Computer Science and Application 计算机科学与应用, 2024, 14(6), 196-207
Published Online June 2024 in Hans. https://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2024.146156

Hans汉斯

一种云环境中的虚拟机回滚安全模型

黄　庆[1]，许阳光[1]，姜文超[2]，代炜琦[3*]

[1]广州市数字政府运营中心，广东　广州
[2]广东工业大学计算机学院，广东　广州
[3]华中科技大学网络空间安全学院，湖北　武汉

收稿日期：2024年5月23日；录用日期：2024年6月22日；发布日期：2024年6月30日

摘　要

虚拟机(VM)回滚可能会被滥用来对系统发动攻击，导致无论是否使用可信计算技术(TPM)，云环境安全问题始终存在，对云计算安全提出严峻挑战。本文报告了在使用TPM环境中和不使用TPM环境中实施成功的攻击实验，分析了导致云安全问题的根本原因，提出一种基于回滚弹性虚拟TPM (rvTPM)的虚拟机回滚安全解决方案。rvTPM基于Xen环境设计实现并测试，实验结果表明：rvTPM在确保虚拟机回滚安全的前提下，不会对云计算环境造成任何明显的性能损失。

关键词

rvTPM，TPM，云计算，虚拟机回滚

A Security Model for Virtual Machine Rollback in Cloud Environments

Qing Huang[1], Yangguang Xu[1], Wenchao Jiang[2], Weiqi Dai[3*]

[1]Guangzhou Digital Government Operations Center, Guangzhou Guangdong
[2]School of Computer Science and Technology, Guangdong University of Technology, Guangzhou Guangdong
[3]School of Cyber Science and Engineering, Huazhong University of Science and Technology, Wuhan Hubei

Received: May 23rd, 2024; accepted: Jun. 22nd, 2024; published: Jun. 30th, 2024

Abstract

Virtual machine (VM) rollback may be misused to launch attacks on the system, leading to persistent security problems in cloud environments, regardless of whether Trusted Platform Module

*通讯作者。

文章引用: 黄庆, 许阳光, 姜文超, 代炜琦. 一种云环境中的虚拟机回滚安全模型[J]. 计算机科学与应用, 2024, 14(6): 196-207. DOI: 10.12677/csa.2024.146156

(TPM) is utilized or not. This poses a serious challenge to the security of the cloud. This paper reports successful attack experiments conducted in both TPM-enabled and non-TPM-enabled environments, analyzes the fundamental causes of cloud security issues, and proposes a secure solution for virtual machine rollback based on rollback-resilient virtual TPM (rvTPM). rvTPM is designed and implemented based on Xen environments. The experimental results demonstrate that rvTPM provides a secure solution for virtual machine rollback without causing any significant performance degradation in cloud computing environments.

Keywords

rvTPM, TPM, Cloud Computing, Virtual Machine Rollback

1. 引言

虚拟机(VM)是云计算的基石[1]，但虚拟化也带来一系列新的安全漏洞问题[2]-[8]。通常的虚拟机回滚机制安全方案[8]要求用户禁用虚拟机回滚机制或虚拟机暂停/恢复功能。另一种解决方案[9]要求用户自主分析，以便判断是否有虚拟机回滚攻击的行为发生，而这在实际应用中无法落实。在未使用 TPM 的环境中，攻击可归因于回滚虚拟机时应用程序状态的丢失，攻击者可以在应用程序毫无察觉的情况下回滚其状态。在采用 TPM 技术的环境中，安全威胁的识别可归因于虚拟机的实际运作状态与由虚拟 TPM (vTPM)记录的虚拟机状态之间存在的差异，即 VM-vTPM 的不匹配情况[10]。

Garfinkel 和 Rosenblum 较早发现虚拟机回滚可能带来严重的安全后果[11]。Xia 等人提出一种解决虚拟机回滚的安全模型，然而其威胁模型假设管理程序是恶意的[9]。鉴于恶意管理软件能够在未征得云服务用户同意的情况下，擅自对虚拟机执行快照拍摄、挂起及恢复等操作，解决方案应聚焦于开发一个精简、专注且可信的管理程序。这样的管理程序能够为虚拟机提供防护，使其免遭恶意通用管理程序的侵害。相比 SaaS 供应商和云用户可能发起的攻击，假设通用管理程序可信，威胁模型将更切合实际。在虚拟机环境中拍摄和恢复快照的相关研究主要集中在如何实现虚拟机回滚[5] [12]-[17]，然而针对不使用 vTPM 的环境，无法在使用 vTPM 的云环境中使用。Parno 等人[18] [19]研究了如何实现程序状态的高效、鲁棒回滚，使回滚操作不会导致系统崩溃。Gofman 等人[20]观察到虚拟机快照机制可能会被滥用来检索敏感数据，建议对于涉及敏感数据的进程不使用快照机制，然而这种解决方案无法解决 VM-vTPM 之间的差异。Goldman 等人[21]研究了如何通过使用计数器来防止重放攻击，计数器在每次 vTPM 操作后都会增加。这意味着虚拟机回滚后，vTPM 功能会因为计数器的减少而受到破坏。England 等人[22]建议将 vTPM 移入管理程序，以保护计数器不被回滚。

本文探讨了虚拟机回滚过程中的一个固有特定安全漏洞，该漏洞被广泛用于攻击恢复等事件[2] [9] [13] [14]，展示了如何利用虚拟机回滚机制对有或没有 TPM 的云环境发起攻击的过程，并提出基于回滚弹性虚拟 TPM (rvTPM)的安全方案，旨在赋予应用程序识别自身状态是否受到虚拟机回滚操作影响的能力。为了有效利用 TPM 进行问题解决，应用程序开发者需熟悉使用 rvTPM 编程。rvTPM 对应用程序编程人员透明，因此 rvTPM 既不会对应用程序进行任何修改，也不要求对客户操作系统(OS)进行任何修改，简单有效且确保云计算环境性能不明显下降。

2. 虚拟机回滚

如图 1 所示，假设云计算环境为用户提供计算资源，软件服务供应商(SaaS) Isaac 通过 SaaS VM 向用户提供软件服务，用户 Peggy 运行用户虚拟机来构建个人网站。

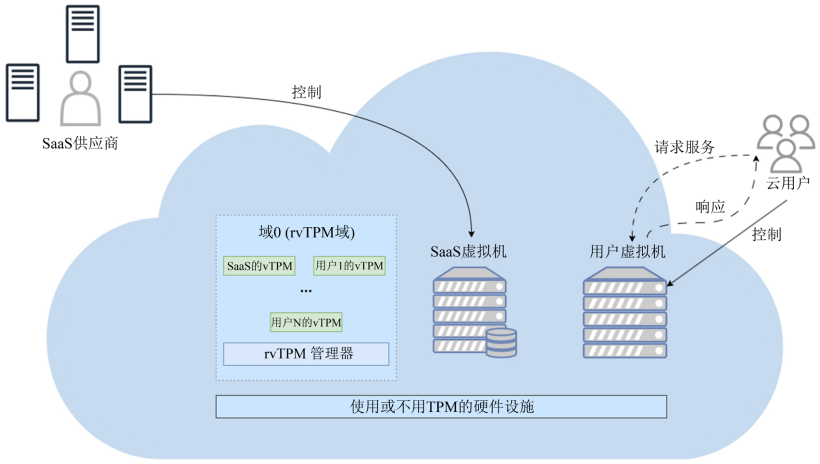

Figure 1. An example of a cloud environment using the current virtual machine roll-back mechanism (regardless of whether TPM technology is used) for attacks
图 1. 利用当前虚拟机回滚机制(无论是否使用 TPM 技术)进行攻击的云环境示例

2.1. 攻击方式 1

成功地重用被泄露的密码。如图 1 所示，假设 Peggy 使用本地计算机(或智能手机)中的电子邮件客户端，连接到位于云端的 Isaac 提供的 SaaS 虚拟机中的 Java Apache 邮件企业服务器[23]。假设服务器通过密码 pw 和基于密码的安全身份验证协议对 Peggy 进行身份验证。具体攻击场景如下。

a) 20:10：Peggy 的密码 pw 被侵入本地计算机的恶意软件破坏，这导致密码信息可能被泄露给了未授权的第三方。

b) 20:20：Peggy 本地计算机中的恶意软件被检测并清除。鉴于密码 pw 可能已经遭到泄露，Peggy 遵循服务器的密码更新流程，将其密码更新为一个新的密码 pw'。

c) 20:30：拥有虚拟机回滚权限(但无其他权限)的攻击者回滚 SaaS 虚拟机到 20:15 时的状态(当时 pw 为有效密码)。或者，SaaS 虚拟机出于合法目的回滚[11]。由于这些回滚操作，攻击者都可能利用密码 pw 验证服务器，从而冒充 Peggy。

该攻击可以在 Amazon EC2 平台上成功实施，其中 SaaS 虚拟机的操作系统为 Amazon LinuxAMI，内核为 3.10.34-37.137.amzn1.x86 64，实例类型为 t1.micro，有 1 个虚拟 CPU 和 0.615GB 内存，服务器为 Apache James 2.3.2。攻击只需要回滚 SaaS 虚拟机的权限，不需要其他权限。

2.2. 攻击方式 2

TPM 能够确保系统的可信引导，向远程验证方证实当前状态，以及对敏感数据进行加密和解密操作[24]。鉴于每台实体计算机都装有一个独立的 TPM，云服务环境中必须实现 TPM 的虚拟化，这样才能保证每台虚拟机都能够利用到 TPM(即 vTPM)。在合法的使用场景中，虚拟机通常需要回滚到早期状态[11]。然而，目前的 VTPM/TPM 设计并未考虑到 vTPM/TPM 的状态回滚问题。这就可能导致记录在 VTPM 中的虚拟机状态与实际虚拟机在执行回滚操作后的状态不一致。这种状态差异，即 VM-VTPM 差异，可能被恶意利用来发起安全攻击。

图 1 展示了恶意的 SaaS 服务提供商 Isaac 如何通过虚拟机回滚功能来盗取用户 Peggy 的敏感数据的过程。在掌握了 SaaS 软件的源代码的情况下，Isaac 有能力植入后门，进而对 Peggy 发起攻击。

a) 21:00：SaaS 虚拟机的客户端操作系统和其对应的 vTPM 均处于 $state_0$，即 vTPM 记录的状态精确反映了虚拟机状态。Isaac 为处于 $state_0$ 的 SaaS 虚拟机拍摄快照。

b) 21:10：新的 SaaS 软件补丁发布。Isaac 更新软件并重启虚拟机。此时虚拟机处于 $state_1$，vTPM 状态也是 $state_1$。Isaac 通过远程验证向 Peggy 证明虚拟机处于 $state_1$(即 SaaS 软件中的漏洞已被修补)。

c) 21:20：Isaac 将 SaaS 虚拟机回滚至 $state_0$，但 vTPM 的状态没有随之更新，即虚拟机与 vTPM 存在差异。vTPM 可能会向 Peggy 证明 SaaS 虚拟机处于 $state_1$ 状态，而它实际上处于 $state_0$。因此，这个不一致可能被利用来窃取 Peggy 在 SaaS 虚拟机中的敏感数据。

攻击者针对未使用 vTPM 或 TPM 系统的攻击之所以有效，根本在于执行虚拟机回滚操作时，应用程序的当前状态可能会无法被追踪或恢复，因为应用程序状态已经回滚到较早状态。为应对这一挑战，必须开发相关机制使应用程序能够识别自身是否因虚拟机的回滚操作遭遇了状态的丢失或变更。为此，需要使应用程序能够检测虚拟机是否已回滚。通过研究，我们观察到如果只是回滚某些 PCR 值，而不是整个 vTPM 状态，同时添加一些永不重置的 PCR，以便远程验证器检测回滚事件，可有效防止回滚攻击。

3. 模型设计

回滚弹性 vTPM (rvTPM)使用 TPM 技术解决攻击问题，为了保护 SaaS 用户免受恶意 SaaS 供应商的侵害，用户应能够识别出存在问题的 SaaS 虚拟机是否经历了回滚操作。为实现这一点，需要依赖一个可信的独立实体来监控并记录虚拟机的回滚行为，同时向 SaaS 用户提供相应的证明，证实这些回滚行为确实已经发生或未发生过。

3.1. 回滚整个 vTPM 状态不安全

为了消除虚拟机与 vTPM 之间的差异，有研究建议回滚整个 vTPM 状态。然而，这种设计并不安全。首先，不应回滚整个 vTPM 实例，否则会破坏 vTPM 计数器的单调性。因此，有必要重放攻击[25]，建议只回滚那些衡量虚拟机完整性的 PCR，以及那些用于 TPM 功能(如密封存储和远程证明)的 PCR。其次，无法回滚系统持久存储(由 TCG 软件栈或 TSS 管理)中的密钥层次结构，由于当前注册的有效密钥(也就是应用程序正在使用的最新一代密钥)有丢失的风险，同时，在虚拟机执行回滚操作之后，那些未被注册且由 TSS 管理的密钥句柄(如已经不再使用或已经失效的受损密钥)可能会重新激活。再次，每次回滚操作后都必须刷新所有句柄，因为回滚操作后虚拟机的所有句柄仍然存在于 vTPM 中，如果不刷新句柄，它们就可以被用来发动攻击。

3.2. 如何检测回滚

为了检测回滚，建议添加 $PCR_{24},\cdots,PCR_{31}$。如表 1 所示。

a) PCR_{24}、PCR_{25} 和 PCR_{26} 被 rvTPM 用于记录虚拟机快照的信息。使用 PCR_{snap} 作为简称，其中 $snap \in \{24,25,26\}$。

b) PCR_{27}、PCR_{28} 和 PCR_{29} 被 rvTPM 用于追踪虚拟机从哪个状态回滚的信息。使用 PCR_{roll} 作为简称，其中 $roll \in \{27,28,29\}$

c) PCR_{30} 用于管理员记录和恢复 vTPM 实例。

d) PCR_{31} 被应用程序用来检测虚拟机回滚。

e) $PCR_{0\sim23}$ 是原始 PCR，简称为 PCR_{orig}，其中 $orig \in \{0,1,\cdots,23\}$。

Table 1. PCR function and permission table
表 1. PCR 功能及权限表

PCR	目的	重置权限	扩展权限
PCR_{24}	快照拍摄时间	rvTPM	rvTPM
PCR_{25}	快照拍摄时的 uid	rvTPM	rvTPM
PCR_{26}	记录拍摄快照时虚拟机状态	rvTPM	rvTPM
PCR_{27}	虚拟机回滚时间	无	rvTPM
PCR_{28}	虚拟机回滚时的 uid	无	rvTPM
PCR_{29}	记录回滚时虚拟机状态	无	rvTPM
PCR_{30}	记录 vTPM 实例数据	无	rvTPM
PCR_{31}	应用检测虚拟机状态	无	应用

由于两个虚拟机快照可能对应 TCG [26]规定的相同 PCR_{orig} 值，而且应用程序可能会在某个快照下封存其敏感数据(即封存的敏感数据不得在任何其他快照下解封)，因此需要区分不同的快照。为此，让 rvTPM 使用 PCR_4 记录虚拟机快照的系统时间(管理程序时间戳)。当云用户封存其敏感数据时，用户可将数据封存在与 PCR_{24} 唯一值相对应的快照下。在进行第 n 次封存操作时，有下式成立：

$$PCR_{24}^{n} = H\left(PCR_{24}^{n-1} \| H\left(snapshot_{time}\right)\right) \tag{1}$$

其中，$PCR_{24} = 0$，H 是 TCG 采用的标准加密哈希函数，||是连接词操作，而快照时间则是拍摄快照时的管理程序时间戳。由于 PCR_{24} 只能由 rvTPM 扩展或重置，因此任何虚拟机都无法对其进行操作。为防止出现上述 VM-rvTPM 差异，PCR_{24} 将与虚拟机一起回滚。为了让云用户(或远程验证者)确定虚拟机何时被回滚以及回滚到何时，使用 PCR_{27} 记录虚拟机回滚的系统时间：

$$PCR_{27}^{n} = H\left(PCR_{27}^{n-1} \| H\left(rollback\_time \| snap\_shot\right)\right) \tag{2}$$

其中，$PCR_{27}^{0} = 0$，回滚时间是发生回滚时的管理程序时间戳。PCR_{27} 只能由 rvTPM 扩展，不能被任何实体重置，从而防止它被任何虚拟机操纵。

由于 SaaS 虚拟机可由多个用户使用，而这些用户通过 PCR_{orig} [26]拍摄了相同的 SaaS 虚拟机快照，并且需要区分不同用户的敏感数据可能被封存的快照，因此推荐利用用户 ID (uid)辨识各用户的快照。为有效纳入该信息，提议增设 PCR_{25}，当拍摄快照时，rvTPM 会扩展 PCR_{25}，以包含拍摄快照的用户的 uid，具体如下：

$$PCR_{25}^{n} = H\left(PCR_{25}^{n-1} \| H\left(uid\_snap\right)\right) \tag{3}$$

其中，$PCR_{25}^{0} = 0$。PCR_{25} 只能由 rvTPM 扩展或重置。为防止出现 VM-vTPM 差异如上所述，PCR_{25} 将与相应的虚拟机一起回滚。

为了让云用户确定是谁将相关虚拟机回滚到谁的快照，添加 PCR_{28} 记录记录回滚虚拟机的用户的 uid：

$$PCR_{28}^{n} = H\left(PCR_{28}^{n-1} \| H1\left(uid\_roll \| uid\_snap\right)\right) \tag{4}$$

其中，$PCR_{28}^{0} = 0$。PCR_{28} 只能由 rvTPM 扩展，但不能重置。

当应用程序封存敏感数据时，数据必须与 PCR_{orig} 所代表的虚拟机状态绑定[26]。为确保这一点，添加了 PCR_{26} 来记录虚拟机状态，用 $VTPM\_snap = PCR_0 \| \cdots \| PCR_{23}$，如下所示：

$$PCR_{26}^{n} = H\left(PCR_{26}^{n-1} \| H\left(vTPM\_snap\right)\right) \tag{5}$$

其中， $\mathrm{PCR}_{26}^{0}=0$。PCR_{26} 只能由 rvTPM 扩展或重置。

与 PCR_{26} 类似，如果虚拟机从状态 B(源状态)回滚到状态 A(目的状态)，添加 PCR_{29}，来记录这一回滚操作：

$$\mathrm{PCR}_{29}^{n}=\mathrm{H}\left(\mathrm{PCR}_{29}^{n-1}\parallel\mathrm{H}\left(\mathrm{vTPM\_snap}^{\mathrm{src}}\parallel\mathrm{vTPM\_snap}^{\mathrm{dest}}\right)\right) \tag{6}$$

其中， $\mathrm{PCR}_{29}^{0}=0$。PCR_{29} 只能由 rvTPM 扩展，但不能重置。

当云管理员需要还原整个 vTPM 实例时，需要记录 vTPM 实例数据(用 vTPM 数据表示)，记为 vTPM_data。为了方便操作，添加 PCR_{29}，使得：

$$\mathrm{PCR}_{29}^{n}=\mathrm{H}\left(\mathrm{PCR}_{29}^{n-1}\parallel\mathrm{H}\left(\mathrm{vTPM\_snap}^{\mathrm{src}}\parallel\mathrm{vTPM\_snap}^{\mathrm{dest}}\right)\right) \tag{7}$$

PCR_{30} 只能由 rvTPM 扩展，但不能重置。

远程挑战者(例如云用户)可以使用 $\mathrm{PCR}_{\mathrm{roll}}$ 来验证虚拟机回滚事件，并判断 SaaS 供应商是否通过回滚事件欺骗他。然而，应用程序不能简单地使用 $\mathrm{PCR}_{\mathrm{roll}}$ 来检测虚拟机回滚时其状态的丢失，因为它无法知道每个 VM 回滚事件的影响，也无法识别哪个虚拟机回滚操作可能会损坏其状态。所以需要添加 PCR_{31} 来允许应用程序记录自己的状态转换，如下所示：

$$\mathrm{PCR}_{30}^{n}=\mathrm{H}\left(\mathrm{PCR}_{30}^{n-1}\parallel\mathrm{H}\left(\mathrm{vTPM\_data}\right)\right) \tag{8}$$

其中， $\mathrm{PCR}_{31}^{0}=0$。E_i 表示第 i 个敏感操作(例如更改密码)对应的信息，该操作将改变应用程序的敏感状态。当应用程序状态发生变化时，它将有关其状态变化的敏感操作信息 E_i 记录到日志中并将 E_i 扩展到 PCR_{31}，然后将秘密消息 m_i(密码或私钥)密封到 PCR_{31}。后面的应用程序可以通过测试哪个步骤的消息 $(m_1,m_2,\cdots,m_n)$可以解封来确定应用程序正在处理哪个状态。如果应用程序处于状态 E_x，则当前 PCR_{31} 值应为 $H\left(H\left(\mathrm{PCR}_{31}^{0}\parallel H\left(E_x\right)\right)\right)$。如果 m_x 消息能够成功解封，则表明应用程序处于状态 E_x 并且不是丢失状态，否则，应用程序能够察觉到虚拟机执行了回滚操作，因为其当前状态与存储在 PCR_{31} 中的真实状态存在差异。应用程序可以通过日志强制恢复与 PCR_{31} 的值匹配的状态，这个 PCR 只能由应用程序扩展但无法重置。

3.3. 安全分析

基于 rvTPM 的虚拟机回滚机制既不会被滥用而导致应用程序状态“丢失”，也不会被滥用而导致虚拟机与 vTPM 之间的差异。rvTPM 旨在保障用户虚拟机中的应用程序能够借助 PCR_{31} 记录其历史状态，并赋予系统强制调整应用程序状态的能力。为了造成应用程序状态的“丢失”，能够回滚虚拟机的攻击者必须能够操纵 PCR_{31} 的值，使其与回滚操作后的应用程序状态相匹配。然而这是不可能的，因为 PCR_{31} 从不重置。

rvTPM 可以确保在进行虚拟机回滚操作后，从相应的 vTPM 快照中恢复 vTPM 状态，还可以确保从一个状态到另一个状态的执行路径是唯一的。为了让攻击者造成 VM-vTPM 差异，攻击者必须确保 SaaS 虚拟机处于可信状态 S_A，以便将证明传递给 SaaS 用户(否则，SaaS 用户就会知道 SaaS 虚拟机不处于安全状态)。假设在成功执行验证后，攻击者将 SaaS 虚拟机拉回到易受攻击的状态 S_B。使用服务后，用户将验证 $\mathrm{PCR}_{\mathrm{roll}}$ 记录的服务时间内 SaaS 虚拟机的状态转换。因此，如果攻击者想消除攻击痕迹，必须通过操纵 $\mathrm{PCR}_{\mathrm{roll}}$ 来欺骗云用户，使虚拟机通过可信状态路径运行。然而这是不可能的，因为 $\mathrm{PCR}_{\mathrm{roll}}$ 只能由 rvTPM 扩展，而且永远不能重置。

4. 实验原型

实验原型系统以 Linux 2.6.18 为域 0 的 Xen 3.3.1 虚拟机管理程序上实施 rvTPM。图 2 给出了 rvTPM

的四个模块。域 0 中的信息收集模块负责搜集回滚操作信息，而 Xen 中的信息注册模块记录和共享状态信息。回滚模块位于域 0，负责执行快照和回滚操作。回滚日志模块位于域 0，负责记录回滚操作。

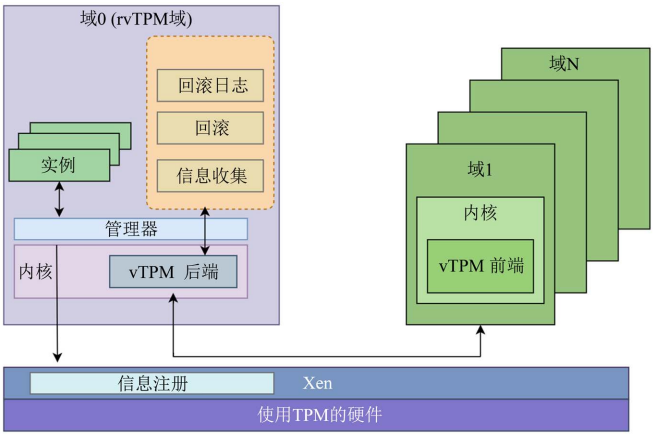

Figure 2. Xen-based rvTPM system architecture, it mainly consists of four modules
图 2. 基于 Xen 的 rvTPM 系统体系结构，它主要由四个模块组成

4.1. 信息收集模块

在拍摄快照或将虚拟机回滚到上一个快照时，会调用该模块来收集以下信息：1) 虚拟机名称；2) 对应的 vTPM ID；3) 用户名：(iv)用户 ID；4) 系统当前时间；5) vTPM 快照的路径。该模块通过一个新定义的超级调用 do_vtpm_rollback，由下面描述的信息注册模块提供，将收集到的信息添加到新添加的全局请求表中，全局请求表的结构是一个链表，每个条目代表一个请求。

4.2. 信息登记模块

信息收集模块收集的信息存储在请求表中。每个 rvTPM 模块都能通过超级调用 do_vtpm_rollback 编辑和检索请求表。为了应对可能出现的并发写入全局表的情况，使用自旋锁[27]确保独占访问。在模块尝试进行编辑请求表操作(如添加、移除或查询条目)时，若需获取锁，它会进入一个循环等待状态，直至锁变为可用。

4.3. 回滚模块

在进行虚拟机快照或将虚拟机回滚到之前的快照时，该模块会记录 snap_time、uid_snap、vTPM_snap 分别扩展到 PCR_{snap} 中的 roll_time || snap_time,uid_roll || uid_snap, $vTPM\_snap^{src}$ || $vTPM\_snap^{dest}$ 分别扩展到 PCR_{roll} 中。并更新日志以包含这些信息项，供远程验证使用。为确保快照和回滚过程的安全性，使用 PCR_{snap}，PCR_{roll} 表示快照。在此机制下，仅允许回滚至这些 PCR 所记录的状态，而 PCR_{roll} 则不允许回滚。一旦状态回滚，相关 vTPM 句柄将被清除，阻止应用程序利用这些句柄访问与回滚虚拟机无关的安全敏感资源(如密钥)。为了确保系统持久性存储不会回滚，将系统持久性存储的目录作为 NFS 运行，以便域 0 挂载和共享。由于域 0 不会回滚，系统持久存储最新状态。

当 rvTPM 管理器接到快照请求，它会查询新建立的全局表中的 vTPM ID。若查询未发现相关信息，即表明用户未发起快照操作。反之，若找到数据，则说明信息收集与注册模块已成功搜集并记录了进行回滚所需的信息。在定位到先前注册的信息之后，rvTPM 管理器将要求 rvTPM 扩展到 PCR_{snap}，并提取 PCR_{orig} 和 PCR_{snap} 中的所有哈希值。rvTPM 管理器将这些 PCR 值存储至一个特定文件中，并复制该文件到某个路径以便回滚。rvTPM 管理器还会将 rvTPM 实例数据保存到该路径，以便管理员执行适当的维护

任务。

首先，rvTPM 管理器会利用 vTPM 的 ID 在全局表中搜索。若查询到相关信息，管理器便会将 vTPM 的快照复制至硬盘。在回滚过程中，由于会生成一个新的 rvTPM 实例，因此可以从已保存的快照中恢复出 rvTPM 的实例。启动过程如下：首先，用未回滚的信息创建 vTPM 实例；其次，检查 rvTPM 的启动类型。只要 rvTPM 的启动类型是 TPM ST STATE，就会调用恢复和扩展 PCR 的功能；第三，在与上一步相同的功能中，确定是否是回滚请求，如果 rvTPM 实例是为其他目的而不是为恢复快照而创建的，则将通过搜索全局表再次确定；第四，一旦确定是回滚请求，该函数将替换 PCR 的值，从 PCR_0…到 PCR_{26}；第五，它将刷新所有与 vTPM 相关的句柄；第六，在执行了所有可能引起 rvTPM 实例数据变更的操作后，保存并存储 rvTPM 实例数据；最后，PCR_{27}…到 PCR_{29} 将被更新来记录此类回滚事件。

4.4. 回滚日志模块

使用新增的 PCR_{snap} 和 PCR_{roll}，可以跟踪回滚事件。然而，尽管 PCR 中存在哈希值，但仍不足以恢复回滚历史。在用户执行拍摄快照或恢复至早期快照的操作时，rvTPM 会在扩展 PCR 时记录日志。日志包括回滚时间(拍摄，恢复快照的时间)、执行的动作(快照或回滚)、用户 ID、虚拟机名称、虚拟机快照的路径(拍摄或恢复到的快照)。日志将有助于远程验证和 TPM_Quote 操作。

5. 性能评估

实验平台基于一台配备四核 1.99 GHz AMD Opteron 处理器、4GB 内存和 v1.2 Broadcom TPM 修订级别 A2 的服务器进行实验。软件环境为 Ubuntu 8.04 LTS 上的准虚拟化 Xen 3.3.1，内核为 2.6.18.8-xen。用户虚拟机运行相同的准虚拟化内核，域 U 分别为 32M、64M、128M、256M、512M 和 lG 内存和 1 个虚拟 CPU 内核。域 0 中运行 NFS 版本 3 服务器，保证 rvTPM 的系统持久存储。

5.1. 使用 rvTPM 的 OpenSSH 服务器

将 rvTPM 与 OpenSSH 服务集成[27]。rvTPM 能让 OpenSSH 服务自动检测密钥丢失，并在虚拟机回滚事件中继续安全工作。为此，需要对 OpenSSH 服务器进行如下修改，该修改包含在 ssh-keygen.c 和 sshd.c 中。扩展 PCR 函数、密封函数和解除密封函数。

a) 当为 OpenSSH 服务器生成新的主机服务器密钥对时，OpenSSH 服务器会将其公钥扩展到 PCRs1，在回滚过程中该公钥不可恢复。然后，OpenSSH 服务器将主密钥封存在 PCR 下。ssh-keygen 命令由 ssh-keygen.c 实现，对主函数稍作修改(添加扩展函数和封存函数)。

b) 在密钥使用前，通过解封操作验证其有效性。解封成功则密钥与 PCR_{31} 一致，不会发生回滚；否则，密钥与 PCR_{31} 不一致，由于 PCR 的值是不可恢复的，指示系统可能已回滚，密钥被恢复到了以前的密钥。创建 sshd 守护进程时，需在主函数中加载并解封主机密钥对。

c) 如果解封操作失败，即虚拟机已回滚到之前的状态，OpenSSH 服务器应放弃当前使用的公钥和私钥对，转而从日志中恢复主机密钥对。使用 rvTPM 系统和更新后的 OpenSSH 服务器，可以有效防止会话密钥在回滚操作中泄露，从而在发生此类事件时通知管理员进行进一步的调查。因为 OpenSSH 服务器可以实时检测回滚操作。

5.2. 系统成本测量

使用不同内存下的不同域 U 配置进行实验，以测试 rvTPM 带来的成本。对于拍摄快照，测量了以下步骤的成本：1) 计算收集信息时的哈希值；2) 超级调用注册信息；3) 超级调用搜索信息；4) PCR 扩展和保存 vTPM 快照；5) 复制 vTPM 快照。对于回滚，测量了以下步骤的成本：1) 计算收集信息时的哈

希值；2) 超级调用注册信息；3) 超级调用搜索信息；4) 回滚 vTPM；5) PCR 扩展。

测试结果如图 3，图 4 所示，在快照和回滚过程中，所有步骤的成本与内存大小并不呈线性关系，可以发现以下几点事实：首先，超级调用非常高效。在快照过程的第 1 步，对当前用户名、时间和 $vTPM_{snap}$ 进行散列。其次，当前用户名和时间的字节数很少，因此计算哈希值花费很少。再次，vTPM 快照的大小固定，因为 24 个 PCR (PCR_0, … ,PCR_{23})的大小固定。在回滚过程的第 1 步中，还对当前用户名、时间和 vTPM 快照进行哈希处理。区别是现在需要对当前虚拟机的 vTPM 快照以及要恢复到的快照的 vTPM 快照(其哈希值等于快照中的 PCR_{26})进行哈希处理，其大小也是固定的。在扩展 PCR 值过程中，将当前时间、用户 ID 和 vTPM 快照的散列分别扩展为 PCR_{24}~PCR_{26} 或 PCR_{27}~PCR_{29} (总共 60 字节)。

此外，如图 5，图 6 所示，通过比较使用 rvTPM 的系统中快照和回滚功能与不使用 rvTPM(但使用 vTPM)的系统中对应功能的成本，可以发现 rvTPM 不会产生任何显著成本。

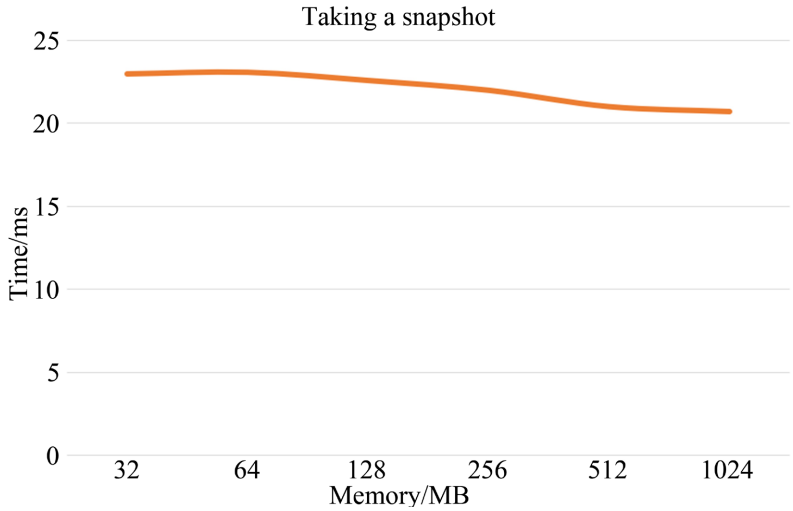

Figure 3. The time cost of rvTPM taking snapshots at different memory sizes
图 3. rvTPM 在不同内存大小下拍摄快照的时间成本

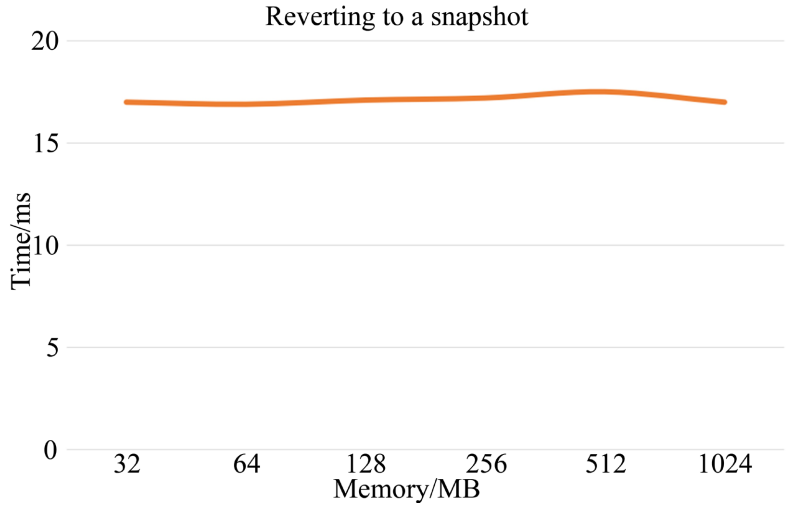

Figure 4. The time cost of rvTPM reverting snapshots at different memory sizes
图 4. rvTPM 在不同内存大小下的恢复快照的时间成本

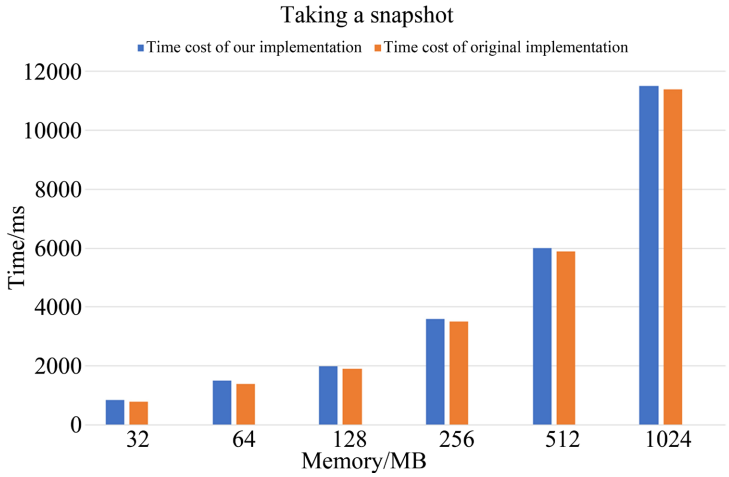

Figure 5. Time cost for taking snapshots of corresponding system functions
图 5. 系统对应功能拍摄快照的时间成本

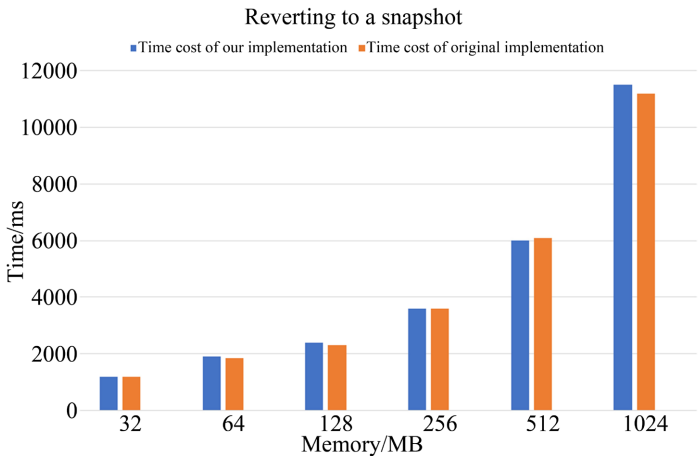

Figure 6. Time cost for reverting snapshots of corresponding system functions
图 6. 系统对应功能恢复快照的时间成本

6. 结论

云环境中的虚拟机回滚机制易导致恶意攻击，本文展示了使用和不使用 TPM 技术的情况下当前的虚拟机回滚机制发起攻击的根源在于状态信息被复用，并基于此设计了回滚弹性虚拟 TPM (rvTPM)回滚机制，并分析了其安全性。原型系统基于 Xen 平台实现 rvTPM，性能评估结果表明：rvTPM 可以有效保证虚拟机回滚安全可靠且不会造成增加显著的效率成本。rvTPM 的提出可以有效促进 TPM 技术在云计算环境中的广泛应用。

基金项目

广东省自然科学基金项目(2024A1515011502)，广东省科技计划项目(2019B010139001)。

参考文献

[1] Chen, P.M. and Noble, B.D. (2001) When Virtual Is Better than Real [Operating System Relocation to Virtual Machines]. *Proceedings Eighth Workshop on Hot Topics in Operating Systems*, Elmau, 20-22 May 2001, 133-138. https://doi.org/10.1109/HOTOS.2001.990073

[2] Garfinkel, T. and Rosenblum, M. (2005) When Virtual Is Harder than Real: Security Challenges in Virtual Machine Based Computing Environments. *Proceedings of the* 10*th conference on Hot Topics in Operating Systems*, Santa Fe, 12-15 June 2005, 20.

[3] Collier, G., Plassman, D. and Pegah, M. (2007) Virtualization's Next Frontier: Security. *Proceedings of the* 35*th Annual ACM SIGUCCS Fall Conference*, Orlando, 7-10 October 2007, 34-36. https://doi.org/10.1145/1294046.1294055

[4] Pearce, M., Zeadally, S. and Hunt, R. (2013) Virtualization: Issues, Security Threats, and Solutions. *ACM Computing Surveys*, **45**, 1-39. https://doi.org/10.1145/2431211.2431216

[5] Chen, L., Xian, M., Liu, J. and Wang, H. (2020) Research on Virtualization Security in Cloud Computing. *Proceedings of the* 2019 *International Conference on AI and Big Data Application*, Guangzhou, 20-22 December 2019, Article 012027. https://doi.org/10.1088/1757-899x/806/1/012027

[6] Zhu, G., Yin, Y., Cai, R. and Li, K. (2017) Detecting Virtualization Specific Vulnerabilities in Cloud Computing Environment. *Proceedings of the* 2017 *IEEE* 10*th International Conference on Cloud Computing* (*CLOUD*), Honololu, 25-30 June 2017, 743-748. https://doi.org/10.1109/cloud.2017.105

[7] Mahipal, S. and Sharmila, V.C. (2021) Virtual Machine Security Problems and Countermeasures for Improving Quality of Service in Cloud Computing. *Proceedings of the* 2021 *International Conference on Artificial Intelligence and Smart Systems* (*ICAIS*), Coimbatore, 25-27 March 2021, 1319-1324. https://doi.org/10.1109/icais50930.2021.9395922

[8] Mansoor, F., Saghar, K., Agha, S.U., *et al*. (2023) Virtual Machine's Network Security. *LC International Journal of STEM*, **4**, 99-127.

[9] Szefer, J. and Lee, R.B. (2012) Architectural Support for Hypervisor-Secure Virtualization. *Proceedings of the Seventeenth International Conference on Architectural Support for Programming Languages and Operating Systems*, London, 3-7 March 2012, 437-450. https://doi.org/10.1145/2150976.2151022

[10] Xia, Y., Liu, Y., Chen, H. and Zang, B. (2012) Defending against VM Rollback Attack. *Proceedings of the IEEE/IFIP International Conference on Dependable Systems and Networks Workshops* (*DSN* 2012), Boston, 25-28 June 2012, 1-5. https://doi.org/10.1109/dsnw.2012.6264690

[11] Berger, S., Cáceres, R., Goldman, K.A., *et al*. (2006) vTPM: Virtualizing the Trusted Platform Module. *Proceedings of the* 15*th USENIX Security Symposium*, Vancouver, 31 July-4 August 2006, 305-320.

[12] Dunlap, G.W., King, S.T., Cinar, S., Basrai, M.A. and Chen, P.M. (2002) ReVirt: Enabling Intrusion Analysis through Virtual-Machine Logging and Replay. *ACM SIGOPS Operating Systems Review*, **36**, 211-224. https://doi.org/10.1145/844128.844148

[13] Dunlap, G.W., Lucchetti, D.G., Fetterman, M.A. and Chen, P.M. (2008) Execution Replay of Multiprocessor Virtual Machines. *Proceedings of the Fourth ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, Seattle, 5-7 March 2008, 121-130. https://doi.org/10.1145/1346256.1346273

[14] Grobauer, B. and Schreck, T. (2010) Towards Incident Handling in the Cloud: Challenges and Approaches. *Proceedings of the* 2010 *ACM Workshop on Cloud Computing Security Workshop*, Chicago, 8 October 2010, 77-86. https://doi.org/10.1145/1866835.1866850

[15] King, S.T., Dunlap, G.W. and Chen, P.M. (2005) Debugging Operating Systems with Time-Traveling Virtual Machines. *Proceedings of the* 2005 *USENIX Annual Technical Conference*, Anaheim, 10-15 April 2005, 1-15.

[16] Liu, H., Jin, H., Liao, X., Hu, L. and Yu, C. (2009) Live Migration of Virtual Machine Based on Full System Trace and Replay. *Proceedings of the* 18*th ACM International Symposium on High Performance Distributed Computing*, Garching, 11-13 June 2009, 101-110. https://doi.org/10.1145/1551609.1551630

[17] Cui, L., Hao, Z., Li, L., *et al*. (2015) Lightweight Virtual Machine Checkpoint and Rollback for Long-Running Applications. *Proceedings of the* 15*th International Conference on Algorithms and Architectures for Parallel Processing*, Zhangjiajie, 18-20 November 2015, 577-596. https://doi.org/10.1007/978-3-319-27137-8_42

[18] Parno, B., Lorch, J.R., Douceur, J.R., Mickens, J. and McCune, J.M. (2011) Memoir: Practical State Continuity for Protected Modules. *Proceedings of the* 2011 *IEEE Symposium on Security and Privacy*, Oakland, 22-25 May 2011, 379-394. https://doi.org/10.1109/sp.2011.38

[19] Cui, L., Hao, Z., Peng, Y. and Yun, X. (2017) Piccolo: A Fast and Efficient Rollback System for Virtual Machine Clusters. *IEEE Transactions on Parallel and Distributed Systems*, **28**, 2328-2341. https://doi.org/10.1109/tpds.2017.2668403

[20] Gofman, M.I., Luo, R., Yang, P. and Gopalan, K. (2011) SPARC: A Security and Privacy Aware Virtual Machine-checkpointing Mechanism. *Proceedings of the* 10*th Annual ACM Workshop on Privacy in the Electronic Society*, Chicago, 17 October 2011, 115-124. https://doi.org/10.1145/2046556.2046571

[21] Goldman, K.A. and Berger, S. (2008) TPM Main Part 3 IBM Commands.

[22] England, P. and Loeser, J. (2008) Para-Virtualized TPM Sharing. *Proceedings of the First International Conference on Trusted Computing and Trust in Information Technologies*, Villach, 11-12 March 2008, 119-132.

[23] James: Java Apache Mail Enterprise Server (2024) http://james.apache.org/

[24] Kinney, S.L. (2006) Trusted Platform Module Basics: Using TPM in Embedded Systems. Newnes.

[25] Sarmenta, L.F.G., van Dijk, M., O'Donnell, C.W., Rhodes, J. and Devadas, S. (2006) Virtual Monotonic Counters and Count-Limited Objects Using a TPM without a Trusted OS. *Proceedings of the First ACM Workshop on Scalable Trusted Computing*, Alexandria, 3 November 2006, 27-42. https://doi.org/10.1145/1179474.1179485

[26] Trusted Computing Group (2013) TCG PC Client Specific TPM Interface Specification (TIS).

[27] OenSSH.org (2012) http://www.openssh.org/