UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

ANDRÉ LUÍS FÁVERO

Metodologia para Detecção de Incoerências entre Regras em Filtros de Pacotes

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação

Prof. Dr. Raul Fernando Weber
Orientador

Porto Alegre, janeiro de 2007.

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Fávero, André Luís

Metodologia para Detecção de Incoerências entre Regras em Filtros de Pacotes / André Luís Fávero – Porto Alegre: Programa de Pós-Graduação em Computação, 2007.

87 f.:il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação. Porto Alegre, BR – RS, 2007. Orientador: Dr. Raul Fernando Weber.

1. Firewall. 2. Filtro de pacotes. 3. Incoerências em regras. 4. Segurança da informação. I. Weber, Raul Fernando. II. Título.

AGRADECIMENTOS

Agradeço àquelas pessoas que me ajudaram durante o período de pesquisa e construção deste trabalho. Especialmente a minha amada Melissa, a pequena mascote Bianca, meus queridos pais, aos amigos sentinelas Serafim e Peres pelas idéias e apoio e ao meu orientador professor Raul Weber.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

DNS	Domain Name System
FDD	Firewall Decision Diagram
FTP	File Transfer Protocol
HTTP	Hyper Text Transfer Protocol
ICMP	Internet Control Message Protocol
ISP	Internet Service Provider
IP	Internet Protocol
NAT	Network Address Translation
OSI	Open Systems Interconnect
PERL	Practical Extraction and Report Language
QoS	Quality of Service
RBS	Rule-Based Software Systems
TCP	Transmission Control Protocol
UDP	User Datagram Protocol

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Embora *firewall* seja um assunto bastante discutido na área de segurança da informação, existem lacunas em termos de verificação de *firewalls*. Verificações de *firewalls* têm o intuito de garantir a correta implementação dos mecanismos de filtragem e podem ser realizadas em diferentes níveis: sintaxe das regras; conformidade com a política; e relacionamento entre as regras. Os aspectos referentes a erros de sintaxe das regras são geralmente tratados pela ferramenta de filtragem em uso. O segundo nível, no entanto, depende da existência de uma política formal e documentada, algo não encontrado na maioria das organizações, e de uma metodologia eficaz que, através da entrada da política de segurança e do conjunto de regras de *firewall* implementado, possa compará-las e apontar as discrepâncias lógicas entre a especificação (política) e a implementação (regras). O último, verificação dos relacionamentos entre regras, não requer qualquer documentação, uma vez que somente o conjunto de regras do *firewall* é necessário para a identificação de incoerências entre elas.

Baseado nessas considerações, este trabalho objetivou o estudo e a definição de uma metodologia para a análise de relacionamentos entre regras, apontando erros e possíveis falhas de configuração. Três metodologias já existentes foram estudadas, analisadas e utilizadas como base inicial para uma nova metodologia que atingisse os requisitos descritos neste trabalho.

Para garantir a efetividade da nova metodologia, uma ferramenta protótipo foi implementada e aplicada a três estudos de caso.

Palavras-Chave: *firewall*, filtros de pacotes, incoerências em regras, segurança da informação.

Methodology for incoherencies identification among packet filters rules

ABSTRACT

Although firewalls are a well discussed issue in the information security field, there are gaps in terms of firewall verification. Firewall verification is aimed at enforce the correct filtering mecanisms implementation and can be executed in three distinct levels: rules syntax, policy compliance and rules relationship. The aspects related to rule syntax errors are usually addressed by the filtering tool in use. However, the second level depends on the existance of a formalized and documented policy, something not usual in most organizations, and on an efficient metodology that, receiving the security policy and the firewall rules set as inputs, could compare them and point logical discrepancies between the specification (policy) and the implementation (rules set). The last level, rules relationship verification, doesn't require any previous documentation, indeed only the firewall rule set is required to conduct a process of rules incoherencies identification.

Based on those considerations, this work aimed at studying and defining a methodology for analysis of rules relationships, pointing errors and possible misconfigurations. Three already existent methodologies were studied, analyzed and used as a initial foundation to a new methodology that achieve the requirements described in this work.

To assure the effectivity of the new methodology, a prototype tool were implemented and applied to three case studies.

Keywords: firewall, packet filters, incoherence in rules, information security.

1 FIREWALLS: FILTROS, PROXYS E POLÍTICAS

Segurança, tradicionalmente, é um aspecto fundamental aos sistemas computacionais. Por essa razão, diferentes tipos de mecanismos para distintas necessidades e ambientes têm sido fatores importantes para a busca de características de segurança para estes sistemas.

A maioria das redes de computadores possui necessidades de segurança e de controle. Por exemplo, o acesso a determinado site na Internet não é permitido pela política de segurança da organização, ou ainda, para uma rede que tem acesso à Internet, não é necessário que as estações de trabalho dos usuários estejam acessíveis de qualquer ponto da Internet, principalmente pelo fato de esses computadores estarem sujeitos a ataques de vírus, *worms* e acessos não permitidos as informações destes usuários. Por atender a estas necessidades, é que os mecanismos de *firewall* são considerados peças fundamentais a qualquer rede.

Na maioria das vezes, *firewalls* e ambientes de *firewalls* são discutidos no contexto de conectividade com a Internet, entretanto, são mecanismos muito importantes para serem empregados também em ambientes de rede que não inclui conectividade com a Internet, mas conectividade com outras unidades da organização, clientes, fornecedores ou parceiros de negócio. Além disso, são importantes mecanismos para efetuar controle de acesso a áreas que possuem funções sensíveis dentro da organização, por exemplo, departamentos financeiros e de recursos humanos.

Um mecanismo de *firewall* pode ser definido como qualquer dispositivo, *software*, arranjo ou equipamento que controla a comunicação entre redes (CHESWICK, BELLOVIN, RUBIN, 2005). Existem diferentes categorias de mecanismos com este mesmo propósito, porém com diferentes capacidades de filtragem. Tal diferença é medida através do protocolo por que cada categoria atua.

Levando em consideração o modelo OSI, existem mecanismos que possuem capacidade para operar nas camadas de enlace (*layer* 2), rede (*layer* 3), transporte (*layer* 4) e aplicação (*layer* 7). Existem basicamente três categorias de mecanismos: filtros de pacotes, conhecidos também como listas de controle de acesso, muito comum em

roteadores e que operam basicamente nas camadas de enlace e rede; *Stateful Inspection Firewalls*, que são filtros de pacotes que adicionam, de forma consistente, os dados da camada de transporte (*layer* 4), acomodando determinadas características do protocolo TCP/IP, que dificultam a construção segura e adequada de *firewalls* que não contam com este tipo de característica; e *gateways* de aplicação, mais comumente chamados de *proxys* trabalham exclusivamente com a camada de aplicação (*layer* 7).

Ambiente de *firewall* é o termo utilizado para descrever todo o conjunto de mecanismos utilizados para prover as funcionalidades de controle de acesso e filtragem em um determinado ponto da rede. Um ambiente simples pode ser constituído somente por um filtro de pacotes. Entretanto, em ambientes mais complexos que necessitam de um grau maior de segurança, vários filtros de pacotes, tanto com características de *stateful* ou não, *proxys* e mudanças específicas na topologia são empregados para a construção do ambiente.

Para configuração de cada mecanismo, conjuntos específicos de regras são utilizados com o intuito de descrever a política de segurança da organização. Através da criação dessas regras, que levam em consideração os campos das camadas anteriormente citadas, o administrador de rede constrói a configuração do seu ambiente ou mecanismo de *firewall*.

Uma regra possui o formato de uma expressão "se condição, então ação", onde a condição é uma expressão do conjunto de campos que devem ser avaliados e ação corresponde à medida a ser tomada pelo *firewall* caso a condição seja satisfeita pelo pacote em questão. Na maioria dos mecanismos, uma ação será de aceite ou de rejeição para aquele pacote, indicando se este pode ou não seguir no fluxo de rede. Desta forma, para seguir a política de segurança da organização, o administrador configura seu mecanismo com indeterminado número de regras expressas, muitas vezes, das mais complexas formas, para que se consiga garantir que o ambiente implemente a política de segurança.

Ainda sobre o conjunto de regras, na maioria dos *firewalls* estas são analisadas pelo mecanismo de forma ordenada. Ou seja, dado uma conjunto de regras e um pacote a ser avaliado, o filtro percorre o conjunto de regras, da primeira à última regra, conforme a ordenação do conjunto até encontrar uma regra cuja expressão satisfaça a condição do pacote. Uma vez que a condição seja satisfeita, o *firewall* aplica a ação da regra que satisfez a condição do pacote, podendo ser um descarte, ou a permissão para que o pacote seja enviado para a próxima rede. As regras posteriores à regra satisfeita são ignoradas pelo mecanismo. Caso nenhuma das regras tenha condição satisfeita, então a política *default* do mecanismo é empregada.

A política *default* possui dois enfoques: pode ser permissiva ou proibitiva. No caso de permissiva, o filtro permite que qualquer pacote que não possa ser selecionado por

nenhuma das regras do conjunto possa seguir para o *host* ou rede de destino; no caso de proibitiva, o pacote é descartado.

1.1 Filtros de Pacotes

Os filtros de pacotes são dos mais antigos e eficientes mecanismos de segurança disponíveis para efetuar o controle de acesso a uma rede. Eles fazem a análise de informações localizadas no cabeçalho de cada pacote, mais especificamente nas camadas de enlace (*layer* 2) e rede (*layer* 3), além de considerarem também alguns campos da camada de transporte (*layer* 4), para determinar se deve ser permitida a entrada ou saída desse pacote para outra rede.

Alguns campos comumente utilizados para formar as expressões de regras são: informações do campo porta de origem e destino; campo ip de origem e destino; interface de entrada e interface de saída do pacote; e tipo de protocolo de transporte.

Por ser um mecanismo simples e ainda assim possuir boa performance e flexibilidade, filtros de pacotes são muito utilizado em roteadores de borda com redes inseguras, por exemplo, em roteadores de acesso à Internet, os quais estão ligados diretamente ao *Internet Service Provider* (ISP).

O funcionamento de um filtro de pacotes, como já anteriormente dito, se dá pela comparação de um conjunto de regras com cada pacote que passa pelo mecanismo. Portanto, o ordenamento das regras é um dos principais fatores para configuração adequada. Na tabela 1.1 é apresentado um exemplo de uma tabela simples de regras de um filtro de pacotes.

Filtros de pacotes também possuem fraquezas e vulnerabilidades. Como vulnerabilidade, o item mais crítico está na configuração do conjunto de regras. Mesmo com um pequeno número de regras, é possível que uma configuração inadequada exponha completamente a organização, pois mesmo acidentalmente é fácil configurar um filtro de pacotes para permitir determinados tipos de tráfego que deveriam ser bloqueados segundo a política de segurança da organização. Uma configuração inadequada pode também prejudicar a disponibilidade de um serviço, provocando que este seja bloqueado involuntariamente pelo administrador do filtro.

Quando uma aplicação cria uma conexão com um *host* remoto, uma porta dita alta é designada no cliente, chamada de porta de origem, com o propósito de ser utilizada para a comunicação com o *host* de destino. De acordo com as especificações do protocolo TCP/IP, a porta de origem deve ser maior que 1023 e menor que 65535. Ainda de acordo com a mesma especificação, serviços devem ser executados preferencialmente em portas de destino consideradas portas baixas, ou seja portas abaixo de 1024. Alguns exemplos bem conhecidos são: http na porta 80; smtp na 25; dns na 53; dentre muitos outros.

Outro aspecto extremamente relevante para a configuração de um filtro de pacotes, são as regras que controlam o retorno das conexões. Para muitas aplicações, principalmente as que utilizam protocolo orientado a conexão (protocolo TCP), o cliente espera um retorno do servidor ao qual solicitou o serviço. Para que isso aconteça, são necessárias regras permitindo tráfego da rede externa (ou da rede na qual está localizado o servidor) para rede interna (ou para a rede onde está o cliente, responsável pela solicitação do serviço). Na linha correspondente à regra 1 da tabela de regras 1.1, é apresentado um exemplo de regra específica ao retorno de conexões.

Principalmente pelo fato de necessitar regras de retorno, aliadas às especificações do TCP/IP, grandes brechas são expostas quando se faz a utilização de filtros de pacote sem características de *stateful*. O fato de o cliente utilizar uma porta alta e diferente a cada sessão da aplicação, obriga que o administrador permita a entrada de pacotes a todas portas altas dos clientes. Isso pode ser explorado tanto pelos clientes, oferecendo serviços que não seriam permitidos pela política de segurança como por atacantes, utilizando tal fraqueza para instalação serviços do tipo *backdoors,* por exemplo, pelos quais um atacante teria acesso garantido a estes *hosts*, passando sem ser bloqueado pelo *firewall*.

Além de pacotes contendo mensagens de retorno, outro tipo de tráfego que tem que ser levado em consideração no momento de configuração do filtro, são os pacotes que carregam mensagens de controle, pacotes do protocolo ICMP. Pacotes ICMPs que fornecem informações relevantes às aplicações e a roteadores, carregam mensagens de controle importantes, geradas, por exemplo, por roteadores ao longo do caminho percorrido pelo pacote. Um exemplo, seriam as mensagens de tipo *destination unreachable*, que indicam que o próximo *hop* da mensagem está inalcançável ou *port unreachable* que o serviço não se encontra disponível.

Tabela 1.1: Exemplo de conjunto de regras de um filtro de pacotes

	Endereço de origem	porta de origem	endereço de destino	porta de destino	ação	descrição
1	any	any	192.168.1.0/24	> 1023	allow	Regra para permitir retorno de conexões TCP para sub-rede interna
2	192.168.1.1	any	any	any	deny	Regra para prever que o próprio *firewall* não posa se conectar a nenhum lugar
3	any	any	192.168.1.1	any	deny	Proíbe acesso de usuários externos diretamente ao *firewall*
4	192.168.1.0/24	any	any	any	allow	Usuários internos podem acessar servidores na rede externa
5	any	any	192.168.1.2	smtp	allow	Usuários externos podem enviar e-mails
6	any	any	192.168.1.3	http	allow	Usuários externos podem acessar servidor www
7		any	any	any	deny	Tudo que não for previamente permitido é bloqueado.

1.2 Filtros *Stateful*

Firewalls Stateful são basicamente filtros de pacotes, entretanto com aprimoramentos que visam resolver algumas fraquezas apresentadas no modelo anterior. Tal aprimoramento se dá principalmente por incorporar completamente a camada *layer* 4 do modelo OSI, bem como as camadas abaixo. Com isso, as fraquezas do modelo anterior referentes ao controle de pacotes de retorno são tratadas de modo que se possa se considerar que *firewalls* do tipo *stateful* são mais seguros. As vulnerabilidades existentes nos filtros de pacotes devido a erros de configuração continuam existindo da mesma forma neste modelo mais aprimorado.

Ter que adicionar regras de retorno, permintido qualquer tráfego em portas altas dos clientes, é realmente muito abrangente e ocasiona problemas de segurança passíveis de serem explorados por usuários maliciosos, se feitas como apresentado na tabela 1.1 que exemplifica um conjunto de regras. Para resolver esta fraqueza, filtros do tipo *stateful*

adicionam uma tabela de estados à sua arquitetura. Para cada nova conexão aceita pelo filtro, este repassa o pacote e também armazena as informações referentes àquela sessão (ip de origem e destino, portas de origem e destino, *flags*, números de seqüência, entre outros) (NORTHCUTT, 2002).

Basicamente, os principais estados utilizados são NEW, ESTABLISHED e RELATED. NEW indica o início de uma nova sessão. No protocolo TCP, o estado NEW ocorre durante o envio de um pacote com *flag* SYN pelo cliente a um servidor em estado *listening*, ou seja, durante o estado SYN_SEND do TCP. Pacotes de início de conexão são tratados por todo conjunto de regras até encontrar uma que satisfaça a condição, ou até ser enquadrado pela política *default*. ESTABLISHED refere-se a conexões já estabelecidas e RELATED a pacotes que são relacionados a conexões já estabelecidas, por exemplo, mensagens ICMP.

Os filtros de pacotes tipo *stateful* adicionam a possibilidade de se utilizar em regras que verificam o estado da sessão, por exemplo:

Tabela 1.2: Regra para controle de estado

	Endereço de origem	porta de origem	endereço de destino	porta de destino	ação	estado	descrição
1	any	any	192.168.1.0/24	any	allow	ESTABLISHED, RELATED	Regra para permitir retorno de sessões iniciadas pela sub-rede interna

A regra da tabela 1.2 indica ao filtro que este deve verificar em sua tabela de estados se existe uma conexão estabelecida para o pacote em questão, ou se o pacote é relacionado a uma comunicação já estabelecida e aprovada pelo filtro. Uma vez que a sessão possa ser identificada na tabela de estados, as demais regras no filtro não são mais checadas, aumentando então a performance do *firewall* para conexões já estabelecidas, se a regra de controle de estado se encontrar no início da tabela de regras.

O fato de os protocolos de transporte (*layer* 4 do OSI) serem diferentes possibilita que existam particularidades em cada tipo de protocolo, para que seus estados possam efetivamente ser utilizados. Tais particularidades também são próprias a cada implementação de mecanismo. Particularidades no nível de aplicação também devem ser levadas em consideração nas implementações.

1.2.1 Funcionamento de *Stateful firewalls* com protocolo TCP

Como TCP é um protocolo orientado à conexão, o estado das sessões de comunicação é bem definido. Identificar o início e fim de comunicação entre as sessões, é relativamente fácil, uma vez que o protocolo possui *flags* utilizados pela máquina de estados TCP e é considerado um protocolo *stateful*. Portanto, o estado de uma comunicação TCP é determinado pela checagem dos *flags* e pelos números de seqüência carregados pelo pacote, juntamente com a combinação de ip, porta de origem e destino. Uma vez, que ocorra o TCP *three-way handshake*, a conexão é considerada ESTABLISHED. Para sessões de comunicação utilizando TCP, pacotes que se enquadram no estado tipo RELATED também são comuns, como o recebimento de mensagens ICMP tipo *host unreachable,* quando da tentativa de estabelecimento de uma conexão TCP.

Entradas na tabela de estados são removidas quando a sessão de comunicação é fechada. Para prevenir que conexões fechadas de forma imprópria não permaneçam por tempo indeterminado na tabela de estados, temporizadores são utilizados. O valor inicial para *timeout* utilizado é relativamente curto (perto de um minuto) e posteriormente alongado (geralmente uma hora ou mais) para conexões estabelecidas.

1.2.2 Funcionamento de *Stateful firewalls* com protocolo UDP

Diferentemente do protocolo TCP, o protocolo de transporte UDP não é orientado à conexão. Os datagramas enviados não necessariamente requerem retorno, como em conexões de *streaming* multimídia, por exemplo. Não possui números de seqüência e nem *flags* de estado e, por estes fatores definir o estado de uma comunicação UDP não é um processo tão simples. Entretanto, filtros *stateful* devem conseguir traçar, ao menos, um "pseudo-estado" de uma transmissão UDP.

Como não há informações de *flags* ou números de seqüência, as informações que podem e são utilizadas para sinalizar uma sessão do tipo ESTABLISHED são ips de origem e destino, bem como portas de origem e destino. Não há, neste protocolo, a indicação bem definida como no TCP, indicando o fim da transmissão. Portanto, o dispositivo *stateful* geralmente utiliza, para remover a informação de tal sessão, um valor pré-configurado para *timeout* (geralmente um minuto ou menos), prevenindo então que a informação de um tráfego UDP permaneça de forma imprópria e indeterminada na tabela de estados.

Um recurso também muito utilizado pelo protocolo UDP são mensagens de controle do tipo ICMP e, portanto, perceber pacotes tipo RELATED também é importante para esse protocolo.

1.2.3 Funcionamento de *Stateful firewalls* com protocolo ICMP

O protocolo ICMP, também como UDP, não é do tipo *stateful*, mas, como UPD, também possui atributos que permitem que as conexões sejam acompanhadas. Como dito

anteriormente, com freqüencia é utilizado para retornar mensagens de controle ou erro, quando o *host* ou protocolo em questão não pode fazê-lo, ou para aplicações do tipo *echo request ou echo reply*.

O mais conhecido exemplo de *echo request* e *echo reply* são as mensagens utilizadas pelo programa *ping*. Uma mensagem deste tipo é útil para identificar se um determinado *host* está apto. Para mensagens do tipo *echo request* ou *echo reply*, traçar o estado da sessão pode ser relativamente mais fácil se comparado a situações em que a mensagem seja do tipo *one-way.*

Assim como UDP, o tempo de permanência na tabela de estados é definido via *time-out*, uma vez que este protocolo também não possui um mecanismo específico para fim da sessão.

1.2.4 Estado em nível de aplicação

Apesar de aparentemente resolvidos todos os problemas utilizando uma tabela de estados, quando se sobe para a camada de aplicação, percebe-se que alguns protocolos possuem regras diferentes das esperadas para comunicação. O protocolo HTTP, por exemplo, segue o estilo de comunicação esperado para aplicações que utilizam como protocolo de transporte TCP. Uma sessão TCP é aberta entre o servidor www e o *browser* do cliente e os comando HTTP, bem como as respostas carregando o conteúdo solicitado pelo cliente seguem o fluxo dentro da sessão de comunicação estabelecida.

Já o protocolo File Transfer Protocol (FTP), que fornece troca de arquivos, exemplifica claramente o problema de particularidades a algumas aplicações, apesar de utilizar TCP como protocolo de transporte, que é do tipo *stateful*. Na tabela 1.3, é demonstrado um exemplo de comunicação entre um cliente e servidor ftp;

Tabela 1.3: Exemplo de comunicação utilizada pelo protocolo ftp

Cliente estabelecendo conexão com servidor (conexão sainte):

```
    16:39:16.401247 IP cliente.39677 > servidor.21: S 2223604035:2223604035(0)
    16:39:16.622829 IP servidor.21 > cliente.39677: S 2880513536:2880513536(0) ack
2223604036
    16:39:16.623790 IP cliente.39677 > servidor.21: . ack 1
```

Após estabelecida conexão, um canal de dados é estabelecido através de uma nova conexão iniciada do lado do servidor (conexão entrante):

```
    16:42:31.019082 IP servidor.20 > cliente.32779: S 3081621326:3081621326(0)
    16:42:31.020121 IP cliente.32779 > servidor.20: S 2412102954:2412102954(0) ack
3081621327
    16:42:31.240087 IP servidor.20 > cliente.32779: . ack 1
```

A primeira parte da comunicação é um *three-way handshake* normal. O cliente utilizando uma porta alta inicia uma sessão com o *host* remoto na porta 21. Entretanto,

após a conexão estabelecida, o servidor também inicia uma conexão para o cliente, utilizando a porta 20, diferente da que originalmente é utilizada. O cliente recebe a conexão em uma porta diferente da originalmente utilizada por ele. Pelo fato de o servidor iniciar um novo canal, utilizando diferentes portas, este não é considerado um tráfego de retorno. O comportamento, neste caso, do protocolo FTP é diferente do esperado dos protocolos carregados por TCP.

Firewalls stateful devem tratar estas particularidades de forma que este tráfego possa ser permitido, mas de forma segura. A sessão de controle FTP é estabelecida conforme especificação, a inicialização do canal de dados é que se torna o problema. Entretanto o *firewall stateful* deverá saber que, após um cliente iniciar uma sessão de controle FTP, utilizando TCP porta de destino 21, o servidor irá iniciar o canal de dados utilizando TCP, porta de origem 20, em uma porta alta do cliente. O *firewall*, então, deverá permitir a entrada de uma conexão vinda com ip de origem do servidor e porta de origem 20, com destino ao ip do cliente.

1.2.5 *Proxys*

Proxys ou *gateways* de aplicações trabalham na camada de aplicação do TCP/IP (*layer* 7 do OSI). Diferentemente dos filtros de pacotes, independente de estes controlarem estado ou não, este tipo de *firewall* não necessita uma interface de entrada e outra de saída para fazer o roteamento dos pacotes.

Outra característica deste tipo de mecanismo é o fato de ele propiciar um controle mais em nível de usuário, uma vez que o mecanismo atua na camada da aplicação e tem acesso a esse tipo de informação, diferentemente dos outros modelos apresentados. Essa característica oferece um nível mais aprofundado de *logs*. Além de registrar informações tradicionais como *host* de origem e destino, também é possível registrar os comandos relativos à nível da aplicação.

1.3 Relação entre Política de Segurança e *Firewalls*

A configuração de um *firewall* se dá pela escrita de suas regras; estas regras, por sua vez, descrevem o que é permitido ou não em termos de transmissões. Para a implementação de um conjunto de regras, o administrador deve avaliar e seguir a política de segurança da organização. É neste documento que são descritas as situações permitidas ou ações que deverão ser tomadas em ocasiões de ocorrências de incidentes de segurança. A política de segurança deve ser utilizada para gerar a uma política mais específica, voltada aos mecanismos de *firewall* (WACK, CUTLER, 2002).

Muitas referências apontam que o primeiro passo para configuração de um *firewall* é ter em mãos a política de segurança da organização. De fato, tal afirmação é realmente muito importante, pois, sem a política, o administrador não tem uma base para guiar quais regras devem ser empregadas e o mecanismo, na verdade, torna-se um

problema, bloqueando, por muitas vezes, a disponibilidade de aplicações importantes e, por outras, permitindo a possibilidade de ações maliciosas.

Para se construir a política do *firewall,* a política de segurança que se encontra descrita em uma linguagem de alto nível deve ser interpretada pelo administrador e transcrita para um formato, no qual fica visível o modo como o *firewall* deverá tratar cada aplicação. Ainda, esta política deverá deixar claro como o *firewall* está implementado e é mantido. Ainda segundo Wack, alguns passos deverão ser seguidos para a criação desta política. Estes passos são os seguintes:

- identificação das aplicações que utilizem rede;
- identificação das vulnerabilidades associadas a cada aplicação levantada;
- análise de custo benefício dos métodos que serão utilizados para garantir a segurança das aplicações;
- criação de uma matriz de aplicações indicando métodos de proteção e outras informações, como mostrado na tabela 3.3; e
- criação das regras com base na matriz de aplicações.

Tabela 1.4: Matriz de aplicações

Aplicação ou serviço TCP/IP	tipo do host interno	política de segurança no host	política de segurança do firewall para rede interna	política de segurança do firewall para rede externa
FTP	Unix	não aceita usuário anônimo, usuário e senha, SSH	permitido	permitido
NetBIOS	Windows	limita o acesso aos compartilhamentos	permitidos somente ao domínio local	negado
Telnet	Roteadores	autenticação com dois níveis de senhas	permitido a partir da máquina do administrador de sistemas	negado

Fonte: WACK, CUTLER, 2002, p. 34

2 ESTADO DA ARTE EM MODELOS DE VERIFICAÇÃO DE CONSISTÊNCIA

O processo de configuração de um ambiente de *firewall* é de fundamental importância para se garantir que as decisões tomadas na política de segurança, sejam efetivamente implementadas; uma falha nesta fase e todo investimento pode ser comprometido, pois uma definição equivocada, mesmo que certa para algum tipo de situação, pode levar a outras não pretendidas pelo administrador e, com isso, uma falsa sensação de segurança. Como dito anteriormente, o processo de escrita das regras pode também ser encarado como processo de configuração de um filtro de pacotes.

A tarefa de configuração desempenhada pelo administrador geralmente é manual, via interface gráfica ou não, mas requer conhecimento prévio das regras já existentes, bem como conhecimentos específicos sobre a sintaxe ou interface utilizada pelo filtro. Uma característica pretendida então para os gerenciadores de filtros de pacotes, é que a escrita destas regras seja uma tarefa simples.

Podem existir três situações que levem a erros de configuração (erros na escrita das regras). A primeira é com relação a falhas de sintaxe para configuração das regras, a segunda é referente a falhas na política de segurança quando por exemplo, o administrador decide equivocadamente e deixa passar alguns tipos de pacotes que deveriam ser bloqueados, gerando uma especificação errada. E, finalmente, a terceira é quando a escrita leva a uma situação adversa à prevista pela política. Esta é a principal fonte de falhas em *firewalls* (CHESWICK, BELLOVIN, RUBIN, 2005).

Atualmente, filtros de pacote já são tecnologias bem estabelecidas e realmente usadas em grande escala. Realizam análise sintática das tabelas de regras, entretanto não realizam nenhuma verificação da semântica das regras que o administrador define. Por isso, é muito importante estabelecer uma metodologia para que tal tarefa passe a ser incluída em novos arranjos de *firewalls,* possibilitando ao administrador procurar, em seu conjunto de regras eventuais, equívocos que levem desde a pequenos problemas até a completa inutilidade de seus filtros de pacotes.

O funcionamento desejado de um filtro de pacotes depende muito da coerência do conjunto de regas. Manter esta coerência pode ser considerado fácil de se fazer quando a tabela de regras possui um pequeno número de registros, entretanto, nem sempre é assim. É muito comum um filtro de pacotes possuir centenas de regras (CHESWICK, BELLOVIN, RUBIN 2005). Também é freqüente a utilização de mais de um filtro de pacotes na topologia da mesma rede, sendo que o primeiro pode interferir diretamente no fluxo de pacotes para o segundo.

Outro aspecto de fundamental importância está ligado a forma dinâmica em que novos serviços ou novas máquinas são agregadas ou retiradas da rede e, por isso, há necessidade de atualizações freqüentes destas regras. Atualizações também são freqüentes e da mesma forma importantes, para que se efetue bloqueios de novas vulnerabilidades, bem como novos ajustes da política implementada. Unindo todos estes aspectos, a gerência de um filtro de pacotes torna-se algo complexo, onde regras podem conflitar, possibilitando falhas de configuração em filtros de pacotes.

A análise de regras de um filtro pode revelar situações em que as mesmas não estão perfeitamente adequadas à política desejada. Existem três situações bem definidas:

- regras erradas: que vão contra a política definida;
- regras que podem ser conflitantes: regras que definem tratamentos diferentes, ou mesmo contrários, para um mesmo conjunto de condições;
- regras mal posicionadas: regras que podem ser reposicionadas com relação às demais, visando uma eventual melhora do desempenho do filtro. Um rearranjo simples seria fazer com que as regras mais utilizadas fossem as primeiras a constarem no filtro, desde que a política seja mantida.

Objetivando estabelecer uma forma eficiente para checagem de regras, este trabalho abordará, nos próximos itens, trabalhos que tenham como preocupação este mesmo propósito, visando apresentar as metodologias existentes.

2.1 Projeto de *firewalls* através de diversidade

Com o objetivo de aumentar o nível de segurança dos mecanismos de filtragem de pacotes, Liu propõe uma metodologia para construção correta e checagem de inconsistências na configuração de filtros. Um *firewall* pode ser considerado em estado correto, se a política encontra-se correta e está implementada de forma a satisfazer os requisitos da especificação, que geralmente está em uma linguagem natural (LIU, 2004).

Para este modelo, a categoria de erros está dividida em erros na especificação da política ou no projeto das regras. Os erros na especificação são decorrentes das ambigüidades inerentes aos requisitos informais da especificação, especialmente se esta é escrita em uma linguagem natural. Já os erros de projeto são causados por incapacidades do administrador do sistema, responsável pela construção das regras. Segundo Liu,

existem administradores com diferentes entendimentos sobre algum ponto da especificação, bem como com diferentes níveis de conhecimentos. Esta observação é o princípio para a metodologia de projetos diversitários de *firewalls* (LIU, 2004-b).

Este modelo baseia-se muito nos projetos diversitários para construção de *hardware* ou *software* da área de tolerância a falhas, mais especificamente, o conceito de programação de N-versões (AVEZIENES, 1977, 1985, 1995). A idéia básica da metodologia de N-versões é entregar a mesma especificação a N equipes para que estas desenvolvam N programas independentes, utilizando diferentes algoritmos, ferramentas e linguagens. A partir da execução em paralelo dos N programas desenvolvidos, um mecanismo de decisão é desenvolvido e utilizado para examinar os N resultados para cada entrada de N programas e, então, selecionar o resultado correto. O elemento principal da metodologia de N-versões é o desenvolvimento diversitário.

O principal motivo para a ocorrência de erros na especificação de configuração de *firewalls* apontado por Liu está no método utilizado para a especificação de regras, a qual é feita diretamente, regra a regra, na configuração do filtro. Esta forma desestruturada gera três características principais no conjunto de regras: falta de consistência, completude, além de um conjunto não compacto de regras.

O problema de consistência é principalmente provocado pelo ordenamento inadequado das regras; quanto à completude, Liu refere-se ao fato de garantir que a especificação seja correta e completamente escrita nas regras, fato que também é difícil de garantir. E, por fim, é comum que o conjunto de regras resultantes possuam redundâncias ou regras que poderiam ser especificadas diferentemente, de modo a gerar um conjunto final mais claro e compacto de regras. Falta de consistência e completude geram erros na configuração do filtro. Já redundâncias acarretam principalmente problemas de performance do filtro.

A metodologia apresentada por Liu, visando resolver os problemas apresentados, consiste em duas grandes fases: uma fase de projeto e uma fase de comparação. Para a fase de projeto, a mesma especificação é fornecida a diferentes equipes de projeto. Estas, de forma independente, produzem o conjunto necessário de regras para a representação da política. É interessante que as equipes possuam diferentes métodos para o desenvolvimento do projeto, buscando assim reduzir a possibilidade de que não se produzam os mesmos erros pelas diferentes equipes. As regras devem ser formalmente especificadas, utilizando-se um modelo definido por Liu, chamado *Firewall Decision Diagram* (FDD) (LIU, 2004-a).

Na fase de comparação, a qual inicia somente após o termino da fase de projeto, os resultados produzidos pelas diferentes equipes são comparados, buscando se encontrar discrepâncias entre estes projetos. Se elas existirem serão investigadas e correções poderão ser aplicadas. Após a fase de comparação e das correções serem aplicadas, caso exista essa necessidade, todas as versões passam a ser equivalentes.

2.2 Modelo de Wang

Wang propõe outra abordagem ao considerar que existam conflitos em sistemas do tipo *Rule-Based Software Systems* (RBS), e classifica os filtros de pacotes como sendo sistemas tipo RBS (WANG, HAO, LEE, 2003). Para detecção de regras conflitantes, propõe um algoritmo de duas-fases, onde primeiramente as regras são normalizadas e, numa segunda fase, são detectados os conflitos. O modelo apenas identifica a existência de conflitos entre regras, sem, no entanto, indicar a existência de um erro propriamente dito.

A seguinte definição é utilizada para regras conflitantes, supondo que todas as regras de uma tabela possam ser convertidas no seguinte formato:

Se (CONDIÇÃO) **então** (AÇÃO)

Como estas regras deverão ser comparadas entre elas, e cada regra possui N partes, então:

R_i : Se (CONDIÇÃO) **então** (AÇÃO) $1 < i < N$

Podem ser definidas como regras conflitantes, duas regras R_i e R_j que possuam ações diferentes e partes da condição levem as regras a serem satisfeitas.

A normalização das regras tem por objetivo tratar tabelas de regras cujo formato não seja ***"if-then".*** Existem filtros de pacotes cuja sintaxe é: **Se** (CONDIÇÃO) **então** (AÇÃO) **senão** (outra AÇÃO), que resultam em um formato ***"if-then-else"***. Este formato agrega a possibilidade de uso de uma nova corrente de regras (visível no *netfilter*, por exemplo). Este formato não pode ser checado pelo algoritmo, por isso a fase de normalização de regras para o formato "*if-then*"

Após a normalização do conjunto de regras, a segunda fase do algoritmo é detectar as que são conflitantes, para isso, primeiramente são checadas as ações de cada par de regras. Se existirem ações conflitantes, então são checadas se as intersecções das expressões condicionais são satisfeitas, caso sejam, então um conflito foi encontrado.

Duas suposições importantes são assumidas pelos autores desse algoritmo, a primeira é referente às ações, só pode existir uma ação por regra; por exemplo, *discart* ou *accept* e estas são conflitantes. A segunda suposição é referente às expressões condicionais de cada regra, esta assume que existe um número limitado de variáveis em cada expressão, os valores possíveis para cada variável também são limitados a um conjunto de números inteiros finitos, contínuos e ordenados. As variáveis são os campos do cabeçalho do protocolo TCP/IP a serem checados. Estas suposições são fundamentais para a segunda fase do algoritmo.

A segunda fase do algoritmo tem como propósito a detecção de regras conflitantes. O primeiro teste para cada par de regras é se este possui ações que sejam conflitantes, se existirem, então são verificadas intersecções entre as duas expressões

condicionais que sejam satisfeitas, se existir, um conflito foi encontrado. Para avaliar a existência de uma intersecção, é verificada a seguinte condição:

- dado um par de regras, para cada variável da expressão condicional, é verificado se o intervalo de valores da primeira regra pertence ao intervalo de valores da segunda regra e *vice-versa*. Se a essa condição for verdadeira, existe uma intersecção.

Portanto, para o modelo de Wang, se duas regras tiverem um único campo sendo uma intersecção, então é indicada a existência de possível conflito.

Tabela 2.1: Algoritmo para checagem de correlações

Entrada: duas condições C1e C2 expressas em forma disjunta
Saída: um valor booleano que indica ocorrência de intersecção entre C1 e C2

```
1 for each clause in C1
2   for each clause in C2
3     for each dimension in set { da, sa, dp, sp, tp} do
4       if both clauses have single interval in current dimension
5         check if these two single interval have intersection
6            if yes, return TRUE
7 return FALSE
```

2.3 Modelo de Al-Shaer

O método de Al-Shaer também apresenta o problema de configuração de regras. Esta proposta considera a análise a partir de um legado de regras, diferentemente do modelo de Liu (2004-b), que parte do modelo trata da especificação formal do conjunto de regras. A partir da análise de regras, o modelo de Al-Shaer considera que existam regras que tratam de forma incoerente o mesmo tráfego e as classifica como falhas, que são anomalias na política do *firewall* e, para o bom funcionamento deste, elas deverão ser detectadas e informadas ao administrador, cabe a este a análise e possível resolução (AL-SHAER, HAMED, 2003).

Para a detecção de regras anômalas, são estabelecidas possíveis relações entre as regras de um filtro e, a partir deste modelo, cria-se uma classificação de tipos de falhas que serão avaliadas mediante a tabela de regras.

O modelo considera somente a utilização de cinco campos para a construção de condição de regra: protocolo, ip de origem e destino, porta de origem e destino. Demais campos como, interface de entrada e saída, dentre outros possíveis de serem avaliados pela maioria dos filtros de pacotes, são considerados, neste modelo, como de uso ocasional e, por isso, não são tratados pelo método.

Com a intenção de fornecer uma visualização simples de toda política de *firewall* e através desta identificar de maneira fácil e compreensível relações e anomalias entre estas regras, pode ser criada uma representação gráfica através de uma árvore, a qual é chamada de árvore de políticas. Para a construção da árvore, cada nodo representa um campo e cada galho representa um possível valor associado a este campo. No nodo folha, é descrita a ação para a regra. Percorrendo-se o caminho do nodo raiz ao folha, é possível recuperar a regra. Regras que têm o mesmo valor para determinado campo compartilham o mesmo nodo para representação deste valor na árvore, a representação é relativamente simples e realmente fornece uma visão de regras correlacionadas (AL-SHAER, HAMED, 2003).

2.3.1 Formalização das relações entre regras

As seguintes definições entre as regras são estabelecidas, visando uma classificação de possíveis falhas na configuração. Todas as relações são destintas e, segundo os autores do método, são as únicas relações possíveis. Os teoremas utilizados para garantir essas definições são os seguintes:

- teorema 1: cada definição é distinta, ou seja, qualquer duas regras somente poderão ser relacionadas por somente uma definição;
- teorema 2: a união de todas as relações representam o conjunto de relações entre qualquer duas regras em uma política de *firewall*.

Supondo a comparação de pares de regras, onde uma é representada como RX e outra RY, Al-Shaer apresenta as seguintes definições:

- Completamente disjuntas: as regras RX e RY não se relacionam se todos os campos em RX não forem subconjuntos, superconjuntos ou iguais aos correspondentes campos de RY;
- Iguais: todos os campos em RX são iguais aos correspondentes em RY.
- Subconjuntos: A regra RX não é igual a RY e todos os campos da regra RY são subconjuntos ou iguais aos campos correspondentes em a regra RY.
- Parcialmente disjuntas: as regras RX e RY serão parcialmente disjuntas, se existir pelo menos um campo em RX que é um subconjunto, ou um superconjunto, ou igual ao campo correspondente em RY e se existir pelo menos um campo em RX que não seja um subconjunto, nem um superconjunto e também não seja igual ao campo correspondente em RY.
- Correlacionadas: Irá existir uma correlação entre regas RX e RY se algum campo em RX for um subconjunto ou igual ao campo correspondente em RY e o resto dos campos em RX são superconjuntos de campos correspondentes em RY.

2.3.2 Classificação de Anomalias

Uma anomalia nas regras do filtro é definida pela existência de duas ou mais regras que tratam o mesmo pacote. Com base nas definições anteriores, é possível se identificar diferentes tipos de anomalias, algumas indicam erros de configuração possíveis a qualquer filtro de pacotes. Entretanto, algumas das anomalias são caracterizadas como prováveis erros e, nesta situação, o modelo visa indicar ao administrador esta ocorrência.

O modelo apresenta a seguinte classificação de anomalias:

2.3.2.1 Anomalia de Sobreposição

Ocorre quando uma regra que precede a regra atual tem condição validada com todos os pacotes que teriam condição validada com esta regra, o que resulta em uma não utilização da regra atual. Este tipo de anomalia é considerado um erro, visto que uma determinada ação da política descrita por esta regra nunca será executada, portanto, uma regra RY será sobreposta por RX se RY é posterior a RX e RY é um subconjunto de RX e se as duas regras apresentarem ações diferentes.

2.3.2.2 Anomalia de Correlação

Existirá uma anomalia de correlação se a primeira regra tem condição validada com alguns pacotes que também teriam condição validada com a segunda regra. A segunda regra também tem condição validada com alguns pacotes que também têm condição validada com a primeira regra. A regra RX e a regra RY são correlacionadas se RX e RY apresentarem uma relação de correlação e se as ações de cada uma das regras forem diferentes.

Observando o exemplo da tabela , onde as regras visam controlar o tráfego HTTP, a regra 1 bloqueia qualquer conexão que seja originada da rede 192.168.0.0/16 a este serviço, entretanto, na regra 2, este tipo de tráfego é liberado. Esta situação pode realmente ocorrer, mas, se houver uma inversão na ordem das duas regras, a política sofre alteração, permitindo acesso ao serviço na porta 80 destinado ao servidor 10.0.0.1 a partir de qualquer origem. Tráfego originado na rede 192.168.0.0/16 com destino a *hosts* diferentes do 10.0.0.1 não seria permitido.

Tabela 2.2: Exemplo de controle ao serviço http (porta de destino 80)

	Endereço de origem	porta de origem	endereço de destino	porta de destino	proto	ação
1	192.168.0.0/16	any	any	80	any	deny
2	any	any	10.0.0.1	80	any	allow

Uma correlação pode não ser um erro, então para este tipo de situação é interessante avisar ao administrador sobre esta correlação, e este deve decidir se a política encontra-se correta.

2.3.2.3 Anomalia de Generalização

Uma regra é uma generalização de outra regra, se tem condição validada com todos os pacotes que teriam condição validada numa regra que a precede. RY é uma generalização da regra RX, se RY for posterior à RX e RY for um superconjunto para RX e as ações das duas regras forem diferentes. Novamente como na anomalia de correlação, invertendo-se a ordem das regras a política é alterada. Este modelo de anomalia não é considerado um erro, mas, segundo o modelo, o administrador deve ser avisado sobre esta generalização, e decidir se a política está correta ou não.

2.3.2.4 Anomalia de Redundância

Regras redundantes existem, quando duas regras executam a mesma ação para o mesmo conjunto de pacotes, mesmo sendo removida, este tipo de regra não altera a política de segurança. Mesmo não alterando a política, redundâncias são consideradas erros visto que não contribuem com as decisões de filtragem e tornam a tabela de regras maior o que eleva a complexidade e o tempo que o filtro de pacotes dispõe para a checagem das regras. Uma regra RY é redundante para RX se RX precede RY e RY é um subconjunto ou igual a RX e as ações são similares. Se RX precede RY e RX é um subconjunto de RY e ações de RX e RY são similares, então a regra RX é redundante para regra RY, provado que RX não esteja envolvido com qualquer anomalia de generalização ou correlação com outras regras que precedem RY.

2.3.2.5 Anomalia de irrelevância

Em outro documento, Al-Shaer classifica também uma quinta anomalia, que diz respeito a regras irrelevantes para aquele contexto de rede em que o filtro de pacotes se encontra. Uma regra será irrelevante se nunca puder ter condição validada com qualquer tráfego que passa através deste *firewall*. Existirá uma regra do tipo irrelevante se os campos ips de origem e destino daquela regra contiverem ips fora do domínio deste *firewall*, ou seja, se o tráfego para o caminho de origem ao destino não passar através deste filtro de pacotes (AL-SHAER, HAMED, 2004).

Este tipo de anomalia é um erro, visto que acresce de forma desnecessária o processo de filtragem, entretanto, para ser possível classificar uma anomalia de irrelevância, é necessário que o modelo tenha, como entradas, informações sobre a topologia em questão, não sendo possível verificar este tipo de anomalia, somente através da análise das regras.

2.3.3 Processo de detecção de anomalias

O processo de descoberta de anomalias tem por objetivo determinar se duas regras quaisquer da política coincidem sobre os mesmos pacotes. Para isso uma série de comparações para cada par de regras são executadas e, ao final, pode-se apontar se as duas regras coincidem e assim existe a detecção de uma potencial anomalia.

Para facilitar a compreensão do modelo de descoberta de anomalias, é utilizado um diagrama de estados, figura 2.1. Para simplificar a explanação sobre o modelo, foram subtraídos alguns estados, sendo que os campos de ip de origem e porta de origem foram integrados em um único estado, o mesmo ocorrendo com os campos de ip de destino e porta de destino (AL-SHAER, HAMED, 2004).

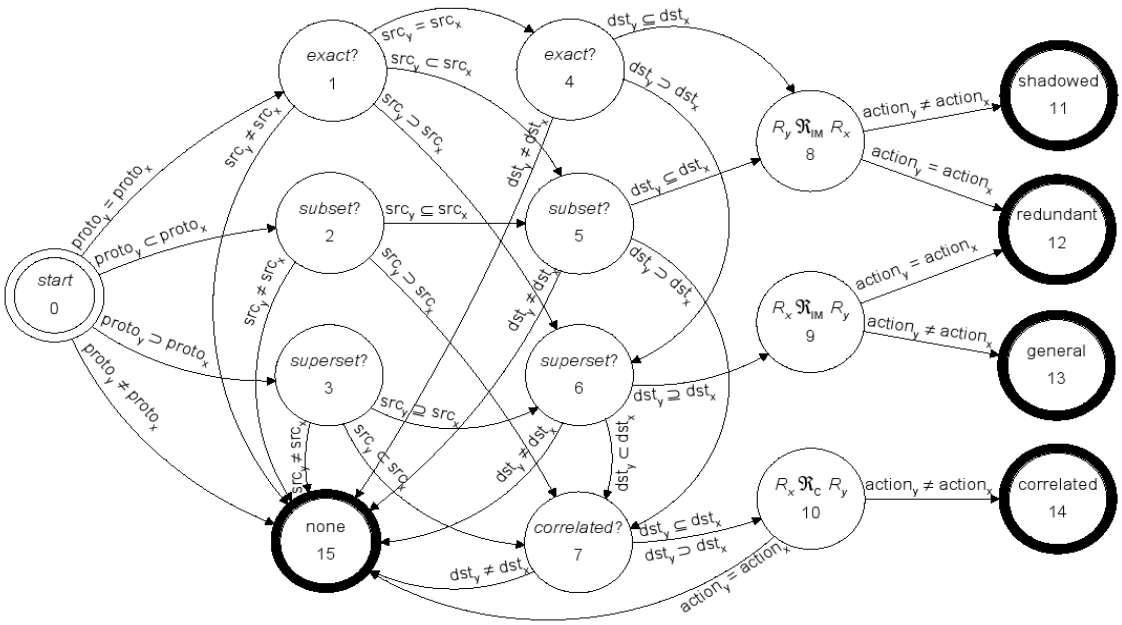


Figura 2.1: Diagrama de estados para detecção de anomalias entre duas regras RX e RY (AL-SHAER, HAMED, 2004)

Inicialmente nenhuma relação é assumida, cada campo da regra RY é comparado com o campo correspondente da regra RX. A ordem de comparação de campos é protocolo, ip de origem, porta de origem, ip de destino, porta de destino. A relação entre duas regras é determinada através das subseqüentes comparações.

Para determinar a classificação da anomalia, as relações são avaliadas, se todos os campos de RY forem subconjuntos ou iguais aos correspondentes em RX e se ambas as regras possuem as mesmas ações, então RY é redundante a RX, mas, se as ações forem diferentes, então RY é sombreado por RX.

Se todos campos de RY forem subconjuntos ou iguais ao correspondente campo em RX, e ambas as regras tiverem a mesma ação, então RX é potencialmente redundante

a RY, enquanto que, se as ações forem diferentes, RY é uma generalização de RX. Apesar de considerar na descrição da anomalia de redundância que uma regra RX é redundante a RY, se provado que RX não está envolvido em anomalias de generalização ou correlação, o modelo não leva em consideração tal requisito, tanto no algoritmo de detecção de anomalias como também na utilização do diagrama de estados.

Se alguns campos de RX forem subconjuntos ou iguais aos campos correspondentes em RY e alguns campos de RX forem superconjuntos aos correspondentes campos em RY e as ações forem diferentes, então RX está correlacionado a RY. Se nenhuma destas situações ocorrerem, então as duas regras não possuem anomalias e o modelo passa a verificar outras regras.

3 NETFILTER/IPTABLES

Netfilter/IPTables é o nome de um *firewall* do tipo *stateful* disponível a sistemas operacionais Linux. O *software* é livre e parte, n*etfilter*, encontra-se em espaço de *kernel* e parte é utilizado em espaço de usuário, *iptables*. O *iptables* é o programa responsável pela manipulação da tabela de regras, enquanto o *netfilter* faz a ponte entre o *kernel* e a estrutura de regras. Nesta pesquisa, utilizar-se-á o termo *iptables* para designar a ferramenta *netfiler/iptables*.

3.1 Arquitetura do *NetFilter/IPTables*

A parte do *netfilter* embutida no *kernel* do sistema operacional é construída a partir de uma série de desvios (*hooks*) em vários pontos da pilha do protocolo IPv4, IPv6 e DECnet. Porém, o importante para este trabalho é a pilha IPv4 e, por isso, os diagramas e informações apresentadas serão específicas deste protocolo. A figura 3.1 apresenta um diagrama do que seria o processo natural de roteamento, diz-se processo natural por não executar nenhum tipo de filtragem.

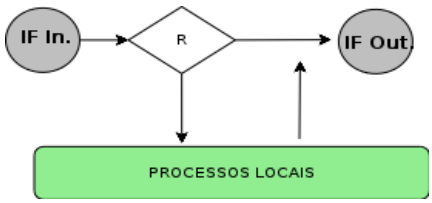


Figura 3.1: Processo natural deroteamento

O pacote inicia o roteamento pela interface de entrada (if in.), ocorre, então, a decisão se o pacote tem que ser repassado a outra interface, de saída (if out.), ou se é destinado a algum processo local. Caso o pacote seja gerado por algum processo local, ele também sofre uma decisão de roteamento e, então, é repassado para a interface de saída correta.

Já com a utilização do *framework netfilter* em pontos específicos, o pacote é desviado ao *netfilter* para que seja verificado conforme as regras específicas deste desvio.

A figura 3.2 apresenta o diagrama do *framework netfilter*. Qualquer pacote percorre este diagrama para passar pela filtragem.

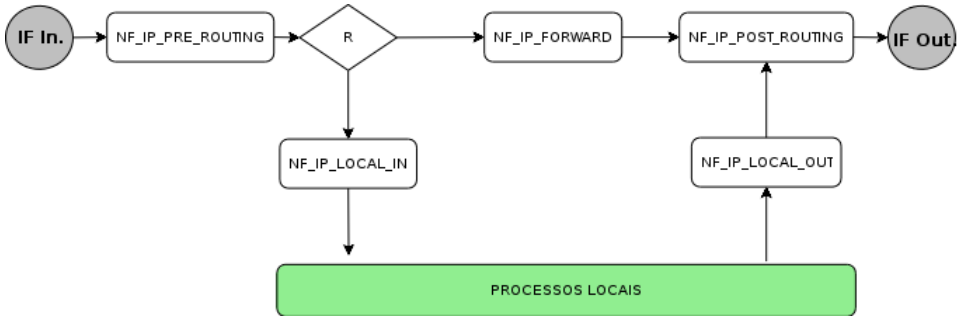


Figura 3.2: *NetFilter Framework*

Após passar pelas verificações simples de consistência – *checksum* correto, não estar truncado e não ser um pacote recebido em modo promíscuo – então o pacote é repassado ao primeiro desvio do *netfilter*, NF_IP_PRE_ROUTING. Após isso, o pacote passa pela decisão de roteamento. Na decisão de roteamento, podem ocorrer três situações:

- O pacote é destinado a um processo local:

  se o pacote é destinado a algum processo da própria máquina, novamente o pacote é desviado ao *netfilter*. Antes de ser entregue ao processo local, o pacote passa pelo desvio NF_IP_LOCAL_IN.

- o pacote é roteado para outra interface;

  se o pacote for roteado para outra rede, então o pacote é desviado para NF_IP_FORWARD. Antes de ser destinado à nova, rede o pacote ainda passa por um último desvio, NF_IP_POST_ROUTING.

- o pacote é descartado pois não pode ser roteado.

Os pacotes são desviados para NF_IP_LOCAL_OUT quando são criados por processos locais.

Portanto, a essência de todo *framework netfilter* é a utilização de desvios ligados ao código de roteamento da pilha, conforme as situações apresentadas acima. Em cada desvio, o *netfilter* poderá executar alguma ação sobre o pacote, na documentação, existem pelo menos cinco ações, porém as duas principais ações são:

- NF_ACCEPT: o pacote continua atravessando normalmente;
- NF_DROP: o pacote é descartado, não pode continuar.

3.2 *Filter*, NAT e *Mangle*: As três Tabelas do *Framework*

Cada um dos desvios apontados anteriormente na arquitetura possui algumas funções específicas, divididas entre filtragem, tradução de endereços (NAT) ou de marcação de pacotes. Estas funções são divididas em tabelas, sendo estas respectivamente *filter*, *nat* e *mangle.* A seleção de pacotes a serem tratados por cada uma das tabelas é feita com o *iptables*.

3.2.1 Tabela *Filter*

A tabela *filter* é destinada especificamente às regras de filtragem. Os pacotes não sofrem nenhuma alteração nesta tabela, somente são filtrados. Esta tabela se aplica aos pacotes que são desviados no *netfilter* em NF_IP_LOCAL_IN, NF_IP_FORWARD e NF_IP_LOCAL_OUT. Cada um destes desvios são associados respectivamente a três listas de regras, denominadas *chains*, em espaço de usuário com *iptabes*: INPUT, FORWARD e OUPUT. A arquitetura desta tabela é apresentada na figura 3.3.

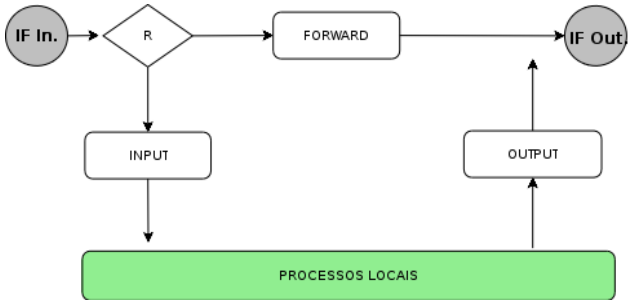


Figura 3.3: Arquitetura da tabela filter do firewall Iptables

Quando o pacote atinge uma destas *chains*, todas as listas de regras desta *chain* é examinada com o objetivo de decidir qual a ação será aplicada ao pacote. Caso não haja nenhuma regra cuja expressão da condição seja satisfeita mediante as informações do pacote, então a política *default* da *chain* é acionada, com isso, o pacote poderá ser descartado por uma regra, ou pela política *default,* ou poderá ser aceito também por uma regra ou pela política *default*. Se o pacote for aceito, ele retorna do desvio e passa para a próxima etapa dentro da arquitetura apresentada.

Pacotes iniciados por processos locais passam pela *chain* de OUPUT antes de serem colocados na rede. Pacotes que chegam até o *host*, primeiro sofrem uma decisão de roteamento, onde o *firewall* verifica quem é o destinatário do pacote. Caso o pacote seja destinado a processos locais do próprio *host*, o pacote então é verificado na *chain* INPUT, se o pacote não for destinado ao próprio *host*, e se este estiver habilitado a fazer repasse de pacotes (fundamental a um filtro de pacotes), então o pacote é destinado a interface de saída, passando pelas regras da *chain* FORWARD. Caso seja bloqueado por alguma regra da *chain* ou se o *host* não possuir capacidade para fazer o repasse, o pacote é descartado.

3.2.2 Tabela NAT

A tabela *nat* é utilizada com a finalidade de se fazer tradução de endereços. Pacotes não originados localmente sofrem influência desta tabela quando desviados em NF_IP_PRE_ROUTING e NF_IP_POST_ROUTING. NF_IP_LOCAL_OUT e NF_IP_LOCAL_IN são utilizados para alterar o destino e origem de pacotes locais.

Três *chains* são observadas na tabela *nat*. PREROUTING, POSTROUTING e OUPUT. PREROUTING associado ao desvio NF_IP_PRE_ROUTING, é utilizada para alterações no destino do pacote. Este tipo de *nat* é chamado de *Destination* NAT ou DNAT. OUTPUT correspondente a qualquer pacote originado localmente. POSTROUTING, associado ao desvio NF_IP_POST_ROUTING, é correspondente a pacotes que terão o endereço de origem alterados, operação chamada de *Source* NAT ou SNAT. Ainda na lista POSTROUTING, pode-se utilizar outra forma especial de *Source* NAT designada *Masquerading*.

3.2.3 Tabela *Mangle*

A *mangle* pode ser utilizada em todos os cinco desvios do *framework netfilter*. O objetivo desta tabela é promover alterações nas informações dos pacotes, ou simplesmente marcações, para que posteriormente sejam roteados de forma diferente ou para que sejam implementadas regras de qualidade de serviço (QoS).

3.3 *Stateful* com *connection tracking*

Iptables é um *firewall* tipo *stateful*, e por isso possui uma tabela de estados que é denominada *conntrack* (*connection tracking*). Os registros da *conntrack* são baseados nas informações de cada protocolo. São criadas regras específicas, indicando que a tabela de estados deva ser consultada. Quando uma conexão é iniciada, *iptables* adiciona uma nova entrada em sua *conntrack* para a sessão em questão. A cada nova entrada na *conntrack*, as seguintes informações são armazenadas:

- protocolo utilizado pela conexão;
- endereço ip de origem e destino;
- portas de origem e destino;
- uma lista com campos endereços ip de origem e destino e portas de origem e destino de forma invertida, para representar um tráfego de resposta;
- tempo que ainda falta para a entrada ser removida;
- o estado da conexão TCP (somente quando o protocolo for TCP); e
- o estado da conexão pela *conntrack*;

Tabela 3.1: Exemplo de entradas na tabela de estados (*conntrack*) do *iptables*

```
    udp      17 17 src=192.168.1.1 dst=192.168.1.255 sport=631 dport=631 packets=1
bytes=120 [UNREPLIED] src=192.168.1.255 dst=192.168.1.1 sport=631 dport=631 packets=0
bytes=0 mark=0 use=1

    tcp      6 103 TIME_WAIT src=200.175.66.56 dst=201.7.178.41 sport=39753 dport=80
packets=16 bytes=2367 src=201.7.178.41 dst=200.175.66.56 sport=80 dport=39753 packets=20
bytes=18224 [ASSURED] mark=0 use=1

    tcp      6 270685 ESTABLISHED src=200.175.83.164 dst=200.96.67.247 sport=50462
dport=22 packets=2324 bytes=162195 src=200.96.67.247 dst=200.175.83.164 sport=22
dport=50462 packets=1793 bytes=480695 [ASSURED] mark=0 use=1
```

Na tabela 3.1, são apresentadas três diferentes entradas da *conntrack*, que representam três diferentes sessões. O primeiro item apresenta o nome do protocolo, seguido pela designação numérica deste protocolo. O seguinte valor (17 para a primeira entrada, 103 na segunda e 270685 na última) indica quanto tempo falta para as referidas entradas serem automaticamente removidas da *conntrack*. Para a segunda e terceira entradas, ambas para conexões TCP, o estado TCP é mostrado. Após aparecem os pares endereço ip (origem e destino) e portas (origem e destino). Na primeira entrada, o próximo item é o estado da conexão, diante da *conntrack*. Como não houve resposta para este pacote, o estado é apresentado como UNREPLIED, motivo pelo qual o valor para *time-out* desta entrada encontra-se tão baixo (17).

Após isto, na lista de cada entrada, são apresentados os valores invertidos para ip e portas de origem e destino, informações que deverão conter os pacotes de retorno. Finalmente, a última informação importante diz respeito ao estado da conexão mediante a *conntrack*. Quando existir um pacote de retorno, sempre após o recebimento do primeiro, o estado que era UNREPLIED é removido e é alterado para ASSURED, com o estabelecimento da conexão, como na segunda e terceira entradas. Uma vez que a conexão é estabelecida, os valores de *time-out* das entradas são aumentados (o valor inicial definido é de 41294 segundos).

3.4 Manipulação das tabelas de regras e listas

Toda política de filtragem depende da configuração que é feita através da manipulação de regras nas listas da tabela *filter*. A construção das regras é feita a partir do uso do programa em espaço de usuário *iptables*. A sintaxe é própria da ferramenta, visto que não existe um padrão definido para as implementações de *firewall* e cada ferramenta possui sua própria sintaxe para a definição das regras.

Cada regra é definida indicando-se a tabela e lista às quais será adicionada, bem como o ordenamento diante das demais regras. Além disso, na definição de cada regra, são utilizadas informações a respeito de protocolo, endereços, portas, interface, estado, dentre outros possíveis para a composição da expressão de condição que identificará os

pacotes que deverão ser controlados. Além da condição, uma ação deverá ser especificada, indicando qual o procedimento adotado pelo filtro em caso da condição especificada na regra combinar com a condição do pacote.

Para a manutenção da tabela de regras, na sintaxe do programa *iptables*, deve-se informar em qual tabela se pretende realizar tal alteração, através do parâmetro -t *tablename,* onde *tablename* poderá ser *filter, nat* ou *mangle*. Por *default,* caso não seja especificada a tabela, a manutenção é realizada sempre na tabela *filter.*

3.4.1 Sintaxe para manipulação de *chains* (listas)

Para a tabela *filter*, sempre irão existir três *chains* principais: INPUT, FORWARD e OUTPUT, estas *chains* não podem ser removidas, entretanto, é possível a utilização de outras *chains* para melhor organização das regras. Caso existam *chains* adicionais criadas pelo administrador, elas deverão ser utilizadas como ações na composição das regras das listas principais.

Para o filtro, a utilização de *chains* adicionais funciona como um desvio, quando uma regra cuja ação seja outra *chain* possuir condição correspondente ao tráfego sendo analisado, o filtro salta para verificação das regras definidas nesta *chain*. Caso alguma tenha condição satisfeita, a ação aplicada ao tráfego é a correspondente desta regra. Se nenhuma regra da *chain* corresponder ao tráfego em análise, o filtro retorna da *chain* adicional e continua a verificação a partir da *chain* principal, utilizando a regra posterior à que ocasionou o desvio. Podem ocorrer novos desvios a partir das *chains* adicionais para outras *chains.*

A figura a seguir demonstra a ação de desvio:

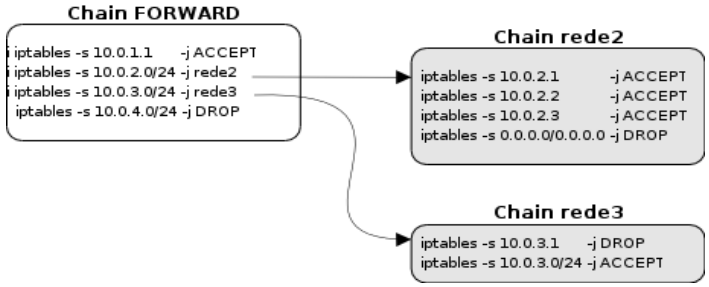


Figura 3.4: Uso de listas adicionais no *iptables*

Para as *chains* principais, INPUT, FORWARD e OUTPUT, existe uma configuração especial para política *default.* A política *default* é utilizada para decidir a ação sobre um pacote, sempre que nenhuma das regras da *chains* tiver condição equivalente ao tráfego. As ações possíveis para a política *default* são DROP, para descartar o pacote e ACCEPT para permitir.

Alguns exemplos de sintaxe para a manipulação de *chains* são apresentados abaixo.

- Para definição da política *default*

iptables -P chain target

Ex.: ***iptables -P INPUT DROP***

Quando listadas as regras de uma *chain*, a política *default* aparece ao lado do nome da *chain.*

- Para manipulação de *chains* adicionais

iptables -[NX] chain

N: para criar nova *chain*

X: para remover uma *chain*

- Para listar as *chains* e respectivas regras de uma tabela

iptables -L [chain] -t filter

a opção -n pode ser utilizada para que não ocorra resolução de nomes

Tabela 3.2: Exemplo de Listagem de regras

```
# iptables -L -n
Chain INPUT  (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD(policy ACCEPT)
target     prot opt source               destination
ACCEPT     all  --  10.0.0.0/8          0.0.0.0/0
ACCEPT     all  --  10.0.1.1            0.0.0.0/0
ACCEPT     all  --  10.0.1.2            0.0.0.0/0
DROP       all  --  10.0.4.0/24         0.0.0.0/0
rede2      all  --  10.0.2.0/24         0.0.0.0/0

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

Chain rede2 (1 references)
target     prot opt source               destination
ACCEPT     all  --  10.0.2.1            0.0.0.0/0
ACCEPT     all  --  10.0.2.2            0.0.0.0/0
DROP       all  --  0.0.0.0/0           0.0.0.0/0
```

3.4.2 Ações

A ação a ser aplicada a um pacote é especificada através do parâmetro "*-j target",* onde *target* é a ação. Os valores possíveis para ação podem ser alguns valores especiais já determinados para ação: ACCEPT, DROP, REJECT, QUEUE, LOG, RETURN ou pode ser o nome de uma *chain* adicional que tenha sido criada pelo usuário.

- ACCEPT: o pacote é aceito;

- DROP: o pacote é descartado;
- REJECT: o pacote é descartado, porém um pacote ICMP com a mensagem de erro é enviado como retorno ao cliente. O tipo ICMP *default* é *icmp-port-unreachable*, porém através da opção *–reject-with*, os seguintes tipos podem utilizados: *icmp-net-unreachable, icmp-host-unreachable, icmp-port-unreachable, icmp-proto-unreachable, icmp-net-prohibited, icmp-host-prohibited ou icmp-admin-prohibited;*
- QUEUE: passa o pacote para algum processo em espaço de usuário;
- LOG: Registra algumas informações do pacote; e
- RETURN: Pára de verificar as regras desta *chain* e retorna à verificação na regra posterior à da *chain* anterior.

3.4.3 Extensões

Uma das principais características do *iptables*, com certeza, é a flexibilidade e a utilização de extensões contribui para isto. A utilização de extensões proporciona a construção de condições mais refinadas para verificação de pacotes, bem como de novas ações.

Extensões são *patchs* que podem ser adicionados ao código do *netfilter* e, a partir daí, compilados para posterior uso.

Para a utilização das funcionalidades de alguma extensão, deve-se adicionar a seguinte sintaxe à construção da regras: ***-m match opções_da_extensão.***

3.4.3.1 Exemplo de utilização da extensão *time*

Esta extensão permite a utilização de parâmetros de data e hora na condição de uma regra. Por exemplo, para aceitar pacotes no intervalo das 8 até as 18 horas de segunda-feira a sexta-feira, pode-se utilizar a seguinte sintaxe:

iptables -A INPUT -m time --timestart 8:00 --timestop 18:00 --days Mon,Tue,Wed,Thu,Fri -j ACCEPT

As opções suportadas por esta extensão são:

- **--timestart**

  horário inicial (HH:MM)

- **--timestop**

  horário final (HH:MM)

- **--days**

lista de dias da semana em que a regra se aplica (*Mon, Tue, Wed, Thu, Fri, Sat, Sun*).

3.4.4 Regras de Estado

A verificação de estados é ativada na condição de regras através da utilização da extensão *state,* a qual permite acesso à *connection tracking* para verificação de estado do pacote.

Possíveis estados:

- INVALID: significa que o pacote não pode ser identificado por algum motivo que inclui falta de memória e mensagens de erros ICMP que não pertencem a nenhuma conexão conhecida;
- ESTABLISHED: o pacote está associado a uma conexão, envia pacotes em ambas direções;
- NEW: o primeiro pacote de uma nova conexão; e
- RELATED: o pacote inicia uma nova conexão, porém é associado com uma conexão já existente, como uma transferência de dados via ftp (*ftp-data*), uma resposta *dns* ou uma mensagem ICMP.

3.4.5 Exemplo de sintaxe para manipulação de regas

A tabela 3.3 apresenta a inserção (ação "-I") de duas regras na estrutura de regras de um *firewall* executando *iptables*. A primeira das regras é inserida na posição 1 da *chain* FORWARD e tem por objetivo permitir, devido à ação ser ACCEPT, o retorno de pacotes a conexões que possam ser identificadas junto à *conntrack*, como pacotes de retorno a uma conexão estabelecida (ESTABLISHED), ou como pacotes de controle que são relacionados àquela conexão.

Tabela 3.3: Exemplo de comandos para construção de regras no *Iptables*

programa	ação na tabela, chain e ordenamento	expressão	ação
iptables	-I FORWARD 1	-m state –state ESTABLISHED, RELATED	-j ACCEPT
iptables	-I FORWARD 3	-p tcp -s 10.0.0.0/8 --dport 80	-j ACCEPT

A segunda regra, a qual é inserida na posição número 3 da *chain* FORWARD, especifica que conexões iniciadas na sub-rede 10.0.0.0/8 (-s 10.0.0.0/8), cujo protocolo de transporte seja TCP (-p tcp), destinadas a quaisquer servidores na porta de destino 80 (--

dport 80), sejam permitidas. Neste exemplo, algumas informações como endereço ip de destino, bem como porta de origem, são suprimidas nas regras, significando então que tais campos indicam qualquer porta ou ip (any).

3.4.6 *iptables-save*: Salvando Regras

O objetivo do utilitário *iptables-save* é gerar um *dump* das regras ativas em um determinado *firewall,* em um formato claro e simples. Como não existe um arquivo de configuração específico, ou algo parecido para configuração da tabela de regras, o utilitário torna-se interessante para a recuperação automatizada de todo conjunto de regras. A partir de outro utilitário, *iptables-restore,* é possível carregar um arquivo de regras salvo pelo *iptables-save,* ou com formato corresponde. A tabela a seguir apresenta um conjunto de regras de uma tabela *filter* salvas com *iptables-save.*

Tabela 3.4: Exemplo tabela de regras salvas com *iptables-save*

```
*filter
:INPUT DROP [4252:1200097]
:FORWARD ACCEPT [6:288]
:OUTPUT ACCEPT [67839:5627879]
:aragorn - [0:0]
:aragorn_ext - [0:0]
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -s 127.0.0.0/255.0.0.0 -d 127.0.0.0/255.0.0.0 -i lo -j ACCEPT
-A INPUT -m state --state INVALID -j DROP
-A INPUT -p icmp -m icmp --icmp-type 8 -j ACCEPT
-A INPUT -s 143.0.0.0/255.0.0.0 -p tcp -m tcp --dport 22 -j ACCEPT
-A INPUT -j aragorn_ext
-A INPUT -s 192.168.1.0/255.255.255.0 -j aragorn
-A INPUT -s 192.168.60.0/255.255.255.0 -j aragorn
-A FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT
-A FORWARD -s 127.0.0.0/255.0.0.0 -d 127.0.0.0/255.0.0.0 -i lo -j ACCEPT
-A FORWARD -m state --state INVALID -j DROP
-A FORWARD -s 192.168.1.0/255.255.255.0 -j ACCEPT
-A FORWARD -s 192.168.60.0/255.255.255.0 -j ACCEPT
-A OUTPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A OUTPUT -s 127.0.0.0/255.0.0.0 -d 127.0.0.0/255.0.0.0 -o lo -j ACCEPT
-A OUTPUT -m state --state INVALID -j DROP
-A OUTPUT -m state --state UNTRACKED -j DROP
-A aragorn -p udp -m udp --dport 67 -j ACCEPT
-A aragorn -p tcp -m tcp --dport 3128 -j ACCEPT
-A aragorn -p tcp -m tcp --dport 143 -j ACCEPT
-A aragorn -p tcp -m tcp --dport 993 -j ACCEPT
-A aragorn -p tcp -m tcp --dport 25 -j ACCEPT
-A aragorn -p udp -m udp --dport 138 -j ACCEPT
-A aragorn -p udp -m udp --dport 137 -j ACCEPT
-A aragorn -p tcp -m tcp --dport 139 -j ACCEPT
-A aragorn -p tcp -m tcp --dport 631 -j ACCEPT
-A aragorn -p udp -m udp --dport 631 -j ACCEPT
-A aragorn -p tcp -m tcp --dport 22 -j ACCEPT
-A aragorn -j LOG
-A aragorn -j REJECT --reject-with icmp-port-unreachable
-A aragorn_ext -p udp -m udp --dport 110 -j ACCEPT
-A aragorn_ext -p udp -m udp --dport 5000 -j ACCEPT
-A aragorn_ext -p tcp -m tcp --dport 5000 -j ACCEPT
-A aragorn_ext -p udp -m udp --dport 53 -j ACCEPT
-A aragorn_ext -p tcp -m tcp --dport 80 -j ACCEPT
-A aragorn_ext -p tcp -m tcp --dport 443 -j ACCEPT
COMMIT
```

4 MODELO TEÓRICO PROPOSTO PARA VERIFICAÇÃO DE INCOERÊNCIAS

4.1 Modelo de Falhas

Durante a configuração de um filtro de pacotes, qualquer uma das regras já implementadas pode sofrer algum tipo de interferência, por exemplo, tratamento diferente para um determinado tipo de tráfego, anulação de execução da regra, dentre outras situações. Este fato tende a ocorrer principalmente pela manutenção constante que é necessária ao conjunto de regras.

Com isso, a tarefa de reconfiguração ou manutenção de um filtro implica uma análise global por parte do administrador, visando à não alteração involuntária da política aplicada pelo filtro. A análise que tem que ser feita entre as regras deve levar em consideração as seguintes variáveis: ordenamento da regra, abrangência dos campos contidos na condição e ação da regra. São estes aspectos que, uma vez alterados, podem provocar que existam regras incoerentes entre si naconfiguração do filtro.

A metodologia proposta por este modelo visa à identificação das incoerências entre regras através da análise envolvendo exclusivamente o conjunto de regras que forma a configuração do filtro. Não são tratadas pelo modelo as incoerências das regras com relação à política de segurança. Sempre que um pacote puder ser condicionado por mais de uma regra, significa que estas são relacionadas, e todas as regras que possuem tratamento para um tráfego comum têm que ser analisadas pois sempre que existir relação entre duas ou mais regras, poderá existir uma incoerência.

O modelo de falhas foi definido utilizando algumas premissas já constituídas no trabalho de Al-Shaer. Al-Shaer classifica, através da utilização do conceito de conjuntos, quais as relações possíveis entre duas regras através da comparação campo a campo das condições da mesma. Para definir o tipo de incoerência, algumas relações entre regras precisam ser estabelecidas. Com referência ao trabalho de Al-shaer e considerando a comparação de duas regras, podem ocorrer as seguintes relações:

- Iguais: todos os campos correspondentes das duas regras possuem valores iguais;
- Superconjunto: As regras não são iguais e todos os campos de uma das regras são iguais ou superconjuntos do campo correspondente à outra regra;
- Subconjunto: As regras não são iguais e todos os campos de uma das regras são iguais ou são subconjuntos do campo correspondente à outra regra;
- Intersecção: As regras são interseccionadas quando não forem iguais, superconjunto ou subconjunto e todos os campos da primeira regra são iguais, subconjunto ou superconjunto aos campos correspondentes da segunda regra.
- Relação parcial: As regras são parcialmente relacionadas quando existir pelo menos um campo na primeira regra que seja igual, subconjunto ou superconjunto ao campo correspondente da segunda regra e se existir pelo menos um campo na segunda regra que não seja igual, subconjunto ou superconjunto ao campo correspondente à primeira regra;
- Sem relação: Nenhum dos valores dos campos correspondentes entre as duas regras tem relação;

Uma relação é definida pela existência de uma igualdade, um superconjunto, um subconjunto, ou uma intersecção entre duas regras. Regras com relação parcial ou sem relação não definem relação e, portanto, não precisam ser analisadas, pois não interferem entre si. Assim sendo, a análise de falhas só será efetuada em regras que possuam relação entre si.

4.2 Caracterização da Falha

A partir da verificação de uma relação entre regras, é necessário então caracterizar se esta relação representa uma falha ou não. Para a caracterização da falha, este modelo leva em consideração os seguintes aspectos:

- tipo de relação entre regras: os tipos de relação que podem apontar incoerência entre as regras relacionadas são: igualdade, subconjunto, superconjunto ou intersecção;
- identidade da ação: comparação entre a ações das regras, caso as duas sejam iguais, existe então identidade na ação;
- ordenação parcial: ordenação entre as regras comparadas. Sendo RA primeira regra e RB segunda regra.

Através da relação destes aspectos, pode-se apontar a um diagnóstico da relação, como apresentado na tabela 4.1, chamada de tabela de caracterização de falha.

As relações que têm como diagnóstico um erro, devem ser desfeitas, visto que um erro indica a existência da falha e esta compromete o correto funcionamento do filtro ou aumenta a complexidade das regras.

Tabela 4.1: Caracterização da Falha

Relação	Grau	Identidade da Ação	Ordenação Parcial	Diagnóstico
Igualdade	-	Sim	-	ERRO, Redundância Desnecessária
Igualdade	-	Não	-	ERRO, Ações Diferentes para a mesma transmissão
Subconjunto	RA ⊆ RB	Sim	RA antes de RB	ERRO, Redundância Desnecessária de RA
Subconjunto	RA ⊆ RB	Não	RA antes de RB	Normal/Aviso
Superconjunto	RB ⊆ RA	Sim	RA antes de RB	ERRO, Redundância Desnecessária de RB
Superconjunto	RB ⊆ RA	Não	RA antes de RB	ERRO, RB nunca será ativado
Intersecção	-	Sim	-	Normal com Redundância Parcial
Intersecção	-	Não	-	Normal/Aviso
Relação Parcial	-	-	-	Normal
Sem Relação	-	-	-	Normal

4.2.1 Significado dos Diagnósticos

4.2.1.1 Redundâncias desnecessárias

Uma das regras nunca será ativada. Essa situação aumenta a complexidade do conjunto de regras, bem como o número total de regras. Orienta-se excluir a que for subconjunto da outra uma vez que isso irá subtrair uma regra e remover a redundância, porém não alterará a política;

4.2.1.2 RB nunca será ativado

RA é mais abrangente que RB. Todas as situações contidas na condição de RB estão contidas em RA e, portanto, como RA está ordenado antes de RB, a ação aplicada é a que está em RA, porém RB não é uma redundância de RA em razão da ação pretendida por RB ser diferente de RA.

4.2.1.3 Ações Diferentes para a mesma transmissão

Indica claramente uma incoerência, pois a mesma situação é tratada igualmente nas duas regras, porém com ações distintas.

4.2.1.4 Normal com redundância parcial

O diagnóstico de "Normal com redundância parcial" indica que a situação é normal, porém as regras contêm uma redundância para algumas condições. Orienta-se que as regras sejam reescritas para eliminar a redundância e facilitar a compreensão.

4.2.1.5 Normal/Aviso

As regras estão corretas uma em relação à outra no diagnóstico Aviso/Normal, porém a situação pode ou não ser errônea; a decisão final neste caso cabe ao administrador. A ocorrência de relações deste tipo são importantes para se definir redundâncias desnecessárias.

4.2.1.6 Normal

As regras comparadas encontram-se sem incoerências uma em relação à outra.

4.3 Definição da política *default* e fluxo de processamento do modelo

A aplicação do modelo em questão segue um fluxo para o processamento das relações, bem como a caracterização destas. Além deste fluxo, existem algumas premissas, principalmente a de ordenação, que são aplicadas no procedimento de comparação, tendo em vista as características de funcionamento de um filtro de pacotes.

A primeira definição é quanto à política *default* do *firewall*. A política *default* sempre será utilizada quando não existir nenhuma regra com condição que possa combinar com o tráfego em avaliação pelo filtro. Nestes casos, a política *default* é aplicada. A configuração da política *default* é global a todo o filtro. Neste modelo, a política é tratada como mais uma regra da tabela, sendo esta a última delas. Caso a política seja proibitiva, é adicionada uma regra com condição abrangente a todo tipo de tráfego, porém com ação *deny*. Em caso de permissiva, a condição novamente é a mesma, abrangendo a todo tipo de tráfego, no entanto com ação *allow*.

Tabela 4.2: Regra para política *default*

	Endereço de origem	porta de origem	endereço de destino	porta de destino	protocolo	estado	ação
ul.	any	any	any	any	any	any	deny ou allow

A comparação aos pares de regras é outra importante definição do modelo. A comparação leva em consideração outras regras, porém, primeiramente para se definir possíveis relações, as regras são comparadas aos pares. Para cada par de regras, este modelo estabelece a relação RA comparada à RB, sendo a primeira regra RA e a segunda regra RB. Como dito anteriormente, não somente a possível detecção de uma relação entre RA x RB vai caracterizar erro ou não nesta relação entre regras, visto que é muito importante para se caracterizar precisamente a relação, levar em consideração outros possíveis relacionamentos que a regra RA em questão tenha possibilitado com outras regras anteriores a esta.

RA deve iniciar com a primeira regra da tabela. No exemplo apresentado na tabela 4.3, RA inicia com a regra 1. RB deve sempre iniciar em RA+1, portanto, como na tabela de exemplo, RB inicia com a regra 2. Se RA estiver posicionado na regra 10, RA deverá ser comparado com as regras RB, iniciando em 11 até a última regra da tabela, no exemplo, regra 19. Supondo o início do procedimento de comparação de regras, RA = 1, será comparada com as regras RB = 2 até RB = 19. Chegando ao fim destas comparações, RA passa para a regra 2 e RB inicia na regra 3 até RB = 19. Quando RA estiver posicionado na última regra, no exemplo da tabela 4.3, RA = 19, RA não será comparado a nenhuma outra regra.

O procedimento de comparação foi definido com base na premissa de ordenação e utilização das regras de um filtro. As regras são utilizadas pelo filtro de forma seqüencial e, uma vez que a condição da regra seja satisfeita pelo tráfego em questão, a ação desta é aplicada e as demais regras são ignoradas pelo filtro. Ainda quanto à ordenação a regra 2, por exemplo, nunca será verificada pelo filtro antes da regra 1 ou depois da regra 3 e, por isso, a metodologia também tem que prever que nunca exista uma comparação em busca de relações entre regras com estas condições, já que seriam detectadas relações incoerentes, visto que o ordenamento parcial das regras é item importante para a definição de caracterização da falha.

Tabela 4.3: Tabela de Regras de filtragem

	Endereço de origem	porta de origem	endereço de destino	porta de destino	proto	estado	ação
1	any	any	10.0.0.0/8	any	any	ESTABLISHED, RELATED	allow
2	any	any	10.0.200.1	22	tcp	NEW	allow
3	any	any	10.0.200.1	443	tcp	NEW	allow
4	any	any	10.0.200.2	25	tcp	NEW	allow
5	any	any	10.0.200.2	110	tcp	NEW	allow
6	any	any	10.0.200.3	53	tcp	NEW	allow
7	any	any	10.0.200.3	53	udp	NEW	allow
8	10.0.0.0/16	any	10.0.200.0/24	22	tcp	NEW	deny
9	10.0.1.0/24	any	10.0.200.0/24	22	tcp	NEW	allow
10	10.0.0.0/16	any	10.0.0.0/16	22	tcp	NEW	allow
11	10.0.200.1	any	10.0.201.1	1433	tcp	NEW	allow
12	10.0.200.0/24	any	10.0.201.1	1433	tcp	NEW	allow
13	10.0.13.0/24	any	10.0.201.1	1433	tcp	NEW	deny
14	10.0.0.0/16	any	10.0.201.1	1433	tcp	NEW	allow
15	any	any	10.0.200.2	25	tcp	NEW	allow
16	any	any	10.0.200.2	110	tcp	NEW	allow
17	any	any	10.0.200.2	110	tcp	NEW	deny
18	10.0.1.1	any	10.0.200.2	22	tcp	NEW	allow
19	any	any	any	any	any	NEW	deny

Ainda, para melhor exemplificar a importância de manter o processo de comparação, de acordo com o fluxo de utilização de regras executado pelo filtro de pacotes, verificam-se as regras apresentadas na tabela 4.4, fazendo-se primeiramente uma análise das duas regras. Nota-se que o objetivo do administrador é bloquear conexões a partir da rede 10.0.13.0/24 destinadas ao serviço na porta tcp/1433 do *host* servidor 10.0.201.1, visto que a primeira regra executa a ação de bloqueio destas conexões, sendo verificada pelo filtro primeiramente a segunda regra que tem ação permissiva a toda rede 10.0.0.0/16, incluindo aqui a rede 10.0.13.0/24, ao mesmo serviço do *host* 10.0.201.1.

Neste exemplo, a remoção da primeira regra ou inversão no ordenamento das mesmas alteraria a política implementada pelo a administrador, pois a regra 14 é mais abrangente no campo de endereço de origem, compreendendo todos os pacotes que combinariam com a regra 13. Portanto, se o modelo de verificação permitisse a comparação RA = 2 X RB =1, seria detectada uma relação de superconjunto e, com base na tabela de caracterização de falhas, tabela 4.1, a ferramenta apresentaria um diagnóstico de erro, indicando que RB nunca seria ativada.

Tabela 4.4: Tabela parcial de regras

	Endereço de origem	porta de origem	endereço de destino	porta de destino	proto	estado	ação
13	10.0.13.0/24	any	10.0.201.1	1433	tcp	NEW	deny
14	10.0.0.0/16	any	10.0.201.1	1433	tcp	NEW	allow

Outro motivo, pelo qual RB deve ser RB = RA + 1 é o fato de garantir que a mesma regra não seja comparada a ela mesma, o que geraria um erro de incoerência, indicando a caracterização de uma redundância desnecessária, visto que existiria uma relação de igualdade, com identidade de ação.

4.4 Procedimento de comparação e caracterização

A partir do fluxo de processamento das regras, avalia-se mais profundamente o procedimento de comparação para a busca das relações, bem como o procedimento de caracterização da falha. O processo de comparação de regras tem como procedimento a comparação campo a campo da condição da regra, para então apontar uma relação final com base nos resultados da comparação como um todo. As relações possíveis para cada campo são de:

- igualdade dos campos;
- contém ou está contido;
- ou de que não há relação entre os campos;

A partir destas três possibilidades de relação entre campos, ao final da comparação de todos os campos, pode-se chegar às relações definidas no modelo de falhas.

4.4.1 Relações Parciais

Quando existir uma relação do tipo parcial, em que, pelo menos em duas regras existe um campo da condição sem relação, que não tenha relação de igualdade,

subconjunto ou superconjunto, e um campo com relação, significa que as regras em comparação não possuem relação e, portanto, não são incoerentes entre si. Independe de qual campo não existe relação, a diferença em um campo significa que o tratamento das regras em questão visa a diferentes conexões e, portanto, como não tratam o mesmo tráfego, também não devem ser comparadas pela ferramenta que implementa o modelo apresentado.

Tabela 4.5: Comparação de regras de relação parcial

	Endereço de origem	porta de origem	endereço de destino	porta de destino	proto	estado	ação
6	any	any	10.0.200.3	53	tcp	NEW	allow
7	any	any	10.0.200.3	53	udp	NEW	allow

A tabela 4.5, apresenta uma comparação entre duas regras com relação parcial. Como os campos de protocolo são diferentes, não existe relação entre as duas e, por isso, também não existe incoerência.

4.4.2 Relações de Igualdade

A relação de igualdade é obtida quando todos os campos da comparação possuem os mesmos valores. Tal relação é apresentada na tabela de regras abaixo:

Tabela 4.6: Comparação de regras de relação de igualdade

	Endereço de origem	porta de origem	endereço de destino	porta de destino	proto	estado	ação
4	any	any	10.0.200.2	25	tcp	NEW	allow
15	any	any	10.0.200.2	25	tcp	NEW	allow

Com base na tabela de caracterização de falha nas relações de igualdade o aspecto de ordenação parcial e grau de relação das regras não é relevante para a caracterização. O único aspecto a ser considerado em relações de igualdade é a identidade ou não da ação. Havendo identidade na ação, é caracterizado um erro de redundância desnecessária.

A existência de uma redundância deve ser eliminada, pois a retirada de uma das regras não afeta a política, porém, se não removida, aumenta o número de regras, aumentando assim a complexidade da política de *firewall* como um todo, além de aumentar também o tempo gasto pelo filtro para a checagem de um número maior de regras.

Caso a ação das regras em comparação seja diferente, também é caracterizado um erro. A segunda regra nunca será utilizada pelo filtro, e a existência de tal relação aponta a incoerência das regras escritas em relação à política que o administrador planeja implementar. Neste tipo de situação, não é possível apontar qual regra tem que ser retirada, pois cabe ao administrador verificar qual está em concordância com a política que deve ser implementada.

4.4.3 Relações de Subconjunto

Na ocorrência de relações do tipo subconjunto, onde RA está contido em RB (RA $\subseteq$ RB), podem existir duas situações. Uma pode resultar em incoerência e a outra pode servir para detectar incoerências entre relações da regra RA em questão com outras regras da tabela. Na primeira relação possível, RA $\subseteq$ RB, as duas regras possuem ações diferentes e a ordenação parcial é RA antes de RB. Com estes aspectos, tem-se a caracterização de uma relação considerada normal/aviso.

Tabela 4.7: Comparação de regras tipo RA subconjunto de RB com ações diferentes.

	Endereço de origem	porta de origem	endereço de destino	porta de destino	proto	estado	ação
13	10.0.13.0/24	any	10.0.201.1	1433	tcp	NEW	deny
14	10.0.0.0/16	any	10.0.201.1	1433	tcp	NEW	allow

Neste exemplo da tabela 4.7, a relação não gera nenhum tipo de incoerência, visto que a primeira regra bloqueia o acesso de uma determinada rede ao serviço tcp/1433 do servidor 10.0.201.1, a qual faz parte da rede 10.0.0.0/16. Portanto, como a regra menos restritiva aparece seqüencialmente antes que a segunda regra, o tráfego da rede 10.0.13.0/24 é bloqueado, enquanto os demais *hosts* da rede 10.0.0.0/16 têm seu acesso permitido.

Tal situação é absolutamente normal na configuração de um filtro, principalmente quando a regra envolve um grande número de *hosts,* como exemplificado na regra 14 da tabela 4.7. Poder-se-ia associar tais regras com uma política que define o acesso de todas as estações de uma determinada empresa a um servidor de banco de dados, porém a rede 10.0.13.0/24 poderia ser associada a uma rede destinada a visitantes, e estes não devem possuir acesso a tal servidor.

Uma outra forma de implementar tal política, sem a utilização da regra 13 que bloqueia o acesso da rede 10.0.13.0/24, desfazendo então o relacionamento entre as regras seria escrever uma regra para cada *host* que poderia ter acesso ao servidor de banco de dados. Tal forma resultaria num número possivelmente elevado de regras, devido a tratar-se de uma rede com máscara 255.255.0.0 e, com isso, o número de regras seria igual ao

número de estações da rede. Este modelo de escrita de regras vai contra os objetivos de diminuição de complexidade de configuração de um filtro, além de manter exatamente a mesma política descrita com as duas regras da tabela 4.7.

De qualquer forma, o exemplo da tabela 4.7, mesmo que seja uma situação normal, contém uma relação entre regras e, portanto, o modelo prevê que tal situação possa ser apresentada ao administrador, para que as regras em questão possam ser avaliadas de forma isolada e mantidas ou não por este, por isso a caracterização desta situação é apontada como Normal/Aviso.

Quando a relação detectada for do tipo subconjunto, e as duas regras possuam ações iguais, juntamente com ordenação parcial RA antes de RB, tem-se evidências da caracterização de uma relação que pode ser considerada um erro, pois existe uma redundância desnecessária da regra RA. Como apresentado na tabela 4.8, as duas regras têm ações iguais, e a regra 12 tem abrangência inclusive sobre o tráfego da regra 11, portanto, se a regra 11 fosse removida, a política implementa pelo *firewall* não sofreria alteração além de ser simplificada.

Tabela 4.8: Comparação de regras tipo RA subconjunto de RB com ações iguais

	Endereço de origem	porta de origem	endereço de destino	porta de destino	proto	estado	ação
11	10.0.200.1	any	10.0.201.1	1433	tcp	NEW	allow
12	10.0.200.0/24	any	10.0.201.1	1433	tcp	NEW	allow

Entretanto, uma característica apropriada deste modelo de verificação é levar em consideração outras relações que podem ter ocorrido com RA em questão, para situações de redundância desnecessária da regra menos abrangente. O problema em questão é apresentado na tabela 4.9.

Tabela 4.9: Comparação de regras tipo RA subconjunto de RB.

	Endereço de origem	porta de origem	endereço de destino	porta de destino	proto	estado	ação
13	10.0.13.0/24	any	10.0.201.1	1433	tcp	NEW	deny
14	10.0.0.0/16	any	10.0.201.1	1433	tcp	NEW	allow
19	any	any	any	any	any	any	deny

O fluxo de comparação será apresentado para expor a necessidade de verificar outras relações, ao contrário de caracterizar a relação somente pela relação parcial. Abaixo seguem as comparações:

- Início do processo:
- Passo 1: Quando RA = regra 13 e RB = regra 14
    - Relação:

      RA X RB = RA é subconjunto de RB, RA precede RB e as ações são diferentes;

    - Caracterização da Relação:

      Normal/Aviso
- Passo 2: Quando RA = regra 13 e RB = regra 19
    - Relação:

      RA X RB = RA é subconjunto de RB, RA precede RB e as ações são iguais;

    - Caracterização da Relação:

      Redundância desnecessária de RA

Neste caso, se o administrador seguir a sugestão da ferramenta de verificação, este irá entender que a regra 13 é desnecessária, pois esta é a caracterização quando verificada a relação entre RA = regra 13 e RB = regra 19, portanto sugere que a regra 13 seja removida; se a regra 13 for removida, a política implementada pelo filtro sofrerá alteração, pois a regra 13 garante que a rede 10.0.13.0/24 não possa ter acesso ao serviço tcp/1433 do *host* 10.0.201.1.

Sabe-se, porém, que a regra 13 participa de uma relação com a regra 14, cuja caracterização é Normal/Aviso. Se a regra RA, considerada redundância desnecessária no passo 2, participar de alguma relação que precede o passo 2, e a caracterização for considerada Normal/Aviso, então esta relação, verificada no passo 2, não pode ser considerada incoerente. Caso não exista nenhuma relação anterior entre RA e outra regra, cuja caracterização tenha sido Normal/Aviso, então a caracterização para a relação do passo 2 será realmente uma redundância desnecessária de RA.

A definição do modelo aponta sempre a regra menos abrangente como a redundância desnecessária, visto que a remoção desta regra não afeta a política implementada, porém sempre cabe ao administrador verificar se a regra mais abrangente atende à política ou não. Podem existir situações em que o correto é manter a regra menos abrangente, pois a regra mais abrangente não mais reflete a política de segurança. De qualquer forma, o modelo aponta a relação entre as regras e, se removida a regra menos abrangente, a política é mantida, diminui-se a complexidade, removendo-se a relação.

4.4.4 Relações de Superconjunto

Ainda no campo das relações, quando comparadas RA x RB e RA for superconjunto de RB, portanto RA ⊆ RB, novamente podem existir duas situações que caracterizam incoerências entre as regras comparadas. Será caracterizado um erro quando o grau de intersecção for RA ⊆ RB e não existir identidade nas ações e RA vier antes de RB. A regra RB nunca será ativada. A tabela 4.10 exemplifica este tipo de incoerência.

Tabela 4.10: Comparação onde RA é superconjunto de RB com ações diferentes

	Endereço de origem	porta de origem	endereço de destino	porta de destino	proto	estado	ação
8	10.0.0.0/16	any	10.0.200.0/24	22	tcp	NEW	deny
18	10.0.1.1	any	10.0.200.2	22	tcp	NEW	allow

Provavelmente a política desejada nesta situação seria bloquear o acesso da rede 10.0.0.0/16 ao serviço tcp/22 da rede 10.0.200.0/24 e RB deveria liberar um acesso específico como exceção à política. Dado ao fato de que, após a condição ser aceita por uma determinada regra, as demais regras são ignoradas, RB nunca será utilizada. Tal situação é apontada ao administrador como erro.

Se a relação for superconjunto, RA ⊆ RB, mas o aspecto de identidade da ação, diferentemente da relação anterior, for de igualdade, a caracterização deverá apontar para uma redundância desnecessária da regra de menor escopo, no caso RB. A tabela 4.11 exemplifica esta relação.

Tabela 4.11: Comparação de regras tipo RA superconjunto de RB com ações iguais

	Endereço de origem	porta de origem	endereço de destino	porta de destino	proto	estado	ação
9	10.0.1.0/24	any	10.0.200.0/24	22	tcp	NEW	allow
18	10.0.1.1	any	10.0.200.2	22	tcp	NEW	allow

A relação aponta claramente um erro, pois a regra 18, ou seja, RB nunca será ativado. Entretanto, a caracterização do erro pode ser mais específica, pois, além de não ser executada, a regra RB tem escopo enquadrado na regra RA, portanto RB resulta em uma redundância desnecessária que, se removida, não altera a política implementada pelo *firewall*.

Diferentemente da relação onde RA é subconjunto de RB, as ações entre as regras são iguais, e RA somente será apresentada como redundância desnecessária se RA não

possuir relação cuja caracterização seja Normal/Aviso. Na ocorrência de uma redundância desnecessária de RB, não é necessário verificar se RB já esteve relacionada anteriormente à outra regra e se a caracterização desta relação foi Normal/Aviso, pois a regra RA que torna RB redundância desnecessária, atende inclusive ao tráfego da relação RB. Portanto, neste contexto, RB é uma clara incoerência, pois somente aumenta a complexidade e o número de regras do filtro. A tabela 4.12 apresenta um pequeno número de regras que exemplifica e comprova esta definição.

Tabela 4.12: Comparação de regras tipo RA subconjunto de RB.

	Endereço de origem	porta de origem	endereço de destino	porta de destino	proto	estado	ação
1	10.0.0.0/16	any	10.0.201.1	1433	tcp	NEW	allow
2	any	any	10.0.201.1	1433	tcp	NEW	deny
3	10.0.13.0/24	any	10.0.201.1	1433	tcp	NEW	deny
4	any	any	any	any	any	any	allow

Abaixo segue o fluxo das comparações:

- Início do processo:
- Passo 1: Quando RA = regra 1 e RB = regra 2
    - Relação:

      RA X RB = RA é subconjunto de RB, RA precede RB e as ações são diferentes;

    - Caracterização da Relação:

      Normal/Aviso;
- Passo 2: Quando RA = regra 1 e RB = regra 3
    - Relação:

      RA X RB = RA é superconjunto de RB, RA precede RB e as ações são diferentes;

    - Caracterização da Relação:

      Erro, RB nunca será ativado;
- Passo 3: Quando RA = regra 1 e RB = regra 4
    - Relação:

      RA X RB = RA é subconjunto de RB, RA precede RB e as ações são iguais;

    - Caracterização da Relação:

Normal, pois RA tem relação caracterizada normal/aviso qdo RA=1 e RB=2;

- Passo 4: Quando RA = regra 2 e RB = regra 3
  - Relação:

    RA X RB = RA é subconjunto de RB, RA precede RB e as ações são iguais;

  - Caracterização da Relação:

    Erro, Redundância desnecessária de RB;
- Passo 5: Quando RA = regra 2 e RB = regra 4
  - Relação:

    RA X RB = RA é subconjunto de RB, RA precede RB e as ações são diferentes;

  - Caracterização da Relação:

    Normal/Aviso;
- Passo 6: Quando RA = regra 3 e RB = regra 4
  - Relação:

    RA X RB = RA é subconjunto de RB, RA precede RB e as ações são diferentes;

  - Caracterização da Relação:

    Normal/Aviso;

Se forem seguidos os apontamentos demonstrados no exemplo acima pelo modelo, a regra 3 apresenta relações incoerentes em duas situações: no passo 2 e no passo 4 e, portanto, deve ser removida. Mesmo que a regra 3 participe de uma relação caracterizada como Normal/Aviso, no passo 6 com a regra 4, esta relação é posterior às incoerências e, por isso, a política continua a mesma.

Tabela 4.13: Resultado após aplicação do modelo

	Endereço de origem	porta de origem	endereço de destino	porta de destino	proto	estado	ação
1	10.0.0.0/16	any	10.0.201.1	1433	tcp	NEW	allow
2	any	any	10.0.201.1	1433	tcp	NEW	deny
3	any	any	any	any	any	any	allow

4.4.5 Relações de Intersecção

A relação de intersecção entre duas regras ocasiona que um determinado tráfego possa ter condição satisfeita pelas duas regras em questão. Uma relação de intersecção existe quando a relação não for de igualdade, subconjunto ou superconjunto, e todos os campos da primeira regra são iguais, subconjunto ou superconjunto aos campos correspondentes da segunda regra. A tabela 4.14 apresenta uma relação de intersecção, onde o endereço de origem da regra 2 é superconjunto da regra 8. O endereço de destino da regra 2 é subconjunto da regra 8. Os demais campos da condição são iguais.

Tabela 4.14: Relação de intersecção entre regras

	Endereço de origem	porta de origem	endereço de destino	porta de destino	proto	estado	ação
2	any	any	10.0.200.1	22	tcp	NEW	allow
8	10.0.0.0/16	any	10.0.200.0/24	22	tcp	NEW	deny

A intersecção apresentada na tabela 4.14, refere-se ao seguinte tráfego:

Tabela 4.15: Condição referente a intersecção

Endereço de origem	porta de origem	endereço de destino	porta de destino	proto	estado
10.0.0.0/16	any	10.0.200.1	22	tcp	NEW

Portanto, as duas regras poderiam ser utilizadas se o tráfego estivesse condicionado à intersecção das duas regras, neste exemplo, para a condição apresentada na tabela 4.15.

Sendo que a ordenação parcial é sempre RA antes de RB, podem existir duas caracterizações em relações de intersecção. O aspecto determinante para as duas situações é a identidade ou não na ação. Quando os campos da ação possuem identidade, a caracterização é de relação normal com redundância parcial. A redundância parcial é a parte da condição definida pelo modelo de intersecção. A tabela 4.16 apresenta uma relação de intersecção com identidade na ação.

Tabela 4.16: Relação de intersecção entre regras resultante em redundância parcial

	Endereço de origem	porta de origem	endereço de destino	porta de destino	proto	estado	ação
2	any	any	10.0.200.1	22	tcp	NEW	allow
9	10.0.1.0/24	any	10.0.200.0/24	22	tcp	NEW	allow

A redundância parcial existe porque as duas regras aceitariam o tráfego da rede 10.0.1.0/24 destinado serviço ao 22/tcp ao *host* 10.0.200.1. A caracterização é apresentada na forma de aviso ao administrador.

No caso de existir intersecção entre as regras, e as ações das mesmas serem diferentes, como presente na tabela 4.14, a caracterização é Normal/Aviso. A caracterização de Normal/Aviso, neste tipo de ocorrência, tem duas funcionalidades importantes. Primeiramente, apresentar a relação para concordância total do administrador; e segundo, como visto na relação de RA subconjunto de RB, para definir com certeza caracterizações de redundância desnecessária de RA.

4.5 Incoerências em regras de controle de estado

Como apresentado no capítulo 1 deste trabalho, existem implementações de filtros do tipo *stateless* cuja implementação não leva em consideração estados de conexões; e filtros do tipo *stateful*, que tratam desta questão. Cabe também salientar que a utilização de controle de estados nos filtros *stateful* tem que ser ativada via configuração de regras pelo administrador, sob pena de poder ocorrer implementações de conjuntos de regras *stateless* em mecanismos *stateful*.

Em qualquer uma das implementações de filtros, pode ocorrer falhas de configuração onde o tráfego de retorno não é condicionado e aceito pelo filtro. Com isso aparece um novo tipo de incoerência, onde um fluxo de tráfego é permitido, porém o retorno não e, com isso, as conexões não são estabelecidas, e o serviço em questão pode não funcionar ou funcionarparcialmente.

Para que as operações de liberações ou bloqueios de novos serviços em configurações de *firewall* possam ser corretas e seguras, é necessário ao administrador do filtro um bom conhecimento de funcionamento do protocolo utilizado por este serviço. Informações como: se a comunicação é orientada à conexão ou não; se possui algum tipo de retorno; se pode gerar algum outro tipo de tráfego relacionado, dentre outras, são de fundamental importância para que se possa construir uma definição correta e exata do tráfego sainte e entrante relacionado a tal serviço.

Nos filtros *stateless*, para cada liberação de tráfego, é necessário que de forma explicita seja adicionada uma regra também permissiva garantindo o tráfego de retorno para o *host* que iniciou a conexão. Nos filtros *stateful*, esta regra também é necessária, entretanto pode ser escrita utilizando-se as vantagens de possuir uma tabela de estados, para todas conexões que tenham passado pelo *firewall*. Com isso se consegue garantir de forma efetiva que somente um tráfego de retorno a uma sessão anteriormente aceita pela condição de determinada regra do filtro possa passar pelo *firewall* com destino ao *host* específico. Tal tarefa é impossível em filtros sem controle de estado, expondo o filtro a aceitar tráfegos que deveriam ser bloqueados, como apresentado no primeiro capítulo deste trabalho.

Para que não existam estas novas incoerências, é necessário que para cada regra exista outra cuja condição do fluxo esteja invertida e desta forma represente o tráfego de retorno. As tabelas 4.17 e 4.18, são respectivamente dois exemplos de regras em filtro *stateless* e filtro *stateful*, onde a segunda regra é responsável pelo tratamento do tráfego de retorno.

Tabela 4.17: Regras tratando o fluxo entrante e sainte em filtro *stateless*

	Endereço de origem	porta de origem	endereço de destino	porta de destino	proto	ação
1	10.0.0.0/16	any	10.0.201.1	22	tcp	allow
2	10.0.201.1	22	10.0.0.0/16	any	tcp	allow

Tabela 4.18: Regras tratando o fluxo entrante e sainte em filtro *stateful*

	Endereço de origem	porta de origem	endereço de destino	porta de destino	proto	estado	ação
1	10.0.0.0/16	any	10.0.201.1	22	tcp	NEW	allow
2	10.0.201.1	22	10.0.0.0/16	any	tcp	ESTABLISHED, RELATED	allow

4.5.1 Procedimento de caracterização de incoerências de estado

A verificação de incoerências de estado, deve também seguir obrigatoriamente alguns requisitos em relação ao processo para caracterizar, de forma correta, possíveis falhas.

Primeiramente, este tipo de falha somente deve ser verificada após adequação das regras em relação aos diagnósticos de incoerências apontados nas comparações diretas de regra a regra, cuja definição foi realizada anteriormente no item 4.4. O motivo para esta definição está em não gerar diagnósticos de erro sobre os erros já anteriormente apresentados, o que aumentaria o número de mensagens possivelmente sem relevância, visto que, removidas as primeiras relações de incoerências, as caracterizações de erro, no ponto da análise de estados, também podem sofrer alterações.

Um segundo item, também de fundamental importância, diz respeito à política geral implementada. Existem duas abordagens principais: por *default deny*, onde o princípio é bloquear tudo e, a partir daí liberar somente os serviços necessários; ou por *default allow*, onde tudo é permitido, porém o tráfego que se quer bloquear é explicitamente condicionado às regras escritas na configuração.

No caso da política *default* ser *allow*, muito provavelmente existirão somente regras de ação *drop*, bloqueando certos tipos de tráfegos e, por isso, não é necessário definir regras permitindo retorno, pois por *default* ele será aceito. Entretanto, podem existir regras ordenadas entre as últimas da configuração cuja condição é ampla e a ação seja *drop*, o que, de certa forma, inverte a política e, neste caso, surge a necessidade de verificar a existência de regras de retorno para regras que sejam precedentes a esta e cuja ação seja *allow*.

Portanto, somente se fará uma verificação de regras de estado em regras cuja ação seja *allow* e, por isso, permite a passagem de um determinado tráfego e, neste caso, deve-se avaliar se existe outra regra que permita tráfego de retorno.

4.5.2 Modelo de verificação para filtros *stateless*

Uma boa prática adotada por administradores de filtros *stateless* é adicionar, logo após cada regra que diga respeito a uma nova transmissão, uma regra que trate do tráfego de retorno desta transmissão, como apresentado na tabela 4.17. A forma mais restritiva é permitir somente o retorno cuja origem seja o ip e a porta do servidor (especificado, nos endereços de destino e porta de destino na regra 1) e o destino seja a faixa de endereços e portas especificados na condições de origem (endereços de origem e porta de origem da regra 1).

Para verificação deste tipo de incoerência, é necessário localizar, no conjunto de regras, a existência de uma que, invertendo-se os campos de origem e destino, a regra tenha relação de igualdade, representando, assim, a regra de retorno.

Existem algumas exceções a esta metodologia: mensagens ICMP de aviso de conectividade, por exemplo, devem ser tratadas de forma individual, sem que formem par com nenhuma outra regra.

4.5.3 Modelo de verificação para filtros *stateful*

Para filtros *stateful*, uma boa prática é, já nas primeiras regras, condiciona-los a uma regra de estado com abrangência a qualquer tráfego.

Tabela 4.19: Boa prática para regra de controle de estado em filtros *stateful*

	Endereço de origem	porta de origem	endereço de destino	porta de destino	proto	estado	ação
1	any	any	any	any	any	ESTABLISHED, RELATED	allow

Após a regra de estado, somente são descritas regras para novas conexões que tratam o primeiro pacote, após isso, toda comunicação que estiver envolvida nesta sessão é tratada pela regra de estado.

Tabela 4.20: Exemplo de definição de regras em *firewall stateful*

	Endereço de origem	porta de origem	endereço de destino	porta de destino	proto	estado	ação
1	any	any	any	any	any	ESTABLISHED, RELATED	allow
2	any	any	10.0.200.1	22	tcp	NEW	allow
3	any	any	10.0.200.1	25	tcp	NEW	allow
4	any	any	any	any	any	any	drop

Dois motivos levam a utilizar este estilo de configuração. Primeiramente, posicionar a regra de controle de estado no início da configuração, sugere uma melhor eficiência do filtro, visto que a maior parte do tráfego diz respeito ao restante da sessão e não ao primeiro pacote que marca o início dela. Desta forma, na maior parte das verificações, o filtro deve condicionar o tráfego a uma sessão cujo estado seja ESTABLISHED ou RELATED, sendo assim mais efetivo para as sessões válidas já iniciadas.

Outro motivo é utilizar uma regra ampla para controle de estado. Desta forma, aceita-se qualquer pacote de retorno considerado válido. Isto porque a filtragem já ocorre naturalmente nas regras que controlam novas sessões (estado NEW) e, portanto, um

tráfego somente será considerado válido pela regra de estado, se anteriormente o início de conexão tenha sido permitido. Com isso, garante-se o controle correto para tráfego de retorno, além de diminuir e simplificar o número de regras para configuração do filtro.

Desta forma, para verificar se todas as regras com estado NEW têm seu tráfego de retorno aceito pela regra de estado, deve-se verificar a seguinte relação para cada regra:

- se os valores para campos de origem forem substituídos pelos valores dos campos de destino e vice-versa, e se a relação resultante da comparação desta regra com alguma regra de estado cuja ação seja *allow* for igual, ou se a regra em questão for subconjunto, significa que o tráfego de retorno para a regra em questão é aceito. Caso contrário, existe uma incoerência.

5 PROTÓTIPO IMPLEMENTADO E TESTES

Visando validar o modelo definido, foi parte deste trabalho a implementação de um protótipo para a verificação de incoerências em regras. Esta ferramenta de verificação foi construída para proceder verificações em conjuntos de regras do *iptables*. O formato utilizado para a descrição das regras é o formato utilizado pela ferramenta *iptables-save*. Para análise pelo protótipo, são aceitos novos conjuntos de regras ou conjuntos legados, salvos pela ferramenta *iptables-save,* não sendo necessária uma nova descrição sintática das regras para a avaliação deste protótipo.

Como parte da validação, obteve-se, junto a duas empresas que fazem o uso do *iptables* como ferramenta de filtragem, dois conjuntos de regras implementadas por seus filtros. Estas regras foram analisadas através do uso do protótipo, e o resultado da análise está publicado neste trabalho. Ainda, realizou-se uma terceira análise, com um conjunto de regras apresentado no trabalho de Al-Shaer (2003), visando comparar a análise feita pelo modelo de Al-Shaer e a análise feita por este modelo.

5.1 Características do protótipo

O protótipo foi construído utilizando-se a linguagem de programação *perl*, por ser uma linguagem de fácil uso, bem como pela característica de fornecer suporte ao uso de expressões regulares, fato que facilita a extração das informações das regras.

O objetivo da ferramenta é ler um arquivo com as regras e, após análise, informar quais possuem relação diagnosticada como incoerente. O administrador pode optar pelo uso da opção "-w", através da qual não só incoerências do tipo erro são informadas, mas também incoerências do tipo Normal/Aviso, por *default* somente incoerências considerada erros são informadas.

O *iptables* é uma ferramenta muito flexível, e um dos principais motivos que reforça esta característica é a possibilidade de construção de regras, utilizando-se várias extensões para composição da condição. Algumas destas extensões são relevantes e freqüentemente utilizadas, porém nem todas.

Tanta flexibilidade às vezes ocasiona maior complexidade, por exemplo, as extensões *udp, tcp, mport* e *multiport*, todas podem ser utilizadas para especificação de portas, porém cada uma com sintaxe diferente. A tabela a seguir demonstra a sintaxe de cada uma destas extensões.

Tabela 5.1: Sintaxe de regras possíveis no *iptables* para especificação de portas

udp e tcp	--source-port port[:port] --sport port[:port]	pode-se especificar uma porta ou um intervalo de portas
multiport	--destination-ports port[,port,port...] --dports port[,port,port...]	pode-se especificar uma porta ou uma seqüência de portas
	--ports port[,port,port]	a seqüência vale para ambos os sentidos, origem e destino
mport	--source-ports port[,port:port,port...]	pode-se especificar um misto de seqüência e intervalo de portas
	--ports port[,port:port,port]	vale para ambos os sentidos

Os exemplos demonstrados na tabela 5.1 podem ser utilizados em todas as formas para especificar origem e também destino. Além disso, o *iptables* permite que se utilize mais de uma extensão e conseqüentemente diferentes especificações de portas na mesma regra. Por exemplo:

iptables -A FORWARD -p tcp --dport 20:23 -m multiport --dports 20,21,22,23 -j ACCEPT

De forma a limitar o escopo deste protótipo, somente são considerados pela ferramenta os campos mais comuns, porém é possível, e de forma fácil, estender também o protótipo, de forma a validar novos campos.

A seguinte lista contém os campos considerados pelo protótipo:

- interface de entrada
- interface de saída
- protocolo
- ip de origem
- ip de destino
- estado (através da extensão *state*)

Se protocolo tcp ou udp (através das extensões *tcp, udp, multiport*):

- porta(s) de origem
- porta(s) e destino

Se protocolo *icmp*:

- icmp-type
- icmp-code

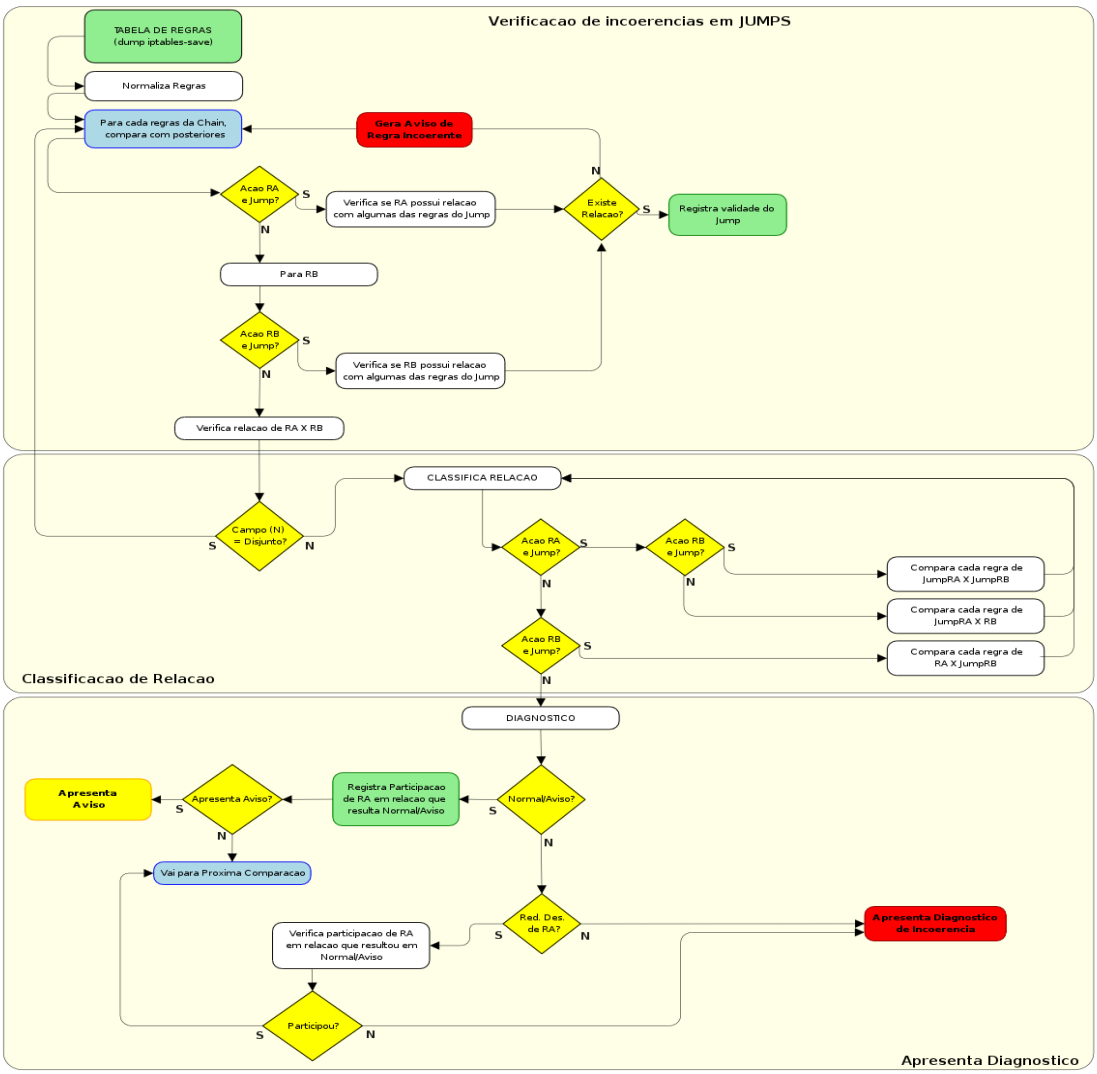
A política *default* para as *chains* principais (INPUT, FORWARD e OUTPUT), conforme apresentado nos requisitos do modelo, é transformada em uma regra abrangente a todo tráfego, com ordenação na última posição da seqüência de regras. A ação desta regra é a ação da política para a *chain* em questão.

Com relação às ações aceitas pelo protótipo, somente as ações ACCEPT, DROP e REJECT são consideradas pela ferramenta. Ações que forem *chains* definidas pelo usuário também são analisadas pela ferramenta.

5.2 Fluxograma de comparações do protótipo

O fluxograma a seguir, na figura 5.1, apresenta o modelo implementado pelo protótipo.

Figura 5.1: Fluxograma do protótipo



A ocorrência de um desvio ou *jump* só existe caso uma regra possua condição validada e ação para outra *chain*. Porém, se na *chain* adicional não existir nenhuma regra com relação de igualdade, subconjunto, superconjunto ou intersecção com a regra que originou o desvio, significa que este é desnecessário e, portanto, a regra pode ser removida. A tabela de regras a seguir demonstra esta incoerência.

Tabela 5.2: Exemplo tabela de regras com desvio incoerente

```
*filter
:INPUT DROP
:FORWARD ACCEPT
:OUTPUT ACCEPT
:rede_rh
-A FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT
-A FORWARD -s 192.168.1.0/255.255.255.0 -j rede_rh
-A FORWARD -s 192.168.2.0/255.255.255.0 -j ACCEPT
-A rede_rh -s 10.0.4.1/255.255.255.255  -j ACCEPT
-A rede_rh -s 10.0.4.2/255.255.255.255  -j ACCEPT
-A rede_rh -s 10.0.4.3/255.255.255.255  -j ACCEPT
COMMIT
```

A tabela 5.2 contém um *dump* de regras do *iptables*. A regra "`-A FORWARD -s 192.168.1.0/255.255.255.0 -j rede_rh`" provoca que pacotes vindos da rede 192.168.1.0/24 sejam verificados pelas regras da *chain* "rede_rh". Entretanto, nenhuma das regras da *chain* "rede_rh" possui condição que trate pacotes com endereço de origem 192.168.1.0/24 e, por isso, desvio não faz sentindo. Neste tipo de situação, a ferramenta gera um aviso de incoerência da regra que originou o desvio.

Outra verificação feita pela ferramenta indica se a *chain* tem validade ou não para o conjunto total de regras. Ainda no exemplo da tabela 5.2, se a regra "`-A FORWARD -s 192.168.1.0/255.255.255.0 -j rede_rh`" for removida, como indicado pela ferramenta pelo motivo exposto no parágrafo anterior, nenhuma outra regra utiliza ação para desviar para verificação de regras da *chain* "rede_rh". Com isso, o conjunto de regras da *chain* "rede_rh" torna-se desnecessário e, portanto, pode ser removido.

Esta última verificação é efetuada pela ferramenta durante o processo de identificação de relação entre a regra com ação de desvio para a *chain* adicional e as regras da *chain* adicional. Caso exista pelo menos uma relação de igualdade, subconjunto, superconjunto ou intersecção da regra que originou o desvio com uma das regras da *chain* adicional, então a *chain* adicional é relevante ao conjunto de regras. O nome da *chain* é registrado num conjunto de *jumps* válidos e, ao final todas as comparações, as *chains* adicionais que não se encontrarem neste conjunto são informadas ao administrador como desnecessárias.

Após a verificação, se o administrador resolver as incoerências apresentadas pela ferramenta, as regras da tabela 5.2 resultariam nas regras da tabela 5.3, abaixo:

Tabela 5.3: Exemplo tabela de regras com desvio incoerente

```
*filter
:INPUT DROP
:FORWARD ACCEPT
:OUTPUT ACCEPT
-A FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT
-A FORWARD -s 192.168.2.0/255.255.255.0 -j ACCEPT
COMMIT
```

5.3 Estudo de casos verificados com o protótipo

O conjunto de regras apresentado no estudo de caso 1 e no estudo de caso 3 foi retirado de ambientes de produção, os quais possuíam filtros *iptables* implementados com as respectivas regras apresentadas. Os endereços IP originais foram substituídos por IPs destinados a documentações de exemplo na RFC 3330.

5.3.1 Estudo de Caso 1

As incoerências apresentadas na análise abaixo são reportadas através da verificação da ferramenta. Somente as incoerências consideradas erros estão reportadas. Relações de Normal/Aviso foram ignoradas. A tabela 5.4 apresenta a tabela de regras que foi extraída. A tabela 5.5 apresenta o resultado após a remoção das incoerências apresentadas.

Tabela 5.4: Tabela de Regras estudo de caso 1

```
1:-A FORWARD -o eth0 -j LOG --log-prefix "BANDWIDTH_OUT:" --log-level 7
2:-A FORWARD -i eth0 -j LOG --log-prefix "BANDWIDTH_IN:" --log-level 7
3:-A FORWARD -d 64.124.41.0/255.255.255.0 -j REJECT --reject-with icmp-port-unreachable
4:-A FORWARD -d 216.35.208.0/255.255.255.0 -j REJECT --reject-with icmp-port-
unreachable
5:-A FORWARD -p tcp -m tcp --dport 6346 -j REJECT --reject-with icmp-port-unreachable
6:-A FORWARD -p tcp -m tcp --dport 6346 -j REJECT --reject-with icmp-port-unreachable
7:-A FORWARD -d 209.61.186.0/255.255.255.0 -j REJECT --reject-with icmp-port-
unreachable
8:-A FORWARD -d 64.49.201.0/255.255.255.0 -j REJECT --reject-with icmp-port-unreachable
9:-A FORWARD -d 209.25.178.0/255.255.255.0 -j REJECT --reject-with icmp-port-
unreachable
10:-A FORWARD -d 206.142.53.0/255.255.255.0 -j REJECT --reject-with icmp-port-
unreachable
11:-A FORWARD -p tcp -m tcp --dport 1214 -j REJECT --reject-with icmp-port-unreachable
12:-A FORWARD -d 213.248.112.0/255.255.255.0 -j REJECT --reject-with icmp-port-
unreachable
13:-A FORWARD -p tcp -m tcp --dport 1214 -j REJECT --reject-with icmp-port-unreachable
14:-A FORWARD -p tcp -m tcp --dport 6346 -j REJECT --reject-with icmp-port-unreachable
15:-A FORWARD -d 64.245.58.0/255.255.254.0 -j REJECT --reject-with icmp-port-
unreachable
16:-A FORWARD -p tcp -m tcp --dport 5190 -j REJECT --reject-with icmp-port-unreachable
17:-A FORWARD -d 64.12.200.89 -j REJECT --reject-with icmp-port-unreachable
18:-A FORWARD -p tcp -m tcp --dport 5190 -j REJECT --reject-with icmp-port-unreachable
19:-A FORWARD -d 64.12.200.89 -j REJECT --reject-with icmp-port-unreachable
20:-A FORWARD -d 216.136.233.128 -j REJECT --reject-with icmp-port-unreachable
21:-A FORWARD -d 216.136.226.208 -j REJECT --reject-with icmp-port-unreachable
22:-A FORWARD -d 216.136.233.137 -j REJECT --reject-with icmp-port-unreachable
23:-A FORWARD -d 216.136.233.138 -j REJECT --reject-with icmp-port-unreachable
24:-A FORWARD -i lo -o eth1 -j ACCEPT
25:-A FORWARD -i eth1 -o lo -j ACCEPT
26:-A FORWARD -i eth0 -o eth1 -m state --state RELATED,ESTABLISHED -j ACCEPT
27:-A FORWARD -i eth1 -o eth0 -j ACCEPT
```

```
28:-A FORWARD -s 201.15.129.10 -d 192.0.2.240/255.255.255.240 -j ACCEPT
29:-A FORWARD -s 201.15.131.91 -d 192.0.2.240/255.255.255.240 -j ACCEPT
30:-A FORWARD -s 200.102.231.34 -d 192.0.2.240/255.255.255.240 -j ACCEPT
31:-A FORWARD -s 200.213.46.2 -d 192.0.2.254 -j ACCEPT
32:-A FORWARD -d 192.0.2.240/255.255.255.240 -p tcp -m tcp --dport 3389 -j ACCEPT
33:-A FORWARD -d 192.0.2.240/255.255.255.240 -p tcp -m tcp --dport 80 -j ACCEPT
34:-A FORWARD -s 192.0.2.224/255.255.255.240 -p tcp -m tcp --dport 25 -j ACCEPT
35:-A FORWARD -d 192.0.2.254 -p tcp -m tcp --dport 25 -j ACCEPT
36:-A FORWARD -d 192.0.2.240/255.255.255.240 -p tcp -m tcp --dport 1494 -j ACCEPT
37:-A FORWARD -d 192.0.2.240/255.255.255.240 -p tcp -m tcp --dport 110 -j ACCEPT
38:-A FORWARD -d 192.0.2.240/255.255.255.240 -p tcp -m tcp --dport 21 -j ACCEPT
39:-A FORWARD -d 192.0.2.240/255.255.255.240 -p tcp -m tcp --dport 443 -j ACCEPT
40:-A FORWARD -d 192.0.2.224/255.255.255.240 -p tcp -m tcp --dport 80 -j ACCEPT
41:-A FORWARD -d 192.0.2.224/255.255.255.240 -p tcp -m tcp --dport 10000 -j ACCEPT
42:-A FORWARD -d 192.0.2.224/255.255.255.240 -p udp -m udp --dport 53 -j ACCEPT
43:-A FORWARD -d 192.0.2.224/255.255.255.240 -p tcp -m tcp --dport 53 -j ACCEPT
44:-A FORWARD -d 192.0.2.224/255.255.255.240 -p udp -m udp --dport 53 -j ACCEPT
45:-A FORWARD -s 192.0.2.240/255.255.255.240 -j ACCEPT
46:-A FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT
47:-A FORWARD -j DROP
48:-A INPUT -i eth0 -j LOG --log-prefix "BANDWIDTH_IN:" --log-level 7
49:-A INPUT -p icmp -j ACCEPT
50:-A INPUT -p tcp -m tcp --dport 80 -j ACCEPT
51:-A INPUT -p tcp -m tcp --dport 25 -j ACCEPT
52:-A INPUT -p udp -m udp --dport 53 -j ACCEPT
53:-A INPUT -p tcp -m tcp --dport 53 -j ACCEPT
54:-A INPUT -s 192.0.2.240/255.255.255.240 -d 192.0.2.224/255.255.255.240 -j ACCEPT
55:-A INPUT -s 192.0.2.224/255.255.255.240 -d 192.0.2.240/255.255.255.240 -j ACCEPT
56:-A INPUT -p tcp -m tcp --dport 10000 -j ACCEPT
57:-A INPUT -s 127.0.0.1 -j ACCEPT
58:-A INPUT -s 192.0.2.226 -j ACCEPT
59:-A INPUT -d 192.0.2.226 -m state --state RELATED,ESTABLISHED -j ACCEPT
60:-A INPUT -d 192.0.2.235 -m state --state RELATED,ESTABLISHED -j ACCEPT
61:-A INPUT -d 192.0.2.241 -m state --state RELATED,ESTABLISHED -j ACCEPT
62:-A INPUT -d 127.0.0.1 -m state --state RELATED,ESTABLISHED -j ACCEPT
63:-A INPUT -i lo -j ACCEPT
64:-A INPUT -p tcp -m tcp --dport 522 -j ACCEPT
65:-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
66:-A INPUT -j DROP
67:-A OUTPUT -o eth0 -j LOG --log-prefix "BANDWIDTH_OUT:" --log-level 7
68:-A OUTPUT -o lo -j loopback
69:-A OUTPUT -j ACCEPT
70:-A loopback -i lo -j ACCEPT
71:-A BLACKLIST -j LOG --log-prefix "(blacklisted drop)" --log-level 5
72:-A BLACKLIST -j DROP
```

5.3.1.1 Verificação de relação entre regra com ação para chain adicional e regras da chain adicional

Na *chain* OUTPUT a condição para interface de saída via lo (*-o lo*) desvia para a *chain* adicional loopback (*-j loopback*):

-A OUTPUT -o lo -j loopback

-A loopback -i lo -j ACCEPT

Na *chain* loopback a única regra existente tem condição disjunta à regra anterior, portanto, a regra "*-A OUTPUT -o lo -j loopback*" é desnecessária.

5.3.1.2 Verificação de chains desnecessárias

As seguintes *chains* não são utilizadas por nenhuma regra e portanto, devem ser removidas: BLACKLIST.

A única referência para a *chain* loopback é desnecessária e portanto, loopback também torna-se desnecessária.

5.3.1.3 Verificações na Chain INPUT:

59x65: ERRO, Redundância Desnecessária de RA
RA -> 59-A INPUT -d 192.0.2.226 -m state --state RELATED,ESTABLISHED -j ACCEPT
RB -> 65-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT

60x65: ERRO, Redundância Desnecessária de RA
RA -> 60-A INPUT -d 192.0.2.235 -m state --state RELATED,ESTABLISHED -j ACCEPT
RB -> 65-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT

61x65: ERRO, Redundância Desnecessária de RA
RA -> 61-A INPUT -d 192.0.2.241 -m state --state RELATED,ESTABLISHED -j ACCEPT
RB -> 65-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT

62x65: ERRO, Redundância Desnecessária de RA
RA -> 62-A INPUT -d 127.0.0.1 -m state --state RELATED,ESTABLISHED -j ACCEPT
RB -> 65-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT

5.3.1.4Verificações na Chain FORWARD:

5x6: ERRO, Redundância Desnecessária
RA -> 5-A FORWARD -p tcp -m tcp --dport 6346 -j REJECT --reject-with icmp-port-unreachable
RB -> 6-A FORWARD -p tcp -m tcp --dport 6346 -j REJECT --reject-with icmp-port-unreachable

5x14: ERRO, Redundância Desnecessária
RA -> 5-A FORWARD -p tcp -m tcp --dport 6346 -j REJECT --reject-with icmp-port-unreachable
RB -> 14-A FORWARD -p tcp -m tcp --dport 6346 -j REJECT --reject-with icmp-port-unreachable

6x14: ERRO, Redundância Desnecessária
RA -> 6-A FORWARD -p tcp -m tcp --dport 6346 -j REJECT --reject-with icmp-port-unreachable
RB -> 14-A FORWARD -p tcp -m tcp --dport 6346 -j REJECT --reject-with icmp-port-unreachable

11x13: ERRO, Redundância Desnecessária
RA -> 11-A FORWARD -p tcp -m tcp --dport 1214 -j REJECT --reject-with icmp-port-unreachable
RB -> 13-A FORWARD -p tcp -m tcp --dport 1214 -j REJECT --reject-with icmp-port-unreachable

16x18: ERRO, Redundância Desnecessária
RA -> 16-A FORWARD -p tcp -m tcp --dport 5190 -j REJECT --reject-with icmp-port-unreachable

RB -> 18-A FORWARD -p tcp -m tcp --dport 5190 -j REJECT --reject-with icmp-port-unreachable

17x19: ERRO, Redundância Desnecessária

RA -> 17-A FORWARD -d 64.12.200.89 -j REJECT --reject-with icmp-port-unreachable
RB -> 19-A FORWARD -d 64.12.200.89 -j REJECT --reject-with icmp-port-unreachable

26x46: ERRO, Redundância Desnecessária de RA

RA -> 26-A FORWARD -i eth0 -o eth1 -m state --state RELATED,ESTABLISHED -j ACCEPT
RB -> 46-A FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT

42x44: ERRO, Redundância Desnecessária

RA -> 42-A FORWARD -d 192.0.2.224/255.255.255.240 -p udp -m udp --dport 53 -j ACCEPT
RB -> 44-A FORWARD -d 192.0.2.224/255.255.255.240 -p udp -m udp --dport 53 -j ACCEPT

Tabela 5.5: Tabela de regras resultante com remoção das incoerências apresentadas

```
1:-A FORWARD -o eth0 -j LOG --log-prefix "BANDWIDTH_OUT:" --log-level 7
2:-A FORWARD -i eth0 -j LOG --log-prefix "BANDWIDTH_IN:" --log-level 7
3:-A FORWARD -d 64.124.41.0/255.255.255.0 -j REJECT --reject-with icmp-port-unreachable
4:-A FORWARD -d 216.35.208.0/255.255.255.0 -j REJECT --reject-with icmp-port-
unreachable
5:-A FORWARD -p tcp -m tcp --dport 6346 -j REJECT --reject-with icmp-port-unreachable
6: A FORWARD -p tcp -m tcp --dport 6346 -j REJECT --reject-with icmp-port-unreachable
7:-A FORWARD -d 209.61.186.0/255.255.255.0 -j REJECT --reject-with icmp-port-
unreachable
8:-A FORWARD -d 64.49.201.0/255.255.255.0 -j REJECT --reject-with icmp-port-unreachable
9:-A FORWARD -d 209.25.178.0/255.255.255.0 -j REJECT --reject-with icmp-port-
unreachable
10:-A FORWARD -d 206.142.53.0/255.255.255.0 -j REJECT --reject-with icmp-port-
unreachable
11:-A FORWARD -p tcp -m tcp --dport 1214 -j REJECT --reject-with icmp-port-unreachable
12:-A FORWARD -d 213.248.112.0/255.255.255.0 -j REJECT --reject-with icmp-port-
unreachable
13:-A FORWARD -p tcp -m tcp --dport 1214 -j REJECT --reject-with icmp-port-unreachable
14:-A FORWARD -p tcp -m tcp --dport 6346 -j REJECT --reject-with icmp-port-unreachable
15:-A FORWARD -d 64.245.58.0/255.255.254.0 -j REJECT --reject-with icmp-port-
unreachable
16:-A FORWARD -p tcp -m tcp --dport 5190 -j REJECT --reject-with icmp-port-unreachable
17:-A FORWARD -d 64.12.200.89 -j REJECT --reject-with icmp-port-unreachable
18:-A FORWARD -p tcp -m tcp --dport 5190 -j REJECT --reject-with icmp-port-unreachable
19:-A FORWARD -d 64.12.200.89 -j REJECT --reject-with icmp-port-unreachable
20:-A FORWARD -d 216.136.233.128 -j REJECT --reject-with icmp-port-unreachable
21:-A FORWARD -d 216.136.226.208 -j REJECT --reject-with icmp-port-unreachable
22:-A FORWARD -d 216.136.233.137 -j REJECT --reject-with icmp-port-unreachable
23:-A FORWARD -d 216.136.233.138 -j REJECT --reject-with icmp-port-unreachable
24:-A FORWARD -i lo -o eth1 -j ACCEPT
25:-A FORWARD -i eth1 -o lo -j ACCEPT
26:-A FORWARD -i eth0 -o eth1 -m state --state RELATED,ESTABLISHED -j ACCEPT
27:-A FORWARD -i eth1 -o eth0 -j ACCEPT
28:-A FORWARD -s 201.15.129.10 -d 192.0.2.240/255.255.255.240 -j ACCEPT
29:-A FORWARD -s 201.15.131.91 -d 192.0.2.240/255.255.255.240 -j ACCEPT
30:-A FORWARD -s 200.102.231.34 -d 192.0.2.240/255.255.255.240 -j ACCEPT
31:-A FORWARD -s 200.213.46.2 -d 192.0.2.254 -j ACCEPT
32:-A FORWARD -d 192.0.2.240/255.255.255.240 -p tcp -m tcp --dport 3389 -j ACCEPT
33:-A FORWARD -d 192.0.2.240/255.255.255.240 -p tcp -m tcp --dport 80 -j ACCEPT
34:-A FORWARD -s 192.0.2.224/255.255.255.240 -p tcp -m tcp --dport 25 -j ACCEPT
35:-A FORWARD -d 192.0.2.254 -p tcp -m tcp --dport 25 -j ACCEPT
36:-A FORWARD -d 192.0.2.240/255.255.255.240 -p tcp -m tcp --dport 1494 -j ACCEPT
37:-A FORWARD -d 192.0.2.240/255.255.255.240 -p tcp -m tcp --dport 110 -j ACCEPT
38:-A FORWARD -d 192.0.2.240/255.255.255.240 -p tcp -m tcp --dport 21 -j ACCEPT
39:-A FORWARD -d 192.0.2.240/255.255.255.240 -p tcp -m tcp --dport 443 -j ACCEPT
40:-A FORWARD -d 192.0.2.224/255.255.255.240 -p tcp -m tcp --dport 80 -j ACCEPT
41:-A FORWARD -d 192.0.2.224/255.255.255.240 -p tcp -m tcp --dport 10000 -j ACCEPT
```

```
42:-A FORWARD -d 192.0.2.224/255.255.255.240 -p udp -m udp --dport 53 -j ACCEPT
43:-A FORWARD -d 192.0.2.224/255.255.255.240 -p tcp -m tcp --dport 53 -j ACCEPT
44:-A FORWARD -d 192.0.2.224/255.255.255.240 -p udp -m udp --dport 53 -j ACCEPT
45:-A FORWARD -s 192.0.2.240/255.255.255.240 -j ACCEPT
46:-A FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT
47:-A FORWARD -j DROP
48:-A INPUT -i eth0 -j LOG --log-prefix "BANDWIDTH_IN:" --log-level 7
49:-A INPUT -p icmp -j ACCEPT
50:-A INPUT -p tcp -m tcp --dport 80 -j ACCEPT
51:-A INPUT -p tcp -m tcp --dport 25 -j ACCEPT
52:-A INPUT -p udp -m udp --dport 53 -j ACCEPT
53:-A INPUT -p tcp -m tcp --dport 53 -j ACCEPT
54:-A INPUT -s 192.0.2.240/255.255.255.240 -d 192.0.2.224/255.255.255.240 -j ACCEPT
55:-A INPUT -s 192.0.2.224/255.255.255.240 -d 192.0.2.240/255.255.255.240 -j ACCEPT
56:-A INPUT -p tcp -m tcp --dport 10000 -j ACCEPT
57:-A INPUT -s 127.0.0.1 -j ACCEPT
58:-A INPUT -s 192.0.2.226 -j ACCEPT
59:-A INPUT -d 192.0.2.226 -m state --state RELATED,ESTABLISHED -j ACCEPT
60:-A INPUT -d 192.0.2.235 -m state --state RELATED,ESTABLISHED -j ACCEPT
61:-A INPUT -d 192.0.2.241 -m state --state RELATED,ESTABLISHED -j ACCEPT
62:-A INPUT -d 127.0.0.1 -m state --state RELATED,ESTABLISHED -j ACCEPT
63:-A INPUT -i lo -j ACCEPT
64:-A INPUT -p tcp -m tcp --dport 522 -j ACCEPT
65:-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
66:-A INPUT -j DROP
67:-A OUTPUT -o eth0 -j LOG --log-prefix "BANDWIDTH_OUT:" --log-level 7
68:-A OUTPUT -o lo -j loopback
69:-A OUTPUT -j ACCEPT
70:-A loopback -i lo -j ACCEPT
71:-A BLACKLIST -j LOG --log-prefix "(blacklisted drop)" --log-level 5
72:-A BLACKLIST -j DROP
```

Eliminando-se estas regras o protótipo não indica mais nenhuma mensagem de erro.

5.3.2 Estudo de Caso 2

As regras apresentadas neste exemplo de caso foram baseadas no artigo de Al-Shaer (2003). O objetivo deste estudo de caso visa fazer uma comparação dos diagnósticos apresentados pelo modelo sugerido por este trabalho com os apresentados por Al-Shaer.

Tabela 5.6: Tabela de regras estudo de caso 2

```
1:-A FORWARD -s 140.192.37.20 -p tcp -m tcp --dport 80 -j DROP
2:-A FORWARD -s 140.0.0.0/255.0.0.0 -p tcp -m tcp --dport 80 -j ACCEPT
3:-A FORWARD -d 161.120.33.40 -p tcp -m tcp --dport 80 -j ACCEPT
4:-A FORWARD -s 140.0.0.0/255.0.0.0 -d 161.120.33.40 -p tcp -m tcp --dport 80 -j DROP
5:-A FORWARD -s 140.192.37.30 -p tcp -m tcp --dport 21 -j DROP
6:-A FORWARD -s 140.0.0.0/255.0.0.0 -p tcp -m tcp --dport 21 -j ACCEPT
7:-A FORWARD -s 140.0.0.0/255.0.0.0 -d 161.120.33.40 -p tcp -m tcp --dport 21 -j ACCEPT
8:-A FORWARD -p tcp -j ACCEPT
9:-A FORWARD -s 140.0.0.0/255.0.0.0 -d 161.120.33.40 -p udp -m udp --dport 53 -j ACCEPT
10:-A FORWARD -d 161.120.33.40 -p udp -m udp --dport 53 -j ACCEPT
11:-A FORWARD -p udp -j DROP
```

5.3.2.1 Diagnóstico segundo modelo de Al-Shaer

- Regra 2 é uma generalização da rega 1;
- regra 3 está correlacionada com regra 1;
- regra 4 está sobreposta pela regra 2;

- regra 4 está sobreposta pela regra 3;
- regra 6 é uma generalização da regra 5;
- regra 7 é correlacionada com regra 5;
- regra 7 é redundante para regra 6;
- regra 8 é uma generalização da regra 1;
- regra 8 é uma generalização da regra 4;
- regra 8 é uma generalização da regra 5;
- regra 9 é redundante para regra 10;
- regra 11 é uma generalização da regra 9;
- regra 11 é uma generalização da regra 10;

Tabela 5.7: Tabela de regras resultante utilizando o modelo de Al-Shaer

```
1:-A FORWARD -s 140.192.37.20 -p tcp -m tcp --dport 80 -j DROP
2:-A FORWARD -s 140.0.0.0/255.0.0.0 -p tcp -m tcp --dport 80 -j ACCEPT
3:-A FORWARD -d 161.120.33.40 -p tcp -m tcp --dport 80 -j ACCEPT
4:-A FORWARD -s 140.0.0.0/255.0.0.0 -d 161.120.33.40 -p tcp -m tcp --dport 80 -j DROP
5:-A FORWARD -s 140.192.37.0 -p tcp -m tcp --dport 21 -j DROP
6:-A FORWARD -s 140.0.0.0/255.0.0.0 -p tcp -m tcp --dport 21 -j ACCEPT
7:-A FORWARD -s 140.0.0.0/255.0.0.0 -d 161.120.33.40 -p tcp -m tcp --dport 21 -j ACCEPT
8:-A FORWARD -p tcp -j ACCEPT
9:-A FORWARD -s 140.0.0.0/255.0.0.0 -d 161.120.33.40 -p udp -m udp --dport 53 -j ACCEPT
10:-A FORWARD -d 161.120.33.40 -p udp -m udp --dport 53 -j ACCEPT
11:-A FORWARD -p udp -j DROP
```

5.3.2.2 Diagnóstico segundo modelo apresentado neste trabalho

Neste estudo de caso as caracterizações que resultaram em diagnósticos tipo aviso também são apresentados:

1x2: NORMAL/AVISO!
RA -> 1-A FORWARD -s 140.192.37.20 -p tcp -m tcp --dport 80 -j DROP
RB -> 2-A FORWARD -s 140.0.0.0/255.0.0.0 -p tcp -m tcp --dport 80 -j ACCEPT

1x3: NORMAL/AVISO!
RA -> 1-A FORWARD -s 140.192.37.20 -p tcp -m tcp --dport 80 -j DROP
RB -> 3-A FORWARD -d 161.120.33.40 -p tcp -m tcp --dport 80 -j ACCEPT

1x4: Normal com Redundância Parcial
RA -> 1-A FORWARD -s 140.192.37.20 -p tcp -m tcp --dport 80 -j DROP
RB -> 4-A FORWARD -s 140.0.0.0/255.0.0.0 -d 161.120.33.40 -p tcp -m tcp --dport 80 -j DROP

1x8: NORMAL/AVISO!
RA -> 1-A FORWARD -s 140.192.37.20 -p tcp -m tcp --dport 80 -j DROP
RB -> 8-A FORWARD -p tcp -j ACCEPT

2x3: Normal com Redundância Parcial
RA -> 2-A FORWARD -s 140.0.0.0/255.0.0.0 -p tcp -m tcp --dport 80 -j ACCEPT
RB -> 3-A FORWARD -d 161.120.33.40 -p tcp -m tcp --dport 80 -j ACCEPT

2x4: ERRO, RB nunca será ativado. Necessário reordenar ou excluir RB
RA -> 2-A FORWARD -s 140.0.0.0/255.0.0.0 -p tcp -m tcp --dport 80 -j ACCEPT
RB -> 4-A FORWARD -s 140.0.0.0/255.0.0.0 -d 161.120.33.40 -p tcp -m tcp --dport 80 -j DROP

2x8: ERRO, Redundância Desnecessária de RA
RA -> 2-A FORWARD -s 140.0.0.0/255.0.0.0 -p tcp -m tcp --dport 80 -j ACCEPT
RB -> 8-A FORWARD -p tcp -j ACCEPT

3x4: ERRO, RB nunca será ativado. Necessário reordenar ou excluir RB
RA -> 3-A FORWARD -d 161.120.33.40 -p tcp -m tcp --dport 80 -j ACCEPT
RB -> 4-A FORWARD -s 140.0.0.0/255.0.0.0 -d 161.120.33.40 -p tcp -m tcp --dport 80 -j DROP

3x8: ERRO, Redundância Desnecessária de RA
RA -> 3-A FORWARD -d 161.120.33.40 -p tcp -m tcp --dport 80 -j ACCEPT
RB -> 8-A FORWARD -p tcp -j ACCEPT

4x8: NORMAL/AVISO!
RA -> 4-A FORWARD -s 140.0.0.0/255.0.0.0 -d 161.120.33.40 -p tcp -m tcp --dport 80 -j DROP
RB -> 8-A FORWARD -p tcp -j ACCEPT

5x6: NORMAL/AVISO!
RA -> 5-A FORWARD -s 140.192.37.30 -p tcp -m tcp --dport 21 -j DROP
RB -> 6-A FORWARD -s 140.0.0.0/255.0.0.0 -p tcp -m tcp --dport 21 -j ACCEPT

5x7: NORMAL/AVISO!
RA -> 5-A FORWARD -s 140.192.37.30 -p tcp -m tcp --dport 21 -j DROP
RB -> 7-A FORWARD -s 140.0.0.0/255.0.0.0 -d 161.120.33.40 -p tcp -m tcp --dport 21 -j ACCEPT

5x8: NORMAL/AVISO!
RA -> 5-A FORWARD -s 140.192.37.30 -p tcp -m tcp --dport 21 -j DROP
RB -> 8-A FORWARD -p tcp -j ACCEPT

6x7: ERRO, Redundância Desnecessária de RB
RA -> 6-A FORWARD -s 140.0.0.0/255.0.0.0 -p tcp -m tcp --dport 21 -j ACCEPT
RB -> 7-A FORWARD -s 140.0.0.0/255.0.0.0 -d 161.120.33.40 -p tcp -m tcp --dport 21 -j ACCEPT

6x8: ERRO, Redundância Desnecessária de RA
RA -> 6-A FORWARD -s 140.0.0.0/255.0.0.0 -p tcp -m tcp --dport 21 -j ACCEPT
RB -> 8-A FORWARD -p tcp -j ACCEPT

7x8: ERRO, Redundância Desnecessária de RA
RA -> 7-A FORWARD -s 140.0.0.0/255.0.0.0 -d 161.120.33.40 -p tcp -m tcp --dport 21 -j ACCEPT
RB -> 8-A FORWARD -p tcp -j ACCEPT

9x10: ERRO, Redundância Desnecessária de RA
RA -> 9-A FORWARD -s 140.0.0.0/255.0.0.0 -d 161.120.33.40 -p udp -m udp --dport 53 -j ACCEPT
RB -> 10-A FORWARD -d 161.120.33.40 -p udp -m udp --dport 53 -j ACCEPT

9x11: NORMAL/AVISO!
RA -> 9-A FORWARD -s 140.0.0.0/255.0.0.0 -d 161.120.33.40 -p udp -m udp --dport 53 -j ACCEPT
RB -> 11-A FORWARD -p udp -j DROP

10x11: NORMAL/AVISO!
RA -> 10-A FORWARD -d 161.120.33.40 -p udp -m udp --dport 53 -j ACCEPT
RB -> 11-A FORWARD -p udp -j DROP

Tabela 5.8: Tabela de regras apresentando as incoerências segundo modelo proposto

```
1:-A FORWARD -s 140.192.37.20 -p tcp -m tcp --dport 80 -j DROP
2: A FORWARD -s 140.0.0.0/255.0.0.0 -p tcp -m tcp --dport 80 -j ACCEPT
3: A FORWARD -d 161.120.33.40 -p tcp -m tcp --dport 80 -j ACCEPT
4: A FORWARD -s 140.0.0.0/255.0.0.0 -d 161.120.33.40 -p tcp -m tcp --dport 80 -j DROP
5:-A FORWARD -s 140.192.37.30 -p tcp -m tcp --dport 21 -j DROP
6: A FORWARD -s 140.0.0.0/255.0.0.0 -p tcp -m tcp --dport 21 -j ACCEPT
7: A FORWARD -s 140.0.0.0/255.0.0.0 -d 161.120.33.40 -p tcp -m tcp --dport 21 -j ACCEPT
8:-A FORWARD -p tcp -j ACCEPT
9: A FORWARD -s 140.0.0.0/255.0.0.0 -d 161.120.33.40 -p udp -m udp --dport 53 -j ACCEPT
10:-A FORWARD -d 161.120.33.40 -p udp -m udp --dport 53 -j ACCEPT
11:-A FORWARD -p udp -j DROP
```

Removendo-se as regras apontadas como incoerentes tem-se como resultado a tabela de regras apresentadas na tabela 5.9:

Tabela 5.9: Tabela de regras resultante após remoção das incoerências apresentadas

```
-A FORWARD -s 140.192.37.20 -p tcp -m tcp --dport 80 -j DROP
-A FORWARD -s 140.192.37.30 -p tcp -m tcp --dport 21 -j DROP
-A FORWARD -p tcp -j ACCEPT
-A FORWARD -d 161.120.33.40 -p udp -m udp --dport 53 -j ACCEPT
-A FORWARD -p udp -j DROP
```

5.3.2.3 Diferenças apresentadas pelos modelos

Com relação a incoerências que resultam em erros na configuração:

- os dois modelos apresentam, independente de classificação ou mensagem de diagnóstico, as regras 4, 7 e 9 como erro;
- as regras 2, 3 e 6 são apresentadas como incoerências somente pelo modelo proposto por este trabalho; isolando-se a relação entre as regras 7 e 8, que resulta no diagnóstico de redundância desnecessária de RA:

7=RA: FORWARD -s 140.0.0.0/255.0.0.0 -d 161.120.33.40 -p tcp -m tcp --dport 21 -j ACCEPT

8=RB: FORWARD -p tcp -j ACCEPT

A regra 7 é um subconjunto da regra 8 e as duas possuem identidade nas ações. Tal caracterização é um erro para os dois modelos. Se a regra 7 for removida, a política do filtro não sofre alterações. Entretanto, o modelo de Al-Shaer não informa tal ocorrência, pelo fato que a regra 7 participou anteriormente de uma correlação com a regra 5 e, segundo definição de Al-Shaer "*a regra RX é redundante para regra RY, provado que RX não esteja envolvido com qualquer anomalia de generalização ou correlação com outras regras que precedem RY*". O mesmo ocorre nas incoerências apontadas nas relações ***2x8***, ***3x8***, ***6x8***.

Com relação a relações que não resultam em aviso:

– avisos alertados somente pelo modelo proposto;

a comparação entre as regras 2 e 3 resulta em uma relação de intersecção e, como existe identidade na ação, pode-se afirmar que existe uma redundância parcial entre as duas regras; A mesma situação ocorre na relação ***1x4.***

2x3: Normal com Redundância Parcial
RA -> 2-A FORWARD -s 140.0.0.0/255.0.0.0 -p tcp -m tcp --dport 80 -j ACCEPT
RB -> 3-A FORWARD -d 161.120.33.40 -p tcp -m tcp --dport 80 -j ACCEPT

Portanto, com base nas diferenças apresentadas, pode-se dizer que o diagnóstico apresentado pelo modelo proposto neste trabalho, mostrou-se efetivo é mais preciso que o modelo de Al-Shaer.

5.3.3 Estudo de Caso 3

O conjunto de regras apresentadas na tabela 5.10 foi retirado de um filtro cuja objetivo era controlar acesso entre a Internet e a rede interna da empresa. O objetivo do uso da ferramenta neste caso foi para remoção de incoerências para após abstrair-se a política que era implementa pelo filtro.

Tabela 5.10: Tabela de regras estudo de caso 3

```
1:-A INPUT -s 172.16.2.0/255.255.255.0 -d 172.16.2.0/255.255.255.0 -i eth0 -j ACCEPT
2:-A INPUT -s 172.16.3.234 -d 172.16.3.234 -i eth0 -j ACCEPT
3:-A INPUT -s 172.16.3.234 -d 172.16.2.2 -i eth0 -j ACCEPT
4:-A INPUT -s 172.16.3.234 -d 172.16.3.44 -i eth0 -j ACCEPT
5:-A INPUT -s 172.16.3.234 -d 172.16.3.41 -i eth0 -j ACCEPT
6:-A INPUT -s 172.16.3.234 -d 172.16.3.17 -i eth0 -j ACCEPT
7:-A INPUT -s 172.16.3.234 -d 172.16.2.8 -i eth0 -j ACCEPT
8:-A INPUT -s 172.16.3.234 -d 172.16.0.0/255.255.0.0 -i eth0 -j ACCEPT
9:-A INPUT -s 172.16.3.234 -d 172.16.2.149 -i eth0 -j ACCEPT
10:-A INPUT -s 172.16.3.234 -d 172.16.2.17 -i eth0 -j ACCEPT
11:-A INPUT -s 172.16.3.234 -d 172.16.2.3 -i eth0 -j ACCEPT
12:-A INPUT -s 172.16.3.234 -d 172.16.3.215 -i eth0 -j ACCEPT
13:-A INPUT -s 172.16.3.234 -d 172.16.2.145 -i eth0 -j ACCEPT
14:-A INPUT -s 172.16.3.234 -d 172.16.2.241 -i eth0 -j ACCEPT
15:-A INPUT -s 172.16.3.234 -d 172.16.2.49 -i eth0 -j ACCEPT
```

```
16:-A INPUT -s 172.16.3.234 -d 172.16.2.214 -i eth0 -j ACCEPT
17:-A INPUT -s 172.16.3.234 -d 172.16.2.222 -i eth0 -j ACCEPT
18:-A INPUT -s 172.16.2.2 -d 172.16.3.234 -i eth0 -j ACCEPT
19:-A INPUT -s 172.16.2.2 -d 172.16.2.2 -i eth0 -j ACCEPT
20:-A INPUT -s 172.16.2.2 -d 172.16.3.44 -i eth0 -j ACCEPT
21:-A INPUT -s 172.16.2.2 -d 172.16.3.41 -i eth0 -j ACCEPT
22:-A INPUT -s 172.16.2.2 -d 172.16.3.17 -i eth0 -j ACCEPT
23:-A INPUT -s 172.16.2.2 -d 172.16.2.8 -i eth0 -j ACCEPT
24:-A INPUT -s 172.16.2.2 -d 172.16.0.0/255.255.0.0 -i eth0 -j ACCEPT
25:-A INPUT -s 172.16.2.2 -d 172.16.2.149 -i eth0 -j ACCEPT
26:-A INPUT -s 172.16.2.2 -d 172.16.2.17 -i eth0 -j ACCEPT
27:-A INPUT -s 172.16.2.2 -d 172.16.2.3 -i eth0 -j ACCEPT
28:-A INPUT -s 172.16.2.2 -d 172.16.3.215 -i eth0 -j ACCEPT
29:-A INPUT -s 172.16.2.2 -d 172.16.2.145 -i eth0 -j ACCEPT
30:-A INPUT -s 172.16.2.2 -d 172.16.2.241 -i eth0 -j ACCEPT
31:-A INPUT -s 172.16.2.2 -d 172.16.2.49 -i eth0 -j ACCEPT
32:-A INPUT -s 172.16.2.2 -d 172.16.2.214 -i eth0 -j ACCEPT
33:-A INPUT -s 172.16.2.2 -d 172.16.2.222 -i eth0 -j ACCEPT
34:-A INPUT -s 172.16.3.44 -d 172.16.3.234 -i eth0 -j ACCEPT
35:-A INPUT -s 172.16.3.44 -d 172.16.2.2 -i eth0 -j ACCEPT
36:-A INPUT -s 172.16.3.44 -d 172.16.3.44 -i eth0 -j ACCEPT
37:-A INPUT -s 172.16.3.44 -d 172.16.3.41 -i eth0 -j ACCEPT
38:-A INPUT -s 172.16.3.44 -d 172.16.3.17 -i eth0 -j ACCEPT
39:-A INPUT -s 172.16.3.44 -d 172.16.2.8 -i eth0 -j ACCEPT
40:-A INPUT -s 172.16.3.44 -d 172.16.0.0/255.255.0.0 -i eth0 -j ACCEPT
41:-A INPUT -s 172.16.3.44 -d 172.16.2.149 -i eth0 -j ACCEPT
42:-A INPUT -s 172.16.3.44 -d 172.16.2.17 -i eth0 -j ACCEPT
43:-A INPUT -s 172.16.3.44 -d 172.16.2.3 -i eth0 -j ACCEPT
44:-A INPUT -s 172.16.3.44 -d 172.16.3.215 -i eth0 -j ACCEPT
45:-A INPUT -s 172.16.3.44 -d 172.16.2.145 -i eth0 -j ACCEPT
46:-A INPUT -s 172.16.3.44 -d 172.16.2.241 -i eth0 -j ACCEPT
47:-A INPUT -s 172.16.3.44 -d 172.16.2.49 -i eth0 -j ACCEPT
48:-A INPUT -s 172.16.3.44 -d 172.16.2.214 -i eth0 -j ACCEPT
49:-A INPUT -s 172.16.3.44 -d 172.16.2.222 -i eth0 -j ACCEPT
50:-A INPUT -s 172.16.3.41 -d 172.16.3.234 -i eth0 -j ACCEPT
51:-A INPUT -s 172.16.3.41 -d 172.16.2.2 -i eth0 -j ACCEPT
52:-A INPUT -s 172.16.3.41 -d 172.16.3.44 -i eth0 -j ACCEPT
53:-A INPUT -s 172.16.3.41 -d 172.16.3.41 -i eth0 -j ACCEPT
54:-A INPUT -s 172.16.3.41 -d 172.16.3.17 -i eth0 -j ACCEPT
55:-A INPUT -s 172.16.3.41 -d 172.16.2.8 -i eth0 -j ACCEPT
56:-A INPUT -s 172.16.3.41 -d 172.16.0.0/255.255.0.0 -i eth0 -j ACCEPT
57:-A INPUT -s 172.16.3.41 -d 172.16.2.149 -i eth0 -j ACCEPT
58:-A INPUT -s 172.16.3.41 -d 172.16.2.17 -i eth0 -j ACCEPT
59:-A INPUT -s 172.16.3.41 -d 172.16.2.3 -i eth0 -j ACCEPT
60:-A INPUT -s 172.16.3.41 -d 172.16.3.215 -i eth0 -j ACCEPT
61:-A INPUT -s 172.16.3.41 -d 172.16.2.145 -i eth0 -j ACCEPT
62:-A INPUT -s 172.16.3.41 -d 172.16.2.241 -i eth0 -j ACCEPT
63:-A INPUT -s 172.16.3.41 -d 172.16.2.49 -i eth0 -j ACCEPT
64:-A INPUT -s 172.16.3.41 -d 172.16.2.214 -i eth0 -j ACCEPT
65:-A INPUT -s 172.16.3.41 -d 172.16.2.222 -i eth0 -j ACCEPT
66:-A INPUT -s 172.16.3.17 -d 172.16.3.234 -i eth0 -j ACCEPT
67:-A INPUT -s 172.16.3.17 -d 172.16.2.2 -i eth0 -j ACCEPT
68:-A INPUT -s 172.16.3.17 -d 172.16.3.44 -i eth0 -j ACCEPT
69:-A INPUT -s 172.16.3.17 -d 172.16.3.41 -i eth0 -j ACCEPT
70:-A INPUT -s 172.16.3.17 -d 172.16.3.17 -i eth0 -j ACCEPT
71:-A INPUT -s 172.16.3.17 -d 172.16.2.8 -i eth0 -j ACCEPT
72:-A INPUT -s 172.16.3.17 -d 172.16.0.0/255.255.0.0 -i eth0 -j ACCEPT
73:-A INPUT -s 172.16.3.17 -d 172.16.2.149 -i eth0 -j ACCEPT
74:-A INPUT -s 172.16.3.17 -d 172.16.2.17 -i eth0 -j ACCEPT
75:-A INPUT -s 172.16.3.17 -d 172.16.2.3 -i eth0 -j ACCEPT
76:-A INPUT -s 172.16.3.17 -d 172.16.3.215 -i eth0 -j ACCEPT
77:-A INPUT -s 172.16.3.17 -d 172.16.2.145 -i eth0 -j ACCEPT
78:-A INPUT -s 172.16.3.17 -d 172.16.2.241 -i eth0 -j ACCEPT
79:-A INPUT -s 172.16.3.17 -d 172.16.2.49 -i eth0 -j ACCEPT
80:-A INPUT -s 172.16.3.17 -d 172.16.2.214 -i eth0 -j ACCEPT
81:-A INPUT -s 172.16.3.17 -d 172.16.2.222 -i eth0 -j ACCEPT
82:-A INPUT -s 172.16.2.8 -d 172.16.3.234 -i eth0 -j ACCEPT
83:-A INPUT -s 172.16.2.8 -d 172.16.2.2 -i eth0 -j ACCEPT
84:-A INPUT -s 172.16.2.8 -d 172.16.3.44 -i eth0 -j ACCEPT
85:-A INPUT -s 172.16.2.8 -d 172.16.3.41 -i eth0 -j ACCEPT
86:-A INPUT -s 172.16.2.8 -d 172.16.3.17 -i eth0 -j ACCEPT
87:-A INPUT -s 172.16.2.8 -d 172.16.2.8 -i eth0 -j ACCEPT
88:-A INPUT -s 172.16.2.8 -d 172.16.0.0/255.255.0.0 -i eth0 -j ACCEPT
89:-A INPUT -s 172.16.2.8 -d 172.16.2.149 -i eth0 -j ACCEPT
90:-A INPUT -s 172.16.2.8 -d 172.16.2.17 -i eth0 -j ACCEPT
```

```
91:-A INPUT -s 172.16.2.8 -d 172.16.2.3 -i eth0 -j ACCEPT
92:-A INPUT -s 172.16.2.8 -d 172.16.3.215 -i eth0 -j ACCEPT
93:-A INPUT -s 172.16.2.8 -d 172.16.2.145 -i eth0 -j ACCEPT
94:-A INPUT -s 172.16.2.8 -d 172.16.2.241 -i eth0 -j ACCEPT
95:-A INPUT -s 172.16.2.8 -d 172.16.2.49 -i eth0 -j ACCEPT
96:-A INPUT -s 172.16.2.8 -d 172.16.2.214 -i eth0 -j ACCEPT
97:-A INPUT -s 172.16.2.8 -d 172.16.2.222 -i eth0 -j ACCEPT
98:-A INPUT -s 172.16.0.0/255.255.0.0 -d 172.16.3.234 -i eth0 -j ACCEPT
99:-A INPUT -s 172.16.0.0/255.255.0.0 -d 172.16.2.2 -i eth0 -j ACCEPT
100:-A INPUT -s 172.16.0.0/255.255.0.0 -d 172.16.3.44 -i eth0 -j ACCEPT
101:-A INPUT -s 172.16.0.0/255.255.0.0 -d 172.16.3.41 -i eth0 -j ACCEPT
102:-A INPUT -s 172.16.0.0/255.255.0.0 -d 172.16.3.17 -i eth0 -j ACCEPT
103:-A INPUT -s 172.16.0.0/255.255.0.0 -d 172.16.2.8 -i eth0 -j ACCEPT
104:-A INPUT -s 172.16.0.0/255.255.0.0 -d 172.16.0.0/255.255.0.0 -i eth0 -j ACCEPT
105:-A INPUT -s 172.16.0.0/255.255.0.0 -d 172.16.2.149 -i eth0 -j ACCEPT
106:-A INPUT -s 172.16.0.0/255.255.0.0 -d 172.16.2.17 -i eth0 -j ACCEPT
107:-A INPUT -s 172.16.0.0/255.255.0.0 -d 172.16.2.3 -i eth0 -j ACCEPT
108:-A INPUT -s 172.16.0.0/255.255.0.0 -d 172.16.3.215 -i eth0 -j ACCEPT
109:-A INPUT -s 172.16.0.0/255.255.0.0 -d 172.16.2.145 -i eth0 -j ACCEPT
110:-A INPUT -s 172.16.0.0/255.255.0.0 -d 172.16.2.241 -i eth0 -j ACCEPT
111:-A INPUT -s 172.16.0.0/255.255.0.0 -d 172.16.2.49 -i eth0 -j ACCEPT
112:-A INPUT -s 172.16.0.0/255.255.0.0 -d 172.16.2.214 -i eth0 -j ACCEPT
113:-A INPUT -s 172.16.0.0/255.255.0.0 -d 172.16.2.222 -i eth0 -j ACCEPT
114:-A INPUT -s 172.16.2.149 -d 172.16.3.234 -i eth0 -j ACCEPT
115:-A INPUT -s 172.16.2.149 -d 172.16.2.2 -i eth0 -j ACCEPT
116:-A INPUT -s 172.16.2.149 -d 172.16.3.44 -i eth0 -j ACCEPT
117:-A INPUT -s 172.16.2.149 -d 172.16.3.41 -i eth0 -j ACCEPT
118:-A INPUT -s 172.16.2.149 -d 172.16.3.17 -i eth0 -j ACCEPT
119:-A INPUT -s 172.16.2.149 -d 172.16.2.8 -i eth0 -j ACCEPT
120:-A INPUT -s 172.16.2.149 -d 172.16.0.0/255.255.0.0 -i eth0 -j ACCEPT
121:-A INPUT -s 172.16.2.149 -d 172.16.2.149 -i eth0 -j ACCEPT
122:-A INPUT -s 172.16.2.149 -d 172.16.2.17 -i eth0 -j ACCEPT
123:-A INPUT -s 172.16.2.149 -d 172.16.2.3 -i eth0 -j ACCEPT
124:-A INPUT -s 172.16.2.149 -d 172.16.3.215 -i eth0 -j ACCEPT
125:-A INPUT -s 172.16.2.149 -d 172.16.2.145 -i eth0 -j ACCEPT
126:-A INPUT -s 172.16.2.149 -d 172.16.2.241 -i eth0 -j ACCEPT
127:-A INPUT -s 172.16.2.149 -d 172.16.2.49 -i eth0 -j ACCEPT
128:-A INPUT -s 172.16.2.149 -d 172.16.2.214 -i eth0 -j ACCEPT
129:-A INPUT -s 172.16.2.149 -d 172.16.2.222 -i eth0 -j ACCEPT
130:-A INPUT -s 172.16.2.17 -d 172.16.3.234 -i eth0 -j ACCEPT
131:-A INPUT -s 172.16.2.17 -d 172.16.2.2 -i eth0 -j ACCEPT
132:-A INPUT -s 172.16.2.17 -d 172.16.3.44 -i eth0 -j ACCEPT
133:-A INPUT -s 172.16.2.17 -d 172.16.3.41 -i eth0 -j ACCEPT
134:-A INPUT -s 172.16.2.17 -d 172.16.3.17 -i eth0 -j ACCEPT
135:-A INPUT -s 172.16.2.17 -d 172.16.2.8 -i eth0 -j ACCEPT
136:-A INPUT -s 172.16.2.17 -d 172.16.0.0/255.255.0.0 -i eth0 -j ACCEPT
137:-A INPUT -s 172.16.2.17 -d 172.16.2.149 -i eth0 -j ACCEPT
138:-A INPUT -s 172.16.2.17 -d 172.16.2.17 -i eth0 -j ACCEPT
139:-A INPUT -s 172.16.2.17 -d 172.16.2.3 -i eth0 -j ACCEPT
140:-A INPUT -s 172.16.2.17 -d 172.16.3.215 -i eth0 -j ACCEPT
141:-A INPUT -s 172.16.2.17 -d 172.16.2.145 -i eth0 -j ACCEPT
142:-A INPUT -s 172.16.2.17 -d 172.16.2.241 -i eth0 -j ACCEPT
143:-A INPUT -s 172.16.2.17 -d 172.16.2.49 -i eth0 -j ACCEPT
144:-A INPUT -s 172.16.2.17 -d 172.16.2.214 -i eth0 -j ACCEPT
145:-A INPUT -s 172.16.2.17 -d 172.16.2.222 -i eth0 -j ACCEPT
146:-A INPUT -s 172.16.2.3 -d 172.16.3.234 -i eth0 -j ACCEPT
147:-A INPUT -s 172.16.2.3 -d 172.16.2.2 -i eth0 -j ACCEPT
148:-A INPUT -s 172.16.2.3 -d 172.16.3.44 -i eth0 -j ACCEPT
149:-A INPUT -s 172.16.2.3 -d 172.16.3.41 -i eth0 -j ACCEPT
150:-A INPUT -s 172.16.2.3 -d 172.16.3.17 -i eth0 -j ACCEPT
151:-A INPUT -s 172.16.2.3 -d 172.16.2.8 -i eth0 -j ACCEPT
152:-A INPUT -s 172.16.2.3 -d 172.16.0.0/255.255.0.0 -i eth0 -j ACCEPT
153:-A INPUT -s 172.16.2.3 -d 172.16.2.149 -i eth0 -j ACCEPT
154:-A INPUT -s 172.16.2.3 -d 172.16.2.17 -i eth0 -j ACCEPT
155:-A INPUT -s 172.16.2.3 -d 172.16.2.3 -i eth0 -j ACCEPT
156:-A INPUT -s 172.16.2.3 -d 172.16.3.215 -i eth0 -j ACCEPT
157:-A INPUT -s 172.16.2.3 -d 172.16.2.145 -i eth0 -j ACCEPT
158:-A INPUT -s 172.16.2.3 -d 172.16.2.241 -i eth0 -j ACCEPT
159:-A INPUT -s 172.16.2.3 -d 172.16.2.49 -i eth0 -j ACCEPT
160:-A INPUT -s 172.16.2.3 -d 172.16.2.214 -i eth0 -j ACCEPT
161:-A INPUT -s 172.16.2.3 -d 172.16.2.222 -i eth0 -j ACCEPT
162:-A INPUT -s 172.16.3.215 -d 172.16.3.234 -i eth0 -j ACCEPT
163:-A INPUT -s 172.16.3.215 -d 172.16.2.2 -i eth0 -j ACCEPT
164:-A INPUT -s 172.16.3.215 -d 172.16.3.44 -i eth0 -j ACCEPT
165:-A INPUT -s 172.16.3.215 -d 172.16.3.41 -i eth0 -j ACCEPT
```

```
166:-A INPUT -s 172.16.3.215 -d 172.16.3.17 -i eth0 -j ACCEPT
167:-A INPUT -s 172.16.3.215 -d 172.16.2.8 -i eth0 -j ACCEPT
168:-A INPUT -s 172.16.3.215 -d 172.16.0.0/255.255.0.0 -i eth0 -j ACCEPT
169:-A INPUT -s 172.16.3.215 -d 172.16.2.149 -i eth0 -j ACCEPT
170:-A INPUT -s 172.16.3.215 -d 172.16.2.17 -i eth0 -j ACCEPT
171:-A INPUT -s 172.16.3.215 -d 172.16.2.3 -i eth0 -j ACCEPT
172:-A INPUT -s 172.16.3.215 -d 172.16.3.215 -i eth0 -j ACCEPT
173:-A INPUT -s 172.16.3.215 -d 172.16.2.145 -i eth0 -j ACCEPT
174:-A INPUT -s 172.16.3.215 -d 172.16.2.241 -i eth0 -j ACCEPT
175:-A INPUT -s 172.16.3.215 -d 172.16.2.49 -i eth0 -j ACCEPT
176:-A INPUT -s 172.16.3.215 -d 172.16.2.214 -i eth0 -j ACCEPT
177:-A INPUT -s 172.16.3.215 -d 172.16.2.222 -i eth0 -j ACCEPT
178:-A INPUT -s 172.16.2.145 -d 172.16.3.234 -i eth0 -j ACCEPT
179:-A INPUT -s 172.16.2.145 -d 172.16.2.2 -i eth0 -j ACCEPT
180:-A INPUT -s 172.16.2.145 -d 172.16.3.44 -i eth0 -j ACCEPT
181:-A INPUT -s 172.16.2.145 -d 172.16.3.41 -i eth0 -j ACCEPT
182:-A INPUT -s 172.16.2.145 -d 172.16.3.17 -i eth0 -j ACCEPT
183:-A INPUT -s 172.16.2.145 -d 172.16.2.8 -i eth0 -j ACCEPT
184:-A INPUT -s 172.16.2.145 -d 172.16.0.0/255.255.0.0 -i eth0 -j ACCEPT
185:-A INPUT -s 172.16.2.145 -d 172.16.2.149 -i eth0 -j ACCEPT
186:-A INPUT -s 172.16.2.145 -d 172.16.2.17 -i eth0 -j ACCEPT
187:-A INPUT -s 172.16.2.145 -d 172.16.2.3 -i eth0 -j ACCEPT
188:-A INPUT -s 172.16.2.145 -d 172.16.3.215 -i eth0 -j ACCEPT
189:-A INPUT -s 172.16.2.145 -d 172.16.2.145 -i eth0 -j ACCEPT
190:-A INPUT -s 172.16.2.145 -d 172.16.2.241 -i eth0 -j ACCEPT
191:-A INPUT -s 172.16.2.145 -d 172.16.2.49 -i eth0 -j ACCEPT
192:-A INPUT -s 172.16.2.145 -d 172.16.2.214 -i eth0 -j ACCEPT
193:-A INPUT -s 172.16.2.145 -d 172.16.2.222 -i eth0 -j ACCEPT
194:-A INPUT -s 172.16.2.241 -d 172.16.3.234 -i eth0 -j ACCEPT
195:-A INPUT -s 172.16.2.241 -d 172.16.2.2 -i eth0 -j ACCEPT
196:-A INPUT -s 172.16.2.241 -d 172.16.3.44 -i eth0 -j ACCEPT
197:-A INPUT -s 172.16.2.241 -d 172.16.3.41 -i eth0 -j ACCEPT
198:-A INPUT -s 172.16.2.241 -d 172.16.3.17 -i eth0 -j ACCEPT
199:-A INPUT -s 172.16.2.241 -d 172.16.2.8 -i eth0 -j ACCEPT
200:-A INPUT -s 172.16.2.241 -d 172.16.0.0/255.255.0.0 -i eth0 -j ACCEPT
201:-A INPUT -s 172.16.2.241 -d 172.16.2.149 -i eth0 -j ACCEPT
202:-A INPUT -s 172.16.2.241 -d 172.16.2.17 -i eth0 -j ACCEPT
203:-A INPUT -s 172.16.2.241 -d 172.16.2.3 -i eth0 -j ACCEPT
204:-A INPUT -s 172.16.2.241 -d 172.16.3.215 -i eth0 -j ACCEPT
205:-A INPUT -s 172.16.2.241 -d 172.16.2.145 -i eth0 -j ACCEPT
206:-A INPUT -s 172.16.2.241 -d 172.16.2.241 -i eth0 -j ACCEPT
207:-A INPUT -s 172.16.2.241 -d 172.16.2.49 -i eth0 -j ACCEPT
208:-A INPUT -s 172.16.2.241 -d 172.16.2.214 -i eth0 -j ACCEPT
209:-A INPUT -s 172.16.2.241 -d 172.16.2.222 -i eth0 -j ACCEPT
210:-A INPUT -s 172.16.2.49 -d 172.16.3.234 -i eth0 -j ACCEPT
211:-A INPUT -s 172.16.2.49 -d 172.16.2.2 -i eth0 -j ACCEPT
212:-A INPUT -s 172.16.2.49 -d 172.16.3.44 -i eth0 -j ACCEPT
213:-A INPUT -s 172.16.2.49 -d 172.16.3.41 -i eth0 -j ACCEPT
214:-A INPUT -s 172.16.2.49 -d 172.16.3.17 -i eth0 -j ACCEPT
215:-A INPUT -s 172.16.2.49 -d 172.16.2.8 -i eth0 -j ACCEPT
216:-A INPUT -s 172.16.2.49 -d 172.16.0.0/255.255.0.0 -i eth0 -j ACCEPT
217:-A INPUT -s 172.16.2.49 -d 172.16.2.149 -i eth0 -j ACCEPT
218:-A INPUT -s 172.16.2.49 -d 172.16.2.17 -i eth0 -j ACCEPT
219:-A INPUT -s 172.16.2.49 -d 172.16.2.3 -i eth0 -j ACCEPT
220:-A INPUT -s 172.16.2.49 -d 172.16.3.215 -i eth0 -j ACCEPT
221:-A INPUT -s 172.16.2.49 -d 172.16.2.145 -i eth0 -j ACCEPT
222:-A INPUT -s 172.16.2.49 -d 172.16.2.241 -i eth0 -j ACCEPT
223:-A INPUT -s 172.16.2.49 -d 172.16.2.49 -i eth0 -j ACCEPT
224:-A INPUT -s 172.16.2.49 -d 172.16.2.214 -i eth0 -j ACCEPT
225:-A INPUT -s 172.16.2.49 -d 172.16.2.222 -i eth0 -j ACCEPT
226:-A INPUT -s 172.16.2.214 -d 172.16.3.234 -i eth0 -j ACCEPT
227:-A INPUT -s 172.16.2.214 -d 172.16.2.2 -i eth0 -j ACCEPT
228:-A INPUT -s 172.16.2.214 -d 172.16.3.44 -i eth0 -j ACCEPT
229:-A INPUT -s 172.16.2.214 -d 172.16.3.41 -i eth0 -j ACCEPT
230:-A INPUT -s 172.16.2.214 -d 172.16.3.17 -i eth0 -j ACCEPT
231:-A INPUT -s 172.16.2.214 -d 172.16.2.8 -i eth0 -j ACCEPT
232:-A INPUT -s 172.16.2.214 -d 172.16.0.0/255.255.0.0 -i eth0 -j ACCEPT
233:-A INPUT -s 172.16.2.214 -d 172.16.2.149 -i eth0 -j ACCEPT
234:-A INPUT -s 172.16.2.214 -d 172.16.2.17 -i eth0 -j ACCEPT
235:-A INPUT -s 172.16.2.214 -d 172.16.2.3 -i eth0 -j ACCEPT
236:-A INPUT -s 172.16.2.214 -d 172.16.3.215 -i eth0 -j ACCEPT
237:-A INPUT -s 172.16.2.214 -d 172.16.2.145 -i eth0 -j ACCEPT
238:-A INPUT -s 172.16.2.214 -d 172.16.2.241 -i eth0 -j ACCEPT
239:-A INPUT -s 172.16.2.214 -d 172.16.2.49 -i eth0 -j ACCEPT
240:-A INPUT -s 172.16.2.214 -d 172.16.2.214 -i eth0 -j ACCEPT
```

```
241:-A INPUT -s 172.16.2.214 -d 172.16.2.222 -i eth0 -j ACCEPT
242:-A INPUT -s 172.16.2.222 -d 172.16.3.234 -i eth0 -j ACCEPT
243:-A INPUT -s 172.16.2.222 -d 172.16.2.2 -i eth0 -j ACCEPT
244:-A INPUT -s 172.16.2.222 -d 172.16.3.44 -i eth0 -j ACCEPT
245:-A INPUT -s 172.16.2.222 -d 172.16.3.41 -i eth0 -j ACCEPT
246:-A INPUT -s 172.16.2.222 -d 172.16.3.17 -i eth0 -j ACCEPT
247:-A INPUT -s 172.16.2.222 -d 172.16.2.8 -i eth0 -j ACCEPT
248:-A INPUT -s 172.16.2.222 -d 172.16.0.0/255.255.0.0 -i eth0 -j ACCEPT
249:-A INPUT -s 172.16.2.222 -d 172.16.2.149 -i eth0 -j ACCEPT
250:-A INPUT -s 172.16.2.222 -d 172.16.2.17 -i eth0 -j ACCEPT
251:-A INPUT -s 172.16.2.222 -d 172.16.2.3 -i eth0 -j ACCEPT
252:-A INPUT -s 172.16.2.222 -d 172.16.3.215 -i eth0 -j ACCEPT
253:-A INPUT -s 172.16.2.222 -d 172.16.2.145 -i eth0 -j ACCEPT
254:-A INPUT -s 172.16.2.222 -d 172.16.2.241 -i eth0 -j ACCEPT
255:-A INPUT -s 172.16.2.222 -d 172.16.2.49 -i eth0 -j ACCEPT
256:-A INPUT -s 172.16.2.222 -d 172.16.2.214 -i eth0 -j ACCEPT
257:-A INPUT -s 172.16.2.222 -d 172.16.2.222 -i eth0 -j ACCEPT
258:-A INPUT -p udp -m udp --sport 32769:65535 --dport 33434:33523 -j ACCEPT
259:-A INPUT -d 192.0.2.162 -j ACCEPT
260:-A INPUT -s 192.0.2.162 -j DROP
261:-A INPUT -s 172.16.2.16 -p udp -m udp --sport 53 -j ACCEPT
262:-A INPUT -s 172.16.2.17 -p udp -m udp --sport 53 -j ACCEPT
263:-A INPUT -s 172.16.2.3 -p udp -m udp --sport 53 -j ACCEPT
264:-A INPUT -d 192.0.2.130 -p tcp -m tcp --dport 5400:5401 -j DROP
265:-A INPUT -d 192.0.2.130 -p udp -m udp --dport 5400:5401 -j DROP
266:-A INPUT -d 192.0.2.130 -p tcp -m tcp --dport 12345:12346 -j DROP
267:-A INPUT -d 192.0.2.130 -p udp -m udp --dport 12345:12346 -j DROP
268:-A INPUT -p tcp -m tcp --dport 20 -j ACCEPT
269:-A INPUT -p udp -m udp --dport 20 -j ACCEPT
270:-A INPUT -p tcp -m tcp --dport 22 -j ACCEPT
271:-A INPUT -p udp -m udp --dport 22 -j ACCEPT
272:-A INPUT -p tcp -m tcp --dport 21 -j ACCEPT
273:-A INPUT -p udp -m udp --dport 21 -j ACCEPT
274:-A INPUT -p tcp -m tcp --dport 25 -j ACCEPT
275:-A INPUT -p udp -m udp --dport 25 -j ACCEPT
276:-A INPUT -p tcp -m tcp --dport 53 -j ACCEPT
277:-A INPUT -p udp -m udp --dport 53 -j ACCEPT
278:-A INPUT -p tcp -m tcp --dport 80 -j ACCEPT
279:-A INPUT -p udp -m udp --dport 80 -j ACCEPT
280:-A INPUT -p tcp -m tcp --dport 5000 -j ACCEPT
281:-A INPUT -p udp -m udp --dport 5000 -j ACCEPT
282:-A INPUT -p tcp -m tcp --dport 8080 -j ACCEPT
283:-A INPUT -p udp -m udp --dport 8080 -j ACCEPT
284:-A INPUT -d 192.0.2.130 -p tcp -m tcp --dport 80 -j ACCEPT
285:-A INPUT -d 192.0.2.130 -p udp -m udp --dport 80 -j ACCEPT
286:-A INPUT -d 192.0.2.130 -p tcp -m tcp --dport 25 -j ACCEPT
287:-A INPUT -d 192.0.2.130 -p udp -m udp --dport 25 -j ACCEPT
288:-A INPUT -d 192.0.2.130 -p tcp -m tcp --dport 53 -j ACCEPT
289:-A INPUT -d 192.0.2.130 -p udp -m udp --dport 53 -j ACCEPT
290:-A INPUT -d 192.0.2.130 -p tcp -m tcp --dport 22 -j ACCEPT
291:-A INPUT -d 192.0.2.130 -p udp -m udp --dport 22 -j ACCEPT
292:-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
293:-A INPUT -j DROP
294:-A INPUT -i tun0 -j ACCEPT
295:-A INPUT -j DROP
296:-A FORWARD -d 172.16.2.159 -p udp -m udp --dport 9010 -j ACCEPT
297:-A FORWARD -d 172.16.2.159 -p tcp -m tcp --dport 9010 -j ACCEPT
298:-A FORWARD -d 172.16.2.159 -p udp -m udp --dport 9005 -j ACCEPT
299:-A FORWARD -d 172.16.2.159 -p tcp -m tcp --dport 9005 -j ACCEPT
300:-A FORWARD -d 172.16.2.136 -p udp -m udp --dport 6000 -j ACCEPT
301:-A FORWARD -d 172.16.2.136 -p tcp -m tcp --dport 6000 -j ACCEPT
302:-A FORWARD -d 172.16.2.1 -p tcp -m tcp --dport 21 -j ACCEPT
303:-A FORWARD -d 172.16.2.8 -p tcp -m tcp --dport 80 -j ACCEPT
304:-A FORWARD -d 172.16.2.1 -p tcp -m tcp --dport 80 -j ACCEPT
305:-A FORWARD -d 172.16.2.6 -p tcp -m tcp --dport 3389 -j ACCEPT
306:-A FORWARD -d 172.16.3.48 -p tcp -m tcp --dport 3389 -j ACCEPT
307:-A FORWARD -d 172.16.3.41 -p udp -m udp --dport 6523 -j ACCEPT
308:-A FORWARD -d 172.16.3.41 -p tcp -m tcp --dport 6523 -j ACCEPT
309:-A FORWARD -d 172.16.3.41 -p udp -m udp --dport 6522 -j ACCEPT
310:-A FORWARD -d 172.16.3.41 -p tcp -m tcp --dport 6522 -j ACCEPT
311:-A FORWARD -d 172.16.3.41 -p udp -m udp --dport 6521 -j ACCEPT
312:-A FORWARD -d 172.16.3.41 -p tcp -m tcp --dport 6521 -j ACCEPT
313:-A FORWARD -d 172.16.3.17 -p udp -m udp --dport 1024 -j ACCEPT
314:-A FORWARD -d 172.16.3.17 -p tcp -m tcp --dport 1024 -j ACCEPT
315:-A FORWARD -d 172.16.3.17 -p udp -m udp --dport 1023 -j ACCEPT
```

```
316:-A FORWARD -d 172.16.3.17 -p tcp -m tcp --dport 1023 -j ACCEPT
317:-A FORWARD -d 172.16.3.17 -p tcp -m tcp --dport 80 -j ACCEPT
318:-A FORWARD -d 172.16.2.6 -p tcp -m tcp --dport 110 -j ACCEPT
319:-A FORWARD -d 172.16.2.6 -p tcp -m tcp --dport 80 -j ACCEPT
320:-A FORWARD -d 172.16.2.1 -p tcp -m tcp --dport 21 -j ACCEPT
321:-A FORWARD -d 172.16.2.3 -p tcp -m tcp --dport 8888 -j ACCEPT
322:-A FORWARD -d 172.16.2.3 -p tcp -m tcp --dport 3265 -j ACCEPT
323:-A FORWARD -d 172.16.3.234 -p tcp -m tcp --dport 80 -j ACCEPT
324:-A FORWARD -s 172.16.2.113 -p tcp -m tcp --dport 80 -j ACCEPT
325:-A FORWARD -s 172.16.2.159 -p tcp -m tcp --dport 80 -j ACCEPT
326:-A FORWARD -s 172.16.2.6 -p tcp -m tcp --dport 80 -j ACCEPT
327:-A FORWARD -s 172.16.3.41 -p tcp -m tcp --dport 80 -j ACCEPT
328:-A FORWARD -s 172.16.3.44 -p tcp -m tcp --dport 80 -j ACCEPT
329:-A FORWARD -s 172.16.2.8 -p tcp -m tcp --dport 80 -j ACCEPT
330:-A FORWARD -s 172.16.2.17 -p tcp -m tcp --dport 80 -j ACCEPT
331:-A FORWARD -s 172.16.2.149 -p tcp -m tcp --dport 80 -j ACCEPT
332:-A FORWARD -s 172.16.2.145 -p tcp -m tcp --dport 80 -j ACCEPT
333:-A FORWARD -s 172.16.2.49 -p tcp -m tcp --dport 80 -j ACCEPT
334:-A FORWARD -s 172.16.2.214 -p tcp -m tcp --dport 80 -j ACCEPT
335:-A FORWARD -s 172.16.2.222 -p tcp -m tcp --dport 80 -j ACCEPT
336:-A FORWARD -s 172.16.3.234 -p tcp -m tcp --dport 80 -j ACCEPT
337:-A FORWARD -s 172.16.3.234 -p tcp -m tcp --dport 80 -j DROP
338:-A FORWARD -s 172.16.2.2 -p tcp -m tcp --dport 80 -j DROP
339:-A FORWARD -s 172.16.3.44 -p tcp -m tcp --dport 80 -j DROP
340:-A FORWARD -s 172.16.3.41 -p tcp -m tcp --dport 80 -j DROP
341:-A FORWARD -s 172.16.3.17 -p tcp -m tcp --dport 80 -j DROP
342:-A FORWARD -s 172.16.2.8 -p tcp -m tcp --dport 80 -j DROP
343:-A FORWARD -s 172.16.0.0/255.255.0.0 -p tcp -m tcp --dport 80 -j DROP
344:-A FORWARD -s 172.16.2.149 -p tcp -m tcp --dport 80 -j DROP
345:-A FORWARD -s 172.16.2.17 -p tcp -m tcp --dport 80 -j DROP
346:-A FORWARD -s 172.16.2.3 -p tcp -m tcp --dport 80 -j DROP
347:-A FORWARD -s 172.16.3.215 -p tcp -m tcp --dport 80 -j DROP
348:-A FORWARD -s 172.16.2.145 -p tcp -m tcp --dport 80 -j DROP
349:-A FORWARD -s 172.16.2.241 -p tcp -m tcp --dport 80 -j DROP
350:-A FORWARD -s 172.16.2.49 -p tcp -m tcp --dport 80 -j DROP
351:-A FORWARD -s 172.16.2.214 -p tcp -m tcp --dport 80 -j DROP
352:-A FORWARD -s 172.16.2.222 -p tcp -m tcp --dport 80 -j DROP
353:-A FORWARD -d 192.0.2.162 -j ACCEPT
354:-A FORWARD -d 220.127.201.7 -p tcp -j DROP
355:-A FORWARD -d 220.127.201.7 -p udp -j DROP
356:-A FORWARD -d 200.146.148.161 -p tcp -j DROP
357:-A FORWARD -d 200.146.148.161 -p udp -j DROP
358:-A FORWARD -d 200.198.152.0/255.255.255.0 -p tcp -j DROP
359:-A FORWARD -d 200.198.152.0/255.255.255.0 -p udp -j DROP
360:-A FORWARD -d 172.16.2.247 -j ACCEPT
361:-A FORWARD -s 172.16.2.247 -j ACCEPT
362:-A FORWARD -o eth1 -p tcp -m tcp --dport 137 -j DROP
363:-A FORWARD -o eth1 -p udp -m udp --dport 137 -j DROP
364:-A FORWARD -o eth1 -p tcp -m tcp --dport 138 -j DROP
365:-A FORWARD -o eth1 -p udp -m udp --dport 138 -j DROP
366:-A FORWARD -o eth1 -p tcp -m tcp --dport 139 -j DROP
367:-A FORWARD -o eth1 -p udp -m udp --dport 139 -j DROP
368:-A FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT
369:-A FORWARD -j DROP
370:-A FORWARD -o tun0 -j ACCEPT
371:-A FORWARD -j DROP
372:-A OUTPUT -d 220.127.201.7 -p tcp -j DROP
373:-A OUTPUT -d 220.127.201.7 -p udp -j DROP
374:-A OUTPUT -d 200.146.148.161 -p tcp -j DROP
375:-A OUTPUT -d 200.146.148.161 -p udp -j DROP
376:-A OUTPUT -d 200.198.152.0/255.255.255.0 -p tcp -j DROP
377:-A OUTPUT -d 200.198.152.0/255.255.255.0 -p udp -j DROP
378:-A OUTPUT -o eth1 -p tcp -m tcp --dport 137 -j DROP
379:-A OUTPUT -o eth1 -p udp -m udp --dport 137 -j DROP
380:-A OUTPUT -o eth1 -p tcp -m tcp --dport 138 -j DROP
381:-A OUTPUT -o eth1 -p udp -m udp --dport 138 -j DROP
382:-A OUTPUT -o eth1 -p tcp -m tcp --dport 139 -j DROP
383:-A OUTPUT -o eth1 -p udp -m udp --dport 139 -j DROP
384:-A OUTPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
385:-A OUTPUT -j DROP
386:-A OUTPUT -o tun0 -j ACCEPT
387:-A OUTPUT -j ACCEPT
```

A tabela 5.11 apresenta o novo conjunto de regras com as incoerências removidas. Somente incoerências reconhecidas como erro foram removidas. Relações que resultam em redundância parcial e Normal/Aviso não foram verificadas.

Tabela 5.11: Conjunto de regras resultante após análise segundo o modelo proposto

```
1:-A INPUT -s 172.16.0.0/255.255.0.0 -d 172.16.0.0/255.255.0.0 -i eth0 -j ACCEPT
2:-A INPUT -p udp -m udp --sport 32769:65535 --dport 33434:33523 -j ACCEPT
3:-A INPUT -d 192.0.2.162 -j ACCEPT
4:-A INPUT -s 192.0.2.162 -j DROP
5:-A INPUT -s 172.16.2.16 -p udp -m udp --sport 53 -j ACCEPT
6:-A INPUT -s 172.16.2.17 -p udp -m udp --sport 53 -j ACCEPT
7:-A INPUT -s 172.16.2.3 -p udp -m udp --sport 53 -j ACCEPT
8:-A INPUT -p tcp -m tcp --dport 20 -j ACCEPT
9:-A INPUT -p udp -m udp --dport 20 -j ACCEPT
10:-A INPUT -p tcp -m tcp --dport 22 -j ACCEPT
11:-A INPUT -p udp -m udp --dport 22 -j ACCEPT
12:-A INPUT -p tcp -m tcp --dport 21 -j ACCEPT
13:-A INPUT -p udp -m udp --dport 21 -j ACCEPT
14:-A INPUT -p tcp -m tcp --dport 25 -j ACCEPT
15:-A INPUT -p udp -m udp --dport 25 -j ACCEPT
16:-A INPUT -p tcp -m tcp --dport 53 -j ACCEPT
17:-A INPUT -p udp -m udp --dport 53 -j ACCEPT
18:-A INPUT -p tcp -m tcp --dport 80 -j ACCEPT
19:-A INPUT -p udp -m udp --dport 80 -j ACCEPT
20:-A INPUT -p tcp -m tcp --dport 5000 -j ACCEPT
21:-A INPUT -p udp -m udp --dport 5000 -j ACCEPT
22:-A INPUT -p tcp -m tcp --dport 8080 -j ACCEPT
23:-A INPUT -p udp -m udp --dport 8080 -j ACCEPT
24:-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
25:-A INPUT -j DROP
26:-A FORWARD -d 172.16.2.159 -p udp -m udp --dport 9010 -j ACCEPT
27:-A FORWARD -d 172.16.2.159 -p tcp -m tcp --dport 9010 -j ACCEPT
28:-A FORWARD -d 172.16.2.159 -p udp -m udp --dport 9005 -j ACCEPT
29:-A FORWARD -d 172.16.2.159 -p tcp -m tcp --dport 9005 -j ACCEPT
30:-A FORWARD -d 172.16.2.136 -p udp -m udp --dport 6000 -j ACCEPT
31:-A FORWARD -d 172.16.2.136 -p tcp -m tcp --dport 6000 -j ACCEPT
32:-A FORWARD -d 172.16.2.8 -p tcp -m tcp --dport 80 -j ACCEPT
33:-A FORWARD -d 172.16.2.1 -p tcp -m tcp --dport 80 -j ACCEPT
34:-A FORWARD -d 172.16.2.6 -p tcp -m tcp --dport 3389 -j ACCEPT
35:-A FORWARD -d 172.16.3.48 -p tcp -m tcp --dport 3389 -j ACCEPT
36:-A FORWARD -d 172.16.3.41 -p udp -m udp --dport 6523 -j ACCEPT
37:-A FORWARD -d 172.16.3.41 -p tcp -m tcp --dport 6523 -j ACCEPT
38:-A FORWARD -d 172.16.3.41 -p udp -m udp --dport 6522 -j ACCEPT
39:-A FORWARD -d 172.16.3.41 -p tcp -m tcp --dport 6522 -j ACCEPT
40:-A FORWARD -d 172.16.3.41 -p udp -m udp --dport 6521 -j ACCEPT
41:-A FORWARD -d 172.16.3.41 -p tcp -m tcp --dport 6521 -j ACCEPT
42:-A FORWARD -d 172.16.3.17 -p udp -m udp --dport 1024 -j ACCEPT
43:-A FORWARD -d 172.16.3.17 -p tcp -m tcp --dport 1024 -j ACCEPT
44:-A FORWARD -d 172.16.3.17 -p udp -m udp --dport 1023 -j ACCEPT
45:-A FORWARD -d 172.16.3.17 -p tcp -m tcp --dport 1023 -j ACCEPT
46:-A FORWARD -d 172.16.3.17 -p tcp -m tcp --dport 80 -j ACCEPT
47:-A FORWARD -d 172.16.2.6 -p tcp -m tcp --dport 110 -j ACCEPT
48:-A FORWARD -d 172.16.2.6 -p tcp -m tcp --dport 80 -j ACCEPT
49:-A FORWARD -d 172.16.2.1 -p tcp -m tcp --dport 21 -j ACCEPT
50:-A FORWARD -d 172.16.2.3 -p tcp -m tcp --dport 8888 -j ACCEPT
51:-A FORWARD -d 172.16.2.3 -p tcp -m tcp --dport 3265 -j ACCEPT
52:-A FORWARD -d 172.16.3.234 -p tcp -m tcp --dport 80 -j ACCEPT
53:-A FORWARD -s 172.16.2.113 -p tcp -m tcp --dport 80 -j ACCEPT
54:-A FORWARD -s 172.16.2.159 -p tcp -m tcp --dport 80 -j ACCEPT
55:-A FORWARD -s 172.16.2.6 -p tcp -m tcp --dport 80 -j ACCEPT
56:-A FORWARD -s 172.16.2.8 -p tcp -m tcp --dport 80 -j ACCEPT
57:-A FORWARD -s 172.16.2.17 -p tcp -m tcp --dport 80 -j ACCEPT
58:-A FORWARD -s 172.16.2.149 -p tcp -m tcp --dport 80 -j ACCEPT
59:-A FORWARD -s 172.16.2.145 -p tcp -m tcp --dport 80 -j ACCEPT
60:-A FORWARD -s 172.16.2.49 -p tcp -m tcp --dport 80 -j ACCEPT
61:-A FORWARD -s 172.16.2.214 -p tcp -m tcp --dport 80 -j ACCEPT
62:-A FORWARD -s 172.16.2.222 -p tcp -m tcp --dport 80 -j ACCEPT
63:-A FORWARD -s 172.16.3.234 -p tcp -m tcp --dport 80 -j ACCEPT
64:-A FORWARD -s 172.16.0.0/255.255.0.0 -p tcp -m tcp --dport 80 -j DROP
65:-A FORWARD -d 192.0.2.162 -j ACCEPT
66:-A FORWARD -d 220.127.201.7 -p tcp -j DROP
```

```
67:-A FORWARD -d 220.127.201.7 -p udp -j DROP
68:-A FORWARD -d 200.146.148.161 -p tcp -j DROP
69:-A FORWARD -d 200.146.148.161 -p udp -j DROP
70:-A FORWARD -d 200.198.152.0/255.255.255.0 -p tcp -j DROP
71:-A FORWARD -d 200.198.152.0/255.255.255.0 -p udp -j DROP
72:-A FORWARD -d 172.16.2.247 -j ACCEPT
73:-A FORWARD -s 172.16.2.247 -j ACCEPT
74:-A FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT
75:-A FORWARD -j DROP
76:-A OUTPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
77:-A OUTPUT -j DROP
```

5.3.3.1 Estatísticas sobre análise de regras do estudo de caso 3

As seguintes estatísticas podem ser observadas após a análise do conjunto de regras do estudo de caso 3:

- número de regras antes da verificação: 387
- número de regras após verificação: 77

Com relação as incoerências:

- removidas por *Redundância Desnecessária de RA:* **108**
- removidas por *Redundância Desnecessária de RB:* **186**
- removidas por *Redundância Desnecessária:* **2**
- removidas por *RB nunca ser ativado*: **3**
- removidas por existir *ações diferentes para mesma transmissão*: **11**

6 CONCLUSÕES E TRABALHOS FUTUROS

A metodologia apresentada por este trabalho visa à identificação de incoerências entre regras de filtros de pacotes, obtendo-se, assim, uma consistência das regras entre si e, com isso, um conjunto compacto onde cada regra reflete parte da política implementa pelo filtro, estando este livre de falhas de configuração que possam mascarar a real utilização e finalidade de alguma regra. A utilização da metodologia aplica-se tanto a legados de regras, como a novas configurações.

A tabela de caracterização de falhas apresentada no capítulo 4, o qual apresenta o modelo proposto, é parte importante deste trabalho visto que a análise apresentada via esta tabela leva em consideração os aspectos importantes para caraterização da relação, bem como o diagnóstico que estes aspectos combinados geram quando comparado um par de regras. A tabela 6.1, reapresenta esta análise de caracterização de falhas.

Tabela 6.1: Caracterização da Falha

Relação	Grau	Identidade da Ação	Ordenação Parcial	Diagnóstico
Igualdade	-	Sim	-	ERRO, Redundância Desnecessária
Igualdade	-	Não	-	ERRO, Ações Diferentes para a mesma transmissão
Subconjunto	RA ⊆ RB	Sim	RA antes de RB	ERRO, Redundância Desnecessária de RA
Subconjunto	RA ⊆ RB	Não	RA antes de RB	Normal/Aviso
Superconjunto	RB ⊆ RA	Sim	RA antes de RB	ERRO, Redundância Desnecessária de RB
Superconjunto	RB ⊆ RA	Não	RA antes de RB	ERRO, RB nunca será ativado
Intersecção	-	Sim	-	Normal com Redundância Parcial
Intersecção	-	Não	-	Normal/Aviso
Relação	-	-	-	Normal

Parcial				
Sem Relação	-	-	-	Normal

Parte das definições deste trabalho seguem características de outros modelos já publicados na literatura, e que fazem parte do estado da arte na área de verificação de filtros. Entretanto, a metodologia apresentada demonstra que é necessário, para tornar a verificação efetiva, levar em consideração não somente a comparação regra a regra, como a literatura sugere, mas também outras relações que uma ou outra regra tenham participado a priori, pois influencia a caracterização da relação e, com isso, o diagnóstico que será apresentado ao administrador. O fato de redefinir fatores que envolvam relações que antecedem uma nova detecção e caracterização de relação, proporciona que se tenha um diagnóstico mais preciso, como o apresentado no estudo de caso 2 do capítulo 5.

As tabelas de regras a seguir demonstram que, com as variações nas ações de algumas regras, os diagnósticos sofrem alterações. Entretanto, o fato de as 3 regras mostradas estarem interrelacionadas entre si torna o diagnóstico mostrado muito genérico e não muito preciso. Assim, existe a necessidade de avaliação da cadeia de regras relacionada como um todo, pois todas possuem relação entre si, e idealmente um diagnóstico deveria englobar as 3 regras, e não somente 3 vezes pares de regras. Isto demonstra a possibilidade de expandir-se a ferramenta para tratar destes casos de forma mais precisa.

```
# TABELA DE REGRAS ####################################################
1:-A INPUT -s 10.0.20.1 -p tcp -m tcp --dport 80 -j ACCEPT
2:-A INPUT -s 10.0.20.0/255.255.255.0 -p tcp -m tcp --dport 80 -j DROP
3:-A INPUT -s 10.0.0.0/255.0.0.0 -d 10.1.1.1 -p tcp -m tcp --dport 80 -j ACCEPT

# VERIFICAÇÕES ############################## tabela FILTER->INPUT
1x2: NORMAL/AVISO!
1x3: Normal com Redundância Parcial
2x3: NORMAL/AVISO!
```

```
# TABELA DE REGRAS ####################################################
1:-A INPUT -s 10.0.20.1 -p tcp -m tcp --dport 80 -j DROP
2:-A INPUT -s 10.0.20.0/255.255.255.0 -p tcp -m tcp --dport 80 -j DROP
3:-A INPUT -s 10.0.0.0/255.0.0.0 -d 10.1.1.1 -p tcp -m tcp --dport 80 -j DROP

# VERIFICAÇÕES ############################## tabela FILTER->INPUT
1x2: ERRO, Redundância Desnecessária de RA
1x3: Normal com Redundância Parcial
2x3: Normal com Redundância Parcial
```

```
# TABELA DE REGRAS ###################################################
1:-A INPUT -s 10.0.20.1 -p tcp -m tcp --dport 80 -j ACCEPT
2:-A INPUT -s 10.0.20.0/255.255.255.0 -p tcp -m tcp --dport 80 -j ACCEPT
3:-A INPUT -s 10.0.0.0/255.0.0.0 -d 10.1.1.1 -p tcp -m tcp --dport 80 -j DROP

# VERIFICAÇÕES ############################## tabela FILTER->INPUT
1x2: ERRO, Redundância Desnecessária de RA
1x3: NORMAL/AVISO!
2x3: NORMAL/AVISO!
```

Este trabalho foi orientado ao *iptables*, e um acréscimo que a metodologia apresentada oferece envolve a verificação de utilização de *chains* adicionais, bem como de regras que possuam desvios para *chains* adicionais e que não tenham relação com estas regras.

Com relação a medidas de performance para a o processo de verificação, estas não foram consideradas por ser a verificação um processo isolado, que deve ser realizado preferencialmente em um equipamento secundário e antes das alterações implementadas ao conjunto de regras. Independente do tempo utilizado para a verificação, este processo em nada influencia o ambiente de produção.

Como trabalhos futuros, pensa-se em expandir o modelo a fim de considerar toda uma cadeia de regras que possua relação e, com isso, refinar ainda mais a metodologia para que se possa obter um diagnóstico ainda mais efetivo. Esta característica deve ainda reduzir o número de mensagens de incoerências, sendo mais pontual em situações em que a ocorrência de um erro acarreta novos avisos de incoerências sem, no entanto, apontar que a remoção da primeira influencia nas seguintes.

É freqüente a existência de vários dispositivos de filtragem em uma rede e, além de considerar que existam incoerências entre as regras de cada um dos equipamentos, considerar que possam existir incoerências entre as regras destes equipamentos, faz parte dos avanços que, nos trabalhos futuros, pretende-se adicionar ao modelo. Para a identificação deste novo tipo de incoerência, pretende-se acrescentar novos aspectos às verificações, como informações sobre a topologia em que estes filtros estão inseridos, visto que um pacote pode atravessar uma cadeia de filtros, bem como alterações por intermédio de NAT.

Para a segurança da informação, é de fundamental importância que se verifique a coerência entre as regas na configuração de filtros, pois este tipo de mecanismo tem função relevante para garantir controles de acesso a serviços exclusivos, assim como o bloqueio de ameaças e vulnerabilidades, as quais são cada vez mais constantes devido à expansão de conectividade que se tem nos dias de hoje. Desta forma, a metodologia proposta vem ao encontro da necessidade de verificação apresentada, uma vez que pode

ser automatizada e, desta forma, auxiliar efetivamente ao administrador para a correta configuração e análise de regras de filtros de pacotes.

REFERÊNCIAS

AVIZIENIS, A. The n-version approach to fault tolerant software. **IEEE Transactions on Software Engineering,** Los Alamitos, v. SE11, n.12, p. 1491–1501, Dec. 1985.

AVIZIENIS, A. The methodology of n-version programming. In: LYU, M.R.(Ed.). **Software Fault Tolerance.** New York: Wiley, 1995. p. 23–46.

AVIZIENIS, A.; CHEN, L. On the implementation of nversion programming for software fault-tolerance during program execution. In: ANNUAL INTERNATIONAL COMPUTER SOFTWARE AND APPLICATIONS CONFERENCE, 1977. **Proceedings...** Los Alamitos: IEEE Computer Society Press, 1977. p. 145–155.

AL-SHAER, E.; HAMED, H. Firewall Policy Advisor for Anomaly Detection and Rule Editing. IFIP/IEEE INTERNATIONAL SYMPOSIUM ON INTEGRATED NETWORK MANAGEMENT, IM, 8., 2003, Colorado. **Integrated network management VIII:** managing it all. Boston: Kluwer Academic, 2003.

AL-SHAER, E.; HAMED, H. Discovery of Policy Anomalies in Distributed Firewalls. In: IEEE CONFERENCE ON COMPUTER COMMUNICATIONS, INFOCOMM, 2004, Chicago, USA. **Proceedings...** Los Alamitos: IEEE Computer Society Press, 2004. v. 4, p. 145–155.

CHESWICK, W. R.; BELLOVIN, S. M.; RUBIN, A. D. **Firewalls e Segurança na Internet:** repelindo o hacker ardiloso. 2.ed. Porto Alegre: Bookman, 2005.

IANA. **Special-Use IPv4 Addresses:** RFC 3330. [S.l.]: Internet Engineering Task Force, Network Working Group, 2002.

LIU, A. X.; GOUDA, M. G. Firewall design: consistency, completeness and compactness. In: INTERNATIONAL CONFERENCE ON DISTRIBUTED COMPUTING SYSTEMS, ICDCS, 2004, Tokyo, Japan. **Proceedings...** Los Alamitos: IEEE Computer Society, 2004. v. 1.

LIU, A. X.; GOUDA, M. G. Diverse Firewall Design. In: INTERNATIONAL CONFERENCE ON DEPENDABLE SYSTEMS AND NETWORKS, DSN, 2004, Florence, Italy. **Proceedings...** Los Alamitos: IEEE Computer Society, 2004. v. 1.

MAYER A. W.; ZISKIND, E. Fang: A firewall analysis engine. In: IEEE SYMPOSIUM ON SECURITY AND PRIVACY, 2000, Oakland, California, USA. **Proceedings...** Los Alamitos: IEEE Computer Society, 2000. v. 1.

NORTHCUTT, S. et al. **Inside Network Perimeter Security**: The Definitive Guide to Firewalls, Virtual Private Networks (VPNs), Routers, and Intrusion Detection Systems. Indianápolis: Pearson Education, 2002.

WANG, D.; HAO, R.; LEE, D. Fault detection in Rule-based Software systems. **Information & Software Technology,** Amsterdam, v. SE45, n. 12, p. 865-871, Oct. 2003.

WACK, J.; CUTLER, K.; POLE, J. **Guidelines on Firewalls and Firewall Policy.** [S.l.]; National Institute of Standards and Technology, Jan. 2002.

WOOL, A. et al. Firmato: A novel firewall management toolkit. In: IEEE SYMPOSIUM ON SECURITY AND PRIVACY, 1999, Oakland, Califórnia, USA. **Proceedings...** Los Alamitos: IEEE Computer Society, 1999. v. 1.