基于切片技术的远程缓冲区溢出攻击检测模型*)

郭 林[1,2] 严 芬[1,2,3] 蔡玮珺[1,2,4] 黄 皓[1,2]

(南京大学软件新技术国家重点实验室 南京 210093)[1] (南京大学计算机科学与技术系 南京 210093)[2]

(扬州大学信息工程学院计算机科学与工程系 扬州 225009)[3] (南昌陆军学院科文教研室 南昌 330103)[4]

摘 要 远程缓冲区溢出漏洞是网络安全领域危害最严重的安全漏洞,提高远程缓冲区溢出攻击防御能力成为安全研究的重要课题。本文提出了一种基于切片技术的远程缓冲区溢出攻击检测模型,给出了模型的架构思想和结构,以及各模块单元的实现技术和方法。最后,通过实验对模型的有效性进行了验证,并对各要素对模型性能的影响进行了客观的分析和评价。

关键词 缓冲区溢出,攻击,切片,指令缓冲,模型

A Model of Defense Remote Buffer Overflow Attack Based On Slice Technology

GUO Lin[1,2] YAN Fen[1,2,3] CAI Wei-Jun[1,2,4] HUANG Hao[1,2]

(State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093)[1]

(Department of Computer Science and Technology, Nanjing University, Nanjing 210093)[2]

(Department of Computer Science and Engineering, Technology Institute, Yangzhou University, Yangzhou 225009)[3]

(Science and Culture Teaching & Research Section, Nanchang Military Academy, Nanchang 330103)[4]

Abstract The remote buffer overflow vulnerability is the most serious security vulnerability in network security domain. How to enhance the defense abilty to remote buffer overflow attack becomes an important research subject. In this paper, the authors put forward a model of defense remote buffer overflow attack based on slice technology. The constructing idea and the structure of the model are given, and the realization technology and method of each module cell are described. In the last of this paper, the authors validate the validity of the model via experimentation. In the same time, give an impersonal analysis and remark to the influence of model performance caused by each factor.

Keywords Buffer overflow, Attacks, Slice, Instrusion cache, Model

1 前言

网络技术的飞速发展,促进了社会信息化和信息网络化。网络已经渗透到社会生活的每个领域,政府、经济、军事等各个社会领域的重要应用越来越依赖计算机网络。随着现代计算机系统复杂度的提高,带来了系统安全漏洞的增多。针对计算机网络的各种恶意攻击和渗透也越来越频繁,因此加强网络安全防御措施是网络安全研究领域面临的严峻课题。

根据近年来 Cert 的统计,由缓冲区溢出造成的安全漏洞占整个被发现安全漏洞的 50%~60%,并且被高性能网络蠕虫利用的几率越来越高。近年来,每一个系统安全漏洞的发现,都会造成一个或者几个网络蠕虫的传播和攻击事件的剧增(例如:Blaster, Slammer, CodeRed, Nimda 等)。所以,缓冲区溢出漏洞成为网络安全领域危害性最大的安全漏洞[1]。2003 年 1 月的 Slammer 蠕虫利用 MSSQL SERVER 的缓冲区溢出漏洞在短短的 10min 内感染了互联网中 90%(75000 台)存在该漏洞的主机。从上面的例子可以看出:利用缓冲区溢出攻击成功率非常高,缓冲区溢出成为黑客和蠕虫制造者最有效的攻击手段。因此,提高缓冲区溢出攻击的检测手段成为强化网络安全防御的一个关键问题。

基于对缓冲区溢出攻击及防御的研究,本文提出了基于切片技术的缓冲区溢出攻击的检测模型。本文第 2 部分介绍了缓冲区溢出攻击的原理和分类;第 3 部分介绍了当前在缓冲区溢出攻击检测方面的相关研究工作;第 4 部分主要阐述了基于切片技术的缓冲区溢出攻击的检测模型的思想、结构和实现方法;第 5 部分对此模型的实验结果和性能进行分析和评价;最后对模型的研究进行了总结和展望。

2 缓冲区溢出攻击

2.1 缓冲区溢出攻击的原理

缓冲区溢出攻击通过往程序的缓冲区写超出其长度的内容,造成缓冲区的溢出,从而破坏应用程序的堆栈,控制应用程序的流程去执行攻击者的恶意代码,以达到攻击的目的。

恶意代码又称为 shellcode,是由可执行机器码组成的字符串,对 exec(sh)之类的代码,利用反汇编后的结果结合操作系统的特点经反复调整而成[2]。本地用户造成溢出执行 shellcode 获得一个本地的 shell,而远程用户通过网络连接生成一个远程 shell。缓冲区溢出给了攻击者进行远程渗透的全部条件:在程序的地址空间里注入恶意的可执行代码、利用该程序的缓冲区溢出漏洞改变程序流程转而执行之。

*)国家 863 高科技发展计划"分布式网络监控与预警系统"(2003AA142010);江苏省高技术研究计划"计算机网络分布式主动防御、监控与预警技术研究"(BG2004030)。郭 林 硕士研究生,工程师,研究方向:网络与信息安全、恶意代码检测;严 芬 博士研究生,讲师,研究方向:网络与信息安全;蔡玮珺 硕士研究生,讲师,研究方向:网络与信息安全;黄 皓 博士,教授,博士生导师,研究方向:网络与信息安全。

2.2 ShellCode 结构分析

通常的 shellcode 按组成分为 3 种类型：NNSSRR 型、RRNNSS 型以及利用环境变量型[3]。N：表示 NOP，空指令。S：攻击行为的可执行代码。RR：应用程序执行流程的返回地址，通常指向 SS 的首地址。表 1 给出了这 3 种类型的 shellcode 的适用环境和构造方法。

表 1 3 种类型的 shellcode 的适用环境和构造方法

Shellcode 类型	适用类型	Shellcode 构造方法
NNSSRR 型	大缓冲区	地址：低——→高 结构：[buffer][EBP][EIP] 组成：[NNNNSS][S][R]
RRNNSS 型	大缓冲区 小缓冲区	地址：低——→高 结构：[buffer]EBP][EIP] 组成：[RRRR][R][R][NNNNSSS]
环境变量型	大缓冲区 小缓冲区	地址：低——→高 结构：[buffer][EBP][EIP] 组成：[AAAAA][A][R]

2.3 缓冲区溢出攻击分类

缓冲区溢出攻击根据攻击源可分为：远程缓冲区溢出攻击和本地缓冲区溢出攻击两类[4]。

①远程缓冲区溢出攻击：运行在主机系统的对外界开放服务程序，为外界用户提供服务（如 Web、ftp、Smtp 等等）。一旦服务程序存在缓冲区溢出漏洞，远程用户就可以通过合法的通信信道向主机系统发送含有恶意代码的数据，造成主机系统的缓冲区溢出，获取主机的控制权。在攻击过程中，网络和主机的安全措施（如：防火墙、IDS、反病毒网关等）对未公开的远程缓冲区溢出攻击的防御是非常困难的。因此，远程缓冲区溢出攻击是对网络安全影响较大的攻击手段。

②本地缓冲区溢出：主机的本地应用程序存在缓冲区溢出漏洞，攻击者对主机注入含有恶意代码的文件或者程序，诱使用户使用这些文件，造成应用程序的缓冲区溢出，获取主机控制权或者提升攻击者在主机上的权限。本地缓冲区溢出攻击在一定程度上是欺骗用户对非法数据文件的使用。对主机数据和文件进行合法性检查是防御本地缓冲区溢出攻击的重要手段。

3 相关研究工作

目前缓冲区溢出攻击实时检测方面的研究工作集中在 3 个方面：基于堆栈保护的检测技术、基于网络数据的检测技术、基于主机行为的检测技术。其中，具有代表性的检测方法有以下 3 种：

①基于 SRAS 技术的检测技术（Secure Return Address Stack）

StackGuard[5,6]是一种“用轻微的性能损失来消除堆栈溢出问题的编译器技术”[7]。它使用一种简单的编译器技术，通过扩展编译器以防止修改函数的返回地址来实现堆栈保护。函数调用时，在堆栈上函数的返回地址前放一个 canary。在函数退出时，检查 canary 的完整性。通过保证 canary 字的完整性来保证程序运行时避免缓冲区溢出漏洞的攻击。

StackShield[8]是一种“消除堆栈溢出问题的预编译技术”。它与 StackGuard 不同，StackShield 是作为一种预编译技术，在中间代码级别上，对堆栈中的返回地址作了保护，消除运行时的堆栈溢出问题。StackShield 创建一个额外的堆栈用来储存函数返回地址的拷贝，并在受保护的函数的入口和出口处各增加一段代码，来保证函数执行过程中，不会被恶意代码注入。

Stackguard 和 StackShield 是目前比较适用的基于编译器技术的缓冲区溢出攻击的实时检测方法。但是，缺点在于不能检测基于 Heap 的缓冲区溢出攻击，在特殊环境中攻击者能够绕过 StackGuard 的检测[9,10]。而且需要对应用程序进行二次编译，使用不方便。

②特征码匹配的检测技术

特征码匹配检测技术[11,12]在检测已知恶意代码的技术中，具有方法简单、速度快等优点，技术比较成熟。目前，商业化的缓冲区溢出攻击检测，大都采用特征码匹配检测技术。当出现新的缓冲区溢出漏洞攻击时，及时获取攻击样本，分析攻击代码要素，提取特征码，然后加入到特征库中去，利用特征库中的特征对缓冲区溢出攻击进行检测。但是特征码检测技术存在较大的弱点：1）它只能检测已知的恶意代码，对未知的、快速繁殖的缓冲区溢出攻击，不能检测；2）工作量大，要搜集和分析大量的缓冲区溢出攻击的特征码；3）恶意代码的多态性使特征码检测的效率大大降低。

③基于系统调用的检测技术

与基于编译器技术和特征码匹配不同，基于系统调用的检测技术[13]是一种异常检测的方法，通过监视 shellcode 中常用的系统服务函数和服务函数调用链对是否是缓冲区溢出攻击做出评估，来对可能的缓冲区溢出攻击进行检测和阻止。基于系统调用的检测技术在一定程度上可以检测和预防未知的缓冲区溢出攻击，但是它也存在缺点：1）误报率高，对缓冲区溢出攻击和正常的系统调用无法做出确切的划分；2）配置复杂，要提高基于系统调用检测的正确率，依赖于系统调用访问控制表的构造。构造系统调用访问控制表需要对系统服务和 shellcode 编写技术有较深的理解，并且随着各种 shellcode 的出现需要对系统调用访问控制表进行经常性的维护。

以上缓冲区溢出攻击的检测方法，在理论上都可以对缓冲区溢出攻击起到一定的防御作用，但是，在实用性和灵活性方面还存在一定的缺陷。通过对缓冲区溢出攻击原理和恶意代码的深入研究，本文运用切片匹配技术检测远程缓冲区溢出，防御远程缓冲区溢出攻击。

4 基于切片技术的远程缓冲区溢出攻击检测模型

基于切片技术的远程缓冲区溢出攻击检测模型（即下文中所指的模型）运用切片技术对主机服务程序的接收数据进行分割切片，计算各切片 HASH 并缓存。同时监视处理器指令 Cache 的执行指令序列并计算其 HASH 值。在两类 HASH 值之间进行匹配。防止远程应用数据被“注入”为程序指令。

4.1 现代处理器的 Cache 体系结构

随着处理器集成度和处理能力的不断提升，处理器的数据吞吐量也逐步加大。Cache 作为一种高效的数据交换解决方案在现代 CPU 中被大量使用。下面以 intel Pentium 系列 CPU 为例，介绍现代 CPU 的 Cache 结构[14,15]。

4.1.1 intel Pentium CPU 的 Cache 结构

Intel 从 Pentium 开始将 Cache 分开，通常分为一级高速缓存 L1 和二级高速缓存 L2。L1 Cache 是集成在 CPU 中的。在 L1 中还分数据 Cache（D-Cache）和指令 Cache（I-Cache）。

它们分别用来存放数据和执行这些数据的指令，而且两个Cache可以同时被CPU访问，减少了争用Cache所造成的冲突，提高了处理器效能。指令Cache直接和执行单元及动态跟踪引擎相连，通过动态跟踪引擎可以很快地找到所执行的指令，并且将指令的顺序存储在追踪缓存里，这样就减少了主执行循环的解码周期，提高了处理器的运算效率。L2 Cache只存储数据(数据和指令的混合体)，不分数据Cache和指令Cache。如图1所示。

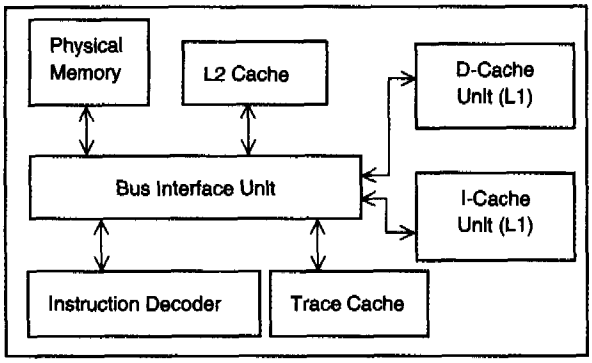

图1 Intel Pentium系列处理器Cache结构

4.1.2 读取命中率分析

CPU在Cache中找到有用的数据被称为命中，当Cache中没有CPU所需的数据时，CPU才访问内存，这种情况称为未命中[16]。对具有两级Cache处理器的数据读取命中率分析结果，如表2所示。

表2 两级Cache处理器的数据读取命中率表

Cache类型	首次命中率	本级命中率	总命中率
L1 Cache	80%		80%
L2 Cache	20%	80%	16%
Main Memory			4%

从表2可以看出，在一颗拥有2级Cache的CPU中，读取L1 Cache的命中率为80%。CPU从L1 Cache中找到的有用数据占数据总量的80%，剩下的20%从L2 Cache读取，同时读取L2的命中率也是80%，从L2读到有用的数据占总数据的16%左右，只有约4%的数据需要从内存中调用。

4.2 基于切片技术的远程缓冲区溢出攻击检测模型思想及结构

造成远程缓冲区溢出攻击的基本条件是服务程序在接收来自外界的用户数据时，发生数据处理错误，服务程序进程被劫持。攻击的最终结果是把含有恶意代码的用户数据“当作”服务进程的指令序列被执行，从而获取主机的控制权。可见，防御远程缓冲区溢出攻击的最根本、最有效的手段是阻止用户数据中的恶意代码的执行。模型通过对进入CPU的L1指令Cache的指令序列进行监视和阻拦，达到准确快速阻止恶意代码执行的目标，从而有效地阻断远程缓冲区溢出攻击。

4.2.1 模型结构

模型包括两个处理单元：HashKey计算单元和L1 I-Cache监视单元。

① HashKey计算单元

图2表示了HashKey计算单元逻辑结构。HashKey计算单元分配一定容量的初始内存空间，定义为HASH Buffer。HASH Buffer采用FIFO的队列存储，本文定义每一个节点为HashKey。Hash计算单元对主机服务程序接收到数据按设定切片大小进行切片，并计算每个数据切片的HashKey存储到Hash Buffer中。Hash计算单元维护Hash Buffer中HashKey的生存时间周期。缓冲区溢出攻击在通常情况下，存在着溢出即执行的特性，为了提高内存的消耗和检测效率，对缓存的数据切片设置了一定的生存周期，超出生存时间周期的切片HashKey将出队、销毁。

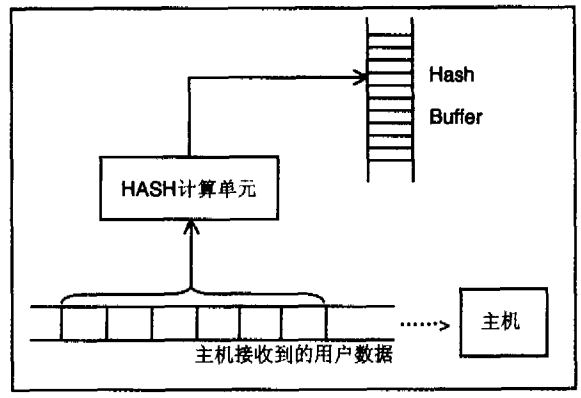

图2 HashKey计算单元逻辑结构图

② L1 I-Cache监视单元

图3表示了L1 I-Cache监视单元的逻辑结构。Hash计算控制单元连接到处理器控制通道PipeLine监视处理器指令I-Cache中指令序列的提交和更新[17]。I-Cache中缓冲指令序列提交完毕，Hash计算控制单元读取I-Cache中的缓冲指令序列计算其Hash值，与Hash Buffer中HashKey进行匹配。若匹配成功，则说明主机服务程序接收到远程数据进入处理器I-Cache并等待执行，这是危害主机安全的高危行为。Hash计算控制单元调用处理器的异常中断指令，投掷异常，阻断I-Cache中指令序列的执行，并销毁Hash Buffer队列中的相关HashKey节点。同时通过处理器控制通道保存程序执行上下文和处理器中断状态。

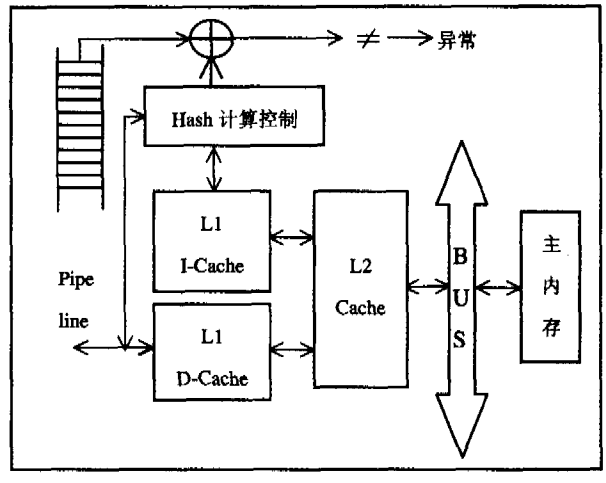

图3 L1 I-Cache监视单元逻辑结构图

4.2.2 HashKey的计算方法

主机服务程序接收到应用数据进行切片后，将取得的数据切片生成HashKey，需要选择一种实用高效的数据加密算法[18]。由于MD5加密算法的高速软件实现特性和输入报文长度任意等特点，因此在本模型中选取128bits的MD5加密算法。同时设定切片长度与MD5加密块长度相等。

HashKey 计算单元中的 Hash 计算单元将接收到应用数据包，按 128bits 进行切片。不足 128bits 的切片补零，对齐到 128bits。对每个切片数据块，使用 MD5 加密算法进行数据加密。对加密的加密数据块按 FIFO 规则缓存到 Hash Buffer 中。L1 I-Cache 监视单元中的计算控制单元使用相同的加密算法和加密过程对 I-Cache 指令序列进行加密。

5 模型仿真原型系统实验结果和性能分析

为验证模型的有效性，本文在 X86 平台下建立了模型的仿真实验环境，进行了仿真实验。

5.1 仿真实验环境

表 3 仿真原型系统实验环境

环境项目	实验环境参数
硬件平台	Intel X86 处理器，2.4G，内存 512MB
仿真软件	SimpleScalar[19,20]
仿真处理器	L1 I-Cache：16KBytes，4-way，32-byte cache line
	L1 D-Cache：16 KBytes，4-way，32-byte cache line
	L2 Cache：1 MBytes，4-way，32-byte cache line
	L1 延迟周期：1 时钟周期
	L2 延迟周期：10 时钟周期
攻击数据包	DARPA 1999 Training Data
网络环境	带宽：100M bps
	发包速率：10M bps

5.2 模型仿真原型系统实验结果

5.2.1 数据切片尺寸与模型检测准确率

设定切片生存周期为 3 秒，数据切片尺寸对模型检测准确率的影响性能曲线如图 4 所示。

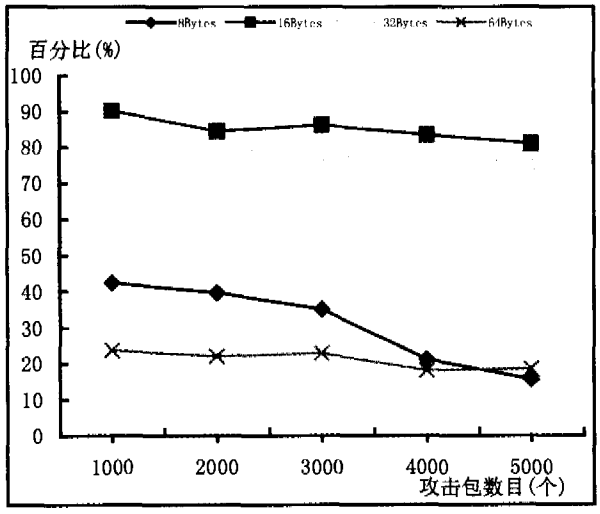

图 4 数据切片尺寸对模型检测准确率的影响

从实验数据可以得出如下结论：在切片生存周期固定的情况下，数据切片的大小对模型的检测准确率影响极大。从 Shellcode 的组成结构和特性分析，攻击代码在通常情况下都是相对简短且有多种组成方式。设定较长（32bytes，64bytes）的数据切片，攻击检出率产生非常大的下降，造成攻击检测正确率较低。设定较短（8bytes）的数据切片，与正常系统调用有大量重叠且随着数据切片的增加，模型误报率升高，同样造成系统攻击检测正确率低。所以，选择适当的数据切片长度，对模型检测性能非常重要。从实验攻击数据包和实验结果看出，8bytes 和 16bytes 的数据切片是比较适合的数据切片长度。

5.2.2 HashKey 生存时间周期与模型内存开销

为保证稳定、高效的模型检测性能，选择 16bytes 的数据切片长度。HashKey 生存时间周期与模型内存开销性能曲线如图 5 所示。

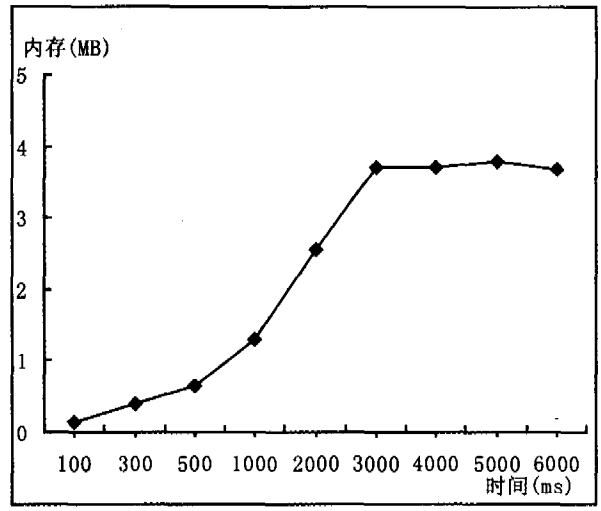

图 5 HashKey 生存时间周期对内存开销的影响

从图 4 可以看出：Hash Buffer 的内存开销在攻击数据包到达的初期，内存开销快速增长。随着 HashKey 的生存周期的到来，内存开销维持在一个相对稳定的状态。内存占用量的大小与攻击数据包速率相关。

5.2.3 Hash Buffer 与模型检测性能

从图 4 的实验数据得到如下结论，随着攻击数据包的增多，Hash Buffer 的容量逐渐增大，L1 I-Cache 监视单元中 HashKey 匹配时间增长，造成模型检测效率出现轻微下降。图 5 的实验数据表明，模型在切片生存周期的到来，Hash Buffer 的容量维持在一个相对稳定的状态。由此可见，Hash Buffer 的容量对模型检测性能的影响相对稳定。

总结与展望 本文阐述了基于切片技术的远程缓冲区溢出攻击检测模型的思想，并详细描述了模型的结构和实现方法。通过验证实验，从原理上证明本模型在防御远程缓冲区溢出攻击的有效性。但是，模型是借助处理器的指令缓冲来阻止注入恶意代码的执行，以达到防御远程缓冲区溢出的目标，因此模型存在处理器平台依赖性。在不同的 CPU 平台上实现本模型，需要根据处理器指令集对模型的各控制单元模块作较大的修改。

随着网络环境越来越复杂，网络应用日益多样化，模型的有效性还需要在复杂的网络环境和应用中进行验证。因此，基于切片技术的远程缓冲区溢出攻击检测模型要成为成熟的商业化模型还有待进一步的研究和完善。

参 考 文 献

1 Cowan C, Wagle P, Pu C, et al. Buffer Overflows : Attacks and Defenses for the Vulnerability of the Decade. http://www.cse.ogi.edu/DISC/projects/immunix/dis2cex00.pdf, 2000-01

2 刘绍翰，许建真，张福炎. 基于缓冲溢出漏洞的攻击及其预防研究综述. 计算机应用与软件，2004，121(1)

3 邵丹，唐世钢，林枫. Windows 平台下的缓冲区溢出漏洞分析. 信息技术，2003，27(2)

4 Backend. Windows 2000 缓冲区溢出入门. IsBaseMagzine，2000

5 StackGuard Development Team. StackGuard Mechanism: Stack Integrity Checking. http : / / www. cse. ogi. edu/ DISC/ projects/ immunix/ StackGuard/ mechanism. html, 1999

6 ÄOzdo~gano~glu H, Vijaykumar T N, Brodley C E, et al. SmashGuard: A Hardware Solution to Prevent Security Attacks on the Function Return Address. School of Electrical and Computer Engineering 465 Northwestern Ave, Purdue University, November 2003
7 Cowan C, Pu C, Maier D, et al. StackGuard: Automatic Adaptive Detection and Prevention of Buffer-Overflow Attacks. http://www.immunix.org/documentation.html.
8 Richarte G. Four different tricks to bypass StackShield and StackGuard protection. Core Security Technologies, April 9, 2002 - June 3, 2002
9 Bulba Kil3r. Bypassing StackGuard And StackShield, May, 2000. http://www.phrack.org/show.php? p=56&a=5 (April 9, 2002).
10 Richarte G. Four different tricks to bypass StackShield and StackGuard protection. Core Security Technologies, April 9, 2002
11 Bergeron J, Debbabi M, Desharnais J, et al. Detection of Malicious Code in COTS Software: A Short Survey. In: First International Software AssuranceCertification Conference (ISACC'99), Washington D C, Mar. 1999
12 Bergeron J, Debbabi M, Erhioui M M, et al. Static Analysis of Binary Code to Isolate Malicious Behaviors. In: Proccedingsof the 8th Workshop on Enabling Technologies on Infrastructure for Collaborative Enterprises, pages, 1999. 184~189
13 尚明磊,黄皓. 缓冲区溢出攻击的分析与实时检测. 计算机工程, 2005, 31 (12): 36~38
14 Intel. IA-32 Intel Architecture Software Developer's Manual Volume 2A: Instruction Set Reference, A-M. http://www.intel.com, 2004
15 AMD. AMD64 Architecture Programmer's Manual Volume 2: System Programming. http://www.amd.com, 2003
16 刘昌勇. CPU 原理分析. http://www.itdoor.net/pages/27, 21932,1,1076917028.html, 2004-02-16
17 Fiskiran A M, Lee R B. Runtime Execution Monitoring (REM) to Detect and PreventMalicious Code Execution. IEEE International Conference on Computer Design ICCD'04), 2004. 452~457
18 Gassend B, Suh G E, Clarke D, et al. Caches and Hash Trees for Efficient Memory Integrity Verification. In: Proc. Int Symposium on High-Performance Computer Architecture (HPCA), 2003. 295~306
19 Burger D, Austin T M. The SimpleScalar Tool Set, Version 2.0. Computer Architecture News, 1997. 13~25
20 Austin T, Larson E, Ernst D. Simplescalar- an infrastructure for computer system modeling. IEEE, February 2002. 59~67

(上接第 61 页)

使用 HC_D 将其恢复(过程中将使用到 *params* 中的解密 D): $\langle CT_8, \langle CT_X \rangle\rangle$。由于 $C_8 \in C_{CGCL}$, CGCL 使用 C_8 将继续看到 $\langle CT_9 \rangle$, 同样的道理, CGCL 使用 C_9 后将可看到访问请求 Re *quest*。

此时, CGCL 为 Tom 的此次访问分配资源 *Cluster*(如"以用户名 CGCL、密码 HP 从 8754 端口登录 192.168.1.240 集群"), 并将使用 Re *sponse* 返回给 Tom。同样, Tom 将使用 Re *sponse* 解密 $P_{Re\ sponse}(Cluster)$ 得到 $P_{Cluster}$; 在提供他的证书来满足便可恢复 *Cluster*。此时, 一个完整的信任协商便成功了。

4 HCBATN 特征分析

隐藏证书在信任协商中, 可带来许多优势, 如: (1)可解决基于 PKI 认证系统中的"谁先"的问题: 自动信任协商系统中, 协商双方都不愿意先暴露自己的策略; (2)访问控制策略不会让非授权的接收者所了解: 策略、资源信息都是密文, 窃听者无法解开, 或者说解密代价太大, 获得信息后已失去意义; (3)证书没固定的格式, 可随意创建和使用等。

除上述特点外, HCBATN 还具有如下特征:

[1]灵活的证书形式, 为证书的创建和使用带了便利。相比 X.509、SPKI/SDSI 等证书, 隐藏证书的使用更为简单、便捷;

[2]没有发现证书的网络开销。在 PKI 信任协商系统中, 证书链的发现, 带来了大量的网络传输开销。同时, 也影响了信任协商的效率和成功率。而在 HCBATN 中, 证书一般只限于一次协商会话, 是临时的, 由协商双方集中管理的, 没有发现证书的网络开销;

[3]减少管理证书的开销。在 HCBATN 中, 证书的创建和使用是任意的, 不需要像 PKI 一样使用 OLAP 来管理发布的证书和 CRL 来管理过期的证书;

[4]单轮回证书交换, 提高了协议的性能。在 HCBATN 中, 除最初一次名称的交换外, 整个过程证书的交换只有一个轮回, 这极大地提高了信任协商的协议性能, 提高了协商效率;

[5]具有更高的证书安全保密性。HCBATN 中, 信息都是以密文的形式进行的, 保证了传递信息、策略的安全性; 同时, 也有效地保护了用户隐私。

结束语 自动信任协商为资源共享、服务调用、商业交易等提供了跨域的安全保障, 是访问控制技术又一道防线。针对当前自动信任协商系统的不足, 本文引入了隐藏证书, 提供了一种基于隐藏证书的自动信任协商模型。HCBATN 对隐藏证书系统进行了改造, 使得其更接近于现实的应用。文中对 HCBATN 的组成与工作原理进行了详细的描述, 介绍了其操作规范, 并通过一个实际的例子来说明其 HCBATN 协议以及其工作流程, 总结了 HCBATN 的特征。

在下一步的工作中, 主要是将 HCBATN 应用到实际的系统中, 或在应用中体现 HCBATN 优势。针对中国教育网格支撑平台[9](ChinaGrid Supporting Platform, CGSP)的多域(一个大学可看作为一个域)、资源复杂、操作系统异构等特征, 将 HCBATN 引入到 CGSP 中, 为校园之间的资源共享与访问提供更高的安全保障。

参 考 文 献

1 Winsborough W H, Li Ninghui. Safety in Automated Trust Negotiation. In: Proceedings of the 2004 IEEE Symposium on Security and Privacy, IEEE Press, 2004. 123~135
2 Winsborough W H, Li N. Protecting sensitive attributes in automated trust negotiation. In: Proceeding of ACM Workshop on Privacy in the Electronic Society, ACM Press, 2002. 102~113
3 Winsborough W, Seamons K, Jones V. Automated Trust Negotiation. In: Proceeding of DARPA Information Survivability Conference and Exposition, ACM Press, 2000. 156~182
4 Holt J E, Bradshaw R, Seamons K E. et al. Hidden crdentials. In: Proceedings of 2nd ACM Workshop on Privacy in the Electronic Society, ACM Press, 2003. 1~8
5 Bradshaw R W, Holt J E, Seamons K E. Concealing Complex Policies with Hidden Credentials. In: Proceedings of the 4th ACM Conference on Computer and Communications Security, ACM Press, 2004. 245~253
6 Frikken K, Atallah M, Li Jiangtao. Hidden Access Control Policies with Hidden Credentials. In: Proceedings of the 3rd ACM Workshop on Privacy in the Electronic Society, ACM Press, 2004. 130~131
7 Boneh D, Franklin M. Identity based encryption from the Weil pairing. In: Proceedings of Crypto2001, Advances in Cryptology, Lecture Notes in Computer Science, Vol 2139, Springer-Verlag, 2001. 213~229
8 Gura N, Eberle H, Shantz S C. Generic implementations of elliptic curve cryptography using partial reduction. In: Proceedings of the 9th ACM Conference on Computer and Communications ecurity, ACM Press, 2002. 1771~89
9 中国教育科研网格公共支撑平台工作组. 中国教育科研网格公共支撑平台设计规范. 北京: 清华大学出版社, 2004