

**UNIVERSIDADE FEDERAL DO RIO DE JANEIRO  
NUCLEO DE COMPUTAÇÃO ELETRÔNICA  
PÓS-GRADUAÇÃO EM GERÊNCIA DE SEGURANÇA DA  
INFORMAÇÃO (MSI)**

**Felipe Martins Rôlla**

**DESENVOLVENDO FIREWALLS SEGUROS EM AMBIENTES DE  
ALTA DISPONIBILIDADE UTILIZANDO SOFTWARE LIVRE**

Rio de Janeiro

2009

**Felipe Martins Rôlla**

**DESENVOLVENDO FIREWALLS SEGUROS EM AMBIENTES DE  
ALTA DISPONIBILIDADE UTILIZANDO SOFTWARE LIVRE**

Monografia apresentada como requisito parcial para a conclusão do curso de especialização em Gerência de Segurança da Informação do NCE-UFRJ.

Orientador: Alexandre Freire

Rio de Janeiro

2009

Agradeço aos professores do Curso de Segurança da Informação pelo incentivo constante no decorrer de todo esse tempo de estudo.

Agradeço também a todos os meus familiares que me apoiaram e me entenderam nos momentos de maior dificuldade.

Dedico esta obra aos meus pais, primeiramente, pelo apoio e a minha esposa pela compreensão em todas as etapas deste trabalho. E, em especial, ao meu orientador, professor Alexandre Freire, por me abrir o campo de visão para novos horizontes.

Se o conhecimento pode criar problemas,  
não é através da ignorância que podemos  
solucioná-los.

Isaac Asimov

# SUMÁRIO

<b>1 INTRODUÇÃO</b> .....	9
1.1 MOTIVAÇÃO .....	11
1.2 OBJETIVOS .....	13
<b>2 REFERENCIAL TEÓRICO</b> .....	10
<b>2.1 AMBIENTES CONHECIDOS</b> .....	11
2.1.1 Conceituação de Firewall .....	13
2.1.2 Pontos de Falha Conhecidos .....	18
<b>2.2 REDUNDÂNCIA DE FIREWALL</b> .....	19
2.2.1 Firewall Redundante .....	13
2.2.2 Mudança de Estrutura .....	16
2.2.3 Problemas Restantes .....	18
<b>2.3 REDUNDANCIA FÍSICA</b> .....	19
2.3.1 Placas Ethernet Redundantes .....	13
2.3.2 Duplicação de Switches .....	16
<b>2.4 ALTA DISPONIBILIDADE</b> .....	19
2.4.1 Conceituação .....	13
2.4.2 Software Bonding .....	16
2.4.3 Software Bonding .....	16
2.4.4 Software Heartbeat .....	16
<b>2.5 SCRIPT DE FIREWALL</b> .....	16
2.5.1 Conceituação .....	16
2.5.2 Bash Scripting .....	16
<b>2.6 REPLICAÇÃO</b> .....	11
2.6.1 Conceituação .....	13
2.6.2 Tipos de Replicação .....	16
2.6.3 Script de Replicação .....	16
<b>2.7 SISTEMA DE PREVENÇÃO DE INTRUSOS</b> .....	11
2.7.1 Conceituação .....	13
2.7.2 Tipos de Softwares .....	16
2.7.3 AIDE .....	18
2.7.3.1 Configurações .....	18
<b>3 CONCLUSÃO</b> .....	23
<b>REFERÊNCIAS</b> .....	24

## LISTA DE FIGURAS

FIGURA 1 – TOPOLOGIA DE FIREWALL PADRÃO.....	3
FIGURA 2 – TOPOLOGIA INTERNET-FIREWALL-LAN UTILIZADA .....	5
FIGURA 3 – TOPOLOGIA FÍSICA SIMPLES.....	6
FIGURA 4 – TOPOLOGIA ALTERADA COM DOIS NÓS DE FIREWALL .....	7
FIGURA 5 – TOPOLOGIA DE FIREWALL HA COM DNS .....	14
FIGURA 6 – TOPOLOGIA DE FIREWALL HA COM DNS COM IP VIP.....	14

## LISTA DE TABELAS

TABELA 1 – CONFIGURAÇÃO DE SWITCHES DUPLICADA .....	9
TABELA 2 – QUADRO DE INTERFACES EM BONDING .....	12
TABELA 3 – TÍTULO DA TABELA 3 .....	16
TABELA 4 – TÍTULO DA TABELA 4 .....	18

## LISTA DE ANEXOS

ANEXO 1 – LEVANTAMENTO LÓGICO DE SWITCHES .....	11
ANEXO 2 – ARQUIVO ALIASES-BOND .....	13
ANEXO 3 – ARQUIVO /ETC/NETWORK/INTERFACES .....	16
ANEXO 4 – AUTHKEYS .....	16
ANEXO 5 – HA.CF .....	18
ANEXO 6 – HARESOURCES .....	18
ANEXO 7 – SCRIPT DE FIREWALL HA .....	18
ANEXO 8 – SCRIPT DE SINCRONISMO SYNC_FIREWALL.SH .....	18

## RESUMO

A preocupação com segurança e disponibilidade é requisito essencial para a grande maioria dos serviços e aplicações que utilizam redes de computadores em geral, principalmente quando conectadas a Internet. Ao contrário do que acontecia no início da Internet, atualmente, as organizações que utilizam a Internet procuram proteger seus ativos através da utilização de dispositivos tecnológicos que impeçam ou dificultem o comprometimento de suas informações sigilosas. Contudo não há como garantir que determinada organização seja vítima de um incidente de segurança ou incidente físico que cause indisponibilidade dos serviços prestados.

As Organizações reconhecem a necessidade da disponibilidade de serviço, para tanto influenciam tanto a parte lógica como física.

Este trabalho tem por objetivo a análise de um caso de sucesso na implementação de Firewalls de borda seguros em ambientes de alta disponibilidade utilizando softwares livres, sua criação, manutenção, escalabilidade, apresentando as boas práticas para seu desenvolvimento e melhoria contínua.

A metodologia utilizada foi o estudo de caso, utilizando como amostra topologias, equipamentos utilizados, softwares, arquivos de configuração, entre outros itens pertinentes.

Os resultados evidenciam que a criação de Firewalls Seguros em Ambientes de Alta Disponibilidade pode ser feita utilizando-se softwares livres, de baixo custo de implementação e de fácil manutenção.

Palavras-chave: Firewall; Seguro; Software Livre; Física; Disponibilidade.

# 1 INTRODUÇÃO

## 1.1 MOTIVAÇÃO

Manter a segurança de sistemas de firewall de uma Organização no ambiente computacional interconectado nos dias atuais é um grande desafio, que se torna mais difícil, à medida que são lançados novos produtos para a Internet, e novas ferramentas de ataque são desenvolvidas. Manter a disponibilidade de serviços no nível 99.9% ideal é praticamente impossível atualmente principalmente também devido ao alto preço de hardware envolvido.

A maioria das Organizações reconhece a existência de diversas soluções de ambientes de firewall, porém dificilmente estão alinhadas com o montante capital disponível para a implementação do ambiente ideal.

Uma das estratégias na redução de custos é a implementação de soluções análogas às comerciais utilizando Software Livre, de fácil aprendizado e livre de royalties com licenciamento comercial. Muitas soluções, mundialmente reconhecidas por sua segurança como a Check Point, utilizam em suas aplicações Softwares Livres, provando seu alto grau de amadurecimento, porém os prendem a licenças comerciais, terminando por elevar o preço desse tipo de solução.

## 1.2 OBJETIVOS

À medida que as organizações começam a implementar soluções de firewall, elas tentam determinar qual a melhor solução para o desenvolvimento de sua estrutura.

As Organizações buscam sempre a melhor solução ao menor custo imediato ou de médio prazo. Este custo deve refletir diretamente na implementação, manutenção, maturidade e escalabilidade da solução de firewall em alta disponibilidade implementada.

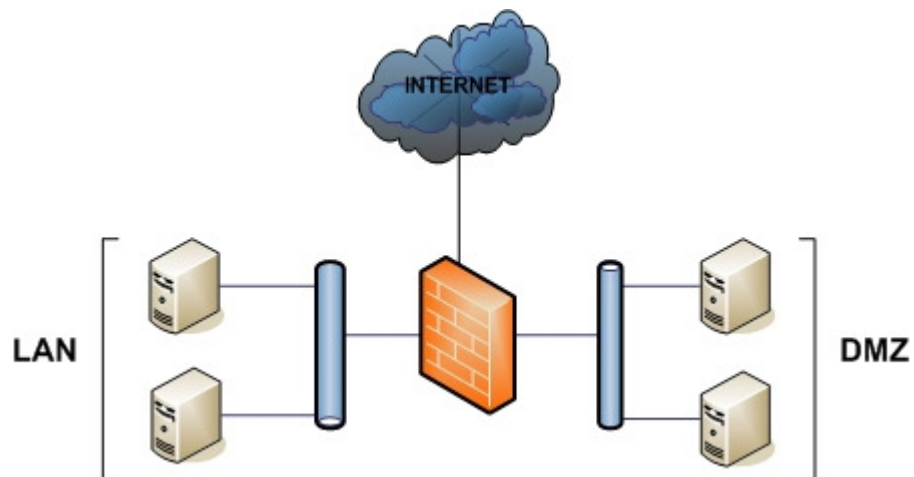
Este trabalho visa responder as seguintes perguntas: Quais soluções de baixo custo de desenvolvimento e manutenção de firewalls seguros utilizando ambientes de alta disponibilidade? Quais são as boas práticas na criação desta solução? Quais são os custos agregados envolvidos nesta solução? Quais são os tipos de serviços providos pelas soluções de softwares escolhidas? Como implementar a solução ?

## 2 REFERENCIAL TEÓRICO

### 2.1 AMBIENTES CONHECIDOS

De forma geral a topologia padrão de *firewall* é composta por 3 a 4 grupos de *hosts*; *Links* Internet, Servidores em uma *DMZ (Demiliarized Zone)*, máquinas em uma rede local e o *firewall* propriamente dito.

A topologia é desenvolvida de forma a proteger todos os nós ligados diretamente ao *firewall* que os interliga, como mostra a **Figura 1**.



**Figura 1: Topologia Padrão de Firewall WAN, LAN, DMZ**

Na topologia apresentada na **Figura 1** é evidente a falta de disponibilidade física, bem como a segurança lógica entre os nós teoricamente protegidos.

### 2.1.1 CONCEITUAÇÃO DE FIREWALL

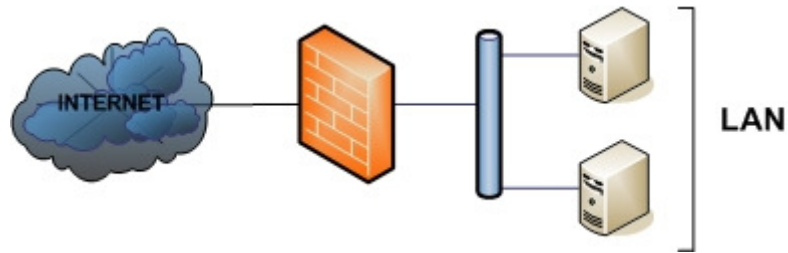
Um servidor de *firewall*, também conhecido como firewall de rede, é uma barreira lógica desenvolvida para prevenir comunicação não autorizada entre sessões de uma rede de computador, agindo nas camadas 3 (Rede) e 4 (Transporte) do modelo conceitual OSI, podendo também, através de implementações *L7*, agir nas camadas 5 (Sessão), 6 (Apresentação) e 7 (Aplicação).

Servidores de *Firewall* Linux utilizam basicamente dois pacotes em seu desenvolvimento, ***netfilter*** e ***iproute***. O pacote ***netfilter*** constitui um framework para filtragem de pacotes *TCP/IP* já incluído por padrão em *kernels vanilla* 2.4 e 2.6, o pacote ***iproute*** contém aplicações que permitem a configuração de roteamento avançado e *Traffic Shaping* (Modelagem de Tráfego) em conexões.

Devido a não utilização de soluções de roteamento avançado, pois não fazem parte do escopo deste documento, o pacote ***iproute*** não será utilizado nesse caso de uso.

### 2.1.2 PONTOS DE FALHA CONHECIDOS

Devido à abordagem de firewall escolhida, pelo caso de uso proteger apenas uma rede, caso mais simples e largamente utilizado em *Data Centers* de *hosting* dedicado, foi adotada uma topologia simples, composta de **INTERNET-FIREWALL-LAN**, conforme visualizado na **Figura 2**, onde a LAN atua como DMZ para servidores de aplicações de rede, entre outros.



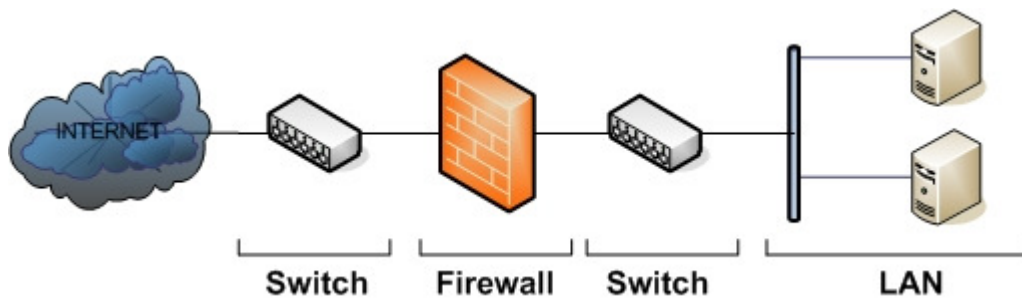
**Figura 2: Topologia Padrão Internet-Firewall-LAN utilizada**

A topologia lógica da **Figura 2**, embora largamente utilizada no meio corporativo, principalmente no mercado de *hosting*, encontra em seu ponto forte, economia de capital no número de dispositivos utilizados, também seu maior ponto de falha facilmente levando a indisponibilidade de serviço em caso de erros físicos em sua estrutura.

A falta de interfaces de rede extras para redundância é um agravante bem como a queima de hardware ou mau funcionamento das interfaces e cabeamento confeccionado fora de padrões de conformidade aceitáveis a estrutura. Uma vez utilizado apenas um caminho físico (interface de rede), no caso de falha em uma interface todo um grupo de hosts se torna indisponível, problema esse que se torna danoso a estrutura quando serviços de missão crítica são fornecidos em período 24x7 na LAN ou DMZ e principalmente se a parte que utiliza (cliente) está protegida contratualmente por termos de SLA, o que dependendo do tempo de indisponibilidade transforma-se em prejuízo capital e de imagem à empresa prestadora do serviço.

A falta de interfaces extras ou somente a utilização de interfaces *Fast Ethernet* e *Gigabit*, limitam fisicamente a largura de banda (*throughput*) da solução tirando-lhe a escalabilidade e flexibilidade quando da necessidade no aumento de carga na estrutura.

Outro ponto principal de falha é a existência de apenas um firewall físico, que independente de falhas de interface, no caso de um eventual problema geral de hardware ou software, pode levar a indisponibilidade total do sistema. Para melhor visualização, a **Figura 3** exibe a topologia física da solução padrão.



**Figura 3: Topologia Física Simples**

## 2.2 REDUNDÂNCIA DE FIREWALL

### 2.2.1 FIREWALL REDUNDANTE

Devido ao ponto de falha física de hardware, foi adicionado à estrutura outro servidor que agirá como redundância física e lógica funcionando em modo *Fail Over* (Ativo-Passivo) onde o firewall **Master** (nó principal), denominado **FWHA01**, funcionará sempre ativo recebendo requisições da LAN e WAN, e o firewall **SLAVE** (nó secundário) denominado **FWHA02**, ficará em *stand-by* assumindo o papel de master no caso de falha do nó de firewall principal.

A adição de um novo firewall à estrutura, evita a instalação de placas sobressalentes no firewall para redundância uma vez que o ponto de falha básico, o firewall, permanecerá ainda o único nó de interligação entre os grupos LAN e INTERNET.

## 2.2.2 MUDANÇAS NA ESTRUTURA

Devido à inclusão de um novo nó de firewall, devem ser executadas mudanças físicas na estrutura, conforme mostra a **Figura 4**. Dessa forma evitamos problemas de indisponibilidade relativa à queda de um dos nós de firewall, de forma a manter os serviços sempre funcionais sem executar mudanças drásticas e aproveitando os ativos de rede existentes.

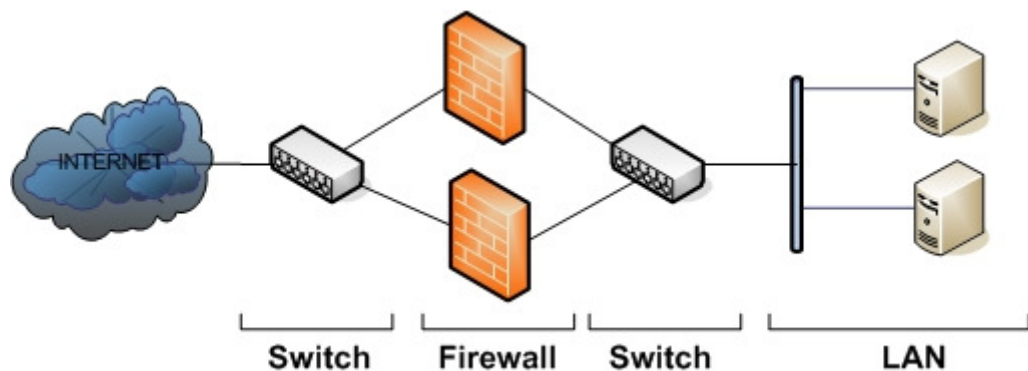


Figura 4: Topologia alterada com dois nós de firewall

## 2.2.3 PROBLEMAS RESTANTES

A adição de outro nó de firewall não elimina um ponto de falha único representado pelos dois *Switches Layer 2* na estrutura, pois independente de queda nos nós de firewall, problemas nos switches ocasionarão indisponibilidade no conjunto de máquinas protegido, em nosso caso a LAN.

Com o novo firewall também surgem problemas relativos à configuração de IP das interfaces de rede WAN e LAN, que devem ser diferentes evitando conflito de IP

(Modelo OSI camada 3) na rede. A configuração de um IP em cada interface gera esforço de administração manual do suporte interno da empresa prestadora de serviço devido a suas configurações de gateway, sendo esses diferentes, caso não sejam automáticos.

## 2.3 REDUNDANCIA FÍSICA

### 2.3.1 PLACAS ETHERNET REDUNDANTES

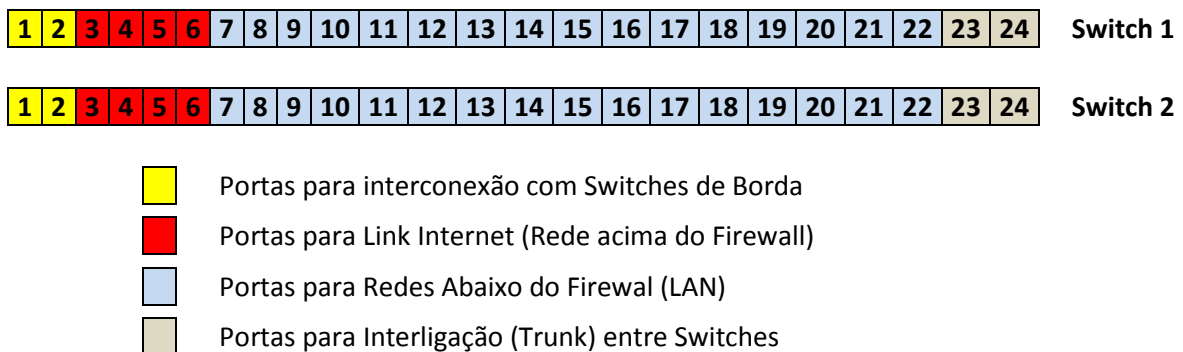
Mesmo com a adição de um novo *switch* a cada grupo de hosts, cada um dos nós de firewall ainda contém apenas uma interface de rede provendo serviço a cada um dos grupos: *Links Internet* e *LAN*.

Obviamente os firewalls funcionam em *Fail Over*, no caso de problemas em um dos nós, o outro assume automaticamente todas as configurações como firewall principal, como endereços IP, redes, regras de filtragem e roteamento. Porém a adição de interfaces evita a total indisponibilidade de um firewall quando da queda de suas interfaces de rede e possibilita a criação de grupos de interface que podem funcionar não só *em Fail Over* como em *Load Balance* e *Link Agregation*.

A utilização de interfaces adicionais, mesmo que em *stand by*, pode diminuir o tempo de paralisação do serviço possibilitando seu uso para substituir uma placa com falhas sem a necessidade de abertura de janela para manutenção, que mesmo por pouco tempo, causará uma indisponibilidade geral para que intervenção seja executada e o problema solucionado.

### 2.3.2 DUPLICAÇÃO DE SWITCHES

A duplicação de switches possibilita uma redundância física a mais na estrutura, porém todas as portas requerem configurações replicadas, como VLAN (Virtual LAN) e IPs, evitando indisponibilidade de alguma rede caso um dos switches fique fora de operação. A **tabela 1** mostra o desenho dos switches necessário para manter a redundância de caminho das conexões.



**Tabela 1: Configuração replicada de Switches**

As portas 1 e 2 são utilizadas para *uplink* de switches de borda acima dos firewalls, mantendo a disponibilidade da estrutura em relação aos switches. As portas 3, 4, 5 e 6 são utilizadas para as 4 interfaces de internet (WAN) dos nós de firewall; 3 e 7 **FWHA01** (eth0 – porta 3, eth1 – porta 7), 4 e 8 **FWHA02** (eth0 – porta 4, eth1 – porta 8). As portas 7 a 22 serão utilizadas pelos hosts servidores abaixo do firewall.

As portas 23 e 24 serão utilizadas para interligação entre os dois switches da estrutura, SW01 e SW02, via configuração *trunk* para que o repasse de pacotes seja executado fisicamente em caso de queda de um dos nós de firewall. Essa

configuração é importante, pois no caso de queda de um ou mais nós de firewall, os hosts continuam se comunicando sem gastar saltos (*hops*) desnecessários sendo obrigados a passar pelos firewalls.

No **anexo 1** pode ser visto um levantamento lógico exemplo completo com os dados de portas, negociação de velocidade de barramento ethernet, identificação de cabeamento por código e cor, hostname do servidor associado, porta do servidor associado, fabricante/modelo do equipamento utilizado e um campo de observações que contém descrição do serviço e escopo de porta utilizado.

## 2.4 ALTA DISPONIBILIDADE

### 2.4.1 CONCEITUAÇÃO

O conceito de alta disponibilidade diz respeito à probabilidade de que um sistema esteja funcional e pronto para uso em um dado instante de tempo. Esse conceito se concretizou como padrão de mercado para produtos e serviços a partir da popularização dos serviços prestados pela rede, protegendo-os de uma eventual queda e mantendo a disponibilidade exigida pelo mercado de 99,999%, principalmente depois do advento da Internet.

A alta disponibilidade é de extrema importância atualmente já que a medida que a demanda pelo consumo de determinado ativo, sendo ele um serviço ou um determinado bem físico, aumenta é necessário que o mesmo tenha um certo grau de disponibilidade que geralmente varia de acordo com a importância do item.

Na solução apresentada foi utilizada a agregação de interfaces de rede em *fail over* fornecendo assim a disponibilidade de hardware necessária para o bom

funcionamento do sistema resguardando-o de eventuais quedas devido a erros físicos e automaticamente aumentando a escalabilidade de rede no caso de uma eventual mudança de topologia exigindo modificações para um eventual aumento de throughput quando necessário.

## 2.4.2 SOFTWARE BONDING

Em sistemas GNU/Linux o software responsável pela agregação (teaming, ou link aggregation) de interfaces de rede é o **ethernet bonding**. Para sua configuração é necessária a utilização do módulo **Bond** já presente por padrão em *kernels vanilla* 2.4, ou mais novos.

Também é de suma importância que os grupos de bonding configurados sejam formados por portas ethernets localizadas em diferentes placas, evitando que a queima de uma determinada interface termine por deixar um grupo de hosts indisponível.

Para a configuração do **Bond** devemos informar ao sistema que o módulo deve ser executado na inicialização com os parâmetros desejados. Para facilitar e padronizar a gerência o arquivo **/etc/modprobe.d/aliases-bond** foi criado com todas as configurações necessárias, cujo conteúdo pode ser visualizado no **anexo 2**. O diretório **/etc/modprobe.d** é padrão em sistemas GNU/Linux e é utilizado para carregar módulos automaticamente, ou seja, qualquer arquivo que contenha configurações de módulo, seguido de seus parâmetros será carregado automaticamente na inicialização do sistema.

Na **tabela 2** abaixo pode ser visualizado o layout de interfaces e grupos de bonding utilizados por cada ativo de firewall.

Bonding	Ethernets	Física	OBS:
Bond0	eth0	1	Rede Acima do Firewall (WAN)
	eth2	2	Rede Acima do Firewall (WAN)
Bond1	eth1	1	Rede Abaixo do Firewall (LAN)
	eth3	2	Rede Abaixo do Firewall (LAN)

**Tabela 2: Configuração de Interfaces em Bonding**

Pode-se notar que o firewall possui duas Ethernets dual port (duas portas ethernet por interface) onde foram criados os *bonds* Bond0 e Bond1 que possuem agrupados duas portas, uma de cada interface de rede *dual port*, mantendo a disponibilidade física do ambiente.

O arquivo de configuração padrão de interfaces de redes, **/etc/network/interfaces**, também é utilizado para informar quais os grupos e placas pertencentes a cada grupo de Bond, conforme pode ser visto no **anexo 3**.

Conforme o arquivo denotado no **anexo 2** o Bond foi configurado em *Fail Over*, de forma a manter uma das interfaces de cada grupo sempre ativa, respondendo pelos serviços de rede e requisições de acesso, enquanto a outra permanece em *stand by*, aguardando um problema físico na primeira para que entre em produção. Essa configuração é denotada pela tag **mode=0**, onde o modo 0 (zero) define o tipo de grupo *Fail Over*. Existem outros grupos como *active-backup* (1), *balance-xor* (2), *broadcast* (3), *802.3ad* (4), *balance-tlb* (5) e *balance-alb* (6).

Os modos de balance supracitados são utilizados para o aumento de throughput uma vez que literalmente somam as velocidades de cada interface em uma grande interface de Bond. A configuração citada não foi necessária para o aumento de banda passante entre redes, portanto não foi utilizada. Para o aumento do throughput também é necessário que as duas interfaces estejam alocadas no mesmo switch e que as portas conectadas sejam configuradas em Port Channel

unindo dois canais em um evitando problemas com spanning tree do switch provenientes da visualização de MAC (Media Access Control) replicado em duas portas diferentes, caracterizando um ataque de Mac Spoofing. Como na solução cada interface encontra-se em switches diferentes não há necessidade.

A configurar o grupo de Bond deve se informar quantas interfaces farão parte do Bond para cada instância carregada do módulo, configuração essa denotada pela tag **max\_bonds=2**. O parâmetro **miimon**, responsável pela especificação da frequência de verificação de link em cada interface do grupo de Bond, foi configurado em 100 mili segundos. Os parâmetros **updelay** e **downdelay** são responsáveis por informar o tempo que o módulo aguardará para respectivamente desativação e ativação dos bonds no caso de uma eventual detecção de queda no link.

### 2.4.3 SOFTWARE HEARTBEAT

Com a adição de um novo nó de firewall há também a necessidade de utilização de um novo IP WAN e LAN para sua gerência, porém esse IP não deve ser utilizado para manter conexões oriundas da rede LAN / DMZ ou WAN, nem tão pouco para criar regras de NAT já que uma vez considerando que um determinado nó encontra-se em link down, os apontamentos de DNS para os servidores da LAN / DMZ se perderiam, bem como suas configurações de NAT. Outro problema é a perda total de configuração de gateway para máquinas internas, uma vez que cada firewall possui um IP diferente da rede abaixo. Esses problemas podem ser melhor visualizados na **Figura 5**.

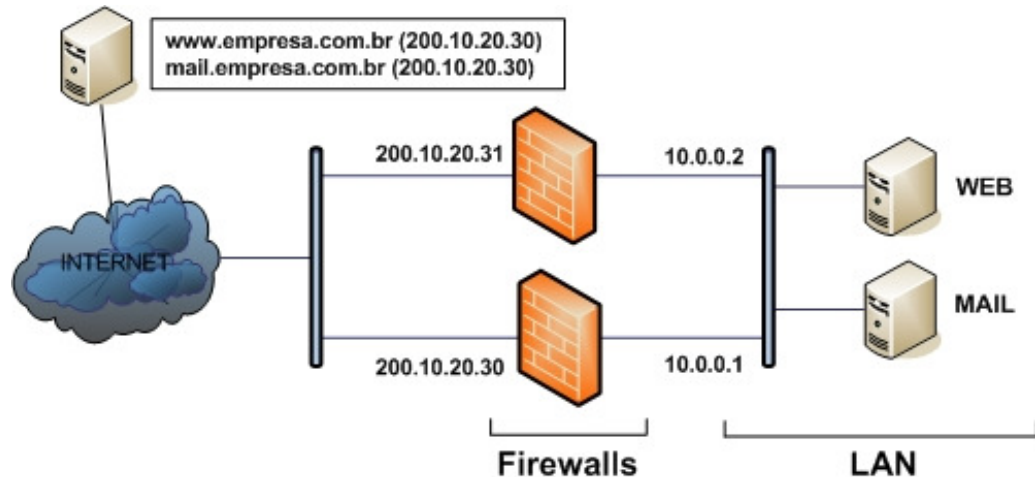


Figura 5: Topologia de Firewall HA com DNS

Para evitar esse tipo de problema adota-se um IP VIP acima e outro abaixo. Um IP VIP, ou virtual, nada mais é que um IP utilizado pelos dois nós de firewall, que fica constantemente alocado no host escolhido como principal e a ele devem ser atribuídas todas as configurações de entrada e saída (NAT) da estrutura. Numa eventual queda de interface (acima ou abaixo) do firewall, ou dele em si, os IPs são automaticamente chaveados para o nó secundário, evitando assim a perda de disponibilidade da estrutura, conforme pode ser melhor visualizado na **Figura 6**.

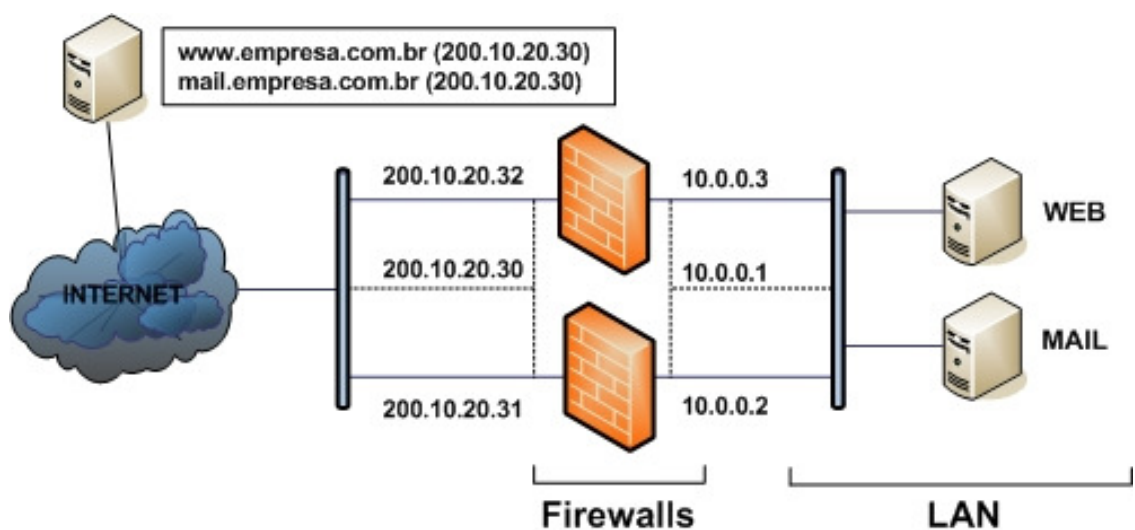


Figura 6: Topologia de Firewall HA com DNS com IP VIP

A esse IP VIP devem ser apontadas todas as configurações de mapeamento de NAT para os hosts servidores abaixo da rede, bem como o apontamento da configuração de gateway da rede LAN/DMZ abaixo.

Na configuração são utilizados três arquivos principais, são eles:

- **/etc/ha.d/authkeys;**
- **/etc/ha.d/ha.cf;**
- **/etc/ha.d/haresources;**

O conteúdo dos arquivos de configuração supracitados deve ser o mesmo em ambos os nós de firewall, caso contrário um não terá permissão de participar do cluster, ocasionando o não chaveamento de IPs, configuração errônea ou diferente nos firewalls que pode levar ao chaveamento parcial ou repetido de IPs.

O arquivo **/etc/ha.d/authkeys (Anexo 3)** é responsável pela configuração de chaves secretas (senha) pela qual permitirá que os nós se comuniquem de forma segura. A chave pode conter qualquer quantidade de caracteres desejada, letras (maiúsculas e minúsculas), símbolos e números. A chave secreta é responsável pelo fechamento de cluster motivo pelo qual a senha deve ser provisionada de forma segura seguindo as boas práticas de segurança atuais.

O arquivo **/etc/ha.d/ha.cf (Anexo 4)** é responsável pela configuração do cluster HA entre os firewalls. Ele especifica a porta e a interface física responsável pela troca de *traps* para verificação dos serviços. Essas são dadas respectivamente pelas tags “**udpport 694**” (para a porta) e “**bcast <interfaces>**”. Por motivos de praticidade e desempenho, o meio escolhido para a transmissão de *traps* foi a interface serial, motivo pelo qual não foi utilizada a tag “**bcast**” e sim “**serial**

`/dev/ttyS0`". Essa alteração faz uso da serial já incluída por padrão em servidores Dell, utilizados no projeto, como um meio reservado apenas para troca de traps, e evita assim que sejam enviadas mensagens broadcast a rede para monitoria de disponibilidade, evitando a perda de desempenho da rede.

Geralmente a tag "**bcast <interfaces>**" é utilizada quando as interfaces para troca de traps são dedicadas. A utilização de interfaces já utilizadas para redes de internet e serviço concorrem tráfego com as já existentes o que dependendo do serviço prestado pode afetar o desempenho de toda a estrutura e contabilizar pacotes em sistemas de gerenciamento de interfaces fazendo com que o administrador do sistema interprete erroneamente os dados levantados pelas interfaces como um aumento de banda decorrente de problemas na rede.

A diretiva **keepalive** é responsável por configurar o intervalo de tempo para o envio de pacotes heartbeat e é especificada em segundos como padrão de unidade de medição de tempo do software. O tempo configurado de três (**keepalive 3**) segundos pode parecer elevado porém se for muito baixo qualquer pequena indisponibilidade do meio físico ou lógico ocasionará chaveamento o que dependendo do caso não simboliza problemas graves no meio e sim pequenas oscilações que podem depender de tráfego ou mal contato em partes móveis do sistema como placas de rede e cabeamento.

A diretiva **deadtime** é utilizada para especificar o quão rápido o software heartbeat deve decidir que um nó do cluster está indisponível. Configurar esse valor muito baixo pode ocasionar ao sistema apresentar falsos positivos na declaração de link down dos nós do cluster, configurar muito alto pode atrasar a retomada de configurações depois de uma eventual falha de um dos nós. Geralmente por motivos

de performance a diretiva é configurada com o triplo de **keepalive** utilizado, no projeto em questão, 9.

A diretiva **node <hostname>** informa quais servidores fazem parte do cluster. O argumento **<hostname>** deve conter o hostname exato de cada um dos nós na ordem crescente de importância no arquivo de configuração, iniciando do nó principal, apresentando o mesmo valor de saída do comando “**uname -n**”. A diretiva deve ser usada uma vez para cada nó, e não há limite de nós. Há formas de adicionar automaticamente nós em um cluster utilizando uma diretiva **autojoin**, porém seguindo os padrões de segurança do projeto, só participam do cluster os servidores que possuem permissão, autenticados pela chave secreta.

A diretiva **ping** é utilizada para declarar nós de ICMP (*Internet Control Management Protocol*) para testes de disponibilidade do protocolo. Cada IP é listado numa diretiva **ping** é considerado independente, ou seja, a conectividade sobre cada nó é considerada com igual importância.

Todos os logs de chaveamento e ocorrência do software heartbeat são configurados pela diretiva **use\_logd <log\_file>** para serem armazenados no diretório **/var/log/ha-debug**.

A diretiva **auto\_failback** determina se um determinado recurso será chaveado automaticamente para seu nó principal, ou permanecerá no nó que estiver servindo até que ele fique indisponível. No cluster do projeto o servidor **FWHA01** foi eleito o principal, portanto caso haja problemas ele chaveia automaticamente suas configurações para o secundário. No momento de retorno do **FWHA01** ele assume automaticamente todas as configurações anteriores e seu papel de principal no cluster.

A diretiva **respawn** é utilizada para especificar o software que será executado e o monitora enquanto está em execução. A diretiva foi configurada para monitorar o **hacluster** utilizando o software **ipfail**. O software **ipfail** provê detecção de erros e falhas na rede, reagindo de forma inteligente e proativa, direcionando o cluster para os recursos de fail over quando necessário através dos nós de ping configurados na diretiva **ping** supracitada. Uma vez que os nós de firewall podem se comunicar via interface serial dedicada, o **ipfail** pode detectar de forma confiável quando um dos nós de rede tem se tornado instável.

O arquivo **/etc/ha.d/haresources** (**Anexo 5**) é responsável pela configuração dos IPs VIPs utilizados no cluster e eleger o nó principal de firewall. O arquivo é formado pela seguinte estrutura:

- Hostname do nó de firewall;
- IPaddr2 seguido do IP VIP com sua máscara e interface utilizada;

A diretiva **hostname** informa o hostname do nó de firewall eleito como o principal para onde todos os IPs VIPs serão chaveados por padrão. O comando **IPaddr2** é utilizado para configurar IPs virtuais nas interfaces escolhidas. Por padrão apenas o nome básico da interface (**eth0**, **eth2**, etc) é utilizado deixando a cargo do software **heartbeat** atribuir a identificação de sub-interface, porém dessa forma perdemos o controle de NAT no script de firewall, devido a esse fato o arquivo **/etc/ha.d/haresources** é configurado definindo-se diretamente a sub-interface por IP utilizado (**eth0:1**, **eth0:2**, etc). Para o projeto em questão foram utilizados dois IPs VIPs, um para WAN, onde serão apontados todos os mapeamentos de serviços via NAT da rede interna, e outro para LAN, que será utilizado como recurso de gateway

abaixo, sobre as interfaces de Bond, já que é utilizada solução para agregation de interfaces (redundância física). Caso necessário, como as redes acima e abaixo são grandes, existem IPs disponíveis na estrutura para aumento da estrutura.

Adicionalmente o arquivo `/etc/ha.d/haresources` podem executar outro software ou script no momento de tomada dos IPs porém para o projeto em questão essa opção não será utilizada.

## 2.5 SCRIPT DE FIREWALL

### 2.5.1 CONCEITUAÇÃO

Procurando aumentar a flexibilidade e facilidade de configuração da solução de firewall a utilização de um script se torna essencial. O script facilita o trabalho de administração uma vez que define hierarquicamente e ordenadamente regras e métodos para filtragem de pacotes e de estados de conexão.

### 2.5.2 BASH SCRIPTING

O script utilizado foi desenvolvido inteiramente em linguagem interpretada bash (*Bourne Again Shell*) script utilizando as boas práticas do mercado para configuração e manutenção.

O script foi desenvolvido de forma estruturada dividida em:

- Definição e inicialização de variáveis;

- Inicialização/Término de módulos do netfilter e configuração de itens de módulos através de `sysctl`;
- Inicialização/Término das políticas padrões de segurança;
- Inicialização de regras de firewall do ambiente;

A função **var\_start()** remete a definição e inicialização de variáveis. Todas as variáveis foram criadas seguindo um padrão de normalização de caixa alta para destaque e melhor visualização no script. As variáveis também seguem um nome padrão definidos por escopo e utilização.

Todos os comandos com maior freqüência de utilização foram atribuídos a variáveis normalizadas em caixa baixa nomeadas com o nome base do comando utilizado. A escolha pela normalização de caixa baixa se deve exatamente a alta freqüência de utilização, que em caixa alta levaria a um maior trabalho administrativo, e a evitar contrastes entre comandos e variáveis que poderia levar a erros da criação de regras. Os comandos **iptables**, **modprobe** e **sysctl** foram normalizados nas variáveis **\$iptables**, **\$modprobe** e **\$sysctl** respectivamente com seu valor incrementado através do comando do sistema Linux, **which**, responsável pelo levantamento de caminho absoluto do binário especificado como argumento. Essa mudança aumenta a flexibilidade do script podendo ser portado para outros sistemas Linux caso necessário pois localiza de forma automatizada o caminho absoluto dos comandos ignorando assim mudanças de caminho devido a alterações do padrão FHS (*File Hierarquical Structure*) executadas por algumas Empresas desenvolvedoras de Distribuições Linux como Red Hat, Suse e Mandrake.

Foram definidas três variáveis para interfaces físicas e lógicas de rede (eth0, eth1, tun01, etc) sendo elas **\$IF\_LO**, **\$IF\_INT** e **\$IF\_EXT**, respectivamente definindo

interfaces de rede loopback, rede interna (LAN) e rede externa (WAN). Foram definidas três variáveis para a utilização dos IPs configurados em cada interface, sendo elas **\$IP\_LO**, **\$IP\_INT** e **\$IP\_EXT**. Foram definidas variáveis para comportar a configuração de rede, definida por IP de rede e máscara de sub-rede em formato CIDR sendo elas **\$NET\_LO**, **\$NET\_EXT**, **\$NET\_INT**, **\$NET\_VPN**, **\$NET\_HA**, respectivamente rede loopback, rede externa, rede interna, rede VPN PPTP L2TP, Rede de traps Heartbeat.

Para o controle de vetores de ataque de DoS (*Denial of Service*) e DDoS (*Distributed Denial of Service*) baseados em protocolo ICMP, como Ping of Death, Smurf, entre outros, foram criadas variáveis para comportar a configuração de IP de broadcast de cada uma das redes protegidas, **\$BRO\_INT**, **\$BRO\_EXT** e **\$BRO\_LO**, e a elas atribuídas regras de filtragem evitando pacotes de ICMP Echo Request (ICMP Type 8) a hosts específicos, multicasts, unicasts ou Broadcasts.

No caso de uso atual foram utilizados na rede abaixo IPs de classe C de segmentos privados sofrendo tradução de endereço IP (NAT) através da interface de saída (**\$IF\_EXT**) do firewall e seu IP (**\$IP\_EXT**), utilizando assim apenas um IP público roteável na Internet, devido a esse fato foram criadas variáveis que comportam os IPs privados da rede DMZ (LAN) a serem utilizados como alvo para reescrita do NAT, de forma a facilitar e diminuir a complexidade do script de firewall. O script pode ser facilmente alterado caso haja mais IPs públicos localizados na interface WAN do servidor e que porventura sofram NAT 1:1 na Interface de rede Interna (LAN). Os IPs acima localizados podem ser denotados em variáveis nomeadas conforme o padrão **\$IP\_NAT<num>\_<Server\_Name>**, onde **\$IP\_NAT** é o subtipo padronizado e normalizado da interface, **<num>** é o número que define

quantidade de IPs públicos utilizados para a tradução, e **<Server\_Name>** o nome do servidor abaixo sofrendo NAT também normalizado em caixa alta.

A função **modules\_start()** é utilizada para execução de todos os módulos netfilter, através do comando **modprobe**, necessários à solução de firewall atual de forma automatizada sem a necessidade de configuração de arquivos padrão **/etc/modules** ou criação de **/etc/modprobe.d/** apenas para atender o firewall. A configuração diretamente no script visa dinamizar o processo de inicialização do serviço de firewall de forma a reduzir os passos necessários para o provisionamento do sistema, reduzindo também o tempo necessário para o levantamento do serviço. Ela também aumenta a integração de configurações de firewall em apenas um script, bem como reduz a complexidade de configuração do serviço.

Essa função também é responsável pela configuração dos itens criados pelo carregamento dos módulos netfilter no kernel utilizando diretamente o software **sysctl** para passar diretamente ao kernel os valores respectivos a cada item.

Os seguintes módulos foram utilizados na solução:

- IP\_CONNTRACK;
- IP\_NAT\_FTP;
- IP\_CONNTRACK\_FTP;

O módulo IP\_CONNTRACK é utilizado para executar rastreamento de conexões, com ele que podemos rastrear conexões de protocolos orientados (TCP, HTTP, etc) e não orientados (IP, ICMP, IGMP, etc) a conexão de forma a manter registros de conexões de todo o firewall em uma tabela ou banco em memória.

O módulo IP\_CONNTRACK\_FTP funciona de forma similar, porém para o protocolo FTP especificamente. Basicamente existem dois modos de funcionamento

de FTP, **passivo** e **ativo**. Em modo passivo as conexões de controle e tráfego de dados são originadas pelo cliente com destino ao servidor. Esse modo não trás problemas caso os servidores estejam atrás de NAT porque ambas as conexões são *outbound* e corretamente traduzidas. No modo ativo a conexão de controle é estabelecida do cliente para o servidor (*outbound*) e na fase final é especificado um par IP/Porta ao qual o servidor deverá conectar para estabelecer conexões *inbound*. Isso dificulta o firewall quando a conexão está atrás de NAT, o módulo IP\_NAT\_FTP resolve esse problema.

Para permitir a passagem de pacotes entre as interfaces foi configurado o item “**net.ipv4.ip\_forward=1**”.

Nessa função foram tomadas medidas de segurança para evitar os tipos de ataques mais comuns a redes de computadores nas camadas 3 e 4 do modelo conceitual OSI. Para evitar ataques baseados em SYN Cookies (ou syn flood) foi configurado o item “**net.ipv4.tcp\_syncookies=1**”. Essa configuração é utilizada para disparar syncookies aos hosts quando a fila de backlog syn do kernel para um determinado socket é transbordada. Isso significa que se o host de firewall é transbordado com pacotes SYN de diferentes hosts, a fila de backlog syn do kernel inicia o envio de cookies para verificar se os pacotes SYN são legítimos.

A função “**net.ipv4.rp\_filter=1**” ativa a filtragem reversa de uma interface específica. A filtragem reversa é utilizada para validar o endereço de origem real utilizado pelos pacotes para correlacionarem corretamente com a tabela de roteamento do firewall, e os pacotes com esse endereço de IP específico que deveriam ter suas respostas retornadas pela mesma interface novamente. Essa configuração é uma contramedida que evita ataques de Spoofing, independente da interface utilizada.

A função “**net.ipv4.icmp\_echo\_ignore\_broadcasts=1**” evita que os IPs de broadcast do servidor sejam atacados por ping (protocolo ICMP) evitando assim ataques do tipo Smurf, entre outros.

A função “**net.ipv4.accept\_source\_route=0**” foi desligada. Essa variável informa ao kernel a não aceitar pacotes com roteamento por source. Pacotes com roteamento por source são geralmente considerados um risco a segurança do sistema, pois permitem que um atacante trafegue dados direcionados a outros roteadores sem que esse tráfego passe pelos roteadores permitidos, ou seja, é uma alteração de roteamento forçada para burlar o bloqueio de roteadores para acessar redes não permitidas.

A opção “**net.ipv4.secure\_redirects=1**” evita que o kernel aceite pacotes ICMP redirecionados de qualquer host de qualquer outro lugar. Quando ligada o kernel aceitará apenas pacotes ICMP de gateways listados na lista de gateways padrões. Isso também evita que o firewall sirva como vetor zumbi de ataques ICMP do tipo Ping of Death, Smurf, e também de ataques Man-in-the-Middle.

A opção “**net.ipv4.log\_martians=1**” é utilizada para informar ao kernel que deve logar todos os pacotes que contenham endereçamentos impossíveis às *facilities* de log do kernel. Um endereço impossível pode ser um IP que o firewall não sabe contatar uma vez que o mesmo não se encontra na tabela de roteamento.

A opção “**net.ipv4.ip\_contrack\_max=300000**” é utilizada para configurar o número máximo de conexões rastreadas pelo kernel. Esse número depende exclusivamente da memória para ser configurado, quanto maior a memória, maior o número de conexões disponível. Cada entrada na tabela de contrack custa 350 bytes de memória ao sistema, portanto o valor de 300.00 foi encontrado devido a análises de performance e carga no servidor utilizado para o projeto.

A função “`policy_start()`” é responsável pela inicialização das políticas de segurança padrão. Foi utilizada uma abordagem Restritiva onde o que não está explicitamente liberado está implicitamente proibido, para isso as chains INPUT e FORWARD da tabela padrão filter foram inicializadas como DROP bloqueando assim qualquer pacote que atravesse ou tenha como destino o firewall em si. Essa função também apaga toda e qualquer chain criada ao longo do script de firewall bem como zera os contadores de pacotes, bytes de todas as tabelas e chains, evitando queda de performance devido a CPU estar ocupada gerenciando contadores que podem ser levantados por outras soluções de gerenciamento. Essa função também é responsável por apagar todas as regras em todas as tabelas e chains do firewall, eliminando por completo as regras. A função “`policy_stop()`” limpa todas as regras, libera todas as políticas retornando o firewall ao estado inicial sem filtragem.

A função “`rules_start()`” é responsável pela inicialização das regras de filtragem do firewall em si, é onde iniciam as regras relativas a rede LAN e WAN. As regras são ordenadas pela ordem de verificação nas *chains* e tabelas do *iptables* bem como a ordem de linhas das regras baseadas no *table traversing* do conjunto de módulos *Netfilter*.

A função supracitada inicia abordando as regras de Statefull Inspection responsáveis pelo rastreamento de conexões baseadas em estados. Foram configurados bloqueios relativos a pacotes inválidos ou não reconhecidos, pelo estado INVALID com destino ao firewall ou atravessando-o em qualquer sentido ou interface. Foram permitidos apenas os estados ESTABLISHED e RELATED relativos à manutenção de conexões de protocolos orientados a conexão e também para regras de retorno. O estado NEW não foi contemplado na configuração, pois ele

subentende qualquer primeiro pacote de conexão destinado ao firewall, podendo esse ser de qualquer tipo como SYN, ACK, entre outros, o que poderia deixar o host vulnerável a diversos tipos de ataques como ACK Poisoning, SYN Flood, entre outros. Os estados ESTABLISHED e RELATED foram liberados com destino e origem do firewall ou atravessando-o em qualquer sentido ou interface.

Todas as regras do script estão orientadas a interface de entrada e saída controlando o tráfego em todos os sentidos, e em conjunto com a opção **sysctl** “**rp\_filter**” evitando ataques de spoofing por interfaces não permitidas.

A Netfilter recomenda que seja feita filtragem de pacotes apenas nas chains existentes na tabela **Filter**, deixando a tabela **Nat** apenas responsável pelo mascaramento (tradução) de IPs, porém a existência de apenas um IP Público roteável no firewall elimina essa boa prática devido a economia de recursos.

Todas as configurações de tradução de IP (NAT) como mapeamento de portas e Nat de saída foram desenvolvidas usando o IP VIP mantendo assim a disponibilidade.

## 2.6 REPLICAÇÃO

### 2.6.1 CONCEITUAÇÃO

A replicação é a operação responsável por sincronizar as configurações de ambos nós de firewall, deixando-os com as informações íntegras para que nenhum dos nós fique desatualizado e se torne indisponível integralmente ou parcialmente devido a chaveamento de IPs pelo software heartbeat.

## 2.6.2 TIPOS DE REPLICAÇÃO

Basicamente existem dois tipos de replicação, **ativo-ativo** e **ativo-passivo**. Na replicação ativo-ativo ambos os hosts recebem incremento de configuração, no caso de queda do nó principal, as configurações de IP são chaveadas para o host secundário que vai assumir como principal, uma vez que suas configurações já estão em conformidade com o primeiro firewall. Caso haja uma nova configuração no nó secundário, essa é replicada para o nó principal sincronamente, mesmo que não esteja disponível para internet. Esse tipo de replicação abre margem para indisponibilidade gerada por erro humano, uma vez havendo uma regra de filtragem mal desenvolvida que torne o nó secundário indisponível, terminará por também deixar indisponível o nó principal, comprometendo todo o ambiente.

Na replicação do tipo **ativo-passivo**, a replicação de regras e configurações funciona apenas um sentido, do nó primário para o secundário. Dessa forma caso alguma configuração seja feita no nó secundário deve ser replicada manualmente para o nó primário. Embora esse tipo de replicação traga um maior esforço administrativo, ela trás mais segurança e disponibilidade ao ambiente, uma vez que se alguma regra for mal desenvolvida no firewall secundário indisponibilizando-o, o ambiente chaveará as configurações para o nó primário que embora possua regras antigas, ainda está ativo, mantendo a disponibilidade do ambiente.

## 2.6.3 SCRIPT DE REPLICAÇÃO

A solução de replicação abordada foi feita a partir do desenvolvimento de script nomeado **sync\_firewall.sh** (Anexo 7) para replicação ativo-passivo replicando as configurações do nó principal para o secundário através de cópia síncrona via tunel **ssh**. O script desenvolvido foi agendado para execução periódica via software **cron** a cada 3 minutos. O script faz uso de uma lista vetorial de arquivos, numerada por posição, que devem sofrer replicação através de seus caminhos absolutos, bem como respectivos comandos necessários para validar as configurações após a replicação.

A lista supracitada deve ser incrementada seguindo o padrão de numerador adotado. O script conecta-se diretamente ao primeiro nó de firewall através do protocolo **ssh** autenticando-se pelo software **sshpas** enviando a senha via túnel criptografado do protocolo. A autenticação é feita através do usuário **fwoperator** criado no nó principal, esse usuário tem somente permissão de execução do comando **cat**, responsável por abrir arquivos textos, sendo essa configuração executada através do software **sudo** também instalado no nó principal. Após a conexão os arquivos apontados no array têm seus hashes verificados, através do software **md5sum**, no nó de origem (FWHA01) e nó de destino (FWHA02), caso sejam iguais a cópia foi feita de forma íntegra e o comando associado ao arquivo é executado, passando assim para o arquivo seguinte, caso o hash md5 seja diferente a cópia do arquivo respectivo é finalizada passando assim para o próximo arquivo.

Caso a conexão não seja possível inicialmente na execução do script, nenhuma sincronização é executada mantendo o sistema íntegro e disponível.

### 3 CONCLUSÃO

Através do caso de uso abordado na prática pode-se concluir que há alternativas viáveis no quesito financeiro e de segurança para a criação e implementação de firewalls corporativos de alta disponibilidade mantendo o grau de maturidade de ferramentas licenciadas já largamente utilizadas no mercado.

A utilização de ferramentas de software livre pode trazer escalabilidade a solução de firewall visto ter código aberto podendo ser atualizadas ou possuir recursos implementados com pouco trabalho administrativo através de profissionais gabaritados e ajuda direta da comunidade de software livre. As ferramentas de software livre também propiciam um corte de custos financeiros que podem ser utilizados na melhoria física da estrutura bem como contratação de pessoal técnico qualificado para mantê-la. Esses cortes evidenciam-se em curto prazo devido ao corte direto de gastos com licenciamento de ferramentas desnecessárias na solução abordada.

Entretanto cabe a organização, procurar estabelecer, como a implementação será executada dependendo do escopo requerido para cada projeto e também da cultura organizacional e tipo de riscos inerentes ao negócio de forma a obterem maior controle.

## REFERÊNCIAS

Albing, C. Vossen, JP. Newham, C, A. *Bash Cookbook*. 1ª Ed O'reilly Media, Inc, 2008.

Daniel J. Barret. Silverman, R. *SSH, The Secure Shell: The Definitive Guide*. 1ª ed. O'Reilly Media, Inc, 2001.

Dulaney, E. *CompTIA Security+ Deluxe Study Guide*. 1ª ed. Sybex / Wiley Publishing, 2009

Gheorghe, L. *Designing and Implementing Linux Firewalls and QoS using netfilter, iproute2, NAT, and L7-filter*. 1ª ed. Packt Publishing, 2006.

Robbins, A. *bash Quick Reference*. 1ª Ed O'reilly Media, Inc, 2006.

Tarreau, W. Gavrilov, C. Tindel N. C. , Girouard, J. Linux Ethernet Bonding Driver mini-howto., 2000. Disponível em : <http://www.kernel.org/pub/linux/kernel/people/marcelo/linux-2.4/Documentation/networking/bonding.txt>. Acesso em 03/07/2010

OFICIAL HEARTBEAT WEBSITE. HA High Availability., 2010. Disponível em : [http://www.linux-ha.org/wiki/Main\\_Page](http://www.linux-ha.org/wiki/Main_Page). Acesso em 03/07/2010

Cisco Systems. Configure Uplink Ethernet PortChannel for Cisco UCS, 2010. Disponível em : [http://www.cisco.com/en/US/products/ps10281/products\\_configuration\\_example09186a0080af091c.shtml](http://www.cisco.com/en/US/products/ps10281/products_configuration_example09186a0080af091c.shtml). Acesso em 03/07/2010

### **Ordenar alfabeticamente e seguir a seguinte norma:**

**Livros:** SOBRENOME, Prenome ou iniciais. *Título*. Edição. Local: Editora, ano, paginação (opcional).

**Fonte eletrônica on line:** SOBRENOME, Prenome ou iniciais. Nome do Artigo. *Nome do periódico* (online), ano. Disponível em: endereço da Internet. Acesso em: dia, mês, ano.

**Obs.:** Quando não há autor, inicia-se pelo título, sendo a 1ª palavra em MAIÚSCULAS. O nome da Instituição ou Entidade Coletiva é usado para iniciar quando se trata de anais de congressos ou trabalhos de cunho administrativo ou legal.

