# TP Attaques matérielles et sécurisation - partie Virtualisation

TLS-SEC

Le but de ce TP est de voir comment il est possible d'utiliser un hyperviseur pour surveiller un système d'exploitation.

## Avant de commencer...

1. Bootez sur la machine `"ubuntu .* no5lvl"`
2. Configurez Wayland pour passer en écran unique
3. Copiez les fichiers `/mnt/gei/TP_VIRTUALISATION/{vm.tgz,vmware.tar}` dans le home de `root`

```
cd .
cp -vr /mnt/gei/TP_VIRTUALISATION/{vm.tgz,vmware.tar} .
```

4. **En attendant que ça copie !** : installez `gcc-12` et les `linux-headers` de votre noyau linux

```
apt update
apt install gcc-12 linux-headers-$(uname -r)
```

5. Désarchivez l'installeur de VMware workstation

```
tar xvf vmware.tar
```

6. Installez VMWare workstation

```
bash VMware-Workstation-*bundle
```

7. Lancez VMware (par la fenêtre)

```
vmware
```

8. Configurez VMware en licence d'essai
   (a) Fenêtre "Before you can run Install" → "Install"
   (b) Fenêtre "Please review the license aggreements" → "I accept ... "
   (c) Fenêtre "Please review the license aggreements" → "I accept ... "
   (d) Fenêtre "Would you like to check" → "No"
   (e) Fenêtre "VMware spyware experience" → "No"

(f) Fenêtre licence → "I want to license VMWare 17 for Personal Use"

9. Décompressez la VM Debian 7 64-bit du tp

```
tar xvzf vm.tgz
```

10. Importez la VM du tp
    (a) "Open a virtual machine"
    (b) Sélectionnez `/root/Debian 7 64-bit.vmwarevm/Debian 7 64-bit.vmx"`
11. Démarrez la VMa pour la tester
    (a) Sélectionnez la VM dans le panneau de gauche
    (b) Cliquez sur le bouton "play"
    (c) Selectionnez "I copied it"
12. Éteignez la VM

# 1 Monitoring de VM via les VProbes (Vmware)

Les VProbes de VMware sont une interface de contrôle de machines virtuelles (Guest) depuis le Host. Ils permettent de lancer de petits scripts donnant un accès très bas niveau à la machine virtuelle durant son exécution sans qu'elle le sache.

Le but de cet exercice est d'utiliser cette interface pour monitorer les droits des shells qui sont exécutés dans la VM. On imagine que cela pourrait servir notamment à savoir si des shells root sont en cours d'exécution.

Une VM Debian est fournie. Elle servira de cible à étudier avec les VProbes.

## 1.1 Les VProbes en bref

Suivant les version de Vmware, il est possible d'utiliser les VProbes avec l'outil `vmware-vprobe`. Une documentation complète peut être trouvée ici : `https://www.vmware.com/products/beta/ws/vprobes_reference.pdf`.

Liste des commandes VProbes de l'outil `vmware-vprobe` :

| VPROBE COMMANDS | PARAMETERS | DESCRIPTION |
|---|---|---|
| -l | Path to vmx file 'VP script text' | Load VP script |
| -r | Path to vmx file | Disable all vprobes |
| -p | Path to vmx file | List probes |
| -g | Path to vmx file | List global variables |

Pour la suite du TP, on appelera "$VMX" le chemin vers le fichier de configuration de la machine virtuelle. Il peut être stocké dans une variable shell et utilisé ensuite dans vos lignes de commande :

```
cd CHEMIN_DE_LA_VM
VMX="$(pwd)/Debian 7 64-bit.vmx"
cd -
```

Par ailleurs, les vprobes, par défaut, écrivent la sortie de leur exécution dans le fichier "vprobe.out" dans le répertoire de votre machine virtuelle. Vous trouverez également un fichier "vprobe.err" en cas d'erreurs.

Il est donc possible (et utile) d'ouvrir un shell dédié à l'affichage de la sortie de vos vprobes, avec la commande : `tail -f vprobe.out`. Ce shell affichera en continue le contenu du fichier de sortie des vprobes exécutées.

Les commandes qui listent les VProbes donnent la liste des événements qu'il est possible d'intercepter grâce a vos scripts de probes. Par exemple :

```
vmware-vprobe -p "$VMX"

Total probes: 45
Guest_TripleFault
Guest_DE
Guest_DB
Guest_CR3Write
...
VMM1Hz
VMM10Hz
...
```

Il est également possible d'accéder à certaines variables vprobes contenant des informations utiles de la VM. La commande suivante, liste toutes ces variables globales accessibles depuis vos scripts :

```
vmware-vprobe -g "$VMX"
Total variables: 100
ARG0
ARG1
...
CR0
CR3
CR4
KERNELGSBASE
CS
RAX
RBX
...
```

## 1.2 Prise en main des VProbes

1. Commencer par vérifier que `vmware-vprobe` fonctionne avec `vmware-vprobe -l Debian.vmx`
2. Pour pouvoir utiliser les VProbes pour analyser une VM, il faut également spécifier les lignes suivantes dans le fichier de configuration `.vmx` de la VM :

```
vprobe.enable = "TRUE"
# Deja definit dans le VMX pas besoin de l'ajouter
# {
debugStub.listen.guest32 = "TRUE"
debugStub.listen.guest64 = "TRUE"
# }
```

3. Ecrire le premier script suivant, dans un fichier hello.vp :

```
(vprobe VMM1Hz
  (printf "hello!\n"))
```

4. Lancer l'exécution du script (avec `vmware-vprobe -l hello.vp $VMX` par exemple). Vous devriez voir des "hello !" remplir le fichier vprobe.out toutes les secondes :

```
hello!
hello!
hello!
```

5. Désactiver ce script à l'aide de la commande suivante : `vmware-vprobe -r $VMX`.
6. Recopier le script suivant, qui utilise la fonction `Guest_CR3Write` de la liste des vprobes pour afficher la valeur du registre CR3 à chaque nouvelle écriture. Le lancer et observer à nouveau `vprobe.out`.

```
# script :
(vprobe Guest_CR3Write
  (printf "0x%x\n" CR3))

# Exemple de vprobe.out :
0x36358000
0x35e85000
0x36250000
```

## 1.3 Mise en application

1. Ecrire un script `.vp` qui affiche la valeur de RIP a chaque écriture dans CR3 de la VM fournie, en s'appuyant sur la fonction VProbes `Guest_CR3Write`.
2. A l'aide de `gdb`, vérifier que les valeurs de RIP obtenues correspondent bien à une écriture dans CR3 :

```
(gdb) target remote :8832
(gdb) x/i <addr> # ex:0xc1854633
0xc1854633: mov    %eax,%cr3
```

3. Afficher à chaque écriture de CR3, la valeur de RSP uniquement s'il s'agit d'une pile noyau (ring0), en s'aidant des tables de pages et se souvenant du fait qu'avec un noyau Linux, les adresses virtuelles ¿3Go sont réservées au noyau.
4. Écrire un script qui comptabilise le nombre de fois que chaque CR3 a été chargé, et affiche le résultat toutes les secondes (utiliser une Aggregate Variable, cf. doc VProbes).
5. A chaque écriture dans CR3, afficher les 8 premiers octets du début de la pile noyau courante.
6. Comprendre la fonction "curprocname" donnée en exemple dans la documentation, puis la réécrire pour la faire fonctionner dans votre VM, pour votre version de noyau Linux. Indices :
   — Récupérer le pointeur de pile noyau
   — L'aligner sur la taille d'une pile noyau (*0xffffe000*)
   — Le début de la pile noyau correspond à la structure thread_info
   — Le premier champ de cette structure est un pointeur vers une task struct
   — Le champ "comm" de la task struct contient le nom du programme lié à la pile noyau courante. Dans le cas de la VM Debian fournie, l'offset de ce champ vaut `0x1cc`.

7. Écrire un script qui affiche les credentials (task-¿cred-¿euid) de chaque shell "bash" qu'il trouve. Pour information, dans le cas de la VM fournie :
   — `cred` est stocké à l'offset `0x1c4` dans `task`
   — `euid` est stocké à l'offset `0x14` dans `cred`.
8. Le tester en créant un utilisateur Linux autre que root et en utilisant bash avec ce nouveau utilisateur.
9. Comment pourrait-on utiliser ce mécanisme pour détecter les élévations de privilèges malveillantes ?

## 1.4 Pour aller plus loin...

1. Comment retrouver les 3 offsets donnés ci-dessus (s'inspirer du code dans `anais/dt.c` dans la vm) ?
2. Appliquer cette méthode à une autre VM linux de votre choix et monitorer ainsi les droits d'exécution de l'ensemble des shells s'exécutant sur votre VM.