Volatilitux

Analyse de la mémoire physique des systèmes Linux

Emilien Girault

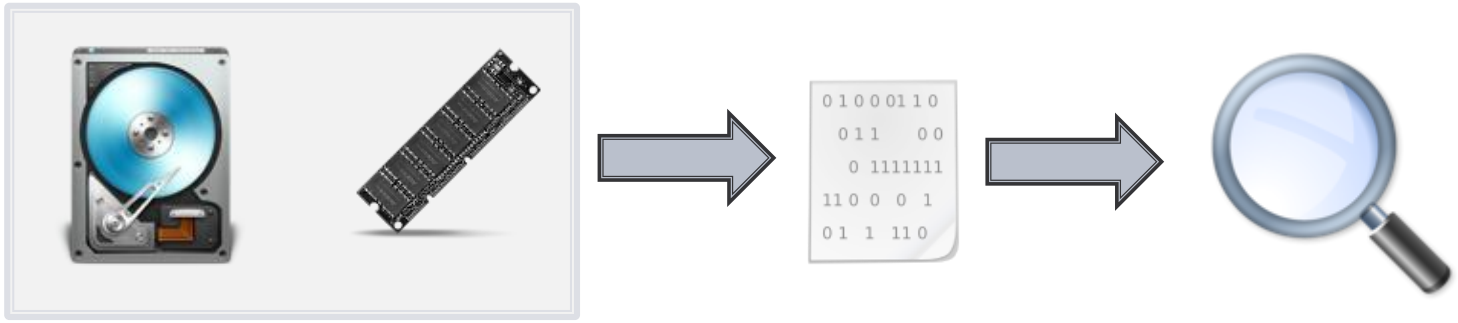
e.girault@sysdream.com
@emiliengirault
www.segmentationfault.fr / www.ghostsinthestack.org

Sommaire

- Introduction
  - Forensics et analyse mémoire
- État de l'art
  - Outils existants
- Volatilitux
  - Motivations
  - Implémentation
  - Résultats
  - Demo

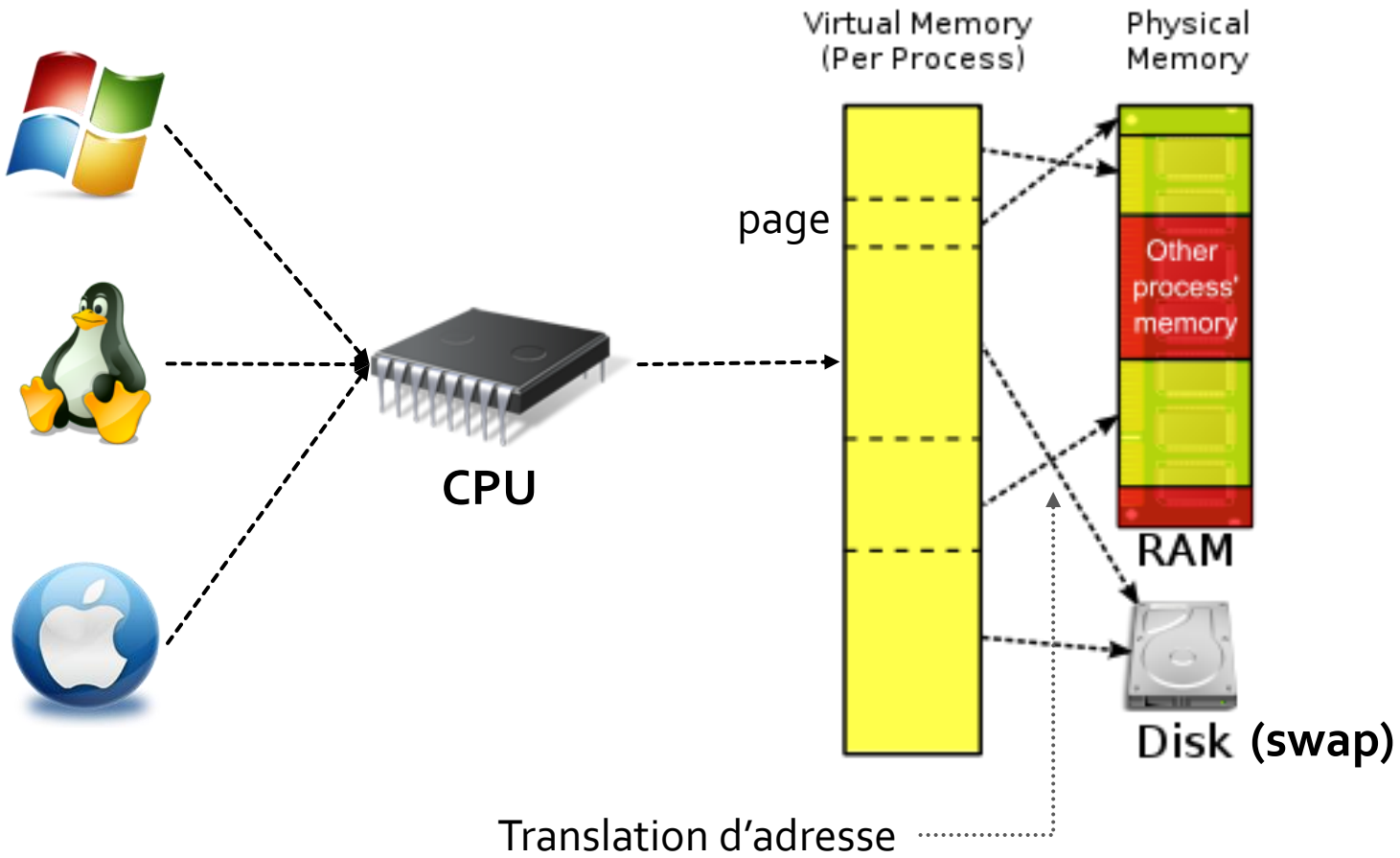
Introduction (1/3)

- Analyse forensique
  - Analyse d'un système informatique après incident
  - Comparable à l'autopsie en biologie
  - Recherche de preuves
    - Processus, connexions, logs, fichiers ouverts/modifiés…



Introduction (2/3)

- Mémoire virtuelle vs mémoire physique



Introduction (3/3)

- L'analyse de RAM : un challenge
  - RAM = pas de structure…
    - Pages non contigües et dans le « désordre »
    - Quelles sont les pages non mappées ? (« trous »)
  - Mémoire virtuelle = structure +/- connue
    - Comment la retrouver ?
      - Où sont les tables de translation d'adresses ?
        - On ne dispose pas des registres CPU…
    - Comment l'analyser ?
      - Dépend de la version de l'OS et des options de compilation
        - Offsets variables, nouvelles structures noyau…

Outils existants

- Volatility [1]
  - Outil libre de référence pour l'analyse de RAM
  - Développé en Python
    - Multi-plateforme
  - Ne supporte que Windows XP SP2 & SP3 ☹
  - Utilisé pour résoudre le challenge Honeynet « Banking Troubles » [2]

[1] https://www.volatilesystems.com/default/volatility
[2] http://www.honeynet.org/challenges/2010_3_banking_troubles

/tmp/Volatility-1.3_Beta

```
$ python volatility

        Volatile Systems Volatility Framework v1.3
        Copyright (C) 2007,2008 Volatile Systems
        Copyright (C) 2007 Komoku, Inc.
        This is free software; see the source for copying conditions.
        There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A PART
ICULAR PURPOSE.

        usage: volatility cmd [cmd_opts]

        Run command cmd with options cmd_opts
        For help on a specific command, run 'volatility cmd --help'

        Supported Internel Commands:
                connections     Print list of open connections
                connscan        Scan for connection objects
                connscan2       Scan for connection objects (New)
                datetime        Get date/time information for image
                dlllist         Print list of loaded dlls for each process
                dmp2raw         Convert a crash dump to a raw dump
                dmpchk          Dump crash dump information
                files           Print list of open files for each process
                hibinfo         Convert hibernation file to linear raw image
                ident           Identify image properties
                memdmp          Dump the addressable memory for a process
                memmap          Print the memory map
                modscan         Scan for modules
                modscan2        Scan for module objects (New)
                modules         Print list of loaded modules
                procdump        Dump a process to an executable sample
                pslist          Print list of running processes
                psscan          Scan for EPROCESS objects
                psscan2         Scan for process objects (New)
                raw2dmp         Convert a raw dump to a crash dump
                regobjkeys      Print list of open regkeys for each process
                sockets         Print list of open sockets
                sockscan        Scan for socket objects
                sockscan2       Scan for socket objects (New)
                strings         Match physical offsets to virtual addresses (may
 take a while, VERY verbose)
                thrdscan        Scan for ETHREAD objects
                thrdscan2       Scan for thread objects (New)
                vaddump         Dump the Vad sections to files
                vadinfo         Dump the VAD info
                vadwalk         Walk the vad tree

        Supported Plugin Commands:
                memmap_ex_2     Print the memory map
                pslist_ex_1     Print list running processes
                pslist_ex_3     Print list running processes
                usrdmp_ex_2     Dump the address space for a process

        Example: volatility pslist -f /path/to/my/file
```

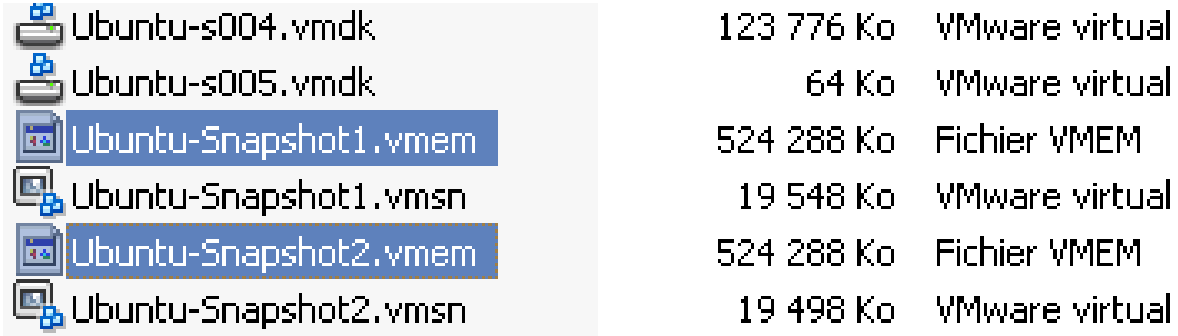
Outils existants

- Foremost [2]
  - Data Carving : Extraction des fichiers basée sur les headers / footers
- Scalpel [3]
  - Optimisation de Foremost
- dd
  - Utilitaire Unix pour l'extraction de blocs binaires
  - 100% manuel…

[2] http://foremost.sourceforge.net/
[3] http://www.digitalforensicssolutions.com/Scalpel/

Outils existants

- MoonSols Windows Memory Toolkit [4]
  - Acquisition de mémoire physique
  - Conversion en crash dump pour analyse avec Windbg
- Snapshots VMWare
  - Fichiers *.vmem



[4] http://www.moonsols.com/

Introduction à Volatilitux

- Challenge SSTIC 2010
  - But : analyser la RAM d'un téléphone Android
- Constat
  - Aucun outil adapté pour Linux
    - A part RAMPARSER [1], mais n'a jamais été publié
  - Nécessite une approche manuelle
    - Editeur hexadécimal, strings, dd
    - Code source du noyau pour comprendre la gestion mémoire de Linux
    - Aspirine…

[1] http://www.dfrws.org/2010/proceedings/2010-304.pdf

Volatilitux Features

- +/- l'équivalent de Volatility pour Linux
- Framework modulaire et évolutif
  - Multi-architectures (ARM, x86 avec/sans PAE)
    - Limité au 32 bits pour le moment
  - Multi-versions (Linux 2.6.x)
  - Commandes
    - pslist, memmap, filelist, memdmp, filedmp
- Développé en Python
- Toujours en développement
  - Release 0.1 à venir

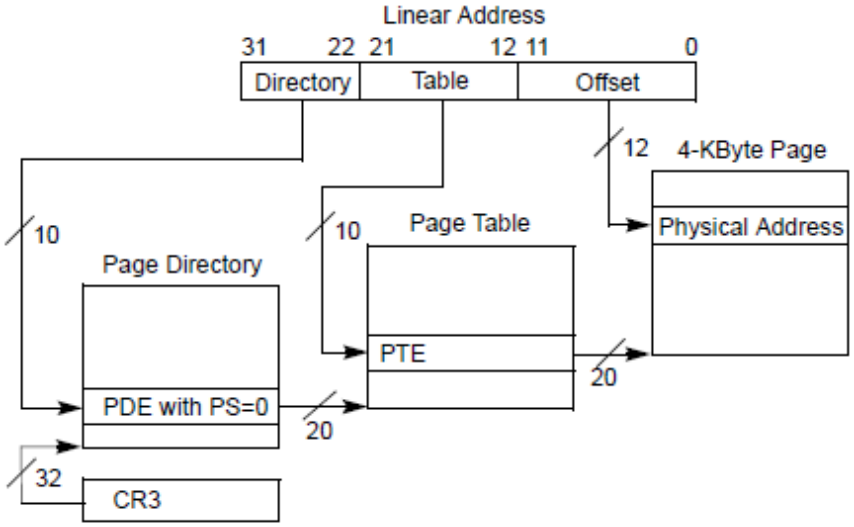
Volatilitux Internals

- Reconstruction de la mémoire virtuelle
  - Noyau (kernelland) :
    - Mappé pour tous les processus (mémoire partagée)
    - Adresses virtuelles >= oxCooooooo (*PAGE_OFFSET*)
      - Adresses physiques <= ox4ooooooo
    - Mappé sur une zone contigüe ☺
      - phys_addr = virt_addr – oxCooooooo

    ➔ Facile !

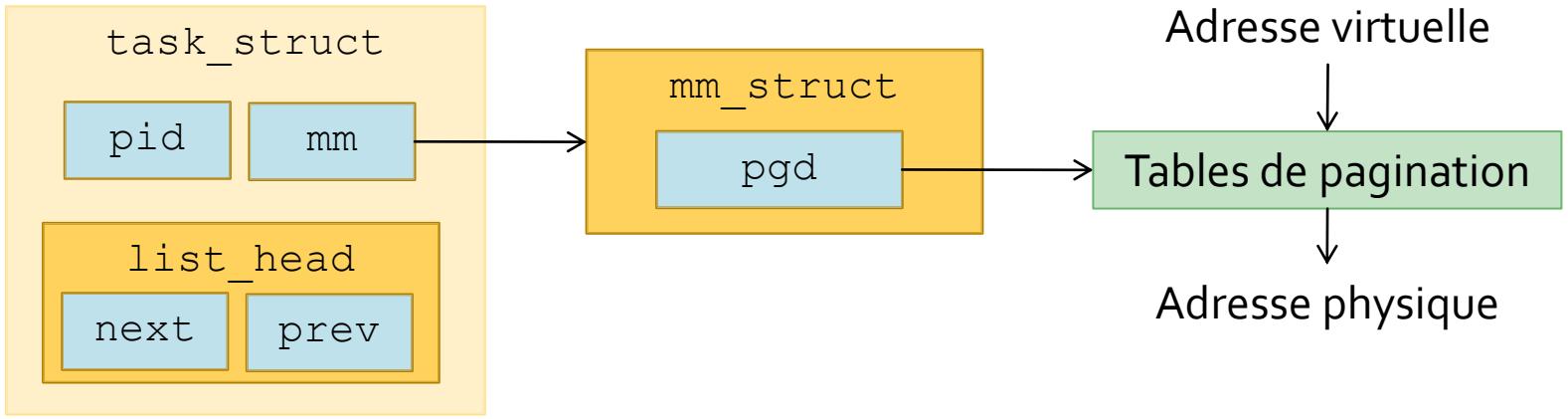
Volatilitux Internals

- Processus (userland)
  - Adresses virtuelles <= oxCooooooo
  - Adresses physiques… ?
    - Registre du CPU permettant de trouver les tables de pagination
  - 1 processus = 1 espace virtuel = 1 registre CR3 (pour x86)
    - Sauvegardé en mémoire ☺



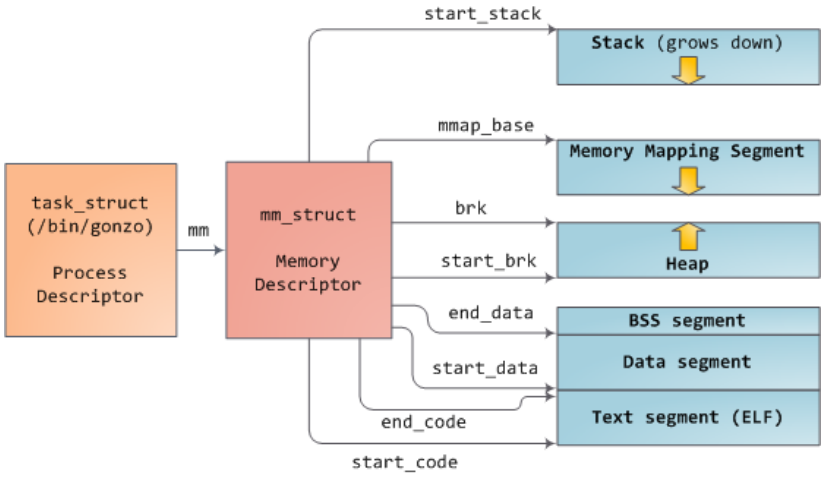
Volatilitux Internals

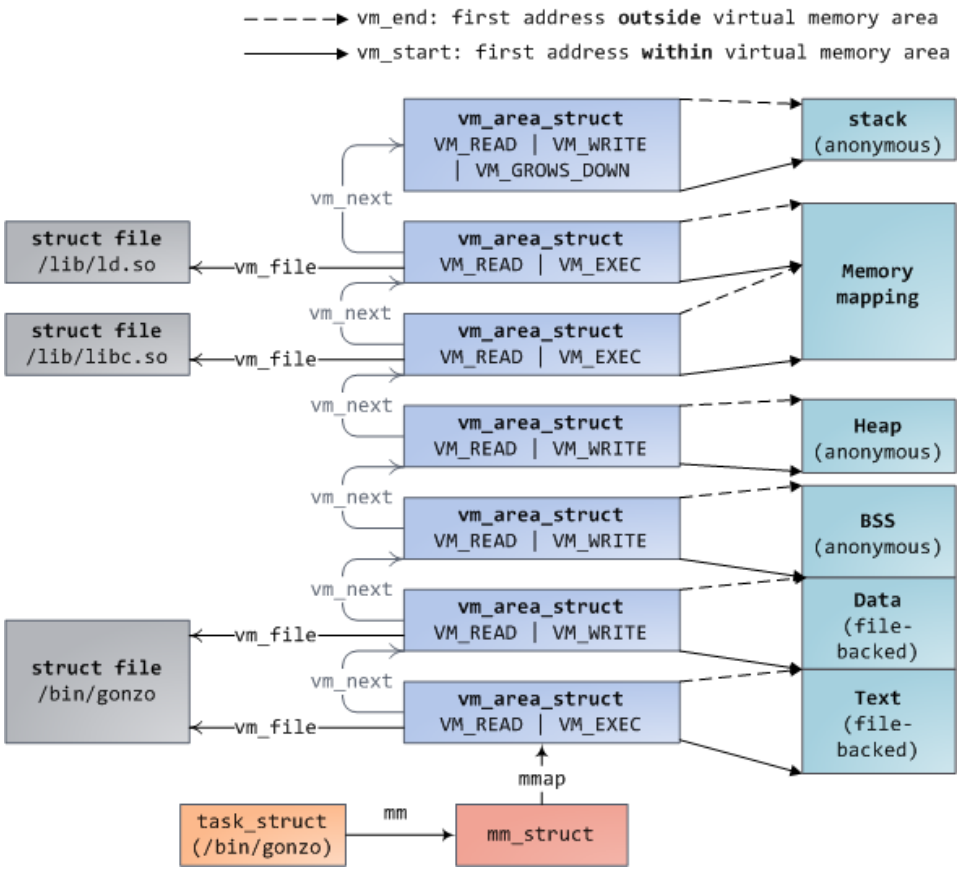
- 1 processus = 1 structure *task_struct*
  - Forment une liste doublement chaînée
    - Structure *list_head* avec 2 pointeurs : *next* et *prev*
  - Possède un pointeur vers une structure *mm_struct*
    - Chaque *mm_struct* possède un champ *pgd* (copie de CR3)



Volatilitux Internals

- *mm_struct* décrit la mémoire d'un processus
  - Zones mémoire
  - Fichiers mappés





Volatilitux Internals

- Nécessité de parser toutes ces structures
  - Point d'entrée : *init_task*
    - Adresse de la 1ère *task_struct* (« swapper »)
- Problème majeur
  - Les offsets varient d'une version à une autre…
- Deux solutions
  - Approche manuelle
    - Module noyau à charger pour récupérer les offsets
  - Approche automatique
    - Détection des offsets à base d'heuristiques

Volatilitux Internals

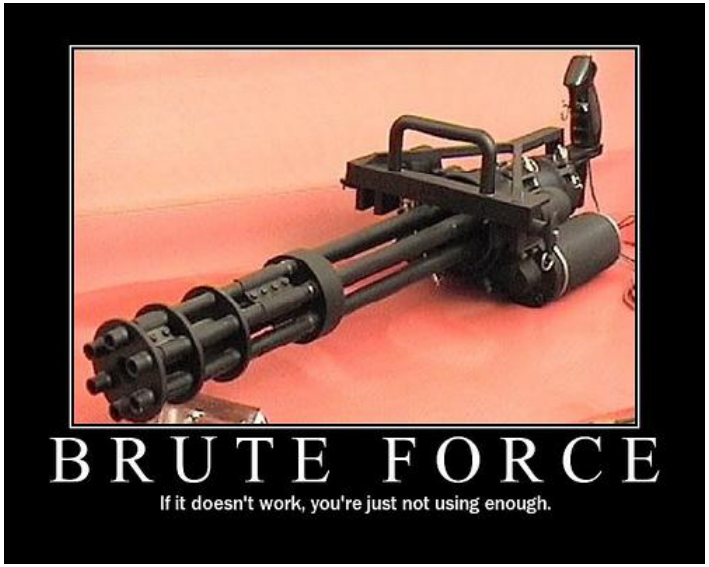
- Détection manuelle
  - Chargement d'un module noyau (LKM) sur la cible
    - Utilisation intensive de la macro offsetof()
  - Génération XML de la configuration
    - Récupération via dmesg (ou /var/log/messages)
  - Exemple :

```
########################################
# Volatilightux Confgen module loaded #
########################################
<config version = "1.0">

# To be completed
<arch name="..."/>
<init_task address="0xc033c300"/>
<struct name = "task_struct">
<field name = "pid" offset = "156" />
<field name = "comm" offset = "432" />
<field name = "tasks" offset = "96" />
<field name = "mm" offset = "120" />
<field name = "parent" offset = "168" />
</struct>
```

Volatilitux Internals

- Détection automatique d'offsets
  - Recherche exhaustive
    - Variable ?
      - Offset, adresse, architecture…
    - Jusqu'à 3 boucles imbriquées
      - S'arrêter dès que possible



  - Elagage par heuristique
    - Vérification d'un test devant être vrai
      - Ex: champs x et y égaux, champ x > oxCooooooo

Volatilitux Internals

- Algorithme simplifié
  - Récupérer la liste des *task_structs*
    - Recherche des chaînes « swapper » et « init » (champ *comm*)
    - Recherche du champ *parent* pointant vers leur début
    - Parcours de la liste des *tasks_structs*
  - Récupérer les offsets de *pid* et *mm* (*mm_struct)*
    - swapper.pid = 0 && init.pid = 1
    - swapper.mm = NULL && init.mm > oxCooooooo
  - Recherche du champ *pgd* et détection de l'architecture
    - Conversion d'une adresse virtuelle dont l'adresse physique est connue
  - Les autres offsets sont hardcodés pour le moment
    - Offsets assez stables selon les versions

En pratique

- Testé sur les distributions suivantes (installées par défaut)
  - Ubuntu 10.10 avec et sans PAE
  - Debian 5
  - Fedora 5 et 8 (CTF 4 et 5 LAMPSecurity)
  - CentOS 5 (CTF 6 LAMPSecurity)
  - Android 2.1 (Challenge SSTIC 2010)

En pratique

- Détection automatique
  - *pslist* fonctionne sur toutes les machines
  - Les autres commandes fonctionnent sauf sur :
    - Ubuntu, CTF4
      - Les offsets hardcodés ne sont pas si stables… Correctif à venir.
- Détection manuelle
  - Chargement du LKM et import du fichier XML
  - Toutes les commandes fonctionnent

Demo

- Challenge SSTIC 2010
  - Extraction des APKs
    - TextViewer ➔ pas de problèmes
    - Secret ➔ 2 pages ne sont plus mappées
      - Extraction manuelle en s'aidant d'un éditeur hexa et dd

Demo

```
D:\Challenges\SSTIC2010\concours_sstic_2010\volatilitux>volatilitux.py -f ..\challv2 pslist | tail
m.android.phone     98        30
d.process.acore     101       30
ndroid.settings     116       30
roid.alarmclock     136       30
d.process.media     147       30
com.android.mms     170       30
m.android.email     183       30
com.svox.pico       207       30
nssi.textviewer     227       30
om.anssi.secret     233       30

D:\Challenges\SSTIC2010\concours_sstic_2010\volatilitux>volatilitux.py -f ..\challv2 memmap -p 227 | grep apk
426f3000-426f8000 r-xs com.anssi.textviewer.apk
426f8000-426fa000 r-xs com.anssi.textviewer.apk
426fa000-426ff000 r-xs com.anssi.textviewer.apk
426ff000-42700000 r-xs data@app@com.anssi.textviewer.apk@classes.dex
427ba000-42c63000 r-xs framework-res.apk
42c63000-42e14000 r-xs framework-res.apk

D:\Challenges\SSTIC2010\concours_sstic_2010\volatilitux>volatilitux.py -f ..\challv2 filedmp -p 227 -t com.anssi.textviewer.apk -o out.apk
Dumping from 426f3000 to 426f8000...
20480 bytes dumped to out.apk

D:\Challenges\SSTIC2010\concours_sstic_2010\volatilitux>unzip -l out.apk
Archive:  out.apk
  Length     Date   Time    Name
 --------    ----   ----    ----
      784  09-04-10 16:00   res/layout/main.xml
     2678  09-04-10 16:00   res/raw/chiffre.txt
     1288  09-04-10 16:00   AndroidManifest.xml
     1660  09-04-10 15:58   resources.arsc
     3966  09-04-10 15:58   res/drawable-hdpi/icon.png
     1537  09-04-10 15:58   res/drawable-ldpi/icon.png
     2200  09-04-10 15:58   res/drawable-mdpi/icon.png
     3092  09-04-10 16:00   classes.dex
      636  09-04-10 16:00   META-INF/MANIFEST.MF
      689  09-04-10 16:00   META-INF/CERT.SF
      689  09-04-10 16:00   META-INF/CERT.RSA
 --------                   -------
    19219                   11 files

D:\Challenges\SSTIC2010\concours_sstic_2010\volatilitux>_
```

Conclusion

- Approche manuelle
  - ☺ Très efficace
  - ☹ Nécessite un accès (root) à la machine
    - ☹ Ou d'extraire sa configuration et recompiler le noyau
- Détection automatique
  - ☺ Simple et rapide
  - ☹ Encore imparfait pour certaines machines
    - ➔ Améliorations à venir…
- Intégration dans Volatility envisageable…
- Sources bientôt disponibles sur

www.segmentationfault.fr

Références

- Linux Cross Reference (code source du noyau)
  - http://lxr.linux.no
- Solutions du challenge SSTIC 2010
  - http://communaute.sstic.org/ChallengeSSTIC2010
  - http://pentester.fr/blog/index.php?post/2010/06/03/Challenge-SSTIC-2010-in-a-nutshell
- Gestion mémoire sous Linux
  - http://duartes.org/gustavo/blog/post/how-the-kernel-manages-your-memory
- Dynamic recreation of kernel data structures for live forensics (A. Case, L. Marziale, C. G. Richard)
  - http://www.dfrws.org/2010/proceedings/2010-304.pdf
- Manuels Intel et ARM
- Code source de Volatility

Questions ?