3) 底层管理程序提供了一个虚拟化捕获处理器,用来处理被捕获到虚拟机管理程序的中断和异常.底层管理程序只提供少量的必要处理,主要的工作都在切换到高级管理程序后由高级管理程序来完成.

高层管理程序在管理模式中运行,它也作为主机 Linux 内核的一部分,可直接利用现有的 Linux 功能,并使用标准的内核软件数据结构和机制.因此,在高层管理程序中更容易实现高级功能.例如,底层管理程序负责虚拟化捕获的处理与上下文的切换,高层管理程序则处理虚拟机的二级页面错误并执行指令仿真.由图 8 可以看出:虚拟机的内核部分与高层管理程序均运行在拥有相同权限等级的管理模式中,但虚拟机的内核部分使用二级页表翻译,并可通过寄存器的设置被底层管理程序捕获到虚拟化模式中.因此,该部分同样处在 KVM/ARM 的监视和管理之下.

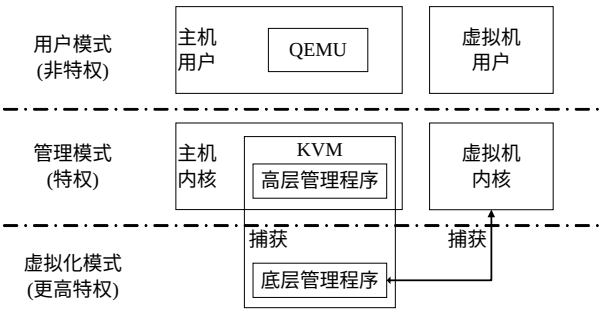

Fig.8 System structure of KVM/ARM

图 8 KVM/ARM 的系统架构图

因为虚拟机管理程序被分割在管理模式和虚拟化模式中,因此在虚拟机和高层管理程序之间的切换涉及多次模式切换.虚拟机中需要被虚拟机管理程序处理的中断或异常,首先要被捕获到底层管理程序中.底层管理程序随后产生另一个异常进入高层管理程序.反之,从高层管理程序进入虚拟机,也需要先从管理模式切换到虚拟化模式,再切换到虚拟机中.因此,分割模式虚拟化在上下文切换时会产生两次模式切换.

3.2.2 系统设计

基于这种分割模式的虚拟化,KVM/ARM 具体实现了多方面的虚拟化功能,主要包括如下 4 点:CPU 虚拟化、内存虚拟化、I/O 虚拟化以及中断虚拟化.

(1) CPU 虚拟化

为实现 CPU 虚拟化,KVM/ARM 必须给虚拟机提供一个与物理 CPU 完全相同的接口,同时确保虚拟机管理程序能控制硬件.因此,KVM/ARM 需确保运行在虚拟机中的软件与运行在物理 CPU 上的软件可以访问到相同的寄存器上下文,同时还要确保在运行虚拟机时,虚拟机管理程序以及其主机内核的物理硬件状态是稳定的.在虚拟机和主机之间相互切换时,通用寄存器的状态可以简单地通过内存读写来实现上下文切换.如果某个对硬件状态的访问可能影响虚拟机管理程序或向虚拟机泄露硬件信息,则 KVM/ARM 将捕获这个敏感指令,并对其进行仿真.

(2) 内存虚拟化

当虚拟机运行时,KVM/ARM 通过开启二级页表翻译来实现内存虚拟化.二级页表翻译只能在虚拟化模式中配置,并对虚拟机完全透明.高层管理程序对二级页表翻译进行管理,使得虚拟机只能访问自己申请的指定内存空间.若虚拟机试图访问其他内存,则产生一个二级页面错误异常,并被虚拟机管理程序捕获与处理.该机制保证了虚拟机不能访问属于虚拟机管理程序或其他虚拟机的内存空间.由于高层管理程序和底层管理程序拥有对系统的完全控制权,因此当它们运行时,二级页表是禁用的.当虚拟机管理程序切换到虚拟机时,会开启二级页表翻译机制,并对二级页表基址寄存器进行相应配置.通过内存虚拟化机制,虽然高层管理程序和虚拟机使用相同的 CPU 模式(管理模式),二级页表翻译确保了高层管理程序不会被其他虚拟机访问.

(3) I/O 虚拟化

KVM/ARM 利用 QEMU 和 Virtio[52]现有的用户空间设备仿真技术来提供 I/O 虚拟化.在硬件级别上,ARM 架构的全部 I/O 机制都是基于对内存映射 I/O(memory mapping I/O,简称 MMIO)设备区域的读取/保存操作.除了被直接分配给虚拟机的设备外,其他 MMIO 设备区域对虚拟机来说都是不可访问的.KVM/ARM 使用二级页表翻译机制来确保物理设备不能被虚拟机直接访问.虚拟机管理程序会捕获虚拟机的每次内存跨界访问,并将相关信息转发给 QEMU 中对应的仿真设备.

(4) 中断虚拟化

由于 KVM/ARM 与 Linux 是紧密结合的,因此它能够重用现有的设备驱动和相关功能,例如中断处理.当程序在虚拟机中运行时,在 KVM/ARM 对 CPU 进行配置后,可将全部的硬件中断捕获到虚拟化模式中.当中断产生时,它会进行状态切换进入高层管理程序,再由主机处理.因此,虚拟机管理程序可以确保对硬件资源的完全控制.当程序在主机以及高层管理程序中运行时,中断就直接被捕获到管理模式中,从而避免了进入虚拟化模式的额外开销.在以上两种情况中,所有硬件中断的处理都在主机中,通过重用 Linux 现有的中断处理功能完成.

3.2.3 基于 KVM/ARM 的扩展

Paolino 等人[53]基于 KVM/ARM 提出了基于内核的可信虚拟机(T-KVM).T-KVM 是一个基于 KVM/ARM 的安全架构,它集成了软件和硬件组件来保护客户操作系统,并支持在 ARM 虚拟机中的可信计算.T-KVM 共包含 4 个独立的组件:ARM 虚拟化扩展、ARM 安全扩展(TrustZone)、可信执行环境(TEE)接口以及 SELinux 的强制访问控制(MAC)模块.T-KVM 架构为客户应用提供了较好的隔离性且支持可信计算.

3.3 OKL4虚拟机管理程序

由于虚拟化技术和微内核技术在很多方面具有相似之处,因此二者的优劣也成为许多人讨论的焦点.Open Kernel Labs 将两者结合在一起,提出了一种基于微内核的虚拟机管理程序——OKL4 虚拟化微内核(OKL4 microvisor)[38].OKL4 虚拟化微内核是 L4 微内核[54]的一个变种,可运行在单核或多核的 ARM,x86 以及 MIPS 处理器平台上.

基于硬件虚拟化扩展,Varanasi[55]在 ARM 上将 OKL4 移植到虚拟化模式中,实现了一个虚拟机管理程序.OKL4 运行在虚拟化模式中,既能发挥其 TCB 小的优势,又能在虚拟化扩展的辅助下实现更好的管理功能.

3.3.1 系统设计

为了让基于 OKL4 的虚拟机管理程序尽可能简单,并保持一个小的 TCB,Varanasi 给出了如下的设计.

- 通过静态编译的方式配置客户虚拟机的上限数量.

一般而言,虚拟机管理程序可通过静态编译配置客户虚拟机的数量上限,也可动态地分配新虚拟机.然而,虚拟机的动态创建比静态编译配置更为复杂,因此需要更多的实现代码,从而增加了 TCB 的代码量,进而增大了系统的安全风险.同时,由于 ARM 平台主要用于移动设备,而一台移动设备上的客户虚拟机数量不会像桌面平台那样大,因此,设计者选择通过静态编译的方式配置客户虚拟机的上限数量.

- 虚拟机管理程序中的内存采用平板映射.

在虚拟机管理程序中,内存映射通常可以采用页表寻址或平板映射.虚拟机管理程序并不会被上层客户操作系统访问,并且采用页表寻址格式需要编写更多代码来实现页表配置.因此,设计者采用简单的平板映射方式来实现虚拟机管理程序中的内存访问.这种方式能使得虚拟机管理程序尽可能的简单,并减小了 TCB.而在客户操作系统中,设计者采用二级页表翻译机制来实现内存访问,如图 9 所示.

- 虚拟机管理程序的组件全部位于虚拟机管理程序中.

不少微内核并没有众多管理组件,而是通过进程间通信(inter-process communication,简称 IPC)和外部管理组件通信来实现管理工作,从而确保微内核的 TCB 足够小.但组件如果存放于上层客户操作系统中,每次操作都需要进行通信,从而对系统性能产生影响.同时,中断管理、二级页表翻译以及状态存取等都是对虚拟机安全至关重要的组件.因此,将这些组件存放于虚拟机管理程序中,在保证系统的性能的同时,也保证了系统的安全性.

- 虚拟机间的通信采用共享内存和异步通信两种方式.

虚拟机间的通信方式主要包括异步通信、同步通信和共享内存.对于内容较大的信息,需采用共享内存的方式;而对于内容较小的信息,通过异步通信的方式可以保证其性能.如果虚拟机之间使用同步机制通信,会让虚拟机因等待返回消息而阻塞.因此,设计者选择共享内存和异步通信的两种方式实现虚拟机之间的通信.

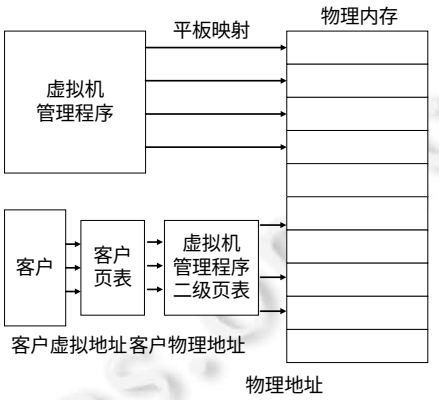

Fig.9 Virtual memory mapping in OKL4 hypervisor

图 9 OKL4 虚拟机管理程序中虚拟内存映射

3.3.2 系统实现

基于 OKL4,Varanasi 在实现中增加了如下的扩展:

- 增加了一个从内核模式进入虚拟化模式的系统调用;
- 为每个客户虚拟机实现了客户页表;
- 增加了对中断和时钟的支持,同时生成中断用于模式切换;
- 扩展了虚拟中断.如果中断不准备进入虚拟机管理程序,则会产生一个虚拟中断传递给客户虚拟机;
- 对一些设备进行了虚拟化;
- 增加了对 Linux 的支持,并可以运行多个 Linux 系统.

4 基于硬件辅助虚拟化技术的安全研究

随着硬件虚拟化技术及其相关研究的不断深入,越来越多的研究者将硬件虚拟化技术和系统安全问题联系起来.在 ARM 虚拟化扩展的辅助下,虚拟机管理程序可以运行在权限更高的虚拟化模式中.因此,虚拟机管理程序可以对操作系统进行有效的管理.因此近几年来,很多人都基于硬件辅助虚拟化技术对系统安全问题进行了研究.

4.1 DroidVisor

由于虚拟机管理程序的权限高于操作系统内核,因此,通过虚拟化来保护内核是一种非常有效的手段.杨永等人[56]利用 ARM 虚拟化扩展实现了一个 Android 内核保护监控器——DroidVisor.

DroidVisor 运行在虚拟化模式中,可以隔离可疑模块,并保护内核中对象的完整性.同时,它还能够通过检测隐藏模块和隐藏进程来检测 rootkit,从而有效地保护内核.

4.1.1 内核完整性保护

操作系统中存在一些可加载内核模块,用来提供额外的功能.攻击者可利用第三方的可加载内核模块进行攻击.DroidVisor 监控可加载内核模块的安装和卸载,如果该模块不在自己的可信列表中,则对其使用单独的二级翻译页表,从而实现内存隔离.当这些模块有异常行为时,DroidVisor 可对这些行为进行拦截和处理.

同时,DroidVisor 还可保护内核中关键对象的完整性.DroidVisor 通过 System.map 文件得到关键对象的虚拟地址,接着,通过二级页表翻译得到对应的物理页表项,并通过设置对应页表项的访问控制位实现访问控制.

DroidVisor 将关键对象的物理页设为只读,当该页被修改时,程序会产生页面错误,该错误会被虚拟机管理程序捕获.虚拟机管理程序捕获到该异常后,可以检查该异常是否由恶意代码导致:如果是由恶意代码导致的,虚拟机管理程序将该异常返回给客户操作系统的异常处理程序,从而实现异常拦截;如果是正常的修改,虚拟机管理程序会负责进行修改,随后返回客户操作系统.

4.1.2 Rootkit 检测

对内核完整性的保护可以防止攻击者对关键对象的修改,但无法有效防护使用进程隐藏和模块隐藏的rootkit.在保护内核完整性的同时,DroidVisor 还在进程切换时对模块列表和进程列表进行检查.如果发现有模块或进程不在已知列表中,则说明存在这种 rootkit.

在进程切换时,操作系统会更改页表的基址寄存器.在 ARM 中通过某些配置,可使该寄存器在修改时被虚拟机管理程序捕获.因此,DroidVisor 选择在进程切换时进行拦截,这样可以检测即将运行的进程是否为隐藏的进程,从而在可疑进程运行前将其阻止.具体的流程如图 10 所示.

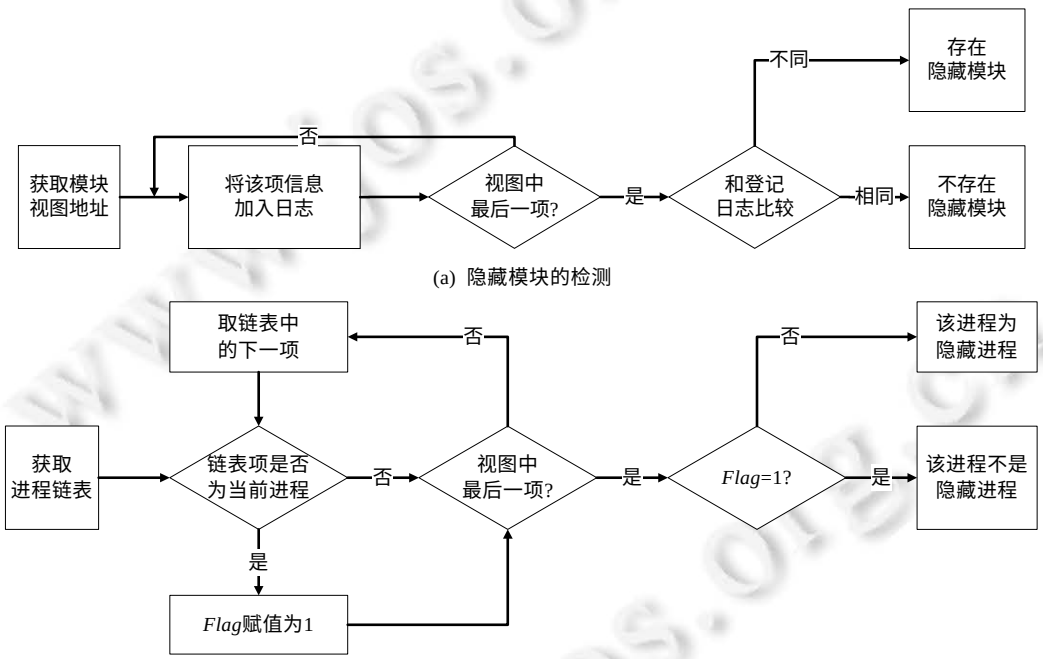

Fig.10 Workflow of rootkit detection

图 10 Rootkit 检测流程图

4.1.3 DroidVisor 自身的保护

DroidVisor 本身是一个轻量级的虚拟机管理程序,因此代码量比较小,即,可信计算基比较小,降低了出现 BUG 的风险.同时,DroidVisor 通过对页表翻译的控制,将虚拟机管理程序的内存空间置于客户操作系统可以访问的范围之外,从而使得客户操作系统无法访问到虚拟机管理程序的内存空间.

同时,对于 DroidVisor 在客户操作系统中加载的辅助模块,DroidVisor 通过二级页表翻译进行保护的同时,也在 TrustZone 的辅助之下实现了一个白名单验证机制,从而更好地保护了 DroidVisor 的功能.

4.2 控制流追踪框架

控制流分析是发现系统是否被攻击的一种重要手段.通过适当的控制流分析,研究者可以了解程序的执行流程,并且通过与正常的控制流对比来确定程序是否正确运行.因此,Horsch 等人[57]使用硬件辅助虚拟化技术建

立了一个控制流追踪的虚拟机管理程序框架,以达到保护程序正常控制流的目的.

该框架共分为 5 个部分:异常处理模块、控制模块、追踪模块、二级页表管理模块和分析模块,如图 11 所示.其中,

- 异常处理模块用于捕获客户操作系统的异常(包括虚拟化调用和页面错误):如果捕获到的异常是虚拟化调用,则传递给控制模块;如果是页面错误,则传递给追踪模块;
- 控制模块负责解析虚拟化调用的参数,并根据参数的值对整个框架的功能进行配置;
- 追踪模块接收到页面错误后,在二级页表管理模块的辅助之下,进行控制流追踪,生成相关的数据,供分析模块使用;
- 分析模块作为一个单独模块,可以根据使用者的不同需求,对拿到的数据进行不同类型的分析,给出相应的结果或进行相应的处理.

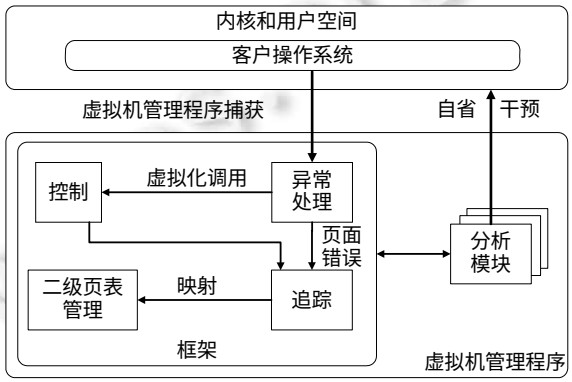

Fig.11 System architecture of control-flow tracing framework

图 11 控制流追踪框架的系统架构

4.2.1 执行流程

在系统启动时,虚拟机管理程序将二级页表的页表项全部设置为不可执行.页表的大小均设置为最小值 4KB,这样可以让控制流的追踪更为精确.虚拟机管理程序随后初始化一个大小为 n 的空列表,用于保存当前的可执行页.其中,n 是可配置的参数,n 越小,表示当前可执行页的总数越小,对控制流的追踪会更为精确;但同时,性能的开销也会更大.因此,通过对控制流分析的精确度和性能的综合考虑,使用者可以配置一个合适的 n 值进行分析.

在系统运行期间,系统的控制流会在页面间进行切换.当页面切换到一个不在可执行页面列表中的页面时,会产生页面错误(预取错误).此时,虚拟机管理程序会拦截该错误,并产生相关的控制流数据,将其发送给分析模块进行分析.

接下来,虚拟机管理程序会将该页设置为可执行,并且将该页加入可执行页面列表中.如果此时可执行页面列表中页面数目大于 n,则虚拟机管理程序依据不同的替换策略,将可执行页面列表中某个页面替换为该页面.

4.2.2 控制流数据

当产生页面错误时,虚拟机管理程序通过追踪模块的分析可以得到多种控制流数据.其中,可以被分析模块用于分析的数据有如下几种.

- 控制流目标:当页面错误产生时,页面错误产生的地址也会保存在相应的寄存器中.该寄存器中保存的是目标页面的虚拟地址,而虚拟机管理程序可根据虚拟地址,通过页表翻译得到对应的物理地址以及物理页;
- 控制流源:控制流源获得的精确程度取决于可执行页面列表的大小 n.由于只有可执行页面列表中的页是可执行的,因此控制流源存在于可执行页面列表中.当 n 越小,这个源就越精确.当 n=1 时,控制流源可

以精确到页.而对于某一种特定的情况来说,如果程序是由带链接分支指令执行的(例如 blx 或 bl),那么链接寄存器(link register,简称 LR)中的值就是精确的源地址;

- 客户上下文:当程序陷入到虚拟机管理程序中时,虚拟机管理程序可以读取到地址空间标识符(address space identifier,简称 ASID)以及进程标识符(process identifier,简称 PID);
- 客户执行状态:当页面中断产生时,虚拟机管理程序可以拿到很多数据,其中包括当前的程序计数器、当前程序状态寄存器(current program status register,简称 CPSR)及堆栈指针等.

4.2.3 基于页表哈希的控制流保护

该框架中的分析模块是相对独立的.开发者可根据需求对收到的控制流数据进行不同种类的分析,从而给出相应的结果.分析的结果可以是系统的运行情况或相关数据,也可以是针对不同情况进行的不同控制或操作.作者给出了一个依据控制流数据进行控制流保护的分析模块的示例.

该分析模块使用白名单机制,它会提前将正常控制流可能的页表跳转的源和目标页通过哈希表的形式连接在一起.当客户操作系统由于页面错误进入虚拟机管理程序时,分析模块会分别对源和目标页进行哈希,随后,通过目标页的哈希找到对应项,再从对应项的链表中寻找对应的源哈希.如果存在对应源哈希,则说明该控制流是正常的,否则说明该控制流是非法的,虚拟机管理程序会返回异常,并中止该控制流,从而保护系统的安全运行.

4.3 XNPro

由于 rootkit 是针对操作系统内核的攻击,而且与操作系统内核拥有相同的权限,因此,在内核中直接抵御 rootkit 攻击比较困难.硬件辅助虚拟化技术可以让虚拟机管理程序在更高权限模式下对内核进行保护,因此,很多人都使用虚拟化技术抵御 rootkit 攻击.Nordholz 等人[58]提出了 XNPro,通过对内核代码执行的严格限制来保证操作系统免于受到代码注入攻击.

4.3.1 工作原理

在正常的操作系统中,在用户模式下运行的程序,无法执行内核内存中的代码;而在管理模式下运行的程序,可以执行用户内存中的代码.而在管理模式下运行时,程序并没有必要执行用户内存中的代码,这种权限只会让被攻击的内核跳转到用户内存中执行一段攻击者编写的代码,并导致系统被攻击.因此,XNPro 对于可执行的代码做了严格的限制,当程序在管理模式下运行时,只有内核中的必要部分(如.text 区域等)被允许执行.

当程序在用户模式下运行时,只有用户模式的内存被设置为可执行并且可写,而内核的内存空间被设置为不可访问.当程序在管理模式下运行时,则只有内核的.text 区域等必要部分被设置为可执行而不可写(防止被恶意篡改),而其他部分被设置为可写而不可执行.这种设计同样可以防止被攻击的内核跳转到内核中某处攻击者编写的代码中.

4.3.2 加载模块的验证

上述机制保护了内核中的固定组件.与此同时,内核中还有一些可加载内核模块.这些合法的内核模块在内核中加载后,同样需要可执行权限.XNPro 采用白名单方式管理这些可加载内核模块.XNPro 预先保存了这些内核模块的哈希值,当这些内核模块被加载时,操作系统会得到这些模块的加载位置,并将其发送给虚拟机管理程序.XNPro 收到消息后,计算对应位置模块的哈希值,并在白名单中寻找对应项.如果白名单中存在相同的哈希值,说明该模块是经过验证的合法模块;否则说明该模块是恶意模块或未验证的可疑模块,XNPro 便会阻止该模块的执行.

5 未来研究展望

随着 ARM 硬件虚拟化扩展的发展,基于硬件辅助虚拟化技术的安全防护技术得到业界的高度重视.由于虚拟机管理程序拥有比客户操作系统更高的权限,因此,多数研究致力于保护系统安全,保护内核完整性.然而,移动平台的安全需求不仅仅限于内核保护.由于移动平台在用户的生活中占有越来越重的地位,移动设备中也存储了更多的用户隐私.因此,硬件虚拟化扩展技术可以用于更多样的安全防护中.总的来说,对于基于 ARM 虚

拟化扩展的安全防护工作,其未来的研究方向可能主要包括以下几个方面.

(1) 系统安全

虽然基于硬件辅助虚拟化技术的系统安全已有不少研究成果,但系统安全依然是一个研究热点.对于虚拟机管理程序来说,语义鸿沟的问题是阻碍虚拟机管理程序保护系统安全的重要问题[59].虚拟机管理程序无法知道操作系统中的关键数据结构,只能通过读取虚拟机内存来分析操作系统状态.如果虚拟机管理程序可以克服语义鸿沟障碍,从而准确知道操作系统状态,那么虚拟机管理程序可以根据设计者的需求保护操作系统的各个方面.此外,虚拟机自省(virtual machine introspection,简称 VMI)技术也是一种有效的分析虚拟机内部信息从而保护系统安全的手段,该技术同样可以移植到 ARM 平台上,用于保护客户操作系统的完整性和安全性.

(2) 隐私保护

随着人们对移动设备的日益依赖,个人和企业的许多隐私信息大量存储在随身的手机、平板以及其他智能设备中.因此,移动设备中的隐私信息成为安全防护的重点.这些信息在移动设备中的输入、传输、保存与读取,均由系统内核以及相关驱动完成的.因此,必须保证这些环节的安全性才能保护用户的隐私.当前,已有许多关于移动平台隐私保护的研究成果[60−62],这些工具均运行在管理模式或用户模式中,一旦系统内核被攻击者攻击,这些保护均会失效.而硬件虚拟化技术可以将虚拟机管理程序架设在虚拟化模式中,该模式拥有高于系统管理模式的权限,同时,对于操作系统来说是完全透明的.因此,在这一层拦截相关的操作,同时对隐私数据进行加密解密,可以有效地保护个人隐私数据.我们的最新工作 H-Binder[63]就利用虚拟化管理程序拦截 Binder 通信数据并进行相关的检查验证,来保护用户 Binder 数据的安全性和完整性.

(3) 可信用户接口

移动设备是由人来进行操作的,这种操作需要用户通过相关的外设实现(如触摸屏、相关按键等).对于攻击者来说,远程控制的操作只能通过控制驱动来实现,外设无法得到响应.虚拟机管理程序可以通过是否存在硬件中断,从而分辨出某个操作是通过直接操作外设还是通过远程控制来实现的.同时,虚拟机管理程序也需要给用户提供一个交互接口,以便从用户处得到指令来决定实现哪种保护.在 x86 平台上,Cheng 等人[64]通过硬件虚拟化技术实现了安全的密码输入,这种技术可以移植到 ARM 平台上实现虚拟机管理程序与用户的交互.

(4) 与安全扩展的结合

在 ARMv7 架构中,同时也包含了安全扩展(security extension)[65].安全扩展又称为可信域(TrustZone),它是一种 ARM 上的硬件安全技术.目前,许多研究者已经针对 ARM 安全扩展研究移动平台的安全防护方法[66−70].ARM 安全扩展和 ARM 虚拟化扩展有不同的特点.ARM 安全扩展基于硬件进行数据保护,其安全性比虚拟化扩展更好.而虚拟化扩展对于异常的拦截更为全面和灵活,同时在模式切换时拥有更好的效率.如果将二者结合在一起,使用安全扩展保存相关的安全密钥以及加密、校验程序,同时利用虚拟化扩展运行虚拟机管理程序对实时的操作系统异常进行拦截处理,便可以同时利用二者的优点,建立一个更为安全高效的系统安全防护体系.

6 结 论

本文主要对当前基于 ARM 虚拟化扩展的安全防护技术的研究进展进行了归纳与综述.早期的 ARM 虚拟化研究由于缺少硬件虚拟化的支持,主要使用全虚拟化与半虚拟化技术.在 ARM 发布了硬件虚拟化扩展之后,人们的研究重心逐渐转向 ARM 硬件虚拟化.ARM 虚拟化扩展提供了新的虚拟化模式,当虚拟机管理程序在虚拟化模式中运行时,拥有比操作系统更高的权限,从而可以更好地管理客户操作系统.在硬件辅助虚拟化的研究中,主要存在 3 种框架:Xen,KVM 以及 OKL4.研究者们也基于硬件辅助虚拟化技术在系统安全的研究中取得了很大进展.最后,基于硬件辅助虚拟化技术的安全研究现状,本文对基于 ARM 虚拟化扩展的安全防护技术的未来发展方向进行了展望.本文认为:基于 ARM 硬件虚拟化,并结合 ARM 安全扩展,研究人员可以在系统安全、隐私保护及可信接口上进行更多有意义的尝试.

References:

[1] Smith B. ARM and Intel battle over the mobile chip's future. Computer, 2008,41(5):15−18. [doi: 10.1109/MC.2008.142]

[2] Aroca RV, Goncalev LMG. Towards green data centers: A comparison of x86 and ARM architectures power efficiency. Journal of Parallel & Distributed Computing, 2012,72(12):1770−1780. [doi: 10.1016/j.jpdc.2012.08.005]

[3] Ou Z, Pang B, Deng Y, Nurminen JK. Energy- and cost-efficiency analysis of ARM-based clusters. In: Proc. of the IEEE/ACM Int'l Symp. on Cluster, Cloud and Grid Computing. 2012. 115−123. [doi: 10.1109/CCGrid.2012.84]

[4] You DH, Noh BN. Android platform based Linux kernel rootkit. In: Proc. of the Int'l Conf. on Malicious & Unwanted Software. 2011. 79−87. [doi: 10.1109/MALWARE.2011.6112330]

[5] Li WX, Wang JB, Mu DJ, Yuan Y. Survey on android rootkit. Microprocessors, 2011,32(2):68−72 (in Chinese with English abstract). [doi: 10.3969/j.issn.1002-2279.2011.02.020]

[6] Mulliner C, Robertson W, Kirda E. VirtualSwndle: An automated attack against in-app billing on Android. In: Proc. of the ACM Asia Conf. on Computer and Communications Security. 2014. 459−470. [doi: 10.1145/2590296.2590335]

[7] Arzt S, Huber S, Rasthofer S, Bodden E. Denial-of-App attack: Inhibiting the installation of Android apps on stock phones. In: Proc. of the 4th ACM Workshop on Security and Privacy in Smartphones & Mobile Devices. 2014. 21−26. [doi: 10.1145/2666620.2666621]

[8] Spaulding J, Krauss A, Srinivasan A. Expoloring an open wifi detection vulnerability as a malware attack vector on iOS devices. In: Proc. of the Int'l Conf. on Malicious and Unwanted Software. 2012. 87−93. [doi: 10.1109/MALWARE.2012.6461013]

[9] Uhlig R, Neiger G, Rodgers D, Santoni AL, Martins FCM, Anderson AV, Bennett SM, Kägi A, Leung FH, Smith L. Intel virtualization technology. Computer, 2005,38(5):48−56. [doi: 10.1109/MC.2005.163]

[10] Chen X, Garfinkel T, Lewis EC, Subrahmanyam P, Waldspurger CA, Boneh D, Dwoskin J, Ports DRK. Overshadow: A virtualization-based approach to retrofitting protection in commodity operating systems. In: Proc. of the Int'l Conf. on Architectural Support for Programming Languages & Operating Systems. 2008. 2−13. [doi: 10.1145/1346281.1346284]

[11] Hofmann OS, Kim S, Dunn AM, Lee MZ, Witchel E. InkTag: Secure applications on an untrusted operating system. In: Proc. of the Int'l Conf. on Architectural Support for Programming Languages & Operating Systems. 2013. 253−264. [doi: 10.1145/2451116.2451146]

[12] Zhou ZW, Gligor VD, Newsome J, McCune JM. Building verifiable trusted path on commodity x86 computers. In: Proc. of the IEEE Symp. on Security and Privacy. 2012. 616−630. [doi: 10.1109/SP.2012.42]

[13] Cheng YQ, Ding XH, Deng RH. Efficient virtualization-based application protection against untrusted operating system. In: Proc. of the 10th ACM Symp. on Information, Computer and Communications Security. 2015. 345−356. [doi: 10.1145/2714576.2714618]

[14] ARM. Architecture Reference Manual (ARMv7-A and ARMv7-R edition). USA: ARM, 2012.

[15] Belady L. A study of replacement algorithms for virtual storage computers. IBM Systems Journal, 1966,5(2):78−101. [doi: 10.1147/sj.52.0078]

[16] Seawright LH, Mackinnon RA. VM/370—A study of multiplicity and usefulness. IBM Systems Journal, 1979,18(1):4−17. [doi: 10.1147/sj.181.0004]

[17] Creasy RJ. The origin of the VM/370 time-sharing system. IBM Journal of Research & Development, 1981,25(5):483−490. [doi: 10.1147/rd.255.0483]

[18] Gum PH. System/370 extended architecture: Facilities for virtual machines. IBM Journal of Research & Development, 1983,27(6):530−544. [doi: 10.1147/rd.276.0530]

[19] Garfinkel T, Ptaff B, Chow J, Rosenblum M, Boneh D. Terra: A virtual machine-based platform for trusted computing. ACM Sigops Operating Systems Review, 2003,37(5):193−206. [doi: 10.1145/1165389.945464]

[20] Huang JB, Ding Y, Fang F. Virtualization and cloud computing. In: Proc. of the Asia-Pacific Conf. on Information Network and Digital Content Security. 2011. 83−86 (in Chinese with English abstract).

[21] Zeng S, Hao Q. Network I/O path analysis in the kernel-based virtual machine environment through tracing. In: Proc. of the Int'l Conf. on Information Science and Engineering. 2009. 2658−2661. [doi: 10.1109/ICISE.2009.776]

[22] Wang J, Niphadkar S, Stavrou A, Ghosh AK. A virtualization architecture for in-depth kernel isolation. In: Proc. of the 43th Hawaii Int'l Conf. on System Sciences. 2010. 1−10. [doi: 10.1109/HICSS.2010.41]

[23] Whitaker A, Shaw M, Gribble SD. Scale and performance in the Denali isolation kernel. ACM SIGOPS Operating Systems Review, 2002,36(SI):195−209. [doi: 10.1145/844128.844147]

[24] Perez R, Doom LV, Sailer R. Virtualization and hardware-based security. IEEE Security & Privacy, 2008,6(5):24−31. [doi: 10.1109/MSP.2008.135]

[25] Wisniewski RW, Inglett T, Keppel P, Murty R, Riesen R. mOS: An architecture for extreme-scale operating systems. In: Proc. of the 4th Int'l Workshop on Runtime and Operating Systems for Supercomputers. 2014. 1−8. [doi: 10.1145/2612262.2612263]

[26] Yu KL, Chen Y, Mao JJ, Zhang L. ARM-MuxOS: A system architecure to support multiple operating systems on single mobile device. Computer Science, 2014,41(10):7−11 (in Chinese with English abstract). [doi: 10.11896/j.issn.1002-137X.2014.10.002]

[27] Nanda S, Chiueh TC. A survey on virtualization technologies. RPE Report, 2005. 1−42.

[28] Smith JE, Nair R. The architecture of virtual machines. Computer, 2005,38(5):32−38. [doi: 10.1109/MC.2005.173]

[29] Rosenblum M, Garfinkel T. Virtual machine monitors: Current technology and future trends. Computer, 2005,38(5):39−47. [doi: 10.1109/MC.2005.176]

[30] Popek GJ, Goldberg RP. Formal requirements for virtualizable third generation architectures. Communications of the ACM, 1974, 17(7):412−421. [doi: 10.1145/361011.361073]

[31] Sites RL, Chernoff A, Kirk MB, Marks MP, Robinson SG. Binary translation. Communications of the ACM, 1993,36(2):69−81. [doi: 10.1145/151220.151227]

[32] Suzuki A, Oikawa S. Implementing a simple trap and emulate VMM for the ARM architecture. In: Proc. of the IEEE Int'l Conf. on Embedded and Real-Time Computing Systems and Applications. 2011. 371−379. [doi: 10.1109/RTCSA.2011.26]

[33] Bellard F. QEMU, a fast and portable dynamic translator. In: Proc. of the Freenix Track: 2005 Usenix Technical Conf. 2005. 41−46.

[34] Bartholomew D. QEMU: A multihost multitarget emulator. Linux Journal, 2006,2006(145):68−71.

[35] Oh SC, Kim KH, Koh KW, Ahn CW. ViMo (virtualization for mobile): A virtual machine monitor supporting full virtualization for ARM mobile systems. In: Proc. of the 1st Int'l Conf. on Cloud Computing, GRIDs, and Virtualization. 2010. 48−53.

[36] Hwang JY, Suh SB, Heo SK, Park CJ, Ryu JM, Park SY, Kim CR. Xen on ARM: System virtualization using Xen hypervisor for ARM-based secure mobile phones. In: Proc. of the 5th IEEE Conf. on Consumer Communications and Networking Conf. 2008. 257−261. [doi: 10.1109/ccnc08.2007.64]

[37] Dall C, Nieh J. KVM for ARM. In: Proc. of the Ottawa Linux Symp. 2010. 45−56.

[38] Heiser G, Leslie B. The OKL4 microvisor: Convergence point of microkernels and hypervisors. In: Proc. of the ACM SIGCOMM Asia-Pacific Workshop on Systems (APSYS 2010). 2010. 19−24. [doi: 10.1145/1851276.1851282]

[39] Lee SM, Suh SB, Jeong B, Mo S. A multi-layer mandatory access control mechanism for mobile devices based on virtualization. In: Proc. of the Consumer Communications and Networking Conf. 2008. 251−256. [doi: 10.1109/ccnc08.2007.63]

[40] Park M, Yoo SH, Yoo C. Real-Time operating system virtualization for xen-arm. In: Proc. of the 4th Int'l Symp. on Embedded Technology. 2009. 1−2.

[41] Liu CL, Layland JW. Scheduling algorithms for muliprogramming in a hard-real-time environment. Journal of the ACM, 2002, 20(1):46−61. [doi: 10.1145/321738.321743]

[42] Li DJ. Multi-Platform extension of lightweight virtual machines [MS. Thesis]. Wuhan: Huazhong University of Science and Technology, 2011 (in Chinese with English abstract). [doi: 10.7666/d.d188307]

[43] Rossier D. EmbeddedXEN: A revisited architecture of the Xen hypervisor to support ARM-based embedded virtualization. White Paper, 2012.

[44] Zhong MZ. Study of embedded system security assurance based on virtualization technology [MS. Thesis]. Tianjin: Nankai University, 2013 (in Chinese with English abstract).

[45] Yang YJ, Gao YW. Study of direct access mapping of image files in Xen virtualized environment. Chines High Technology Letters, 2012,22(5):483−489 (in Chinese with English abstract). [doi: 10.3772/j.issn.1002-0470.2012.05.006]

[46] Zhao YH. Study of Linux kernel virtual machine technology implemented on ARM platform [MS. Thesis]. Wuhan: Huazhong University of Science and Technology, 2011 (in Chinese with English abstract). [doi: 10.7666/d.d187115]

[47] Ding JH, Lin CJ, Chang PH, Tsang CH, Hsu WC, Chung YC. ARMvisor: System virtualization for ARM. In: Proc. of the Ottawa Linux Symp. 2012. 95−109.

[48] Dall C, Nieh J. KVM/ARM: The design and implementation of the Linux ARM hypervisor. In: Proc. of the 19th Int'l Conf. on Architectural Support for Programming Languages and Operating Systems. 2014. 333−348. [doi: 10.1145/2541940.2541946]

[49] Barham P, Dragovic B, Fraser K, Hand S, Harris T, Ho A, Neugebauer R, Pratt I, Warfield A. Xen and the art of virtualization. ACM Sigops Operating System Review, 2003,37(5):164−177. [doi: 10.1145/1165389.945462]

[50] Lengyel TK, Kittel T, Pfoh J, Eckert C. Multi-Tiered security architecture for ARM via the virtualization and security extensions. In: Proc. of the Security in Highly Connected IT Systems. 2014. 308−312. [doi: 10.1109/DEXA.2014.68]

[51] Kivity A, Kamay Y, Laor D, Lublin U, Liguori A. KVM: The Linux virtual machine monitor. In: Proc. of the Linux Symp. 2007. 225−230.

[52] Russel R. Virtio: Towards a de-facto standard for virtual I/O devices. ACM SIGOPS Operating System Review, 2008,42(5): 95−103. [doi: 10.1145/1400097.1400108]

[53] Paolino M, Rigo A, Spyridakis A, Fanguède J, Lalov P, Raho D. T-KVM: A trusted architecture for KVM ARM v7 and v8 virtual machines securing virtual machines by means of KVM, TrustZone, TEE and SELinux. In: Proc. of the 6th Int'l Conf. on Cloud Computing, GRIDs, and Virtualization. 2015. 39−45.

[54] Liedtke J. On μ-kernel construction. In: Proc. of the 15th ACM Symp. on Operating System Principles. 1995. 237−250. [doi: 10.1145/224056.224075]

[55] Varanasi P. Implementing hardware-supported virtualization in OKL4 on ARM [Bachelor Thesis]. Sydney: University of New South Wales, 2010. [doi: 10.1145/2103799.2103813]

[56] Yang Y, Qian ZJ, Huang H. A lightweight monitor for Android kernel protection. Computer Engineering, 2014,40(4):48−52 (in Chinese with English abstract). [doi: 10.3969/j.issn.1000-3428.2014.04.009]

[57] Horsch J, Wessel S. Transparent page-based kernel and user space execution tracing from a custom minimal ARM hypervisor. In: Proc. of the IEEE Int'l Conf. on Trust, Security and Privacy in Computing and Communications. 2015. 408−417. [doi: 10.1109/Trustcom.2015.401]

[58] Nordholz J, Vetter J, Peter M, Junker-Petschick M, Danisevskis J. XNPro: Low-Impact hypervisor-based execution prevention on ARM. In: Proc. of the Int'l Workshop on Trustworthy Embedded Devices. 2015. 55−64. [doi: 10.1145/2808414.2808415]

[59] Chen PM, Noble BD. When virtual is better than real. In: Proc. of the 8th Workshop on Hot Topics in Operating Systems. 2001. 133−138.

[60] Enck W, Gilbert P, Han S, Tendulkar V, Chun BG, Cox LP, Jung J, Mcdaniel P, Sheth AN. TaintDroid: An information-flow tracking system for realtime privacy monitoring on smartphones. ACM Trans. on Computer Systems, 2014,32(2):5:1−5:29. [doi: 10.1145/2619091]

[61] Lin JH, Ren W, Jia LL. The design and implementation of a system for privacy protection in Android. Netinfo Security, 2013,(7): 16−19 (in Chinese with English abstract). [doi: 10.3969/j.issn.1671-1122.2013.07.004]

[62] Jia P, He X, Liu L, Gu B, Fang Y. A framework for privacy information protection on Android. In: Proc. of the Int'l Workshop on Sensor, Peer-to-peer and Social Networks. 2015. 1127−1131. [doi: 10.1109/ICCNC.2015.7069508]

[63] Shen D, Zhang ZK, Ding XH, Li ZJ, Deng RH. H-Binder: A hardened binder framework on android systems. In: Proc. of the 12th EAI Int'l Conf. on Security and Privacy in Communication Networks. 2016. 24−43.

[64] Cheng Y, Ding X. Virtualization based password protection against malware in untrusted operating systems. In: Proc. of the Int'l Conf. on Trust and Trustworthy Computing. 2012. 201−218. [doi: 10.1007/978-3-642-30921-2_12]

[65] ARM. ARM security technology—Building a secure system using trustzone technology. ARM Technical White Paper, 2009.

[66] Santos N, Raj H, Saroiu S, Wolman A. Using ARM trustzone to build a trusted language runtime for mobile applications. In: Proc. of the Int'l Conf. on Architectural Support for Programming Languages and Operating Systems. 2014. 67−80. [doi: 10.1145/2541940.2541949]

[67] Azab AM, Ning P, Shah J, Chen Q, Bhutkar R, Ganesh G, Ma J, Shen W. Hypervision across worlds: real-time kernel protection from the ARM trustzone secure world. In: Proc. of the ACM Sigsac Conf. on Computer and Communications Security. 2014. 90−102. [doi: 10.1145/2660267.2660350]

[68] Zhao S, Zhang Q, Hu G, Qin Y, Feng D. Providing root of trust for ARM trustzone using on-chip SRAM. In: Proc. of the 4th Int'l Workshop on Trustworthy Embedded Devices. 2014. 25−36. [doi: 10.1145/2666141.2666145]

[69] Pinto S, Oliveira D, Pereira J, Cardoso N, Ekpanyapong M, Cabral J, Tavares A. Towards a lightweight embedded virtualization architecture exploiting ARM TrustZone. In: Proc. of the IEEE Int'l Conf. on Emerging Technologies and Factory Automation. 2014. 1−4. [doi: 10.1109/ETFA.2014.7005255]

[70] Sun H, Sun K, Wang Y, Jing J. TrustOTP: Transforming smartphones into secure one-time password tokens. In: Proc. of the ACM Sigsac Conf. on Computer and Communications Security. 2015. 976−988. [doi: 10.1145/2810103.2813692]

附中文参考文献:

[5] 李文新,王姜博,慕德俊,袁源.Android 系统 Rootkit 技术综述.微处理机,2011,32(2):68−72. [doi: 10.3969/j.issn.1002-2279.2011.02.020]

[20] 黄建波,丁扬,方芳.虚拟化与云计算.见:亚太信息网络与数字安全会议论文集.2011.83−86.

[26] 余宽隆,陈瑜,茅俊杰,张磊.ARM-MuxOS:一台手机,多个世界.计算机科学,2014,41(10):7−11. [doi: 10.11896/j.issn.1002-137X.2014.10.002]

[42] 李大江.轻量级虚拟机的多平台扩展[硕士学位论文].武汉:华中科技大学,2011. [doi: 10.7666/d.d188307]

[44] 钟木忠.基于虚拟化技术的嵌入式系统安全保证研究[硕士学位论文].天津:南开大学,2013.

[45] 杨亚军,高云伟.Xen 虚拟化环境中镜像文件的访问直接映射研究.高技术通讯,2012,22(5):483−489. [doi: 10.3772/j.issn.1002-0470.2012.05.006]

[46] 赵亚辉.ARM 平台上实现 Linux 内核虚拟机技术研究[硕士学位论文].武汉:华中科技大学,2011. [doi: 10.7666/d.d187115]

[56] 杨永,钱振江,黄皓.一种轻量级的 Android 内核保护监控器.计算机工程,2014,40(4):48−52. [doi: 10.3969/j.issn.1000-3428.2014.04.009]

[61] 林佳华,任伟,贾磊雷.Android 手机隐私保护系统的设计与实现.信息网络安全,2013(7):16−19. [doi: 10.3969/j.issn.1671-1122.2013.07.004]

李舟军(1963－),男,湖南湘乡人,博士,教授,博士生导师,主要研究领域为网络与信息安全,数据挖掘,智能信息处理.

苏晓菁(1989－),女,助理研究员,主要研究领域为高可靠性器件与集成技术,先进计算光刻技术.

沈东(1989－),男,博士生,CCF 学生会员,主要研究领域为移动安全,虚拟化安全.

马金鑫(1986－),男,博士,助理研究员,主要研究领域为软件安全,程序分析.