CONCEITOS E IMPLEMENTAÇÃO DE CGNAT

Marcelo Gondim
Fernando Frediani

O Brasil Peering Fórum

B . P . F
Brasil Peering Fórum

É um **NOG** (**Network Operators Group**) onde profissionais da área trabalham compartilhando conteúdo técnico com o objetivo de fazer uma Internet Brasileira melhor.

Possui uma Wiki aberta (**https://wiki.brasilpeeringforum.org/**) para facilitar o compartilhamento de conteúdo em forma de artigos e tutoriais e também uma Lista de Discussão (**https://listas.brasilpeeringforum.org/**).

Qualquer pessoa cadastrada pode contribuir com novos artigos ou tutoriais ou acrescentar informações complementares.

Acesso gratuito.

O Brasil Peering Fórum

Crie uma conta Entrar

Página principal | Discussão | Ler | Ver código-fonte | Ver histórico | Pesquisar em Wiki BPF

Página principal

Seja bem vindo à Wiki do BPF (Brasil Peering Forum).

O **Brasil Peering Forum** é um NOG (Network Operators Group) onde vários profissionais trabalham com o objetivo de fazer uma Internet Brasileira melhor. Engajados com a comunidade de operadores de Redes e Telecomunicações no Brasil, desempenham papéis instrutivos e participam nos principais eventos do setor, colaborando para o crescimento técnico e operacional dos ISP's e empresas da área de Internet. Contribuem de forma ativa com várias listas de discussão técnicas no Brasil e no mundo, e estão sempre abertos a um bom bate-papo sobre processos para elevar o nível dos ISP's nacionais.

A participação é aberta para a comunidade Internet e gratuita e acontece através dos grupos de trabalho e listas de discussão. Convidamos todos a se inscreverem e participar das discussões na **Lista Geral de Discusão BPF**. Ali são discutidos os assuntos de interesse geral, realizados anúncios para a comunidade, aviso de publicação de novos materiais, etc. O intuito principal da lista é promover a troca de informações, aprendizado e networking entre os participantes. Para se inscrever acesse a página sobre Participação (Listas de Discussão / Task-Forces).

Conheça os detalhes do trabalho desenvolvido pelo BPF nos links abaixo

Artigos em Destaque

Acesso rápido à artigos em destaque e de uso frequente.

- **Como Escrever na Wiki** - Passo a Passo de como criar um novo artigo e contribuir com a Wiki do BPF.
- **CDN Peering e PNI - Brasil** - Lista com as principais CDNs, instruções de como solicitar Servidores, sessões Bilaterias nos IXs e PNIs.

Últimos Artigos Publicados

Lista completa de todos os artigos e materiais publicados na área **Conteúdos**.

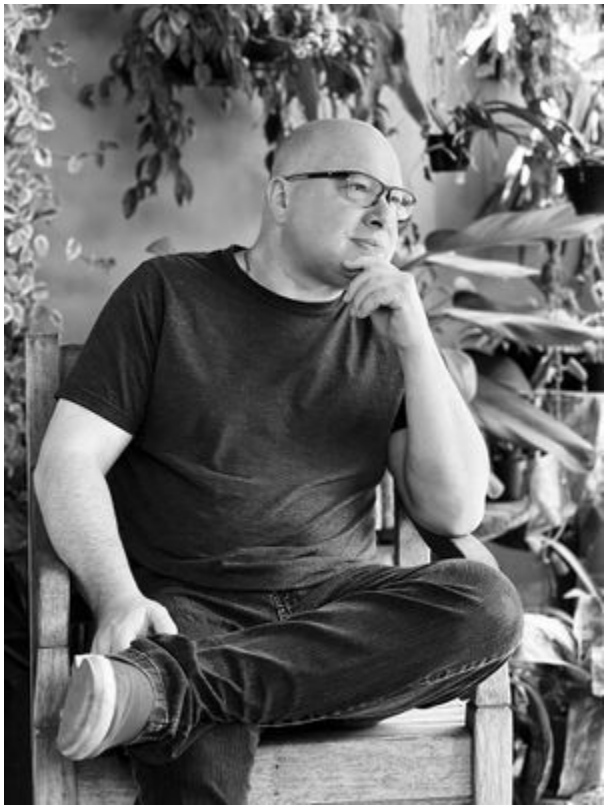
- **Assinatura MoU BPF** - Assinatura do Memorando de Entendimento entre os membros da Board e Comitê de Programa do BPF.
- **O Minimo que Voce precisa saber sobre IRR** - Artigo explicando o que é IRR, a importância do uso, principais bases e com um tutorial de como adicionar informações em uma base.
- **Informativo Infra 07** - 29/12/2019
- **Boas praticas para a implantação do OSPF em ambientes de ISP** - Artigo discorrendo sobre 12 boas práticas em situações envolvendo OSPF em ambientes ISP.
- **Introdução aos Conceitos de Programabilidade de Infraestruturas de Redes** - Artigo um tanto extenso e completo cobrindo os fundamentos de programabilidade de redes
- **Informativo Infra 08** - 26/01/2019
- **Informativo Infra 09** - 16/02/2020
- **Informativo Infra 10** - 09/03/2020
- **DNSSEC - Seguranca do DNS** - Artigo conceitual explicando o funcionando do DNSSEC baseado no documento DNSSEC: Securing DNS publicado pela ICANN.
- **UTRS - Registro e Configuração** - Artigo que explica o funcionamento do serviço UTRS do Team Cymru, passo a passo para solicitação e configurações exemplo.
- **Soluções para o Gerenciamento Efetivo do BGP em um Sistema Autônomo** - Artigo bastante completo dissertando sobre o gerenciamento e monitoramento do BGP em um Sistema Autônomo.

Página principal
Mudanças recentes
Página aleatória
Ajuda

Menu
Quem Somos
Participação
Conteúdos Úteis
Categorias
Documentos Públicos
Agenda

Ferramentas
Páginas afluentes
Mudanças relacionadas
Páginas especiais
Versão para impressão
Ligação permanente
Informações da página

Apresentadores - Marcelo Gondim



- Começou sua carreira como desenvolvedor de software em COBOL e Clipper entre 1992 e 1995. Em 1996 foi responsável por desenvolver um sistema concorrente com o RENPAC da Embratel para acesso ao SISCOMEX e implantou a Internet para fins comerciais na empresa DATABRAS.
- Trabalhou como consultor e instrutor de Linux na Conectiva S/A em 2000.
- Em 2003 se tornou consultor de diversos Provedores de Internet na Região dos Lagos - RJ e onde acabou se tornando CTO da Nettel Telecomunicações (AS53135) com 42.000 assinantes. Implantou IPv6 iniciando em 2013 e se tornou participante do MANRS com diversas contribuições com artigos e palestras.
- Atualmente é Especialista em Redes, cuida do SOC (Security Operations Center) da Brasil TecPar AS262907, onde desenvolve as boas práticas, tratamentos de incidentes relacionados ao ASN e desenvolve as estratégias de Mitigação DDoS da empresa. Também desenvolveu uma Rede de DNS Recursivo Anycast espalhada pelo RS e MT/MS, também certificada KINDNS.

Apresentadores - Fernando Frediani

Engenheiro de Computação graduado pela Pontifícia Universidade Católica de Campinas. Possui especialização pela Universidade de Cranfield no Reino Unido e MBA em Gestão Empresarial pela Fundação Getúlio Vargas no Brasil.

Na empresa Americanet/Ultrawave exerce o cargo de Gerente de Engenharia. Atuou como consultor de diversos Provedores de Serviços e Banda Larga com foco em Infraestrutura. Também exerceu a função de Gerente de Engenharia e Infraestrutura na empresa UPX Technologies, Systems Architect na NTT Europe e Lead Systems Engineer na

Qube Managed Services, ambas em Londres, Reino Unido.

Desenhou e implantou diversos projetos de Cloud e Infraestrutura como Serviço em países como Reino Unido, Estados Unidos, Espanha, França, Suíça e Alemanha.

Membro fundador e atualmente membro da Diretoria do Brasil Peering Fórum (https://wiki.brasilpeeringforum.org), participa também de diversos fóruns relacionados à Governança de Internet como Fórum de Políticas do LACNIC, ARIN e AfriNic. Foi aluno da Escola Brasileira de Governança da Internet (EGI.br)

Palestrante em eventos do setor de Internet no Brasil e no exterior.

B.P.F
Brasil Peering Fórum

Introdução

- Surgiu devido à escassez de IPv4 disponível para os Provedores de Acesso.
- Alocação de Endereços definida pela RFC6598.
  - Range 100.64.0.0/10.
  - Não é o mesmo que RFC1918.
- Uma maneira sustentável e organizada para continuar provendo acesso até a transição completa para IPv6.
- CGNAT é NAT.
- CGNAT “não é NAT”.

Aspectos Legais

- Importância do registro e guarda de logs para identificação do usuário.
    - Art. 10, Art. 13 e Art. 15 do Marco Civil.
- Somente o endereço IP de origem não é suficiente. É necessário haver o registro da porta de origem também.
    - Interpretações do Judiciário no sentido da obrigatoriedade da guarda também da porta de origem.
- Não se deve jamais registrar endereço de destino para este propósito (violação da privacidade).
- Provedores de conteúdo devem também guardar os registros de porta de origem, caso contrário a identificação não é possível.

Tipos de CGNAT - Determinístico

- Mais utilizado pelos provedores em geral pela facilidade de implementação.
- Define um range limitado de portas TCP e UDP por usuário para ser utilizado.
- Permite uma economia razoável de endereços IPv4 Públicos à depender do nível de compartilhamento realizado.
- Requer uma quantidade bem menor de log (apenas os de autenticação e atribuição do IP da range de CGNAT).

Tipos de CGNAT - Determinístico

- Exemplo 1 – Compartilhamento 1:32
  - 32 assinantes compartilham o mesmo IPv4 Público
  - 2016 portas de origem alocadas para cada IP Privado
- Exemplo 2 – Compartilhamento 1:16
  - 16 assinantes compartilham o mesmo IPv4 Público
  - 4032 portas de origem alocadas para cada IP Privado
- Exemplo 3 – Compartilhamento 1:8
  - 8 assinantes compartilharão o mesmo IPv4 Público
  - 8064 portas de origem alocadas para cada IP Privado

```
iptables -t nat -A CGNAT -s 100.64.18.10 -p tcp -j SNAT --to
192.0.0.1:3040-5055
iptables -t nat -A CGNAT -s 100.64.18.10 -p udp -j SNAT --to
192.0.0.1:3040-5055
```

Tipos de CGNAT – Bulk Port Allocation

- Realiza a atribuição de portas de origem para cada IP de CGNAT de maneira dinâmica e em blocos, conforme a necessidade de cada assinante.
- Define um range máximo de portas TCP e UDP inicial por usuário e blocos adicionais à serem alocados posteriormente conforme a necessidade de cada um.
- Permite uma economia maior de endereços IPv4 Públicos pois a maioria dos usuários não utilizam um alto número de portas e um mesmo IPv4 pode ser utilizado por uma quantidade maior de usuários.
- Requer registro de log devido às alocações serem realizadas de maneira dinâmica.

Tipos de CGNAT – Bulk Port Allocation

- Exemplo 1
  - Assinante recebe inicialmente uma alocação de 512 portas para uso.
  - Quando atingir uso do número de portas alocadas o sistema alocará blocos adicionais de 512 portas (não contíguas) para o mesmo IP Privado utilizado pelo assinante.
- Exemplo 2
  - Assinante recebe inicialmente uma alocação de 256 portas para uso.
  - Quando estiver perto de atingir uso do número de portas alocadas o sistema alocará blocos adicionais de 128 portas (não contíguas) para o mesmo IP Privado utilizado pelo assinante.
- Cada nova alocação gera uma entrada nos logs.

Processo de Identificação de Usuário

IP Destino: 200.160.2.3
Porta de Destino: 443
IP de Origem: 198.51.100.22
Porta de Origem: 48122

IP Destino: 200.160.2.3
Porta de Destino: 443
IP de Origem: 100.64.1.22
Porta de Origem: 21402

B . P . F
Brasil Peering Fórum

Processo de Identificação de Usuário

Log do Servidor

```
198.51.100.22 48122 - servidor-web [28/Apr/2023:10:22:44 -0300] "GET
/index.html HTTP/1.1" 201 1126 "-" ""Mozilla/5.0 (Windows NT 10.0; Win64;
x64; rv:59.0) Gecko/20100101 Firefox/59.0"
```

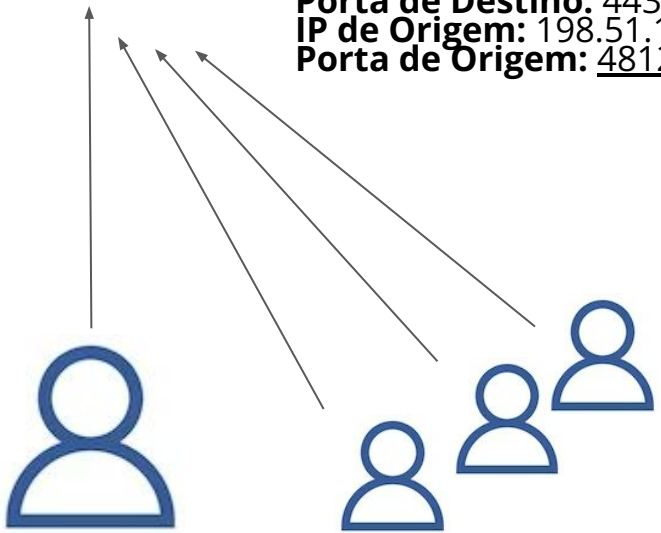
Processo de Identificação de Usuário

IP Destino: 200.160.2.3
Porta de Destino: 443
IP de Origem: 198.51.100.22
Porta de Origem: 23482

IP Destino: 200.160.2.3
Porta de Destino: 443
IP de Origem: 198.51.100.22
Porta de Origem: 18680

IP Destino: 200.160.2.3
Porta de Destino: 443
IP de Origem: 198.51.100.22
Porta de Origem: 48122

IP Destino: 200.160.2.3
Porta de Destino: 443
IP de Origem: 198.51.100.22
Porta de Origem: 14288



IP Destino: 200.160.2.3
Porta de Destino: 443
IP de Origem: 100.64.1.22
Porta de Origem: 21402

B . P . F
Brasil Peering Fórum

Processo de Identificação de Usuário

- **Informações necessárias serem fornecidas pelo Provedor de Conteúdo para identificação**
  - IP de Origem
  - Porta de Origem
  - Data e horário do acesso ao conteúdo (com fuso horário)
- Com essas informações é solicitado ao **Provedor de Acesso** que consulta seus registros.
  - No caso do **IP de Origem ser IP Público alocado para CGNAT** verifica seus registros/logs para identificar qual o IP Interno de CGNAT foi utilizado
    - Para isso ser possível é **necessária a informação da Porta de Origem**
  - Verifica-se então nos registros de autenticação qual usuário recebeu aquele IP naquela data e horário.

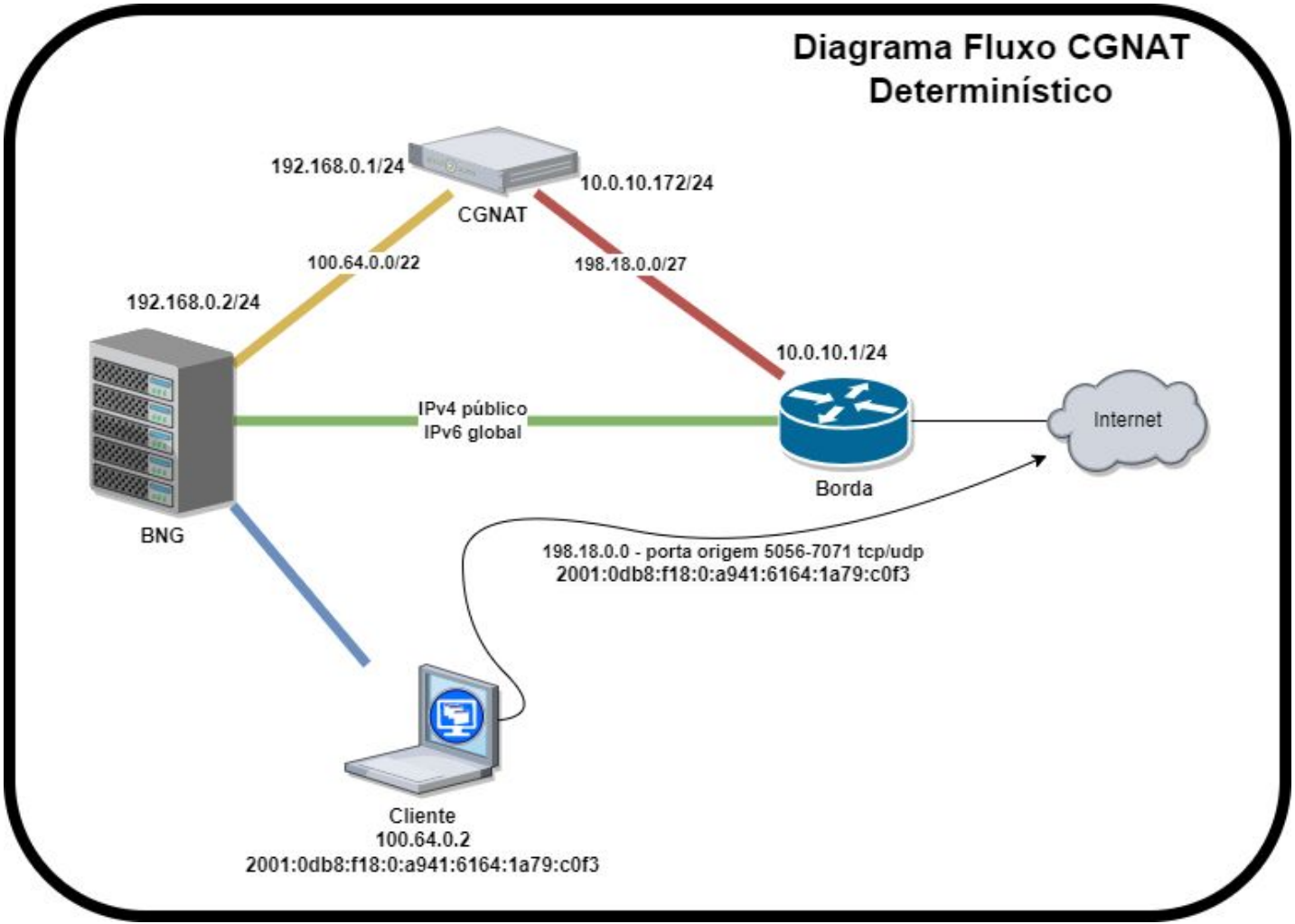


Diagrama Fluxo CGNAT
Determinístico
192.168.0.1/24
CGNAT
10.0.10.172/24
100.64.0.0/22
198.18.0.0/27
192.168.0.2/24
10.0.10.1/24
IPv4 público
IPv6 global
Internet
Borda
BNG
198.18.0.0 - porta origem 5056-7071 tcp/udp
2001:0db8:f18:0:a941:6164:1a79:c0f3
Cliente
100.64.0.2
2001:0db8:f18:0:a941:6164:1a79:c0f3

B . P . F
Brasil Peering Fórum

CGNAT Determinístico 1/32 com GNU/Linux Debian 11 (Bullseye)

debian

Hardware e Sistema que utilizaremos neste tutorial:

B . P . F
Brasil Peering Fórum

- 2x Intel® Xeon® Silver 4215R Processor (3.20 GHz, 11M Cache, 8 Ambiente **NUMA (non-uniform memory access)**.
- 32Gb de ram.
- 2x SSD 240 Gb RAID1.
- 2x Interfaces de rede Intel XL710-QDA2 (2 portas de 40 Gbps).
- GNU/Linux Debian 11 (Bullseye).

Vamos configurar um **LACP** com as duas portas de cada interface, para que possamos ter um backup, caso algum módulo apresente algum problema. Seu ambiente de produção pode ser diferente e por isso precisamos ter alguns cuidados na hora de montarmos o conjunto de hardware e não obtermos surpresas.

1º Verifique algumas especificações da interface de rede que será usada. Por exemplo a **Intel XL710-QDA2**:

- 2 portas de 40 Gbps.
- PCIe 3.0 x8 (8.0 GT/s).

Com essa informação seu equipamento não poderá possuir slots PCIe inferiores a esta especificação, caso contrário terá problemas de desempenho.

Você também precisa estar atento para as limitações de barramento por **versão** x **lane (x1)**:

- PCIe 1.0/1.1 - 2.5 GT/s - (8b/10b encoding) - 2 Gbps.
- PCIe 2.0/2.1 - 5.0 GT/s - (8b/10b encoding) - 4 Gbps.
- PCIe 3.0/3.1 - 8.0 GT/s - (128b/130b encoding) - ~7,88 Gbps.
- PCIe 4.0 - 16 GT/s - (128b/130b encoding) - ~15,76 Gbps.

Calculando a capacidade:

Se observarmos a **XL710-QDA2** é **PCIe 3.0 x8 (8 lanes)** ou seja o barramento irá suportar:

- 8.0 GT/s * (128b/130b encoding) * 8 lanes = 63,01 Gbps

O objetivo do **LACP** nesse caso, não seria alcançar os **80 Gbps** de capacidade em cada interface, mesmo porque cada barramento das interfaces é limitado em **63,01 Gbps**, mas manteremos um backup dos **40 Gbps**.

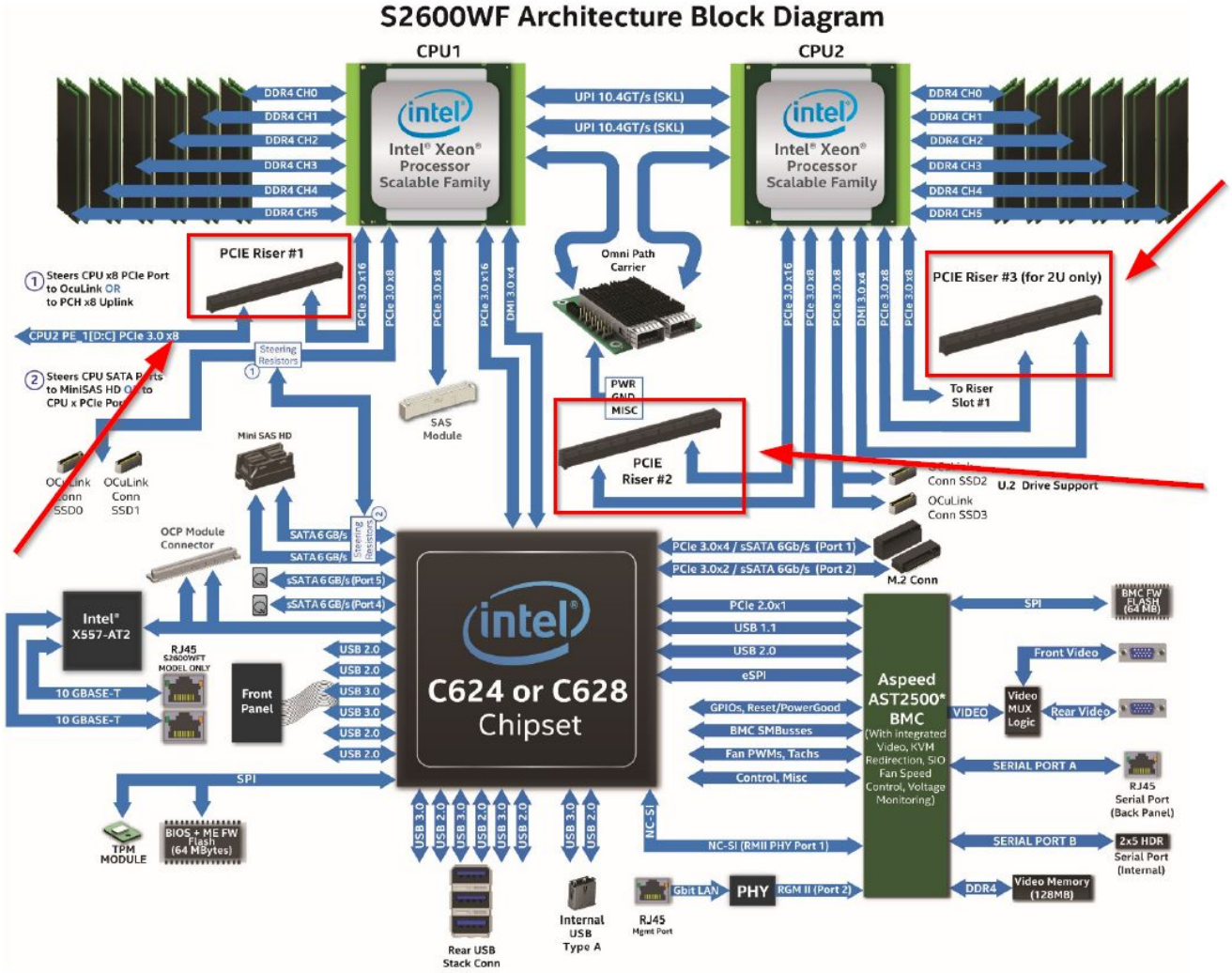
Nessa configuração teríamos teoricamente **63,01 Gbps de entrada** e **63,01 Gbps de saída**. Mas para esse cenário precisaremos fazer uma coisa chamada **CPU Affinity**. Nesse caso colocaríamos um processador dedicado para cada interface de rede. É um cenário mais complexo do que com 1 processador apenas, inclusive necessitamos de olhar o **datasheet da motherboard** e identificar quais slots PCIe são diretamente controlados por qual CPU. Se temos a **CPU0** e **CPU1**, uma interface precisará ficar no **slot controlado pela CPU0** e a outra interface no **slot controlado pela CPU1** e observar a quantidade de lanes no slot para ver se suporta a mesma quantidade de lanes da interface de rede.

Falando um pouco sobre **PPS (Packet Per Second)** para calcular por exemplo **1 Gbps de tráfego** na ethernet, a quantidade de PPS que o sistema precisaria suportar encaminhar teríamos: **1.000.000.000/8/1518 = 82.345 packets per second**.

Existe um comando no **GNU/Linux** para você saber se o seu equipamento com **processadores físicos**, conseguirá trabalhar com o **CPU Affinity**:

```
# cat /sys/class/net/<interface>/device/numa_node
```

Se o resultado do comando acima for **-1** então esse equipamento não trabalhará com o **CPU Affinity**. Isso porque cada interface precisa estar sendo gerenciada por um **node específico**. Se são **2 processadores** então o resultado deveria ser **0** de **CPU0** ou **1** de **CPU1**.
No próximo slide veremos um exemplo de datasheet da **motherboard S2600WF**.


S2600WF Architecture Block Diagram
CPU1
CPU2
Intel® Xeon® Processor Scalable Family
UPI 10.4GT/s (SKL)
DDR4 CH0
DDR4 CH1
DDR4 CH2
DDR4 CH3
DDR4 CH4
DDR4 CH5
PCIE Riser #1
PCIE Riser #3 (for 2U only)
PCIE Riser #2
Omni Path Carrier
PWR
GND
MISC
SAS Module
Mini SAS HD
OCuLink Conn SSD0
OCuLink Conn SSD1
OCuLink Conn SSD2
OCuLink Conn SSD3
U.2 Drive Support
To Riser Slot #1
OCP Module Connector
M.2 Conn
Intel® X557-AT2
RJ45
10 GBASE-T
Front Panel
C624 or C628 Chipset
SPI
TPM MODULE
BIOS + ME FW Flash (64 MBytes)
Rear USB Stack Conn
Internal USB Type A
RJ45 Mgmt Port
Gbit LAN
PHY
RGM II (Port 2)
NC-SI (RMII PHY Port 1)
Aspeed AST2500* BMC
(With integrated Video, KVM Redirection, SIO Fan Speed Control, Voltage Monitoring)
VIDEO
Video MUX Logic
Front Video
Rear Video
BMC FW FLASH (64 MB)
SERIAL PORT A
RJ45 Serial Port (Back Panel)
SERIAL PORT B
2x5 HDR Serial Port (Internal)
DDR4
Video Memory (128MB)
GPIOs, Reset/PowerGood
BMC SMBusses
Fan PWMs, Tachs
Control, Misc
PCIe 2.0x1
USB 1.1
USB 2.0
eSPI
PCIe 3.0x4 / sSATA 6Gb/s (Port 1)
PCIe 3.0x2 / sSATA 6Gb/s (Port 2)
sSATA 6 GB/s (Port 5)
sSATA 6 GB/s (Port 4)
SATA 6 GB/s
Steering Resistors
CPU2 PE_1[D:C] PCIe 3.0 x8
1 Steers CPU x8 PCIe Port to OcuLink OR to PCH x8 Uplink
2 Steers CPU SATA Ports to MiniSAS HD OR to CPU x PCIe Port

Se observarmos o datasheet acima veremos que temos o **PCIe Riser** **Riser #3**. Cada Riser possui **slots PCIe** que são gerenciados por de
Se colocássemos as duas interfaces de rede nos slots do **Riser #**
estaríamos pendurando tudo apenas no **processador 2**.

Isso foi apenas para mostrar a complexidade de quando usamos um equipamento **NUMA** e estamos somente escolhendo o hardware adequado. Ainda não chegamos na configuração do **CPU Affinity**.

Para sabermos quais cores estão relacionados para uma determinada CPU, utilizamos os comandos abaixo:

```
# cat /sys/devices/system/node/node0/cpulist
0-7

# cat /sys/devices/system/node/node1/cpulist
8-15
```

No exemplo acima a **CPU0** tem os cores de **0** a **7** e a **CPU1**, os cores de **8** a **15**, ou seja, é um equipamento com **16 cores**.

Também é importante, para aumento de performance, que seja desabilitado na BIOS o **HT (Hyper Threading)**.

Antes de configurarmos algumas coisas no nosso ambiente, precisarem ferramenta importante para o nosso tuning; vamos instalar o pacote **ethtool**. fazermos alguns ajustes na nossa interface de rede. Alguns fabricantes poc certas alterações mas com as interfaces da Intel sempre obtive os resultados esperados.


```
# apt install ethtool
```

No nosso exemplo acima vimos que o equipamento possui **16 cores** sendo que **8 cores por CPU**. Então, para esse caso, faremos um ajuste nas interfaces para ficarem preparadas para receberem 8 cores em cada através das IRQs. Usamos o **parâmetro -l** do **ethtool** para listar o **Pre-set maximums combined** da interface e o **parâmetro -L** para alterar esse valor. Façamos então a alteração:

```
# ethtool -L enp5s0f0 combined 8
# ethtool -L enp5s0f1 combined 8
# ethtool -L enp6s0f0 combined 8
# ethtool -L enp6s0f1 combined 8
```

Com os comandos acima deixamos preparadas as interfaces para aceitarem 8 cores em cada uma através das IRQs.

Não podemos usar o programa **irqbalance** para o **CPU Affinity**, pois este faz migração de contextos entre os cores e isso é ruim. Como no nosso exemplo estamos usando uma interface Intel, utilizaremos um script da própria Intel para realizar o CPU Affinity de forma mais fácil. Esse script se chama **set_irq_affinity** e vem acompanhado com os fontes do driver da interface. Aqui por exemplo:

Intel Network Adapter

Agora que preparamos as interfaces, façamos os apontamentos dos cores da seguinte forma. Vamos supor que colocamos o script em **/root/scripts**:

```
# /root/scripts/set_irq_affinity 0-7 enp5s0f0
# /root/scripts/set_irq_affinity 0-7 enp5s0f1
# /root/scripts/set_irq_affinity 8-15 enp6s0f0
# /root/scripts/set_irq_affinity 8-15 enp6s0f1
```

Vamos fazer mais alguns ajustes nas interfaces com o **ethtool**. Dessa vez vamos aumentar os **Rings RX** e **TX**. Mas antes vamos listar os valores que podemos usar:

```
# ethtool -g enp5s0f0
Ring parameters for enp5s0f0:
Pre-set maximums:
RX:             4096
RX Mini:        n/a
RX Jumbo:       n/a
TX:             4096
Current hardware settings:
RX:             512
RX Mini:        n/a
RX Jumbo:       n/a
TX:             512
```

Acima vemos que o valor máximo é de **4096** tanto para **TX**, quanto para **RX** mas está configurado para **512** em **RX** e **TX**. Façamos então:

```
# ethtool -G enp5s0f0 rx 4096 tx 4096
# ethtool -G enp5s0f1 rx 4096 tx 4096
# ethtool -G enp6s0f0 rx 4096 tx 4096
# ethtool -G enp6s0f1 rx 4096 tx 4096
```

Vamos desabilitar as seguintes options das interfaces: **TSO**, **GRO** e **GSO**.

```
# ethtool -K enp5s0f0 tso off gro off gso off
# ethtool -K enp5s0f1 tso off gro off gso off
# ethtool -K enp6s0f0 tso off gro off gso off
# ethtool -K enp6s0f1 tso off gro off gso off
```

Aumentaremos o **txqueuelen** para **10000**:

```
# ip link set enp5s0f0 txqueuelen 10000
# ip link set enp5s0f1 txqueuelen 10000
# ip link set enp6s0f0 txqueuelen 10000
# ip link set enp6s0f1 txqueuelen 10000
```

Tudo que fizemos até o momento será perdido no próximo reboot do sistema, então faremos com que esses comandos sejam executados sempre que o sistema iniciar. Para isso vamos deixar o nosso arquivo **/etc/network/interfaces** configurado conforme nosso diagrama, usando **LACP** e executando nossos comandos anteriores.

Antes precisaremos instalar o pacote **ifenslave** para que o **bonding** funcione:

```
# apt install ifenslave
# modprobe bonding
# echo "bonding" >> /etc/modules
```

O nosso **/etc/network/interfaces**

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

auto bond0
iface bond0 inet static
        bond-slaves enp5s0f0 enp5s0f1
        bond_mode 802.3ad
        bond-ad_select bandwidth
        bond_miimon 100
        bond_downdelay 200
        bond_updelay 200
        bond-lacp-rate 1
        bond-xmit-hash-policy layer2+3
        address 10.0.10.172/24
        gateway 10.0.10.1
        pre-up /usr/sbin/ethtool -L enp5s0f0 combined 8
        pre-up /usr/sbin/ethtool -L enp5s0f1 combined 8
        pre-up /root/scripts/set_irq_affinity 0-7 enp5s0f0
        pre-up /root/scripts/set_irq_affinity 0-7 enp5s0f1
        pre-up /usr/sbin/ethtool -G enp5s0f0 rx 4096 tx 4096
        pre-up /usr/sbin/ethtool -G enp5s0f1 rx 4096 tx 4096
        pre-up /usr/sbin/ethtool -K enp5s0f0 tso off gro off gso off
        pre-up /usr/sbin/ethtool -K enp5s0f1 tso off gro off gso off
        pre-up /usr/sbin/ip link set enp5s0f0 txqueuelen 10000
        pre-up /usr/sbin/ip link set enp5s0f1 txqueuelen 10000
```

B.P.F
Brasil Peering Fórum

```
auto bond1
iface bond1 inet static
        bond-slaves enp6s0f0 enp6s0f1
        bond mode 802.3ad
        bond-ad select bandwidth
        bond miimon 100
        bond downdelay 200
        bond updelay 200
        bond-lacp-rate 1
        bond-xmit-hash-policy layer2+3
        address 192.168.0.1/24
        pre-up /usr/sbin/ethtool -L enp6s0f0 combined 8
        pre-up /usr/sbin/ethtool -L enp6s0f1 combined 8
        pre-up /root/scripts/set irq affinity 8-15 enp6s0f0
        pre-up /root/scripts/set irq affinity 8-15 enp6s0f1
        pre-up /usr/sbin/ethtool -G enp6s0f0 rx 4096 tx 4096
        pre-up /usr/sbin/ethtool -G enp6s0f1 rx 4096 tx 4096
        pre-up /usr/sbin/ethtool -K enp6s0f0 tso off gro off gso off
        pre-up /usr/sbin/ethtool -K enp6s0f1 tso off gro off gso off
        pre-up /usr/sbin/ip link set enp6s0f0 txqueuelen 10000
        pre-up /usr/sbin/ip link set enp6s0f1 txqueuelen 10000
```

Colocaremos o **kernel do backports**. Para isso deixe o seu **/etc/apt/sources** conforme abaixo e rode os comandos na sequência:

```
deb http://security.debian.org/debian-security bullseye-security main contrib non-free
deb http://deb.debian.org/debian bullseye main non-free contrib
deb http://deb.debian.org/debian bullseye-updates main contrib non-free
deb http://deb.debian.org/debian bullseye-backports main contrib non-free
```

```
# apt update
# apt install -t bullseye-backports linux-image-amd64
# reboot
```

Protegendo contra static loop e preparando o ambiente do CGNAT:

B . P . F
Brasil Peering Fórum

O **static loop** é algo que, definitivamente, pode derrubar toda a sua opera devidamente tratado e pode ser facilmente explorado por pessoas m
A causa do problema é uma rota estática para um prefixo IP (seja IPv4 ou IPv6), que aponta para um next-hop e nesse destino não existe nenhuma informação sobre o prefixo IP na tabela de rotas local, obrigando o pacote a retornar para o seu gateway default e ficando nesse loop até que expire o **TTL (Time To Live)** do pacote. Isso ocorre muito nos casos em que temos **concentradores PPPoE (BNG)** e **caixas CGNAT** como esta que estaremos fazendo. Na wiki do **Brasil Peering Fórum** temos um artigo que fala sobre esse problema e outras recomendações de segurança:

Recomendações sobre Mitigação DDoS

Crie um arquivo **/etc/rc.local** e dentro colocaremos algumas coisas como as blackholes para cada prefixo IPv4 público que usaremos no nosso servidor de exemplo e rotas de retorno para o nosso BNG:

```
# > /etc/rc.local
# chmod +x /etc/rc.local
```

Dentro teremos:

```
#!/bin/sh -e
/usr/sbin/ip route add blackhole 198.18.0.0/27 metric 254
/usr/sbin/route add -net 100.64.0.0/22 gw 192.168.0.2
```

No exemplo acima estamos colocando em **blackhole** o nosso prefixo IPv4 público deste tutorial que é o **198.18.0.0/27** e adicionando uma rota de retorno do prefixo **100.64.0.0/22** usado no nosso BNG para o **next-hop 192.168.0.2**.

Redução dos tempos de timeouts:

Os **tempos padrões** dos **timeouts** de **tcp** e **udp** são altos para o no CGNAT, ainda mais quando estamos diminuindo a quantidade de portas tcp/udp por assinante e com isso podemos rapidamente estourar esse limite, fazendo com que o sistema pare de funcionar. Abaixo estou colocando os valores que sempre usei e não percebi problemas, mas você pode ajustar conforme achar mais prudente. Adicionaremos as configurações abaixo também no nosso **/etc/rc.local**:

```
echo 5 > /proc/sys/net/netfilter/nf conntrack tcp timeout syn sent
echo 5 > /proc/sys/net/netfilter/nf conntrack tcp timeout syn recv
echo 86400 > /proc/sys/net/netfilter/nf conntrack tcp timeout established
echo 10 > /proc/sys/net/netfilter/nf conntrack tcp timeout fin wait
echo 10 > /proc/sys/net/netfilter/nf conntrack tcp timeout close wait
echo 10 > /proc/sys/net/netfilter/nf conntrack tcp timeout last ack
echo 10 > /proc/sys/net/netfilter/nf conntrack tcp timeout time wait
echo 10 > /proc/sys/net/netfilter/nf conntrack tcp timeout close
echo 300 > /proc/sys/net/netfilter/nf conntrack tcp timeout max retrans
echo 300 > /proc/sys/net/netfilter/nf conntrack tcp timeout_unacknowledged
echo 10 > /proc/sys/net/netfilter/nf_conntrack_udp_timeout
echo 180 > /proc/sys/net/netfilter/nf_conntrack_udp_timeout_stream
echo 10 > /proc/sys/net/netfilter/nf_conntrack_icmp_timeout
echo 600 > /proc/sys/net/netfilter/nf_conntrack_generic_timeout
```

Faremos alguns tunings no sistema antes:

Em **/etc/sysctl.conf** adicionaremos:

```
net.core.default qdisc=fq
net.ipv4.tcp_congestion_control=bbr
net.core.rmem max = 2147483647
net.core.wmem max = 2147483647
net.ipv4.tcp_rmem = 4096 87380 2147483647
net.ipv4.tcp wmem = 4096 65536 2147483647
net.ipv4.conf.all.forwarding=1
net.netfilter.nf conntrack helper=1
net.netfilter.nf conntrack buckets = 512000
net.netfilter.nf_conntrack_max = 4096000
vm.swappiness=10
```

As configurações acima melhoram o uso de memória, habilita o encaminhamento dos pacotes e aumenta a quantidade máxima de **conntracks** do sistema para **4096000**

Se o **conntrack** estourar, seu CGNAT terá problemas e causará indisponibilidades. Para consultar a quantidade de **conntracks** em uso:

```
# cat /proc/sys/net/netfilter/nf_conntrack_count
```

Para listar as **conntracks**:

```
# cat /proc/net/nf_conntrack
```

Ajustando a data e horário do sistema:

Uma tarefa muito importante a ser feita nos servidores, é garantir que o horário e data estejam corretos e para isso usaremos o programa **chrony**. Eu prefiro usar sempre horário **UTC** nos servidores e fazer a conversão quando necessário:

```
# apt install chrony
```

```
# cat << EOF > /etc/chrony/chrony.conf
confdir /etc/chrony/conf.d
sourcedir /run/chrony-dhcp
sourcedir /etc/chrony/sources.d
keyfile /etc/chrony/chrony.keys
driftfile /var/lib/chrony/chrony.drift
ntsdumpdir /var/lib/chrony
logdir /var/log/chrony
maxupdateskew 100.0
rtcsync
makestep 1 3
leapsectz right/UTC
EOF
```

```
# cat << EOF > /etc/chrony/sources.d/nic.sources
server a.st1.ntp.br iburst nts
server b.st1.ntp.br iburst nts
server c.st1.ntp.br iburst nts
server d.st1.ntp.br iburst nts
EOF
```

```
# systemctl restart chronyd.service
# timedatectl set-timezone "UTC"
```

No arquivo **/etc/modules** adicionaremos os módulos que usaremos no nosso CGNAT, inclusive os **ALGs (Application Layer Gateway)**. Sem eles alguns serviços, ainda muito utilizados, apresentarão problemas.

Em **/etc/modules** adicionaremos mais os módulos abaixo:

```
nf_conntrack
nf_nat_pptp
nf_nat_h323
nf_nat_sip
nf_nat_irc
nf_nat_ftp
nf_nat_tftp
```

Antes de começarmos nossas regras de CGNAT precisaremos de alguns pacotes:

```
# apt install python3-pip nftables
# pip install ipaddress
```

Vamos precisar também de um gerador de regras de CGNAT para **nftables**. Porque criar as regras manualmente não é uma tarefa rápida e para isso usaremos um programa em python criado por **José Beiriz** e disponibilizado aqui: **GRCN**

Nosso script será dividido em 2 partes:

- O script base que colocaremos em **/root/scripts** chamado de **frw-nft.sh**. Esse script conterá as regras básicas do CGNAT e este incluirá a chamada para os outros arquivos de regras propriamente ditos do CGNAT.
- Essa outra parte é composta pelos arquivos de regras de CGNAT, onde são feitas as traduções de IPs privados **100.64.0.0/10 (Shared Address Space - RFC6598)**, para os IPs públicos. No próximo slide apresentamos o **frw-nft.sh**.

Nosso script de CGNAT base **/root/scripts/frw-nft.sh**:

```
#!/usr/sbin/nft -f
# limpa todas as regras da memoria
flush ruleset

# regras base para o CGNAT
add table ip nat
add chain ip nat POSTROUTING { type nat hook postrouting priority100; policy accept; }
add chain ip nat CGNATOUT

# libera o proprio CGNAT para acessar a Internet - para atualizacoes por exemplo
add rule ip nat POSTROUTING oifname "bond0" ip saddr 10.0.10.172 counter snat to
198.18.0.0

# faz o jump para as regras de CGNAT
add rule ip nat POSTROUTING oifname "bond0" counter jump CGNATOUT

# carrega os arquivos de regras de CGNAT
include "/root/scripts/cgnat-0-31.conf"
```

A última linha do script acima, em vermelho, é o arquivo de regras CGNAT que iremos gerar e será chamado pelo script quando for executado.

Após a criação do script, alteramos a permissão dele para ficar como executável e adicionamos ele em nosso **/etc/rc.local**:

```
# chmod 700 /root/scripts/frw-nft.sh
# echo "/root/scripts/frw-nft.sh" >> /etc/rc.local
```

Gerando nossas regras de CGNAT:

B . P . F
Brasil Peering Fórum

Após baixarmos nosso script gerador de regras CGNAT **(GRCN)**, coloca **/root/scripts/**. Como estamos trabalhando no modelo **determinístico d**, pegarmos nosso **bloco privado 100.64.0.0/22 (1024 IPs)** e nosso **bloco público 198.18.0.0/27 (32 IPs)** e executarmos em linha de comando:

```
# cd /root/scripts
# ./cgnat-nft.py 0 198.18.0.0/27 100.64.0.0/22 1/32
```

Se digitar apenas **./cgnat-nft.py** será apresentado um help dos parâmetros mas é bem simples o seu uso. No comando acima temos o número **0** como índice. Muito cuidado com o índice, porque ele é muito importante para a performance e para cada novo arquivo gerado, esse índice precisará ser incrementado. O comando acima criará automaticamente o arquivo chamado **cgnat-0-31.conf**, aquele mesmo visto no script base sendo carregado com o **include**. Onde esse **0-31** quer dizer que nesse arquivo os índices vão de **0 a 31**. Se for gerar um novo arquivo com o comando acima, o próximo índice a ser usado seria o 32. Por exemplo:

```
# ./cgnat-nft.py 32 198.18.0.32/27 100.64.4.0/22 1/32
```

Esse comando acima criará novas regras no arquivo **cgnat-32-63.conf**, na sequência inclua esse novo arquivo em **/root/scripts/frw-nft.sh** e execute o **/root/scripts/frw-nft.sh** novamente para carregar as novas regras. No próximo slide daremos uma olhada nas regras geradas nesses arquivos.

Executando o gerador de regras:

```
./cgnat-nft.py 0 198.18.0.0/27 100.64.0.0/22 1/32
```

B.P.F
Brasil Peering Fórum

```
################################################################################################################
    GRCN - Gerador de Regras CGNAT em nftables - Beiriz - v4.0 - 27/07/2020 (25/03/2023)
################################################################################################################

        [ Índice inicial: 0 | público: 198.18.0.0/27 | privado: 100.64.0.0/22 | 2016 portas/IP (1/32)]

 - Índice das regras: 0;
 - Rede pública: 198.18.0.0/27 (32 IPs);
 - Rede privada: 100.64.0.0/22 (1024 IPs);
 - Quantidade de IPs privados por IP público: 32 (32 sub-redes /27);
 - Total de portas públicas: 64512;
 - Portas por IP privado: 2016;
 - Arquivo de destino (conf): 'cgnat-0-31.conf';


Tecle [ENTER]...
```

```
# GRCN - Gerador de Regras CGNAT em nftables - Beiriz - v4.0 - 27/07/2020 (25/03/2023)
# - blocos 100.64.0.0/22 -> 198.18.0.0/27;
# - /0 de IPs privados / IP público;
# - 2016 portas / IP privado;
# ------------------------------------ #INDICE 0 / IP PUBLICO 198.18.0.0
add chain ip nat CGNATOUT_0
flush chain ip nat CGNATOUT_0
add rule ip nat CGNATOUT_0 ip protocol tcp ip saddr 100.64.0.0 counter snat to 198.18.0.0:1024-3039
add rule ip nat CGNATOUT_0 ip protocol udp ip saddr 100.64.0.0 counter snat to 198.18.0.0:1024-3039
add rule ip nat CGNATOUT_0 ip protocol tcp ip saddr 100.64.0.1 counter snat to 198.18.0.0:3040-5055
add rule ip nat CGNATOUT_0 ip protocol udp ip saddr 100.64.0.1 counter snat to 198.18.0.0:3040-5055
add rule ip nat CGNATOUT_0 ip protocol tcp ip saddr 100.64.0.2 counter snat to 198.18.0.0:5056-7071
add rule ip nat CGNATOUT_0 ip protocol udp ip saddr 100.64.0.2 counter snat to 198.18.0.0:5056-7071
add rule ip nat CGNATOUT_0 ip protocol tcp ip saddr 100.64.0.3 counter snat to 198.18.0.0:7072-9087
add rule ip nat CGNATOUT_0 ip protocol udp ip saddr 100.64.0.3 counter snat to 198.18.0.0:7072-9087
add rule ip nat CGNATOUT_0 ip protocol tcp ip saddr 100.64.0.4 counter snat to 198.18.0.0:9088-11103
add rule ip nat CGNATOUT_0 ip protocol udp ip saddr 100.64.0.4 counter snat to 198.18.0.0:9088-11103
add rule ip nat CGNATOUT_0 ip protocol tcp ip saddr 100.64.0.5 counter snat to 198.18.0.0:11104-13119
add rule ip nat CGNATOUT_0 ip protocol udp ip saddr 100.64.0.5 counter snat to 198.18.0.0:11104-13119
add rule ip nat CGNATOUT_0 ip protocol tcp ip saddr 100.64.0.6 counter snat to 198.18.0.0:13120-15135
add rule ip nat CGNATOUT_0 ip protocol udp ip saddr 100.64.0.6 counter snat to 198.18.0.0:13120-15135
add rule ip nat CGNATOUT_0 ip protocol tcp ip saddr 100.64.0.7 counter snat to 198.18.0.0:15136-17151
add rule ip nat CGNATOUT_0 ip protocol udp ip saddr 100.64.0.7 counter snat to 198.18.0.0:15136-17151
add rule ip nat CGNATOUT_0 ip protocol tcp ip saddr 100.64.0.8 counter snat to 198.18.0.0:17152-19167
add rule ip nat CGNATOUT_0 ip protocol udp ip saddr 100.64.0.8 counter snat to 198.18.0.0:17152-19167
add rule ip nat CGNATOUT_0 ip protocol tcp ip saddr 100.64.0.9 counter snat to 198.18.0.0:19168-21183
add rule ip nat CGNATOUT_0 ip protocol udp ip saddr 100.64.0.9 counter snat to 198.18.0.0:19168-21183
add rule ip nat CGNATOUT_0 ip protocol tcp ip saddr 100.64.0.10 counter snat to 198.18.0.0:21184-23199
add rule ip nat CGNATOUT_0 ip protocol udp ip saddr 100.64.0.10 counter snat to 198.18.0.0:21184-23199
add rule ip nat CGNATOUT_0 ip protocol tcp ip saddr 100.64.0.11 counter snat to 198.18.0.0:23200-25215
add rule ip nat CGNATOUT_0 ip protocol udp ip saddr 100.64.0.11 counter snat to 198.18.0.0:23200-25215
add rule ip nat CGNATOUT_0 ip protocol tcp ip saddr 100.64.0.12 counter snat to 198.18.0.0:25216-27231
add rule ip nat CGNATOUT_0 ip protocol udp ip saddr 100.64.0.12 counter snat to 198.18.0.0:25216-27231
add rule ip nat CGNATOUT_0 ip protocol tcp ip saddr 100.64.0.13 counter snat to 198.18.0.0:27232-29247
add rule ip nat CGNATOUT_0 ip protocol udp ip saddr 100.64.0.13 counter snat to 198.18.0.0:27232-29247
add rule ip nat CGNATOUT_0 ip protocol tcp ip saddr 100.64.0.14 counter snat to 198.18.0.0:29248-31263
add rule ip nat CGNATOUT_0 ip protocol udp ip saddr 100.64.0.14 counter snat to 198.18.0.0:29248-31263
add rule ip nat CGNATOUT_0 ip protocol tcp ip saddr 100.64.0.15 counter snat to 198.18.0.0:31264-33279
add rule ip nat CGNATOUT_0 ip protocol udp ip saddr 100.64.0.15 counter snat to 198.18.0.0:31264-33279
add rule ip nat CGNATOUT_0 ip protocol tcp ip saddr 100.64.0.16 counter snat to 198.18.0.0:33280-35295
"cgnat-0-31.conf" 2212L, 223100B
```

B.P.F
Brasil Peering Fórum

```
add rule ip nat CGNATOUT_0 ip protocol udp ip saddr 100.64.0.18 counter snat to 198.18.0.0:37312-39327
add rule ip nat CGNATOUT_0 ip protocol tcp ip saddr 100.64.0.19 counter snat to 198.18.0.0:39328-41343
add rule ip nat CGNATOUT_0 ip protocol udp ip saddr 100.64.0.19 counter snat to 198.18.0.0:39328-41343
add rule ip nat CGNATOUT_0 ip protocol tcp ip saddr 100.64.0.20 counter snat to 198.18.0.0:41344-43359
add rule ip nat CGNATOUT_0 ip protocol udp ip saddr 100.64.0.20 counter snat to 198.18.0.0:41344-43359
add rule ip nat CGNATOUT_0 ip protocol tcp ip saddr 100.64.0.21 counter snat to 198.18.0.0:43360-45375
add rule ip nat CGNATOUT_0 ip protocol udp ip saddr 100.64.0.21 counter snat to 198.18.0.0:43360-45375
add rule ip nat CGNATOUT_0 ip protocol tcp ip saddr 100.64.0.22 counter snat to 198.18.0.0:45376-47391
add rule ip nat CGNATOUT_0 ip protocol udp ip saddr 100.64.0.22 counter snat to 198.18.0.0:45376-47391
add rule ip nat CGNATOUT_0 ip protocol tcp ip saddr 100.64.0.23 counter snat to 198.18.0.0:47392-49407
add rule ip nat CGNATOUT_0 ip protocol udp ip saddr 100.64.0.23 counter snat to 198.18.0.0:47392-49407
add rule ip nat CGNATOUT_0 ip protocol tcp ip saddr 100.64.0.24 counter snat to 198.18.0.0:49408-51423
add rule ip nat CGNATOUT_0 ip protocol udp ip saddr 100.64.0.24 counter snat to 198.18.0.0:49408-51423
add rule ip nat CGNATOUT_0 ip protocol tcp ip saddr 100.64.0.25 counter snat to 198.18.0.0:51424-53439
add rule ip nat CGNATOUT_0 ip protocol udp ip saddr 100.64.0.25 counter snat to 198.18.0.0:51424-53439
add rule ip nat CGNATOUT_0 ip protocol tcp ip saddr 100.64.0.26 counter snat to 198.18.0.0:53440-55455
add rule ip nat CGNATOUT_0 ip protocol udp ip saddr 100.64.0.26 counter snat to 198.18.0.0:53440-55455
add rule ip nat CGNATOUT_0 ip protocol tcp ip saddr 100.64.0.27 counter snat to 198.18.0.0:55456-57471
add rule ip nat CGNATOUT_0 ip protocol udp ip saddr 100.64.0.27 counter snat to 198.18.0.0:55456-57471
add rule ip nat CGNATOUT_0 ip protocol tcp ip saddr 100.64.0.28 counter snat to 198.18.0.0:57472-59487
add rule ip nat CGNATOUT_0 ip protocol udp ip saddr 100.64.0.28 counter snat to 198.18.0.0:57472-59487
add rule ip nat CGNATOUT_0 ip protocol tcp ip saddr 100.64.0.29 counter snat to 198.18.0.0:59488-61503
add rule ip nat CGNATOUT_0 ip protocol udp ip saddr 100.64.0.29 counter snat to 198.18.0.0:59488-61503
add rule ip nat CGNATOUT_0 ip protocol tcp ip saddr 100.64.0.30 counter snat to 198.18.0.0:61504-63519
add rule ip nat CGNATOUT_0 ip protocol udp ip saddr 100.64.0.30 counter snat to 198.18.0.0:61504-63519
add rule ip nat CGNATOUT_0 ip protocol tcp ip saddr 100.64.0.31 counter snat to 198.18.0.0:63520-65535
add rule ip nat CGNATOUT_0 ip protocol udp ip saddr 100.64.0.31 counter snat to 198.18.0.0:63520-65535
add rule ip nat CGNATOUT_0 counter snat to 198.18.0.0
add rule ip nat CGNATOUT ip saddr 100.64.0.0/27 counter jump CGNATOUT_0
# ---------------------------------------- #INDICE 1 / IP PUBLICO 198.18.0.1
add chain ip nat CGNATOUT_1
flush chain ip nat CGNATOUT_1
add rule ip nat CGNATOUT_1 ip protocol tcp ip saddr 100.64.0.32 counter snat to 198.18.0.1:1024-3039
add rule ip nat CGNATOUT_1 ip protocol udp ip saddr 100.64.0.32 counter snat to 198.18.0.1:1024-3039
add rule ip nat CGNATOUT_1 ip protocol tcp ip saddr 100.64.0.33 counter snat to 198.18.0.1:3040-5055
add rule ip nat CGNATOUT_1 ip protocol udp ip saddr 100.64.0.33 counter snat to 198.18.0.1:3040-5055
add rule ip nat CGNATOUT_1 ip protocol tcp ip saddr 100.64.0.34 counter snat to 198.18.0.1:5056-7071
add rule ip nat CGNATOUT_1 ip protocol udp ip saddr 100.64.0.34 counter snat to 198.18.0.1:5056-7071
add rule ip nat CGNATOUT_1 ip protocol tcp ip saddr 100.64.0.35 counter snat to 198.18.0.1:7072-9087
add rule ip nat CGNATOUT_1 ip protocol udp ip saddr 100.64.0.35 counter snat to 198.18.0.1:7072-9087
```

B.P.F
Brasil Peering Fórum

```
table ip nat {
        chain POSTROUTING {
                type nat hook postrouting priority srcnat; policy accept;
                oifname "ens18" ip saddr 10.0.10.172 counter packets 0 bytes 0 snat to 198.18.0.0
                oifname "ens18" counter packets 0 bytes 0 jump CGNATOUT
        }

        chain CGNATOUT {
                ip saddr 100.64.0.0/27 counter packets 0 bytes 0 jump CGNATOUT_0
                ip saddr 100.64.0.32/27 counter packets 0 bytes 0 jump CGNATOUT_1
                ip saddr 100.64.0.64/27 counter packets 0 bytes 0 jump CGNATOUT_2
                ip saddr 100.64.0.96/27 counter packets 0 bytes 0 jump CGNATOUT_3
                ip saddr 100.64.0.128/27 counter packets 0 bytes 0 jump CGNATOUT_4
                ip saddr 100.64.0.160/27 counter packets 0 bytes 0 jump CGNATOUT_5
                ip saddr 100.64.0.192/27 counter packets 0 bytes 0 jump CGNATOUT_6
                ip saddr 100.64.0.224/27 counter packets 0 bytes 0 jump CGNATOUT_7
                ip saddr 100.64.1.0/27 counter packets 0 bytes 0 jump CGNATOUT_8
                ip saddr 100.64.1.32/27 counter packets 0 bytes 0 jump CGNATOUT_9
                ip saddr 100.64.1.64/27 counter packets 0 bytes 0 jump CGNATOUT_10
                ip saddr 100.64.1.96/27 counter packets 0 bytes 0 jump CGNATOUT_11
                ip saddr 100.64.1.128/27 counter packets 0 bytes 0 jump CGNATOUT_12
                ip saddr 100.64.1.160/27 counter packets 0 bytes 0 jump CGNATOUT_13
                ip saddr 100.64.1.192/27 counter packets 0 bytes 0 jump CGNATOUT_14
                ip saddr 100.64.1.224/27 counter packets 0 bytes 0 jump CGNATOUT_15
                ip saddr 100.64.2.0/27 counter packets 0 bytes 0 jump CGNATOUT_16
                ip saddr 100.64.2.32/27 counter packets 0 bytes 0 jump CGNATOUT_17
                ip saddr 100.64.2.64/27 counter packets 0 bytes 0 jump CGNATOUT_18
                ip saddr 100.64.2.96/27 counter packets 0 bytes 0 jump CGNATOUT_19
                ip saddr 100.64.2.128/27 counter packets 0 bytes 0 jump CGNATOUT_20
                ip saddr 100.64.2.160/27 counter packets 0 bytes 0 jump CGNATOUT_21
                ip saddr 100.64.2.192/27 counter packets 0 bytes 0 jump CGNATOUT_22
                ip saddr 100.64.2.224/27 counter packets 0 bytes 0 jump CGNATOUT_23
                ip saddr 100.64.3.0/27 counter packets 0 bytes 0 jump CGNATOUT_24
                ip saddr 100.64.3.32/27 counter packets 0 bytes 0 jump CGNATOUT_25
                ip saddr 100.64.3.64/27 counter packets 0 bytes 0 jump CGNATOUT_26
                ip saddr 100.64.3.96/27 counter packets 0 bytes 0 jump CGNATOUT_27
                ip saddr 100.64.3.128/27 counter packets 0 bytes 0 jump CGNATOUT_28
                ip saddr 100.64.3.160/27 counter packets 0 bytes 0 jump CGNATOUT_29
                ip saddr 100.64.3.192/27 counter packets 0 bytes 0 jump CGNATOUT_30
                ip saddr 100.64.3.224/27 counter packets 0 bytes 0 jump CGNATOUT_31
                ip saddr 100.64.4.0/27 counter packets 0 bytes 0 jump CGNATOUT_32
--Mais--
```

B.P.F
Brasil Peering Fórum

Explicando a função dos índices:

O sistema de avaliação de regras de filtros de pacotes e NAT no GNU/Linux é do tipo **First Match Win**, o que significa que a pesquisa das regras se encerra quando o sistema encontra uma regra que dê match.

O sistema fica muito mais otimizado e performático quando quebramos as regras e separamos em **CHAINS** e é aí que entram os **índices**. Porque as **CHAINS** não podem ter o mesmo nome, senão não haveria separação das regras.

Ao lado vemos por exemplo que quando houver um pacote relacionado com o prefixo de origem **100.64.0.0/27**, este será encaminhado para a chain **CGNATOUT_0**, que é onde estão as regras de CGNAT para esse bloco IP. Desse jeito a checagem para esse prefixo não percorre todas as regras de NAT contidas na memória.

Simulando um acesso do cliente e observando os resultados:

Infelizmente criar um ambiente de laboratório para participantes do treinamento na modalidade de tutorial, ficaria complexo de mensurar e dimensionar. No entanto, para testar as regras, fizemos um ambiente virtual de laboratório usando um **Proxmox** e criando 3 VMs: **CGNAT**, **BNG** e **CLIENTE**.

Do router de testes capturei os pacotes para demonstrar como funciona o **CGNAT** e a identificação do cliente em caso de recebimento de Ofício com os logs para busca.

Veremos ainda nesse tutorial como montar um **Servidor de Logs (SYSLOG + Netflow)** para recebimento dos logs de CGNAT e também para armazenar logs de **Prefix Delegation IPv6** gerados pelo Mikrotik RouterOS 6.x.

No próximo slide teremos o acesso por parte do cliente e a captura dos pacotes somente para uma POC (Proof of Concept), para demonstrarmos que o CGNAT está funcionando e alocando a porta, dentro do range de portas, corretamente para um determinado cliente.

Abaixo temos um exemplo de captura bem simples de pacote mostrando que o IP **198.18.0.0** com **porta origem 6767/TCP** acessou o **200.147.41.220** na porta 443/TCP, um acesso para o site do UOL. Neste caso poderia chegar um Ofício solicitando quem acessou com esses dados:

B . P . F
Brasil Peering Fórum

Data: 20/03/2023 às 08:27:17 UTC-3
IP: 198.18.0.0
Porta origem: 6767

Packets Captured

```
08:27:17.529751 IP 198.18.0.0.6767 > 200.147.41.220.443: tcp 0
08:27:17.530652 IP 198.18.0.0.6767 > 200.147.41.220.443: tcp 85
08:27:17.540885 IP 200.147.41.220.443 > 198.18.0.0.6767: tcp 43
08:27:17.542010 IP 198.18.0.0.6767 > 200.147.41.220.443: tcp 788
08:27:17.558616 IP 200.147.41.220.443 > 198.18.0.0.6767: tcp 1408
08:27:17.558636 IP 200.147.41.220.443 > 198.18.0.0.6767: tcp 13
08:27:17.558747 IP 200.147.41.220.443 > 198.18.0.0.6767: tcp 1408
08:27:17.558768 IP 200.147.41.220.443 > 198.18.0.0.6767: tcp 13
08:27:17.558876 IP 200.147.41.220.443 > 198.18.0.0.6767: tcp 1408
08:27:17.558897 IP 200.147.41.220.443 > 198.18.0.0.6767: tcp 1408
08:27:17.558917 IP 200.147.41.220.443 > 198.18.0.0.6767: tcp 1408
08:27:17.558937 IP 200.147.41.220.443 > 198.18.0.0.6767: tcp 39
08:27:17.558998 IP 200.147.41.220.443 > 198.18.0.0.6767: tcp 1196
08:27:17.559604 IP 198.18.0.0.6767 > 200.147.41.220.443: tcp 0
08:27:17.559738 IP 198.18.0.0.6767 > 200.147.41.220.443: tcp 0
08:27:17.559756 IP 198.18.0.0.6767 > 200.147.41.220.443: tcp 0
08:27:20.635644 IP 198.18.0.0.6767 > 200.147.41.220.443: tcp 0
08:27:20.645147 IP 200.147.41.220.443 > 198.18.0.0.6767: tcp 0
08:27:20.645922 IP 198.18.0.0.6767 > 200.147.41.220.443: tcp 0
```

Se olharmos os dados do log do Ofício e procurarmos pelo IP **198.18.0.0** e **porta 6767** no nosso arquivo de configuração do CGNAT, acharemos o IP **100.64.0.2** que utiliza o range de portas entre **5056** e **7071**. Na sequência só fazermos a busca no servidor radius, por exemplo, para achar o login pppoe do cliente que usou o IP **100.64.0.2** em **20/03/2023 às 08:27:17 UTC-3**.

B.P.F
Brasil Peering Fórum

```
# GRCN - Gerador de Regras CGNAT em nftables - Beiriz - v4.001 - 27/07/2020 (31/03/2023)
# - blocos 100.64.0.0/22 -> 198.18.0.0/27;
# - /0 de IPs privados / IP público;
# - 2016 portas / IP privado;
# ------------------------------------- #INDICE 0 / IP PUBLICO 198.18.0.0
add chain ip nat CGNATOUT_0
flush chain ip nat CGNATOUT_0
add rule ip nat CGNATOUT_0 ip protocol tcp ip saddr 100.64.0.0 counter snat to 198.18.0.0:1024-3039
add rule ip nat CGNATOUT_0 ip protocol udp ip saddr 100.64.0.0 counter snat to 198.18.0.0:1024-3039
add rule ip nat CGNATOUT_0 ip protocol tcp ip saddr 100.64.0.1 counter snat to 198.18.0.0:3040-5055
add rule ip nat CGNATOUT_0 ip protocol udp ip saddr 100.64.0.1 counter snat to 198.18.0.0:3040-5055
add rule ip nat CGNATOUT_0 ip protocol tcp ip saddr 100.64.0.2 counter snat to 198.18.0.0:5056-7071
add rule ip nat CGNATOUT_0 ip protocol udp ip saddr 100.64.0.2 counter snat to 198.18.0.0:5056-7071
add rule ip nat CGNATOUT_0 ip protocol tcp ip saddr 100.64.0.3 counter snat to 198.18.0.0:7072-9087
add rule ip nat CGNATOUT_0 ip protocol udp ip saddr 100.64.0.3 counter snat to 198.18.0.0:7072-9087
add rule ip nat CGNATOUT_0 ip protocol tcp ip saddr 100.64.0.4 counter snat to 198.18.0.0:9088-11103
add rule ip nat CGNATOUT_0 ip protocol udp ip saddr 100.64.0.4 counter snat to 198.18.0.0:9088-11103
add rule ip nat CGNATOUT_0 ip protocol tcp ip saddr 100.64.0.5 counter snat to 198.18.0.0:11104-13119
add rule ip nat CGNATOUT_0 ip protocol udp ip saddr 100.64.0.5 counter snat to 198.18.0.0:11104-13119
add rule ip nat CGNATOUT_0 ip protocol tcp ip saddr 100.64.0.6 counter snat to 198.18.0.0:13120-15135
add rule ip nat CGNATOUT_0 ip protocol udp ip saddr 100.64.0.6 counter snat to 198.18.0.0:13120-15135
add rule ip nat CGNATOUT_0 ip protocol tcp ip saddr 100.64.0.7 counter snat to 198.18.0.0:15136-17151
add rule ip nat CGNATOUT_0 ip protocol udp ip saddr 100.64.0.7 counter snat to 198.18.0.0:15136-17151
add rule ip nat CGNATOUT_0 ip protocol tcp ip saddr 100.64.0.8 counter snat to 198.18.0.0:17152-19167
add rule ip nat CGNATOUT_0 ip protocol udp ip saddr 100.64.0.8 counter snat to 198.18.0.0:17152-19167
add rule ip nat CGNATOUT_0 ip protocol tcp ip saddr 100.64.0.9 counter snat to 198.18.0.0:19168-21183
add rule ip nat CGNATOUT_0 ip protocol udp ip saddr 100.64.0.9 counter snat to 198.18.0.0:19168-21183
add rule ip nat CGNATOUT_0 ip protocol tcp ip saddr 100.64.0.10 counter snat to 198.18.0.0:21184-23199
add rule ip nat CGNATOUT_0 ip protocol udp ip saddr 100.64.0.10 counter snat to 198.18.0.0:21184-23199
add rule ip nat CGNATOUT_0 ip protocol tcp ip saddr 100.64.0.11 counter snat to 198.18.0.0:23200-25215
add rule ip nat CGNATOUT_0 ip protocol udp ip saddr 100.64.0.11 counter snat to 198.18.0.0:23200-25215
```

198.18.0.0 porta origem 6767

Monitorando o tráfego Mbps/PPS com a ferramenta **bmon**. Para instalar o software no Debian basta fazer:

```
# apt install bmon
```

Para monitorar as interfaces faríamos algo assim onde **-b** para **bits/s** e o **-p** para **selecionar as interfaces** que quer monitorar. Para monitorar nosso **bond0** e **bond1**. A imagem abaixo é apenas um exemplo.

```
# bmon -b -p bond0,bond1
```

B.P.F
Brasil Peering Fórum

CGNAT Determinístico usando Mikrotik RouterOS

Utilizando Mikrotik RouterOS 6.x como caixa CGNAT:

B.P.F
Brasil Peering Fórum

Uma boa opção para caixa CGNAT com custo acessível seria uma **CCR1036-8G-2S+** onde se for configurada somente para fazer CGNAT, com o mínimo de regras de filtro e **Fasttrack** habilitado, já alcancei **13 Gbps de tráfego** ou **26 Gbps agregado** fazendo um **bonding** com as 2 interfaces ópticas de 10Gbps.

Essa imagem abaixo foi retirada do datasheet da CCR1036-8G-2S+:

Ethernet test results

CCR1036-8G-2S+r2		Tile 36 core max possible throughput test					
Mode	Configuration	1518 byte		512 byte		64 byte	
		kpps	Mbps	kpps	Mbps	kpps	Mbps
Bridging	none (fast path)	2275.7	27636.1	6578.9	26947.2	41666.7	21333.4
Bridging	25 bridge filter rules	2275.7	27636.1	5179.7	21216.1	5163.5	2643.7
Routing	none (fast path)	2275.7	27636.1	6578.9	26947.2	41666.7	21333.4
Routing	25 simple queues	2275.7	27636.1	6553.3	26842.3	7643.1	3913.3
Routing	25 ip filter rules	1825.7	22171.3	3033.9	12426.9	3049.7	1561.4

1. All tests are done with Xena Networks specialized test equipment (XenaBay),and done according to RFC2544 (Xena2544)
2. Max throughput is determined with 30+ second attempts with 0,1% packet loss tolerance in 64, 512, 1518 byte packet sizes
3. Test results show device maximum performance, and are reached using mentioned hardware and software configuration, different configurations most likely will result in lower results

Configurando o sistema:

B . P . F
Brasil Peering Fórum

Instale um Mikrotik RouterOS do zero, procure versão mais estável possível. Como não utilizei ainda em produção o RouterOS 7.x, sugiro utilizar a versão **6.48.6 Long-term**, que até o momento, é a versão considerada mais estável.

O processo de configurar um **CGNAT Determinístico** no **Mikrotik RouterOS** será bem mais simples que no **Debian GNU/Linux** mas a capacidade alcançada com o GNU/Linux será bem superior ao visto aqui.

Sobre Fasttrack:

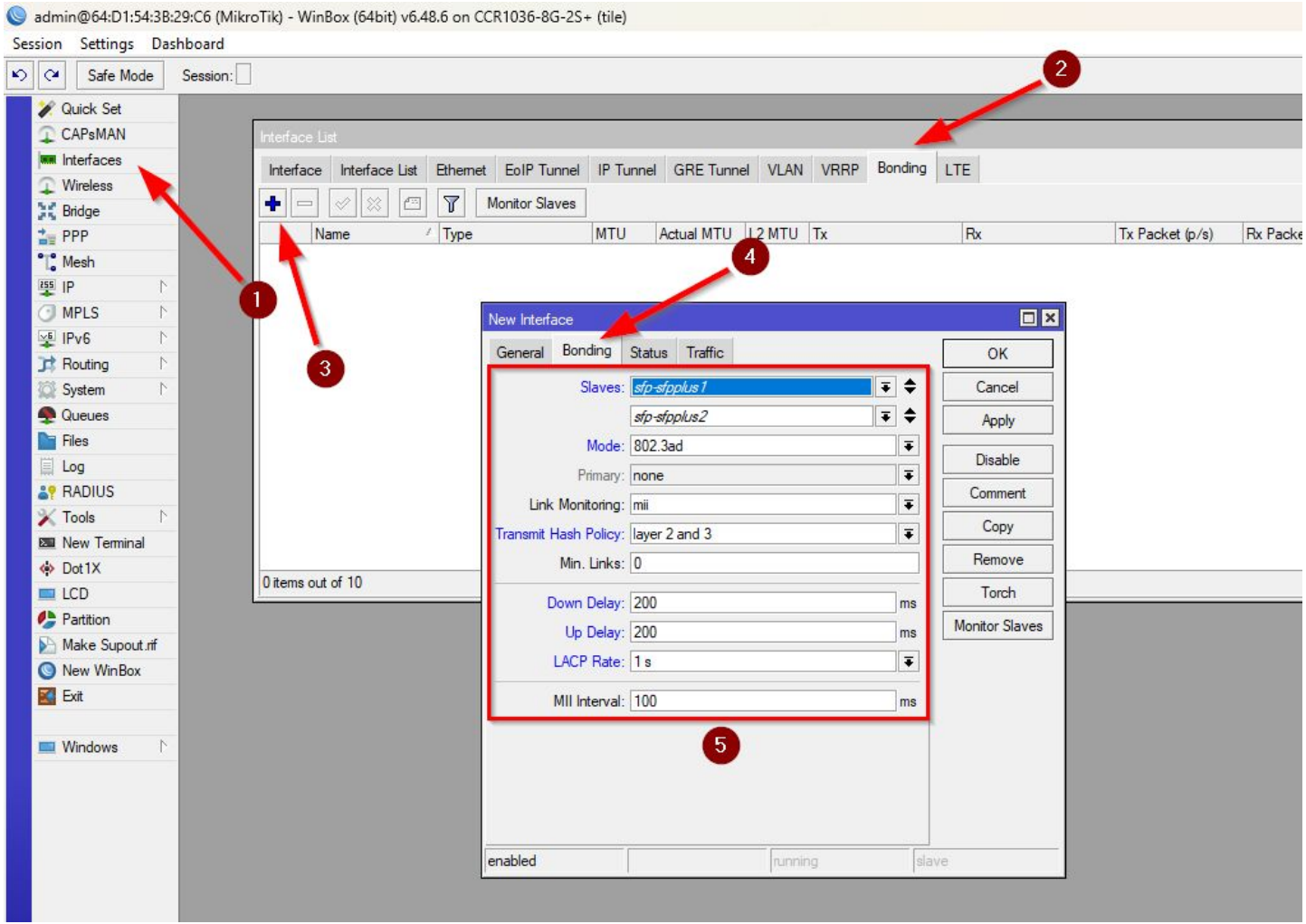
O **Fasttrack** é um recurso muito importante que aumentará a performance da sua caixa CGNAT, acelerando o encaminhamento de pacotes e diminuindo o consumo de CPU. Neste momento não faremos isso. Quando chegarmos no processo de criação das regras de CGNAT, ele será habilitado e mostraremos quais as regras que fazem isso.

Configurando o bonding:

Como usaremos as duas portas de 10GbE sfp+ da CCR, utilizaremos vlans para separar a rede que se comunicará com a Internet, da rede com o BNG. No próximo slide veremos como deixar o nosso bonding.

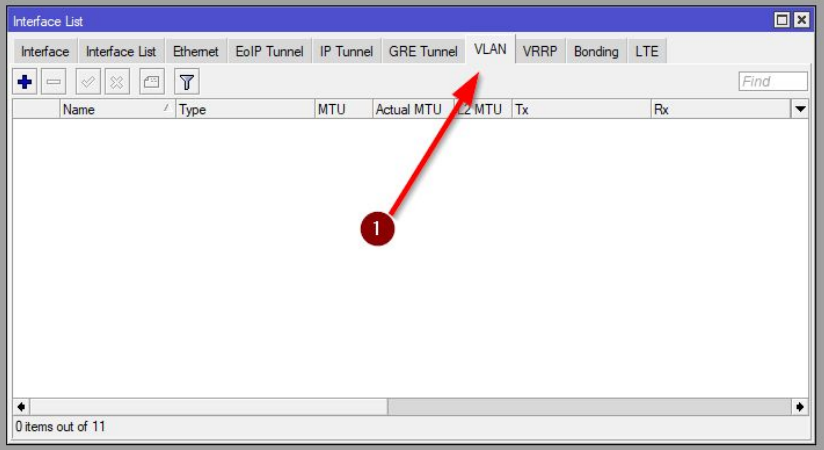
Na sequência configuramos nossas vlans de entrada e saída e em cima delas os IPs do diagrama, como fizemos com o Debian.

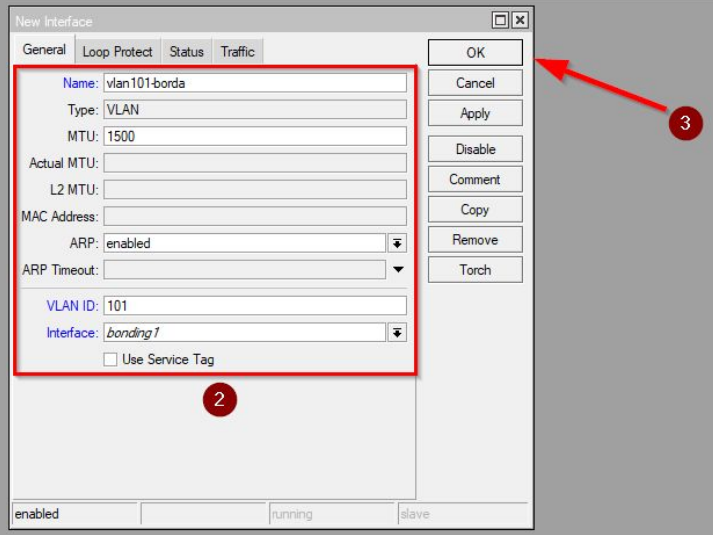
Vamos definir a **vlan 101** para a interface que fará a comunicação com a Internet e por onde será feito o **CGNAT** e a **vlan 102** que fará a comunicação com o **BNG**.

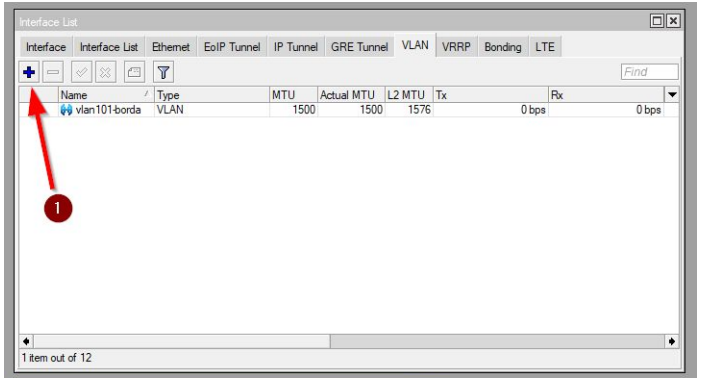


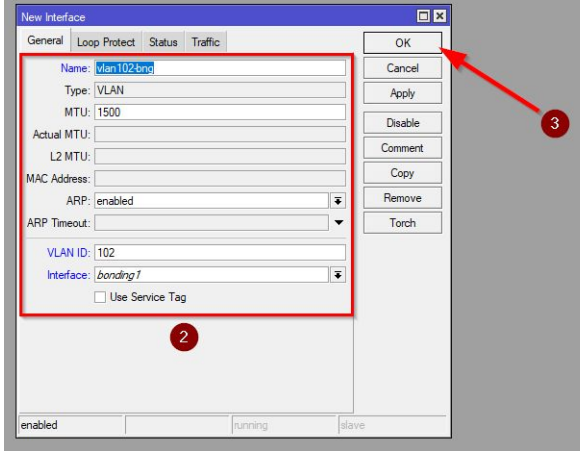
B.P.F
Brasil Peering Fórum

B . P . F
Brasil Peering Fórum









Configurando os IPs e rotas:

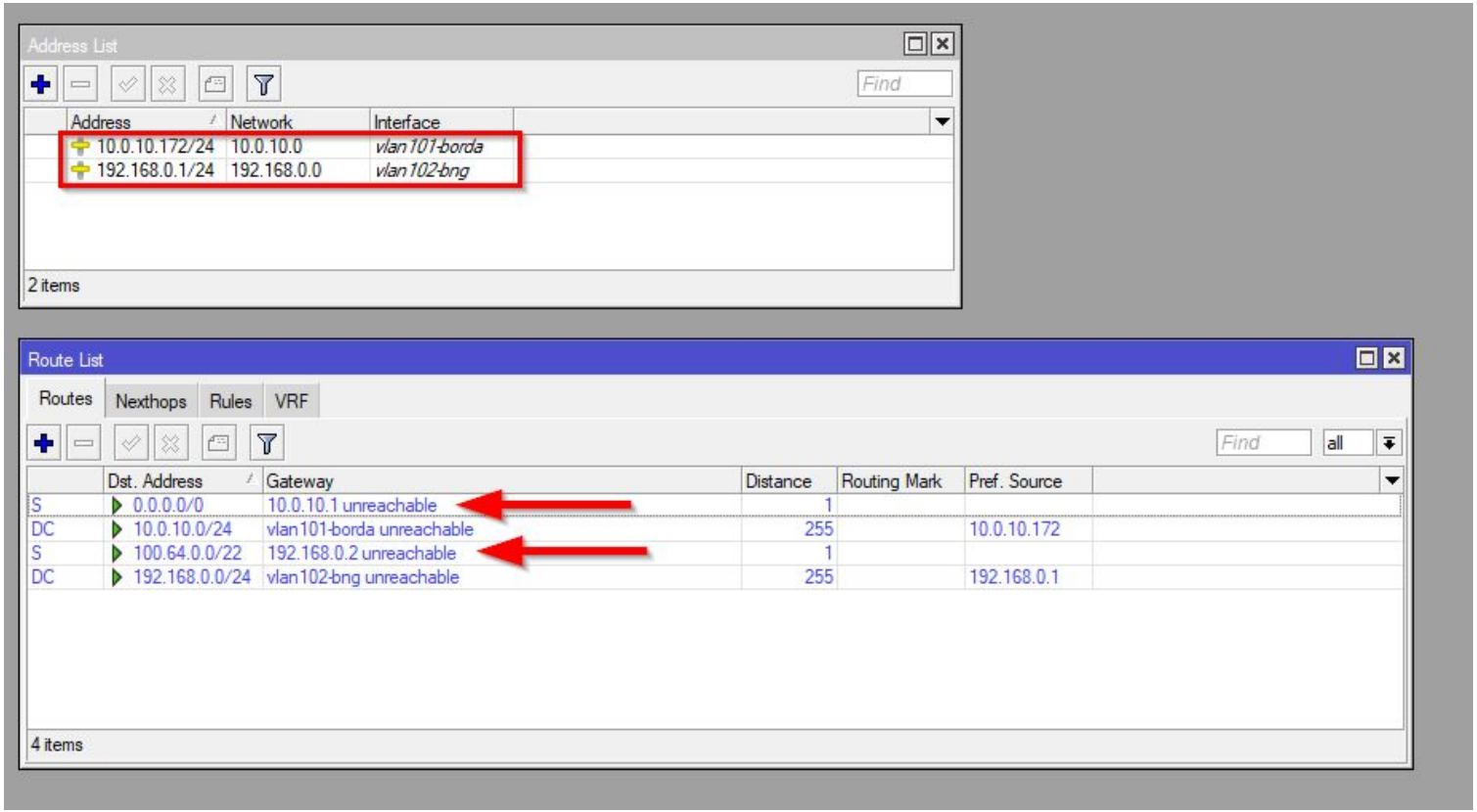
B . P . F

Brasil Peering Fórum

O objetivo deste tutorial é ser bem simples para entendermos os conceitos e por isso estamos utilizando rotas estáticas e não estamos envolvendo outros protocolos como o OSPF. Nada impediria de utilizar a mesma técnica apresentada aqui em um cenário com OSPF, por exemplo.

No próximo slide veremos que na **vlan-101-borda** configuramos o **IP 10.0.10.172/24** e na **vlan-102-bng** configuramos o **IP 192.168.0.1/24**.

Como rotas criamos uma **default route** apontando para o **IP 10.0.10.1**, criamos uma rota para **100.64.0.0/22** com **next-hop 192.168.0.2** e para nos protegermos de **static loop** teremos nossas rotas de **blackhole** quando formos gerar as regras de CGNAT.


Address List
Find
Address
Network
Interface
10.0.10.172/24
10.0.10.0
vlan101-borda
192.168.0.1/24
192.168.0.0
vlan102-bng
2 items
Route List
Routes
Nexthops
Rules
VRF
Find
all
Dst. Address
Gateway
Distance
Routing Mark
Pref. Source
S
0.0.0.0/0
10.0.10.1 unreachable
1
DC
10.0.10.0/24
vlan101-borda unreachable
255
10.0.10.172
S
100.64.0.0/22
192.168.0.2 unreachable
1
DC
192.168.0.0/24
vlan102-bng unreachable
255
192.168.0.1
4 items

Recomendações de segurança:

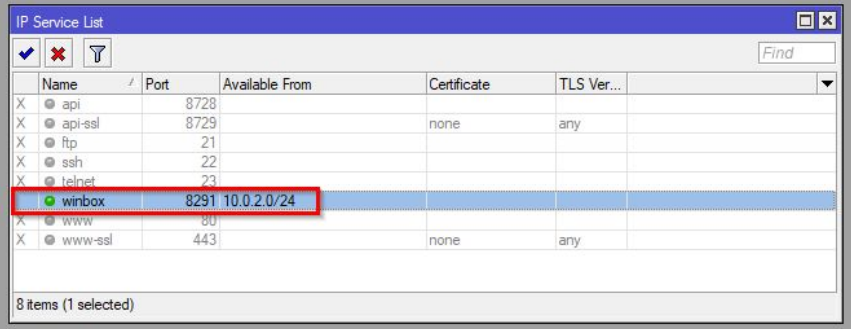
B . P . F
Brasil Peering Fórum

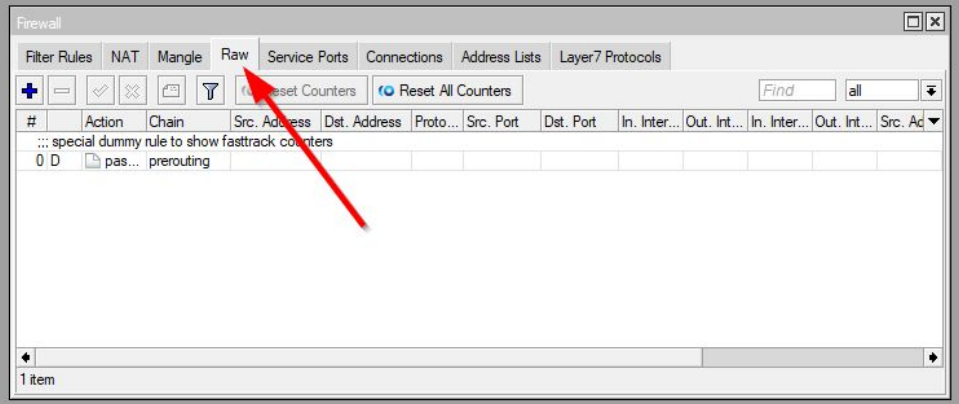
- Utilize credenciais de acesso com senhas fortes, não esqueça o **login admin sem senha** (padrão no Mikrotik RouterOS).

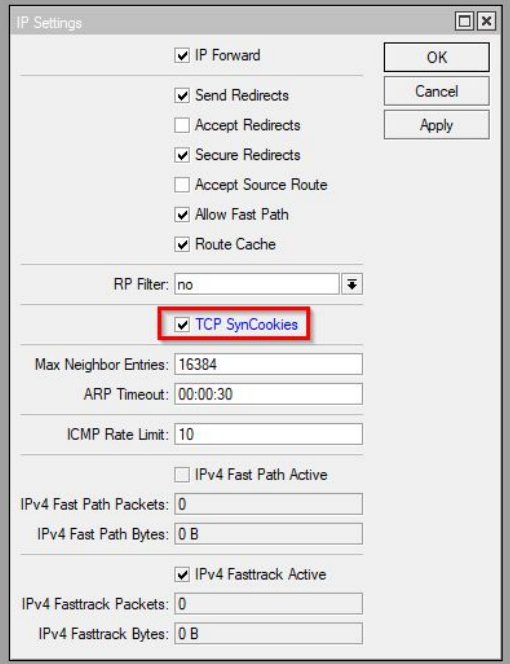
- Desabilite todos os serviços que não for utilizar e os que ficarem abertos, especifique neles o acesso apenas da sua rede de gerência. Não deixe qualquer serviço aberto para a Internet.

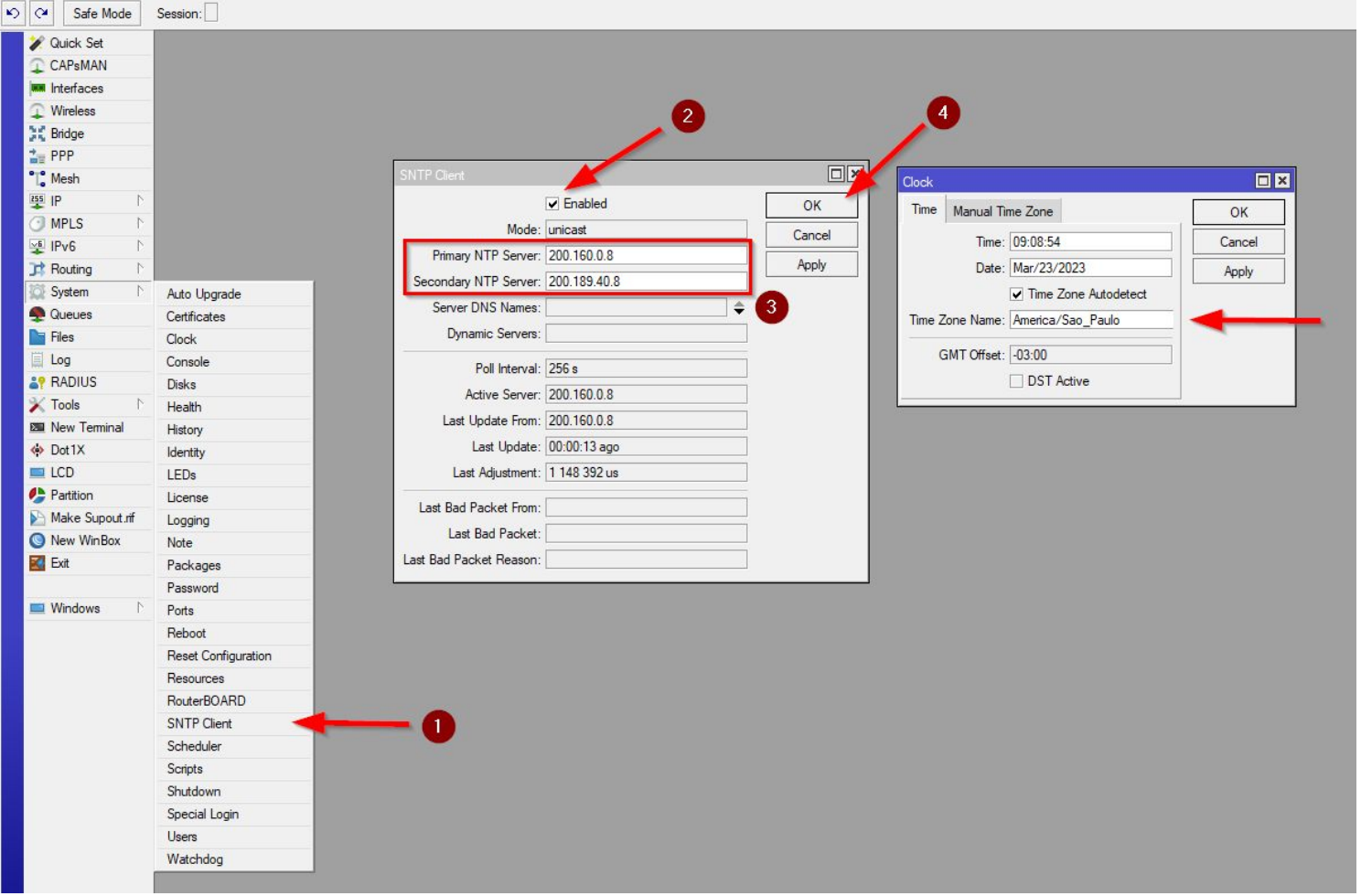
- Habilite o **TCP SynCookies**.

- Procure criar suas regras de filtros de pacotes sempre na **Table Raw**, ela não agride tanto a performance do equipamento mas necessita de muita atenção porque ela pode afetar os acessos dos assinantes. Isso porque uma regra genérica demais será analisada tanto com destino a caixa, quanto destino ao cliente e o mesmo pode ocorrer no sentido inverso, do cliente para a Internet.


IP Service List
Find
Name
Port
Available From
Certificate
TLS Ver...
api 8728
api-ssl 8729 none any
ftp 21
ssh 22
telnet 23
winbox 8291 10.0.2.0/24
www 80
www-ssl 443 none any
8 items (1 selected)


Firewall
Filter Rules
NAT
Mangle
Raw
Service Ports
Connections
Address Lists
Layer7 Protocols
Reset Counters
Reset All Counters
Find
all
Action Chain Src. Address Dst. Address Proto... Src. Port Dst. Port In. Inter... Out. Int... In. Inter... Out. Int... Src. Ad
::: special dummy rule to show fasttrack counters
0 D pas... prerouting
1 item


IP Settings
IP Forward
Send Redirects
Accept Redirects
Secure Redirects
Accept Source Route
Allow Fast Path
Route Cache
RP Filter: no
TCP SynCookies
Max Neighbor Entries: 16384
ARP Timeout: 00:00:30
ICMP Rate Limit: 10
IPv4 Fast Path Active
IPv4 Fast Path Packets: 0
IPv4 Fast Path Bytes: 0 B
IPv4 Fasttrack Active
IPv4 Fasttrack Packets: 0
IPv4 Fasttrack Bytes: 0 B
OK
Cancel
Apply



B.P.F
Brasil Peering Fórum

Configure o NTP client da caixa e mantenha a data e horário adequados.

Criando as regras de CGNAT:

B . P . F
Brasil Peering Fórum

Para simplificar nossa vida, **Rudimar Remontti** criou em seu blog, um sistema para gerar regras de CGNAT Determinístico de forma simples e performática, utilizando regras **netmap** da Mikrotik. Para tanto o link é este:
https://cgnat.remontti.com.br/

O sistema é bem completo, simples, irá gerar as regras de CGNAT e nossas blackholes para bloqueio de **static loop**. Também no final teremos uma tabela de associação que devemos guardar para fazer as quebras de sigilo solicitadas nos Ofícios Judiciais.

Ao acessar o site e seguindo o nosso diagrama completaremos as informações conforme mostrado no próximo slide.

B . P . F
Brasil Peering Fórum

Gerador de CGNAT para RouterOS

Networks inicial privado: 100.64.0.0

Prefixo público (Network): 198.18.0.0/27

Lembram dos índices que fizemos no Debian?

1 Público para quantos Privados: 32 Clientes [~2000 Portas]

Sequencial Chain(s): 0

Ignorar destino(s): Nenhum

Ignorar Prefixo/Lista: LISTA_SERVIDORES

No Track (RAW): Sim

Interface Uplink: Nome da Interface (Recomendado)

Nome/Lista da interface de Uplink: vlan101-borda

Uso de Blackhole: Sim - Prefixo (Recomendado)

Nome/Lista da interface de Uplink: Loopback

Protocolos

- (•) TCP/UDP (Recomendado Judicialmente)
- ( ) Apenas TCP (Problemas Judicial*)

Fasttrack

- (•) Sim (Recomendado)
- ( ) Não

RouterOS

- (•) Versão 6.x
- ( ) Versão 7.x

Gerar Script

RAR rudimar@remontti

Regras geradas: 49146

Se está feliz em ter encontrado uma solução fácil e rápida para gerar as regras CGNAT, lhe poupando horas de trabalho, por favor, considere fazer uma doação para apoiar o desenvolvimento e me ajudar a manter esta ferramenta acessível e ***gratuita para todos****.*
Rudimar Remontti

Quero ajudar

Contato

RANGE IPs PRIVADOS PARA CGNAT **100.64.0.0-100.64.3.255** | TOTAL DE IPS **1024** | PORTAS POR CLIENTE: **2016**

AO FINAL DAS REGRAS SE ENCONTRA O MAPEAMENTO DAS PORTAS.

*O **motivo** de ser tudo impresso na tela **não tendo opções para download** é que todos os dados preenchidos nunca serão armazenado. Ética!*

```
# BLACKHOLE
/ip route add type=blackhole dst-address=198.18.0.0/27 comment=CGNAT_BLACKHOLE
```

Nossa blackhole contra static loop

```
# FASTTRACK
/ip firewall filter add chain=forward action=fasttrack-connection connection-state=established,related
/ip firewall filter add chain=forward action=accept connection-state=established,related
```

Instruções para habilitar o Fasttrack

```
#CGNAT
/ip firewall nat add chain=srcnat src-address=100.64.0.0/24 out-interface="vlan101-borda"  action=jump jump-target=CGNAT_100_64_0
/ip firewall nat add chain=srcnat src-address=100.64.1.0/24 out-interface="vlan101-borda"  action=jump jump-target=CGNAT_100_64_1
/ip firewall nat add chain=srcnat src-address=100.64.2.0/24 out-interface="vlan101-borda"  action=jump jump-target=CGNAT_100_64_2
/ip firewall nat add chain=srcnat src-address=100.64.3.0/24 out-interface="vlan101-borda"  action=jump jump-target=CGNAT_100_64_3

/ip firewall nat add chain=CGNAT_100_64_0 src-address=100.64.0.0/27 action=jump jump-target=CGNAT_0
/ip firewall nat add chain=CGNAT_100_64_0 src-address=100.64.0.32/27 action=jump jump-target=CGNAT_1
/ip firewall nat add chain=CGNAT_100_64_0 src-address=100.64.0.64/27 action=jump jump-target=CGNAT_2
/ip firewall nat add chain=CGNAT_100_64_0 src-address=100.64.0.96/27 action=jump jump-target=CGNAT_3
/ip firewall nat add chain=CGNAT_100_64_0 src-address=100.64.0.128/27 action=jump jump-target=CGNAT_4
/ip firewall nat add chain=CGNAT_100_64_0 src-address=100.64.0.160/27 action=jump jump-target=CGNAT_5
/ip firewall nat add chain=CGNAT_100_64_0 src-address=100.64.0.192/27 action=jump jump-target=CGNAT_6
/ip firewall nat add chain=CGNAT_100_64_0 src-address=100.64.0.224/27 action=jump jump-target=CGNAT_7
/ip firewall nat add chain=CGNAT_100_64_1 src-address=100.64.1.0/27 action=jump jump-target=CGNAT_8
/ip firewall nat add chain=CGNAT_100_64_1 src-address=100.64.1.32/27 action=jump jump-target=CGNAT_9
/ip firewall nat add chain=CGNAT_100_64_1 src-address=100.64.1.64/27 action=jump jump-target=CGNAT_10
/ip firewall nat add chain=CGNAT_100_64_1 src-address=100.64.1.96/27 action=jump jump-target=CGNAT_11
/ip firewall nat add chain=CGNAT_100_64_1 src-address=100.64.1.128/27 action=jump jump-target=CGNAT_12
/ip firewall nat add chain=CGNAT_100_64_1 src-address=100.64.1.160/27 action=jump jump-target=CGNAT_13
/ip firewall nat add chain=CGNAT_100_64_1 src-address=100.64.1.192/27 action=jump jump-target=CGNAT_14
/ip firewall nat add chain=CGNAT_100_64_1 src-address=100.64.1.224/27 action=jump jump-target=CGNAT_15
/ip firewall nat add chain=CGNAT_100_64_2 src-address=100.64.2.0/27 action=jump jump-target=CGNAT_16
/ip firewall nat add chain=CGNAT_100_64_2 src-address=100.64.2.32/27 action=jump jump-target=CGNAT_17
/ip firewall nat add chain=CGNAT_100_64_2 src-address=100.64.2.64/27 action=jump jump-target=CGNAT_18
/ip firewall nat add chain=CGNAT_100_64_2 src-address=100.64.2.96/27 action=jump jump-target=CGNAT_19
/ip firewall nat add chain=CGNAT_100_64_2 src-address=100.64.2.128/27 action=jump jump-target=CGNAT_20
/ip firewall nat add chain=CGNAT_100_64_2 src-address=100.64.2.160/27 action=jump jump-target=CGNAT_21
/ip firewall nat add chain=CGNAT_100_64_2 src-address=100.64.2.192/27 action=jump jump-target=CGNAT_22
/ip firewall nat add chain=CGNAT_100_64_2 src-address=100.64.2.224/27 action=jump jump-target=CGNAT_23
/ip firewall nat add chain=CGNAT_100_64_3 src-address=100.64.3.0/27 action=jump jump-target=CGNAT_24
/ip firewall nat add chain=CGNAT_100_64_3 src-address=100.64.3.32/27 action=jump jump-target=CGNAT_25
/ip firewall nat add chain=CGNAT_100_64_3 src-address=100.64.3.64/27 action=jump jump-target=CGNAT_26
/ip firewall nat add chain=CGNAT_100_64_3 src-address=100.64.3.96/27 action=jump jump-target=CGNAT_27
/ip firewall nat add chain=CGNAT_100_64_3 src-address=100.64.3.128/27 action=jump jump-target=CGNAT_28
```

Nossas regras de CGNAT propriamente ditas.

O site irá gerar automaticamente os comandos ao lado de onde faremos uma cópia e executaremos no nosso equipamento Mikrotik RouterOS.

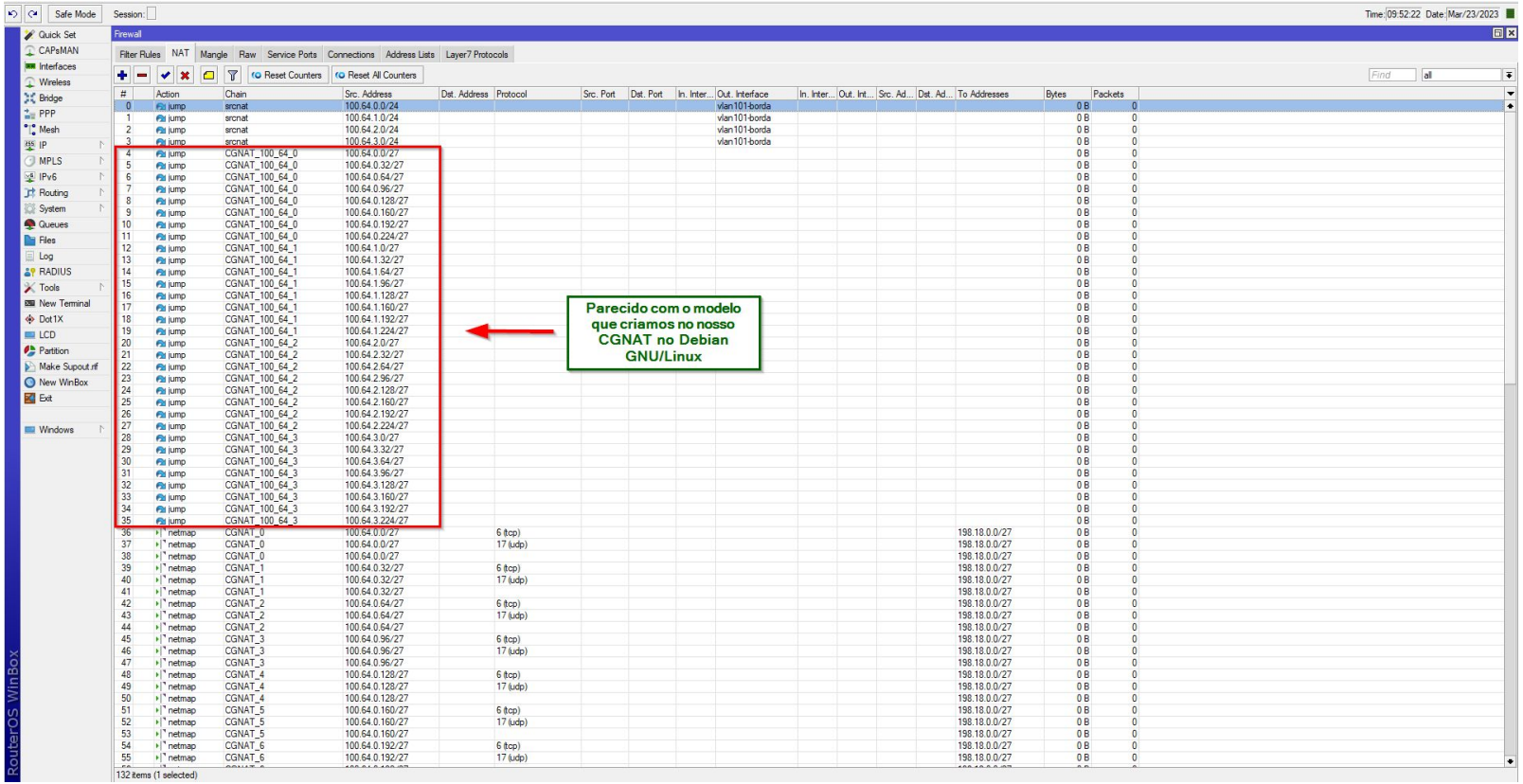
No final da página é gerado uma tabela do mapeamento das portas, isso deve ser salvo como documento importante pois será usado para quebra de sigilo tecnológico.

B.P.F
Brasil Peering Fórum

MAPEAMENTO DAS PORTAS

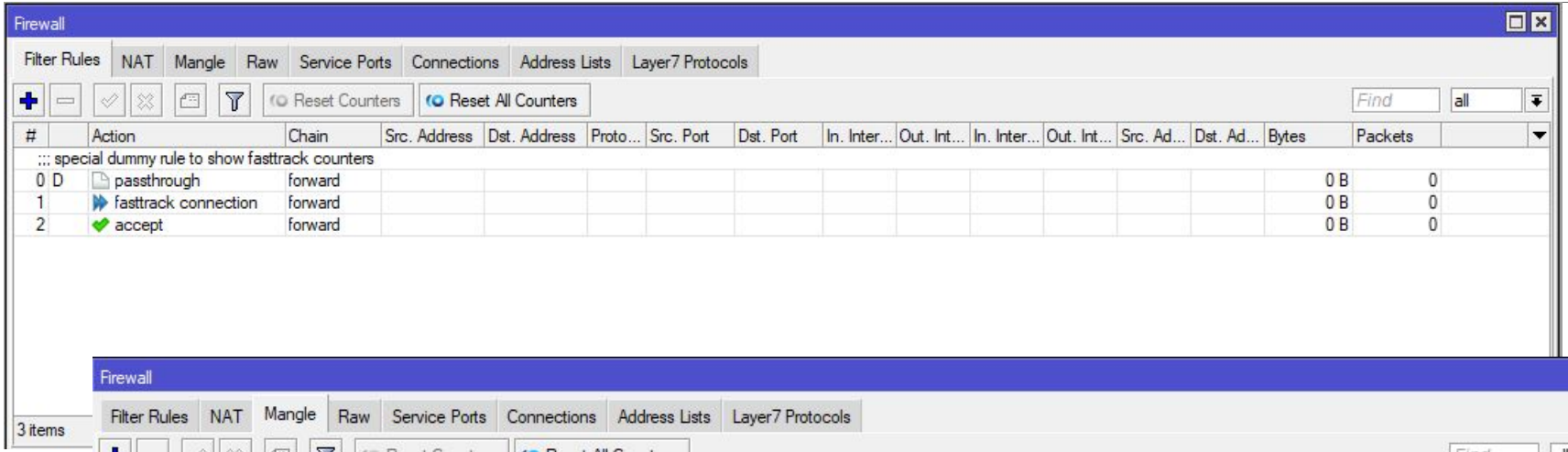
IP Público	Range de Portas	IP Privado
198.18.0.0	1024 à 3040	100.64.0.0
198.18.0.1	1024 à 3040	100.64.0.1
198.18.0.2	1024 à 3040	100.64.0.2
198.18.0.3	1024 à 3040	100.64.0.3
198.18.0.4	1024 à 3040	100.64.0.4
198.18.0.5	1024 à 3040	100.64.0.5
198.18.0.6	1024 à 3040	100.64.0.6
198.18.0.7	1024 à 3040	100.64.0.7
198.18.0.8	1024 à 3040	100.64.0.8
198.18.0.9	1024 à 3040	100.64.0.9
198.18.0.10	1024 à 3040	100.64.0.10
198.18.0.11	1024 à 3040	100.64.0.11
198.18.0.12	1024 à 3040	100.64.0.12
198.18.0.13	1024 à 3040	100.64.0.13
198.18.0.14	1024 à 3040	100.64.0.14
198.18.0.15	1024 à 3040	100.64.0.15
198.18.0.16	1024 à 3040	100.64.0.16
198.18.0.17	1024 à 3040	100.64.0.17
198.18.0.18	1024 à 3040	100.64.0.18
198.18.0.19	1024 à 3040	100.64.0.19
198.18.0.20	1024 à 3040	100.64.0.20
198.18.0.21	1024 à 3040	100.64.0.21
198.18.0.22	1024 à 3040	100.64.0.22
198.18.0.23	1024 à 3040	100.64.0.23
198.18.0.24	1024 à 3040	100.64.0.24
198.18.0.25	1024 à 3040	100.64.0.25
198.18.0.26	1024 à 3040	100.64.0.26
198.18.0.27	1024 à 3040	100.64.0.27
198.18.0.28	1024 à 3040	100.64.0.28
198.18.0.29	1024 à 3040	100.64.0.29
198.18.0.30	1024 à 3040	100.64.0.30
198.18.0.31	1024 à 3040	100.64.0.31
198.18.0.0	3041 à 5056	100.64.0.32
198.18.0.1	3041 à 5056	100.64.0.33
198.18.0.2	3041 à 5056	100.64.0.34

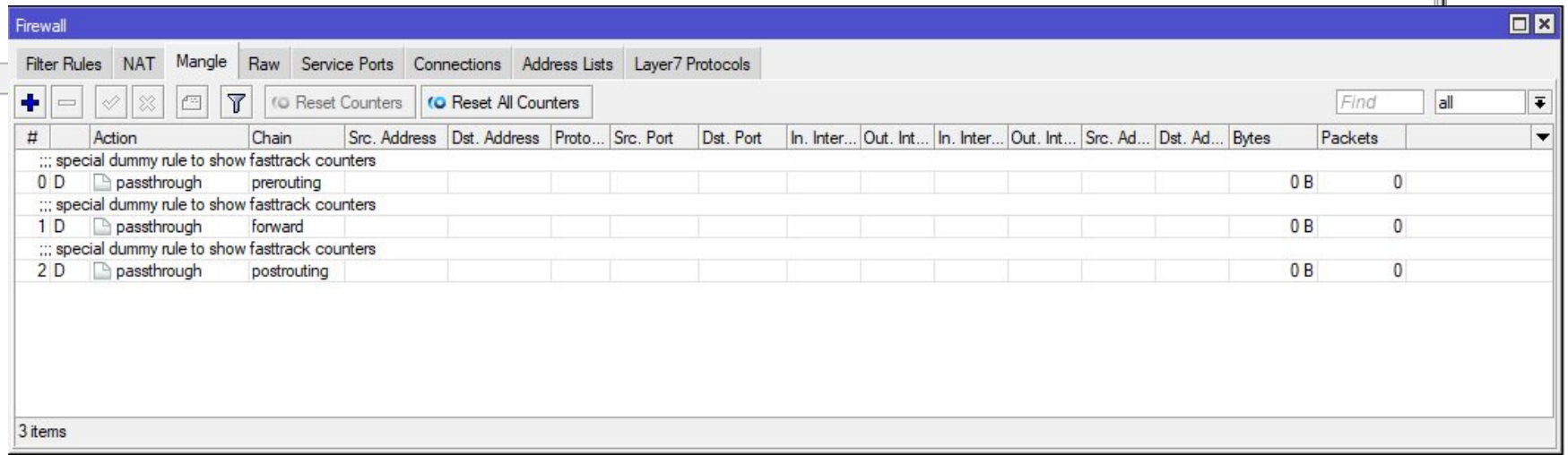
O conceito é o mesmo, quebrar as regras em blocos menores para chegarmos no nosso **First Match Win** mais rápido e não termos que percorrer todas as regras em memória.



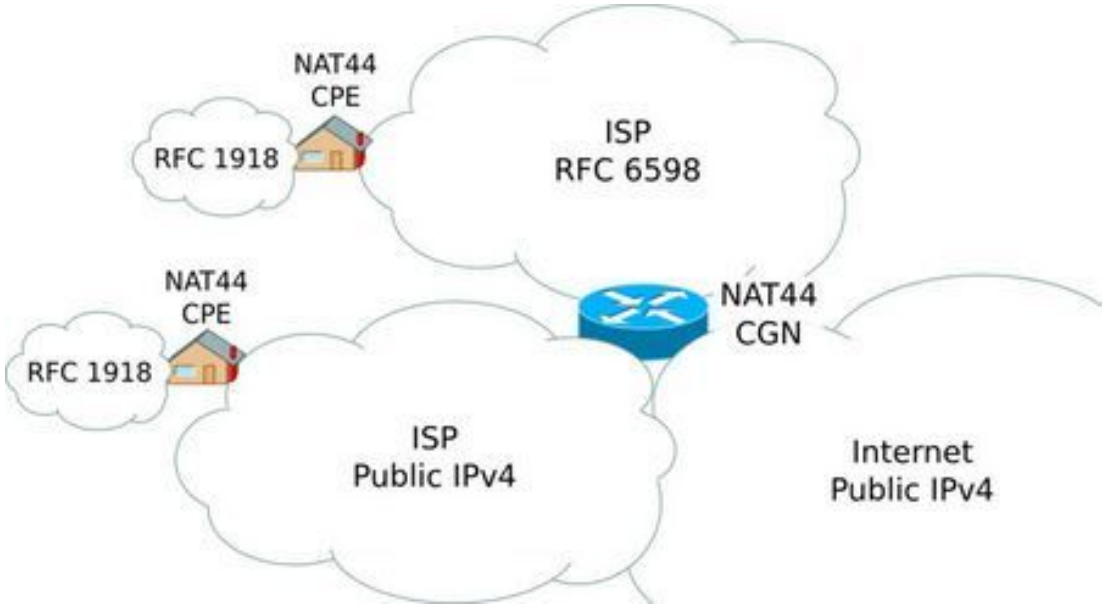
Abaixo como ficaram as regras que habilita o **Fasttrack** no nosso equipamento, aumentando em muito a performance de encaminhamento dos pacotes.

B . P . F
Brasil Peering Fórum





Servidor de Logs CGNAT SYSLOG+NETFLOW



Introdução:

Todo sistema de CGNAT necessita que tenhamos uma maneira de identificar um cliente quando for requisitado pela Justiça, logo precisamos de recursos para tal. Quando utilizamos CGNAT Determinístico vimos que através de tabelas tanto no GNU/Linux com NFTables ou IPTables e nos Mikrotiks RouterOS, é bem fácil identificar os clientes sem precisar de logs de acesso.

O problema começa quando utilizamos **CGNAT BPA (Bulk Port Allocation)** e existem configurações que podem te gerar uma quantidade enorme de logs diários e outras configurações que podem te gerar logs muito menores.

Um exemplo de configuração que gera uma quantidade excessiva de logs é quando o sistema envia logs para cada nova conexão aberta. Nessa modalidade, cada acesso que o cliente faz, gera um log de IP público de origem e porta de origem.

A outra configuração, que é a mais recomendada, definimos blocos de portas e dizemos ao sistema para ir alocando para o cliente conforme a necessidade dele. Um exemplo de configuração que costumo recomendar nesse caso seria de:

- Blocos de **256 portas**, com até **16 blocos** por cliente dando um total de até **4096 portas**.

Se formos analisar os dados trafegados, sem identificar o cliente, vamos perceber que a maioria dos clientes utilizam poucas portas e ficam dentro do primeiro bloco de 256 portas, outros necessitarão de mais e **alguns heavy** users podem chegar perto das 4096 portas. Com essa abordagem podemos economizar muito recurso IP e chegar em casos de **relação 1/100 ou mais**.

Ainda utilizando essa abordagem de logs por alocação de blocos de portas, teremos logs muito menores para armazenar. Isso porque só será registrado quando um cliente alocar o bloco de porta e não por cada porta que utilizar.

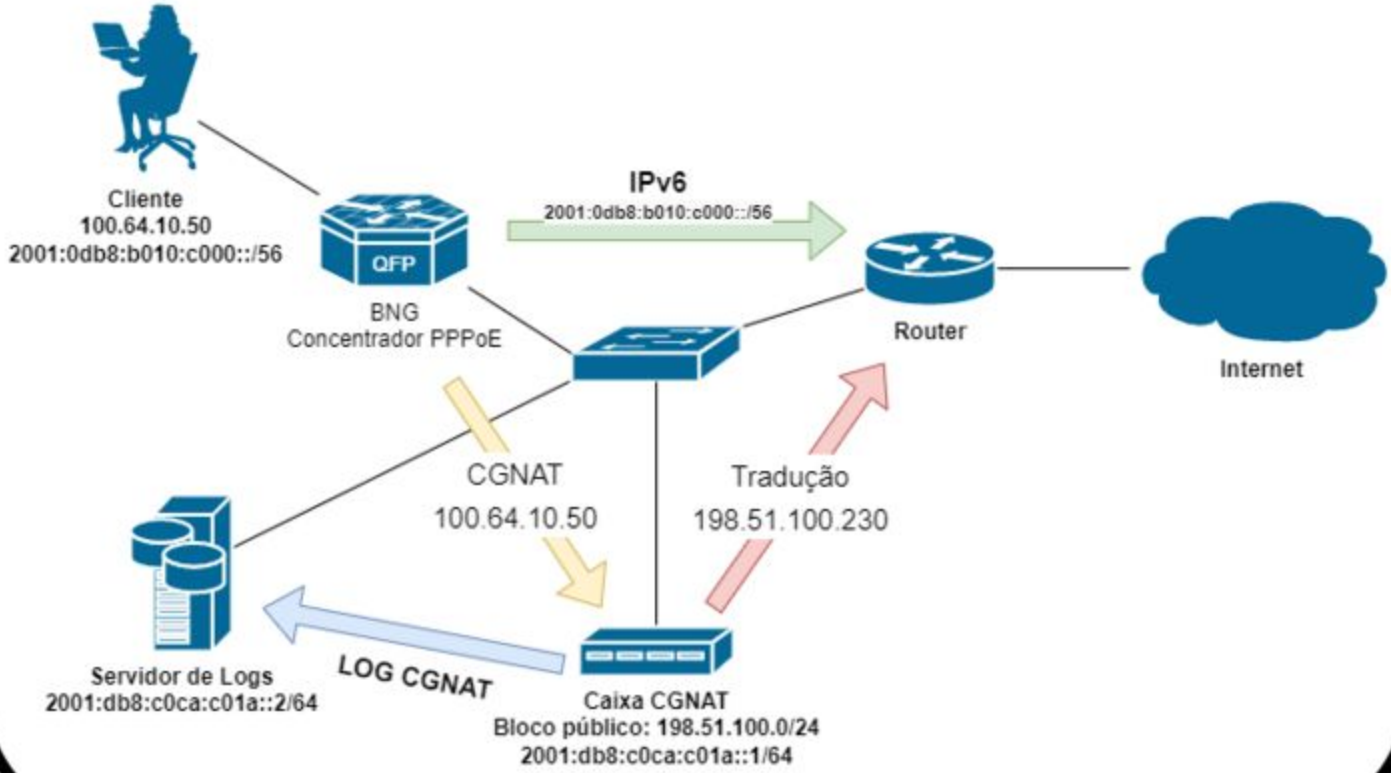
Enquanto os fabricantes de equipamentos como **Cisco**, **Huawei**, **Juniper** enviam os logs de **Prefix Delegation IPv6** para o Radius, temos um problema com o **Mikrotik RouterOS 6.x**. Este não faz, mas será mostrado como enviar os **logs de IPv6** para o nosso servidor de Logs via SYSLOG. Porque você pode receber um Ofício para quebra de sigilo com logs contendo IP de origem IPv6.

Os formatos que trabalharemos neste artigo, são os mais utilizados nos equipamentos de Redes:

- **Syslog**: é um serviço muito utilizado para armazenar logs sejam eles quais forem; mensagens de outros serviços, erros de sistema, um acesso indevido ou tentativas de burlar um sistema, enfim, qualquer coisa que possa dar uma informação útil para uma análise da situação. Além de tudo isso, podemos utilizar esse recurso para armazenar logs de CGNAT. O problema é que o syslog não é tão robusto e não lida muito bem com muitas requisições simultâneas, aumentando em muito o uso do processamento e dependendo da situação pode ocorrer alguma falha no registro do dado. O que vemos é que para registrar logs de sistemas e processos de um servidor, ele se sai muito bem, mas quando temos diversos BNGs enviando logs de CGNAT, ele não seria a melhor opção.

- **Netflow**: o envio de flow é utilizado em muitos ambientes para análise por amostragem e com diversas aplicações como por exemplo: sistemas de detecção e mitigação DDoS, sistemas que geram estatísticas úteis para análise de tráfego de Redes, dando informações valiosíssimas para o pessoal da Engenharia de Redes, consumos de determinados conteúdos para avaliar se pode ser solicitado algum Cache de CDN para a sua Operação, detecção de Botnets, enfim, muita coisa pode ser feita usando Netflow inclusive tem uma função especialmente elaborada para logs de CGNAT. Sem falar que ao contrário do nosso amigo **syslog**, esse sistema se mostra muito mais robusto, preparado para receber muitas requisições simultâneas e com menor consumo de CPU.

Neste artigo, iremos configurar o nosso servidor para receber ambos os formatos mas por experiência própria, quando falamos de logs de CGNAT, procure sempre que puder, utilizar o **Netflow** em detrimento ao **Syslog**.


Cliente
100.64.10.50
2001:0db8:b010:c000::/56
QFP
BNG
Concentrador PPPoE
IPv6
2001:0db8:b010:c000::/56
Router
Internet
CGNAT
100.64.10.50
Tradução
198.51.100.230
Servidor de Logs
2001:db8:c0ca:c01a::2/64
LOG CGNAT
Caixa CGNAT
Bloco público: 198.51.100.0/24
2001:db8:c0ca:c01a::1/64

Como base para a nossa configuração usaremos o diagrama acima e para poder visualizar o que ocorre em um ambiente com CGNAT. Nesse exemplo temos um cliente, que ao se conectar no Provedor de Internet, recebeu um IPv4 **100.64.10.50** (**RFC6598**) e um Prefix Delegation IPv6 **2001:0db8:b010:c000::/56**.

Todos os acessos a conteúdos em IPv6 **não passarão pelo CGNAT**, seguindo diretamente para a Internet e sendo registrado o log da conexão no seu servidor Radius, por exemplo. Já para o cliente acessar os conteúdos em IPv4, será necessário que os pacotes passem pela Caixa CGNAT, seja feita uma tradução do IP **100.64.10.50** para um IPv4 público, que no nosso exemplo é o **198.51.100.230**.

Sempre que eu tiver que passar pela **Caixa CGNAT** e for feita uma tradução, será necessário armazenar esses dados através de logs que serão enviados para o nosso servidor de logs CGNAT em **2001:db8:c0ca:c01a::2**.

Se você já tem IPv6 rodando em sua Operação, procure utilizar esse protocolo para se comunicar entre seus sistemas, não precisa ser um IPv6 Global, pode ser um endereço **ULA (Unique Local Address)** que utiliza o prefixo **FC00::/7**, dessa forma você vai inclusive economizar seus IPv4 públicos. O **ULA** seria o equivalente a usarmos prefixos IPv4 privados da **RFC1918**: **127.0.0.0/8**, **10.0.0.0/8**, **172.16.0.0/12** e **192.168.0.0/16**.

Lembre-se que segurança também é importante e se for usar IPs públicos, tanto em IPv4 quanto em IPv6, você precisa filtrar os acessos e garantir que só a sua gerência tenha acesso.

Requisitos para o sistema:

- Debian GNU/Linux 11 (Bullseye) amd64 com LVM (Logical Volume Manager).
- Nfdump (Netflow).
- Syslog-ng.
- Pigz (compactação dos logs).

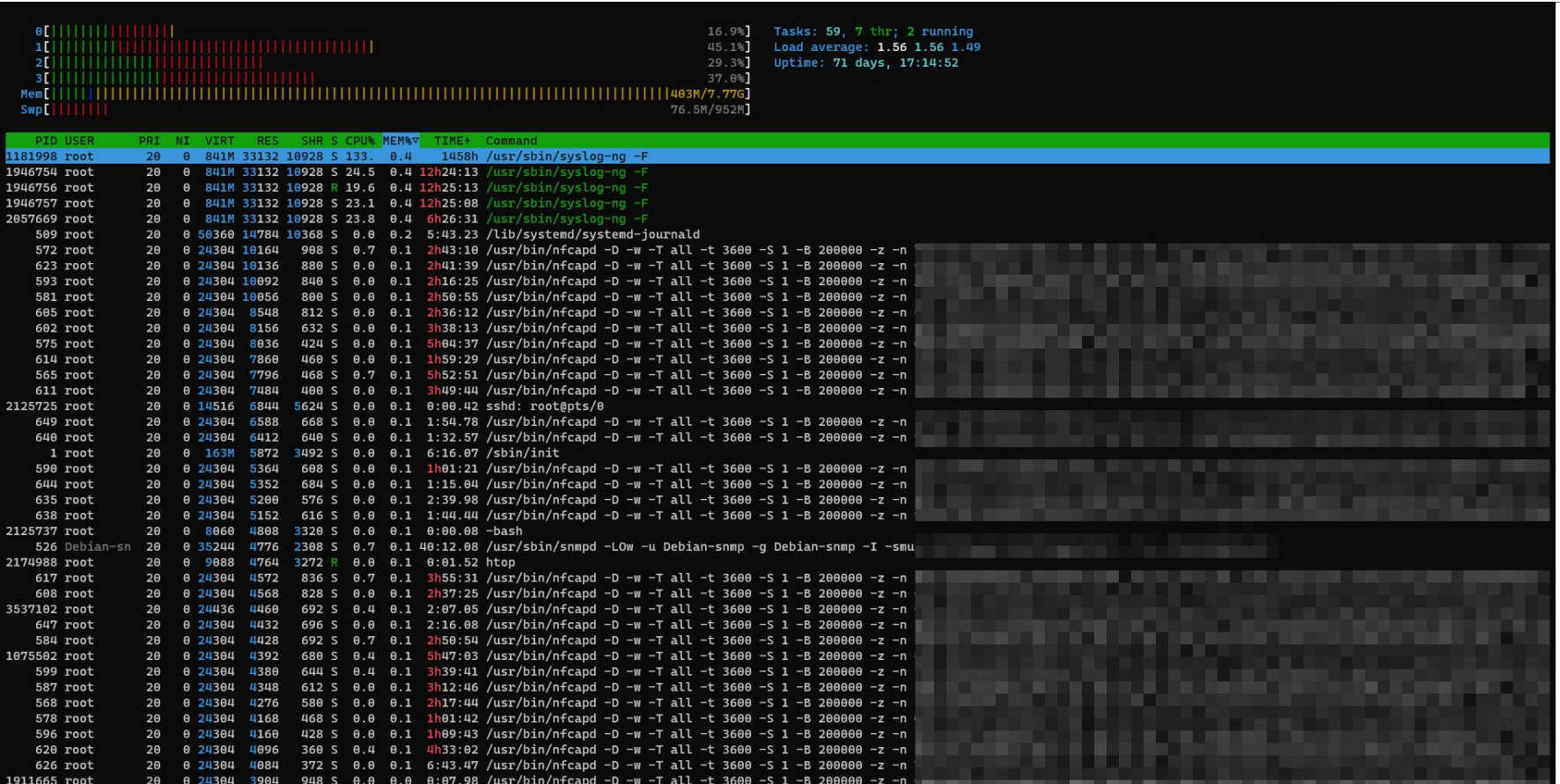
Hardware:

Um servidor de logs não necessita ter uma alta capacidade de processamento, se você não estiver abusando dos recursos como por exemplo: muitas caixas de CGNAT enviando milhares de conexões via syslog. Agora uma coisa é certa de que você vai precisar: espaço em disco e que tenha uma ótima performance de I/O. Disco é de fato o ponto chave na solução e por isso não utilize discos usados de outros servidores e monitore sempre a saúde dos discos para não ser pego de surpresa. O hardware que você vai precisar, dependerá do tamanho da sua Organização e do volume de dados que você enviará para o seu servidor ou até mesmo servidores. Você poderia ter, por exemplo, um servidor de logs CGNAT em cada cidade que atender.

Outro detalhe importante: pense no crescimento do armazenamento dos dados e que em determinado momento você precisará expandir isso, seja adicionando mais um disco ou aumentando o uso em uma Cloud. Se você for montar uma infraestrutura própria, virtualizada e utilizando um storage, já é um bom começo.

Esse é um **htop** de um servidor virtualizado, que recebe milhares de requisições de envio de logs por segundo tanto de **syslog**, quanto de **netflow**:

B.P.F
Brasil Peering Fórum



Quanto à armazenagem desse sistema acima:

B.P.F
Brasil Peering Fórum

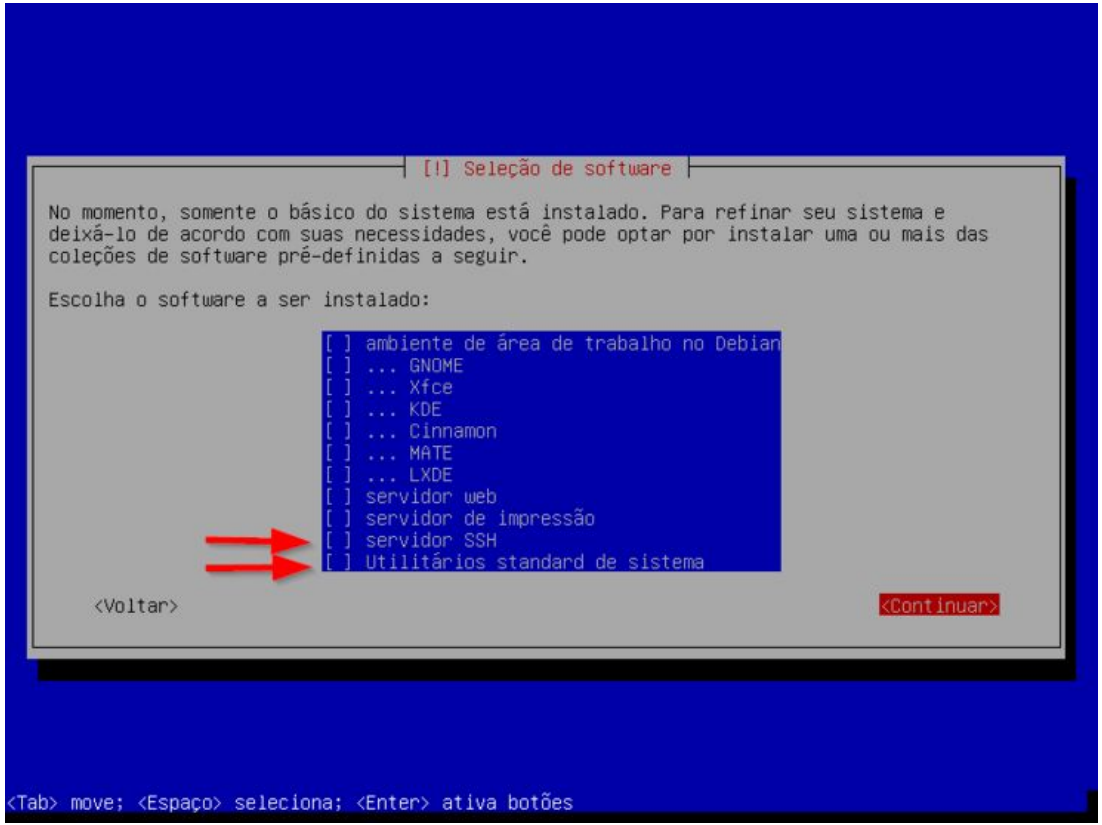
```
└── # df -h
Sist. Arq.              Tam. Usado Disp. Uso% Montado em
udev                    3,9G     0  3,9G   0% /dev
tmpfs                   796M   81M  716M  11% /run
/dev/mapper/root-root    46G  1,5G   42G   4% /
tmpfs                   3,9G  504K  3,9G   1% /dev/shm
tmpfs                   5,0M     0  5,0M   0% /run/lock
/dev/mapper/boot-boot   451M   86M  338M  21% /boot
/dev/mapper/var-data     19T   13T  4,8T  73% /var
tmpfs                    38M     0   38M   0% /run/user/0
```

Esse é um sistema novo, com armazenamento de 6 meses de logs de CGNAT e sabendo-se que precisaremos armazenar pelo menos **1 ano de log sempre**. Sim, esse espaço precisará de um novo aumento em breve através do uso do **LVM**. A sua realidade pode não ser essa e você pode necessitar de muito menos recurso que isso, então comece por baixo mas pensando em possível expansão de espaço de disco:

CPU	Memória	Disco
2.4Ghz 4 cores	8G DDR4	19Tb

Preparando o Servidor:

Quando for instalar o Debian, não esqueça de configurar o **LVM** na área de armazenagem dos logs e instalar o mínimo de pacotes que puder, deixando apenas os pacotes básicos e o servidor ssh para acesso remoto. Na imagem abaixo deixe tudo desmarcado e selecione apenas os 2 últimos itens.



Após a instalação vamos deixar o nosso **/etc/apt/sources.list** da seguinte forma:


```
deb http://security.debian.org/debian-security bullseye-security main contrib non-free
deb http://deb.debian.org/debian bullseye main non-free contrib
deb http://deb.debian.org/debian bullseye-updates main contrib non-free
deb http://deb.debian.org/debian bullseye-backports main contrib non-free
```

Costumo instalar alguns pacotes além dos necessários para ter algumas ferramentas à mão:

```
# apt install net-tools htop iotop sipcalc tcpdump curl gnupg rsync wget host
dnsutils mtr-tiny bmon sudo tmux whois syslog-ng nfdump pigz chrony irqbalance
```

```
# systemctl enable irqbalance
# echo "vm.swappiness=10" >> /etc/sysctl.conf
# sysctl -p
```

Acertando o horário do servidor:

Não existe nada tão importante em um sistema de armazenagem de logs, do q estarem devidamente certos e sincronizados. Se isso estiver errado, seu sis e não conseguirá identificar corretamente quando os eventos ocorreram realm deixar uma dica de como eu trabalho, mas fica apenas como dica.

Eu utilizo sempre horário **UTC** nos servidores de logs, isso porque podemos trabalhar com sistemas espalhados em diversos lugares, em fusos horários diferentes e também forço no **syslog-ng** para registrar o log sempre no **horário do servidor de logs**, ao invés do horário enviado pelo equipamento remoto. Isso porque o equipamento remoto pode estar com data e hora errados, mas no momento que os logs são enviados para o nosso servidor de logs, ele fica registrado corretamente, mesmo que em horário **UTC**.

Edite o arquivo **/etc/chrony/chrony.conf** e comente a linha com "**pool 2.debian.pool.ntp.org iburst"** e na sequência adicione as linhas conforme abaixo:

```
#pool 2.debian.pool.ntp.org iburst
server a.st1.ntp.br iburst nts
server b.st1.ntp.br iburst nts
server c.st1.ntp.br iburst nts
server d.st1.ntp.br iburst nts
```

Salve o arquivo e reinicie o serviço **chronyd**:

```
# systemctl restart chronyd.service
```

Agora vamos configurar o tzdata para horário **UTC**:

```
# timedatectl set-timezone UTC
```

Configurando o Syslog-ng:

Todas as configurações que faremos neste tutorial serão para armazenar os logs de forma estruturada e que facilite mais as buscas e limpeza do sistema quando necessário.

A estrutura de armazenamento que utilizaremos para o syslog será esta abaixo:

/var/log/cgnat/syslog/<HOSTNAME>/<ANO>/<MÊS>/<DIA>/server-<HORA>.log

```
# mkdir -p /var/log/cgnat/syslog
```

Onde:

- **<HOSTNAME>** é o nome do host, recebido do equipamento que estamos coletando nossos dados.
- **<ANO>**, **<MÊS>**, **<DIA>** e **<HORA>** serão usados pelo **syslog-ng** para automaticamente criar essa estrutura de diretórios e arquivos de forma organizada.

Dessa forma fica mais rápido de localizar quem usou determinado IP e em que data e hora. Repare que os arquivos são criados por hora de **00** até **23** facilitando a localização da informação:

```
┌─[root@ ]─[/var/log/cgnat/syslog/ CE01/2022/10/24]─[14:50:57]
└── # l
total 406100
-rw-r----- 1 root adm 22546030 out 24 00:59 server-00.log.gz
-rw-r----- 1 root adm 22295409 out 24 01:59 server-01.log.gz
-rw-r----- 1 root adm 19511381 out 24 02:59 server-02.log.gz
-rw-r----- 1 root adm 15104138 out 24 03:59 server-03.log.gz
-rw-r----- 1 root adm 12159922 out 24 04:59 server-04.log.gz
-rw-r----- 1 root adm 10203229 out 24 05:59 server-05.log.gz
-rw-r----- 1 root adm  8778175 out 24 06:59 server-06.log.gz
-rw-r----- 1 root adm  8152663 out 24 07:59 server-07.log.gz
-rw-r----- 1 root adm  9560271 out 24 08:59 server-08.log.gz
-rw-r----- 1 root adm 11836170 out 24 09:59 server-09.log.gz
-rw-r----- 1 root adm 14059478 out 24 10:59 server-10.log.gz
-rw-r----- 1 root adm 14372733 out 24 11:59 server-11.log.gz
-rw-r----- 1 root adm 15007419 out 24 12:59 server-12.log.gz
-rw-r----- 1 root adm 16748918 out 24 13:59 server-13.log.gz
-rw-r----- 1 root adm 20060654 out 24 14:59 server-14.log.gz
-rw-r----- 1 root adm 23116700 out 24 15:59 server-15.log.gz
-rw-r----- 1 root adm 23987038 out 24 16:59 server-16.log.gz
-rw-r----- 1 root adm 22258049 out 24 17:59 server-17.log.gz
-rw-r----- 1 root adm 21300876 out 24 18:59 server-18.log.gz
-rw-r----- 1 root adm 20743376 out 24 19:59 server-19.log.gz
-rw-r----- 1 root adm 17252096 out 24 20:59 server-20.log.gz
-rw-r----- 1 root adm 19978342 out 24 21:59 server-21.log.gz
-rw-r----- 1 root adm 22689108 out 24 22:59 server-22.log.gz
-rw-r----- 1 root adm 24072178 out 24 23:59 server-23.log.gz
┌─[root@ ]─[/var/log/cgnat/syslog/ CE01/2022/10/24]─[14:51:00]
└── #
```

Observação: o **/var/log/cgnat** precisa estar dentro do volume **LVM** conforme c
nele que serão armazenados todos os logs recebidos.

Para configurarmos essa estrutura precisamos alterar o arquivo **/etc/syslog-ng/sy**
o **options** para esse abaixo:

```
options { chain_hostnames(off); flush_lines(0); use_dns(no); use_fqdn(no)keep_hostname (yes);
          dns_cache(no); owner("root"); group("adm"); perm(0640); dir_perm(0700)create_dirs (yes);
          stats_freq(0); bad_hostname("^gconfd$");keep-timestamp(off);
};
```

Vamos criar um arquivo de configuração que vai ser o responsável por receber e criar a estrutura acima. Crie o seguinte novo arquivo **/etc/syslog-ng/conf.d/isp.conf** com o conteúdo de exemplo abaixo:

```
source s_net {
   udp6(ip("2001:db8:c0ca:c01a::2") port(514));
};

destination d_ce { file("/var/log/cgnat/syslog/${HOST}/${YEAR}/${MONTH}/${DAY}/server-${HOUR}.log"); };

filter f_ce { facility(daemon) and not message(".*SSH.*"); };
filter f_ce_ipv6 { facility(syslog); };

log { source(s_net); filter(f_ce); destination(d_ce); };
log { source(s_net); filter(f_ce_ipv6); destination(d_ce); };
```

Reinicie o serviço e o sistema começará a tratar os dados recebidos conforme especificamos acima:

```
# systemctl restart syslog-ng.service
```

Compactando os logs de syslog diariamente:

Abaixo um script que irá varrer diariamente os arquivos de log automaticamente. Não se preocupe se quiser rodá-lo manu vezes, ele só compacta o que ainda não tiver sido compactado e sempre os logs do dia anterior.

```
# mkdir -p /root/scripts
```

Daremos o nome do script de **/root/scripts/compacta_syslog.sh** e o seu conteúdo está logo abaixo:

```
#!/bin/bash

ANO=$(date -d "-1 day" '+%Y')
MES=$(date -d "-1 day" '+%m')
DIA=$(date -d "-1 day" '+%d')
for lista in /var/log/cgnat/syslog/*; do
   if [ -d $lista/$ANO/$MES/$DIA ]; then
        pigz -p4 --fast $lista/$ANO/$MES/$DIA/*
   fi
done
```

```
# chmod 700 /root/scripts/compacta syslog.sh
# echo "00 4    * * *   root    /root/scripts/compacta_syslog.sh" >> /etc/crontab
```

Configurando o Netflow:

Para receber e armazenar os logs em netflow usaremos o **nfcapd** do pacote **nfdum** parâmetros que já facilitam nossa vida criando uma estrutura de diretórios de for de buscar o que precisamos. Procure configurar o **Netflow** do equipamento que enviará os logs para **versão 9**. Precisaremos ter em mãos, os seguintes dados do equipamento para liberação da coleta dos dados em nosso servidor:

- **HOSTNAME** do equipamento.
- **IP do equipamento** que será usando para envio dos logs.
- **Porta UDP** onde será recebido o flow daquele equipamento. Para cada equipamento que for enviar um flow, você precisa especificar uma porta UDP dedicada para ele.

Com esses dados montamos a linha de comando que ficará escutando o flow. Abaixo um exemplo:

```
/usr/bin/nfcapd -D -w -T all -t 3600 -S 1 -B 200000 -z -n
RJO-DC01-CGNAT-01,2001:db8:c0ca:c01a::1,/var/log/cgnat/flow/RJO-DC01-CGNAT-01 -b 2001:db8:c0ca:c01a::2 -p 2055
```

Onde:

HOSTNAME é o **RJO-DC01-CGNAT-01**
IP é **2001:db8:c0ca:c01a::1**
PORTA UDP é **2055**

O **"-D"** diz ao programa rodar como daemon, **"-w"** e **"-t"** dizem de quanto em quanto tempo o arquivo é rotacionado. Nesse caso **3600s**, ou seja, de hora em hora. O **"-T all"** especifica que o **nfdump** vai suportar todas as implementações de **Netflow v9**. O **"-S 1"** diz para arquivar os logs no formato **year/month/day** automaticamente. O **"-B 200000"** é para aumentar o buffer e não termos problemas com perdas de flow. O **"-z"** comprime o flow em **LZO1X-1**. O **"-n"** é onde passamos o **HOSTNAME**, **IP** e **diretório base**. O **"-b 2001:db8:c0ca:c01a::2"** seria o IP do servidor de logs CGNAT e o **"-p 2055"** a especificação da porta udp que irá receber o flow daquele equipamento.

Precisamos sempre antes de executar o comando, criar o diretório base, senão o serviço não irá levantar. No exemplo acima faremos assim:

```
# mkdir -p /var/log/cgnat/flow/RJO-DC01-CGNAT-01
```

Com os dados dos equipamentos podemos montar um arquivo **/etc/rc.local** e colocar uma entrada por linha conforme as regras que passei acima. Então nosso arquivo ficaria assim:

```
#!/bin/sh -e
#
# rc.local
#
# This script is executed at the end of each multiuser runlevel.
# Make sure that the script will "exit 0" on success or any other
# value on error.
#
# In order to enable or disable this script just change the execution
# bits.
#
# By default this script does nothing.

/usr/bin/nfcapd -D -w -T all -t 3600 -S 1 -B 200000 -z -n
RJO-DC01-CGNAT-01,2001:db8:c0ca:c01a::1,/var/log/cgnat/flow/RJO-DC01-CGNAT-01 -b 2001:db8:c0ca:c01a::2 -p 2055

exit 0
```

```
# chmod +x /etc/rc.local
```

Podemos fazer um reboot no sistema e checar com um **ps afx**, se nosso serviço está rodando como configuramos. Como estamos nos baseando no nosso diagrama, só teríamos esse servidor enviando flow e no seu ambiente de produção provavelmente existirão outros equipamentos fazendo CGNAT. Nesse caso só ir adicionando novas linhas do **nfcapd** mas prestando atenção nos parâmetros e principalmente na **porta UDP**.

Compactando os logs de netflow diariamente:

B . P . F
Brasil Peering Fórum

Assim como fizemos no **syslog-ng**, configuramos um script qu
diariamente os logs armazenados via **netflow**. Vamos
/root/scripts/compacta_flow.sh e adicioná-lo também ao nosso cron para rodar diariamente. Abaixo o conteúdo do script:

```
#!/bin/bash

ANO=$(date -d "-1 day" '+%Y')
MES=$(date -d "-1 day" '+%m')
DIA=$(date -d "-1 day" '+%d')
for FOLDER in /var/log/cgnat/flow/*; do
  if [ -d $FOLDER/$ANO/$MES/$DIA ]; then
     cd $FOLDER/$ANO/$MES/$DIA
     echo "Compactando: ${FOLDER}/$ANO/$MES/$DIA/"
     pigz -p4 --fast nfcapd*
  fi
done
```

```
# chmod 700 /root/scripts/compacta_flow.sh
# echo "00 4    * * *   root    /root/scripts/compacta_flow.sh" >> /etc/crontab
```

Fazendo consultas do flow armazenado:

B . P . F
Brasil Peering Fórum

A forma de consultar dependerá de como você configurou sua Caixa C
comentei que existe uma configuração de flow que pode ocupar muito
e que devemos evitá-la? Abaixo mostro um exemplo. Repare também que estamos usando a mesma estrutura de organização de diretórios e arquivos, que usamos na configuração do syslog. Essa configuração tenho casos de gerar **mais de 400Mb de log/hora**.

```
# cd /var/log/cgnat/flow/RJO-DC05-CGNAT01/2022/08/04
# cat nfcapd.202208042300|nfdump -r -
2022-08-04 22:59:59.992     ADD  Ignore TCP     100.64.202.178:50419 ->          0.0.0.0:0      170.XXX.XXX.XXX:50419 ->
0.0.0.0:0            0        0
2022-08-04 23:00:31.369  DELETE  Ignore TCP     100.64.202.178:50419 ->          0.0.0.0:0      170.XXX.XXX.XXX:50419 ->
0.0.0.0:0            0        0
2022-08-04 23:13:34.417     ADD  Ignore UDP     100.64.199.100:50419 ->          0.0.0.0:0      170.XXX.XXX.XXX:50419 ->
0.0.0.0:0            0        0
2022-08-04 23:15:35.908  DELETE  Ignore UDP     100.64.199.100:50419 ->          0.0.0.0:0      170.XXX.XXX.XXX:50419 ->
0.0.0.0:0            0        0
2022-08-04 23:19:48.215     ADD  Ignore TCP     100.64.199.100:50419 ->          0.0.0.0:0      170.XXX.XXX.XXX:50419 ->
0.0.0.0:0            0        0
2022-08-04 23:20:19.283  DELETE  Ignore TCP     100.64.199.100:50419 ->          0.0.0.0:0      170.XXX.XXX.XXX:50419 ->
0.0.0.0:0            0        0
2022-08-04 23:41:02.639     ADD  Ignore UDP      100.64.204.78:50419 ->          0.0.0.0:0      170.XXX.XXX.XXX:50419 ->
0.0.0.0:0            0        0
2022-08-04 23:43:02.435  DELETE  Ignore UDP      100.64.204.78:50419 ->          0.0.0.0:0      170.XXX.XXX.XXX:50419 ->
0.0.0.0:0            0        0
```

Nesse exemplo acima temos o arquivo **nfcapd.202208042300** e esses arquivos são gerados de hora em hora com os dados recebidos. O nome dele é composto pelo **ano**, **mês**, **dia** e a **hora de rotação,** que nesse caso seria **23:00**. Podemos reparar que a cada solicitação de conexão do cliente, é enviado um log da porta que foi utilizada, neste nosso exemplo a porta **50419 TCP** e **UDP**. Isso realmente ocupa muito espaço em disco.

Agora vamos olhar para um BPA que está configurado para trabalhar com alocação dinâmica de blocos de portas. A consulta é um pouco diferente porque precisa ser formatada. Detalhe de logs **menores que 500Kb/hora**.

```
# cd /var/log/cgnat/flow/CGN-BOX1-SPO01/2023/01/15
# cat nfcapd.202301151100 | nfdump -o "fmt:%ts %nevt %pr %sa %nda %nsa %pbstart %pbend" -r -
```

Date first seen	Event	Proto	Src IP Addr	X-late Dst IP	X-late Src IP	Pb-Start	Pb-End
2023-01-15 10:54:48.746	ADD	255	100.113.127.23	0.0.0.0	186	8192	8447
2023-01-15 10:54:48.858	ADD	255	100.113.24.254	0.0.0.0	186.	5888	6143
2023-01-15 10:54:51.271	DELETE	255	100.113.13.121	0.0.0.0	170	31488	31743
2023-01-15 10:54:51.070	ADD	255	100.113.94.129	0.0.0.0	18	13568	13823
2023-01-15 10:54:52.510	ADD	255	100.113.42.63	0.0.0.0	170	22784	23039
2023-01-15 10:54:53.925	DELETE	255	100.113.124.135	0.0.0.0	186	13312	13567
2023-01-15 10:54:56.343	ADD	255	100.113.110.240	0.0.0.0	170	19200	19455
2023-01-15 10:54:57.087	ADD	255	100.113.13.11	0.0.0.0	170	18944	19199
2023-01-15 10:54:57.850	DELETE	255	100.113.73.36	0.0.0.0	186.	34048	34303
2023-01-15 10:54:58.797	DELETE	255	100.113.109.3	0.0.0.0	186.	8704	8959
2023-01-15 10:54:58.659	DELETE	255	100.113.65.240	0.0.0.0	170	46848	47103
2023-01-15 10:54:59.603	DELETE	255	100.113.37.179	0.0.0.0	170	52224	52479
2023-01-15 10:55:00.213	ADD	255	100.113.0.250	0.0.0.0	186	65280	65535
2023-01-15 10:55:00.826	DELETE	255	100.113.75.141	0.0.0.0	170	21760	22015
2023-01-15 10:55:00.662	ADD	255	100.113.66.204	0.0.0.0	170	38144	38399
2023-01-15 10:55:01.889	ADD	255	100.113.123.228	0.0.0.0	186	23808	24063
2023-01-15 10:55:02.433	DELETE	255	100.113.4.173	0.0.0.0	170	37888	38143
2023-01-15 10:55:01.579	ADD	255	100.113.42.169	0.0.0.0	186	41472	41727
2023-01-15 10:55:01.611	ADD	255	100.113.67.40	0.0.0.0	186	4864	5119
2023-01-15 10:55:03.107	ADD	255	100.113.31.84	0.0.0.0	170	35584	35839
2023-01-15 10:55:07.126	ADD	255	100.113.27.244	0.0.0.0	170	59392	59647
2023-01-15 10:55:07.274	DELETE	255	100.113.103.238	0.0.0.0	186.	39424	39679
2023-01-15 10:55:06.591	ADD	255	100.113.77.243	0.0.0.0	170	64512	64767
2023-01-15 10:55:07.907	DELETE	255	100.113.10.22	0.0.0.0	170	50176	50431
2023-01-15 10:55:08.305	DELETE	255	100.113.82.142	0.0.0.0	170	20992	21247
2023-01-15 10:55:08.050	DELETE	255	100.113.30.31	0.0.0.0	170	32512	32767
2023-01-15 10:55:09.330	ADD	255	100.113.25.52	0.0.0.0	186	48896	49151
2023-01-15 10:55:10.129	ADD	255	100.113.42.35	0.0.0.0	170	14848	15103
2023-01-15 10:55:10.545	ADD	255	100.113.37.250	0.0.0.0	170	65024	65279
2023-01-15 10:55:10.689	ADD	255	100.113.57.38	0.0.0.0	170	55808	56063
2023-01-15 10:55:12.583	ADD	255	100.113.115.48	0.0.0.0	170	13056	13311
2023-01-15 10:55:12.554	ADD	255	100.113.114.74	0.0.0.0	170	30464	30719
2023-01-15 10:55:14.067	DELETE	255	100.113.92.178	0.0.0.0	170	47872	48127
2023-01-15 10:55:15.354	ADD	255	100.113.11.47	0.0.0.0	170	63232	63487
2023-01-15 10:55:16.694	ADD	255	100.113.116.137	0.0.0.0	170	49152	49407
2023-01-15 10:55:16.694	DELETE	255	100.113.116.137	0.0.0.0	170	49152	49407
2023-01-15 10:55:16.694	ADD	255	100.113.116.137	0.0.0.0	170	20224	20479
2023-01-15 10:55:18.263	ADD	255	100.113.115.6	0.0.0.0	170	16128	16383
2023-01-15 10:55:18.866	DELETE	255	100.113.110.28	0.0.0.0	170	23040	23295
2023-01-15 10:55:19.137	DELETE	255	100.113.116.52	0.0.0.0	170	18688	18943
2023-01-15 10:55:19.593	ADD	255	100.113.78.31	0.0.0.0	186	40192	40447
2023-01-15 10:55:19.949	DELETE	255	100.113.4.123	0.0.0.0	170	63744	63999
2023-01-15 10:55:20.441	DELETE	255	100.113.37.91	0.0.0.0	18	19968	20223
2023-01-15 10:55:21.379	DELETE	255	100.113.7.32	0.0.0.0	186	37376	37631
2023-01-15 10:55:20.898	DELETE	255	100.113.116.114	0.0.0.0	170	59904	60159
2023-01-15 10:55:21.190	ADD	255	100.113.42.220	0.0.0.0	170	32768	33023
2023-01-15 10:55:21.978	ADD	255	100.113.14.187	0.0.0.0	186	27648	27903
2023-01-15 10:55:23.067	DELETE	255	100.113.29.242	0.0.0.0	170	29696	29951
2023-01-15 10:55:23.137	ADD	255	100.113.91.149	0.0.0.0	170	25088	25343

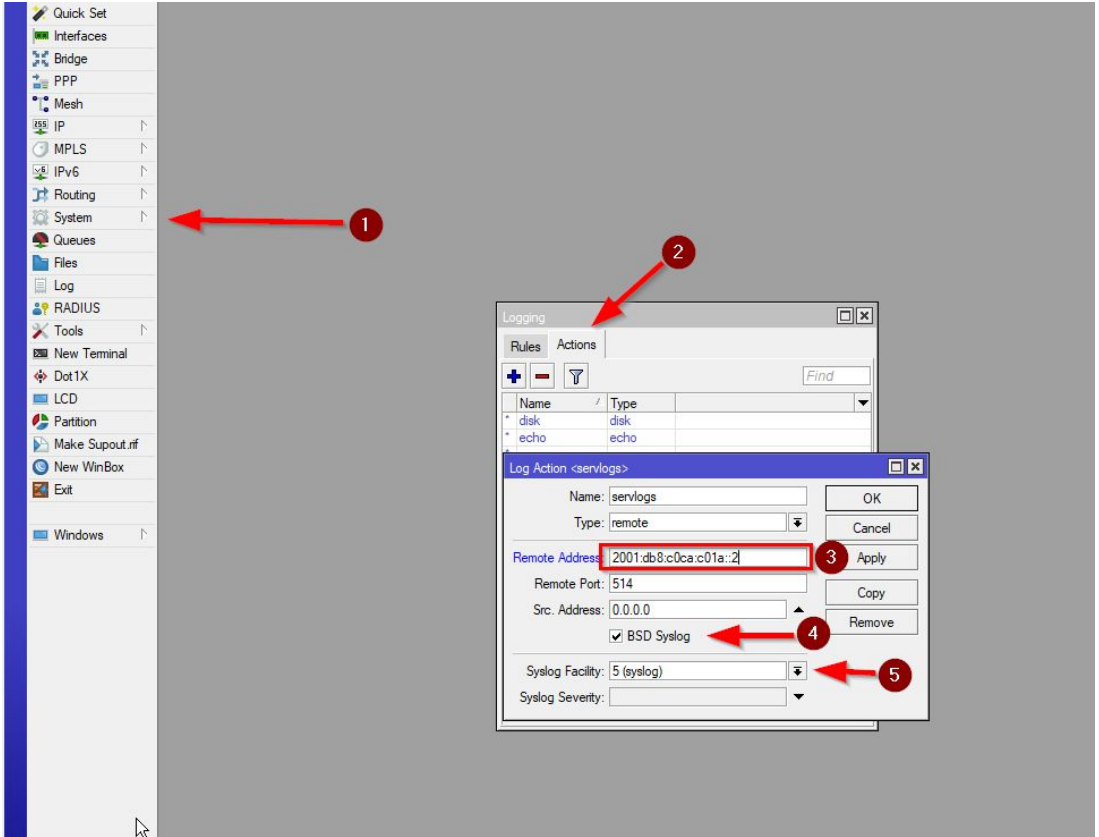
--Mais--

B.P.F
Brasil Peering Fórum

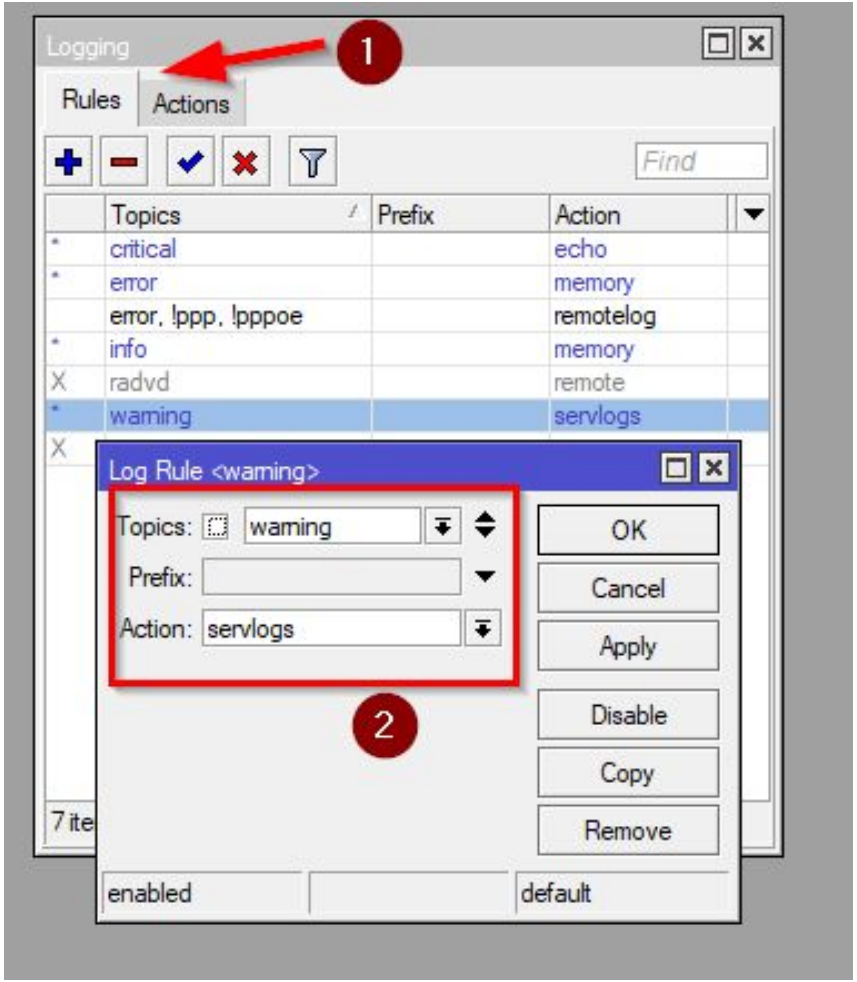
Configurando um Mikrotik RouterOS para envio de logs IPv6:

Brasil Peering Fórum

Infelizmente o Mikrotik até a versão 6.x pelo menos, não sei se já corrigiram na versão 7, não enviava para o Radius o registro do **IPv6 Prefix Delegation entregue para o cliente** e por isso se você receber uma solicitação de Quebra de Sigilo com logs em IPv6, dificilmente você irá conseguir identificar o cliente estando nessa situação. Para isso trago uma solução que veio da contribuição do **Bruno Viviani** neste artigo dele**.**

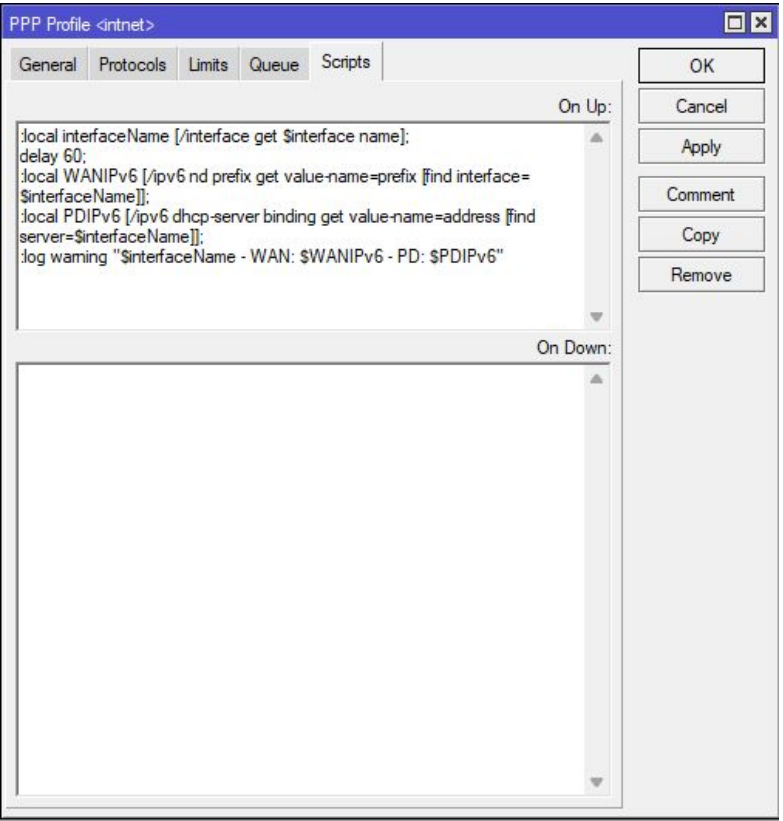


Após isso modificamos a **Rule warning** e apontamos para o nosso **servlogs**:



Por último temos que adicionar no **profile do PPPoE**, na parte de scripts em **On Up**:

```
:local interfaceName [/interface get $interface name];
delay 60;
:local WANIPv6 [/ipv6 nd prefix get value-name=prefix [find interface=$interfaceName]];
:local PDIPv6 [/ipv6 dhcp-server binding get value-name=address [find server=$interfaceName]];
:log warning "$interfaceName - WAN: $WANIPv6 - PD: $PDIPv6"
```



B.P.F
Brasil Peering Fórum

Nos logs do servidor aparecerão algo assim:

B.P.F
Brasil Peering Fórum

```
Jan 25 02:26:56 SPO-SEDE-CE02 <pppoe-jose> - WAN: 2804:0db8:8005:5668::/64 - PD: 2804:0db8:8050:7b00::/56
Jan 25 02:37:08 SPO-SEDE-CE02 <pppoe-maria> - WAN: 2804:0db8:8005:5669::/64 - PD: 2804:0db8:8050:a000::/56
```

Desta forma o ISP conseguirá fazer a Quebra de Sigilo Tecnológico de logs com IPv6 recebidos de Ofícios.