

[Artigos](#) > **DevOps**

Protegendo seu servidor com IPTables

```
ri@laptop: ~  
p:~$ ping 192.168.0.1  
68.0.1 (192.168.0.1) 56(84) bytes of data.  
rom 192.168.0.1: icmp_seq=1 ttl=64 time=0.290 ms  
rom 192.168.0.1: icmp_seq=2 ttl=64 time=0.272 ms  
rom 192.168.0.1: icmp_seq=3 ttl=64 time=0.284 ms  
rom 192.168.0.1: icmp_seq=4 ttl=64 time=0.269 ms  
rom 192.168.0.1: icmp_seq=5 ttl=64 time=0.284 ms  
  
8.0.1 ping statistics ---  
transmitted, 5 received, 0% packet loss, time 4063ms  
g/max/mdev = 0.269/0.279/0.290/0.022 ms  
p:~$
```

**Yuri Matheus**

07/08/2017

COMPARTILHE



Esse artigo faz parte da
Formação DevOps

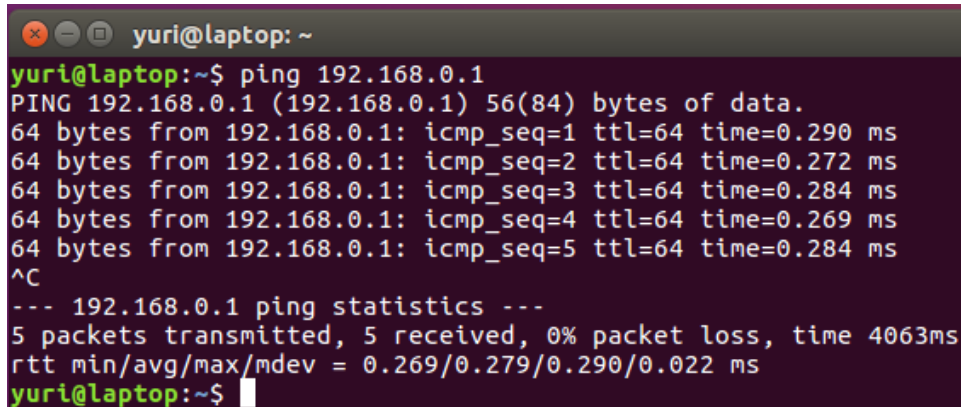
Estamos configurando um novo servidor de banco de dados aqui na Alura para armazenar os dados dos alunos, vamos como configurar o IPTables no nosso servidor, bloqueando protocolos, portas e permitindo acesso de uma máquina. Na reunião ficou decidido, por questões de segurança que:

- Para que usuários maliciosos, ou vírus, não encontrem nossa máquina, não será

possível realizar [ping](#);

- Para prevenir de hackers , ou vírus, tentar acessar nosso banco vamos **bloqueá-lo de acessos remotos**.

Legal, já que temos essas duas tarefas, começaremos pelo **ping**. Antes de tentarmos bloquear o ping, vamos verificar se o mesmo funciona, fazendo nossa máquina pingar (`ping`) o nosso servidor (`192.168.0.1`):



```
yuri@laptop: ~  
yuri@laptop:~$ ping 192.168.0.1  
PING 192.168.0.1 (192.168.0.1) 56(84) bytes of data.  
64 bytes from 192.168.0.1: icmp_seq=1 ttl=64 time=0.290 ms  
64 bytes from 192.168.0.1: icmp_seq=2 ttl=64 time=0.272 ms  
64 bytes from 192.168.0.1: icmp_seq=3 ttl=64 time=0.284 ms  
64 bytes from 192.168.0.1: icmp_seq=4 ttl=64 time=0.269 ms  
64 bytes from 192.168.0.1: icmp_seq=5 ttl=64 time=0.284 ms  
^C  
--- 192.168.0.1 ping statistics ---  
5 packets transmitted, 5 received, 0% packet loss, time 4063ms  
rtt min/avg/max/mdev = 0.269/0.279/0.290/0.022 ms  
yuri@laptop:~$
```

Viram? Nosso servidor está respondendo as requisições ping...

Como a resposta ao ping já vem habilitada por padrão na maioria das distribuições Linux, principalmente nas destinadas a serem servidores, precisamos bloqueá-las. Mas como podemos fazer isso?

Bloqueando o ping com o iptables

Bom, para bloquear o ping precisamos de uma ferramenta capaz de barrar protocolos, ou acesso a portas.

Para barrá-los vamos construir uma parede [firewall](#), na qual conterà regras que irão bloquear essas requisições. No Linux nós temos o `iptables` para resolver isso.

Para isso, temos que estar logados como **superusuário**, ou seja, como `root`. Feito isso vamos usar o `iptables` , inserir uma nova regra (`-I`) que entrará (`INPUT`) no servidor , no topo da nossa tabela de regras, isto é, na posição `1` :

```
# iptables -I INPUT 1
```

Agora que indicamos a nossa ação de bloqueio, precisamos adicionar mais algumas informações para especificar o nosso objetivo, isto é, bloquear o ping.

Como o ping funciona pelo protocolo **ICMP** , é ele que vamos indicar. Sendo assim, vamos

Como o ping funciona pelo protocolo [ICMP](#), e ele que vamos indicar. Sendo assim, vamos bloquear o protocolo (-p) ICMP (icmp):

```
# iptables -I INPUT 1 -p icmp
```

Temos também que informar o tipo que queremos bloquear no protocolo, que no caso é a resposta do servidor --icmp-type echo-request , após isso, definimos a ação (-j) para bloquear (DROP) os pacotes :

```
# iptables -I INPUT 1 -p icmp --icmp-type echo-request -j DROP
```

Note que o nosso comando é um pouco grande, será que funciona? Vamos testar:

```
[root@srv01 ~]# iptables -I INPUT 1 -p icmp --icmp-type echo-request -j DROP
[root@srv01 ~]# _
```

Bom, ele executou, mas como podemos ter certeza que está funcionando? Testando, né? Portanto, faremos novamente o teste do ping:

```
yuri@laptop: ~
yuri@laptop:~$ ping 192.168.0.1 -c 4
PING 192.168.0.1 (192.168.0.1) 56(84) bytes of data.

--- 192.168.0.1 ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 3050ms
```

Note que agora ele não responde mais ao ping! Ou seja, resolvemos a nossa primeira necessidade. Agora, se alguém tentar pingar o nosso servidor, ele não será encontrado.

Vamos dar uma olhada na nossa tabela e ver a regra que acabamos de criar.

Para o iptables mostrar a tabela de regras, basta falarmos para ele listar (-L) a tabela que queremos, no nosso caso é a tabela de entrada (INPUT).

```
# iptables -L INPUT
```

```
[root@srv01 ~]# iptables -L INPUT
Chain INPUT (policy ACCEPT)
target     prot opt source                destination            icmp echo-request
DROP      icmp -- anywhere              anywhere               icmp echo-request
[root@srv01 ~]# _
```

Nossa regra já está na tabela.

Porém, o nosso banco de dados ainda não foi bloqueado, portanto, a gente precisa fazer isso

Porém, o nosso banco de dados ainda não foi bloqueado, portanto, a gente precisa fazer isso agora.

Bloqueando uma porta com o iptables

Estamos utilizando um banco de dados [MariaDB](#).

Em geral, a instalação pergunta se desejamos permitir ou não o acesso a outras máquinas. Com isso você já pode se proteger. Senão vamos nos proteger agora autorizando o acesso somente a máquina local. Mas como podemos bloquear um banco de dados?

Para acessar um banco de dados, precisamos informar o endereço dele, junto da sua porta. Como iremos acessar internamente, o endereço será o nosso **localhost**, isto é, a nossa máquina local e, no caso, queremos que o acesso local seja realizado.

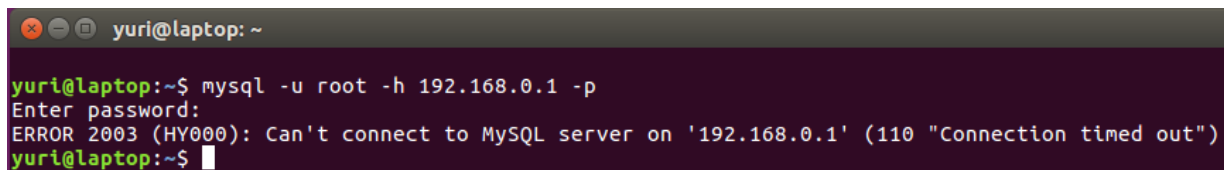
Em outras palavras, precisamos apenas bloquear a porta que dá acesso ao serviço do banco, pois, dessa forma, garantimos que ninguém o acesse.

O comando para bloquear uma porta é parecido com o anterior. A diferença é que precisamos informar a porta que desejamos bloquear.

Neste caso, é a porta de destino da requisição. Então basta informar o `--destination-port` junto do número, que para o banco de dados MariaDB é `3306`.

```
# iptables -I INPUT 1 -p tcp --destination-port 3306 -j DROP
```

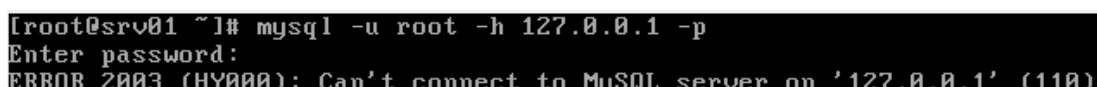
Se alguém tentar acessar remotamente nosso banco de dados obterá fracasso:



```
yuri@laptop: ~  
yuri@laptop:~$ mysql -u root -h 192.168.0.1 -p  
Enter password:  
ERROR 2003 (HY000): Can't connect to MySQL server on '192.168.0.1' (110 "Connection timed out")  
yuri@laptop:~$
```

Agora nosso banco de dados está bloqueado dos acessos externos. Ele só será acessado do próprio servidor, ou então, se alguém [acessar o servidor remotamente](#).

Muito bem! Conseguimos bloquear a porta. Vamos agora tentar acessar nosso banco diretamente do nosso servidor:



```
[root@srv01 ~]# mysql -u root -h 127.0.0.1 -p  
Enter password:  
ERROR 2003 (HY000): Can't connect to MySQL server on '127.0.0.1' (110)
```

```
python 2000 (11000) can't connect to nginx server on 127.0.0.1 (1107)
[root@srv01 ~]# _
```

//

"Hum... por quê não conseguimos acessar nosso banco?"

Quando bloqueamos o acesso a porta de todas as máquinas, isso inclui a nossa própria.

Libertando acesso de um endereço a uma porta

Para resolver esse problema, precisamos apenas permitir o acesso do nosso próprio servidor.

O comando para isso é parecido com o de bloquear. A única diferença é que, após informarmos a porta, precisamos dizer o endereço de origem (`-s`), que no nosso caso é o **localhost**, ou seja, o `127.0.0.1`. Lembrando que ao invés de bloquear (`DROP`), precisamos aceitar requisições, portanto `ACCEPT`.

Novamente, o comando completo fica assim:

```
# iptables -I INPUT 1 -p tcp --destination-port 3306 -s 127.0.0.1 -j ACCEPT
```

```
[root@srv01 ~]# iptables -I INPUT 1 -p tcp --destination-port 3306 -s 127.0.0.1 -j ACCEPT
[root@srv01 ~]# _
```

//

Mas se eu colocar INPUT 1, não vai sobrescrever a outra regra?

O comando `-I` insere uma nova regra, logo, quando digitarmos o comando acima, ele ficará na primeira posição de nossa tabela de regras, enquanto o que estava nessa posição irá para a segunda.

Vamos listar nossa tabela novamente para ter certeza disso:

```
# iptables -L INPUT
```

```
[root@srv01 ~]# iptables -L INPUT
Chain INPUT (policy ACCEPT)
target     prot opt source               destination          tcp dpt:mysql
ACCEPT     tcp  --  localhost            anywhere
```

```
DROP tcp -- anywhere anywhere tcp dpt:mysql
DROP icmp -- anywhere anywhere icmp echo-request
[root@srv01 ~]#
```

Viram? Nossas três regras estão na tabela do firewall.

Bom agora que já temos todas as regras criadas podemos falar para o iptables salvá-las. Então `iptables-save` para salvá-las.

Agora todas as nossas regras estão salvas no servidor.

Pontos de atenção

Vale lembrar que o **Linux** é um sistema **case sensitive**, isto é, letras maiúsculas e minúsculas são diferenciadas. Mudar um parâmetro pode fazer o comando não funcionar como esperado.

Também temos que lembrar que todos os comando acima são feitos como **superusuário**, ou seja, precisamos utilizar o `sudo` ou estar logados como root para funcionar.

Vale lembrar também que é preciso cuidado ao executar comandos como superusuário para não comprometer o sistema.

Conclusão

Nesse artigo vimos como configurar um firewall no Linux utilizando o iptables. Bloqueando tanto um protocolo, quanto uma porta. Também vimos como aceitar a requisição de um endereço IP.

Gostou do IPTables? Esse é apenas o começo quando falamos de segurança. Existem muitas coisas mais.

Aqui na Alura temos um curso de [segurança de redes com iptables e pfSense!](#)

Nele você vai aprender a configurar um firewall para proteger seu servidor de possíveis ataques e ameaças, como trabalhar com virtualização, proteger sua instância na Amazon EC2 e muito mais.





Yuri Matheus

Yuri é desenvolvedor e instrutor. É estudante de Sistemas de Informação na FIAP e formado como Técnico em Informática no Senac SP. O seu foco é nas plataformas Java e Python e em outras áreas como Arquitetura de Software e Machine Learning. Yuri também atua como editor de conteúdo no blog da Alura, onde escreve, principalmente, sobre Redes, Docker, Linux, Java e Python.

[Artigo Anterior](#)

[Próximo Artigo](#)

[**Virtual Hosts, virtualizando vários sites em um mesmo servidor**](#)

[**SSH: o acesso remoto aos servidores**](#)

Leia também:

- [SSH: o acesso remoto aos servidores](#)
- [Shell Script: Introdução e Como Automatizar Tarefas](#)
- [SSH, Telnet e as diferenças para conectar em um servidor](#)
- [Compartilhando arquivos com o Samba](#)
- [Virtual Hosts, virtualizando vários sites em um mesmo servidor](#)

Veja outros artigos sobre
[DevOps](#)

**Quer mergulhar em
tecnologia e aprendizagem?**

Receba a newsletter que o nosso CEO escreve pessoalmente, com insights do mercado de trabalho, ciência e desenvolvimento de software