云计算环境中的虚拟机同驻安全问题综述

沈晴霓 李 卿

（北京大学软件与微电子学院 北京大学网络与软件安全保障教育部重点实验室 北京 100871）

摘 要 在云计算环境中，为了实现资源共享，不同租户的虚拟机可能运行在同一台物理机器上，即虚拟机同驻，这将带来新的安全问题。为此，文章重点讨论同驻虚拟机所面临的一些新的安全威胁，包括资源干扰、隐蔽通道/侧信道、拒绝服务与虚拟机负载监听等，介绍现有虚拟机同驻探测方法，总结针对虚拟机同驻威胁的四种防御思路，并分析未来的研究趋势。

关键词 云计算；虚拟化；虚拟机；同驻；安全

中图分类号 TP 309 **文献标志码** A

Review on Co-residency Security Issues of Virtual Machines in Cloud Computing

SHEN Qingni LI Qing

(*MoE Key Lab of Network and Software Assurance of Peking University*, *School of Software and Microelectronics*, *Peking University*, *Beijing* 100871, *China*)

Abstract In cloud computing, in order to achieve resource sharing, virtual machines (VMs) of different tenants might be scheduled to run on the same physical machine, namely VMs co-residency, which would bring many new security issues. Therefore, security threats due to VMs co-residency, including resources interference, covert or side channel, denial of service and virtual machine load monitoring were reviewed in this paper. Besides, existing detection methods of co-residency were introduced, four kinds of defense about VMs co-residency were summarized and further trends were also pointed out.

Keywords cloud computing; virtualization; virtual machine; co-residency; security

1 引 言

云计算具有资源配置动态化、需求服务自助化、以网络为中心、服务可计量化、资源池化和透明化等特点，不仅节省了应用成本，提高了计算能力，而且还降低了能源的消耗。目前主流的基础设施即服务（Infrastructure as a Service，IaaS）平台包括：亚马逊的弹性计算云（Elastic Compute Cloud，EC2）[1]、Google 的谷歌计算引擎[2]，微

收稿日期：2015-06-22 修回日期：2015-07-29
基金项目：国家自然科学基金（61073156，61232005）；国家高技术研究发展计划（2015AA016009）；深圳市科技计划（JSGG20140516162852628）
作者简介：沈晴霓（通讯作者），博士，教授，研究方向为操作系统与虚拟化安全、云计算与大数据安全、可信计算，E-mail：qingnishen@ss.pku.edu.cn；李卿，硕士，研究方向为云计算安全。

软的 Azure[3]，Rackspace 的 Mosso[4]以及开源的 OpenStack[5]等，它们将底层的计算资源虚拟化，为上层的租户提供强大的计算能力。

在云计算技术方面，研究者们最初关注的主要是虚拟机间高效通信问题[6-7]，提出了 XenLoop[8]、XenSocket[9]、Xway[10]、IVC[11]以及 MMNet[12]等一系列解决方案。但是，随着云计算技术的发展，云计算安全问题越来越引起人们重视[13-17]，特别是美国加州大学圣迭戈分校的 Ristenpart 等[17]在 2009 年首次指出了云计算环境中虚拟机同驻安全问题，即云服务供应商为了有效利用物理资源，常将不同租户的虚拟机分配给同一台物理机器(称为同驻)。这就导致攻击者能够有机会将自己的虚拟机与受害者的虚拟机同驻，从而威胁受害者虚拟机资源的机密性和可用性等。近年来，人们研究发现多种同驻安全威胁[18-24]，包括资源干扰、隐蔽通道/侧信道、拒绝服务与虚拟机负载监听等。例如：美国马萨诸塞大学阿默斯特分校[18]对同驻虚拟机间的资源干扰进行了细致的分析与实验；美国威斯康星大学麦迪逊分校[19]基于同驻虚拟机间的资源冲突问题，提出了资源释放攻击；美国密歇根大学[20]对同驻虚拟机间的第二级高速缓存(L2 Cache)隐蔽通道/侧信道问题进行了深入的分析；美国东北大学[21]对虚拟机 VCPU 公平调度器的脆弱性进行了分析，指出了针对同驻虚拟机 VCPU 进行的拒绝服务攻击等。针对上述威胁，研究者们又陆续提出了一系列同驻探测方法和防御技术[22-26]。例如，美国俄勒冈大学[22]基于同驻虚拟机的网络包时延问题，提出了同驻水印探测技术；美国北卡罗来纳大学和 RSA 实验室[23,24]分析了 L2 Cache 问题，提出了一种探测虚拟机同驻的工具——HomeAlone；微软公司[27]基于最低级高速缓存的隐蔽通道/侧信道问题，提出了提高虚拟机资源隔离性的防御对策；美国孟菲斯大学[28]针对同驻虚拟机的网络通信拒绝服务攻击，提出了一种基于博弈论的防御机制。

2 虚拟机同驻安全威胁

同驻虚拟机的安全问题是基于这样一个假设：云平台的 IaaS 基础架构和云服务供应商都是可信的，而云服务的使用者(租户)之间是互不信任的。攻击者指的是利用云平台进行攻击的恶意租户，而受害者指的是利用云平台提供的服务执行某些机密性相关程序的正常租户。像其他普通租户一样，攻击者可以在云平台中开启并控制多个虚拟机实例，由于云服务供应商对物理资源需要有效利用，可能会将攻击者的实例与受害者实例分配到同一台物理机器上，这样攻击者就可以利用共享的物理资源(如 CPU、内存、磁盘和网络等)窃取受害者的私密数据或者达到拒绝服务目的等。

虚拟机同驻威胁[17]如图 1 所示，通常地，攻击者首先针对受害者虚拟机所属区域和类型，租用和启动大量同区域和同类型的虚拟机实例，然后再利用这些虚拟机实例，探测其中哪些实例与受害者虚拟机同驻，之后再通过这些同驻的实例对受害者虚拟机及其平台实施攻击。目前，虚拟机同驻可能带来的安全威胁主要包括：(1)对受害者虚拟机资源的干扰，包含 CPU、磁盘和网络资源的干扰；(2)利用虚拟机间共享资源(如 L2 Cache)构造隐蔽通道/侧信道，获取受害者虚拟机上的机密信息[17,20,29]，例如 RSA/AES 密钥；(3)对受害者虚拟机造成拒绝服务，利用网络传输队列(Transmit and Receive Queue，T/R Queue)[28]或 VCPU 调度器的漏洞[20]等使受害者虚拟机的可用性降低；(4)资源释放，利用同驻虚拟机间存在资源冲突问题，使受害者虚拟机释放资源[19]；(5)监听并探测受害者虚拟机的负载状况[17,22]。

2.1 同驻虚拟机间的资源干扰

尽管虚拟机监控器(Virtual Machine

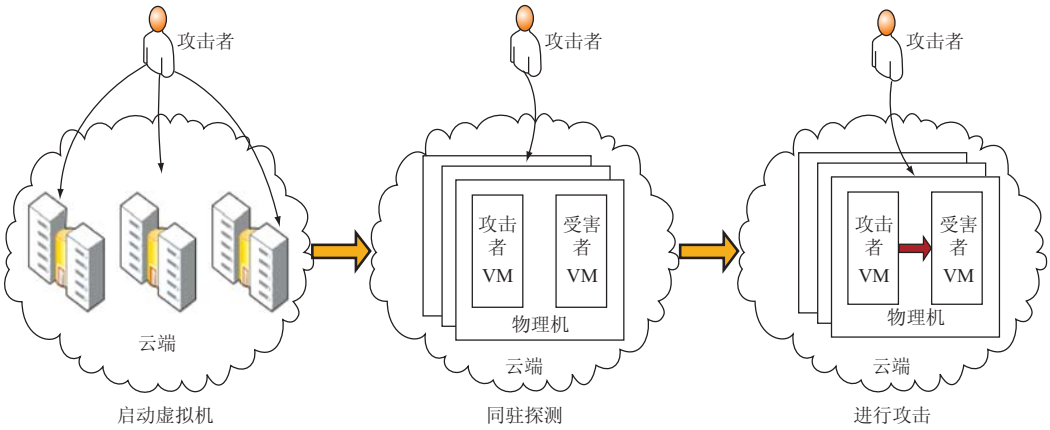

图 1　虚拟机同驻威胁

Fig. 1　Co-residency threats of virtual machine

Monitor，VMM）对虚拟机提供隔离性，但是当资源争用时仍然会导致相互干扰，影响性能[18,30,31]。美国马萨诸塞大学阿默斯特分校的 Barker 等[18]分别从 CPU、磁盘和网络三种资源角度分析了在一个虚拟机上运行的时间敏感型程序是否会受到同驻虚拟机上运行的其他程序的干扰。基于 EC2 公有云平台和实验环境云平台上的实验与分析，他们得到结论：在虚拟机上运行的时间敏感型程序确实会在一定程度上受到其同驻虚拟机上运行程序的干扰，根据竞争资源的不同，其受影响的程度也不同，其中受干扰最明显的是磁盘资源——在同驻虚拟机的干扰下，磁盘密集型程序的运行效率下跌幅度接近 75%。下面分别从 CPU、磁盘和网络三种资源干扰角度分析。

2.1.1 CPU 资源

在同一台物理机器上运行的虚拟机共享 CPU 资源。每个虚拟机创建时，VMM 都会为其分配一个或多个虚拟 CPU，即 VCPU，并且同时创建的虚拟机数量可以大于物理 CPU 核心的数量。VMM 支持灵活的 VCPU 分配方案，可以将 VCPU 与物理 CPU 的任意一个核心进行绑定，不同虚拟机的 VCPU 也可以共享同一个物理 CPU 核心[32-35]。

针对 CPU 资源的干扰，研究者[18]启动多个同驻的虚拟机实例，并在其中一个实例上运行 CPU 密集型测试程序：只执行浮点数运算的线程，不执行输入/输出（Input/Outpit，I/O）操作，并且每 5 分钟记录一次测试程序的运行时间。经验证，测试程序的运行时间分布较为分散，运行时间出现较大的波动，从而可以说明 VMM 分配给该虚拟机的 CPU 资源发生了变化，而这正是由其他同驻虚拟机上的负载变化导致的。

2.1.2 磁盘资源

每个虚拟机都有一个虚拟的磁盘，这些虚拟的磁盘共享实际的物理磁盘，而且 VMM 控制着每个虚拟机对磁盘的 I/O 请求，但是目前的虚拟机技术（如 Xen）还不支持磁盘资源的精确分配[35]。因此，某个虚拟机如果过度或恶意使用磁盘操作，就会干扰与其同驻的虚拟机磁盘 I/O 操作。

针对磁盘资源的干扰，研究者[18,30]启动多个同驻的虚拟机实例，并在其中一个实例上运行磁盘 I/O 密集型的测试程序，测试两种情形：（1）测试程序流式读取/写入磁盘 5 MB 大小的文件；（2）测试程序随机读取/写入一个 1 KB 大小的磁盘块，每 5 分钟记录一次测试程序的运行时间。经验证，无论是流式的读写大规模磁盘数

据，还是随机的读写小块数据，测试程序的执行时间都呈现出分散的状态，尤其以写数据时最为明显。

2.1.3 网络资源

同驻的虚拟机也共享物理主机上的网络资源，因为它们的网络通信都是经由 VMM 管理的网桥连接到外部的物理网卡。VMM 提供一种网络带宽管理工具——tc，通过它可以控制每个虚拟机可用的最大网络带宽，严格限制每个虚拟机只能使用已分配的带宽；也可以对带宽进行灵活设置，即一旦某个虚拟机需要的带宽增加而其他虚拟机需要的带宽较小时，可以将分配给其他虚拟机的带宽再分配[32,35]。

针对网络资源的干扰，研究者[18,31]在实验平台中启动多个同驻虚拟机实例，其中一个部署游戏服务器，同时在外部运行一个收集游戏服务器时延的应用程序 qstat，这里的服务器时延只和网络状况有关，统计的是不同干扰条件下的平均网络时延和超时丢包率，如表 1 所示。当其他同驻虚拟机也在运行网络相关程序时，部署的游戏服务器的网络时延明显增加，同时也出现了一定的超时丢包率。其中 tc 工具采取的网络带宽控制策略不同也会对网络时延和超时几率有一定影响：当采用严格的带宽分配机制时，时延较低，但超时丢包率较高；当采用灵活的带宽分配机制时，超时丢包率较低，但时延较高。

表 1 网络资源干扰

Table 1 Network Resource Interference

干扰项	平均时间（ms）	超时丢包率（%）
空闲	8.1	0
CPU＋磁盘	6.2	1.7
网络（无 tc）	N/A	100
网络（tc，严格）	23.6	6.7
网络（tc，灵活）	33.9	1.7

2.2 LLC 隐蔽通道/侧信道

云平台中隐蔽通道/侧信道也发生在虚拟机同驻的条件下。Osvik 和 Shamir 等[36,37]在 2006 年以 AES 为例，分析了利用高速缓存的隐蔽通道/侧信道攻击和对抗方案，这种通道在云平台中依然存在，但是由于云平台存在更复杂的影响因素，如 CPU 核心迁移、粗粒度的调度算法、两级内存映射以及其他虚拟机的干扰，使得该通道实现起来难度更大也更复杂。

美国加州大学圣迭戈分校的 Ristenpart 等[17]最早提出了云平台中基于最低级高速缓存（Last Level Cache，LLC）的虚拟机间隐蔽通道/侧信道。但他们只是简单讨论了该隐蔽通道/侧信道的构造以及可能带来的威胁，并没有对其进行详细的分析。美国密歇根大学的 Xu 与 AT&T 研究实验室的 Joshi 等以 L2 Cache 为例，详细给出了其在云平台下的实现方案[20]，测量了带宽，并评估该通道可对云平台造成实际危害。除了在 X86 架构上的研究，德国弗劳恩霍夫研究所的 Weiß 等[29]也研究了基于嵌入式 ARM 架构的虚拟化平台中的 Cache 隐蔽通道/侧信道问题。

2.2.1 通信方案

LLC 隐蔽通道的通信方案[17]：首先，将所有的 Cache Line 分为两个子集合——集合 a 和集合 b。当发送方发送 1 bit 数据时，发送方访问与某个 Cache Line 子集合相映射的内存地址，这样将会使得与该子集合相关的接收方 Cache 内容从 Cache Line 中挤出，而与另一子集合相关的部分不变。然后，接收方分别访问与这两个子集合相映射的内存地址，通过比较访问时间的差别对发送的信息进行解码。如果访问子集合 a 的时间明显大于访问子集合 b 的时间，则发送的是 1，否则发送的是 0。最初实测该通道的带宽仅有 0.2 bps，这意味着要传送一个 2 048 位的 RSA 私钥（实际占用 1743 字节）需要花费 20 个小时之久。Xu 等[21]改进了通信方案，使其在实验环境下的理论带宽可以达到 262.47 bps，这意味着传输一个 2 048 位的 RSA 私钥仅需 53 秒。

2.2.2 应用场景

在云平台上，LLC 侧信道有多种应用场景。例如，利用 LLC 探测同驻虚拟机的网络负载状况[17]、基于 LLC 侧信道探测同驻的虚拟机[23]、结合 LLC 侧信道实现 Keystroke Timing 攻击[38]等。

前两种将在 2.5 和 3.3 节中介绍，这里重点介绍利用 L2 Cache 侧信道实现的 Keystroke timing 攻击。该攻击的基本思想如下：首先，攻击者利用 L2 Cache 侧信道技术，探测由同驻受害者键入 SSH 终端密码等敏感信息而引起的 Cache 变化；然后，攻击者利用这些信息恢复出受害者键入的敏感信息。具体地，攻击者首先申请一块和 Cache 大小相同的内存，并通过访问这个内存填充 L2 Cache，之后不断访问当前物理机器的 Cache，当同驻虚拟机上的受害者键入敏感信息到 shell 时，攻击者就可以探测到自己访问 Cache 的时间出现了变化，即探测到了 Keystroke timing。这样攻击者就可以利用已有的 Keystroke timing 攻击技术恢复受害者键入的信息。该方案基于这样一个假设：在一台物理机器上，Cache 负载上出现的某个峰值就意味着用户向同驻虚拟机的 SSH 终端键入了一个单词或者命令。

2.3 拒绝服务攻击

虚拟机同驻安全威胁还包括基于共享资源的拒绝服务(Denial of Service，DoS)。由于在虚拟化环境下，运行在同一台物理机器上的虚拟机共享底层物理资源(如 CPU、内存、网络、磁盘等)，若攻击者试图绕过 VMM 的管理，就可以使得受害者虚拟机无法正常使用物理资源，破坏同驻虚拟机的可用性，达到拒绝服务攻击目的。本文将讨论针对网络通信的拒绝服务攻击[26]以及针对 VCPU 的拒绝服务攻击[21]。

2.3.1 网络通信拒绝服务攻击

美国孟菲斯大学的 Bedi 等[28]研究同驻虚拟机网络资源的共享问题，提出了一种通过阻塞网络传输队列达到拒绝服务攻击的方案，并提出了一种基于博弈论(Game Theoretic)的防御机制。

图 2 描述了 Xen 的网络架构[35]，其上虚拟机在通过网卡(Network Interface Card，NIC)使用网络时均共享同一个 T/R Queue，而 T/R Queue 则易受网络阻塞的影响，会引发潜在的拒绝服务攻击。具体方案如下[28]：攻击者通过部署一系列与受害者同驻的恶意虚拟机耗尽物理机的网络带宽，这些恶意虚拟机不断使用网络发送无意义的网络数据，由于受害者虚拟机与恶意虚拟机共享同一条 T/R Queue，受害者正常的网络通信就会被占用，无法使用 T/R Queue 进行通信。由于每一台恶意的虚拟机占用的带宽相对并不高，使得云服务供应商很难检测到这种网络通信拒绝服务攻击。

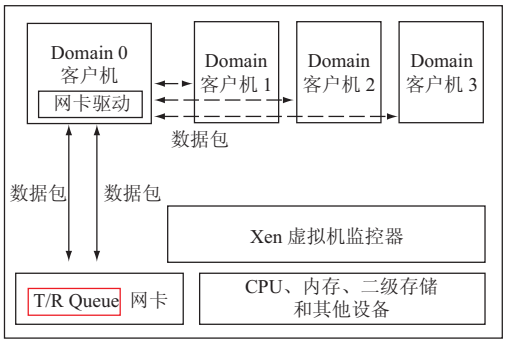

图 2　Xen 虚拟化网络架构

Fig. 2　Virtualized network architecture of Xen

利用博弈论对该问题进行建模，可实现一种防御方案，该模型中有两个角色：攻击者和受害者。攻击者的动作集包括选择实施攻击的虚拟机数量以及每台虚拟机攻击时占用的网络带宽；受害者动作集是设定防火墙的阈值来防止恶意虚拟机对网络通信的阻塞。一旦某台虚拟机使用的网络带宽超过阈值，特权虚拟机(Domain 0)就中断其网络通信，目的是使 Domain 0 能够选取最优阈值以中断掉尽可能多的恶意虚拟机网络带宽，保护合法用户对网络带宽的正常使用。

2.3.2 VCPU 拒绝服务攻击

Xen 默认采用基于额度的 VCPU 调度算法

(Credit Scheduler)[32-34]。它可以应用于对称多处理器(Symmetric Multi Peocessor，SMP)系统中，也可以支持连续工作和断续工作两种模式。美国东北大学的 Zhou 等[21]分析了 Xen 的 Credit 调度算法，并发现了其存在的一个漏洞，利用此漏洞可构造对同驻虚拟机的 VCPU 拒绝服务攻击。Credit 调度算法[32-34]是一种按比例公平共享的非抢占式调度算法。它将各个 VCPU 分为 under 队列和 over 队列，只调度 under 队列中的 VCPU。最开始，所有的 VCPU 都在 under 队列，每个虚拟机的初始 credit 为其对应的 weight 值。每当 VCPU 被调度时，其对应虚拟机的 credit 就会减少，当 credit 为负数时，它被放入 over 队列。VCPU 每执行一个调度周期(10 ms)，credit 值都需要重新计算，并且一次最多可以运行 3 个周期，即 30 ms。之后无论 credit 是否为正，都要让出物理 CPU。同时，为了解决响应延迟时间过长的问题，credit 调度算法加入了一个 BOOST 状态，处于 BOOST 态的虚拟机 VCPU 具有最高的调度优先级。

上述 Credit 算法存在两个问题[21]：(1)算法的粒度太粗。每 10 ms 调度一次，只在调度时进行检测，在运行期间不做检测。如果一个 VCPU 被调度上物理 CPU，在 10 ms 内时放弃运行(被 I/O 阻塞或其他原因)，则在 10 ms 结束时调度器没有检测到该 VCPU，不会减少其额度。(2) Credit 调度器无法区别正在运行的 VCPU 是主动让出物理 CPU 还是被 I/O 阻塞，在这个 VCPU 被唤醒时，都会被设置 BOOST 标志，获得最高调度优先级。图 3 为利用上述问题的拒绝服务攻击流程。其中恶意虚拟机运行一个 VCPU 攻击程序，该程序一旦被调度上物理 CPU，在运行 $10-\varepsilon$ ms 后调用 Halt()函数使自己运行 idle 进程，主动让出物理 CPU，使调度器调度其他同驻虚拟机的 VCPU 在物理 CPU 上运行。在该运行周期快结束之前主动唤醒攻击程序，由 Xen 置 BOOST 位而获得最高调度优先级，下一个调度时刻将再次被调度上物理 CPU，而且不会被减少额度。如此反复，攻击程序所在的 VCPU 可以一直获得最高调度优先级，在每个运行周期都运行 $10-\varepsilon$ ms，而物理机器上其他所有虚拟机只能分享剩下的 ε ms，便实现了对同驻虚拟机进行 VCPU 拒绝服务攻击。

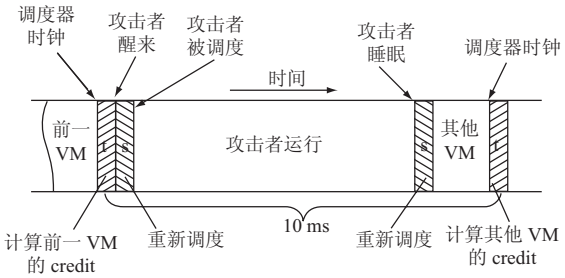

图 3 VCPU 拒绝服务攻击

Fig. 3 Scheme of VCPU DoS attack

2.4 资源释放攻击

来自美国威斯康星大学麦迪逊分校的 Varadarajan 等[19]在分析同驻虚拟机资源冲突的基础上，提出了一种新的攻击类型，即资源释放攻击(Resource-Freeing Attack)，通过改变受害者虚拟机上的资源使用状况后释放其资源给同驻攻击者的虚拟机使用。该方案在实验环境下提升了攻击程序 60% 的性能，在亚马逊 EC2 环境中也获得 13% 性能提升。

2.4.1 攻击原理

资源释放攻击[19]要求：(1)使得受害者虚拟机对某个资源的使用达到瓶颈；(2)改变受害者的资源使用状况，使得受害者的绝大部分处理时间都花费在瓶颈资源上，无法使用其他资源。

同驻虚拟机间资源的争用和干扰是实现资源释放攻击的基础。在同一个物理机器上可能会造成资源争用的资源有如下几类：CPU、Cache、磁盘、内存以及网络。经实验发现，磁盘资源和 LLC 在发生竞争时受到的干扰较为明显。其中，LLC 资源在同驻虚拟机运行网络密集型程序时受到

的干扰最大，所以资源释放攻击选取的攻击程序为 LLC 密集型应用程序，而受害者为运行网络密集型应用程序的虚拟机。资源释放攻击通过修改受害者资源使用情况，使其处于处理网络请求的瓶颈状态，从而提升 LLC 密集型攻击程序的性能。

2.4.2 攻击过程

图 4 中的 beneficiary 指在攻击虚拟机上运行的 LLC 密集型程序；victim 指在受害者虚拟机上运行的 Apache Web Server；load generator 指处于物理节点外，向受害者虚拟机发送静态网页请求的代理终端；helper 指处于物理节点外，向受害者虚拟机发送公共网关接口（Common Gateway Interface，CGI）脚本请求的代理终端。由于受害者虚拟机在处理静态网页请求时抢占了 beneficiary 与 victim 共享的 LLC，使得 beneficiary 程序执行时间显著增加，程序性能下降。因此，攻击者通过使用外部程序 helper 向受害者虚拟机发送 CGI 脚本请求，使受害者处理 CGI 请求时使用的 CPU 资源达到瓶颈状态，导致其所能响应的、由 load generator 产生的正常网页请求效率显著下降，从而减少了对物理机器上 LLC 资源的使用。由于受害者虚拟机资源使用状况的转移，攻击程序 beneficiary 的性能就得以提高。

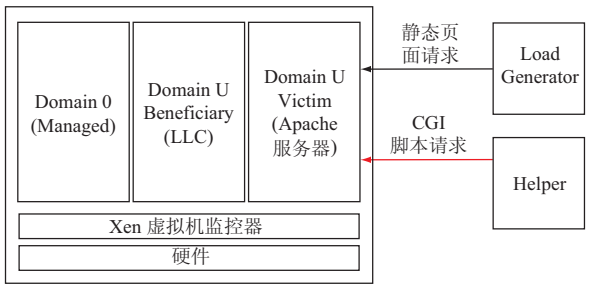

图 4　资源释放攻击

Fig. 4　Scheme of resource-freeing attack

2.5　同驻虚拟机负载探测

同驻虚拟机的负载探测也存在着一些安全隐患，如可以估算受害者虚拟机中网页服务器的访问用户量或探测哪一个页面被频繁访问[17]。通常这些信息是私密的，如果攻击者与受害者存在商业竞争关系，那么泄露这些信息将会造成严重危害。目前虚拟机负载探测方法主要有两种：基于 Cache 侧信道的负载探测[17,20]和通过网络 TCP 会话的负载探测[22]。

2.5.1 基于 Cache 侧信道的探测

利用 2.2 节 Cache 侧信道，可以探测同驻虚拟机的负载状况。它通过测量自己访问 Cache 映射地址的时间估算出同驻虚拟机的 HTTP 通信带宽，具体方法为[17,20]：建立两台同驻虚拟机，在一台攻击者虚拟机上执行 Cache 测量程序，在另一台受害者虚拟机上运行一个 Web Server；外部用户向 Web Server 发送不同速率的 HTTP 请求，分别为每分钟 0 次、50 次、100 次、200 次等。实验表明：攻击者虚拟机上测量到的 Cache 映射地址的访问时间与 HTTP 发送请求频率有明显的关系，这样便可以估算出受害者虚拟机的网络负载状况。

2.5.2 通过网络 TCP 会话的探测

通过网络 TCP 会话的同驻水印技术[22]（见 3.3 节），可以利用外部应用程序和受害者 Web Server 之间 TCP 会话的通信量来估算其负载状况。具体方法如下：首先，攻击者利用同驻水印技术确认自己的虚拟机与受害者虚拟机运行在同一个物理平台上；然后，攻击者自己也运行 Web Server，并在平台外部开启一个应用程序，这个应用程序同时与攻击者和受害者虚拟机初始化一个 TCP 会话，外部程序通过测量 TCP 会话流，能够得到其分别与攻击者和受害者通信带宽的比例，设为 R。除非这两台虚拟机自身负载出现变化，物理平台上其他虚拟机的负载变化并不会影响 R，因为其他虚拟机对这两个虚拟机的影响效果是等同的；网络阻塞的影响也可以忽略，因为攻击者和受害者共享同一网络路径，R 保持不变。这样，由于攻击者的负载由攻击者控制并保持恒定，比例 R 的波动就表明受害者虚拟机上的

负载出现了变化。

3 虚拟机同驻探测技术

云平台中的虚拟机同驻探测，既可以指攻击者通过已知的探测方法确认自己的虚拟机与受害者虚拟机在同一个物理平台上运行，也可以指某租户通过已知的探测手段确认是否与其他租户的虚拟机在同一台物理机器上运行。虚拟机同驻探测技术主要有三种：(1)基于网络信息的同驻探测技术[17]；(2)同驻水印技术(Co-Residency Watermarking)[22]；(3)利用 L2 Cache 的 HomeAlone[23]。

3.1 基于网络信息的同驻探测技术

通过网络信息探测技术[17]，如果两个虚拟机具有相同的 Domain 0 的 IP 地址，或具有很短的网络包往返时间(Round-Trip Times，RTTs)，或具有数字上接近的内部 IP 地址，则可说明这两个虚拟机是同驻的。

对于方法(1)，在 Xen 中，每个 Guest 虚拟机在进行网络通信时的第一跳地址是 Domain 0 的 IP 地址，攻击者可以据此确定当前物理平台上 Domain 0 的 IP 地址。此外，攻击者可以利用外部的一台虚拟机向受害者虚拟机发送一个 TCP SYN 追踪路由，并探测最后一跳的 IP 地址，通过它即可获得受害者虚拟机所在物理平台上 Domain 0 的 IP 地址。若后者与前者 IP 地址相同，就说明这两个虚拟机实例可能是同驻的。

对于方法(2)，运行在同一物理平台上的一个虚拟机向另一个虚拟机发送网络包时，由于不需云平台路由，只需物理机内部路由，所以网络包的 RTTs 相比非同驻情况短很多。因此，攻击者可以通过向不同的虚拟机多次发送网络包，并记录每次探测的平均包往返时间，其中平均包往返时间最短的那个虚拟机就很有可能是与攻击者同驻。

对于方法(3)，由于每个虚拟机实例创建时都会根据如下算法分配到一个内部 IP 地址：具有相同 Domain 0 IP 地址的虚拟机实例的内部 IP 地址按照一定线性递增顺序进行分配。由于具有相同 Domain 0 IP 地址的虚拟机是同驻的，这样在数字上较为接近的内部 IP 地址的虚拟机可能是同驻的。

实际上，为了提高准确率，通常综合利用以上三种方法：先利用方法(3)比较两个虚拟机实例的内部 IP 地址，若 IP 地址数字接近，再利用方法(1)确定虚拟机的 Domain 0 IP 地址，若相同，则说明两个虚拟机同驻。为了增加探测的准确性，还可以进一步利用方法(2)测量虚拟机间的网络包往返时间。

3.2 同驻水印技术

在实际云平台中，云服务供应商可以通过多种方法阻止攻击者利用网络信息进行同驻探测，如设置 Domain 0 对外部发起的路由追踪不作应答，在创建虚拟机实例时随机地分配内部 IP 地址，利用虚拟 LAN 隔离不同的租户等。这样就不得不去寻找更为有效的同驻探测方法，美国俄勒冈大学 Bates 等[22]利用同驻虚拟机共用网卡的特性提出了一种更为有效的同驻探测方案——同驻水印技术。它是基于物理网卡多路复用带来的网络包延时问题[39]提出的。具体地，进行探测的虚拟机周期性地向目标虚拟机的网络包中注入水印标记，干扰其正常网络数据包的发送，之后通过测量收集目标虚拟机的网络包通信状况，判断自己是否与目标虚拟机同驻。

假定 SERVER 是在云平台中一个物理机器上运行的虚拟机实例，FLOODER 是攻击者为了同驻探测而创建的虚拟机实例，CLIENT 是在云平台外运行、与 FLOODER 串通好的代理终端。则探测方法如下：首先，CLIENT 与已知 IP 的 SERVER 建立一个 TCP 会话。接着，CLIENT 周期性地向 FLOODER 发送信号，FLOODER 接收

到信号后去占用物理机器网卡，并向外发送无意义的 UDP 包。若 FLOODER 与 SERVER 是同驻的，由于它们对同一物理主机网卡的多路复用，就会对正常的 CLIENT-SERVER 网络通信数据流带来一定延时，这个延时即同驻水印。最后，通过收集 CLIENT-SERVER 每个时间间隔的网络包数量，分析有水印时间间隔和无水印时间间隔收到网络包的数量分布，可以判断 FLOODER 与 SERVER 是否同驻。

3.3　HomeAlone 技术

如 2.3 节所述，LLC 隐蔽通道/侧信道应用于秘密传递信息。但是，美国北卡罗来纳大学和 RSA 实验室的 Zhang 等[23,24]则将它用作防御方案，即租户使用 HomeAlone[23]工具验证云服务提供商是否遵循了 SLA 协议，租户是否独占了某个物理平台上的资源，而没有让其他租户虚拟机与其同驻。

HomeAlone 通过让租户在其独占物理机器上创建一些友好虚拟机实例(friendly VMs)，并约定在某个时间周期内，它们均不使用选定的 Cache 区域映射地址，并测量该周期内 Cache 映射地址的访问时间，若发现 Cache 使用状况发生变化，则认为有不期望的虚拟机(foe VM)与租户的虚拟机发生同驻。该技术实现的难点在于准确地区分 friendly VMs 与同驻 foe VMs 的 Cache 行为，以及保证 friendly VMs 性能不会降低。

HomeAlone 架构如图 5 所示，由三个子部件构成：运行在用户态的 coordinator、运行在内核态的 address remapper 和 co-residency detector。HomeAlone 工具安装在每个 friendly VMs 中，仅需修改 friendly VMs 内核，无需对 hypervisor 修改或云服务供应商支持。探测周期开始时，第一个 coordinator(称为 initiator)首先启动，并确定某个染色的 Cache 集(即在探测周期所有 friendly VMs 都要减少访问的 Cache 区域），并将这条命令发送给其他 friendly VM 的 coordinator。其他 coordinator 接收到命令后会调用 address remapper 空出相应的 Cache 集，并尽量少访问这个区域。一旦 address remapper 完成了 Cache 集的空出操作，coordinator 就向 initiator 发送确认信息。Initiator 收到所有 coordinator 发回的确认信息后创建一个 token，并随机选择一个 friendly VM，将这个 token 发送给它。该 friendly VM(称为 token holder)调用 co-residency detector 进行 Cache 访问状况的测量，Token holder 收集测量结果 r，并将 token 和 r 发送给另一个 friendly VM，如此执行 n 次之后，最后那个虚拟机对收集的结果进行分析，并对物理平台上是否存在同驻的 foe 虚拟机做出判断。

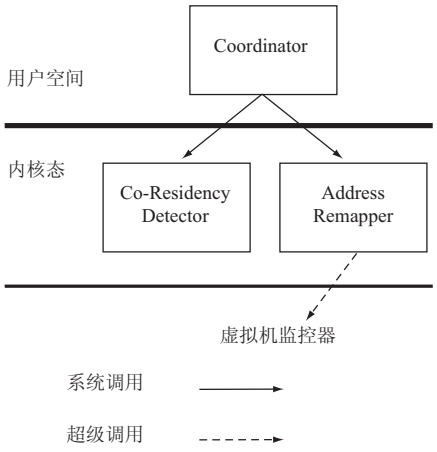

图 5　HomeAlone 架构

Fig. 5　Architecture of HomeAlone

4　虚拟机同驻防御技术

虚拟机同驻可能破坏云平台中数据的机密性和资源的可用性，会产生极大安全威胁问题。为了有效应对这类安全问题，本节将从四个方面讨论同驻防御技术：(1)加入安全机制防御云平台中的 LLC 隐蔽通道/侧信道；(2)采用断续工作型调度策略增强同驻虚拟机之间的隔离性；(3)借鉴多核心处理器芯片系统(CMPs)实现防干扰的

进程调度算法；(4)硬件协同防御技术。

4.1 防御同驻虚拟机间隐蔽通道/侧信道

2.3 节的分析已经证明，云平台中的 LLC 隐蔽通道/侧信道会破坏多租户虚拟机之间的隔离性。微软公司的 Raj 等[27]针对 Hyper-V[40]提出了两种防御 LLC 隐蔽通道/侧信道的资源管理方案：基于 Cache 的处理器分配策略以及基于 Cache 的内存页染色技术。

4.1.1 基于 Cache 的处理器分配策略

SMP(Symmetric Multi-Processing)通常是在 package 层共享 LLC，而且许多云计算平台的服务器都配置有多个 package。这样，可以利用 SMP 特性对虚拟机分配处理器核心，使它们互相不共享 Cache。具体做法如下：首先，根据 LLC 的共享情况，将物理机上的处理器核心进行分组，所有共享同一 LLC 的处理器核心放入一组；然后，分配算法在为某个虚拟机分配物理 CPU 时，会从还未分配给任何其同驻虚拟机的组中选择一个处理器核心进行分配，并且这个组中的其他处理器核心也只允许分配给这个虚拟机的其他 VCPU。

通过该方案，同驻的虚拟机不再共享 LLC，也就不会存在 LLC 隐蔽通道/侧信道的安全问题。但是这种方案也存在一个致命的缺陷，即物理平台的 CPU 资源并没有得到充分利用。比如一个虚拟机申请的 VCPU 小于某个组中的处理器核心数量时，那些未分配的处理器核心将永远得不到使用。

4.1.2 基于 Cache 的内存页染色技术

内存页染色技术[42,43]是一种软件方法，它通过控制应用程序使用的物理内存来间接保证同驻虚拟机不共享 Cache 集。内存页染色的颜色数量通常是由 Cache Line 的大小乘以 Cache 集的数量再除以内存页大小计算得到。在虚拟化环境中，hypervisor 对虚拟机内存页面的分配会影响虚拟机中应用程序对 Cache 的使用。基于 Cache 的内存页染色技术，通过隔离内存页的颜色集来隔离共享 LLC 处理器核心对 Cache 的使用。具体来说就是修改并控制 virtual-to-physical 内存映射表，隔离同一个物理平台上不同虚拟机对内存页的访问(一个虚拟机只能访问某一种颜色的内存页)，并间接隔离虚拟机对 Cache 集的使用。

该方法并不会造成 CPU 资源的浪费，但有可能会引起内存资源的浪费以及虚拟机性能的下降。比如一台虚拟机并没有完全使用分配给它的某一种颜色的内存页，属于该颜色的其他内存页也不会再分配给其他的虚拟机，那么这些内存页面就会一直得不到使用。另外，一台虚拟机的行为可能与 Cache 集的分布不符，该方案可能会使该虚拟机性能下降。

4.2 防同驻的断续工作型调度策略

在虚拟化技术中，资源的调度策略[32]共分为两种：一种是连续工作型调度策略；另一种是断续工作型调度策略。其中连续工作型调度策略指的是 VMM 在分配物理资源时尽量做到对物理资源的充分利用。当某个虚拟机对物理资源的使用超过分配给它的量时，调度器将已经分配给其他相对空闲的虚拟机资源调度给这个虚拟机使用。断续工作型调度策略指的是 VMM 在分配物理资源时支持精确的资源分配。调度器通过给每个虚拟机设置阈值限制其对资源的使用，即使某个虚拟机相对空闲，调度器也不会将属于该虚拟机的资源调度给另外一个繁忙的虚拟机使用。

在防御同驻虚拟机导致网络通信拒绝服务攻击时[28]，可通过设置网络防火墙限制每个虚拟机实例使用带宽，一旦某个虚拟机网络带宽超过设定的阈值，VMM 就中断该虚拟机的网络通信，阻止攻击者对同驻虚拟机的拒绝服务攻击。在防御资源释放攻击时[19]，若采用断续工作型的物理资源分配方案，受害者虚拟机对某个资源的使用就不会达到瓶颈状态，这样攻击者便不能将受害者的 CPU 时间集中于处理该瓶颈资源并实施资

源释放攻击。由此可见，采用断续工作型调度策略可以明显提高同驻虚拟机间的隔离性，削弱同驻威胁。但是，由于它严格限制虚拟机对物理资源的使用，必然存在某些资源未充分利用的缺陷。在开放云平台的虚拟机资源分配过程中，人们正在研究可以防止同驻攻击的安全调度策略[44-46]。

4.3　防同驻与 CMP 进程隔离技术

针对 CMP 上多道程序运行的隔离性和性能提高的调度算法研究[47,48]对减少虚拟机同驻威胁提供了参考。美国西蒙弗雷泽大学的 Fedorova 和哈佛大学 Seltzer 等[47]提出了一种新的 Cache-Fair 调度算法，保证每个程序公平分配 Cache，几乎消除了同驻程序间性能干扰，提高了多道程序运行的隔离性。美国康奈尔大学的 Bhadauria 和瑞典查尔姆斯理工大学的 McKee[48]分析了 CMP 系统中多道程序资源使用的冲突问题，结合性能提升，设计了三种协同调度算法：HOLISYN、Greedy 和 Oracle 等，使用统计方法及处理器性能计数器来探测冲突资源的使用情况，并尝试在不同的时刻调度它们，提高了 CMPs 系统上程序的总吞吐量。

4.4　硬件协同防同驻技术

除了在软件层防御同驻，还可以从硬件设计层做防御[49,50]，这也是未来的重要方向。如前所述，LLC 隐蔽通道/侧信道问题[17,20]是由同一个 package 中的处理器核心共享 Cache 造成的，除了利用内存页染色技术和处理器分配策略之外，也可以从 CPU 设计角度考虑消除该隐蔽通道/侧信道。例如重新设计 CPU 架构，使其严格地控制虚拟机对 Cache 的使用，彻底消除 LLC 隐蔽通道/侧信道。再譬如，利用同驻水印探测虚拟机同驻的方案[22]是基于物理网卡多路复用带来的网络包延时提出，可以考虑重新设计物理网卡，使其支持虚拟化技术中多个虚拟机对网卡的多路复用，为同驻的虚拟机间提供良好的网络隔离性。

5　未来研究趋势

通过分析总结云计算环境中的虚拟机同驻安全问题研究现状以及当前研究方向所存在的问题，可以将未来的研究趋势概括为三大方向：

(1) 针对云计算环境中的虚拟机同驻威胁研究，将在高带宽、低干扰的隐蔽通道/侧信道攻击以及更加隐蔽、潜伏时间更长的拒绝服务攻击等方面进一步深入研究；

(2) 结合 IaaS 云服务采用的不同类型虚拟化实现技术（如 Xen，KVM，DOCER 等）开展研究，建立准确率高、用户可自主验证或者可委托可信第三方验证的虚拟机同驻探测技术；

(3) 虚拟机同驻安全隐患已经成为云服务发展亟待解决的关键问题，业界与学术界期望能够从防同驻的资源调度和负载均衡策略、多核进程隔离技术以及硬件协同设计等角度来提升云服务的可信性。因此，深入研究解决同驻安全问题的软、硬件结合技术是未来的重要发展方向。

6　总　结

随着云计算技术的发展与应用，多租户虚拟机之间的同驻安全威胁日益突出，成为云计算环境中不容忽视、亟待解决的问题。为此，本文总结并分析了近年来业界和学术界对虚拟机同驻安全问题的研究，包括虚拟机同驻所面临的多种安全威胁、虚拟机同驻探测技术以及虚拟机同驻威胁防御方案，并探讨了未来研究趋势。

参 考 文 献

[1] Amazon Elastic Compute Cloud (EC2)[EB/OL]. [2015-03-25]. http://aws.amazon.com/ec2/.

[2] Google Compute Engine (GCE)[EB/OL]. 2013-06-03 [2015-05-28]. http://www.freehao123.com/tag/google-compute-engine/.

[3] Microsoft Azure Services Platform [EB/OL]. 2014-02-10 [2015-06-01]. http://www.microsoft.com/azure/default.mspx.

[4] Rackspace Mosso [EB/OL]. [2014-06-20]. http://www.mosso.com/.

[5] OpenStack [EB/OL]. [2015-06-01]. http://www.openstack.org/.

[6] Gebhardt C, Tomlinson A. Challenges for inter virtual machine communication [R]. RHUL-MA-2010-12, England: University of London, 2010.

[7] Wang J. Survey of state-of-the-art in inter-VM communication mechanisms [J]. Research Proficiency Report, 2009: 1-25.

[8] Wang J, Wright KL, GopalanK. XenLoop: a transparent high performance inter-VM network loopback [J]. Cluster Computing, 2009, 12(2): 141-152.

[9] Zhang X, McIntosh S, Rohatgi P, et al. XenSocket: a high-throughput interdomain transport for virtual machines [M] // Middleware 2007. Springer Berlin Heidelberg, 2007: 184-203.

[10] Kim K, Kim CY, Jung SI, et al. Inter-domain socket communications supporting high performance and full binary compatibility on Xen [C] // Proceedings of the Fourth ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments, 2008: 11-20.

[11] Huang W, Koop MQ, Gao Q, et al. Virtual machine aware communication libraries for high performance computing [C] // Proceedings of the 2007 ACM/IEEE Conference on Supercomputing, 2007: 1-12.

[12] Radhakrishnan P, Srinivasan K. MMNet: an effcient inter-VM communication mechanism [C] // Proceedings of Xen Summit, 2008: 1-3.

[13] Vaquero L, Rodero-Merino L, Morán D. Locking the sky: a survey on IaaS cloud security [J]. Computing, 2011, 91(1): 93-118.

[14] Wei JP, Zhang XL, Ammons G, et al. Managing security of virtual machine images in a cloud environment [C] // Proceedings of the 2009 ACM Workshop on Cloud Computing Security, 2009: 91-96.

[15] 俞能海, 郝卓, 徐甲甲, 等. 云安全研究进展综述 [J]. 电子学报, 2013, 41(2): 371-381.

[16] 冯登国, 张敏, 张妍, 等. 云计算安全研究 [J]. 软件学报, 2011, 22(1): 71-83.

[17] Ristenpart T, Tromer E, Shacham H, et al. Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds [C] // Proceedings of the 16th ACM Conference on Computer and Communications Security, 2009: 199-212.

[18] Barker S, Shenoy PJ. Empirical evaluation of latency-sensitive application performance in the cloud [C] // Proceedings of the First Annual ACM SIGMM Conference on Multimedia Systems, 2010: 35-46.

[19] Varadarajan V, Kooburat T, Farley B, et al. Resource-freeing attacks: improve your cloud performance (at your neighbor's expense) [C] // Proceedings of the 2012 ACM Conference on Computer and Communications Security, 2012: 281-292.

[20] Xu YJ, Bailey M, Jahanian F, et al. An exploration of L2 cache covert channels in virtualized environments [C] // Proceedings of the 3rd ACM Workshop on Cloud Computing Security Workshop, 2011: 29-40.

[21] Zhou FF, Goel M, Desnoyers P, et al. Scheduler vulnerabilities and coordinated attacks in cloud computing [C] // 2011 10th IEEE International Symposium on Networking Computing and Applications, 2011: 123-130.

[22] Bates A, Mood B, Pletcher J, et al. Detecting co-residency with active traffic analysis techniques [C] // Proceedings of the 2012 ACM Workshop on Cloud Computing Security Workshop, 2012: 1-12.

[23] Zhang YQ, Juels A, Oprea A, et al. HomeAlone: co-residency detection in the cloud via side-channel analysis [C] // 2011 IEEE Symposium on Security and Privacy, 2011: 313-328.

[24] Zhang YQ, Juels A, Reiter MK, et al. Cross-VM side channels and their use to extract private keys [C] // Proceedings of the 2012 ACM Conference on Computer and Communications Security, 2012: 305-316.

[25] Godfrey M, Zulkernine M. A server-side solution to cache-based side-channel attacks in the cloud [C] // 2013 IEEE Sixth International Conference on Cloud Computing, 2013: 163-170.

[26] 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.

[27] Raj H, Nathuji R, Singh A, et al. Resource management for isolation enhanced cloud services [C] // Proceedings of the 2009 ACM Workshop on Cloud Computing Security, 2009: 77-84.

[28] Bedi HS, Shiva S. Securing cloud infrastructure against co-resident DoS attacks using game theoretic defense mechanisms [C] // Proceedings of the International Conference on Advances in Computing Communications and Informatics, 2012: 463-469.

[29] Weiß M, Heinz B, Stumpf F. A cache timing attack on AES in virtualization environments [M] // Financial Cryptography and Data Security. Springer Berlin Heidelberg, 2012: 314-328.

[30] Pu X, Liu L, Mei YD, et al. Understanding performance interference of I/O workload in virtualized cloud environments [C] // 2010 IEEE 3rd International Conference on Cloud Computing, 2010: 51-58.

[31] Wang GH, Ng TSE. The impact of virtualization on network performance of amazon EC2 data center [C] // 2010 Proceedings IEEE on INFOCOM, 2010: 1-9.

[32] Chisnall D. The Definitive Guide to the Xen Hypervisor [M]. Boston: Prentice Hall PTR, 2007.

[33] 石磊, 邹德清, 金海. Xen 虚拟化技术 [M]. 武汉: 华中科技大学出版社, 2009.

[34] 沈晴霓, 卿斯汉. 操作系统安全设计 [M]. 北京: 机械工业出版社, 2013.

[35] Barham P, Dragovic B, Fraser K, et al. Xen and the art of virtualization [J]. ACM SIGOPS Operating Systems Review, 2003, 37(5): 164-177.

[36] Osvik DA, Shamir A, Tromer E. Cache attacks and countermeasures: the case of AES [M] // Topics in Cryptology–CT-RSA 2006. Springer Berlin Heidelberg, 2006: 1-20.

[37] Tromer E, Osvik DA, Shamir A. Efficient cache attacks on AES and countermeasures [J]. Journal of Cryptology, 2010, 23(1): 37-71.

[38] Song DX, Wagner D, Tian XQ. Timing analysis of keystrokes and SSH timing attacks [C] // The 10th USENIX Security Symposium, 2001: 337-352.

[39] Whiteaker J, Schneider F, Teixeira R. Explaining packet delays under virtualization [J]. ACM SIGCOMM Computer Communication Review, 2011, 41(1): 38-44.

[40] Hyper-V overview [EB/OL]. 2015-03-31 [2015-06-12]. https://technet.microsoft.com/en-us/library/hh831531.aspx.

[41] 王于丁, 杨家海, 徐聪, 等. 云计算访问控制技术研究综述 [J]. 软件学报, 2015, 26(5): 1129-1150.

[42] Kessler RE, Hill MD. Page placement algorithms for large real-indexed caches [J]. ACM Transactions on Copmuter Systems, 1992, 10(4): 338-359.

[43] Page D. Partitioned cache architecture as a side-channel defence mechanism [J]. IACR Cryptology ePrint Archive, 2005: 280.

[44] Varadarajan V, Ristenpart T, Swift M. Scheduler-based defenses against cross-VM side-channels [C] // Proceedings of the 23rd USENIX Security Symposium, 2014: 687-702.

[45] Azar Y, Kamara S, Menache I, et al. Co-location-resistant clouds [C] // Proceedings of the 6th ACM Workshop on Cloud Computing Security, 2014: 9-20.

[46] Han Y, Chan J, Alpcan T, et al. Virtual machine allocation policies against co-resident attacks in cloud computing [C] // 2014 IEEE International Conference on Communications, 2014: 786-792.

[47] Fedorova A, Seltzer M, Smith MD. Improving performance isolation on chip multiprocessors via an operating system scheduler [C] // Peoceedings of the 16th International Conference on Parallel Architecture and Compilation Techniques, 2007: 25-38.

[48] Bhadauria M, McKee SA. An approach to resource-aware co-scheduling for CMPs [C] // Proceedings of the 24th ACM International Conference on Supercomputing. ACM, 2010: 189-199.

[49] Pattuk E, Kantarcioglu M, Lin ZQ, et al. Preventing cryptographic key leakage in cloud virtual machines [C] // Proceedings of the 23rd USENIX Security Symposium, 2014: 703-718.

[50] Godfrey M, Zulkernine M. Preventing cache-based side-channel attacks in a cloud environment [J]. IEEE Transactions on Cloud Computing, 2014, 2(4): 395-408.