App vs Wild

Protection d'applications en environnement hostile

Stéphane Duverger

2 Juin 2016

AIRBUS
GROUP

Contexte

Objectif : protéger le code d'une application

- environnement hostile, non maintenu
- d'un noyau malveillant et d'elle-même
- sans modification, recompilation
- ni supposition sur le fonctionnement de l'OS
  - filtrage, classification des appels systèmes
  - identification, terminaison d'un processus

Contexte

Objectif : protéger le code d'une application

- environnement hostile, non maintenu
- d'un noyau malveillant et d'elle-même
- sans modification, recompilation
- ni supposition sur le fonctionnement de l'OS
  - filtrage, classification des appels systèmes
  - identification, terminaison d'un processus

Solution : virtualisation

- contrôler le CPU, la mémoire et les périphériques
- dépendances matérielles (x86)
- micro-virtualisation (déployable sur le *cloud*)
- fondée sur Ramooflax

Ramooflax 4tehwin

Rappel

- micro-hyperviseur de type 1 (*bare-metal*), à VM unique
- s'appuie sur Intel-VT et AMD-V
- présenté durant SSTIC 2011
- open-source `https://github.com/sduverger/ramooflax`

Ramooflax 4tehwin

Rappel

- micro-hyperviseur de type 1 (*bare-metal*), à VM unique
- s'appuie sur Intel-VT et AMD-V
- présenté durant SSTIC 2011
- open-source `https://github.com/sduverger/ramooflax`

Mode autonome

- sans interaction distante (*python, gdbstub, drivers*)
- dédié aux applications à protéger
- virtualiser la mémoire physique des applications

Vue d'ensemble de la protection

Que protège-t-on ?

- code, algorithmes
- pas les données
- ne gère pas les attaques avec accès physique

Vue d'ensemble de la protection

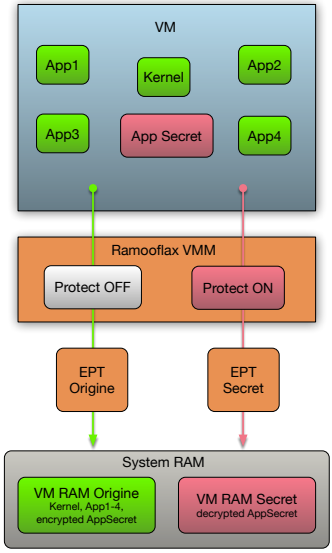
Que protège-t-on ?

- code, algorithmes
- pas les données
- ne gère pas les attaques avec accès physique

Principe

- applications déployées chiffrées (*toolchain*)
- présenter plusieurs vues de la mémoire physique
  - *Secret(s)*, pour les applications protégées
  - *Origine*, pour le reste
- Ramooflax s'occupe
  - du déchiffrement à la volée
  - des transitions d'environnements mémoire

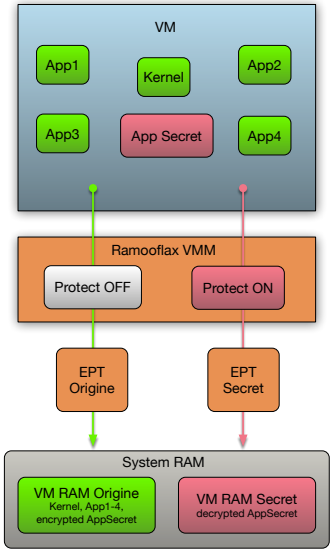
Proposer deux vues de la mémoire physique



Nested Page Tables

- virtualisation de la MMU
- Intel EPT et AMD RVI
- *mapping guest* vers *system physical*
- création dynamique d'EPT Secret
- déchiffrement à la demande

Proposer deux vues de la mémoire physique



Nested Page Tables

- virtualisation de la MMU
- Intel EPT et AMD RVI
- *mapping guest* vers *system physical*
- création dynamique d'EPT Secret
- déchiffrement à la demande

Propriété de sécurité

- page déchiffrée et intégrité vérifiée
  - marquée *eXecute-only*
  - Intel EPT uniquement
- autres pages marquées *RWX*
  - données, tas, pile
  - bibliothèques partagées
  - *gdt, idt, tss, pgd, ptb*

Changements de niveau de privilèges

Quand/comment basculer d'EPT ?

- si on schedule un processus *Secret* on installe son EPT dédié
- sinon on met en place l'EPT Origine
- identifier un processus *Secret* ?
- intercepter les transitions ring 3 <=> ring 0 ?

Changements de niveau de privilèges

Quand/comment basculer d'EPT ?

- si on schedule un processus *Secret* on installe son EPT dédié
- sinon on met en place l'EPT Origine
- identifier un processus *Secret* ?
- intercepter les transitions ring 3 <=> ring 0 ?

Identification d'un processus

- sans paradigme lié à l'OS (*thread info*, *task struct*, etc)
- un espace d'adressage par processus (*cr3* == *pgd*)
- filtrer les écritures dans *cr3*

Changements de niveau de privilèges

Quand/comment basculer d'EPT ?

- si on schedule un processus *Secret* on installe son EPT dédié
- sinon on met en place l'EPT Origine
- identifier un processus *Secret* ?
- intercepter les transitions ring 3 <=> ring 0 ?

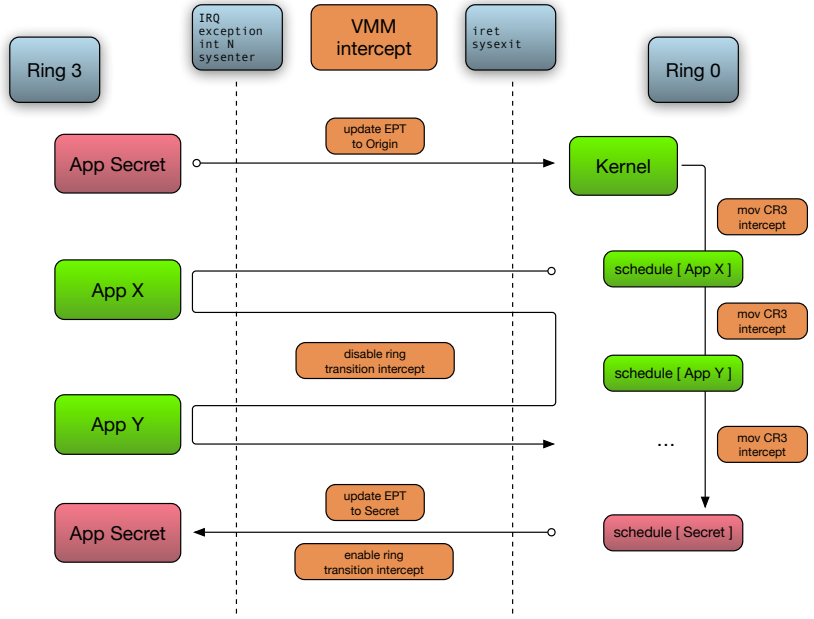
Identification d'un processus

- sans paradigme lié à l'OS (*thread info*, *task struct*, etc)
- un espace d'adressage par processus (*cr3* == *pgd*)
- filtrer les écritures dans *cr3*

Interception des transitions

- transition ring 3 vers ring 0
  - interruption matérielle *IRQ*
  - exception du processeur (*#PF*, *#BP*, etc)
  - appel système : *int N*, *sysenter*
- transition ring 0 vers ring 3
  - *iret*, *sysexit*

Intercepter les transitions de privilèges



Préparation d'une application Secret

Toolchain

- supporte binaires ELF32 statiques, dynamiques, PIE/PIC
- outil de génération de clefs

```
$ ./genkey.bin -c
AES         : 56fe883e73031f01a1765d57fb8f45034b19bc2f7837bed74ee0d4a4e2ba717e
IV          : d85d8c0c05b09170cd5486342d94250d
HMAC        : 0c617fe4fc90336ffc17e45a515b84c1cb1c9c4c6db6c479782ca83046985a62
```

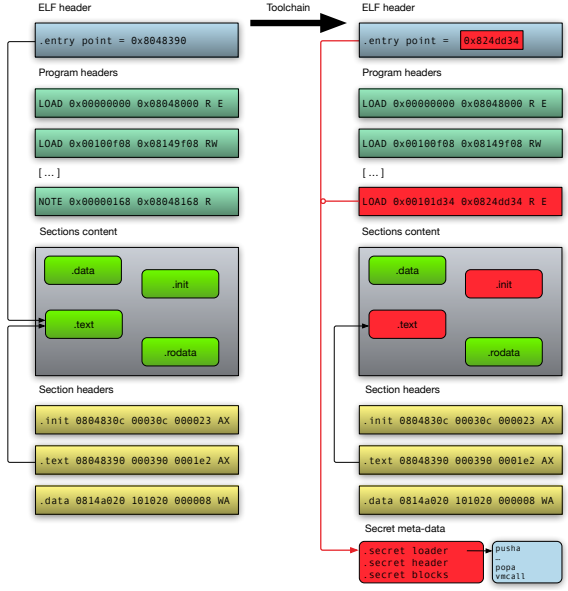
- outil de chiffrement/modification

```
$ ./elfix.bin
FAIL : missing arguments
usage : ./elfix.bin <elf> <hex aes key> <hex iv> <hex hmac key>

  <elf>            ELF binary file to encrypt
  <hex aes key>    hexadecimal string of AES KEY     (32 bytes)
  <hex iv>         hexadecimal string of AES CBC IV  (16 bytes)
  <hex hmac key>   hexadecimal string of HMAC key
```

Modifications apportées au binaire

- chiffrement des sections de code
- signature des éléments chiffrés
- modification d'un *program header*
- ajout de méta-informations
- ajout d'un pré-chargeur minimaliste
- modification du point d'entrée



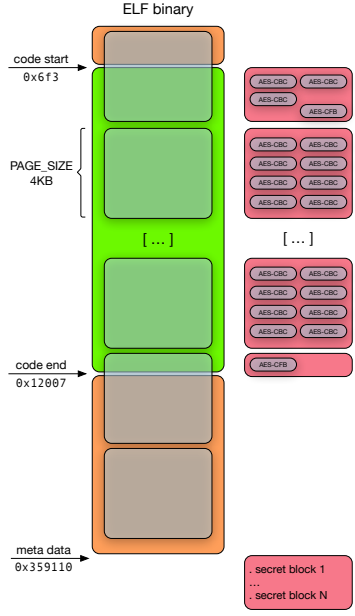
Chiffrement et signature des sections de code

- regroupe les sections de code par page(s) de 4KB
- facilite le déchiffrement à la demande
- crée des *secret_process_block*

```
typedef struct secret_process_block
{
    uint64_t size ;              // block size
    uint64_t offset ;            // from load base
    uint8_t  hash[HMAC_OUT_SZ] ; // hash of encrypted block

} __attribute__((packed)) sec_blk_t ;
```

- chiffrement AES-256-CBC/CFB
- signature SHA-256 HMAC



Injection et chargement des meta-data

Meta-data header

- spécifique au processus *Secret*
- à destination de l'hyperviseur
- *offsets* relatifs à l'adresse de chargement

```
typedef struct secret_program_header
{
    uint64_t  aep;    // actual ELF entry point
    uint64_t  oep;    // original ELF entry point
    uint64_t  end;    // secret info ending
    uint64_t  ret;    // sysenter caller
    uint64_t  sys;    // sysexit offset
    uint64_t  cnt;    // block count
    sec_blk_t blk[0]; // secret blocks

} __attribute__((packed)) sec_phdr_t;
```

Injection et chargement des meta-data

Meta-data header

- spécifique au processus *Secret*
- à destination de l'hyperviseur
- *offsets* relatifs à l'adresse de chargement

```
typedef struct secret_program_header
{
    uint64_t  aep;    // actual ELF entry point
    uint64_t  oep;    // original ELF entry point
    uint64_t  end;    // secret info ending
    uint64_t  ret;    // sysenter caller
    uint64_t  sys;    // sysexit offset
    uint64_t  cnt;    // block count
    sec_blk_t blk[0]; // secret blocks

} __attribute__((packed)) sec_phdr_t;
```

Modification d'un Program Header

- *PT_NOTE* présent mais optionnel
- remplacé par *PT_LOAD*
- référence toutes les meta-data (*header*, *blocks*)
- permet leur futur chargement

Injection et chargement des meta-data

Meta-data header

- spécifique au processus *Secret*
- à destination de l'hyperviseur
- *offsets* relatifs à l'adresse de chargement

```
typedef struct secret_program_header
{
    uint64_t  aep;    // actual ELF entry point
    uint64_t  oep;    // original ELF entry point
    uint64_t  end;    // secret info ending
    uint64_t  ret;    // sysenter caller
    uint64_t  sys;    // sysexit offset
    uint64_t  cnt;    // block count
    sec_blk_t blk[0]; // secret blocks

} __attribute__((packed)) sec_phdr_t;
```

Modification d'un Program Header

- *PT_NOTE* présent mais optionnel
- remplacé par *PT_LOAD*
- référence toutes les meta-data (*header*, *blocks*)
- permet leur futur chargement

Injection d'un pré-chargeur

- devient l'*entry point*
- force l'OS à mapper les meta-data
- exécute *vmcall*
- finalement saute sur l'*OEP*

Fonctionnement de l'hyperviseur

Compilation

- embarque les clefs cryptographiques (AES, IV, HMAC)
- la définition des *sec_blk_t*, *sec_phdr_t*
- indépendant des applications *Secret* (nombre, nature)

Fonctionnement de l'hyperviseur

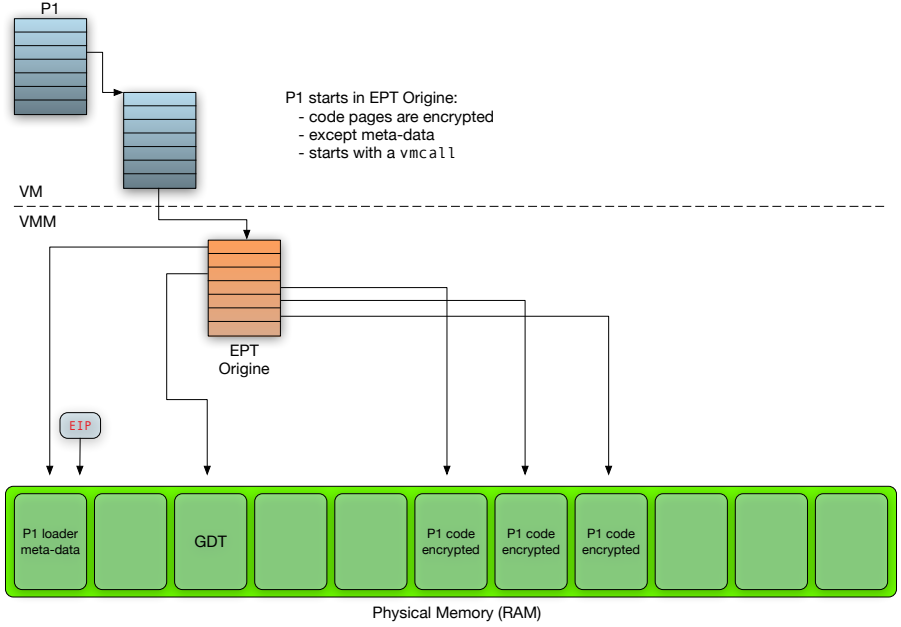
Compilation

- embarque les clefs cryptographiques (AES, IV, HMAC)
- la définition des *sec_blk_t*, *sec_phdr_t*
- indépendant des applications *Secret* (nombre, nature)

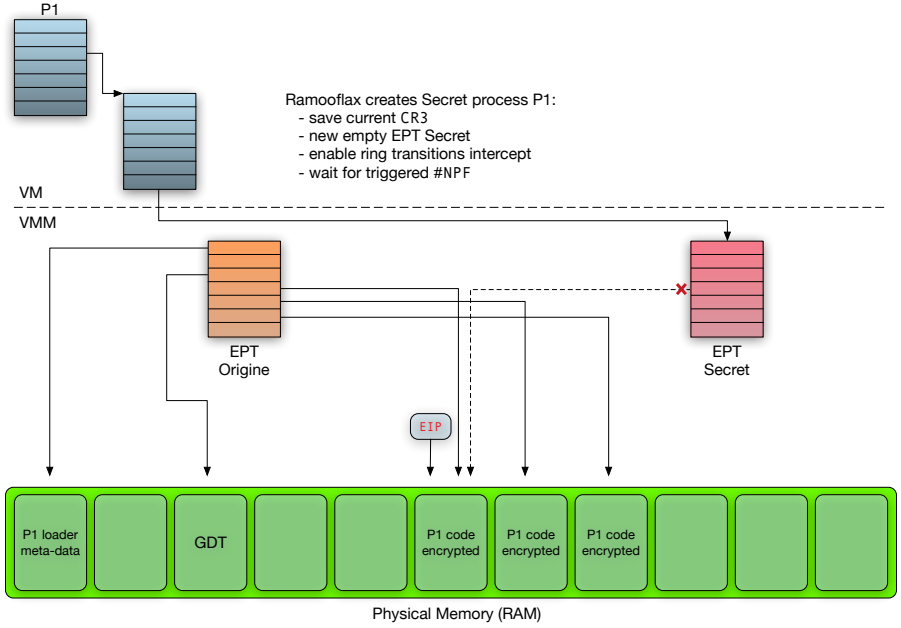
Gestionnaire d'évènements : *#VMEXIT*

- détection de processus *Secret*
- interception des écritures dans *cr3*
- traitement des *Nested Page Fault* (*#NPF*)
- interception des changements de privilèges (*#GP, #NP*)

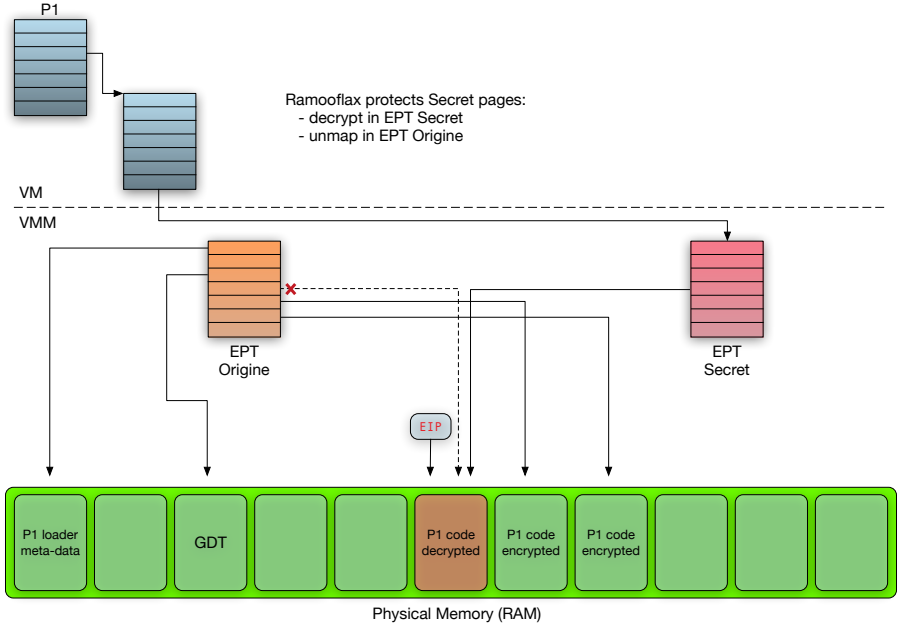
Création d'un processus *Secret*



Création d'un processus *Secret*



Création d'un processus *Secret*



Gestion des Nested Page Fault (*#NPF*)

En bref

- fautes de page au niveau EPT (*guest* vers *system* physique)
- surviennent après les *#PF* traditionnels
- différentes natures : read/write/execute/present
- utilisation de deux primitives : **sécuriser** et **rendre**

Gestion des Nested Page Fault (*#NPF*)

En bref

- fautes de page au niveau EPT (*guest* vers *system* physique)
- surviennent après les *#PF* traditionnels
- différentes natures : read/write/execute/present
- utilisation de deux primitives : **sécuriser** et **rendre**

Sécuriser une page

- validation du HMAC
- si le hash est invalide, **rendre** la page à la VM
- sinon marquage non présent dans EPT *Origine*
- déchiffrement *in-situ*
- marque la page *eXecute-only* dans EPT *Secret*

Gestion des Nested Page Fault (*#NPF*)

En bref

- fautes de page au niveau EPT (*guest* vers *system* physique)
- surviennent après les *#PF* traditionnels
- différentes natures : read/write/execute/present
- utilisation de deux primitives : **sécuriser** et **rendre**

Sécuriser une page

- validation du HMAC
- si le hash est invalide, **rendre** la page à la VM
- sinon marquage non présent dans EPT *Origine*
- déchiffrement *in-situ*
- marque la page *eXecute-only* dans EPT *Secret*

Rendre une page

- chiffrement *in-situ* si elle était déchiffrée
- on la marque *USER* | *RWX* dans EPT *Origine*
- on la marque non présente dans tous les EPTs *Secret* concernés (*reverse-mapping*)

Mécanisme d'interception

Basé sur la segmentation

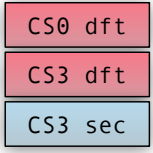
- Ramooflax parcourt la GDT et détecte les segments de code ring 0/3
- crée un nouveau descripteur de code ring 3 (*cs3 sec*)
- mise à jour des descripteurs de segment : provoquer $\#NP$

Mécanisme d'interception

Basé sur la segmentation

- Ramooflax parcourt la GDT et détecte les segments de code ring 0/3
- crée un nouveau descripteur de code ring 3 (*cs3 sec*)
- mise à jour des descripteurs de segment : provoquer *#NP*

Secret



Mécanisme d'interception

Basé sur la segmentation

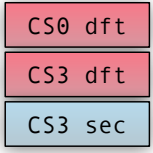
- Ramooflax parcourt la GDT et détecte les segments de code ring 0/3
- crée un nouveau descripteur de code ring 3 (*cs3 sec*)
- mise à jour des descripteurs de segment : provoquer *#NP*

Mécanisme d'interception

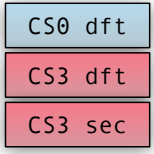
Basé sur la segmentation

- Ramooflax parcourt la GDT et détecte les segments de code ring 0/3
- crée un nouveau descripteur de code ring 3 (*cs3 sec*)
- mise à jour des descripteurs de segment : provoquer *#NP*

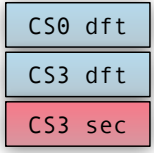
Secret



Noyau



Désactivé

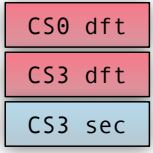


Mécanisme d'interception

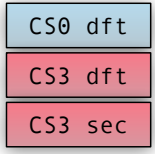
Basé sur la segmentation

- Ramooflax parcourt la GDT et détecte les segments de code ring 0/3
- crée un nouveau descripteur de code ring 3 (*cs3 sec*)
- mise à jour des descripteurs de segment : provoquer *#NP*

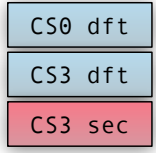
Secret



Noyau



Désactivé



Cas complexes

- terminaison de *Secret* et réutilisation de *cr3*
- détection implicite suite à un *fork()*
- *sysenter/sysexit* n'utilisent pas la GDT
- gestion des signaux, *clone()* qui modifient la pile noyau

État de la protection

Ayez confiance

- proriété de sécurité repose sur EPT *eXecute-only*
- *hardware cookies* assurent l'exécution des processus *Secret*
  - interception des changements de privilèges
  - segment spécial *cs3_sec*
  - *fork()* et trampoline *sysexit*
  - *Page Table magic*
- confidentialité et intégrité, pas disponibilité
- peu importe que les mécanismes soient détournés

État de la protection

Ayez confiance

- proriété de sécurité repose sur EPT *eXecute-only*
- *hardware cookies* assurent l'exécution des processus *Secret*
  - interception des changements de privilèges
  - segment spécial *cs3_sec*
  - *fork()* et trampoline *sysexit*
  - *Page Table magic*
- confidentialité et intégrité, pas disponibilité
- peu importe que les mécanismes soient détournés

Limitations

- pas d'I/O MMU donc AMT (ring -3), DMA, etc
- mode SMM > VMX
- *side-channels*
- *jit-compiler* et *self-modifying* code
- processeurs AMD (SEGMEXEC ?)

Scénario d'attaque

Prédiction d'instructions

- tentatives infinies
- analyser les effets sur les GPRs
- instruction par instruction

Scénario d'attaque

Prédiction d'instructions

- tentatives infinies
- analyser les effets sur les GPRs
- instruction par instruction

Contre-mesures

- filtrer *#BP*, *#DB*
- sauver/effacer/restorer les GPRs avant IRQ et exception
- difficile pour les appels systèmes
- est-il possible de deviner Insn X sachant :
  - fixe GPRs(E1)
  - Insn X
  - N Insn connues
  - *int 0x80* ou *sysenter/syscall*
  - lit GPRs(E2)

Évolutions et Performances

John The Ripper 1.8.0

```
$ john-1.8.0/run/relbench john.clear.bench john.sec.bench
Number of benchmarks : 13
Minimum : 0.93487 real, 0.93500 virtual
Maximum : 1.04503 real, 1.04293 virtual
Median : 0.98855 real, 0.98333 virtual
Median absolute deviation : 0.02517 real, 0.01947 virtual
Geometric mean : 0.98990 real, 0.98371 virtual
Geometric standard deviation : 1.03304 real, 1.03151 virtual
```

Évolutions et Performances

John The Ripper 1.8.0

```
$ john-1.8.0/run/relbench john.clear.bench john.sec.bench
Number of benchmarks : 13
Minimum : 0.93487 real, 0.93500 virtual
Maximum : 1.04503 real, 1.04293 virtual
Median : 0.98855 real, 0.98333 virtual
Median absolute deviation : 0.02517 real, 0.01947 virtual
Geometric mean : 0.98990 real, 0.98371 virtual
Geometric standard deviation : 1.03304 real, 1.03151 virtual
```

Quelques idées

- Windows, PE, ELF64, *syscall/sysret*
- toolchain collaborative
  - sections spécifiques à chiffrer
  - sans appels systèmes
- protéger des bibliothèques partagées

Merci ! Greetz

aux collègues pour les idées, la patience

... à ma compagne enceinte de jumelles : 35 SA, *man fork*

Challenge

`https://github.com/sduverger/AppVsWild`