N00b Hack

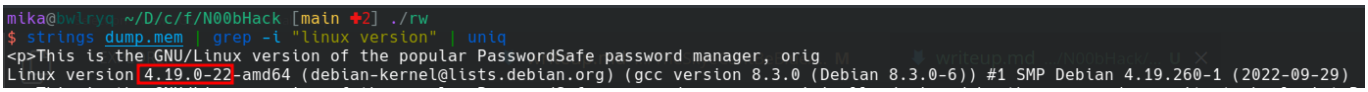
Auteur: bWlrYQ

1. Reconnaissance

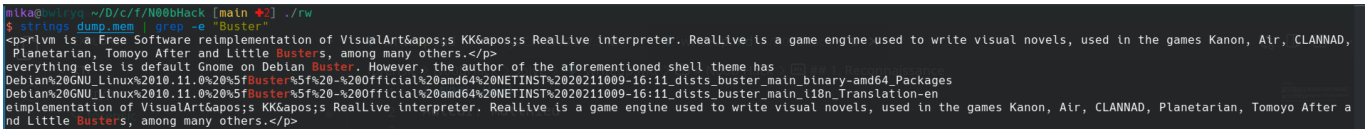
Un dump de la RAM nous est fourni, l'énoncé parle de déterminer la version du Kernel ainsi que l'OS utilisé. Une question sur la version du kernel est peu courante en analyse de mémoire Windows, on peut donc dores et déjà comprendre que nous ne somme pas sur une analyse de méoire Windows mais plutôt d'un autre OS.

Cela complique immédiatement les choses car Volatility ne fournit la table des symboles que pour Windows. Sur un OS de type Linux,MacOS ou encore Android, c'est à nous de fournir le profil adéquat pour avoir les bons symboles.

On va tout de même vérifier nos dires avec un petit `strings dump.mem | grep -i "linux version" | uniq`



On voit donc qu'on est bien sur une distribution Linux (ici Debian), nous avons la version du Kernel mais pas celle de Debian. On va donc recommencer avec `strings dump.mem | grep -e "Buster"`. Buster est ici le nom de la Version de Debian utilisée, on aurait pu essayer au préalable Bullseye etc... Dans notre cas on est sur Debian 10.11.0 (voir screen).



Premier flag: FMCTF{Debian:4.0.19-22}

2. Reconstruction du profil

Afin de pouvoir analyser le dump avec volatility, il nous faut maintenant construire le profil. Ce qui signifie qu'il nous faut la bonne version de Debian ainsi que le kernel associé. Pour cela nous allons nous mettre en quête d'un ISO de Debian 10.11.0 que l'on peut trouver ici.

Une fois l'ISO téléchargé on peut l'installer sur une machine virtuelle. On peut vérifier le version du kernel avec `uname -a`. Dans notre cas la version du Kernel correspond à celle du dump. Nous n'avons donc pas besoin de downgrade la version du kernel (ce qui est une bonne chose, car c'est pénible).

Pour construire le profil il va nous falloir deux choses, volatility (qui implémente une méthode toute faire pour le générer), et les paquets dwarfdump et zip. On va donc utiliser la série de commandes suivante (en root):

```
apt install -y dwarfdump zip
cd ~/
git clone https://github.com/volatilityfoundation/volatility.git
```

```
cd volatility/tools/linux/ && make
cd ~/
zip $(lsb_release -i -s)_$(uname -r)_profile.zip
./volatility/tools/linux/module.dwarf /boot/System.map-$(uname -r)
```

Et voilà, notre profil est généré. On peut récupérer le profil sur notre machine hôte (par exemple avec un bête `python3 -m http.server 80`) puis téléchargr le profil depuis notre hôte. On peut encore monter un dossier partagé entre l'hôte et la VM.

3. Utilisation du profil dans Volatility

Sur notre hôte, on va installer volatility. Si votre système est compatible APT on peut utiliser ce super script de AbyssWatcher qui permet de faire une installation propre.

Une fois volatility installé on peut copier notre profil dans `/opt/volatility/volatility/plugins/overlays/linux` (ou ailleurs en fonction de votre installation). Si vous utilisez simplement le rep github ce sera juste `./volatility/volatility/plugins/overlays/linux`.

Pour vérifier que notre profil a bien été importé on peut faire `python2 vol.py --info | grep "Debian"`

```
mika@bwlryq ~/D/volatility [master ✔] ./rw
$ python2 vol.py --info | grep Debian
Volatility Foundation Volatility Framework 2.6.1
LinuxDebian_4_19_0-22-amd64_profilex64 - A Profile for Linux Debian_4.19.0-22-
amd64_profile x64
```

Notre profil est importé on peut désormais l'utiliser pour analyser notre dump.

4. Analyse du dump de mémoire

Pour commencer on va pouvoir regarder quelles sont les commandes bash qui ont été tapées dans la console avec `python2 vol.py -f dump.mem --profile=LinuxDebian_4_19_0-22-amd64_profilex64 linux_bash`

```
mika@bwlryq ~/D/volatility [master ✔] ./rw
$ python2 vol.py --profile=LinuxDebian_4_19_0-22-amd64_profilex64  -f
../ctf/forensic/N00bHack/dump.mem linux_bash
↵ 2
Volatility Foundation Volatility Framework 2.6.1
Pid      Name                 Command Time                   Command
-------- -------------------- ------------------------------ -------
    2031 bash                 2022-11-01 16:23:31 UTC+0000   ./TP_R314
                        [...SNIPPED...]
    2068 bash                 2022-11-01 16:25:21 UTC+0000   sudo rm -r azazel/
```

On ne remarque rien de très intéressant, si ce n'est qu'un dossier azazel a été supprimé et qu'une programme TP_R314 a été exécuté. En se renseignant un petit peu on peut trouver sur Interne qu'azazel est un rootkit.

Aucune trace de son exécution n'a été détectée on peut donc le laisser de côté. En revanche on pourrait creuser du côté du programme TP_R314. Pour cela on peut utiliser `python2 vol.py -f dump.mem --profile=LinuxDebian_4_19_0-22-amd64_profilex64 linux_pstree | grep TP_R314 -A 5` afin de voir ce qui a été exécuté par la suite par le programme.

```
mika@bwlryq ~/D/volatility [master ✔] ./rw
$ python2 vol.py --profile=LinuxDebian_4_19_0-22-amd64_profilex64 -f ../ctf/forensic/N00bHack/dump.mem linux_pstree | grep "TP_R314" -A 5
Volatility Foundation Volatility Framework 2.6.1
..TP_R314             1910             1000
...sh                 1914             1000
..gnome-terminal-     1934             1000
```

Il semble que que le programme `TP_314` ait seulement lancé une instance sh ce qui semble un peu suspect. Essayons de voir si ce programme a essayé de sortir sur Internet avec `python2 vol.py -f dump.mem --profile=LinuxDebian_4_19_0-22-amd64_profilex64 linux_pstree | grep TP_R314 -A 5`.

```
mika@bwlryq ~/D/volatility [master ✔] ./rw
$ python2 vol.py --profile=LinuxDebian_4_19_0-22-amd64_profilex64 -f ../ctf/forensic/N00bHack/dump.mem linux_netstat | grep "TP_R314" -A 5
Volatility Foundation Volatility Framework 2.6.1
UNIX 27040                TP_R314/1910
UNIX 27041                TP_R314/1910
TCP      192.168.5.79    :38656 51.75.247.236   : 4444 ESTABLISHED                TP_R314/1910
TCP      192.168.5.79    :38656 51.75.247.236   : 4444 ESTABLISHED                     sh/1914
```

On peut voir qu'en plus d'ouvrir uniquement un shell, celui-ci est envoyé à un client distant, cela fait clairement penser à un reverse shell, nous avons donc identifié notre programme malveillant ainsi que l'adresse IP distante à laquelle il est lié.

Nos flags sont donc FMCTF{TP_R314} et FMCTF{51.75.247.236}.

5. Pour aller plus loin

Pour aller plus loin dans l'analyse de ce dump, on pourrait dumper ce qui est spécifique au PID de notre processus malveillant ainsi que l'exécutable afin de faire de la rétro ingénéerie. Ici c'est simplement un payload msfvenom tout ce qu'il y a de plus classique avec un shell meterpreter.