RAPPORT SUR L'AVANCEMENT DU PROJET

Ceci constitue un rapport sur l'avancement du projet relatif au A06: Composants Vulnérables et Obsolètes. Durant ces 15 derniers jours, nous avons principalement travaillé sur l'établissement de scénarios d'attaque afin de comprendre et d'expliciter les vulnérabilités relatives aux composants (Obsolètes ou non).

Pour notre cas, nous avons prévu deux scénarios d'attaque pour le moment

Scénario 1 :

Les composants s'exécutent généralement avec le même niveau de privilèges que l'application, et donc les failles d'un quelconque composant peuvent aboutir à un impact sévère. Les failles peuvent être accidentelles (ex : erreur de développement) ou intentionnelles (ex : porte dérobée dans un composant). Voici quelques exemples de découvertes de vulnérabilités de composants exploitables :

- CVE-2017-5638, une vulnérabilité d'exécution à distance de Struts 2, qui permet l'exécution de code arbitraire sur le serveur, a été responsable d'importantes violations ;
- Bien que l'internet des objets (IoT) soit souvent difficile, voire impossible à mettre à jour, l'importance de ces mises à jour peut être énorme (ex : objets biomédicaux).

Pour la mise en pratique, nous allons utiliser les vulnérabilités de apache apache struts2 Apache Struts est un framework gratuit et open source utilisée pour créer des applications Web Java. Nous avons examiné plusieurs vulnérabilités d'exécution de code à distance (RCE) signalées dans Apache Struts et avons observé que dans la plupart d'entre elles, les attaquants ont utilisé des expressions OGNL (Object Graph Navigation Language). L'utilisation d'OGNL facilite l'exécution de code arbitraire à distance, car Apache Struts l'utilise pour la plupart de ses processus

Scénario de l'attaque :

Cette vulnérabilité particulière peut être exploitée si l'attaquant envoie une requête spécialement conçue pour télécharger un fichier sur un serveur vulnérable qui utilise un plug-in basé à Jakarta pour traiter la requête de téléchargement. L'attaquant peut alors envoyer du code malveillant dans l'en-tête Content-Type pour exécuter la commande sur un serveur vulnérable.

Scénario 2: RC4StaticKeyAttack

Ceci est une attaque sur les implémentations RC4 faibles qui utilisent des clés statiques. Cependant, des conditions doivent être respectées au préalable, à savoir :

1. **L'attaquant doit avoir la capacité de chiffrer un texte en clair connu**

Pour que cela fonctionne, vous devez pouvoir chiffrer un texte en clair connu ou vous avez besoin d'un texte en clair connu et du texte chiffré correspondant.

2. **Le système de chiffrement doit utiliser une clé statique.**

Une implémentation vulnérable devrait ressembler à ceci.

```
key = 'Never ever use the same key more than once!!!!'
encrypted = enc(key, plaintext)
print(base64.b64encode(encrypted))
```

Remarquez la clé statique codée en dur.

S'il n'y a pas d'accès au code source du système de chiffrement, une clé statique peut être identifiée en chiffrant deux fois un texte en clair connu. Si les deux textes chiffrés résultants sont identiques, il est presque certain que la même clé de chiffrement est utilisée.

Un exemple d'une telle implémentation vulnérable peut être trouvé dans crypt.py. Utilisez le script crypt.pypour chiffrer deux fois un texte en clair donné.

```
python crypt.py testData/knownPlainTextSmall.txt
H4FXzKw=
python crypt.py testData/knownPlainTextSmall.txt
H4FXzKw=
```

Remarquez les textes chiffrés identiques.

3. **Usage**

Trouvez un système de chiffrement vulnérable et laissez-le chiffrer un texte en clair connu. Enregistrez le texte chiffré correspondant. Transmettez ensuite le texte en clair connu, le texte chiffré connu et le texte chiffré que vous souhaitez craquer à rc4Cracker.py

```
python3 rc4Cracker.py testData/knownPlainText.txt testData/knownPlainText.rc4 testData/unknownPlainText.rc4
```

Le script renverra le texte en clair précédemment inconnu.