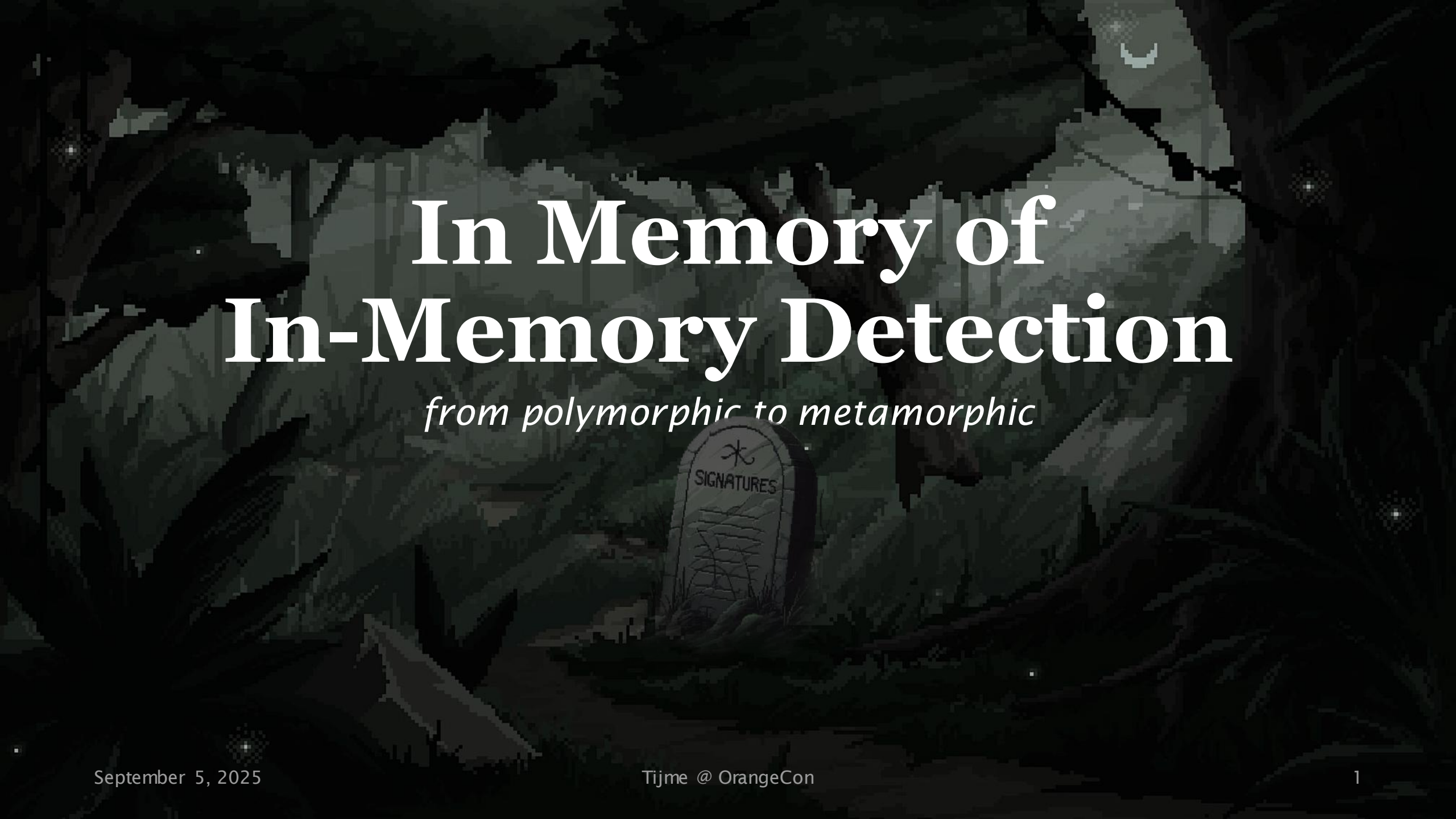




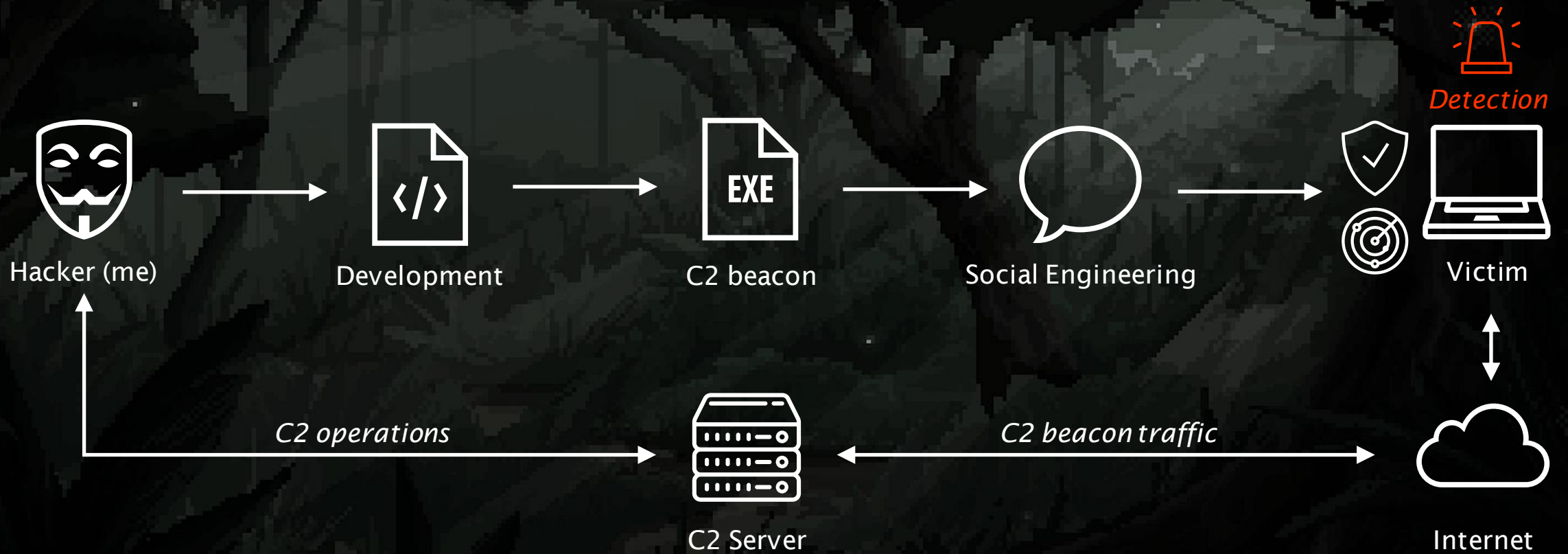
In Memory of In-Memory Detection

from polymorphic to metamorphic

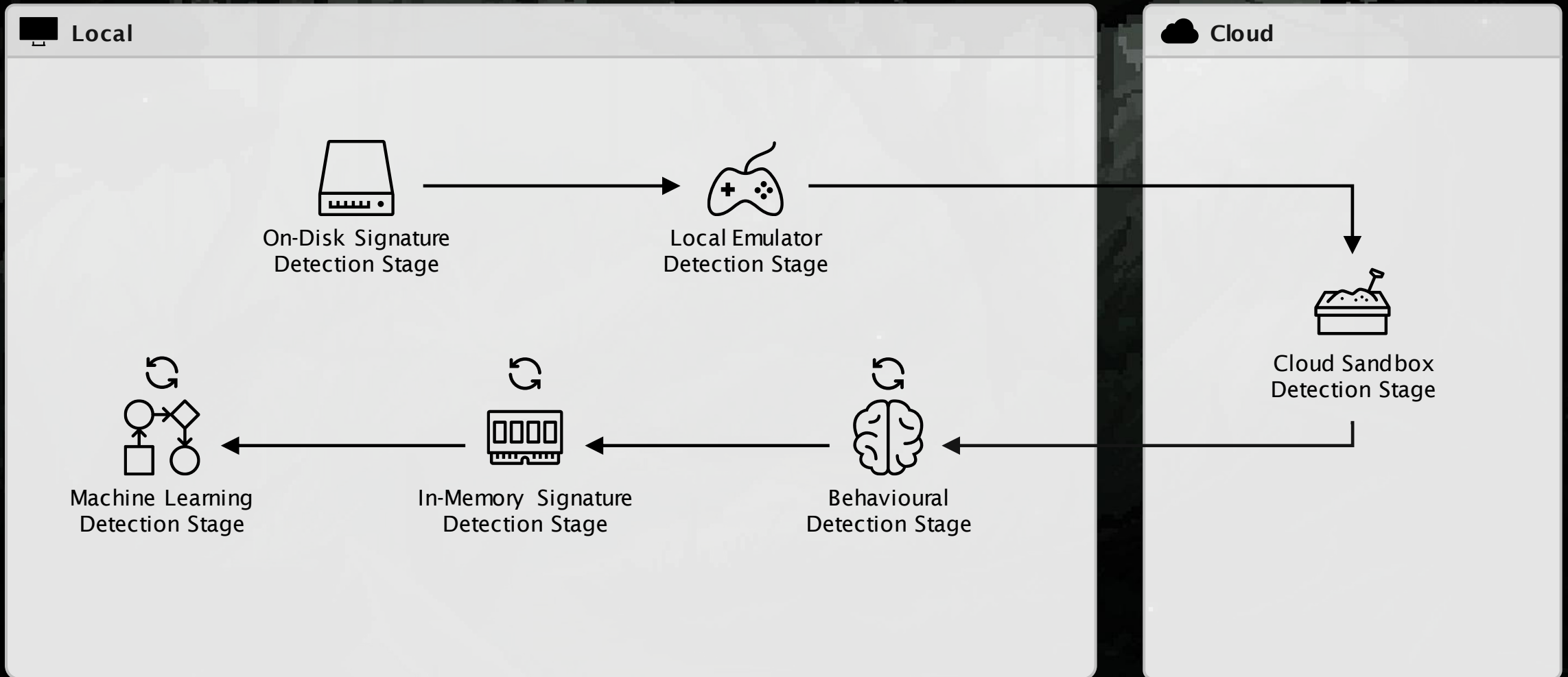
About Tijme (me)

- Offensive Cyber @ ABN AMRO Bank (Netherlands)
- Previously Red Teaming @ Northwave (Netherlands)
- Author of exploits & malwarez
- Socials username is @tijme
[Bluesky](#) - [GitHub](#) - [LinkedIn](#) - [Twitter](#)

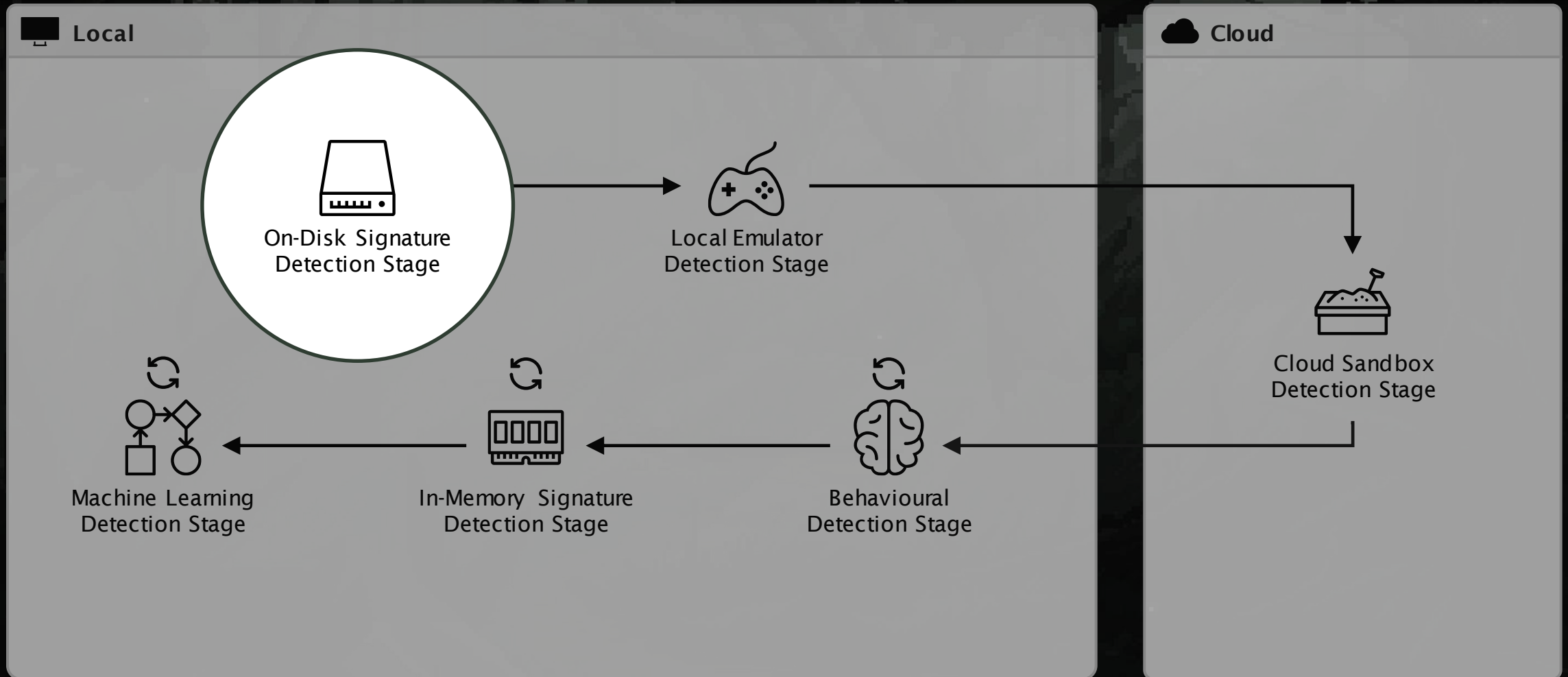
Malware



Detection stages



Detection stages



Talk outline

1. **Polymorphic malware**
Often found in malware antique stores
 2. **Metamorphic malware**
Often seen on malware buzzword bingo cards
 3. **Dittobytes project**
A true metamorphic cross-compiler
- ❖ **Demo**

Packer

telepath9000 / elf-packer Public

Code Issues 1 Pull requests Actions Projects Security Insights

master 1 Branch 0 Tags

Go to file

Code

telepath9000 removed test, main bin can be used c20efdf · 4 months ago 44 Commits

include	renamed fill... to inject_payload	4 months ago
src	changed encryption iteration to 64 bit	4 months ago
.gitignore	added compiledb json to gitignore	6 years ago
LICENSE	complete rewrite	7 years ago
Makefile	fixed segfault in prepare_elf	4 months ago
README.md	Update README.md	6 years ago

README MIT license

elf packer

About

Encrypts 64-bit elf files that decrypt at runtime.

linux encryption binary-analysis elf-parser self-modifying-code injection-attacks elf64 elf-binaries elf-format

Readme MIT license Activity 32 stars 2 watching 6 forks Report repository

Packer

Your original code

```
#include <stdio.h>
#include <stdlib.h>

int main (int argc, char** argv) {
    printf("Hello world");

    WinExec("whoami");

    return 0;
}
```

packed.exe

Packer

Your original code

```
#include <stdio.h>
#include <stdlib.h>

int main (int argc, char** argv) {
    printf("Hello world");

    WinExec("whoami");

    return 0;
}
```

Encrypted

packed.exe

.text segment: Encrypted code

```
0x050 01001100100011100000110101011001010011011100111000011
0x060 00011011000100110010001100110001101010011000000110101
0x070 00110000001100000011011000110111011001010011001101100
0x080 10101100001011000010110011001100001001101000110001000
0x090 11001100110000001110010011011001100100001100100110000
0x100 10110011000110110001110010110010100111001001100000011
0x110 10010110000101100100001100100110001000110001001100110
0x120 01101100011000000110111011001000011100100110100001100
0x130 01001110000011000000110110011001100011011100110100001
0x140 10100011001100110011001100010001101100110000100110000
0x150 01100110011001100110010100110010011000100011010000110
0x160 01001100010001100000011010100110100001101110110010000
0x170 01101100011000000110111011001000011100100110100001100
0x180 1110000011100000111000011000110110001
```

Packer

New stub by packer

```
#include <stdio.h>
#include <stdlib.h>

int stub () { argc, char** argv) {
    decrypt(lpEncryptedCode, lpKey);
    return lpEncryptedCode();
}
WinExec("whoami");

return 0;
}
```

Encrypted
Added

packed.exe

```
.text segment: DecryptStub
0x000 00000000100010000000000000001100100001001000000000000011
0x010 0001001000100001100000011001000010000000000000000110000
0x020 001000000010000000100010001100100100000000010000001000
0x030 0000110000001000000011001000100000000010010001100000000
0x040 100010001100000010000000100010010001000000000110000
0x100 10110011000110110001110010110010100111001001100000011
0x110 10010110000101100100001100100110001000110001001100110
0x120 01101100011000000110111011001000011100100110100001100
0x130 01001110000011000000110110011001100011011100110100001
0x140 10100011001100110011001100010001101100110000100110000
0x150 01100110011001100110010100110010011000100011010000110
0x160 01001100010001100000011010100110100001101110110010000
0x170 01101100011000000110111011001000011100100110100001100
0x180 1110000011100000111000011000110110001
```

Packer

New stub by packer

```
#include <stdio.h>
#include <stdlib.h>

int stub () {
    decrypt(lpEncryptedCode, lpKey);
    return lpEncryptedCode();
}
```

.text segment: Encrypted code

```
0x050 01001100100011100000110101011001010011011100111000011
0x060 00011011000100110010001100110001101010011000000110101
```

Your original code

```
#include <stdio.h>
#include <stdlib.h>

int main (int argc, char** argv) {
    printf("Hello world");
    WinExec("whoami");
    return 0;
}
```

1. Stub is first to be executed

2. Decrypts the original code

3. Jumps to the original code

4. Original code is executed

Polymorphic stub

Sample 1

.text segment: Decryption stub

```
0x000 00110001100110011011001010011011100110110011000100011
0x010 00010110010000111001001101100011000001100100001110010
0x020 01100010011010100110001001110010110010100110100011000
0x030 01001110000110010100110110011000110011001100110010001
0x040 10011001110010011000100110001011000110010011000
```

Sample 2

.text segment: Decryption stub

```
0x000 00110001100110011011001010011011100110110011000100011
0x010 00010110010000111001001101100011000001100100001110010
0x020 01100010011010100110001001110010110010100110100011000
0x030 01001110000110010100110110011000110011001100110010001
0x040 10011001110010011000100110001011000110010011000
```

In both examples, the stub is static (signatureable) each compilation.

```
0x050 01001100100011100000110101011001010011011100111000011
0x060 00011011000100110010001100110001101010011000000110101
0x070 00110000001100000011011000110111011001010011001101100
0x080 10101100001011000010110011001100001001101000110001000
0x090 11001100110000001110010011011001100100001100100110000
0x100 10110011000110110001110010110010100111001001100000011
0x110 10010110000101100100001100100110001000110001001100110
0x120 01101100011000000110111011001000011100100110100001100
0x130 01001110000011000000110110011001100011011100110100001
0x140 10100011001100110011001100010001101100110000100110000
0x150 01100110011001100110010100110010011000100011010000110
0x160 01001100010001100000011010100110100001101110110010000
0x170 01101100011000000110111011001000011100100110100001100
0x180 1110000011100000111000011000110110001
```

```
0x050 01100011001110010001000101100000010100010000000100000
0x060 00100000001000000010000000100000001000000010000000100
0x070 00000100000001000000010000000100010011010110110010101
0x080 11100101011111011011110111000001110011001000100011101
0x090 00010000001011011000010100010000000100000001000000010
0x100 00000010000000100000001000000010000000100000001000000
0x110 01000000010000000100000001000000010000000100000001000
0x120 10011101100110010101110010011010010110011001111001001
0x130 00010000010100010000000100000001000000010000000100000
0x140 0010000001000000010000000100000001000000010000000100
0x150 00001011101001011000000101000100000001000000010000000
0x160 10000000100000001000000010000000100000001000000010000
0x170 000100000001000000010001001101011011011001011001000010
0x180 00100011101000100000001000100101000001110101011000100
```

Ultimate polymorphic form

The screenshot shows a GitHub repository page for 'PELock/Simple-Polymorphic-Engine-SPE32'. The repository description states: 'Simple Polymorphic Engine (SPE32) is a simple polymorphic engine for encrypting code and data. SPE32 allows you to encrypt any data and generate a unique decryption code for this data. The encryption algorithm uses randomly selected instructions and encryption keys. The generated decryption code will be different each time.' Below this, there is a section titled 'Polymorphic decryption code as viewed in x86dbg debugger' which displays a screenshot of the x86dbg debugger. The debugger window shows the decryption code for 'spe32_test.exe'. The code is a sequence of instructions that decrypts the data. The instructions are listed in a table with columns for address, disassembly, and comment. The address column shows addresses from 00401400 to 00401444. The disassembly column shows instructions like 'jmp', 'push', 'call', 'pushad', 'pushfd', 'mov', 'neg', 'add', 'mov', 'sub', 'xor', and 'mov'. The comment column shows comments like 'spe32_test.40141A', 'VirtualFree', and 'ExitProcess'. The debugger also shows the CPU registers and the current instruction pointer (EIP) at 0040141A.

GitHub - PELock/Simple-Polymorphic-Engine-SPE32

Simple Polymorphic Engine — SPE32

Simple Polymorphic Engine (SPE32) is a simple polymorphic engine for encrypting code and data.

SPE32 allows you to encrypt any data and generate a unique decryption code for this data.

The encryption algorithm uses randomly selected instructions and encryption keys.

The generated decryption code will be different each time.

Polymorphic decryption code as viewed in x86dbg debugger

spe32_test.exe - PID: 2018 - Module: spe32_test.exe - Thread: Main Thread 2788 - x32dbg [Elevated]

File View Debug Trace Plugins Favourites Options Help Sep 1 2019

Graph Log Notes Breakpoints Memory Map Call Stack CPU SEH Script Symbols Source References Threads Handles Trace

Address	Disassembly	Comment
00401400	EB 15	
00401405	58	
00401406	68 00800000	jmp spe32_test.40141A
00401408	6A 00	push eax
0040140D	50	push 0
0040140E	E8 5F100000	call <JMP.&VirtualFree>
00401413	6A 00	push 0
00401415	E8 46100000	call <JMP.&ExitProcess>
0040141A	68	pushad
0040141B	9C	pushfd
0040141C	BE E00BFF	mov esi, FF8FDE0
00401421	F7DE	neg esi
00401423	03F5	add esi, ebp
00401425	BF 10000000	mov edi, 10
0040142A	8B06	mov eax, duword ptr [esi]
0040142C	81E8 33B4C07	sub eax, 078CB433
00401432	81F0 8C2DEF75	xor eax, 75EF200C
00401438	81E8 A3CB7B9D	sub eax, 9D7BCB43
0040143E	81F0 00F25E67	xor eax, 675EF200
00401444	8906	mov dword ptr [esi], eax

Hide FPU

Register	Value	Comment
EAX	FF8FDE0	
EBX	00480000	
ECX	00000004	
EDX	00000003	
EBP	00000000	
ESP	0019FF80	
ESI	00402460	<spe32_test.JMP.&ExitProcess>
EDI	0040141A	spe32_test.0040141A
EIP	0040141A	spe32_test.0040141A

EFLAGS 00000293

Flag	Value
ZF	0
PF	0
AF	1
OF	0
SF	1
DF	0
CF	1
TF	0
IF	1

LastError: 000036B7 (ERROR_SXS_KEY_NOT_FOUND)

Readme

Activity

148 stars

6 watching

36 forks

Report repository

Releases

No releases published

Packages

No packages published

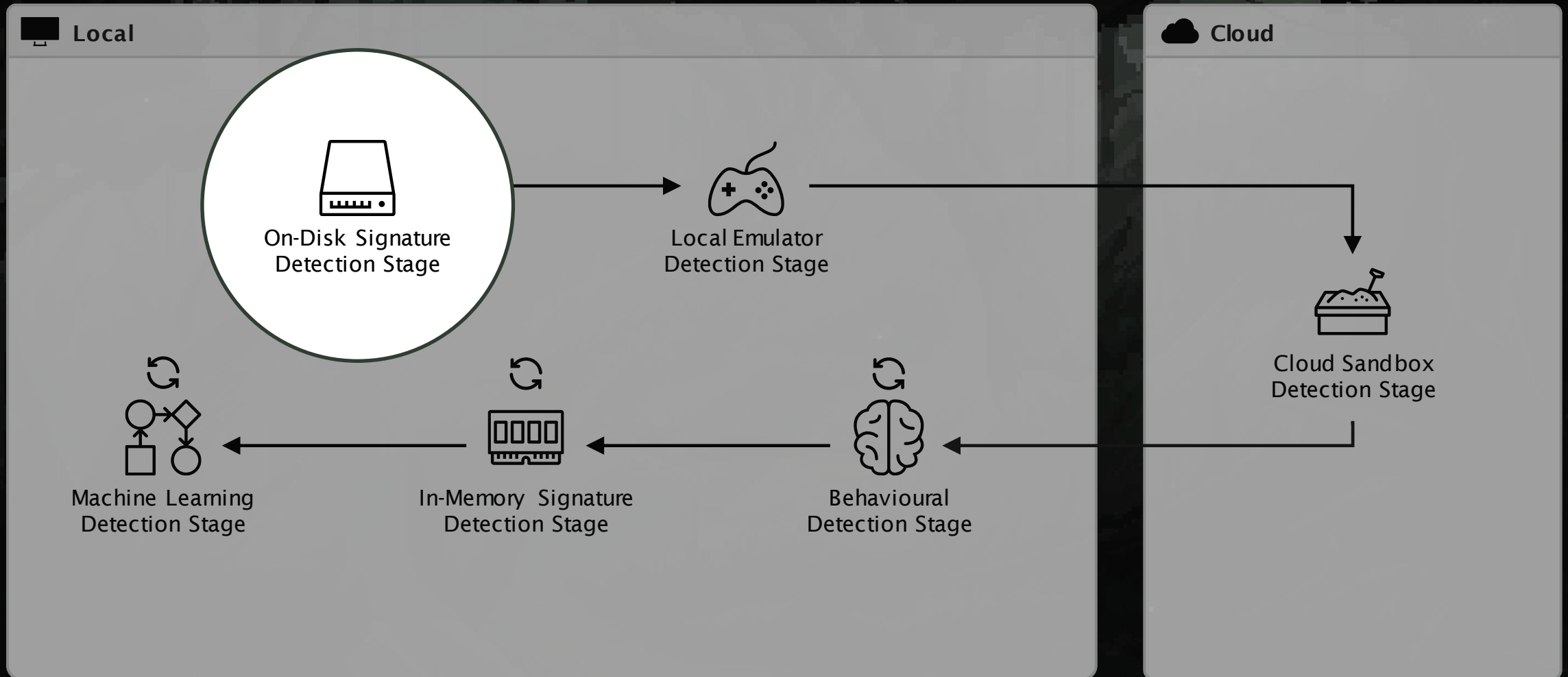
Languages

Assembly 99.3%

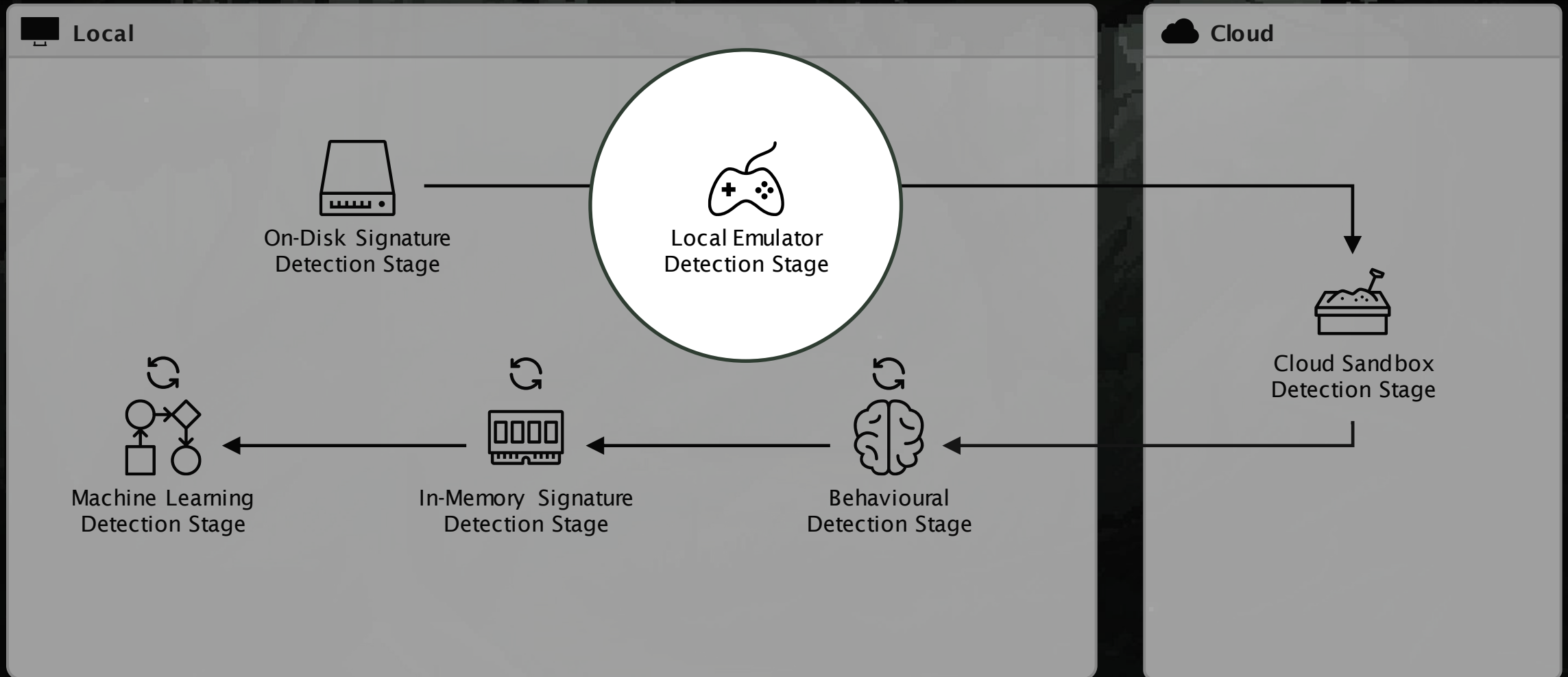
Batchfile 0.7%

Virus.exe	Malware.exe	InstallMe.exe	VeryLegit.exe	NotMalware	Undetected	ClickMe	Stealthy	Test	Win	Win		
.text segment: Decryption step 0x000 011001010011100000110 0x010 000101100100001110010 0x020 011000100110101001100 0x030 010011100001100101001 0x040 100110011100100110001	.text segment: Decryption step 0x000 0110010101101010011 0x010 10001110000001110011 0x020 0001111000111100111 0x030 0000011101100101010 0x040 1110011111010010010	.text segment: Decryption step 0x000 10101110101011110 0x010 1101100110101110 0x020 0000110010001111 0x030 0101110111010010 0x040 0100001111001010	.text segment: Decryption step 0x000 1010110011001100 0x010 01011110100 0x020 11100110110 0x030 11010100011 0x040 10000010101	.text segment: Decryption step 0x000 100110011001100 0x010 01001100100 0x020 1010100 0x030 010100 0x040 000010	.text segment: Decryption step 0x000 0110011001100 0x010 000100 0x020 000 0x030 10 0x040 10	.text segment: Decryption step 0x000 0000000000000000 0x010 0000000000000000 0x020 0000000000000000 0x030 0000000000000000 0x040 0000000000000000	.text segment: Decryption step 0x000 0000000000000000 0x010 0000000000000000 0x020 0000000000000000 0x030 0000000000000000 0x040 0000000000000000	.text segment: Decryption step 0x000 0000000000000000 0x010 0000000000000000 0x020 0000000000000000 0x030 0000000000000000 0x040 0000000000000000	.text segment: Decryption step 0x000 0000000000000000 0x010 0000000000000000 0x020 0000000000000000 0x030 0000000000000000 0x040 0000000000000000	.text segment: Decryption step 0x000 0000000000000000 0x010 0000000000000000 0x020 0000000000000000 0x030 0000000000000000 0x040 0000000000000000		
.text segment: Encrypted code 0x050 010011001000111000001 0x060 000110110001001100100 0x070 001100000011000000110 0x080 101011000010110000101 0x090 110011001100000011100 0x100 101100110001101100011 0x110 100101100001011001000 0x120 011011000110000001101 0x130 010011100000110000001 0x140 101000110011001100110 0x150 011001100110011001100 0x160 010011000100011000000 0x170 011011000110000001101 0x180 111000001110000011100	.text segment: Encrypted code 0x050 0000010011101100001 0x060 0000100101000001000 0x070 0010100011111110111 0x080 0001010001111111100 0x090 0000000110110100011 0x100 0011001010110110110 0x110 0000010000001110111 0x120 1001100001010110011 0x130 1000011000001001011 0x140 1101101100010101001 0x150 1110011110100111101 0x160 1101101010111111111 0x170 1001111001000111011 0x180 0100011011011010001	.text segment: Encrypted code 0x050 00010110010010110 0x060 1101100101101010 0x070 0110110001011110 0x080 1001011100111111 0x090 0011000110010010 0x100 0101110000000100 0x110 1001000001000101 0x120 0001110101001101 0x130 1110000101000011 0x140 1000100100111000 0x150 1110010101000001 0x160 0011101001001111 0x170 1000110100100000 0x180 0011110011010101	.text segment: Encrypted code 0x050 1110110000000000 0x060 10001101100 0x070 01111111110 0x080 10010001010 0x090 00011001001 0x100 11011100100 0x110 00101001000 0x120 11000111011 0x130 01000010101 0x140 11100011101 0x150 10101011111 0x160 00110110010 0x170 00101001101 0x180 11100001001	.text segment: Encrypted code 0x050 110110011001100 0x060 101100101 0x070 010100 0x080 01000 0x090 01010 0x100 101100100 0x110 000100 0x120 101100 0x130 001100 0x140 11100 0x150 1000100 0x160 000100 0x170 111100 0x180 1100000	.text segment: Encrypted code 0x050 0000000000000000 0x060 0100000000000000 0x070 0000000000000000 0x080 0000000000000000 0x090 0000000000000000 0x100 0000000000000000 0x110 0000000000000000 0x120 0000000000000000 0x130 0000000000000000 0x140 0000000000000000 0x150 0000000000000000 0x160 0000000000000000 0x170 0000000000000000 0x180 0000000000000000	.text segment: Encrypted code 0x050 0000000000000000 0x060 0000000000000000 0x070 0000000000000000 0x080 0000000000000000 0x090 0000000000000000 0x100 0000000000000000 0x110 0000000000000000 0x120 0000000000000000 0x130 0000000000000000 0x140 0000000000000000 0x150 0000000000000000 0x160 0000000000000000 0x170 0000000000000000 0x180 0000000000000000	.text segment: Encrypted code 0x050 0000000000000000 0x060 0000000000000000 0x070 0000000000000000 0x080 0000000000000000 0x090 0000000000000000 0x100 0000000000000000 0x110 0000000000000000 0x120 0000000000000000 0x130 0000000000000000 0x140 0000000000000000 0x150 0000000000000000 0x160 0000000000000000 0x170 0000000000000000 0x180 0000000000000000	.text segment: Encrypted code 0x050 0000000000000000 0x060 0000000000000000 0x070 0000000000000000 0x080 0000000000000000 0x090 0000000000000000 0x100 0000000000000000 0x110 0000000000000000 0x120 0000000000000000 0x130 0000000000000000 0x140 0000000000000000 0x150 0000000000000000 0x160 0000000000000000 0x170 0000000000000000 0x180 0000000000000000	.text segment: Encrypted code 0x050 0000000000000000 0x060 0000000000000000 0x070 0000000000000000 0x080 0000000000000000 0x090 0000000000000000 0x100 0000000000000000 0x110 0000000000000000 0x120 0000000000000000 0x130 0000000000000000 0x140 0000000000000000 0x150 0000000000000000 0x160 0000000000000000 0x170 0000000000000000 0x180 0000000000000000	.text segment: Encrypted code 0x050 0000000000000000 0x060 0000000000000000 0x070 0000000000000000 0x080 0000000000000000 0x090 0000000000000000 0x100 0000000000000000 0x110 0000000000000000 0x120 0000000000000000 0x130 0000000000000000 0x140 0000000000000000 0x150 0000000000000000 0x160 0000000000000000 0x170 0000000000000000 0x180 0000000000000000	.text segment: Encrypted code 0x050 0000000000000000 0x060 0000000000000000 0x070 0000000000000000 0x080 0000000000000000 0x090 0000000000000000 0x100 0000000000000000 0x110 0000000000000000 0x120 0000000000000000 0x130 0000000000000000 0x140 0000000000000000 0x150 0000000000000000 0x160 0000000000000000 0x170 0000000000000000 0x180 0000000000000000	.text segment: Encrypted code 0x050 0000000000000000 0x060 0000000000000000 0x070 0000000000000000 0x080 0000000000000000 0x090 0000000000000000 0x100 0000000000000000 0x110 0000000000000000 0x120 0000000000000000 0x130 0000000000000000 0x140 0000000000000000 0x150 0000000000000000 0x160 0000000000000000 0x170 0000000000000000 0x180 0000000000000000

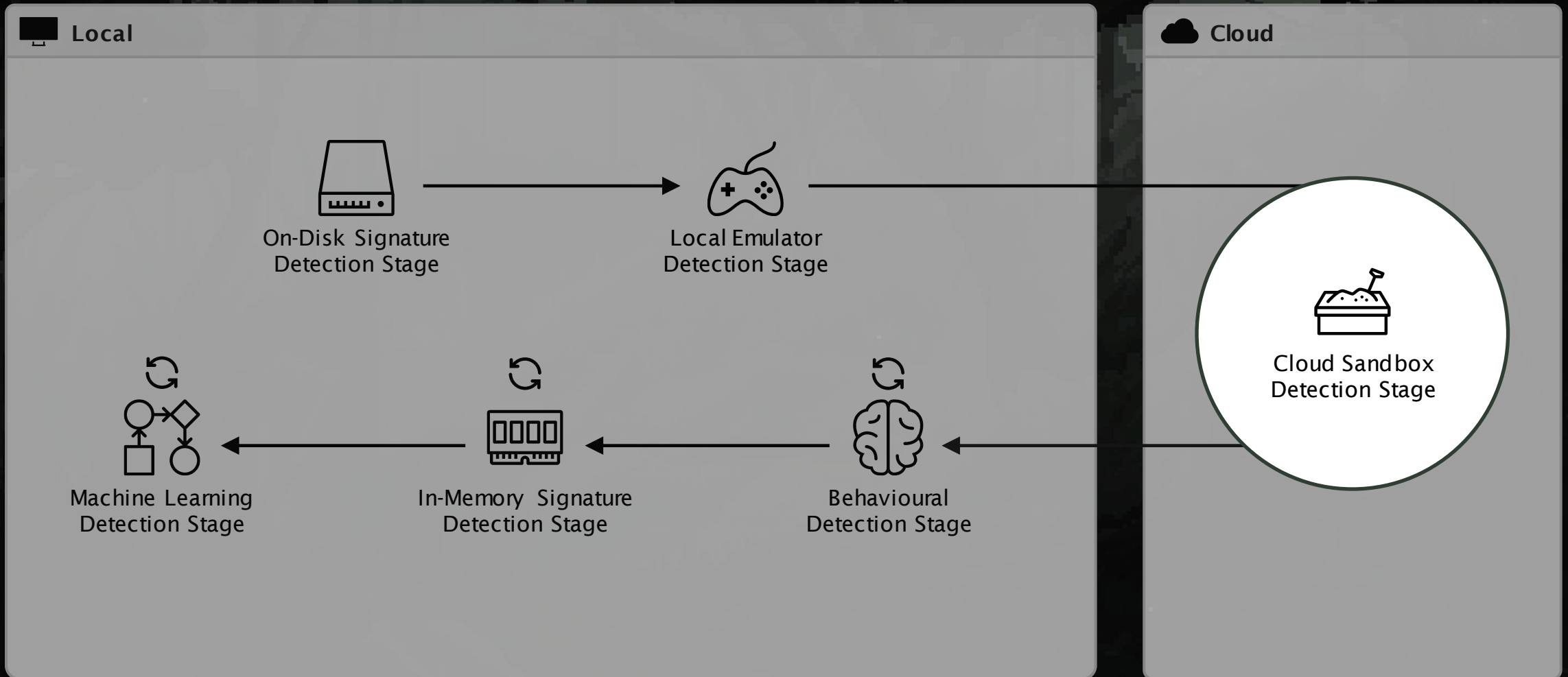
Detection stages



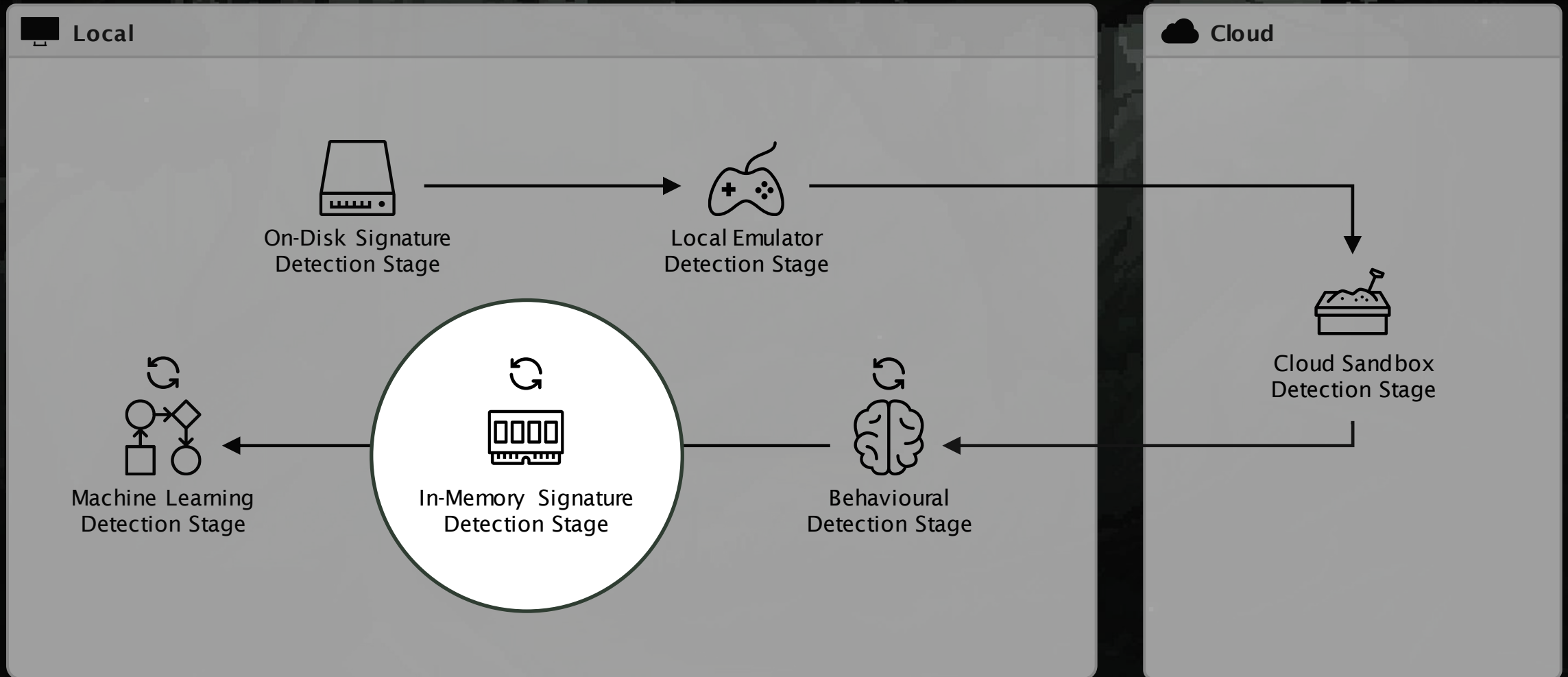
Detection stages



Detection stages



Detection stages



On-disk vs in-memory

 packed.exe (on disk)

.text segment: Decryption stub (random)

```
0x000 00110001100110011011001010011011100110110011000100011
0x010 00010110010000111001001101100011000001100100001110010
0x020 01100010011010100110001001110010110010100110100011000
0x030 01001110000110010100110110011000110011001100110010001
0x040 10011001110010011000100110001011000110010011000
```

.text segment: Encrypted code (random)

```
0x050 01001100100011100000110101011001010011011100111000011
0x060 00011011000100110010001100110001101010011000000110101
0x070 00110000001100000011011000110111011001010011001101100
0x080 10101100001011000010110011001100001001101000110001000
0x090 11001100110000001110010011011001100100001100100110000
0x100 10110011000110110001110010110010100111001001100000011
0x110 10010110000101100100001100100110001000110001001100110
0x120 01101100011000000110111011001000011100100110100001100
0x130 01001110000011000000110110011001100011011100110100001
0x140 10100011001100110011001100010001101100110000100110000
0x150 01100110011001100110010100110010011000100011010000110
0x160 01001100010001100000011010100110100001101110110010000
0x170 01101100011000000110111011001000011100100110100001100
0x180 1110000011100000111000011000110110001
```

 packed.exe (in memory)

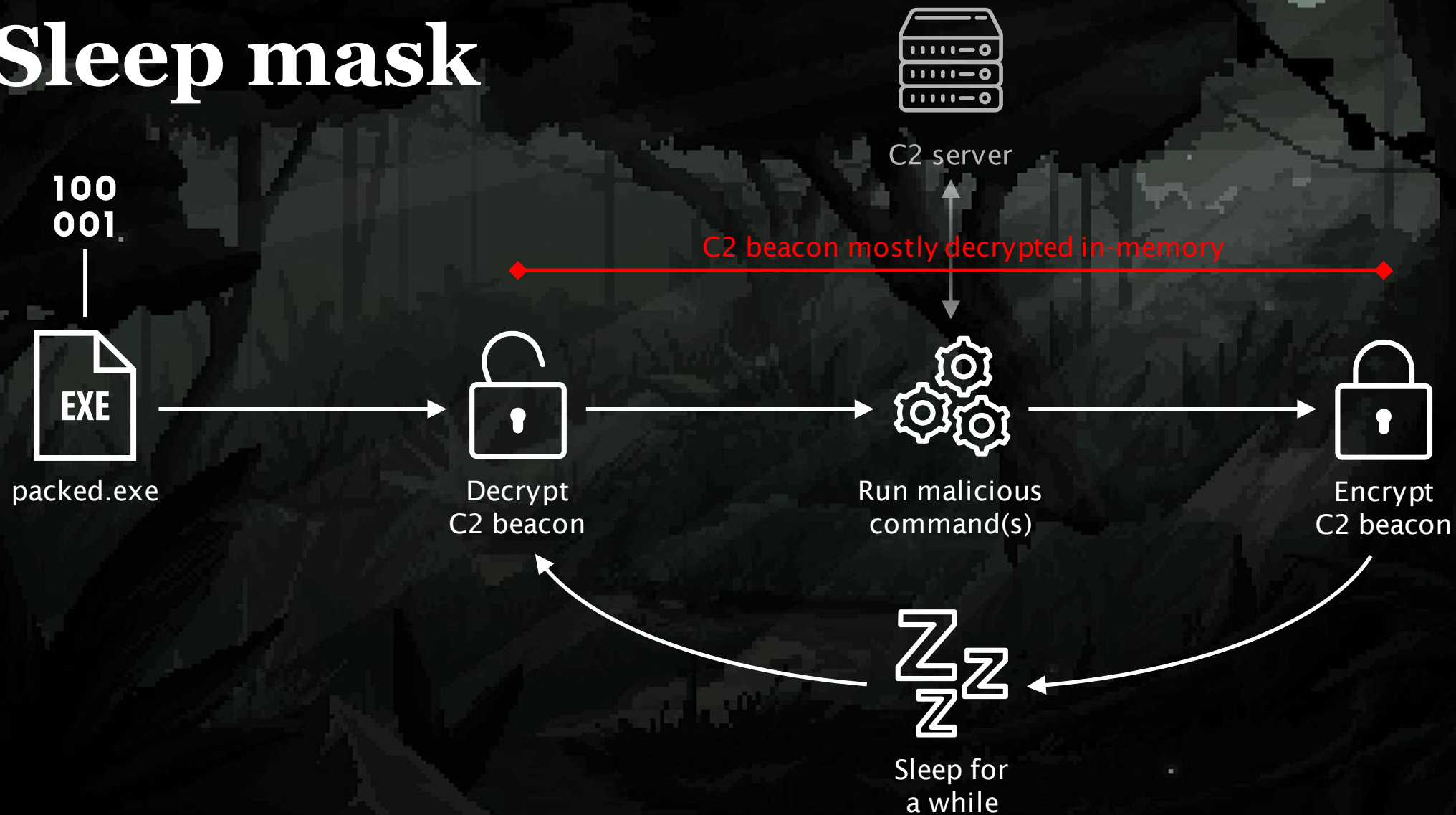
.text segment: Decryption stub (random)

```
0x000 00110001100110011011001010011011100110110011000100011
0x010 00010110010000111001001101100011000001100100001110010
0x020 01100010011010100110001001110010110010100110100011000
0x030 01001110000110010100110110011000110011001100110010001
0x040 10011001110010011000100110001011000110010011000
```

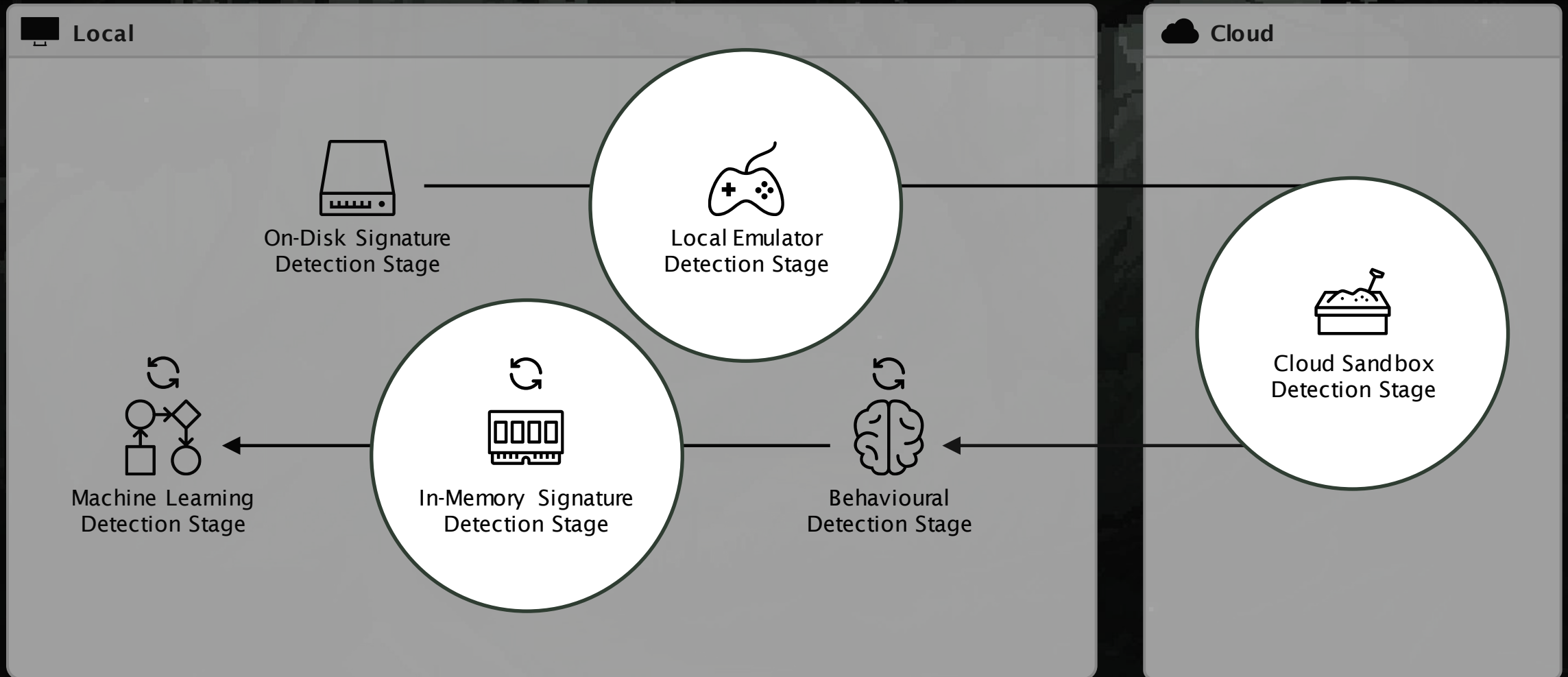
.text segment: Decrypted code (running)

```
0x050 010011001000111000001011 push rbp
0x060 000110110001001100100011 mov rbp, rsp
0x070 001100000011000000110110 mov DWORD PTR [rbp-0xC],0x1
0x080 101011000010110000101100 mov DWORD PTR [rbp-0x8],0x2
0x090 110011001100000011100100 mov eax, DWORD PTR [rbp-0xC]
0x100 101100110001101100011100 cdq
0x110 100101100001011001000011 idiv DWORD PTR [rbp-0x8]
0x120 011011000110000001101111 mov DWORD PTR [rbp-0x4],eax
0x130 010011100000110000001101 mov r15, [r11] # Hello
0x140 101000110011001100110011 mov r14, [r11+0x40] # World
0x150 011001100110011001100101 mov r13, [r11+0x50] # !
0x160 010011000100011000000110 mov eax, 0x0
0x170 011011000110000001101110 pop rbp
0x180 111000001110000011100001 ret
```

Sleep mask



Detection stages



Talk outline

1. **Polymorphic malware**
Often found in malware antique stores
 2. **Metamorphic malware**
Often seen on malware buzzword bingo cards
 3. **Dittobytes project**
A true metamorphic cross-compiler
- ❖ **Demo**

Metamorphic malware

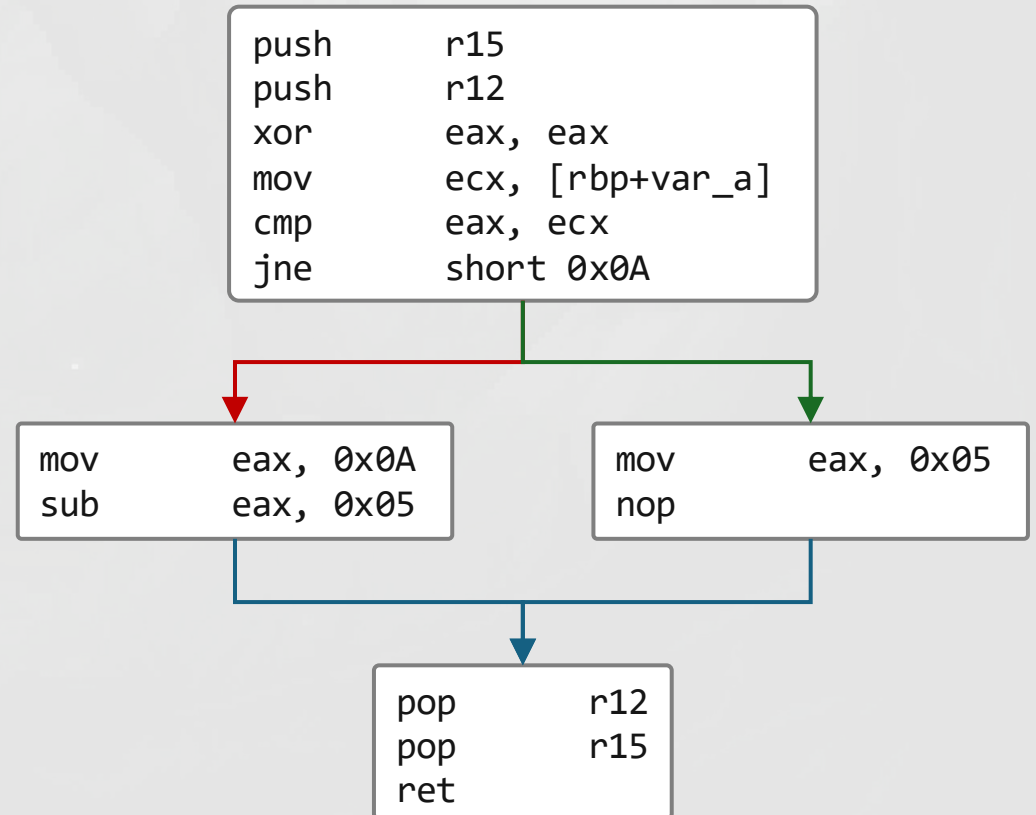


Results in *exact same* assembly, every time you compile it.

Code

```
int test_function(int arg1) {  
  
    int result;  
  
    if (arg1 == 0xA) {  
        result = 0x0A;  
        result = result - 0x05;  
    } else {  
        result = 0x05;  
    }  
  
    return result;  
}
```

Disassembly



Metamorphic malware



Results in *different* assembly,
every time you compile it.

Disassembly

```
push    r15
push    r12
xor     eax, eax
mov     ecx, [rbp+var_a]
cmp     eax, ecx
jne     short 0x0A
```

```
mov     eax, 0x0A
sub     eax, 0x05
```

```
mov     eax, 0x05
nop
```

```
pop     r12
pop     r15
ret
```

Metamorphicated (sample 1)

```
push    r15
push    r12
mov     eax, 0
mov     edx, [rbp+var_b]
cmp     eax, edx
jne     short 0x0A
```

```
mov     eax, 0x04
inc     eax
```

```
mov     eax, 0x0A
xor     eax, 0x0F
```

```
pop     r12
pop     r15
ret
```

Metamorphic malware



Results in *different* assembly, every time you compile it.

Disassembly

```
push    r15
push    r12
xor     eax, eax
mov     ecx, [rbp+var_a]
cmp     eax, ecx
jne     short 0x0A
```

```
mov     eax, 0x0A
sub     eax, 0x05
```

```
mov     eax, 0x05
nop
```

```
pop     r12
pop     r15
ret
```

Metamorphicated (sample 1)

```
push    r15
push    r12
mov     eax, 0
mov     edx, [rbp+var_b]
cmp     eax, edx
jne     short 0x0A
```

```
mov     eax, 0x04
inc     eax
```

```
mov     eax, 0x0A
xor     eax, 0x0F
```

```
pop     r12
pop     r15
ret
```

Metamorphic malware



Results in *different* assembly, every time you compile it.

Disassembly

```
push    r15
push    r12
xor     eax, eax
mov     ecx, [rbp+var_a]
cmp     eax, ecx
jne     short 0x0A
```

```
mov     eax, 0x0A
sub     eax, 0x05
```

```
mov     eax, 0x05
nop
```

```
pop     r12
pop     r15
ret
```

Metamorphicated (sample 2)

```
push    r15
push    r12
mov     eax, 0
mov     edx, [rbp+var_b]
cmp     eax, edx
jne     short 0x0A
```

```
pop     r9
push    r9
mov     rdi, 0x11
sub     rdi, eax
shr     rdi, 0x01
jmp     short 0xB
```

```
mov     eax, 0x0A
xor     eax, 0x0F
```

Metamorphic malware

The screenshot shows a web browser window displaying the GitHub Topics page for 'metamorphic-engine'. The address bar shows the URL 'https://github.com/topics/metamorphic-engine'. The page header includes the GitHub logo and navigation links: Product, Solutions, Resources, Open Source, Enterprise, and Pricing. A search bar is present with the text 'Search or jump to...'. Below the header, the 'Topics' tab is selected, showing a list of related topics: Explore, Topics, Trending, Collections, Events, and GitHub Sponsors. The main content area features a large heading '# metamorphic-engine' with a 'Star' button. Below this, it states 'Here is 1 public repository matching this topic...'. The repository listed is 'jakydibe / Zone', which has 16 stars. The repository is categorized under 'Code', 'Issues', and 'Pull requests'. It is associated with the topics 'keystone', 'capstone', 'obfuscator', 'crypter', 'metamorphism', 'metamorphic-engine', and 'pe-obfuscator'. The repository was updated on May 11 and is written in Python. On the right side, there are instructions on how to improve the topic page, create the topic, and add it to a repository.

metamorphic-engine · GitHub Topics

https://github.com/topics/metamorphic-engine

Product Solutions Resources Open Source Enterprise Pricing

Search or jump to...

Explore Topics Trending Collections Events GitHub Sponsors

metamorphic-engine Star

Here is 1 public repository matching this topic...

jakydibe / Zone Star 16

<> Code Issues Pull requests

keystone capstone obfuscator crypter metamorphism metamorphic-engine pe-obfuscator

Updated on May 11 Python

Improve this page

Add a description, image, and links to the metamorphic-engine topic page so that developers can more easily find about it.

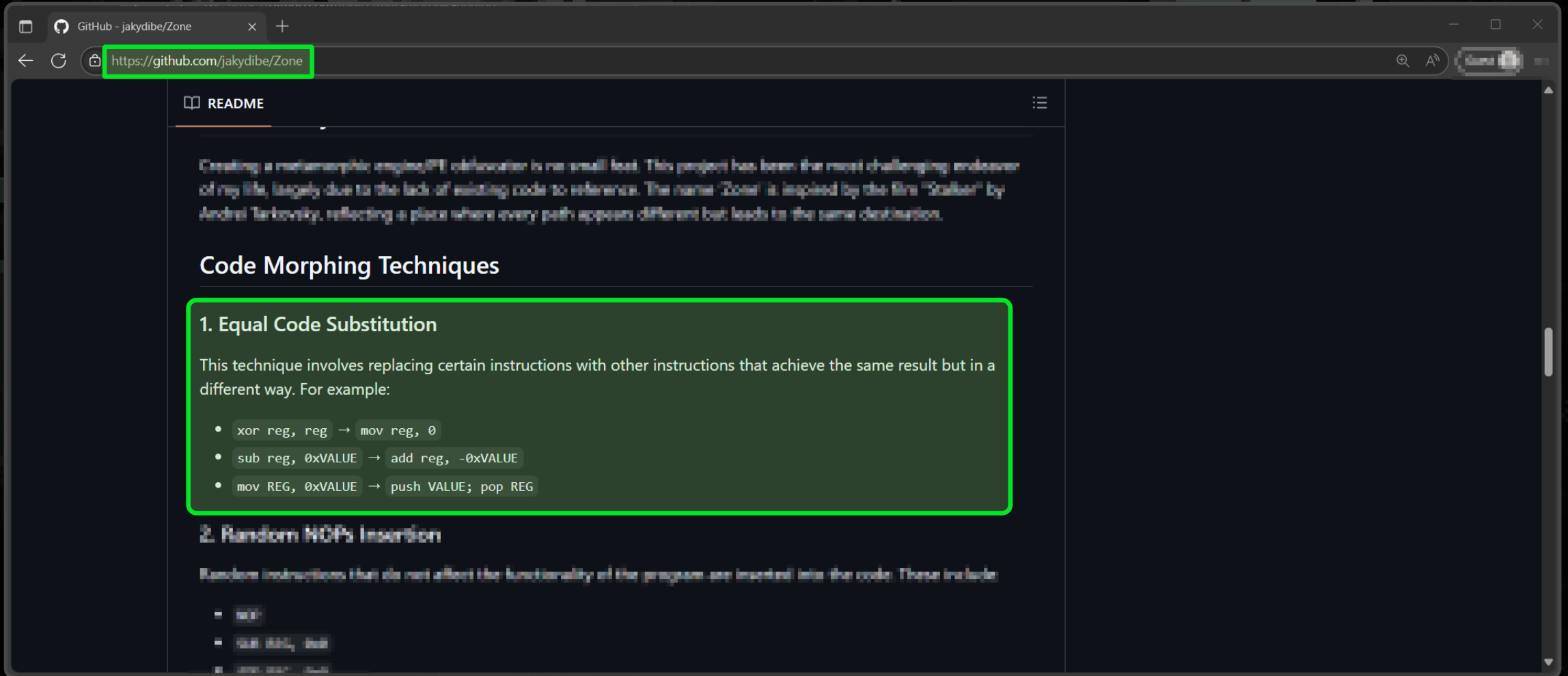
Create this topic

Add this topic to your repo

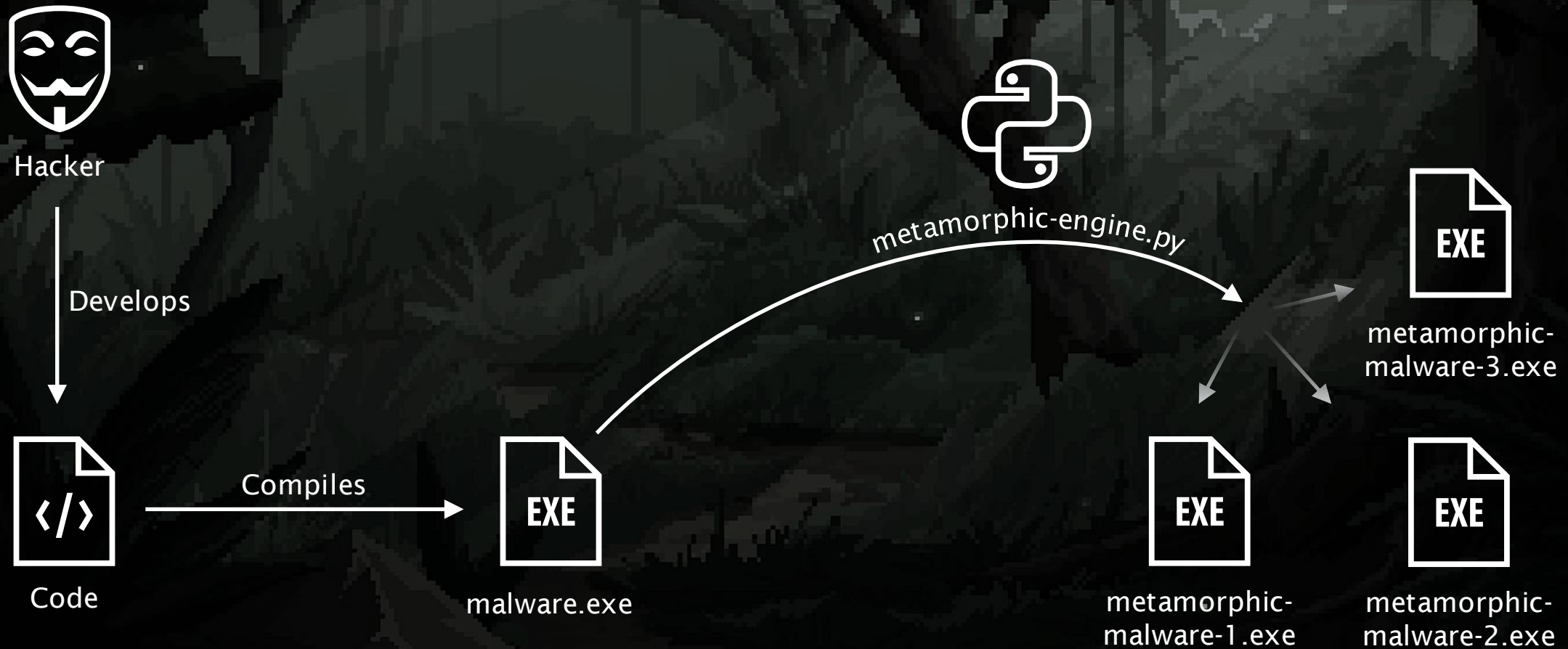
To associate your repository with the metamorphic-engine topic, visit your repo's landing page and select "manage topics."

Learn more

Metamorphic malware



Post-compile metamorphics



Post-compile metamorphics

malware.exe (original)

```
// Syscall 'print' reference
0x0000 mov rax, 1
// Write to 'stdout'
0x0004 mov rdi, 1
// Reference to the bytes to print
0x0008 lea rsi, [rip+0x0]
// Amount of bytes to print
0x000E mov rdx,
// Call the 'print' function and print H1À
0x0012 syscall
// The three bytes (inlined) (ascii: H1À)
0x0014 0x48, 0x31, 0xC0
```

malware.exe (metamorphicated)

```
// Syscall 'print' reference
0x0000 mov rax, 1
// Write to 'stdout'
0x0004 mov rdi, 1
// Reference to the bytes to print
0x0008 lea rsi, [rip+0x0]
// Amount of bytes to print
0x000E mov rdx,
// Call the 'print' function and print H1À
0x0012 syscall
// The three bytes (inlined) (ascii: H1À)
0x0014 xor rax, rax
```

cmd.exe

```
tijme@vm $ .\malware.exe
H1À
```

Post-compile metamorphics

malware.exe (original)

```
// Syscall 'print' reference
0x0000 mov rax, 1
// Write to 'stdout'
0x0004 mov rdi, 1
// Reference to the bytes to print
0x0008 lea rsi, [rip+0x0]
// Amount of bytes to print
0x000E mov rdx,
// Call the 'print' function and print H1À
0x0012 syscall
// The three bytes (inlined) (ascii: H1À)
0x0014 0x48, 0x31, 0xC0
```

malware.exe (metamorphicated)

```
// Syscall 'print' reference
0x0000 mov rax, 1
// Write to 'stdout'
0x0004 mov rdi, 1
// Reference to the bytes to print
0x0008 lea rsi, [rip+0x0]
// Amount of bytes to print
0x000E mov rdx,
// Call the 'print' function and print H1À
0x0012 syscall
// The three bytes (inlined) (metamorphicated)
0x0014 mov rax, 0
```

cmd.exe

```
tijme@vm $ .\malware.exe
H1À
```

cmd.exe

```
tijme@vm $ .\malware.exe
HÇÀ
```

And this is why...

We need pre-compile metamorphic engines!



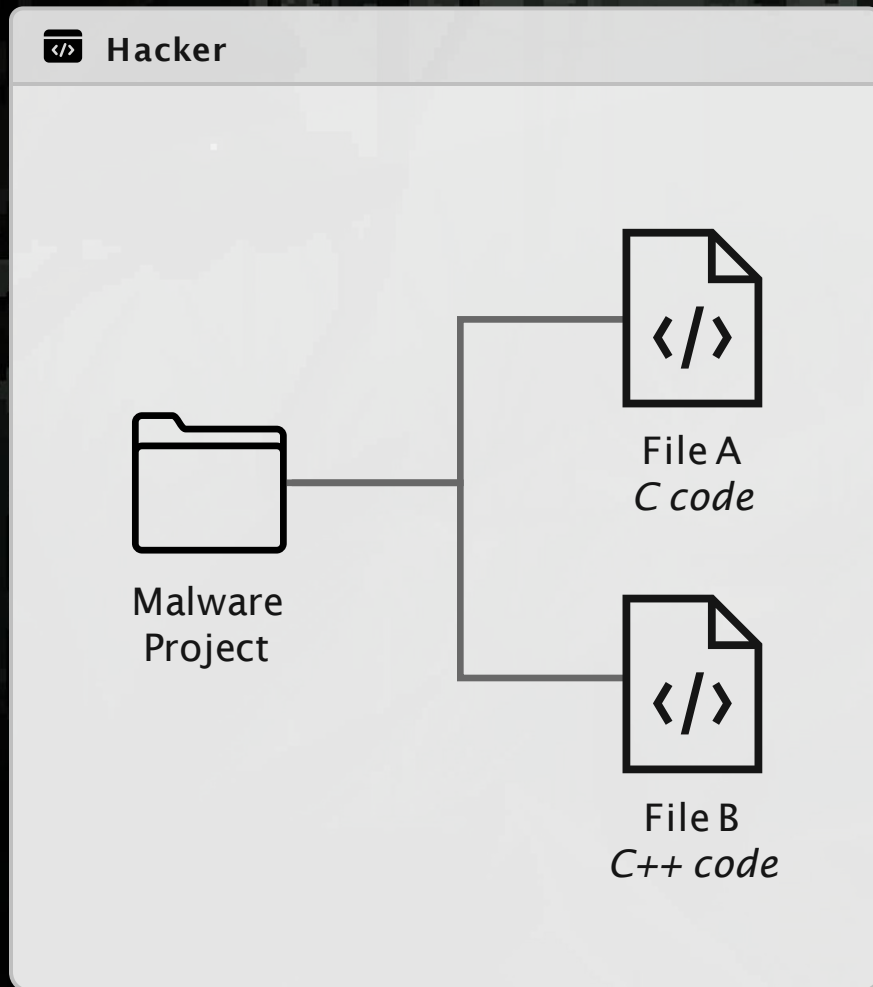
Pre-compile metamorphics



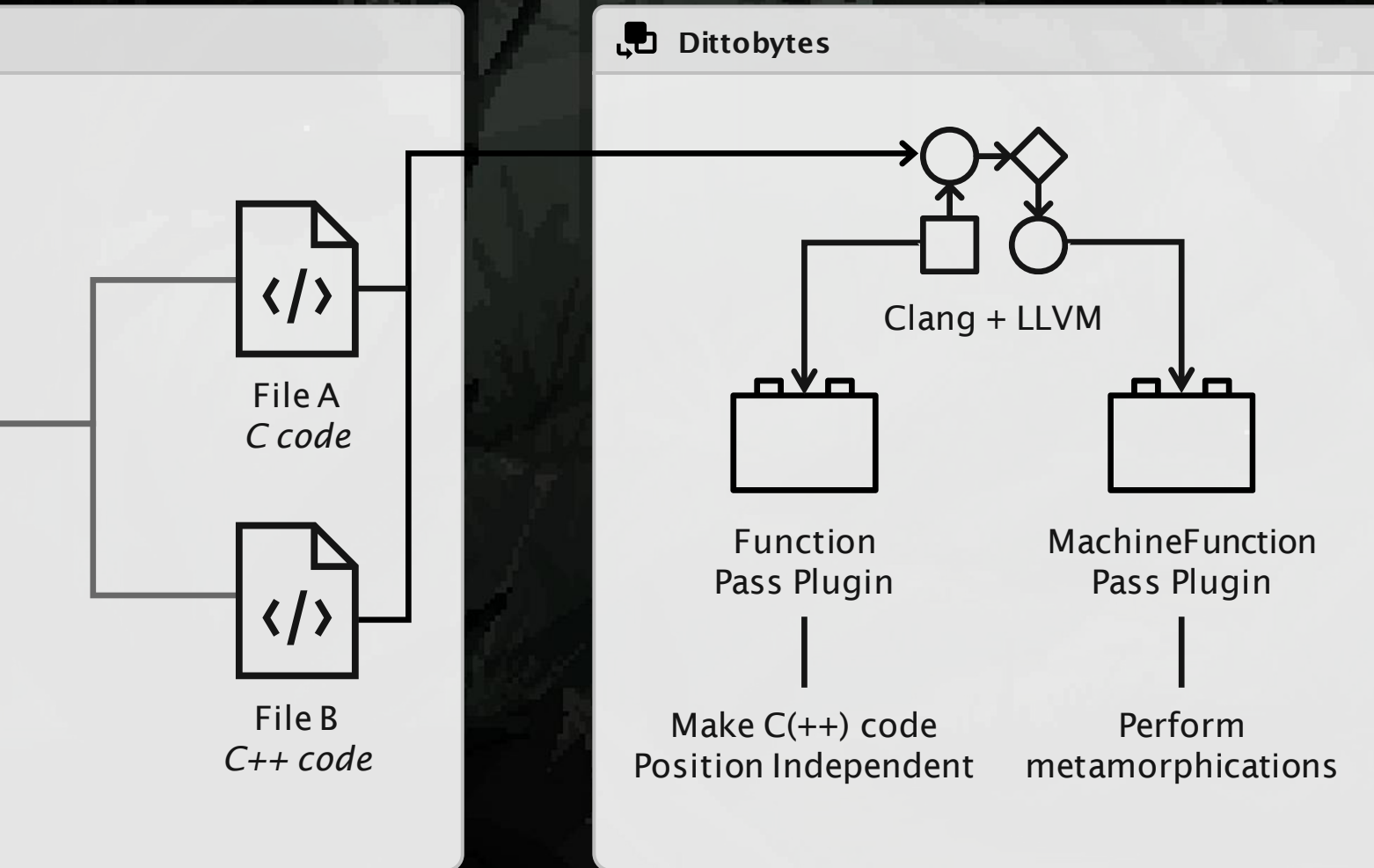
Talk outline

1. **Polymorphic malware**
Often found in malware antique stores
 2. **Metamorphic malware**
Often seen on malware buzzword bingo cards
 3. **Dittobytes project**
A true metamorphic cross-compiler
- ❖ **Demo**

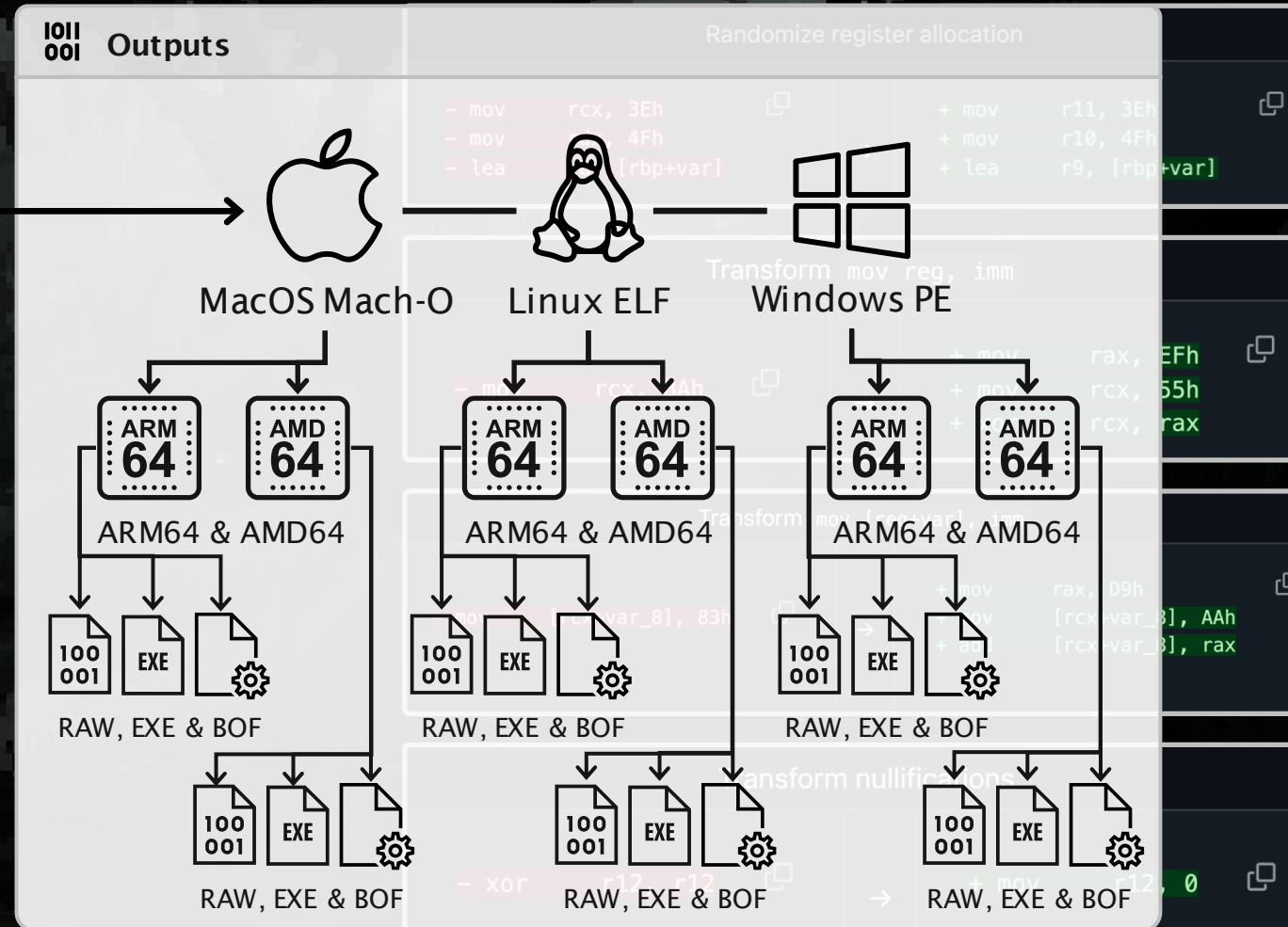
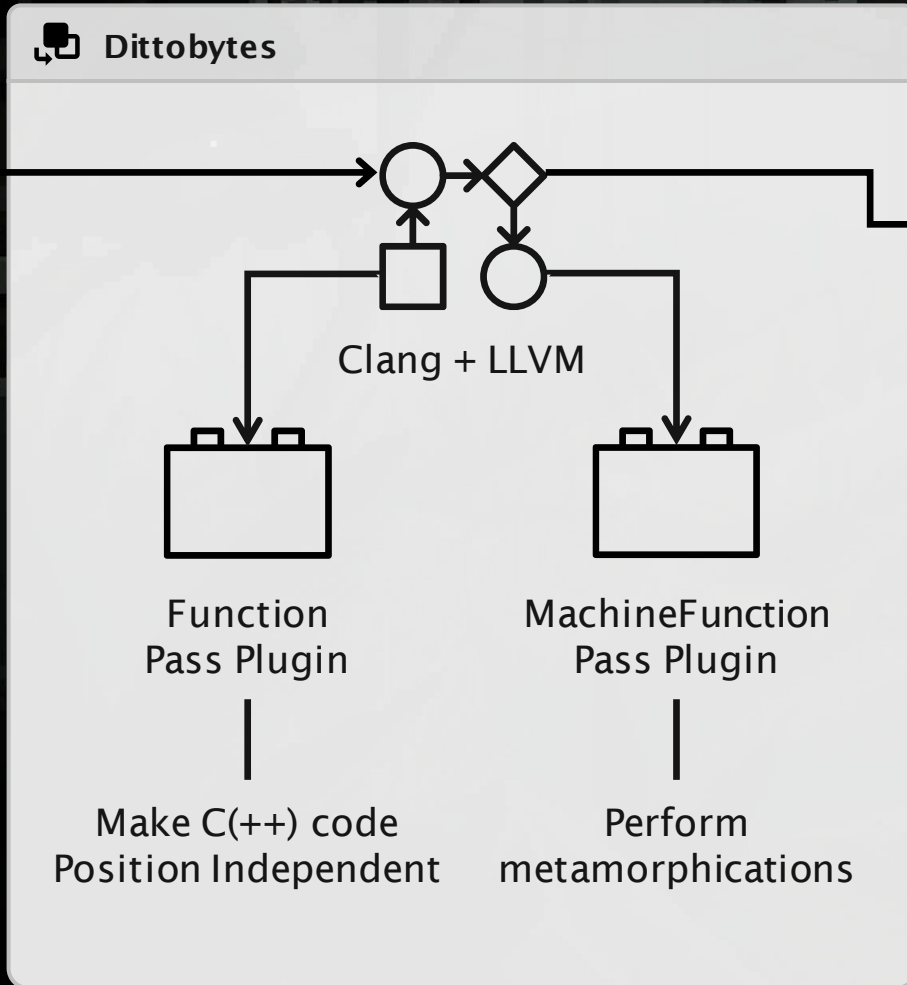
Dittobytes



Dittobytes



Dittobytes



Dittobytes

Dittobytes GitHub project

The screenshot shows a web browser window displaying the GitHub repository for 'tijme/dittobytes'. The browser's address bar shows the URL 'https://github.com/tijme/dittobytes', which is highlighted with a green rectangle. The repository page has a dark theme. At the top, there are tabs for 'README' and 'MPL-2.0 license'. Below this, the 'Getting started' section is visible, featuring a red and white icon. Under 'Getting started', there are three sub-sections: 'Overview', 'Preparing', and 'Developing'. Each sub-section has a list of topics with right-pointing chevrons. The 'Preparing' section includes two items with difficulty levels: 'Cloning the repository' (difficulty: easy) and 'Building the build tools in a Docker container' (difficulty: easy). The 'Developing' section includes two items: 'The basics' and 'A hello world'.

tijme/dittobytes: Metamorphic c x +

← ↻ 🏠 🔒 <https://github.com/tijme/dittobytes> 📦 🔊 ☆ 30 🔍 ⚙️ 👤 ⋮

📖 README 📄 MPL-2.0 license ✎ ☰

Getting started

Overview

- ▶ Directory structure

Preparing

- ▶ Cloning the repository
- ▶ Building the build tools in a Docker container
Difficulty: **easy**
- ▶ Installing the build tools on Windows Subsystem for Linux instead
Difficulty: **intermediate**

Developing

- ▶ The basics
- ▶ A hello world

Dittobytes

Creating the Dittobytes build env (Docker)



Apple M3 Pro:
20 minutes



X64 laptop:
2,5 hours

🌐 %2

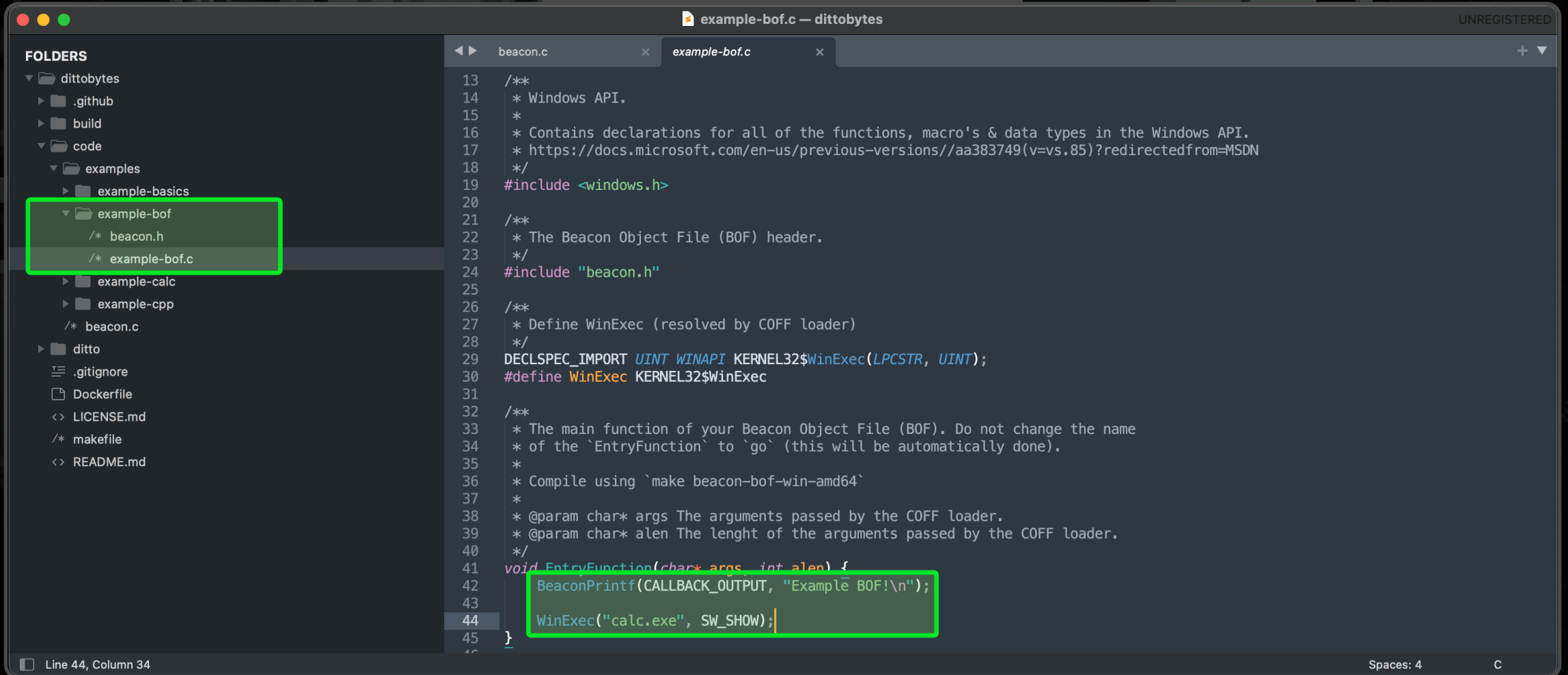
docker-buildx

```
docker buildx build -t dittobytes .
```

```
=> [internal] load build definition from Dockerfile                                0.0s1
=> => transferring dockerfile: 3.37kB                                           0.0s
=> [internal] load metadata for docker.io/library/debian:bookworm-slim          0.5s9
=> [internal] load .dockerignore                                                0.0s
=> => transferring context: 2B                                                  0.0s4
=> [internal] load build context                                                0.0s
=> => transferring context: 2.16kB                                              0.0s9
=> [ 1/27] FROM docker.io/library/debian:bookworm-slim@sha256:2424c1850714a4d94666ec928e24d86de958646737b1d113f5b2207be44d37d8 0.0s
=> CACHED [ 2/27] RUN apt update -y                                           0.0s1
=> CACHED [ 3/27] RUN apt install -y --no-install-recommends gnupg2 wget ca-certificates apt-transport-https autoconf automake cmake dpkg-dev file make patch libc6-dev mingw-w64 nano pyth 0.0s
=> [ 4/27] COPY your-root-ca.crt /usr/local/share/ca-certificates/tia.crt      0.1s0
=> [ 5/27] RUN update-ca-certificates                                          0.3s
=> [ 6/27] COPY ditto/scripts/requirements.txt /tmp/requirements.txt           0.0s6
=> [ 7/27] RUN python3 -m pip install -r /tmp/requirements.txt --break-system-packages 11.5s
=> [ 8/27] WORKDIR /opt                                                        0.0s0
=> [ 9/27] RUN git clone --depth 1 https://github.com/tijme/forked-dittobytes-macos-sdk.git macos-sdk 138.3s
=> [10/27] WORKDIR /opt                                                        0.1s6
=> [11/27] RUN wget https://github.com/mstorsjo/llvm-mingw/releases/download/20240619/llvm-mingw-20240619-msvcrt-ubuntu-20.04-x86_64.tar.xz 169.9s
=> [12/27] RUN tar -xf llvm-mingw-20240619-msvcrt-ubuntu-20.04-x86_64.tar.xz   7.4s9
=> [13/27] RUN mv llvm-mingw-20240619-msvcrt-ubuntu-20.04-x86_64 llvm-w64lin 4.9s
=> [14/27] RUN rm llvm-mingw-20240619-msvcrt-ubuntu-20.04-x86_64.tar.xz       0.1s
=> [15/27] WORKDIR /opt                                                        0.0s
=> [16/27] RUN git clone --depth 1 --branch release/18.x https://github.com/tijme/forked-dittobytes-llvm-project.git llvm-source 75.6s
=> [17/27] WORKDIR /opt/llvm-source/build                                     0.0s
=> [18/27] RUN cmake -G Ninja ../llvm -DLLVM_ENABLE_PROJECTS="clang;lld" -DCMAKE_BUILD_TYPE=Release -DLLVM_TARGETS_TO_BUILD="X86;AArch64" -DCMAKE_INSTALL_PREFIX=/opt/llvm -DBUILD_ 7.6s
=> [19/27] RUN ninja                                                         815.9s
=> => # [3097/4186] Building CXX object tools/clang/lib/Driver/CMakeFiles/obj.clangDriver.dir/ToolChains/RISCVToolchain.cpp.o
=> => # [3098/4186] Building CXX object tools/clang/lib/Driver/CMakeFiles/obj.clangDriver.dir/ToolChains/PPCLinux.cpp.o
=> => # [3099/4186] Building CXX object tools/clang/lib/Driver/CMakeFiles/obj.clangDriver.dir/ToolChains/InterfaceStubs.cpp.o
=> => # [3100/4186] Building CXX object tools/clang/lib/Driver/CMakeFiles/obj.clangDriver.dir/Types.cpp.o
=> => # [3101/4186] Building CXX object tools/clang/lib/Driver/CMakeFiles/obj.clangDriver.dir/ToolChains/WebAssembly.cpp.o
=> => # [3102/4186] Building CXX object tools/clang/lib/Driver/CMakeFiles/obj.clangDriver.dir/ToolChains/ZOS.cpp.o
```

Dittobytes

Example code to compile with Dittobytes



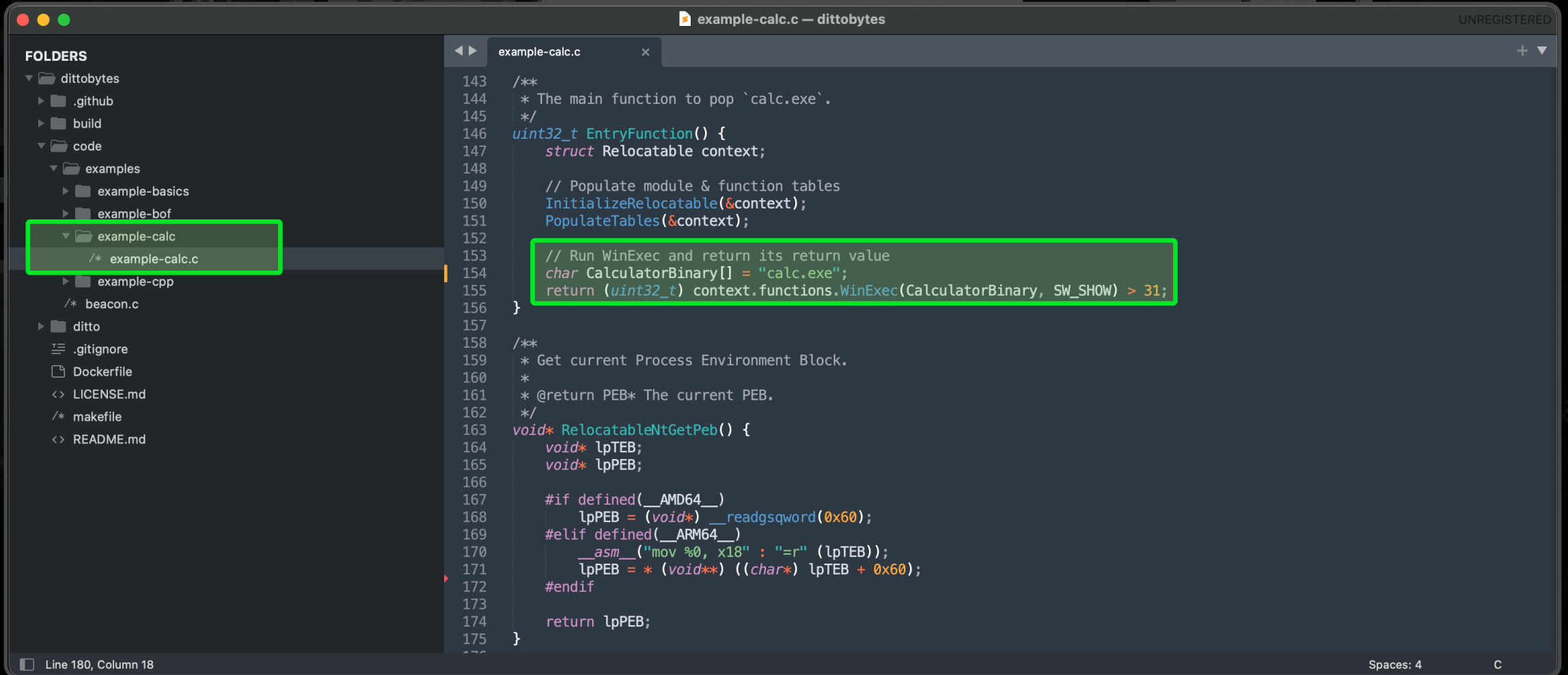
```
13 /**
14  * Windows API.
15  *
16  * Contains declarations for all of the functions, macro's & data types in the Windows API.
17  * https://docs.microsoft.com/en-us/previous-versions//aa383749(v=vs.85)?redirectedfrom=MSDN
18  */
19 #include <windows.h>
20
21 /**
22  * The Beacon Object File (BOF) header.
23  */
24 #include "beacon.h"
25
26 /**
27  * Define WinExec (resolved by COFF loader)
28  */
29 DECLSPEC_IMPORT UINT WINAPI KERNEL32$WinExec(LPCSTR, UINT);
30 #define WinExec KERNEL32$WinExec
31
32 /**
33  * The main function of your Beacon Object File (BOF). Do not change the name
34  * of the 'EntryFunction' to 'go' (this will be automatically done).
35  *
36  * Compile using `make beacon-bof-win-amd64`
37  *
38  * @param char* args The arguments passed by the COFF loader.
39  * @param char* alen The lenght of the arguments passed by the COFF loader.
40  */
41 void EntryFunction(char* args, int alen) {
42     BeaconPrintf(CALLBACK_OUTPUT, "Example BOF!\n");
43
44     WinExec("calc.exe", SW_SHOW);
45 }
```

Line 44, Column 34

Spaces: 4 C

Dittobytes

Example code to compile with Dittobytes



```
143  /**
144  * The main function to pop `calc.exe`.
145  */
146  uint32_t EntryFunction() {
147      struct Relocatable context;
148
149      // Populate module & function tables
150      InitializeRelocatable(&context);
151      PopulateTables(&context);
152
153      // Run WinExec and return its return value
154      char CalculatorBinary[] = "calc.exe";
155      return (uint32_t) context.functions.WinExec(CalculatorBinary, SW_SHOW) > 31;
156  }
157
158  /**
159  * Get current Process Environment Block.
160  *
161  * @return PEB* The current PEB.
162  */
163  void* RelocatableNtGetPeb() {
164      void* lpTEB;
165      void* lpPEB;
166
167      #if defined(__AMD64__)
168          lpPEB = (void*) __readgsqword(0x60);
169      #elif defined(__ARM64__)
170          __asm__ ("mov %0, x18" : "=r" (lpTEB));
171          lpPEB = * (void**) ((char*) lpTEB + 0x60);
172      #endif
173
174      return lpPEB;
175  }
```

Line 180, Column 18

Spaces: 4 C

Dittobytes

Building the example code using Dittobytes

user@xps: /mnt/c/Workspace

```
user@xps:/mnt/c/Workspace/Repositories/github.com-tijme/dittobytes$ |
```

Dittobytes

Building the example code using Dittobytes

```
user@xps: /mnt/c/Workspace x + v
user@xps:/mnt/c/Workspace/Repositories/github.com-tijme/dittobytes$ make help
[+] Building your code:
  - make beacon-[format]-[platform]-[arch]           // (Re-)compile your code
    ↳ Available formats: exe,raw,bof
    ↳ Available platforms: win,lin,mac
    ↳ Available architectures: amd64,arm64
  - make beacon-bof-win-amd64                       // (Re-)compile your code to Windows AMD64 BOF/COFF
  - make beacon-raw-mac-arm64                       // (Re-)compile your code to MacOS ARM64 raw shellcode
  - make beacon-raw-lin-all                         // (Re-)compile your shellcode to raw shellcode for Linux and any architecture
  - make beacon-raw-all-all                       // (Re-)compile your shellcode to raw shellcode for any platform and architecture
  - make beacon-all-all-all                     // (Re-)compile your shellcode to executable, BOF/COFF and raw shellcode for any platform and architecture
[+] Dittobytes internals
  - make ditto-loaders                             // (Re-)compile all pre-shipped Ditto shellcode loaders
  - make ditto-transpilers                         // (Re-)compile the pre-shipped Ditto LLVM transpilers/passes
[+] Test suite:
  - make test-suite-build                          // (Re-)compile all feature tests
  - make test-suite-test                          // Run all feature tests (for the current architecture)
[+] Cleanup:
  - make clean                                     // Remove all your code builds ('beacon-*) from the build folder
  - make clean-ditto-loaders                      // Remove all loader builds ('loaders-*) from the build folder
  - make clean-ditto-transpilers                  // Remove all transpiler builds from the transpiler build folders
[+] Help:
  - make help                                     // Show this help message
user@xps:/mnt/c/Workspace/Repositories/github.com-tijme/dittobytes$ |
```

Dittobytes

Running the example code using Dittobytes

```
user@xps: /mnt/c/Workspace x + v
  ✓ Modified immediate value using random option `XOR`.
  ↳ MachineTranspiler passing function `RelocatablePreliminaryGetProcAddress(...)` for step `1`.
  ↳ Running ARM64 module: TransformRegMovImmediates(option=XOR,modifyAll=1).
- Intermediate compile of build/beacon-win-arm64.meta2.mir.
- Intermediate compile of build/beacon-win-arm64.meta3.mir.
  ↳ MachineTranspiler passing function `go(...)` for step `2`.
  ↳ Running ARM64 module: TransformNullifications(modifyAll=0).
  ↳ MachineTranspiler passing function `InitializeRelocatable(...)` for step `2`.
  ↳ Running ARM64 module: TransformNullifications(modifyAll=0).
  ↳ MachineTranspiler passing function `PopulateTables(...)` for step `2`.
  ↳ Running ARM64 module: TransformNullifications(modifyAll=0).
  ↳ MachineTranspiler passing function `RelocatableNtGetPeb(...)` for step `2`.
  ↳ Running ARM64 module: TransformNullifications(modifyAll=0).
  ↳ MachineTranspiler passing function `RelocatableGetDataTableEntry(...)` for step `2`.
  ↳ Running ARM64 module: TransformNullifications(modifyAll=0).
  ↳ MachineTranspiler passing function `RelocatableStrCmp(...)` for step `2`.
  ↳ Running ARM64 module: TransformNullifications(modifyAll=0).
  ↳ MachineTranspiler passing function `RelocatablePreliminaryGetProcAddress(...)` for step `2`.
  ↳ Running ARM64 module: TransformNullifications(modifyAll=0).
- Intermediate compile of build/beacon-win-arm64.obj.
- Intermediate cleanup of build files.
- Done building BOF beacon-bof-win-arm64.
```

```
user@xps:/mnt/c/Workspace/Repositories/github.com-tijme/dittobytes$ ls -lah build/beacon*
-rwxrwxrwx 1 user user 4.0K Aug  3 18:53 build/beacon-win-amd64.exe
-rwxrwxrwx 1 user user 3.1K Aug  3 18:53 build/beacon-win-amd64.obj
-rwxrwxrwx 1 user user 2.1K Aug  3 18:53 build/beacon-win-amd64.raw
-rwxrwxrwx 1 user user 4.0K Aug  3 18:53 build/beacon-win-arm64.exe
-rwxrwxrwx 1 user user 3.1K Aug  3 18:53 build/beacon-win-arm64.obj
-rwxrwxrwx 1 user user 2.2K Aug  3 18:53 build/beacon-win-arm64.raw
user@xps:/mnt/c/Workspace/Repositories/github.com-tijme/dittobytes$
```



Dittobytes

On-disk: original (left) vs metamorphicated (right)

0 554883EC 60488D6C 24604888 63616C63 2E657865 488945F7 C645FF02 488D4DD0 UH...H.L\$H...E...E...H...
32 E86B0000 00488D4D D0E80202 00008A45 F78845C7 8A45F888 45C88A45 F98845C9 .K H.M...E...E...E...E...
64 8A45FA88 45CA8A45 FB8845CB 8A45FC88 45CC8A45 FD8845CD 8A45FE88 45CE8A45 .E...E...E...E...E...E...E...
96 FF8845CF 488B45E8 488D4DC7 BA050000 00FFD083 F81F0F97 C024010F B6C04883 .E...E...H.M...E...E...E...H...
128 C4605DC3 6666662E 0F1F8400 00000000 554881EC 90000000 488DAC24 80000000 .].fff. UH...H...\$...
160 488B4845 524E454C 33324889 4503C745 0B2E646C 6CC645F0 0048884C 6F61644C H.KERNEL32.H.E.dll.E H.LoadL
192 69627248 8945F6C7 45FE6172 7941C645 02004888 47657450 726F6341 488945E7 ibrH.E...E.aryA.E H.GetProcAH.E
224 C745EF64 64726566 C745F373 73C645F5 0048894D D88A4503 8845C88A 45048845 .E.ddref.E.ss.E H.M...E...E...E...
256 CC8A4505 8845CD8A 45068845 CE8A4507 8845CF8A 45088845 D88A4509 8845D18A .E...E...E...E...E...E...E...
288 450A8845 D28A450B 8845D38A 450C8845 D48A450D 8845D58A 450E8845 D68A450F E...E...E...E...E...E...E...
320 8845D78A 45F68845 BE8A45F7 8845BF8A 45F88845 C08A45F9 8845C18A 45FA8845 .E...E...E...E...E...E...E...
352 C28A45FB 8845C38A 45FC8845 C48A45FD 8845C58A 45FE8845 C68A45FF 8845C78A .E...E...E...E...E...E...E...
384 45008845 C88A4501 8845C98A 45028845 C8A45E7 8845AF8A 45E88845 B08A45E9 E...E...E...E...E...E...E...
416 8845B18A 45EA8845 B28A45EB 8845B38A 45EC8845 B48A45ED 8845B58A 45EE8845 .E...E...E...E...E...E...E...
448 B68A45EF 8845F78A 45F08845 B88A45F1 8845B98A 45F28845 B8A45F3 45FA8845 .E...E...E...E...E...E...E...
480 45F48845 BC8A45F5 8845BD48 8D4DCB48 8D55BE8E 28020000 4889C148 8B45D848 E...E...E...H.M.H.U...C H.H.E.H
512 89480848 8D4DCB48 8D55AE8E 10020000 4889C148 8B45D848 89481048 81C49000 H.H.M.H.U...H.H.E.H.H...
544 00005DC3 6666662E 0F1F8400 00000000 554883EC 60488D6C 24604888 4845524E .].fff. UH...H.L\$H...
576 45C3332 488945F3 C745F82E 646C6CCE 45FF0048 8B57696E 45786563 00488945 E.L32.H.E...E.dll.E H.WinExec
608 EB48894D E08A45F3 8845D38A 45F48845 D48A45F5 8845D58A 45F68845 D68A45F7 .H.M...E...E...E...E...E...E...
640 8845D78A 45F88845 D88A45F9 8845D98A 45FA8845 DA8A45FB 8845D88A 45FC8845 .E...E...E...E...E...E...E...
672 DC8A45FD 8845D08A 45FE8845 DE8A45FF 8845D0F8 8845E048 88400848 8D4DD3FF .E...E...E...E...E...H.E.H...H.M...
704 D04889C1 488B45E0 4889088A 45EB8845 CB8A45EC 8845C68A 45ED8845 CD8A45EE H.H.E.H...H...E...E...E...E...
736 8845CE8A 45EF8845 CF8A45F0 8845D08A 45F18845 D18A45F2 8845D248 8B45E048 .E...E...E...E...E...E...E...H.E.H
768 8B401048 8B4DE048 8B09488D 55CBFFD0 4889C148 8B45E048 89481848 83C4605D .H.M.H.H.U...H.H.E.H.H...
800 C3666666 6666662E 0F1F8400 00000000 554883EC 20488D6C 2420C745 FC600000 .fff. UH...H.L\$H...
832 008B45FC 65488B00 488945F0 488945F0 488945F0 488945F0 488945F0 488945F0 .E.eH.H.E.H...H.E.H...H...].f...
864 554883EC 18488D6C 24104889 4D0048C7 45F80000 0000488B 45F84883 C0104889 UH...H.L\$H.M.H.E...H.E.H...H...
896 45F0488B 450031C9 482B4DF0 4801C848 83C4185D C366662E 0F1F8400 00000000 E...E...H.M.H...H...].f...
928 554883EC 18488D6C 24104889 4D004889 55F8488B 45000FBE 0831C083 F9008845 UH...H.L\$H.M.H.U...H...E...I...
960 F7741648 8B45000F BE00488B 4DF08FBE 0939C80F 94C08845 F78A45F7 A8017502 .t.H.E...H.M...9...E...E...E...
992 EB1A488B 45004883 C0014889 4500488B 45F84883 C0014889 45F8EB86 488B4500 .H.E.H...H.E.H.E.H...H.E...H.E...
1024 0FBE0048 8B4DF80F BE0939C8 0F94C024 010FB6C0 4883C418 5DC3660F 1F440000 .H.M...9...\$...H...].f...
1056 554881EC A0000000 488DAC24 80000000 48894D10 48895508 E8F3FEFF FF488945 UH...H...\$...H.M.H.U...H...E...
1088 00488B45 00488B40 18488B40 20488945 F8488B45 F8488945 F0488B4D F08FEFE H.E.H...H...H.E.H.E.H.E.H.M...
1120 FFFF4889 45E8488B 45F0488B 00488945 F0488B45 E8488B40 30488945 E048837D .H.E.H.E.H...H.E.H.E.H...H.E.H...
1152 E0007505 E9F00000 00488B45 E0488945 D8488B45 E0488B4D D8486349 3C4801C8 u...H.E.H.E.H.E.H.M.HcH...
1184 488945D0 488B45D0 8B080800 00008945 C837DCC 007505E9 C9000000 488B45E0 H.E.H.E...E...].u...H.E...

Dittobytes



In-memory: original (left) vs metamorphicated (right)

```
RAX .text:00007FF6FAFC1000 push rbp
RDX .text:00007FF6FAFC1001 sub rsp, 60h
RIP .text:00007FF6FAFC1005 lea rbp, [rsp+60h]
R9 .text:00007FF6FAFC100A mov rax, 6578652E636C6163h
.text:00007FF6FAFC1014 mov [rbp+var_9], rax
.text:00007FF6FAFC1018 mov [rbp+var_1], 0
.text:00007FF6FAFC101C lea rcx, [rbp+var_30]
.text:00007FF6FAFC1020 call sub_7FF6FAFC1090
.text:00007FF6FAFC1025 lea rcx, [rbp+var_30]
.text:00007FF6FAFC1029 call sub_7FF6FAFC1230
.text:00007FF6FAFC102E mov al, byte ptr [rbp+var_9]
.text:00007FF6FAFC1031 mov [rbp+var_39], al
.text:00007FF6FAFC1034 mov al, byte ptr [rbp+var_9+1]
.text:00007FF6FAFC1037 mov [rbp+var_38], al
.text:00007FF6FAFC103A mov al, byte ptr [rbp+var_9+2]
.text:00007FF6FAFC103D mov [rbp+var_37], al
.text:00007FF6FAFC1040 mov al, byte ptr [rbp+var_9+3]
.text:00007FF6FAFC1043 mov [rbp+var_36], al
.text:00007FF6FAFC1046 mov al, byte ptr [rbp+var_9+4]
.text:00007FF6FAFC1049 mov [rbp+var_35], al
.text:00007FF6FAFC104C mov al, byte ptr [rbp+var_9+5]
.text:00007FF6FAFC104F mov [rbp+var_34], al
```

```
RAX .text:00007FF7D4581000 push rbp
RDX .text:00007FF7D4581001 push r13
RIP .text:00007FF7D4581003 push r15
R9 .text:00007FF7D4581005 push r14
.text:00007FF7D4581007 push rbx
.text:00007FF7D4581008 push r12
.text:00007FF7D458100A sub rsp, 68h
.text:00007FF7D458100E lea rbp, [rsp+60h]
.text:00007FF7D4581013 mov r14, 0FD1EE24A962E9956h
.text:00007FF7D458101D mov rdx, 98668764F542F835h
.text:00007FF7D4581027 xor rdx, r14
.text:00007FF7D458102A mov [rbp+30h+var_31], rdx
.text:00007FF7D458102E mov cl, 56h ; 'V'
.text:00007FF7D4581030 mov [rbp+30h+var_29], 56h ; 'V'
.text:00007FF7D4581034 xor [rbp+30h+var_29], cl
.text:00007FF7D4581037 lea r15, [rbp+30h+var_58]
.text:00007FF7D458103B mov rcx, r15
.text:00007FF7D458103E call sub_7FF7D45810E0
.text:00007FF7D4581043 lea r13, [rbp+30h+var_58]
.text:00007FF7D4581047 mov rcx, r13
.text:00007FF7D458104A call sub_7FF7D4581340
.text:00007FF7D458104F mov r11b, byte ptr [rbp+30h+var_31]
```

Dittobytes



In-memory: original (left) vs metamorphicated (right)

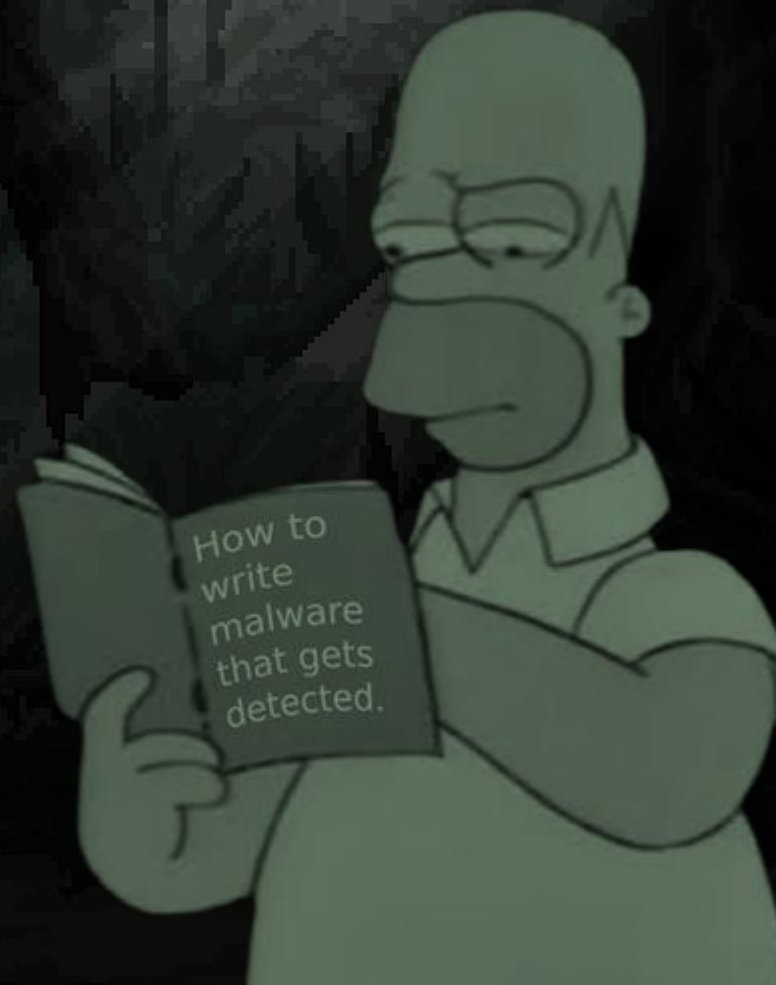


```
1 .text:00007FF6FAFC1090 push rbp
2 .text:00007FF6FAFC1091 sub rsp, 90h
3 .text:00007FF6FAFC1098 lea rbp, [rsp+80h]
4 .text:00007FF6FAFC10A0 mov rax, 32334C454E52454Bh
5 .text:00007FF6FAFC10AA mov [rbp+10h+var_D], rax
6 .text:00007FF6FAFC10AE mov [rbp+10h+var_5], 6C6C642Eh
7 .text:00007FF6FAFC10B5 mov [rbp+10h+var_1], 0
8 .text:00007FF6FAFC10B9 mov rax, 7262694C64616F4Ch
9 .text:00007FF6FAFC10C3 mov [rbp+10h+var_1A], rax
10 .text:00007FF6FAFC10C7 mov [rbp+10h+var_12], 41797261h
11 .text:00007FF6FAFC10CE mov [rbp+10h+var_E], 0
12 .text:00007FF6FAFC10D2 mov rax, 41636F7250746547h
13 .text:00007FF6FAFC10DC mov [rbp+10h+var_29], rax
14 .text:00007FF6FAFC10E0 mov [rbp+10h+var_21], 65726464h
15 .text:00007FF6FAFC10E7 mov [rbp+10h+var_1D], 7373h
16 .text:00007FF6FAFC10ED mov [rbp+10h+var_1B], 0
17 .text:00007FF6FAFC10F1 mov [rbp+10h+var_38], rcx
18 .text:00007FF6FAFC10F5 mov al, byte ptr [rbp+10h+var_D]
19 .text:00007FF6FAFC10F8 mov [rbp+10h+var_45], al
20 .text:00007FF6FAFC10FB mov al, byte ptr [rbp+10h+var_D+1]
21 .text:00007FF6FAFC10FE mov [rbp+10h+var_44], al
22 .text:00007FF6FAFC1101 mov al, byte ptr [rbp+10h+var_D+2]
```

```
1 .text:00007FF6D5471000 push rbp
2 .text:00007FF6D5471001 push r14
3 .text:00007FF6D5471003 push r12
4 .text:00007FF6D5471005 push rdi
5 .text:00007FF6D5471006 push rsi
6 .text:00007FF6D5471007 push r15
7 .text:00007FF6D5471009 push r13
8 .text:00007FF6D547100B sub rsp, 60h
9 .text:00007FF6D547100F lea rbp, [rsp+60h]
10 .text:00007FF6D5471014 mov r13, 1246845D81FC9E55h
11 .text:00007FF6D547101E mov rax, 773EE173E290FF36h
12 .text:00007FF6D5471028 xor rax, r13
13 .text:00007FF6D547102B mov [rbp+30h+var_39], rax
14 .text:00007FF6D547102F mov r9b, 2Fh ; '/'
15 .text:00007FF6D5471032 mov [rbp+30h+var_31], 2Fh ; '/'
16 .text:00007FF6D5471036 sub [rbp+30h+var_31], r9b
17 .text:00007FF6D547103A lea r9, [rbp+30h+var_60]
18 .text:00007FF6D547103E mov rcx, r9
19 .text:00007FF6D5471041 call sub_7FF6D54710E0
20 .text:00007FF6D5471046 lea r12, [rbp+30h+var_60]
21 .text:00007FF6D547104A mov rcx, r12
22 .text:00007FF6D547104D call sub_7FF6D5471330
```

But...

Does it now properly bypass
in-memory detections?



Evasion

Automatic Yara rule generator



By Florian Roth

GitHub - Neo23x0/yarGen: yarC x +

https://github.com/Neo23x0/yarGen

Neo23x0 / yarGen Public

Notifications Fork 296 Star 1.7k

<> Code Issues 11 Pull requests 8 Actions Projects Wiki Security Insights

master 1 Branch 14 Tags Go to file <> Code

Neo23x0 Merge pull request #54 from ShobhitMishra-bot/patch-1 c16faff · 4 months ago 206 Commits

3rdparty	v0.18.0	8 years ago
screens	v0.14.2	10 years ago
tools	Cleanup	10 years ago
.gitignore	Update .gitignore	2 years ago
LICENSE	License	11 years ago
README.md	v0.24.0 - AI output support	2 years ago
prepare-release.sh	chore: prepare release script	5 years ago
requirements.txt	replace pefile with lief & add support of opcode extractio...	2 years ago
yarGen.py	fix invalid escape sequence	4 months ago

About

yarGen is a generator for YARA rules

python malware malwareanalysis malware-analysis malware-research yara

Readme View license Activity 1.7k stars 90 watching 296 forks Report repository

Releases 7

Evasion



Automatic Yara rule generator

The screenshot shows the yarGen application window titled "yargen_rules.yar — yarGen". The interface includes a sidebar with a "FOLDERS" section listing the project structure: yarGen, .dbs, 3rdparty, dbs, dropzone, screens, tools, .gitignore, LICENSE, /* prepare-release.sh, <> README.md, requirements.txt, /* yarGen.py, and yargen_rules.yar. The main editor displays the content of "yargen_rules.yar" with line numbers 1 through 15. The Yara rule is for "beacon_win_amd64" and includes meta-information, a strings section with three hex-encoded strings (\$op0, \$op1, \$op2), and a condition. A green box highlights the strings section, and a callout points to the file path in the description. The status bar at the bottom indicates "Line 15, Column 2", "Spaces: 3", and "Plain Text".

```
1 rule beacon_win_amd64 {
2   meta:
3     description = "memsamples - file memsample-beacon-win-amd64-1.exe"
4     author = "Tijme Gommers"
5     reference = "https://github.com/Neo23x0/yarGen"
6     date = "2025-08-19"
7     hash1 = "5dd006e88b691ba22d6b4507ac9036e3aaaa8efb1f13552590709521a7d46b74"
8   strings:
9     $op0 = { 55 41 57 57 41 54 41 55 48 83 ec 70 48 8d 6c 24 }
10    $op1 = { 4c 8d 45 c0 4c 89 c1 e8 fe 02 00 00 8a 4d ef 88 }
11    $op2 = { 55 57 53 41 55 56 41 56 41 57 41 54 48 81 ec 98 }
12   condition:
13     uint16(0) == 0x5a4d and filesize < 20KB and
14     all of ($op*)
15 }
```

Evasion



Automatic Yara rule generator

```
~$1 -zsh
[09:17:22] ~/D/yarGen >>> yara -c -r yargen_rules.yar ~/Downloads/memsample-beacon-win-amd64-1.exe
1 (DETECTED)
[09:17:24] ~/D/yarGen >>> yara -c -r yargen_rules.yar ~/Downloads/memsample-beacon-win-amd64-2.exe
0 (undetected)
[09:17:26] ~/D/yarGen >>> yara -c -r yargen_rules.yar ~/Downloads/memsample-beacon-win-amd64-3.exe
0 (undetected)
[09:17:28] ~/D/yarGen >>> yara -c -r yargen_rules.yar ~/Downloads/memsample-beacon-win-amd64-4.exe
0 (undetected)
[09:17:31] ~/D/yarGen >>> yara -c -r yargen_rules.yar ~/Downloads/memsample-beacon-win-amd64-5.exe
0 (undetected)
[09:17:34] ~/D/yarGen >>> yara -c -r yargen_rules.yar ~/Downloads/memsample-beacon-win-amd64-6.exe
0 (undetected)
[09:17:34] ~/D/yarGen >>> 
```

 Sample used to generate signature (DETECTED)

 Samples 2 ... 6 (undetected)



Dittobytes compiled all samples based on the same code.

Evasion



Automatic Yara rule generator

UNREGISTERED

yargen_rules.yar — yarGen

FOLDERS

- yarGen
 - .dbs
 - 3rdparty
 - dbs
 - dropzone
 - screens
 - tools
- .gitignore
- LICENSE
- /* prepare-release.sh
- <> README.md
- requirements.txt
- /* yarGen.py
- yargen_rules.yar

```
1 rule beacon_win_amd64 {
2   meta:
3     description = "memsamples - file memsample-beacon-win-amd64-1.exe"
4     author = "Tijme Gommers"
5     reference = "https://github.com/Neo23x0/yarGen"
6     date = "2025-08-19"
7     hash1 = "5dd006e88b691ba22d6b4507ac9036e3aaaa8efb1f13552590709521a7d46b74"
8   strings:
9     $op0 = { 55 41 57 57 41 54 41 55 48 83 ec 70 48 8d 6c 24 }
10    $m1 = { 41 57 57 41 54 41 55 48 83 ec 70 48 8d 6c 24 }
11    55                                push    rbp
12    41 57                            push    r15
13    57                                push    rdi
14    41 54                            push    r12
15    41 55                            push    r13
16    48 83 EC 70                      sub     rsp, 70h
17    48 8D 6C 24 70                   lea     rbp, [rsp+70h]
```

Line 15, Column 2

Spaces: 3 Plain Text

Evasion

Tijme's Signature Generator



```
1 .text:00007FF7D4581000 push rbp
2 .text:00007FF7D4581001 push r13
3 .text:00007FF7D4581003 push r15
4 .text:00007FF7D4581005 push r14
5 .text:00007FF7D4581007 push rbx
6 .text:00007FF7D4581008 push r12
7 .text:00007FF7D458100A sub rsp, 68h
8 .text:00007FF7D458100E lea rbp, [rsp+60h]
9 .text:00007FF7D4581013 mov r14, 0FD1EE24A962E9956h
10 .text:00007FF7D458101D mov rdx, 98668764F542F835h
11 .text:00007FF7D4581027 xor rdx, r14
12 .text:00007FF7D458102A mov [rbp+30h+var_31], rdx
13 .text:00007FF7D458102E mov cl, 56h ; 'V'
14 .text:00007FF7D4581030 mov [rbp+30h+var_29], 56h ; 'V'
15 .text:00007FF7D4581034 xor [rbp+30h+var_29], cl
16 .text:00007FF7D4581037 lea r15, [rbp+30h+var_58]
17 .text:00007FF7D458103B mov rcx, r15
18 .text:00007FF7D458103E call sub_7FF7D45810E0
19 .text:00007FF7D4581043 lea r13, [rbp+30h+var_58]
20 .text:00007FF7D4581047 mov rcx, r13
21 .text:00007FF7D458104A call sub_7FF7D4581340
22 .text:00007FF7D458104F mov r11b, byte ptr [rbp+30h+var_31]
```

```
1 .text:00007FF6D5471000 push rbp
2 .text:00007FF6D5471001 push r14
3 .text:00007FF6D5471003 push r12
4 .text:00007FF6D5471005 push rdi
5 .text:00007FF6D5471006 push rsi
6 .text:00007FF6D5471007 push r15
7 .text:00007FF6D5471009 push r13
8 .text:00007FF6D547100B sub rsp, 60h
9 .text:00007FF6D547100F lea rbp, [rsp+60h]
10 .text:00007FF6D5471014 mov r13, 1246845D81FC9E55h
11 .text:00007FF6D547101E mov rax, 773EE173E290FF36h
12 .text:00007FF6D5471028 xor rax, r13
13 .text:00007FF6D547102B mov [rbp+30h+var_39], rax
14 .text:00007FF6D547102F mov r9b, 2Fh ; '/'
15 .text:00007FF6D5471032 mov [rbp+30h+var_31], 2Fh ; '/'
16 .text:00007FF6D5471036 sub [rbp+30h+var_31], r9b
17 .text:00007FF6D547103A lea r9, [rbp+30h+var_60]
18 .text:00007FF6D547103E mov rcx, r9
19 .text:00007FF6D5471041 call sub_7FF6D54710E0
20 .text:00007FF6D5471046 lea r12, [rbp+30h+var_60]
21 .text:00007FF6D547104A mov rcx, r12
22 .text:00007FF6D547104D call sub_7FF6D5471330
```



Evasion



Tijme's Signature Generator

```
[20:41:39] ~/Downloads >>> python3 sigmakeatron.py memsample-beacon-win-amd64-1.exe memsample-beacon-win-amd64-2.exe
Found 2 potential signatures:
- Byte sequence at offset 12 also found in other metamorphic sample:
  48 81 ec e8 03 00 00 48 8d ac 24 80 00 00 00
- Byte sequence at offset 22830 also found in other metamorphic sample:
  48 8b 85 00 04 00 00 4c
[20:41:39] ~/Downloads >>> 
```



Find similarities in two samples that are based on the same code.

Evasion



Tijme's Signature Generator

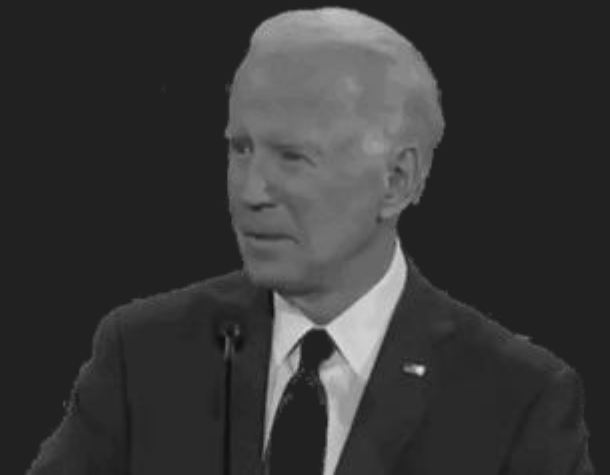
```
[09:48:29] ~/Downloads >>> yara -c -r sigmakeatron.yar ~/Downloads/memsample-beacon-win-amd64-1.exe
1 (DETECTED)
[09:48:31] ~/Downloads >>> yara -c -r sigmakeatron.yar ~/Downloads/memsample-beacon-win-amd64-2.exe
1 (DETECTED)
[09:48:32] ~/Downloads >>> yara -c -r sigmakeatron.yar ~/Downloads/memsample-beacon-win-amd64-3.exe
0 (undetected)
[09:48:34] ~/Downloads >>> yara -c -r sigmakeatron.yar ~/Downloads/memsample-beacon-win-amd64-4.exe
1 (DETECTED)
[09:48:36] ~/Downloads >>> yara -c -r sigmakeatron.yar ~/Downloads/memsample-beacon-win-amd64-5.exe
0 (undetected)
[09:48:38] ~/Downloads >>> yara -c -r sigmakeatron.yar ~/Downloads/memsample-beacon-win-amd64-6.exe
0 (undetected)
[09:48:38] ~/Downloads >>> █
```

 Samples used to generate signature (DETECTED)

 Another metamorphic sample is also DETECTED.



Dittobytes compiled all samples based on the same code.



Evasion



Tijme's Signature Generator

~%2

-zsh



```
[20:41:39] ~/Downloads >>> python3 sigmakeatron.py memsample-beacon-win-amd64-1.exe memsample-beacon-win-amd64-2.exe
Found 2 potential signatures:
- Byte sequence at offset 12 also found in other metamorphic sample:
  48 81 ec e8 03 00 00 48 8d ac 24 80 00 00 00
- Byte sequence at offset 22830 also found in other metamorphic sample:
  48 8b 85 00 04 00 00 4c
```

```
[20:41:39] ~/Downloads >>>
```

48 81 EC E8 03 00 00	sub	rsp, 3E8h
48 8D AC 24 80 00 00 00	lea	rbp, [rsp+80h]
48 8B 85 00 04 00 00	mov	rax, [rbp+400h]
4C 8B 58 50	mov	r11, [rax+50h]



Dittobytes compiled all samples based on the same code.

Evasion

Tijme's Signature Generator



Google TI - Search - p:0 type:exe x

https://www.virustotal.com/gui/search/p%253A0%2520type%253Aexecutable%2520s%253A100%252B%2520tag%253Assigned%...

p:0 type:executable s:100+ tag:signed content:{48 81 ec e8 03 00 00 48}

Seen in the last month

Search for all executables that:

- Have a valid code signature.
- Have been submitted 100+ times.
- Are not marked as malicious.
- Contain the specific byte sequence.

1b727cd6e9b6277488dcdab444	1 / 100	0 / 72	2025-06-29 01:01:43	2025-08-10 14:00:16	1.3 K	11.78 MB
cc550dff570e0b7a8fc36e7146...	1 / 100	0 / 53	2025-06-29 14:59:45	2025-08-08 02:16:40	300	32.08 MB
Logitech G HUB	1 / 100	0 / 69	2025-07-01 16:18:44	2025-08-11 17:03:49	2.6 K	66.20 MB
OfficeClickToRun.exe	0 / 100	0 / 73	2025-07-01 21:09:47	2025-08-11 16:07:47	100	12.44 MB
C:\Program Files\WizTr...	1 / 100	0 / 73	2025-07-02 01:53:23	2025-08-11 12:05:04	243	10.34 MB

Evasion



Tijme's Signature Generator

Google TI - Search - p:0 type:e x Google TI - File - 5121084f056 x +

https://www.virustotal.com/gui/file/5121084f0561cad856bd2350e04f5eb155d37f71cbaaf4ec9e3aadb7d002319/details

5121084f0561cad856bd2350e04f5eb155d37f71cbaaf4ec9e3aadb7d002319

Smart search | Tijme Gommers

Signers

- Microsoft Corporation
 - Name: Microsoft Corporation
 - Status: Valid
 - Issuer: Microsoft Windows Code Signing PCA 2024
 - Valid From: 06:08 PM 04/17/2025
 - Valid To: 06:08 PM 04/15/2026
 - Valid Usage: Code Signing, 1.3.6.1.4.1.311.61.6.1, 1.3.6.1.4.311.76.22.6.1
 - Algorithm: sha384RSA
 - Thumbprint: 883B405B08B7039DEECA731F53EB236E3CF198A3
 - Serial Number: 33 00 00 00 54 53 E5 67 39 FB 6E 57 BB 00 00 00 00 54
- + Microsoft Windows Code Signing PCA 2024
- + Microsoft Root Certificate Authority 2010

Counter Signers

- + Microsoft Time-Stamp Service
- + Microsoft Time-Stamp PCA 2010
- + Microsoft Root Certificate Authority 2010

OfficeClickToRun.exe

Evasion



Tijme's Signature Generator

Google TI - Search - p:0 type:e x Google TI - File - 0b7c0f0bebb x +

https://www.virustotal.com/gui/file/0b7c0f0bebb593ecfd4a2486454b41bb89aaf323ac8ed91c528107356f2d6442/details

0b7c0f0bebb593ecfd4a2486454b41bb89aaf323ac8ed91c528107356f2d6442

Smart search | Tijme Gommers

Signers

— Rockstar Games, Inc.

Name	Rockstar Games, Inc.
Status	Valid
Issuer	DigiCert Trusted G4 Code Signing RSA4096 SHA384 2021 CA1
Valid From	12:00 AM 01/18/2023
Valid To	11:59 PM 01/20/2026
Valid Usage	Code Signing
Algorithm	sha256RSA
Thumbprint	565932392989B3616F2968E1B1D6F974561B1F32
Serial Number	0D 88 C0 8F 56 6D 2B 1F 0C 19 4D B1 F8 CA C9 A9

+ DigiCert Trusted G4 Code Signing RSA4096 SHA384 2021 CA1

+ DigiCert Trusted Root G4

+ DigiCert

Counter Signers

+ DigiCert Timestamp 2024

+ DigiCert Trusted G4 RSA4096 SHA256 TimeStamping CA

Rockstar-Games-Launcher.exe

So how do we detect it?

Entropy-based detection

No, metamorphications do not encrypt or compress (parts of) the code/binary.

RWX-memory detection

No, metamorphic malware does not need to allocate writable and executable memory regions.

Stack-based detection

No, stack-based strings/sequences not present if you use existing string encryption techniques.

API-import detection

No, Dittobytes produces Position Independent Code (PIC), which does not use the import table.

API-call sequence detection

Detection possible if your malware produces a suspicious sequence of API-calls.

Network traffic detection

Detection possible if your malware produces a suspicious network traffic pattern.



In Memory of In-Memory Detection

Thank you

Questions? I'm around all day at OrangeCon!

