

嵌入式系统抗缓冲区溢出攻击的硬件防御机制研究

万毓西,刘政林,霍文捷,邹雪城

(华中科技大学 电子科学与技术系,湖北 武汉 430074)

摘 要: 嵌入式系统设计时由于成本和功耗等方面的考虑而较少重视安全性,而一般采用的软件防御方式无法满足嵌入式系统在实时性和可靠性上的要求,缓冲区溢出作为最常见的软件安全漏洞对嵌入式系统安全构成严重威胁.文中构建了一种基于细粒度指令流监控(FIFM)的硬件防御机制,通过虚拟执行单元虚拟执行程序,在攻击发生之前检测攻击行为.实验结果表明 FIFM 能很好的防御典型的缓冲区溢出攻击,而且 FIFM 不需要修改程序,不破坏流水线完整性,对系统的性能影响小,本文的防护机制可以应用于其他嵌入式系统设计中以动态防御缓冲区溢出攻击.

关键词: 缓冲区溢出攻击;虚拟执行单元;硬件防御

中图分类号: TN4

文献标识码: A

文章编号: 1000-7180(2011)04-0135-05

Study of a Hardware Defense Mechanism to Against Buffer Overflow Attacks for Embedded Systems

WAN Yu-xi, LIU Zheng-lin, HUO Wen-jie, ZOU Xue-cheng

(Department of Electronic Science & Technology,

Huazhong University of Science and Technology, Wuhan 430074, China)

Abstract: The embedded systems design pay less attention to safety out of consideration for cost and power consumption, and the software defense mechanism unable to achieve the requirement of the embedded systems in real-time and reliability, as the most common software vulnerabilities, buffer overflow poses a serious threat to embedded systems in safety. So we build a hardware defense mechanism based on fine granularity instruction flow monitor to against buffer overflow attacks, through virtual implement the procedures by ours virtual execution units, we can detect attacks before it come comply. The result shows that the hardware defense mechanism can defense these attacks completely, and neither change the source code or destroy the pipeline integrity, nor affects performance, our protection mechanism could be used to prevent run-time buffer overflow attacks in other embedded systems.

Key words: buffer overflow attacks; virtual execution units; hardware defense

1 引言

缓冲区溢出是最常见的软件安全漏洞之一^[1],美国国家标准和技术局^[2]的报告显示,自2001年来缓冲区溢出漏洞在所有漏洞中所占比例从未低于14%。随着嵌入式系统的快速发展,嵌入式系统的安全越来越重要,在嵌入式系统上运行的程序需满足

实时性和可靠性的要求,而通常的嵌入式系统在设计初期没有考虑安全性问题,一个简单的缓冲区溢出攻击就可获得系统处理的敏感数据,甚至能获取系统的使用权.另一方面,目前已有的防御方法多采用软件来侦测缓冲区溢出攻击.这些方法有的需要增加新的指令,有的需要改动编译器,这些都增加了程序的开销,影响系统的实时性,而且软件防御机制

收稿日期:2010-07-10;修回日期:2010-08-23

基金项目:国家自然科学基金项目(60973034);2007 新世纪优秀人才支持计划资助(NCET-07-0328);中央高校基本科研业务费资助(2010ZD033)

滞后于漏洞的发展,对系统的可靠性提出了严峻的挑战,这些都约束了软件防御方式在嵌入式系统中的应用。

文中从硬件的角度出发,研究缓冲区溢出攻击的防御机制,以 SPARC 架构的 LEON3 处理器为参考,针对嵌入式系统可能出现的缓冲区溢出攻击,提出了一种轻量级的硬件防御机制,从而改善嵌入式系统的安全性。

2 相关工作

为了防御缓冲区溢出攻击,前人已经进行了一些研究。文献[3]提出的 StackGuard 是通过编译器在返回地址附近保存一个 canary,当函数返回时检查这个 canary 是否正确,若检查结果有误则认为攻击者已经篡改了返回地址;文献[4]提出的 StackShield 是将返回地址压入一个特殊的安全堆栈;文献[5]采用的是类似于 StackShield 的防御技术,只是加强了对安全堆栈的保护力度;文献[6]对数据段进行重组,数组的存放位置与全局和静态变量物理隔离,使得数组的溢出不会篡改它们,以上这四种机制只能防御针对返回地址的攻击;文献[7]提出的 PointGuard 是一种指针保护的技术,当指针存储在内存之前,通过一个 key 对这个指针加密,在返回时解密,该机制虽然可以防御针对指针的攻击,却不能防御篡改返回地址的攻击;文献[8]提出的 HSAP 技术通过在流水线中增加了一个处理段来执行硬件边界检查,以及类似 PointGuard 的技术来保护指针,可以防御篡改返回地址和针对指针的攻击。

以上技术中文献[2-5]只是保护了返回地址,文献[7]只能防范特定的攻击,文献[8]执行了边界检查,但是检查粒度不够细,防御范围有限,这些方法都降低了安全防御性能。另外文献[3,4,7]通过更改编译器加强了程序的安全性,虽然不改动硬件,但是他们都增加了程序的复杂度;文献[5]不仅改动了编译器,在硬件上也做了较大改动;文献[6]的技术虽然不改动硬件和软件,但防御性能有限;文献[8]破坏了原流水线的旁路机制,影响了流水线的性能,同时需要软件开发者对变量重新布局,增加了软件开发者的设计难度,硬件和软件代价较高。

文中在处理器体系结构中构建一种细粒度指令流监控的硬件防御机制,在保持流水线完整性以及不更改应用程序的前提下,通过粗细两种粒度的边界检查,返回地址保护和检查跳转指令等方式增强了系统运行过程中的安全性。

3 攻击模型分析

一般的缓冲区溢出攻击模型^[8]如图 1 所示。攻击者利用 strcpy 使 fun 函数中的 buffer 溢出至 main 函数的栈,然后用攻击代码的入口地址篡改 fun 函数的返回地址,当 fun 函数返回时,实际返回的是攻击代码的入口地址,程序转而执行攻击代码。

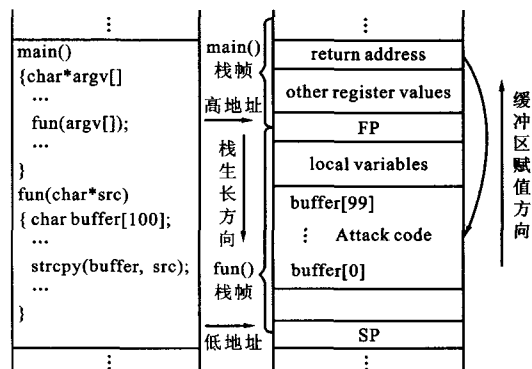


图 1 model 1

一个双溢出的攻击模型^[8]如图 2。指针 ptr 开始指向 buffer,第一个 strcpy 注入攻击代码,同时更改 ptr 使其指向共享函数指针偏移表(GOT),然后第二个 strcpy 用攻击代码的入口地址覆盖了 printf 函数在 GOT 中的入口地址,当程序调用 printf 时,实际上运行的是攻击代码。

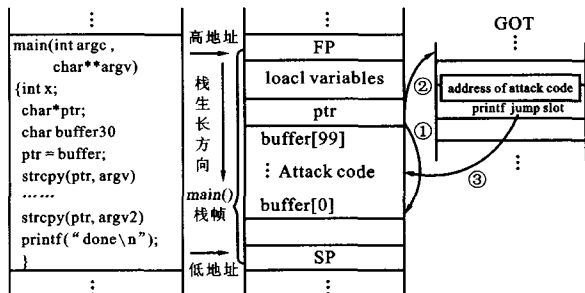


图 2 model 2

这两个攻击模型的关键是攻击者利用缓冲区漏洞使其溢出,然后精心设计缓冲区溢出后的行为,篡改返回地址或其他数据,迫使程序执行攻击代码。

4 监控机制

缓冲区溢出攻击的关键点在于迫使缓冲区溢出,因此若能够对缓冲区执行细粒度边界检查,同时对栈进行粗粒度边界检查、保护返回地址和检查跳转指令,则可对缓冲区溢出攻击进行全程监控。

利用 sparc-elf 编译工具,对攻击模型进行反汇编研究,分析在指令层次的攻击行为,构建指令级硬件监控单元。通过对缓冲区溢出攻击代码进行指令

级细化,分析完成攻击的指令流,提取关键指令,构造基于指令的攻击步骤,结合 c 语言层次的攻击行为分析,缓冲区溢出攻击可分为三个阶段:准备阶段,构造阶段和完成阶段. 监控单元在这三个阶段依次展开.

在准备阶段,攻击者寻找程序中的缓冲区漏洞,随后向这些存在漏洞的缓冲区写入超过其长度的内容,因此,监控单元需要在指令流中识别缓冲区. 利用 `sparc-elf` 工具分析缓冲区在指令层次的存在形式,提取相应特征,通过设计硬件识别模块,在指令流中获取缓冲区的边界信息:起始地址和长度,为对缓冲区执行细粒度边界检查提供原始监控数据.

在构造阶段,攻击者在写入缓冲区的过程中完成溢出动作,因此监控单元需要精确监控缓冲区的写入动作. 通过 `sparc-elf` 工具分析攻击模型中写缓冲区操作的指令流,精确定位写缓冲区的指令,监控单元计算这些写指令的目标地址,结合缓冲区的边界信息,对缓冲区执行细粒度边界检查,在缓冲区的溢出阶段检测到攻击行为. 另外,攻击者对缓冲区溢出常常演变为跨栈帧的溢出,其目的是篡改保存在前一个栈中的返回地址,这个溢出动作必然会跨越当前栈帧的边界,监控单元在指令流中保存当前栈帧的指针(SP),利用监控的写指令目标地址,实现对栈的粗粒度边界检查,防御跨栈帧的溢出攻击. 这两种粒度的边界检查可监控大多数缓冲区溢出攻击在构造阶段的行为.

在完成阶段,攻击者已经完成攻击动作,等待程序跳转执行攻击代码,而这一动作的实现,往往是通过前阶段已经篡改的返回地址或通过跳转指令来实现. 监控单元设计硬件堆栈来保存返回地址,当子程序返回时,对原系统的返回地址与保存的返回地址作比较,若两者有差异,则产生报警信号. 另一方面,攻击代码往往存放在栈中,监控跳转指令的目标地址,若发现目标地址在栈的地址域,就产生报警信号. 这两方面的监控可阻断大多数程序跳转到攻击代码的跳转路线.

以 SPARC 架构的 LEON3 处理器^[9,10]为平台,构建基于细粒度指令流监控机制(FIFM),对缓冲区溢出攻击在准备、构造和完成这三个阶段执行全程监控. FIFM 在 LEON3 处理器外构建监控单元. 其结构如图 3 所示. FIFM 位于流水线与 L1 Cache 之间.

在 FIFM 中有一个虚拟执行单元 VEU,VEU 是整个监控单元的控制核心,监控将在流水线中执

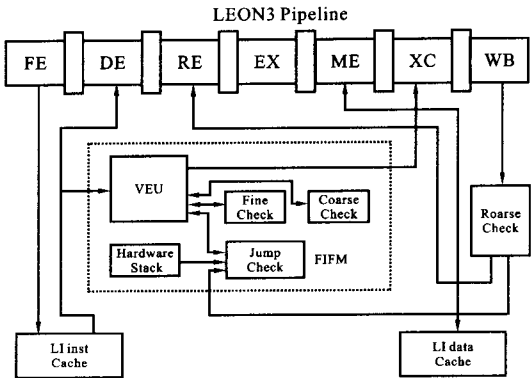


图 3 监控系统架构

行的指令,虚拟执行与攻击有关的指令:识别缓冲区边界信息以及写缓冲区指令的目标地址,把这些信息输入细粒度边界检查单元(fine check);把 SP 以及写缓冲区指令的目标地址输入粗粒度边界检查单元(coarse check);控制保存返回地址的硬件堆栈(hardware stack),把待分析的返回地址以及跳转指令目标地址输入跳转检查单元(jump check). 同时 VEU 接收并处理各模块的报警信号,随后把处理结果传递至流水线. 设立 VEU 是为保持流水线硬件完整性,且 FIFM 的处理时间与流水线处理时间重叠,不影响原系统的处理性能;FIFM 处理周期较短,可以在指令执行完之前获得监控结果,确保攻击代码不会破坏流水线.

以模型 1 来解释 FIFM 的工作原理. 如图 4 所示. VEU 根据监控的指令流,计算出缓冲区的边界地址,即 start address 和 end address, fine check 单元据此监控写指令的目标地址,当出现连续越界行为便置位报警信号;coarse check 接收 VEU 监控的栈帧的边界信息,即 sp 和 fp,同样当出现连续越界行为就产生报警信号;设立硬件堆栈保存返回地址,当程序使用原返回地址时进行对比;监控所有的跳转指令,禁止程序跳转到存储在栈中的攻击代码.

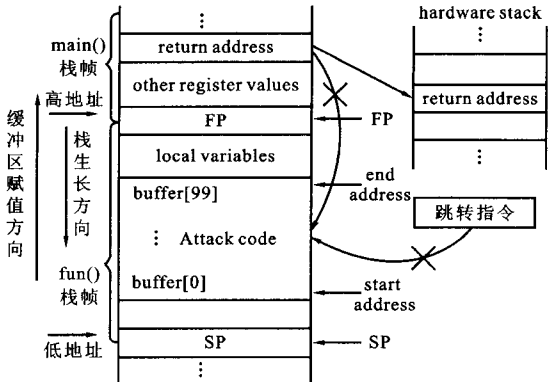


图 4 FIFM 监控示例

5 实验结果及安全性分析

将 LEON3 和相应的 FIFM 下载到 Xilinx 公司的 virtex5-xc5vfx70t FPGA 开发板上, FIFM 的实现开销如表 1 所示.

表 1 FIFM 面积和性能代价

参数	原 CPU 核	增加 FIFM	开销 %
硬件资源(slices)	3410	4044	18.5
时钟周期(ns)	8.678	8.695	0.1

由表 1 知, FIFM 带来了 18.5% 的硬件开销, 但是大约有 60% 的开销来源于 VEU 单元, 由前面的分析和实验结果可知, 设立的 VEU 单元, 可以在保持流水线完整性的基础上, 为子监控模块提供各种监控数据, 确保快速检测到攻击, 保证了流水线的安全, 而 FIFM 对原系统在时钟周期的开销只有 0.1%.

通过设置报警信号不中断程序的执行, 在原平台和实验平台上分别运行了 MiBench 中 5 个典型测试基准程序: sha 是在安全运算中使用的一种标准安全 hash 算法; bitcnts 是用于测试处理器的 bit 处理能力; stringsearch 是一种字符串搜索算法; bathmath 是用于检测嵌入式处理器中处理数学计算的能力; fft 是在数字信号处理中计算频率的快速傅里叶算法.

表 2 MiBench 测试统计结果

Mibench program	jmp	branch	ret	call/deep
sha	36	588	30	35/7
bitcnts	383	2756	307	311/13
stringsearch	190	18418	184	186/4
bathmath	4787	31921	4781	4833/9
fft	380	2371	374	400/30

表 2 是对这些程序中各种跳转指令个数的统计, call 指令用于函数调用, 在最后一列统计了这些程序的函数调用深度. 实验结果表明, 测试程序中存在大量跳转指令, 数量由 684 个到 46322 个, 而且 call 指令的个数代表了返回地址的个数, 可见 FIFM 需要处理大量的监控信息, 但测试程序在这两个平台的上运行时间几乎没有差别, 一方面由于对时钟周期只有 0.1% 的开销, 另一方面采用了指令级监控机制, 可完全识别指令并进行连续监控, 同时隐藏了监控处理时间, 所以不影响原系统的正常运行.

为了分析 FIFM 的安全性能, 分别把第 3 节中的两种攻击注入到实验平台, 检验 FIFM 抗攻击能力, 具体情况如表 3 所示.

表 3 各种机制防御能力统计

防御机制	模型 1	模型 2	快速检测
FIFM	✓	✓	✓
文献[3]	✓	×	×
文献[4]	✓	×	×
文献[5]	✓	×	×
文献[6]	✓	×	×
文献[7]	×	✓	×
文献[8]	✓	✓	✓

表 3 的第 2 列和第 3 列是每种机制的防御能力测试结果. FIFM 可以检测到这两种攻击模型; 对于模型 1, FIFM 的两个边界检查单元可检查攻击动作, 而且硬件堆栈可直接保护返回地址; 而在模型 2 中, fine check 单元可检查到溢出缓冲区的动作, 而且当程序被迫跳转到攻击代码时, 这个跳转指令可以被 jump check 单元捕捉到. 由表 3 可知, 除了文献[8]可检查到这两个攻击模型, 其他技术只能识别一种攻击模型. 表 3 的第 4 列表明只有 FIFM 和文献[8]可以实现快速检测.

虽然 FIFM 和文献[8]提出的 HSAP 技术都可以快速检测两种攻击模型, 但是和 HSAP 相比, FIFM 有如下优势: FIFM 设计了 VEU 单元, 不需要改动流水线架构, 隐藏了监控处理时间, FIFM 监控功能的实现不需要修改程序, 所以 FIFM 对硬件和软件影响较小, 而 HSAP 技术需要在流水线中增加一个处理段才能执行粗粒度边界检查, 不仅降低了防御能力, 破坏了流水线的旁路机制, 同时该技术还需要修改程序, 所以 HSAP 技术的硬件代价和软件开销都较高; 另外, HSAP 的检测单元设立在流水线的写回段之前, 而 FIFM 与流水线同步, 能比 HSAP 提前 3 个时钟周期检测到攻击, 避免了攻击代码破坏流水线.

因此 FIFM 不仅能防御这两种攻击模型, 而且不破坏流水线, 不修改程序, 对系统的性能影响小, 可以快速检测攻击, 保护了流水线的安全.

6 结束语

文中描述了一种针对缓冲区溢出攻击的硬件防御机制—FIFM, 并在以 LEON3 处理器搭建的实验

平台上实现了该机制,通过注入两种典型的缓冲区溢出攻击模型,检验了 FIFM 的抗攻击能力. 实验结果表明, FIFM 不仅能完全防御这两种攻击模型,而且还可以不对程序进行修改,不破坏流水线完整性,对原系统的性能影响小. 因此,文中设计的轻量级硬件防御机制,能有效改善嵌入式系统抗缓冲区溢出攻击的能力.

参考文献:

- [1] 付剑晶,黄金钟. 一种缓冲区溢出攻击的编译数据空间对策[J]. 微电子学与计算机,2008,25(6):196—199.
- [2] NIST, National Vulnerability Database [EB/OL]. (2010) [2010—07—10]. <http://web.nvd.nist.gov>. 2010.
- [3] Cowan C, Pu C, Maier D, et al. Stackguard: automatic adaptive detection and prevention of buffer—overflow attacks[C]// Proceedings of the USENIX Security Symposium. San Antonio, Texas: USENIX, 1998:3—5.
- [4] Vendicator, StackShield: A stack smashing technique protection tool for Linux[EB/OL]. (2000) [2010—07—10]. <http://www.angelfire.com/sk/stackshield/info.html>, 2000.
- [5] Eugen Leontie, Gedare Bloom, Olga Gelbart, Bhagirath Narahari and Rahul Simha. A compiler—hardware technique for protecting against buffer overflow attacks[J]. Journal of Information Assurance and Security, 2010 (5):1—8.
- [6] Yves Younan, Frank Piessens, Wouter Joosen. Protecting global and static variables from buffer overflow attacks[C]// Availability, Reliability and Security, 2009. Fukuoka; IEEE, 2009(3):798—803.
- [7] Cowan C, Beattie S, Johansen J, et al. Pointguard: protecting pointers from buffer overflow vulnerabilities [C]// Proceedings of the USENIX Security Symposium. Berkeley, CA, USA: USENIX, 2003:7—7.
- [8] Zili Shao, Edwin Sha. Defending embedded systems against buffer overflow via hardware/software[C]// Proceedings of 19th Annual Computer Security Applications Conference. USA:IEEE, 2003:352—361.
- [9] LEON3 SPARC Processor[EB/OL]. [2010—07—10]. <http://www.gaisler.com>.
- [10] 孔璐,吴武臣,侯立刚,等. 基于 LEON3 的 SOC 平台设计与 SPI 嵌入[J]. 微电子学与计算机,2010,27(3):173—176.

作者简介:

万毓西 男,(1985—),硕士研究生. 研究方向为嵌入式安全、数字 IC 设计.

刘政林 男,(1968—),教授. 研究方向为集成电路设计、无线传感器网络、嵌入式安全等.

霍文捷 男,(1982—),博士. 研究方向为计算机架构设计、安全硬件设计.

邹雪城 男,(1964—),教授. 研究方向为 RFID 系统和高性能计算、信息安全.