Reverse engineering : une introduction

Antoxyde

Securimag

Octobre 2020

Securimag

Définition théorique vague et incompréhensible

"Déconstruire" quelque chose, pour comprendre comment la chose est implémentée, comme est-ce qu'elle fonctionne, qu'est ce qu'elle fait..

Point vocabulaire

"détricoter" en francais (source: bitoduc.fr), donc on va rester en anglais.

Securimag

Ce qui nous intéresse

Principalement, partir d'un programme dont on n'a pas les sources, et appliquer la définition.

Securimag

Buts

- Analyse de malware
- Crack de logiciel (création d'un keygen par ex.)
- Recherche de vulnérabilité
- Réimplementer du code perdu, faire de l'interopératibilité
- Pour les jeux vidéo: création de cheat, de bots, etc
- ou bien simplement satisfaire sa curiosité :-)

Securimag

Exemple: Analyse de malware

Malware

2017: Marcus Hutchins (@MalwareTech) et WannaCry.

Securimag

Exemple: Analyse de malware

- Le malware utilisait une 0day sur SMB (EternalBlue, leak de la NSA) qui lui permettait de se propager sur les PC non patchés sur le même réseau local
- Nom de domaine "anti-sandbox" : www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com

Securimag

Exemple: "recréation" d'un logiciel closed source/dont on a perdu le code source

- Driver nvidia sous licence propriétaire et closed-source
- Le driver "nouveau" est développé en reversant le driver propriétaire

Securimag

Point de départ

On a seulement du "code machine" (ie un amas d'octets), et on veut comprendre ce que l'ordinateur va faire en l'exécutant
On peut avoir:

- Du Bytecode .pyc, .class, .luac, .mrb,
- Du "vrai" code machine: un binaire (ELF, PE, Mach-O)

Securimag

Bytecode

Bytecode: exécuté sur une machine virtuelle, un interpreter.

```
2           0 LOAD_GLOBAL              0 (print)
            2 LOAD_CONST               1 ('Hello, World!')
            4 CALL_FUNCTION            1
```

En général, on arrive à remonter à la source (uncompyle6 pour Python par exemple)

Securimag

Code machine

Contient du code directement executé par le CPU.

Type d'exécutable

ELF (Linux), PE (Windows, les .exe), Mach-O (MacOS)

L'architecture

Le "type" du CPU: Intel, ARM, MIPS, .. et 32/64 bits.

Si l'archi du binaire ne correspond pas à celle de votre PC : VM/émulation (VirtualBox, Qemu, ..) ou un raspi qui traine si c'est de l'ARM.

Securimag

Flags de compilation

- Dynamique (link avec les librairies au runtime) vs statique (embarque les librairies)
- Strippé : les symboles (noms de fonctions par ex.) sont retirés

Statique et strippé : tu pleures et tu reverse printf pendant 3h sans t'en rendre compte.

Securimag

Techniques générales

Analyse statique

- Désassembleur : affiche le code assembleur, tout beau tout propre (ex: Radare2, Binary ninja, Cutter, IDA, Ghidra, ..)
- Décompilateur: affiche du C/C++ pas toujours très lisible.

Securimag

Exemple désassemblage (Binary-Ninja)

```
main:
00001239  push    rbp {__saved_rbp}
0000123a  mov     rbp, rsp {__saved_rbp}
0000123d  sub     rsp, 0x50
00001241  mov     rax, qword [fs:0x28]
0000124a  mov     qword [rbp-0x8 {var_10}], rax
0000124e  xor     eax, eax  {0x0}
00001250  lea     rdi, [rel data_2008]  {"Whats the password ?"}
00001257  call    puts
0000125c  lea     rax, [rbp-0x40 {var_48}]
00001260  mov     rsi, rax {var_48}
00001263  lea     rdi, [rel data_201d]  {"%49s"}
0000126a  mov     eax, 0x0
0000126f  call    __isoc99_scanf
00001274  lea     rax, [rbp-0x40 {var_48}]
00001278  mov     rdi, rax {var_48}
0000127b  call    strlen
```

Exemple décompilation (Binary-Ninja)

```
main:
   0 @ 00001241  void* fsbase
   1 @ 00001241  int64_t rax = *(fsbase + 0x28)
   2 @ 00001257  puts(str: "Whats the password ?")
   3 @ 0000126f  void var_48
   4 @ 0000126f  __isoc99_scanf(format: "%49s", &var_48)
   5 @ 00001294  if (verify_password(&var_48, zx.q(strlen(&var_48):0.d)):0.d == 0)
```

Securimag

Techniques générales

Analyse dynamique

- Débuggage: gdb, IDA, x64dbg (windows),..
- Emulation, Instrumentation (Frida, ..), Hooking (LD_PRELOAD, ..), execution symbolique (Angr, miasm)..

Securimag

Show time !

Securimag

Le reverse en CTF

CTF Reversing BINGO!!!!

"more layers means more difficulty!"	MD-5, SHA-1 or CRC	All challenges are Linux	Self-unpacking binary	RC4 or other shitty cipher
Solution requires recognizing crypto impl	Flag checker	Custom virtual machine	Statically-linked, stripped binary	Intended solution involves guessing or side-channel
Involves some kind of graph traversal	Unintended pintool solution	**FREE** Waste of time	C++ STL hell	Solution requires brute-forcing
Shitty custom or ancient architecture	Binary has ABI unsup-ported by IDA Pro	rwx segments, code caves, self-decrypting code, etc.	Binary uses a COTS packer (e.g. vmprotect,	Crackme/ keygenme

Securimag

Site de challs de reverse

- La catégorie "reverse" de root-me.org (débutant/confirmé)
- crackmes.one: banque de crackmes (3246 à l'heure actuelle)
- Challenge du Flareon : 2020.flare-on.com (challenge annuel, niveau hard)
- ctf.securimag.org :sunglasses:

Securimag

Vague de chall sur la plateforme (ctf.securimag.org)

4 nouveaux challenges cette semaine

- PE Switch - 100 pts - (PE, x86_x64)
- aIARM - 150 pts - (ELF, ARM 32bits)
- Oh My Lib - 400 pts - (ELF, x86_64)
- Aïe, ca pyc ! - 400 pts - (ELF/Bytecode python)

Et toujours Memtrap (300pts) et Cequetudisnaaucunsens (100pts)

Securimag

Toolz en vrac

- Unix: file, ltrace, strace, strings (oui oui)
- Windows: Flare VM , cf github.com/fireeye/flare-vm#installed-tools pour une liste de tools
- Android/Java: apktool, dex2jar, JAD, frida,..
- Dotnet: DNSpy, DotPeek,..
- Python: uncompyle, pyinstaller,..

Securimag

Sujets à creuser pour aller plus loin

en vrac aussi

- Unpacking (UPX, ..)
- Execution symbolique (angr, miasm, ..)
- Taint analysis
- Techniques d'anti-debug (Ptrace, signal trap, nanomites, ..)
- Technique d'obfuscation (VMprotect, movfuscator)
- Les tools de quarkslab

Securimag

BIERE

BIERE

Securimag