

@Maxou56800, @rbnctl

CONIX CyberSecurity

—

8 September 2017

Machoke

Who are we?

Maxou56800

Twitter: ▶ @Maxou56800

CERT-Conix

Robin Marsollier

Twitter: ▶ @rbnctl

robin.marsollier@conix.fr

CERT-Conix

- 1 Apparently used by AV
- 2 Used by academics
- 3 Used by other actors

- 1 Apparently used by AV
- 2 Used by academics
- 3 Used by other actors
- 4 But yet no public implementation ...

- 1 Get something better than md5/sha* (resistant to small changes inside samples notably, etc.)
- 2 A fuzzy hash better than good old ssdeep
- 3 Get a small and independent tool easy to use and deploy at large
- 4 Let other tools do the clustering

- 1 Designed by ANSSI, published with Polichombr (<https://github.com/ANSSI-FR/polichombr>)
- 2 CFG-based fuzzy hash
- 3 2 implementations: Ruby/miasm || Python/IDAPython (Machoc lost in lots of ruby/python/whatever code)

Machoke

Naming



- 1 Radare2 + r2pipe
- 2 Python

Machoke

Machoke algorithm

```
[0x660] ;[gb]
;-- main:
(fcn) main 54
main ();
; var int local_14h @ rbp-0x14
; var int local_4h @ rbp-0x4
; CALL XREF from 0x000006a6 (sym.function1)
; DATA XREF from 0x0000054d (entry0)
push rbp
mov rbp, rsp
sub rsp, 0x20
mov dword [local_14h], edi
mov dword [local_4h], 0
mov dword [local_4h], 0
jmp 0x689;[ga]
```

v

```
0x689 ;[ga]
; JMP XREF from 0x00000679 (main)
; [0x4:4]=0x10102
cmp dword [local_4h], 4
jle 0x67b;[gd]
```

t f

```
0x67b ;[gd]
; JMP XREF from 0x0000068d (main)
mov eax, dword [local_4h]
mov edi, eax
call sym.function1;[gc]
add dword [local_4h], 1
```

```
0x68f ;[ge]
mov eax, 0
leave
ret
```

Machoke

Machoke algorithm

1 Blocks and call labelling

```
[0x660] ;[gb]
;-- main:
(fcn) main 54
main ();
; var int local_14h @ rbp-0x14
; var int local_4h @ rbp-0x4
; CALL XREF from 0x000006a6 (sym.function1)
; DATA XREF from 0x0000054d (entry0)
push rbp
mov rbp, rsp
sub rsp, 0x20
mov dword [local_14h], edi
mov dword [local_4h], 0
mov dword [local_4h], 0
jmp 0x689;[ga]
```

1

v

```
0x689 ;[ga]
; JMP XREF from 0x00000679 (main)
; [0x4:4]=0x10102
cmp dword [local_4h], 4
jle 0x67b;[gd]
```

2

t f

```
0x67b ;[gd]
; JMP XREF from 0x0000068d (main)
mov eax, dword [local_4h]
mov edi, eax
call sym.function1;[gc]
add dword [local_4h], 1
```

3

```
0x68f ;[ge]
mov eax, 0
leave
ret
```

4

Machoke

Machoke algorithm

```
[0x660] ;[gb]
;-- main:
(fcn) main 54
main ();
; var int local_14h @ rbp-0x14
; var int local_4h @ rbp-0x4
; CALL XREF from 0x000006a6 (sym.function1)
; DATA XREF from 0x0000054d (entry0)
push rbp
mov rbp, rsp
sub rsp, 0x20
mov dword [local_14h], edi
mov dword [local_4h], 0
mov dword [local_4h], 0
jmp 0x689;[ga]
```

1

v

```
0x689 ;[ga]
; JMP XREF from 0x00000679 (main)
; [0x4:4]=0x10102
cmp dword [local_4h], 4
jle 0x67b;[gd]
```

2

t f

```
0x67b ;[gd]
; JMP XREF from 0x0000068d (main)
mov eax, dword [local_4h]
mov edi, eax
call sym.function1;[gc]
add dword [local_4h], 1
```

3

```
0x68f ;[ge]
mov eax, 0
leave
ret
```

4

- 1 Blocks and call labelling
- 2 Translate to text:
1:2;

Machoke

Machoke algorithm

```
[0x660] ;[gb]
;-- main:
(fcn) main 54
main ();
; var int local_14h @ rbp-0x14
; var int local_4h @ rbp-0x4
; CALL XREF from 0x000006a6 (sym.function1)
; DATA XREF from 0x0000054d (entry0)
push rbp
mov rbp, rsp
sub rsp, 0x20
mov dword [local_14h], edi
mov dword [local_4h], 0
mov dword [local_4h], 0
jmp 0x689;[ga]
```

1

v

```
0x689 ;[ga]
; JMP XREF from 0x00000679 (main)
; [0x4:4]=0x10102
cmp dword [local_4h], 4
jle 0x67b;[gd]
```

2

t f

```
0x67b ;[gd]
; JMP XREF from 0x0000068d (main)
mov eax, dword [local_4h]
mov edi, eax
call sym.function1;[gc]
add dword [local_4h], 1
```

3

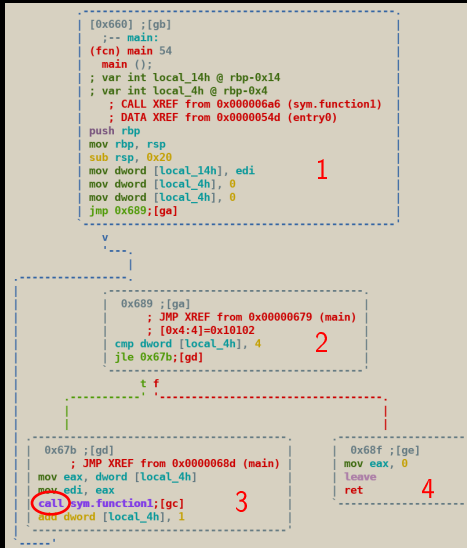
```
0x68f ;[ge]
mov eax, 0
leave
ret
```

4

- 1 Blocks and call labelling
- 2 Translate to text:
1:2;2:3,4;

Machoke

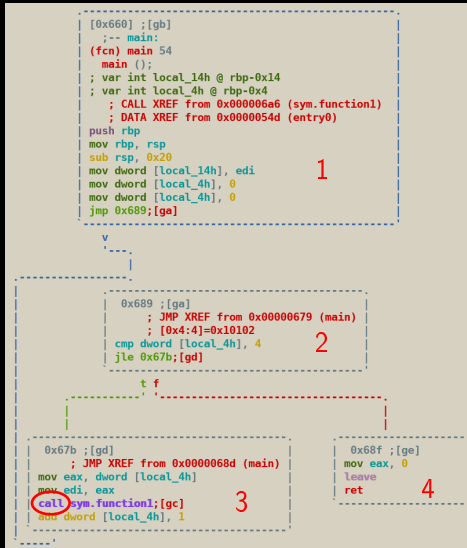
Machoke algorithm



- 1 Blocks and call labelling
- 2 Translate to text:
1:2;2:3,4;3:c,2;

Machoke

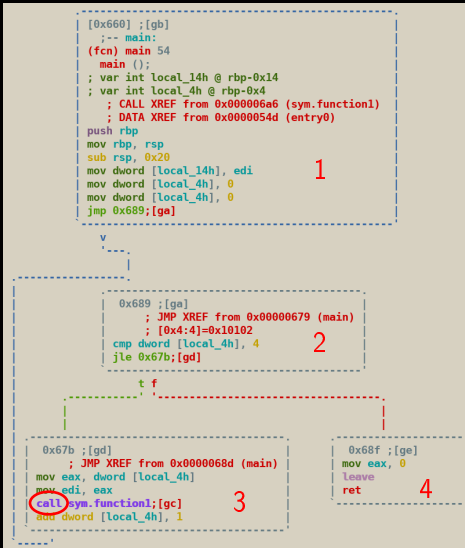
Machoke algorithm



- 1 Blocks and call labelling
- 2 Translate to text:
1:2;2:3,4;3:c,2;4:;

Machoke

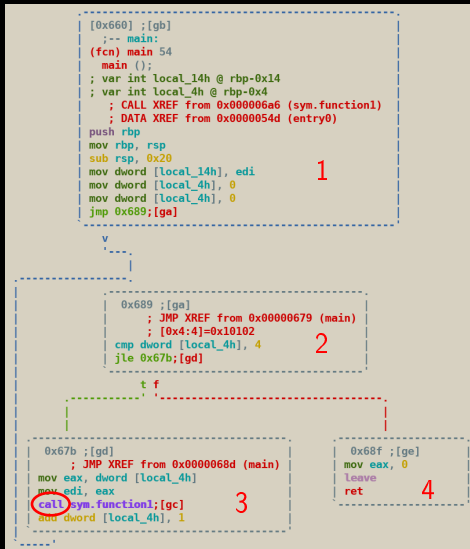
Machoke algorithm



- 1 Blocks and call labelling
- 2 Translate to text:
1:2;2:3,4;3:c,2;4;;
- 3 Murmurhash3: e38a5cbb

Machoke

Machoke algorithm



- 1 Blocks and call labelling
- 2 Translate to text:
1:2;2:3,4;3:c,2;4;
- 3 Murmurhash3: e38a5cbb
- 4 Repeat for each function in
sample, concatenate hashes

Machoke

r2 commands used

- 1 aa
- 2 ilj
- 3 aflj
- 4 agj

First analysis on a collection of samples classified by Yara rules (21915 samples):

- 1 21915 unique MD5/SHA256 (as expected)

First analysis on a collection of samples classified by Yara rules (21915 samples):

- 1 21915 unique MD5/SHA256 (as expected)
- 2 10691 unique ssdeep

First analysis on a collection of samples classified by Yara rules (21915 samples):

- 1 21915 unique MD5/SHA256 (as expected)
- 2 10691 unique ssdeep
- 3 Only 4674 unique machoke hashes

Second analysis on the BUSURPER tool family from the shadow broker leak:

Second analysis on the BUSURPER tool family from the shadow broker leak:

- 1 80 different samples (80 different md5/sha*)
- 2 Only 3 different machoke hashes for all the 80 versions of the tool

Machoke

Machoke in the future...

- 1 Adding ability to "machoke" functions of malware
- 2 Integrate machoke inside other tools (MISP, viper etc.)
- 3 Build correlation/clusterisation tools on top of it

<https://github.com/conix-security/machoke>