Sécurité pour les applications

1. Préambule

L'objectif de ce TP est de découvrir différents mécanismes de sécurité ciblant des applications.

2. Prérequis

Deux machines virtuelles au format VirtualBox sont utilisées pour ce TP :

- SSE_TP2_Debian : système Debian 9 utilisé pour les autres TP (pour mémoire, le compte utilisateur est *tp* et les mots de passes sont également *tp*), en fonction de ce qui a été fait aux TP précédents, il peut être préférable de repartir d'un état zéro (snapshot ou redéploiement de la machine virtuelle) ;
- SSE_Win10 : dérivée de la machine virtuelle Windows 10[1] mise à disposition par Microsoft pour tester le navigateur Edge, le clavier a été passé en français pour éviter les désagréments de saisie avec un clavier QWERTY, l'antivirus Windows Defender et la mise à jour Windows Update ont été désactivés, et plusieurs outils ont été ajoutés (Mimikatz, John-The-Ripper et la suite Sysinternals). Le mot de passe du compte IEUser est *Passw0rd!* .

3. ASLR

Exercice 3-1

Objectif : comparer 2 implémentation du mécanisme ASLR pour les processus utilisateurs

Distribution Linux

La VM à utiliser pour cette partie de l'exercice est la VM Debian. Avant de commencer, vérifier que l'ASLR est bien activé pour les processus utilisateur.

Générer 2 programmes différents à partir du code source suivant.

```
/*
demoASLR.c

compilation
gcc -o demo_sansPIE -fno-pie -no-pie demoASLR.c
gcc -o demo_avecPIE -fpie -pie  demoASLR.c

*/
```

1 https://developer.microsoft.com/en-us/microsoft-edge/tools/vms/

```
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>

int main(void)
{
      pid_t proc_pid;

      proc_pid = getpid();
      printf("Le PID du processus courant est %d\n\n", proc_pid);
      printf("Appuyer sur la touche Entrée pour arrêter le programme\n");
      getchar();
      return 0;
}
```

Lancer le programme demo_sansPIE, noter le PID puis, dans un autre terminal, afficher les informations sur la mémoire du processus (XXX représente le PID) :

```
cat /proc/XXX/maps
```

Collecter les différentes *Base Address* dans le tableau suivant lors de 3 exécutions des 2 applications, la troisième exécution intervenant après un redémarrage du système.

Les sections exécutables sont du type r-xp.

		demo_sansPIE	demo_avecPIE
1er lancement	Section exécutable de demo_nnnnPIE		
	Section exécutable de la libc (libc-2.24.so)		
	[heap]		
	[stack]		
2è lancement	Section exécutable de demo_nnnnPIE		
	Section exécutable de la libc (libc-2.24.so)		
	[heap]		
	[stack]		
Après redémarrage du système	Section exécutable de demo_nnnnPIE		
	Section exécutable de la libc (libc-2.24.so)		

	[heap]		
	[stack]		

Les 2 binaires se comportent-ils de la même manière au niveau des adresses de chargement du programme en lui-même, des bibliothèques qu'ils utilisent, de leur pile et de leur tas ? Quel est l'effet d'un redémarrage sur ces adresses ?

Les zones de mémoires affectées au tas et à la pile sont-elles exécutables ? Quel principe est appliqué ici ?

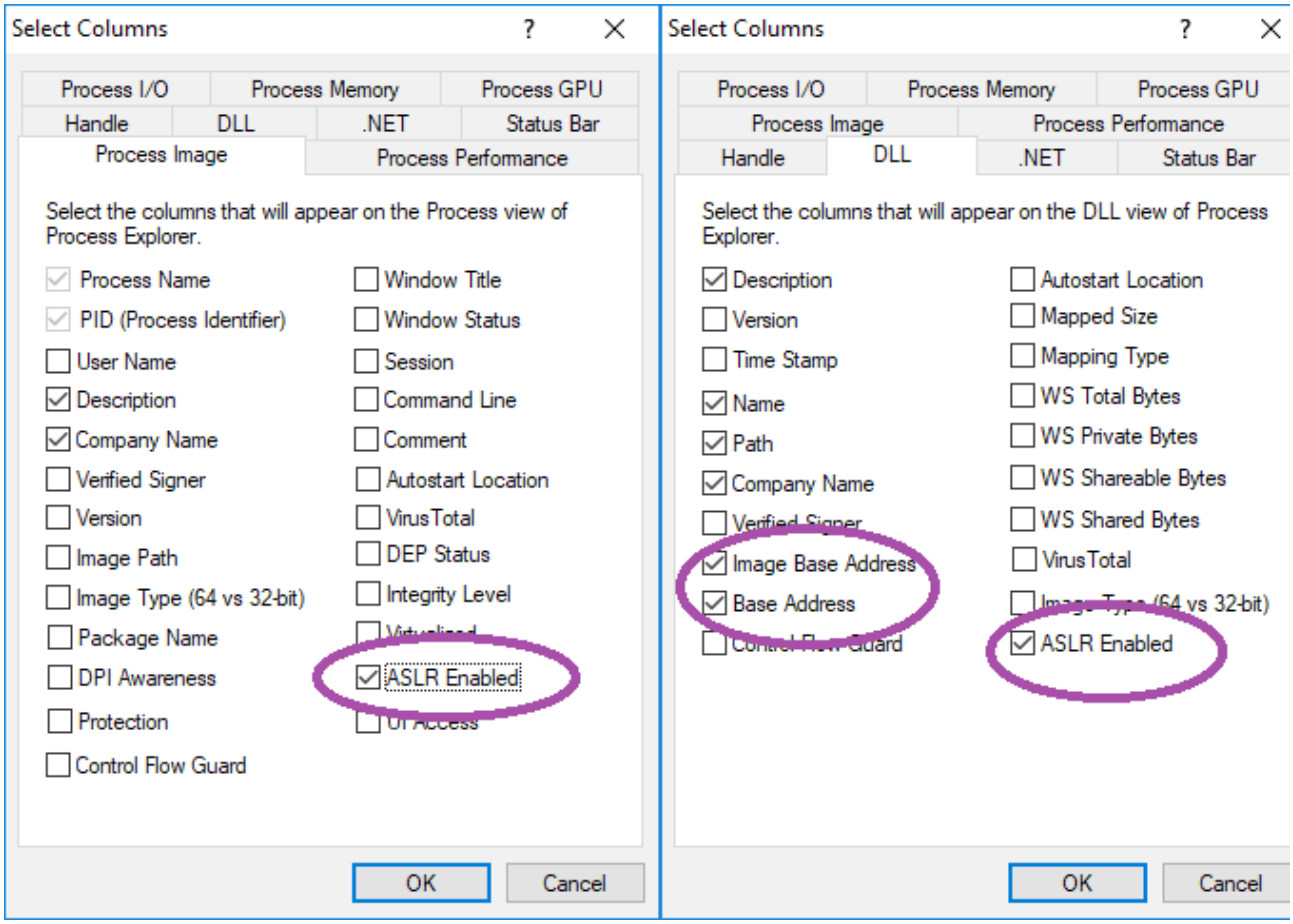
Système Windows

La VM à utiliser pour cette partie de l'exercice est la VM Windows.

Les programmes DemoASLR1.exe et DemoASLR2.exe ne sont pas dans la machine virtuelle, ils sont fournis avec le support de TP. Ils peuvent être insérés dans la machine virtuelle par l'option Glisser-Déposer (*Drag&Drop*) si celle-ci est activée pour la VM, en montant une clé USB dans la VM ou chargeant l'image ISO fournit comme lecteur de CD de la VM. Autre option, se connecter à Moodle depuis la VM si celle-ci a accès à Internet.

Lancez l'outil Process Explorer puis les programmes DemoASLR1.exe et DemoASLR2.exe sans les fermer.

Dans Process Explorer, afficher les colonnes « ASLR » pour les processus et « ASLR », « Image Base Address » et « Base Address » pour les DLL.



Collecter les différentes *Base Address* dans le tableau suivant lors de 3 exécutions des 2 applications, la troisième exécution intervenant après un redémarrage du système.

		DemoASLR1	DemoASLR2
1er lancement	DemoASLRn.exe		
	Kernel32.dll		
	Ntdll.dll		
2è lancement	DemoASLRn.exe		
	Kernel32.dll		
	Ntdll.dll		
Après redémarrage du système	DemoASLRn.exe		
	Kernel32.dll		
	Ntdll.dll		

Les 2 binaires se comportent-ils de la même manière au niveau des adresses de chargement du programme en lui-même et des bibliothèques qu'ils utilisent ? Quel est l'effet d'un redémarrage sur ces adresses ?

Sur le site https://winitor.com/binaries.html, récupérer l'application PE Studio et analyser les 2 binaires. Quelle information vous permet d'expliquer le comportement observé ?

Le fonctionnement du mécanisme ASLR est-il identique sur les deux systèmes d'exploitation considérés ? Qu'en pensez-vous ?

Mandatory ASLR (Windows)

La page suivante décrit les conditions pour, en principe, forcer l'ASLR pour tous les processus : https://blogs.technet.microsoft.com/srd/2017/11/21/clarifying-the-behavior-of-mandatory-aslr/

Activer la configuration décrite avec la commande suivante (en vérifiant sa bonne prise en compte dans le registre) puis redémarrer.

```
Set-ProcessMitigation -Name DemoASLR4 -Enable ForeceRelocateImages, BottomUp
```

Le comportement du programme DemoASLR2 est-il différent ? Qu'en pensez-vous ?

4. Contrôle applicatif sous Windows

Exercice 4-1

Objectif : mettre en œuvre *AppLocker* et identifier des contournements simples

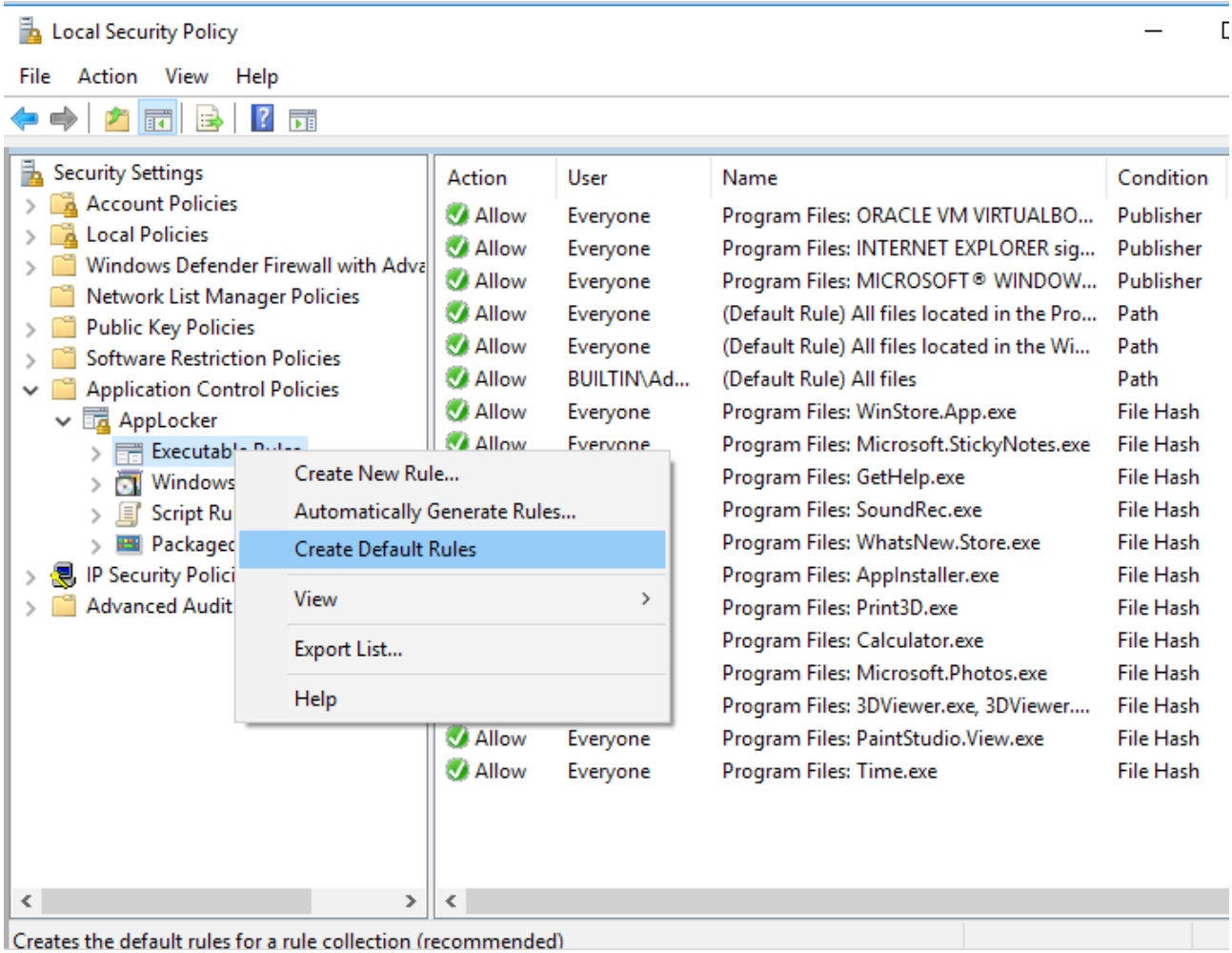
La VM à utiliser pour cet exercice est la VM Windows.

Copier le fichier tp-applocker.xml (fourni avec le support de TP) dans le répertoire C:\tp.

A l'aide d'une invite de commande Powershell élevée (administrateur), lancer les commandes suivantes :

```
Set-AppLockerPolicy -xml C:\tp\tp-applocker.xml
secpol.msc
```

Observer les règles qui ont été importées dans « Application Control Policies\AppLocker\Executable Rules » (elles ont été créées à partir des options « Create Default Rules » et « Automatically Generates Rules » puis simplifier pour une question de lisibilité).



Taper les commandes Powershell suivantes (utilisant la cmdlet Test-AppLockerPolicy) :

```
Test-AppLockerPolicy -xml C:\tp\tp-applocker.xml -Path C:\Windows\notepad.exe
Test-AppLockerPolicy -xml C:\tp\tp-applocker.xml -Path  C:\tp\mimikatz\mimikatz.exe
```

Quels sont les informations obtenues et que peut-on en déduire sur l'exécution de ces 2 applications ?

_____

_____

_____

Il est nécessaire d'activer le service *Application Identity* pour que *AppLocker* fonctionne. Avec une invite de commande élevée, taper les commandes suivantes :

```
sc.exe config AppIdSvc start=auto
sc.exe start AppIdSvc
```

Identifier un utilisateur non-privilégié[2] (i.e. issu d'un TP précédent) ou en créer un (par ex. à l'aide de la commande `net user /add util_std`). Ouvrir une session avec ce compte utilisateur.

Nota : si le menu démarrer ne s'affiche pas, le clic droit sur l'icône Windows ou les raccourcis clavier « touche Windows »+E ou ALT+F4 pourront s'avérer pratique.

Pouvez-vous lancer les applications Notepad et Mimikatz ciblée précédemment avec la cmdlet Test-AppLockerPolicy ? Est-ce conforme la stratégie déployée ?

Avec une invite de commande non-élevée, taper les commandes suivantes :

```
cp C:\tp\mimikatz.exe C:\Windows\temp
C:\Windows\temp\mimikatz.exe
```

Que se passe-t-il ? Comment peut-on expliquer ce comportement.

Noter que l'auto-complétion ne fonctionne pas sur le fichier mimikatz.exe. L'ACL en place sur le répertoire C:\Windows\temp permet-elle d'expliquer ce comportement ?

Ce comportement est connu et documenté par Microsoft :

https://docs.microsoft.com/fr-fr/windows/device-security/applocker/select-types-of-rules-to-create#determine-how-to-allow-system-files-to-run

Que faut-il penser des répertoires C:\Windows\Tasks et C:\Windows\System32\spool\drivers\color ?

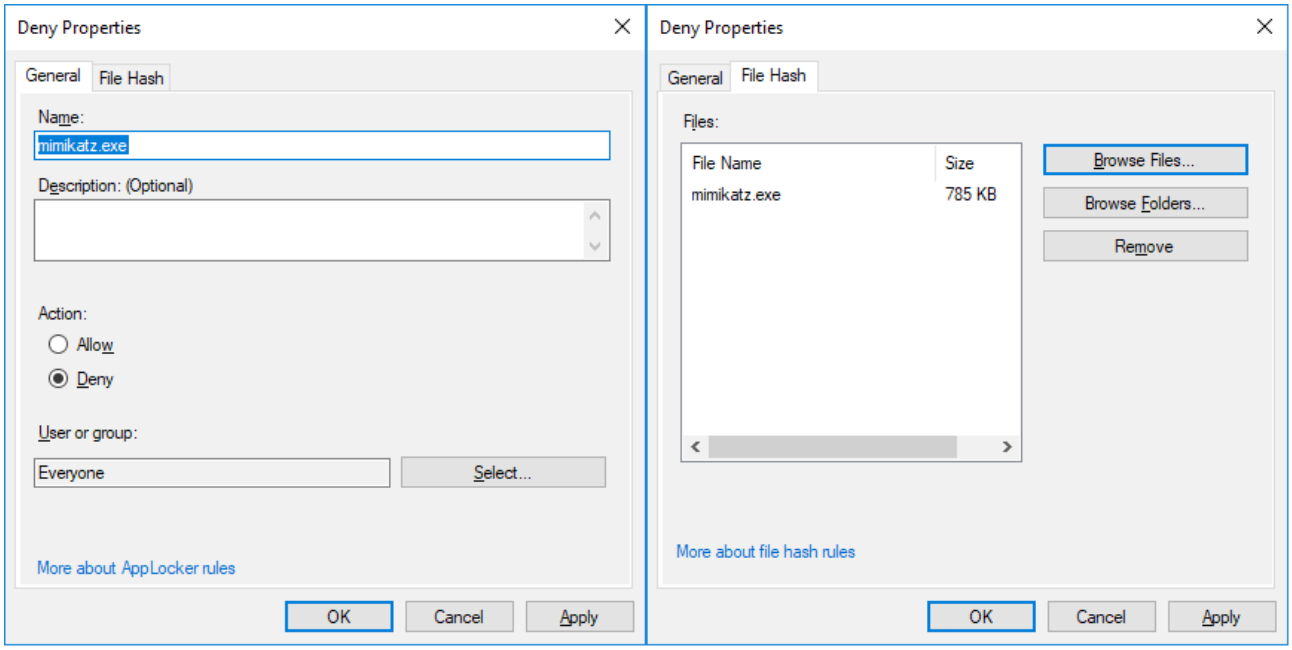
Avec le compte IEUser, relancer l'outil secpol.msc et créer une nouvelle règle pour les fichiers exécutables avec les options suivantes :

- Action : deny
- User or group : Everyone

2 Appartenant uniquement au groupe Users

- Conditions : File hash
- Files : C:\tp\mimikatz\mimikatz.exe

Une fois que cette règle est créée, vérifier que ses propriétés sont bien conformes.



Tester le lancement de Mimikatz à partir des différents répertoires où il a été copié. La règle est-elle efficace ?

Avec une invite de commande Powershell, taper la commande suivante :

```
Get-AppLockerFileInformation -EventLog -EventType Denied -Statistics
```

Les informations apportés peuvent-elles être utiles du point de vue de la sécurité ?

Cette méthode consistant à ajouter des règles de refus ciblant des condensats cryptographiques de fichiers vous parait-elle efficace, notamment vis-à-vis d'une règle de chemin ?