Gunpack

SSTIC 2016

Julien Lenoir - julien.lenoir@airbus.com

1er juin 2016

AIRBUS
GROUP

Outline

1. Concepts
2. Mécanismes internes de Windows
3. Conception
4. Exemples d'utilisation
5. Conclusion

Plan

1 Concepts
2 Mécanismes internes de Windows
3 Conception
4 Exemples d'utilisation
5 Conclusion

Gunpack

C'est fait pour :

- Unpacker des binaires sous Windows 32 bits
- Développer des scripts d'unpacking en Python

Pour s'attaquer à :

- packers "COTS"
- packers Custom
- l'analyse de certains *malwares*

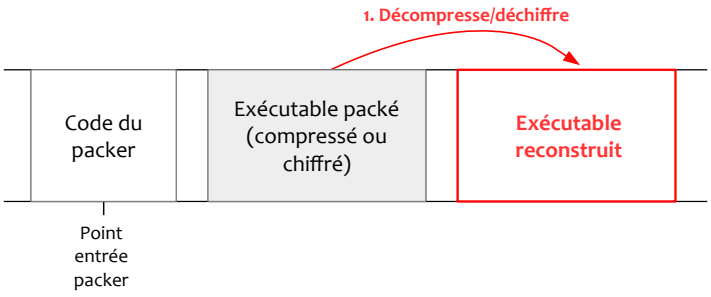
Pas particulièrement adapté à :

Analyse des virtualiseurs (ex : VMProtect)

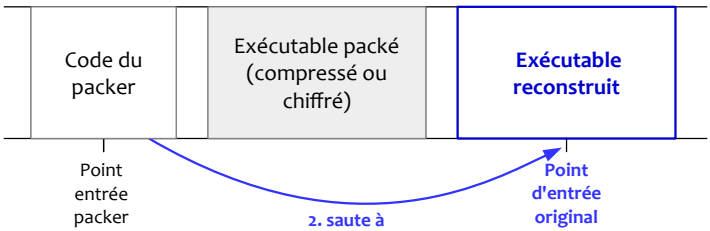
Un packer ?


Code du packer
Exécutable packé (compressé ou chiffré)
Point entrée packer

Un packer ?


1. Décompresse/déchiffre
Code du packer
Exécutable packé (compressé ou chiffré)
Exécutable reconstruit
Point entrée packer

Un packer ?


Code du packer
Exécutable packé (compressé ou chiffré)
Exécutable reconstruit
Point entrée packer
2. saute à
Point d'entrée original

Choix d'implémentation

Instrumentation de l'OS de l'intérieur

Avantage

Utilisation sur une machine physique ou virtuelle

Inconvénient

Détection de l'outil (fichiers, driver)

Limitation

Uniquement Windows 7 en mode PAE

État de l'art

Outils existants

- Renovo (2007)
- Omniunpack (2007)
- Justin (2008)
- MutantX-S (2013)
- Packer Attacker (2015)

Problèmes

- Très peu sont opensource
- Pas vraiment scriptables

Démarche initiale

Implémenter MutantX-S

- Amélioration de l'architecture
- A servi de base à notre design

Défaut

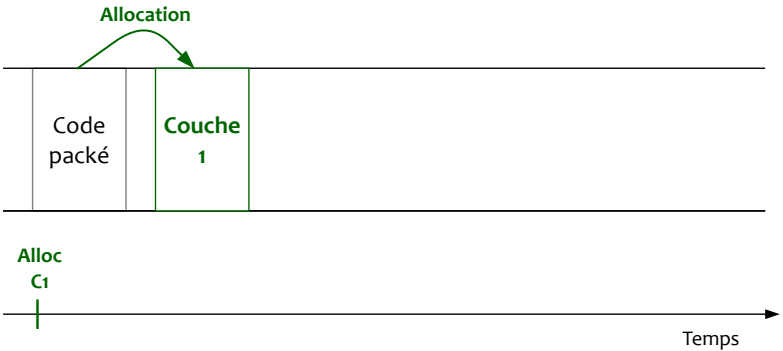
Trop dépendant du malware packé

Nécessité

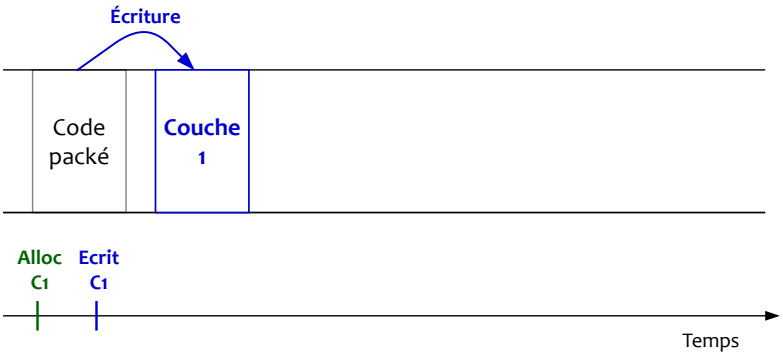
Trouver une solution plus souple

Exemple de packer

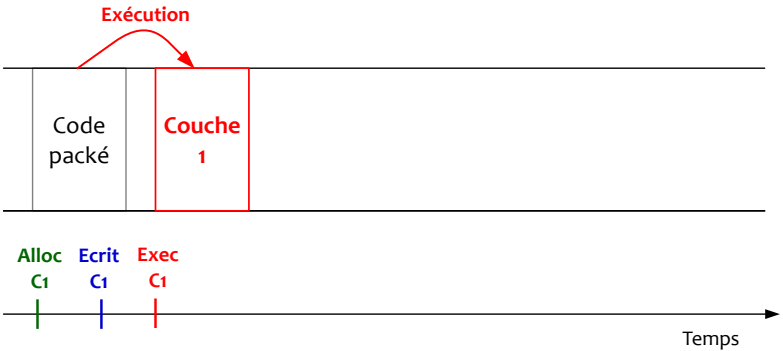
Exemple de packer



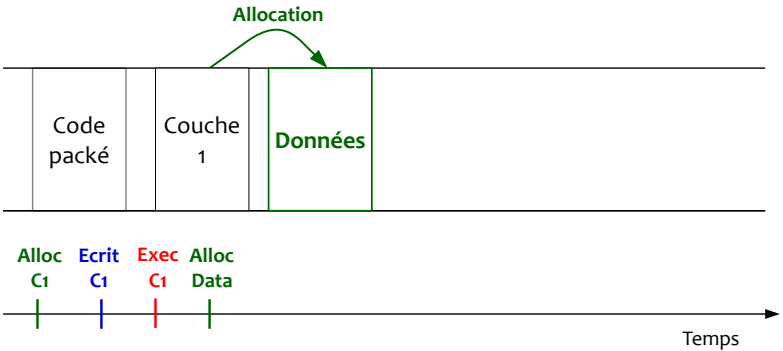
Exemple de packer



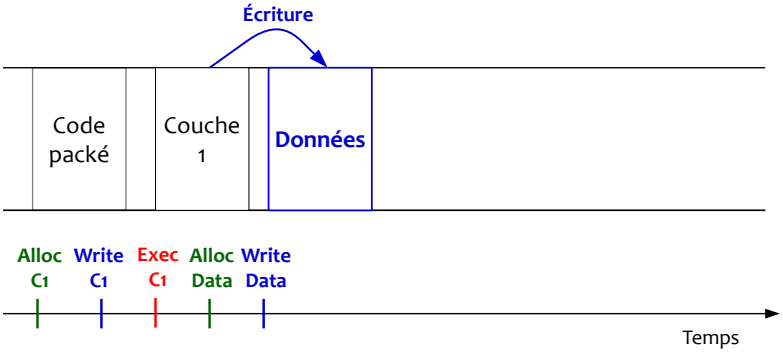
Exemple de packer



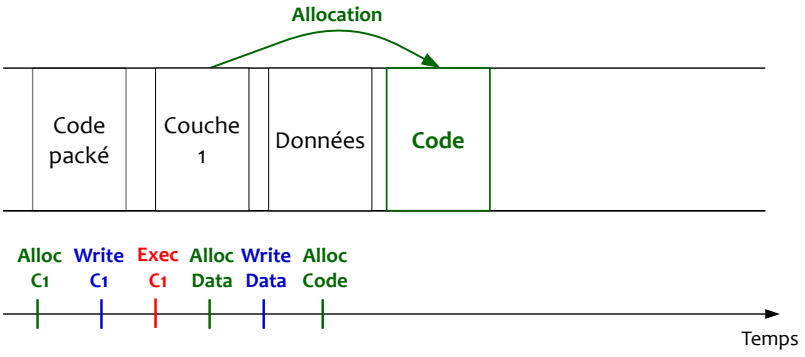
Exemple de packer



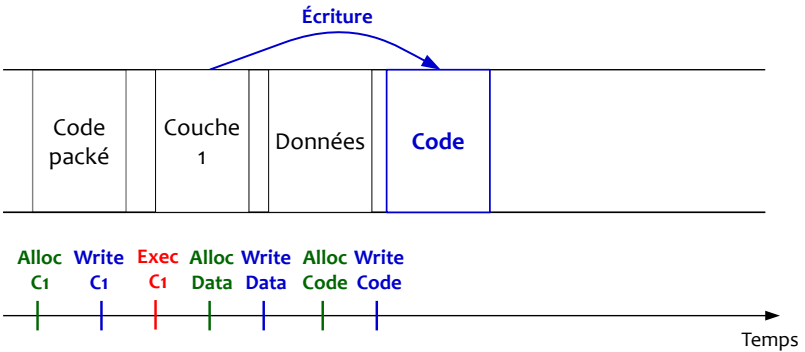
Exemple de packer



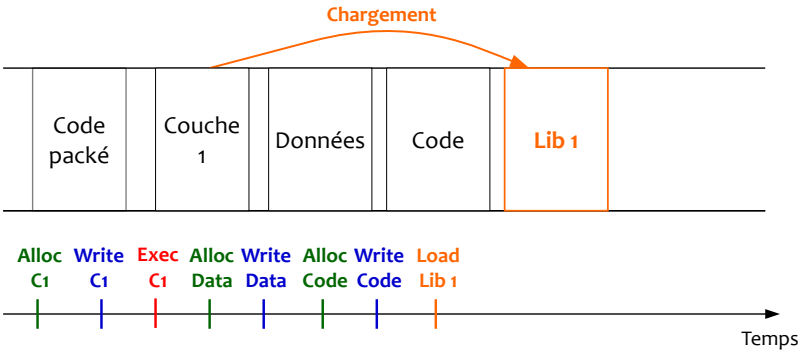
Exemple de packer



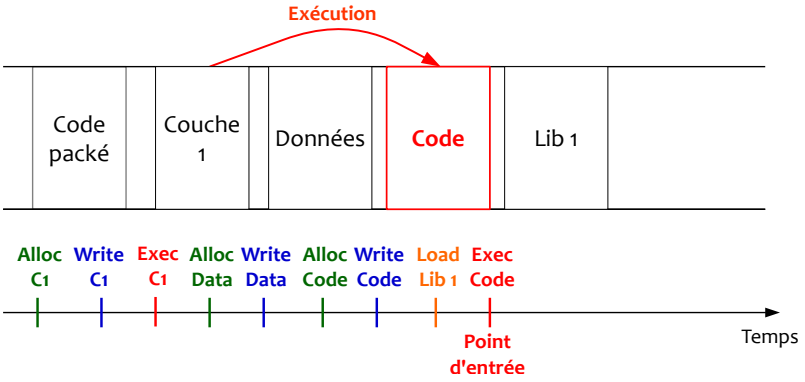
Exemple de packer



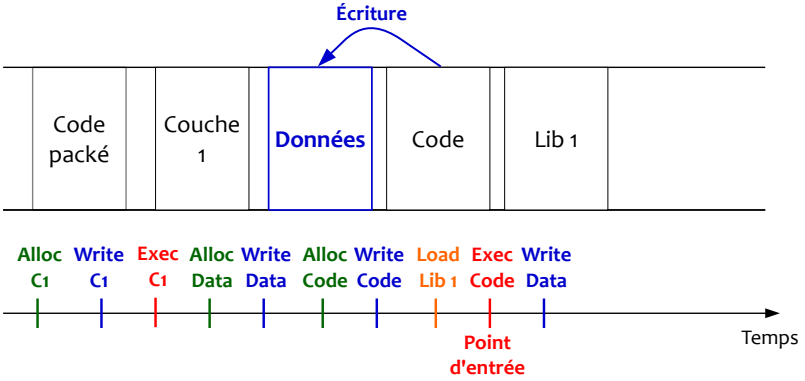
Exemple de packer



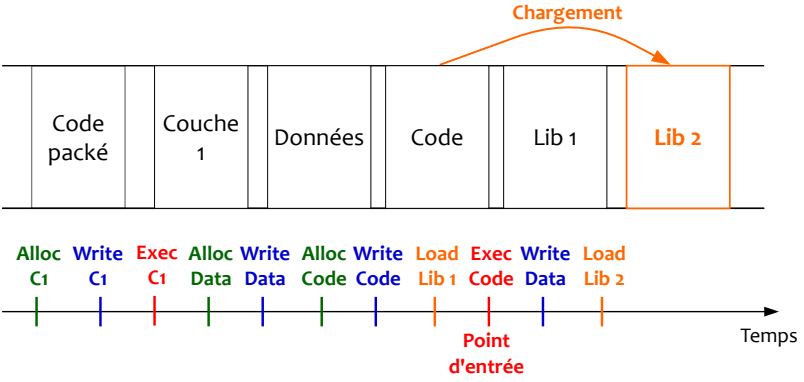
Exemple de packer



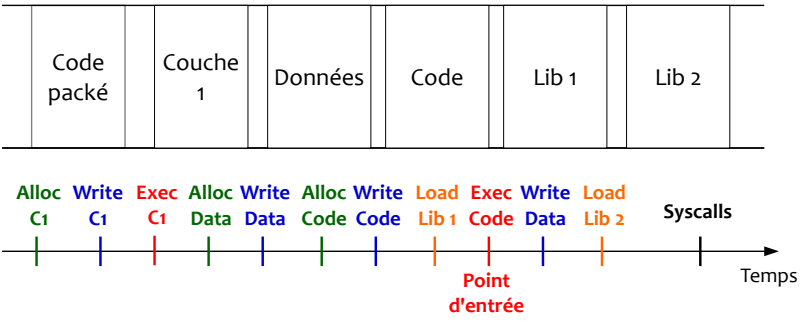
Exemple de packer



Exemple de packer



Exemple de packer



Problématique

Déterminer l'OEP ?

Stratégies :

- Détecter la génération de code
- Surveiller les appels système ≪ sensibles ≫
- Combinaison d'heuristiques

Solution

- Collecter un maximum d'informations sur la vie d'un processus
- Permettre de développer des scripts d'unpacking

Événements collectés

Événements

- Appels systèmes mémoire
  - Allocation
  - Protection
  - Libération
- Création de thread
- Création de processus fils
- Chargement de librairies
- Accès mémoire

Événements collectés

Pour chaque évènement

- Pid du processus
- Tid du thread

A chaque accès mémoire

- Type d'accès (écriture ou exécution)
- Contexte (registres)
- Information sur le loader
- Adresse virtuelle
- Adresse physique

Accès mémoire

Idée

Détection des accès en écriture et en exécution sur les pages de mémoire

Comment ?

- Modification des droits des pages à l'insu du processus
- Le processus effectue des accès invalides
- Fautes générées interceptées

Invariant

Une page est **Inscriptible** xor **Exécutable**

Accès mémoire

Fonctionne avec la granularité d'une page

Deux types de fautes

- En écriture : la page devient **W only**, l'exécution reprend
- En exécution : la page devient **X only**, l'exécution reprend

Permet d'avoir une bonne vue du comportement du processus

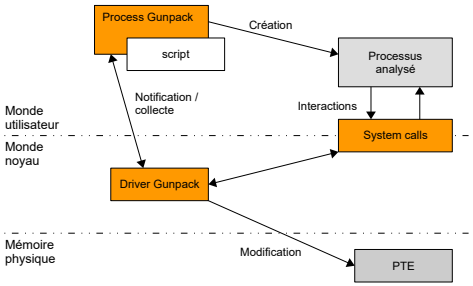
Difficulté

Garder l'OS et le processus fonctionnels

Architecture

Deux composants :

1. Un pilote :
   - Modifie la mémoire du processus
   - Collecte les évènements
   - Filtre les appels système
2. Un processus python :
   - Crée le processus cible
   - Notifie le pilote
   - Rapatrie les évènements
   - Dump la mémoire du processus
3. Des scripts :
   - Implémentent l'intelligence de l'outil



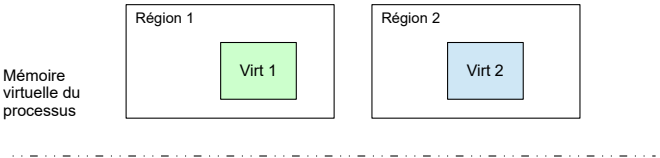
Vidéo

Vidéo

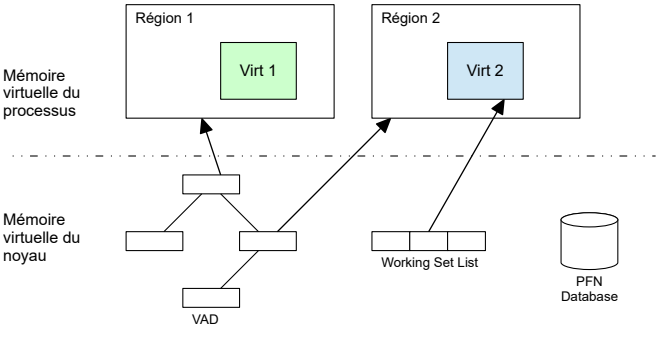
Plan

1 Concepts
2 Mécanismes internes de Windows
3 Conception
4 Exemples d'utilisation
5 Conclusion

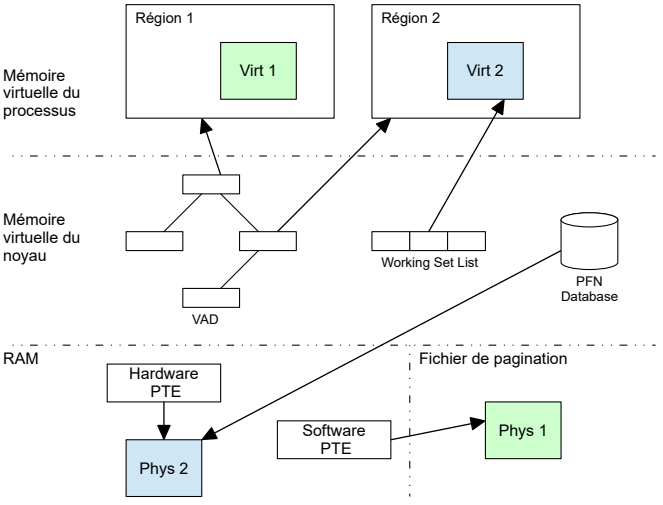
Mémoire


Région 1
Virt 1
Région 2
Virt 2
Mémoire virtuelle du processus

Mémoire

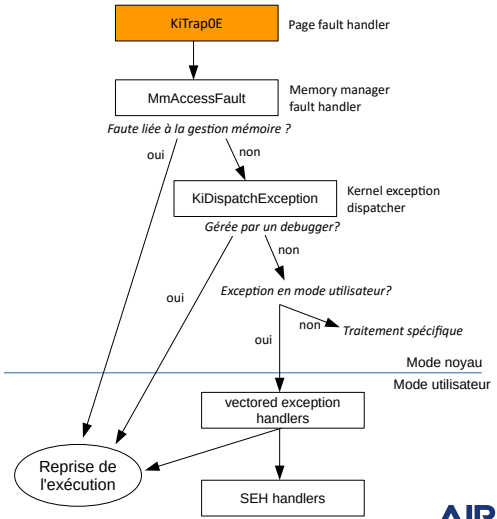

Région 1
Virt 1
Région 2
Virt 2
Mémoire virtuelle du processus
Mémoire virtuelle du noyau
Working Set List
PFN Database
VAD

Mémoire


Région 1
Virt 1
Région 2
Virt 2
Mémoire virtuelle du processus
Mémoire virtuelle du noyau
Working Set List
PFN Database
VAD
RAM
Fichier de pagination
Hardware PTE
Software PTE
Phys 1
Phys 2

Exceptions

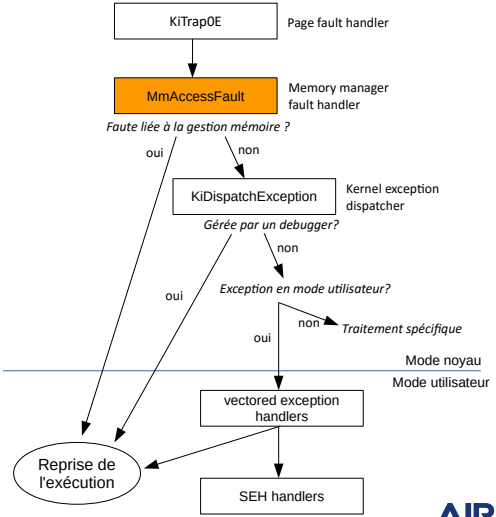
1. Le processeur donne la main au noyau



Exceptions

2. Gestion des fautes liées à au fonctionnement du gestionnaire de mémoire comme :

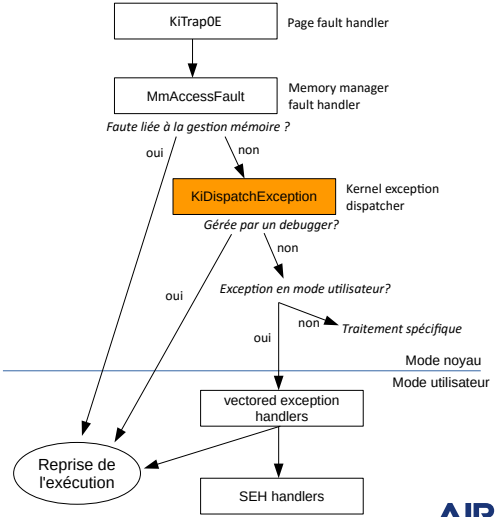
- Page dans le *page file*
- Page *copy on write*
- Page de type *demand paging*



Exceptions

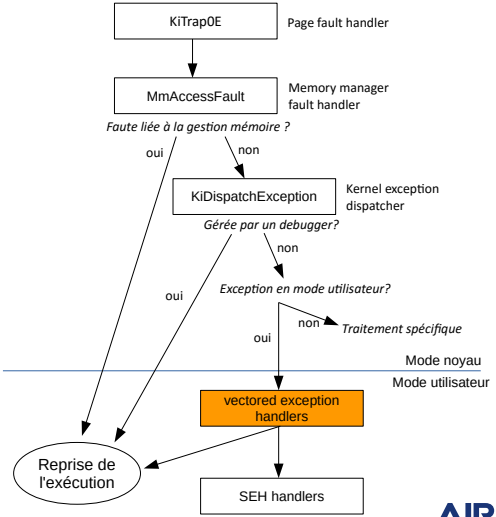
3. La faute est devenue une exception et doit être traitée :

- Par un débugger s'il y en a
- Par un gestionnaire d'exception



Exceptions

4. Traitement de l'exception confié au processus



Plan

1 Concepts

2 Mécanismes internes de Windows

3 Conception

4 Exemples d'utilisation

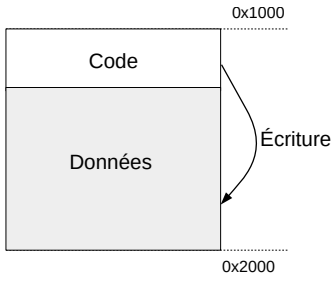
5 Conclusion

Pages auto-modifiantes

Problème posé par l'invariant

- Faute en écriture : la page devient **RW**, reprise de l'exécution
- Faute en exécution : la page devient **RX**, reprise de l'exécution
- ...

Boucle infinie



Besoin

Rompre temporairement l'invariant (W xor X)

Pages auto-modifiantes

Deux possibilités

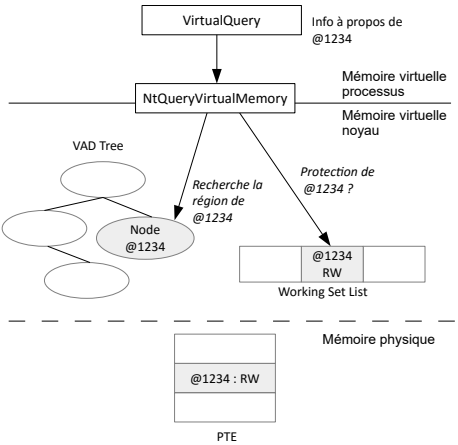
1. Mettre la page RWX et exécuter pas à pas une seule instruction
   - Avantage : très précis
   - Inconvénient : lent
2. Mettre la page en RWX jusqu'au prochain accès invalide
   - Avantage : très rapide
   - Inconvénient : peu précis

Choix

Paramétrable à l'initialisation du moteur

Modification de la mémoire

Impératif : présenter au processus sa *vue* de l'état de sa mémoire



Modification de la mémoire

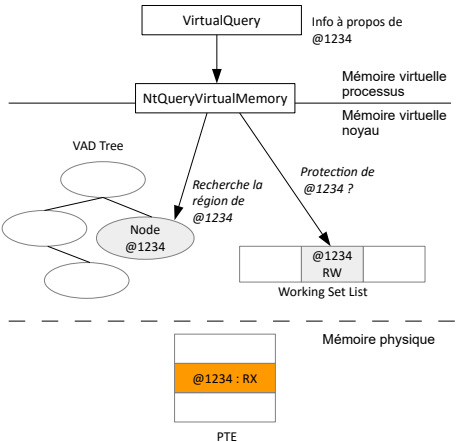
Impératif : présenter au processus sa *vue* de l'état de sa mémoire

Choix

Modifier le PTE directement

Avantages :

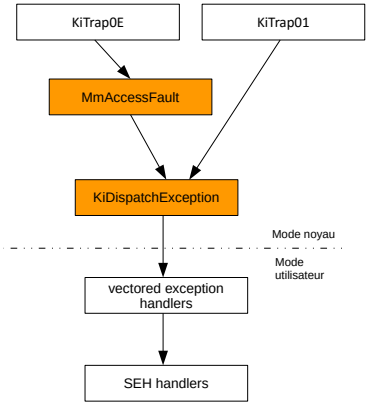
- Simple : pas besoin de modifier les structures internes du noyau
- Efficace : invisible pour le processus



Hook des exceptions

Interceptées à deux endroits :

1. *MmAccessFault* :
   - Empêcher le noyau changer le PTE
   - Restaurer notre PTE si la page a été *swapée*
2. *KiDispatchException* :
   - Pour gérer l'exécution pas à pas



Collecte d'évènements

Accès mémoires

Depuis notre gestionnaire d'exceptions

Interception des *syscalls* mémoire :

- *NtAllocateVirtualMemory*
- *NtProtectVirtualMemory*
- *NtMapViewOfSection*

Utilisation des *NotifyRoutines* :

- Création de processus fils
- Création de thread
- Chargement de librairie

Plan

1 Concepts
2 Mécanismes internes de Windows
3 Conception
4 Exemples d'utilisation
5 Conclusion

Initialisation :

- Ligne de commande
- Paramétrage du moteur :
  - Droits initiaux des pages
  - Stratégie pour les pages auto-modifiantes
  - Durée maximum d'analyse

```
    def __init__(self, file_path):

        self.command_line = file_path
        self.set_rwe_policy(RWE_PAGE_RWX)
        self.set_max_unpack_time(120)
        self.set_initial_nx_state(INITIAL_READ_ONLY)
```

Gestion des évènements :

- Thread suspendu
- Retour :
  - 1 : exécution continue
  - 0 : moteur suspend le processus

```
def event_handler(self, event_type, event_obj):
    if (event_type == EVENT_EXCEPTION):
        e = event_obj
        if (e.AccessedType == EXECUTE_ACCESS):
            if (e.Ctx.Esp == self.packer_oep_esp):
                self.oep = e.Ctx.Eip
                return 0
            if (e.AccessedAddress == self.packer_oep):
                self.packer_oep_esp = e.Ctx.Esp
    return 1
```

Post traitement :

- Processus suspendu

```
def post_treatment(self):

    dump_name = "%s\dumped.exe" % self.output_directory
    unpack_name = "%s\unpacked.exe" % self.output_directory

    self.process.DumpPE(self.process.pid, self.oep,
                        dump_name)
    self.process.Iat_Rebuild(dump_name, unpack_name,
                             self.oep)
    print "Process dump with Oep = 0x%x" % self.oep
```

Analyse multi-processus

Possibilité

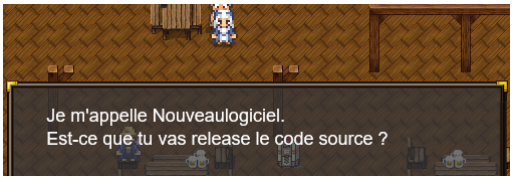
- Collecter les évènements d'autres processus
- function *add_tracked_process*

Cas d'applications

- Etude de malware
  - PlugX
  - Dyre
- Analyse d'exploits

Plan

1 Concepts
2 Mécanismes internes de Windows
3 Conception
4 Exemples d'utilisation
5 Conclusion



nouveaulogiciel si tu es parmis nous :

`https://bitbucket.org/iwseclabs/gunpack.git`

GPL v3

Merci de votre attention !