Data flow analysis in order to construct control flow graphs of obfuscated x86 binary codes

Jean-Yves MARION - Université de Lorraine, Loria
Sylvain CECCHETTO - Cyber-Detect, Nancy

France-Japan
Cybersecurity workshop
February 25 and 26, 2021

Data flow analysis in order to construct control flow graphs of obfuscated x86 binary codes

Jean-Yves MARION - Université de Lorraine, Loria
Sylvain CECCHETTO - Cyber-Detect, Nancy

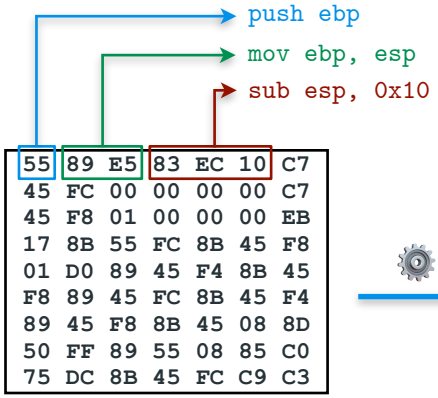
France-Japan
Cybersecurity workshop
February 25 and 26, 2021

Introduction

Binary analysis

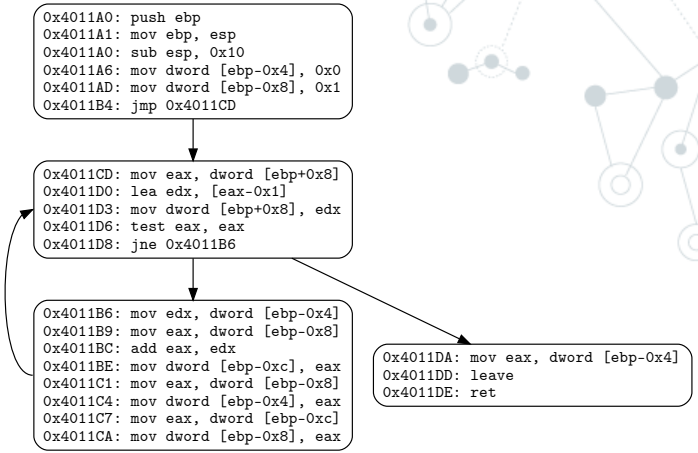
Disassembly and control flow graph

[Schwarz, Debray and Andrews 2002], [Shoshitaishvili, Wang, Salls, Stephens, Polino, Dutcher, Grosen, Feng, Hauser, Kruegel and Vigna 2016] [Biondi, Rigo, Zennou and Mehrenberger 2017]



Disassembly

```
push ebp
mov ebp, esp
sub esp, 0x10
mov dword [ebp-0x4], 0x0
mov dword [ebp-0x8], 0x1
jmp 0x2d
mov edx, dword [ebp-0x4]
mov eax, dword [ebp-0x8]
add eax, edx
mov dword [ebp-0xc], eax
mov eax, dword [ebp-0x8]
mov dword [ebp-0x4], eax
mov eax, dword [ebp-0xc]
mov dword [ebp-0x8], eax
mov eax, dword [ebp+0x8]
lea edx, [eax-0x1]
mov dword [ebp+0x8], edx
test eax, eax
jne 0x16
mov eax, dword [ebp-0x4]
leave
ret
```



Problems

- Variable length instructions
- Mixed data in code bytes
- Indirect jumps
- …

Example : disassembly of EB0168C3909090 code

```
0x0: EB01        jmp  0x3
0x2: 68C3909090  push 0x909090c3
```

```
0x0: EB01  jmp  0x3
0x2: 68    ;data
0x3: C3    ret
0x4: 90    nop
0x5: 90    nop
0x6: 90    nop
```

Obfuscation

[Collberg, Thomborson and Low 1997]

Transforming a program P into a program P' so that P' works like P but is harder to understand

Build the control flow graph of a binary

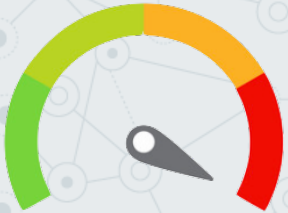
Build the control flow graph of a binary

Malwares are protected (obfuscations)

Build the control flow graph of a binary

Malwares are protected (obfuscations)

Build the **control flow graph** of a **malware**

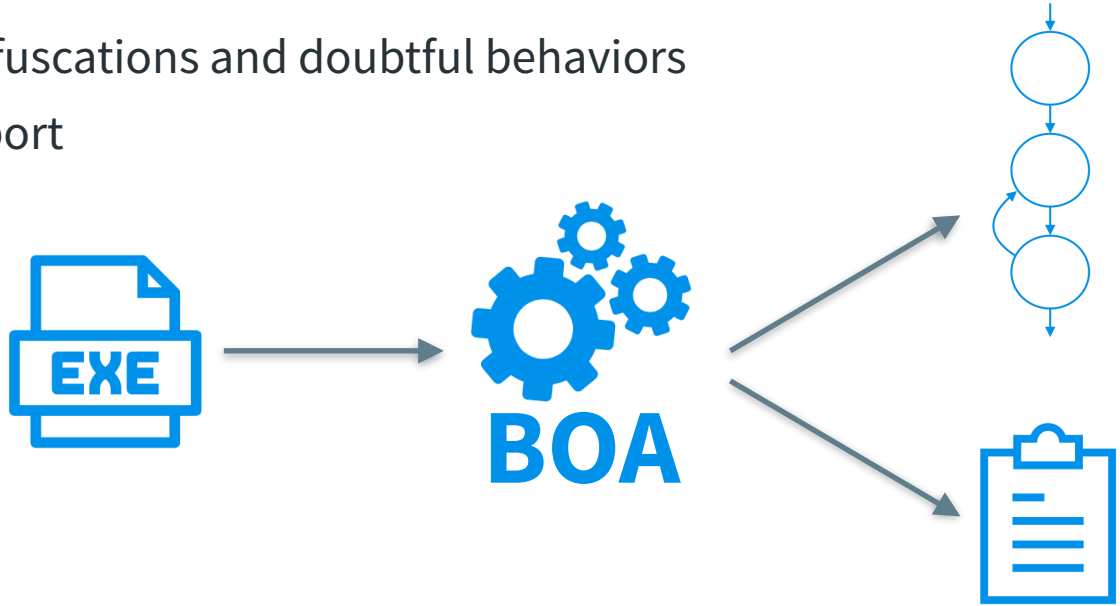


BOA - platform

Goals

Disassemble and build CFG of obfuscated binaries

- **Main goal:**
  - Disassemble and build CFG of obfuscated binaries
- **Others goals:**
  - Detect obfuscations and doubtful behaviors
  - Create report



*An approach combining the **advantages** of a **static analysis** with the **strengths** of a **dynamic analysis**?*

*An approach combining the **advantages** of a **static analysis** with the **strengths** of a **dynamic analysis**?*

➡ **Symbolic execution**…

… obfuscation-resistant

Machine state

⊥ : unknown value

- Given by the triplet $(\texttt{ip}, \sigma, p)$, with:

$\texttt{ip}$ Instruction pointer

$$\sigma: \begin{cases} r \in \mathbf{Reg} \mapsto v \in \mathbf{Addr} \cup \{ \bot \} \\ f \in \mathbf{Flag} \mapsto v \in \{0,1,\bot\} \\ a \in \mathbf{Addr} \mapsto v \in \mathbf{Bytes} \cup \{ \bot \} \end{cases}$$

Value of CPU, flags and memory cells

$$p : a \in \mathbf{Addr} \mapsto p \in \{\varnothing, R, RX, RW, RWX\}$$

Memory cells permissions

Execution

$$(\texttt{ip}, \sigma, p) \rightarrow (\texttt{ip}', \sigma', p') =_{def} \begin{cases} (\texttt{ip}, \sigma, p) \xrightarrow{\sigma[\texttt{ip}]^*} (\texttt{ip}', \sigma', p') \\ (\texttt{ip}, \sigma, p) \xrightarrow{\mathcal{O}} \mathcal{O}(\texttt{ip}, \sigma, p) \end{cases}$$

Normal mode, when $\sigma[\texttt{ip}]^*$ can be executed

Else, kernel mode

Self-modification support :

- Instructions live in σ
- σ can be modified by $\sigma[\texttt{ip}]^*$

BOA
In a nutshell

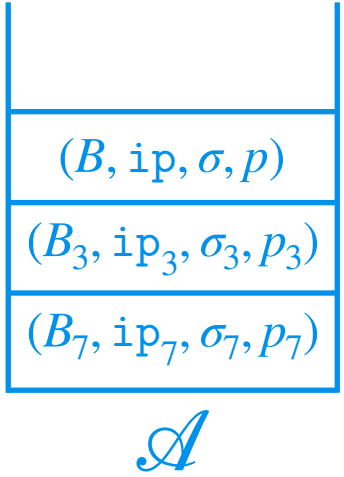
Data flow analysis to rescue in order to
build the control flow graph of obfuscated binaries

- **Analysis approach:** recursive disassembling + symbolic execution at basic block level
- Resolve **indirect jumps** by partial **machine state computation**
- **Multiple obfuscation support:**
  - Self-modification
  - Exceptions
  - Indirect jumps and call stack tampering
  - Opaque predicate and dead branches
  - On the fly import table construction

BOA

Algorithm : main loop

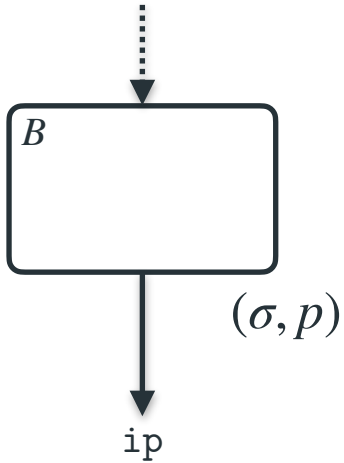
```
while 𝒜:
```

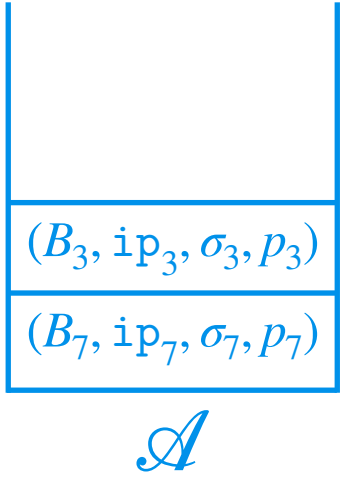


BOA

Algorithm : main loop

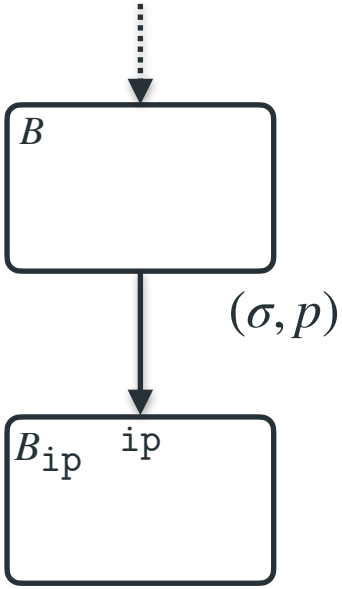
```
while 𝒜:
// Retrieve a 4-uplet to be processed
  |   (B, ip, σ, p) = 𝒜.pop()
```

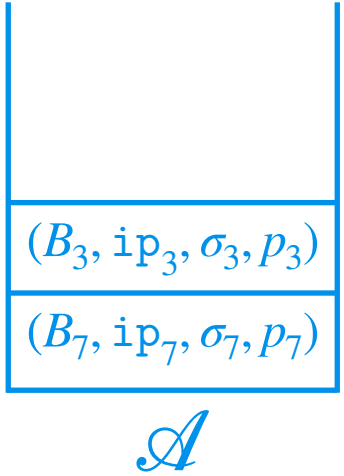




BOA
Algorithm : main loop

```
while 𝒜:
// Retrieve a 4-uplet to be processed
  |   (B, ip, σ, p) = 𝒜.pop()
// Disassemble ip basic block
  |   B_ip = DisasBasicBlock(ip)
```

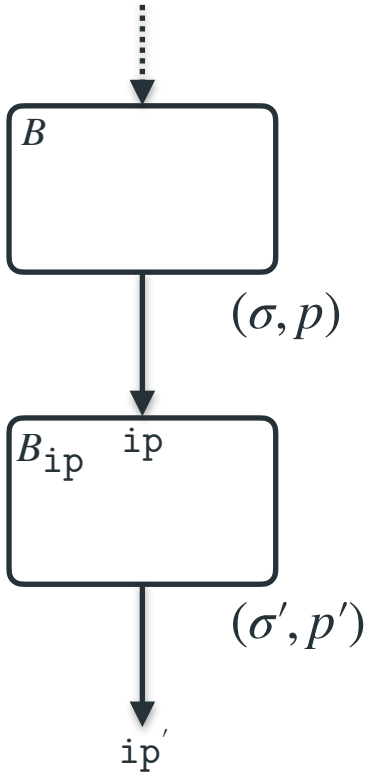


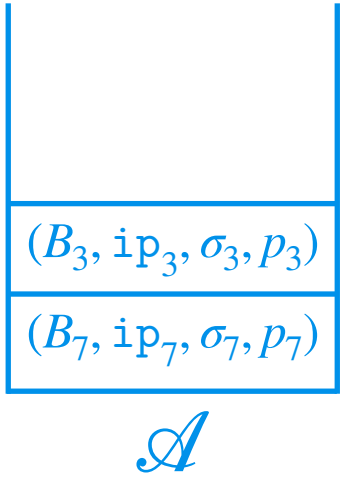


BOA

Algorithm : main loop

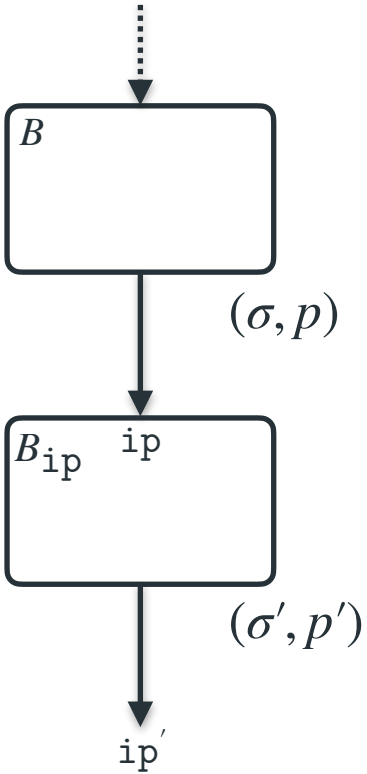
```
while 𝒜:
// Retrieve a 4-uplet to be processed
  |   (B, ip, σ, p) = 𝒜.pop()
// Disassemble ip basic block
  |   B_ip = DisasBasicBlock(ip)
// Apply symbolic execution of B_ip in σ and p
  |   (ip', σ', p') = MachineState(B_ip, σ, p)
```

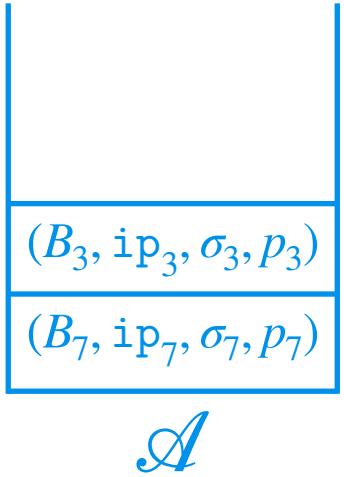




BOA
Algorithm : main loop

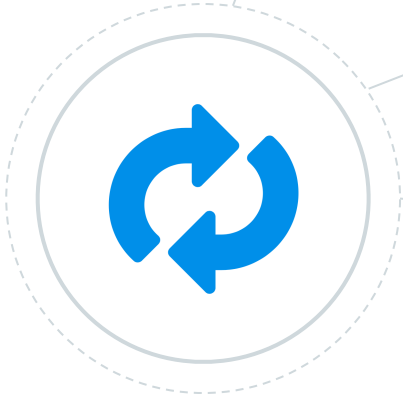
while $\mathscr{A}$:
// Retrieve a 4-uplet to be processed
  | $(B, \texttt{ip}, \sigma, p) = \mathscr{A}.\texttt{pop}()$
// Disassemble ip basic block
  | $B_{\texttt{ip}}$ = DisasBasicBlock(ip)
// Apply symbolic execution of $B_{\texttt{ip}}$ in σ and p
  | $(\texttt{ip}', \sigma', p')$ = MachineState($B_{\texttt{ip}}, \sigma, p$)
// Add new 4-uplet in $\mathscr{A}$
  | if $\texttt{ip}' \neq \bot$:
  | | $\mathscr{A}.\texttt{push}((B_{\texttt{ip}}, \texttt{ip}', \sigma', p'))$
  | else if $type(B_{\texttt{ip}})$ == COND:
  | | $\mathscr{A}.\texttt{push}((B_{\texttt{ip}}, next\_mem, \sigma', p'))$
  | | $\mathscr{A}.\texttt{push}((B_{\texttt{ip}}, target, \sigma', p'))$





BOA - applications

Self-modifications



- **Two goals**
  - Detect self-modification
  - Continue self-modification

BOA
Self modification and execution waves

- **Detect :** [Bonfante, Fernandez, Marion, Rouxel, Sabatier et Thierry 2015]
  - Keep a list of modified memory cells
  - Check every instruction before symbolic execution
- **Handle :**
  - Concept of execution waves
  - Each instruction is associated to a wave

BOA
Self modification and execution waves

- **Detect :** [Bonfante, Fernandez, Marion, Rouxel, Sabatier et Thierry 2015]
  - Keep a list of modified memory cells
  - Check every instruction before symbolic execution
- **Handle :**
  - Concept of execution waves
  - Each instruction is associated to a wave

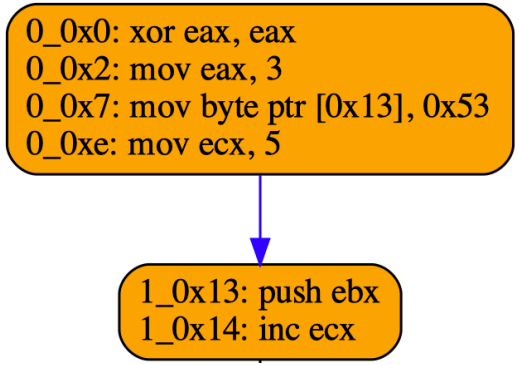
```
main:
        xor eax, eax
        mov eax, 0x3
        mov BYTE [A], 0x53
        mov ecx, 0x5
A       push eax
        inc ecx
```

⇦ self-modification (`@[A]` replaced by `0x53`)

⇦ self-modified instruction

Self modification and execution waves

- **Detect :** [Bonfante, Fernandez, Marion, Rouxel, Sabatier et Thierry 2015]
  - Keep a list of modified memory cells
  - Check every instruction before symbolic execution
- **Handle :**
  - Concept of execution waves
  - Each instruction is associated to a wave




```
main:
        xor eax, eax
        mov eax, 0x3
        mov BYTE [A], 0x53
        mov ecx, 0x5
A       push eax
        inc ecx
```

⇦ self-modification (`@[A]` replaced by `0x53`)

⇦ self-modified instruction

Type d'analyse :	Statique	Dynamique								
Type de détection :		Instruction						Page mémoire	Appel système	Table d'import
Packer	**BOA**	**CoDisasm** [15]	**PinSec** [8]	**Poly-Unpack** [61]	**Renovo** [41]	**PinDemonium** [49]	**RePEconstruct** [45]	**Omni-Unpack** [50]	**Eureka** [66]	**Bin-Unpack** [20]
ACPacker	12/✓/×	635/✓/✓	-	-	-	-	-	-	-	-
ACProtect 2.0	811/✓/×	971/✓/✓	4/✓/×	-	-	-/✓	-	-/✓	-	-/✓
Armadillo 3.78	×/×	-	1/✓/×	✓/×	×/×	-	-	-/~	-/✓	-/✓
Armadillo 9.64	×/×	165/✓/✓	-	-	-	-	-	-	-	-
ASPack 2.12	3/✓/✓	3/✓/✓	2/✓/✓	✓/~	✓/✓	-/✓	-	-/✓	-/✓	-/✓
ASPack 2.40	3/✓/✓	3/✓/✓	-	-	-	-	-	-	-	-
EP Protector 0.3	2/✓/✓	2/✓/✓	1/✓/✓	-	-	-	-	-	-	-
eXPressor	2/✓/✓	2/✓/✓	1/✓/✓	-	-	-/✓	-	-	-	-/✓
FSG 2.0	2/✓/✓	2/✓/✓	1/✓/✓	✓/✓	✓/✓	-/✓	×/×	-/✓	-/✓	-/✓
JDPack 2.0	2/✓/×	3/✓/✓	×/×	-	-	-	-	-	-	-
MEW	2/✓/✓	2/✓/✓	1/✓/✓	✓/✓	✓/✓	-/✓	-	-/✓	-/✓	-/✓
MoleBox	2/✓/×	3/✓/✓	2/✓/✓	×/×	✓/✓	-	✓/✓	-/✓	-/✓	-/✓
Morphine 2.7	2/✓/×	3/✓/✓	-	✓/✓	✓/~	-	-	-/✓	-/✓	-
Mystic	×/×	4/✓/✓	1/✓/✓	-	-	-	-	-	-	-
Neolite 2.0	×/×	2/✓/✓	1/✓/✓	-	-	-	✓/✓	-	-	-
nPack 1.1.300	2/✓/✓	2/✓/✓	1/✓/✓	-	-	-	-	-	-	-/✓
Obsidium 1364	6/✓/×	16/✓/✓	×/×	×/×	×/×	-/✓	-	-	-/✓	-/✓
Packman 1.0	2/✓/✓	2/✓/✓	1/✓/✓	-	-	-	✓/✓	-	-	-
PE Compact 2.20	4/✓/✓	4/✓/✓	1/✓/✓	×/×	✓/✓	-/✓	✓/✓	-	-/✓	-/✓
PE Compact 3.02.2	4/✓/✓	4/✓/✓	-	-	-	-	-	-	-	-
PELock	2/✓/×	15/✓/×	6/✓/✓	-	-	-/✓	-	-	-	-/✓
PESpin 1.1	5/✓/×	80/✓/×	×/×	-	-	-	✓/×	-	-	-/✓
Petite 2.2	2/✓/×	3/✓/✓	×/×	-	-	-	✓/✓	-	-	-/✓
RLPack	2/✓/✓	2 /✓/✓	1/✓/✓	-	-	-	-	-	-	-/✓
Setisoft 2.7.1	×/×	32/✓/×	4/✓/×	-	-	-	-	-	-	-
SVK 1.43f	2/✓/×	35/✓/×	×/×	-	-	-	-	-	-	-
tElock 0.51	5/✓/×	17/✓/✓	5/✓/×	-	-	-	✓/×	-/×	-	-/✓
tElock 0.99	14/✓/×	18 /✓/✓	-	-	-	-	-	-	-	-
Themida 1.8	3/✓/×	-	×/×	×/×	✓/~	-	✓/×	-/✓	-/~	-/✓
Themida 2.0.3	3/✓/×	106 /✓/✓	-	-	-	-	-	-	-	-
Upack 0.39	2/✓/×	3/✓/✓	2/✓/✓	-	-	-	-	-	-	-
UPX 2.90	2/✓/✓	2/✓/✓	1/✓/✓	✓/✓	✓/✓	-/✓	✓/✓	-/✓	-/✓	-
WinUpack	3/✓/✓	3/✓/✓	2/✓/✓	✓/~	✓/✓	-/✓	×/×	-/✓	-/✓	-/✓
Yoda's Crypter 1.3	4/✓/✓	4/✓/✓	3/✓/×	-	-	-	-	-	-	-/✓
Yoda's Protector 1.02	2/✓/×	6/✓/✓	×/×	✓/~	✓/~	-	-	-	-/✓	-/✓

$n/d/r$: nombre de vagues détectées (n) / code auto-modifiant détecté (d) / récupération du code du binaire original (r)

✓ : réussite, ~ : réussite partielle, × : échec, - : absence de résultat

La version exacte des packers utilisés est seulement connue pour BOA, CoDisasm et PinSec.

Type d'analyse :	Statique	Dynamique								
Type de détection :		Instruction						Page mémoire	Appel système	Table d'import
Packer	**BOA**	**CoDisasm** [15]	**PinSec** [8]	**Poly-Unpack** [61]	**Renovo** [41]	**PinDemonium** [49]	**RePEconstruct** [45]	**Omni-Unpack** [50]	**Eureka** [66]	**Bin-Unpack** [20]
ACPacker	12/✓/×	635/✓/✓	-	-	-	-	-	-	-	-
ACProtect 2.0	811/✓/×	971/✓/✓	4/✓/×	-	-	-/✓	-	-/✓	-	-/✓
Armadillo 3.78	×/×	-	1/✓/×	✓/×	×/×	-	-	-/~	-/✓	-/✓
Armadillo 9.64	×/×	165/✓/✓	-	-	-	-	-	-	-	-
ASPack 2.12	3/✓/✓	3/✓/✓	2/✓/✓	✓/~	✓/✓	-/✓	-	-/✓	-/✓	-/✓
ASPack 2.40	3/✓/✓	3/✓/✓	-	-	-	-	-	-	-	-
EP Protector 0.3	2/✓/✓	2/✓/✓	1/✓/✓	-	-	-	-	-	-	-
eXPressor	2/✓/✓	2/✓/✓	1/✓/✓	-	-	-/✓	-	-	-	-/✓
FSG 2.0	2/✓/✓	2/✓/✓	1/✓/✓	✓/✓	✓/✓	-/✓	×/×	-/✓	-/✓	-/✓
JDPack 2.0	2/✓/×	3/✓/✓	×/×	-	-	-	-	-	-	-
MEW	2/✓/✓	2/✓/✓	1/✓/✓	✓/✓	✓/✓	-/✓	-	-/✓	-/✓	-/✓
MoleBox	2/✓/×	3/✓/✓	2/✓/✓	×/×	✓/✓	-	✓/✓	-/✓	-/✓	-/✓
Morphine 2.7	2/✓/×	3/✓/✓	-	✓/✓	✓/~	-	-	-/✓	-/✓	-
Mystic	×/×	4/✓/✓	1/✓/✓	-	-	-	-	-	-	-
Neolite 2.0	×/×	2/✓/✓	1/✓/✓	-	-	-	✓/✓	-	-	-
nPack 1.1.300	2/✓/✓	2/✓/✓	1/✓/✓	-	-	-	-	-	-	-/✓
Obsidium 1364	6/✓/×	16/✓/✓	×/×	×/×	×/×	-/✓	-	-	-/✓	-/✓
Packman 1.0	2/✓/✓	2/✓/✓	1/✓/✓	-	-	-	✓/✓	-	-	-
PE Compact 2.20	4/✓/✓	4/✓/✓	1/✓/✓	×/×	✓/✓	-/✓	✓/✓	-	-/✓	-/✓
PE Compact 3.02.2	4/✓/✓	4/✓/✓	-	-	-	-	-	-	-	-
PELock	2/✓/×	15/✓/×	6/✓/✓	-	-	-/✓	-	-	-	-/✓
PESpin 1.1	5/✓/×	80/✓/×	×/×	-	-	-	✓/×	-	-	-/✓
Petite 2.2	2/✓/×	3/✓/✓	×/×	-	-	-	✓/✓	-	-	-/✓
RLPack	2/✓/✓	2 /✓/✓	1/✓/✓	-	-	-	-	-	-	-/✓
Setisoft 2.7.1	×/×	32/✓/×	4/✓/×	-	-	-	-	-	-	-
SVK 1.43f	2/✓/×	35/✓/×	×/×	-	-	-	-	-	-	-
tElock 0.51	5/✓/×	17/✓/✓	5/✓/×	-	-	-	✓/×	-/×	-	-/✓
tElock 0.99	14/✓/×	18 /✓/✓	-	-	-	-	-	-	-	-
Themida 1.8	3/✓/×	-	×/×	×/×	✓/~	-	✓/×	-/✓	-/~	-/✓
Themida 2.0.3	3/✓/×	106 /✓/✓	-	-	-	-	-	-	-	-
Upack 0.39	2/✓/×	3/✓/✓	2/✓/✓	-	-	-	-	-	-	-
UPX 2.90	2/✓/✓	2/✓/✓	1/✓/✓	✓/✓	✓/✓	-/✓	✓/✓	-/✓	-/✓	-
WinUpack	3/✓/✓	3/✓/✓	2/✓/✓	✓/~	✓/✓	-/✓	×/×	-/✓	-/✓	-/✓
Yoda's Crypter 1.3	4/✓/✓	4/✓/✓	3/✓/×	-	-	-	-	-	-	-/✓
Yoda's Protector 1.02	2/✓/×	6/✓/✓	×/×	✓/~	✓/~	-	-	-	-/✓	-/✓

$n/d/r$: nombre de vagues détectées (n) / code auto-modifiant détecté (d) / récupération du code du binaire original (r)
✓ : réussite, ~ : réussite partielle, × : échec, - : absence de résultat
La version exacte des packers utilisés est seulement connue pour BOA, CoDisasm et PinSec.

- **original binary :** *hostname.exe*
- 35 pecked versions
- Comparison with 9 dynamic tools

Type d'analyse :	Statique	Dynamique								
Type de détection :		Instruction						Page mémoire	Appel système	Table d'import
Packer	**BOA**	**CoDisasm** [15]	**PinSec** [8]	**Poly-Unpack** [61]	**Renovo** [41]	**PinDe-monium** [49]	**RePE-construct** [45]	**Omni-Unpack** [50]	**Eureka** [66]	**Bin-Unpack** [20]
ACPacker	12/✓/×	635/✓/✓	-	-	-	-	-	-	-	-
ACProtect 2.0	811/✓/×	971/✓/✓	4/✓/×	-	-	-/✓	-	-/✓	-	-/✓
Armadillo 3.78	×/×	-	1/✓/×	✓/×	×/×	-	-	-/~	-/✓	-/✓
Armadillo 9.64	×/×	165/✓/✓	-	-	-	-	-	-	-	-
ASPack 2.12	3/✓/✓	3/✓/✓	2/✓/✓	✓/~	✓/✓	-/✓	-	-/✓	-/✓	-/✓
ASPack 2.40	3/✓/✓	3/✓/✓	-	-	-	-	-	-	-	-
EP Protector 0.3	2/✓/✓	2/✓/✓	1/✓/✓	-	-	-	-	-	-	-
eXPressor	2/✓/✓	2/✓/✓	1/✓/✓	-	-	-/✓	-	-	-	-/✓
FSG 2.0	2/✓/✓	2/✓/✓	1/✓/✓	✓/✓	✓/✓	-/✓	×/×	-/✓	-/✓	-/✓
JDPack 2.0	2/✓/×	3/✓/✓	×/×	-	-	-	-	-	-	-
MEW	2/✓/✓	2/✓/✓	1/✓/✓	✓/✓	✓/✓	-/✓	-	-/✓	-/✓	-/✓
MoleBox	2/✓/×	3/✓/✓	2/✓/✓	×/×	✓/✓	-	✓/✓	-/✓	-/✓	-/✓
Morphine 2.7	2/✓/×	3/✓/✓	-	✓/✓	✓/~	-	-	-/✓	-/✓	-
Mystic	×/×	4/✓/✓	1/✓/✓	-	-	-	-	-	-	-
Neolite 2.0	×/×	2/✓/✓	1/✓/✓	-	-	-	✓/✓	-	-	-
nPack 1.1.300	2/✓/✓	2/✓/✓	1/✓/✓	-	-	-	-	-	-	-/✓
Obsidium 1364	6/✓/×	16/✓/✓	×/×	×/×	×/×	-/✓	-	-	-/✓	-/✓
Packman 1.0	2/✓/✓	2/✓/✓	1/✓/✓	-	-	-	✓/✓	-	-	-
PE Compact 2.20	4/✓/✓	4/✓/✓	1/✓/✓	×/×	✓/✓	-/✓	✓/✓	-	-/✓	-/✓
PE Compact 3.02.2	4/✓/✓	4/✓/✓	-	-	-	-	-	-	-	-
PELock	2/✓/×	15/✓/×	6/✓/✓	-	-	-/✓	-	-	-	-/✓
PESpin 1.1	5/✓/×	80/✓/×	×/×	-	-	-	✓/×	-	-	-/✓
Petite 2.2	2/✓/×	3/✓/✓	×/×	-	-	-	✓/✓	-	-	-/✓
RLPack	2/✓/✓	2 /✓/✓	1/✓/✓	-	-	-	-	-	-	-/✓
Setisoft 2.7.1	×/×	32/✓/×	4/✓/×	-	-	-	-	-	-	-
SVK 1.43f	2/✓/×	35/✓/×	×/×	-	-	-	-	-	-	-
tElock 0.51	5/✓/×	17/✓/✓	5/✓/×	-	-	-	✓/×	-/×	-	-/✓
tElock 0.99	14/✓/×	18 /✓/✓	-	-	-	-	-	-	-	-
Themida 1.8	3/✓/×	-	×/×	×/×	✓/~	-	✓/×	-/✓	-/~	-/✓
Themida 2.0.3	3/✓/×	106 /✓/✓	-	-	-	-	-	-	-	-
Upack 0.39	2/✓/×	3/✓/✓	2/✓/✓	-	-	-	-	-	-	-
UPX 2.90	2/✓/✓	2/✓/✓	1/✓/✓	✓/✓	✓/✓	-/✓	✓/✓	-/✓	-/✓	-
WinUpack	3/✓/✓	3/✓/✓	2/✓/✓	✓/~	✓/✓	-/✓	×/×	-/✓	-/✓	-/✓
Yoda's Crypter 1.3	4/✓/✓	4/✓/✓	3/✓/×	-	-	-	-	-	-	-/✓
Yoda's Protector 1.02	2/✓/×	6/✓/✓	×/×	✓/~	✓/~	-	-	-	-/✓	-/✓

$n/d/r$: nombre de vagues détectées (n) / code auto-modifiant détecté (d) / récupération du code du binaire original (r)
✓ : réussite, ~ : réussite partielle, × : échec, - : absence de résultat
La version exacte des packers utilisés est seulement connue pour BOA, CoDisasm et PinSec.

- **original binary :** *hostname.exe*
- 35 pecked versions
- Comparison with 9 dynamic tools
- **Failed : 5**

Type d'analyse :	Statique	Dynamique								
Type de détection :		Instruction						Page mémoire	Appel système	Table d'import
Packer	**BOA**	**CoDisasm** [15]	**PinSec** [8]	**Poly-Unpack** [61]	**Renovo** [41]	**PinDemonium** [49]	**RePEconstruct** [45]	**Omni-Unpack** [50]	**Eureka** [66]	**Bin-Unpack** [20]
ACPacker	12/✓/×	635/✓/✓	-	-	-	-	-	-	-	-
ACProtect 2.0	811/✓/×	971/✓/✓	4/✓/×	-	-	-/✓	-	-/✓	-	-/✓
Armadillo 3.78	×/×	-	1/✓/×	✓/×	×/×	-	-	-/~	-/✓	-/✓
Armadillo 9.64	×/×	165/✓/✓	-	-	-	-	-	-	-	-
ASPack 2.12	3/✓/✓	3/✓/✓	2/✓/✓	✓/~	✓/✓	-/✓	-	-/✓	-/✓	-/✓
ASPack 2.40	3/✓/✓	3/✓/✓	-	-	-	-	-	-	-	-
EP Protector 0.3	2/✓/✓	2/✓/✓	1/✓/✓	-	-	-	-	-	-	-
eXPressor	2/✓/✓	2/✓/✓	1/✓/✓	-	-	-/✓	-	-	-	-/✓
FSG 2.0	2/✓/✓	2/✓/✓	1/✓/✓	✓/✓	✓/✓	-/✓	×/×	-/✓	-/✓	-/✓
JDPack 2.0	2/✓/×	3/✓/✓	×/×	-	-	-	-	-	-	-
MEW	2/✓/✓	2/✓/✓	1/✓/✓	✓/✓	✓/✓	-/✓	-	-/✓	-/✓	-/✓
MoleBox	2/✓/×	3/✓/✓	2/✓/✓	×/×	✓/✓	-	✓/✓	-/✓	-/✓	-/✓
Morphine 2.7	2/✓/×	3/✓/✓	-	✓/✓	✓/~	-	-	-/✓	-/✓	-
Mystic	×/×	4/✓/✓	1/✓/✓	-	-	-	-	-	-	-
Neolite 2.0	×/×	2/✓/✓	1/✓/✓	-	-	-	✓/✓	-	-	-
nPack 1.1.300	2/✓/✓	2/✓/✓	1/✓/✓	-	-	-	-	-	-	-/✓
Obsidium 1364	6/✓/×	16/✓/✓	×/×	×/×	×/×	-/✓	-	-	-/✓	-/✓
Packman 1.0	2/✓/✓	2/✓/✓	1/✓/✓	-	-	-	✓/✓	-	-	-
PE Compact 2.20	4/✓/✓	4/✓/✓	1/✓/✓	×/×	✓/✓	-/✓	✓/✓	-	-/✓	-/✓
PE Compact 3.02.2	4/✓/✓	4/✓/✓	-	-	-	-	-	-	-	-
PELock	2/✓/×	15/✓/×	6/✓/✓	-	-	-/✓	-	-	-	-/✓
PESpin 1.1	5/✓/×	80/✓/×	×/×	-	-	-	✓/×	-	-	-/✓
Petite 2.2	2/✓/×	3/✓/✓	×/×	-	-	-	✓/✓	-	-	-/✓
RLPack	2/✓/✓	2 /✓/✓	1/✓/✓	-	-	-	-	-	-	-/✓
Setisoft 2.7.1	×/×	32/✓/×	4/✓/×	-	-	-	-	-	-	-
SVK 1.43f	2/✓/×	35/✓/×	×/×	-	-	-	-	-	-	-
tElock 0.51	5/✓/×	17/✓/✓	5/✓/×	-	-	-	✓/×	-/×	-	-/✓
tElock 0.99	14/✓/×	18 /✓/✓	-	-	-	-	-	-	-	-
Themida 1.8	3/✓/×	-	×/×	×/×	✓/~	-	✓/×	-/✓	-/~	-/✓
Themida 2.0.3	3/✓/×	106 /✓/✓	-	-	-	-	-	-	-	-
Upack 0.39	2/✓/×	3/✓/✓	2/✓/✓	-	-	-	-	-	-	-
UPX 2.90	2/✓/✓	2/✓/✓	1/✓/✓	✓/✓	✓/✓	-/✓	✓/✓	-/✓	-/✓	-
WinUpack	3/✓/✓	3/✓/✓	2/✓/✓	✓/~	✓/✓	-/✓	×/×	-/✓	-/✓	-/✓
Yoda's Crypter 1.3	4/✓/✓	4/✓/✓	3/✓/×	-	-	-	-	-	-	-/✓
Yoda's Protector 1.02	2/✓/×	6/✓/✓	×/×	✓/~	✓/~	-	-	-	-/✓	-/✓

$n/d/r$: nombre de vagues détectées (n) / code auto-modifiant détecté (d) / récupération du code du binaire original (r)

✓ : réussite, ~ : réussite partielle, × : échec, - : absence de résultat

La version exacte des packers utilisés est seulement connue pour BOA, CoDisasm et PinSec.

- **original binary :** *hostname.exe*
- 35 pecked versions
- Comparison with 9 dynamic tools
- **Failed : 5**
- **Partially unpacked : 16**

Type d'analyse :	Statique	Dynamique								
Type de détection :		Instruction						Page mémoire	Appel système	Table d'import
Packer	**BOA**	**CoDisasm** [15]	**PinSec** [8]	**Poly-Unpack** [61]	**Renovo** [41]	**PinDemonium** [49]	**RePEconstruct** [45]	**Omni-Unpack** [50]	**Eureka** [66]	**Bin-Unpack** [20]
ACPacker	12/✓/✓	635/✓/✓	-	-	-	-	-	-	-	-
ACProtect 2.0	811/✓/×	971/✓/✓	4/✓/×	-	-	-/✓	-	-/✓	-	-/✓
Armadillo 3.78	×/×	-	1/✓/×	✓/×	×/×	-	-	-/~	-/✓	-/✓
Armadillo 9.64	×/×	165/✓/✓	-	-	-	-	-	-	-	-
ASPack 2.12	3/✓/✓	3/✓/✓	2/✓/✓	✓/~	✓/✓	-/✓	-	-/✓	-/✓	-/✓
ASPack 2.40	3/✓/✓	3/✓/✓	-	-	-	-	-	-	-	-
EP Protector 0.3	2/✓/✓	2/✓/✓	1/✓/✓	-	-	-	-	-	-	-
eXPressor	2/✓/✓	2/✓/✓	1/✓/✓	-	-	-/✓	-	-	-	-/✓
FSG 2.0	2/✓/✓	2/✓/✓	1/✓/✓	✓/✓	✓/✓	-/✓	×/×	-/✓	-/✓	-/✓
JDPack 2.0	2/✓/×	3/✓/✓	×/×	-	-	-	-	-	-	-
MEW	2/✓/✓	2/✓/✓	1/✓/✓	✓/✓	✓/✓	-/✓	-	-/✓	-/✓	-/✓
MoleBox	2/✓/×	3/✓/✓	2/✓/✓	×/×	✓/✓	-	✓/✓	-/✓	-/✓	-/✓
Morphine 2.7	2/✓/×	3/✓/✓	-	✓/✓	✓/~	-	-	-/✓	-/✓	-
Mystic	×/×	4/✓/✓	1/✓/✓	-	-	-	-	-	-	-
Neolite 2.0	×/×	2/✓/✓	1/✓/✓	-	-	-	✓/✓	-	-	-
nPack 1.1.300	2/✓/✓	2/✓/✓	1/✓/✓	-	-	-	-	-	-	-/✓
Obsidium 1364	6/✓/×	16/✓/✓	×/×	×/×	×/×	-/✓	-	-	-/✓	-/✓
Packman 1.0	2/✓/✓	2/✓/✓	1/✓/✓	-	-	-	✓/✓	-	-	-
PE Compact 2.20	4/✓/✓	4/✓/✓	1/✓/✓	×/×	✓/✓	-/✓	✓/✓	-	-/✓	-/✓
PE Compact 3.02.2	4/✓/✓	4/✓/✓	-	-	-	-	-	-	-	-
PELock	2/✓/×	15/✓/×	6/✓/✓	-	-	-/✓	-	-	-	-/✓
PESpin 1.1	5/✓/×	80/✓/×	×/×	-	-	-	✓/×	-	-	-/✓
Petite 2.2	2/✓/×	3/✓/✓	×/×	-	-	-	✓/✓	-	-	-/✓
RLPack	2/✓/✓	2 /✓/✓	1/✓/✓	-	-	-	-	-	-	-/✓
Setisoft 2.7.1	×/×	32/✓/×	4/✓/×	-	-	-	-	-	-	-
SVK 1.43f	2/✓/×	35/✓/×	×/×	-	-	-	-	-	-	-
tElock 0.51	5/✓/×	17/✓/✓	5/✓/×	-	-	-	✓/×	-/×	-	-/✓
tElock 0.99	14/✓/×	18 /✓/✓	-	-	-	-	-	-	-	-
Themida 1.8	3/✓/×	-	×/×	×/×	✓/~	-	✓/×	-/✓	-/~	-/✓
Themida 2.0.3	3/✓/×	106 /✓/✓	-	-	-	-	-	-	-	-
Upack 0.39	2/✓/×	3/✓/✓	2/✓/✓	-	-	-	-	-	-	-
UPX 2.90	2/✓/✓	2/✓/✓	1/✓/✓	✓/✓	✓/✓	-/✓	✓/✓	-/✓	-/✓	-
WinUpack	3/✓/✓	3/✓/✓	2/✓/✓	✓/~	✓/✓	-/✓	×/×	-/✓	-/✓	-/✓
Yoda's Crypter 1.3	4/✓/✓	4/✓/✓	3/✓/×	-	-	-	-	-	-	-/✓
Yoda's Protector 1.02	2/✓/×	6/✓/✓	×/×	✓/~	✓/~	-	-	-	-/✓	-/✓

$n/d/r$: nombre de vagues détectées (n) / code auto-modifiant détecté (d) / récupération du code du binaire original (r)
✓ : réussite, ~ : réussite partielle, × : échec, - : absence de résultat
La version exacte des packers utilisés est seulement connue pour BOA, CoDisasm et PinSec.

- **original binary :** *hostname.exe*
- 35 pecked versions
- Comparison with 9 dynamic tools
- **Failed : 5**
- **Partially unpacked : 16**
- **Totally unpacked : 14**

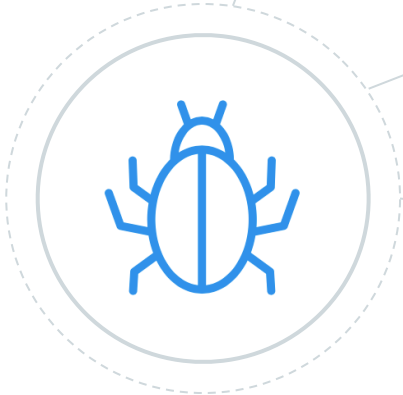
Type d'analyse :	Statique	Dynamique								
Type de détection :		Instruction						Page mémoire	Appel système	Table d'import
Packer	**BOA**	**CoDisasm** [15]	**PinSec** [8]	**Poly-Unpack** [61]	**Renovo** [41]	**PinDemonium** [49]	**RePEconstruct** [45]	**Omni-Unpack** [50]	**Eureka** [66]	**Bin-Unpack** [20]
ACPacker	12/✓/×	635/✓/✓	-	-	-	-	-	-	-	-
ACProtect 2.0	811/✓/×	971/✓/✓	4/✓/×	-	-	-/✓	-	-/✓	-	-/✓
Armadillo 3.78	×/×	-	1/✓/×	✓/×	×/×	-	-	-/~	-/✓	-/✓
Armadillo 9.64	×/×	165/✓/✓	-	-	-	-	-	-	-	-
ASPack 2.12	3/✓/✓	3/✓/✓	2/✓/✓	✓/~	✓/✓	-/✓	-	-/✓	-/✓	-/✓
ASPack 2.40	3/✓/✓	3/✓/✓	-	-	-	-	-	-	-	-
EP Protector 0.3	2/✓/✓	2/✓/✓	1/✓/✓	-	-	-	-	-	-	-
eXPressor	2/✓/✓	2/✓/✓	1/✓/✓	-	-	-/✓	-	-	-	-/✓
FSG 2.0	2/✓/✓	2/✓/✓	1/✓/✓	✓/✓	✓/✓	-/✓	×/×	-/✓	-/✓	-/✓
JDPack 2.0	2/✓/×	3/✓/✓	×/×	-	-	-	-	-	-	-
MEW	2/✓/✓	2/✓/✓	1/✓/✓	✓/✓	✓/✓	-/✓	-	-/✓	-/✓	-/✓
MoleBox	2/✓/×	3/✓/✓	2/✓/✓	×/×	✓/✓	-	✓/✓	-/✓	-/✓	-/✓
Morphine 2.7	2/✓/×	3/✓/✓	-	✓/✓	✓/~	-	-	-/✓	-/✓	-
Mystic	×/×	4/✓/✓	1/✓/✓	-	-	-	-	-	-	-
Neolite 2.0	×/×	2/✓/✓	1/✓/✓	-	-	-	✓/✓	-	-	-
nPack 1.1.300	2/✓/✓	2/✓/✓	1/✓/✓	-	-	-	-	-	-	-/✓
Obsidium 1364	6/✓/×	16/✓/✓	×/×	×/×	×/×	-/✓	-	-	-/✓	-/✓
Packman 1.0	2/✓/✓	2/✓/✓	1/✓/✓	-	-	-	✓/✓	-	-	-
PE Compact 2.20	4/✓/✓	4/✓/✓	1/✓/✓	×/×	✓/✓	-/✓	✓/✓	-	-/✓	-/✓
PE Compact 3.02.2	4/✓/✓	4/✓/✓	-	-	-	-	-	-	-	-
PELock	2/✓/×	15/✓/×	6/✓/✓	-	-	-/✓	-	-	-	-/✓
PESpin 1.1	5/✓/×	80/✓/×	×/×	-	-	-	✓/×	-	-	-/✓
Petite 2.2	2/✓/×	3/✓/✓	×/×	-	-	-	✓/✓	-	-	-/✓
RLPack	2/✓/✓	2 /✓/✓	1/✓/✓	-	-	-	-	-	-	-/✓
Setisoft 2.7.1	×/×	32/✓/×	4/✓/×	-	-	-	-	-	-	-
SVK 1.43f	2/✓/×	35/✓/×	×/×	-	-	-	-	-	-	-
tElock 0.51	5/✓/×	17/✓/✓	5/✓/×	-	-	-	✓/×	-/×	-	-/✓
tElock 0.99	14/✓/×	18 /✓/✓	-	-	-	-	-	-	-	-
Themida 1.8	3/✓/×	-	×/×	×/×	✓/~	-	✓/×	-/✓	-/~	-/✓
Themida 2.0.3	3/✓/×	106 /✓/✓	-	-	-	-	-	-	-	-
Upack 0.39	2/✓/×	3/✓/✓	2/✓/✓	-	-	-	-	-	-	-
UPX 2.90	2/✓/✓	2/✓/✓	1/✓/✓	✓/✓	✓/✓	-/✓	✓/✓	-/✓	-/✓	-
WinUpack	3/✓/✓	3/✓/✓	2/✓/✓	✓/~	✓/✓	-/✓	×/×	-/✓	-/✓	-/✓
Yoda's Crypter 1.3	4/✓/✓	4/✓/✓	3/✓/×	-	-	-	-	-	-	-/✓
Yoda's Protector 1.02	2/✓/×	6/✓/✓	×/×	✓/~	✓/~	-	-	-	-/✓	-/✓

$n/d/r$: nombre de vagues détectées (n) / code auto-modifiant détecté (d) / récupération du code du binaire original (r)
✓ : réussite, ~ : réussite partielle, × : échec, - : absence de résultat
La version exacte des packers utilisés est seulement connue pour BOA, CoDisasm et PinSec.

- **original binary :** *hostname.exe*
- 35 pecked versions
- Comparison with 9 dynamic tools
- **Failed : 5**
- **Partially unpacked : 16**
- **Totally unpacked : 14**

BOA :
100% static

Analysis of a real malware: Emotet



- **Features**
  - Steal password and banking information
  - Network propagation
  - Loader and botnet
- **Analysis of a « recent » sample**
  - October 14, 2020: detected as malware by 7/63 of VirusTotal's antivirus

Emotet: BOA analysis

- **First wave :**
  - 3 341 instructions
  - No exception
  - 145 `RET` instructions without call stack tampering
  - 10 external functions address retrieved by *GetProcAddress*
- **Construction of the second wave** (after ~28 000 executed instructions) **:**
  - Allocation of 117 043 bytes with *VirtualAllocExNuma*
  - Un-pack loop execution (117 042 iterations)
  - Jump on self-modified code with `call ebx` ⇨ enter in second wave

Emotet: BOA analysis

- **Second wave dump:**
  - « Raw » dump
  - Detected as *Emotet* by 13/60 VirusTotal's antivirus

Ad-Aware	Generic.EmotetAC.F21A24A9	ALYac	Generic.EmotetAC.F21A24A9
Arcabit	Generic.EmotetAC.F21A24A9	Avast	Win32:IcedID-A [Trj]
AVG	Win32:IcedID-A [Trj]	BitDefender	Generic.EmotetAC.F21A24A9
DrWeb	Trojan.Emotet.1031	Emsisoft	Generic.EmotetAC.F21A24A9 (B)
eScan	Generic.EmotetAC.F21A24A9	FireEye	Generic.EmotetAC.F21A24A9
GData	Generic.EmotetAC.F21A24A9	Gridinsoft	Trojan.Emotet.B.sdlyf
MAX	Malware (ai Score=86)	AegisLab	Undetected

Conclusion

- **Yet another tool**
  - A static analysis tool…
  - …with some features of a dynamic analysis
- **Needs some improvements**
  - Improve SMT solvers part
  - Multi-thread, multi-process, best OS simulation,…

- Bergeron, Debbabi, Desharnais, Erhioui, Lavoie et Tawbi - 2001
  Static Detection of Malicious Code in Executable Programs
- Bonfante, Fernandez, Marion, Rouxel, Sabatier et Thierry - 2015
  CoDisasm: Medium Scale Concatic Disassembly of Self-Modifying Binaries with Overlapping Instructions
- Biondi, Rigo, Zennou et Mehrenberger - 2017
  BinCAT: purrfecting binary static analysis
- Collberg, Thomborson et Low - 1997
  A Taxonomy of Obfuscating Transformations
- Djoudi et Bardin - 2015
  BINSEC: Binary Code Analysis with Low-Level Regions
- Junod, Rinaldini, Wehrli et Michielin - 2015
  Obfuscator-LLVM -- Software Protection for the Masses
- Kiss, Lalande, Leslous et Tong - 2016
  Kharon dataset: Android malware under a microscope
- Pietrek - 1997
  A crash course on the depths of Win32 structured exception handling
- Qiu, Su et Ma - 2014
  Deobfuscate Non-Returning Calls and Call-Stack Tampering in Instruction Traces
- Robin - 2017
  Formal Approaches for Automatic Deobfuscation and Reverse-engineering of Protected Codes
- Salwan - 2020
  L'usage de l'exécution symbolique pour la déobfuscation binaire en milieu industriel
- Schwarz, Debray et Andrews - 2002
  Disassembly of executable code revisited
- Shoshitaishvili, Wang, Salls, Stephens, Polino, Dutcher, Grosen, Feng, Hauser, Kruegel, Vigna - 2016
  SOK: (State of) The Art of War: Offensive Techniques in Binary Analysis