面向云计算平台的虚拟机同驻方法

刘维杰[1,2]，王丽娜[1,2]，王丹磊[1,2]，尹正光[3]，付楠[1,2]
（1. 空天信息安全与可信计算教育部重点实验室，湖北 武汉 430079；
2. 武汉大学国家网络安全学院，湖北 武汉 430079；3. 阿里云计算有限公司，浙江 杭州 311121）

摘 要：若攻击者想攻击云平台上某一目标虚拟机，则其必须与目标虚拟机同驻。基于此，提出一种虚拟机同驻方法，通过构建云环境中自适应的隐蔽信道，结合基于隐蔽信道的虚拟机同驻检测方法和自动化虚拟机洪泛策略，并在国内某知名商业云平台上进行同驻验证。实验表明，所构建的自适应隐蔽信道传输正确率可高达 95%以上；所提出的同驻检测方法置信度高，误检率不超过 5‰。同驻方法不会破坏云平台本身隔离性且具有一定的通用性，但潜在威胁极大，亟需重视与防范。

关键词：云计算平台；虚拟机同驻；隐蔽信道；虚拟机洪泛
中图分类号：TP309
文献标识码：A
doi: 10.11959/j.issn.1000−436x.2018241

Virtual machine co-residency method on cloud computing platform

LIU Weijie[1,2], WANG Li'na[1,2], WANG Danlei[1,2], YIN Zhengguang[3], FU Nan[1,2]
1. Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education, Wuhan 430079, China
2. School of Cyber Science and Engineering, Wuhan University, Wuhan 430079, China
3. Alibaba Cloud Computing Co., Ltd., Hangzhou 311121, China

Abstract: If the attacker wants to compromise a target virtual machine on a cloud platform, the malicious virtual machine must be co-resident with the target. Based on this, a virtual machine co-residency method was proposed. The method combined a co-residency detection scheme based on covert channel construction and an automatic virtual machine flooding strategy, and was evaluated on a well-known domestic cloud platform. Experiment shows that the adaptive covert channel can achieve accuracies of 95%, the proposed detection scheme has strong robustness whose false positive rate is less than 5 ‰, the proposed method is versatile and keeps the virtualization isolation barrier intact, which has great potential threat and should be paid great attention and precaution.

Key words: cloud computing platform, virtual machine co-residency, covert channel, virtual machine flooding

1 引言

云计算的概念从提出到广泛应用，在短短几年的时间里经历了巨大的变革。目前，国内外的商用云计算业务都已进入蓬勃发展的重要时期，云计算也越来越多地被视为存储数据和部署服务的下一代 IT 基础设施。然而，多租户动态聚合、边界泛化的特点使云计算平台难以抵抗虚拟机之间共享计算资源所带来的安全威胁。其中，最主要的是虚拟机同驻威胁，即将属于攻击者的虚拟机实例运行在目标虚拟机所在的物理主机上。由于云环境“虚拟隔离、物理共存”的特点，云平台允许同驻的虚拟机共享物理主机的大部分资源，因此同驻威胁难以避免，其主要包括资源干扰、拒绝服务、隐蔽/

收稿日期：2018−04−03；**修回日期**：2018−06−29
通信作者：王丽娜，lnwang@whu.edu.cn
基金项目：国家自然科学基金资助项目（No.U1536204）；中央高校基本科研业务费专项基金资助项目（No.2042018kf1028）
Foundation Items: The National Natural Science Foundation of China (No.U1536204), The Central University Basic Business Expenses Special Funding for Scientific Research Project (No.2042018kf1028)

侧信道、虚拟机跳跃、虚拟机逃逸和迁移间隙等[1-9]。恶意的虚拟机同驻会破坏云平台中数据的机密性和资源的可用性，导致严重的安全问题，对大规模企业用户和普通云租户都会造成极大危害。

如果攻击者意图实施针对云计算平台的攻击，则必须先实现其恶意虚拟机与目标虚拟机的同驻。早在 2009 年，Ristenpart 等[10]首次提出了 Amazon EC2（elastic compute cloud）平台上的虚拟机同驻方案，但是其检测同驻的方法是基于两台虚拟机之间的网络分组往返时延（RTT, round trip time）会随同驻与否而发生变化的原理，该原理过于简单，且未能考虑到云平台中复杂的网络环境所造成网络分组时延测量不准确的问题。

在判断虚拟机是否同驻方面，Bates 等[11]提出利用“同驻水印”来进行同驻检测。其原理为利用检测虚拟机向外部周期性地发送网络分组，同时通过外部代理向目标虚拟机发送网络分组，通过测量网络分组通信是否有足够的时延来判断虚拟机是否同驻。然而，此方法与 Ristenpart 等[10]提出的同驻检测方法都依赖云平台允许虚拟机自由地和外部进行网络通信，很容易由于云平台因安全考虑将 ICMP（internet control message protocol）禁止而失效[12]，且云平台一旦对不同的私有专用网进行隔离[13]，基于网络信息的同驻方案将会变得毫无用武之地。

Zhang 等[14]提出了一种虚拟机同驻探测方案，其让租户将一些暂不使用的处理器缓存行留为警报行，若探测到警报行被使用或其负载与之前不同，则表明其他虚拟机使用了该处理器缓存（下文简称 cache）区域，同驻检测器进程就会发出报警。同时，余思等[15]将虚拟机同驻检测问题具体化为 cache 负载的差异性度量，基于 cache 负载特征匹配来推断目标虚拟机与攻击者虚拟机是否同驻。这两种方案都根据 cache 的负载差异进行同驻判断，但是云平台为了降低能耗，提高资源利用率，引入了各种基于硬件特性的资源优化机制和资源隔离机制[16]，使 cache 的负载测量并不准确，并且计算 cache 负载特征方法复杂，很难保证同驻检测的实时性。

基于隐蔽信道的同驻检测方案效果好，但难以在此基础上实现虚拟机同驻[17]。针对这一问题，本文以“先假设，后验证”的思路，提出了一种面向云计算平台的虚拟机同驻方案，如图 1 所示。攻击者在实施同驻的过程中，需要对同驻情况进行判定，因此需要准确率高的虚拟机同驻检测手段来确定同驻情况。在攻击者有目的性地进行虚拟机洪泛后，还需借助合谋方案对目标虚拟机进行精确定位。

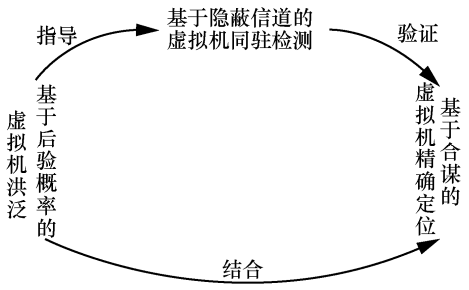


图 1　虚拟机同驻总体方案

本文方案充分考虑具体的云计算环境，力求同驻检测正确率高且易于部署。该方案改进了现有的两种隐蔽信道，并将其运用到虚拟机的同驻检测方案中，打破了基于隐蔽信道的检测方案在现实环境难以实现同驻的壁垒；提出了一种基于后验概率的自动化虚拟机洪泛方法，有针对性地确定目标虚拟机与洪泛虚拟机的同驻。利用本方案在国内某知名商用云平台上进行实际验证，对增强现有商用云平台的安全性有重要意义。

2　云计算环境下同驻威胁模型

攻击者要实现其控制的虚拟机与被攻击虚拟机被云平台管理系统分配到同一物理主机，看似天方夜谭，其实有规律可循。即如果能够事先了解到被攻击虚拟机开启的精确时间，使用简单的虚拟机洪泛（VM flooding），就能让攻击者所拥有的约占总量 $\frac{1}{12}$ 的虚拟机开启在被攻击虚拟机的物理主机上。本文在此基础上进行了改进，使攻击者同驻攻击的成功率增加，并且不需要额外的社会工程信息。攻击者的威胁模型描述如下。

在公有云环境中，攻击者理论上可以通过合法渠道获得部分资源，即攻击者能够创建任意的虚拟机并对其进行管理。本文基于该事实提出如下假设。

假设 1　云平台系统有良好的隔离性，云平台管理系统和其上的客户操作系统均是干净且安全的（及时更新，补丁完备），同时云管理员的操作均是善意且合规的，不存在内部威胁[18]。如此，本文中的攻击者仅通过侧信道的方式实现与目标虚拟机同驻。

假设 2　云平台的各个租户之间是互不信任的，且不具有任何特殊权限。攻击者只能期望利用

侧信道攻击来获取其他用户的隐私信息[19]。攻击者的目标为使用云平台的正常租户，他们利用虚拟机运行某些机密性的相关应用。

假设 3 攻击者有足够的资金保证其能够开启足够多的虚拟机，同时还能够通过社会工程的方式获取目标开启虚拟机的准确时间[11]。

上述假设均符合实际应用场景，假设 2 中针对攻击者背景知识的假设也具有相当的普适意义。为了方便描述，下面将攻击者创建并拥有的虚拟机称为攻击虚拟机，将目标用户创建并拥有的虚拟机称为目标虚拟机。

3 基于隐蔽信道的虚拟机同驻检测

如果攻击者意图实现与目标虚拟机同驻，则必须拥有检测攻击虚拟机是否与目标虚拟机同驻的能力。于是，在描述同驻方案前，本节提出其实现的基础——基于隐蔽信道的同驻检测方案。

现有研究中，基于网络信息的同驻检测方法实现简单，检测效率较高，但是检测准确率低，只能作为其他同驻检测方法的辅助手段使用；基于资源干扰的同驻检测方法虽然实现复杂，但检测准确率较高，适用范围广。但是，随着虚拟机资源管理机制的优化以及硬件技术的不断发展，同驻虚拟机之间的相互干扰越来越难被发现[17]。基于硬件的隐蔽信道作为云平台中不可避免地存在，可被充分地应用于同驻检测中[21]。

因此，本文以准确性、高效性和实用性为目标，提出基于隐蔽信道的同驻检测方案，并以基于内存总线冲突的隐蔽信道和基于最高级缓存（LLC, last level cache）的隐蔽信道为例，实现了这两种同驻检测方案。由于该方案以云平台中的共享硬件资源为渠道构建，因此具备极强的通用性[17]。

3.1 云环境下隐蔽信道机理与构建

基于微处理器架构的隐蔽信道是多用户计算机系统中最常见的隐蔽信道类型，其主要存在于具有共享资源的系统中，如云计算环境中。不过在嘈杂的公有云环境中，微处理器架构隐蔽信道会面临更多的问题。虚拟机迁移、vCPU 调度以及 Hypervisor 的活动增加了许多噪声，这些都给隐蔽信道精确同步带来了挑战。

3.1.1 基于 LLC 的隐蔽信道

与其他隐蔽信道介质相比，cache 对于攻击者更具吸引力。因为 cache 的高操作速度可产生高带宽，并且不受软件系统限制，可以绕过许多高级隔离机制，所以近年来基于 cache 的隐蔽信道备受重视[20-21]。其中，最常用的交替通信技术是 Prime + Probe 方法[22]和 Flush + Reload[23]方法。

然而，在实现基于 LLC 的隐蔽信道之前，首先需要解决两种地址映射不确定的问题。1）用户层应用程序操纵的是虚拟地址，而 cache 是采用物理标记的，数据在 cache 中的位置由其物理内存地址决定，物理地址和虚拟地址在操作系统内的转换是由硬件 MMU（memory management unite）进行的，用户层很难获取该信息；2）新式的 Intel 处理器使用了未在官方文档里声明的散列函数来对 cache 地址进行映射，即 LLC 的非公开散列索引机制。在新式的 Intel 处理器中，为了增强有效性，LLC 被分为了多个分片（slice）。LLC 各分片与物理内存地址之间的映射关系由散列函数确定，即使攻击者可以判断一个 cache 组中包含有哪些 cache 行（line），也无法知晓这些 cache 行所对应的 LLC 分片[24]。针对上述问题，使用 Linux super pages 机制来尽可能地映射额外的地址偏移，从而维持 LLC 物理地址与内存虚拟地址之间的映射关系，如图 2 所示。此时，通信双方只能定位到每一个 cache 分片的索引值 y（第 16 位到第 6 位），而 slice ID 是未知的。

文献[24-25]提出了如何对该散列函数进行逆向，但这种逆向工程需要进行大量的实验，消耗大量时间。于是，文献[7]提出了一种通过构建冲突集的方法来寻找能够映射到相同 cache set 而虚拟地址

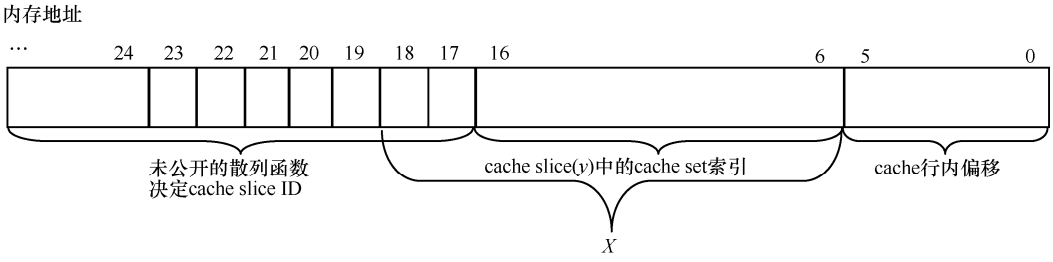


图 2 内存与新型 LLC 的映射索引

不同的内存块。本文对这种方法进行了改进，提出了一种快速构建冲突内存块的方法，能够进一步地缩短寻找时间。改进的方法描述如算法 1 所示。其中，函数 *probe*()为所定义的 Probe 访存原语。

算法 1　基于 LLC 的隐蔽信道冲突集构建算法

输入　候选内存行 lines

输出　驱逐集，每个分片一个驱逐集 *eviction_set*

probe(*set*；*candidate*)

读取 *candidate*；

for each *l* in set do

读取 *l*；

end for

end function

构建 *conflict_set*

随机化 *lines*；

for each *candidate* ∈*lines* do

if not *probe*(*conflict_set*；　*candidate*)

将 *candidate* 放入 *conflict_set*；

end for

构建 *eviction_set*

for each *candidate* in *lines-conflict_set* do

if *probe*(*conflict_set*，　*candidate*)

for each *l* in *conflict_set* do

if not *probe*(*conflict_set*；　*candidate*)

将 *l* 放入 *eviction_set*；

end for

if *eviction_set*∪*eviction_set_before*≠∅

eviction_set_before=e*viction_set* ∪ *eviction_set_before*

输出 *eviction_set*；

conflict_set= *conflict_set-eviction_set*；

end for

通过将具有同样元素的驱逐集合并，使本文构建驱逐集的方案更加圆满，能够在每个 slice 对应的驱逐集中找到超过组相联路数的可供驱逐的内存行。这样，在隐蔽信道构建时，就可以更加灵活地选用任意的 slice ID 和 set ID 作为 LLC 冲突集的 ID。

在信道实际构建过程中，发送方在算法 1 构建 *confict_set* 的步骤中可以同时访问 00|*y*、01|*y*、10|*y*、11|*y* 这 4 块内存块，来清空目的 cache set。由于 LLC 分片的共享环机制[7]，连续访问 4 块 slice 能保证该 cache set 一定被发送方的数据所占据。

3.1.2　基于内存总线的隐蔽信道

内存总线负责处理器与主存之间的数据传输，是现代计算机系统中不可或缺的组成部分。处理器内的 CPU 核心都共用该内存总线。因此，内存总线上的冲突将会导致系统范围可观测的内存访问时延。因此，利用某些特定的程序行为所触发的内存总线冲突就可以构建出一种隐蔽信道[26]。x86 架构中，处理器利用总线锁（memory bus lock）实现内存原子指令。对于未对齐的内存区域，执行内存原子指令会触发内存总线锁。利用这种机制，可以构建一种高速、可靠、跨虚拟机的隐蔽信道，如图 3 所示。

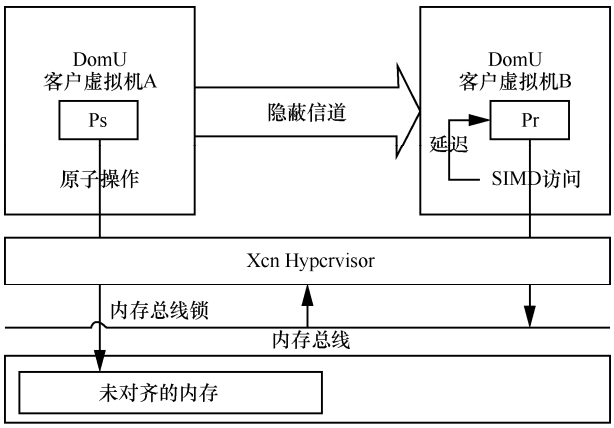


图 3　基于内存总线的隐蔽信道

同样地，对现有的基于内存总线的隐蔽信道进行了改进：当发送方使用特定的原子操作时，接收方使用 SSE (streaming SIMD extension)指令集的某些访存指令操作，避免访问 cache，使正常访问内存总线时的访存时间不会因 cache 命中而产生影响，从而保证每次正常访存（总线不冲突）时的时间基本一致，进而保证了信道传输的准确性。具体如算法 2 所示。

算法 2　基于内存总线的隐蔽信道构建算法

MExotic：异常操作数原子指令

DSend[*N*]，*DRecv*[*N*]：发送和接收数据

发送方：

for *i*=1 to *N*-1 do

if *DSend*[*i*]=1

for *j*=0 to *T*-1 do

M_{Exotic} 内存访问；

end for

else

for *j*=0 to *kT*-1 do

SIMD 内存访问；
end for
end for
接收方：
for *i*=1 to *N*-1 do
for *i*=0 to *T*-1 do
SIMD 内存访问；
end for
if 平均访问时间>*threshold*
DRecv[*i*]=1；
else
DRecv[*i*]=0；
end for

3.2 面向码元识别的自适应信道优化

为了解决信道在嘈杂的公有云环境中传输正确率会受到影响的问题，本文引入了自适应的码元识别技术，以此来提升传输正确率。

本文通过使用数据帧（frame）来加强通信协议对时钟漂移和调度中断的弹性。传输的数据被分成固定长度，发送方使用伪随机数生成算法来产生每一个数据帧的载荷[27]，并用起始模式和停止模式将每个数据段构造成数据帧。这种方式具备两个优点：①当发送方检测到传输中断时，不需要重传全部数据，仅需重发相关数据帧；②在传输期间总是不可避免地会丢失部分数据，利用数据帧，接收方能更加容易地定位错误，进行纠错处理。

基于 LLC 的隐蔽信道与基于内存总线的隐蔽信道实现均包含 3 个参数：*threshold* 、*S* 、*T* 。其中，*threshold* 为判断码元为 1 的时延阈值；*S* 为码元周期，即 *S* 个连续的 0 识别为一位“0”，*S* 个连续的 1 识别为一位“1”；对于基于 LLC 的隐蔽信道，*T* 为每个码元执行原子操作的次数，对于基于内存总线的隐蔽信道，*T* 为 Prime+Probe 方法中的发送方暂定时延。

3.2.1 信号的平滑与连续识别

由于嘈杂的云环境带来的噪声，信道若未经过去噪便进行码元识别操作，将会引入相当大的传输错误。平滑滤波技术能够过滤低频信号中的噪声，它不经由傅里叶转换，在图像处理领域被广泛应用。本文利用平滑滤波来实现信号的初筛。

平滑滤波器的输出是滤波器邻域所含像素的平均，过滤窗口越大，平滑效果越好；然而若窗口过大，平滑效果则会使边缘的信息失真过于严重，导致输出的信号过度模糊，因此需合理选择窗口的大小。在本文方案中，窗口大小 W_{size} 与码元 *S* 周期设为相同，这样既符合接收码率与滤波速度一致性，又能够最大可能地不遗漏任何有效比特位。

经过去噪操作后，所接收的“0”和“1”的区别已经较为明显（在基于内存总线的信道中，发送“0”仅消耗约 100 个 CPU 时钟周期，发送“1”需消耗几千个 CPU 时钟周期；在基于 LLC 的信道中，发送“0”消耗约 30 个 CPU 时钟周期，而发送“1”需要消耗约 100 个时钟周期）。因此，采取聚类的方式来区分接收的信息即可。在此情况中，仅需要将数据分为两类，即仅需要求出聚类中心，之后再进行简单判断。由于不用考虑时间序列，需要聚类的数据仅为一维数据。在此本文采用最为简单有效的 *K* 均值聚类法来实现该功能。

3.2.2 信道自适应同步与校准

由于发送方和接收方分属不同的虚拟机，它们之间的调度差异将导致数据传输和检测过程不同步，进而对信道的调制造成很大的问题。本文通过自时钟编码克服时钟失步，并使用差分曼彻斯特编码传输数据位。更重要的是，当通信双方传输起始时间不同时，有很大的概率导致传输的数据极不连续。同时，为了减少重发次数，节省时间开销，必须确保发送方和接收方的时隙是对齐的。如果没有对齐，两者将在重叠的时间内争用该信道并引入位翻转错误。这种不受限制的争用会导致“0”和“1”的值的可分离分布较少，加剧发送方和接收方的竞争，明显降低信道的容量与可用性。针对这个问题，本文提出时钟同步与接收确认机制来提高信道可靠性，如算法 3 所示。其中，*send_covert_bits*()与 *receive_signal*()分别表示发送帧与接收帧的函数方法，*calculate_best_round*()表示搜索并微调当前帧延时的函数方法。

算法 3 自适应的信道同步算法
DSend[*N*]，*DRecv*[*N*]：发送和接收数据
calculate_best_round(): 信道传输正确率计算函数
发送方：
for *n*<1 to 20 do
send_covert_bits(*DSend*[*N*]);
start_time=*start_time*+1；/* 微调 1 μs */
end for
确定 *clock_offset*；
接收方：

for n<1 to 20 do

receive_signal(*DRecv*[*N*]);

end for

best_round=calculate_best_round(*DRecv*[*N*]);

本文提出的同步协议能够实现在几百 μs 的时钟差异内，实现高精度的对齐传输。通过在隐蔽通道上传送一个已知的比特序列来进行多轮处理。在每次通信之后，发送方将其时钟移动 1 μs，稍微改变两个进程的执行起始时间。随着时钟变得更加同步，两个进程的争用变得更加一致。一旦达到最高正确率后，发送方的每次移位都会降低信道的传输正确率。此时，接收方通过测量最高正确率便可能定位到最佳的移位大小。同步过程的耗时取决于用于测试的分组大小，基本上每个虚拟机只需要执行一次即可完成同步。

通信过程中，发送方和接收方必须就信道参数（如频率，即 1 bit 的时隙持续时间）达成一致。于是，在实施攻击前，必须根据目标机器的特性，分析目标机器以确定这些参数。在离线分析参数阶段，通过微调 S 和 T 的值使通信效率达到最大，并将其应用于真实攻击场景中。具体来说，攻击者通过求极值的手段来确定阈值以使信道传输的正确率最大，并且在之后的攻击中，使用该参数进行测量和判断。

4　面向云平台的虚拟机同驻方案

由于虚拟机同驻造成的危害越来越大，研究人员在云平台的虚拟机投放策略上提出了抵御同驻危害的方法。已有研究[28-29]提出虚拟机动态部署算法，基于云提供商协助的 VM 迁移服务，通过不断地重新部署 VM，使任意两台 VM 间不存在足够长的同驻时间，从而限制由侧信道引发的信息泄露。上述方法虽然能够在一定程度上抵御同驻威胁，但其防御窗口过长，仍无法从本质上防止虚拟机实例洪泛造成的同驻。

虚拟机实例洪泛，即攻击者在同一时间、同一个云平台可用区，开启大量的虚拟机实例。这些实例可能是攻击者利用自身账号投放，也可能是通过其控制的虚拟机僵尸网络投放。虚拟机实例洪泛本质上是对云平台虚拟机分布策略的滥用，但其并不违反云平台的商业策略和服务等级协议。

当攻击者实例开启时间接近受害者虚拟机实例开启时间时，攻击者可以有效地利用云平台平行放置的部署算法来实现一定概率的攻击者实例与受害者实例同驻。由于云计算的动态特性，服务器通常在实例上运行，不需要时终止，稍后再运行。例如，攻击者可以监视服务器的状态（如通过网络探测）直到该实例消失，然后在发现一个新实例（该服务）重新开启时进行实例洪泛。攻击者甚至可以利用该特点主动触发新的目标虚拟机实例。

4.1　基于后验概率的虚拟机洪泛覆盖

本节提出了一种基于后验概率的自动化虚拟机方案。通过该方案可以得出：①被攻击的虚拟机分布在各台物理主机的概率；②本次投放 VM flooding 虚拟机时，每一台攻击虚拟机与被攻击的目标虚拟机同驻的概率。该方案的总体思路为先大量投放虚拟机，再对所投放的虚拟机进行同驻检测，将确定为同驻的虚拟机归类，最后对所投放的虚拟机进行约减。具体步骤如下。

首先，通过云平台提供的合法途径，同时申请一批虚拟机集群。该批虚拟机集群中，记一台虚拟机落在各台物理主机的概率为 $p=(p_1,p_2,\cdots,p_m)$，m 为物理主机总数，且

$$\sum p_i = 1 \tag{1}$$

对 p 做出估计，从而确定 VM flooding 所能达到的最大同驻概率。

当外部请求发送到云平台时，记请求开启虚拟机实例数量为 n，经过同驻检测得知，所有的 n 个虚拟机实例在 m 台物理平台（物理主机）上的分布情况为 $x=(x_1,x_2,\cdots,x_m)$，x_i 表示第 i 台物理机上的实例数，则有

$$\sum x_i = n \tag{2}$$

在实验结果上有

$$P(x)=\prod p_i^{x_i} \tag{3}$$

通过极大似然法来估计 p，然后在实验结果上最大化对数似然函数，有

$$\ln\big(L(p)\big)=\sum x_i \ln p_i \tag{4}$$

$$\hat{p}=\arg\max_P L(p) \tag{5}$$

记

$$g(p)=\sum p_i = 1 \tag{6}$$

显然式(4)为凸函数，在式(6)中引入拉格朗日乘子，代入式(4)得

$$\ln\big(L(p)\big)=\ln\big(L(p)+\lambda(1-g(p))\big) \tag{7}$$

当式(7)取得最大值时，自变量偏导数为 0，即

$$\frac{\partial}{\partial p_i}\ln\left(L(p)+\lambda(1-g(p))\right)=0 \tag{8}$$

将式(4)、式(6)代入式(8)，有

$$\frac{x_i}{p_i}-\lambda=0 \tag{9}$$

由式(2)、式(9)可得

$$\hat{p}_i=\frac{x_i}{\sum x_i}=\frac{x_i}{n} \tag{10}$$

在得知其持有的 VM flooding 虚拟机群的同驻情况后，攻击者可以将每台物理主机上的攻击实验用虚拟机数量缩减到每台物理主机上仅一台，这将大量减少之后同驻攻击所需的成本和开销。若此时的目标虚拟机处于第 i 台物理主机上，其上已有 N_i 台虚拟机同驻，则在同一时间再开 $\dfrac{\ln(1-\delta)}{\ln\left(1-\dfrac{N_i}{m}\right)}$ 台虚拟机，有 δ 的概率使攻击虚拟机与目标虚拟机同驻。

上述概率可以告知攻击者如何进行下一步的攻击实验，并给予攻击者理论上的指导。例如，优先从同驻发生率较高的物理主机开始进行同驻检测，从而加快发现与目标同驻的速度。

4.2 基于合谋的同驻虚拟机远程定位

当洪泛虚拟机数量达到一定程度时，可实现云平台可用区物理主机的全部覆盖。尽管如此，攻击虚拟机如何定位目标虚拟机依旧困难。

文献[4]提出，在云计算环境下，攻击虚拟机可借助前面所述的内存总线冲突来实现云环境的拒绝服务（DoS, denial of service）攻击，具体操作方式为不断触发内存总线冲突，正如构建隐蔽信道一般[30]，造成大量内存访问时延，从而达到平台上其他虚拟机无法正常使用内存的目的。除了该内存总线占用漏洞外，利用内存行锤击漏洞也可造成虚拟化平台的服务等级下降甚至停止。攻击虚拟机可通过不断锤击具有关键数据的内存行（如 Hypervisor 所维护的 EPT/NPT 页表），使整个虚拟化平台暂停服务[2-3]。本文基于上述 DDoS 攻击，结合合谋攻击的思路，提出一种能够快速定位目标虚拟机的方法。

假设攻击者已采用如 4.1 节所述方案，成功将攻击虚拟机分布到各物理平台上，为了节省开销，此时攻击者可将各物理主机上的洪泛虚拟机减少至一台。为了叙述方便，本文将各台物理主机上运行的被攻击者持有的虚拟机称为 probe 虚拟机，并设其集群为 $P=\{probe\,|\,p_1,p_2,\cdots,p_m\}$；将攻击者所拥有的远程设备（可访问云中某服务）称为 remote 端。

当攻击者已通过社会工程学手段得知目标虚拟机位于某可用区且部署服务时，可使用 remote 端访问该服务，同时，在 probe 虚拟机中执行基于内存漏洞的 DoS 攻击，并观察 remote 端上目标服务的响应情况。若发现 P 中某台虚拟机 p_i 在执行 DoS 攻击时，服务中断或时延极大，则可判定 p_i 已与目标虚拟机同驻。

5 同驻实验与行业验证

在对本文提出的同驻方案验证之前，首先对所提的基于隐蔽信道的同驻检测方案进行了实验验证，以确保该同驻检测（判定）方案在之后的方案验证中得到最大程度的效用。

5.1 虚拟机同驻判定

为了使同驻检测的正确率尽可能高，首先必须确保前面所提的两种隐蔽信道的通信传输正确率达到峰值，参数 *threshold* 的设定对接收方能否识别信号、建立通信起着决定性作用，而 T 和 S 这两个参数的配置则对隐匿信道的传输正确率具有显著影响。本文以 Xen 为虚拟化平台，对这两种基于隐蔽信道的同驻检测技术以及上述优化措施进行了编码实现与一系列实验，结果如图 3 所示。其中，实验平台采用 Xen 4.4.2 作为虚拟机管理器，CPU 采用 Intel i5-3470，每台虚拟机分配 1 024 MB 内存。由于在隐蔽信道的工作过程中，经常会发生传输序列的某一个或几比特位丢失、翻转或错误检测而导致的比特位增加，因此不能使用常规的序列按位比对来计算传输正确率。此处采用 Needleman-wunsch 序列比对算法[31]进行隐蔽信道的正确率计算。经过优化后的两种隐蔽信道均可达到 95%以上的传输正确率。由于每个码元执行的原子操作次数越大，其产生的时间周期和平均时延越均匀、稳定，从而使传输正确率越高。然而由于 T 的增大会降低传输速率，因此在以传输数据为目的的隐蔽信道中，T 的取值需权衡传输正确率与传输速率。

5.1.1 同驻阈值计算

为了确定传输正确率与虚拟机是否同驻之间

的关系，选取分布在两台物理主机上的共 10 台虚拟机之间进行隐蔽信道通信实验。以基于内存总线的隐蔽信道为例，其传输正确率与同驻状态如图 4 所示。

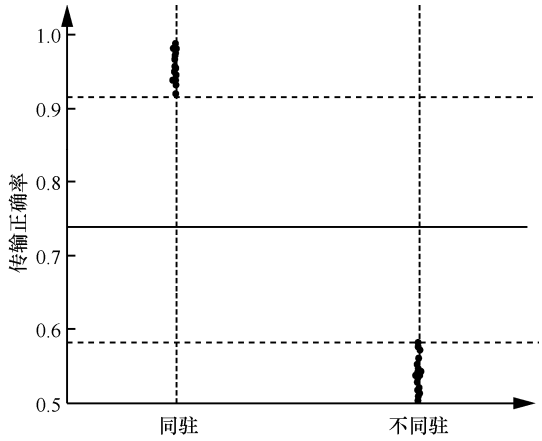


图 4　传输正确率的多次采样

其中，实际同驻的虚拟机共 20 对，非同驻的虚拟机共 25 对。实验中，同驻虚拟机传输正确率最小值为 90.7%，非同驻虚拟机传输正确率最大值为 57.9%。两类样本间有明显间隔，因此，可以通过设定传输准确率阈值判定虚拟机是否同驻。

实际上，当隐蔽信道的参数一定时，同驻虚拟机传输正确率与非同驻虚拟机传输正确率均为客观上的定值。由于实验数据的局限性和某些可能出现的噪声的影响，传输正确率符合正态分布。为了确定虚拟机是否同驻的判定阈值，需要先由样本估计出正态分布的参数，然后经过计算得到最优的判定阈值。

记同驻虚拟机传输正确率 X_t 服从正态分布 $N(\mu_1,\sigma_1)$，非同驻虚拟机传输正确率 X_b 服从正态分布 $N(\mu_2,\sigma_2)$。由极大似然估计，有

$$\hat{\mu}=\frac{1}{n}\sum x_i \tag{11}$$

$$\sigma^2=\frac{1}{n}\sum x_i^2-\left(\frac{1}{n}\sum x_i\right)^2 \tag{12}$$

由式(11)和式(12)得：μ_1=0.946，σ_1=0.023 3，μ_2=0.540，σ_2=0.233 。

同时，记 $\Phi(g)$ 为标准正态分布的概率密度函数，则同驻时传输正确率分布函数 p 为

$$p(x_t\mid\mu_1,\sigma_1)=\Phi\left(\frac{x_t-\mu_1}{\sigma_1}\right)$$

不同驻时传输正确率分布函数 q 为

$$p(x_b\mid\mu_1,\sigma_1)=\Phi\left(\frac{x_b-\mu_1}{\sigma_1}\right)$$

本文认为漏判率与误判率占有同样的比重，因此引入 F_1 分数来衡量 x 的准确性，此时

$$F_1=\frac{2pq}{p+q} \tag{13}$$

最优判定阈值应满足

$$\hat{X}_t=\arg\min F_1$$

为使 F_1 分数在 x 的取值区间内取得最大值，而由于 X_t 无解析解，极值无法计算，在此通过查表，以一定精度搜索 X_t，得到其极值解为 x=74.3%，如图 4 中实线所示。

5.1.2　检测效果评估

通过上述计算得到判定阈值后，重新对 3 台同样的物理主机上的共 15 台确定编号的虚拟机进行隐蔽信道通信实验。为了尽可能地模拟实际云平台环境，所有虚拟机每 15 min 进行一次随机迁移(目标主机不确定)。

以基于内存总线的隐蔽信道实现为测试工具，同样每 15 min 对 105 对虚拟机进行同驻检测，每次检测持续 8 s，持续测量 1 h，实验结果如表 1 所示。

表 1　　同驻检测效果评估

参数	值
实际同驻对数	218
判断正确次数	217
误检率	4.6‰
实际不同驻对数	202
判断正确次数	202
漏检率	0%

通过表 1 可以看出，本文提出的同驻检测方法具有极低的误检率和漏检率。这一方面是由于对两种隐蔽信道的实现做出了改进，且充分考虑到云环境中的噪声和其他不良影响，对信道的通信进行了针对性的优化；另一方面是由于所设的判定阈值具有较好的置信度。随后，本文提出的同驻检测方法在实验室环境中检测成功的前提下，将此方法应用于商用云平台的同驻检测中，通过虚拟机洪泛碰撞的方式使两台云实例同驻。

5.2 同驻方案行业应用验证

现阶段，国内公有云服务竞争激烈。其中，某云平台（下文中简称 α 云）作为行业标杆，其他公有云服务平台架构与其极为类似。因此，为了使实验结果具有标准性和普适性，本文选取 α 云作为目标，对其商业策略和服务等级协议进行研究，并在 α 云上有针对性地利用前面提出的 VM flooding 攻击方案进行了一系列实验。

目前，在物理基础设施方面，α 云提供了 5 个配置自由度，分别是：①地域；②可用区，可用区是指在同一地域内，电力和网络互相独立的物理区域，通常为一个机房；③~⑤实例系列，实例系列代表着不同的硬件系列，分别为系列Ⅰ、系列Ⅱ、系列Ⅲ。其中，不同系列采用不同的 CPU 类型，且其内存也不尽相同。在上述 5 个自由度中，前两个（地域和可用区）属于对地理位置的配置，而后 3 个属于对实例类型的配置。本文选择华南 1-可用区 A 进行投放实验，后续如无特殊说明，虚拟机实例都在该可用区创建。

5.2.1 自动化同驻检测

为了能够对每个虚拟机实例进行同驻检测，每个虚拟机实例上都必须安装隐蔽信道通信程序。因此，首先创建了一个含有隐蔽信道发送和接收程序的镜像 Image-covert-channel，其用户操作系统为 64 bit Ubuntu14.04。后续的虚拟机实例均以此镜像为模板创建。

然而，当创建 n 台虚拟机时，需要对 C_n^2 对虚拟机进行同驻检测，其工作非常繁琐，因此，本文设计并实现了自动化同驻检测工具（ACD, automatic co-resident detector），该工具简化了同驻检测的操作流程，为本文的后续实验提供了十分有效的帮助。ACD 通过脚本执行，远程控制 α 云中的虚拟机运行 sender 或 receiver 程序，检测同驻情况，其工作流程如图 5 所示。

为了更进一步实现同驻检测的自动化，本文基于 Ansible 开发了配置管理与自动化运行工具，通过 SSH（secure shell）连接服务器并运行配置任务。它非常适合将 bash 脚本转换成 Ansible 任务，并且由于其基于 SSH，因此更易于检查运行结果。Ansible 同样支持在 α 云或其他大型商用云（如 Amazon EC2）中采用组（group）的形式管理虚拟机并遍历组内的所有主机。由此，使用 add_host 模块动态创建一个由这些新实例组成的 group，便于在后续任务中立即在主机上执行所需操作。

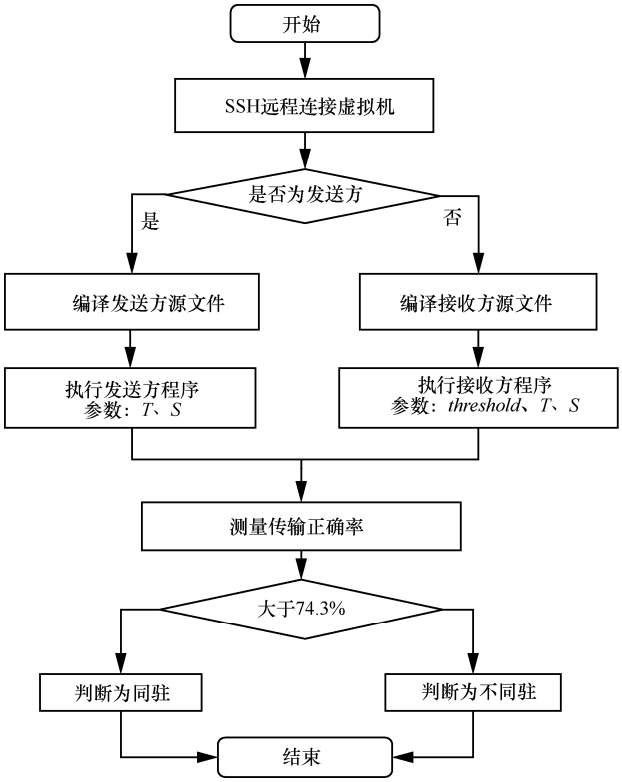


图 5 ACD 工作流程

5.2.2 同驻实验案例研究

由于考虑到 CPU 核数和内存大的虚拟机可能被分配到更具特殊性的物理机中，首先用账号甲同时创建了 n_1.small 和 n_1.medium 虚拟机实例各 50 台，基于第 3 节提出的同驻检测方法，对其进行两两检测。检测结果显示，任意两台虚拟机都不同驻。再考虑到独享型的虚拟机可能被分配到更具特殊性的物理机。继续用账号甲同时创建了 n_2.small 和 n_2.medium 虚拟机实例各 50 台，对其进行两两检测。检测结果显示，任意两台虚拟机都不同驻。结果如表 2 所示。

表 2 单一账号同时创建多台虚拟机

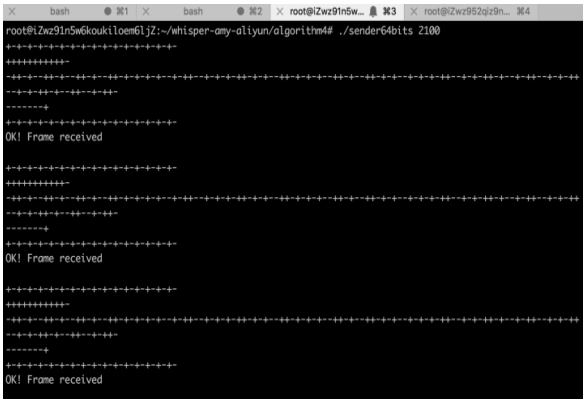
账号	地域-可用区	实例类型	数量	同驻对数
甲	华南 1-A	n_1.small	50	0
甲	华南 1-A	n_2.small	50	0
甲	华南 1-A	n_1.medium	50	0
甲	华南 1-A	n_2.medium	50	0

基于上述推测可知，单一账号的同驻情况并不理想。于是，本文调整最初的洪泛策略，使用多个账号进行实验。

首先，改用两个账号分别同时创建了 5 台 sn_1.medium 虚拟机实例，对其进行同驻检测，检测到

一对虚拟机同驻，其实例 ID 分别为 i-wz91n5w6koukiloem6lj 和 i-wz952qiz9n49r2vj5yyg。

以 i-wz91n5w6koukiloem6lj 作为发送方，i-wz952qiz9n49r2vj5yyg 作为接收方，采用经过调试的最优参数：*threshold*=6 700、*T*=300、*S*=7，测量隐蔽信道传输正确率高达 95.3%，其传输效果如图 6 所示。其中，发送信息被封装到一个 frame 中，“+”“−”分别代表所发送的“1”和“0”。后续如无特殊说明，所有同驻实验均采用上述参数作为输入。



(a) 发送方

(b) 接收方

图 6　隐蔽信道在 α 云上的检测效果

之后，用甲、乙、丙这 3 个账号分别同时各创建了 5 台 sn_1.medium 型实例，共 15 台实例。为了进一步具体测量同驻产生率，本文分别使用两个账号同时创建 5 台 sn_1.medium 虚拟机及 3 个账号同时创建 5 台 sn_1.medium 虚拟机，并进行 50 次重复实验。其中，当两个不同账号同时创建 medium 型实例各 5 台时，有 48%的概率实现一对虚拟机同驻；当 3 个不同账号同时创建实例各 5 台时，有 90%的概率出现至少一对虚拟机同驻，有 82%的概率出现至少两对虚拟机同驻，有 52%的概率出现 3 对虚拟机同驻。另外，观察到同驻的这两台虚拟机分别处于两个不同的内网网段，因此判断 α 云有极大可能使用了类似 VPE[13]或 VxLan 的网络隔离措施来构建安全组或私有云的网络边界。

采用更多的账号进行了相同的实验。当使用 4 个账号甲、乙、丙、丁同时创建 sn_1.medium 虚拟机实例时，共出现了 35 对同驻。其中，有两台物理机上同驻了 4 台虚拟机，3 台物理机上同驻了 3 台虚拟机，14 台物理机上同驻了两台虚拟机，另外，75 台物理机上只分配到一台虚拟机，总计 120 台虚拟机被部署到了 94 台物理机上。同时，5 个账号与 6 个账号同时创建共 120 台虚拟机的同驻结果如表 3 所示。由表 3 可知，当账号越多时，同驻情况越好，实证了 5.2.1 节所提出的假设。

接着，为了探究虚拟机同驻概率与实例类型之间的关系，本文对 n_1.tiny、sn_1.medium、sn_2.medium 这 3 种类型的实例进行了实验。此次实验使用了 12 个账号同时各创建一台虚拟机，重复实验 50 次，同驻结果如表 4 所示。可以看出，sn_1.medium 型实例的平均同驻对数略高于其他两种类型。

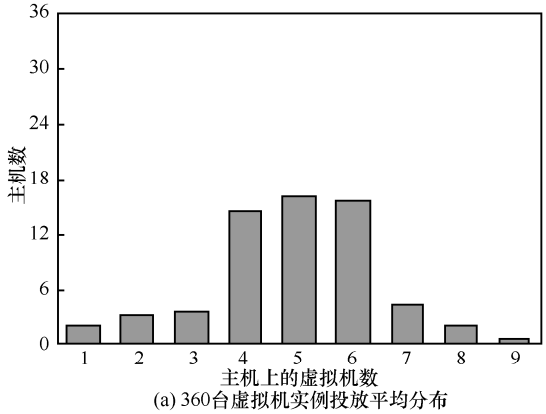
表 3　　多账号同时创建多台虚拟机

账号	地域 - 可用区	实例类型	数量	同驻对数
甲、乙	华南 1-A	sn_1.medium	10	1
甲、乙、丙	华南 1-A	sn_1.medium	15	2
甲、乙、丙、丁	华南 1-A	sn_2.medium	120	35
甲、乙、丙、丁、戊	华南 1-A	sn_2.medium	120	53
甲、乙、丙、丁、戊、己	华南 1-A	sn_2.medium	120	59

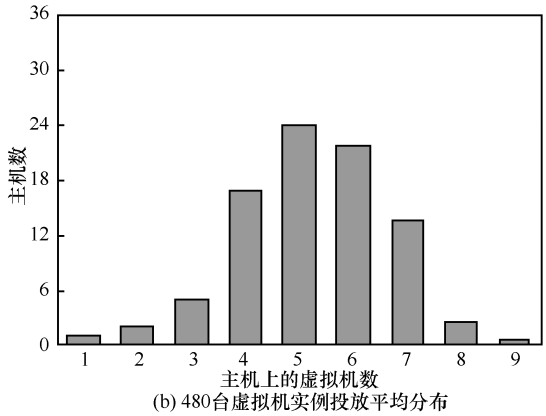
表 4　　多账号同时创建不同类型实例

实例类型	数量	平均同驻对数
n_1.tiny	12	2.46
sn_1.medium	12	2.88
sn_2.medium	12	2.62

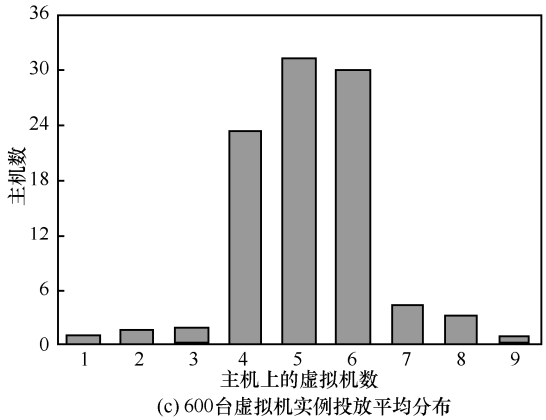
由上述分析可知：当账号越多时，同驻比例越大；当实例类型为 sn_1.medium 时，同驻比例最大。因此，为了进一步对本文所提的洪泛策略进行改进与实验验证，在 36 d 时间内 α 云使用高峰时间段（晚 8 时—晚 9 时），使用 12 个账号同时开启共 360 台、480 台和 600 台 sn_1.medium 型实例各 10 次。具体情况如图 7 所示。



(a) 360台虚拟机实例投放平均分布



(b) 480台虚拟机实例投放平均分布



(c) 600台虚拟机实例投放平均分布

图 7　大规模虚拟机实例洪泛

由图 7(c)可知，110 台物理主机中，载有 5 个和 6 个同驻虚拟机的数量最多，分别有 32 台和 31 台。所有的物理主机均有 3 个及以上同驻虚拟机，单台物理主机上最大的同驻虚拟机数为 9。由第 4 节结论可知：在同一时间再开启 10~15 个虚拟机，有约 15%~21%的概率使攻击虚拟机与这 9 个目标虚拟机同驻。另外，在所有的 12 个账号中，每 50 个实例均被投放到了 40~43 台物理主机上，其中，载有两个虚拟机的物理主机均不少于 6 台。

图 8 显示了采用上述基于后验概率洪泛策略进行有针对性的虚拟机投放时，仅使用 2、5、10 个账号迭代开启虚拟机时同驻的覆盖率。从图 8 可以看到，当投放数量超过 100 时，同驻覆盖率有大幅提升；当使用 5 个不同账号投放 72 次（360 台虚拟机实例）时，同驻覆盖率就已经接近 90%；当使用 10 个不同账号投放 25 次时，覆盖率超过 95%；当投放实例数超过 500 时，无论使用多少数量的账号，都可基本实现全部覆盖。

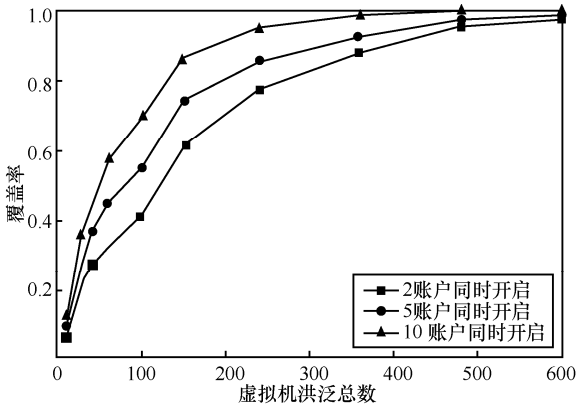


图 8　同驻覆盖比

5.2.3　实验结果分析

通过历时多个月的同驻实验，可以发现与验证如下规律。

1) 现阶段 α 云对于同驻威胁做出了基本的防护，采取了一些简单的应对措施。首先，ICMP 分组被禁止使用于网络探测，不同用户的虚拟机之间的网络与其所处的物理主机无关。其次，同一账号的虚拟机在正常情况下无法被分配到同一物理机。

2) 不同账号同时创建相同可用区、相同实例类型的虚拟机时，产生同驻的可能性极大，且采用的账号越多，攻击成功率越高。实例类型对于同驻率的影响不大。

3) 当启动大量的虚拟机来进行 VM flooding 实验时，由于物理主机的数量远小于开启的实例数，同一账号的虚拟机无法被分配到同一物理机的规律被打破。在同一时间大量虚拟机被分布在百余台物理主机上，基本覆盖了该可用区子网的物理主机，充分说明了该洪泛策略的可行性。

6　结束语

本文对适用于商用云平台的虚拟机同驻方案进行了研究。以“先假设，后验证”的方式来完成攻击虚拟机与目标虚拟机的同驻，且在强安全条件下依旧有效。该方案能够使攻击者快速实现恶意虚

拟机与目标虚拟机的同驻，尽可能地减少不必要的开销。

本文通过自适应的侧信道优化手段对两种可用于同驻检测的隐蔽信道进行了优化；提出了一种虚拟机实例洪泛策略，结合该策略与同驻检测方法可得虚拟机洪泛过程中任意两台虚拟机的分布情况，进而能够得到攻击者实现与目标虚拟机同驻的精确概率；最后本文结合实际，在α云实施了本文同驻方案，并取得了成功，相比同类方案[10]8.4%的成功率有大幅提升，同时也优于文献[32]所提方案20%~100%的成功率。整个过程并不需要破坏α云本身的防护机制，且通过自动化脚本实施，开销小、灵活性高。当同驻覆盖率高达一定程度时，此攻击甚至可演变为大规模 DoS 攻击。

本文所做工作本质上属于对云平台的攻击，是一切基于同驻的攻击的必要前提。虽以α云作为实验目标，但由于其他 IaaS 云平台与α云的本质架构一致，其结果仍可为其他商用云平台提供参考。之后，将会针对此威胁研究其检测与防御方案，尽可能用通用的方法来解决同驻所带来的信息泄露问题。

参考文献：

[1] DESNOS A, FILIOL E, LEFOU I, et al. Detecting (and creating!) a HVM rootkit (aka BluePill-like)[J]. Journal in Computer Virology, 2011, 7(1): 23-49.

[2] RAZAVI K, GRAS B, BOSMAN E, et al. Flip feng shui: hammering a needle in the software stack[C]//Usenix Security Symposium. 2016: 1-18.

[3] XIAO Y, ZHANG X, ZHANG Y, et al. One bit flips, one cloud flops: cross-VM row hammer attacks and privilege escalation[C]//Usenix Security Symposium. 2016: 19-35.

[4] ZHANG T, ZHANG Y, LEE R. DoS attacks on your memory in the cloud[C]//ACM Symposium on Information, Computer and Communications Security. 2017: 253-265.

[5] IRAZOQUI G, INCI M S, EISENBARTH T, et al. Fine grain cross-VM attacks on Xen and VMware[C]//ACM Conference on Cloud Computing. 2014: 737-744.

[6] IRAZOQUI G, INCI M S, EISENBARTH T, et al. Wait a minute! a fast, cross-VM attack on AES[C]//International Symposium on Recent Advances in Intrusion Detection. 2014: 299-319.

[7] LIU F, YAROM Y, GE Q, et al. Last-level cache side-channel attacks are practical[C]//IEEE Symposium on Security and Privacy. 2015: 605-622.

[8] IRAZOQUI G, EISENBARTH T, SUNAR B, et al. Cross processor cache attacks[C]//ACM Conference on Computer and Communications Security. 2016: 353-364.

[9] YAROM Y, GENKIN D, HENINGER N, et al. CacheBleed: a timing attack on OpenSSL constant time RSA[C]//International Workshop on Cryptographic Hardware and Embedded Systems. 2016: 346-367.

[10] RISTENPART T, TROMER E, SHACHAM H, et al. Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds[C]//ACM Conference on Computer and Communications Security. 2009: 199-212.

[11] BATES A, MOOD B, PLETCHER J, et al. Detecting co-residency with active traffic analysis techniques[C]//ACM Conference on Cloud Computing. 2012: 1-12.

[12] XU Z, WANG H, WU Z, et al. A measurement study on co-residence threat inside the cloud[C]//Usenix Security Symposium. 2015: 929-944.

[13] 王丽娜, 张浩, 余荣威, 等. 基于 VPE 的可信虚拟域构建机制[J]. 通信学报, 2013, 34(12): 167-177.
WANG L N, ZHANG H, YU R W, et al. Building mechanism of trusted virtual domain via the VPE[J]. Journal on Communications, 2013, 34(12): 167-177.

[14] ZHANG Y, JUELS A, OPREA A, et al. HomeAlone: co-residency detection in the cloud via side-channel analysis[C]//IEEE Symposium on Security and Privacy. 2011: 313-328.

[15] 余思, 桂小林, 张学军, 等. 云环境中基于 cache 共享的虚拟机同驻检测方法[J]. 计算机研究与发展, 2013, 50(12): 2651-2660.
YU S, GUI X L, ZHANG X J, et al. Co-residency detection scheme based on shared cache in the cloud[J]. Journal of Computer Research and Development, 2013, 50(12): 2651-2660.

[16] LIU F, GE Q, YAROM Y, et al. CATalyst: defeating last-level cache side channel attacks in cloud computing[C]//IEEE International Symposium on High Performance Computer Architecture. 2016: 406-418.

[17] 梁鑫, 桂小林, 戴慧珺, 等. 云环境中跨虚拟机的 cache 侧信道攻击技术研究[J]. 计算机学报, 2017, 40(2): 317-336.
LIANG X, GUI X L, DAI H J, et al. Cross-VM cache side channel attacks in cloud: a survey[J]. Chinese Journal of Computers, 2017, 40(2): 317-336.

[18] 王国峰, 刘川意, 潘鹤中, 等. 云计算模式内部威胁综述[J]. 计算机学报, 2017, 40(2): 296－316.
WANG G F, LIU C Y, PAN H Z, et al. Survey on insider threats to cloud computing[J]. Chinese Journal of Computers, 2017, 40(2): 296-316.

[19] WANG L, LIU W, KUMAR N, et al. A novel covert channel detection method in cloud based on XSRM and improved event association algorithm[J]. Security and Communication Networks, 2016, 9(16): 3543-3557.

[20] IRAZOQUI G, EISENBARTH T, SUNAR B, et al. S$A: a shared cache attack that works across cores and defies VM sandboxing-and its application to AES[C]//IEEE Symposium on Security and Privacy. 2015: 591-604.

[21] 沈晴霓, 李卿. 云计算环境中的虚拟机同驻安全问题综述[J]. 集成技术, 2015, 4(5): 5-17.
SHEN Q N, LI Q. Review on co-residency security issues of virtual machines in cloud computing[J]. Journal of Integration Technology, 2015, 4(5): 5-17.

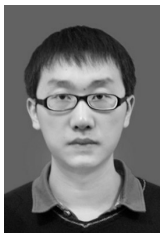
[22] OSVIK D A, SHAMIR A, TROMER E. Cache attacks and countermeasures: the case of AES[C]//Cryptographers' Track at the RSA Conference. 2006: 1-20.

[23] YAROM Y, FALKNER K. FLUSH+RELOAD: a high resolution, low noise, L3 cache side-channel attack[C]//Usenix Security Symposium. 2014: 719-732.

[24] MAURICE C, SCOUARNEC N L, NEUMANN C, et al. Reverse engineering intel last-level cache complex addressing using performance counters[C]//International Symposium on Recent Advances in

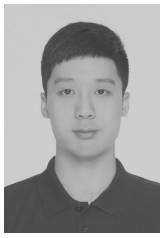
Intrusion Detection. 2015: 48-65.
[25] IRAZOQUI G, EISENBARTH T, SUNAR B, et al. Systematic reverse engineering of cache slice selection in intel processors[C]//Euromicro Conference on Digital Systems Design. 2015: 629-636.
[26] WU Z, XU Z, WANG H, et al. Whispers in the hyper-space: high-bandwidth and reliable covert channel attacks inside the cloud[J]. IEEE/ACM Transactions on Networking, 2015, 23(2): 603-615.
[27] PAAR C, PELZL J. Understanding cryptography: a textbook for students and practitioners[M]. Springer Science & Business Media, 2009.
[28] LI P, GAO D, REITER M K, et al. Replica placement for availability in the worst case[C]//International Conference on Distributed Computing Systems. 2015: 599-608.
[29] MOON S, SEKAR V, REITER M K, et al. Nomad: mitigating arbitrary cloud side channels via provider-assisted migration[C]//ACM Conference on Computer and Communications Security. 2015: 1595-1606.
[30] LIU W, GAO D, REITER M K. On-demand time blurring to support side-channel defense[C]//European Symposium on Research in Computer Security. 2017:210-228.
[31] NEEDLEMAN S B, WUNSCH C D. A general method applicable to the search for similarities in the amino acid sequence of two proteins[J]. Journal of Molecular Biology, 1970, 48(3): 443-453.
[32] VARADARAJAN V, ZHANG Y, RISTENPART T, et al. A placement vulnerability study in multi-tenant public clouds[C]//Usenix Security Symposium. 2015:913-928.

[作者简介]



刘维杰（1991–），男，湖北武汉人，武汉大学博士生，主要研究方向为虚拟化安全、图像处理等。

王丽娜（1964–），女，辽宁营口人，博士，武汉大学教授、博士生导师，主要研究方向为多媒体安全、云计算安全、可信计算等。



王丹磊（1992–），男，湖北武汉人，武汉大学硕士生，主要研究方向为机器学习、信息内容安全等。

尹正光（1989–），男，江西吉安人，硕士，阿里云计算有限公司高级开发工程师，主要研究方向为 cloud native 和 IaaS 架构等。

付楠（1993–），女，江西九江人，武汉大学硕士生，主要研究方向为网络安全、云计算安全等。