钉枪：突破 ARM 特权隔离

张锋巍　宁振宇
南方科技大学

关键词：硬件辅助调试　ARM 架构　系统安全

随着智能手机、智能电视等智能终端和物联网(IoT) 行业的飞速发展，ARM 架构逐渐在市场上崭露头角。目前，绝大多数的智能终端和 IoT 设备都配备了低能耗高性能的 ARM 架构处理器。与此同时，系统安全始终是一个绕不开的话题，对于智能终端和 IoT 设备来说更是如此。在这些终端和设备上，安全防线一旦被攻破，将导致大范围的隐私信息泄露，大额度的资产流失，甚至可能是大规模的安全生产事故。因此，安全相关的研究人员开始将研究重心放在基于 ARM 架构的软件及操作系统安全上，比如对于智能手机操作系统的安全分析以及对恶意手机软件的研究。然而，对于 ARM 架构的硬件安全研究却较为少见，目前还处于较为初期的阶段。众所周知，硬件是软件运行的基础，一旦硬件的安全架构出现问题，必将危及到所有上层软件的安全性。

“钉枪”[1] 就是这样一个基于 ARM 架构硬件漏洞的攻击方式。通过时常被忽略的 ARM 硬件调试特性以及该特性在多核系统中的设计漏洞，“钉枪”能够绕过 ARM 系统的权限隔离机制，达到以低权限完成高权限操作的目的。例如，在智能手机上，“钉枪”可以通过“非安全模式”下的应用来窃取保存在“安全模式”下的用户指纹信息。

“钉枪”的设计

“钉枪”攻击源于 COMPASS 实验室，它的全称是计算机和系统安全 (COMPuter And System Security) 实验室。该实验室依托于南方科技大学计算机科学与工程系和美国韦恩州立大学计算机系，致力于计算机系统安全方向的前沿研究，所做工作得到了国际安全界的高度认可。在该实验室的项目 Ninja[2] 的研究过程中，我们使用了 ARM 架构的嵌入式追踪宏单元 (Embedded Trace Macrocell, ETM) 来实现与透明追踪 (transparent tracing) 相关的功能。在使用 ETM 这一硬件特性时，我们意外地发现 ARM 架构的权限管理机制在这一特性上表现得较为模糊。简单来说，就是 ARM 允许你在低权限下使用这一硬件特性来追踪高权限下软件的执行情况。最初这一发现只是让我们感到疑惑，当进一步确认可行性之后，我们马上意识到这可能会是一个安全漏洞，并对其展开研究。

能够以低权限追踪高权限的执行情况，这当然是一个可以挖掘的漏洞，但如果仅仅只能实现追踪的话，这一漏洞的影响力必然有限。那么，是否能够通过它来实现在高权限中执行任意代码呢？带着这一问题继续细读 ARM 提供的硬件手册，我们有了崭新的发现。在计算机系统中，与追踪系统齐名的是调试 (debugging) 系统，追踪系统用来观测目标运行情况，而调试系统则可以用来完全控制目标的运行。在对系统进行调试的过程中，通常会有“调试者”和“被调试者”两种角色。一般来说，“调试者”和“被调试者”是两台不同的设备，而“调试者”通过特定的硬件连接线和接口（如 JTAG）来调试“被调试者”的运行情况。根据 ARM 硬件手册的描述，在检查调试权限的过程中，硬件只会检查“被调试者”

的权限而忽略“调试者”的权限，这也是为什么追踪系统会允许以低权限来追踪高权限执行的原因。

如果“调试者”和“被调试者”之间需要有硬件连接的话，这就意味着需要对“被调试者”有物理访问(physical access)，这样做的前提必然会严重限制漏洞的触发场景。那么，如何能绕过这一限制呢？多核系统可以做到。随着硬件技术的发展，现在的终端设备处理器往往都不止包含一个核，如果把每个核都视作一个单独的设备，那能不能用一个核扮演“调试者”的角色来调试另一个核呢？经过进一步研究，我们发现这是可行的。如图1所示，在多核系统中，我们完全可以用一个核通过内存映射接口来调试和控制另外一个核。总结来说，我们可以用一个低权限的核通过调试系统来控制另一个高权限的核，并通过调试命令来实现对高权限资源（寄存器、内存和外设等）的完全控制，从而达到在高权限中执行任意代码的效果。

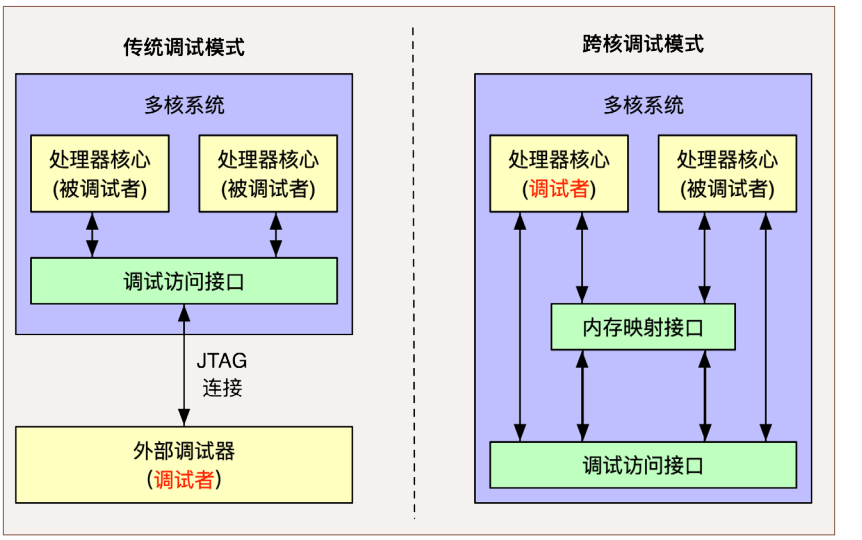

图1　多核系统中的调试模式

“钉枪”的实现

虽然已经通过可行性分析，但是验证的过程却是艰苦万分。无论是让目标核进入和退出调试模式，还是让目标核在调试模式下执行指令、实现权限升级等，每一步成功之前都是无数次的系统崩溃。与普通软件崩溃的情况不同，这类系统崩溃往往不会留下任何有帮助的日志，只能依靠经验和不断地尝试来定位崩溃的原因。而且，由于几乎没有相关的研究，如果想解决碰到的问题，我们唯一可以依赖的就是ARM提供的几千页的硬件手册。在经过了无数次的尝试之后，我们终于在ARM Juno开发板上实现了第一个“钉枪”的雏形。

然而这还只是个开始。为了验证“钉枪”确实能在商用终端和设备上实现，我们测试了市面上各类智能手机、物联网设备和基于ARM架构的云服务器。与ARM Juno开发板情况不同的是，这类终端设备往往缺少公开的硬件手册，有些甚至连内核都未公开，这又进一步加大了验证的难度。比如，在红米6手机上，由于没有合适的内核源码，我们只能通过反编译并修改原有内核模块的方式来达到验证的目的。又比如，在华为Mate 7手机上，为了找出用户指纹在“安全模式”下的存储信息，我们只能通过反编译安全固件的方式来进行蛛丝马迹的寻找。总结下来，我们花了数月的时间在二十多台不同的商用终端设备上进行了验证分析，其中一大半设备都会受到“钉枪”的影响。然而，由于ARM架构的使用极其广泛，我们有理由相信目前在市场上还有大量的未知设备受影响。

“钉枪”的防护

对我们来说，任何安全研究的最终目的都是为了更好地加强计算机系统的安全性，“钉枪”的研究也是如此。发现这一漏洞后，我们立即向ARM进行了详细的报告。同样地，我们也与各大手机厂商和芯片厂商取得了联系，将漏洞以及相关设备上进行的分析研究反馈给他们，并协助进行后续的漏洞

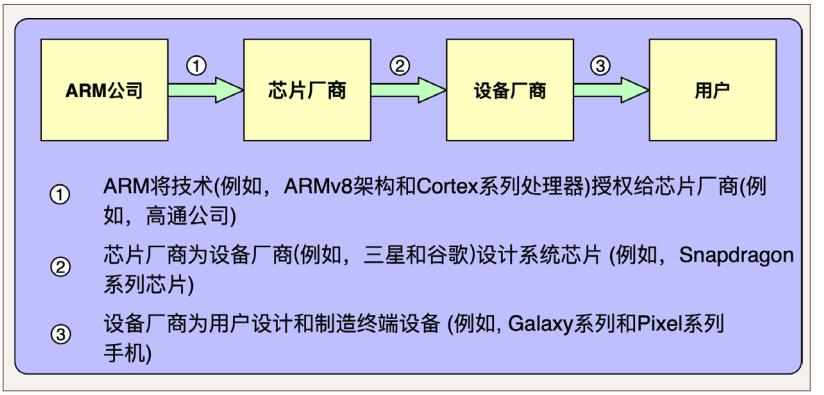

图 2　ARM 生态系统中

修复工作。为了方便对漏洞进行统一管理，我们也已经向美国 MITRE[1] 申请了 CVE ID[2] 来对其进行详细的描述和管理。

与此同时，我们也分别从 ARM 生态系统中不同角色的角度出发，提出了针对“钉枪”的防护方案。图 2 展示了 ARM 生态系统中不同的角色。由于“钉枪”攻击牵涉到这个生态系统中的每一个角色，我们从 ARM 公司、芯片厂商、设备厂商各自的角度出发，提供了一整套具有针对性的有效防御机制。通过与芯片厂商和设备厂商技术团队的交流，我们也在不断优化和完善这一整套“钉枪”防护方案，以便后续研究者参考。

“钉枪”的影响和意义

“钉枪”攻击发表在了计算机安全顶级会议 IEEE S&P 2019 上 [1]，此攻击对于 ARM 平台生态系统产生了很大的影响，有几十亿台终端设备是“钉枪”攻击的潜在受害者，手机厂商（华为、小米、摩托罗拉等）、物联网设备厂商（树莓派等）、云服务厂商（亚马逊等）都可能遭受“钉枪”的攻击。同时，我们也积极联系了这些厂商，他们对“钉枪”攻击都进行了回应和修复。我们也获得了美国 MITRE 给我们对于“钉枪”攻击的 CVE 号码 (CVE-2018-18068)。

追溯“钉枪”攻击发生的始末，我们认为相关漏洞产生的根本原因是在技术革新的过程中，一个原本安全的硬件特性未能及时进行相应的更新和审视，导致其在新的技术环境下出现了安全问题。在“钉枪”攻击中，硬件调试功能在单核环境下原本是一个安全的硬件特性，而随着多核处理器和多核系统的诞生，硬件调试功能的相关特性并没有随之更新，依然保留了单核环境下最初的安全设计，这才导致了攻击的最终发生。因此，我们建议在引入任何新的技术时，一定要谨慎地审视原有功能和特性的安全问题。

一些思考

当我们介绍“钉枪”的时候，很多读者可能会问:“你们是怎样发现这样有意思的攻击的？”或“怎样才能发现漂亮的漏洞？”其实发现好的漏洞有时候是可遇而不可求的，或者说是靠“运气”。如何让自己拥有这份“运气”，或许以下几点可以提供一些帮助：

第一,要有扎实的系统功底。其实在发现“钉枪”攻击前,笔者已经完成了“忍者”(Ninja) 的工作 [2]。“忍者”是一个基于 ARM 的恶意软件分析平台，通过这个项目的开发实现，让我们对 ARM 架构有了非常深入的了解。在这个基础上,我们才能够发现“钉枪”攻击。就好像当一个人看到一条指令或者一个寄存器值时，只有熟悉这个领域才有可能发现它背后潜在的漏洞；不熟悉的人，可能看一眼就过去了。

第二，要有一颗好奇的心。当我们做完“忍者”

[1] MITRE 是美国一个非盈利性组织，它管理联邦政府资助的研发中心，支持几个美国政府机构。

[2] CVE（Common Vulnerabilities and Exposures，公共漏洞和披露）：一种公开信息安全缺陷和暴露的编号形式，类似于漏洞字典。MITRE 是 CVE 的核心运营组织。

工作后，意外地发现 ARM 架构的 ETM 可以追踪高权限的代码。此时“忍者”工作已经完成，并且简单地追踪高权限的代码不足以让同行们感到惊讶。如果没有一颗好奇的心，我们就不会进一步研究这个问题。研究完之后才发现，这不止是一个“小坑”，而是一个“大洞”，并且越挖越深。

第三，要有“自救的韧性”。很多时候科研是极其郁闷的，碰到的具体问题可能没有人能够帮你。就像我们从事的 ARM 架构硬件安全特性研究，前人的工作和经验是少之又少。在“钉枪”的实现过程中，我们经历了无数次的系统崩溃。我们唯一可以依靠的就是 ARM 几千页的手册，在这种情况下，我们没有被无数次的系统崩溃和极其郁闷的心情打败，而是保持着一种“自救的韧性”，通过现有的信息（ARM 手册），最后实现了“钉枪”攻击。 ■

张锋巍

CCF 专业会员。南方科技大学副教授。主要研究方向为系统安全、可信执行、硬件辅助安全、透明调试、车载安全及可否认加密。zhangfw@sustech.edu.cn

宁振宇

南方科技大学访问学生，美国韦恩州立大学博士生。主要研究方向为 ARM 系统安全、可信执行环境及车载安全。

参考文献

[1] Ning Z, Zhang F. Understanding the security of ARM debugging features[C]//Proceedings of the 40th IEEE Symposium on Security and Privacy. IEEE, 2019.

[2] Ning Z, Zhang F. Ninja: Towards transparent tracing and debugging on ARM[C]//Proceedings of the 26th USENIX Security Symposium. USENIX, 2017: 33-49.