Sécurité et Systèmes d'exploitation

François Armand
M2 / Paris7

Quelques références

- Sécurité des noyaux de systèmes d'exploitation, Thèse Eric Lacombe.
  - http://tel.archives ouvertes.!r/docs/" "/#$/%&/ ' #/( ) */these eric+lacombe.pd!

Quelques références

- Slap6S;: http:// / / .o / %.or0/vie / /=ctivities) ashboard/Slap6S
- http:// / / .ssi.0ouv.!r/archive/!r/sciences/!ichiers/rapports/rapport +orientation+ssi+%" ">.pd!
- http:// / / .invisiblethin0slab.com/
- http:// / / .sstic.or0/
- Trusted ,omputin0 ?roup: http:// / / .trustedcomputin00roup.or0/
- ,rypto(a0e/@>$. ?uillaume)uc. http://enstb.or0/A3eryell/publications/con!/%" "&/Symp=/article.pd!
- http:// / / .polyxene.!r/
- http://pax.0rsecurity.net/
- http:// / / .ertos.nicta.com.au/research/l#.veri!ied/

Quelques exemples

- Stuxnet
- TDL4
  - 4,5 millions de machines infectées
  - Utilise communications encryptées P2P (Kadmelia)
  - Combat les autres virus
  - Infecte le MBR de la machine
- ILOVEYOU

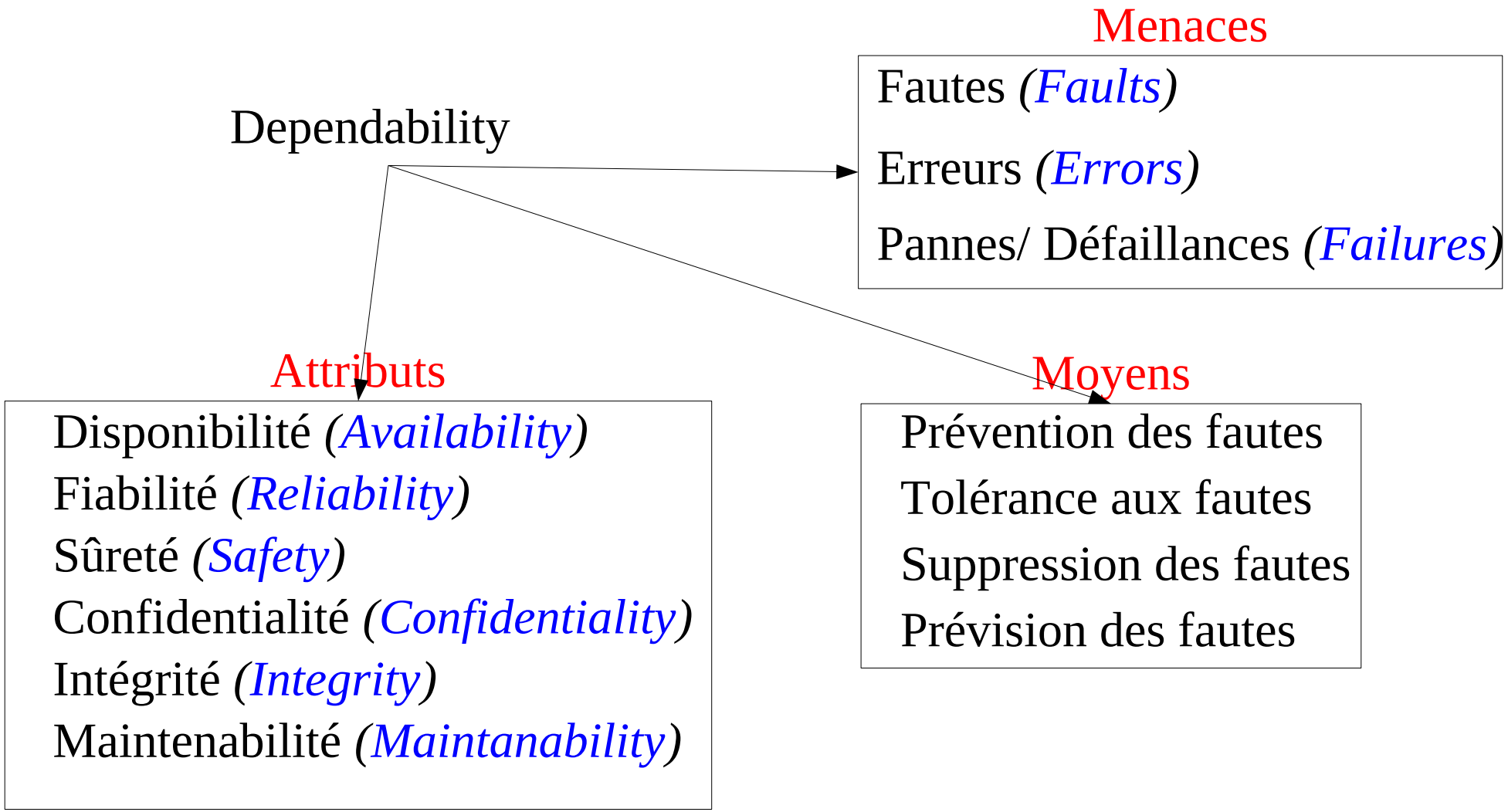
Exemples suite

- Bugzilla : 7 March 2012
  - Bugzilla does not properly validate form attributes passed to xmlrpc.cgi, enabling cross-site request forgery attacks.
- Gnash : 7 March 2012
  - The gnash flash player stores cookies in world-readable files with predictable names.
- Httpd : 7 March 2012
  - The Apache HTTPD server is subject to denial-of-service attacks using partial requests.

Exemples suite

- Stunnel : 1/03/2012 *(tunnel ssl)*
  - The vulnerability may possibly be leveraged to perform remote code execution or a Denial of Service attack.
- Spamdyke : 06/03/2012 *(filtre anti spam)*
  - Boundary errors related to the "snprintf()" and "vsnprintf()" functions in spamdyke could cause a buffer overflow. A remote attacker could possibly execute arbitrary code or cause a Denial of Service.

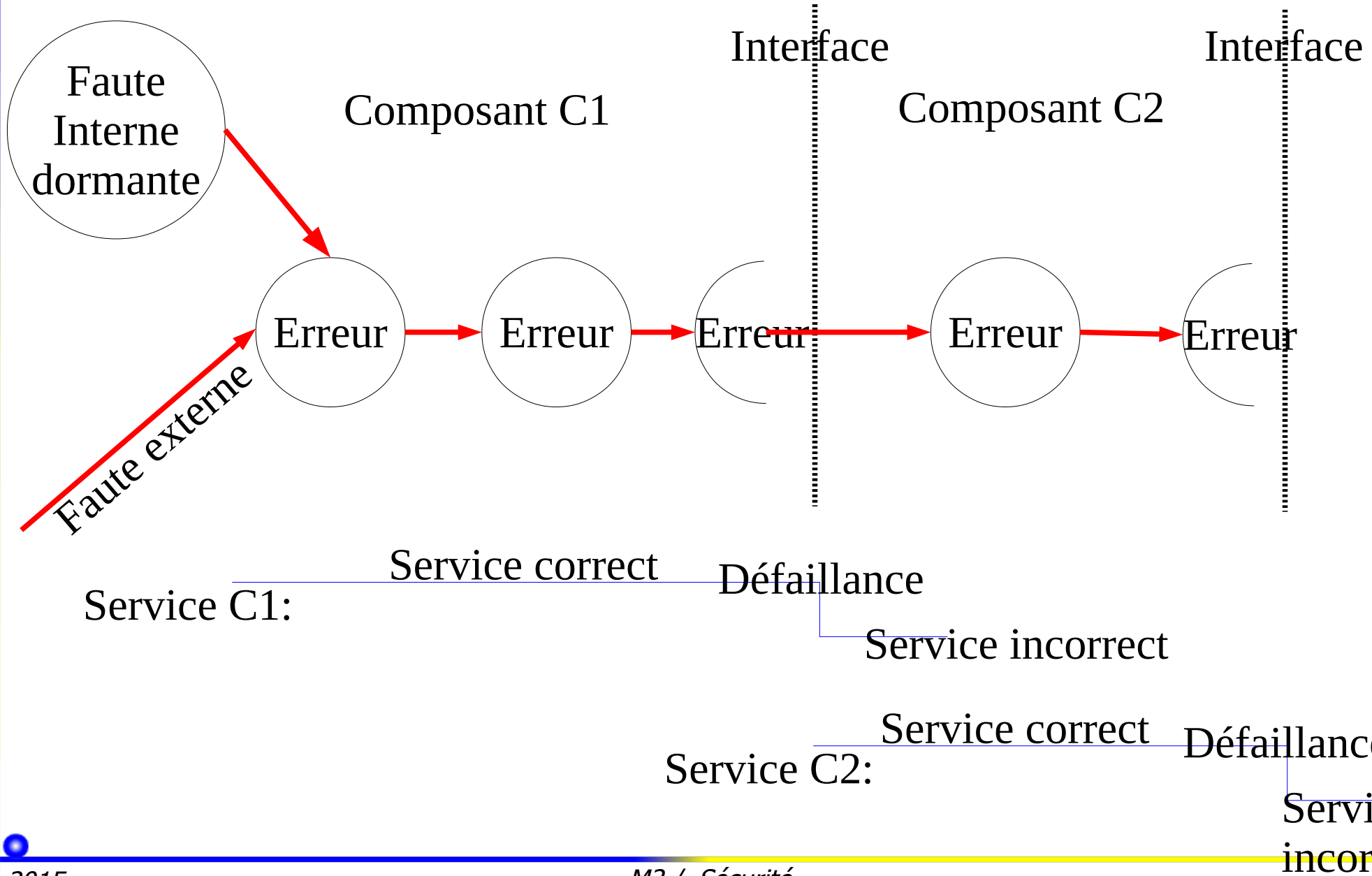
Dépendance *(Dependability)*



Menaces

- Faute:
  - Défaut physique du matériel ou du logiciel
- Erreur:
  - Une valeur incorrecte dans le système
- Défaillance / Panne:
  - Déviation du système par rapport à sa spécification

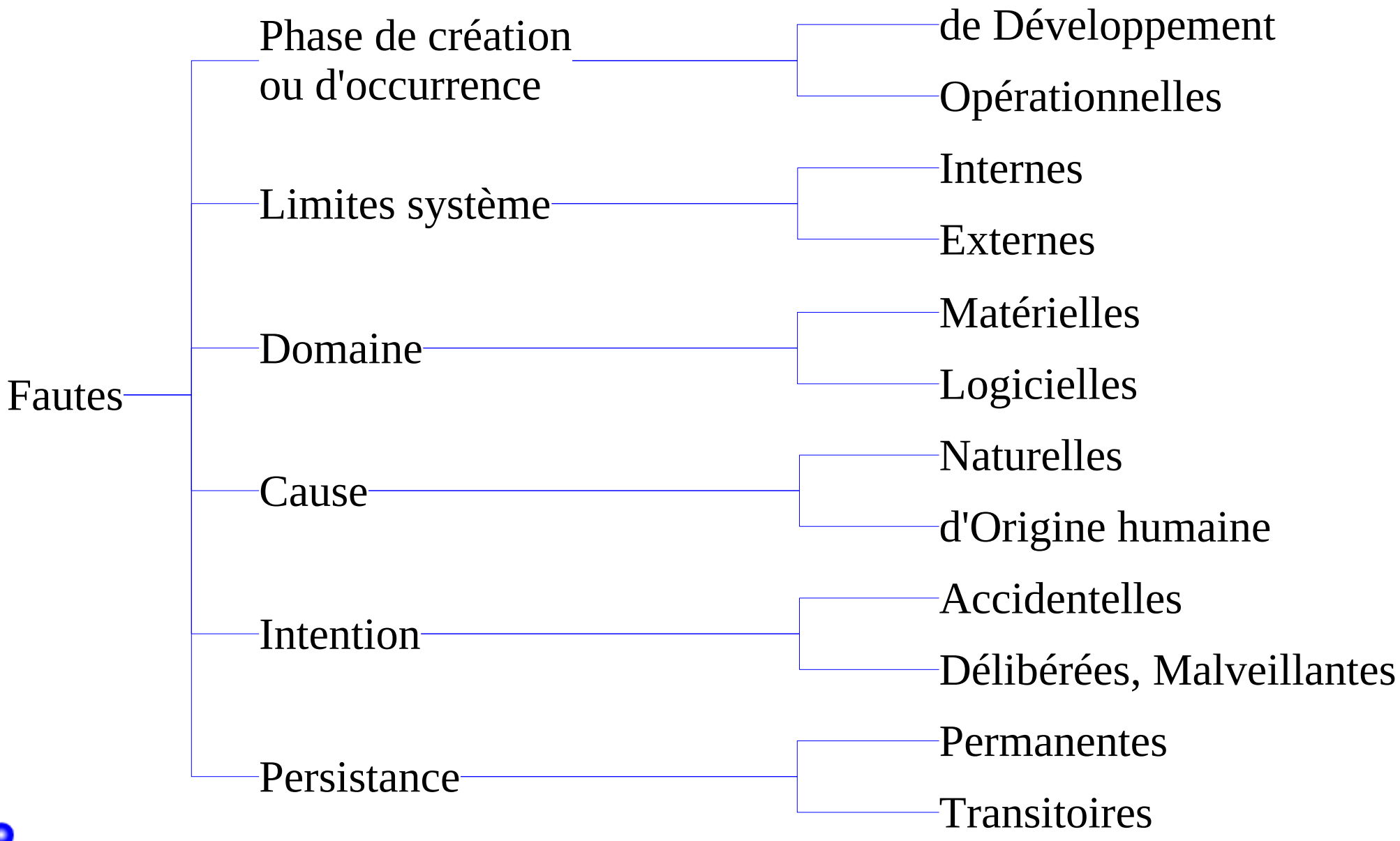
Menaces


Faute
Interne
dormante
Composant C1
Interface
Composant C2
Interface
Erreur
Erreur
Erreur
Erreur
Erreur
Faute externe
Service C1:
Service correct
Défaillance
Service incorrect
Service C2:
Service correct
Défaillance
Service
incorrect

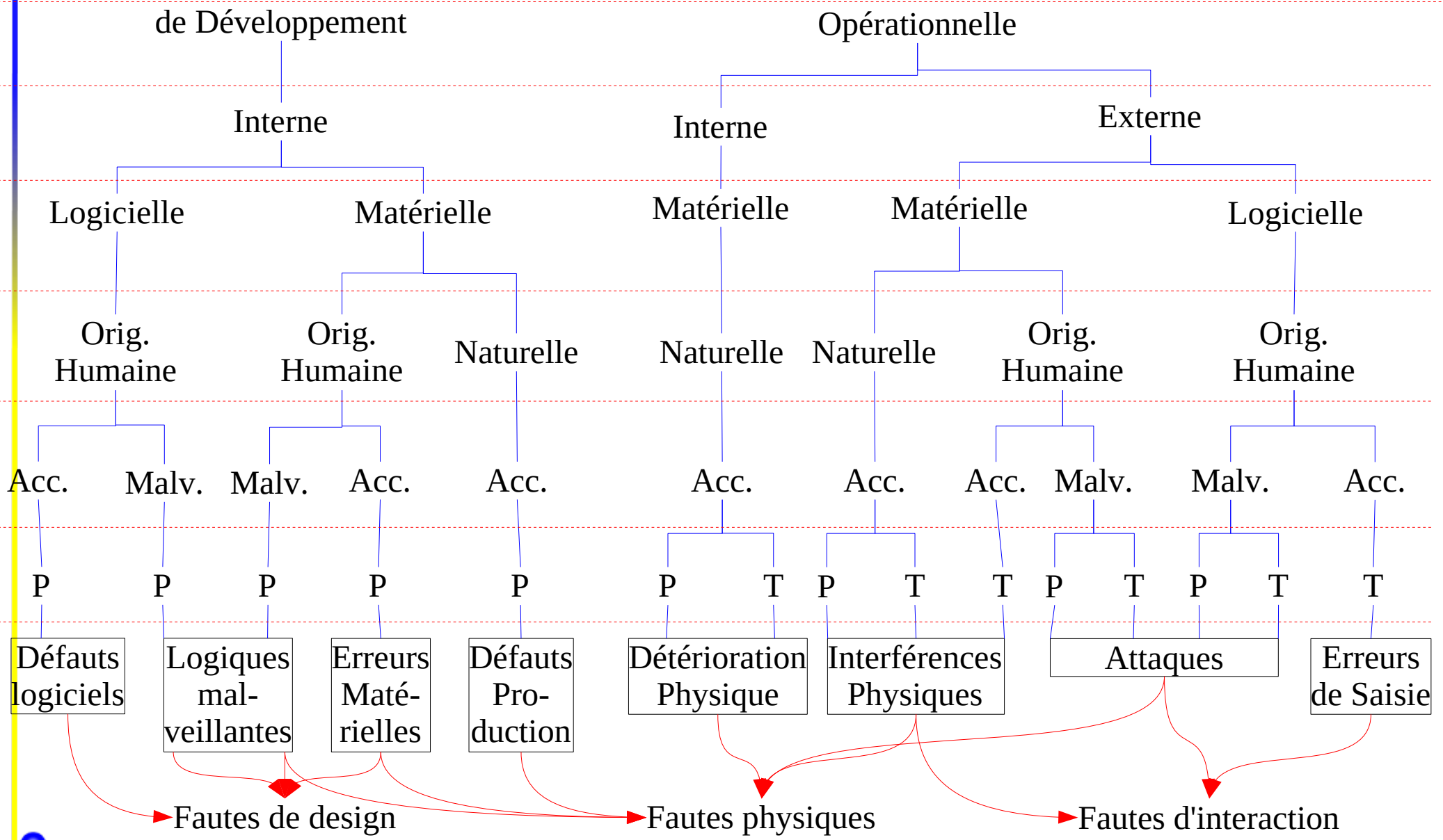
Modes de Défaillance

- Domaine:
  - Valeur erronée, défaillance temporelle
- Perception par plusieurs utilisateurs:
  - Défaillance cohérente ou incohérente
- Conséquences:
  - Mineures
  - ....
  - Catastrophiques

Classes Élémentaires de Fautes



Classes Combinées de Fautes



Attributs

- Fiabilité:
  - Fournir un service correct de manière continue
- Sûreté:
  - Absence de conséquences catastrophiques
- Maintenabilité:
  - Capacité à supporter des modifications et des réparations

Attributs

- Confidentialité:
  - Ne pas dévoiler des informations sans autorisation
  - Donc empêcher
    - Utilisateur non autorisé d'accéder à une information à laquelle il ne devrait pas avoir accès,
    - Utilisateur autorisé de transmettre une information à un tiers non autorisé

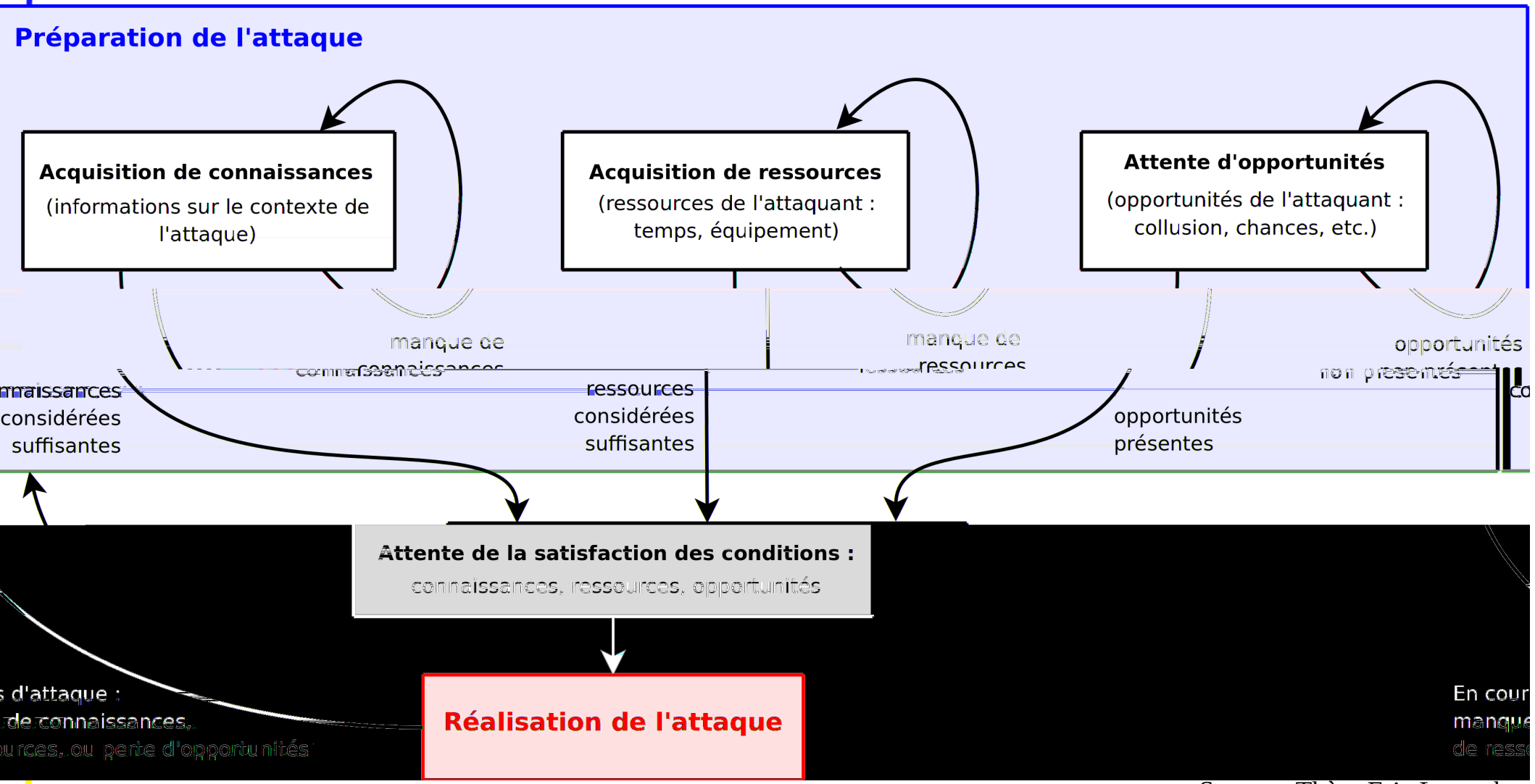
Attributs

- Intégrité:
  - Absence d'altérations incorrectes de l'état du système
  - Empêcher les modifications illégitimes
    - Par des utilisateurs non autorisés
    - Ou par des utilisateurs autorisés
  - Empêcher qu'on puisse empêcher les modifications légitimes

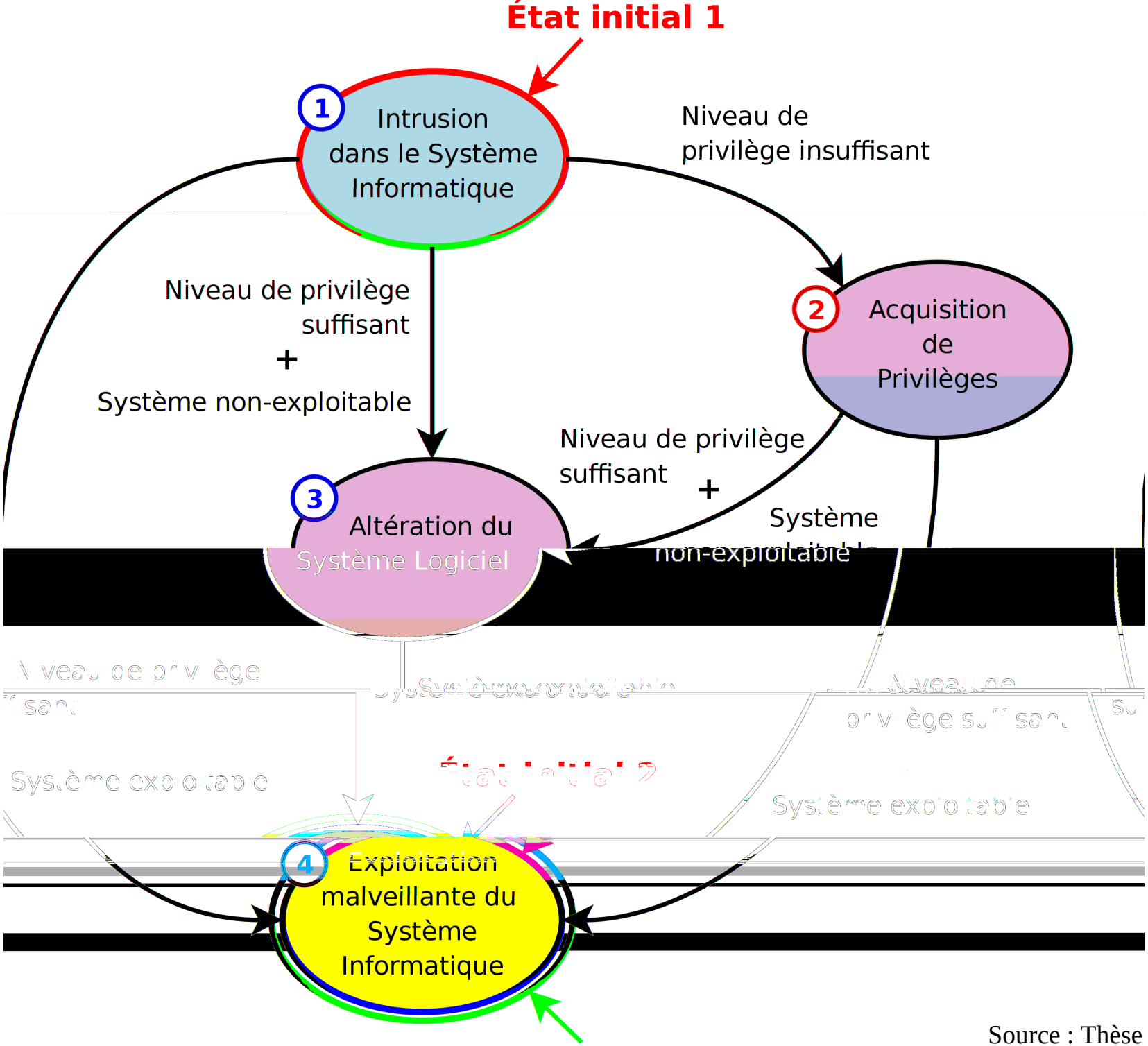
Attributs

- Disponibilité:
  - Fournir accès à un utilisateur autorisé quand il en a besoin
    - En lecture et/ou en écriture
  - Empêcher qu'on puisse empêcher un tel accès
    - Empêcher les « dénis de service »

Processus de résolution du problème de l'attaquant



Source : Thèse Eric Lacombe



Source : Thèse Eric Lacombe

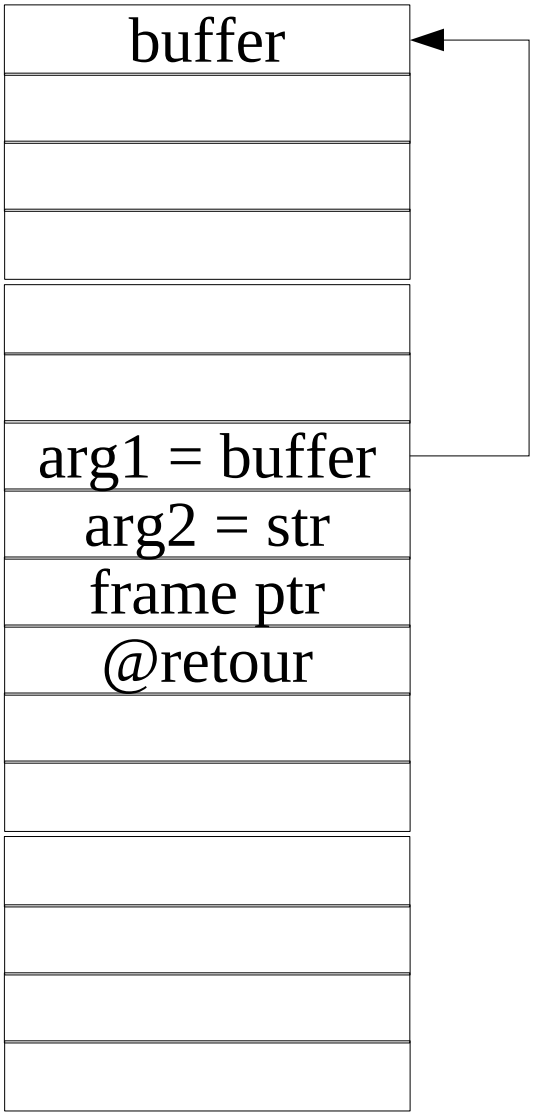
Quelques Attaques

- Buffer Overflow
- Integer Overflow
- Return to libc
- Denial Of Service

Débordement de tampon (pile)

```
char buffer[20] ;
strcpy(buffer, str);
```

- Si strlen > 20
  - Débordement buffer
  - Écrasement @retour
  - => saut vers code choisi par l'attaquant.



Débordement pile : exemple

```
#include <string.h>
 !id f!!(c!nst char" buf) #
    char buffer[$00];
    strcpy(buffer, buf);
%
int &ain(int argc, char "arg []) #
    if(argc > 0) f!!(arg [$]);
    return 0;
%
```

Débordement pile : exemple

```
# ./vuln `perl -e 'print "A"x200;''
(rreur de seg&entati!n (c!re du&ped)
# gdb  uln
...
)!re *as generated by '.+ uln
,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
,,,,,,,,,,,,,,,,,,,,,,-.
.r!gra& ter&inated *ith signal $$, /eg&entati!n
fault.
#O O01$1$1$1$ in 22 ()
(gdb) p 3ebp
3$ 4 ( !id ") O01$1$1$1$
(gdb) p 3eip
32 4 ( !id ") O01$1$1$1$
(gdb) 0+$O0 3esp
O0bfa215206 O01$1$1$1$ O01$1$1$1$ O01$1$1$1$
O01$1$1$1$
```

Débordement pile : exemple

- Générer une chaîne à passer en argument telle que :
  - Adresse de retour soit une adresse valide vers du code
    - Déjà présent
    - Mieux : vers du code amené dans la chaîne !
- Problèmes :
  - Convertir du code en chaîne de caractères
  - Pas de caractères nuls
  - Trouver l'adresse (sans caractère nul)

Déroutement vers du code déjà présent

```
void bar(void)
#
   printf(78ac9ing ,tte&pt.:n7);
%

void foo(char * arg)
#
   int i;
   char buffer[100];
   strcpy(buffer, arg);
   i 4 0;
   printf(7;uit:n7);
%

int main(int argc, char *argv[])
#
   f!!(arg [$]);
   return 0;
%
```

Déroutement vers du code déjà présent

- Récupérer l'adresse de la fonction « bar »
  - `# n& < grep bar`
  - `0=01=2$0 > bar`
- Préparer une chaîne contenant cette adresse au « bon endroit »

```
# .+ uln2 'perl ?e -print 7,70$0=;
                         print 7:0$0:0=2:001:00=7:-'

;uit

8ac9ing ,tte&pt.

/eg&entati!n fault
```

La fonction bar a été exécutée !

Déroutement vers du code fourni

```
@A>/ 52
gl!bal Bstart
seg&ent .te0t
Bstart6
        C&p sh!rt d!nnees
r&dir6  p!p eb0
        0!r ea0, ea0
        &!  al, 0x28 ; syscall r&dir
        int 00=0
        0!r ea0, ea0
        &!  al, 0x01 ; syscall e0it
        0!r eb0, eb0
        int 00=0
d!nnees6  call r&dir
        n!& db 7testdir7
```

Écriture du code à injecter
- rmdir ./testdir
- exit 0

Fichier : shellcode.nasm

Vérification du code malicieux

- nasm shellcode.asm, transformation, gcc, execution

```
char sc[] 4
7:0eb:00f:0Db:05$:0c0:0b0:02=:0cd:0=0:05$:0
c0:0b0:00$:05$:0db:0cd:0=0:0e=:0ec:0ff:0ff:
0ff:0E1:0FD:0E5:0E1:0F1:0FG:0E27;

int &ain(int argc, char ""arg )
#
   int ("shellc!de)() 4 (int (")())sc;
   shellc!de();
   return 0;
%
```

Étapes suivantes

- Chaîne testdir pas terminée par un caractère nul
  - On peut générer la chaîne et le caractère nul à l'exécution du code malicieux
- Fabriquer un « buffer » qui
  - Contienne le code malicieux
  - L'adresse de ce code
    - On ne veut plus aller exécuter une fonction prédéfinie (bar)
    - Ce code sera dans le buffer « écrasé », ici dans la pile.
    - Trouver l'adresse du buffer (sans caractère nul)
    - Mettre des nop.

Integer Overflow

- Extrait OpenSSH (2.9.9 à 3.3)

```
++ e0trait un entier d-un paHuet reIu
nresp 4 pac9etBgetBint();
if (nresp > 0) #
    ++ all!ue un ta&p!n de6 nresp " 1 !ctets
    resp!nse 4 0&all!c(nresp"siJe!f(char"));
    f!r (i 4 0; i > nresp; iKK)
        resp!nse[i] 4 pac9etBgetBstring(LMNN);
%
```

- Si nresp = 1 073 741 824
  - Malloc (0) => buffer / heap overflow

Débordement de Tas

- Écrasement de fonction de type « .dtors »
  - Effectif seulement lors de l'arrêt du processus
- Utilisation de séquences de copie de la libc
  - Par exemple : malloc / free
  - Si les pointeurs ont été écrasés...

Déni de Service

- Consommation de ressources
  - Ex : Bande passante réseau, espace disque, temps CPU, mémoire, ressources systèmes (fork bomb...)
- Changement configuration
  - Ex : Routage réseau
- Changement état
  - Ex : Reset connexions TCP
- Attaque composants physiques
- Obstruction canaux de communication

Réponses possibles

- Vérifications compilateur
  - Ex : Débordement de tableaux
  - Ex : Utilisation de « canaris »
    - `gcc ?f stac9?pr!tect!r`
- Contrôle mémoire
  - Ex : Espaces adressages distincts
  - Pas d'exécution des régions de données
- Limitations utilisation ressources
  - Ex : Par processus, par utilisateur..
- Contrôle anti intrusion
  - Firewall, inspection paquets...

Gestion Mémoire

- Protection des régions mémoires
  - Lecture, Écriture
  - Exécution
    - Support matériel
      - relativement récent sur x86 (XD – Intel, NX – AMD)
      - XN ARM v6
    - Permet interdire exécution de données :
    - Tas et pile : empêche certaines attaques de « buffer overflow » (Sasser, Blaster)

Allocations Mémoire

- Mettre les mémoires allouées à zéro
  - Évite certains canaux cachés
- Gestion des caches (L1, L2..)
  - Lors des changements de contexte de processus (ordonnancements)

Gestion Mémoire

- Linux Exec Shield
  - Émulation de la protection en exécution
  - Basé sur limite du segment de code (registre CS)
    - Systèmes Fedora et RedHat
  - Pas inclus dans kernel.org
- Linux PaX
  - Similaire
- Windows DP
  - Data Execution Prevention
- OpenBSD
  - W^X

ASLR

- Address Space Layout Randomization
  - Idée : ne pas placer les régions mémoires toujours à la même adresse
  - Complexifie la tâche des attaques
  - Réponse statistique
    - Il faut que l'entropie soit suffisamment grande pour que la protection soit efficace (mais pas garantie)
  - Pile, tas, librairies, mmap
  - Éventuellement code et données... si Code « PIC »

Altération Mémoire Noyau

- Via un module noyau ou un pilote de périphérique (chargé dynamiquement)
- Accès à la mémoire via /dev/kmem ou /dev/mem
- Débordement de tampon
- Chaîne de format
- Données incorrectes, déréférencement pointeur incorrect...
- Via accès DMA
- Accès direct pépriphérique / périphérique (PCI)

Utilisateurs (Unix)

- Utilisateurs
  - Représentés par des « credentials »
  - Identifiant Utilisateur :
    - Reél / Effectif
  - Identifiant(s) de groupe(s)
    - Réels / Effectifs
    - Un groupe primaire + des groupes additionnels (optionnels)
  - Identifiants sur 32 bits *(15 à l'origine)* stockés dans
    - /etc/passwd
    - /etc/group
- Droits universels à « root »: (uid =0)

/etc/passwd

- Historiquement : jusqu'en 1988

Terminal de Contrôle

- Ne pas le lire depuis stdin !!!
  - Permettrait de stocker le mdp en ligne
- Acquisition depuis le terminal de contrôle du processus
  - /dev/tty
  - Associé à la « session » (en pratique au terminal ayant servi à la connexion utilisateur)
  - Utilisé aussi pour les signaux générés depuis le clavier : ^C, ^\, ^Z...

Connexion au système

- Demander systématiquement le mot de passe sans même vérifier si l'utilisateur est connu
  - Empêche de différencier les utilisateurs connus, des utilisateurs inconnus
- Limiter le nombre de tentatives d'accès infructueuses
  - Temporisation (longue) après 3 échecs consécutifs
  - Prévenir des échecs successifs

Droits utilisateurs

- Associés aux fichiers
  - rwx rwx rwx
    - Lecture / Écriture /
      - Exécution : binaires exécutables, librairies, scripts...
      - Traversée / recherche dans les répertoires
  - Certains système de fichiers gèrent aussi des ACL
    - Access Control List
- Associés aux processus
  - Envoi de signaux, debug

ACL (linux)

- Permet une gestion plus fine des droits d'accès
  - CONFIG_EXT3_FS_POSIX_ACL=y
  - mount -o remount,acl /dev/hda1
  - getfacl foo
    - `# file6 f!!`
    - `# !*ner6 franc!is`
    - `# gr!up6 franc!is`
    - `user66r*?`
    - `gr!up66r??`
    - `&as966r??`
    - `!ther66r??`

ACL (linux)

- Ajouter des droits à un utilisateur
  - `setfacl ?& u6la&bdaMser6rx f!!`
  - `03 getfacl f!!`
  - `# file6 f!!`
  - `# !*ner6 franc!is`
  - `# gr!up6 franc!is`
  - `user66r*?`
  - `user6la&bdaMser6r-x`
  - `gr!up66r??`
  - `&as966r?0`
  - `!ther66r??`

Set user / group Id

- Changer d'utilisateur effectif pendant la durée d'exécution d'un programme :
- Exemple : passwd

```
?r?sr?sr?0 $ r!!t sys 2E211 ,ug $0 20$0 +bin+pass*d
```

  - Permet d'avoir les droits de root et donc de modifier le mot de passe stocké dans un fichier inaccessible à l'utilisateur normal

Autorisations permanentes

- Permettre à certains utilisateurs d'effectuer des commandes réservées à root, sans leur accorder un privilège universel (shell)
- `sud!`
  - Permet à des utilisateurs pré-enregistrés dans (`+etc+sud!ers`) d'exécuter des commandes également pré-enregistrées

Changement de propriétaire

- ch ! * n
  - Dans les faits, réservé à « root »
    - Limite les possibilités de « Cheval de Troie »
  - Remet les bits suid, sgid à 0 si effectué par non root
  - Rien n'est dit si effectué par root

« Sticky bit »

- Répertoire /tmp
- `dr*0r*0r*t 5E r!!t r!!t 2252520 &ars 2E $562$ +t&p`
- Les fichiers dans le répertoire tmp ne peuvent être détruits que par :
  - Root
  - Leur propriétaire
- Bien que tout le monde ait le droit d'écriture dans tmp.
- Empêche la substitution de fichiers temporaires pendant l'exécution d'un programme

Montage de systèmes de fichiers

- `&!unt +de +hda= +&nt`
  - Réservé à « root »
  - Sinon, moyen simple d'amener du code malicieux
- Empêcher l'exécution de programmes en provenance de systèmes de fichiers montés
  - `n!e0ec`
  - `n!suid`

Limitations Ressources

- POSIX :
- Chaque ressource a une limite « soft » et une limite « hard »
  - On peut changer une limite soft entre 0 et « hard »
  - On peut baisser limite « hard »
- Si super utilisateur (ou CAP_SYS_RESOURCE) on peut augmenter une limite « hard ».

Capacités (*capabilities*)

- Au lieu d'avoir un (super) utilisateur ayant tous les droits, donner des droits restreints en fonction des opérations à effectuer.
- Les opérations du noyau sont contrôlées par des services distincts les uns des autres.
  - Les services (hooks) par défaut vérifient si la capacité correspondante est détenue par le processus
- Par défaut « root » a toutes les capacités

Capacités (*capabilities*)

- CAP_CHOWN
  - Permission de changer propriétaire d'un fichier
- CAP_KILL
  - Outrepasse les vérifications lors de l'envoi de signal
- CAP_MKNOD
  - Droit de créer des fichiers de type périphérique...
- CAP_SETGID, CAP_SETUID
  - Droit de modifier les uid/gid d'un process

Capacités (*capabilities*)

- CAP_SYS_ADMIN
  - Mount, swap, sethostname...
- CAP_SYS_BOOT
  - Permet invoquer « reboot »
- CAP_SYS_CHROOT
  - Permet de changer la racine
- CAP_SYS_MODULE
  - Droit de charger / décharger des modules
- CAP_SYS_TIME
  - Droit de changer l'heure

Capacités (*capabilities*)

- Pour être réellement utile, il faudrait lier ce mécanisme avec la gestion des binaires exécutables :
  - Quel programme a besoin de quelle capacité ?
  - Pas fait à ce jour
- Utilisation par les Linux Security Modules
  - SELinux
  - Android ?

Processus

- Limites
  - Définie au niveau système pour un utilisateur lambda
  - Définie comme un « pourcentage » mémoire physique au niveau global.
- « Problème » des « fork bomb »
  - Utilisateur lambda => « facile à résoudre »
  - Root :
    - variations sur session/group
    - Disparition des processus parents...

SE Linux

- Permet d'implémenter :
  - Politique *Mandatory Access Control* (MAC)
  - Au lieu de la politique habituelle :
  - *Discretionary Access Control* (DAC)
- S'appuie sur les « hooks » du noyau :
  - Linux Security Modules

AppArmor

- Utilisé par Suse
- Principes similaires
  - Mais identification des objets par leur « chemin »
- On peut enregistrer un profile d 'exécution
  - Sert à détecter ensuite les déviations par rapport au profile enregistré.

Architectures Matérielles

- TPM : Trusted Platform Module
- ARM TrustZone
- Crypto X86

TPM

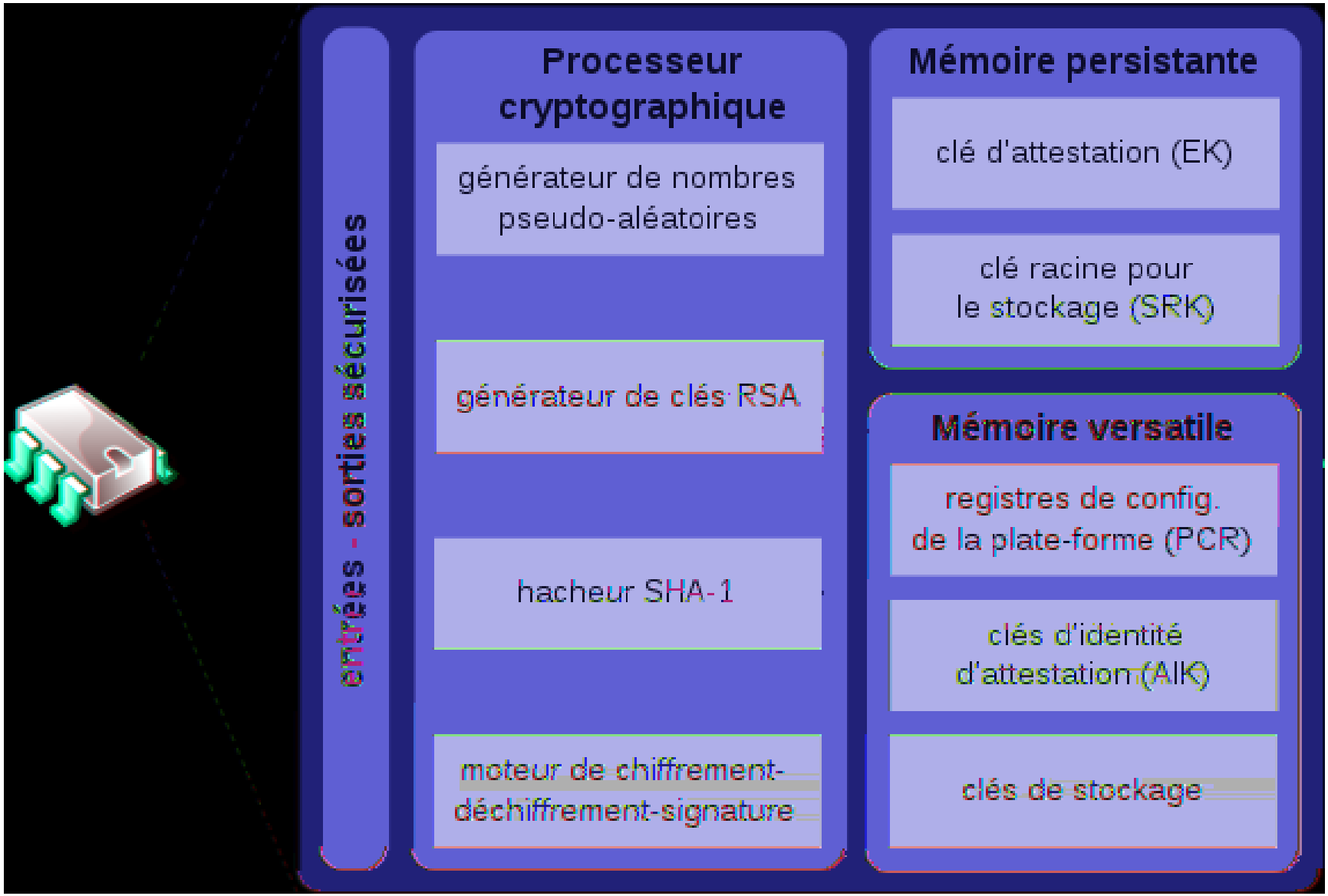


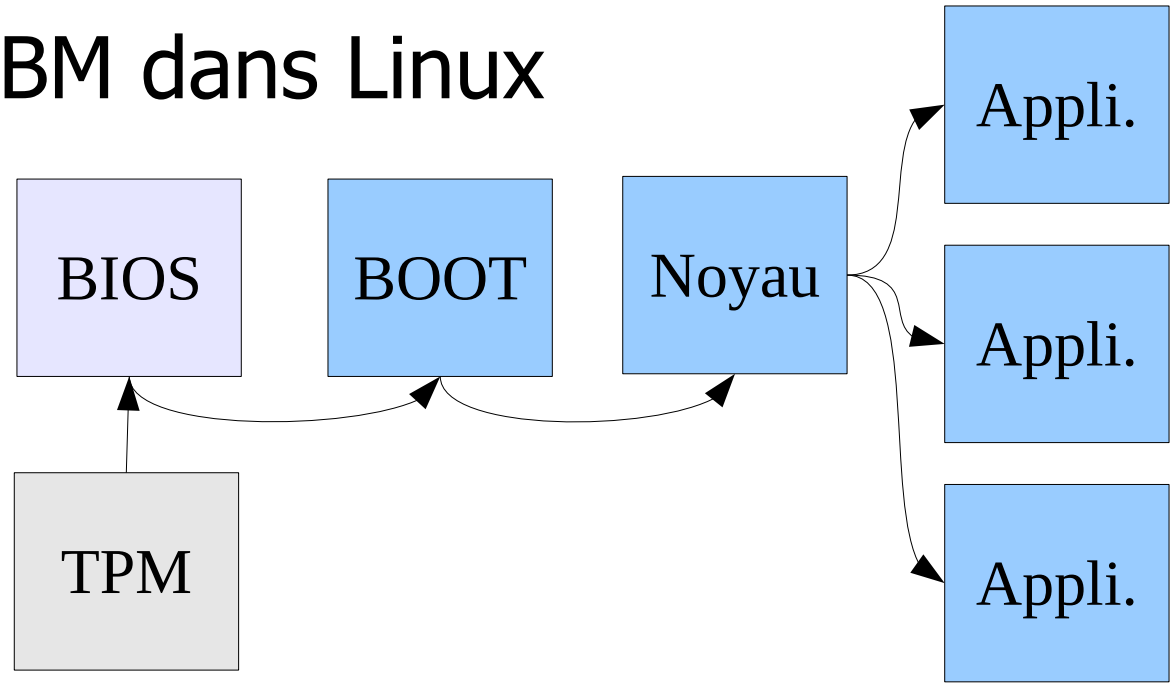
Image wikipedia de Eusebius (Guillaume Piolle).

TPM

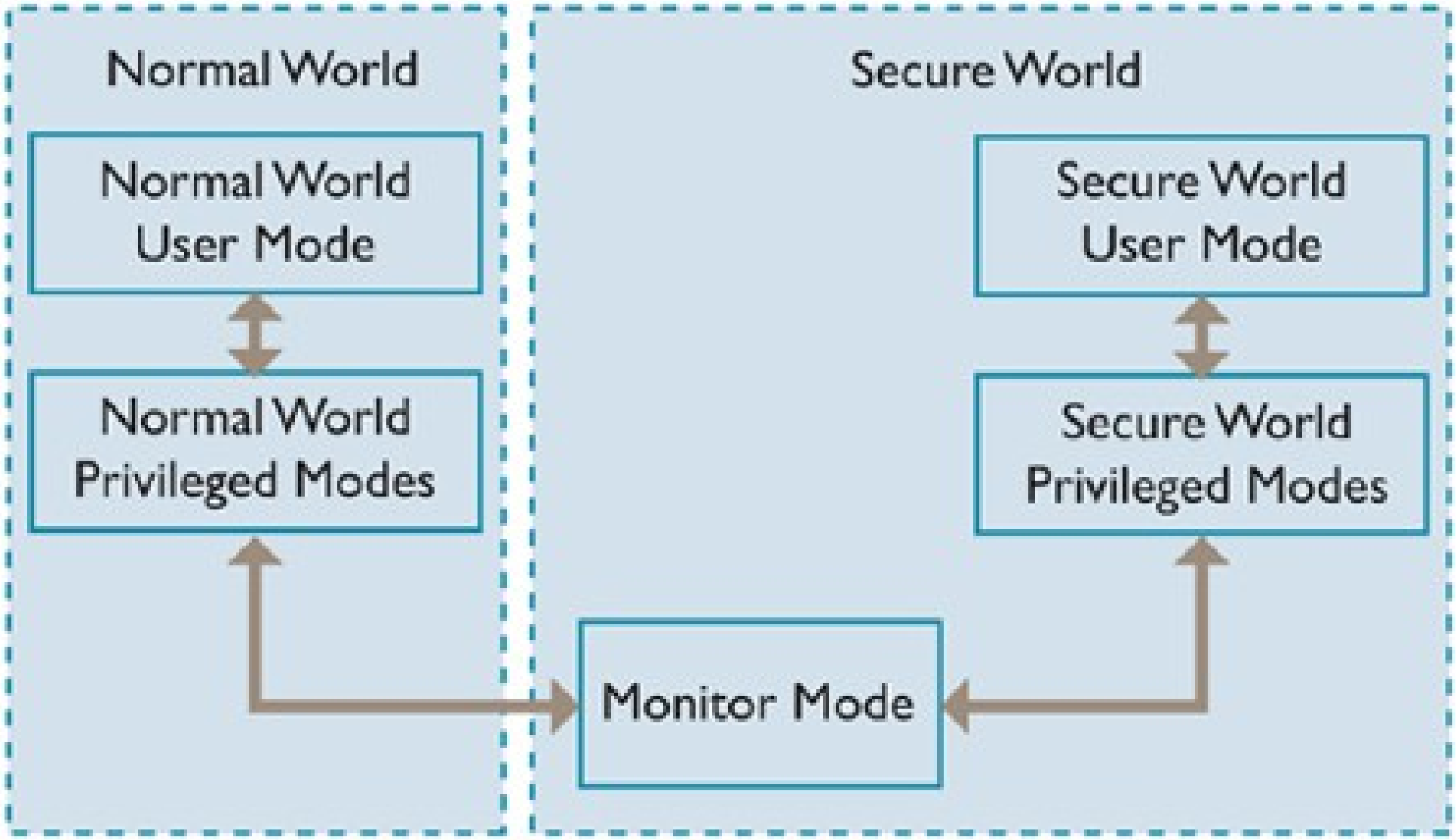
- Controversé
  - Opposition de la communauté Logiciels Libres
  - A été très en vogue pour les applications DRM
- Vulnérable à certaines attaques physiques
  - (Onéreuses)
- Limitation :
  - Impose SHA-1 qui n'est plus recommandé par les agences de sécurité...
- Nouvelles versions...

TPM et OS

- Le TPM vérifie la signature du BIOS
- De manière transitive, chaque composant vérifie la signature du composant « suivant »
- Implémenté par IBM dans Linux
- Trusted boot
- Peu utilisé

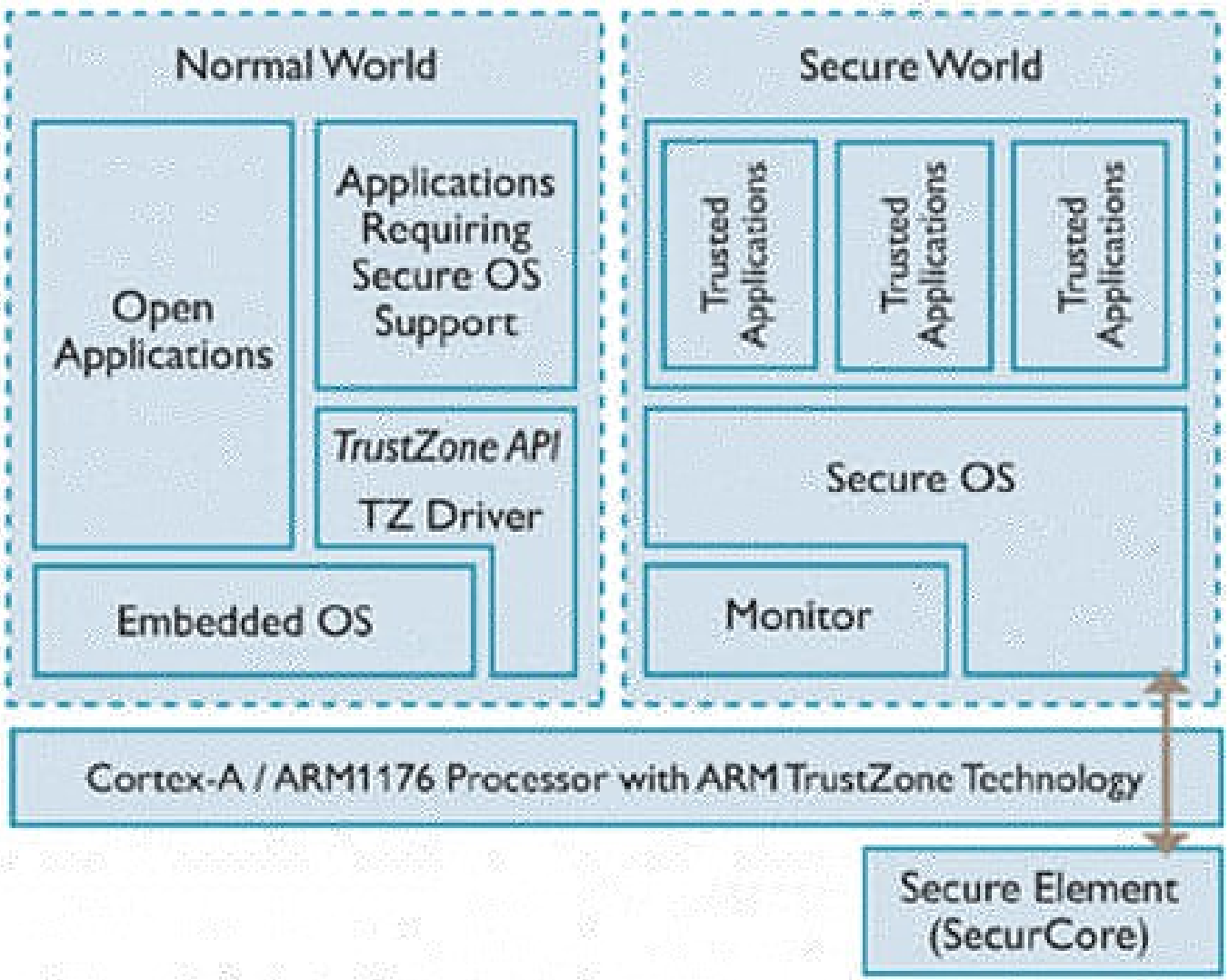


ARM TrustZone
Hardware architecture



Source: http://www.arm.com/products/processors/technologies/trustzone.php

ARM TrustZone



Source: http://www.arm.com/products/processors/technologies/trustzone.php

CryptoPage / X86

- Architecture matérielle :
  - Toutes les données et le code sont chiffrés en mémoire
  - Décryptés au moment du chargement dans le processeur / cache
  - Techniques pour cacher les « modèles d'accès »
    - Permutations régulières des lignes de cache dans un bloc mémoire.
- Implémentation sur QEMU