最新的 TIP 包之前紧接着一个 FUP 包,则代表最新的跳转是由中断产生的.这时候,我们只检查被保护模块内的控制流,而不检查从模块中跳出的指令的合法性,因为这是一个中断的处理,我们不认为攻击者有能力在虚拟机中制造中断.

5 系统评测

我们的机器使用支持了 IPT 的 Intel Skylake 处理器,现在市面上可以方便地买到并且已被广为使用.我们已实现了本系统的原型.所有的测试在上述机器上运行,处理器为 Intel i7-6700K,操作系统为 Ubuntu 16.04,内核版本为 Linux 3.16.38.使用的虚拟机管理器为 Linux 所带 kvm 模块,使用的虚拟机为 Debian 8 操作系统,内核版本为 Linux 4.3.3.静态分析 CFG 的实现基于 Dyninst 框架开发.

5.1 安全性评测

本系统可以防御内核模块中的真实 ROP 攻击.为了验证这一点,我们手动编写了一个带有缓冲区溢出漏洞的内核模块,并编写了一个 ROP 攻击程序,利用内核模块的缓冲区溢出漏洞来覆写掉内核栈上的返回地址,劫持控制流,最终达成向某一特定文件内写入任意值的目的.开启了本系统的保护之后,在被保护的内核模块执行到返回指令时,读取的返回地址是被修改的内核核心内的地址,控制流检查器检查出这个地址和函数调用栈上存储的值不一样,将结果通知虚拟机管理器,虚拟机管理器暂停这个虚拟机的执行并报告给用户,等待后续处理.

本系统可以确保在 ROP 攻击生效前检测到攻击并中止虚拟机运行,保证攻击不会造成危害.在先前的研究中,有些攻击者会想办法避免触发控制流检查以隐藏攻击,然而这在本系统中是行不通的:假如攻击者想在被保护的模块内部寻找 gadgets,即用来在 ROP 攻击中执行的代码,修改一些寄存器和栈上的值,之后再返回到正常的返回地址,达到修改之后执行的函数参数的目的,进而实现一些攻击.然而在控制流返回到正常的返回地址时,由于它跳出了被保护的内核模块,所以会触发控制流检查,模块内部的 gadgets 会被发现,攻击不可能顺利完成.还有一种情况是:攻击者希望一直在模块内部进行跳转来修改一些关键内存或寄存器,使控制流不跳出模块,从而使攻击不被发现.然而,在模块内部能够做的事情很有限:即使攻击者修改了很多关键数据,如果控制流不从模块中跳出去,那么那些数据也不会被用到,攻击也不会造成实际效果;而且系统每隔一定时间会发出中断,在跳到中断处理函数的时候即跳出了被保护的模块,会触发控制流的检查.

还有一种可能的攻击是:攻击者通过内核模块的内存漏洞去修改属于内核核心部分的栈,而不修改内核模块部分的栈.这样一来,内核模块执行返回指令时,会返回到正确的内核核心中的地址,但是内核核心接下来会使用攻击者修改过的栈.然而,现在并没有这种攻击发表,并且也超出了本文的防止在内核模块内发生的 ROP 攻击的范畴.这种攻击可以通过换栈来防御:我们为内核模块准备一个单独的栈,在切换至内核模块执行时,扩展页表错误处理函数将虚拟机的栈指针映射到一个新的页上,和之前的内核栈区分开,防止攻击者修改内核栈;之后,在控制流返回内核核心时再将栈切换回内核栈,保证正常执行.

由于本系统的目标是防御内核模块中的 ROP 攻击,在内核模块通过间接函数调用来调用内核核心中的函数时,并没有进行检查.由于我们不依赖于内核核心的源码或二进制,我们无法对内核核心进行分析,也就无法得出内核模块可能会调用内核核心中哪些函数.现在被公布出的在内核中的控制流劫持攻击只有 ROP 攻击这 1 种,在内核模块中修改函数调用地址来进行控制流攻击的难度较大,不易实现,防御 ROP 攻击已经提供了极强的安全性.本系统可以防御所有内核模块中的 ROP 攻击,有力地阻止攻击者利用内核模块中的漏洞来攻击内核.

5.2 性能评测

我们对本系统进行了性能测试,用于多方面评测本系统所带来的额外开销.

5.2.1 Linux 命令的性能测试

我们在虚拟机中测试了 Linux 自带的一些命令,并记录其运行时间,之后再开启本系统的保护,重新测试相同的命令,将其结果比较,得到额外的性能开销.每个测试都被重复 10 次,并取平均值作为该条测试的结果.我们选择保护被监控虚拟机的 ext4 文件系统内核模块,并测试在保护 ext4 时运行各个基准测试的性能,并算出了造

成的额外性能开销的几何平均值.

我们选取了一些有代表性的 Linux 命令进行测试,使用默认配置运行这些命令,并使用 time 命令计算运行所用的时间.之后开启本系统保护,再次运行相同保护并记录所用的时间,并比较两次运行所用时间.我们选择的命令有:用于查看当前文件夹下所有文件的 ls、用于解压缩的 tar、用于文件操作的 dd 以及用于浏览网页的 curl.通过结果发现:对于频繁使用文件操作的 ls、tar、dd 命令,所造成的额外性能开销较大,而主要开销在网络传输的 curl 命令则开销很小.平均的额外性能开销为 38.75%,如图 3 所示.

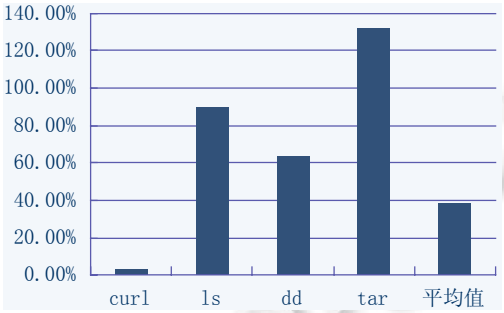


Fig.3 Overhead when running Linux utilities

图 3 运行 Linux 命令造成的额外性能开销

5.2.2 服务器程序的性能测试

因为本系统目标为保护客户的虚拟机,而服务器程序是攻击者最常使用的攻击载体,也是客户虚拟机上运行较多的程序,所以本系统的一个主要应用场景是在运行着服务器程序的客户虚拟机上实施保护.

为了测试本系统会造成的最大的性能开销,我们选取的被保护的内核模块是服务器程序中利用最频繁的网卡驱动模块,并测试在保护它时运行各个服务器程序时的性能开销.我们在 Web 服务器 nginx-1.6.3、FTP 服务器 vsftpd-3.0.3、SSH 服务器 OpenSSH-7.1p1 上进行了测试:在 nginx 上,我们使用 Apache 基准测试(apache benchmarks,简称 ab)来模拟 10 个并发的客户同时产生 20K 个请求,其中,每个请求都请求一个大小为 20B 的文件;在 FTP 服务器上,我们使用 pyftpbench 来下载 10MB 大小的文件;在 OpenSSH 上,我们使用了登陆并执行一般命令的脚本来进行测试.每种测试同样进行了 10 次并取平均值.图 4 展示了开启本系统时运行服务器程序造成的额外性能开销.如图可以看出,造成的平均额外性能开销为 61%.

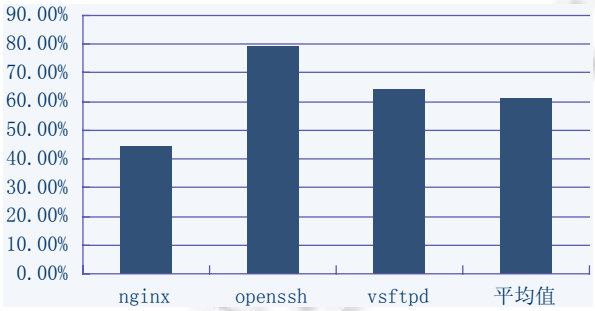


Fig.4 Overhead when running server applications

图 4 运行服务器程序造成的额外性能开销

6 讨 论

6.1 对返回指令的保护

本系统目前只支持对返回指令的保护,只防御通过篡改返回地址来劫持控制流的 ROP 攻击,不能防御所有

的控制流劫持攻击.这是因为本系统设计的目标是不依赖内核核心的源代码和二进制,无法对内核核心进行分析,所以本系统无法获得较精确的内核中的合法跳转目标地址集合.现在流行的控制流劫持攻击大多数都是 ROP 攻击,并且现有的公布出来的针对操作系统内核的控制流攻击全部为 ROP 攻击.因此我们认为:本系统基本能够防御针对内核的控制流劫持攻击,已提供足够强的安全性.除本系统之外,也有一些保护 CFI 的系统只保护返回指令,不保护间接函数调用指令.如,PT-CFI[33]就是一个这样的系统.文献[33]中提到,目前还没有一个完美的保护间接函数调用指令的机制,因为各种分析方法都或多或少存在漏洞,无法准确地分析出每一条间接函数调用指令的合法目标地址.

除操作系统内核核心外,内核的地址空间还包括除了被保护的内核模块以外的其他内核模块.我们已经做了一些工作来分析其他内核模块.经过分析发现:我们可以通过被保护的内核模块的信息获得它所依赖的内核模块和依赖它的内核模块,然后获得这些内核模块中合法的跳转目标地址.被保护的内核模块的间接函数调用只能调用如下几个地方:(1) 该内核模块内部;(2) 某个该内核模块所依赖的内核模块内部;(3) 某个依赖该内核模块的内核模块内部;(4) 内核核心中.情况(1)的情况已经通过静态分析生成的 CFG 解决,而在情况(2)和情况(3)中的合法跳转目标地址我们可以也通过静态分析得到.所以,如果被保护的内核模块的一条间接函数调用指令的目标的地址为内核模块的地址范围,而非内核核心的地址范围,就可以确保它是合法的.然而,即使我们保护了调用其他内核模块中的函数调用的合法性,攻击者仍然可以直接调用内核核心中的函数来实行攻击.我们尝试对内核核心的二进制做了一些分析,但是由于内核本身过于庞大,而且其中有很多非常规的函数调用方式,仅仅使用基于二进制的分析无法很好地限制非法的函数调用.所以,完全的基于二进制来保护内核的控制流完整性是一项比较艰巨的工作.

6.2 额外性能开销

如第 5.2 节所述,本系统目前造成的额外性能开销相对较大,经测试:在开启本系统后,对 Linux 基本命令造成的额外性能开销约为 40%,而对服务器程序造成的最大额外性能开销约为 60%.这主要有两个原因:第一,我们在进行性能测试时,选取的是会有较高负载的内核模块,如果我们运行的程序和被保护的内核模块关系不大,如在保护 ext4 文件系统模块时运行 curl 来浏览网页,则只会造成 2.99%的额外性能开销;第二,本系统对返回指令做了最精细的保护,保证本系统可以防御所有的 ROP 攻击,所以需要较频繁地进行控制流检查,来保证虚拟机的安全性.同时,我们也做到了极好的透明性和兼容性:我们不需要内核核心的源代码和二进制,使得本系统可以用于几乎所有的操作系统;我们也不需要对内核核心和内核模块的二进制进行修改,使本系统可以用于几乎所有的虚拟化云服务场景.我们用相对较大的额外性能开销换来了极强的安全性、兼容性和透明性,我们认为这是值得的.

对于本系统目前性能开销相对较大的问题,我们也拟出了一些优化方案.如,在每次控制流跳出内核模块的时候,在虚拟机内部检查这个跳转的源地址和目标地址对是否在之前已经遇到过.如遇到过,则直接认为这个跳转是合法的,不触发虚拟机下陷从而提高效率.但是这种做法必须向虚拟机中加入新的内核模块,无法维持我们提供的保护的透明性.再如,我们可以将一些被经常调用的函数(如中断处理函数)放在保护范围之外,即调用它们的时候我们不会触发扩展页表错误,减少虚拟机下陷.但是这样的做法给了攻击者将控制流跳转指令重定向到这些函数内部并实施攻击的机会,会影响系统的安全性.可以看出,目前对本系统尝试做的性能优化的方案都会在一定程度上牺牲本系统提供的保护精确性、透明性、兼容性.

7 总 结

本文针对目前流行的 ROP 攻击的问题,以及很少有保护操作系统内核不受 ROP 攻击的系统的现状,提出了一套基于硬件的使用虚拟化手段来保护针对内核模块的 ROP 攻击的系统.本系统分为内核模块静态分析、控制流记录、动态检查几个模块.利用 IPT 高效记录的特点,本系统可以准确地捕捉被保护虚拟机的控制流.通过修改扩展页表错误处理函数,在控制流跳出内核模块的时候触发一个扩展页表错误,在扩展页表错误处理函数中触发控制流检查,将记录的控制流与静态分析生成并重构的 ITC-CFG 进行比对,可以精确地确保被保护虚拟

机的返回指令是合法的,保证虚拟机不受 ROP 攻击.测试结果表明:本系统可以精确地检测到 ROP 攻击,并不会影响正常状态下虚拟机的运行,提供了极强的安全性、兼容性、高效性.

References:

[1] Abadi M, Budiu M, Erlingsson Ú, Ligatti J. Control-Flow integrity. In: Proc. of the 12th ACM Conf. on Computer and Communications Security (CCS 2005). New York: ACM Press, 2005. 340−353. [doi: 10.1145/1102120.1102165]

[2] Seshadri A, Luk M, Qu N, Perrig A. SecVisor: A tiny hypervisor to provide lifetime kernel code integrity for commodity OSes. ACM SIGOPS Operating Systems Review, 2007,41(6):335−350. [doi: 10.1145/1323293.1294294]

[3] Hund R, Holz T, Freiling FC. Return-Oriented rootkits: Bypassing kernel code integrity protection mechanisms. In: Proc. of the 18th Conf. on USENIX Security Symp. (SSYM 2009). Berkeley: USENIX Association, 2009. 383−398.

[4] Criswell J, Dautenhahn N, Adve V. KCoFI: Complete control-flow integrity for commodity operating system kernels. In: Proc. of the 2014 IEEE Symp. on Security and Privacy (SP 2014). Washington: IEEE Computer Society, 2014. 292−307. [doi: 10.1109/SP.2014.26]

[5] Jr. Petroni NL, Hicks M. Automated detection of persistent kernel control-flow attacks. In: Proc. of the 14th ACM Conf.on Computer and Communications Security (CCS 2007). New York: ACM Press, 2007. 103−115. [doi: 10.1145/1315245.1315260]

[6] Ge XY, Talele N, Payer M, Jaeger T. Fine-Grained control-flow integrity for kernel software. In: Proc. of the 2016 IEEE European Symp. on Security and Privacy (EuroS&P). IEEE, 2016. 179−194. [doi: 10.1109/EuroSP.2016.24]

[7] João M, Rigo S. Go With the FLOW: Fine-Grained Control-Flow Integrity for the Kernel. In: Proc. of the Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais. 2016.

[8] Li X, Huang H. Approach of kernel integrity monitoring using hardware virtualization. Computer Science, 2011,38(12):68−72 (in Chinese with English abstract). [doi: 10.3969/j.issn.1002-137X.2011.12.015]

[9] Liu Y, Shi P, Wang X, Chen H, Zang B, Guan H. Transparent and efficient cfi enforcement with Intel processor trace. In: Proc. of the 2017 IEEE Int'l Symp. on High Performance Computer Architecture (HPCA). IEEE, 2017. 529−540. [doi: 10.1109/HPCA.2017.18]

[10] Niu B, Tan G. Modular control-flow integrity. In: Proc. of the 35th ACM SIGPLAN Conf. on Programming Language Design and Implementation (PLDI 2014). New York: ACM Press, 2014. 577−587. [doi: 10.1145/2594291.2594295]

[11] Tice C, Roeder T, Collingbourne P, Checkoway S, Erlingsson Ú, Lozano L, Pike G. Enforcing forward-edge control-flow integrity in GCC & LLVM. In: Proc. of the 23rd USENIX Conf. on Security Symp. (SEC 2014). Berkeley: USENIX Association, 2014. 941−955.

[12] Mashtizadeh AJ, Bittau A, Boneh D, Mazières D. CCFI: Cryptographically enforced control flow integrity. In: Proc. of the 22nd ACM SIGSAC Conf. on Computer and Communications Security (CCS 2015). New York: ACM Press, 2015. 941−951. [doi: 10.1145/2810103.2813676]

[13] Arthur W, Mehne B, Das R, Austin T. Getting in control of your control flow with control-data isolation. In: Proc. of the 2015 IEEE/ACM Int'l Symp. on Code Generation and Optimization (CGO). 2015. 79−90. [doi: 10.1109/CGO.2015.7054189]

[14] Wang MH, Yin H, Bhaskar AV, Su PR, Feng DG. Binary code continent: Finer-Grained control flow integrity for stripped binaries. Journal of Cyber Security, 2016,1(2):61−72 (in Chinese with English abstract).

[15] Zhang MW, Sekar R. Control flow integrity for COTS binaries. In: Proc. of the 22nd USENIX Conf. on Security (SEC 2013). Berkeley: USENIX Association, 2013. 337−352.

[16] Zhang C, Wei T, Chen ZF, Duan L, Szekeres L, McCamant S, Song D, Zou W. Practical control flow integrity and randomization for binary executables. In: Proc. of the 2013 IEEE Symp. on Security and Privacy (SP 2013). Washington: IEEE Computer Society, 2013. 559−573. [doi: 10.1109/SP.2013.44]

[17] Payer M, Barresi A, Gross TR. Fine-Grained control-flow integrity through binary hardening. In: Proc. of the Int'l Conf. on Detection of Intrusions and Malware, and Vulnerability Assessment. New York: Springer-Verlag, 2015. 144−164. [doi: 10.1007/978-3-319-20550-2_8]

[18] Qiao R, Zhang MW, Sekar R. A principled approach for ROP defense. In: Proc. of the 31st Annual Computer Security Applications Conf. (ACSAC 2015). New York: ACM Press, 2015. 101−110. [doi: 10.1145/2818000.2818021]

[19] Vishwath M, Larsen P, Brunthaler S, Hamlen KW, Franz M. Opaque control-flow integrity. In: Proc. of the NDSS, Vol.26. 2015. 27−30.

[20] van der Veen V, Göktas E, Contag M, Pawlowski A, Chen X, Rawat S, Bos H, Holz T, Athanasopoulos E, Giuffrida C. A tough call: Mitigating advanced code-reuse attacks at the binary level. In: Proc. of the 2016 IEEE Symp. on Security and Privacy (SP). 2016. 934−953.

[21] Kayaalp M, Schmitt T, Nomani J, Ponomarev D, Abu-Ghazaleh N. SCRAP: Architecture for signature-based protection from code reuse attacks. In: Proc. of the 2013 IEEE 19th Int'l Symp. on High Performance Computer Architecture (HPCA 2013). Washington: IEEE Computer Society, 2013. 258−269. [doi: 10.1109/HPCA.2013.6522324]

[22] Arnautov S, Fetzer C. Controlfreak: Signature chaining to counter control flow attacks. In: Proc. of the 2015 IEEE 34th Symp. on Reliable Distributed Systems (SRDS). 2015. 84−93.

[23] Vasudevan A, Qu N, Perrig A. XTRec: Secure real-time execution trace recording on commodity platforms. In: Proc. of the 2011 44th Hawaii Int'l Conf. on System Sciences (HICSS 2011). Washington: IEEE Computer Society, 2011. 1−10. [doi: 10.1109/HICSS.2011.500]

[24] Yuan LW, Xing WC, Chen HB, Zang BY. Security breaches as PMU deviation: Detecting and identifying security attacks using performance counters. In: Proc. of the 2nd Asia-Pacific Workshop on Systems (APSys 2011). New York: ACM Press, 2011. Article 6. [doi: 10.1145/2103799.2103807]

[25] Xia YB, Liu YT, Chen HB, Zang BY. CFIMon: Detecting violation of control flow integrity using performance counters. In: Proc. of the 42nd Annual IEEE/IFIP Int'l Conf. on Dependable Systems and Networks (DSN 2012). Washington: IEEE Computer Society, 2012. 1−12. [doi: 10.1109/DSN.2012.6263958]

[26] Pappas V, Polychronakis M, Keromytis AD. Transparent ROP exploit mitigation using indirect branch tracing. In: Proc. of the 22nd USENIX Conf. on Security (SEC 2013). Berkeley: USENIX Association, 2013. 447−462.

[27] Cheng YQ, Zhou ZW, Miao Y, Ding XH, Deng HJ. ROPecker: A generic and practical approach for defending against ROP attack. In: Proc. of the 21th Annual Network and Distributed System Security Symp. (NDSS 2014). San Diago, 2014. 1−14. [doi: 10.14722/ndss.2014.23156]

[28] van der Veen V, Andriesse D, Göktaş E, Gras B, Sambuc L, Slowinska A, Bos H, Giuffrida C. Practical context-sensitive CFI. In: Proc. of the 22nd ACM SIGSAC Conf. on Computer and Communications Security (CCS 2015). New York: ACM Press, 2015. 927−940. [doi: 10.1145/2810103.2813673]

[29] Wang Z, Jiang XX. HyperSafe: A lightweight approach to provide lifetime hypervisor control-flow integrity. In: Proc. of the 2010 IEEE Symp. on Security and Privacy (SP 2010). Washington: IEEE Computer Society, 2010. 380−395. [doi: 10.1109/SP.2010.30]

[30] Ge X, Talele N, Payer M, Jaeger T. Fine-Grained control-flow integrity for kernel software. In: Proc. of the 2016 IEEE European Symp. on Security and Privacy (EuroS&P). IEEE, 2016. 179−194. [doi: 10.1109/EuroSP.2016.24]

[31] Team P. Rap: Riprop. 2015. https://pax.grsecurity.net/docs/PaXTeam-H2HC15-RAP-RIP-ROP.pdf

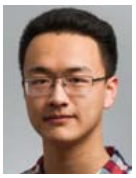
[32] Zhang MW, Qiao R, Hasabnis N, Sekar R. A platform for secure static binary instrumentation. In: Proc. of the 10th ACM SIGPLAN/SIGOPS Int'l Conf. on Virtual Execution Environments (VEE 2014). New York: ACM Press, 2014. 129−140. [doi: 10.1145/2576195.2576208]

[33] Gu YF, Zhao QC, Zhang YQ, Lin ZQ. PT-CFI: Transparent backward-edge control flow violation detection using Intel processor trace. In: Proc. of the 7th ACM on Conf. on Data and Application Security and Privacy (CODASPY 2017). New York: ACM Press, 2017. 173−184. [doi: 10.1145/3029806.3029830]

附中文参考文献:

[8] 李珣,黄皓.一个基于硬件虚拟化的内核完整性监控方法.计算机科学,2011,38(12):68−72. [doi: 10.3969/j.issn.1002-137X.2011.12.015]

[14] 王明华,尹恒,Bhaskar AV,苏璞睿,冯登国.二进制代码块:面向二进制程序的细粒度控制流完整性校验方法.信息安全学报,2016,1(2):61−72.



王心然(1995－),男,天津人,硕士生,主要研究领域为系统安全.

陈海波(1982－),男,博士,教授,博士生导师,CCF 杰出会员,主要研究领域为系统软件,系统结构.

刘宇涛(1989－),男,博士,主要研究领域为系统安全.