入侵诱骗系统中无缝环境切换技术的研究

夏春和 王海泉 胡恒一
(北京航空航天大学计算机学院 北京 100083)

〔摘要〕 网络攻击的目标可以归纳为:目标端的数据资源、计算资源及连接资源。为抵御以数据资源为目标的攻击,提出一种基于入侵诱骗的网络动态防御系统,探讨了入侵诱骗系统当中的重定向技术;分析其存在的问题,提出了一种无缝重定向解决方案,包括基于socket的重定向技术和环境状态同步技术;保证了入侵诱骗系统的主动性与安全性。
〔关键词〕 入侵诱骗 虚拟环境 无缝环境切换 环境状态 重定向 〔中图分类号〕 TP393.08
〔文献标识码〕 A 〔文章编号〕 1000-7857(2005)04-0016-04

RESEARCH ON SEAMLESS SWITCH ENVIRONMENT TECHNOLOGY OF INTRUSION DECEPTION

XIA Chun-he WANG Hai-quan HU Heng-yi
(School of Computer, Beijing University of Aeronautics & Astronautics, Beijing 100083, China)

Abstract: The targets of net attacks could be summed up to data resources, computational resources and connection resources. To resist net attack, which focuses on data resource, an intrusion-deception-based dynamic network defensive system was present in this paper. and redirection technology in the Intrusion Deception System were also discussed, The problems of redirection technology were analyzed and a seamless redirection solution was described, including the technology of redirection base on Socket and Environment States synchronization. The initiative and security of the Intrusion Deception System were assured.
Key Words: intrusion deception, seamless switch environment, environment states, redirection
CLC Number: TP393.08 **Document Code**: A **Article ID**: 1000-7857(2005)04-0016-04

1 引言

近年来随着计算机网络安全防御技术的研究与发展,静止的、平面的防御体系逐渐向动态的、多方位网络动态防御体系转变。寻找有效的动态安全防御方法,确保网络系统免受攻击,改变防御者的消极被动地位,是当今网络安全研究的新热点。已提出的PPDR (Policy /Protect/ Detect/ Response)与PDRR(Protect /Detect /React /Restore)等网络动态防御体系模型,强调网络系统在受到攻击的情况下,网络系统的稳定运行能力。PDRR与PPDR模型是重要的动态防御模型。

本文提出了基于入侵诱骗的网络动态防御体系,并探讨在入侵诱骗系统当中最为重要的部分——无缝环境切换技术。所谓无缝环境切换,就是将攻击者从真实环境中切换到诱骗环境中,而攻击者察觉不到自己已经被切换,从而攻击的目标由真实主机变为诱骗主机。本文通过对现有的入侵诱骗技术的研究,分析了现有的环境切换技术的优缺点,提出了自己的无缝环境切换技术。

2 入侵诱骗的概念及体系结构

入侵诱骗系统也被称为蜜罐(honeypot)。从一个广义的角度来讲,入侵诱骗系统可以设计成任意系统,设计它的目的是创建一个看起来容易被攻击的系统。更具体一点,入侵诱骗系统是一个分析、学习工具,是专门设计用来引诱入侵者的系统[1]。

建立一个容易受到攻击并且有着强大日志功能的系统,我们就可以实时地俘获对计算机的攻击。设想入侵诱骗系统就是一个监视器,在这个监视器里面攻击者能够被更仔细地监控和研究。研究和监视攻击者的攻击方法与技术,可以帮助我们判断和阻止未知的攻击。

入侵诱骗系统也能够减少真实系统被攻击的可能性。大多数的攻击者会入侵一个容易受到攻击的系统(入侵诱骗系统)而不是一个安全级别较高的系统,这样就可以潜在地减少对合法的真实系统的攻击。入侵诱骗系统的另外一个特点在于:任何尝试连接入侵诱骗系统的行为都被定义成未授权的行为。因为正常的流量访问没必要连接入侵诱骗系统,故任何在入侵诱骗系统的活动都能够被网络管理员认为是一个可能的攻击活动。

入侵诱骗系统[1]各部分的功能简述如下。(1)环境切换子系统:根据IDS检测结果进行操作环境的切换;(2)虚拟环境子系统:用来模拟操作系统、服务进程、协议、应用软件的漏洞、脆弱性;(3)信息收集子系统:收集虚拟环境系统事件,它

收稿日期:2005-03-01
基金项目:航空基金项目(03F51060),总装备部"十五"预研资助项目(418010703)
作者简介:夏春和,男,1965~;北京市海淀区学院路37号北京航空航天大学计算机学院,教授,北京市网络技术重点实验室主任,主要研究方向为网络安全与网络测量技术等;E-mail: xch@buaa.edu.cnb

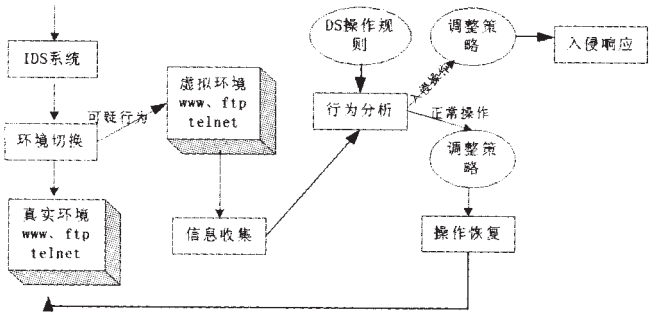

图 1 入侵诱骗系统的体系结构

们对入侵者的行为进行详细的记录,进行行为跟踪;(4)行为分析子系统:从行为操作序列的角度对可疑行为进行监视与分析,判断可疑行为的性质;(5)操作恢复子系统:保存行为操作命令序列及环境状态,对判断为正常的行为切换至真实环境中,恢复执行操作命令,给出正常的访问结果。

本文的工作重点是在分析现有的切换技术的基础上,提出自己的环境切换技术。

3 现有的环境切换技术及存在的问题

3.1 现有诱骗环境存在的问题

尽管入侵诱骗技术已经被证明有很大的发展前景,但其"守株待兔"式的防御方式,限制了入侵诱骗系统作用的进一步发挥。

很多用户在使用入侵诱骗系统时,比较关心的是诱骗环境的冒险性。所谓冒险性,是指一旦攻击者发现自己处于诱骗环境中,攻击者就会退出而不再访问诱骗环境,甚至会伪造一些虚假的信息迷惑和误导管理员,从而使诱骗环境失效甚至会成为攻击者所利用的工具。不同于其它安全工具,诱骗系统并不只是收集信息的工具,而是充当攻击的牺牲者。允许攻击者进入诱骗系统,从而可以监控他们的行为,这是诱骗系统的特征之一,而诱骗系统的这一本质特征,也决定了其冒险性。而且随着现在诱骗技术的发展,越来越多的人们意识到诱骗环境的重要性,已经出现了一些针对诱骗环境的探测工具。诱骗主机越多,冒险性就越大,尤其是将诱骗环境放置在网络的核心位置。因此入侵诱骗技术就好比一把双刃剑,能否降低其负面效应是入侵诱骗技术能否长期发展的关键。Honeynet Project 已经定义了数据控制技术,可以限制攻击者在诱骗环境中的能力,但是对于各种不同的网络,限制是不一样的。这样就会给网络管理员增加沉重的负担。

为了降低诱骗环境的冒险性,一方面可以增强诱骗环境的真实性,增加攻击者检测诱骗环境的难度;另一方面可以主动地将攻击者引入到诱骗环境中,并让攻击者察觉不到已经被切换到诱骗环境之中。比较两种方法,第二种方法更为实际。

3.2 相关研究

3.2.1 蜜罐场

Lance Spotzner[2]提出了一个新的体系结构——蜜罐场(Honeypot Farms)。认为只要将攻击流量重定向到蜜罐中,就不需要将蜜罐遍布于网络中;应制定一种路由协议,将符合标准的流量重定向到蜜罐中。相比于传统的蜜罐技术,蜜罐场有如下几个优点。(1)中心日志收集功能。能够收集所有网络上面发生的攻击行为,而不用在每个子网的蜜罐系统里面收集信息,并潜在地降低了日志被修改的可能性,减轻了收集日志的工作量。(2)通过重定向技术可以搭建一个柔性的网络体系。改变蜜罐系统的位置,只需要改变路由表和相关的路由文件即可。(3)增强蜜罐系统的安全性。传统的蜜罐收集攻击本身的信息,如果攻击者不攻击蜜罐则不会收集到任何信息。而通过重定向技术,就可以将攻击真实主机的流量重定向到蜜罐系统中,增强真实主机的安全性。(4)能够扩大监控范围。

3.2.2 Bait & Switch

另一个流量重定向的解决方法是 Bait & Switch[3]。系统重定向恶意流量到 B&S 所指向的蜜罐中。重定向是基于 Snort 签名的匹配。当一个数据包匹配 Snort 签名,它会被 iptables[4] 打上标记,并且一个特定的路由表将这个数据包进行重定向。通过这种方式,B&S 能够将访问真实服务的恶意流量重定向到蜜罐里面做进一步的观察。同时,所有访问真实服务的正常流量不会受到影响。为了增强蜜罐的真实性,蜜罐可以是真实服务的镜像,这会给攻击者造成足够的迷惑。

Bait & Switch 工具的问题在于它选择使用 Fail-Shut(失败则关闭)的方式。这种方式并不能够给用户提供一种无缝的重定向。

3.2.3 NetBait

NetBait 利用了 Disinformation Security(假情报安全)。假情报安全是基于网络上的"提供入侵者关于网络配置和拓扑节点的错误信息"概念[5]。尽管 Netbait 并不认为它的产品是一个蜜罐工具,但是它的许多特性都符合 Lance Spitzner 关于 Honeypot Farms 体系结构的特征。

NetBait 通过一个路由设备重定向流量。这个路由设备有一系列的规则,会重定向任何访问真实主机的打标记的数据包,并将这些数据包发送到虚拟机器上进行数据收集。重定向的规则主要是根据目标端的端口。例如,如果一个 Web 服务放置在 NetBait 路由的后面,它能够将所有对 80 端口的请求重定向到一个虚拟主机上或能够收集数据包的客户端。NetBait 的优势在于它能够按照自己喜欢的方式配置虚拟的主机。这些机器完全可以通过 Web 来配置。Web 访问权限允许进行远程管理。

然而 NetBait 不能提供将真实系统上的攻击流量进行转移的能力。因为重定向的完成是基于目标端口和目标 ip 的。攻击被转移到真实服务的其它开放端口并不是重定向。例如,受 Netbait 保护的 ftp 服务能够潜在地将所有 21 端口的需求重定向到其它端口上。通过将攻击转移到非 21 端口保护机器。

可以看到,现有的关于环境切换技术的研究,主要是通过网络数据包的重定向,在网络层或 ip 层来完成。但是这种方法的实现需要路由器等路由设备的支持,增加了诱骗系统配置的复杂度,而且可移植性差、透明性不高。

4 无缝环境切换技术

在现有入侵诱骗的切换技术中,关注的重点在于如何将可疑流量重定向到诱骗环境当中,而这样的重定向很可能会使攻击者产生怀疑。因为攻击者在真实主机上面会有

一些记录,当攻击者的环境被切换到诱骗主机上后,若发现自己的记录没有了,则会引起攻击者的怀疑。因此,我们研究了一种以用户环境为切换对象、基于 TCP 层的无缝环境切换技术。

无缝环境切换:一个用户访问真实主机资源,当入侵检测系统发现此用户为可疑用户时,将用户环境从真实主机切换到诱骗主机的行为;再通过行为收集分析子系统来判断此可疑用户为合法用户还是攻击者。对于合法用户,对其提供真实服务,将用户环境再次切换回真实主机;对于攻击者,交给入侵响应系统处理。但也不能让可疑用户一直处于诱骗主机中,因此,必须规定一个时间限额:MAXTIME。可疑用户在诱骗环境所待的时间只能小于等于 MAXTIME,当时间超过 MAXTIME 后,除非判断此用户为攻击者,否则需要将此用户环境再次切换回真实主机中。

用户环境:用户访问真实环境时操作系统状态的改变,包括内核信息、进程上下文、网络连接状态等。

4.1 环境切换整体流程

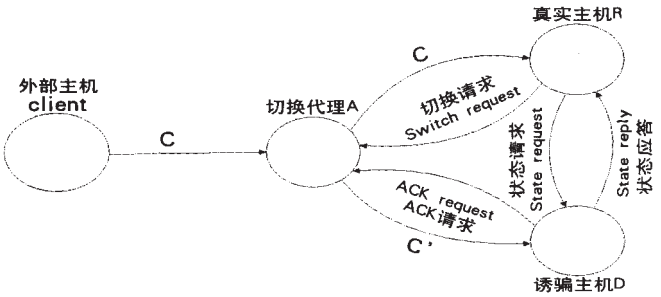

图 2 环境切换子系统整体流程

切换整体流程图如图 2 所示。具体切换流程如下:(1)首先,外部主机发送对真实主机 R 的访问请求 C,切换代理 A 接受请求后与真实主机 R 建立连接,提供真实服务;(2)当某个连接被判断为可疑用户时,真实主机发起对这个连接的环境切换;(3)发送<state request>到诱骗主机 D,告诉诱骗主机 D 需要进行切换,同时真实主机开始收集用户环境状态;(4)诱骗主机 D 接受请求后做一定的初始化,发送<state reply>给真实主机 R,告诉真实主机可以进行状态迁移;(5)真实主机发送状态数据(包括网络连接状态和环境状态)给诱骗主机;(6)诱骗主机 D 接受数据后改变相应初始化的值,使得诱骗主机 D 的状态和真实主机 R 状态一致;(7)处理完之后,诱骗主机发送一个应答请求给切换代理 A,告诉切换代理 A 已经初始化结束,可以进行通信,同时真实主机发送切换请求<switch request>给切换代理 A,告诉切换代理 A 这个连接的请求已经进行切换,以后这个连接的请求都发往诱骗主机 D;(8)切换代理 A 等待诱骗主机 D 的应答请求,收到应答请求后,将这个可疑连接的后续请求发送给诱骗主机 D,完成环境切换。

从上面的分析可以看出,实现无缝环境切换,首要问题是明确用户的环境包括哪些信息。由于本文是基于开放源码的 Linux 操作系统上设计开发的,在 Linux 系统中,用户对主机的访问,从内核角度来讲,可以看成一个带网络连接的用户环境进程,因此我们将环境切换子系统中所需要迁移的状态信息定义为用户环境状态和用户连接状态的集合。下面介绍用户环境切换和用户连接切换中所需迁移的状态信息。

4.2 用户环境切换

用户环境切换技术是环境切换子系统实现的重要组成部分之一。一般地,环境切换子系统中的执行是以用户环境为单位进行的。用户环境是表征计算机上用户对象的一个实例(Instance)的抽象操作系统。一个用户环境从 OS 内核角度来讲,也就是一个用户进程。用户环境切换也是将用户环境进程从真实主机迁移到诱骗主机,因此该技术和分布式系统中的进程迁移技术类似,所以可以借助进程迁移的思想来实现用户环境切换技术。

将一个正在运行的用户从真实主机转移到虚拟主机继续运行而不破坏用户的数据完整性和可用性,需要从操作系统的内核中分离出用户所有进程的各种数据发送到虚拟主机中,然后根据这些信息创建一个新的用户进程,并营造一个与原来进程运行环境相似的新环境[6]。

4.2.1 用户环境切换的状态信息

为了实现真实主机和诱骗主机之间的用户环境切换,最重要的就是确定以下需要迁移的用户环境信息。

(1)环境的执行状态(Execution State):表示当前运行的处理器状态,包括内核在上下文切换时保存和恢复的信息,如通用寄存器内容、浮点寄存器内容、堆栈指针、程序计数器以及状态寄存器内容。执行状态与机器是密切相关的。

(2)环境的控制状态(Process Control State):指操作系统用来控制进程的所有信息,一般包括进程优先级、进程标识(PID)、父进程标识(PPID)等。需要注意的是,一旦控制信息被冻结后,该用户环境就应该被挂起。

(3)环境的 Memory 状态和地址空间:包括用户环境的所有虚存信息、进程数据和进程的堆栈信息等,是进程状态最主要的一部分,也是迁移最耗时的部分。

(4)环境的消息(Message)状态:包括被缓存的消息和关于网络连接的控制信息。在进程迁移期间的网络连接的管理和进程迁移完成后网络连接的重建是迁移消息状态的主要问题,是本文第五章所解决的问题。

(5)文件状态:用户环境的文件状态包括文件描述符和文件缓存块。

4.2.2 用户环境切换流程

用户环境切换模块运行后,在真实主机上产生一个内核守护进程 Switch_RealH,在诱骗主机上产生一个内核守护进程 Switch_JailH。它们都是作为一个内核态线程运行的。Linux 的内核线程是没有虚拟存储空间的进程,它们运行在内核态中,直接使用物理地址空间。Switch_JailH 在固定端口 SWITCH_PORT 监听切换请求。环境切换过程和切换代理无关。当真实主机 A 上的用户环境进程 a 要被切换到诱骗主机 B 上,整个切换过程如下。

(1)向诱骗主机 B 上的切换守护进程 Switch_JailH 发送建立连接请求(切换请求)。B 收到切换请求后,会分叉(fork)出一个新进程 b 来处理切换请求,自己则继续监听切换请求信息。b 被标识为 remote 进程。此时,进程 b 和进程 a 已经建立一条 TCP 通讯连接,用于相互交换信息。

(2)当连接建立好后,A 上守护进程 Switch_RealH 开始获取 a 的进程状态信息,并通过 a 与 b 之间打开的连接将状态发送到进程 b。进程 b 接收到数据后,根据相应信息修改自身的进程状态。

(3)迁移完成后,a 将成为进程的 deputy 部分,一直处

于核心态,直到 b 被切换回真实主机或被处理。B 上,进程 b 改为就绪状态,加入运行队列等待 schedule()调度。

(4)等待 MAXTIME 时间,如果这段时间内此用户没有被判断出为攻击者, 将 b 切换回真实主机。切换过程为:B 上的守护进程 Switch_JailH 获取 b 的进程状态信息,并传送到进程 a;a 接收到数据后, 根据相应信息修改自身的进程状态,因为进程 a 的框架已经存在,因此只需修改相关进程状态即可;b 被杀死,释放资源。至此,一次环境切换完成。

4.3 用户连接切换

用户环境切换是环境切换子系统实现的基础。但同时,必须保证客户端对真实主机的访问不被中断, 因此除了用户环境需要切换之外,还需要对用户连接也进行切换。

Linux 操作系统中是通过 Socket 套接字接口,使用 TCP 和 UDP 作为底层的网络通讯机制。然而,发送和接受网络消息会改变 OS 的缓冲区状态。用户环境和 OS 保持着紧密相关的一些描述信息(如 Socket 标识符、IP 地址等),用户环境迁移后, 将这些描述信息所指向的实体迁移到诱骗主机中是一个非常复杂的过程。

4.3.1 用户连接切换的状态信息

对于客户端来说, 是通过 Socket 接口来操纵网络连接的。当创建一个 Socket 以后,系统返回一个 Socket 描述符。客户端可以利用这个 Socket 描述符,通过虚拟文件系统接口(read,write)或 Socket 接口(send,recive)来操作。无论是通过虚拟文件系统接口还是 Socket 接口来读写或发送接收数据,客户端只需通过对应的文件描述符即可。而具体的路由选择、正确性检验等都是通过内核来处理的。因此,只要保证文件描述符这一层的接口不变,保证数据的收发正确性即可。

对于一个 Socket 描述符 fd 来讲,其关联着许多内核数据结构,如 struct file 结构、struct sock 结构等。它们之间的关系往往是错综复杂、相互交织在一起的。另外,由于协议的层次模型,每层对状态的处理和检测也不一致。

(1)TCP 连接 Socket 的状态包括地址变量、发送和接收顺序变量、当前段变量、拥塞控制状态、定时器状态。另外,如果主机所处网络的拓扑结构和网络接口设备不同,RTT 和 MSS 相关的状态则要在进程所处的新主机上重新计算。

(2)接收和发送的数据状态是指切换用户连接时,除了要迁移用户连接的状态,还要迁移数据。每条连接都拥有一个接收和发送缓存队列。接收队列保存的是已从网络中接收被 TCP 层处理完毕但是还未被应用层所读取的数据。发送队列则是保存着应用层提交给内核等待发送的数据。当用户环境切换后,将在诱骗主机上恢复运行,从真实主机处读取和发送数据。因此,接收和发送队列必须也在切换时在诱骗主机上重建,否则就会造成数据丢失的错误。

4.3.2 用户连接切换流程

我们以下面的例子来说明如何进行用户连接的切换。三元组 T<用户进程,本机 IP,本机 port>表示本地进程。客户端 A 的进程 A1 的 T1<A1,IP_A,Port1>与真实主机的切换代理 Switchproxy 的 Ts<Ts,IP_s,Ports>建立了一条 TCP 连接,然后 Switchproxy 与真实主机上的进程 B1 的 T2<B1,IP_B,Port2>建立一条 TCP 连接 P1(IP_s,Ports,IP_B,Port2), 于是客户端可以得到真实主机所提供的真实服务。当怀疑此客户为可疑用户时进行以下用户环境切换:

(1)真实主机向诱骗主机 C 发送切换请求,将 B1 进行用户环境切换为诱骗主机上的进程 C1,有关切换过程参见第 4.1.2 节;

(2) 真实主机 B 上的 Socket 状态控制模块向 Switchproxy 发出迁移通告, 标志该连接已处于准备迁移状态,抑制 Switchproxy 向该连接继续发送数据;

(3)真实主机 B 上的 Socket 状态控制模块向诱骗主机 C 发送用户连接切换请求, 诱骗主机分配一个新的 Socket 描述符并与 C1 进程绑定;

(4)真实主机 B 上的 Socket 状态发送模块开始收集用户连接状态并打包发送到诱骗主机 C,C 解包, 根据包内容恢复该连接状态,如果该连接的端口已被占用,则将得到一个新的端口;

(5)C1 向 B1 发送 SWITCH_SOCKET_REPLY 消息,通知 B1 连接状态已被恢复, 并返回新的地址和端口(IP_C,Port3);

(6)C1 向切换代理发送 CHANGE_ADDR 消息(IP_s,Ports,IP_B,Port2,IP_C,Port3),告知它由(IP_s,Ports,IP_B,Port2)标识的网络连接的另一端已变成为(IP_C,Port3);

(7)切换代理根据(IP_s,Ports,IP_B,Port2)找到匹配的连接的 sock 控制结构,修改该连接的目的地址和端口分别为 IP_C 和 Port3, 然后发送 CHANGE_ADDR_REPLY 响应消息;

(8)C1 收到 CHANGE_ADDR_REPLY 响应消息,该连接切换完毕;

(9)可疑用户所有数据都被传送到诱骗主机的 C1 进程中,当 C1 发生缺页等情况时将通过真实主机 B 上的 B1 获取相关信息,当到达 MAXTIME 时,C1 重新被切换回真实主机的 B1。

5 结论

入侵诱骗系统是动态网络防御体系研究当中的热点。本文通过对现有的入侵诱骗系统切换技术的研究,提出了一种以用户环境为切换对象、基于 TCP 层的无缝环境切换技术,给出了环境切换中所需要迁移状态信息的定义,阐述了切换流程,并以 Linux 系统为实例进行了验证,验证结果表明该系统能够将攻击者引导进诱骗环境中,而且并不会引起攻击者的怀疑。这对构建网络动态防御体系具有重要意义。

参考文献(References)

[1] 夏春和.基于入侵诱骗的网络防御体系及关键技术研究[D].北京:北京航空航天大学博士论文,2003
[2] Lance Spitzner Honeypot Farms [EB/OL]. http://www.securityfocus.com/infocus/1720
[3] The Bait & Switch Honeypot[EB/OL]. http://violating.us/projects/baitnswitch/
[4] The iptables project[EB/OL].http://www.netfilter.org/
[5] Employing Disinformation Security to Protect Corporate Networks with NetBait [EB/OL]. http://www.netbaitinc.com/products/disinfo_wp.pdf
[6] 陆铭璐,谢立.进程的动态迁移技术[J].计算机研究与发展,1997,34(9)

(责任编辑 胡春华)