Reversing In Wonderland

Neural Network Based Malware Detection Techniques

HITCON 2020

aaaddress1@chroot.org

#Windows #Reversing #Pwn #Exploit

- Master degree at CSIE, NTUST
- **Security Researcher** - chr0.ot
- **Speaker** - BlackHat, DEFCON, HITCON, CYBERSEC
- aaaddress1@chroot.org
- 30cm.tw & Hao's Arsenal

- Associate Professor of CSIE, NTUST
- Joint Associate Research Fellow of CITI, Academia Sinica
- smcheng@mail.ntust.edu.tw

#4G #5G #LTE_Attack #IoT

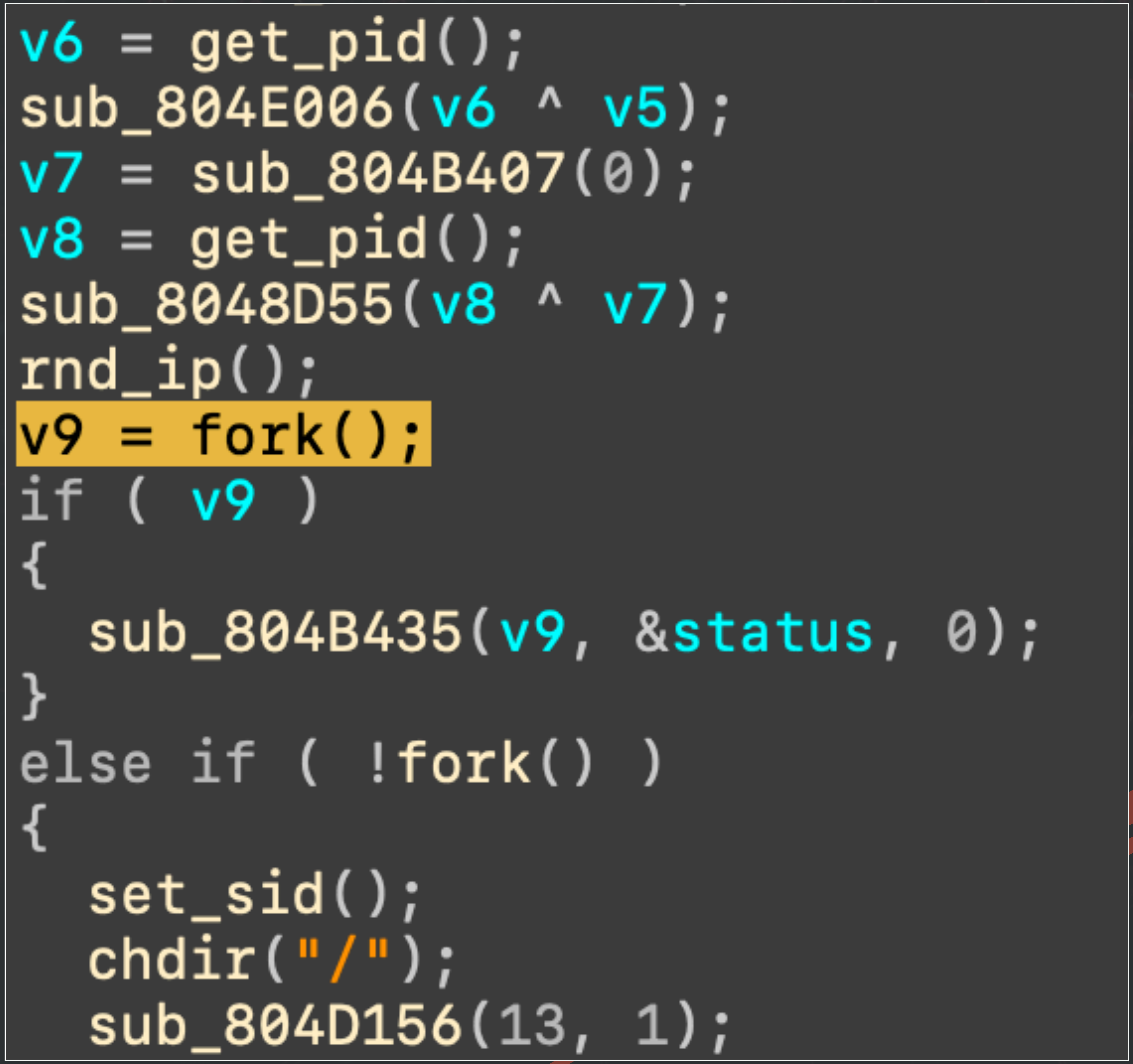
/?outline

1. Malware in the Wild
2. Semantics
3. Semantic-Aware: PV-DM
4. Asm2Vec & Experiment
5. Challenge

Malware In the Wild

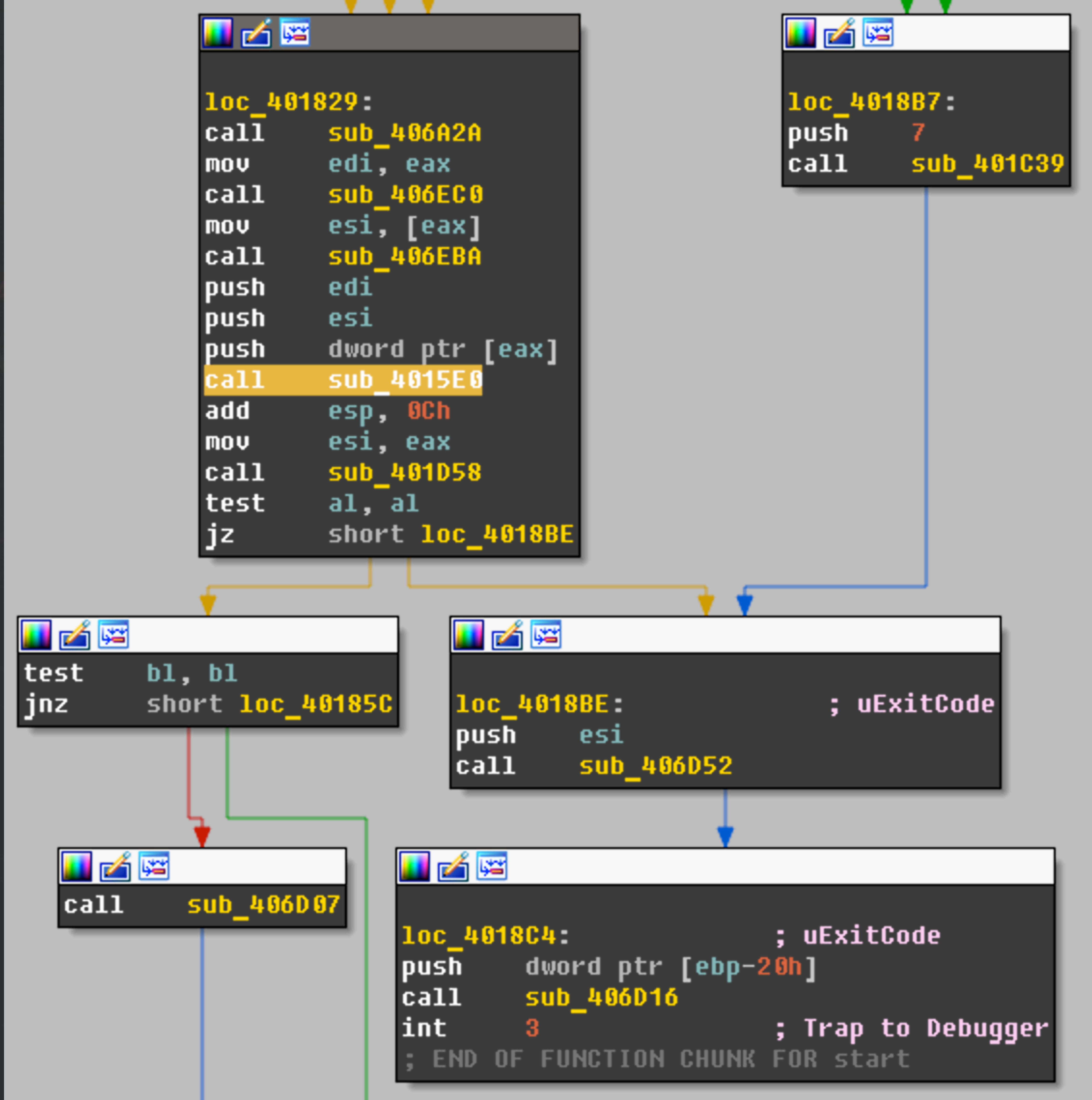
aaaddress1@chroot.org

#behavior



```
v6 = get_pid();
sub_804E006(v6 ^ v5);
v7 = sub_804B407(0);
v8 = get_pid();
sub_8048D55(v8 ^ v7);
rnd_ip();
v9 = fork();
if ( v9 )
{
  sub_804B435(v9, &status, 0);
}
else if ( !fork() )
{
  set_sid();
  chdir("/");
  sub_804D156(13, 1);
```

#behavior



```
loc_401829:
call    sub_406A2A
mov     edi, eax
call    sub_406EC0
mov     esi, [eax]
call    sub_406EBA
push    edi
push    esi
push    dword ptr [eax]
call    sub_4015E0
add     esp, 0Ch
mov     esi, eax
call    sub_401D58
test    al, al
jz      short loc_4018BE
```

```
loc_4018B7:
push    7
call    sub_401C39
```

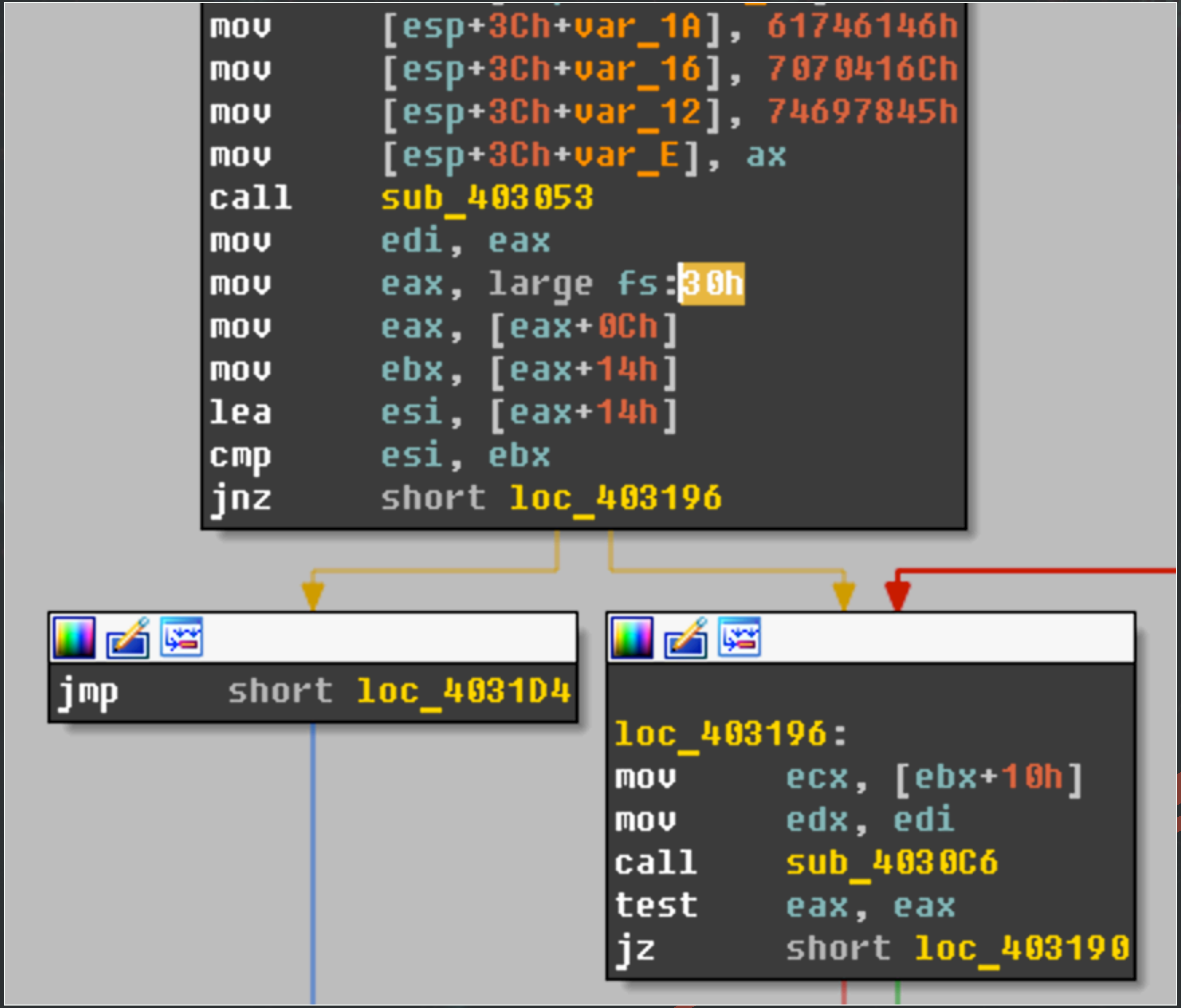
```
test    bl, bl
jnz     short loc_40185C
```

```
loc_4018BE:                 ; uExitCode
push    esi
call    sub_406D52
```

```
call    sub_406D07
```

```
loc_4018C4:                 ; uExitCode
push    dword ptr [ebp-20h]
call    sub_406D16
int     3                   ; Trap to Debugger
; END OF FUNCTION CHUNK FOR start
```

#behavior



```
mov     [esp+3Ch+var_1A], 61746146h
mov     [esp+3Ch+var_16], 7070416Ch
mov     [esp+3Ch+var_12], 74697845h
mov     [esp+3Ch+var_E], ax
call    sub_403053
mov     edi, eax
mov     eax, large fs:30h
mov     eax, [eax+0Ch]
mov     ebx, [eax+14h]
lea     esi, [eax+14h]
cmp     esi, ebx
jnz     short loc_403196
```

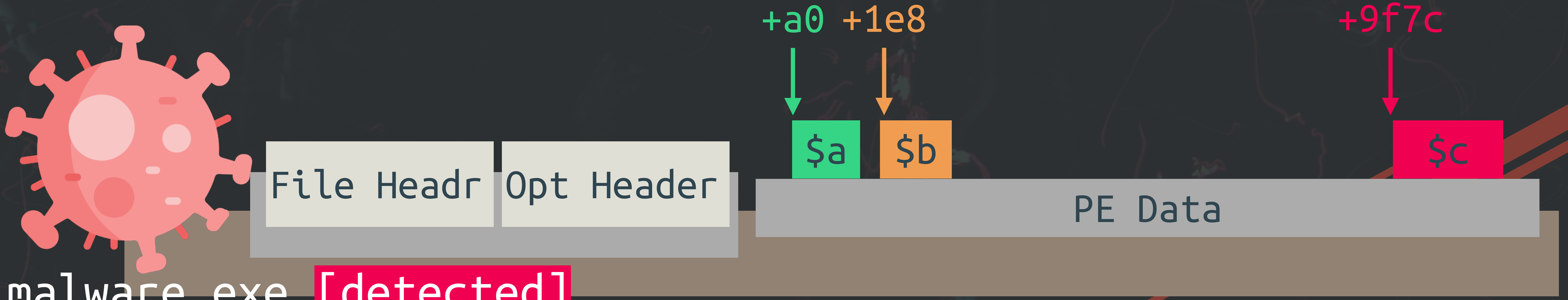
```
jmp     short loc_4031D4
```

```
loc_403196:
mov     ecx, [ebx+10h]
mov     edx, edi
call    sub_4030C6
test    eax, eax
jz      short loc_403190
```

#YARA

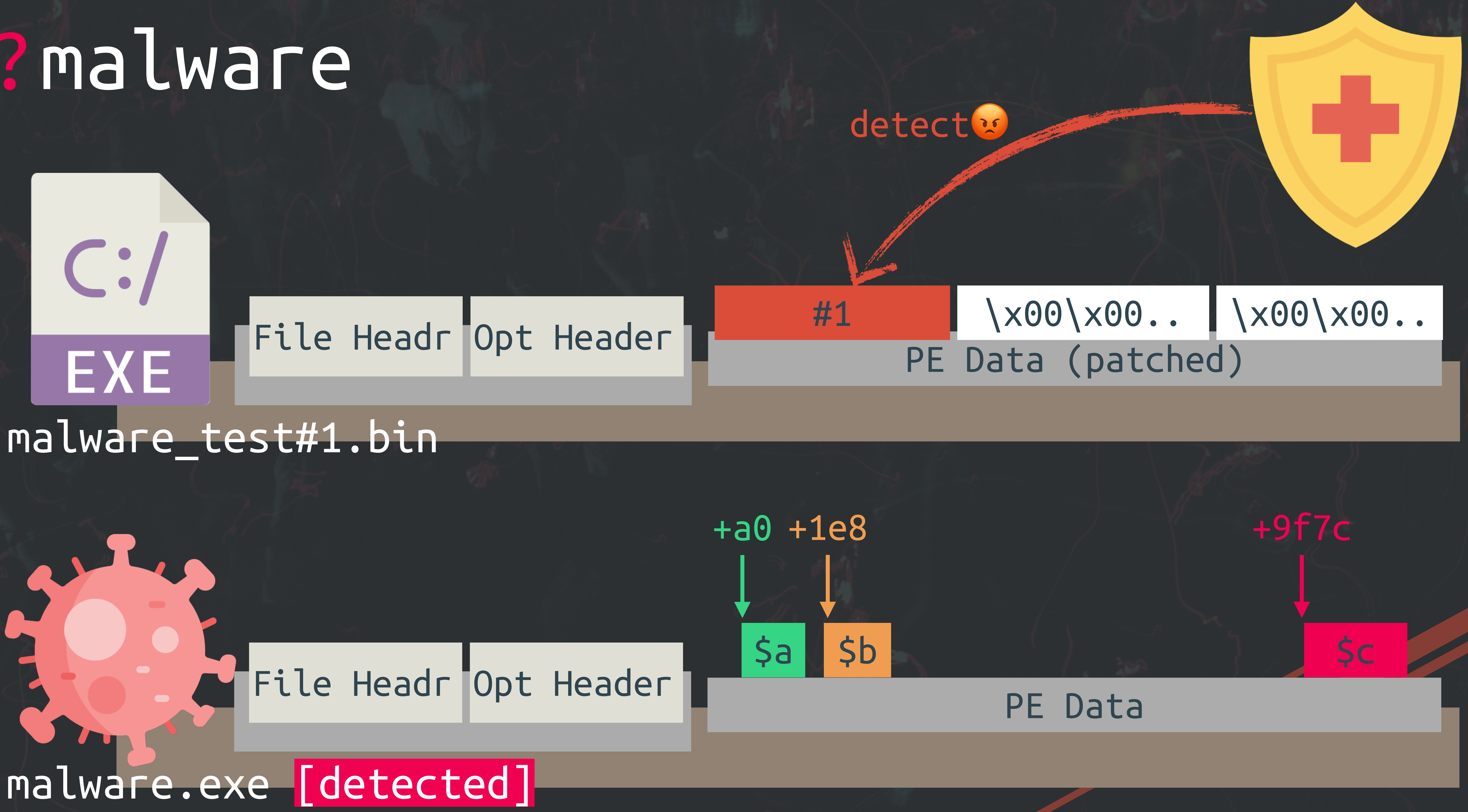
```
rule silent_banker : banker {
    meta:
        description = "malware in the wild"
        threat_level = 3
        in_the_wild = true
    strings:
        $a = {6A 40 68 00 30 00 00 6A 14 8D 91}
        $b = {8D 4D B0 2B C1 83 C0 27 59 F7 F9}
        $c = "UVODFRYSIHLNWPEJXQZAKCBGMT"
    condition:
        $a or $b or $c
}
```

/?malware

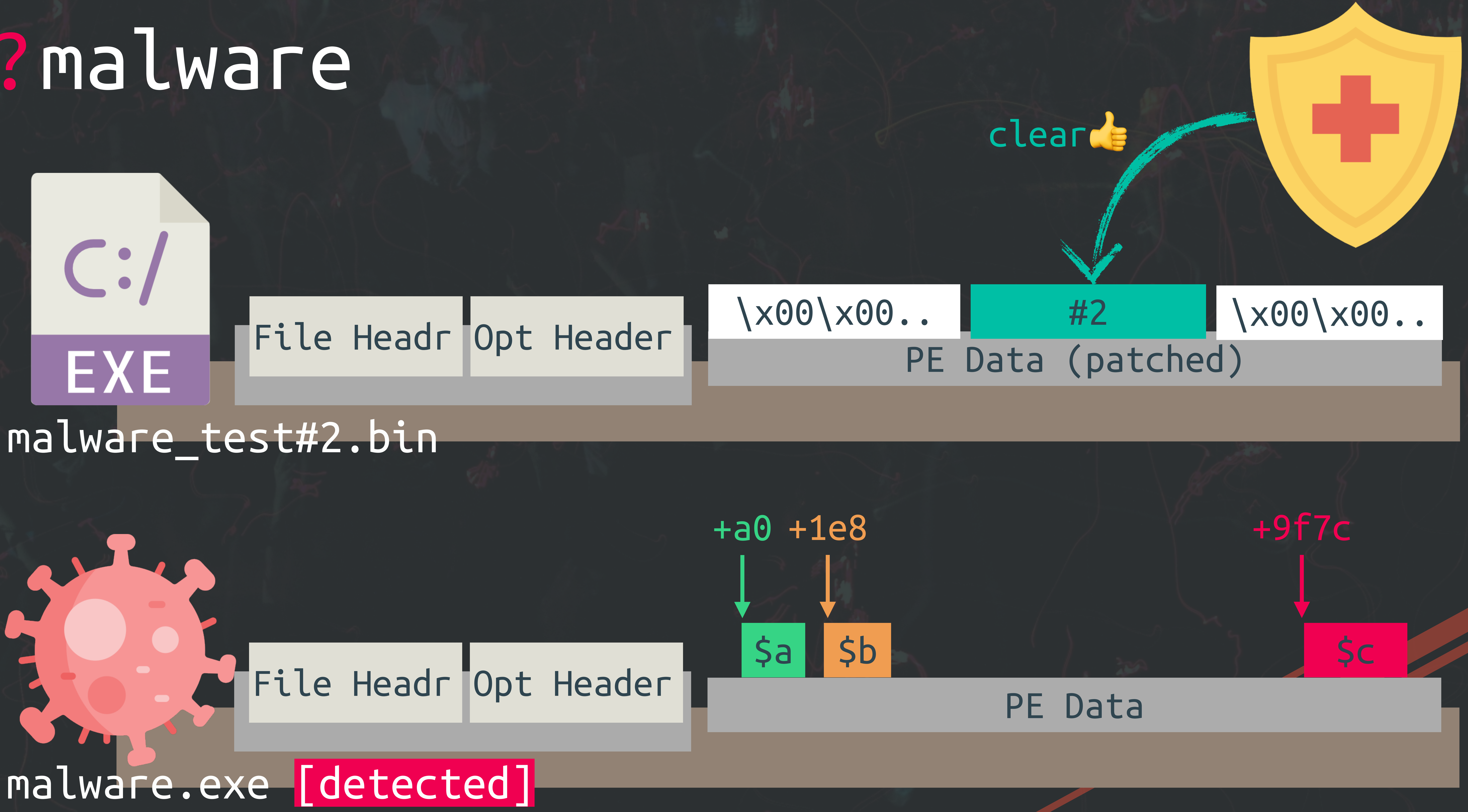


malware.exe [detected]

/?malware


detect😡
C:/
EXE
File Headr
Opt Header
#1
\x00\x00..
\x00\x00..
PE Data (patched)
malware_test#1.bin
+a0
+1e8
+9f7c
$a
$b
$c
File Headr
Opt Header
PE Data
malware.exe [detected]

/?malware



clear👍

\x00\x00.. | #2 | \x00\x00..

PE Data (patched)

File Headr | Opt Header

malware_test#2.bin

+a0 +1e8 +9f7c

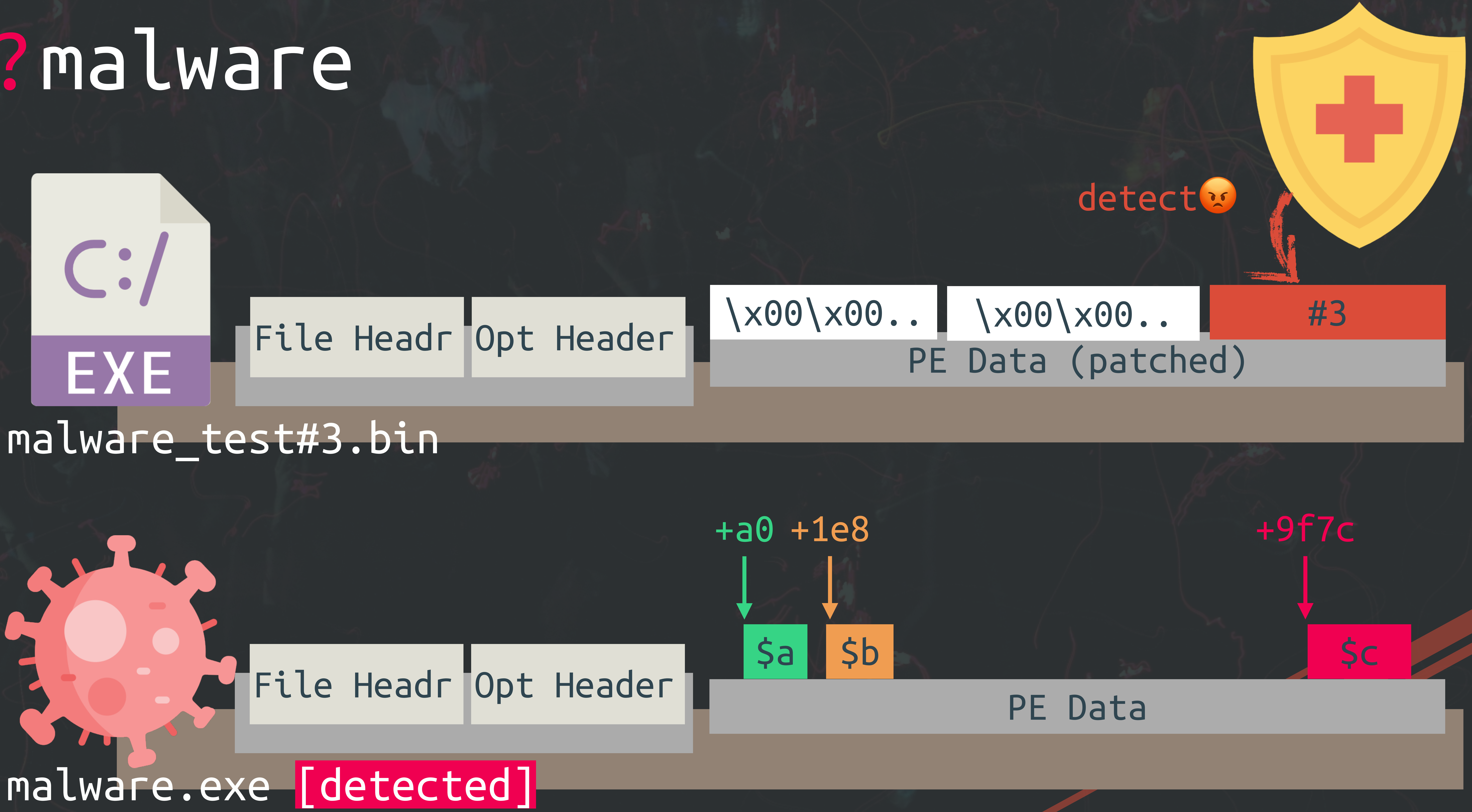
$a | $b | $c

PE Data

File Headr | Opt Header

malware.exe [detected]

/?malware



detect😡

File Headr

Opt Header

\x00\x00..

\x00\x00..

#3

PE Data (patched)

malware_test#3.bin

+a0 +1e8

+9f7c

$a

$b

$c

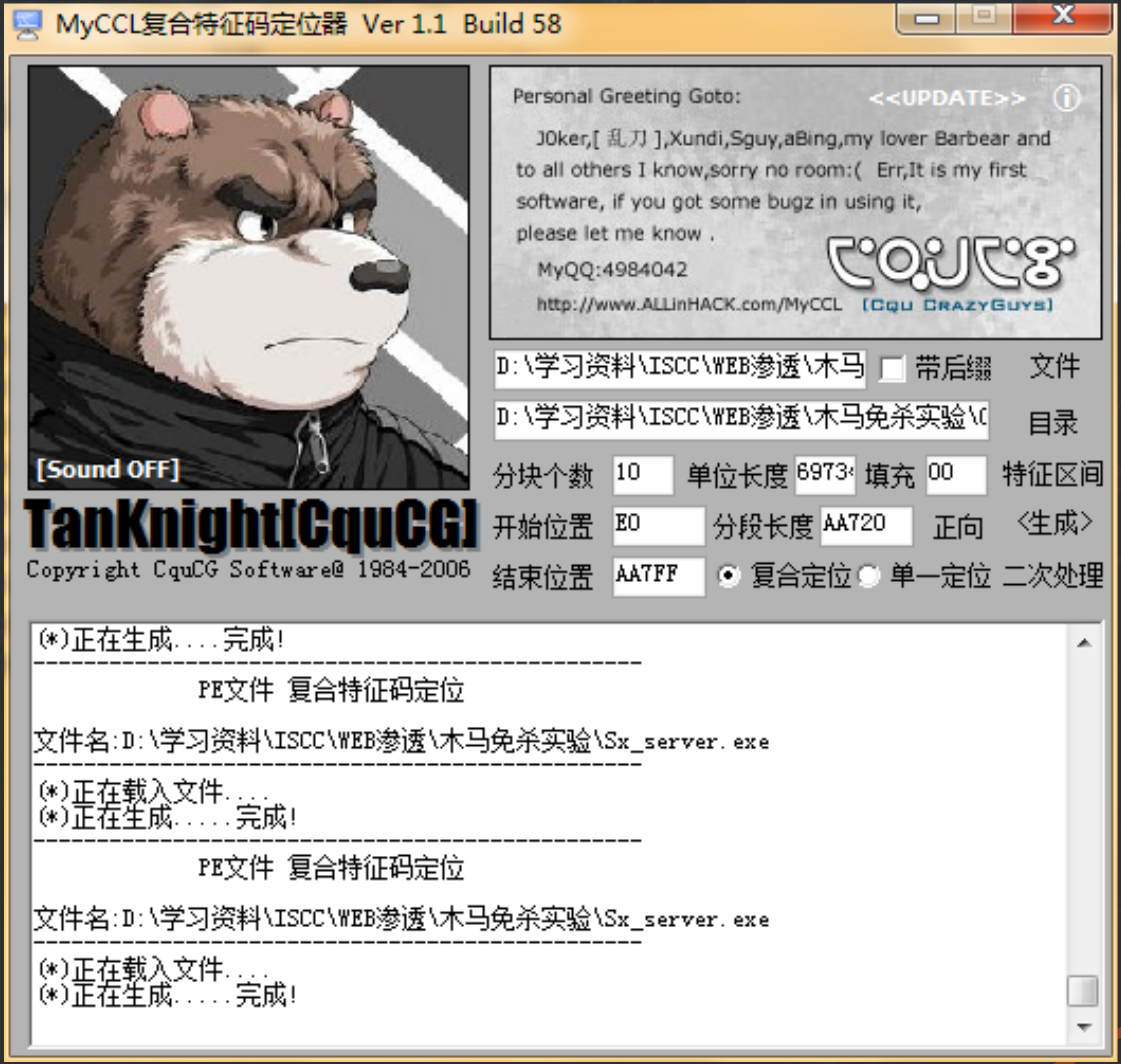
File Headr

Opt Header

PE Data

malware.exe [detected]

#免殺



MyCCL复合特征码定位器 Ver 1.1 Build 58

Personal Greeting Goto: <<UPDATE>>

J0ker,[乱刀],Xundi,Sguy,aBing,my lover Barbear and to all others I know,sorry no room:(Err,It is my first software, if you got some bugz in using it, please let me know .

MyQQ:4984042

http://www.ALLinHACK.com/MyCCL [Cqu CrazyGuys]

[Sound OFF]

TanKnight[CquCG]

Copyright CquCG Software@ 1984-2006

D:\学习资料\ISCC\WEB渗透\木马 ☐ 带后缀 文件

D:\学习资料\ISCC\WEB渗透\木马免杀实验\(目录

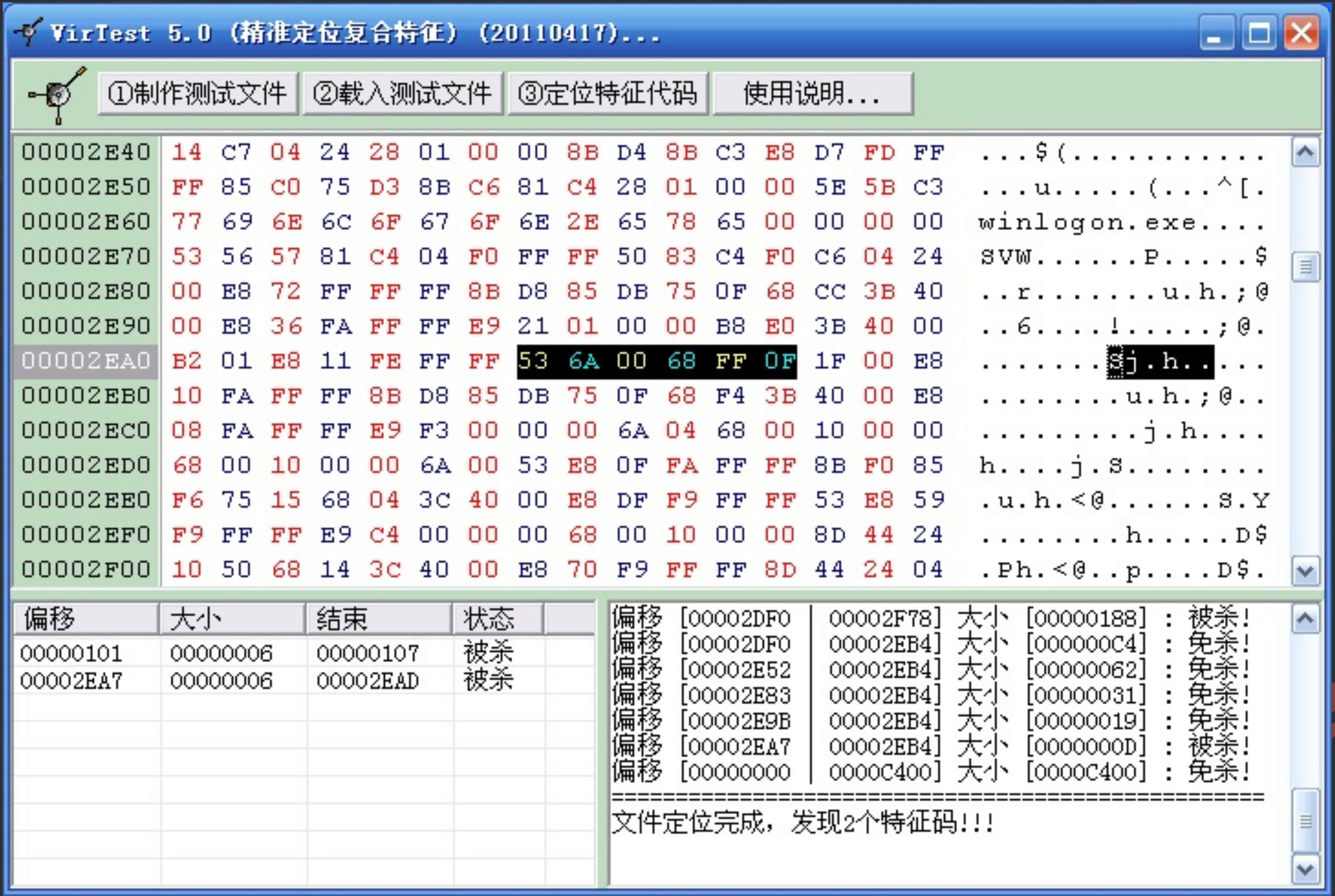
分块个数 10 单位长度 6973(填充 00 特征区间

开始位置 E0 分段长度 AA720 正向 <生成>

结束位置 AA7FF ● 复合定位 ○ 单一定位 二次处理

```
(*)正在生成....完成!
----------------------------------------------------
              PE文件 复合特征码定位

文件名:D:\学习资料\ISCC\WEB渗透\木马免杀实验\Sx_server.exe
----------------------------------------------------
(*)正在载入文件....
(*)正在生成.....完成!
----------------------------------------------------
              PE文件 复合特征码定位

文件名:D:\学习资料\ISCC\WEB渗透\木马免杀实验\Sx_server.exe
----------------------------------------------------
(*)正在载入文件....
(*)正在生成.....完成!
```

#免殺



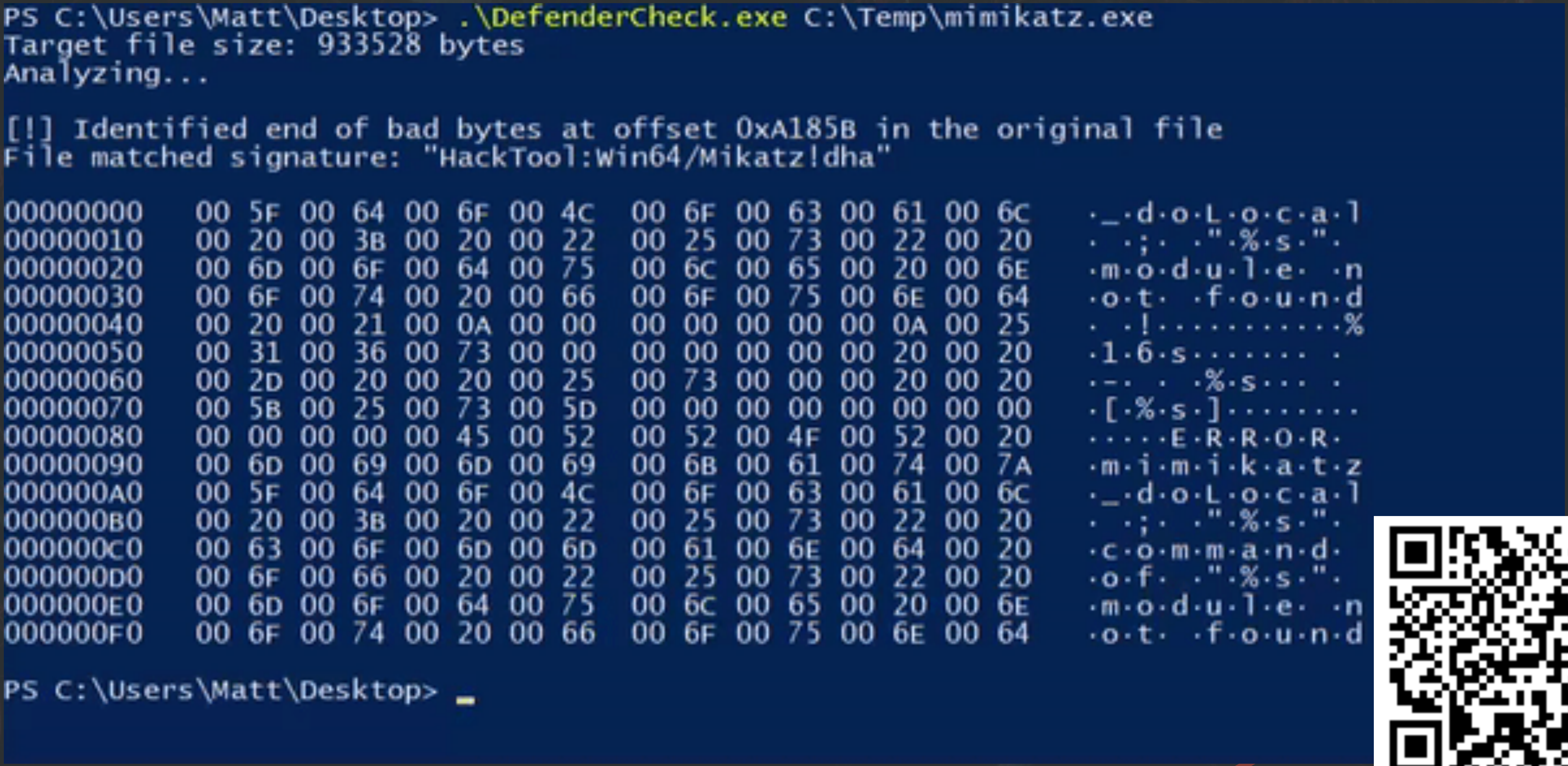
```
00002E40  14 C7 04 24 28 01 00 00 8B D4 8B C3 E8 D7 FD FF   ...$(...........
00002E50  FF 85 C0 75 D3 8B C6 81 C4 28 01 00 00 5E 5B C3   ...u.....(...^[.
00002E60  77 69 6E 6C 6F 67 6F 6E 2E 65 78 65 00 00 00 00   winlogon.exe....
00002E70  53 56 57 81 C4 04 F0 FF FF 50 83 C4 F0 C6 04 24   SVW......P.....$
00002E80  00 E8 72 FF FF FF 8B D8 85 DB 75 0F 68 CC 3B 40   ..r.......u.h.;@
00002E90  00 E8 36 FA FF FF E9 21 01 00 00 B8 E0 3B 40 00   ..6....!.....;@.
00002EA0  B2 01 E8 11 FE FF FF 53 6A 00 68 FF 0F 1F 00 E8   .......Sj.h.....
00002EB0  10 FA FF FF 8B D8 85 DB 75 0F 68 F4 3B 40 00 E8   ........u.h.;@..
00002EC0  08 FA FF FF E9 F3 00 00 00 6A 04 68 00 10 00 00   .........j.h....
00002ED0  68 00 10 00 00 6A 00 53 E8 0F FA FF FF 8B F0 85   h....j.S........
00002EE0  F6 75 15 68 04 3C 40 00 E8 DF F9 FF FF 53 E8 59   .u.h.<@......S.Y
00002EF0  F9 FF FF E9 C4 00 00 00 68 00 10 00 00 8D 44 24   ........h.....D$
00002F00  10 50 68 14 3C 40 00 E8 70 F9 FF FF 8D 44 24 04   .Ph.<@..p....D$.
```

偏移	大小	结束	状态
00000101	00000006	00000107	被杀
00002EA7	00000006	00002EAD	被杀

```
偏移 [00002DF0 | 00002F78] 大小 [00000188] : 被杀!
偏移 [00002DF0 | 00002EB4] 大小 [000000C4] : 免杀!
偏移 [00002E52 | 00002EB4] 大小 [00000062] : 免杀!
偏移 [00002E83 | 00002EB4] 大小 [00000031] : 免杀!
偏移 [00002E9B | 00002EB4] 大小 [00000019] : 免杀!
偏移 [00002EA7 | 00002EB4] 大小 [0000000D] : 被杀!
偏移 [00000000 | 0000C400] 大小 [0000C400] : 免杀!
==========================================================
文件定位完成，发现2个特征码!!!
```

aaaddress1@chroot.org

#AMSI



```
PS C:\Users\Matt\Desktop> .\DefenderCheck.exe C:\Temp\mimikatz.exe
Target file size: 933528 bytes
Analyzing...

[!] Identified end of bad bytes at offset 0xA185B in the original file
File matched signature: "HackTool:Win64/Mikatz!dha"

00000000   00 5F 00 64 00 6F 00 4C  00 6F 00 63 00 61 00 6C   ·_·d·o·L·o·c·a·l
00000010   00 20 00 3B 00 20 00 22  00 25 00 73 00 22 00 20   · ·;· ·"·%·s·"· 
00000020   00 6D 00 6F 00 64 00 75  00 6C 00 65 00 20 00 6E   ·m·o·d·u·l·e· ·n
00000030   00 6F 00 74 00 20 00 66  00 6F 00 75 00 6E 00 64   ·o·t· ·f·o·u·n·d
00000040   00 20 00 21 00 0A 00 00  00 00 00 00 00 0A 00 25   · ·!···········%
00000050   00 31 00 36 00 73 00 00  00 00 00 00 00 20 00 20   ·1·6·s······· · 
00000060   00 2D 00 20 00 20 00 25  00 73 00 00 00 20 00 20   ·-· · ·%·s··· · 
00000070   00 5B 00 25 00 73 00 5D  00 00 00 00 00 00 00 00   ·[·%·s·]········
00000080   00 00 00 00 00 45 00 52  00 52 00 4F 00 52 00 20   ·····E·R·R·O·R· 
00000090   00 6D 00 69 00 6D 00 69  00 6B 00 61 00 74 00 7A   ·m·i·m·i·k·a·t·z
000000A0   00 5F 00 64 00 6F 00 4C  00 6F 00 63 00 61 00 6C   ·_·d·o·L·o·c·a·l
000000B0   00 20 00 3B 00 20 00 22  00 25 00 73 00 22 00 20   · ·;· ·"·%·s·"· 
000000C0   00 63 00 6F 00 6D 00 6D  00 61 00 6E 00 64 00 20   ·c·o·m·m·a·n·d· 
000000D0   00 6F 00 66 00 20 00 22  00 25 00 73 00 22 00 20   ·o·f· ·"·%·s·"· 
000000E0   00 6D 00 6F 00 64 00 75  00 6C 00 65 00 20 00 6E   ·m·o·d·u·l·e· ·n
000000F0   00 6F 00 74 00 20 00 66  00 6F 00 75 00 6E 00 64   ·o·t· ·f·o·u·n·d

PS C:\Users\Matt\Desktop> _
```

/?challenge

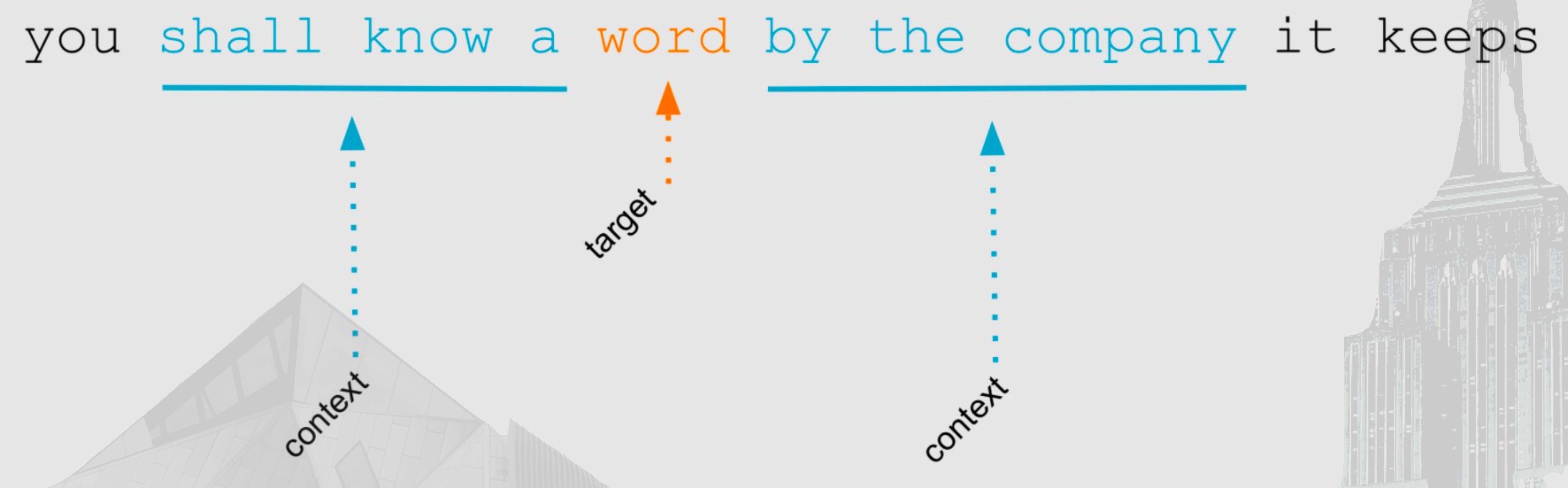
- Active Protection System
  - rule-based, not strong enough against unkown attacks
- Malware Pattern based on Reversing
  - lack of lexical semantic of assembly → false positive
  - too slow against variability malware
- Known Challenges
  - compiler optimization
  - Mirai, Hakai, Yowai, SpeakUp
  - Anti-AntiVirus Techniques
- Word Embedding Techniques (NLP)
  - use only few samples to predict income binary files
  - learn lexical semantic from instruction sequences

aaaddress1@chroot.org

Semantics

/?semantics

“You shall know a word by the company it keeps“
(Firth, J. R. 1957:11)

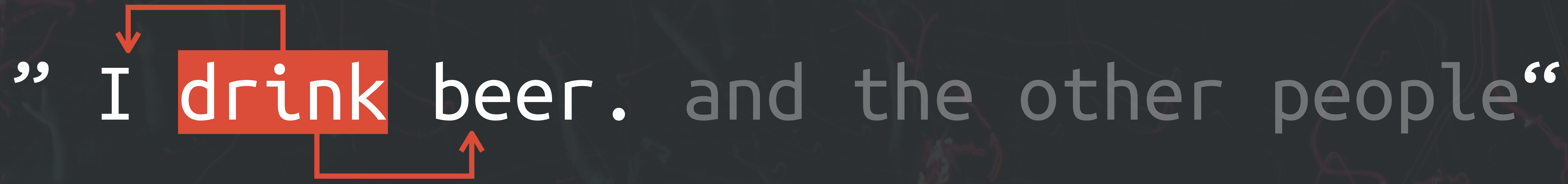


you shall know a word by the company it keeps

/?semantics

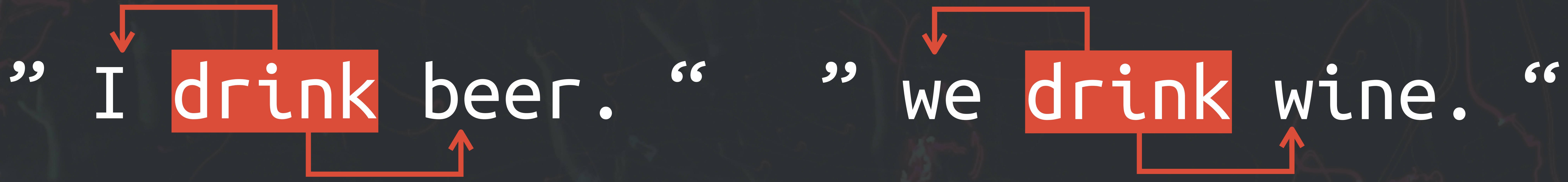
> “... I can show you the world. Shining,
> shimmering, splendid. Tell me, princess,
> now when did. You last let your heart
> decide? I can open your eyes, Take you
> wonder by wonder ...”

/?semantics



” I drink beer. and the other people“

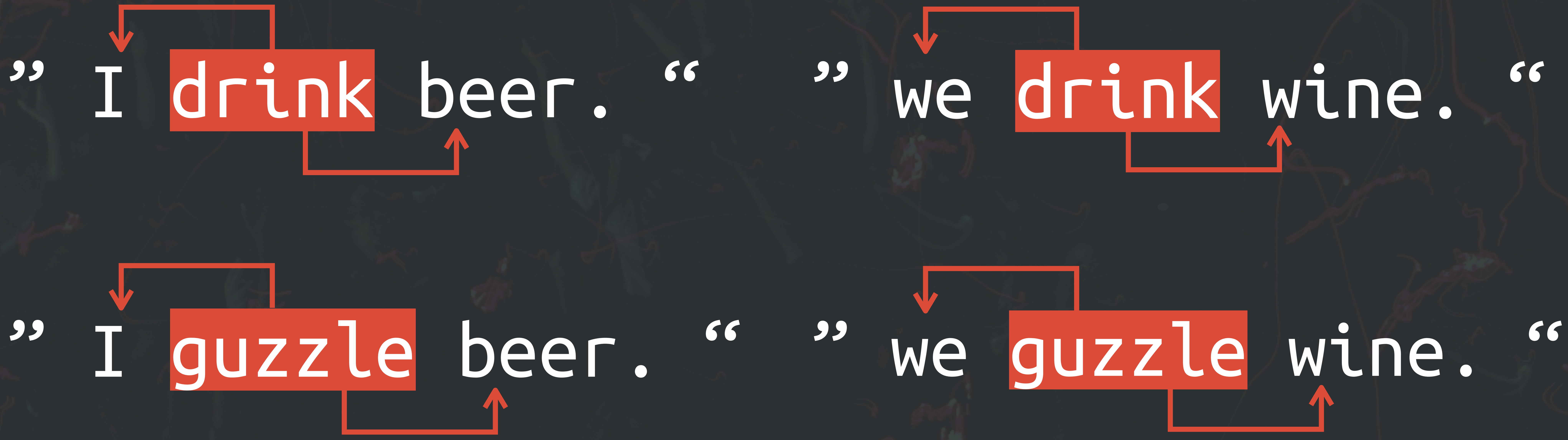
/?semantics



” I drink beer. “

” we drink wine. “

/?semantics



” I drink beer. “

” we drink wine. “

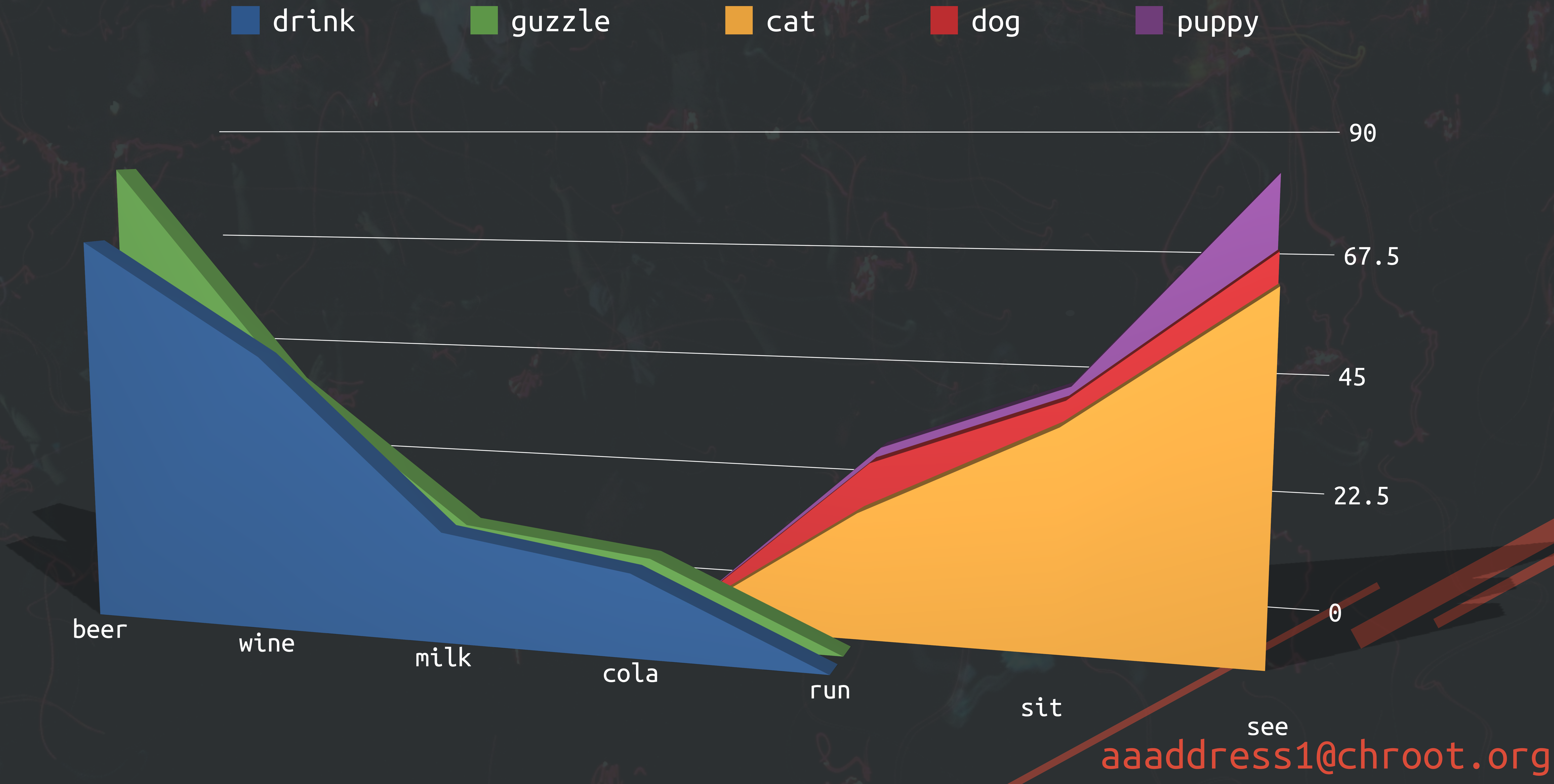
” I guzzle beer. “

” we guzzle wine. “

/?tokenFreq

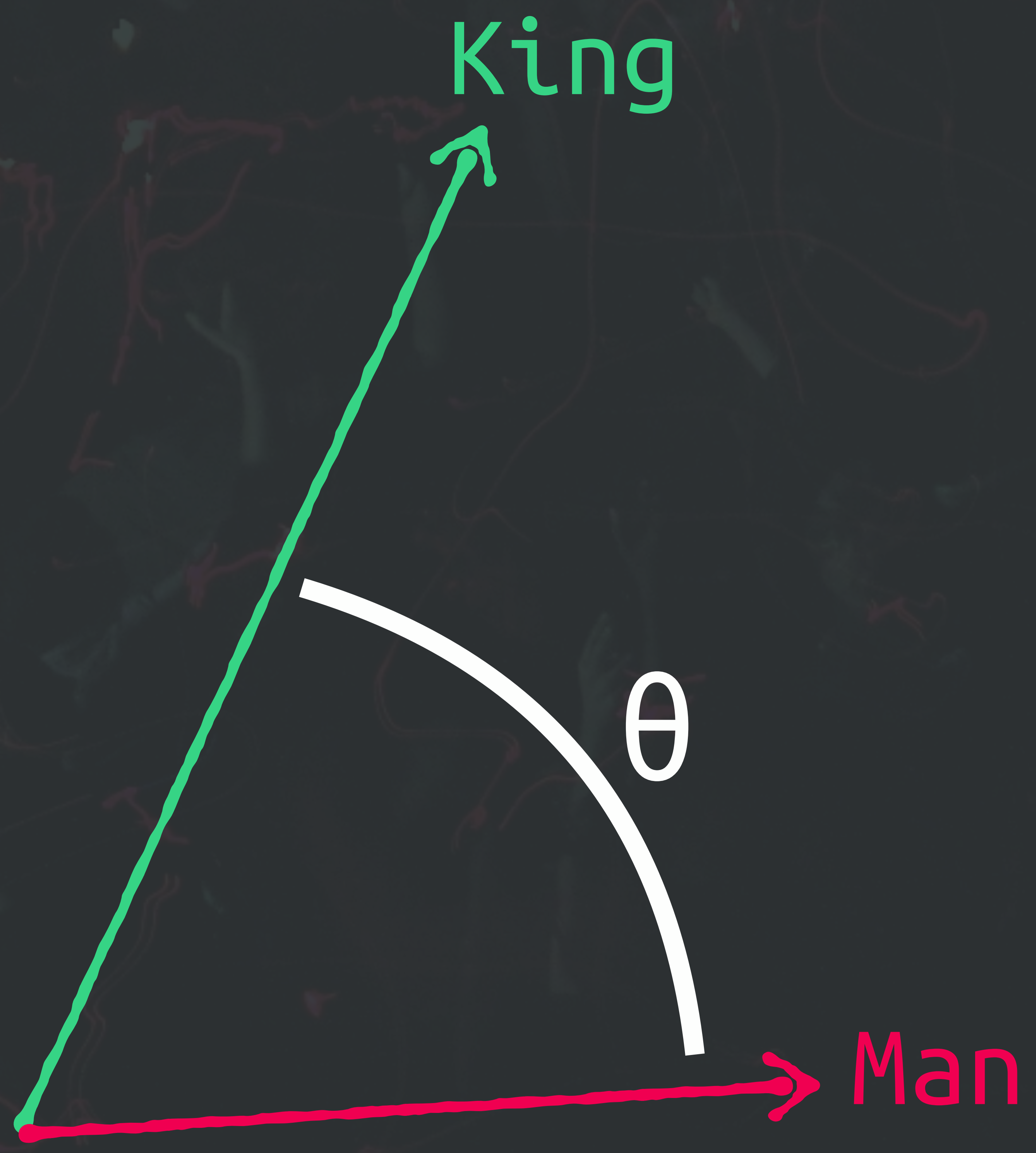
		beer	wine	milk	cola	run	sit	see
🟦	drink	70	50	20	15	0	0	0
🟩	guzzle	83	44	19	15	0	0	0
🟧	cat	0	0	0	0	23	40	65
🟥	dog	0	0	0	0	30	43	70
🟪	puppy	0	0	0	0	31	44	83

/?freq



aaaddress1@chroot.org

/?cos(θ)



$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum\limits_{i=1}^{n} A_i B_i}{\sqrt{\sum\limits_{i=1}^{n} A_i^2} \sqrt{\sum\limits_{i=1}^{n} B_i^2}}$$

#semantics

- Co-Occurrence Matrix
  - count based, token frequency
  - able to capture lexical semantic
  - Cosine Similarity
- Issues
  - vocabulary
  - online training

→ Paragraph Vector Distributed Memory (PV-DM)

Word2Vec

/?tokenFreq

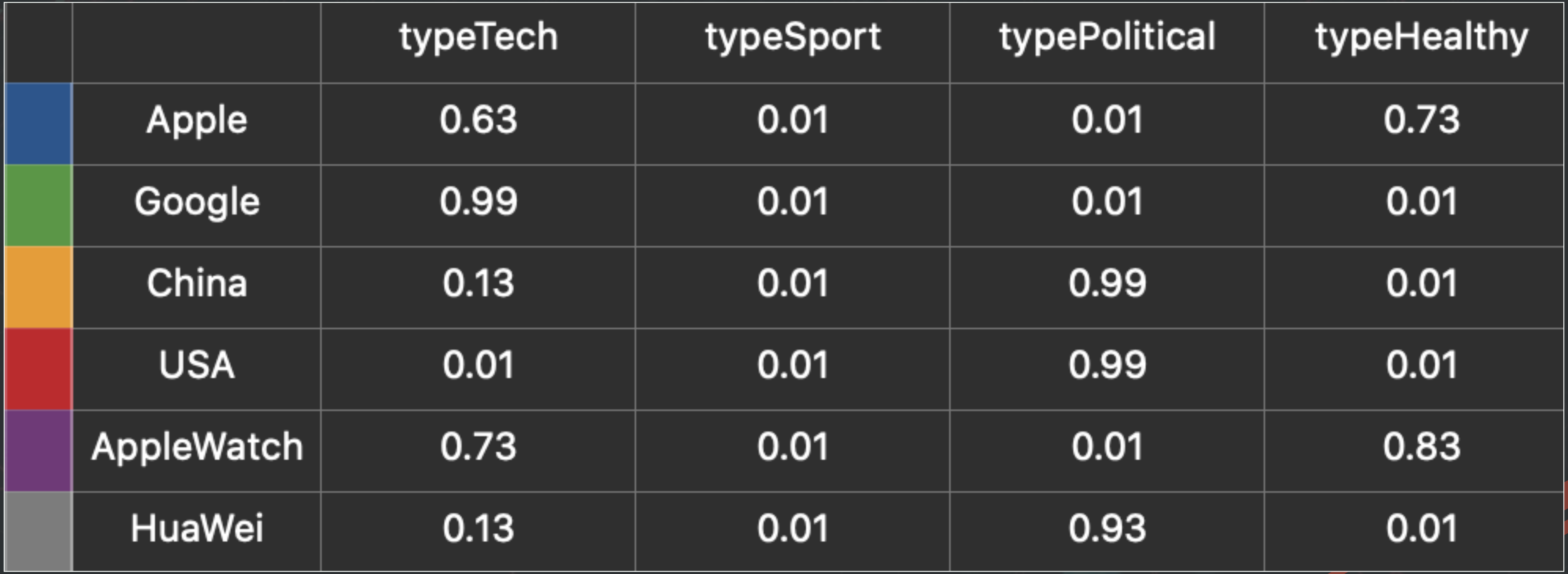
drink

		beer	wine	milk	cola	run	sit	see
🟦	drink	70	50	20	15	0	0	0
🟩	guzzle	83	44	19	15	0	0	0
🟧	cat	0	0	0	0	23	40	65
🟥	dog	0	0	0	0	30	43	70
🟪	puppy	0	0	0	0	31	44	83

behavior

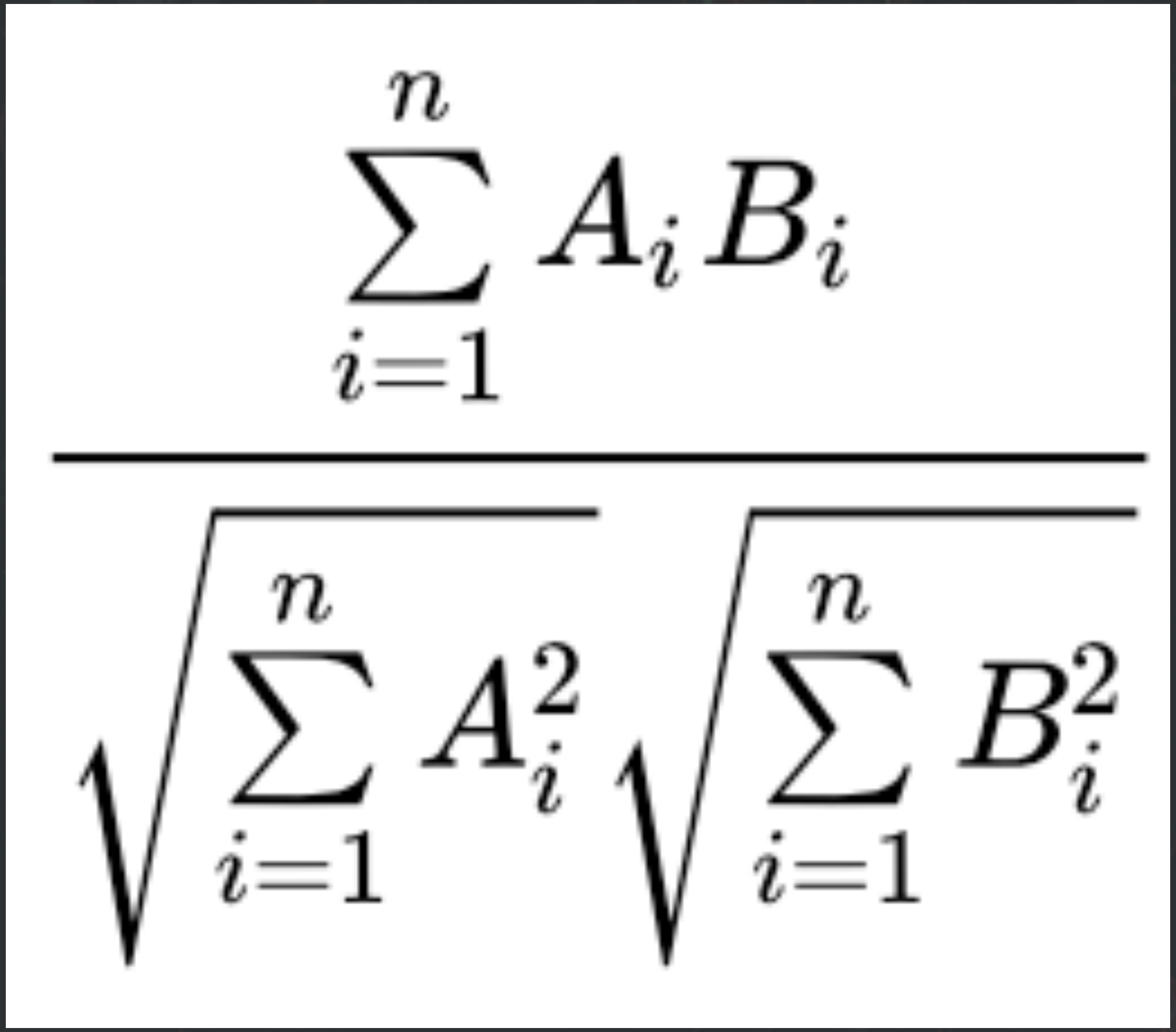
/?tokenFreq

4 dim



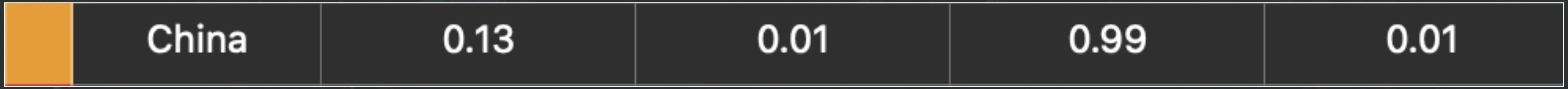
		typeTech	typeSport	typePolitical	typeHealthy
🟦	Apple	0.63	0.01	0.01	0.73
🟩	Google	0.99	0.01	0.01	0.01
🟧	China	0.13	0.01	0.99	0.01
🟥	USA	0.01	0.01	0.99	0.01
🟪	AppleWatch	0.73	0.01	0.01	0.83
⬜	HuaWei	0.13	0.01	0.93	0.01

#Sim



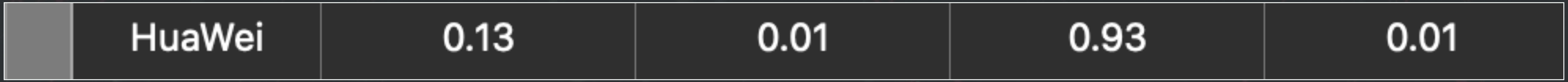
$$\frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

#Sim



🟧	China	0.13	0.01	0.99	0.01

similar()

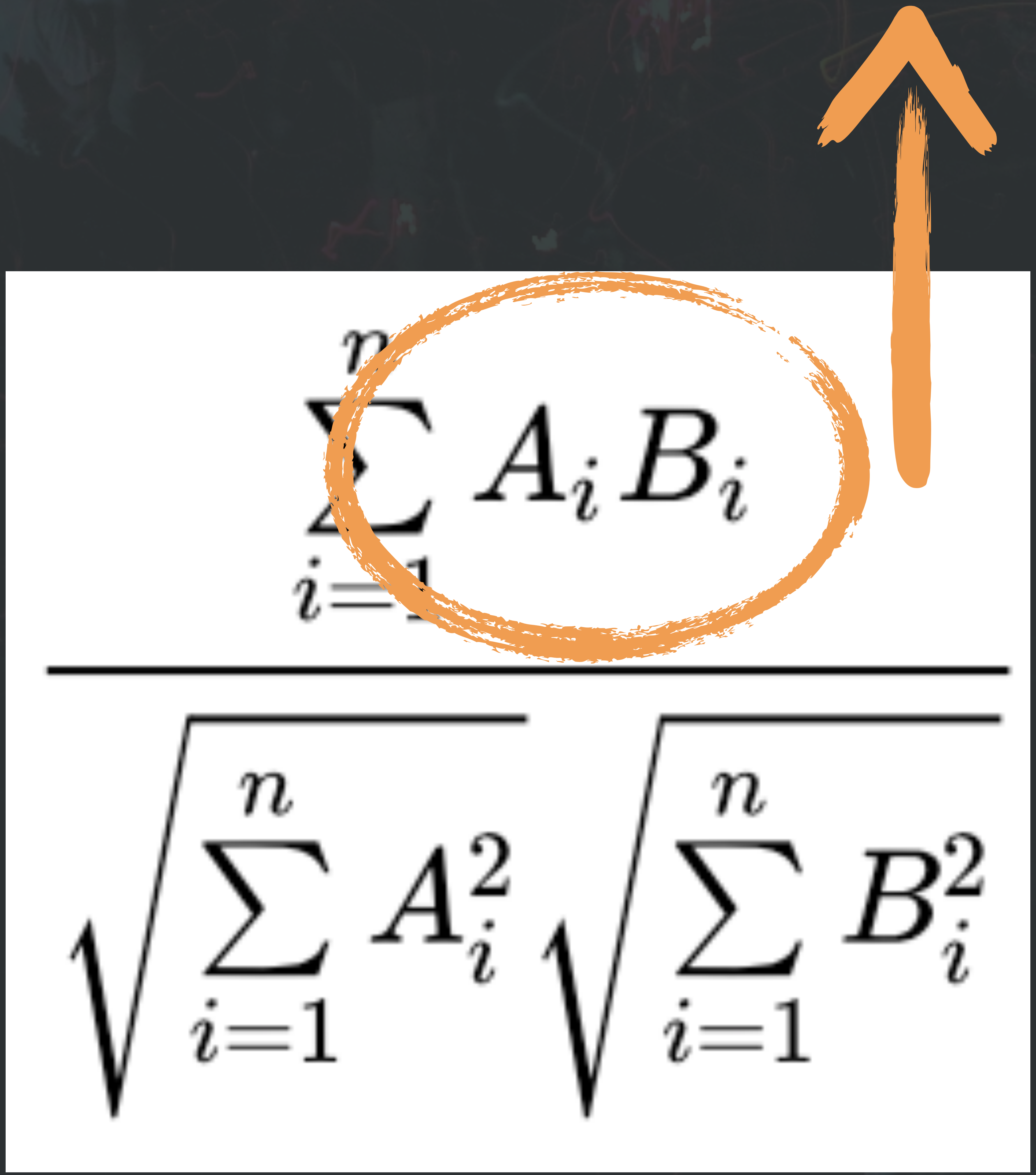


⬜	HuaWei	0.13	0.01	0.93	0.01

$$= \frac{0.13*0.13 + 0.01*0.01 + 0.99*0.93 + 0.01*0.01}{\text{sqrt}(0.13\hat{}2 + 0.01\hat{}2 + 0.99\hat{}2 + 0.01\hat{}2) \times \text{sqrt}(0.13\hat{}2 + 0.01\hat{}2 + 0.93\hat{}2 + 0.01\hat{}2)}$$

$$= 0.9999650034397828$$

#Sim

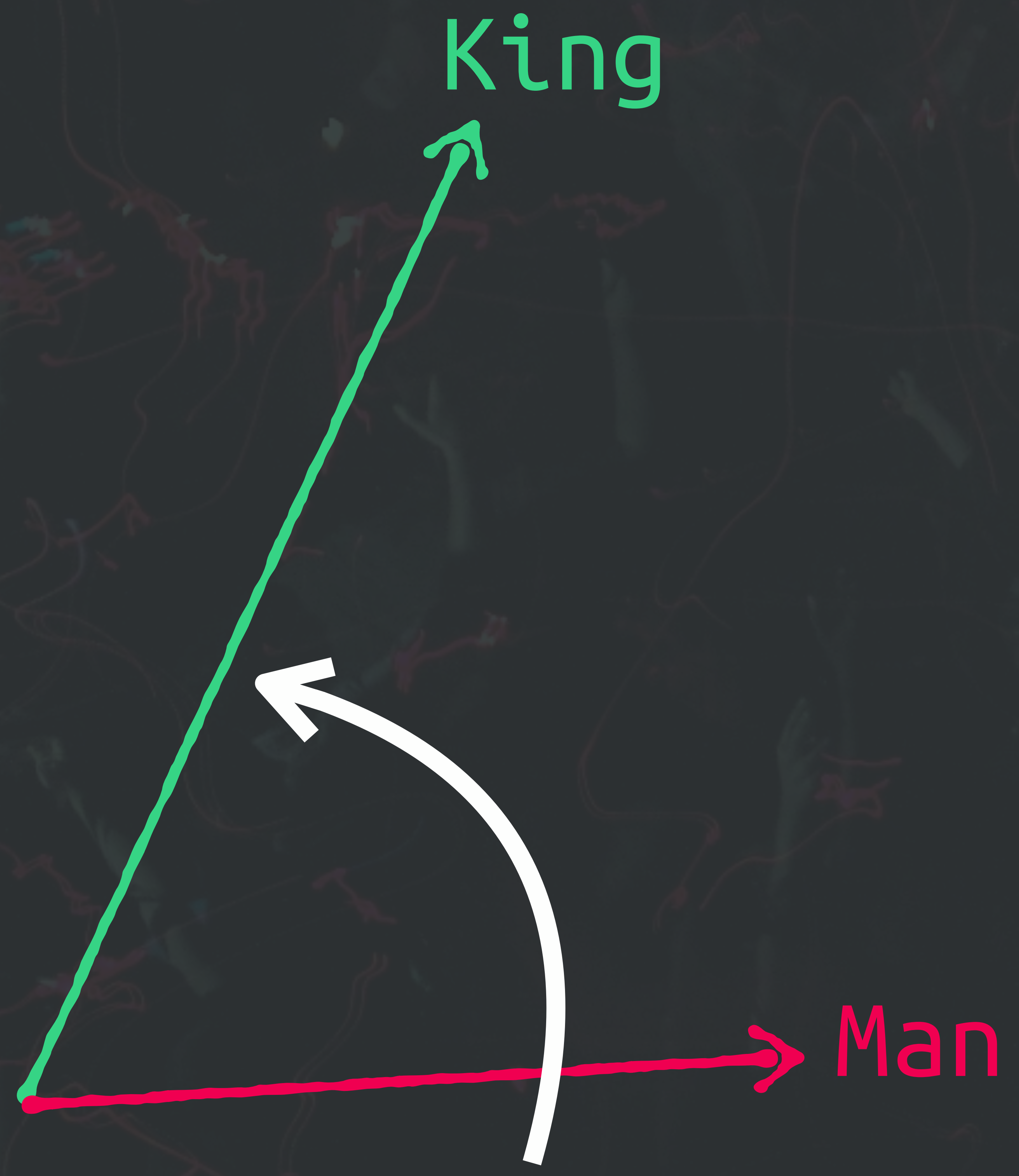


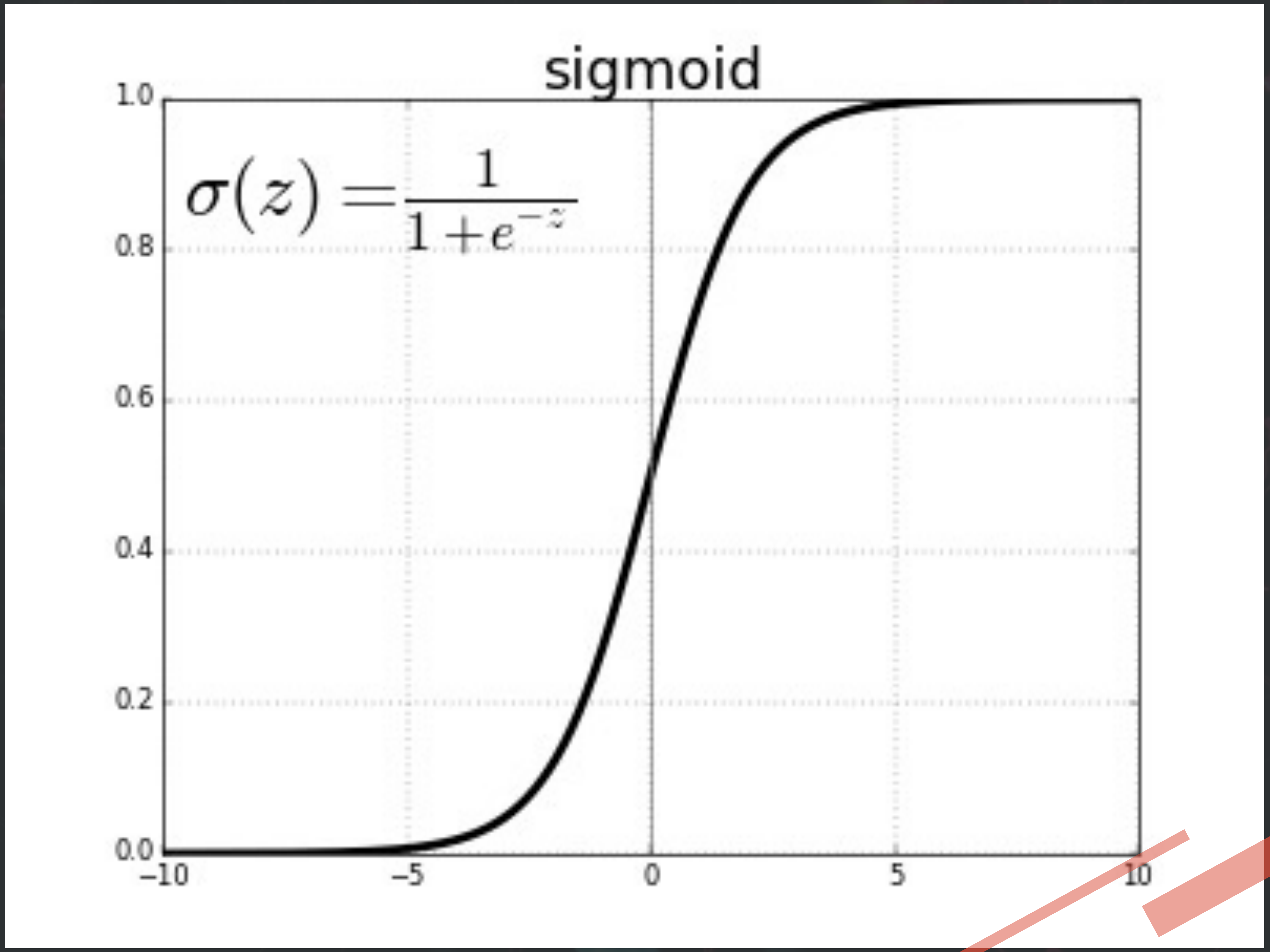
$$\frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$

more similar

#Sim

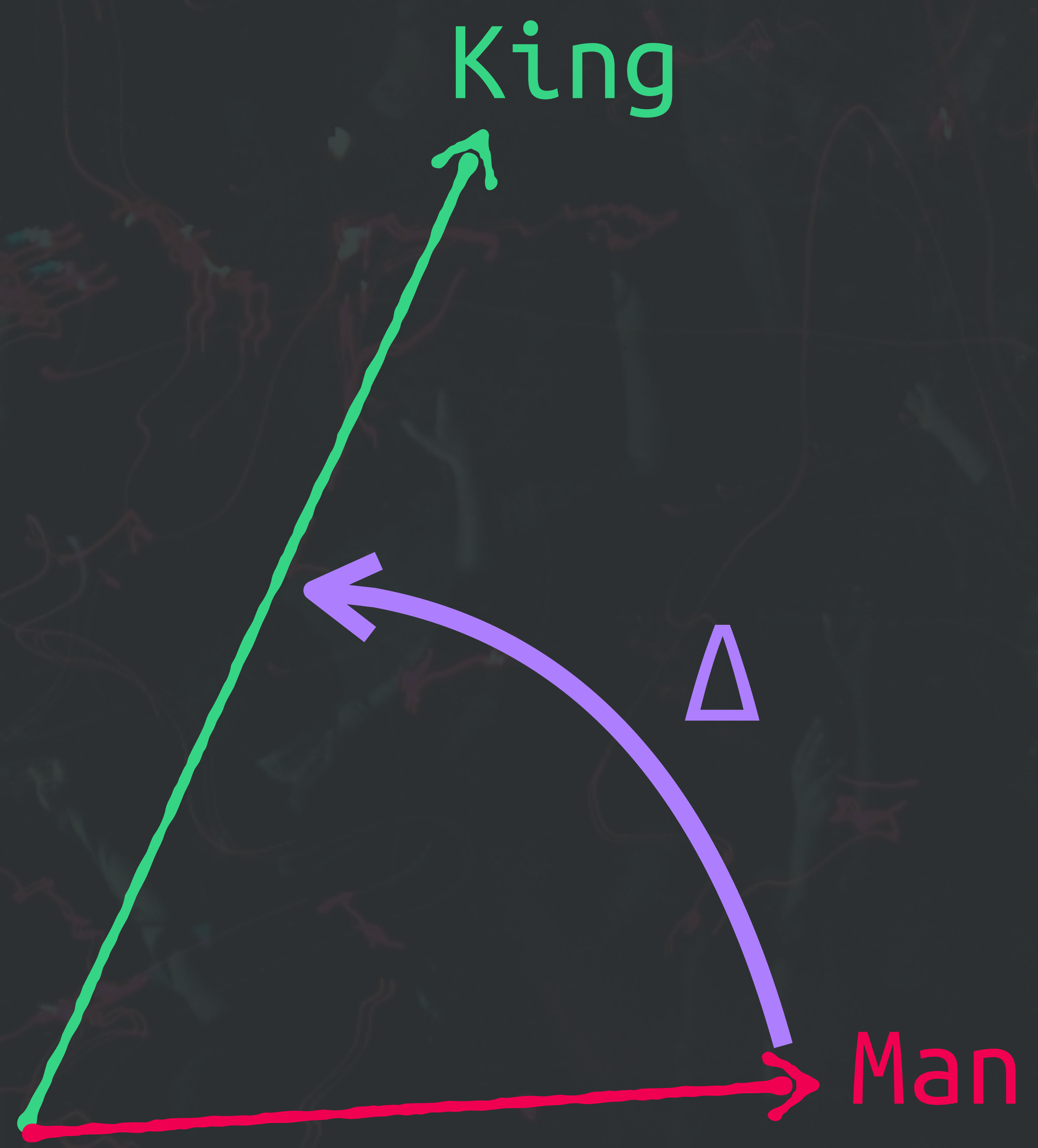
sim(King - Man) ≒ sigmoid(King · Man)





aaaddress1@chroot.org

#Sim

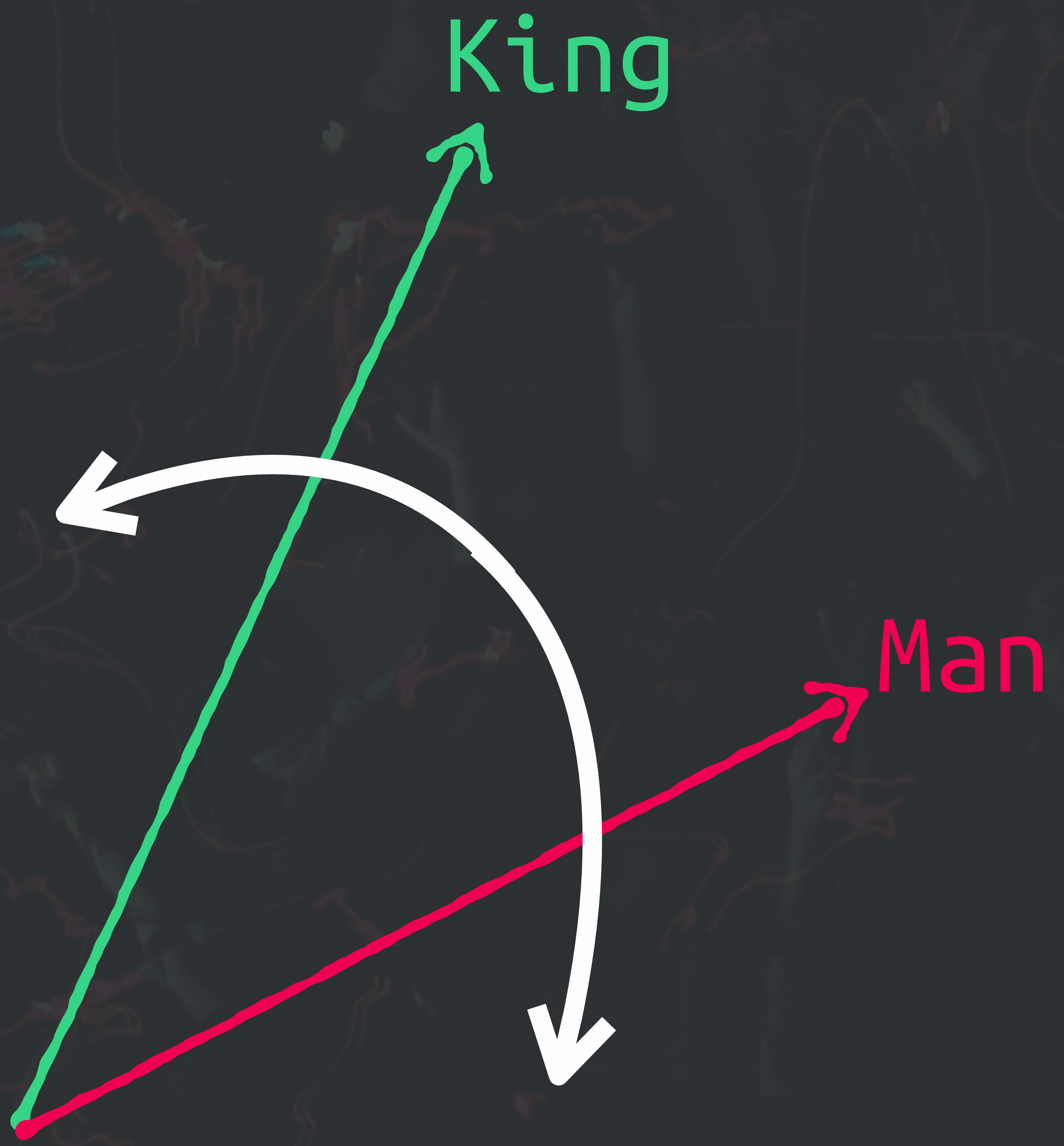


sim(King - Man) ≒ sigmoid(King · Man)

Δ(King - Man) = (1 - sim(King - Man)) · King

[BACKWARD]: Man = Man - Δ(King - Man) * learningRate

#negative



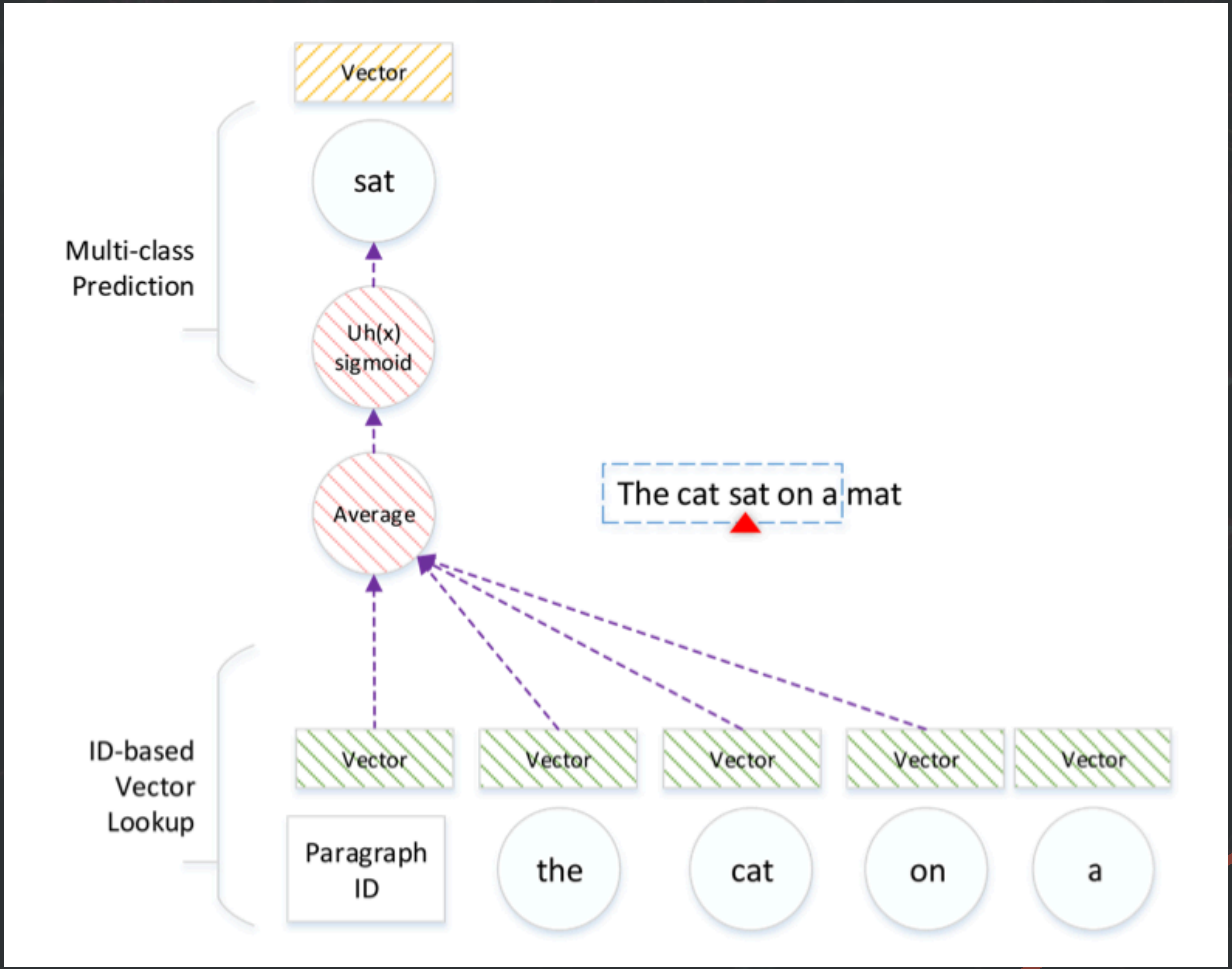
sim(King - Man) ≒ sigmoid(King · Man)

Δ(King - Man) = sim(King - Man) · King

[BACKWARD]: Man = Man - Δ(King - Man) * learningRate

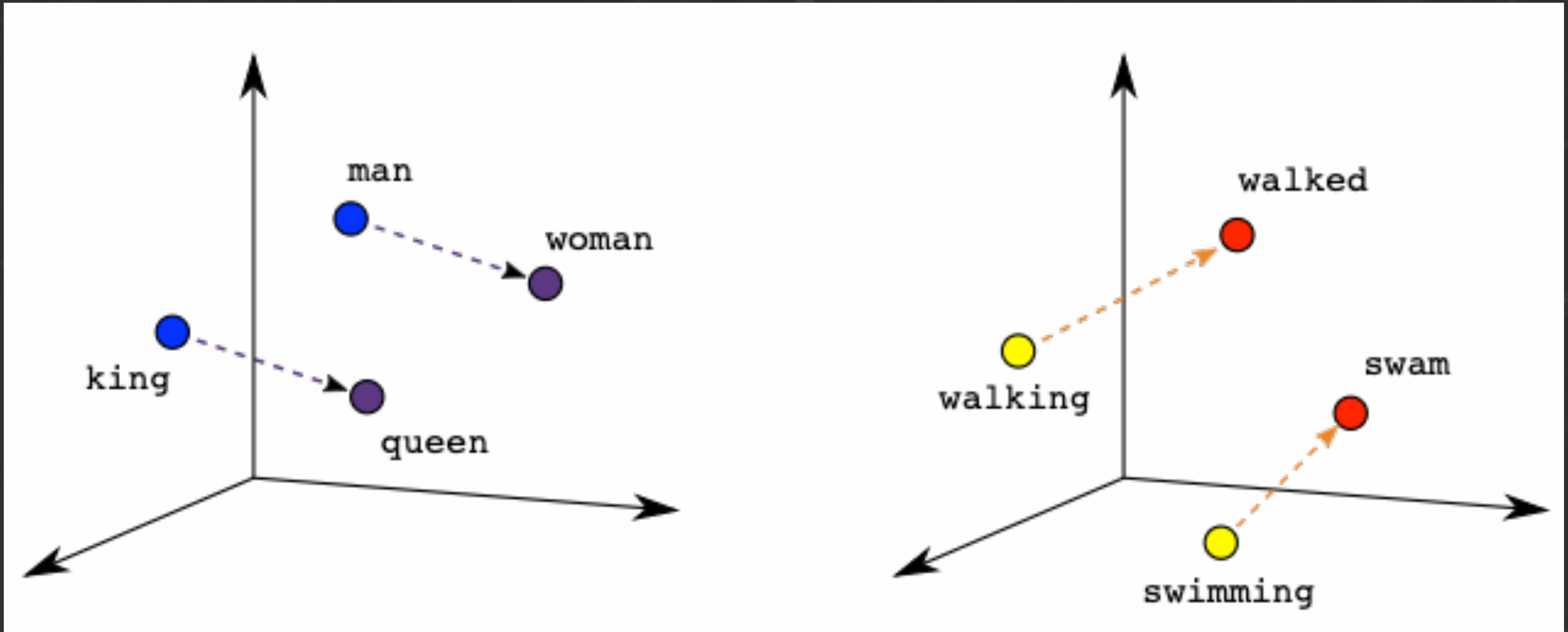
aaaddress1@chroot.org

#PV-DM



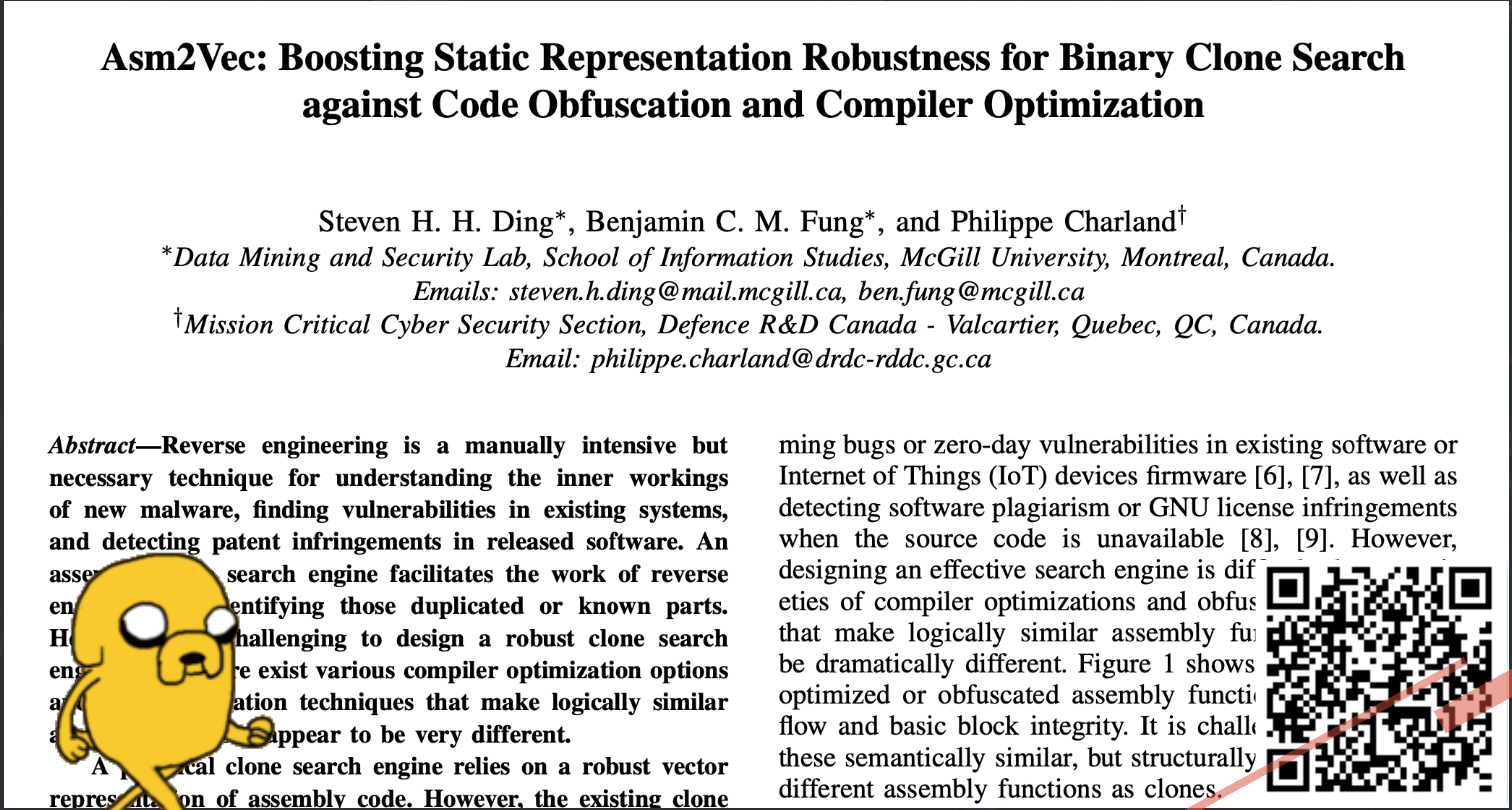
aaaddress1@chroot.org

#Word2Vec



Asm2Vec

#Asm2Vec



Asm2Vec: Boosting Static Representation Robustness for Binary Clone Search against Code Obfuscation and Compiler Optimization

Steven H. H. Ding*, Benjamin C. M. Fung*, and Philippe Charland†

*Data Mining and Security Lab, School of Information Studies, McGill University, Montreal, Canada.
Emails: steven.h.ding@mail.mcgill.ca, ben.fung@mcgill.ca
†Mission Critical Cyber Security Section, Defence R&D Canada - Valcartier, Quebec, QC, Canada.
Email: philippe.charland@drdc-rddc.gc.ca

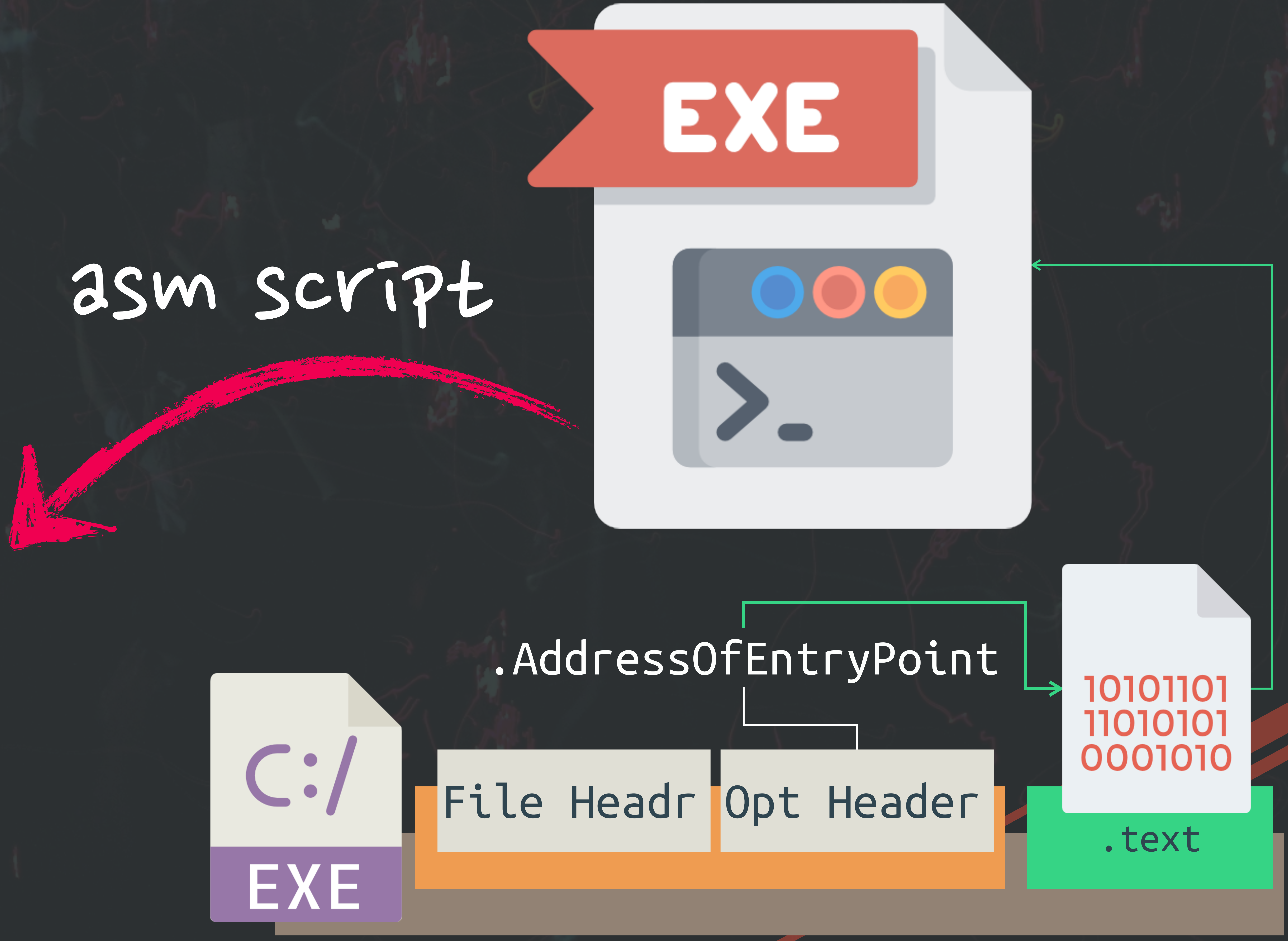
***Abstract*—Reverse engineering is a manually intensive but necessary technique for understanding the inner workings of new malware, finding vulnerabilities in existing systems, and detecting patent infringements in released software. An asse search engine facilitates the work of reverse en entifying those duplicated or known parts. H hallenging to design a robust clone search eng re exist various compiler optimization options a ation techniques that make logically similar a ppear to be very different.**

A p cal clone search engine relies on a robust vector repres on of assembly code. However, the existing clone

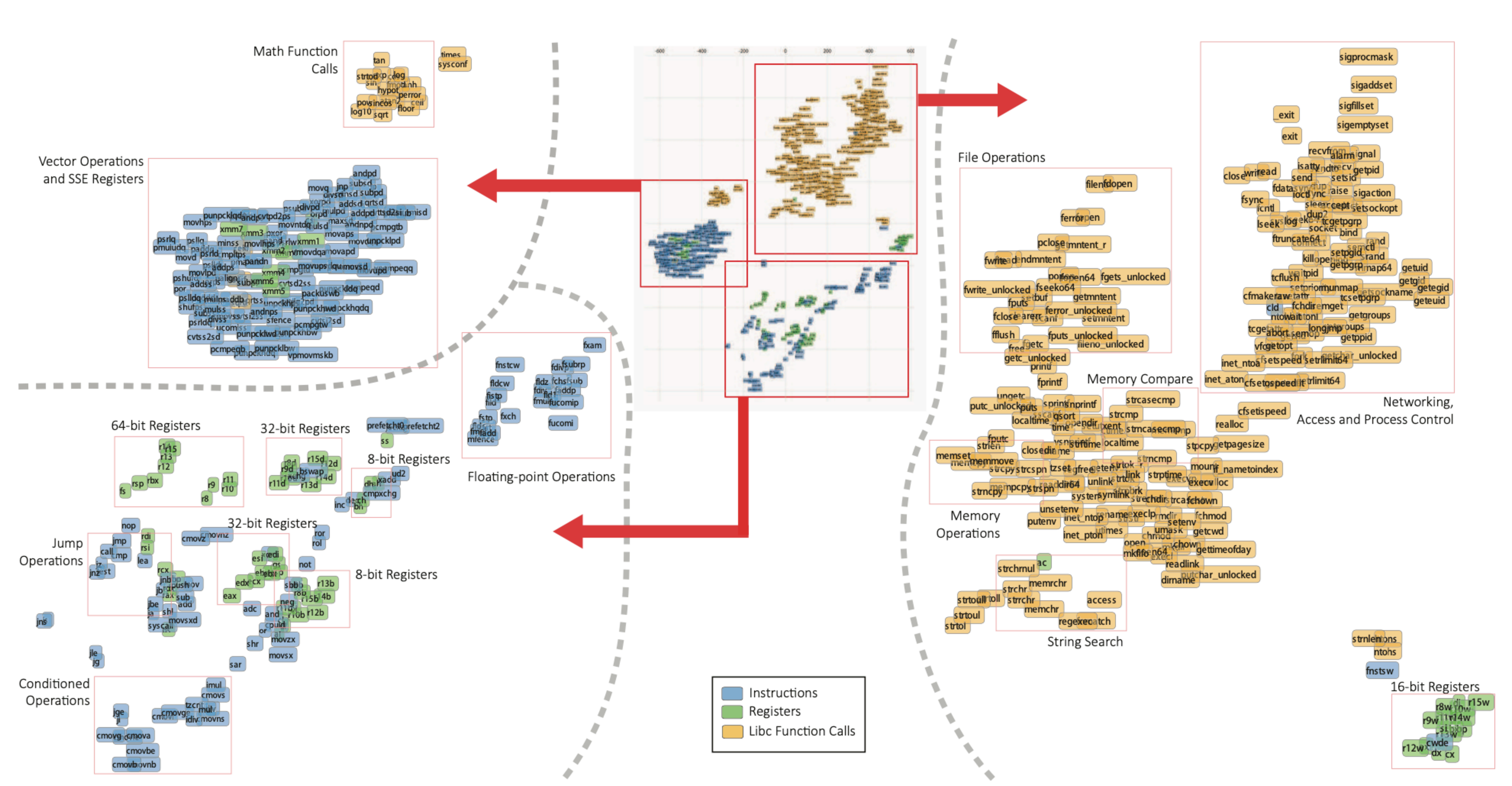
ming bugs or zero-day vulnerabilities in existing software or Internet of Things (IoT) devices firmware [6], [7], as well as detecting software plagiarism or GNU license infringements when the source code is unavailable [8], [9]. However, designing an effective search engine is dif eties of compiler optimizations and obfus that make logically similar assembly fu be dramatically different. Figure 1 shows optimized or obfuscated assembly functi flow and basic block integrity. It is chall these semantically similar, but structurally different assembly functions as clones.

#paragraph

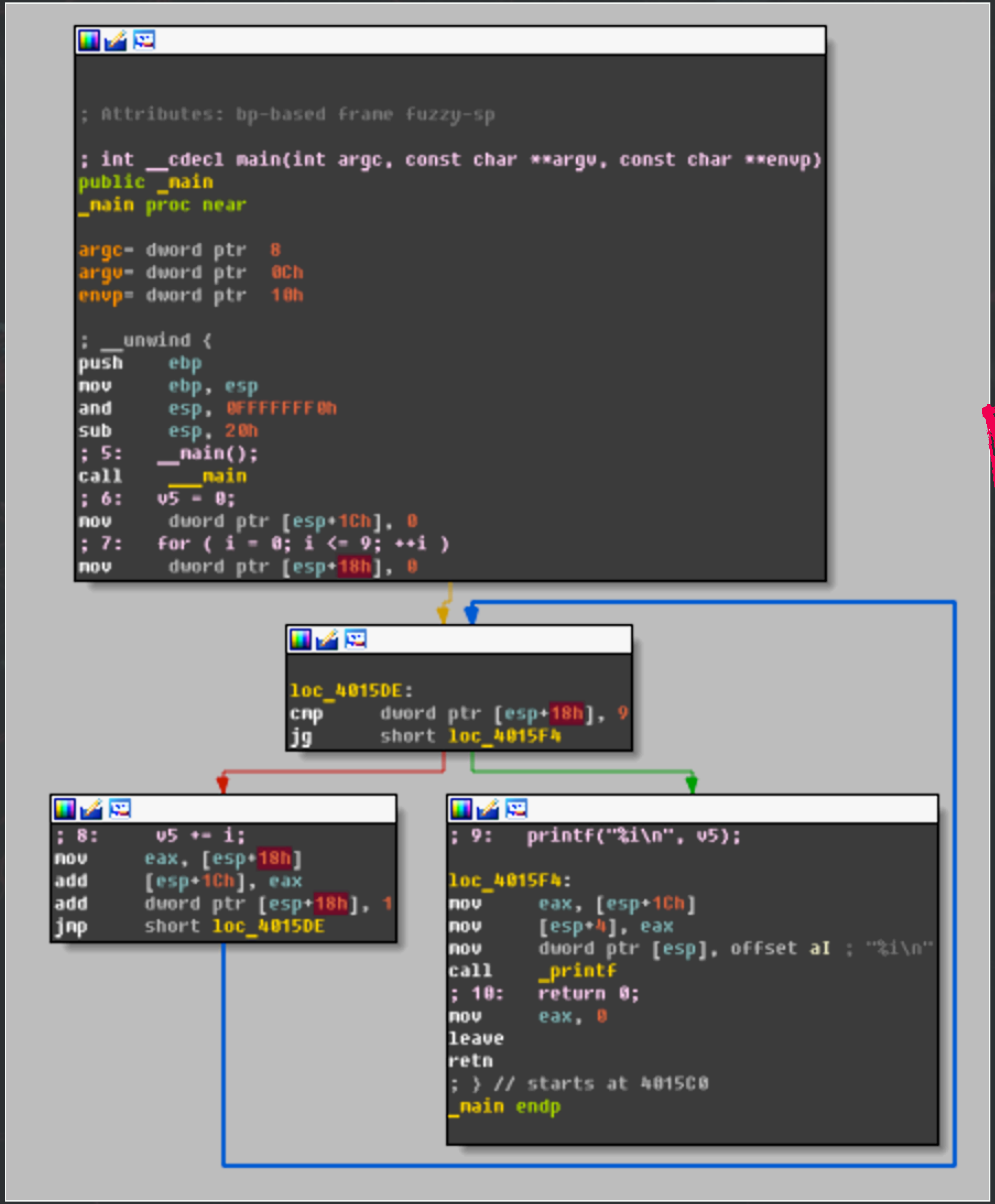
```
mov [ebp-0x04], 00
jmp block_c
cmp [ebp-0x04], Ah
jg Exit
push 0x3E8
call Sleep
jmp block_b
mov eax, [ebp-0x04]
add eax, 1
mov [ebp-0x04], eax
cmp [ebp-0x04], Ah
jg Exit
push 0x3E8
call Sleep
jmp block_b
...
```



#Asm2Vec



Math Function Calls
Vector Operations and SSE Registers
Floating-point Operations
64-bit Registers
32-bit Registers
8-bit Registers
Jump Operations
32-bit Registers
8-bit Registers
Conditioned Operations
File Operations
Memory Compare
Memory Operations
String Search
Networking, Access and Process Control
16-bit Registers
Instructions
Registers
Libc Function Calls

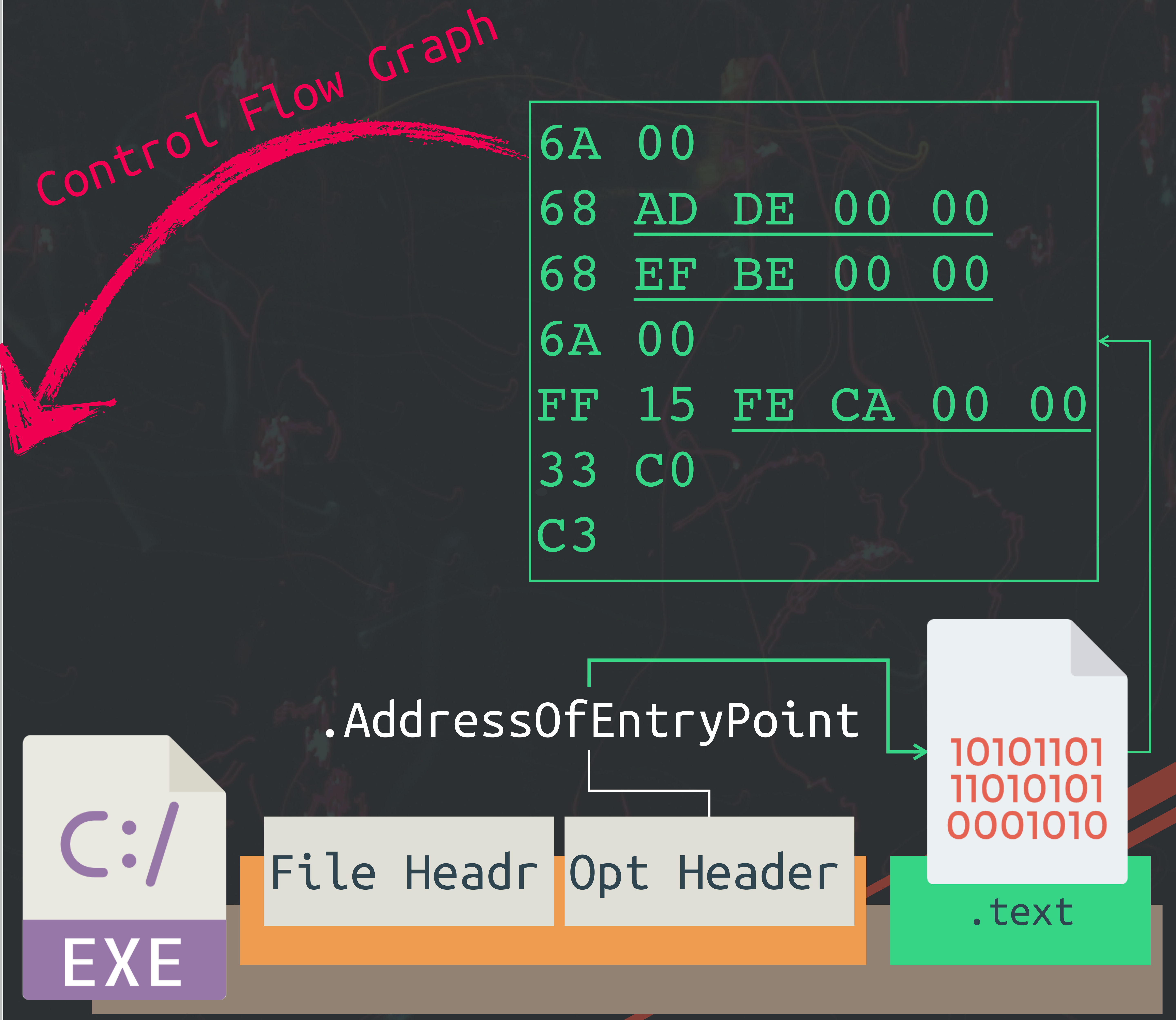


```
; Attributes: bp-based frame fuzzy-sp

; int __cdecl main(int argc, const char **argv, const char **envp)
public _main
_main proc near

argc= dword ptr  8
argv= dword ptr  0Ch
envp= dword ptr  10h

; __unwind {
push    ebp
mov     ebp, esp
and     esp, 0FFFFFFF0h
sub     esp, 20h
; 5:   __main();
call    ___main
; 6:   v5 = 0;
mov     dword ptr [esp+1Ch], 0
; 7:   for ( i = 0; i <= 9; ++i )
mov     dword ptr [esp+18h], 0
```

```
loc_4015DE:
cmp     dword ptr [esp+18h], 9
jg      short loc_4015F4
```

```
; 8:     v5 += i;
mov     eax, [esp+18h]
add     [esp+1Ch], eax
add     dword ptr [esp+18h], 1
jmp     short loc_4015DE
```

```
; 9:   printf("%i\n", v5);

loc_4015F4:
mov     eax, [esp+1Ch]
mov     [esp+4], eax
mov     dword ptr [esp], offset aI ; "%i\n"
call    _printf
; 10:  return 0;
mov     eax, 0
leave
retn
; } // starts at 4015C0
_main endp
```



```
6A 00
68 AD DE 00 00
68 EF BE 00 00
6A 00
FF 15 FE CA 00 00
33 C0
C3
```

aaaddress1@chroot.org

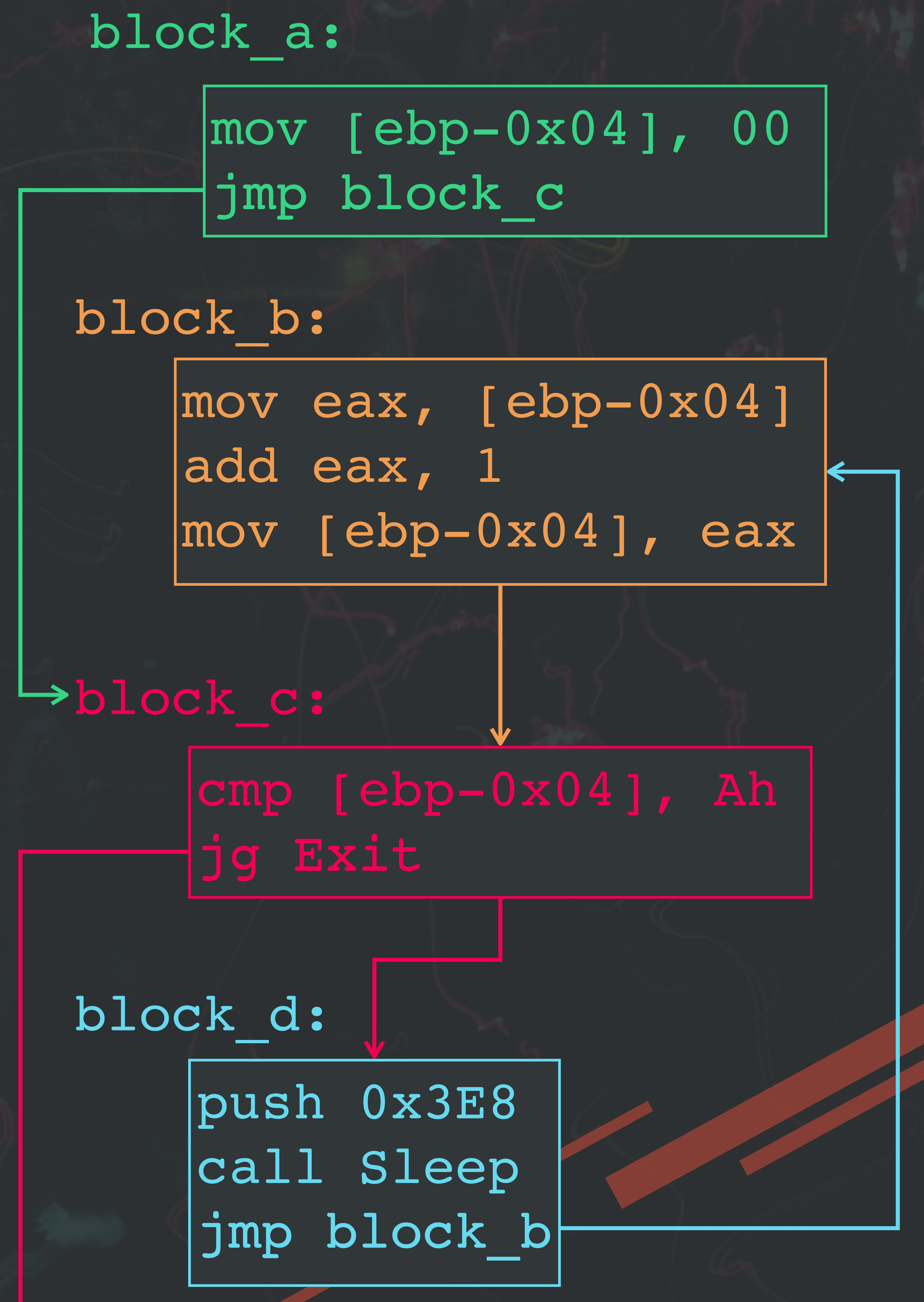
/?rndWalk

#1: block_a → block_c → Exit

#2: block_a → block_c → block_d →
block_b → block_c → Exit

#3: block_a → block_c → block_d →
block_b → block_c → block_d →
block_b → block_c → Exit

#4: block_a → block_c → block_d →
block_b → block_c → block_d →
block_b → block_c → block_d →
block_b → block_c → Exit



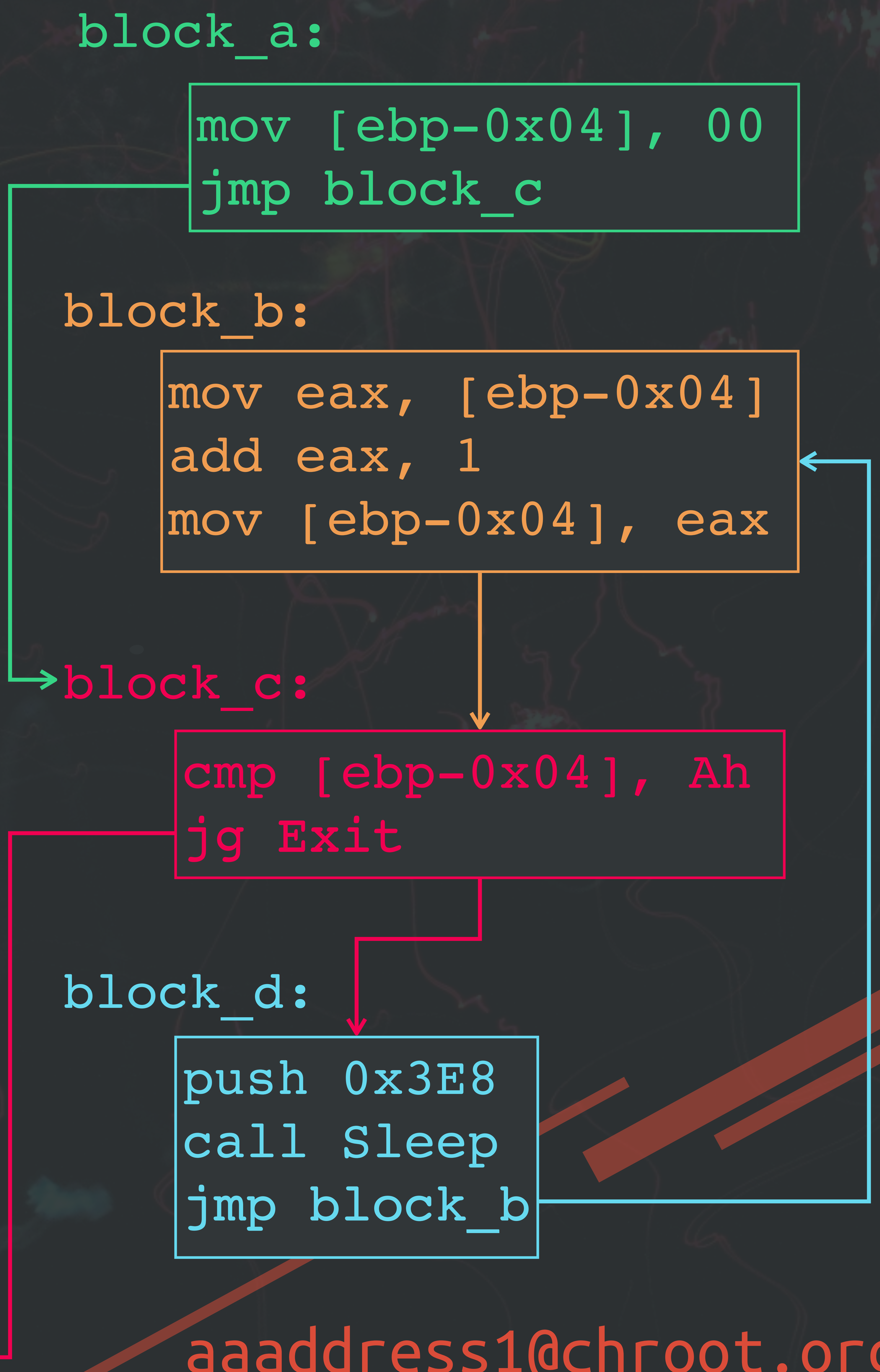
```
block_a:
    mov [ebp-0x04], 00
    jmp block_c

block_b:
    mov eax, [ebp-0x04]
    add eax, 1
    mov [ebp-0x04], eax

block_c:
    cmp [ebp-0x04], Ah
    jg Exit

block_d:
    push 0x3E8
    call Sleep
    jmp block_b
```

/?rndWalk

asm

script

```
mov [ebp-0x04], 00
jmp block_c
cmp [ebp-0x04], Ah
jg Exit
push 0x3E8
call Sleep
jmp block_b
mov eax, [ebp-0x04]
add eax, 1
mov [ebp-0x04], eax
cmp [ebp-0x04], Ah
jg Exit
push 0x3E8
call Sleep
jmp block_b
...
```

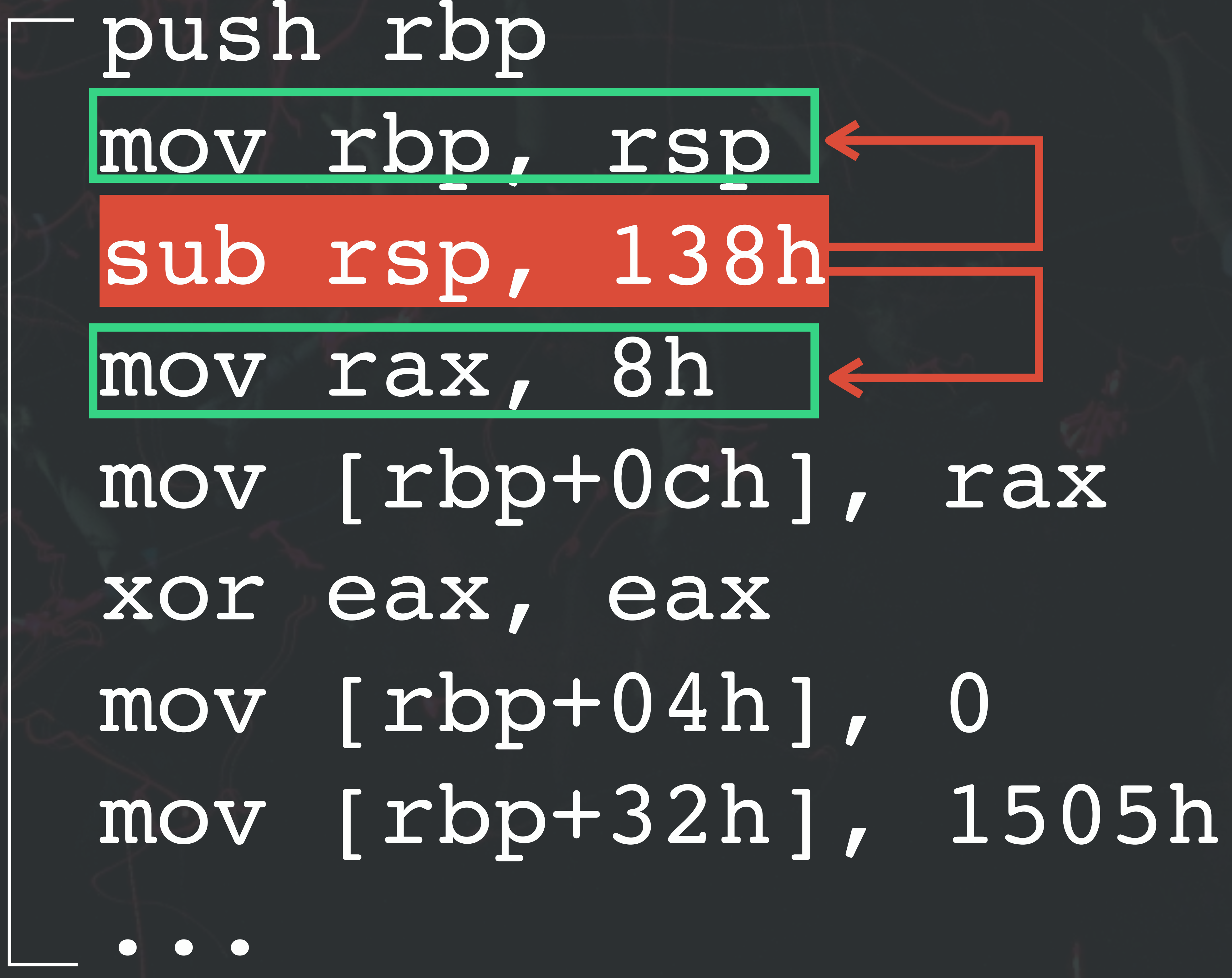


```
block_a:
    mov [ebp-0x04], 00
    jmp block_c

block_b:
    mov eax, [ebp-0x04]
    add eax, 1
    mov [ebp-0x04], eax

block_c:
    cmp [ebp-0x04], Ah
    jg Exit

block_d:
    push 0x3E8
    call Sleep
    jmp block_b
```

#Asm2Vec

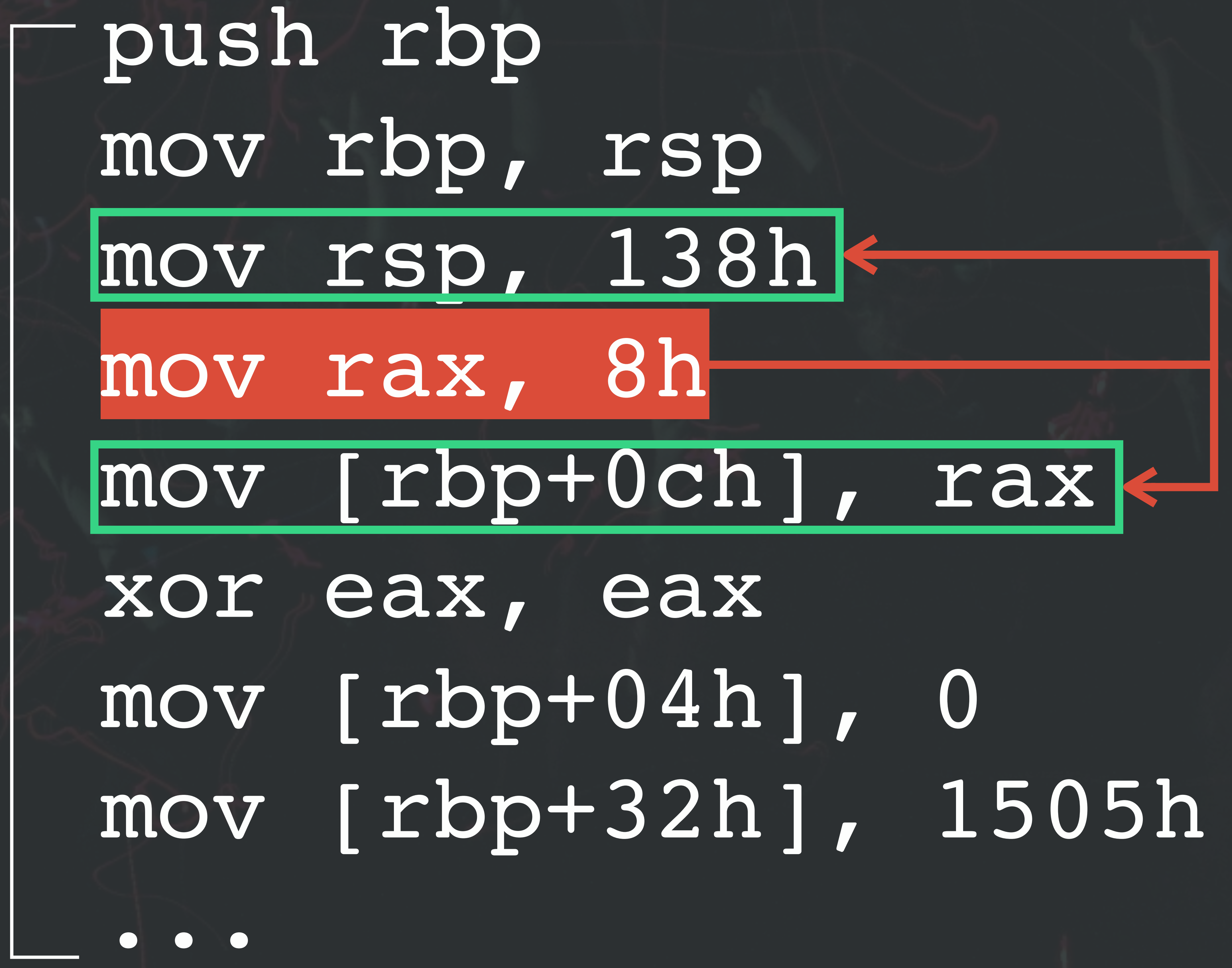
```
push rbp
mov rbp, rsp
sub rsp, 138h
mov rax, 8h
mov [rbp+0ch], rax
xor eax, eax
mov [rbp+04h], 0
mov [rbp+32h], 1505h
...
```

#Asm2Vec



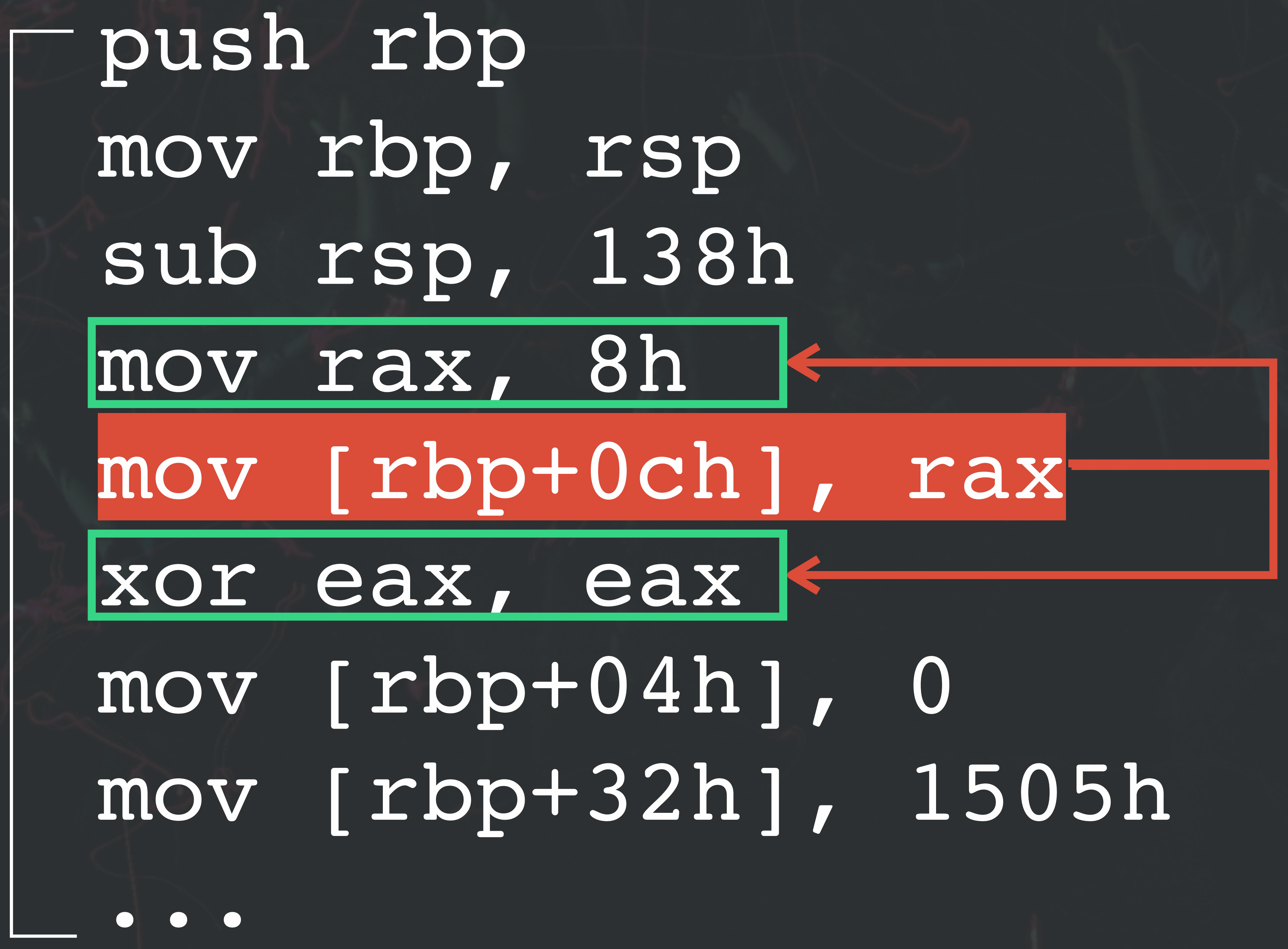
```
push rbp
mov rbp, rsp
sub rsp, 138h
mov rax, 8h
mov [rbp+0ch], rax
xor eax, eax
mov [rbp+04h], 0
mov [rbp+32h], 1505h
...
```

#Asm2Vec



```
push rbp
mov rbp, rsp
mov rsp, 138h
mov rax, 8h
mov [rbp+0ch], rax
xor eax, eax
mov [rbp+04h], 0
mov [rbp+32h], 1505h
...
```

#Asm2Vec



```
push rbp
mov rbp, rsp
sub rsp, 138h
mov rax, 8h
mov [rbp+0ch], rax
xor eax, eax
mov [rbp+04h], 0
mov [rbp+32h], 1505h
...
```

#Asm2Vec

```
push rbp
mov rbp, rsp
sub rsp, 138h
mov rax, 8h
mov [rbp+0ch], rax
xor eax, eax
mov [rbp+04h], 0
mov [rbp+32h], 1505h
...
```

#Asm2Vec

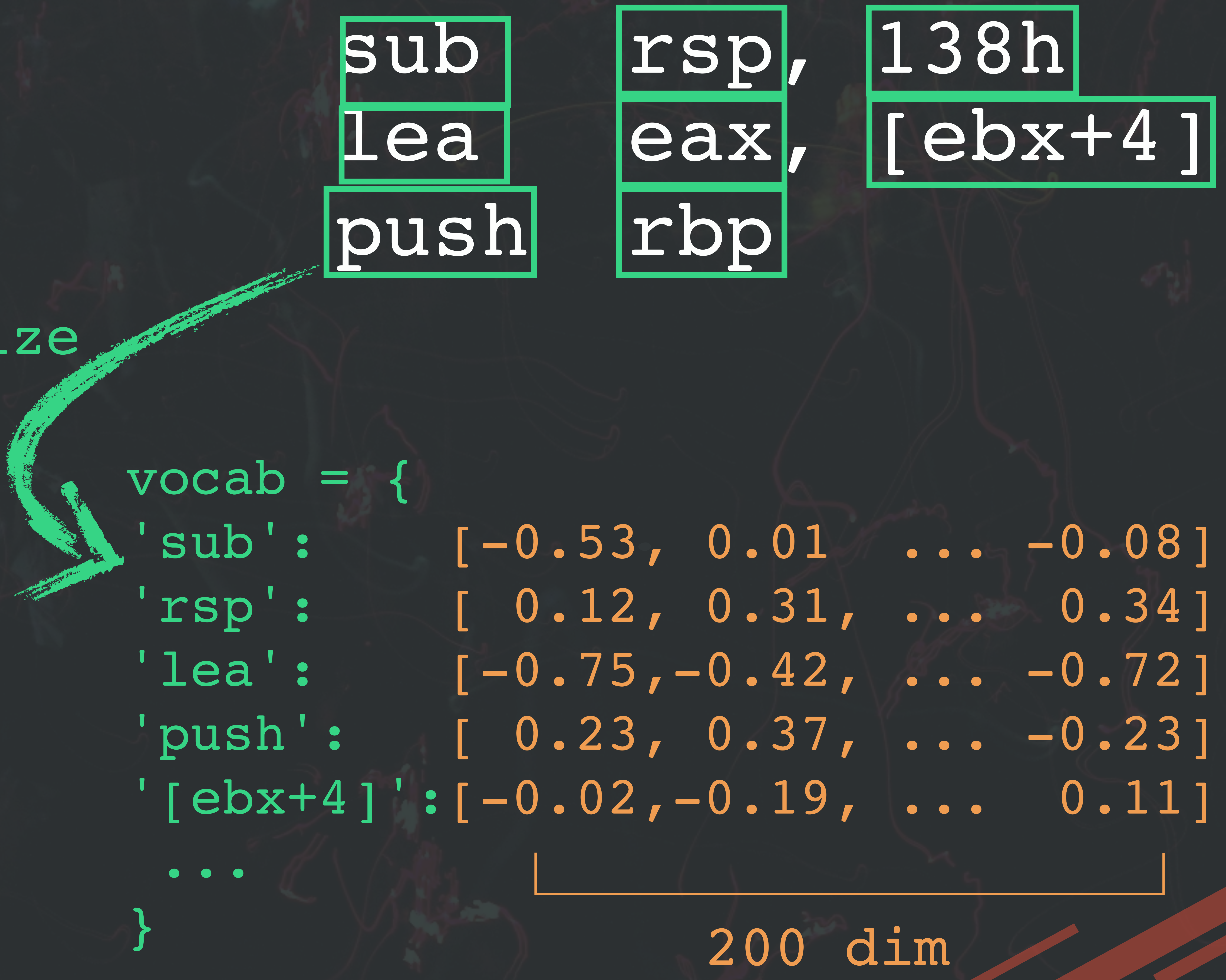
```
push rbp
mov rbp, rsp
sub rsp, 138h
mov rax, 8h
mov [rbp+0ch], rax
xor eax, eax
mov [rbp+04h], 0
mov [rbp+32h], 1505h
...
```

#Asm2Vec

```
push rbp
mov rbp, rsp
sub rsp, 138h
mov rax, 8h
mov [rbp+0ch], rax
xor eax, eax
mov [rbp+04h], 0
mov [rbp+32h], 1505h
...
```

#Asm2Vec

```
push rbp
mov rbp, rsp
sub rsp, 138h
mov rax, 8h
mov [rbp+0ch], rax
xor eax, eax
mov [rbp+04h], 0
mov [rbp+32h], 1505h
...
```



```
sub    rsp,  138h
lea    eax,  [ebx+4]
push   rbp
```

Tokenize

```
vocab = {
'sub':    [-0.53, 0.01  ... -0.08],
'rsp':    [ 0.12, 0.31, ...  0.34],
'lea':    [-0.75,-0.42, ... -0.72],
'push':   [ 0.23, 0.37, ... -0.23],
'[ebx+4]':[-0.02,-0.19, ...  0.11],
 ...
}
```

200 dim

aaaddress1@chroot.org

#Asm2Vec

```
push rbp
mov rbp, rsp
sub rsp, 138h
mov rax, 8h
mov [rbp+0ch], rax
xor eax, eax
mov [rbp+04h], 0
mov [rbp+32h], 1505h
...
```

```
                 operands
operator
sub          rsp, 138h
lea          eax, [ebx+4]
push         rbp
...
```

#Asm2Vec

```
push rbp
mov rbp, rsp
sub rsp, 138h
mov rax, 8h
mov [rbp+0ch], rax
xor eax, eax
mov [rbp+04h], 0
mov [rbp+32h], 1505h
...
```

operator: sub

operands: rsp, 138h

$$T(\text{instruction}) =$$

$$T(\text{sub}) \;||\; (\; T(\text{rsp})/2 + T(\text{138h})/2 \;)$$

#Asm2Vec

```
push rbp
mov rbp, rsp
sub rsp, 138h
mov rax, 8h
mov [rbp+0ch], rax
xor eax, eax
mov [rbp+04h], 0
mov [rbp+32h], 1505h
...
```

operator operands

push rbp

T(instruction) = T(push) || (T(rbp))

#Asm2Vec

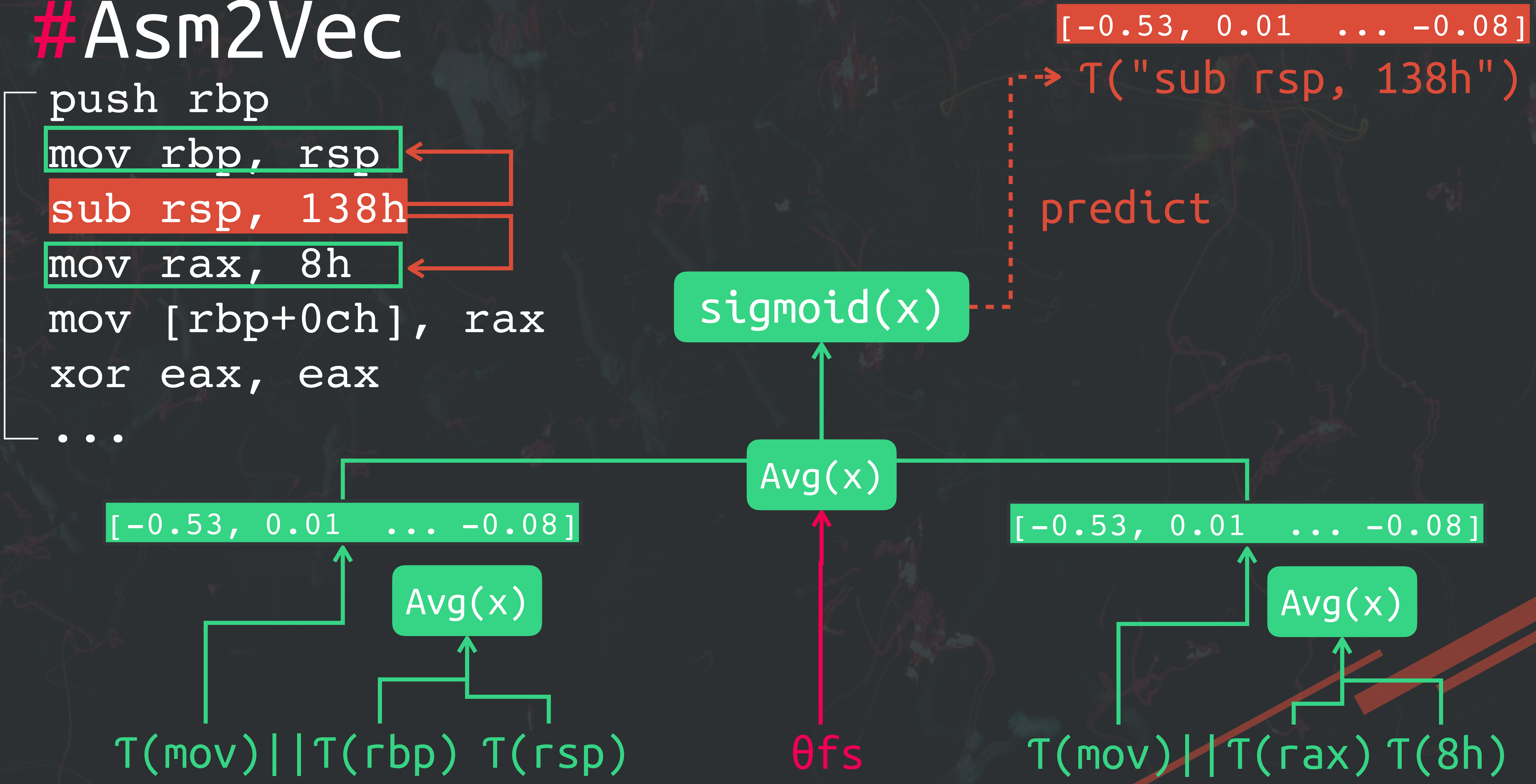
```
push rbp
mov rbp, rsp
sub rsp, 138h
mov rax, 8h
mov [rbp+0ch], rax
xor eax, eax
mov [rbp+04h], 0
mov [rbp+32h], 1505h
nop
```

$$\overset{\text{operator}}{\texttt{nop}}\quad (\overbrace{\texttt{null}}^{\text{operands}})$$

$$T(\text{instruction}) = T(\text{nop}) \;||\; (\text{ null })$$

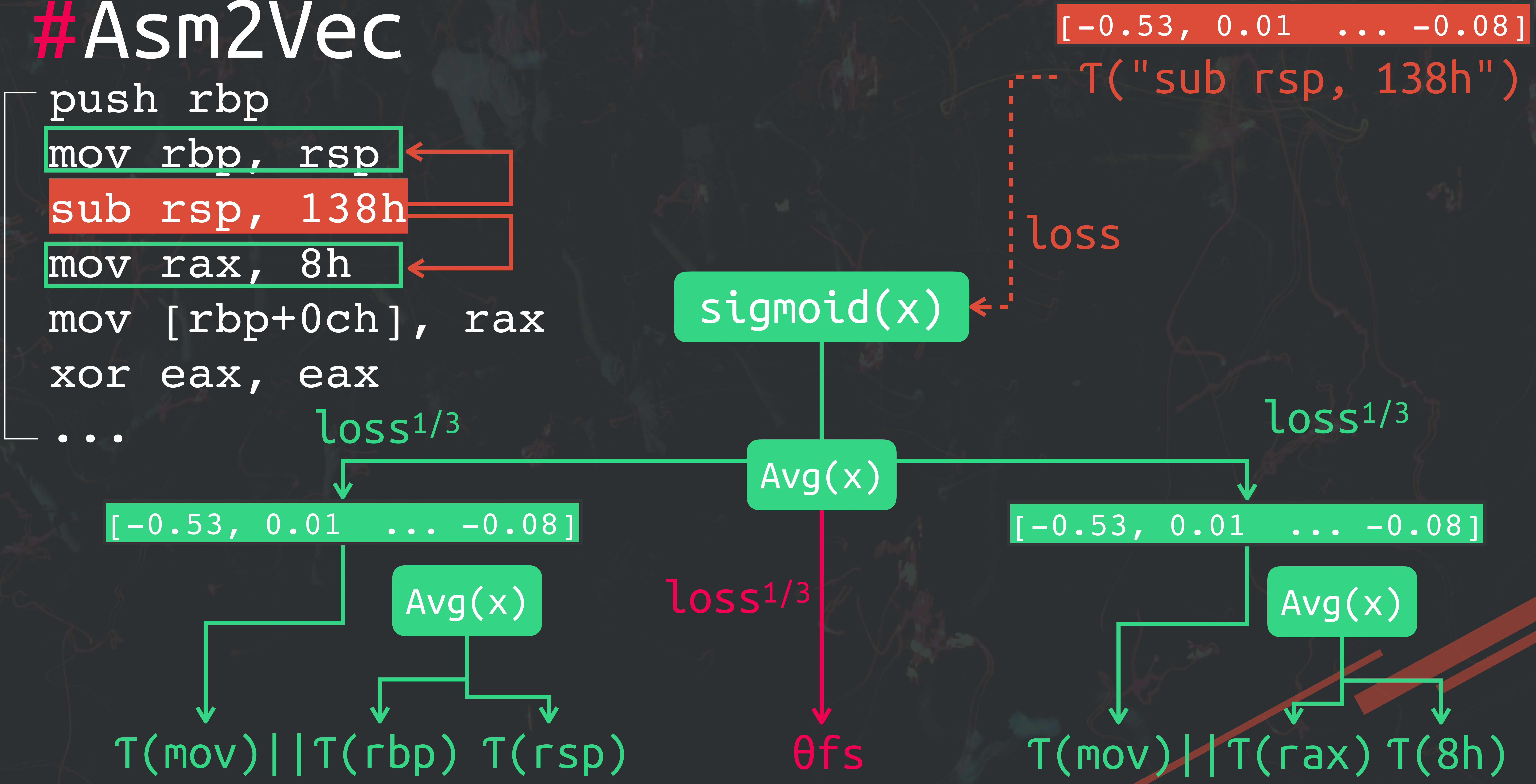
#Asm2Vec

```
push rbp
mov rbp, rsp
sub rsp, 138h
mov rax, 8h
mov [rbp+0ch], rax
xor eax, eax
...
```



aaaddress1@chroot.org

#Asm2Vec



```
push rbp
mov rbp, rsp
sub rsp, 138h
mov rax, 8h
mov [rbp+0ch], rax
xor eax, eax
...
```

aaaddress1@chroot.org

$./exp

- Dataset

  - malware: Mirai samples from VirusTotal (40000+)
  - benign: ELF from Linux-based IoT firmware (3600+)
  - stripped binary

- Training
  - random choose only 25 Mirai samples to train
  - each token represented by 200-dim vector (random)
  - negative sampling: 25 tokens
  - decreasing learning rate: 0.025 → 0.0025

- Cross validation: 10 times
- Malicious: Similarity(binary, model) >= 95%

$./exp

- MIPS

```
- Mirai:  96.75% (18467 samples)
- Benign: 96.41% (348   samples)
```

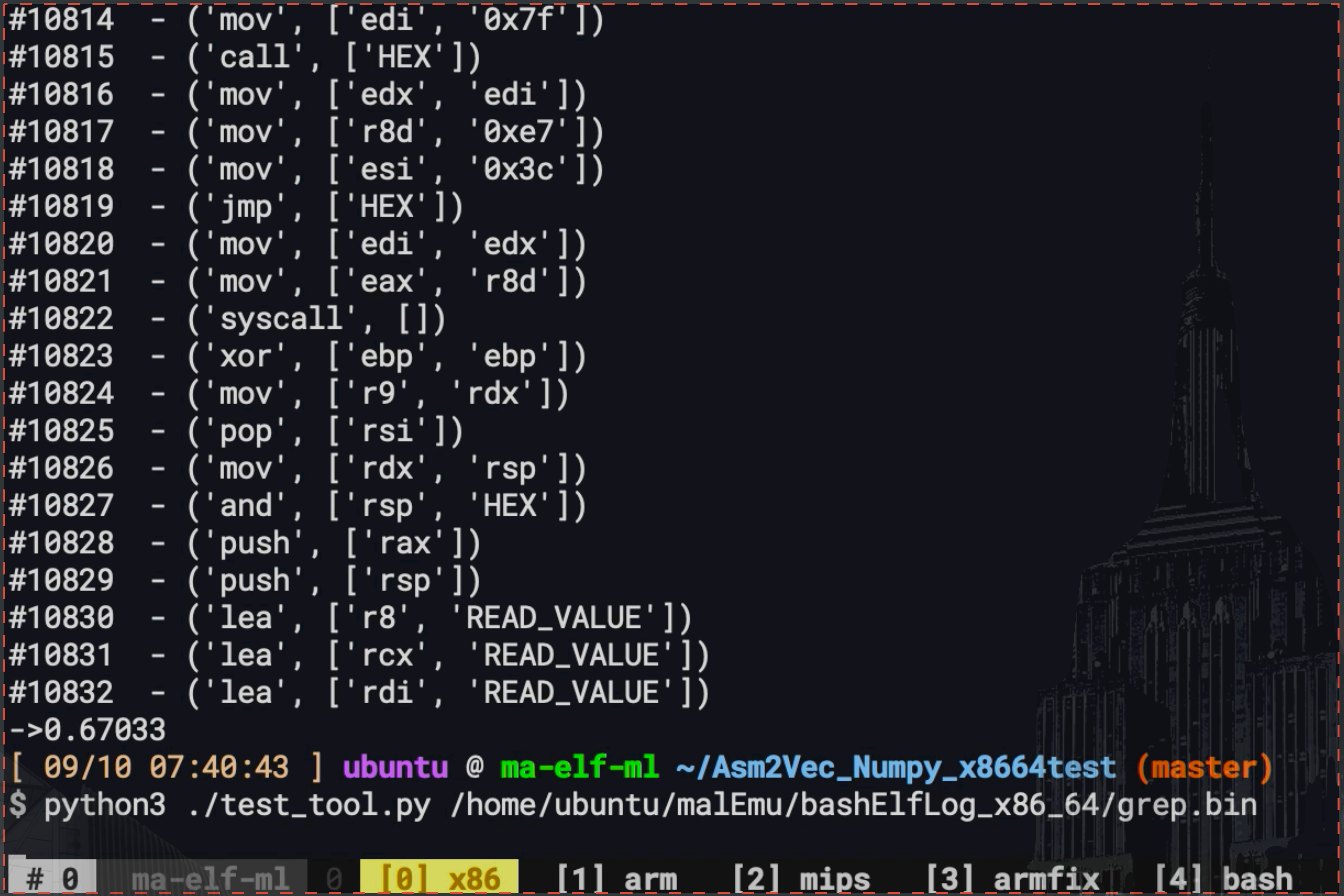
- x86

```
- Mirai:  96.75% (2564  samples)
- Benign: 99.93% (1567  samples)
```

- ARM

```
- Mirai:  98.53% (23827 samples)
- Benign: 93.87% (1699  samples)
```

/>Demo

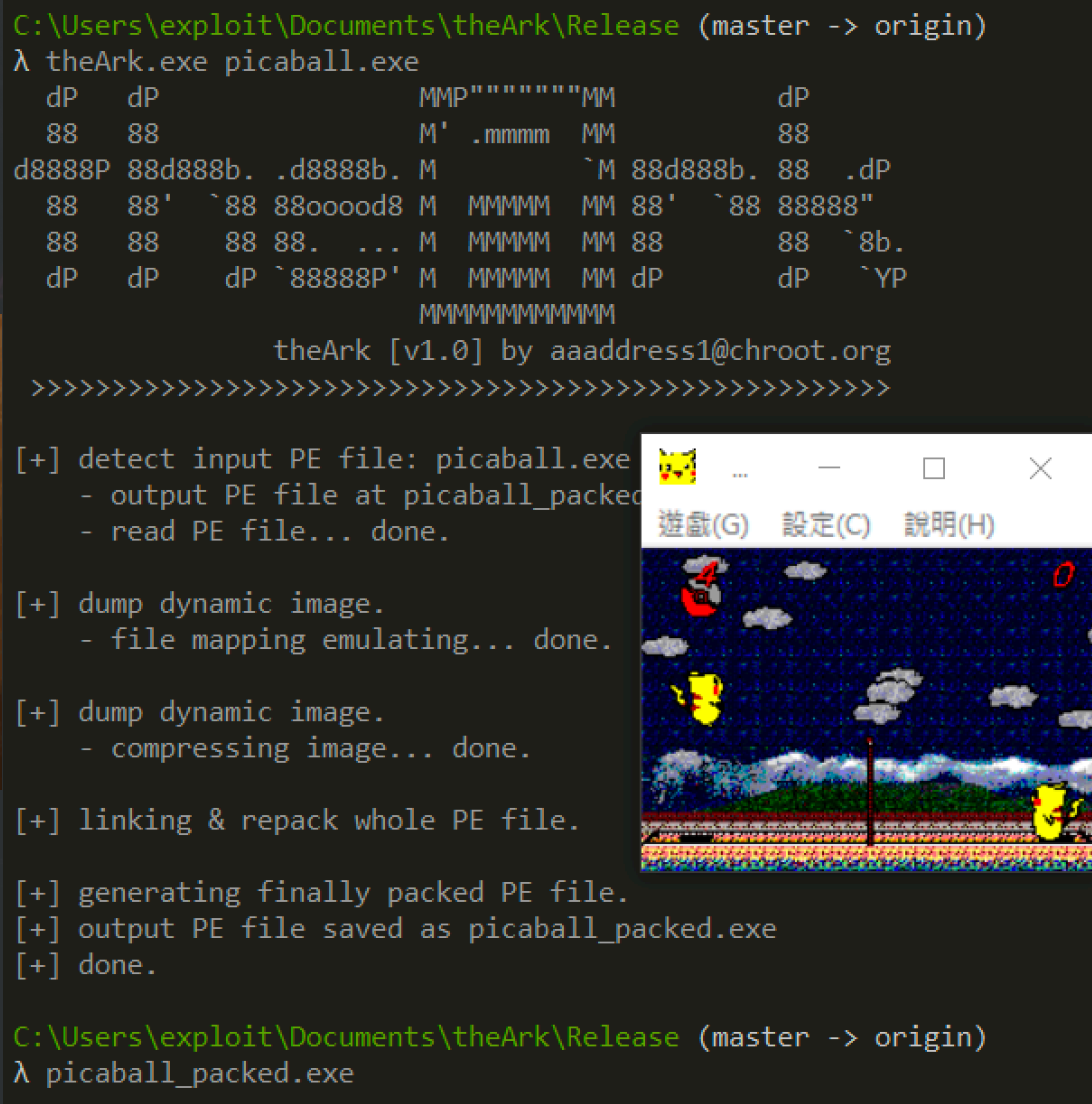


```
#10814  - ('mov', ['edi', '0x7f'])
#10815  - ('call', ['HEX'])
#10816  - ('mov', ['edx', 'edi'])
#10817  - ('mov', ['r8d', '0xe7'])
#10818  - ('mov', ['esi', '0x3c'])
#10819  - ('jmp', ['HEX'])
#10820  - ('mov', ['edi', 'edx'])
#10821  - ('mov', ['eax', 'r8d'])
#10822  - ('syscall', [])
#10823  - ('xor', ['ebp', 'ebp'])
#10824  - ('mov', ['r9', 'rdx'])
#10825  - ('pop', ['rsi'])
#10826  - ('mov', ['rdx', 'rsp'])
#10827  - ('and', ['rsp', 'HEX'])
#10828  - ('push', ['rax'])
#10829  - ('push', ['rsp'])
#10830  - ('lea', ['r8', 'READ_VALUE'])
#10831  - ('lea', ['rcx', 'READ_VALUE'])
#10832  - ('lea', ['rdi', 'READ_VALUE'])
->0.67033
[ 09/10 07:40:43 ] ubuntu @ ma-elf-ml ~/Asm2Vec_Numpy_x8664test (master)
$ python3 ./test_tool.py /home/ubuntu/malEmu/bashElfLog_x86_64/grep.bin

# 0  ma-elf-ml  0  [0] x86   [1] arm   [2] mips   [3] armfix   [4] bash
```

Challenge

/!challenge

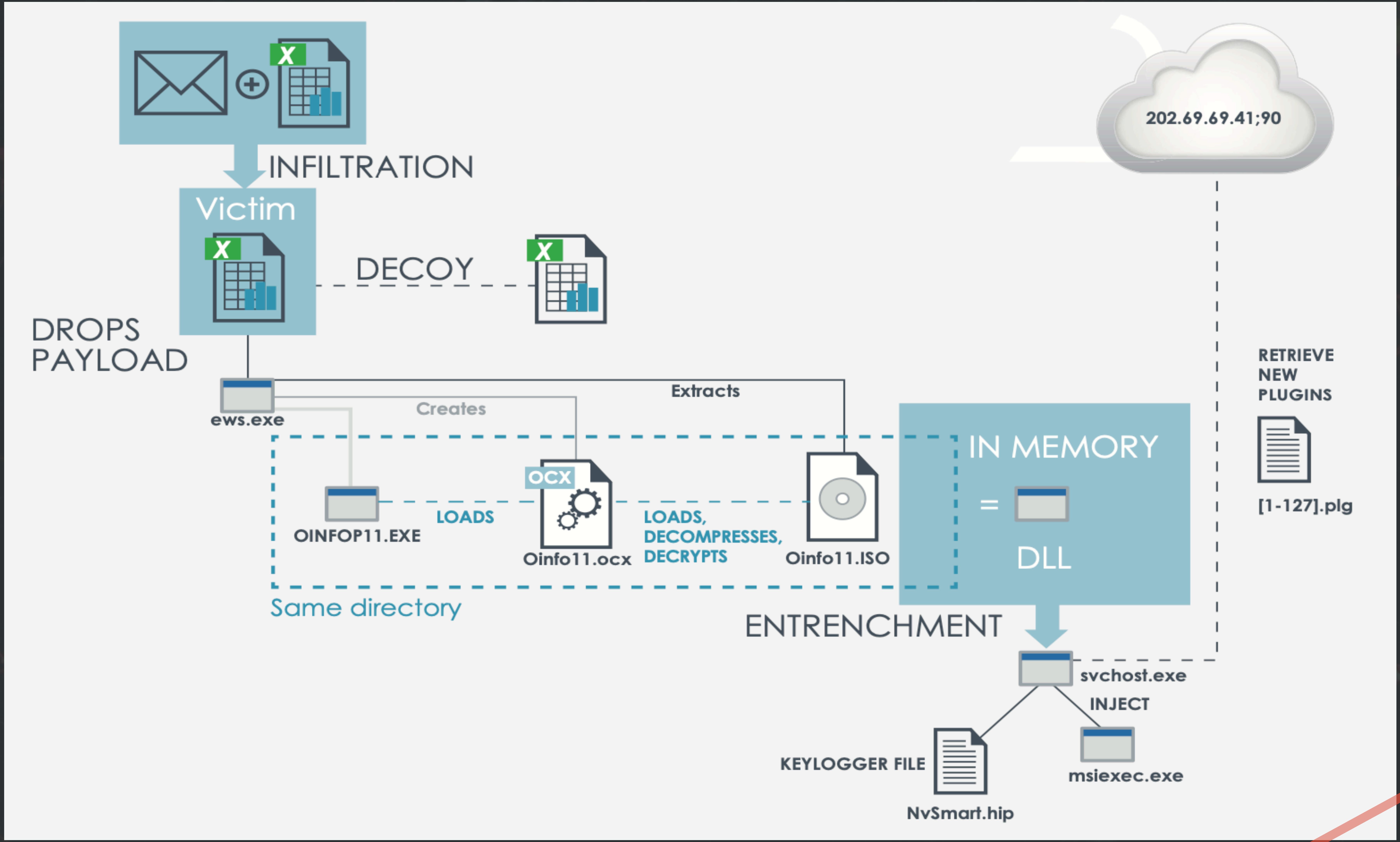
github.com/aaaddress1/theArk



```
C:\Users\exploit\Documents\theArk\Release (master -> origin)
λ theArk.exe picaball.exe
  dP   dP                MMP"""""""MM          dP
  88   88                M' .mmmm  MM          88
d8888P 88d888b. .d8888b. M         `M 88d888b. 88  .dP
  88   88'  `88 88ooood8 M  MMMMM  MM 88'  `88 88888"
  88   88    88 88.  ... M  MMMMM  MM 88       88  `8b.
  dP   dP    dP `88888P' M  MMMMM  MM dP       dP   `YP
                         MMMMMMMMMMMM
                theArk [v1.0] by aaaddress1@chroot.org
 >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

[+] detect input PE file: picaball.exe
    - output PE file at picaball_packed
    - read PE file... done.

[+] dump dynamic image.
    - file mapping emulating... done.

[+] dump dynamic image.
    - compressing image... done.

[+] linking & repack whole PE file.

[+] generating finally packed PE file.
[+] output PE file saved as picaball_packed.exe
[+] done.

C:\Users\exploit\Documents\theArk\Release (master -> origin)
λ picaball_packed.exe
```

/!PluginX

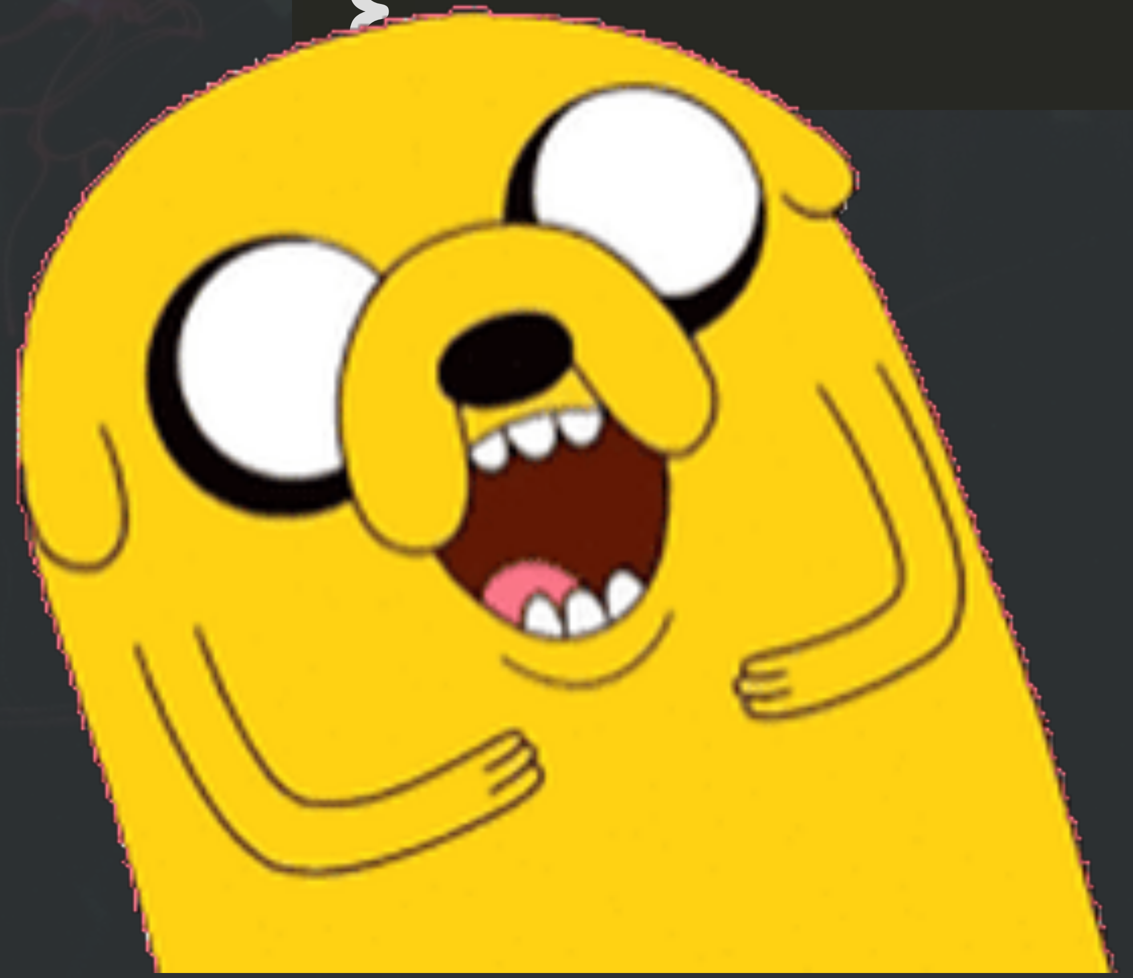
DLL SIDE-LOADING: A Thorn in the Side of the Anti-Virus Industry

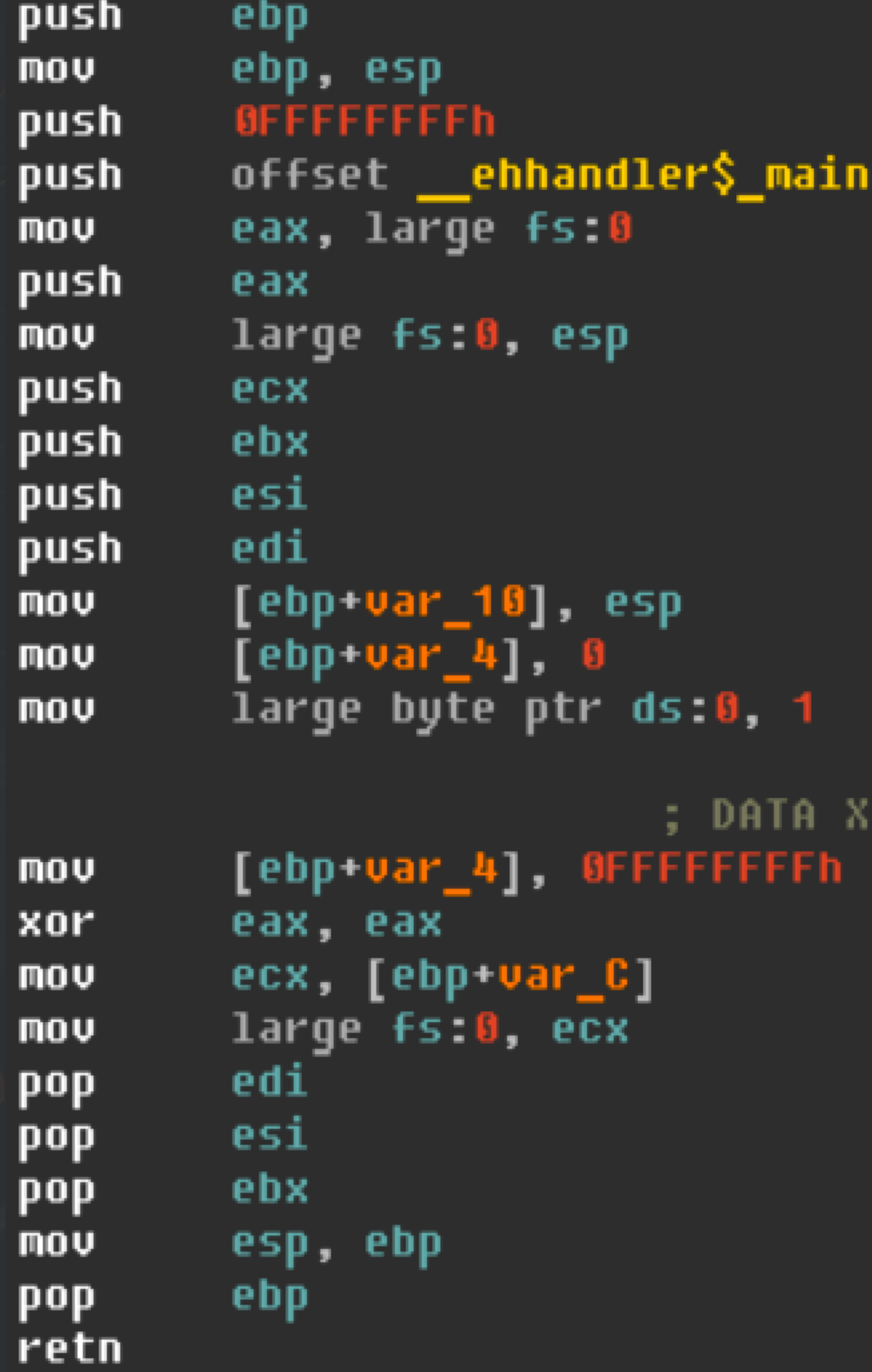


aaaddress1@chroot.org

/!challenge

```
int main(void) {
  try {
    *(char*)NULL = 1;
  } catch (...) {
    puts("Hell Kitty");
  }
}
```

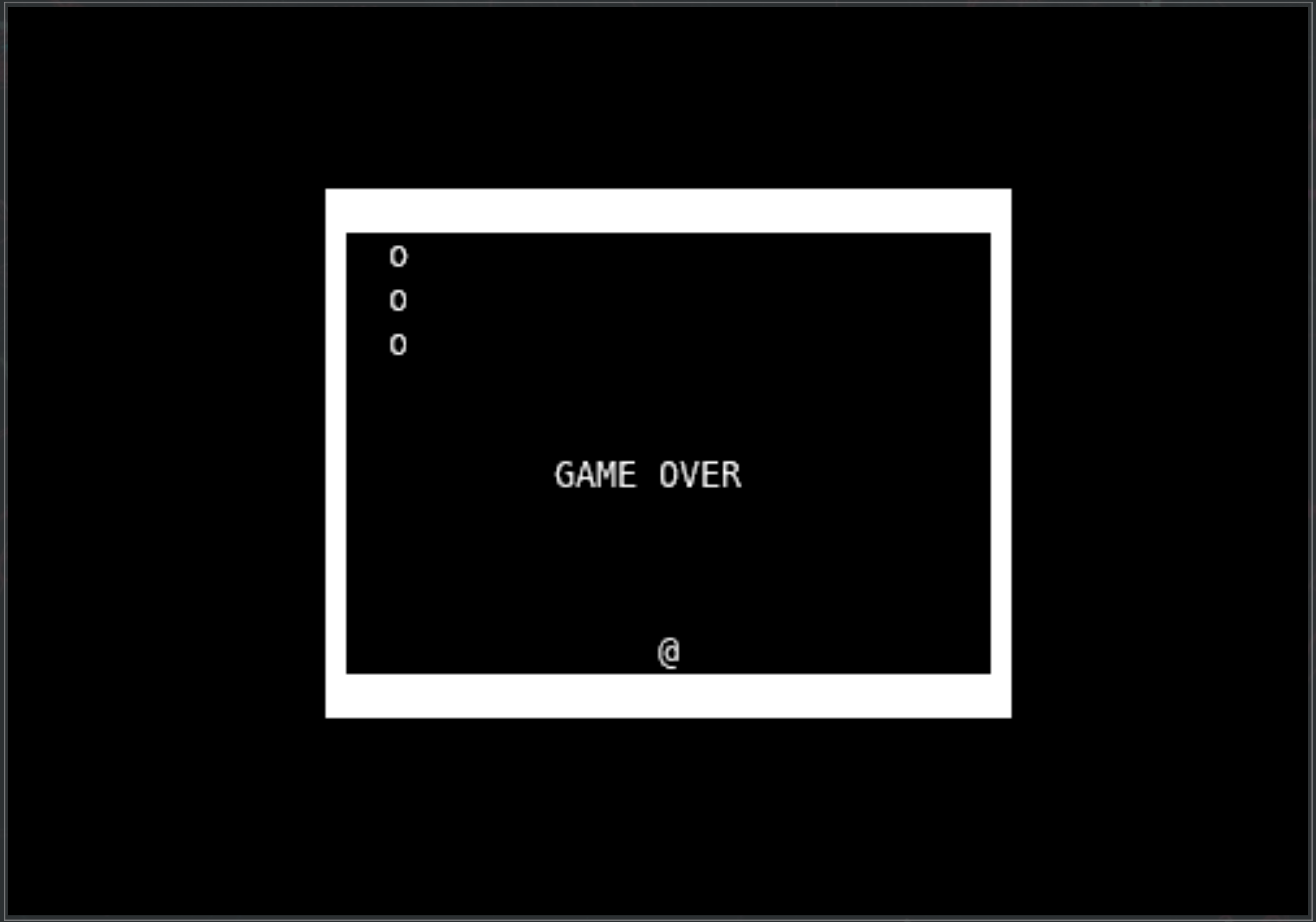




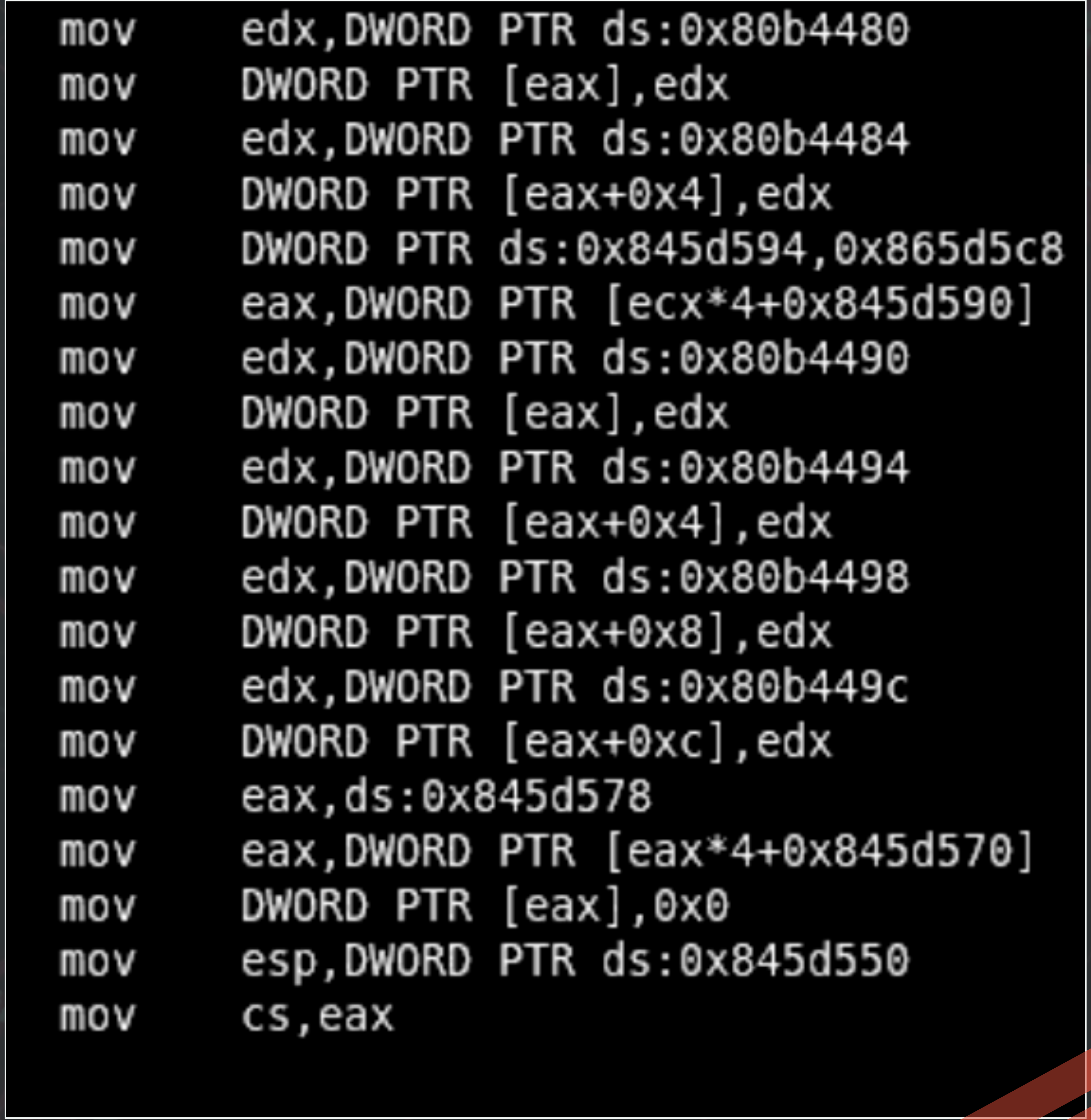
```
push    ebp
mov     ebp, esp
push    0FFFFFFFFh
push    offset __ehhandler$_main
mov     eax, large fs:0
push    eax
mov     large fs:0, esp
push    ecx
push    ebx
push    esi
push    edi
mov     [ebp+var_10], esp
mov     [ebp+var_4], 0
mov     large byte ptr ds:0, 1

                                    ; DATA X
mov     [ebp+var_4], 0FFFFFFFFh
xor     eax, eax
mov     ecx, [ebp+var_C]
mov     large fs:0, ecx
pop     edi
pop     esi
pop     ebx
mov     esp, ebp
pop     ebp
retn
```

/!challenge

github.com/xoreaxeaxeax/movfuscator



GAME OVER



```
mov    edx,DWORD PTR ds:0x80b4480
mov    DWORD PTR [eax],edx
mov    edx,DWORD PTR ds:0x80b4484
mov    DWORD PTR [eax+0x4],edx
mov    DWORD PTR ds:0x845d594,0x865d5c8
mov    eax,DWORD PTR [ecx*4+0x845d590]
mov    edx,DWORD PTR ds:0x80b4490
mov    DWORD PTR [eax],edx
mov    edx,DWORD PTR ds:0x80b4494
mov    DWORD PTR [eax+0x4],edx
mov    edx,DWORD PTR ds:0x80b4498
mov    DWORD PTR [eax+0x8],edx
mov    edx,DWORD PTR ds:0x80b449c
mov    DWORD PTR [eax+0xc],edx
mov    eax,ds:0x845d578
mov    eax,DWORD PTR [eax*4+0x845d570]
mov    DWORD PTR [eax],0x0
mov    esp,DWORD PTR ds:0x845d550
mov    cs,eax
```

/!challenge

- Issue based on Control Flow Walking
  - Self modifying code
    1. Software Packer e.g. VMProtect, Themida
    2. Shellcode Encoder
  - Control Flow Rerouting
    1. Error handling e.g. SEH
    2. MultiThread
  - Exported malicous function
  - Virtual Method Table
- Vector Obfuscation
  - 95% benignware / 5% injected shellcode
  - Use common instructions as gadgets to build a obfuscation chain e.g. movfuscator

Thanks!

aaaddress1@chroot.org

Github

Slide

Facebook

@aaaddress1