文章编号: 1001 - 9081(2008) S1 - 0302 - 03

一种支持动态演化的防火墙软件的设计与实现

李开拓,胡 羽,张家晨
(吉林大学 计算机科学与技术学院,长春 130012)
(zhangjc@jlu.edu.cn)

摘 要:对于软件防火墙,如果具备了动态更新的能力,会使被保护的系统更安全,即使在更新时也能对被保护系统进行保护。为达到这一目的,在分析 netfilter和 OSGi框架的原理的基础上,提出采用这两种框架分别处理防火墙安全策略和功能模块更新的方案,实现了一种基于这两种框架的支持动态演化的防火墙系统。

关键词:软件动态演化;软件在线更新; OSGi; netfilter
中图分类号: TP311.53 **文献标志码**: A

Design and implementation of software firewall supporting dynamic evolution

LI Kai-tuo, HU Yu, ZHANG Jia-chen
(*College of Computer Science and Technology, Jilin University, Changchun Jilin* 130012, *China*)

Abstract: If software firewall could support dynamic evolution, the protected systems will be safer, for firewall could be updated as it ran. To accomplish this goal, based on the analysis of the principles of netfilter and OSGi framework, a proposal that security policy update in firewall was processed by netfilter framework, as well as the function module updated by OSGi framework. The firewall supporting dynamic evolution was also implemented.

Key words: software dynamic evolution; software runtime update; OSGi; netfilter

0 引言

软件动态演化是指软件在运行期间的演化。当前,大量的应用系统运行于 Internet这样一个开放的、动态的和多变的环境之中,面对的是硬件资源和客户功能需求的频繁变化,它们常常需要动态调整自身的行为以处理新型的信息对象,需要伸缩自身的处理能力以适应可能会不断变化的任务环境与需求。而要使软件满足这样的要求就必须使其有动态演化的能力,以动态地增减其组成构件,调整内部结构。特别对诸如生命维持软件、生命控制系统、电信交换系统和武器控制系统等关键任务系统来说,常常需要在不停机的前提下进行软件的动态更新。同时保护这些安全敏感系统的防火墙软件也必须时刻通过对流经它的网络通信进行监控来实现安全防护。但是和其他软件一样,防火墙软件也避免不了需要升级或者修改 bug之类的问题。关闭这些高可用系统上的防火墙是不允许的。例如银行交易系统,系统上防火墙的片刻中断都有让银行系统裸露在众多恶意黑客程序窥伺之下的危险,随时可能造成客户巨大的财产损失。此时最好的选择就是进行在线的演化。本文试图实现一个能够动态演化的防火墙系统。

通过整合支持软件动态演化的 OSGi框架和允许在内存中随意增删改查安全规则的 netfilter框架,本文设计了一个满足动态演化要求的防火墙软件 DFW。当 DFW正在运行的时候,用户可以更新已有的安全策略和功能模块。

1 系统设计方案

1.1 系统研制目标

对于本文设计的防火墙软件 DFW其研制目标如下:

1) DFW可以行使一般防火墙所应该有的过滤数据包、管理进出网络的访问行为、封堵某些禁止的访问行为和记录通过防火墙的信息内容和活动等功能。

2) DFW允许用户动态的改变其功能模块,改变前的数据和状态能够被正确无误的过渡给改变后的 DFW使用。

3) DFW允许用户在不影响网络安全保护的前提下,让用户删除之前定义的一套安全策略,而换用新定义的安全策略。

1.2 系统运行环境

DFW用 Java和 XML语言编写,在 Eclipse平台上开发。运行于 Linux/Unix操作系统,要求操作系统内核实现了 netfilter框架,并且系统内安装了 Java运行时环境 (JRE)。对于 Linux来说, 2.4和 2.6版本都在内核中实现了 netfilter框架。

1.3 系统的功能模块

本系统是一个基于 OSGi框架和 netfilter框架、在 Linux/Unix操作系统中运行的、支持动态演化的防火墙工具。具体功能包括:安装/卸载安全策略、添加/删除安全规则、对防火墙进行动态演化,如图 1所示。

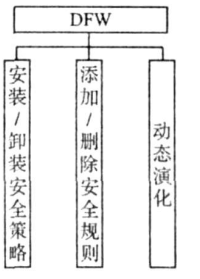

图 1 系统功能模块

对系统各功能模块说明如下:

1)使用者可以根据需要安装或卸载安全策略。当主机

收稿日期: 2007 - 11 - 27。

作者简介:李开拓 (1985 -),男,湖南岳阳人,主要研究方向:软件维护; 胡羽 (1981 -),女,吉林榆树人,硕士研究生,主要研究方向:软件维护; 张家晨 (1969 -),男,吉林德惠人,教授,博士, CCF高级会员,主要研究方向:软件维护。

没有用防火墙保护时,主机默认接受一切到达主机的数据包。当使用者点击安装防火墙策略的命令按钮时,系统会弹出一个窗口要求用户选择安全策略文件路径,此时用户有两个选择:一是选择一个原来用 XML格式保存的安全策略文件,系统读取安全策略文件,执行文件中记录的防火墙规则,这有利于简化用户的操作,使原来的防火墙配置能够马上执行,用户不必每次都从头开始配置防火墙;二是预先建立一个空文件,系统会自动做初始化工作,用户只需做相应的选择就可以了。当使用者点击卸载防火墙策略的命令按钮时,系统会弹出一个窗口要求用户选择安全策略文件路径,这是因为防火墙运行的时候安全策略记录通过 Policy类保存在内存中,Policy类会随着用户设定动态的改变其数据域的内容,因此当用户要结束某个运行着的安全策略时,用户可以选择将安全策略从内存中写回到硬盘上的 XML文件。

2)使用者可以依据 iptables语法规则按照需要随时配置各种安全规则,只要添加规则成功,规则即刻在内存中执行,开始过滤数据包。传统的防火墙中,建立一套新的安全策略一般要先停止当前安全策略的执行,然后从内存中卸载掉旧的安全策略,重新安装新版本的安全策略,最后才能开始执行新版本的安全策略。在 DFW中,因为是通过 netfilter框架中的 iptables命令配置安全策略,依据 netfilter的最先匹配原则(数据包会在最先被匹配的地方停下来),可以在不影响旧有防火墙安全策略的前提下配置一套新的安全策略,配置过程中旧有的防火墙安全策略仍然可以对主机行使安全保护。等新的安全策略配置好之后,可以逐一的删除旧有的安全策略,在这个阶段新、旧两种安全策略共同提供主机的网络安全保护。当旧的安全策略删除完毕之后,新的安全策略将完全取代旧的安全策略保护主机的网络运行。这就是动态更新防火墙的安全策略的过程。

3)使用者可以对防火墙软件的功能进行动态演化。使用者通过可视化的防火墙软件和安全策略打交道,而安全策略和系统内核打交道。iptables语法规则通过表目标扩展和表匹配扩展机制可以将防火墙的安全规则表示方法无限升级,随时适应计算机网络技术的发展。这就是说安全策略的组织方式和配置方式是变化的,另外防火墙软件的功能也需要经常升级:或者是修改 bug,或者是为了完善软件功能、让用户更高效、更安全、更方便地使用软件。考虑到这些因素,DFW完全按照支持动态演化的 OSGi框架编程规范进行设计和编码。用户可以在 OSGi控制台用 start命令启动 DFW,用 stop命令停止 DFW的运行,用 update命令更新已经在内存中运行的 DFW。在进行更新的相当短的一段时间里,旧版本的防火墙软件的设置仍然有效,旧版本的防火墙软件的状态和数据都将传递给新版本,最终新版本防火墙在内存中将完全替代旧版本防火墙。

2 系统关键模块的实现

2.1 安全策略的动态更新

2.1.1 在 netfilter框架下动态更新安全规则的原理

netfilter/iptables框架是从 Linux 2.4内核开始启用的一个全新的体制,它把新的特性加入到内核中并不需要重新启动内核。netfilter/iptables的框架包括数据包过滤 filter、网络地址转换 nat、数据包处理 mangle三个功能表[1-3]。数据包过滤表 filter包括三个链:INPUT、FORWARD、OUTPUT;网络地址转换表 nat包括三个链:PREROUTING、OUTPUT、POSTROUTING;数据包处理表 mangle包括五个链:PREROUTING、INPUT、FORWARD、POSTROUTING、OUTPUT。每个链包括具体的数条数据包过滤规则。使用 iptables命令就可以在这些表中的各个插入点上制定一些 IP包处理规则链,经过每个插入点的数据包,都会按照这些规则链定义的动作来处理,绝大部分包处理功能都可以通过在这些内建的表格中填入规则完成。每一条规则都是这样定义的:如果数据包头符合这样的条件,就这样处理这个数据包。当一个数据包到达一个链时,系统就会从第一条规则开始检查,看是否符合该规则所定义的条件:如果满足,系统将根据该条规则所定义的方法处理该数据包;如果不满足则继续检查下一条规则。这是 netfilter框架的最先匹配准则。

安全规则的动态更新是通过安全规则的增加和删除操作完成的。查看 Linux-2.6.20.1/net/ipv4/netfilter/ip_tables.c文件中的 do_replace()函数,就可以知道 netfilter框架不区分安全规则的增加和删除操作,而把它们通通看作对规则集的修改,新的规则集会替换旧的规则集,这是通过指针值的变化来实现的,这也是为什么用户可以随时添加或修改内核中的规则而无需重启内核的原因。最后,do_replace()函数会把老规则集和计数器所占用的空间释放掉。

2.1.2 DFW中安全规则动态更新的实现

使用者可以依据 iptables语法规则随时配置各种安全规则,只要添加规则成功,规则即刻在内存中执行,开始过滤数据包。DFW中增加和删除安全规则首先都要将用户使用图形工具配置的安全规则变量转换为可以执行的 iptables命令,然后执行这个脚本,最后将内存中安全策略的变化反映到图形界面显示上。

将安全规则变量转换为可以执行的 iptables命令是很关键的一步。由于目前 DFW只支持对 netfilter框架中 filter表上的操作,它考虑的规则链只有三种:INPUT、OUTPUT和 FORWARD。将安全规则变量转换为 iptables命令的算法流程见图 2。

数组 FilterChain中的元素表示 INPUT、OUTPUT、FORWARD、反向 INPUT、反向 OUTPUT、反向 FORWARD等 6条链是否设定为匹配。将它们的初值都赋值为 false,也就是说默认情况下所有的 6条链都不匹配数据包。后三条链是为了简化有 TCP服务或是没有指定服务的防火墙安全规则的配置而虚造的,分别对应前 3条链来说的。对应于一条在 INPUT链中的 iptables规则(-A INPUT -s a -d b),反向 INPUT链中对应的 iptables规则是(-A OUTPUT -d a -s b)。对应于一条在 OUTPUT链中的 iptables规则(-A OUTPUT -s a -d b),反向 OUTPUT链中对应的 iptables规则是(-A INPUT -d a -s b)。对应于一条在 FORWARD链中的 iptables规则(-A FORWARD -s a -d b),反向 FORWARD链中对应的 iptables规则是(-A FORWARD -d a -s b)。考虑到 TCP连接提供的是全双工服务,如果一个方向上的数据是可信任的,那么绝大多数情况下,另一个方向上的数据也应该是可信任的。上述 3条虚造的规则链可以帮助用户自动的构造另一方向上的数据信任规则。另外,如果防火墙安全规则没有指定匹配的服务,说明用户希望这条防火墙规则不使用服务匹配选项,即防火墙规则可以匹配任何服务,当然也包括 TCP服务,所以碰到这种情况时,DFW的处理方法和碰到 TCP服务匹配时是一样的。

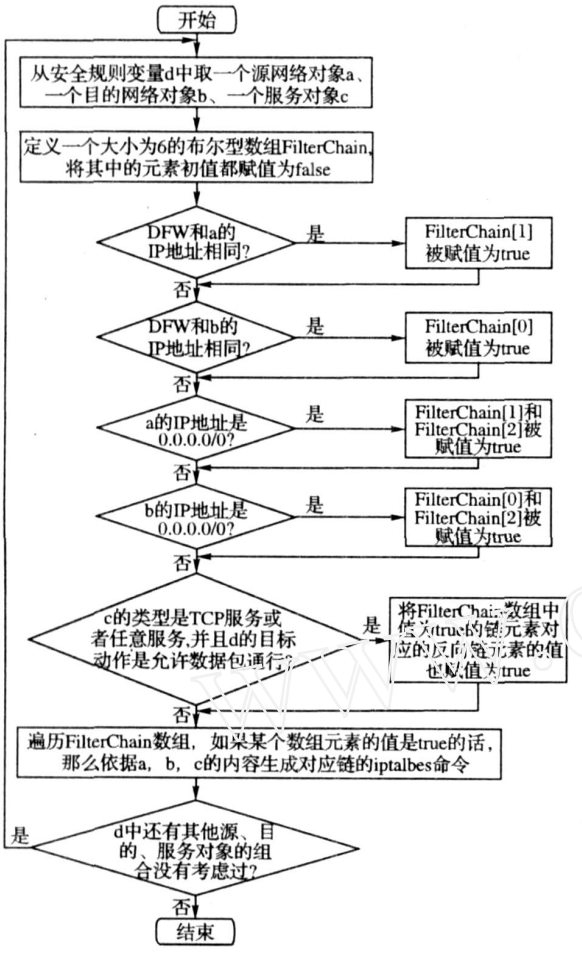

图 2 转换算法流程

2.2 防火墙功能的动态更新

2.2.1 在OSGi框架下动态更新Bundle的原理

OSGi框架规范是开放服务网关组织指定的一个规范。它的最大特点之一是其热插拔特性。OSGi规范的核心是Bundle。一个Bundle就是一个JAR文件,这个jar文件和普通的jar文件唯一不同的地方就是Meta-inf目录下的MANIFEST.MF文件的内容,关于这个Bundle的属性和与其他Bundle之间的依赖关系都在MANIFEST.MF中进行描述[4]。这样,每个Bundle都是自描述性的。

OSGi基于Java技术,其能够实现超强的动态性的机制完全是利用Java的ClassLoader的强大功能[5]。在Java中,ClassLoader负责从类文件中装载类。Java虚拟机中有一个默认的类装载器即系统类装载器,它只负责从JVM的"java class path"属性指定的路径或".jar"文件中装载类,并不装载和已经装载过的类同名的类定义。所以,系统类装载器不支持类的在线替换。OSGi的实现中重载了系统默认的ClassLoader,提出了模块隔离ClassLoader的机制,同时也增强了ClassLoader按版本加载的功能。由于同一个名字的class,运行时类型可以分属不同的ClassLoader命名空间,这样在JVM中可以加载同一个类的不同版本,起到模块隔离和按版本加载的作用。按照在OSGi的Framework规范的实现中,每个Bundle都有自己的ClassLoader,可以加载自己私有的class,同时Bundle也可以通过对外共享或者注册为服务依赖和其他Bundle交互。

2.2.2 DFW中防火墙功能动态演化的实现

在动态更新的过程中,OSGi框架会先调用旧版本软件的Bundle.stop方法停止旧版本软件的运行,然后调用新版本软件的Bundle.start方法启动新版本软件的运行。这说明只要在旧版本软件的Bundle.stop方法中做好数据保存工作,在新版本软件的Bundle.start方法做好数据恢复工作,就可以实现软件的动态演化。至于何时调用旧版本软件的Bundle.stop方法,又是何时调用新版本软件的Bundle.start方法则由OSGi控制台说了算。

一个Bundle通过它的执行器(Activator)来启动。执行器是一个实现了BundleActivator接口的类。这个类会在其自述文件的Bundle-Activator属性里声明。BundleActivator接口中包含start方法和stop方法,DFW实现了该接口,并在start和stop方法中指明Bundle启动、停止时所需要进行的工作。

另外,DFW中还有一个配置文件DFW.conf用来记录Bundle上一次是否在没有卸载安全策略的情况下停止了运行。如果是的话,说明内存中有一套安全规则在过滤数据包,DFW的界面能够反映正在执行的安全策略的内容;如果不是的话,则可以开始一个新的会话。

DFW里start方法的实现过程如下所示:

1)声明安全策略变量policy,建立一个新的JFrame——f;

2)在f中添加控件:包括工具条、按钮、滚动面板;

3)读取配置文件DFW.conf,如果返回的结果说明内存中有一套安全策略在发挥作用,则从指定位置读取内存中安全策略的XML文件表示,根据XML文件中各个元素的值对policy赋值;否则将policy赋值为空;

4)如果policy不为空的话,那么使用policy对最新的TableModel赋值,然后在滚动面板的表格里显示数据;否则,在滚动面板里显示一幅欢迎使用的图。

5)设置f的大小、位置、可见性。

DFW里stop方法的实现过程如下所示:

1)如果内存中有一套安全规则在过滤数据包,那么将policy变量转换为XML文件表示,并将XML文件存储在指定路径。修改配置文件DFW.conf指定字段,说明内存中有安全策略在执行;

2)销毁窗口,回收f所占用的内存。

3 结语

本文设计并实现了支持动态演化的防火墙工具DFW。DFW中能够实现的动态演化包括安全策略的动态演化和防火墙软件本身的动态演化。传统的防火墙中,建立一套新的安全策略一般要先停止当前安全策略的执行,然后从内存中卸载掉旧的安全策略,重新安装新版本的安全策略,最后才能开始执行新版本的安全策略。DFW可以动态更新安全策略而不影响安全保护。另外,用户可以通过OSGi控制台实施防火墙功能的动态更新,以动态调整安全策略的组织方式和配置方式。

参考文献:

[1] RUSSELL R. Packet filtering howto[EB/OL]. [2007-06-10]. http://www.netfilter.org/documentation/index.html

[2] ANDREASSON O. Iptables tutorial 1.2.2[EB/OL]. [2007-06-10]. http://iptablestutorial.frozentux.net/iptables-tutorial.html

[3] RUSSELL R. Netfilter hacking howto[EB/OL]. [2007-06-10]. http://www.netfilter.org/documentation/index.html

[4] The OSGi Alliance. OSGi technology[EB/OL]. [2007-06-10]. http://www.osgi.org/osgi_technology/index.asp?section=2.

[5] The OSGi Alliance. OSGi service platform core specification (Release 4)[EB/OL]. [2005-08-01]. http://www2.osgi.org/Specifications/HomePage.