

Rétro-ingénierie binaire

INF600C
Sécurité des logiciels et exploitation de vulnérabilités

Auteur: Jean Privat, Adapté par: Philippe Pépos Petitclerc

Hiver 2025

Université du Québec à Montréal

Notes

Rétro-ingénierie binaire

Notes

Plan

Les semaines qui s'en viennent

- Rétro-ingénierie binaire (rev)
- Corruption de mémoire et exploits (pwn)
- Contre-mesures classiques et exploits (rop)
- Contre-mesures modernes et exploits (hard)

Notes

Prérequis

INF2170 Organisation des ordinateurs et assembleur

- Comprendre le comportement du processeur et de la RAM
- Savoir lire et écrire des petits programmes en assembleur

INF3135 Construction et maintenance de logiciels

- Comprendre la programmation procédurale
- Comprendre l'utilisation de la mémoire et des pointeurs
- Savoir lire et écrire des petits programmes en C

3

Notes

Difficulté

- C'est très technique
 - C'est assez hermétique
 - Ça nécessite beaucoup de pratique
- On va aller lentement en cours
- Vous devez faire **activement** les labs

4

Notes

Rétro-ingénierie binaire

Rétro-ingénierie binaire

PIN1: le code c'est la vérité

Qu'est-ce que la rétro-ingénierie ?

Langages machine et d'assemblage

Rétro-ingénierie, c'est difficile

PIN2: des bogues

PIN3: prendre le contrôle

5

Notes

PIN1

Un programme demande un PIN.

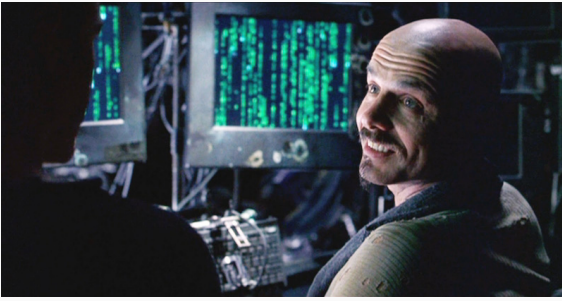
```
$ ./pin
PIN:42
Erreur!
$ ./pin
PIN:1111
Erreur!
$ ./pin
PIN:hello
Erreur!
```

Objectif: trouver le PIN

6

[illegible]

Comprendre le binaire?



« ...there's way too much information to decode the Matrix. You get used to it, though. Your brain does the translating. I don't even see the code. All I see is blonde, brunette, redhead. »
— Cypher, Matrix (1999)

8

Notes

Binaire exécutable

Fichier qui contient le programme exécutable

- le code machine du programme (et sous-programmes)
- du code machine ajouté par le compilateur/assembleur
- des données (dites statiques)
- des métadonnées pour éditeurs de liens, chargeurs, débogueurs...

9

Notes

Machine?

Langage machine

- Langage natif du processeur
- Composée d'instructions et de données codées en binaire
- Spécifique à une architecture (type de processeur)

Code machine

- Programme en langage machine
- Séquence de bits
- Interprétable directement par le processeur

10

Notes

Commande strings(1)

Rappel INF2170

- Tout n'est que bits
- Il n'y a pas de magie

```
$ strings pin1
__isoc99_scanf
printf
stdout
t$,U
PIN:
INF600C{%d}
```

Des octets dans le binaire servent à coder des chaînes

- Chaînes littérales écrites par un programmeur
- Noms de fonctions, de symboles, de sections, etc.
- Informations de débogage

11

Notes

Qu'est-ce que la rétro-ingénierie ?

Notes

Rétro-ingénierie (ingénierie inverse)

Comprendre le fonctionnement d'un programme

Objectifs

- Le maintenir
- Le faire inter-opérer
- S'assurer de son bon fonctionnement et de sa robustesse
- S'assurer de son innocuité
- S'en protéger, en cas de logiciel malveillant (*malware*)
- Créer une version compatible sans vol de copyright
- Percer les façons de faire des concurrents
- Récupérer des secrets embarqués
- Trouver des failles de sécurité

12

Notes

Légalité de la rétro-ingénierie

Complexe

- Droit d'auteur (*copyright*)
- Brevet d'invention (*patent*)
- Droit des contrats (*end user license agreement*)

Varié

- Canada, [Loi sur le droit d'auteur](#) (C-42), 1985
- États-unis, DMCA, 1998
- Europe, EUCD, 2001
- France, DADVSI, 2006

13

Notes

En gros dans le monde

Interopérabilité

- + ou - protégée pour l'utilisation personnelle
- Il y a des contraintes en cas de diffusion (ex. *clean room design*)

Mesures techniques de protection (DRM)

- Le contournement est interdit
- La promotion, la distribution, la vente ou l'utilisation de logiciels et/ou de services de contournement est interdit
- Sauf à des fins de recherche, de sécurité ou d'interopérabilité (sous contraintes)

14

Notes

Langages machine et d'assemblage

Notes

Fichiers exécutables

```
$ ls -l
-rwxr-xr-x 1 privat privat 7344 mar  7 09:59 pin1
-rwxr-xr-x 1 privat privat 8520 mar  7 09:59 pin1_64
-rw-r--r-- 1 privat privat 117 mar  7 09:59 pin1.pepo

$ file *
pin1:      ELF 32-bit LSB executable, Intel 80386
pin1_64:   ELF 64-bit LSB executable, x86-64
pin1.pepo: ASCII text

$ cat pin1.pepo
41 00 20 31 00 37 C1 00 37 B0 24 B6 0C 00 1C 41
00 25 39 00 37 50 00 7D 50 00 0A 00 41 00 2E 00
50 49 4E 3A 00 49 4E 46 36 30 30 43 7B 00 45 72
72 65 75 72 21 0A 00 00 00 zz
```

15

Notes

Pep/8 Rappel

Pédagogique

- Pour apprendre la programmation assembleur
- Représentatif des processeurs actuels
- Livré avec un simulateur graphique

Simple

- 16 bits
- 37 instructions (mnémoniques)
- 5 registres
- 8 modes d'adressages

16

Notes

Assembleur

Langage d'assemblage (ou assembleur)

- Représentation du code machine lisible par un humain
- Directives, littéraux, symboles, étiquettes

Assembler et assemblage

- Transformer du code d'assemblage en code machine équivalent

Assembleur

- Outil faisant l'assemblage. Exemple: `as`, `nasm`, `masm`, `asem8`

Désassembler

- Transformer du code machine en code d'assemblage équivalent
- C'est une **analyse statique** du binaire d'un programme

17

Notes

pin1: désassemblage

pep/8

```
0006:  c10024      lda  0x0024,d
0009:  b024b6      cpa  0x24b6,i
000c:  0c0018      brne 0x0018
```

Où est le PIN?

- Le bon PIN est dans le binaire
- C'est le même principe que strings
- Mais en plus technique

Contre-mesures

- Ne pas mettre de secrets dans le binaire

18

Notes

Un vrai processeur?

```
$ objdump -d -Mintel pin1
8048502: a12ca00408   mov  eax,ds:0x804a02c
8048507: 3d023a0100   cmp  eax,0x13a02
804850c: 7516         jne  8048524

$ objdump -d -Mintel pin1_64
40062c: 48b051d0a200 mov  rax,QWORD PTR [rip+0x200a1d]
400633: 483dd8210000 cmp  rax,0x21d8
400639: 7519         jne  400654
```

En vrai c'est pareil

- Le bon PIN est dans le binaire
- Mais décompiler à la main c'est pénible
- Un outil c'est plus simple: `objdump(1)` de GNU binutils.

19

Notes

Architecture x86

Jeu d'instruction (*Instruction set architecture, ISA*)

- 1978: 16 bits Intel 8086
- 1985: 32 bits Intel 80386
- 2001: 64 bits Intel Itanium (lol)
- 2003: 64 bits AMD64 x86-64

Complexe

- *Complex instruction set computer (CISC)*
- 981 mnémoniques (et 3684 variations)
- Redondant
- Contraintes et noms bizarres/historiques
- Plein de trucs obsolètes: MMX, BCD, etc.
- Plein de trucs sales: alignement `nop`, `repz ret`, etc.

20

Notes

2018 CVE List

LEAKED LIST OF MAJOR 2018 SECURITY VULNERABILITIES

CVE-2018-????? APPLE PRODUCTS CRASH WHEN DISPLAYING CERTAIN TELUGU OR BENGALI LETTER COMBINATIONS.
CVE-2018-????? AN ATTACKER CAN USE A TIMING ATTACK TO EXPLOIT A RACE CONDITION IN GARBAGE COLLECTION TO EXTRACT A LIMITED NUMBER OF BITS FROM THE WIKIPEDIA ARTICLE ON CLAUDE SHANNON.
CVE-2018-????? AT THE CAFE ON THIRD STREET, THE POST-IT NOTE WITH THE WiFi PASSWORD IS VISIBLE FROM THE SIDEWALK.
CVE-2018-????? A REMOTE ATTACKER CAN INJECT ARBITRARY TEXT INTO PUBLIC-FACING PAGES VIA THE COMMENTS BOX.
CVE-2018-????? MYSQL SERVER 5.5.145 SECRETLY RUNS TWO PARALLEL DATABASES FOR PEOPLE WHO SAY "S-Q-L" AND "SEQUEL".
CVE-2018-????? A FLAW IN SOME x86 CPUs COULD ALLOW A ROOT USER TO DE-ESCALATE TO NORMAL ACCOUNT PRIVILEGES.
CVE-2018-????? APPLE PRODUCTS CRASH FIRE WHEN DISPLAYING EMOTI WITH DIACRITICS.
CVE-2018-????? AN OVERSIGHT IN THE RULES ALLOWS A DOG TO JOIN A BASKETBALL TEAM.
CVE-2018-????? HASKELL ISN'T SIDE-EFFECT-FREE AFTER ALL; THE EFFECTS ARE ALL JUST CONCENTRATED IN THIS ONE COMPUTER IN MISSOURI THAT NO ONES CHECKED ON IN A WHILE.
CVE-2018-????? NOBODY REALLY KNOWS HOW HYPERVISORS WORK.
CVE-2018-????? CRITICAL UNDER LINUX 3.14.8 ON SYSTEM390 IN A UTC+H TIME ZONE, A LOCAL USER COULD POTENTIALLY USE A BUFFER OVERFLOW TO CHANGE ANOTHER USER'S DEFAULT SYSTEM CLOCK FROM 12-HOUR TO 24-HOUR.
CVE-2018-????? x86 HAS WAY TOO MANY INSTRUCTIONS.
CVE-2018-????? NUMPY 1.8.0 CAN FACTOR PRIMES IN O(LIN) TIME AND MUST BE QUICKLY DEPRECATED BEFORE ANYONE NOTICES.
CVE-2018-????? APPLE PRODUCTS GRANT REMOTE ACCESS IF YOU SEND THEM WORDS THAT BREAK THE "I BEFORE E" RULE.
CVE-2018-????? SKYLAKE x86 CHIPS CAN BE PRIED FROM THEIR SOCKETS USING CERTAIN FLASHHEAD SOLDERWORKERS.
CVE-2018-????? APPARENTLY LINUX TORVALDS CAN BE BROOD PRETTY EASILY.
CVE-2018-????? AN ATTACKER CAN EXECUTE MALICIOUS CODE ON THEIR OWN MACHINE AND NO ONE CAN STOP THEM.
CVE-2018-????? APPLE PRODUCTS EXECUTE ANY CODE PRINTED OVER A PHOTO OF A DOG WITH A SADDLE AND A BABY RIDING IT.
CVE-2018-????? UNDER RARE CIRCUMSTANCES, A FLAW IN SOME VERSIONS OF WINDOWS COULD ALLOW FLASH TO BE INSTALLED.
CVE-2018-????? TURNS OUT THE CLOUD IS JUST OTHER PEOPLE'S COMPUTERS.
CVE-2018-????? A FLAW IN MYSQL'S CVE DATABASE ALLOWS ARBITRARY CODE INSERTION. [←CLICK HERE FOR CHEAP VAGRA→]

Source: <https://xkcd.com/1957/> (2018)

21

Notes

Syntaxe assembleur AT&T vs. Intel

Syntaxe AT&T

- Préfixe: valeurs \$, registres %, destinations *
- Ordre: mov source,destination
- Adressage: %segreg:disp(base,index,scale)
- Taille mémoire dans l'instruction si besoin movb, movw, movl, movq
- Défaut chez Unix et les outils GNU

Syntaxe Intel

- Pas de préfixe
- Ordre: mov destination,source
- Adressage: segreg:[base+index*scale+disp]
- Taille mémoire explicite: BYTE, WORD, DWORD, QWORD
- Plus populaire en sécu et dans les outils Windows
- Plus proche de Pep/8

22

Notes

Registres importants

Pep/8: 16 bits

- A et X: registres généraux
- SP: pointeur de pile (*stack pointer*)
- IP: pointeur d'instruction (compteur ordinal)

80386: 32 bits

- EAX, ECX, EBX, EDX, ESI et EDI: registres généraux
- ESP: pointeur de pile (haut de la pile)
- EBP: pointeur de base
- EIP: pointeur d'instruction

x86-64: 64 bits

- 8 de base (RAX → RBP), 8 nouveaux (R8 → R15) et RIP

23

Notes

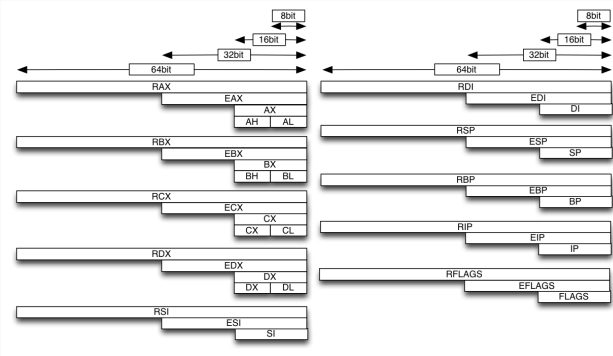
Pep/8: NZVC

- N: négatif (signe)
- Z: zéro
- V: débordement (*overflow*)
- C: retenue (*carry*)

80386 et x86-64: EFLAGS

- SF: signe (négatif)
- ZF: zéro
- OF: débordement (*overflow*)
- CF: retenue (*carry*)

Notes



Notes

Instructions 0x86

Transfert de valeur

- Copie une valeur
- Immédiate, en mémoire ou registre
- En x86: `mov`
- En Pep/8: `ld, st`

Opération arithmétique et logique

- Opérations unaires ou binaires
- En x86: `add, sub, cmp, mul, and, sar, etc.`
- En Pep/8: `add, sub, cp, ..., and, asr, etc.`

Notes

Instructions usuelles

Contrôle du flot d'exécution

- Branchements et sous-programmes
- En x86: `jmp`, `jl`, `call`, `ret`, etc.
- En Pep/8: `br`, `brlt`, `call`, `ret0`, etc.

LEA: *Load Effective Address*

- Détermine l'adresse effective d'un `mov` (sans accès mémoire)
Ça calcule un pointeur
- Sert aussi à faire de l'arithmétique pas chère
`segreg:[base+index*scale+disp]`

Pile

- `push` et `pop`: empile et dépile une valeur; modifie ESP
- `enter` et `leave`: empile et dépile un cadre; modifie EBP et ESP

27

Notes

Rétro-ingénierie, c'est difficile

Notes

Rétro-ingénierie, c'est difficile

Que fait le programme Pep/8 suivant?

C0 FE FE 16 00 0D 70 65 70 38 00 41 00 06 00 0B 58

| | | | |
|-------|--------|------|---------|
| 0000: | C0FEFE | LDA | -258,i |
| 0003: | 16000D | CALL | 0xD |
| 0006: | 706570 | ADDA | 25968,i |
| 0009: | 380041 | DECO | 65,i |
| 000C: | 00 | STOP | |
| 000D: | 06000B | BRLE | 0xB |
| 0010: | 58 | RET0 | |

28

Notes

L'interprétation dépend de l'observateur

C0 FE FE 16 00 0D 70 65 70 38 00 41 00 06 00 0B 58

Si on interprète les octets différemment...

| | | | |
|-------|------------|--------|------------|
| 0000: | C0FEFE | LDA | -258, i |
| 0003: | 16000D | CALL | 0xD |
| 000D: | 06000B | BRLE | 0xB |
| 000B: | 410006 | STRO | 0x6, i |
| 000E: | 00 | STOP | |
| 0006: | 7065703800 | .ASCII | "pep8\x00" |

29

Notes

Qu'est-ce que la vérité ?



« Boy: Do not try and bend the spoon. That's impossible. Instead only try to realize the truth.

Neo: What truth?

Boy: There is no spoon. » – Matrix (1999)

30

Notes

Code impénétrable (*Code obfuscation*)

Rendre le code difficile à comprendre

- Enlever toute information de débogage (`strip(1)`)
- Complexifier les algorithmes
- Forcer le désassembleur à mal désassembler

Techniques d'anti-désassemblage:

- Utiliser un même octet en RAM pour des significations différentes
- Abuser de branchements indirects calculés
- Générer/muter le code machine lors de l'exécution
- Ajouter de l'aléa pour rendre les choses faussement non-déterministes

31

Notes

Qui est aussi méchant?



32

Notes

Qui est aussi méchant?

- Développeurs de logiciels malveillants (*malwares*)
- Développeurs utilisant la sécurité par l'obscurité (DRM)
- Développeurs paranoïaques
- Amateurs de défis et de casse-têtes (CTF)
- Développeurs de compilateurs optimisants (dégât collatéral)

33

Notes

PIN2: des bogues

Notes

PIN2

Objectif: trouver le PIN

```
$ ./pin2
PIN: 12
Erreur!

$ ls -l
-rwxr-xr-x 1 privat privat 7352 mar  9 12:59 pin2
-rwxr-xr-x 1 privat privat 8528 mar  9 12:59 pin2_64
-rw-r--r-- 1 privat privat 260 mar  9 13:27 pin2.pepo

$ cat pin2.pepo
41 00 4B 31 00 49 C0 06 50 C8 00 04 16 00 1C B1
00 49 0C 00 1B C1 00 49 16 00 3C 00 68 00 02 E3
00 00 B8 00 00 06 00 38 1E 70 00 01 73 00 00 E3
00 00 88 00 01 04 00 22 C3 00 00 5A 68 00 02 41
00 50 3B 00 00 51 00 7D 5A 00 00 50 49 4E 3A 00
46 4C 41 47 7B 00 zz
```

Notes

pin2.pepo désassemblé (1)

```
main:

0000 41004B main:  STRO  0x004B,d
0003 310049      DECI  0x0049,d
0006 C00650      LDA  0x650,i
0009 C80004      LDX  0x4,i
000C 16001C      CALL get_pin
000F B10049      CPA  0x0049,d
0012 0C001B      BRNE 0x001B
0015 C10049      LDA  0x0049,d
0018 16003C      CALL print
001B 00          STOP
```

- Le bon PIN est calculé par get_pin

Notes

pin2.pepo désassemblé (2)

```
get_pin:

001C 680002 get_pin: SUBSP 2,i
001F E30000      STA  0,s
0022 B80000      CPX  0,i
0025 060038      BRLE 0x0038
0028 1E          ASRA
0029 700001      ADDA 0x1,i
002C 730000      ADDA 0,s
002F E30000      STA  0,s
0032 880001      SUBX 0x1,i
0035 040022      BR   0x0022
0038 C30000      LDA  0,s
003B 5A          RET2
```

- get_pin est compliqué
- On peut tenter de comprendre l'algo
- Mais il y a plus simple...

Notes

Exécution contrôlée d'un programme

- Pas à pas, instruction par instruction
- Voir le contenu de la mémoire et des registres
- Surveiller les appels

C'est une **analyse dynamique** d'un programme

Objectif

- Diagnostiquer certains bugs
- Rétro-ingénierie

37

Notes



« The pill you took is part of a trace program. It's design to disrupt your input/output carrier signal so we can pinpoint your location. »

— Morpheus, Matrix (1999)

38

Notes

GNU Debugger

- 1986 (Richard Stallman)
- Supporte de nombreux langages et architectures
- Interface texte (console)
- Nombreuses interfaces graphique (tierces parties)
- Débogage de processus en cours
- Débogage réseau

39

Notes

Python Exploit Development Assistance for GDB

- Améliore l'affichage de GDB
- Ajoute des fonctions d'aide à l'ingénierie inverse
- Ajoute des fonctions d'aide au développement d'exploits

```
$ git clone https://github.com/longld/peda.git ~/peda
$ echo "source ~/peda/peda.py" >> ~/.gdbinit
$ echo "set disassembly-flavor intel" >> ~/.gdbinit
```

40

Notes

Commandes gdb/peda utiles

Exécuter

- `run args`: exécute depuis le début avec des arguments
- `start (peda)`: exécute jusqu'au début du main
- `si, stepi`: exécute une instruction, entre dans les fonctions
- `ni, nexti`: exécute une instruction, n'entre pas dans les fonctions
- `finish`: exécute jusqu'à la fin de la fonction
- `nextcall (peda)`: exécute jusqu'au prochain `call`
- `nextjmp (peda)`: exécute jusqu'au prochain `jmp`
- `c, continue`: reprend l'exécution
- `b *adresse`: met un point d'arrêt

Divers

- `entrée`: refait la dernière commande
- `q, quit`: quitter
- `h cmd, help cmd`: affiche l'aide
- `peda`: affiche les commandes peda

41

Notes

Commandes gdb/peda utiles

Inspecter

- `p expr`: calcule et affiche une expression (en hexa par défaut)
- `p/d expr`: pareil mais en décimal (d'autres formats existent)
- `x adresse`: affiche le contenu d'une adresse
- `x/3db adresse`: affiche **3** décimaux, chacun d'un octet (**b** byte)
- `telescope adresse (peda)`: affiche et déréférence
- `pdisass fonction (peda)`: désassemble une fonction
- `bt, backtrace`: affiche la pile d'appels

42

Notes

PIN3: prendre le contrôle

Notes

PIN3

```
$ ./pin3  
PIN:1234  
Erreur!
```

Objectif: Ignorer le PIN et avoir le flag

Notes

43

Plus de commandes gdb/peda utiles

Modifier

- `set $reg = expr`: modifier la valeur d'un registre
- `goto adresse`: modifier le compteur ordinal
- `skipi (peda)`: ignorer une instruction (ça fait des bonds)
- `return`: quitter de force une fonction sans l'exécuter
- `patch adress valeur (peda)`: écrire une valeur en mémoire

Pourquoi modifier?

- Mieux comprendre ce qui se passe

Notes

44

Prendre le contrôle



« I don't like the idea that I'm not in control of my life. »
— Neo, Matrix (1999)

45

Notes

Cercle de confiance

Qui peu déboguer ?

- Seul l'utilisateur légitime peut contrôler le comportement
- Mais il n'y a aucun privilège à gagner

Ça ne fonctionne pas

- Sur un processus d'un autre utilisateur
- Sur un binaire suid
- Sur un processus d'une autre machine

46

Notes

Un débogueur, comment ça marche

Un outil surpuissant

- Suspendre et reprendre l'exécution
- Lire toute la mémoire
- Modifier toute la mémoire... même celle en lecture seule
- Lire et modifier les registres
- Intercepter les signaux
- Mettre des points d'arrêts

Pas de magie

- `gdb(1)` est un programme normal non privilégié
- Le système lui permet d'observer et de contrôler d'autres processus
- Appel système `ptrace(2)`
- Utilisé aussi par `strace(1)` et `ltrace(1)`

47

Notes

Principe de ptrace

- Observé et observateur sont des processus **indépendants**
Un observateur, plusieurs observés
- L'observé peut être
Un processus fils (`PTTRACE_TRACEE`)
Un processus existant de l'utilisateur (`PTTRACE_ATTACH`)
- Lorsque l'observé reçoit un signal
Le système **arrête** l'observé (état stoppé)
L'observateur est notifié (via `wait`)
- Quand l'observé est stoppé, l'observateur peut
L'inspecter et le bricoler
Le faire repartir (*continue*)
- L'observateur est laissé à lui-même
Interpréter les octets de la mémoire et des registres
Bricoler et restaurer le code machine
ex. points d'arrêts via `int 3` (`0xCC`)

48

Notes

Observateur vs. observé



« First there was darkness. Then came the strangers. They abducted us and brought us here. This city, everyone in it, is their experiment. They mix and match our memories as they see fit, trying to divine what makes us unique. » — Dr. Daniel P. Schreber, *Dark City* (1998)

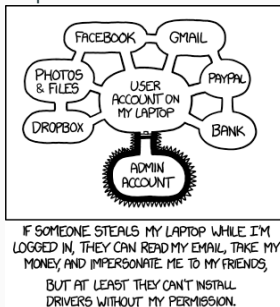
49

Notes

Rappel de sécurité traditionnelle

Sauf contre-ordre validé par le système d'exploitation:

- Les contrôles d'accès sont au niveau des utilisateurs
- Un processus a le complet contrôle de son espace mémoire
- Un processus ne peut accéder à la mémoire d'autres processus



50

Notes

Sécurité et ptrace?

Mode d'accès `ptrace`

- Pour les appels systèmes
- qui permettent d'accéder à la mémoire d'autres processus

Règles de base:

- Restreint aux mêmes utilisateurs et groupes
- modulo *root*
- modulo *setuid*

Lire et écrire la mémoire

- `/proc/pid/mem` (nécessite d'être ptracé et ptrace-stoppé)
- `ptrace(2)`; commandes `PTRACE_PEEKDATA` et `PTRACE_POKEDATA`
- `process_vm_readv(2)`, `process_vm_writev(2)`
- `gcore(1)` pour générer une image mémoire

51

Notes

Vol de secret

Un logiciel malveillant peut utiliser `ptrace` pour surveiller ou contrôler tout processus d'un utilisateur

- vol de secrets dans la mémoire de `ssh`, `gpg`, etc.
- vol de terminaux (*keylogger*)

52

Notes

Contre-mesures

Réduction de la surface d'attaque ?

- Interdire d'attacher. On ne trace que les fils. (défaut Ubuntu)
`/proc/sys/kernel/yama/ptrace_scope` (module de sécurité)
- Désactiver le traçage.
`prctl(PR_SET_DUMPABLE, 0), ptrace(PTRACE_TRACEME)`

Vraies contre-mesures ?

- Isoler les applications dans des conteneurs
Exemples: `firejail(1)`, `flatpack`, `snap`, etc.
- Isoler les applications dans des *users*.
Exemple: `Android`
- Contrôler les applications directement
Exemple: `MAC` (`SELinux`, `AppArmor`, etc.)

53

Notes
