防火墙中规则的翻译及检测方法的研究

周华平
(安徽理工大学 计算机科学与技术系,安徽 淮南 232001)

摘 要:iptables的Web配置系统,虽然使防火墙的规则输入变得非常容易操作,避免了输入规则的语法错误,而同时又保持了iptables的强大功能,但没有考虑规则之间的联系。在以往的防火墙规则输入过程中,规则都由用户判定其语义是否正确,规则之间是否矛盾,是否有无用的规则。但因规则的语义是与其在规则表中的位置相关的,因此,当规则较多时,很容易出错。因此文中针对基于Linux防火墙的包过滤系统,提出了如何对包过滤规则进行翻译和正确性检测。

关键词:iptables;防火墙;Netfilter;网络地址转换;规则
中图分类号:TP393.08 **文献标识码**:A **文章编号**:1673-629X(2007)11-0135-04

Research of Firewall Rules Translation and Examination Method

ZHOU Hua-ping
(Dept. of Computer Sci. and Techn., Anhui Univ. of Sci. and Techn., Huainan 232001, China)

Abstract: Although iptables' Web disposition system makes the firewall rules easy to input, avoids grammatical error, and simultaneously maintains the iptables' formidable function, it does not considerate the relation between the rules. In the former firewall rules inputing process, they are completely determined by the user whether the semantics is correct, whether they are contradictory between the rules, whether it exists the useless rules. Because the rule semantics is connected with the place in the rules table, it is very easy to make a mistake when it has more rules. This paper proposes how to translate packets filtering rules and check their validations based on the Linux firewall's packet filtering system.

Key words: iptables; firewall; Netfilter; NAT; rule

1 利用Netfilter实现包过滤

一个基于Netfilter框架的[1]、称为iptables的数据报选择系统在Linux2.4内核中被应用,其实它就是ipchains的后继工具,但却有更强的可扩展性。

内核模块可以注册一个新的规则表(table),并要求数据报流经指定的规则表。这种数据报选择用于实现数据报过滤(filter表),网络地址转换(Nat表)及数据报处理(mangle表)。Linux2.4内核提供的这三种数据报处理功能都基于Netfilter的钩子函数和IP表。它们是独立的模块,相互之间是独立的。它们都完美地集成到由Netfilter提供的框架中。包过滤filter表格不会对数据报进行修改,而只对数据报进行过滤。iptables优于ipchains的一个方面就是它更为小巧和快速。它是通过钩子函数NF_IP_LOCAL_IN,NF_IP_FORWARD及NF_IP_LOCAL_OUT接入Netfilter框架的。因此对于任何一个数据报只有一个地方对其进行过滤。这相对ipchains来说是一个巨大的改进,因为在ipchains中一个被转发的数据报会遍历三条链。

IPv4[2]数据包流经线路如图1所示。

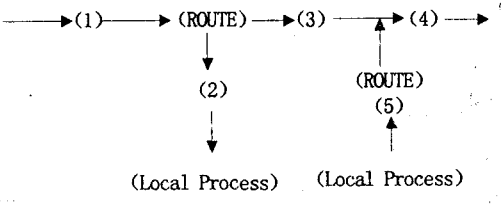

图1 IPv4数据包流经线路(filter)

2 防火墙规则语义翻译

因防火墙规则表中的规则是顺序检测的,因此规则表中的顺序是非常重要的,如前面的规则可以允许被后面的规则拒绝的数据包通过,而反过来就不成立,即后面的规则无法做到允许被前面的规则拒绝的数据包通过。如表1中的两条规则。

收稿日期:2007-02-07
基金项目:安徽省教育厅青年基金资助项目(2005jq1102);安徽理工大学青年科学研究基金资助项目
作者简介:周华平(1979-),女,河南唐河人,硕士,研究方向为网络安全及应用。

表 1 规则示例

序号	方向	本地主机	外部主机	本地端口	外部端口	动作
1	进	ALL	HC1	ALL	ALL	阻止
2	进	WWW	ALL	80	ALL	通过

两条规则的语义为:

第一条:阻止外部主机 HC1 的所有访问;

第二条:允许所有外部主机访问本地主机 SMTP_MAIL 的 80 号端口。

虽然第二条允许所有外部主机访问本地主机 WWW 的 80 号端口,但是第一条已阻止了来自 HC1 的所有访问,所以,HC1 仍是无法访问 WWW 的 80 号端口。如将上面规则的第一条与第二条互换,如表 2 所示。

表 2 互换后的规则

序号	方向	本地主机	外部主机	本地端口	外部端口	动作
1	进	WWW	ALL	80	ALL	通过
2	进	ALL	HC1	ALL	ALL	阻止

此规则的语义为:允许所有外部主机访问本地主机 WWW 的 80 号端口;禁止来自 HC1 的其他连接。这样的结果是,HC1 也可以访问主机 WWW 的 80 号端口。

由上面可以看出,规则的顺序是非常重要的,两条规则的顺序调换以后,其语义已完全不同,为此,应在每条规则进入之前,将其翻译成自然语言,向用户显示,由用户判定语义的正确性,但因每条规则的语义是与其在规则表中的位置是紧密相关的,是与其上面的规则相联系的,所以首先将每条规则形式化,然后根据上面的规则对本条规则进行修正,最后,将其翻译成自然语言,显示给用户。这将完整地表达该条规则的语义,从而用户可判断是否符合其初衷,以决定本条规则是否可以录入。

形式化地,把过滤规则表中的规则按次序表示为:

A→B; C→D; E→F …表示顺序三条规则,其中 A→B 表示由前提条件 A 推出结论 B,在这里结论 B 如允许、拒绝和日志等[3]。

因规则是顺序执行的,所以规则的完整形式应为:

A→B; ~A∧C→D; ~A∧~C∧E→F…

即后面规则的前提条件与前面所有规则前提条件的非做合取,构成其前提条件。

对表 1 中的规则表设:

RA(x):远程主机地址是 x;

LA(x):本地主机地址是 x;

RP(x):远程主机端口号是 x:

LP(x):本地主机端口号是 x;

P:通过。

则:

~RA(x):远程主机地址不是 x;

~LA(x):本地主机地址不是 x;

~RP(x):远程主机端口号不是 x;

~LP(x):本地主机端口号不是 x;

~P:阻止。

则表 1 中的防火墙规则可以写为如下的形式:

1) ∀x∀y∀z∀u(RA(HC1)∧RP(y)∧LA(z)∧LP(u))→~P

若忽略全称量词及其约束的谓词,可简化为:

RA(HCI)→~P

2) ∀X∀y(~RA(HC1)∧RA(X)∧RP(y)∧LA(MAIL)∧LP(80))→P

若忽略全称量词及其约束的谓词,可简化为:

~RA(HC1)∧LA(WWW)∧LP(80)→P

若忽略全称量词及其约束的谓词,可简化为:

~RA(HC1)∧~(LA(WWW)∧LP(80))→~P

按上面的形式可以译为自然语言:

(1)来自 HC1 的包访问全部阻止;

(2)除来自 HC1 的包,访问本地 WWW:80 号端口的包,可通过。

iptables 对数据包的检测分为三条链,分别为: input,output,forward,所以实际应用时要三条链分开进行,分三部分进行语义分析,三条链的处理方法是一样的,具体的算法可描述为:

a. 新建三个文件,一个存放形式化的规则(File1),一个存放处理后的形式化的规则(File2),一个存放翻译后的规则(File3)。

b. 输入一条规则 Rule。

c. 将 Rule 表示成合式公式,并在忽略全称量词及其约束的谓词后,成如下形式:Condition→Conclusion,若 Rule 是第一条规则,则直接追加到 File1 和 File2 中,否则,设 File1 中所有规则的前件分别为 Condl,cond2,…,condn,Rule 的前件为 Condition,则将 Condition→Conclusion 追加到 File1,将~Condl∧~Cond2∧…∧~Condn∧Condition→Conclusion 追加到 File2。

d. 将 Rule 利用追加到 File2 中的形式译为自然语言,追加到 File3,并显示给用户。

e. 如要继续输入则转 b.。

(结束)

在第三步中生成一新的条件时,当规则很多时可能会生成很长的条件,大多数情况条件是可以化简的,例如:

在表 2 的例子中

第一条:RA(HC1)→~P

现在增加一条,HC2 可访问 WWW 的 80 号端口,

则生成的条件为:

第二条:~RA(HC1)∧RA(HC2)∧LA(WWW)∧LP(80)→P

若 HCI 与 HC2 不相同则第二条可以简化为:RA(HC2)∧LA(WWW)∧LP(80)→P

3　规则正确性检查

对规则集的考虑应该是整体是否有效、规范、正确,而不仅仅是规则集中的每条规则是否有效、规范、正确。有时,若干条均正确有效的规则构成的规则集的整体可能是有安全隐患的。如果当管理员按照包过滤规则编辑器的步骤和实际的安全策略制定过滤规则在应用之前进行规则的冲突检测,这样一来不仅可以避免规则之间出现矛盾、冗余而且便于规则的管理和维护,也便于定义较为完备的安全策略,实现防火墙包过滤的功能。

规则相容性检查是指检查进入过滤规则表中的所有规则的相容性问题,如新输入的规则与现有规则表中其他规则不相容,或为规则表中所有规则的逻辑结论,则这条规则不能进入过滤规则表中,需提醒用户。

如果新加入的规则与前面所有规则的合取式是永假的,即与前面的规则不相容,此时该规则不应加入规则表中,如果新加入的规则是前面规则的逻辑结论,则这是一条多余的规则,也不应加入规则表中。

规则的相容性判断原理[4]如下:

NewRule 与规则集{R1,R2,…,Rn}不相容,当且仅当存在 Ri{R1,R2,…,Rn},Ri∧NewRule=空,(R1∧R2∧…∧Rn)∧NewRule 为空;

规则的冗余性判断原理如下:

NewRule 与规则表冗余,即 NewRule 为规则集{R1,R2,…,Rn}的逻辑结论,当且仅当(R1∧R2∧…Rn)=>NewRule。

由于:

A→B⇔~A∨B⇔~(A∧~B)

所以,规则的冗余性判断原理可表示为如下等价形式:

NewRule 与规则表冗余,即 NewRule 为规则集{R1,R2,…,Rn}的逻辑结论,当且仅当(R1∧R2∧…∧Rn)∧~NewRule 为空。

可以发现,冗余性判定和相容性判定可采用相同的方法。

对于不相容性的测试可使用归结原理,首先将欲证为不可满足的谓词公式变化为斯柯林标准形,然后变为子句集并用鲁宾逊消解原理证明其不可满足性。先用一例子看看手工判断规则的冗余性的步骤。

表 3 中的两条防火墙规则,显然第二条是多余的,即它为上面规则的逻辑结论。

表 3　两条规则示例

序号	方向	动作	本地主机	外部主机	本地端口	外部端口
1	进	通过	ALL	All	ALL	ALL
2	进	通过	ALL	All	ALL	ALL

下面是手工的判断过程,规则可写成如下形式:

(1)将自然语言转化为逻辑符号表示。

∀x∀y∀z∀u(RA(x)∧RP(y)∧LA(z)∧LP(u)→P)　①

∀y∀z∀u(RA(a)∧RP(y)∧LA(z)∧LP(u)→P)　②

欲证②是①是逻辑结论只须证~②与①是不可满足的。

(2)变成斯柯林标准型。

①⇔∀X∀y∀Z∀u~RA(X)∨~RP(y)∨~LA(z)∨~LP(u)∨P)

省去全称量词:

~RA(x)∨~RP(y)∨~LA(z)∨~LP(u)∨P

~②⇔~(∀∨∀z∀u~(RA(a)∧RP(y)∧LA(z)∧LP(u))∨P))

⇔∃y∃z∃u(RA(a)∧RP(y)∧LA(z)∧LP(u)∧~P)

去掉存在量词:

RA(a)∧RP(b)∧LA(c)∧LP(d)∧~P

(3)消解。

~RA(x)∨~RP(y)∨~LA(z)∨~LP(u)∨P　①

RA(a)　②

RP(b)　③

LA(c)　④

LP(d)　⑤

~P　⑥

~RP(y)∨~LA(z)∨~LP(u)∨P　⑦

(①,② δ={a/x})

~LA(z)∨~LP(u)∨P　⑧

(⑦,③ δ={b/y})

~LP(u)∨P ⑨(⑧,④ δ={c/z})

P ⑩(⑨,⑤ δ={d/u})

口(⑩,⑥)

相应的归结演绎树如图 2 所示。

由上面的判断可知防火墙第二条规则是第一条规则的逻辑结论。第二条规则是重复的,不应存入防火墙规则表中。可以看出规则冗余性检查的手工实现过程,从这里可以得到把这一过程用算法实现的步骤:

1)新建一个文件 file;

2)规则集中的字句集置入 file 中;

3)若空字句 NIL 在 file 中,则新规则是已存规则的逻辑结论,结束;

4)按某种策略在 file 表中寻找可归结的字句对,若存在则归结之,并将归结式并入 file 中,转 2);

5)归结失败,不是已存规则的逻辑结论,退出。

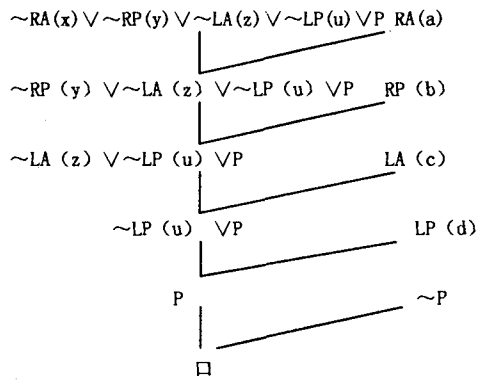

图 2 归结演绎树

4 程序流程

程序首先将刚输入的规则翻译成自然语言,如果觉得不符合要求,继续输入规则。如果用户觉得符合初衷,继续进行判断看该规则是否符合语法检查,如果符合,还要检查该规则是否是前面规则的逻辑结论,如是则相当于存在这条规则。再根据情况做出处理,向用户给出提示。如不是,说明该规则符合要求,把刚刚输入的规则存放到文件中[5]。流程见图 3。

系统判断某条规则是否存在时,只要判断这条规则是不是前面的规则的逻辑结论即可,不是要判定是否有这样一条完全相同的规则。

规则的自动检测与语义翻译,在实际应用中对网站的管理人员是很有帮助的,可以避免规则的输入错误,或是及早发现错误,以免输入了不正确的规则而给系统造成安全漏洞。该检测方式不只是适用于本系统,凡是基于包过滤的防火墙规则存在规则的冲突问题,也同样可以用此方法进行检测。

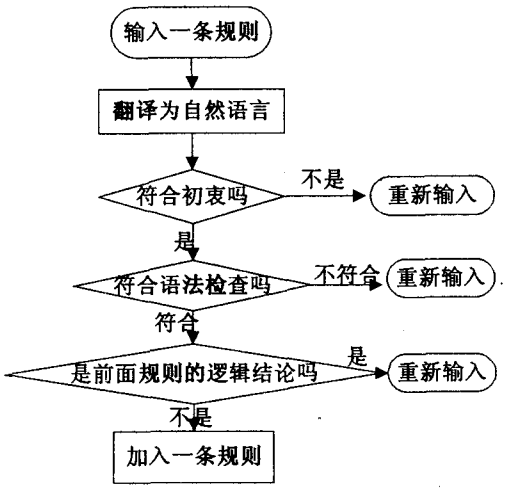

图 3 加入一条规则执行流程

5 结束语

针对 Linux 下的包过滤防火墙,由于利用 Netfilter 中的 iptables 命令配置防火墙规则的复杂性和易错性,在此基础上讨论了防火墙规则的翻译和正确性检测。试验证明它对辅助管理员进行防火墙规则表设置是有一定帮助的。

参考文献:

[1] 巴卡卡蒂. Red Hat Linux 奥秘[M]. 第 3 版. 北京:电子工业出版社,2000.

[2] 张 双. 应用代理防火墙中央日志审计子系统的设计与实现[D]. 北京:中国科学院软件研究所,2001.

[3] Ziegler R L. Linux 防火墙[M]. 北京:人民邮电出版社,2001.

[4] Callan R. 人工智能[M]. 北京:电子工业出版社,2004.

[5] 陈 颖. 基于包过滤的个人防火墙的研究与实现[D]. 西安:西安交通大学,2002.

(上接第 127 页)

参考文献:

[1] Mooney J K. UDDI technical white paper[EB/OL]. 2002 - 09 - 06. http://www.uddi.org/pubs/Uru UDDI Technical WhitePaper.pdf.

[2] OWL Services Coalition OWL - S: Semantic Makeup for Web Services OWL - S v. 1.1, White Paper[EB/OL]. 2004 - 11. http://www.daml.org/services/owl - s/1.

[3] OWL - S Semantic Makeup for Web Services[EB/OL]. 2004 - 11 - 22. http: www.w3.org/Submission/2004/SUBM - OWL - S - 20041122/2004/11/22.

[4] 胡继才,万福钧,吴珍权,等. 应用模糊数学[M]. 武汉:武汉测绘科技大学出版社,1998.

[5] Paolucci M, Kawamura T, Payne T, et al. Semantic Matching of Web Services Capabilities[C]//First International Semantic Web Conference. Sardinia, Italy:[s.n.],2002.

[6] Srinivasan N, Paolucci M, Sycara K. Adding OWL - S to UDDI, implementation and throughput[C]//First International Workshop on Semantic Web Services and Web Process Composition (SWSWPC 2004). San Diego, California, USA:[s.n.],2004.

[7] Chao Kuo - Ming, Younas M, Lo Chi - Chun, et al. Fuzzy Matchmaking for Web Services[C]//Proceedings of 19 IEEE Conference on Advanced Network and Information Applications. [s.l.]:IEEE CS,2005:721 - 726.