KCon洞见未来 2017

KCon

探索虚拟化技术在漏洞检测中的应用

By:仙果

目录
CONTENTS

01 自我介绍

我是谁？

仙果
安全研究员

看雪学院

准爸爸

高级安全工程师

02 困境的“城内和城外”

困境

- 针对性的检测漏洞攻击
- 时间和效率的考量
- 人工分析的难度较大
- 人才培养周期长
- 攻防对抗成本显著增加

虚拟化（Virtualization）

虚拟化技术应用于漏洞攻防对抗

精确对抗漏洞攻击

节省人力成本

更多的时间在技术研究上

Virtualzation

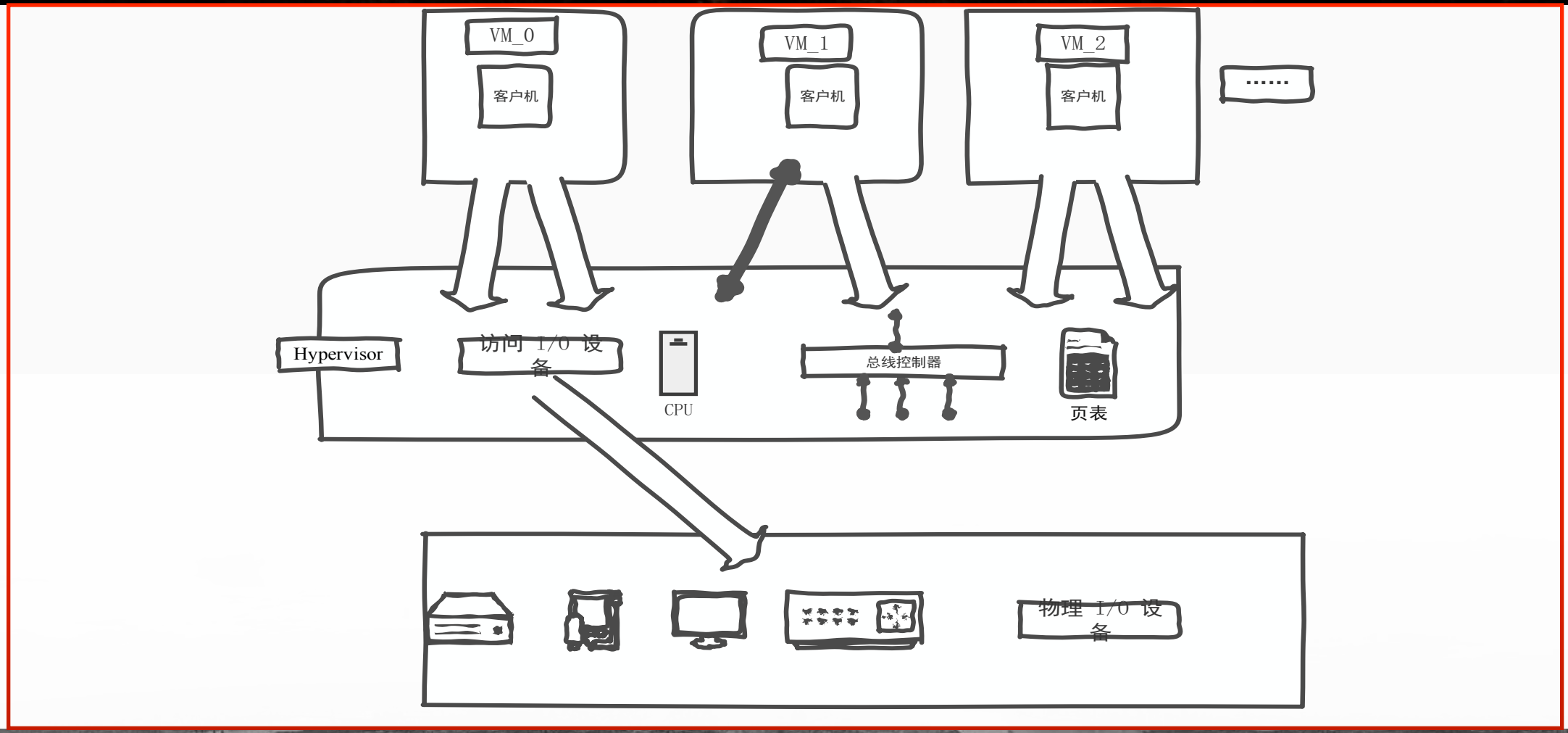
对计算机资源抽象

隐藏了系统、应用程序员和终端用户赖以交互的计算机资源物理性的一面，把单一物理资源转换为多个逻辑资源，反之亦可。

Hardware Enabled Virtualzation

硬件虚拟化技术

硬件层面上，CPU对虚拟化技术提供直接支持，提高虚拟效率，降低开发难度；硬件上实现内存地址甚至是I/O设备的映射，支持二次寻址。


VM_0
客户机
VM_1
客户机
VM_2
客户机
……
Hypervisor
访问 I/O 设备
CPU
总线控制器
页表
物理 I/O 设备


虚拟机 1
虚拟机 2
虚拟机 3
应用程序
操作系统
应用程序
操作系统
应用程序
操作系统
HEV/Hypervisor
物理设备层

Intel VT

Intel Virtualzation Technology VT-x/VT-d

为每个虚拟机提供虚拟处理器
VMM层可以控制处理器资源，物理内存，管理中断和I/O操作
各虚拟机采用相同接口处理虚拟机设备
每个虚拟机可以独立运行，相互独立
VMM层对虚拟机完全透明

AMD SVM

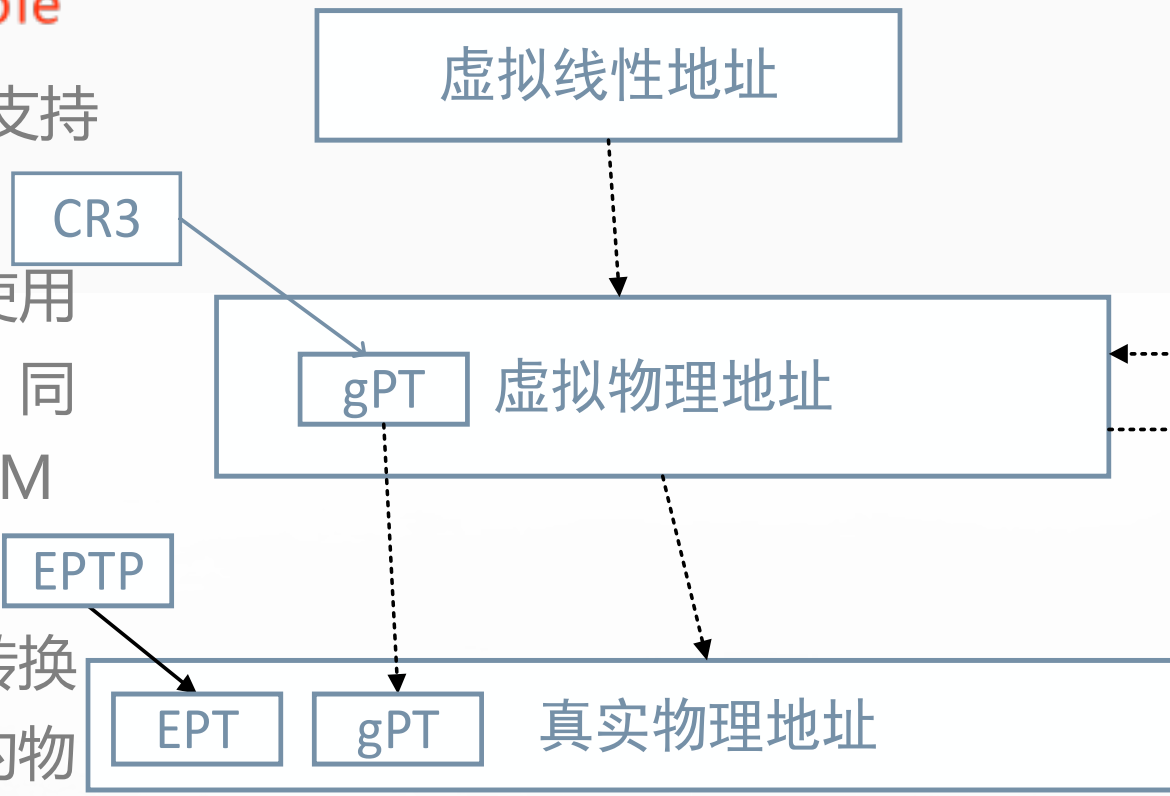
AMD Secure Virtual Machine

客户机模式
Hypervisor 与Guest 直接快速切换
中断Guest 中特定的指令或事件
DMA保护
虚拟中断
嵌套页实现地址翻译
TLB(Translation Lookaside Buffer)减少性能下降

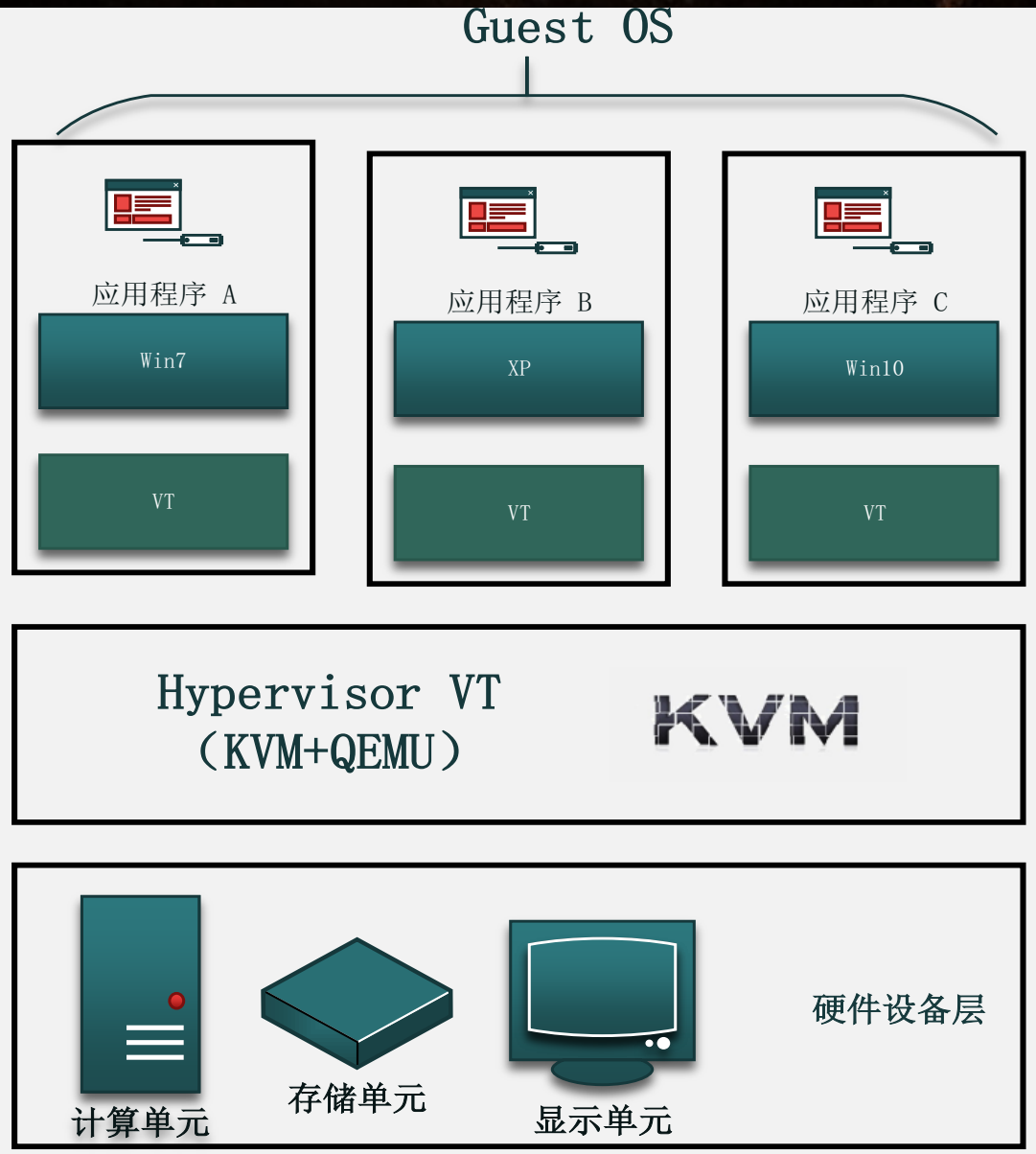
Intel VT---EPT Intel VT Extended Page Table

intel CPU在处理器级别加入对内存虚拟化的支持，支持两级地址翻译

一个逻辑CPU处于非根模式下运行客户机代码时，使用的地址是客户机虚拟地址，而访问这个虚拟地址时，同样会发生地址的转换，这里的转换还没有设计到VMM层，和正常的系统一样，依然是采用CR3作为基址，用客户机页表进行地址转换，只是到这里虽然已经转换成物理地址，是客户机物理地址，不等同于宿主机的物理地址，所以并不能直接访问，需要借助于第二次的转换，也就是EPT的转换。



03 VT VS Exploit

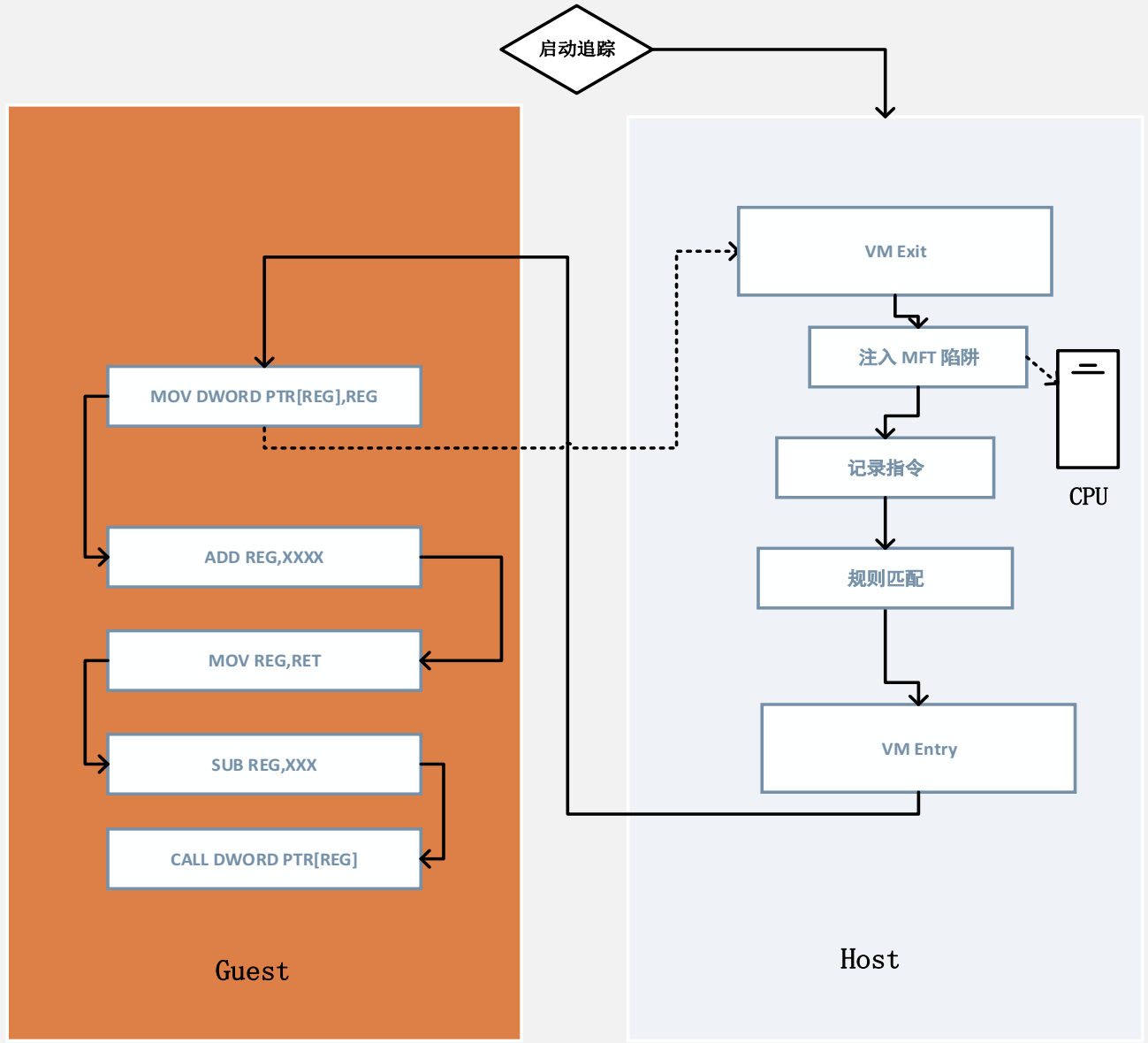


VT 检测框架

硬件各个单元合并虚拟之后，使用KVM 硬件虚拟化框架，进行硬件单元的管理和调度。

KVM使用 Intel VT 技术来提升虚拟化处理性能，在 KVM 上层通过嵌套的 nested VT技术，使检测框架运行在 QEMU 虚拟机中。

利用VT技术，从而实现传统技术无法实现的技术。例如MSR HOOK，EPT hook，特权指令监控等。



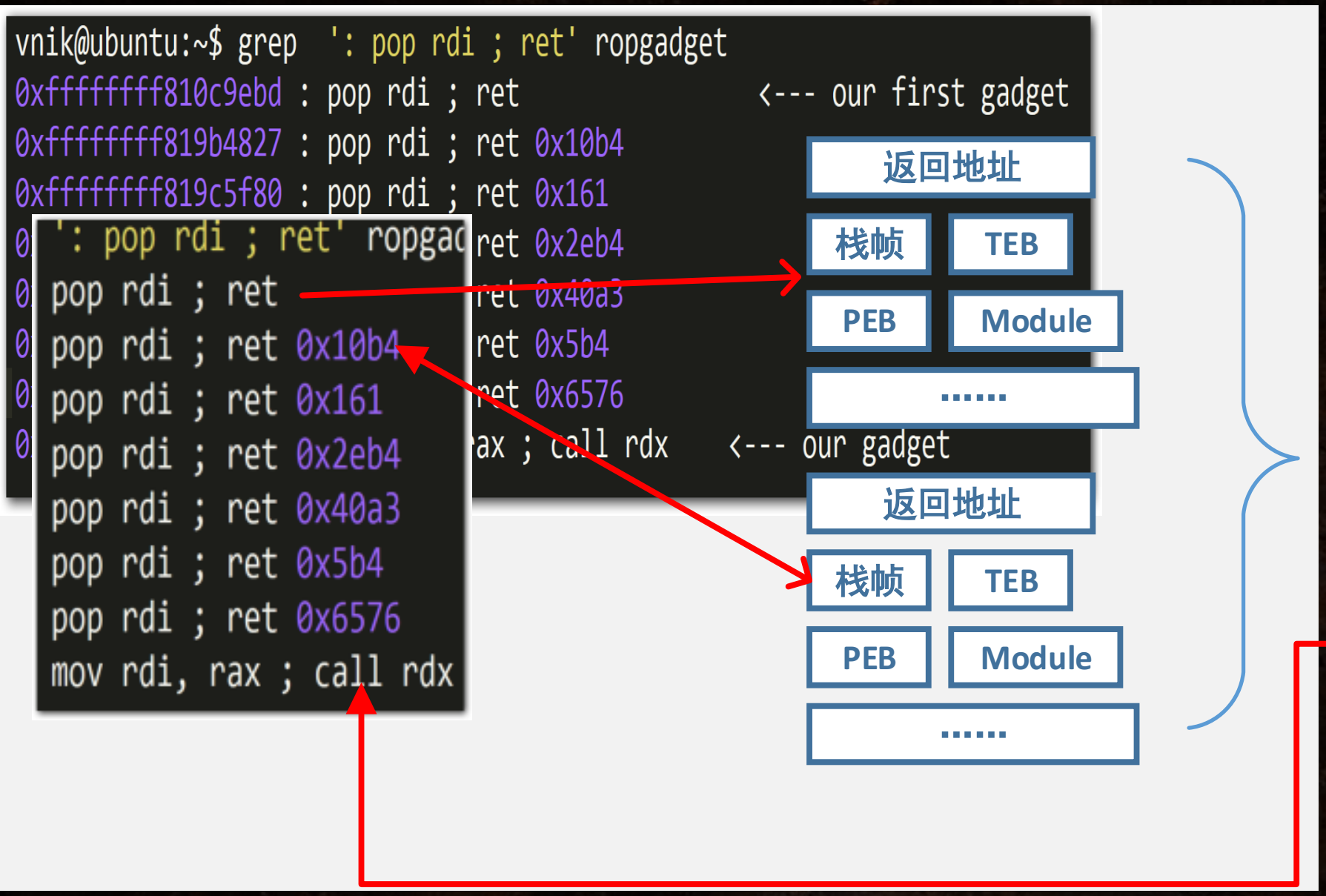
指令追踪流程图

指令追踪通过在 HOST 层针对CPU指令注入MTF陷阱，当CPU 汇编指令执行完毕后，触发 VM Exit 事件，陷入 HOST 处理流程中。

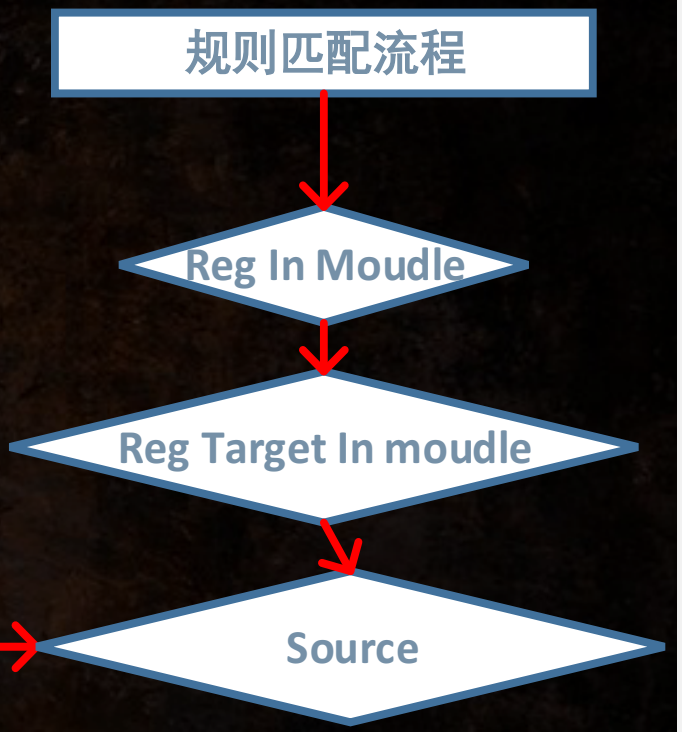
进而能够记录具体的指令流程和当前CPU的运行环境，从而能够进行命中规则的指令流记录。

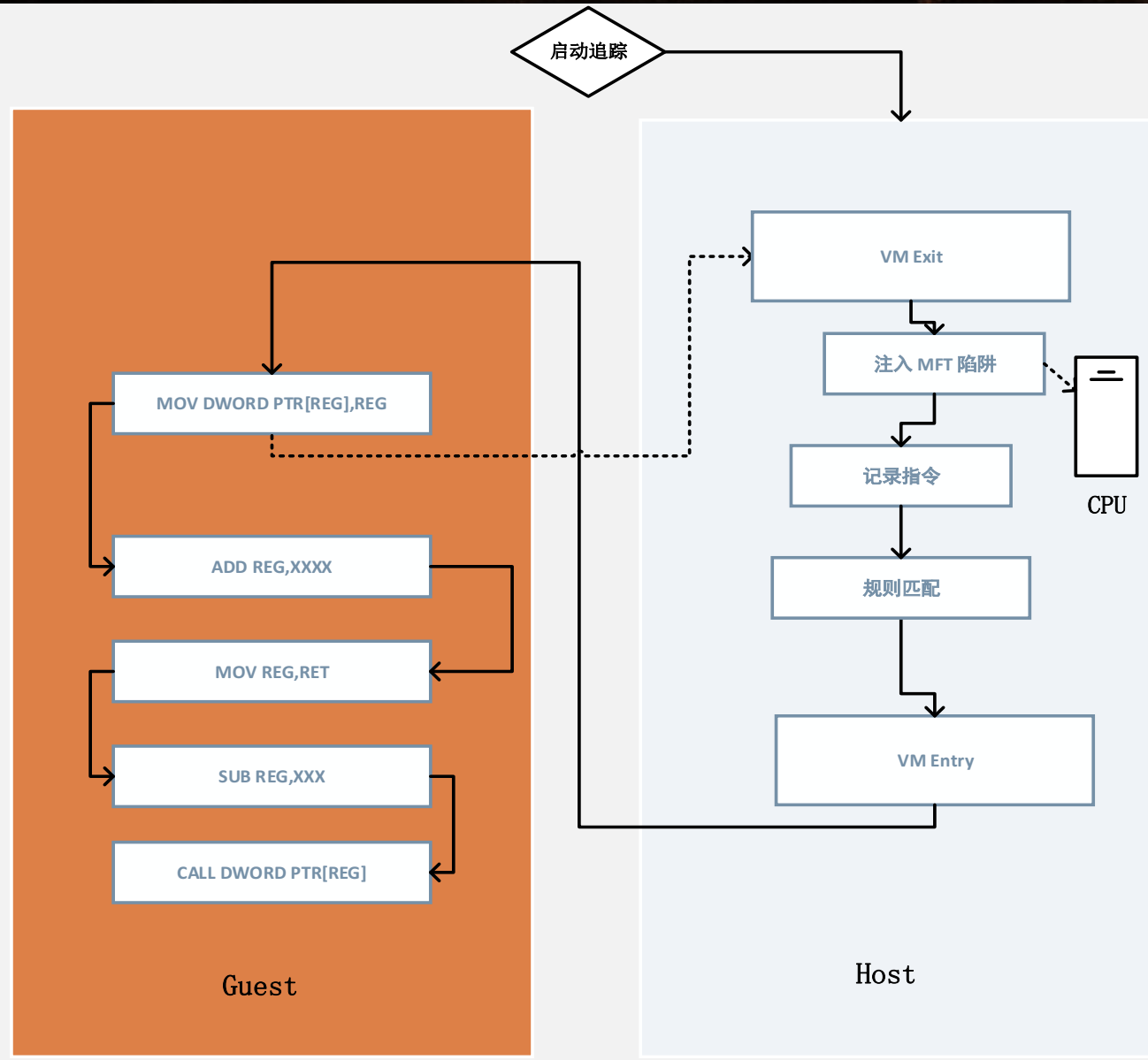
当记录完成之后，就可以进行规则上的匹配，规则匹配完整之后，通过 VMResume 指令产生 VM Entry 事件交回到 Guest进行执行下一条指令。

依次进行循环处理。


vnik@ubuntu:~$ grep ': pop rdi ; ret' ropgadget
0xffffffff810c9ebd : pop rdi ; ret
<--- our first gadget
0xffffffff819b4827 : pop rdi ; ret 0x10b4
0xffffffff819c5f80 : pop rdi ; ret 0x161
ret 0x2eb4
ret 0x40a3
ret 0x5b4
ret 0x6576
ax ; call rdx
<--- our gadget
': pop rdi ; ret' ropgad
pop rdi ; ret
pop rdi ; ret 0x10b4
pop rdi ; ret 0x161
pop rdi ; ret 0x2eb4
pop rdi ; ret 0x40a3
pop rdi ; ret 0x5b4
pop rdi ; ret 0x6576
mov rdi, rax ; call rdx
返回地址
栈帧
TEB
PEB
Module
.....
返回地址
栈帧
TEB
PEB
Module
.....

指令追踪 VS ROP


规则匹配流程
Reg In Moudle
Reg Target In moudle
Source



VT VS 提权

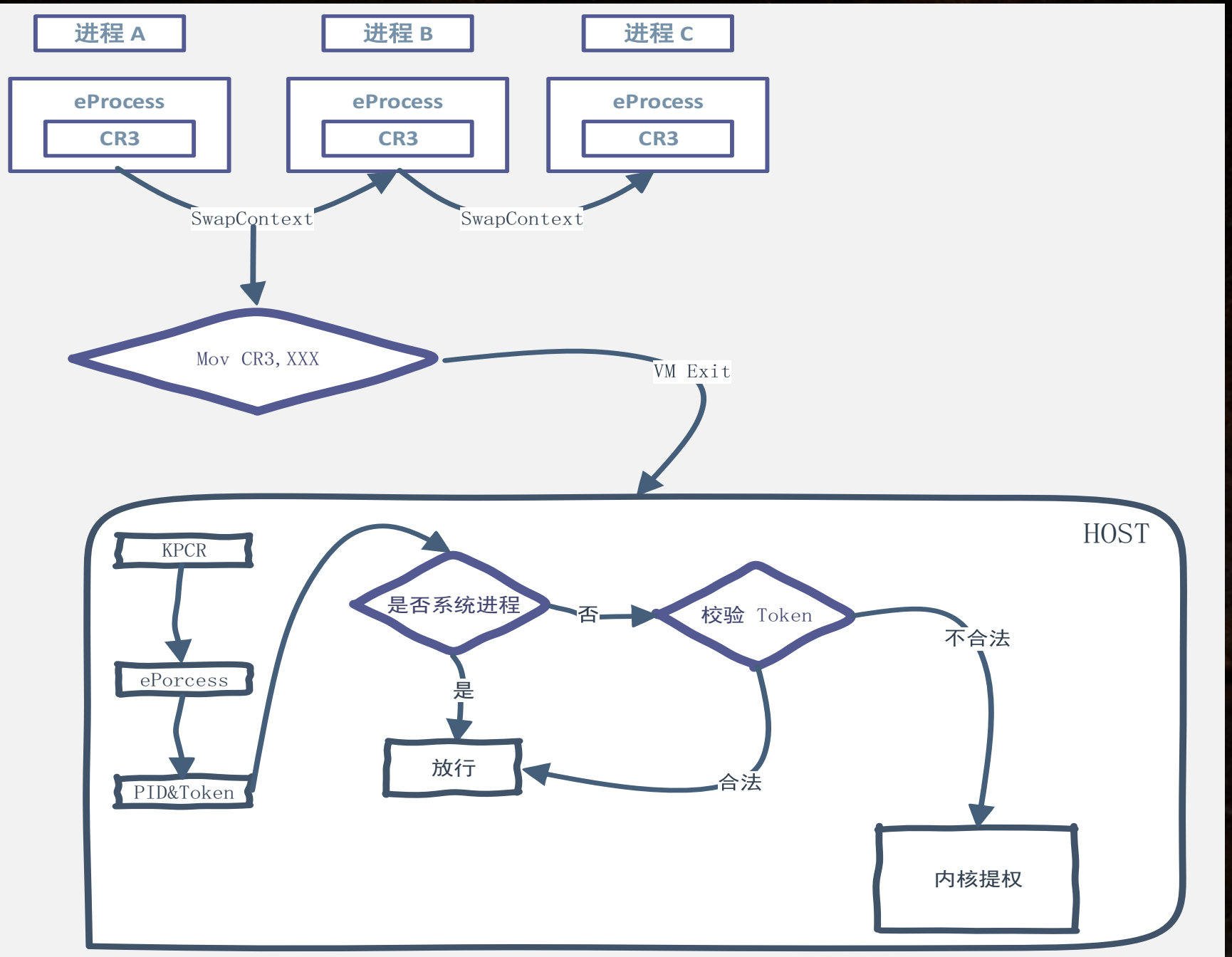
指令追踪通过在 HOST 层针对CPU指令注入MTF陷阱，当CPU 汇编指令执行完毕后，触发 VM Exit 事件，陷入 HOST 处理流程中。

进而能够记录具体的指令流程和当前CPU的运行环境，从而能够进行命中规则的指令流记录。

当记录完成之后，就可以进行规则上的匹配，规则匹配完整之后，通过 VMResume 指令产生 VM Entry 事件交回到Guest进行执行下一条指令。依次进行循环处理。

```
BITS 64

global start

section .text

start:
mov     rax, [gs:0x188]         ;Get current ETHREAD in
mov     rax, [rax+0x68]         ;Get current EPROCESS address
mov     rcx, rax                ;Copy current EPROCESS address to RCX

find_system_process:
mov     rax, [rax+0xe0]         ;Next EPROCESS ActiveProcessLinks.Flink
sub     rax, 0xe0               ;Go to the beginning of the EPROCESS structure
mov     r9 , [rax+0xd8]         ;Copy PID to R9
cmp     r9 , 0x4                ;Compare R9 to SYSTEM PID (=4)
jnz short find_system_process   ;If not SYSTEM got to next EPROCESS

stealing:
mov     rdx, [rax+0x160]        ;Copy SYSTEM process token address to RDX
mov     [rcx+0x160], rdx        ;Steal token with overwriting our current process's token address
retn    0x10
```

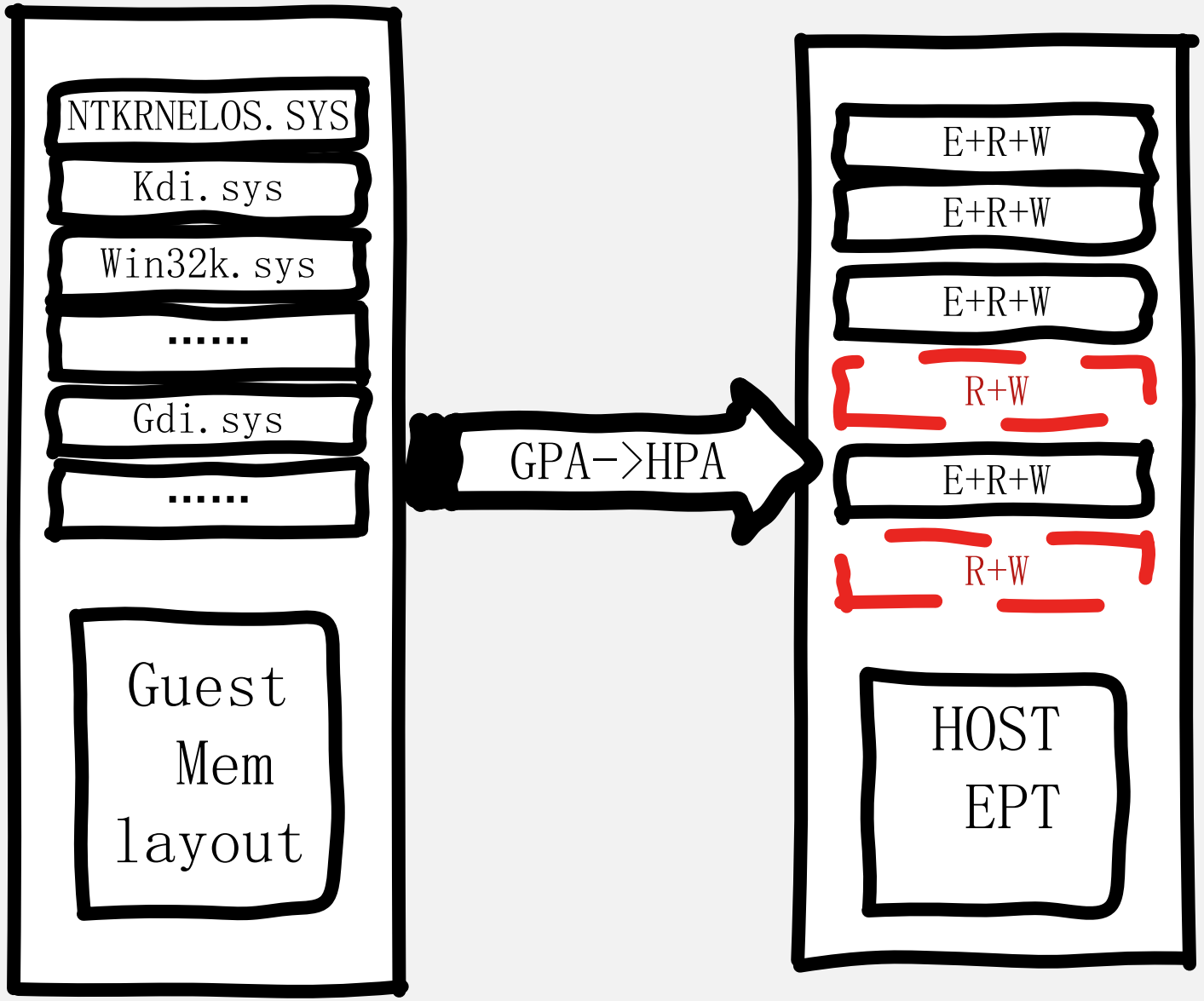
VT VS 提权



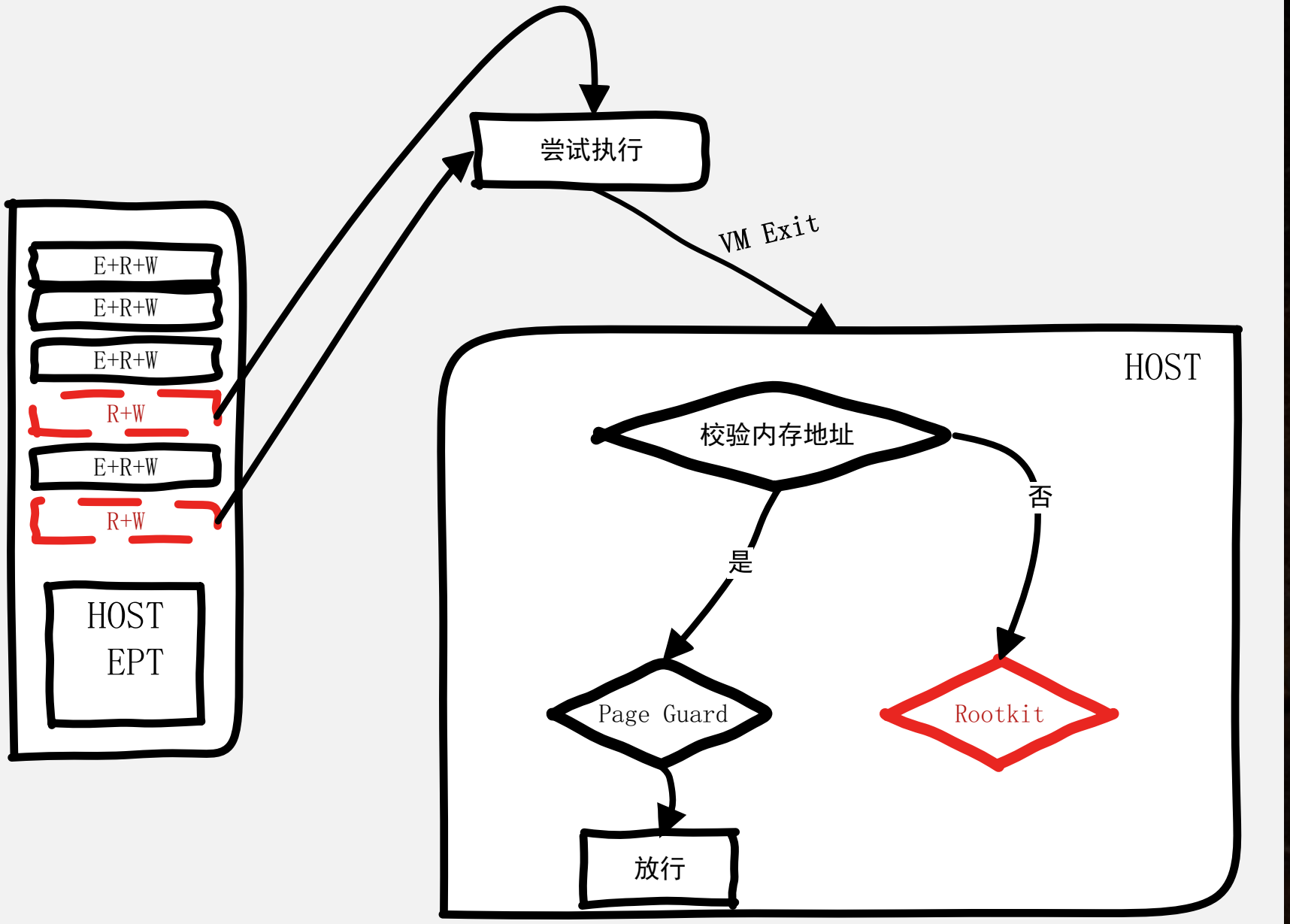
VT VS 提权

操作系统中每个进程都有属于自身权限的Token，漏洞需要提权则要获取到NT system 权限的token，进行覆盖。

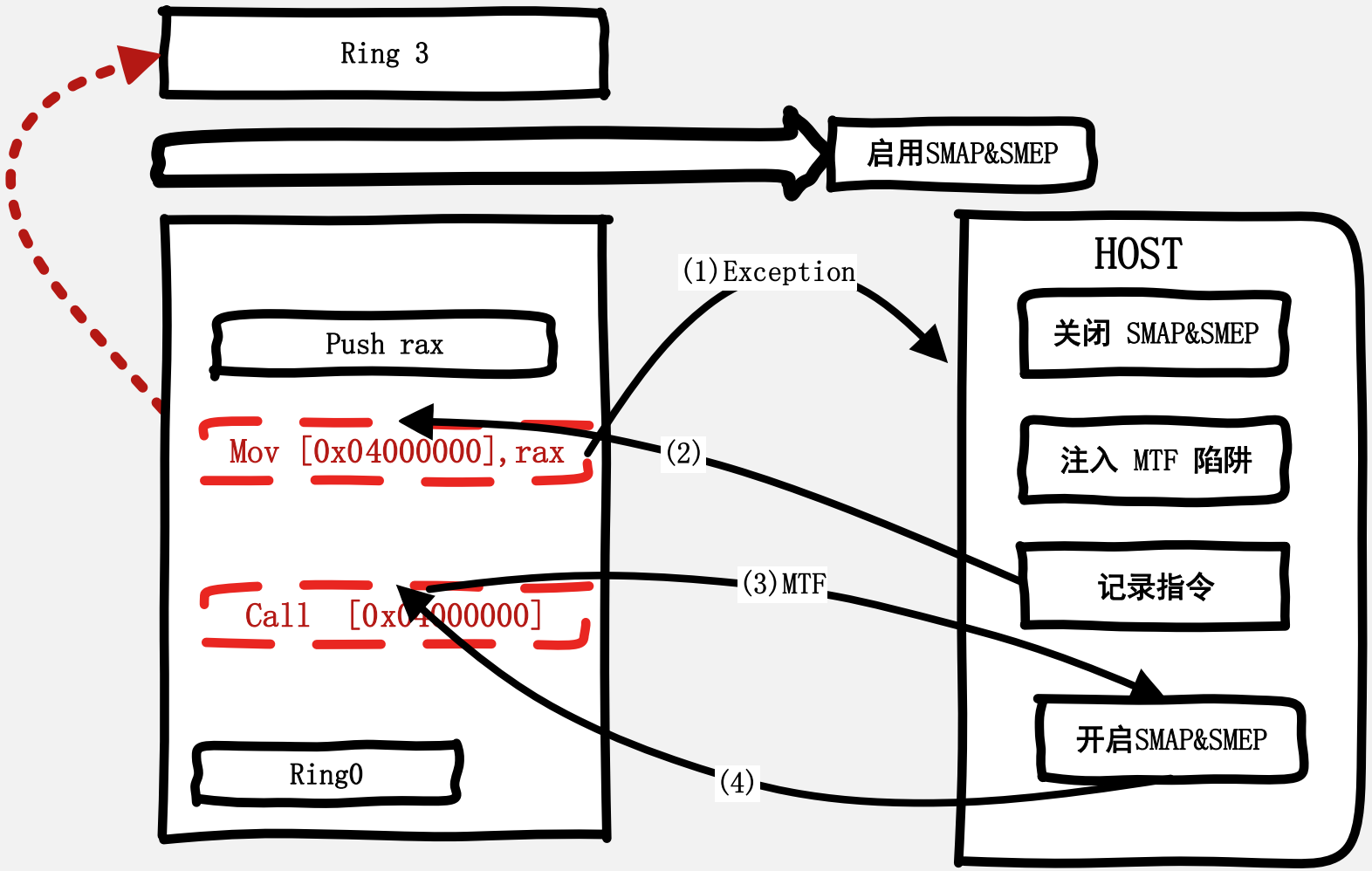
当进行CR3 的赋值时，Guest 就会陷入到 Host 中，进而可以进行PID和Token的判断。


NTKRNELOS.SYS
Kdi.sys
Win32k.sys
.....
Gdi.sys
.....
Guest
Mem
layout
GPA->HPA
E+R+W
E+R+W
E+R+W
R+W
E+R+W
R+W
HOST
EPT

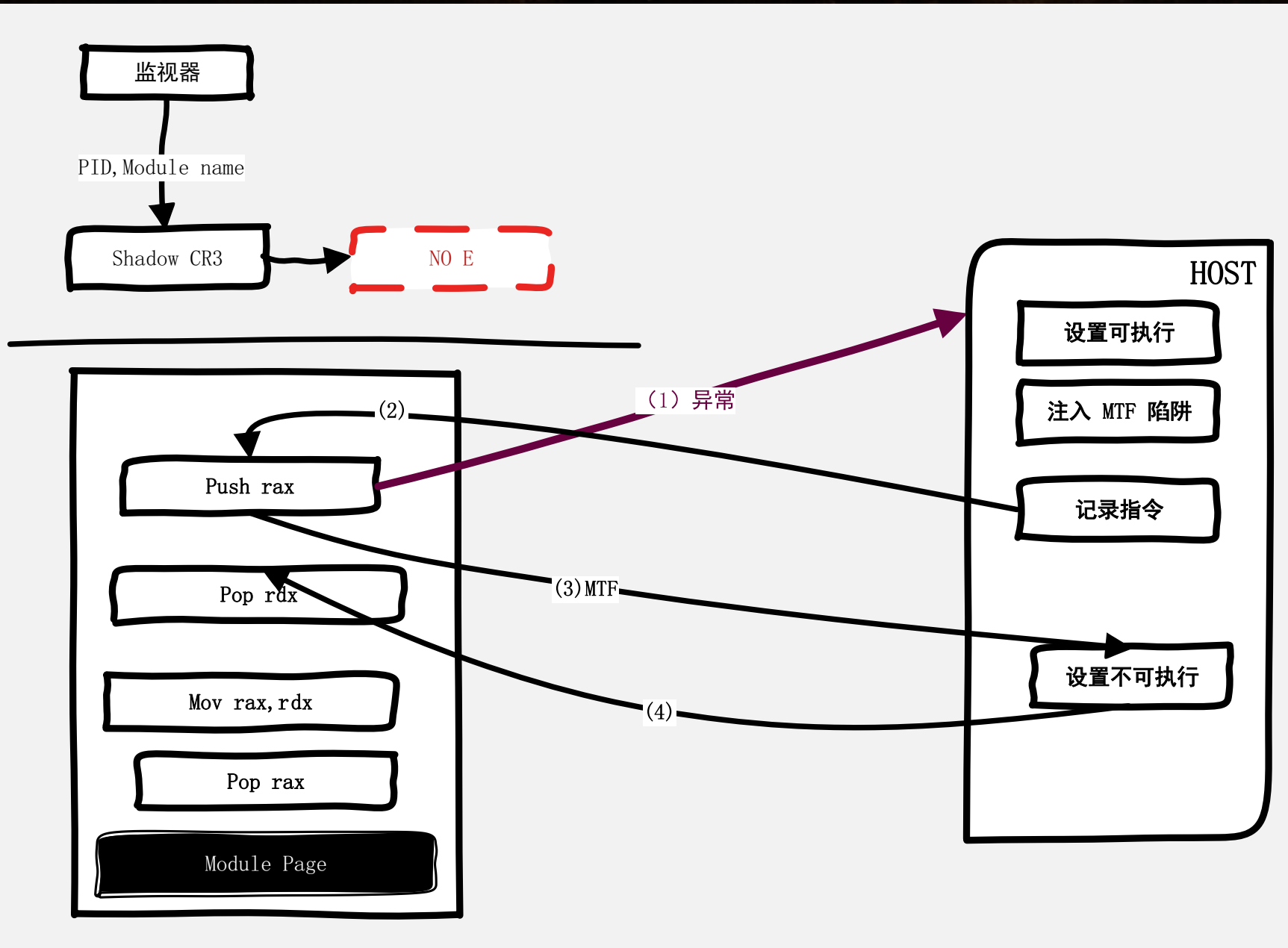
VT VS Rookit


尝试执行
VM Exit
E+R+W
E+R+W
E+R+W
R+W
E+R+W
R+W
HOST
EPT
HOST
校验内存地址
否
是
Page Guard
Rootkit
放行

VT VS Rookit



VT VS Ring0->Ring3


监视器
PID,Module name
Shadow CR3
NO E
HOST
设置可执行
注入 MTF 陷阱
记录指令
设置不可执行
(1) 异常
(2)
(3)MTF
(4)
Push rax
Pop rdx
Mov rax,rdx
Pop rax
Module Page

VT VS 内核信息泄漏

04 攻防技术对抗

攻防技术对抗

05 结语

路漫漫其修远兮
吾将上下而求索

KCon 洞见未来 2017

Thank you!

KCon 洞见未来 2017