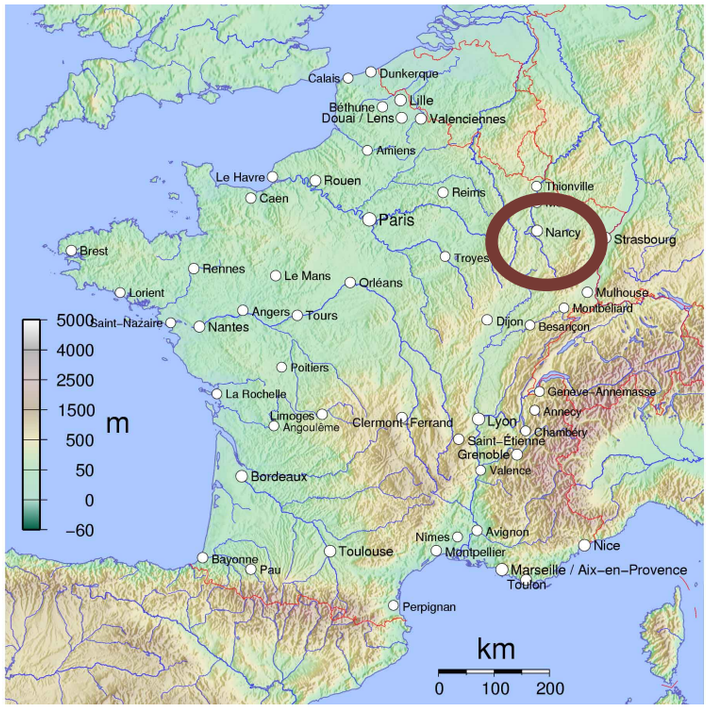
Loria
Laboratoire lorrain de recherche
en informatique et ses applications

CoDisasm

Medium Scale Concatic Disassembly of Self-Modifying Binaries with Overlapping Instructions

Jean-Yves Marion
Professor at l'Université de Lorraine
Institut Universitaire de France

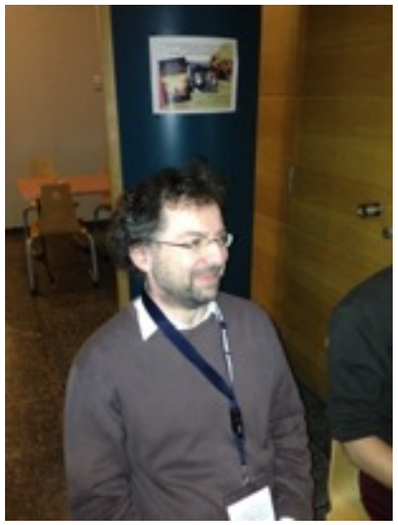


informatiques mathématiques
Inria

cnrs

UNIVERSITÉ DE LORRAINE

This is a joint works with

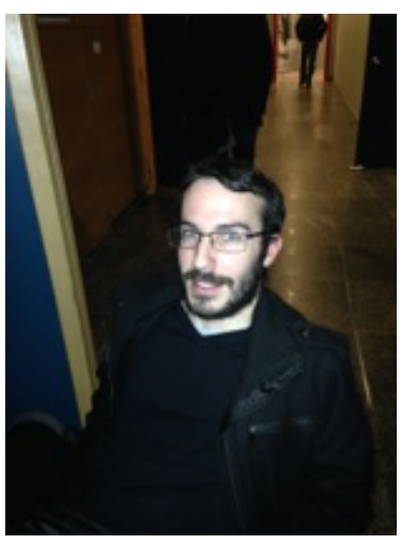


Guillaume Bonfante

José Fernandez

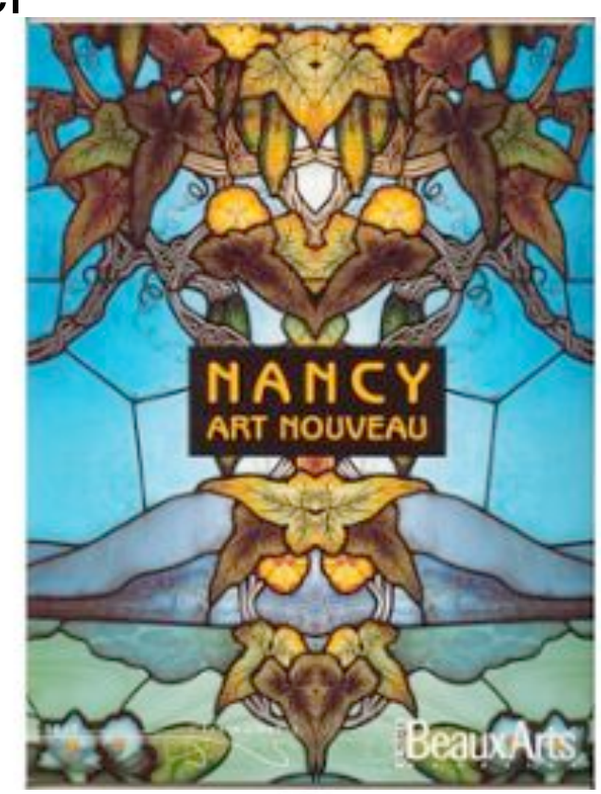
Benjamin Rouxel

ÉCOLE POLYTECHNIQUE MONTRÉAL



Aurélien Thierry

Fabrice Sabatier



Loria Laboratoire lorrain de recherche en informatique et ses applications

Jean-Yves Marion

UNIVERSITÉ DE LORRAINE

Problem
What really makes this program?

Input:
an x86 obfuscated binary code

What Happen After You Hit Return ?

Loria
Laboratoire lorrain de recherche en informatique et ses applications

Jean-Yves Marion

UNIVERSITÉ DE LORRAINE

Towards a high level semantics

Disassembly

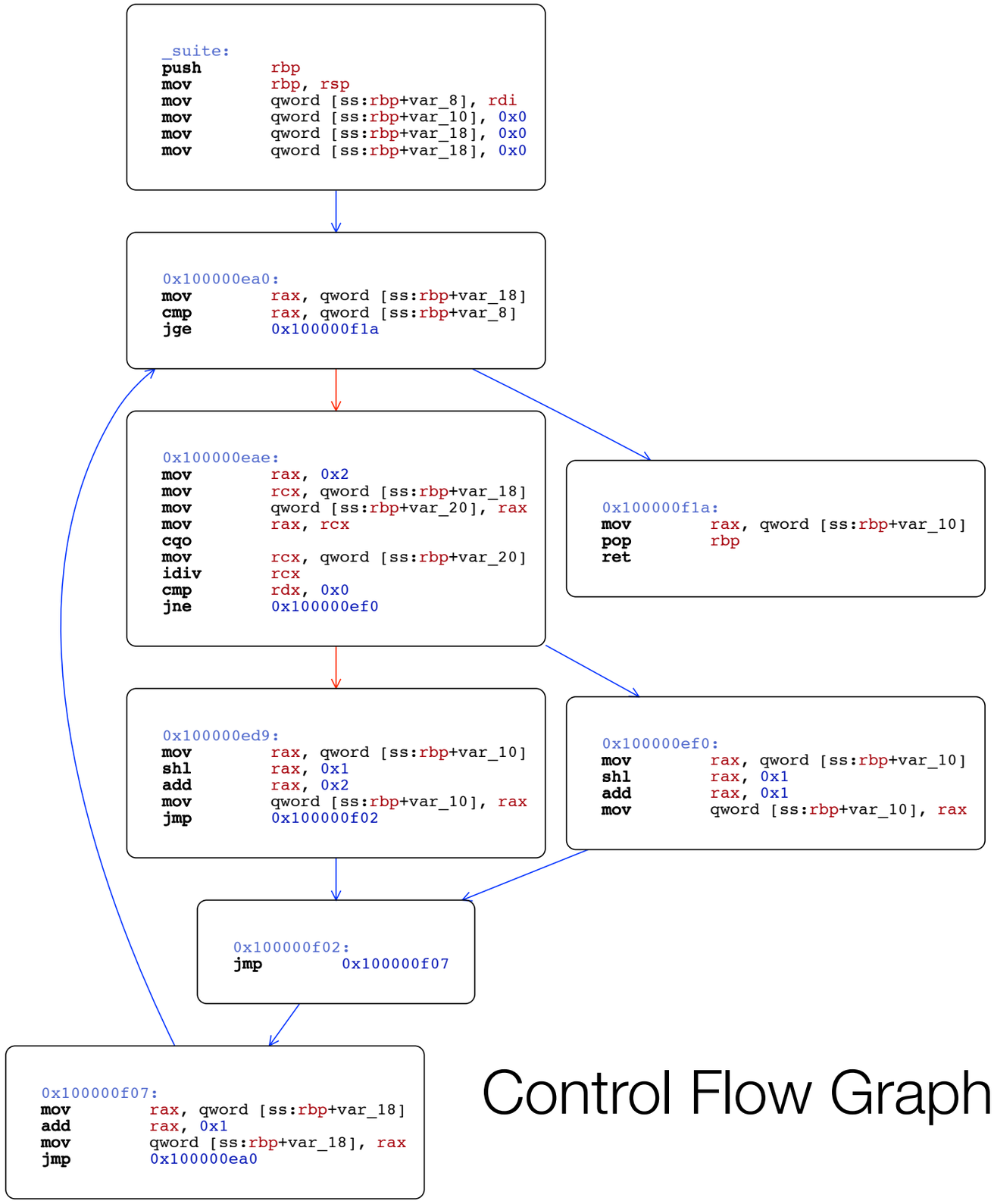
Memory or PE file

```
E80 : 55 48 89 E5 48 89 7D F8 48 C7 45 F0 00 00 00 00 48 C7 45 E8 00 00
E90 : 00 00 48 C7 45 E8 00 00 00 00 48 8B 45 E8 48 3B 45 F8 0F 8D 6C 00
EA0 : 00 00 48 B8 02 00 00 00 00 00 00 00 48 8B 4D E8 48 89 45 E0 48 89
EB0 : C8 48 99 48 8B 4D E0 48 F7 F9 48 81 FA 00 00 00 00 0F 85 17 00 00
EC0 : 00 48 8B 45 F0 48 C1 E0 01 48 05 02 00 00 00 48 89 45 F0 E9 12 00
ED0 : 00 00 48 8B 45 F0 48 C1 E0 01 48 05 01 00 00 00 48 89 45 F0 E9 00
EE0 : 00 00 00 48 8B 45 E8 48 05 01 00 00 00 48 89 45 E8 E9 86 FF FF FF
EF0 : 48 8B 45 F0 5D C3
```

```
0000000100000e80    push    rbp
0000000100000e81    mov     rbp, rsp
0000000100000e84    mov     qword [ss:rbp+var_8], rdi
0000000100000e88    mov     qword [ss:rbp+var_10], 0x0
0000000100000e90    mov     qword [ss:rbp+var_18], 0x0
0000000100000e98    mov     qword [ss:rbp+var_18], 0x0

0000000100000ea0    mov     rax, qword [ss:rbp+var_18]
0000000100000ea4    cmp     rax, qword [ss:rbp+var_8]
0000000100000ea8    jge     0x100000f1a

0000000100000eae    mov     rax, 0x2
0000000100000eb8    mov     rcx, qword [ss:rbp+var_18]
0000000100000ebc    mov     qword [ss:rbp+var_20], rax
0000000100000ec0    mov     rax, rcx
0000000100000ec3    cqo
0000000100000ec5    mov     rcx, qword [ss:rbp+var_20]
0000000100000ec9    idiv    rcx
0000000100000ecc    cmp     rdx, 0x0
0000000100000ed3    jne     0x100000ef0

0000000100000ed9    mov     rax, qword [ss:rbp+var_10]
0000000100000edd    shl     rax, 0x1
0000000100000ee1    add     rax, 0x2
0000000100000ee7    mov     qword [ss:rbp+var_10], rax
0000000100000eeb    jmp     0x100000f02

0000000100000ef0    mov     rax, qword [ss:rbp+var_10]
0000000100000ef4    shl     rax, 0x1
0000000100000ef8    add     rax, 0x1
0000000100000efe    mov     qword [ss:rbp+var_10], rax

0000000100000f02    jmp     0x100000f07

0000000100000f07    mov     rax, qword [ss:rbp+var_18]
0000000100000f0b    add     rax, 0x1
0000000100000f11    mov     qword [ss:rbp+var_18], rax
0000000100000f15    jmp     0x100000ea0

0000000100000f1a    mov     rax, qword [ss:rbp+var_10]
0000000100000f1e    pop     rbp
0000000100000f1f    ret
```

Towards a high level semantics

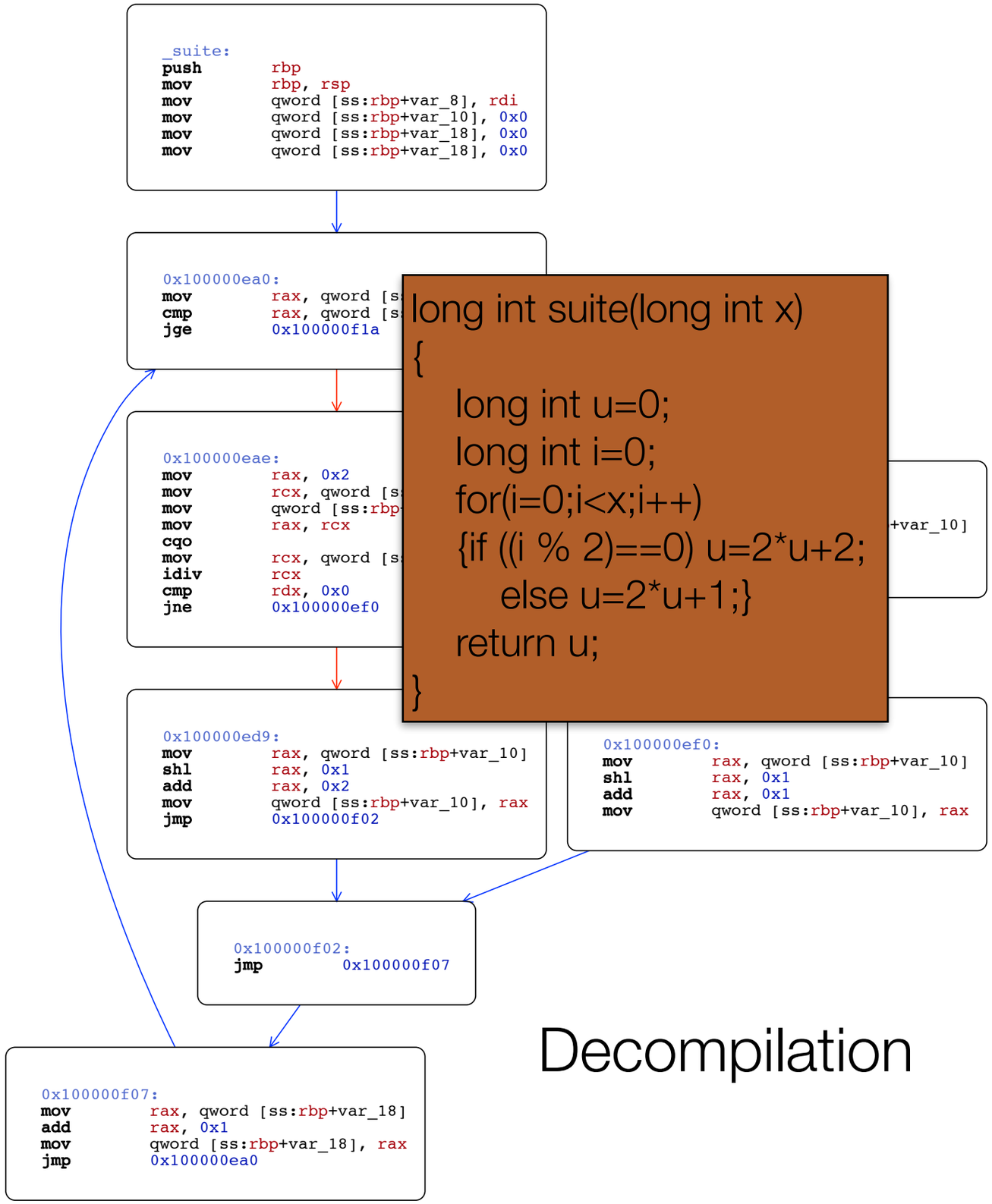
```
E80 : 55 48 89 E5 48 89 7D F8 48 C7 45 F0 00 00 00 00 48 C7 45 E8 00 00
E90 : 00 00 48 C7 45 E8 00 00 00 00 48 8B 45 E8 48 3B 45 F8 0F 8D 6C 00
EA0 : 00 00 48 B8 02 00 00 00 00 00 00 00 48 8B 4D E8 48 89 45 E0 48 89
EB0 : C8 48 99 48 8B 4D E0 48 F7 F9 48 81 FA 00 00 00 00 0F 85 17 00 00
EC0 : 00 48 8B 45 F0 48 C1 E0 01 48 05 02 00 00 00 48 89 45 F0 E9 12 00
ED0 : 00 00 48 8B 45 F0 48 C1 E0 01 48 05 01 00 00 00 48 89 45 F0 E9 00
EE0 : 00 00 00 48 8B 45 E8 48 05 01 00 00 00 48 89 45 E8 E9 86 FF FF FF
EF0 : 48 8B 45 F0 5D C3
```



Control Flow Graph

Towards a high level semantics

```
0000000100000e80    push    rbp
0000000100000e81    mov     rbp, rsp
0000000100000e84    mov     qword [ss:rbp+var_8], rdi
0000000100000e88    mov     qword [ss:rbp+var_10], 0x0
0000000100000e90    mov     qword [ss:rbp+var_18], 0x0
0000000100000e98    mov     qword [ss:rbp+var_18], 0x0

0000000100000ea0    mov     rax, qword [ss:rbp+var_18]
0000000100000ea4    cmp     rax, qword [ss:rbp+var_8]
0000000100000ea8    jge     0x100000f1a

0000000100000eae    mov     rax, 0x2
0000000100000eb8    mov     rcx, qword [ss:rbp+var_18]
0000000100000ebc    mov     qword [ss:rbp+var_20], rax
0000000100000ec0    mov     rax, rcx
0000000100000ec3    cqo
0000000100000ec5    mov     rcx, qword [ss:rbp+var_20]
0000000100000ec9    idiv    rcx
0000000100000ecc    cmp     rdx, 0x0
0000000100000ed3    jne     0x100000ef0

0000000100000ed9    mov     rax, qword [ss:rbp+var_10]
0000000100000edd    shl     rax, 0x1
0000000100000ee1    add     rax, 0x2
0000000100000ee7    mov     qword [ss:rbp+var_10], rax
0000000100000eeb    jmp     0x100000f02

0000000100000ef0    mov     rax, qword [ss:rbp+var_10]
0000000100000ef4    shl     rax, 0x1
0000000100000ef8    add     rax, 0x1
0000000100000efe    mov     qword [ss:rbp+var_10], rax

0000000100000f02    jmp     0x100000f07

0000000100000f07    mov     rax, qword [ss:rbp+var_18]
0000000100000f0b    add     rax, 0x1
0000000100000f11    mov     qword [ss:rbp+var_18], rax
0000000100000f15    jmp     0x100000ea0

0000000100000f1a    mov     rax, qword [ss:rbp+var_10]
0000000100000f1e    pop     rbp
0000000100000f1f    ret
```

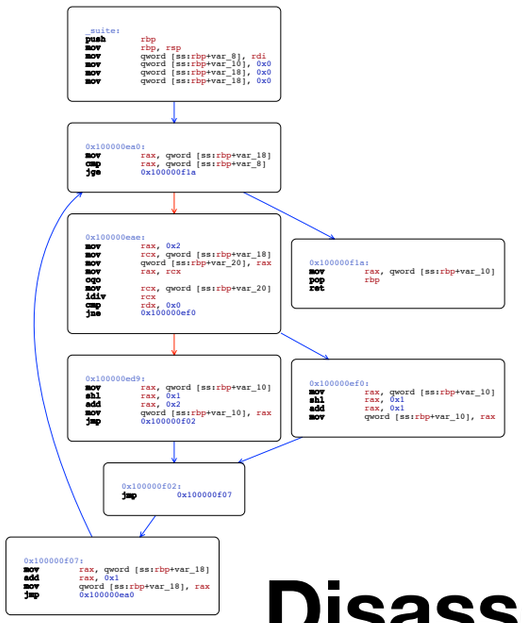


Decompilation

Abstraction by reverse-refinement

High-level semantics

```
long int suite(long int x)
{
   long int u=0;
   long int i=0;
   for(i=0;i<x;i++)
   {if ((i % 2)==0) u=2*u+2;
      else u=2*u+1;}
   return u;
}
```



Disassembly is a crucial step

```
0000000100000e80    push    rbp
0000000100000e81    mov     rbp, rsvp
0000000100000e84    mov     qword [ss:rbp+var_8], rdi
0000000100000e88    mov     qword [ss:rbp+var_10], 0x0
0000000100000e90    mov     qword [ss:rbp+var_18], 0x0
0000000100000e98    mov     qword [ss:rbp+var_18], 0x0
```

```
E80 : 55 48 89 E5 48 89 7D F8 48 C7 45 F0 00 00 00 00 48 C7 45 E8 00 00 E90 : 00 00 48
C7 45 E8 00 00 00 00 48 8B 45 E8 48 3B 45 F8 0F 8D 6C 00 EA0 : 00 00 48 B8 02 00 00 00
00 00 00 00 48 8B 4D E8 48 89 45 E0 48 89 EB0 : C8 48 99 48 8B 4D E0 48 F7 F9 48 81 FA
00 00 00 00 0F 85 17 00 00 EC0 : 00 48 8B 45 F0 48 C1 E0 01 48 05 02 00 00 00 48 89 45
F0 E9 12 00 ED0 : 00 00 48 8B 45 F0 48 C1 E0 01 48 05 01 00 00 00 48 89 45 F0 E9 00 EE0
: 00 00 00 48 8B 45 E8 48 05 01 00 00 00 48 89 45 E8 E9 86 FF FF FF EF0 : 48 8B 45 F0 5D
C3
```

Low-level semantics

Loria Laboratoire lorrain de recherche en informatique et ses applications

Jean-Yves Marion

UNIVERSITÉ DE LORRAINE

Context

Practical issues related to binary analysis

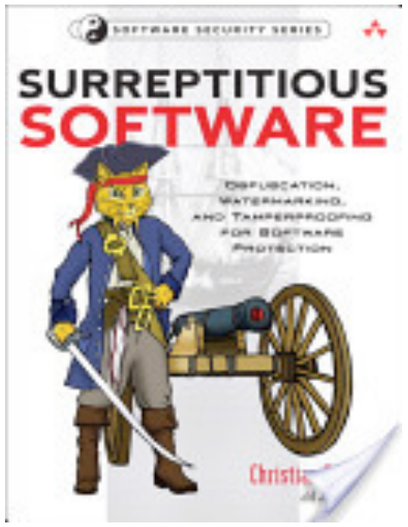
Code Protections against analysis

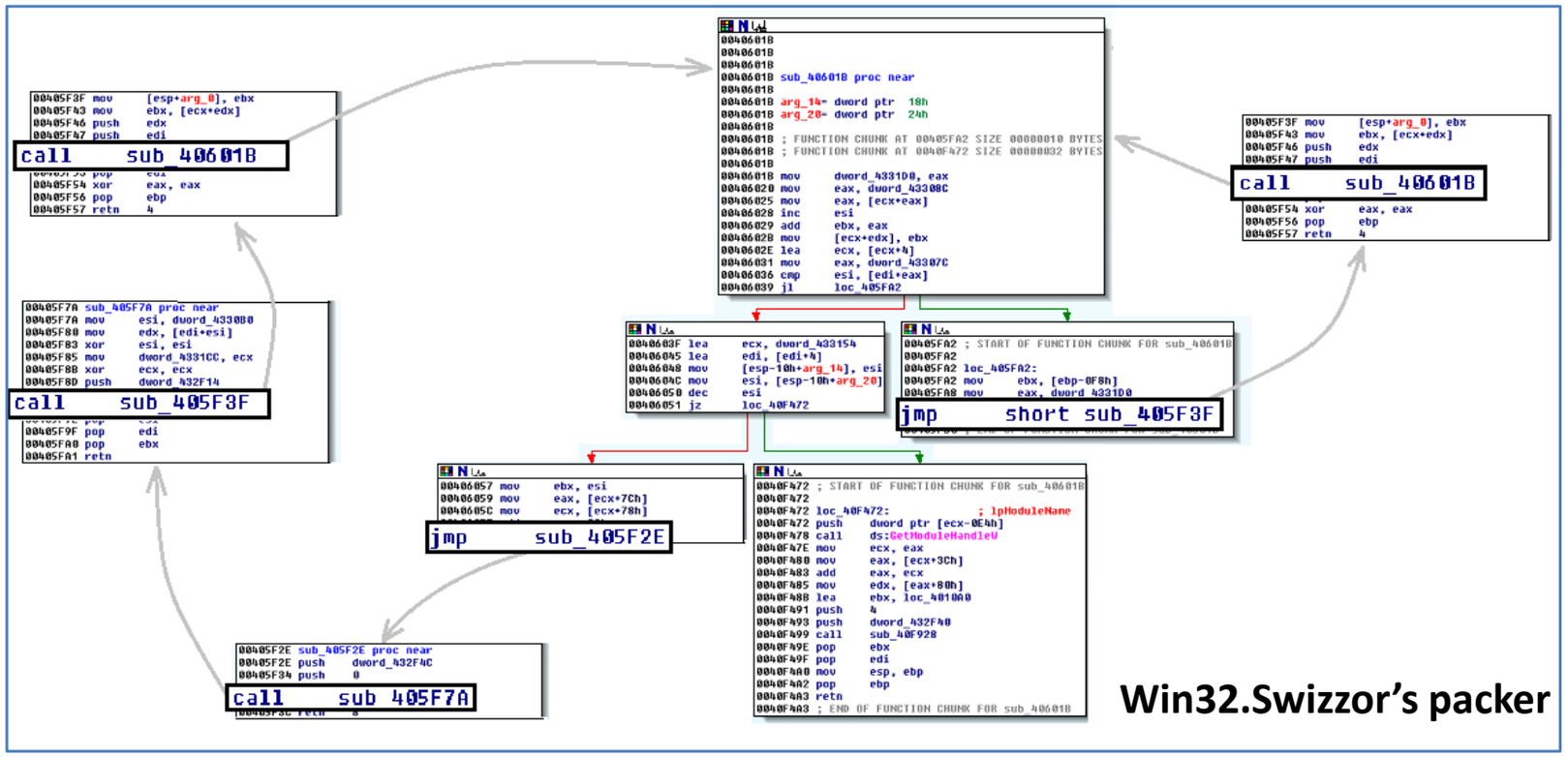
Loria
Laboratoire lorrain de recherche en informatique et ses applications

Jean-Yves Marion

UNIVERSITÉ DE LORRAINE

Malware protections





Win32.Swizzor's packer

1. Obfuscation
2. Cryptography
3. *Self-modification*
4. *Code overlapping*
5. Anti-analysis tricks

Malware analysis is very hard

Rational approach to help Felix the cat !

Loria Laboratoire lorrain de recherche en informatique et ses applications

Jean-Yves Marion

UNIVERSITÉ DE LORRAINE

A common protection scheme for malware

Decrypt

Decrypt

Decrypt

Wave 1		
01005000	pushfd	
01005001	push	0x3
01005003	jae	0x1005010
01005005	jmp	0x1005009
01005007	db	0x75 ; 'u'
01005008	db	0x75 ; 'u'
01005009	call	0x1005014
0100500e	xor	ax, 0xf773
01005012	jmp	0x1005031
01005014	add	esp, 0x4
01005017	jmp	0x100501b
01005019	db	0x75 ; 'u'
0100501a	db	0x75 ; 'u'
0100501b	dec	dword

Wave 2		
01007088	call	0x100708d
0100708d	sub	dword [ss:esp], 0x23a
01007094	jmp	dword [ss:esp+0x4]

.....

Wave 18
1006ba7 mov ebx, dword ptr [ebp+0x403783]
1006bad xor esi, esi
1006baf not ebx
1006bb1 or esi, ebx
1006bb3 jnz 0xa
1006bbd add ebx, dword ptr [ebp+0x403763]

This is a run of the **packer** Telock

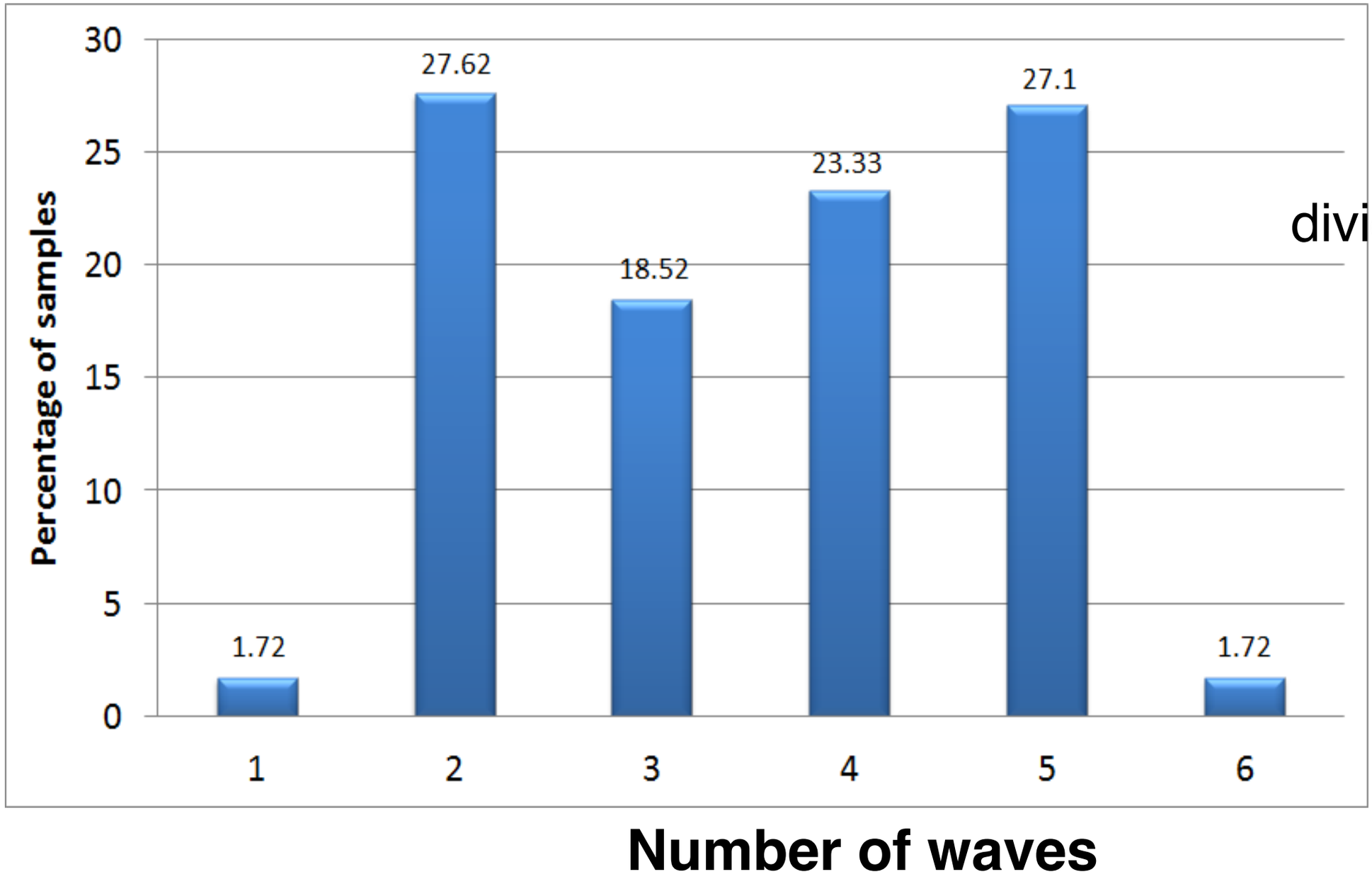
with 18 waves

payload

A run is a sequence of waves

Self-modifying program schema

Almost all malware is self-modifying !



600 samples
divided in 6 families

Based on 100 samples of Koobface, Conficker, Cutwail, Ramnit, Storm Worm, Waledac

See Virus Bulletin 2015

Problem
How to obtain a correct Control Flow Graph of a self-modifying code ?

Disassembly of self-modifying codes

Loria
Laboratoire lorrain de recherche en informatique et ses applications

Jean-Yves Marion

UNIVERSITÉ DE LORRAINE

Where are we ?

The input:
An obfuscated x86 binary file with no information except an entry point (e.g. a malware)

Can we recover the assembly code of the original program inside a binary file ?

- **when the binary file is self-modifying**

Goal
The disassembly of each wave of code of a self-modifying code (e.g. packers).

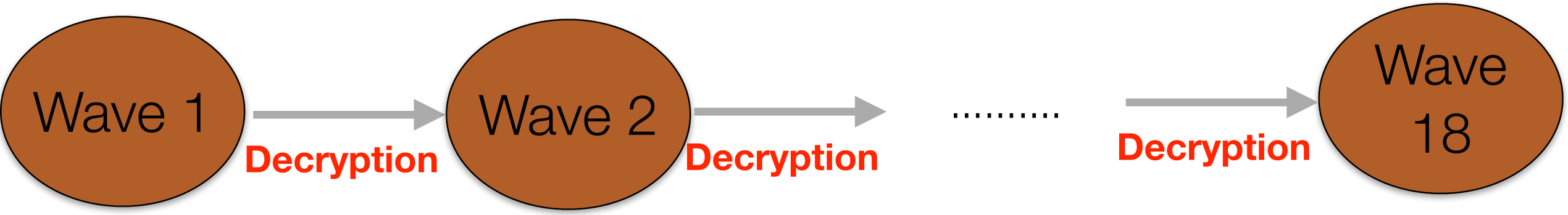
tELock genrates 18 waves

A payload installer

Unpack a tELock-file

The payload is decrypted in 0x01005090

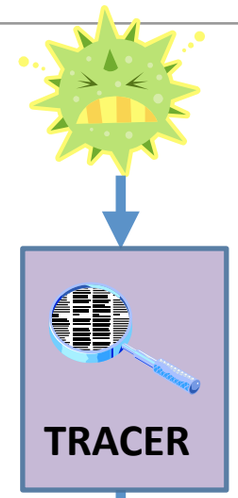
```
; data size
01006e62                mov ecx, 0x1dc2
; ebx is the pointer on the block to decrypt
; ebx=0x1005090
01006e67                inc ebx
01006e6f   loop:        rol byte ptr [ebx+ecx], 0x5
01006e73                add byte ptr [ebx+ecx], cl
01006e76                xor byte ptr [ebx+ecx], 0x67
01006e7a                inc byte ptr [ebx+ecx]
01006e7d                dec ecx
01006e80                jnle loop
; jump to the decrypted data
 01006e82               jmp 0x01005090
```



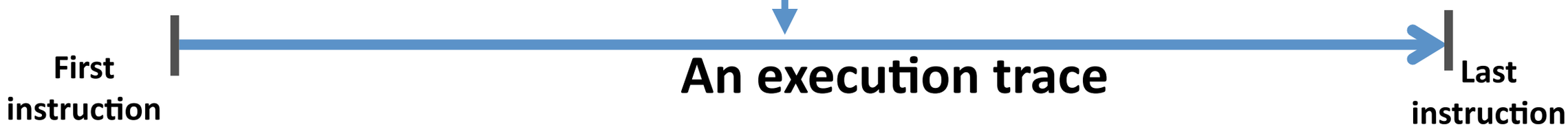
Self-modifying program schema

Wave extraction

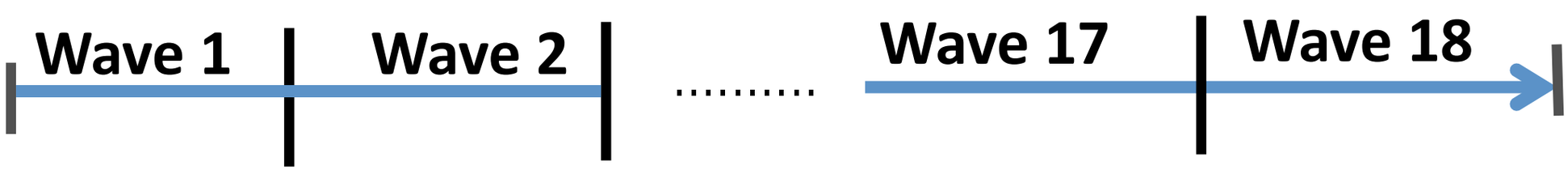
Secure sand boxing



Dynamic analysis



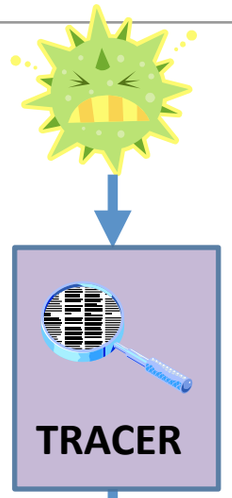
A trace indicates a sequence of code waves



Next Goal : Disassemble each PE-wave separately determined by the trace

TeLock with 18 waves

Secure sand boxing



Dynamic analysis

First instruction

An execution trace

Last instruction

A trace indicates a sequence of code waves

PE-Wave 1

PE-Wave 2

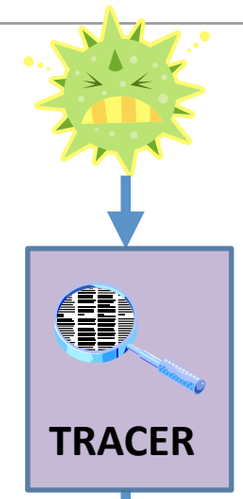
.....

PE-Wave 16

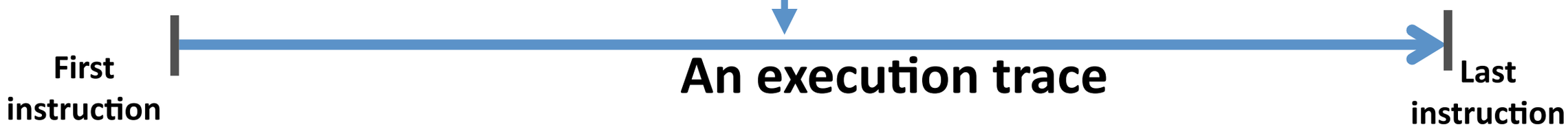
PE-Wave 18

Armadillo (V9) : 2 processes 11 threads

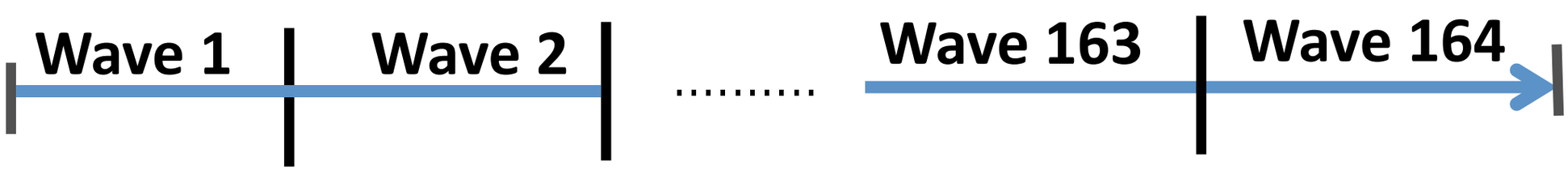
Secure sand boxing



Dynamic analysis



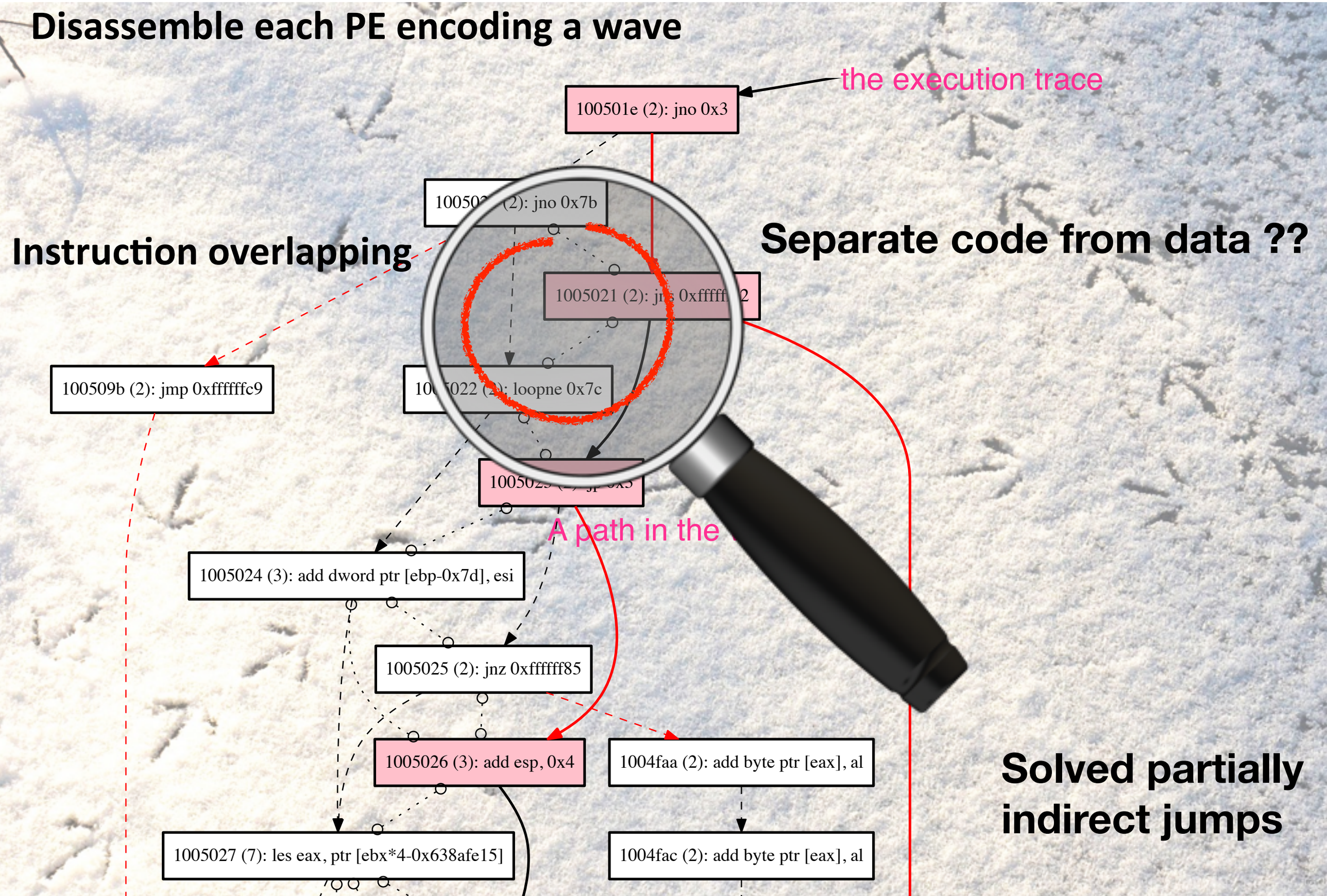
A trace indicates a sequence of code waves



PE-Wave 162 Contains the packed file

Wave disassembly

Disassemble each PE encoding a wave



Instruction overlapping

Separate code from data ??

Solved partially indirect jumps

Instruction overlapping in UPX packer

```
010059f0    89  f9                  mov ecx,edi
010059f2    79  07                  jnz +9
010059f4    0f  b7  07              movzx eax, word [edi]
010059f7    47                      inc edi
010059f8    50                      push eax
010059fa    b9  57  48  f2  ae      mov ecx, aef24857
010059ff    55                      push ebp
```

Inside Wave 1

Instruction overlapping in UPX packer

010059f0	89	f9				mov ecx,edi
010059f2	**79**	**07**				**jnz +9**
010059f4	0f	b7	07			movzx eax, word [edi]
010059f7	47					inc edi
010059f8	50					push eax
010059fa	b9	57	48	f2	ae	mov ecx, aef24857
010059fb		*57*				*push edi*
010059fc			*48*			*dec eax*
010059fd				*f2*	*ae*	*repne scasb*
010059ff	55					push ebp

overlapped instructions

Inside Wave 1

Jump Search View Debugger Options Windows Help

IDA - /Users/JYM/Recherche/Conférences/Exposé Virus/Botconf 2014/Archive/pack/upx-hostname.exe/upx-hostname.exe_wave0

No debugger

IDA View-A | Hex View-A | Structures | Enums | Imports

```
UPX1:010059CE                 or      eax, eax
UPX1:010059D0                 jz      short loc_1005A17
UPX1:010059D2                 mov     ebx, [edi+4]
UPX1:010059D5                 lea     eax, [eax+esi+541Ch]
UPX1:010059DC                 add     ebx, esi
UPX1:010059DE                 push    eax
UPX1:010059DF                 add     edi, 8
UPX1:010059E2                 call    dword ptr [esi+5494h]
UPX1:010059E8                 xchg    eax, ebp
UPX1:010059E9
UPX1:010059E9 loc_10059E9:                            ; CODE XREF: UPX1:01005A0F↓j
UPX1:010059E9                 mov     al, [edi]
UPX1:010059EB                 inc     edi
UPX1:010059EC                 or      al, al
UPX1:010059EE                 jz      short loc_10059CC
UPX1:010059F0                 mov     ecx, edi
UPX1:010059F2                 jns     short near ptr loc_10059FA+1
UPX1:010059F4                 movzx   eax, word ptr [edi]
UPX1:010059F7                 inc     edi
UPX1:010059F8                 push    eax
UPX1:010059F9                 inc     edi
UPX1:010059FA
UPX1:010059FA loc_10059FA:                            ; CODE XREF: UPX1:010059F2↑j
UPX1:010059FA                 mov     ecx, 0AEF24857h
UPX1:010059FF                 push    ebp
UPX1:01005A00                 call    dword ptr [esi+5498h]
UPX1:01005A06                 or      eax, eax
UPX1:01005A08                 jz      short loc_1005A11
UPX1:01005A0A                 mov     [ebx], eax
UPX1:01005A0C                 add     ebx, 4
UPX1:01005A0F                 jmp     short loc_10059E9
UPX1:01005A11 ; ---------------------------------------------------------------------------
UPX1:01005A11
UPX1:01005A11 loc_1005A11:                            ; CODE XREF: UPX1:01005A08↑j
UPX1:01005A11                 call    dword ptr [esi+54A8h]
```

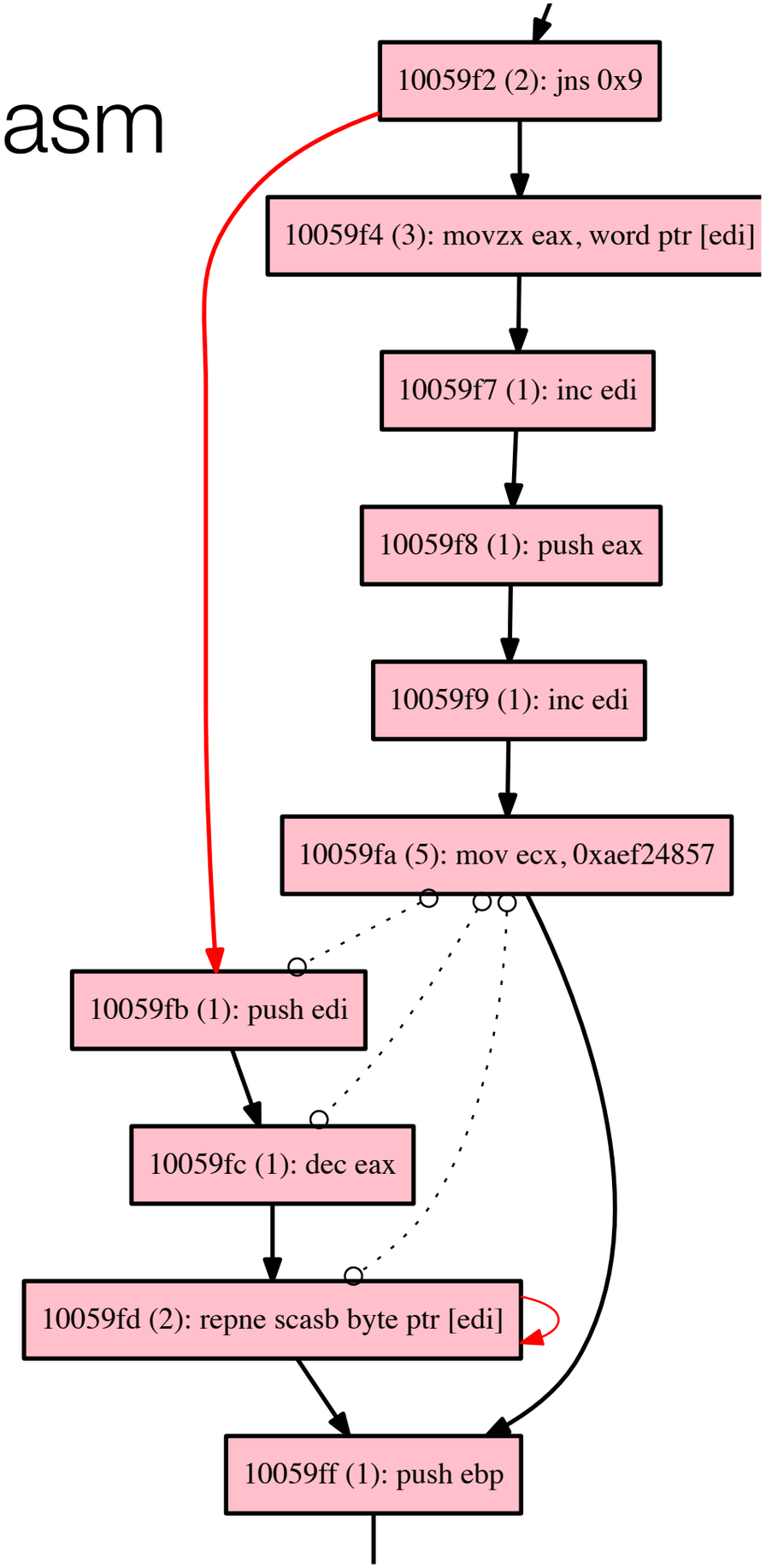
000059F0 010059F0: UPX1:010059F0

Missing code

```
gment  (01001000-01005000) ... ... OK
gment  (01005000-01006000) ... ... OK
ory...
ng convention by default
 SEH for vc7/8
equences...
se wait...ok
rche/Conférences/Exposé Virus/Botconf 2014/Archive/pack/upx-hostname.exe/upx-hostname.exe_wave0' has been successfully loaded into the database.
cations/IDA Starter 6.2/idaq.app/Contents/MacOS/idc/ida.idc'...
in'...
cations/IDA Starter 6.2/idaq.app/Contents/MacOS/idc/onload.idc'...
Load'...
nput file...
re the input file right now.
--------------------------------------------------------------------------------
 Mar  9 2014, 22:16:00)
Apple LLVM 5.0 (clang-500.0.68)]
 (serial 0) (c) The IDAPython Team <idapython@googlegroups.com>
--------------------------------------------------------------------------------
 SEH for vc7/8
```

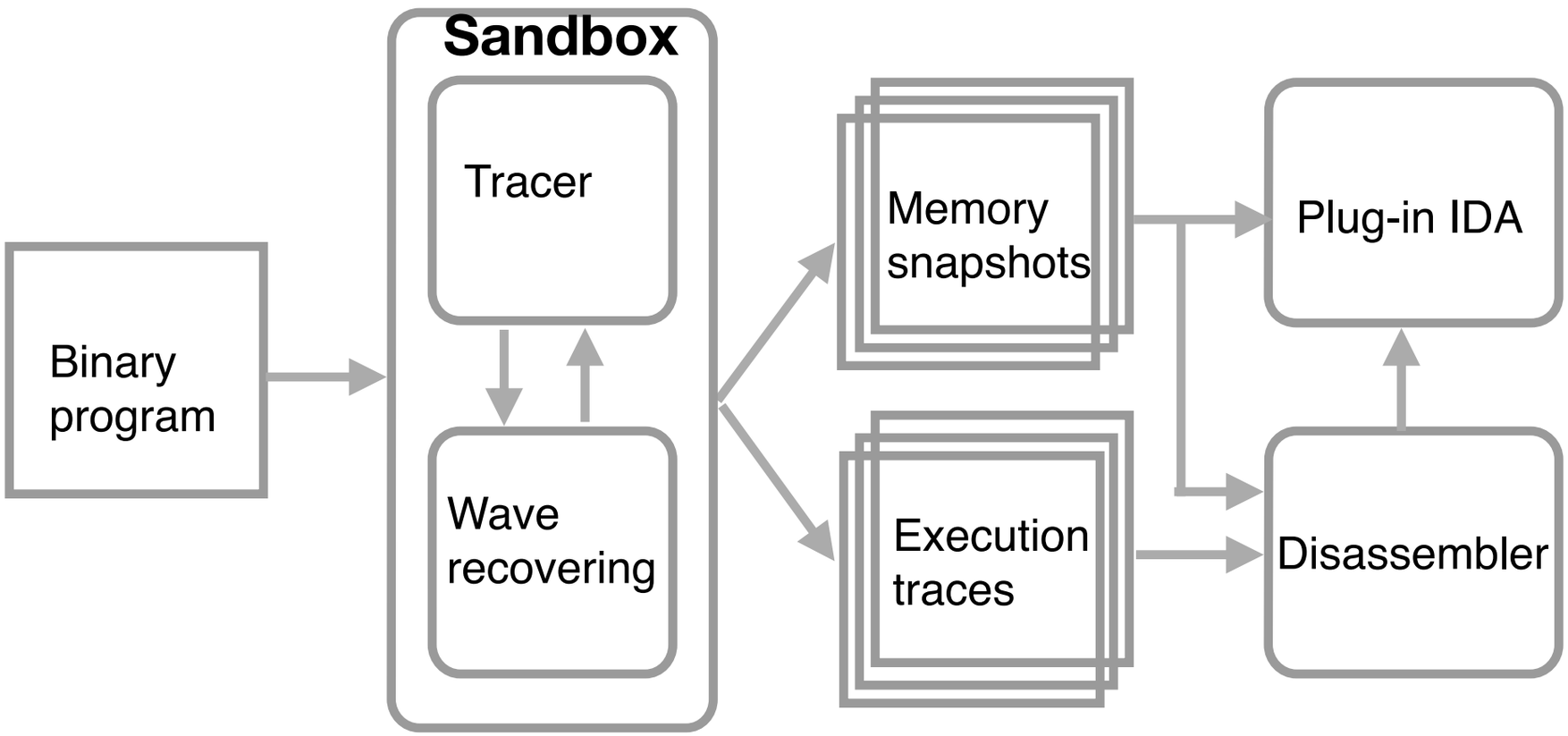
Screenshot of Codisasm

Overlapping



The overall architecture of the disassembler

We combine concrete path execution and static analysis



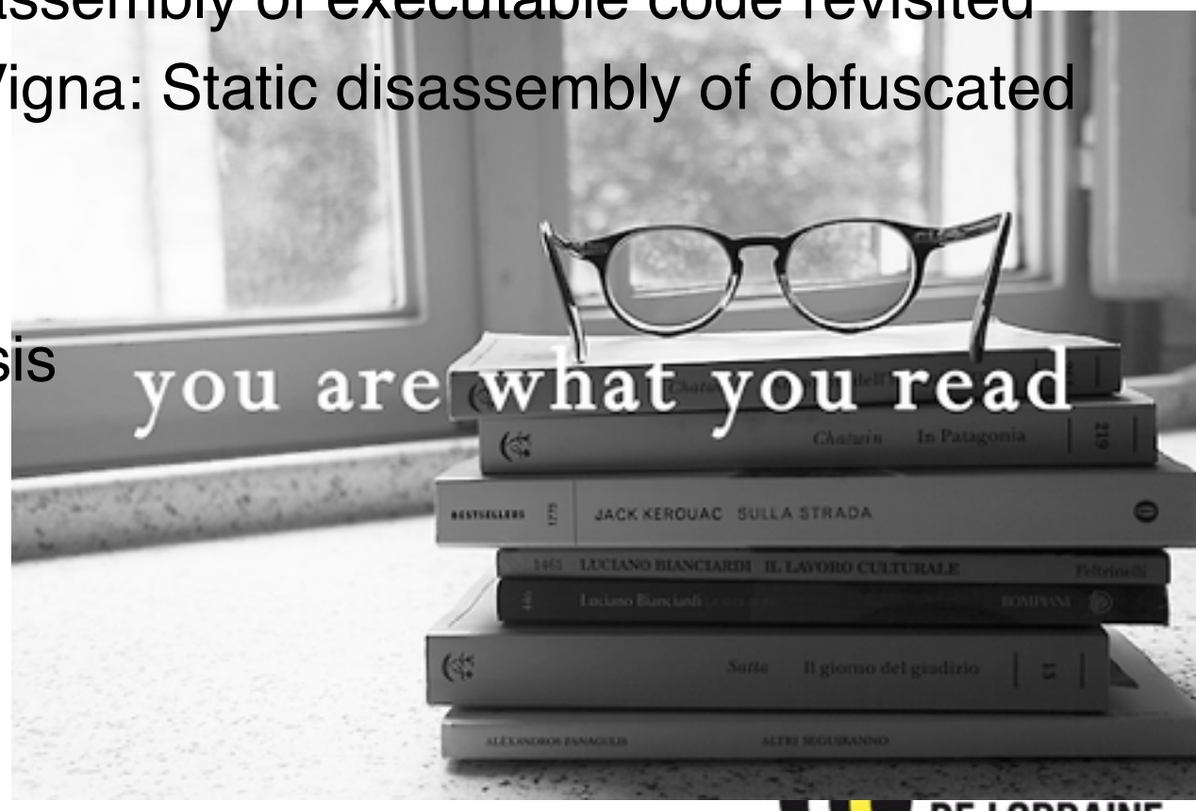
Worflow :

x86 file → Server → compact traces + Memory dumps → Disassemble

(available at http://marionjy.loria.fr/)

We are what we read !

- J. Caballero, N. M. Johnson, S. Mccamant, and D. Song: Binary code extraction and interface identification for security applications
- J. Kinder, F. Zuleger, and H. Veith: An abstract interpretation-based framework for control flow reconstruction from binaries
- T. W. Reps and G. Balakrishnan: Improved memory-access analysis for x86 executables
- S. Debray and J. Patel: Reverse engineering self-modifying code: Unpacker extraction
- N. Krishnamoorthy, S. Debray, and K. Fligg: Static detection of disassembly errors
- B. Schwarz, S. Debray, and G. Andrews: Disassembly of executable code revisited
- C. Kruegel, W. Robertson, F. Valeur, and G. Vigna: Static disassembly of obfuscated binaries
- J. Kinder. Static analysis of x86 executables
- G. Vigna. Static disassembly and code analysis



Pitfalls :

- Call/ret obfuscations
- Indirect jumps
- Opaque Predicates

Another key issue : Big code !

Answers to these questions should be automatic and should deal with vast amounts of samples

- *about 6 millions of samples inside our lab*
- *about **300 000** files by day for Google via VirusTotal !*
- *400 millions of samples for Google*

80 hours (=3 days) to analyze
300 000 files
with a time put of 1 sec. by file

Summary : Codisasm

- A model of self-modifiying computations based on Waves
- Deals with Self modifying codes
- Deals with overlapping instructions
- Follows processes and implements a sequence of anti-anti-debugging tricks
  - *The last wave of Mystic packer generates 3 processes*

Packer disassembly

Packer name	#proc.	#thr.	#Wave	DM
ACProtect v2.0	1	1	635	N
Armadillo v9.64	2	11	165	Y
Aspack v2.12	1	1	3	N
BoxedApp v3.2	1	15	6	Y
EP Protector v0.3	1	1	2	N
Expressor	1	1	2	N
FSG v2.0	1	1	2	N
JD Pack v2.0	1	1	3	N
MoleBox	1	1	3	N
Mystic	1	1	4	Y
Neolite v2.0	1	1	2	N
nPack v1.1.300	1	1	2	N
Packman v1.0	1	1	2	N
PE Compact v2.20	1	1	4	Y
PECrypt V1.02	1	4	99	Y
PE Lock	1	1	15	Y
PE Spin v1.1	1	1	80	Y
Petite v2.2	1	1	3	N
RLPack	1	1	2	N
Setisoft v2.7.1	1	5	32	Y
TELock v0.99	1	1	18	Y
Themida v2.0.3.0	1	28	106	Y
Upack v0.39	1	1	3	N
Upx v2.90	1	1	2	N
VM Protect v1.50	1	1	1	N
WinUPack	1	1	3	N
Yoda's Crypter v1.3	1	1	4	Y
Yoda's Protector v1.02	1	1	6	N

Epilogue : a malware writer scenario

1. We sent the backdoor hupigon.eyf toVirusTotal :
   - **45 AV detected hupigon.eyf over 57 AV**

virustotal (/fr/) Français Rejoindre notre communauté Se connecter

SHA256: fdd726007bdec7ef8a22e79f2f9e7115a86f1271f0be583858e4d09546860d33

Nom du fichier : Hupigon.eyf

Ratio de détection : 45 / 57

Date d'analyse : 2015-05-16 14:05:50 UTC (il y a 4 mois, 3 semaines)

0 0

Analyse | File detail | Informations supplémentaires | Commentaires 1 | Votes | Informations comportementales

Antivirus	Résultat	Mise à jour
ALYac	Backdoor.Agent.YRI	20150516
AVG	BackDoor.Hupigon5.RUD	20150516
AVware	Trojan.Win32.Generic!BT	20150516

Epilogue : a malware writer scenario

1.We sent the backdoor hupigon.eyf toVirusTotal :

- **45 AV detected hupigon.eyf over 57 AV**

2. We packed hupigon.eyf with Mystic packer: *Mystic[hupigon.eyf]*

3. **Only 22 AV** identified *Mystic[hupigon.eyf]* has malicious **without identifying** hupigon.eyf

virustotal (/fr/)

Français Rejoindre notre communauté Se connecter

SHA256:	601231c1b38053337822369de812b3e7f47811baa45192cfc59636c3a3289ed0
Nom du fichier :	codisasm-mystic-Hupigon.eyf
Ratio de détection :	22 / 57
Date d'analyse :	2015-05-16 16:25:43 UTC (il y a 4 mois, 3 semaines) Voir les derniers (/fr/file/601231c1b38053337822369de812b3e7f47811baa45192cfc59636c3a3289ed0/analysis/)

0 0

Analyse | File detail | Informations supplémentaires | Commentaires 0 | Votes | Informations comportementales

Antivirus	Résultat	Mise à jour
ALYac	Gen:Heur.Zygug.1	20150516
AVG	Win32/Cryptor	20150516

Loria Laboratoire lor en informatiqu

UNIVERSITÉ DE LORRAINE

Epilogue : a malware writer scenario

1. We sent the backdoor hupigon.eyf toVirusTotal :
   - **45 AV detected hupigon.eyf over 57 AV**

2. We packed hupigon.eyf with Mystic packer: *Mystic[hupigon.eyf]*

3. **Only 22 AV** identified *Mystic[hupigon.eyf]* as malicious **without identifying** hupigon.eyf

4. We uses Codisasm on *Mystic[hupigon.eyf]*. We obtained 4 waves

5. We checked that **one of the processes in the last wave contains hupigon.eyf**

6. We sent the PE file of the last wave to VirusTotal : **20** AV identified correctly hupigon.eyf

virustotal (/fr/)

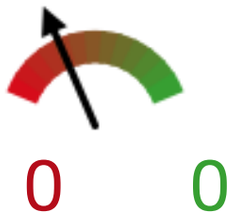
Français Rejoindre notre communauté Se connecter

SHA256: 42516a3a7cc6b33804c5a62c5e36e8e7ecb28a92c5416d5c772e601d7ad66003

Nom du fichier : mystic2-Hupigon.eyf_wave3-WithAllocSec

Ratio de détection : 20 / 57

Date d'analyse : 2015-05-16 16:38:34 UTC (il y a 4 mois, 3 semaines)



Analyse | File detail | Informations supplémentaires | Commentaires 0 | Votes

Antivirus	Résultat	Mise à jour
ALYac	Backdoor.Agent.YRI	20150516
AVG	BackDoor.Hupigon5.RUD	20150516
Ad-Aware	Backdoor.Agent.YRI	20150516
AhnLab-V3	Win-Trojan/Hupigon.34304.AA	20150516
Antiy-AVL	Trojan[Backdoor]/Win32.Hupigon	20150516
Avast	Win32:Neptunia-WC [Trj]	20150516

Questions ?

Code available at http://marionjy.loria.fr/

Thank you