云环境中跨虚拟机的Cache侧信道攻击技术研究

梁 鑫 桂小林 戴慧珺 张 晨
(西安交通大学电子与信息工程学院 西安 710049)
(陕西省计算机网络重点实验室(西安交通大学) 西安 710049)

摘 要 在云计算环境中,不同租户的虚拟机可能运行于同一台物理主机之上,即虚拟机同驻.同驻的虚拟机之间共享物理主机的计算资源,并依赖于虚拟机监控器进行系统资源的分配与调度.这种跨域共享虽然提高了资源使用效率,但也给用户的隐私安全造成严重威胁.恶意租户通过探测共享资源的状态信息,建立泄漏模型,便可绕过虚拟化提供的隔离性,窃取其它同驻虚拟机的隐私信息,这种攻击模式通常称为跨虚拟机的侧信道攻击.文中深入分析了跨虚拟机Cache侧信道攻击的机理和实现方式,对跨虚拟机Cache侧信道攻击技术的研究现状与进展进行总结.首先,分析总结了Cache侧信道信息泄露的本质原因;其次,回顾了跨虚拟机Cache侧信道攻击的起源与研究进展,讨论了其与传统Cache侧信道攻击的关系,并提出跨虚拟机访问驱动Cache侧信道攻击的通用模型;然后,分类归纳并重点阐述了虚拟机同驻相关问题以及当前用于跨虚拟机Cache侧信道信息探测的主流方法;最后,分析了目前研究中存在的问题,并展望了未来的研究方向.

关键词 云计算;缓存;侧信道攻击;虚拟机同驻;虚拟化
中图法分类号 TP309 **DOI号** 10.11897/SP.J.1016.2017.00317

Cross-VM Cache Side Channel Attacks in Cloud: A Survey

LIANG Xin GUI Xiao-Lin DAI Hui-Jun ZHANG Chen
(*School of Electronic and Information Engineering, Xi'an Jiaotong University, Xi'an* 710049)
(*Shaanxi Province Key Laboratory of Computer Network (Xi'an Jiaotong University), Xi'an* 710049)

Abstract In cloud-computing, virtual machines(VMs)of different tenants might be scheduled to run on the same physical machine, namely VMs co-residency. Co-resident VMs would share the underlying computing resources of the physical machine, relying on the virtual machine monitor to allocate and schedule system resources. Cross-domain sharing of underlying computing resources, albeit improving the utilization efficiency of available resources extremely, poses a serious threat to users' privacy concerns. A malicious VM could break the isolation mechanism and extract private information from other co-resident VMs, simply by probing the responses of shared resources and establishing a special leakage model. This attack pattern described above is usually called side-channel attacks. This paper deeply studied the mechanism and implementation of cross-VM cache side-channel attacks, and summarized its research status and advances. First, the essential cause of cache-based side-channel information leakage is analyzed and summarized. Next, the origin and research progress of cross-VM cache side-channel attacks are reviewed, the differences and relations between classic cache side-channel attacks and cross-VM cache side-channel attacks are discussed, followed by presentation of the universal model of access-driven cross-VM

收稿日期:2016-03-31;在线出版日期:2016-09-13.本课题得到国家自然科学基金(61472316)、陕西省科技计划项目(2013SZS16,2016ZDJC-05)和中央高校基本科研业务费项目(XKJC2014008)资助.**梁 鑫**,女,1987年生,博士研究生,主要研究方向为云计算、虚拟化安全、侧信道攻击.E-mail:xliang@stu.xjtu.edu.cn.**桂小林**(通信作者),男,1966年生,博士,教授,博士生导师,中国计算机学会(CCF)会员,主要研究领域为云安全、网络与信息安全、隐私保护.E-mail:xlgui@mail.xjtu.edu.cn.**戴慧珺**,女,1979年生,博士,工程师,主要研究方向为网络路由、隐私安全.**张 晨**,男,1992年生,硕士研究生,主要研究方向为云计算、分布式系统.

cache side-channel attacks. Then, the related issues of VMs co-residency and the latest mainstream methods for cross-VM cache-based side-channel information probing are categorized and expounded in detail. Finally, the current problems existing in the research and the future research directions of this field are presented.

Keywords cloud computing; cache; side-channel attacks; VMs co-residency; virtualization

1 引 言

云计算[1-2]作为一种新型的网络计算模式,通过网络以按需、易扩展的方式为用户提供各种虚拟的IT资源和应用服务,支持用户在任意位置、使用多种终端进行访问.云计算不仅大大减轻了用户的计算和存储负担,降低了用户对各种IT基础设施的购置和管理维护成本,也使得企业用户可以根据需求的变化随时进行应用的快速重新部署和动态扩展.然而,在已经实现的云计算服务中,安全和隐私问题[3-6]一直令人担忧,并已经成为阻碍云计算推广和普及的主要因素之一.基于虚拟化环境提供的逻辑隔离,采用访问控制[7]、入侵检测[8-9]等方法可以增强云计算环境的安全性;但是,隐私泄露问题依然存在,因为底层共享的硬件资源容易引发侧信道攻击(Side-Channel-Attacks,SCA)[10]的威胁.

侧信道攻击是一个经典的研究课题,由Kocher等人[11]于1996年首先提出.侧信道攻击是针对密码算法实现的一种攻击方式,当密码算法具体执行时,执行过程中可能泄露与内部运算紧密相关的多种物理状态信息,比如声光信息、功耗、电磁辐射以及运行时间等.这些通过非直接传输途径泄露出来的物理状态信息被研究人员称为侧信道信息(Side-Channel Information,SCI).攻击者通过测量采集密码算法执行期间产生的侧信道信息,再结合密码算法的具体实现,就可以进行密钥的分析与破解.而这种利用侧信道信息进行密码分析的攻击方法则被称为侧信道攻击.

相比于传统Cache侧信道攻击①,跨虚拟机Cache侧信道攻击的相关研究起步较晚,直到2009年由Ristenpart等人[10]首先提出.云计算环境中,云服务供应商为了有效利用物理资源,通常将多个不同租户的虚拟机分配到同一台物理机器上运行,称为虚拟机同驻.虚拟机同驻为跨虚拟机实施攻击提供了便利条件,而同驻虚拟机之间对于CPU、Cache及内存等底层物理资源的共享和争用则天然地为Cache侧信道信息泄露提供了泄露通道.因此,恶意租户可以方便地利用底层共享资源构建侧信道,绕过虚拟化环境提供的逻辑隔离,隐秘地窃取其它同驻虚拟机的隐私信息[12].现有的云安全防护机制很难发现和应对这种非入侵式的攻击,跨虚拟机Cache侧信道攻击已成为云计算环境中威胁用户隐私安全的重要挑战,并得到了国内外研究人员的普遍关注[10,13-14].

工业界同样关注跨虚拟机侧信道攻击相关研究,尤其是针对跨虚拟机侧信道攻击的有效防御方法.2010年,Intel公司发布了第1款支持AES-NI指令集的处理器②,用于防御针对AES的Cache侧信道攻击.AES-NI指令集在处理器芯片的定制硬件上执行AES加密/解密操作,由于不需要在内存中建立查找表,其指令在运行时都不会访问内存,因此,可有效防御针对AES的Cache侧信道攻击.随后,AMD公司发布的Bulldozer处理器中加入了对AES-NI的硬件支持③,Intel的Cryptography API以及密码库OpenSSL 1.0.1④和NSS 3.12.2⑤等也增加了对AES-NI指令集的支持.此外,公有云平台Amazon EC2提供的Virtual Private Cloud(VPC)⑥服务允许租户选择"专用硬件模式"创建虚拟机实例⑦,并承诺租户可以独占使用某个物理平

① 为了便于区分,本文将单机环境(非虚拟化环境)中的Cache侧信道攻击称为传统Cache侧信道攻击;将云计算环境中跨虚拟机的Cache侧信道攻击称为跨虚拟机Cache侧信道攻击;而发生在虚拟机内部的Cache侧信道攻击与传统Cache侧信道攻击类似,这里不做考虑.

② Securing the Enterprise with Intel AES-NI. http://www.intel.com/content/dam/doc/white-paper/enterprise-security-aes-ni-white-paper.pdf. 2010

③ Following instructions. https://web.archive.org/web/20101126155830/; http://blogs.amd.com:80/work/2010/11/22/following-instructions/. 2010

④ Intel Advanced Encryption Standard Instructions (AES-NI). https://software.intel.com/en-us/articles/intel-advanced-encryption-standard-instructions-aes-ni/. 2012

⑤ AES-NI enhancements to NSS on Sandy Bridge systems. https://bugzilla.mozilla.org/show_bug.cgi?id=706024. 2012

⑥ Virtual private cloud. Pro Powershell for Amazon Web Services: DevOps for the AWS Cloud. Springer, 2014: 67-88

⑦ AWS security whitepaper: Overview of security processes. https://zh.scribd.com/document/317587233/AWS-Security-Whitepaper. 2016

台上的硬件资源.虽然租户需要为此额外付费,但这极大地加强了租户之间的物理隔离性,基本杜绝了跨虚拟机 Cache 侧信道攻击的可能性. SecludIT 公司的云基础设施自动检测软件 Elastic Detector 则宣称可以检测发现云环境中访问驱动的侧信道攻击①.

目前,国内对跨虚拟机 Cache 侧信道攻击的关注和研究相对较少,且尚未有详细全面介绍其攻击机理与研究成果的综述论文.为深入理解跨虚拟机 Cache 侧信道攻击的机理和实现方式,总体把握其研究趋势,并促进国内在该方向上的研究,本文对云计算环境中跨虚拟机 Cache 侧信道攻击的相关研究工作进行了分析和总结.

本文第 2 节讨论共享 Cache 带来的信息泄露问题;第 3 节对跨虚拟机 Cache 侧信道攻击进行概述;第 4 节阐述虚拟机同驻的相关问题;第 5 节对主流的 Cache 侧信道信息探测方法进行介绍;第 6 节讨论现有研究中存在的问题并展望未来可能的研究方向;第 7 节对全文进行总结.

2　Cache 侧信道信息泄露分析

Cache 是攻击者构建侧信道时最常用的一类共享资源,因为:(1) CPU Cache 是最常用的共享资源,只要程序运行,就离不开 CPU,也就离不开 CPU Cache;(2) Cache 在系统中位于较低的层级,且缺乏有效的访问控制机制,利用其构建侧信道可以绕过许多高层的隔离机制,比如虚拟化环境提供的逻辑隔离等;(3) Cache 拥有一个系统上数据访问和计算操作最细粒度且详细的状态信息,可以为攻击者提供丰富的信息;(4) 不同的 Cache 行为(命中或失效)具有不同的时间特征,可以通过采集 Cache 时间特征得到 Cache 侧信道信息,而 Cache 时间特征的采集方法比较简单;(5) Cache 的操作和刷新频率较高,可以提供细粒度的观察结果.

分析探讨共享 Cache 带来的信息泄露问题,对于深入理解 Cache 侧信道攻击的机理与实现方式来说十分重要.因此,本节将对 Cache 侧信道信息泄露问题进行详细地分析讨论,其中,2.1 节分析 Cache 可能泄露哪些信息;2.2 节讨论 Cache 泄露的侧信道信息可以用于密钥分析的原因;2.3 节讨论 Cache 侧信道信息为什么会被泄露;2.4 节讨论 Cache 侧信道信息可能通过哪些方式泄露.

2.1　Cache 侧信道信息

Cache 侧信道信息是指系统中由 Cache 行为产生的物理状态信息,包括访问时间、功率、电磁辐射等.通常,CPU 访问内存中的数据或指令时,根据涉及到的目标数据或指令当前是否位于 Cache 中,将导致不同的 Cache 行为,包括 Cache 命中和 Cache 失效.而不同 Cache 行为产生的 Cache 侧信道信息是可区分的,例如,相比 Cache 命中,Cache 失效会产生对更高一级 Cache 或内存的访问,从而导致更长的访问时间或者产生更多的功耗等.因此,如果能够观察或测量到算法执行期间产生的 Cache 侧信道信息,攻击者就可以推断对应的 Cache 行为,并进一步得到内存访问的一些关键信息,包括:

(1) 内存访问的部分地址信息

Cache 是物理标记的,数据在 Cache 中的位置由其物理内存地址决定.因此,Cache 可能泄露内存访问的部分地址信息.例如,目标程序执行期间,攻击者根据访问指定的 Cache 组产生的侧信道信息,可以推断目标程序是否使用了该 Cache 组;如果使用了,再根据 Cache 与内存地址之间的映射关系,就可以得到目标程序执行期间内存访问的部分地址信息.

(2) 历史内存访问的数据信息或指令信息

CPU 访问指定的数据或指令时,根据其当前是否被缓存于 Cache 中,将产生不同的侧信道信息,具体地,当访问一个数据块或指令块时,如果其已经被缓存于 Cache 中,将发生 Cache 命中;那么,相对 Cache 失效来说,将产生较短的访问时间.因此,一个数据块或指令块的访问时间将泄露其是否已被缓存于 Cache 中这一信息;进一步地,这将泄露历史内存访问的数据信息或指令信息.例如,目标程序执行期间,攻击者访问指定的数据块或指令块,根据访问时间就可以判断它们是否已被缓存,进一步地可以推断目标程序执行期间是否访问过这些数据或指令.

综上,利用 Cache 侧信道信息可以推测得到一些程序执行期间的内存访问信息,包括访问了哪些内存位置的数据,使用了哪些内存页面,或者执行了哪些指令序列等.

2.2　Cache 侧信道信息与密钥之间的相关性

算法实现中通常存在一些条件语句、分支语句等,导致算法执行具有数据依赖特性,即相同算法在输入参数不同时,可能具有不同的执行特征.因此,攻击者可以利用算法的执行特征推测输入参数的取值.

① How to detect side-channel attacks in cloud infrastructures. https://elastic-security.com/2013/09/11/how-to-detect-side-channel-attacks-in-cloud-infrastructures/. 2013

如图 1 所示. 其中,图 1(a)为输入依赖的指令执行,如果 A 和 B 的运行时间存在可观察的差异,攻击者就可以根据采集到的时间信息判断执行了哪一分支,进而确定 x 的取值;图 1(b)所示为输入依赖的数据访问,如果攻击者能够确认用户访问了 m 还是 n,就可以确定 y 的取值;图 1(c)所示为输入依赖的执行时间,由于循环次数由输入参数决定,而循环次数又直接影响算法的执行时间,因此,算法执行时间将泄露输入参数 z 的取值信息.

```
if (x==1)                if (y==1)                int z=input();
  then A;                  then m++;              (for i=0; i<z; i++)
else if (x==0)           else if (y==0)           { sleep(10); }
  B;                       n++;
```

(a) 输入依赖的指令执行　(b) 输入依赖的数据访问　(c) 输入依赖的执行时间

图 1　算法执行具有数据依赖特性

密码算法的执行通常具有密钥依赖特性,主要包括密钥依赖的数据访问模式和密钥依赖的指令执行序列两种. 因此,攻击者可以利用密码算法执行中产生的侧信道信息推断其执行特征,再根据执行特征以及具体的依赖关系来反向推测密码算法使用的密钥. 例如:

现代分组密码算法在实现中通常使用查找表来提高性能,且查表索引通常是密钥相关的,因此,分组密码算法的执行通常具有密钥依赖的数据访问模式. 如果攻击者能够获取密码算法查表访问的 Cache 组地址集合,并将其转换为查表索引,再结合明文或密文就可以进行密钥分析[15],推断出密码算法使用的部分或全部密钥.

现代公钥密码算法在加密/解密过程中所要执行的指令序列,由于密钥各二进制位取值不同一般具有很大区别,因此,公钥密码算法的执行通常具有密钥依赖的指令执行序列. 比如,模幂运算是 RSA 加解密的核心算法,其中一种简单的实现方式为平方-乘算法,即将模幂运算分解为一系列平方、乘法以及取模运算. 如图 2 所示,采用平方-乘算法,当指数 e 的二进制表示中的某一位为 1 时将比为 0 时多执行两步运算[13]. 那么,如果攻击者能够确定算法运行期间执行的指令序列,就可以推测指数 e 的取值,并进一步破解密钥.

```
算法:平方-乘模幂算法
输入:底数x,模m,
    指数e=(e_{n-1}…e_0)_2
输出:y=x^e mod m
FOR i=n-1 DOWNTO 0 DO
    y←y^2
    y←y mod m
    IF e_i=1 THEN
        y←y*x                  ←仅当
        y←y mod m              e_i=1时
    END IF                     执行
END FOR
RETURN y
```

图 2　平方-乘算法

2.3　Cache 侧信道信息泄露来源

在云计算环境中,被分配到同一个物理 CPU 上运行的多个虚拟机之间共享硬件 Cache,虽然 Cache 中的数据是受存储器保护的,攻击者无法直接获取. 但是,这些虚拟机在 Cache 中的数据可能被映射到同一个或多个 Cache 组甚至 Cache 行中,这为攻击者提供了一个观察被攻击虚拟机 Cache 行为的侧信道:攻击者访问其私有数据或执行私有指令时产生的侧信道信息,可以用来推测其它虚拟机的 Cache 行为.

目前,由于共享 Cache 产生的信息泄露可以分为两类:Cache 争用产生的信息泄露和数据重用产生的信息泄露.

2.3.1　Cache 争用产生的信息泄露

通常,Cache 失效表明发生了 Cache 争用. 如图 3 所示,虚拟机 VM1 的内存数据 $A1$,$A2$ 与虚拟机 VM2 的内存数据 $B1$,$B2$,$B3$ 被映射到同一个 Cache 组中. 假设 VM2 在较短的时间内连续两次访问 $B1$、$B2$ 和 $B3$,如果第 2 次访问 $B1$ 时发生了 Cache 失效,则可以判断在 VM2 的两次访问之间有其它的虚拟机(VM1)使用了相同的 Cache 组,导致 VM2 的 B1 已被驱逐出 Cache.

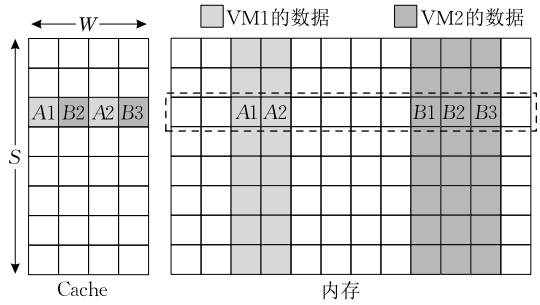


图 3　Cache 争用产生的信息泄露

在图 3 中,每个小方块表示 Cache 行大小的数据块,是 Cache 与内存之间进行数据传输与分配的基本单位;图中左边表示 Cache,每行表示一个 Cache 组,每个 Cache 组有 W 个 Cache 行,总共有 S 个 Cache 组;右边表示内存,每行表示映射到同一 Cache 组中的所有内存块,即对应于左边同一高度的 Cache 组.

Prime-Probe[10,13-14] 攻击方法是 Cache 争用产生信息泄露的一个典型应用. Prime-Probe 攻击中,

攻击者利用进程或虚拟机之间的外部 Cache 访问冲突，在密码进程运行前后分别访问同样的数据来填充 Cache，并根据第 2 次访问各个 Cache 组时发生了 Cache 命中（没有争用）还是 Cache 失效（存在争用），来推测密码进程运行期间访问的 Cache 组地址集合，再在此基础之上进行密钥分析.

2.3.2　数据重用产生的信息泄露

Cache 命中信息能够记录 CPU 对 Cache 的历史访问情况. Cache 命中表明存在相同数据的重复使用，即 CPU 曾经访问过相同的数据. 如图 4 所示，假设虚拟机 VM1 和 VM2 共享部分内存页面，VM2 访问页面 $A1$，$A2$ 时，发现 $A1$，$A2$ 已经位于 Cache 中，则可以推断有其它的虚拟机（VM1）在之前使用了 $A1$，$A2$.

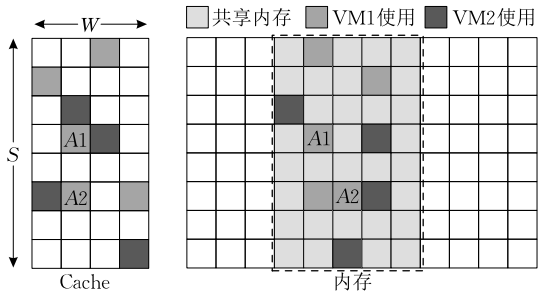


图 4　数据重用产生的信息泄露

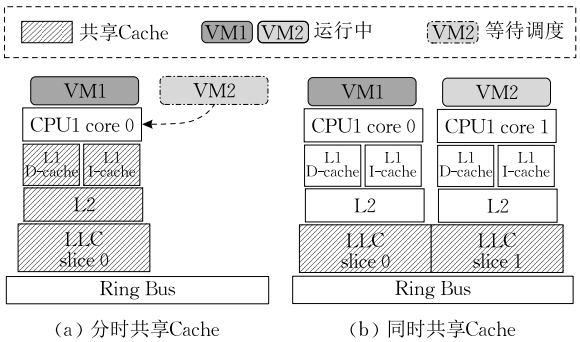
Yarom 等人[16]提出的 Flush-Reload 攻击方法是数据重用产生信息泄露的一个典型例子，其假设攻击者和攻击目标共享内存页面，在密码进程执行前将指定地址的一个或多个共享内存块驱逐出 Cache，并在加密完成后立即访问这些内存块，然后根据访问时是否发生了 Cache 命中来推测密码进程执行期间是否使用了其中的一些内存块，进一步地，可以推测密码进程执行了哪些加密指令等.

2.4　Cache 侧信道信息泄露方式

共享 Cache 是 Cache 侧信道信息泄露的主要方式. 云计算环境中，攻击虚拟机和目标虚拟机共享 Cache 的方式可以分为图 5 所示.

（1）分时共享

当攻击虚拟机和目标虚拟机被分配到同一个 CPU 核心上运行时，如图 5（a）所示，它们之间分时复用该物理 CPU 核心，并由虚拟机监控器（Virtual Machine Monitor，VMM）负责调度、切换运行的虚拟机. 这种情况下，攻击虚拟机和目标虚拟机可以共享该 CPU 的全部多级 Cache，但是只能分时共享，并不能同时对 Cache 进行访问.



（a）分时共享Cache　　（b）同时共享Cache

图 5　虚拟机间共享 Cache 的方式

VMM 每次切换运行的虚拟机时，并不会清空该 CPU Cache 中的数据，因此，下一个运行的虚拟机可以探测得到上一个运行的虚拟机的 Cache 访问信息. 通常，攻击者需要寻找并利用虚拟机调度算法的漏洞，来抢占目标虚拟机的调度[13]，才能保证攻击虚拟机能够紧随目标虚拟机之后被调度运行；这样，才能在虚拟机切换后探测目标虚拟机之前的 Cache 访问信息.

（2）同时共享

当攻击虚拟机和目标虚拟机被分配到同一个 CPU 的不同核心上运行时，它们之间可以并行运行. 这种情况下，如图 5（b）所示，攻击虚拟机和目标虚拟机只能共享该 CPU 的末级 Cache（Last Level Cache，LLC），并不能共享各个核心私有的 L1 Cache 或 L2 Cache. 但是，由于它们之间可以同时对 LLC 进行访问，因此，攻击者可以同步地在目标虚拟机运行期间探测 LLC 来得到其 Cache 访问信息[14,17].

（3）通过 Cache 一致性协议共享 Cache 数据

在多处理器系统中，如果多个 CPU 访问了同样的一块内存数据，那么，这块内存数据可能在不同 CPU 的多个 Cache 中都存有副本. Cache 一致性协议就是用来解决内存数据与它的多个 Cache 副本之间的内容一致性问题. 即使攻击虚拟机和目标虚拟机被分配到不同的 CPU 之上运行时，如图 6 所示，虽然它们之间并不物理上共享 Cache；但是，由于

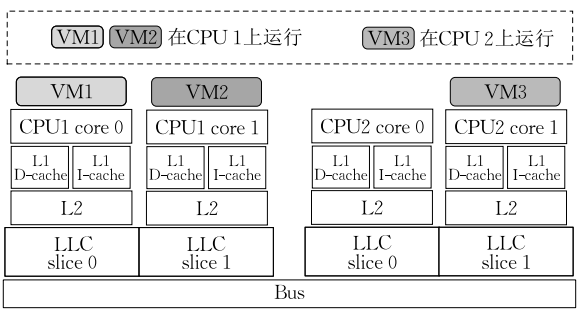


图 6　通过 Cache 一致性协议共享 Cache 数据

Cache 一致性协议,它们也可能会共享一部分 Cache 数据(同一内存数据在不同 Cache 中的副本),因此,Cache 一致性协议也可以看作逻辑上共享 Cache 的一种方式.

当一个处理器发生 Cache 失效时,失效的数据是否缓存于另一个处理器 Cache 中,将产生从另一个处理器 Cache 读取数据和直接从内存读取数据两种可能,这将导致可区分的 Cache 访问时间差异[18].因此,攻击者也可以利用 Cache 一致性协议来探测目标虚拟机是否使用了指定的内存数据[18].

本文没有考虑被分配到同一个 CPU 核心上的虚拟机之间并行运行的情况,这主要是由于以下两方面的原因:一是出于安全性的考虑,公有云平台一般禁止使用超线程或对称多线程技术[13],因此,基本不会发生这种情况;二是即使发生这种情况,也可以同时参考上述两种共享方式产生的泄露问题.

3 跨虚拟机 Cache 侧信道攻击概述

在云计算环境中,被分配到同一个物理 CPU 上运行的多个虚拟机之间共享底层的硬件资源,这为恶意租户跨虚拟机实施 Cache 侧信道攻击提供了极大的便利条件.跨虚拟机 Cache 侧信道攻击会破坏多租户虚拟机之间的隔离性,从而给其它同驻虚拟机带来严重的安全威胁.同时,由于跨虚拟机 Cache 侧信道攻击不是利用系统本身的漏洞进行攻击,而是通过共享的底层硬件环境进行攻击,因此极难防御.

跨虚拟机 Cache 侧信道攻击可以看作是传统 Cache 侧信道攻击在云计算环境下跨虚拟机攻击场景中的一种扩展和应用.因此,3.1 节介绍传统 Cache 侧信道攻击,3.2 节分析总结跨虚拟机 Cache 侧信道攻击与传统 Cache 侧信道攻击的联系与区别,3.3 节介绍跨虚拟机 Cache 侧信道攻击的研究进展.

3.1 传统缓存侧信道攻击

1998 年,Kesley 等人[19]提出 Cache 命中率可用于密码分析的思想.此后,Cache 侧信道攻击技术获得研究人员的极大关注并得以迅速发展.最初,Cache 侧信道攻击的相关研究工作都是围绕单机环境展开,以数据缓存[20-21]、指令缓存[22-23]等作为研究对象,研究人员提出了多种可行的 Cache 侧信道攻击方法,并且可以利用这些攻击方法窃取受害者系统中的机密信息,例如 AES[20-21,24-25]、DES[26]、RSA[27-30]等加密算法的密钥,这给大部分加密算法的安全性带来了严重威胁.

根据采集信息不同,可将传统 Cache 侧信道攻击分为时序驱动攻击、访问驱动攻击以及踪迹驱动攻击[29].时序驱动攻击[24,26,31]需要采集密码算法一次加密/解密的整体时间,并利用统计分析方法推测密钥.访问驱动攻击[20,25,27,32]需要通过间谍进程采集密码算法一次加密/解密过程中访问的 Cache 组地址集合,再利用直接分析或排除分析方法推测密钥.踪迹驱动攻击[28-30]需要采集密码算法一次加密/解密过程中所有 Cache 访问的命中和失效序列,再结合明文或密文推测密钥.

传统 Cache 侧信道攻击方法大多存在先决成立的假设条件,例如时序驱动攻击需要能够获得目标系统的详细配置参数并重建相同的本地对照环境;访问驱动攻击需要能够侵入目标机器并植入间谍进程;踪迹驱动攻击通过计时手段很难实现,需要物理上接触目标机器并精确采集其功率消耗或电磁辐射信息等.这些假设条件在现实环境的攻击场景中往往难以满足,这严重影响了传统 Cache 侧信道攻击在现实环境中的可行性.

3.2 跨虚拟机 Cache 侧信道攻击的起源

2009 年,Ristenpart 等人[10]首次指出云计算环境中存在跨虚拟机 Cache 侧信道攻击的安全威胁.不同于传统 Cache 侧信道攻击,同驻虚拟机之间对于 Cache、内存的共享和争用天然地为跨虚拟机 Cache 侧信道攻击的实现提供了便利条件和重要基础,极大地增强了跨虚拟机 Cache 侧信道攻击在现实环境中的可行性.参考传统 Cache 侧信道攻击分类,可将跨虚拟机 Cache 侧信道攻击分为时序驱动攻击和访问驱动攻击,下面分别对其与传统 Cache 侧信道攻击的关系进行评述.而踪迹驱动攻击在云计算环境中很难实现,目前尚未有相关的研究成果.

3.2.1 跨虚拟机时序驱动攻击

传统时序驱动攻击可以分为远程攻击和本地攻击两种.远程攻击属于非入侵式攻击,攻击者通过网络远程采集计时信息,由于网络传输时延和抖动时延的存在,很难采集到精确的计时信息,而不精确的计时信息对攻击成功率影响较大,因此,远程攻击在真实网络环境下的适用性不强.本地攻击属于入侵式攻击,攻击者需要在攻击目标的操作系统中植入间谍进程来采集计时信息,虽然可以完全消除网络传输时延和抖动时延带来的影响,但是也降低了本

地攻击在现实环境中的可行性.

由于发生在两个虚拟机操作系统之间，跨虚拟机时序驱动攻击本质上仍属于远程攻击. 当攻击者的虚拟机和攻击目标的虚拟机不同驻时，攻击场景与传统远程时序驱动攻击完全一样，因此，跨虚拟机时序驱动攻击通常发生在同驻的虚拟机之间. 由于同驻虚拟机之间一般通过宿主机的本地通信信道进行数据交换，因此，跨虚拟机时序驱动攻击采集的计时信息比较精确[33]. 除此之外，跨虚拟机时序驱动攻击与传统时序驱动攻击并无明显区别.

3.2.2　跨虚拟机访问驱动攻击

与传统访问驱动攻击方法类似，跨虚拟机访问驱动攻击通过监测宿主机的 Cache 或共享内存，并利用计时手段采集目标进程运行期间通过 Cache 泄露出来的侧信道信息，再结合其具体实现进行分析，即可能推测得到其它同驻虚拟机的隐私信息. 但是，如表 1 所示，跨虚拟机访问驱动攻击与传统访问驱动攻击也存在许多不同之处，表现在：

表 1　传统跨进程与跨虚拟机访问驱动攻击比较

攻击类型	攻击场景	运行环境	攻击方式	可检测性
传统访问驱动攻击	发生在单机环境中的两个进程之间	真实	入侵式	攻击目标可以检测到攻击
跨虚拟机访问驱动攻击	发生在同驻一台宿主机的两个虚拟机之间	虚拟	非入侵式	攻击目标无法检测到攻击

(1) 攻击场景

传统访问驱动攻击发生在单机环境中的两个进程之间. 攻击者需要将间谍进程植入攻击目标的操作系统，再利用系统内多个进程间共享 Cache 的特点使间谍进程与密码进程同步执行，探测采集密码进程执行期间的 Cache 侧信道信息.

跨虚拟机访问驱动攻击发生在同驻一台宿主机的两个虚拟机之间. 攻击者不需要侵入目标虚拟机的操作系统，仅需要和目标虚拟机同驻，就可以探测采集密码进程执行期间的 Cache 侧信道信息.

(2) 运行环境

传统访问驱动攻击中，密码进程与间谍进程运行在真实物理硬件之上的同一个真实操作系统环境中，它们之间具有完全相同的运行环境，且可以共享该机的全部硬件资源.

跨虚拟机访问驱动攻击中，攻击者和攻击目标分别为同驻一台宿主机的攻击虚拟机和目标虚拟机中的目标进程. 攻击虚拟机和目标虚拟机共享覆盖的或相邻的底层硬件资源，可能具有相同的或完全不同的操作系统. 系统级虚拟化为跨虚拟机攻击增加了难度，例如，内存虚拟化使得虚拟机内存系统中增加了一级物理地址到机器地址的地址转换，导致内存地址与 Cache 之间的映射关系更为复杂；同时，系统级虚拟化也使得攻击者拥有其虚拟机操作系统的全部权限，这可以为其提供一些新的能力，例如可以修改操作系统的内存页面大小等.

(3) 攻击方式

传统访问驱动攻击需要侵入攻击目标的操作系统，植入并运行间谍进程，是一种入侵式的攻击方式.

跨虚拟机访问驱动攻击的攻击目标从操作系统内部转移到了操作系统外部，不需要侵入攻击目标的操作系统，是一种非入侵式的攻击方式.

(4) 可检测性

传统访问驱动攻击会在目标操作系统中留下攻击痕迹，因此，存在被攻击目标检测发现的可能性.

跨虚拟机访问驱动攻击不会在攻击目标的虚拟机中留下任何的攻击痕迹，因此，攻击目标无法检测到这种攻击行为.

3.3　跨虚拟机 Cache 侧信道攻击的研究进展

近年来，跨虚拟机 Cache 侧信道攻击研究得到了国内外研究人员的极大关注，并获得了一些新的研究进展，图 7 为其研究时间轴，其中标示出了跨虚拟机 Cache 侧信道攻击研究起源与发展过程中具有代表性的文献. 下面从时序驱动攻击和访问驱动攻击两个方面分别进行介绍.

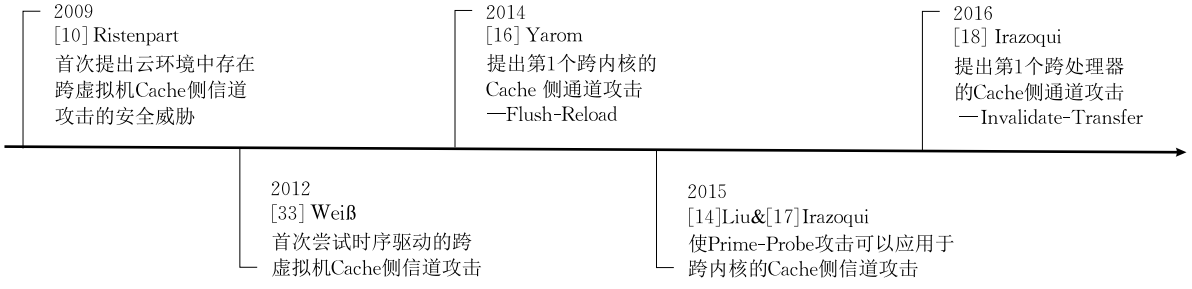


图 7　跨虚拟机 Cache 侧信道攻击研究时间轴

3.3.1 跨虚拟机时序驱动攻击研究进展

目前,跨虚拟机时序驱动攻击的相关研究较少,已有研究成果也只是将传统时序驱动攻击方法直接应用于跨虚拟机攻击场景中.

2012 年,Weiß 等人[33]以运行 L4Re 微内核的虚拟化平台为例,首先尝试将 Bernstein 攻击[31]应用于虚拟化环境中同驻的虚拟机之间. 2014 年,Irazoqui 等人[34-35]将 Bernstein 攻击应用于 Xen、VMware 和 KVM 虚拟化环境以及 Amazon EC2 云平台中,并成功恢复了其它同驻虚拟机中使用的 AES 密钥的部分比特位.

3.3.2 跨虚拟机访问驱动攻击研究进展

传统访问驱动攻击为 Cache 的探测和分析奠定了坚实的基础,也为跨虚拟机访问驱动攻击提供了可以借鉴的技术和方法.已有跨虚拟机访问驱动攻击可分为:(1)将传统的访问驱动攻击方法迁移至跨虚拟机攻击场景中,并解决迁移过程中产生的各种问题和挑战;(2)设计实现新的攻击方法.

(1)迁移传统攻击方法

2009 年,Ristenpart 等人[10]利用 Prime-Probe 方法[20]在 Amazon EC2 云平台中探测得到同驻虚拟机的 Cache 负载状态信息以及用户击键间隔时间信息等. 2012 年,Zhang 等人[36]基于 Prime-Probe 方法设计了 HomeAlone 工具,租户可利用 HomeAlone 来检测是否有其他租户的虚拟机与其同驻.随后,Zhang 等人[13]利用 Prime-Probe 方法对 L1 指令 Cache 进行探测,并设计了一个隐马尔科夫模型过滤探测噪声,成功恢复了其它同驻虚拟机上使用的 ElGamal 加密密钥. 2015 年,Younis 等人[37]将虚拟机的虚拟地址转换为物理地址,再利用 Prime-Probe 方法监控这些物理地址是否被其它虚拟机访问过. Liu 等人[14]和 Irazoqui 等人[17]利用大页映射得到的额外物理地址知识来保留 LLC 和物理内存之间的映射关系,绕过了 LLC 与内存地址映射不透明的问题,使得 Prime-Probe 方法可用于探测 LLC.在此之前,Prime-Probe 方法只能用于探测容量较小的 L1 Cache.随后,Inci 等人[38]和 Kayaalp 等人[39]对 LLC 的索引哈希机制进行逆向工程,得到了内存地址与 LLC 各分片之间的映射关系,解决了 LLC 与内存地址映射不透明的问题.

(2)设计新的攻击方法

2014 年,Yarom 等人[16]假设攻击虚拟机和目标虚拟机共享内存页面,并利用 Cache 的包容性特点,将 Gullasch 攻击[21]扩展之后用于探测 LLC,提出了第 1 个可以跨内核进行攻击的 Cache 侧信道攻击方法(Flush-Reload 攻击).

2015 年,Gruss 等人[40]发现缓存刷新指令的执行时间取决于被刷新的内存块是否位于 Cache 中,他们利用这一点对 Flush-Reload 攻击进行了改进,提出了 Flush-Flush 攻击,在缩短了攻击执行时间的同时提高了攻击的隐匿性.

2016 年,Irazoqui 等人[18]基于 Cache 一致性协议,提出并实现了第 1 个跨 CPU 的 Cache 侧信道攻击方法(Invalidate-Transfer 攻击),成功恢复了另一个 CPU 上运行的加密软件库中使用的 AES 密钥和 ElGamal 密钥.

通过上述分析发现,跨虚拟机访问驱动攻击的攻击方法更为多样化,随着攻击媒介从 L1 数据 Cache 和 L1 指令 Cache,到 LLC,再到双路 CPU 上 Cache 的变化,攻击的限制条件越来越现实:从要求攻击虚拟机和目标虚拟机同驻一个 CPU 核心,到同驻一个 CPU,再到跨 CPU,威胁范围也越来越广.

3.4 跨虚拟机访问驱动 Cache 侧信道攻击的通用模型

根据 3.2.1 节的分析,跨虚拟机时序驱动攻击的攻击模式和传统时序驱动攻击并无区别,同时,发表的文献中,已有一些[15,32]对传统时序驱动攻击的攻击模型进行了详细介绍及总结,可供参考,本文不再赘述.而 3.2.2 节的分析表明,跨虚拟机访问驱动攻击的攻击场景、执行环境以及攻击方式等均与传统访问驱动攻击不同.因此,本节探讨跨虚拟机访问驱动 Cache 侧信道攻击的攻击模式,给出此类攻击的一个通用模型.

3.4.1 相关假设

为了方便描述,我们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.云计算环境中的 Cache 侧信道攻击一般基于如下假设:

假设 1. 云服务供应商及其提供的底层基础设施都是可信的,而使用云平台的各个租户之间是互不信任的.

假设 2. 攻击者为使用云平台的恶意租户,其拥有目标用户所运行应用的相关背景知识,但不具有任何特殊权限,期望利用侧信道攻击来获取目标用户的隐私信息.

假设 3. 攻击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相关的应用,并对外提供公开的服务访问接口.

上述假设均符合实际应用场景，即使假设 2 中针对攻击者背景知识的假设，也具有一定的普适意义. 例如，针对特定的目标用户发起攻击时，攻击者通常已经拥有目标用户的背景知识；即使没有，攻击者也可以通过其它方式得到所需背景知识，例如，采用文献[41]和文献[42]中的方法，攻击者可以判断同驻虚拟机是否与其使用相同的操作系统，或者探测得到同驻虚拟机中运行的应用等.

3.4.2　攻击过程

跨虚拟机 Cache 侧信道攻击本质上是一种基于统计分析的攻击方法. 攻击过程一般如下：

(1) 实现同驻

基于一定策略，攻击者创建并运行多个虚拟机实例，并利用虚拟机同驻检测方法逐一判断这些虚拟机是否和目标虚拟机同驻；重复这一过程，直到实现至少一个攻击虚拟机和目标虚拟机同驻为止.

(2) 探测 Cache 侧信道信息

攻击虚拟机首先根据具体攻击场景(包括攻击虚拟机和目标虚拟机共享 Cache 的方式，目标虚拟机所运行应用的具体实现特征等)的不同，设计选取合适的 Cache 侧信道信息探测方法；然后，将 Cache 设置为一个已知状态，并访问部署在目标虚拟机中的应用；最后，探测并收集目标虚拟机运行过程中产生的 Cache 侧信道信息.

(3) 分析 Cache 侧信道信息

将攻击者想要获取的隐私信息标记为 $K=\{K_1,K_2,\cdots,K_n\}$. 目标虚拟机中的应用为了响应攻击者的服务请求，将执行一些操作 $OP=\{OP_1,OP_2,\cdots,OP_n\}$，而操作 OP 与隐私信息 K 之间通常具有依赖关系，即由隐私信息 K 决定执行哪些操作，可以表示为函数 $f:K\rightarrow OP$，即 $OP=f(K)$；在执行 OP 时，目标虚拟机访问共享 Cache，这将产生一些 Cache 侧信道信息 $L=\{L_1,L_2,\cdots,L_n\}$；当 Cache 初始状态相同时，不同的操作 OP 将产生不同的 Cache 行为，从而产生不同的 Cache 侧信道信息，即操作 OP 与执行操作 OP 产生的 Cache 侧信道信息 L 之间具有强相关性，可以表示为函数 $g:OP\rightarrow L$，即 $L=g(OP)$.

通过上述分析，攻击虚拟机可以通过分析探测到的 Cache 侧信道信息 L 来推测得到目标虚拟机中的应用具体执行了哪些操作，再结合这些操作(OP)与隐私信息(K)之间的关系，减小 K 的取值范围或直接推测 K 的取值. 具体地，通过对目标应用的具体实现进行分析，可以建立函数 f，并得到其逆函数 $K=f^{-1}(OP)$；对具体 Cache 行为与其产生的 Cache 侧信道信息进行关联，可以建立函数 g，并得到其逆函数 $OP=g^{-1}(L)$；在此基础之上，建立泄露模型 $K=f^{-1}(g^{-1}(L))$，即可通过对 Cache 侧信道信息 L 进行分析，推测得到目标虚拟机的隐私信息 K.

图 8 给出了跨虚拟机 Cache 侧信道攻击的通用模型. 其中，虚拟机同驻是实现跨虚拟机 Cache 侧信道攻击的重要基础，我们将在第 4 节中介绍；跨虚拟机 Cache 侧信道信息探测方法将在第 5 节中介绍；跨虚拟机 Cache 侧信道信息分析方法与传统单机环境中的分析方法是通用的，且一般针对具体算法及其实现进行设计，在文献[15,29]中已经详细介绍，本文不再赘述.

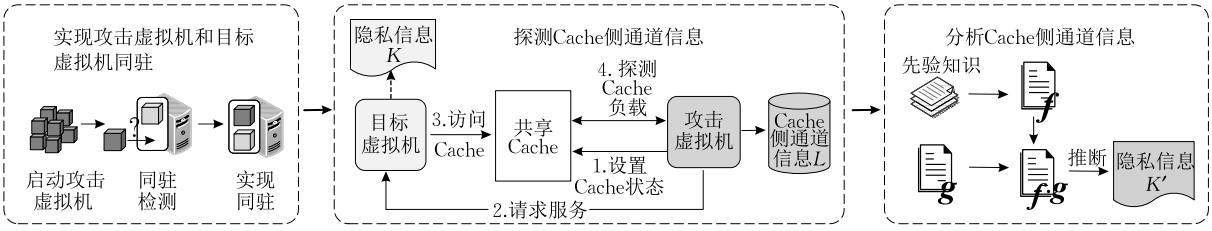


图 8　跨虚拟机 Cache 侧信道攻击通用模型

4　虚拟机同驻

攻击虚拟机和目标虚拟机同驻是攻击者可以跨虚拟机实施侧信道攻击的必要条件和重要基础. 然而，随着云计算的快速发展，数据中心拥有的物理机器数量不断增加，例如 Amazon EC2 的单个数据中心目前一般拥有超过 50 000[①] 台物理机器. 在这一应用背景下，攻击者可能需要花费大量的时间和精力，才能实现和特定目标虚拟机的同驻.

根据 3.4.2 节有关实现同驻的过程，攻击者创建虚拟机的策略以及同驻检测方法的准确性和效率是影响同驻实现过程的重要因素. 因此，4.1 节分析

① AWS innovation at scale, AWS re: Invent 2014(SPOT301). http://www.slideshare.net/AmazonWebServices/spot301-aws-innovation-at-scale-aws-reinvent-2014. 2014

公有云中虚拟机资源调度策略的脆弱性,这可以帮助攻击者可以更有效率地实现同驻;4.2 节分类介绍虚拟机同驻检测方法;4.3 节探讨虚拟机同驻检测评估问题,并对已有的各种虚拟机同驻检测方法进行对比分析.

4.1 虚拟机资源调度策略

云平台的虚拟机资源调度策略主要解决的是虚拟机与物理主机的映射问题,包括虚拟机的创建、迁移以及管理等.作为云计算底层的核心技术之一,虚拟机资源调度策略的好坏将直接影响到云平台的整体性能、运营成本以及服务质量等.目前,公有云平台的虚拟机资源调度策略还未形成统一的标准和规范,各大云服务供应商都是根据自身资源的特点以及调度目标等采用不同的调度策略[43],例如,IBM 云平台采用性能优先的调度策略,HP 云平台采用成本优先的调度策略,Amazon EC2 的调度策略综合考虑了成本优先、负载均衡、高可靠性、满足用户不同租用请求等优化目标.

通过上述分析可以发现,公有云平台目前采用的虚拟机资源调度策略多是从降低能耗、减少成本、提高资源利用率以及实现负载均衡等角度出发进行优化,而大多忽略了安全上的考虑.因此,攻击者可以分析并利用虚拟机资源调度策略的脆弱性,更有针对性地创建或启动虚拟机来实现与目标虚拟机的同驻[10].例如,为了降低能耗,并提高资源利用率,虚拟机资源调度策略可能将同一时刻提交启动申请的多个虚拟机分配到一台物理主机上运行;再如,为了便于管理,虚拟机资源调度策略可能将业务分类、可用区域、实例类型等启动信息相同的虚拟机分配到一台物理主机上运行.

为了应对同驻威胁,公有云平台对虚拟机资源调度策略和网络管理功能进行了优化,在一定程度上增加了实现同驻的难度[44],但是,攻击者仍可以较低的开销实现与目标虚拟机的同驻[44-45].

4.2 虚拟机同驻检测方法

虚拟机同驻检测方法用来判断两个虚拟机是否运行于同一台物理主机之上,高效、准确的同驻检测方法有助于降低攻击者实现同驻的开销.根据同驻检测的基本原理不同,可将虚拟机同驻检测方法分为基于网络信息的同驻检测、基于资源干扰的同驻检测和基于隐蔽信道的同驻检测.

4.2.1 基于网络信息的虚拟机同驻检测

Ristenpart 等人[10]发现在 Amazon EC2 中同驻的两个 Xen 虚拟机之间具有相同的第 1 跳 IP 地址,较短的网络包往返时间,以及数字上接近的内部 IP 地址;因此,攻击者可以利用两个虚拟机的网络信息来判断它们是否同驻.然而,2015 年,Inci 等人[38]发现,Amazon EC2 已经修复了这些网络信息泄露漏洞,上述利用网络信息进行虚拟机同驻检测的方法在 Amazon EC2 中都已不再适用.但是,基于网络信息的同驻检测方法仍可能适用于其它的云平台.

基于网络信息的同驻检测,利用两个虚拟机的网络信息来判断它们之间是否同驻.优点是实现简单,检测效率较高,且不会对目标虚拟机的运行造成任何影响;缺点是检测准确率不高,且云平台可以很容易地阻止此类检测方法[44],例如利用 VPC 对不同虚拟机的网络进行隔离等.

4.2.2 基于资源干扰的虚拟机同驻检测

虽然 VMM 在同驻的虚拟机之间提供了逻辑隔离,但是对于底层共享资源的竞争使用仍然会导致虚拟机之间的相互干扰[10,36,46].

物理网卡的多路复用会带来网络包延时问题[47],基于此,Bates 等人[48-49]设计了一种同驻水印方法进行虚拟机同驻检测.具体地,进行同驻检测的虚拟机(攻击虚拟机)周期性地占用宿主机的物理网卡,向外发送无意义的网络包;与此同时,通过代理与目标虚拟机进行通信,并测量收集它们之间的网络包通信状况.如果攻击虚拟机与目标虚拟机同驻,每个检测周期内它们将多路复用宿主机的物理网卡,就会导致代理与目标虚拟机之间的网络通信存在一定的延时,这个延时就是同驻水印.因此,通过检测周期内代理与目标虚拟机之间的网络通信是否产生同驻水印,即可判断攻击虚拟机与目标虚拟机是否同驻.

Zhang 等人[36]设计了 HomeAlone 工具,租户可利用 HomeAlone 来检测是否有其他租户的虚拟机与其同驻,验证租户虚拟机对于物理主机的独占使用.其基本思想如下:在每个检测周期内,租户控制其所有虚拟机均不使用某一随机选取的 Cache 区域,探测该 Cache 区域的负载状态;若探测到 Cache 负载状态大于设定的阈值,则表明有其它虚拟机使用了该 Cache 区域,从而表明存在其他租户的虚拟机与该租户的虚拟机同驻.

当目标虚拟机提供公开服务时,其响应服务请求要使用宿主机的 Cache 资源;如果攻击虚拟机与目标虚拟机同驻,那么攻击虚拟机探测到的宿主机 Cache 负载值将明显增大;否则,Cache 负载值将基本保持不变[10].基于上述事实,余思等人[12]将虚拟

机同驻检测问题抽象为 Cache 负载值集合差异性计算问题：首先，在访问和不访问目标虚拟机的情况下，攻击者可以获取两个 Cache 负载值集合；然后，对这两个集合进行对比分析，得到 Cache 负载状态的变化情况；最后，基于 Cache 负载特征匹配的方式推断目标虚拟机与攻击虚拟机是否同驻.

基于资源干扰的虚拟机同驻检测，利用两个虚拟机竞争使用共享资源时是否相互干扰来判断它们是否同驻. 优点是检测准确率较高；缺点是实现相对复杂，容易受到同一宿主机上其它虚拟机的干扰，并且可能破坏宿主机的资源可用性. 此外，Inci 等人[38]发现云平台通过优化资源管理机制和虚拟机隔离机制，可以有效降低同驻虚拟机之间的相互干扰；同时，随着硬件技术的不断发展，云平台中使用的硬件也更加复杂，例如固态硬盘允许多个读/写操作同时进行，在 Amazon EC2 云平台中已经很难观察到硬件性能上的衰减.

4.2.3　基于隐蔽信道的虚拟机同驻检测

隐蔽信道[50]是指两个合谋的恶意进程通过预先约定的方式操作系统中的共享资源而实现的一种信息传输方式. 云计算环境中，合谋的两个虚拟机利用宿主机的共享资源创建隐蔽信道，就可以绕过 VMM，以一种隐秘地方式进行通信. Ristenpart 等人[10]利用基于内存总线和硬盘访问冲突的隐蔽信道来判断两个虚拟机是否同驻；Inci 等人[38]提出一种基于 LLC 访问冲突的隐蔽信道通信方式用于虚拟机同驻检测.

基于隐蔽信道的虚拟机同驻检测基于两个虚拟机是否能够合谋操作同一宿主机的共享资源，来判断它们是否同驻. 这种同驻检测方法的优点是误检率低，检测结果比较准确；但是，存在一个致命的缺点，即只能应用于检测与被检测双方均为受控虚拟机的情况，并不适用于现实应用场景中只有攻击虚拟机受控的情况. 因此，基于隐蔽信道的同驻检测方法一般用于研究人员在实验环境中确定其创建的两个虚拟机同驻这一事实，并在此基础上，分析同驻虚拟机之间的一些特性.

4.3　虚拟机同驻检测方法的评估

为了客观而准确评估相关虚拟机同驻检测方法的性能，研究虚拟机同驻检测方法的评测指标与测评方法具有重要的理论价值和实际意义. 但是，目前有关虚拟机同驻检测方法的研究刚刚起步，而有关其评测研究，则尚未有成熟和完善的理论与方法. 本节分析讨论适用于虚拟机同驻检测评估的测评原则，并根据提出的测评原则对已有的各类检测方法进行对比分析.

4.3.1　虚拟机同驻检测方法的测评原则

虚拟机同驻检测方法需要在保证检测效果的同时兼顾检测效率与检测能力，另外，一个好的同驻检测方法应该在现实环境中可行. 因此，可以从以下几个方面对虚拟机同驻检测方法进行评估.

(1) 检测效果

可以将同驻检测问题抽象为一个二分类问题：根据是否与某个特定的虚拟机同驻，可以将其它所有虚拟机分为同驻和不同驻两类. 表 2 展示了用于衡量同驻检测的分类准确性的混淆矩阵，其中包含同驻检测的全部 4 种可能结果.

正确接受：表示两个虚拟机实际同驻，检测结果也为同驻；

正确拒绝：表示两个虚拟机实际不同驻，检测结果也为不同驻；

错误拒绝：表示两个虚拟机实际同驻，但检测结果为不同驻；

错误接受：表示两个虚拟机实际不同驻，但检测结果为同驻.

表 2　同驻检测的二分问题的混淆矩阵

		同驻检测结果	
		同驻	不同驻
实际是否同驻	同驻	正确接受	错误拒绝
	不同驻	错误接受	正确拒绝

根据混淆矩阵，可以使用分类器的评价指标，包括真正类率、漏报率、误报率、假负类率、正确率、精确度、准确率、召回率以及 F-measure 等[51]，来对同驻检测方法的检测效果进行度量.

(2) 检测效率

检测效率是衡量同驻检测方法性能的一个重要方面，可以利用完成一次同驻检测所需的时间来对检测效率进行评估.

(3) 检测能力

检测能力主要用于反映同驻检测方法能够检测的同驻级别. 由于虚拟机之间对于 CPU 的共享情况直接决定它们之间的 Cache 共享关系，因此，本文中我们根据虚拟机之间共享 CPU 的情况，将它们之间的同驻级别分为：

内核级同驻. 两个虚拟机被分配到同一个 CPU 的同一个内核之上交替运行；

CPU 级同驻. 两个虚拟机被分配到同一个 CPU 之上运行，包括交替地运行于同一个内核之上

(内核级同驻)以及并行地运行于不同内核之上两种情况;

宿主机级同驻.两个虚拟机被分配到同一个物理主机之上运行,它们之间可能被分配到同一个CPU之上运行(CPU级同驻),也可能被分配到不同的CPU上运行.

(4)局限性

局限性主要用于衡量同驻检测方法在现实应用环境下的可行性,可以使用限制条件来描述同驻检测算法的局限性,通常来说,限制条件越严格,算法的局限性越强、可行性越低.

(5)实用性

实用性主要用于衡量同驻检测方法在现实应用场景下的实用价值.

4.3.2 虚拟机同驻检测方法的比较

在4.2节中,先后介绍了3类虚拟机同驻检测方法,本节对这3类虚拟机同驻检测方法进行比较,列举其原理及优缺点,见表3;同时,对各种虚拟机同驻检测方法进行对比分析,见表4.其中,检测效果用"好"、"较好"、"一般"、"差",检测效率、实用性用"高"、"较高"、"中"、"低",检测能力用"内核级"、"CPU级"、"宿主机级"来分别描述.

表3 3类虚拟机同驻检测方法的比较

方法分类	原理	主要优点	主要缺点
基于网络信息的同驻检测方法	利用两个虚拟机的网络信息进行判断	实现简单,检测效率较高,不会对目标虚拟机的运行造成任何影响	检测准确率不高
基于资源干扰的同驻检测方法	基于两个虚拟机竞争使用共享资源时是否相互干扰来判断	检测准确率较高	实现相对复杂,容易受到干扰,可能破坏宿主机的资源可用性
基于隐蔽信道的同驻检测方法	基于两个虚拟机是否能够合谋操作同一宿主机的共享资源进行判断	误检率低,检测结果比较准确	只适用于检测与被检测双方均为受控虚拟机的情况

表4 各种虚拟机同驻检测方法的比较

方法分类	代表文献	检测效果	检测效率	检测能力	限制条件	实用性
基于网络信息的同驻检测方法	[10]	一般	高	宿主机级	无	低
基于资源干扰的同驻检测方法	[12]	较好	中	内核级	目标虚拟机对外提供公开服务	高
	[48-49]	好	较高	宿主机级	需要一台云平台外运行的代理终端配合检测	
基于隐蔽信道的同驻检测方法	[10]	好	中	宿主机级	进行同驻检测的两个虚拟机均为受控虚拟机	低
	[38]	好	低	CPU级		

可以看出:基于隐蔽信道的同驻检测方法误检率低、检测效果最好,但其限制条件较为严格,在现实攻击场景中很难实现,实用性低;基于网络信息的同驻检测方法实现简单,检测效率较高,但其检测准确率低、检测效果一般,而且由于大多数云平台已经支持使用VPC来加强网络隔离,导致其实用性严重降低;基于资源干扰的同驻检测方法相对来说实用性最高,是目前虚拟机同驻检测的主流方法.

5 Cache侧信道信息探测方法

当前跨虚拟机Cache侧信道信息探测方法主要有四种:Prime-Probe方法、Flush-Reload方法、Flush-Flush方法以及Invalidate-Transfer方法.其中,Flush-Flush方法和Invalidate-Transfer方法与Flush-Reload方法的探测原理类似,因此,本节主要对Prime-Probe方法和Flush-Reload方法进行详细的分析和讨论,并将Flush-Flush方法和Invalidate-Transfer方法作为Flush-Reload方法的扩展,进行简单的介绍.

5.1 Prime-Probe方法

Prime-Probe(PP)方法最初由Osvik等人[20]针对单机计算环境提出,2009年,Ristenpart等人[10]首先将其应用于云环境中.

5.1.1 方法简介

利用PP探测Cache侧信道信息时,攻击虚拟机重复下面的步骤:

(1) Prime.用预先准备的数据填充一个或多个Cache组;

(2) Trigger.访问部署在目标虚拟机中的应用,并且等待一段预设的Prime-Probe间隔时间;在这个过程中,由于要响应服务请求,目标虚拟机执行应用并使用Cache;

(3) Probe.重新读取Prime阶段加载的数据,测量并记录各个Cache组的读取时间.

实际探测过程中,为了避免硬件预取导致的时延隐藏问题,通常采用Pointer-Chasing[52]技术对预先准备的数据进行组织和操作.如图9所示,如果目标虚拟机在Trigger阶段使用了一些攻击虚拟机在Prime阶段填充的Cache组,攻击虚拟机

位于这些 Cache 组中的数据将被驱逐出 Cache，导致攻击虚拟机在 Probe 阶段重新读取数据时发生 Cache 失效，从而产生一个较长的读取时间. 因此，根据 Probe 阶段探测到的各个 Cache 组的重载时间，可以判断目标虚拟机在应用执行中使用了哪些 Cache 组.

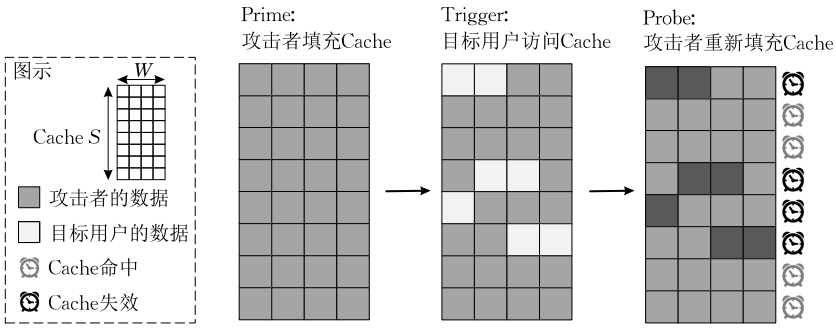


图 9　Prime-Probe 方法

PP 算法基于以下关键因素实现：(1) 攻击者能够识别出与目标用户内存访问活动相关的 Cache 访问，即需要确定目标应用中安全性关键的数据或代码缓存于 Cache 中的哪一个或哪几个 Cache 组中；(2) 攻击虚拟机能够填充指定 Cache 组中的全部 Cache 行，即要求攻击者知道 Cache 与内存地址之间的映射关系；(3) 高分辨率的时钟.

5.1.2　应用 PP 方法探测 L1 Cache

利用 PP 方法探测 L1 Cache 时，通常针对整个 L1 Cache 进行探测，然后再使用机器学习方法识别出与目标用户内存访问活动相关的 Cache 使用[13-14,27].

PP 方法最初只能应用于探测 L1 Cache，这主要是因为 L1 Cache 的容量较小，攻击者可以利用数据或代码的虚拟地址确定其在 L1 Cache 中的具体位置(称为虚拟定址). 具体地，Cache 是物理标记的，即使用物理地址来确定数据或代码在 Cache 中的存储位置(称为物理定址). 对于容量较小的 L1 Cache，攻击者可能拥有其地址映射所需的全部物理地址知识.

如图 10 所示，与传统计算机相比，虚拟机系统中共包含 3 种地址，其中：虚拟地址是指客户机操作系统提供给其应用程序使用的线性地址空间，对应于传统计算机系统中的虚拟地址；物理地址是指虚拟机能够看到的经过 VMM 抽象的伪物理地址；机器地址是指真实硬件的机器地址，即地址总线上应该出现的地址信号，对应于传统计算机系统中的物理地址. 系统在进行虚拟地址到物理地址再到机器地址的转换过程中，只对页号部分进行映射，页内地址偏移部分则保留不变；因此，当 Cache 容量小于等于内存页面大小时，虚拟地址的页内地址偏移部分将保留 Cache 与物理内存之间进行映射所需的全部地址信息，使 Cache 由物理定址变为虚拟定址. 但是，当 Cache 容量大于内存页面大小时，Cache 只能物理定址，此时，攻击者只能控制其数据或代码的虚拟地址，不知道也无法控制其物理地址和机器地址，也就无法控制其私有数据填充指定的 Cache 组.

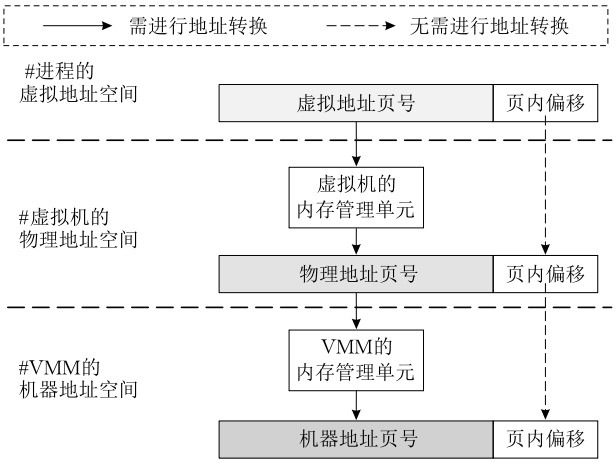


图 10　机器地址、物理地址和虚拟地址

5.1.3　应用 PP 方法探测 LLC

除了物理定址问题，使用 PP 探测 LLC 的挑战还在于：(1) LLC 的非公开哈希索引机制. LLC 各分片与物理内存地址之间的映射关系由一个非公开的哈希函数确定，即使攻击者可以判断一个 Cache 组中包含有哪些 Cache 行，也不知道这些 Cache 行分别对应于 LLC 的哪个分片；(2) 安全性关键访问相关的 Cache 组. 对于容量较大的 LLC(通常大于 2 MB)来说，对整个 LLC 进行一次 Prime 或 Probe 的时间较长，如果每次都探测整个 LLC，将无法以足够的分辨率探测得到有用的信息. 导致使用 PP 探测 LLC 时，只能监控那些与目标应用安全性关键

访问相关的 Cache 组，因此需要确定目标应用中安全性关键的数据或代码缓存于 LLC 中的哪一个或几个 Cache 组中.

针对上述问题，Liu 等人[14]通过在攻击虚拟机中使用大内存页，利用大页映射产生的额外页内地址偏移来保留 LLC 与虚拟地址之间的映射关系；同时，利用 Cache 组内各 Cache 行之间的驱逐关系来按照分片进行 Cache 行划分，绕过了 LLC 地址映射不透明的问题；最后，通过对 LLC 进行扫描，一次监控一个 Cache 组在探测周期内的时态访问模式，并判断是否与目标应用的安全性关键访问一致，直到确定目标应用中安全性关键的数据或代码对应的 Cache 组. 与此同时，Irazoqui 等人[17]也利用大页映射解决 LLC 的地址映射问题，并通过将探测数据从除 LLC 之外的多级 Cache 中驱逐的方法来降低其它高级别 Cache 对于探测结果的影响. Inci 等人[38]和 Kayaalp 等人[39]以及文献[53-55]通过对 LLC 的索引哈希机制进行逆向工程，恢复了物理内存地址与 LLC 的地址映射关系，从根本上解决了 LLC 与物理内存地址映射不透明的问题. Kayaalp 等人[39]则通过访问一个 Cache 组时是否对目标应用造成影响来判断该 Cache 组是否对应于目标应用的安全性关键访问.

5.2 Flush-Reload 方法

2013 年，Yarom 等人[16]利用 Cache 刷新指令(Clflush)可以将指定内存块从全部多级 Cache 中驱逐以及 Cache 具有包容性的特点，对文献[21]中的方法进行了扩展，提出了 Flush-Reload(FR)攻击方法. Flush-Reload 基于共享内存实现，是虚拟化环境中第 1 个跨内核、跨虚拟机的 Cache 探测方法.

5.2.1 方法简介

利用 FR 探测 Cache 侧信道信息时，攻击虚拟机重复下面的操作：

(1) Flush. 将共享内存中指定位置的内存块驱逐出 Cache；

(2) Trigger. 访问部署在目标虚拟机中的应用，并且等待一个预先设定的 Flush-Reload 间隔；在这个过程中，由于要响应服务请求，目标虚拟机执行应用并使用 Cache；

(3) Reload. 重新加载这些指定位置的内存块，测量并记录各个数据块的重载时间.

如图 11 所示，Flush 阶段，攻击虚拟机将指定位置的内存块驱逐出 Cache，如果在 Trigger 阶段目标虚拟机访问了其中的一些内存块，这些数据将被重新加载到 Cache；那么，Reload 阶段对这些内存块的重载将发生缓存命中，从而得到一个显著降低的重载时间. 因此，根据 Reload 阶段的重载时间，可以判断目标虚拟机是否访问了共享内存中指定位置的数据块，并进一步判断目标虚拟机访问了哪些数据或执行了哪些指令.

FR 算法实现基于以下关键因素：(1) 攻击虚拟机和目标虚拟机共享内存页面；(2) 攻击虚拟机可以无限制地使用 Cache 刷新指令；(3) 攻击虚拟机知道目标应用中安全性关键的数据或代码的内存位置；(4) Cache 多级架构具有包容性的特点；(5) 高分辨率的时钟.

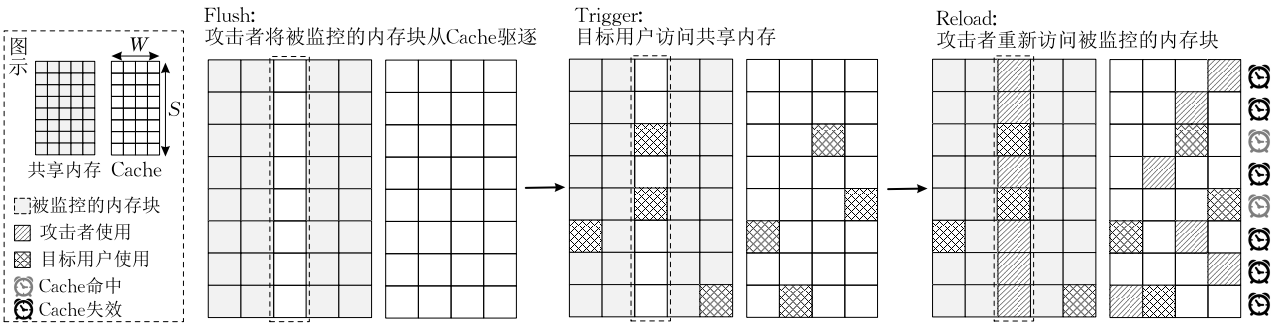


图 11 Flush-Reload 方法

5.2.2 Flush-Reload 方法的应用

实际应用时，为了得到目标应用中安全性关键的数据或代码的内存位置，攻击者通常需要对被攻击的目标应用进行逆向工程，来定位这些安全性关键的内存块的具体位置. Flush-Reload 攻击基于假设攻击虚拟机和目标虚拟机共享内存页面实现，这种共享可以基于页面的来源，比如共享的代码库或加密库；或者基于合并内容相同的页面，比如 Linux 的内核同页合并(Kernel Same Page Merging，KSM)[56]，VMWare ESX 的透明页面共享(Transparent Page Sharing，TPS)[57]以及 Xen 的内存写时复制(Memory Copy on Write，Memory COW)[58]

等.共享内存页面可以为攻击者提供丰富的信息,例如:其它同驻虚拟机中使用的 ECDSA 签名算法的密钥[59-61]和 AES 密钥[62-63],运行的软件加密库[64],加密消息的明文[65],同驻租户的敏感数据[66]以及击键信息[67]等.

5.2.3　Flush-Reload 方法的扩展

(1) Flush-Flush 方法

Gruss 等人[40]发现 Cache 刷新指令 Clflush 的执行时间可以用来判断被刷新的内存块是否位于 Cache 中:被刷新的内存块不在 Cache 中时,Clflush 指令将提前终止,从而产生较短的执行时间;被刷新的内存块在 Cache 中时,Clflush 指令需要将其从全部多级 Cache 中驱逐,导致执行时间较长.基于此,Gruss 等人去掉了 Flush-Reload 方法中的 Reload 步骤,提出了 Flush-Flush 攻击方法.Flush-Flush 方法连续两次使用 Clflush 指令将指定位置的内存块驱逐出 Cache,并根据第 2 次 Clflush 指令的执行时间来判断在这期间目标虚拟机是否使用了这些内存块.由于去掉了 Reload 步骤,Flush-Flush 方法缩短了一轮探测所需时间,可以提供更高的探测分辨率,同时也增加了攻击的隐匿性.但是,与 Cache 命中和失效产生的 Cache 访问时间差异信息相比,Clflush 指令执行时间的差异信息较小,导致 Flush-Flush 方法的探测准确率没有 Flush-Reload 方法高.

(2) Invalidate-Transfer 方法

2016 年,Irazoqui 等人[18]发现在多处理器配置下,由于需要保持多处理器间 Cache 内容的一致性,当一个处理器发生 Cache 失效时,失效的数据是否缓存于另一个处理器 Cache 中,将产生从另一个处理器 Cache 读取数据和直接从内存读取数据两种可能,这将导致 Cache 访问时间差异(大约 50 个时钟周期).基于上述事实,Irazoqui 等人首次提出并实现了跨处理器的 Invalidate-Transfer 攻击方法,并恢复了另一个处理器上运行的加密软件库中使用的 AES 密钥和 ElGamal 密钥.Invalidate-Transfer 方法先将指定的共享内存块在 Cache 中的副本标记为无效,然后等待一段时间后,再重新访问上述指定的内存块,并根据重载时间来判断在此期间攻击目标是否使用了这些内存块.进一步地,根据重载时间的差异还可以判断攻击者和攻击目标是否运行于同一个 CPU 之上.

5.3　Cache 侧信道信息探测方法的比较

本节从信息泄露来源、应用场景、实现条件、优缺点等方面对 Prime-Probe 方法和 Flush-Reload 方法进行对比分析,见表 5.

从表 5 可以看出,Prime-Probe 方法和 Flush-Reload 方法针对不同的应用需求,有各自的优缺点及应用挑战,具体方法的选取取决于应用场景和攻击者具备的能力等.当攻击虚拟机和目标虚拟机之间共享内存时,采用 Flush-Reload 方法,攻击者可以得到更细粒度且更为丰富的信息;当攻击虚拟机和目标虚拟机没有共享内存时,攻击者只能采用 Prime-Probe 方法.因此,Flush-Reload 方法的适用范围受到了严格的限制,Prime-Probe 方法则更为通用.

表 5　Prime-Probe 方法和 Flush-Reload 方法比较

方法	信息泄露来源	泄露信息	应用场景	实现条件	主要优点	主要缺点	代表文献
PP	Cache 争用	目标虚拟机访问了哪些Cache组	分时共享 L1 Cache	为了得到足够的探测分辨率,攻击虚拟机需要频繁地抢占物理 CPU 调度,保证和目标虚拟机交替运行于同一个 CPU 核心之上	(1)实现简单 (2)对于目标应用不敏感,通用性强	(1)容易受噪声影响 (2)只能得到目标应用内存访问的地址信息 (3)分析方法相对复杂	[10,13-14,17,37-39,68-69]
			同时共享 LLC	(1)需要识别目标应用中安全性关键的代码或数据缓存于哪些 Cache 组中; (2)需要恢复 LLC 与内存地址之间的映射关系			
FR	数据重用	目标虚拟机访问了哪些内存块	只要求共享内存,不限制 Cache 共享方式	(1)要求多级 Cache 具有包容性特点 (2)需要知道目标应用中安全性关键的数据或代码的物理内存地址 (3)可以无限制地使用 Cache 刷新指令	(1)能够得到更精确的信息 (2)不受 Cache 与内存之间地址映射以及地址多样化技术(如 ASLR)的影响 (3)分析方法相对简单	(1)应用范围受到限制,只适用于共享内存的攻击场景 (2)依赖于 Cache 刷新指令的使用	[16,59-60,62-64,66-67]

6　总结与展望

作为云计算环境面临的一种重要安全威胁,跨虚拟机 Cache 侧信道攻击已引起了学术界和工业界的广泛关注.本文分析了跨虚拟机 Cache 侧信道攻击的攻击机理和实现方式,并着重对虚拟机同驻相关问题和 Cache 侧信道信息探测技术进行了阐述.

总体来说,跨虚拟机 Cache 侧信道攻击还是一个较新的研究领域,许多研究工作仍然处于起步阶段,在理论和应用上都还存在一些有价值的研究问题.

6.1 虚拟机同驻检测方面

攻击虚拟机和目标虚拟机同驻是跨虚拟机 Cache 侧信道攻击的必要条件和重要基础,而虚拟机同驻检测方法则是实现同驻的过程中比较重要且关键的一个部分.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,只能作为其它同驻检测方法的辅助手段使用;基于隐蔽信道的同驻检测方法误检率低、检测效率也比较高,但其实用性低,只能用于检测双方均为受控虚拟机的情况,这在现实攻击场景中很难实现;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,适用范围最广,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观察发现.因此,如何设计更为准确、高效、实用的虚拟机同驻检测方法,还需要进一步研究.

6.2 跨虚拟机时序驱动攻击方面

目前,跨虚拟机时序驱动攻击的相关研究较少,主要是将传统时序驱动攻击方法直接应用于跨虚拟机攻击场景中.一方面,跨虚拟机时序驱动攻击并没有解决传统时序驱动攻击中存在的两个主要问题:一是攻击条件过于严格,为了采集加密时间以创建模板,攻击者需预先获取目标系统的详细配置参数并重建相同的本地对照环境,这在实际攻击场景中很难实现;二是创建模板时所需样本量大,一般都以百万计,而且离线分析方法比较复杂[32].因此,如何在没有对照环境的情况下,设计能够直接采集目标端的加密时间来创建模板的攻击方案是一个富有挑战性的工作;如何将更多的数学分析方法与现有的离线分析方法结合起来,从而简化分析过程,提高密钥分析效率,从而减少创建模板所需要采集的样本量,也是颇具挑战性的问题.另一方面,根据分析方法不同,传统时序驱动攻击可以分为 Cache 碰撞计时攻击和 Cache 计时模板攻击两类[70],而现有跨虚拟机时序驱动攻击都属于 Cache 计时模板攻击.因此,将传统 Cache 碰撞计时攻击引入跨虚拟机攻击场景中,并解决迁移过程中可能遇到的问题,也具有重要的意义.

6.3 跨虚拟机访问驱动攻击方面

跨虚拟机访问驱动攻击是一种非入侵式的攻击方式,由于其利用虚拟机之间共享的底层硬件资源进行攻击,而不是利用系统本身的逻辑漏洞进行攻击,因此,极难防御.目前,跨虚拟机访问驱动攻击已经有一些研究成果,一方面,利用虚拟化系统环境为攻击者带来的新能力,例如,修改内存页面大小等,对传统的 Prime-Probe 攻击方法进行了扩展,使其能够支持跨内核攻击;另一方面,基于某些系统共享内存页面的特点,设计提出了 Flush-Reload、Flush-Flush 以及 Invalidate-Transfer 攻击方法,并且已经可以支持跨 CPU 进行攻击.

但是,现有跨虚拟机访问驱动攻击方法大多是在较为理想的假设条件下进行,如假设攻击者具有密码库及密码算法实现方式的背景知识、可以精确地在密码算法执行前后探测采集 Cache 侧信道信息,以及可以准确地定位密码算法使用的 Cache 组地址等.因此,在以后的研究中,如何摆脱上述假设条件的限制,提高跨虚拟机访问驱动攻击在现实环境中的实用性和通用性,也是一个极具挑战性的研究方向.另外,现有攻击方法大多没有考虑虚拟化环境中的各种软硬件特征、VMM 系统负载以及其它同驻虚拟机活动等给 Cache 探测结果引入的噪声和干扰,这也将严重影响攻击的实用性,因为在现实环境中这类噪声和干扰是真实且普遍存在的,很难避免.因此,对虚拟化环境中存在的各种噪声和干扰进行分类、建模、量化,并有针对性地设计过滤方法,也是一个值得研究的方向.

6.4 跨虚拟机 Cache 侧信道攻击方法评估方面

由于跨虚拟机 Cache 侧信道攻击是针对密码算法实现的一种攻击方式,其攻击效果受底层硬件配置、系统环境、密码库版本、密码实现方式、信息采集手段以及攻击者和攻击目标的同驻等级等多个因素的影响,导致很难对不同攻击方法的可行性及威胁性进行统一的量化评估.因此,如何建立评估指标体系和评估模型,量化各种因素对于攻击方法的影响,从而对众多的攻击方法进行统一客观的性能评估,也是今后的一个重要研究方向.

6.5 其它方面

云计算包含 3 种不同的服务类型:基础设施即服务(Infrastructure as a Service,IaaS)、平台即服务(Platform as a Service,PaaS)以及软件即服务(Software as a Service,SaaS).目前,云环境中有关跨虚拟机 Cache 侧信道攻击的研究工作,除文献[66]以外,都是围绕 IaaS 云平台展开.因此,将 Cache 侧信道攻击扩展到 IaaS 层之上的 PaaS 或 SaaS,甚至轻量级虚拟化方案 Docker 中,也具有重

要的意义.

7　结束语

跨虚拟机 Cache 侧信道攻击是云安全领域的热点问题.随着云计算的持续快速发展,该问题得到了国内外研究人员的极大关注.针对这一问题,本文首先讨论了共享 Cache 带来的信息泄露问题,这是深入理解 Cache 侧信道攻击机理与方法的理论基础;然后介绍了跨虚拟机 Cache 侧信道攻击的起源与研究进展,分析总结了其与传统 Cache 侧信道攻击的区别与联系,并给出了跨虚拟机访问驱动 Cache 侧信道攻击的通用模型;接下来,介绍了虚拟机同驻相关问题,讨论了虚拟机资源调度策略对于虚拟机同驻的影响,介绍了虚拟机同驻检测的各种方法,分析讨论了适用于虚拟机同驻检测方法评估的测评原则,并根据提出的测评原则对各种虚拟机同驻检测方法进行了对比分析;最后,重点阐述了当前用于跨虚拟机 Cache 侧信道信息探测的主要方法,并从信息泄露来源、应用场景、实现条件等方面进行了分析比较.

本文还重点讨论了跨虚拟机 Cache 侧信道攻击研究中存在的问题和挑战,并展望了未来可能的研究方向.

致　谢　在此,我们向对本文提出宝贵修改意见的评审老师和同行表示衷心的感谢!

参　考　文　献

[1] Chen Kang, Zheng Wei-Min. Cloud computing: System instances and current research. Journal of Software, 2009, 20(5): 1337-1348(in Chinese)
(陈康,郑纬民. 云计算:系统实例与研究现状. 软件学报, 2009, 20(5): 1337-1348)

[2] Foster I, Zhao Y, Raicu I, et al. Cloud computing and grid computing 360-degree compared//Proceedings of the 2008 Grid Computing Environments Workshop. Austin, USA, 2008: 1-10

[3] Kandukuri B R, V R P, Rakshit A. Cloud security issues//Proceedings of the 2009 IEEE International Conference on Services Computing. Bangalore, India, 2009: 517-520

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

[5] Fernandes D A, Soares L F, Gomes J V, et al. Security issues in cloud environments: A survey. International Journal of Information Security, 2014, 13(2): 113-170

[6] Ardagna C A, Asal R, Damiani E, et al. From security to assurance in the cloud: A survey. ACM Computing Surveys (CSUR), 2015, 48(1): 1-50

[7] Sailer R, Jaeger T, Valdez E, et al. Building a MAC-based security architecture for the Xen opensource hypervisor//Proceedings of the 21st Annual Computer Security Applications Conference. Tucson, USA, 2005: 276-285

[8] Wang Z, Jiang X. HyperSafe: A lightweight approach to provide lifetime hypervisor control-flow integrity//Proceedings of the 2010 IEEE Symposium on Security and Privacy. Oakland, USA, 2010: 380-395

[9] Tupakula U K, Varadharajan V. Dynamic state-based security architecture for detecting security attacks in virtual machines. Computer Journal, 2012, 55(4): 397-409

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

[11] Kocher C P. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems//Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology. Santa Barbara, USA, 1996: 104-113

[12] Yu Si, Gui Xiao-Lin, Zhang Xue-Jun, et al. Co-residency detection scheme based on shared cache in the cloud. Journal of Computer Research and Development, 2013, 12(12): 2651-2660(in Chinese)
(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. 计算机研究与发展, 2013, 12(12): 2651-2660)

[13] Zhang Y, Juels A, Reiter M K, et al. Cross-VM side channels and their use to extract private keys//Proceedings of the 2012 ACM Conference on Computer and Communications Security. Raleigh, USA, 2012: 305-316

[14] Liu F, Yarom Y, Ge Q, et al. Last-level cache side-channel attacks are practical//Proceedings of the 2015 IEEE Symposium on Security and Privacy. San Jose, USA, 2015: 605-622

[15] Zhao Xin-Jie, Wang Tao, Guo Shi-Ze, et al. Cache attacks on block ciphers. Journal of Computer Research and Development, 2012, 49(3): 453-468(in Chinese)
(赵新杰,王韬,郭世泽等. 分组密码 Cache 攻击技术研究. 计算机研究与发展, 2012, 49(3): 453-468)

[16] Yarom Y, Falkner K. Flush + Reload: A high resolution, low noise, L3 cache side-channel attack//Proceedings of the the 23rd USENIX Conference on Security Symposium. San Diego, USA, 2014: 719-732

[17] Irazoqui G, Eisenbarth T, Sunar B. S$A: A shared cache attack that works across cores and defies VM sandboxing—and its application to AES//Proceedings of the 2015 IEEE Symposium on Security and Privacy. San Jose, USA, 2015: 591-604

[18] Irazoqui G, Eisenbarth T, Sunar B. Cross processor cache attacks//Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security. Xi'an, China, 2016: 353-364

[19] Kelsey J, Schneier B, Wagner D, et al. Side channel cryptanalysis of product ciphers//Proceedings of the 5th European Symposium on Research in Computer Security. Louvain-la-Neuve, Belgium, 1998: 97-110

[20] Osvik D A, Shamir A, Tromer E. Cache attacks and countermeasures: The case of AES//Proceedings of the 2006 the Cryptographers' Track at the RSA Conference on Topics in Cryptology. San Jose, USA, 2006: 1-20

[21] Gullasch D, Bangerter E, Krenn S. Cache games—bringing access-based cache attacks on AES to practice//Proceedings of the 2011 IEEE Symposium on Security and Privacy. Berkeley, USA, 2011: 490-505

[22] Aciiçmez O. Yet another micro-architectural attack: Exploiting I-cache//Proceedings of the 2007 ACM Workshop on Computer Security Architecture. Fairfax, USA, 2007: 11-18

[23] Aciiçmez O, Brumley B B, Grabher P. New results on instruction cache attacks//Proceedings of the 12th International Conference on Cryptographic Hardware and Embedded Systems. Santa Barbara, USA, 2010: 110-124

[24] Bonneau J, Mironov I. Cache-collision timing attacks against AES//Proceedings of the 8th International Conference on Cryptographic Hardware and Embedded Systems. Yokohama, Japan, 2006: 201-215

[25] Zhao Xin-Jie, Wang Tao, Guo Shi-Ze, et al. Access driven Cache timing attack against AES. Journal of Software, 2011, 22(3): 572-591(in Chinese)
(赵新杰, 王韬, 郭世泽等. AES访问驱动Cache计时攻击. 软件学报, 2011, 22(3): 572-591)

[26] Tsunoo Y, Saito T, Suzaki T, et al. Cryptanalysis of DES implemented on computers with cache//Proceedings of the 5th International Workshop Cryptographic Hardware and Embedded Systems. Cologne, Germany, 2003: 62-76

[27] Percival C. Cache missing for fun and profit//Proceedings of the BSDCan 2005. Ottawa, Canada, 2005: 1-13

[28] Chen C S, Wang T, Chen X C, et al. An improved trace driven instruction cache timing attack on RSA. Journal of Software, 2011, 2011(7): 1683-1694

[29] Chen Cai-Sen, Wang Tao, Guo Shi-Ze, et al. Research on trace driven instruction Cache timing attack on RSA. Journal of Software, 2013, 24(7): 1683-1694(in Chinese)
(陈财森, 王韬, 郭世泽等. RSA踪迹驱动指令Cache计时攻击研究. 软件学报, 2013, 24(7): 1683-1694)

[30] Chen Cai-Sen, Wang Tao, Guo Shi-Ze, et al. Research on trace driven data Cache timing attack against RSA. Chinese Journal of Computers, 2014, 37(5): 1039-1051(in Chinese)
(陈财森, 王韬, 郭世泽等. 针对RSA算法的踪迹驱动数据Cache计时攻击研究. 计算机学报, 2014, 37(5): 1039-1051)

[31] Bernstein D J. Cache-timing attacks on AES. VLSI Design IEEE Computer Society, 2005, 51(2): 218-221

[32] Zhao Xin-Jie, Wang Tao, Zheng Yuan-Yuan. Research on access driven Cache timing attacks against Camellia. Chinese Journal of Computers, 2010, 33(7): 1153-1164(in Chinese)
(赵新杰, 王韬, 郑媛媛. Camellia访问驱动Cache计时攻击研究. 计算机学报, 2010, 33(7): 1153-1164)

[33] Weiß M, Heinz B, Stumpf F. A cache timing attack on AES in virtualization environments//Proceedings of the 16th International Conference on Financial Cryptography and Data Security. Bonaire, Netherlands, 2012: 314-328

[34] Irazoqui G, Inci M S, Eisenbarth T, et al. Fine grain cross-VM attacks on Xen and VMware are possible//Proceedings of the 4th IEEE International Conference on Big Data and Cloud Computing. Sydney, Australia, 2014: 506-512

[35] Irazoqui G, Inci M S, Eisenbarth T, et al. Fine grain cross-VM attacks on Xen and VMware//Proceedings of the 2014 IEEE 4th International Conference on Big Data and Cloud Computing. Sydney, Australia, 2014: 737-744

[36] Zhang Y, Juels A, Oprea A, et al. HomeAlone: Co-residency detection in the cloud via side-channel analysis//Proceedings of the 2011 IEEE Symposium on Security and Privacy. Oakland, USA, 2011: 313-328

[37] Younis Y A, Kifayat K, Shi Q, et al. A new prime and probe cache side-channel attack for cloud computing//Proceedings of the 2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing. Liverpool, UK, 2015: 1718-1724

[38] Inci M S, Gulmezoglu B, Irazoqui G, et al. Seriously, get off my cloud! Cross-VM RSA key recovery in a public cloud. IACR Cryptology ePrint Archive, Report 2015/898, 2015: 1-15

[39] Kayaalp M, Abu-Ghazaleh N, Ponomarev D, et al. A high-resolution side channel attack on the last-level cache//Proceedings of the 53rd Annual Design Automation Conference. Austin, Texas, 2016: 1-6

[40] Gruss D, Maurice C, Wagner K. Flush+Flush: A stealthier last-level cache attack. arXiv e-prints: 1511.04594, 2015: 1-14

[41] Owens R, Wang W. Non-interactive OS fingerprinting through memory de-duplication technique in virtual machines//Proceedings of the 30th IEEE International Performance Computing and Communications Conference. Orlando, USA, 2011: 1-8

[42] Suzaki K, Iijima K, Yagi T, et al. Memory deduplication as a threat to the guest OS//Proceedings of the Fourth European Workshop on System Security. Salzburg, Austria, 2011: 1-6

[43] Zuo Li-Yun, Cao Zhi-Bo. Review of scheduling resarch in cloud computing. Application Research of Computers, 2012, 29(11): 4023-4027(in Chinese)
(左利云, 曹志波. 云计算中调度问题研究综述. 计算机应用研究, 2012, 29(11): 4023-4027)

[44] Xu Z, Wang H, Wu Z. A measurement study on co-residence threat inside the cloud//Proceedings of the 24th USENIX Conference on Security Symposium. Washington, USA, 2015: 929-944

[45] Varadarajan V, Zhang Y, Ristenpart T, et al. A placement vulnerability study in multi-tenant public clouds//Proceedings of the 24th USENIX Conference on Security Symposium. Washington, USA, 2015: 913-928

[46] Barker S K, Shenoy P. Empirical evaluation of latency-sensitive application performance in the cloud//Proceedings of the 1st Annual ACM Sigmm Conference on Multimedia Systems. Phoenix, USA, 2010: 35-46

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

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

[49] Bates A, Mood B, Pletcher J, et al. On detecting co-resident cloud instances using network flow watermarking techniques. International Journal of Information Security, 2014, 13(2): 171-189

[50] Zhenyu W, Zhang X, Haining W. Whispers in the hyper-space: High-bandwidth and reliable covert channel attacks inside the cloud. IEEE/ACM Transactions on Networking, 2015, 23(2): 603-615

[51] Wang Cheng, Liu Ya-Feng, Wang Xin-Cheng, et al. Appraisal identification of classifier's performance. Electronic Design Engineering, 2011, 19(8): 13-15(in Chinese)
(王成，刘亚峰，王新成等. 分类器的分类性能评价指标. 电子设计工程，2011，19(8)：13-15)

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

[53] Maurice C, Scouarnec N L, Neumann C, et al. Reverse engineering Intel last-level cache complex addressing using performance counters//Proceedings of the 18th International Symposium on Research in Attacks, Intrusions, and Defenses-Volume 9404 (RAID 2015). Kyoto, Japan, 2015: 48-65

[54] Irazoqui G, Eisenbarth T, Sunar B. Systematic reverse engineering of cache slice selection in Intel processors//Proceedings of the 2015 Euromicro Conference on Digital System Design. Funchal, Portugal, 2015: 629-636

[55] Yarom Y, Ge Q, Liu F, et al. Mapping the Intel last-level cache. IACR Cryptology ePrint Archive, Report 2015/905, 2015: 1-12

[56] Arcangeli A, Eidus I, Wright C. Increasing memory density by using KSM//Proceedings of the Linux Symposium. Quebec, Canada, 2009: 19-28

[57] Waldspurger C A. Memory resource management in VMware ESX server. ACM SIGOPS Operating Systems Review, 2003, 36(SI): 181-194

[58] Barham P, Dragovic B, Fraser K, et al. Xen and the art of virtualization//Proceedings of the 9th ACM Symposium on Operating Systems Principles. Bolton Landing, USA, 2003: 164-177

[59] Benger N, van de Pol J, Smart N P, et al. "Ooh aah… just a little bit": A small amount of side channel can go a long way//Proceedings of the 16th International Workshop on Cryptographic Hardware and Embedded Systems—CHES 2014-Volume 8731. New York, USA, 2014: 75-92

[60] Yarom Y, Benger N. Recovering OpenSSL ECDSA nonces using the Flush+Reload cache side-channel attack. IACR Cryptology ePrint Archive, Report 2014/140, 2014: 140

[61] van de Pol J, Smart N P, Yarom Y. Just a little bit more//Nyberg K ed. Topics in Cryptology—CT-RSA 2015. Switzerland: Springer International Publishing, 2015: 3-21

[62] Irazoqui G, Inci M S, Eisenbarth T, et al. Wait a minute! A fast, cross-VM attack on AES//Stavrou A, Bos H, Portokalidis G eds. Research in Attacks, Intrusions and Defenses. Switzerland: Springer International Publishing, 2014: 299-319

[63] Gülmezoğlu B, Inci M S, Irazoqui G, et al. A faster and more realistic Flush+Reload attack on AES//Mangard S, Poschmann A Y eds. Constructive Side-Channel Analysis and Secure Design. Switzerland: Springer International Publishing, 2015: 111-126

[64] Irazoqui G, Inci M S, Eisenbarth T, et al. Know thy neighbor: Crypto library detection in cloud. Proceedings on Privacy Enhancing Technologies, 2015, 1(1): 25-40

[65] Irazoqui G, Inci M S, Eisenbarth T, et al. Lucky 13 strikes back//Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security. New York, USA, 2015: 85-96

[66] Zhang Y, Juels A, Reiter M K, et al. Cross-tenant side-channel attacks in PaaS Clouds//Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security. Scottsdale, USA, 2014: 990-1003

[67] Gruss D, Spreitzer R, Mangard S. Cache template attacks: Automating attacks on inclusive last-level caches//Proceedings of the 24th USENIX Conference on Security Symposium. Washington, USA, 2015: 897-912

[68] Gajrani J, Mazumdar P, Sharma S, et al. Challenges in implementing cache-based side channel attacks on modern processors//Proceedings of the 2014 27th International Conference on VLSI Design and 2014 13th International Conference on Embedded Systems. Mumbai, India, 2014: 222-227

[69] Pratiba D, Shobha G, Tandon S, et al. Cache based side channel attack on AES in cloud computing environment. International Journal of Computer Applications, 2015, 119(13): 14-17

[70] Wang Tao, Zhao Xin-Jie, Guo Shi-Ze, et al. Research of Cache timing template attacks on AES. Chinese Journal of Computers, 2012, 35(2): 325-341(in Chinese)
(王韬, 赵新杰, 郭世泽等. 针对AES的Cache计时模板攻击研究. 计算机学报, 2012, 35(2): 325-341)

LIANG Xin, born in 1987, Ph. D. candidate. Her current research interests include cloud computing, virtualization security and side-channel attacks.

GUI Xiao-Lin, born in 1966, Ph. D., professor, Ph. D. supervisor. His research interests include cloud security, network and information security, privacy protection.

DAI Hui-Jun, born in 1979, Ph.D., engineer. Her research interests include network routing and privacy security.

ZHANG Chen, born in 1992, M. S. candidate. His research interests include cloud computing and distributed systems.

Background

Cloud computing is a kind of emerging network service mode delivering information infrastructures and computing resources as IT services. As cloud computing provides remarkable convenience to the customers (enterprises or individuals), its characteristics, such as resource sharing, multi-tenant cross-domain sharing and platform' openness, make cloud security problems more complex. The existing research has shown that cross-VM side channel attacks (CSCA) are becoming a new security challenge faced by cloud computing.

This paper provides a review of cross-VM cache side channel attacks (CSCA) in cloud computing environment, analyzing its mechanism and implementations and summarizing its research status and advances. Followed by the analysis of cache side information leakage and presentation of universal attack model, this paper also category and expound the problem of VMs co-residency and cache side information probing in detail, and point out the existing problems and possible research trends in the future. We hope that all of these can help other researchers.

This work is partly supported by the National Natural Science Foundation of China under Grant No. 61472316, the Science and Technology Project of Shaanxi Province under Grant Nos. 2013SZS16 and 2016ZDJC-05, and the Fundamental Research Funds for the Central Universities under Grant No. XKJC2014008. These projects aim to promote the healthy and development of cloud computing, and reach the international advanced level of the cloud security field. The group has been working on the attack patterns and attack schemes of SCA, side information probing, VMs co-residency detection and SCA detection etc. Many papers have been published in international conferences and journals, such as ICOIN, Electronics and Electrical Engineering, Journal of Internet Technology and the Scientific World Journal etc.