) paramètre(s) à modifier ? Modifiez-le(s) et lancez l'exploit
9. [/0.2] Grâce à l'exploit précédent, ajoutez un utilisateur "h4x0r" et créer un répertoire "owned" sur le répertoire /home/inf4420a
10. [/0.3] Comment corriger cette vulnérabilité ?

5 Question 3 - Vulnérabilités WEB [/4.5]

5.1 Mise en marche

Vous avez une application web composée de deux Docker inf4420a-app et inf4420a-db qui représentent respectivement la partie applicative et la partie base de données. Vous aurez à manipuler l'application à travers un menu qui contient les deux vulnérabilités à tester.

1. Connectez-vous avec le compte root sur la VM inf4420a
2. Lancez le docker de la base de données avec la commande

```
# docker run -d -p 3306:3306 inf4420a-db
```

3. Lancez le docker de l'application web avec la commande

```
# docker run -d -p 3000:3000 inf4420a-app
```

4. Accédez à l'adressse de votre vm inf4420a avec votre navigateur pour confirmer le bon fonctionnement http ://@ip inf4420a :3000. Testez le menu.
5. [/0.3] Refaites le scan de port avec nmap et reportez les nouveaux services observés.
6. Lancez Burp[3] sur votre machine kali
7. Configurez le proxy de votre navigateur pour passer à travers Burp.
8. Reconnectez-vous sur l'application web et observez les changements dans Burp. Désactivez le mode intercept.

5.2 Vulnérabilité XSS [4]

1. Allez à la page XSS
2. Réactivez le mode intercept sur Burp
3. [/0.1] Sur la page des produits, ajoutez un nouveau produit.
4. Observez la requête sur Burp, et passez là au serveur
5. [/0.1] Ajoutez un nouveau produit, et modifiez la catégorie pour qu'elle correspond à "Hacked" sur Burp.
6. Désactivez le mode intercept sur Burp
7. [/0.1] Ajoutez un nouveau produit et précisez dans le nom du produit

```
<script>alert("xss sur le nom du produit")</script>
```

8. [/0.2] Quel est le type de cette XSS ?
9. [/0.4] Qu'en est il pour les autres champs ? Sont-ils vulnérables ? Voir les deux listings 1 & 2
10. [/0.3] Utilisez l'attaque XSS pour afficher les cookies (il se peut qu'il n'y en ait pas)
11. [/0.4] Comment corriger cette vulnérabilité et à quel niveau (Frontend or Backend) ? Justifiez votre réponse.

```
app.post('/add', (req, res) => {
   if(req.body.name === undefined || req.body.name === ''
    || req.body.cat === undefined  || req.body.cat === ''
    || req.body.fournisseur === undefined  || req.body.fournissuer === ''
    ||  req.body.prix === undefined || req.body.prix === ''  ){
        res.render('pages/add_produit',{error:"Une erreur s'est produite !"})
    } else {
        alpha=req.body.name
        var sql ="INSERT INTO produit SET ?"
        let post = {
            name:req.body.name,
            categorie:req.body.cat,
            fournisseur:req.body.fournisseur,
            prix:req.body.prix
        }
        connection.query(sql, post, (err, result) => {
            if(err) throw err;
            res.redirect('/add')
        });
    }
})
```

Listing 1 – Code pour l'ajout d'un nouveau produit

```
  <tbody>
            <%
            for (var i=0; i <result.length; i++) { %>
                   <tr>
                        <td><%-result[i].id%></td>
                        <td><%-result[i].name%></td>
                        <td><%-result[i].categorie%></td>
                        <td><%-result[i].fournisseur%></td>
                        <td><%-result[i].prix%></td>
                    </tr>
                    <% } %>

            </tbody>
```

Listing 2 – Sorite du tableau

5.3 Vulnérabilité d'injection SQL [5]

1. Allez à la Page SQLi
2. Réactivez le mode intercept sur Burp
3. [/0.1] Recherchez le produit avec l'id 24, observez la requête sur Burp, et passez là au serveur. Désactivez le mode intercept sur Burp.

4. [/0.2] Introduisez le caractère ' sur le champ id. À quoi correspond le message et que permet-il d'identifier ?
5. [/0.2] Utilisez le champ de recherche et introduisez :

```
24 Order by [num]
```

, num varie de 1 à 10. Quelle information peut-on conclure sur la table produit ?

6. [/0.3] Utiliser le code suivant à la place du champ de recherche,

```
-1 Union select 1,2,3,4,5
```

Pourquoi avons-nous choisi les options -1 et les cinq chiffres après le select ?

7. [/0.1] Utilisez le texte suivant à la place du champ de recherche :

```
-1 Union select database(),2,3,4,5
```

Quel est le nom de la base de données ?

8. [/0.3] Changez le texte précédent pour identifier l'utilisateur de la base de données. Que pouvez vous conclure ?
9. [/0.8] En utilisant information schema de Mysql identifiez la deuxième table de la base de données inf4420a, et récupérez son contenu manuellement.
10. [/0.4] Utilisez sqlmap[7] pour faire la question précédente.
11. [/0.4] Le listing 3 reprends le code utilisé au niveau de l'application. Comment-peut on l'améliorer pour corriger la vulnérabilité sql ?

```
app.post('/search', (req, res) => {
    id=req.body.id
    var sql ="SELECT * FROM produit WHERE id="+id
    connection.query(sql,(err,result)=>{
            if (err) {
                res.render('pages/search_produit',{message:err.message,
                                                   sql:err.sql})
                    }
            res.render('pages/search_produit',{result})
                                        })
})
```

Listing 3 – Code de recherche par id

6 Question 4 - Hacking facile [/2]

Pour répondre à cette question, démarrez uniquement la machine virtuelle Bufferoverflows que vous trouverez dans votre répertoire /home/INF4420a/A2022/TP2/.

Ouvrez la session Invité et allez sur la page http ://ip. Vous devez trouver comment entrer dans le système suivant, sans trouver les noms d'usagers et les mots de passe (qui sont évidents dans ce cas-ci, puisqu'ils sont en texte clair dans le programme). Il faut utiliser une attaque basée sur un problème de sécurité dans le programme. Vous devez vous baser uniquement sur ce code et non supposer qu'il y a des trous de sécurité dans d'autres composantes du système (comme des injections SQL). Votre seule manière d'interagir avec le système est donc d'envoyer et de recevoir des caractères au programme. Si votre attaque est un succès, vous n'aurez entré aucun des mots de

passe de la liste puis le programme écrira "Bienvenu sur ce système ..." et terminera correctement (il ne doit pas y avoir de faute système). Le fichier exécutable Windows et le code source en C sont disponibles dans le dossier « hack1 » de l'archive « Utilitaires TP2 » sur le site Moodle (utilisez le fichier exécutable fourni et non pas une version que vous avez compilée).

1. [/0.4] Identifiez les adresses ou commencent le nom d'utilisateur saisi et la première instance du tableau des utilisateurs (l'utilisateur "root")
2. [/0.2] Calculez le nombre de caractères nécessaires pour atteindre la première instance "root" à partir de l'utilisateur.
3. [/0.8] Donnez la séquence exacte de caractères à entrer pour accéder au système. Expliquez brièvement comment votre « hack » fonctionne.
4. [/0.6] Que faudrait-il changer dans le programme pour enlever ce problème de sécurité ?

Références

[1] **nmap** => https://nmap.org/

[2] **Framework Metasploit** => https://www.offensive-security.com/metasploit-unleashed/exploits/

[3] **Configurer Burp** => https://portswigger.net/support/configuring-your-browser-to-work-with-burp

[4] **XSS** => https://owasp.org/www-community/attacks/xss/

[5] **Injection SQL** => https://owasp.org/www-community/attacks/SQL_Injection

[6] **Injecction de commandes** => https://portswigger.net/web-security/os-command-injection

[7] **sqlmap** => http://sqlmap.org/

[8] **Buffer overflow** => http://en.wikipedia.org/wiki/Buffer_overflow#Basic_example

[9] **Stack buffer overflow** => http://en.wikipedia.org/wiki/Stack_buffer_overflow

[10] **Smashing The Stack For Fun and Profit** => http://phrack.org/issues/49/14.html