https://github.com/25077667/VMPilot

VMPilot

New VMP aims for secure by design.

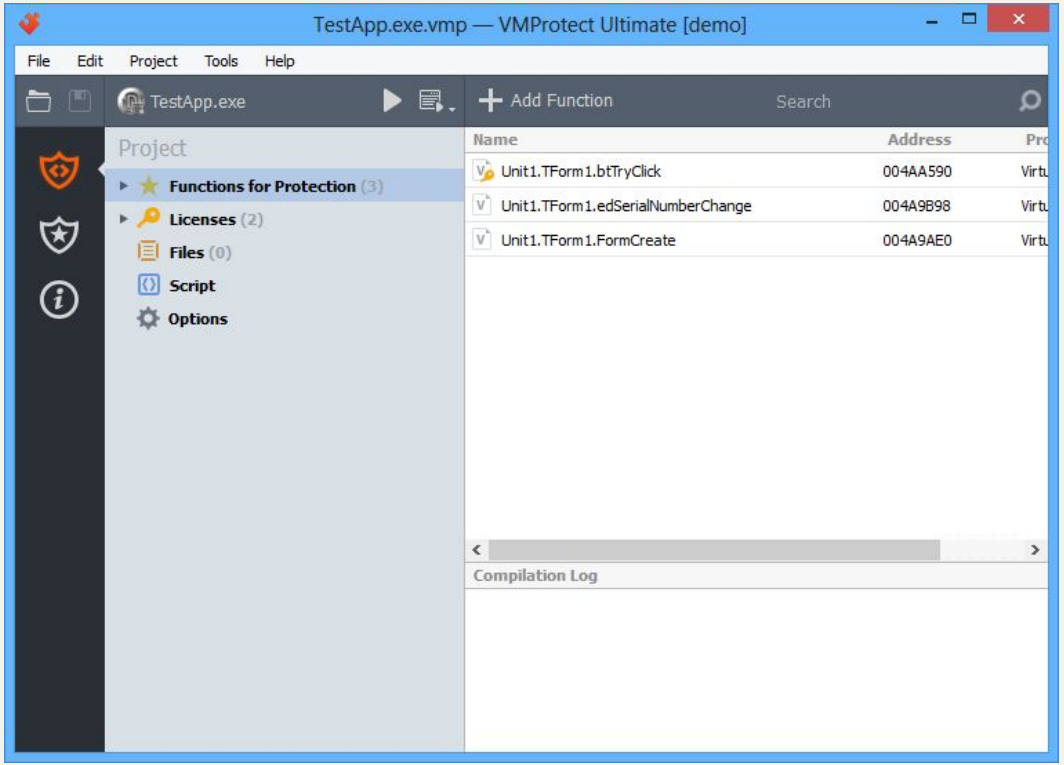
scc@teamt5.org

Outline

1. Introduction
2. Mechanism SDK
3. Proof
4. Mechanism Run time
5. Tiny demo

Introduction

What's VMP (original)



Also refer to:
https://github.com/JonathanSalwan/VMProtect-devirtualization

https://vmpsoft.com/support/user-manual/working-with-vmprotect/main-window/project-section/functions-for-protection-section/

Some issues before

- Inlining from compiler
- Legacy cryptography scheme
- Black-boxed behavior
- Windows CET
  - RIP ROP
  - Intel docs

```
7   #include "lib/VMProtectSDK.h"
8
9   using namespace std;
10  bool IsReturnFalse7()
11  {
12      return false;
13  }
14
15  static inline uint8_t FFFF(uint8_t n, unsigned int c)···
21
22  static inline uint8_t GGGG(uint8_t n, unsigned int c)···
28
29  char* FuxkVMPDecrypt(char* Data, size_t size)
30  {
31      VMProtectBegin("VMP: fuxk_vmp_decrypt");
32      for (; IsReturnFalse7();)
33      {
34      }
35
36      auto key = VMProtectDecryptStringA("**************");
37      for (; IsReturnFalse7();)
38      {
39      }ö
40      // SOME LOGIC HERE...
41
42      VMProtectFreeString(key);
43      VMProtectEnd();
44      return Data;
45  }
46
```

Expected Usage

Similar to VMProtect:

```
#include <vmpilot/sdk.hpp>

template <typename T>
T square(T x) {
    VMPilot_Begin(__FUNCTION__);
    auto result = x * x;
    VMPilot_End(__FUNCTION__);
    return result;
}
```

Output:

```
square:
    push rbp
    call    _Z13VMPilot_BeginPKc    ; VMPilot_Begin(__FUNCTION__);
    ... garbage code ...
    ... garbage code ...
    ... garbage code ...
    call    _Z11VMPilot_EndPKc      ; VMPilot_End(__FUNCTION__);
    pop rbp
    ret
```

Balanced Parentheses?

TEAMT5 杜浦數位安全

```
VMPilot_Begin("A");
auto result = x * x;
VMPilot_End("A");
```

```
VMPilot_Begin("A");
VMPilot_Begin("B");
auto result = x * x;
VMPilot_End("B");
VMPilot_End("A");
```

Expected Usage

Similar to VMProtect:

```
#include <vmpilot/sdk.hpp>

template <typename T>
T square(T x) {
    VMPilot_Begin(__FUNCTION__);
    auto result = x * x;
    VMPilot_End(__FUNCTION__);
    return result;
}
```

Output:

```
square:
    push rbp
    call    _Z13VMPilot_BeginPKc    ; VMPilot_Begin(__FUNCTION__);
    ... garbage code ...
    ... garbage code ...
    ... garbage code ...
    call    _Z11VMPilot_EndPKc      ; VMPilot_End(__FUNCTION__);
    pop rbp
    ret
```

Balanced Parentheses?

TEAMT5
杜 浦 數 位 安 全

```
VMPilot_Begin("A");
auto result = x * x;
VMPilot_End("A");
```

(...)

```
VMPilot_Begin("A");
VMPilot_Begin("B");
auto result = x * x;
VMPilot_End("B");
VMPilot_End("A");
```

[(...)]

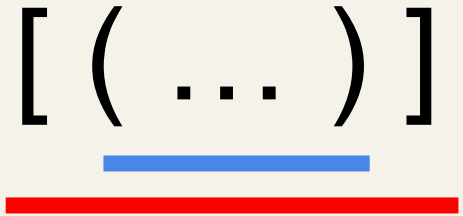
Balanced Parentheses?

TEAMT5
杜 浦 數 位 安 全

```
VMPilot_Begin("A");
auto result = x * x;
VMPilot_End("A");
```

(...)

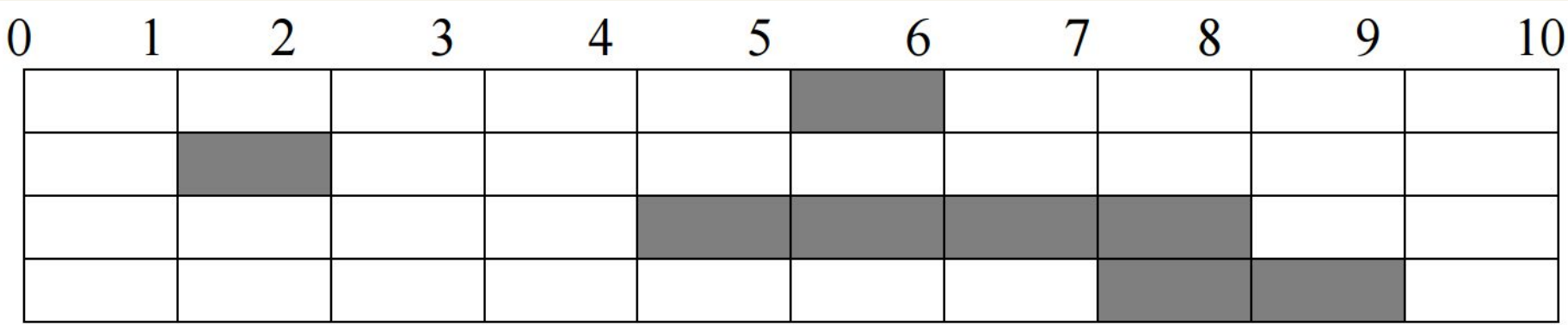
```
VMPilot_Begin("A");
VMPilot_Begin("B");
auto result = x * x;
VMPilot_End("B");
VMPilot_End("A");
```



b966. 第 3 題 線段覆蓋長度

給定一維座標上一些線段，求這些線段所覆蓋的長度，注意，重疊的部分只能算一次。

例如給定 4 個線段：(5, 6)、(1, 2)、(4, 8)、(7, 9)，如下圖，線段覆蓋長度為 6 。

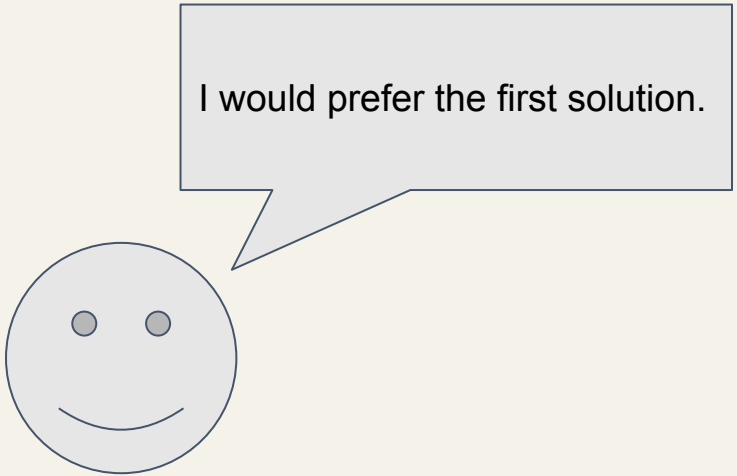


https://zerojudge.tw/ShowProblem?problemid=b966

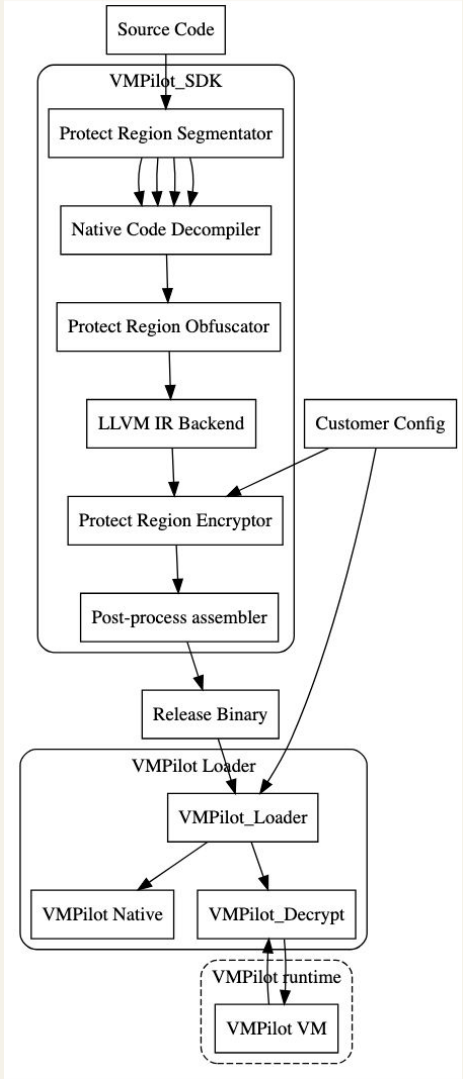
Two solutions

TEAMT5
杜 浦 數 位 安 全

1. Emitting warns
2. Overriding



Mechanism



Segmentator

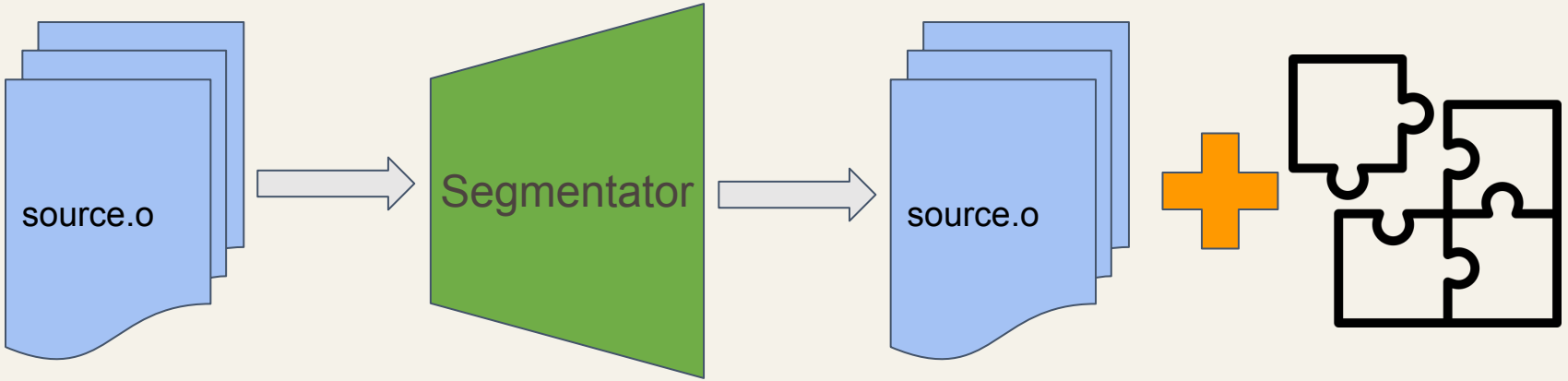
TEAMT5 杜浦數位安全

```
namespace VMPilot::SDK::Segmentator {
class Segmentator {
std::string m_filename;
std::function<void(std::string)> m_callback;

public:
Segmentator(const std::string& filename);
~Segmentator() = default;

void do_segmentation() noexcept;
};
} // namespace VMPilot::SDK::Segmentator
```

```
m_callback = [](const std::string& filename) {
    auto me = ELF_Segmentator(filename);
    me();
};
```

Segmentator

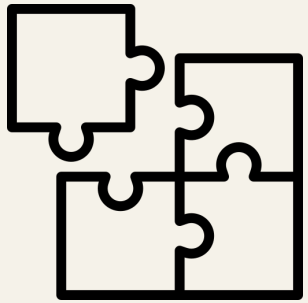
TEAMT5
杜 浦 數 位 安 全



Segments asm

```
_Z6squareIiET_S0_:
        push rbp
        mov rbp, rsp
        sub rsp, 32
        mov DWORD PTR [rbp-20], edi
        mov edi, OFFSET FLAT:.LC0
        call _Z13VMPilot_BeginPKc
        mov eax, DWORD PTR [rbp-20]
        imul eax, eax
        mov DWORD PTR [rbp-4], eax
        mov edi, OFFSET FLAT:.LC0
        call _Z11VMPilot_EndPKc
        mov eax, DWORD PTR [rbp-4]
        leave
        ret
```

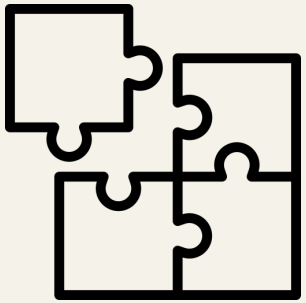
TEAMT5
杜 浦 數 位 安 全



```
template <typename T>
T square(T x) {
    VMPilot_Begin(__FUNCTION__)
    ;
    auto result = x * x;
    VMPilot_End(__FUNCTION__);
    return result;
}
```

Segments IR

TEAMT5
杜 浦 數 位 安 全



```
define linkonce_odr dso_local noundef i32 @int square<int>(int)(i32 noundef %x) comdat {
entry:
%x.addr = alloca i32, align 4
%result = alloca i32, align 4
store i32 %x, ptr %x.addr, align 4
call void @VMPilot_Begin(char const*)(ptr noundef
@__FUNCTION__.int square<int>(int))
%0 = load i32, ptr %x.addr, align 4
%1 = load i32, ptr %x.addr, align 4
%mul = mul nsw i32 %0, %1
store i32 %mul, ptr %result, align 4
call void @VMPilot_End(char const*)(ptr noundef
@__FUNCTION__.int square<int>(int))
%2 = load i32, ptr %result, align 4
ret i32 %2
}
```

Obfuscation

TEAMT5 杜浦數位安全

scc-Taipei ❐ 0 ● 1 zsh (ssh)

```
1570
1571 0000000000001b80 <_Z24SampleFiniteStateMachinev>:
1572     1b80:       sub    rsp,0x18
1573     1b84:       mov    esi,0x249
1574     1b89:       lea    rdi,[rip+0x0]        # 1b90 <_Z24SampleFiniteStateMachinev+0x10>
1575     1b90:       mov    rax,QWORD PTR fs:0x28
1576     1b99:       mov    QWORD PTR [rsp+0x8],rax
1577     1b9e:       xor    eax,eax
1578     1ba0:       lea    rdx,[rsp+0x4]
1579     1ba5:       mov    DWORD PTR [rsp+0x4],0x2a
1580     1bad:       call   10 <_ZN8andrivet13ADVobfuscator17ObfuscatedCallRetINS0_8Machine17MachineEiNS0_17ObfuscatedAddressIPFiiEEEJiEEET0_T1_DpOT2_.isra.0>
1581     1bb2:       mov    rax,QWORD PTR [rsp+0x8]
1582     1bb7:       sub    rax,QWORD PTR fs:0x28
1583     1bc0:       jne    1bc7 <_Z24SampleFiniteStateMachinev+0x47>
1584     1bc2:       add    rsp,0x18
1585     1bc6:       ret
1586     1bc7:       call   1bcc <_Z24SampleFiniteStateMachinev+0x4c>
1587
```

https://polarply.medium.com/build-your-first-llvm-obfuscator-80d16583392b

Obfuscati

TEAMT5
杜 浦 數 位 安 全

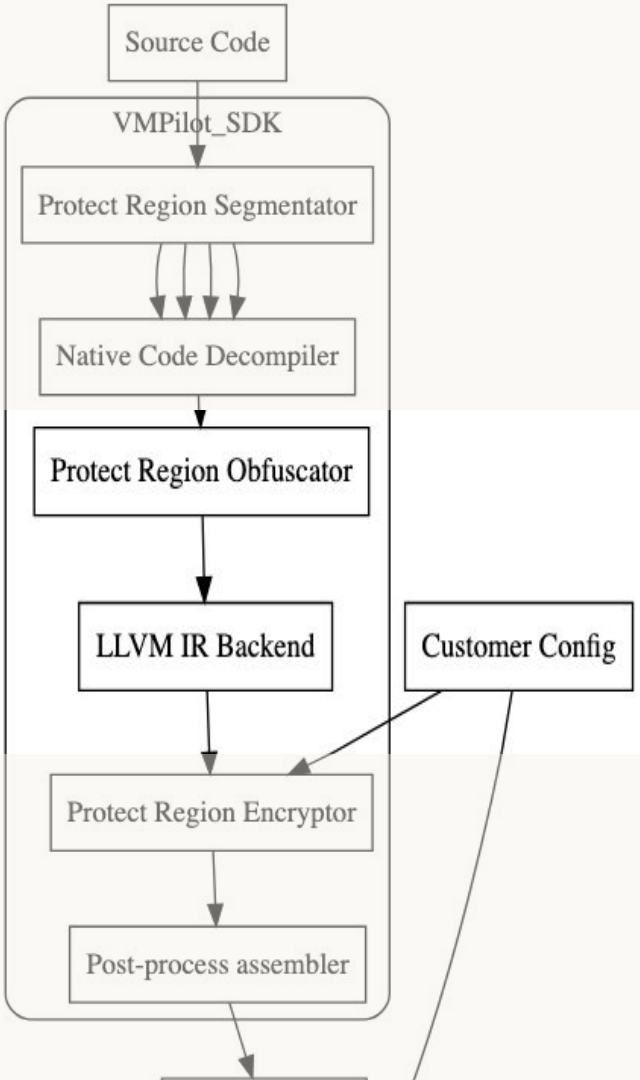
```
1570
1571 0000000000001b80 <_Z24SampleFini
1572     1b80:       sub    rsp,0x18
1573     1b84:       mov    esi,0x249
1574     1b89:       lea    rdi,[rip+
1575     1b90:       mov    rax,QWORD
1576     1b99:       mov    QWORD PTR
1577     1b9e:       xor    eax,eax
1578     1ba0:       lea    rdx,[rsp+
1579     1ba5:       mov    DWORD PTR
1580     1bad:       call   10 <_ZN8a
1581     1bb2:       mov    rax,QWORD
1582     1bb7:       sub    rax,QWORD
1583     1bc0:       jne    1bc7 <_Z2
1584     1bc2:       add    rsp,0x18
1585     1bc6:       ret
1586     1bc7:       call   1bcc <_Z2
1587
```

```
scc-Taipei ❐ 0 ● 1 zsh (ssh)                                                    ⌘2

1570
1571 0000000000001b80 <_Z24SampleFiniteStateMachinev>:
1572     1b80:       sub    rsp,0x18
1573     1b84:       mov    esi,0x249
1574     1b89:       lea    rdi,[rip+0x0]        # 1b90 <_Z24SampleFiniteStateMachinev+0x10>
1575     1b90:       mov    rax,QWORD PTR fs:0x28
1576     1b99:       mov    QWORD PTR [rsp+0x8],rax
1577     1b9e:       xor    eax,eax
1578     1ba0:       lea    rdx,[rsp+0x4]
1579     1ba5:       mov    DWORD PTR [rsp+0x4],0x2a
1580     1bad:       call   10 <_ZN8andrivet13ADVobfuscator17ObfuscatedCallRetINS0_8Machine17MachineEiNS0_17ObfuscatedAddressIPFiiEEEJiEEET0_T1_DpOT2_.isra.0>
1581     1bb2:       mov    rax,QWORD PTR [rsp+0x8]
1582     1bb7:       sub    rax,QWORD PTR fs:0x28
1583     1bc0:       jne    1bc7 <_Z24SampleFiniteStateMachinev+0x47>
1584     1bc2:       add    rsp,0x18
1585     1bc6:       ret
1586     1bc7:       call   1bcc <_Z24SampleFiniteStateMachinev+0x4c>
1587
1588 Disassembly of section .text._ZN5boost6detail8function21function_obj_invoker0INS_3_bi6bind_tINS_3msm4back11HandledEnumENS_4_mfi3mf2IS7_NS6_13state_machineIN8andrivet13ADVobf
1588 uscator8Machine17MachineINSC_SeventIiNSC_17ObfuscatedAddressIPFiiEEEJRiEEEiEENS_9parameter5void_ESO_SO_SO_EERKNSM_6event5EhEENS3_5list3INS3_5valueIPSP_EENSV_ISQ_EENSV_IhEEEE
1588 EES7_E6invokeERNS1_15function_bufferE:
1589
1590 0000000000000000 <_ZN5boost6detail8function21function_obj_invoker0INS_3_bi6bind_tINS_3msm4back11HandledEnumENS_4_mfi3mf2IS7_NS6_13state_machineIN8andrivet13ADVobfuscator8Mac
1590 hine17MachineINSC_SeventIiNSC_17ObfuscatedAddressIPFiiEEEJRiEEEiEENS_9parameter5void_ESO_SO_SO_EERKNSM_6event5EhEENS3_5list3INS3_5valueIPSP_EENSV_ISQ_EENSV_IhEEEEES7_E6invo
1590 keERNS1_15function_bufferE>:
1591    0:   mov    rax,QWORD PTR [rdi]
1592    3:   mov    rdi,QWORD PTR [rax+0x8]
1593    7:   movzx  edx,BYTE PTR [rax+0x19]
1594    b:   lea    rsi,[rax+0x18]
1595    f:   add    rdi,QWORD PTR [rax+0x10]
1596   13:   mov    rax,QWORD PTR [rax]
1597   16:   test   al,0x1
1598   18:   je     22 <_ZN5boost6detail8function21function_obj_invoker0INS_3_bi6bind_tINS_3msm4back11HandledEnumENS_4_mfi3mf2IS7_NS6_13state_machineIN8andrivet13ADVobfuscator8Ma
1598 chine17MachineINSC_SeventIiNSC_17ObfuscatedAddressIPFiiEEEJRiEEEiEENS_9parameter5void_ESO_SO_SO_EERKNSM_6event5EhEENS3_5list3INS3_5valueIPSP_EENSV_ISQ_EENSV_IhEEEEES7_E6inv
1598 okeERNS1_15function_bufferE+0x22>
1599   1a:   mov    rcx,QWORD PTR [rdi]
1600   1d:   mov    rax,QWORD PTR [rcx+rax*1-0x1]
1601   22:   jmp    rax
1602
1603 Disassembly of section .text._ZN5boost6detail8function21function_obj_invoker0INS_3_bi6bind_tINS_3msm4back11HandledEnumENS_4_mfi3mf2IS7_NS6_13state_machineIN8andrivet13ADVobf
1603 uscator8Machine17MachineINSC_SeventIiNSC_17ObfuscatedAddressIPFiiEEEJRiEEEiEENS_9parameter5void_ESO_SO_SO_EERKNSM_6event2EhEENS3_5list3INS3_5valueIPSP_EENSV_ISQ_EENSV_IhEEEE
1603 EES7_E6invokeERNS1_15function_bufferE:
1604
1605 0000000000000000 <_ZN5boost6detail8function21function_obj_invoker0INS_3_bi6bind_tINS_3msm4back11HandledEnumENS_4_mfi3mf2IS7_NS6_13state_machineIN8andrivet13ADVobfuscator8Mac
1605 hine17MachineINSC_SeventIiNSC_17ObfuscatedAddressIPFiiEEEJRiEEEiEENS_9parameter5void_ESO_SO_SO_EERKNSM_6event2EhEENS3_5list3INS3_5valueIPSP_EENSV_ISQ_EENSV_IhEEEEES7_E6invo
1605 keERNS1_15function_bufferE>:
1606    0:   mov    rax,QWORD PTR [rdi]
1607    3:   mov    rdi,QWORD PTR [rax+0x8]
1608    7:   movzx  edx,BYTE PTR [rax+0x19]
1609    b:   lea    rsi,[rax+0x18]
1610    f:   add    rdi,QWORD PTR [rax+0x10]
1611   13:   mov    rax,QWORD PTR [rax]
1612   16:   test   al,0x1
1613   18:   je     22 <_ZN5boost6detail8function21function_obj_invoker0INS_3_bi6bind_tINS_3msm4back11HandledEnumENS_4_mfi3mf2IS7_NS6_13state_machineIN8andrivet13ADVobfuscator8Ma
1613 chine17MachineINSC_SeventIiNSC_17ObfuscatedAddressIPFiiEEEJRiEEEiEENS_9parameter5void_ESO_SO_SO_EERKNSM_6event2EhEENS3_5list3INS3_5valueIPSP_EENSV_ISQ_EENSV_IhEEEEES7_E6inv
1613 okeERNS1_15function_bufferE+0x22>
1614   1a:   mov    rcx,QWORD PTR [rdi]
1615   1d:   mov    rax,QWORD PTR [rcx+rax*1-0x1]
1616   22:   jmp    rax
1617
1618 Disassembly of section .text._ZN5boost6detail8function21function_obj_invoker0INS_3_bi6bind_tINS_3msm4back11HandledEnumENS_4_mfi3mf2IS7_NS6_13state_machineIN8andrivet13ADVobf
1618 uscator8Machine17MachineINSC_SeventIiNSC_17ObfuscatedAddressIPFiiEEEJRiEEEiEENS_9parameter5void_ESO_SO_SO_EERKNSM_6event4EhEENS3_5list3INS3_5valueIPSP_EENSV_ISQ_EENSV_IhEEEE
1618 EES7_E6invokeERNS1_15function_bufferE:
1619
1620 0000000000000000 <_ZN5boost6detail8function21function_obj_invoker0INS_3_bi6bind_tINS_3msm4back11HandledEnumENS_4_mfi3mf2IS7_NS6_13state_machineIN8andrivet13ADVobfuscator8Mac
1620 hine17MachineINSC_SeventIiNSC_17ObfuscatedAddressIPFiiEEEJRiEEEiEENS_9parameter5void_ESO_SO_SO_EERKNSM_6event4EhEENS3_5list3INS3_5valueIPSP_EENSV_ISQ_EENSV_IhEEEEES7_E6invo
1620 keERNS1_15function_bufferE>:
1621    0:   mov    rax,QWORD PTR [rdi]
1622    3:   mov    rdi,QWORD PTR [rax+0x8]
1623    7:   movzx  edx,BYTE PTR [rax+0x19]
1624    b:   lea    rsi,[rax+0x18]
1625    f:   add    rdi,QWORD PTR [rax+0x10]
1626   13:   mov    rax,QWORD PTR [rax]
1627   16:   test   al,0x1
1628   18:   je     22 <_ZN5boost6detail8function21function_obj_invoker0INS_3_bi6bind_tINS_3msm4back11HandledEnumENS_4_mfi3mf2IS7_NS6_13state_machineIN8andrivet13ADVobfuscator8Ma
1628 chine17MachineINSC_SeventIiNSC_17ObfuscatedAddressIPFiiEEEJRiEEEiEENS_9parameter5void_ESO_SO_SO_EERKNSM_6event4EhEENS3_5list3INS3_5valueIPSP_EENSV_ISQ_EENSV_IhEEEEES7_E6inv
1628 okeERNS1_15function_bufferE+0x22>

47.2111°C | 23:54 | 25 Jun  scc  scc-Taipei
```

```
essIPFiiEEEJiEEET0_T1_DpOT2_.isra.0>
```

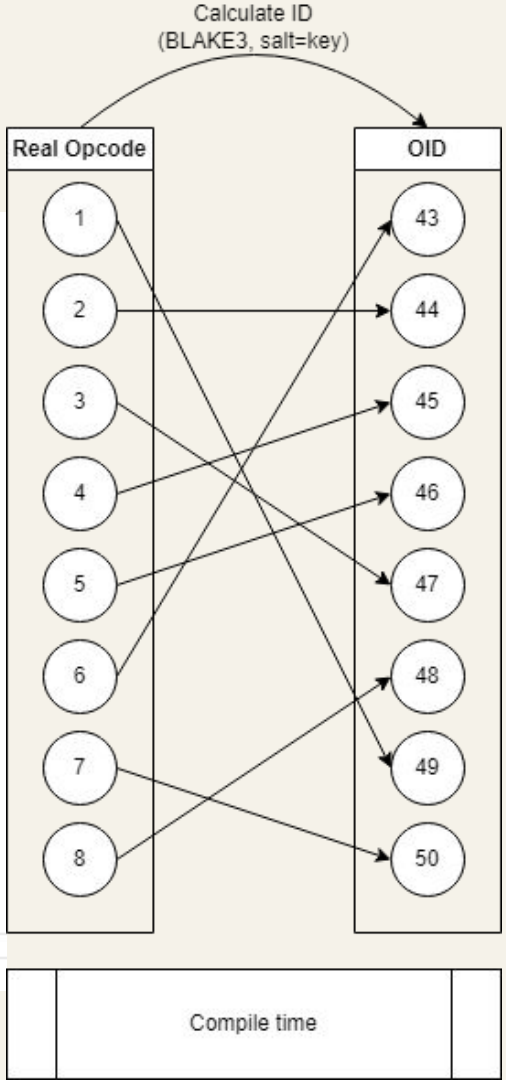
Introduced above

Still **traditional** technique



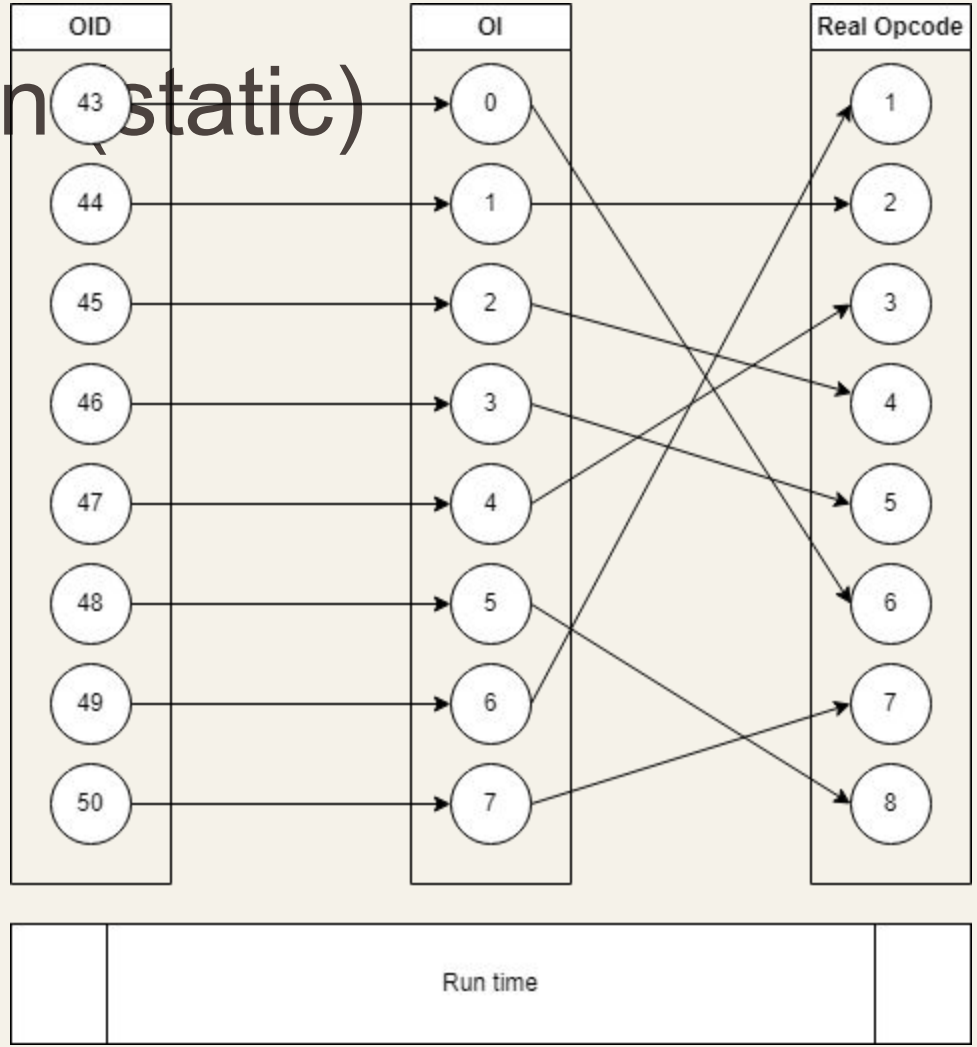
Encryption (static)

```
93  /**
94   * @brief Opcode enum class
95   *
96   * The bit level view of the opcode enum class (into sizeof(Opcode_t)) like:
97   * low bit                                                          high bit
98   * | DataMovement | ArithmeticLogic | ControlTransfer | ThreadingAtomic |
99   * |--------------|-----------------|-----------------|-----------------|
100  * | 0            | 0               | 0               | 0               |
101  *
102  * Each enum class has a size of sizeof(Opcode_t) / 4.
103  *
104  * Example: We would like a DataMovement::Move. And the opcode would be:
105  * | DataMovement::Move | 0 | 0 | 0 |
106  *
107  * Example: We would like a ArithmeticLogic::Add. And the opcode would be:
108  * | 0 | ArithmeticLogic::Add | 0 | 0 |
109  *
110  * Example: We would like a ControlTransfer::Jmp. And the opcode would be:
111  * | 0 | 0 | ControlTransfer::Jmp | 0 |
112  *
113  * Example: We would like a ThreadingAtomic::Lock. And the opcode would be:
114  * | 0 | 0 | 0 | ThreadingAtomic::Lock |
115  *
116  * CAUTION:
117  * It is invalid, if it combine two or more enum classes.
118  *
119  */      You, 上週 • feat(runtime): implement opcode table mechanism …
120
```



Encryption (static)

Prove later



Encryption (dynamic)

TEAMT5
杜 浦 數 位 安 全

Encryption

Not

Implement

Yet

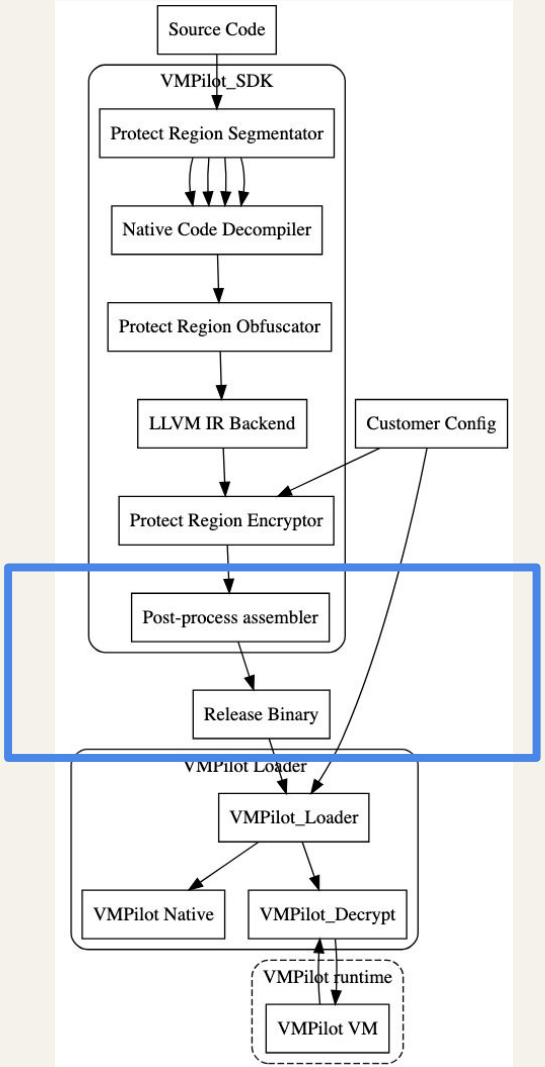
```
26  void VMPilot::Common::Instruction::decrypt(Instruction_t& inst,
27                                             const std::string& key) noexcept {
28      // Decrypt the instruction using the key
29      // We use AES-256-CBC from the OpenSSL library to decrypt the instruction.
30      const auto data = flatten(inst);
31
32      // If key is too short, we pad it with 0
33      std::string padded_key = key;
34      if (key.size() < 32)
35          padded_key.resize(32, 0);
36      else if (key.size() > 32)
37          padded_key.resize(32);
38
39      // Decrypt the instruction
40      EVP_CIPHER_CTX* ctx = EVP_CIPHER_CTX_new();
41      EVP_DecryptInit_ex(
42          ctx, EVP_aes_256_cbc(), NULL,
43          reinterpret_cast<const unsigned char*>(padded_key.data()), NULL);
44      EVP_DecryptUpdate(ctx, reinterpret_cast<unsigned char*>(&inst), NULL,
45                        reinterpret_cast<const unsigned char*>(data.data()),
46                        sizeof(inst));
47      EVP_DecryptFinal_ex(ctx, reinterpret_cast<unsigned char*>(&inst), NULL);
48      EVP_CIPHER_CTX_free(ctx);
49  }
```

Encryption (dynamic)

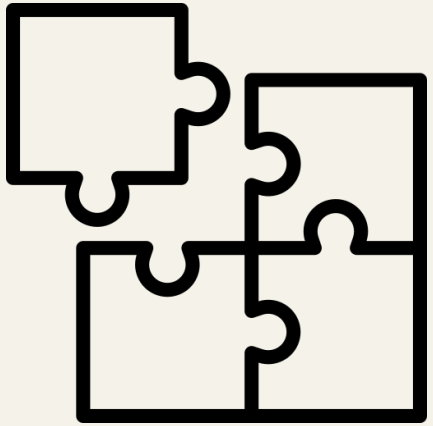
Notice that

It's **full instruction.**

```
26  void VMPilot::Common::Instruction::decrypt(Instruction_t& inst,
27                                             const std::string& key) noexcept {
28      // Decrypt the instruction using the key
29      // We use AES-256-CBC from the OpenSSL library to decrypt the instruction.
30      const auto data = flatten(inst);
31
32      // If key is too short, we pad it with 0
33      std::string padded_key = key;
34      if (key.size() < 32)
35          padded_key.resize(32, 0);
36      else if (key.size() > 32)
37          padded_key.resize(32);
38
39      // Decrypt the instruction
40      EVP_CIPHER_CTX* ctx = EVP_CIPHER_CTX_new();
41      EVP_DecryptInit_ex(
42          ctx, EVP_aes_256_cbc(), NULL,
43          reinterpret_cast<const unsigned char*>(padded_key.data()), NULL);
44      EVP_DecryptUpdate(ctx, reinterpret_cast<unsigned char*>(&inst), NULL,
45                        reinterpret_cast<const unsigned char*>(data.data()),
46                        sizeof(inst));
47      EVP_DecryptFinal_ex(ctx, reinterpret_cast<unsigned char*>(&inst), NULL);
48      EVP_CIPHER_CTX_free(ctx);
49  }
```



TEAMT5
杜 浦 數 位 安 全



```
square:
    push rbp
    call    _Z13VMPilot_BeginPKc    ; VMPilot_Begin(__FUNCTION__);
    ... garbage code ...
    ... garbage code ...
    ... garbage code ...
    call    _Z11VMPilot_EndPKc      ; VMPilot_End(__FUNCTION__);
    pop rbp
    ret
```

Proofs

Proofs

TEAMT5
杜 浦 數 位 安 全

Formal verification

> Axiomatic and Algebraic semantics

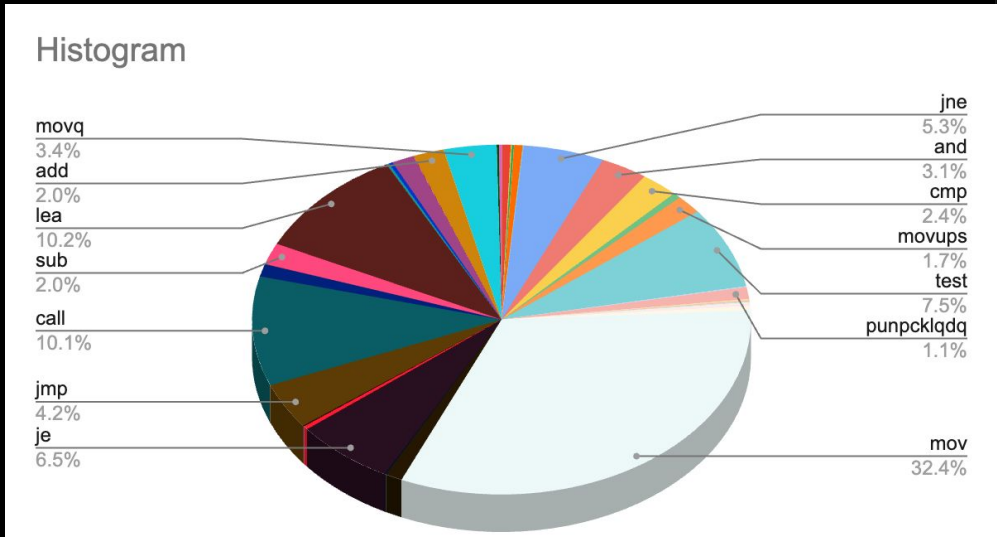
Perfect secrecy

Random oracle

Perfect secrecy

Side channel attack of frequency analysis

```
→ ObfuscatedCalls git:(master) ✗ objdump -d main.o | sed -n -E 's/^\s*[0-9a-f]+:\s+([0-9a-f]{2}\s+){1,15}([0-9a-f]{2})\s+([^\t]+)/\3/p' | sed -n 's/\(\w\+\).*/\1/p' | awk '{ count[$
0]++ } END { for (instruction in count) print instruction, count[instruction] }'

shl 2
movdqu 42
shr 4
cs 11
pop 41
jle 3
jne 385
and 224
cmp 178
push 38
movups 125
test 552
jae 5
punpcklqdq 82
sar 4
movhps 1
xchg 10
or 6
imul 1
cmpb 25
cltq 25
pshufd 1
movabs 1
mov 2372
nopl 80
data16 1
jb 5
00 18
je 474
nopw 23
03 1
movslq 7
jmp 310
call 739
movb 84
movd 2
sub 145
lea 749
jbe 1
movl 12
movaps 18
xor 106
adc 1
pxor 1
add 146
movq 250
movdqa 11
movzbl 12
→ ObfuscatedCalls
```



Perfect secrecy

In 1946, Claude Shannon introduced the idea of perfect secrecy, which became a cornerstone in modern cryptography. [1, 2]

在金鑰空間K內任取一個金鑰k_i, k_j，加密方式為E，m為隨機明文，c為隨機密文則概率關係有

$$P(E(m, k_i) = c) = P(E(m, k_j) = c)$$

該處c, m為相同文段。

也可定義為：

任意$x \in P, y \in C$，有$P(x) = P(x|y)$。

即通過觀察密文無法得到關於明文的任何資訊。

或是： 一組在(K, M, C)上的密碼系統(D, E)滿足

$\forall m_0, m_1 \in M(len(m_0) = len(m_1))$ 且 $\forall c \in C$

$Pr[E(k_0, m_0) = c] = Pr[E(k_1, m_1) = c]$

其中 k_0, k_1 是由 K 當中以完全均等的機率隨機取樣

(註：$Pr[$某事件$]$ 表示該事件發生的機率，K, M, C 各為密鑰、明文及密文空間，D, E 為解密與加密函式)

[1] https://doi.org/10.1016/B978-1-59749-969-9.00008-0
[2] https://doi.org/10.1016/B978-0-12-802459-1.00011-7

Perfect secrecy

TEAMT5
杜 浦 數 位 安 全

Stream cipher or Block encryption?

No, instruction set architecture is the best practice place!

S. W. Boyd, G. S. Kc, M. E. Locasto, A. D. Keromytis and V. Prevelakis, "On the General Applicability of Instruction-Set Randomization," in IEEE Transactions on Dependable and Secure Computing, vol. 7, no. 3, pp. 255-270, July-Sept. 2010, doi: 10.1109/TDSC.2008.58.

Recall

TEAMT5 杜浦數位安全

```
11  struct Instruction_t {
12      // Instruction_t Data layout:
13      //
14      // (Opcode:          16 bits)
15      // (left operand:    64 bits)
16      // (right operand:   64 bits)
17      // (Nounce:          32 bits)
18      // (Checksum:        16 bits)
19      //
20
21      uint16_t opcode;
22      uint64_t left_operand;
23      uint64_t right_operand;
24      uint32_t nounce;
25      uint16_t checksum;
26  };
```

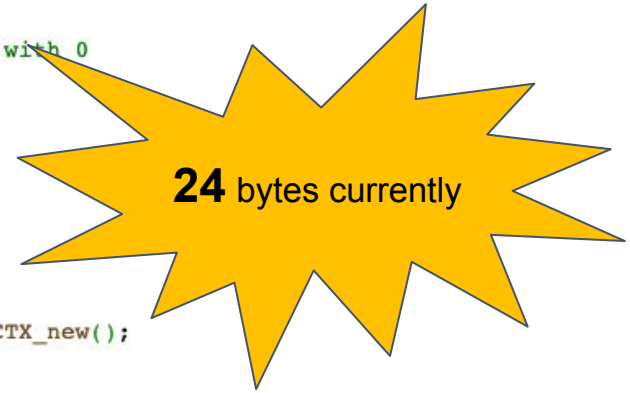
```
26  void VMPilot::Common::Instruction::decrypt(Instruction_t& inst,
27                                             const std::string& key) noexcept {
28      // Decrypt the instruction using the key
29      // We use AES-256-CBC from the OpenSSL library to decrypt the instruction.
30      const auto data = flatten(inst);
31
32      // If key is too short, we pad it with 0
33      std::string padded_key = key;
34      if (key.size() < 32)
35          padded_key.resize(32, 0);
36      else if (key.size() > 32)
37          padded_key.resize(32);
38
39      // Decrypt the instruction
40      EVP_CIPHER_CTX* ctx = EVP_CIPHER_CTX_new();
41      EVP_DecryptInit_ex(
42          ctx, EVP_aes_256_cbc(), NULL,
43          reinterpret_cast<const unsigned char*>(padded_key.data()), NULL);
44      EVP_DecryptUpdate(ctx, reinterpret_cast<unsigned char*>(&inst), NULL,
45                        reinterpret_cast<const unsigned char*>(data.data()),
46                        sizeof(inst));
47      EVP_DecryptFinal_ex(ctx, reinterpret_cast<unsigned char*>(&inst), NULL);
48      EVP_CIPHER_CTX_free(ctx);
49  }
```

Recall

TEAMT5 杜浦數位安全

```
11  struct Instruction_t {
12      // Instruction_t Data layout:
13      //
14      // (Opcode:          16 bits)
15      // (left operand:    64 bits)
16      // (right operand:   64 bits)
17      // (Nounce:          32 bits)
18      // (Checksum:        16 bits)
19      //
20
21      uint16_t opcode;
22      uint64_t left_operand;
23      uint64_t right_operand;
24      uint32_t nounce;
25      uint16_t checksum;
26  };
```

```
26  void VMPilot::Common::Instruction::decrypt(Instruction_t& inst,
27                                             const std::string& key) noexcept {
28      // Decrypt the instruction using the key
29      // We use AES-256-CBC from the OpenSSL library to decrypt the instruction.
30      const auto data = flatten(inst);
31
32      // If key is too short, we pad it with 0
33      std::string padded_key = key;
34      if (key.size() < 32)
35          padded_key.resize(32, 0);
36      else if (key.size() > 32)
37          padded_key.resize(32);
38
39      // Decrypt the instruction
40      EVP_CIPHER_CTX* ctx = EVP_CIPHER_CTX_new();
41      EVP_DecryptInit_ex(
42          ctx, EVP_aes_256_cbc(), NULL,
43          reinterpret_cast<const unsigned char*>(padded_key.data()), NULL);
44      EVP_DecryptUpdate(ctx, reinterpret_cast<unsigned char*>(&inst), NULL,
45                        reinterpret_cast<const unsigned char*>(data.data()),
46                        sizeof(inst));
47      EVP_DecryptFinal_ex(ctx, reinterpret_cast<unsigned char*>(&inst), NULL);
48      EVP_CIPHER_CTX_free(ctx);
49  }
```



Recall the definition

在金鑰空間K內任取一個金鑰k_i, k_j，加密方式為E，m為隨機明文，c為隨機密文則概率關係有

$$P(E(m, k_i) = c) = P(E(m, k_j) = c)$$

該處c, m為相同文段。

也可定義為：

任意$x \in P, y \in C$，有$P(x) = P(x|y)$。

即通過觀察密文無法得到關於明文的任何資訊。

或是： 一組在(K, M, C)上的密碼系統(D, E)滿足

$\forall m_0, m_1 \in M(len(m_0) = len(m_1))$ 且 $\forall c \in C$

$Pr[E(k_0, m_0) = c] = Pr[E(k_1, m_1) = c]$

其中 k_0, k_1 是由 K 當中以完全均等的機率隨機取樣

(註：$Pr[$某事件$]$ 表示該事件發生的機率，K, M, C 各為密鑰、明文及密文空間，D, E 為解密與加密函式)

Recall the definition

在金鑰空間K內任取一個金鑰k_i, k_j，加密方式為E，m為隨機明文，c為隨機密文則概率關係有

$$P(E(m, k_i) = c) = P(E(m, k_j) = c)$$

該處c, m為相同文段。

也可定義為：

任意$x \in P, y \in C$，有$P(x) = P(x|y)$。

即通過觀察密文無法得到關於明文的任何資訊。

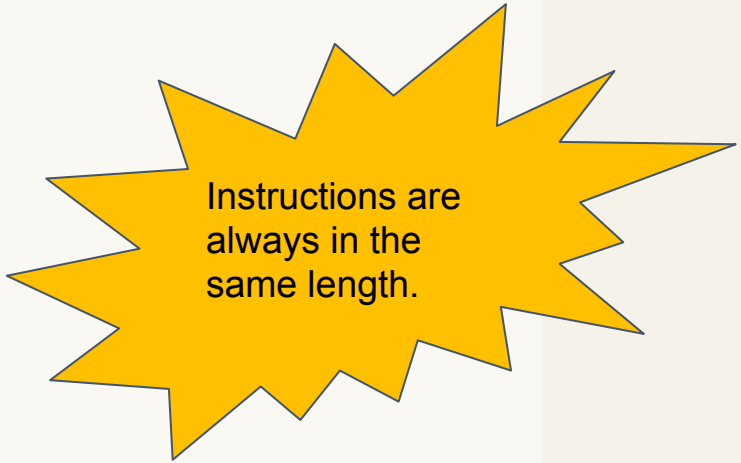
或是：一組在(K, M, C)上的密碼系統(D, E)滿足

$$\forall m_0, m_1 \in M(len(m_0) = len(m_1)) \text{ 且 } \forall c \in C$$

$$Pr[E(k_0, m_0) = c] = Pr[E(k_1, m_1) = c]$$

其中 k_0, k_1 是由 K 當中以完全均等的機率隨機取樣

(註：$Pr[\text{某事件}]$ 表示該事件發生的機率，K, M, C 各為密鑰、明文及密文空間，D, E 為解密與加密函式)



Recall the definition

在金鑰空間K內任取一個金鑰k_i, k_j，密方式為E，m為隨機明文，c為隨機密文則概率關係有

$$P(E(m, k_i) = c) =$$

該處c, m為相同文

也可定義為：

任意$x \in P, y \in C$，有$P($

即通過觀察密文無法得到關於明文何資訊

或是：一組在(K, M, C)上的密碼系統(D, E)滿足

$$\forall m_0, m_1 \in M(len(m_0) = len(m_1)) \text{ 且 } \forall c \in C$$

$$Pr[E(k_0, m_0) = c] = Pr[E(k_1, m_1) = c]$$

其中 k_0, k_1 是由 K 當中以完全均等的機率隨機取樣

(註：$Pr[\text{某事件}]$ 表示該事件發生的機率，K, M, C 各為密鑰、明文及密文空間，D, E 為解密與加密函式)

But we use a symmetric cryptography.

Instructions are always in the same length.

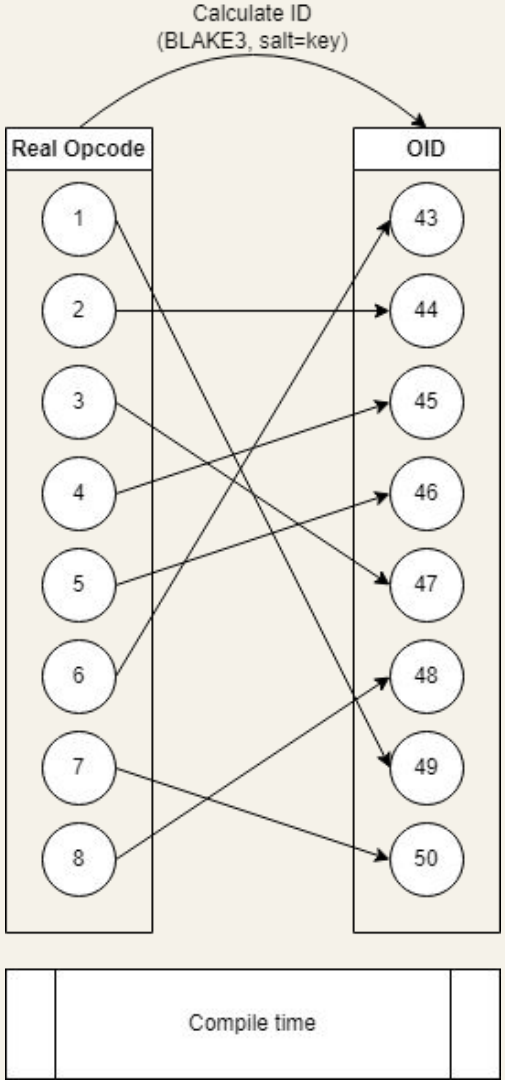
Random oracle

TEAMT5 杜浦數位安全

在密碼學裡面，**隨機預言機**（英語：Random oracle）是一部預言機（可以理解為理論的黑箱），對任何輸入都回傳一個真正均勻隨機的輸出（請參考離散型均勻分佈），不過對相同的輸入，該預言機每次都會用同一方法輸出。換句話說，隨機預言機是一個將所有可能輸入與輸出作隨機映射的函數。

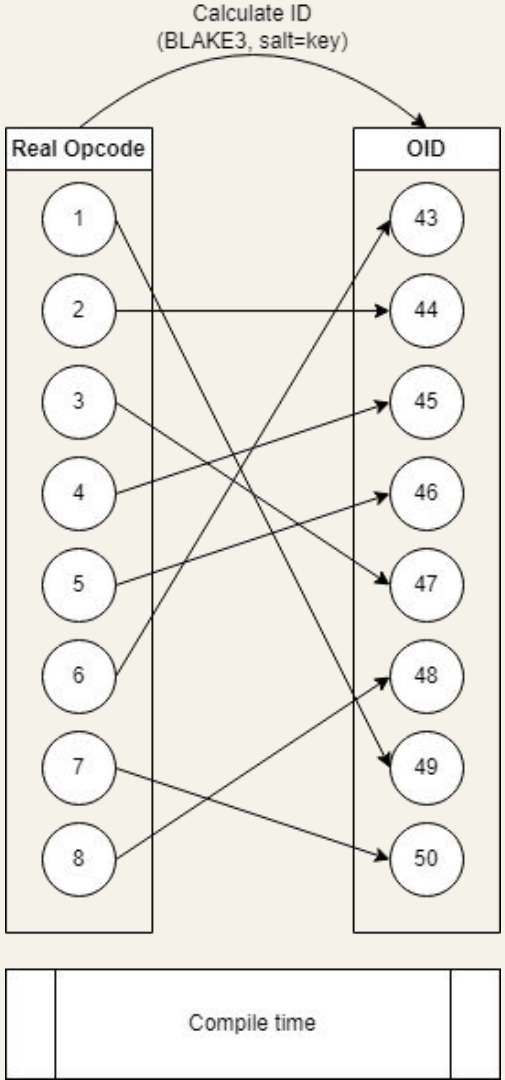
Claim 1: The key is **selected** from a **uniform** distribution random Oracle space.

Proof: We assume that the key used in the cryptographic scheme is selected from a random Oracle space with a uniform distribution. This means that each possible key is equally likely to be chosen, ensuring randomness in the selection process.



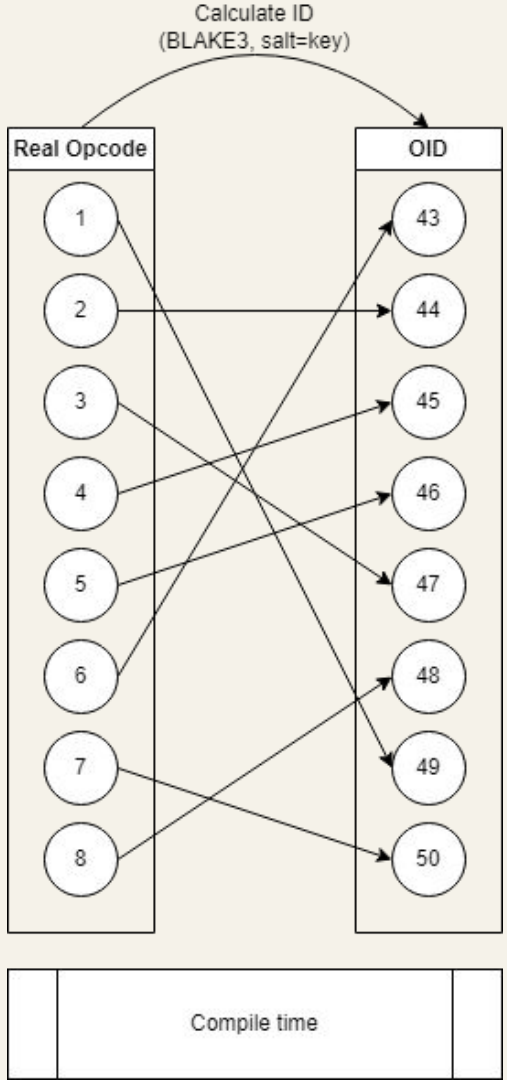
Claim 2: The key is inputted into an **one-way hash** function.

Proof: The selected key is used as an input to an one-way hash function. An one-way hash function takes an input and produces a fixed-size output, such that it is computationally infeasible to retrieve the original input from the output.



Claim 3: The cryptographic scheme **exhibits properties** of a random oracle.

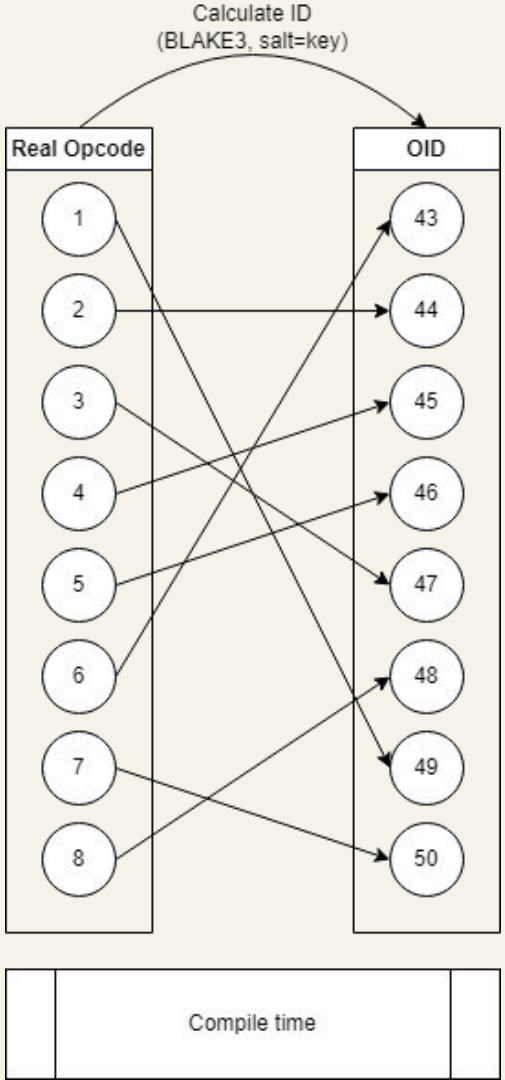
Proof: Based on Claims 1 and 2, we can conclude that the cryptographic scheme approximates a random oracle. A random oracle is a theoretical black box that provides random and independent outputs for each unique input. In this case, the one-way hash function, when applied with the selected key, generates bytes output that appears random and unpredictable. As long as the key remains confidential and the hash function remains secure, the scheme's outputs are indistinguishable from random strings, exhibiting properties of a random oracle.



TEAMT5
杜 浦 數 位 安 全

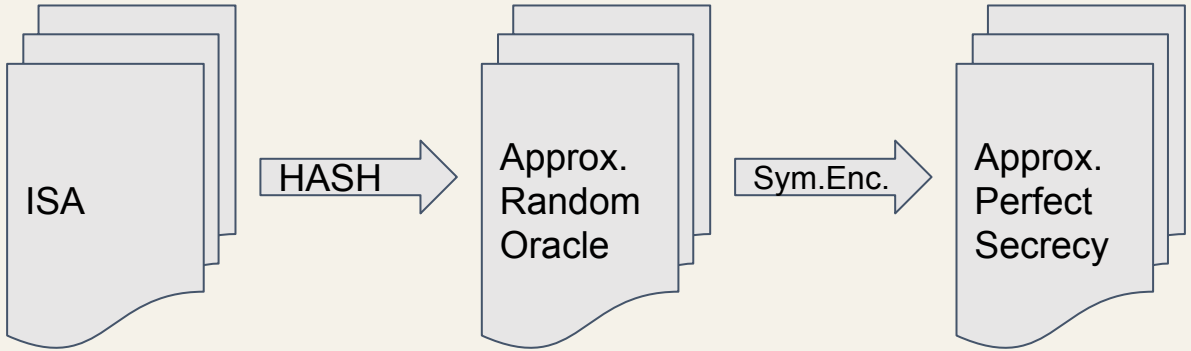
Conclusion: **This cryptographic scheme is approximately a random oracle.**

The provided proof demonstrates that the cryptographic scheme, utilizing a key selected from a uniform distribution random Oracle space and an one-way hash function, exhibits properties similar to a random oracle. While it may not perfectly emulate a random oracle due to potential vulnerabilities or weaknesses in the hash function, it approximates its behavior, providing random and unpredictable outputs for each unique input.

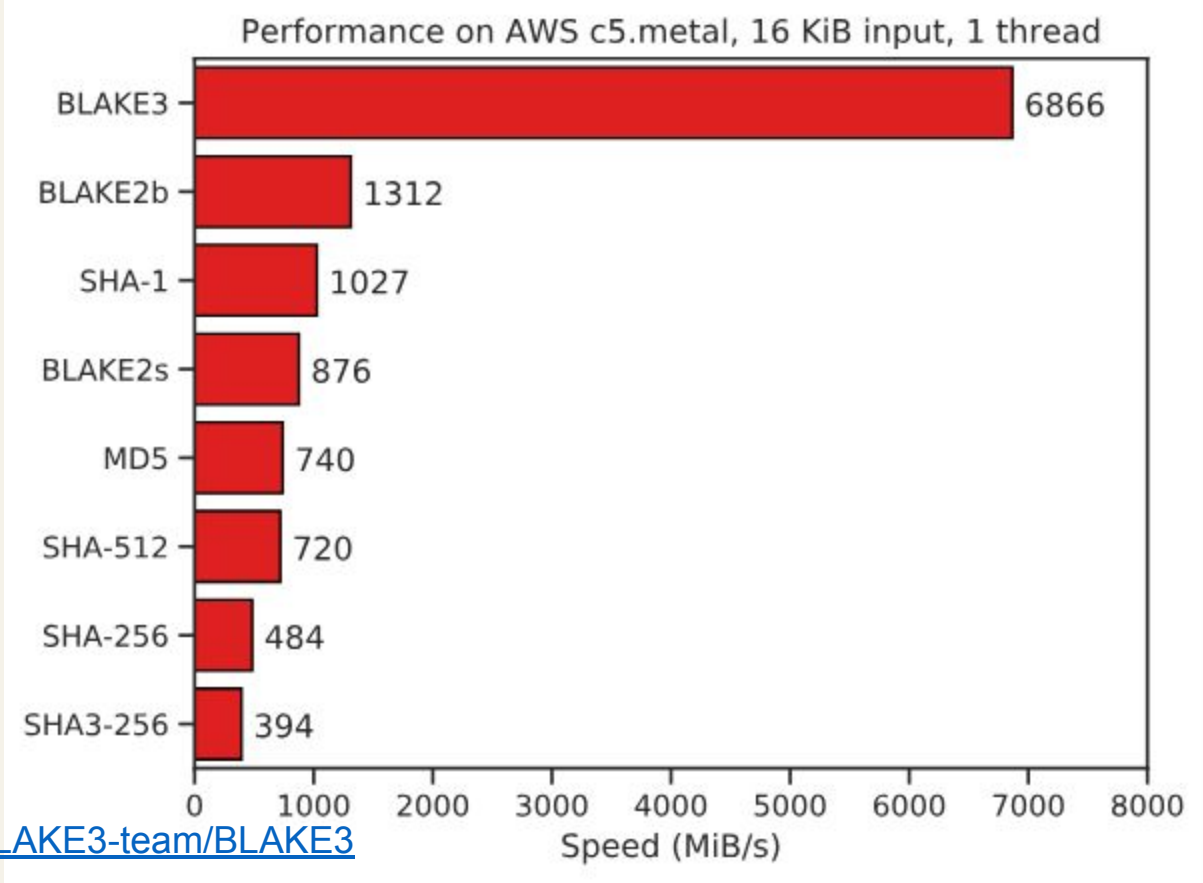


Cryptographic scheme

TEAMT5
杜 浦 數 位 安 全



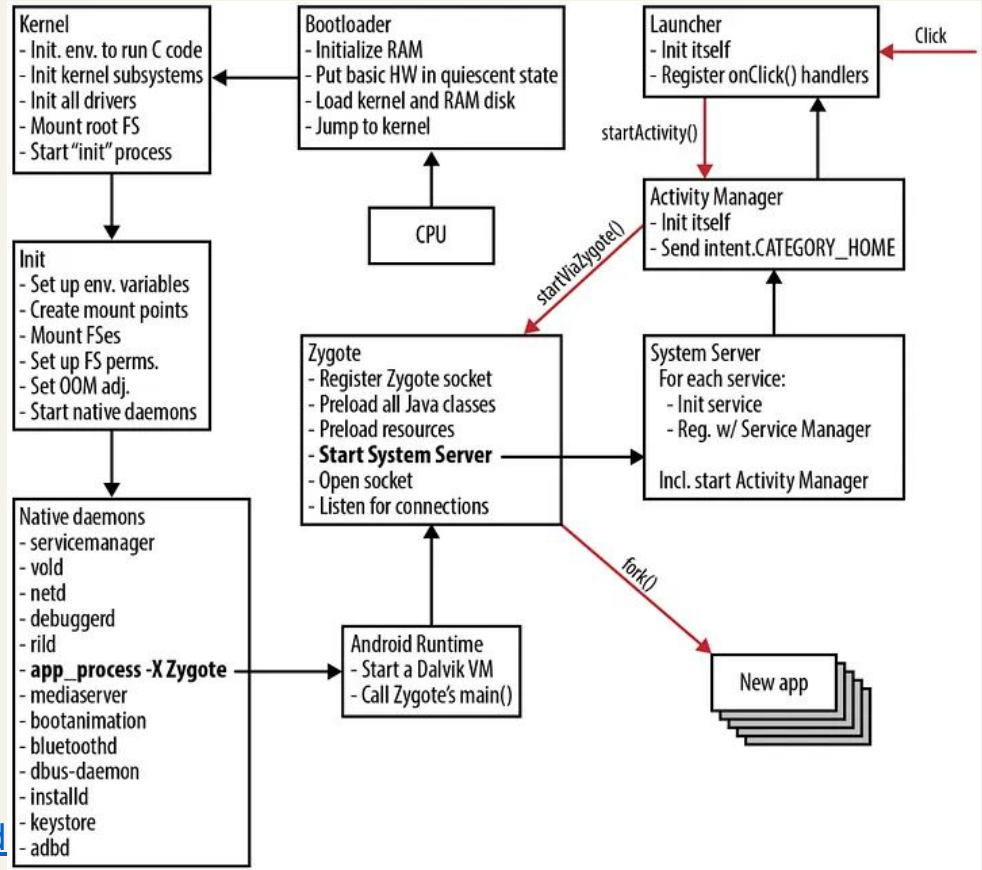
Perfect hash function?



https://github.com/BLAKE3-team/BLAKE3

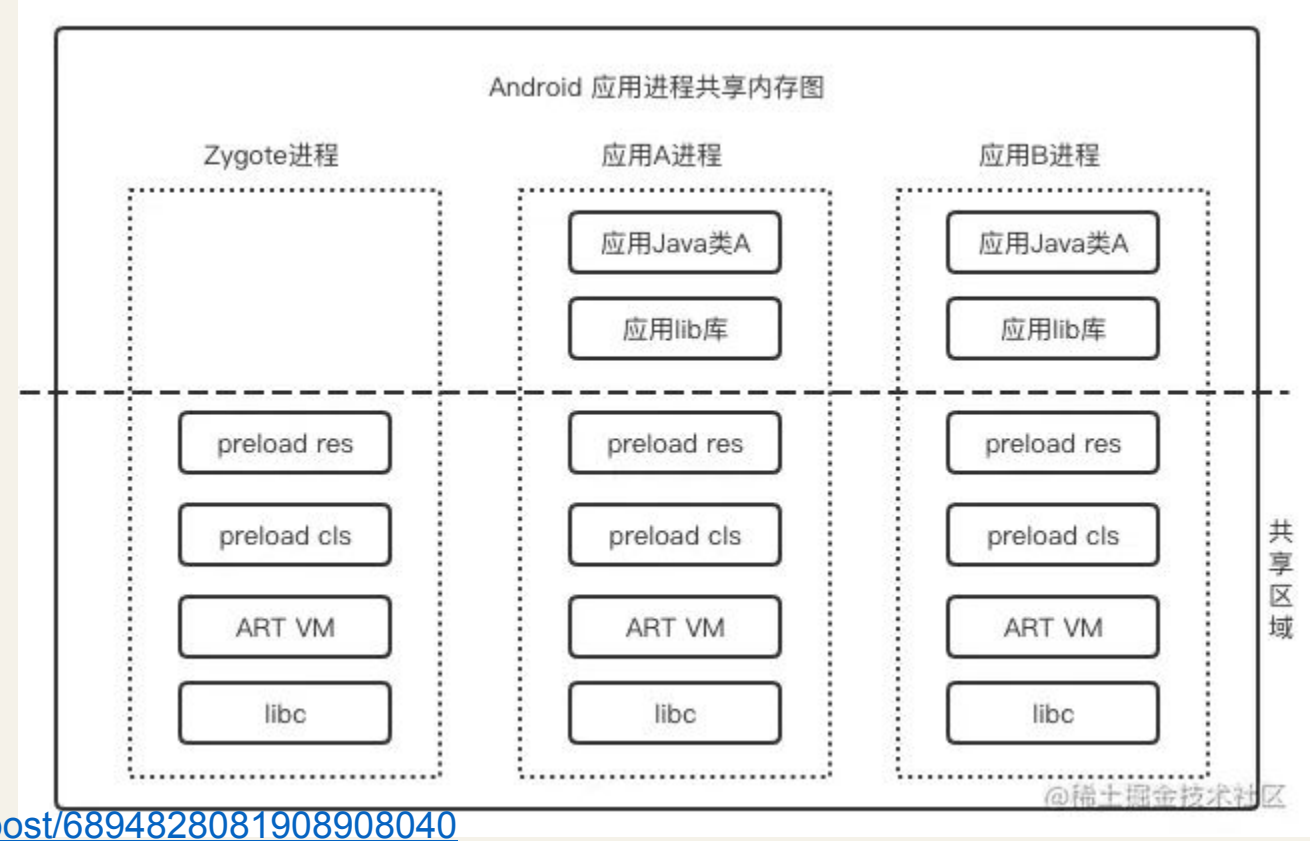
Mechanism Run time

Zygote in Android



https://medium.com/@vood

Share memory



https://juejin.cn/post/6894828081908908040

Producer-Consumer

TEAMT5
杜 浦 數 位 安 全

```
VMPilot_Begin("A");
auto result = x * x;
VMPilot_End("A");
```

- Get pid, tid…
- Parse stack frame
- Conditional notify
- Send stack info
- Fetch data
- Send data
- …
- Move PC

https://www.flaticon.com/free-icon/machine_9780637?term=virtual+machine&related_id=9780637

Producer-Consumer

TEAMT5
杜 浦 數 位 安 全

```
VMPilot_Begin("A");
auto result = x * x;
VMPilot_End("A");
```

- Get pid, tid…
- Parse stack frame
- Conditional notify
- Send stack info
- Clean up

https://www.flaticon.com/free-icon/machine_9780637?term=virtual+machine&related_id=9780637

Another function call?

TEAMT5
杜 浦 數 位 安 全

```
VMPilot_Begin("A");
auto result = foo(x);
VMPilot_End("A");
```

- Push vm state
- Push trap function
- Find function addr
- Call it
- Entrance trap function
- Continue producer-consumer
- Move PC

https://www.flaticon.com/free-icon/machine_9780637?term=virtual+machine&related_id=9780637

Multi-threading

TEAMT5
杜 浦 數 位 安 全

```
VMPilot_Begin("A1");
auto result = ...;
VMPilot_End("A1");

VMPilot_Begin("A2");
auto result = ...;
VMPilot_End("A2");

VMPilot_Begin("A3");
auto result = ...;
VMPilot_End("A3");
```

- Get pid, tid…
- Parse stack frame
- **Conditional notify**
- **Send stack info**
- Fetch data
- **Send data**
- …
- Move PC

https://www.flaticon.com/free-icon/machine_9780637?term=virtual+machine&related_id=9780637

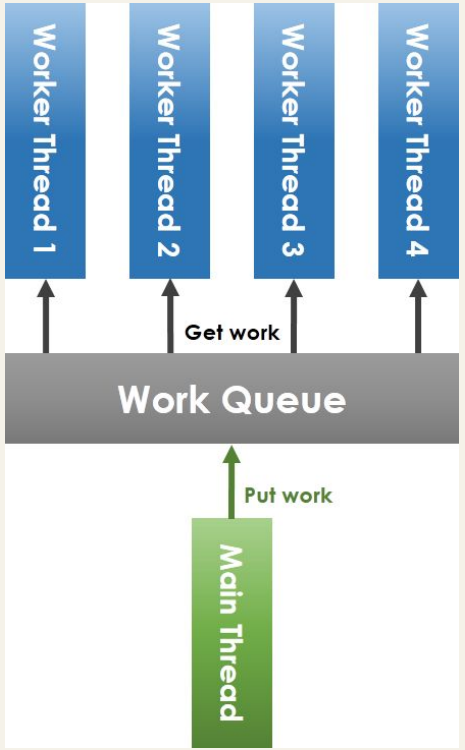
Multi-threading

TEAMT5 杜浦數位安全

Like CMWQ in kernel

```
VMPilot_Begin("A1");
auto result = ...;
VMPilot_End("A1");

VMPilot_Begin("A2");
auto result = ...;
VMPilot_End("A2");

VMPilot_Begin("A3");
auto result = ...;
VMPilot_End("A3");
```

- Get pid, tid…
- Parse stack frame
- **Conditional notify**
- **Send stack info**
- Fetch data
- **Send data**
- …
- Move PC



https://hackmd.io/@sysprog/linux-io-model/https%3A%2F%2Fhackmd.io%2F%40sysprog%2Ffast-web-server

Inner threading

TEAMT5
杜 浦 數 位 安 全

```
VMPilot_Begin("A");
auto f =[](...) {};
std::thread t(f);
t.join();
VMPilot_End("A");
```

Thinking …

https://www.flaticon.com/free-icon/machine_9780637?term=virtual+machine&related_id=9780637

Tiny demo

Remaining issues

TEAMT5
杜 浦 數 位 安 全

Segmentator	Runtime lib	TODO	Further
Junk code ISA extension	Producer-consumer Pass CET (Zero hacks RIP)	x86 seg. first LLVM IR backend Loader Runtime Win/Lin first	FFI More arch More platform

Open to pull requests.

https://github.com/25077667/VMPilot

Thank you.

scc@teamt5.org