TRYBE

# Arquitetura de Computadores e Redes

Dia 2 - Arquitetura de Redes II

# Índice

1. Segurança
2. Firewalls
3. Ping e Traceroute
4. Mão na massa
5. Fail2ban e Iptables

# Segurança

- **Contexto**

  No conteúdo conhecemos alguns cuidados que precisamos ter ao expor nossas aplicações na internet.

  Vimos também **como utilizar protocolos mais seguros**, assim como alguns dispositivos que podem nos ajudar no controle das redes como os **proxies e firewalls**.

  Além disso, existem diversas ferramentas, gratuitas ou pagas, que oferecem camadas adicionais de segurança, e serviços que propõem algumas soluções bem interessante.

# Segurança

Home → TECNOLOGIA → Desenvolvedor encontra falhas de segurança no Ingresso.com

TECNOLOGIA

## Desenvolvedor encontra falhas de segurança no Ingresso.com

Ambas as falhas encontradas colocariam em risco informações pessoais de usuários do serviço e, possivelmente, senhas

Por **Gustavo Gusmão**

Especialista explica que uma das vulnerabilidades aparece na hora de gerar um ingresso para impressão, após a compra. A outra, por sua vez, se dá na forma com que a companhia armazenaria as palavras-chaves de clientes.

O endereço referente à entrada adquirida podia ser facilmente modificado. Dessa forma, um usuário qualquer – ou alguém mal-intencionado mesmo – podia acessar as compras de outros clientes mudando alguns caracteres.

# Segurança

HOME / SEGURANÇA

## Falha de segurança no Android permite desvio de dados sigilosos

Bug foi encontrado na biblioteca Play Core pela startup de segurança Oversecured; senhas e números de cartões de crédito de dentro dos aplicativos puderam ser acessados

Vinicius Szafran, editado por Fabiana Rolfini 28/08/2020 16h04

Um aplicativo malicioso no mesmo dispositivo pode explorar essa vulnerabilidade ao injetar módulos maliciosos em outros apps que dependem da biblioteca, assim conseguindo acessar dados sigilosos dos usuários, como senhas e números de cartões de crédito de dentro dos aplicativos.

# Segurança

## Aplicativos de combate à Covid-19 apresentam falhas de segurança e privacidade, diz estudo

Por: João Victor Escovar /// 19 de maio de 2020

*Levantamento feito por especialistas da Internet Lab, centro de pesquisa em direito e tecnologia, analisou plataformas governamentais ligadas à pandemia e detectou riscos aos dados dos usuários*

O aplicativo do SUS foi medido com uma categoria de exposição “baixa”, uma “média” e duas “altas”.

Já o Auxílio Emergencial, da Caixa, só foi avaliado em dois quesitos por questões técnicas, obtendo exposição “alta” em “transparência” e “média” em “segurança”.

# Segurança

Home → TECNOLOGIA → Após polêmica, Zoom corrige a maior falha de segurança do aplicativo

TECNOLOGIA

## Após polêmica, Zoom corrige a maior falha de segurança do aplicativo

Aplicativo para a realização de videoconferências volta atrás e vai garantir que todos os usuários tenham o mesmo nível de segurança

Por **Rodrigo Loureiro**
Publicado em: 18/06/2020 às 10h20
Tempo de leitura: **2 min**

Vale lembrar que o número de usuários mensais ativos da Zoom **passou de 10 milhões em dezembro de 2019 para mais de 300 milhões**. Isso graças aos efeitos de isolamento social da pandemia do novo coronavírus. Tal crescimento fez com que as ações da companhia chegassem a triplicar de valor desde o começo do ano.

# Segurança

FOLHA INFORMAÇÕES · ELEIÇÕES 2020 · TEXTOS LIBERADOS

## Ataque de hackers no sistema do TSE não viola segurança da eleição

Bancos de dados acessados por invasores não têm nenhuma relação com a votação

Barroso afirmou que não há indícios de que o ataque tenha provocado vazamentos de dados de ex-ministros do TSE e de servidores. Segundo ele, a divulgação de dados de hoje foi resultado de um ataque que partiu de Portugal e foi realizado em 23 de outubro.

# Segurança

FOLHA INFORMAÇÕES · ELEIÇÕES 2020 · TEXTOS LIBERADOS

## Ataque de hackers no sistema do TSE não viola segurança da eleição

Bancos de dados acessados por invasores não têm nenhuma relação com a votação

"Foram vazadas informações antigas e irrelevantes sobre ministros aposentados e antigos funcionários do TSE. Foi um vazamento sem nenhuma relevância ou consequência para o processo eleitoral", disse.

# Segurança

10 falhas em 2020

1. iFood – Brecha de segurança expõe dados do aplicativo
2. EasyJet – Ciberataque expõe dados de 9 milhões de clientes
3. Natura – 50 mil clientes Natura tem informações vazadas na web
4. Honda Motor – Empresa de automóveis suspende sua produção
5. Samsung – Falha de segurança permite invasão de celulares Galaxy desde 2014
6. Zoom – Problemas de privacidade e invasões na plataforma de videoconferências
7. Grupo Energisa – Invasor teria criptografado servidores e pedido R$ 5 mi
8. Twitter – Hackers postam fraude no Twitter de bilionários e famosos
9. Nubank – Dados de correntistas do Nubank estavam disponíveis no Google
10. Austrália – Onda de ataques cibernéticos ao governo

# Firewalls

- **Contexto**

  Os firewalls podem ser softwares, ou um dispositivo físico, de hardware com um software para realizar esse gerenciamento.

  Os firewalls normalmente operam na camada 2 e 3 do modelo OSI, ou na camada de transporte do TCP/IP.

  Isso significa que os firewalls **não irão analisar explicitamente o conteúdo do pacote**, porém, ele é capaz de executar regras de acordo com o protocolo e porta.

# Firewalls

- **Contexto**

  Vimos que podemos aplicar diversos tipos de filtros de pacotes utilizando firewalls, porém, para computadores pessoais, normalmente não é necessário o uso de regras muito complexas, sendo que as interfaces padrões de firewall nativa já são o suficiente.

  Porém, ao lidar com **servidores**, esse cenário muda, pois tratam-se de máquinas muito mais expostas e que requerem uma atenção maior. Durante essa aula iremos abordar **o uso do firewall padrão do Linux**. Provavelmente, vocês precisarão lidar com esse tipo de ambiente no dia-a-dia, por exemplo, para liberar o tráfego em uma porta em um servidor, para tornar sua aplicação disponível.

# Firewalls

- **Ping e Traceroute**

  Antes de iniciar a nossa prática, vamos ver dois comandos bastante populares, o **Ping** e o **Traceroute**.

  Esses comandos rodam tanto no *Linux* quanto no *MacOS*.

# Firewalls

- **Ping**

  O primeiro é o **ping** (*Packet InterNet Grouper*), esse comando é utilizado para medir o tempo de resposta da conexão do computador com outros dispositivos da rede local ou internet, assim como se o outro dispositivo está acessível ou não.

  O comando ping utiliza um protocolo próprio também, o **ICMP** - **Protocolo de mensagens de controle de rede**, enviando pequenos pacotes de dados e calculando o tempo que demora para a outra parte responder. Esse delay da resposta é denominado **latência** e, quanto menor o tempo/latência, melhor (as pessoas que jogam online sabem a importância disso).

# Firewalls

- **Ping**

  O ping pode ser útil para diagnosticar problemas de redes em computadores pessoais ou em servidores, por exemplo, se quisermos saber se um computador tem acesso a outro, podemos disparar esse comando contra o outro e ver se há resposta.

  Podemos comparar o comando como um jogo de "ping e pong", onde ao bater a bola de um lado, o ping, a bola realiza o trajeto até o outro lado, onde é rebatida e, então, temos sua resposta, o "pong".

  A maioria dos sistemas operacionais já possuem o comando instalado, para instalá-lo, basta executá-lo, rode o comando: **ping + o endereço do alvo**.

# Firewalls

- **Ping**

  **$ ping google.com**

  No resultado do comando, a primeira coisa legal que podemos ver é o IP correspondente àquele ***DNS***.

  Para localizar o alvo na rede o comando precisa que o ***Domain Name*** seja resolvido, ou seja, ele precisa traduzir aquele endereço para o seu endereço IP. Na resposta do comando também é indicado o tempo de resposta em milissegundos e se houve ou não perda de pacotes no caminho.

# Firewalls

- **Ping**

  **$ ping google.com**

  O parâmetro **-c** define quantos pacotes queremos enviar, por exemplo:

  **$ ping -c 4 google.com**

  Assim enviaremos 4 pacotes e, então, nos é mostrado o resumo desses envios.

# Firewalls

- **Traceroute**

  Outra ferramenta interessante é o **traceroute**.

  O traceroute é uma ferramenta que permite descobrir o caminho feito pelos pacotes desde a sua origem até o seu destino.

  Ele é utilizado para **testes, medidas e gerenciamento de rede**. Com ele é possível detectar falhas no caminho de um pacote e identificar onde está acontecendo a falha, por exemplo, **se é um roteador da nossa rede ou outro dispositivo**.

# Firewalls

- **Traceroute**

  Para executá-lo é bem simples também:

  **$ traceroute google.com**

  Vamos relembrar os conceitos da aula passada e perceber como os nossos pacotes trafegam por diversos dispositivos para chegar até seu destino, relembrando o conceito de redes de computadores. Cada endereço IP é um dispositivo por onde um pacote passou e essa rota é diferente dependendo de onde ele será executado, executem vocês também e vejam a saída do comando.

# Firewalls

- **Traceroute**

  Outro conceito importante para relembrarmos é o de um endereço de IP interno e externo, ou válido.

  Os IPs internos, são utilizados na nossa LAN, ou seja, são válidos somente na nossa rede local, isso porque, como vimos na aula anterior, temos um limite de dispositivos utilizando IPv4. Desse modo, não necessariamente todo dispositivo da rede precisa possuir um IP para internet, somente o roteador/modem da operadora precisa ter um IP "válido" para a internet.

  **No caso, os IPs que iniciam com 192.168.x.x são dispositivos locais da rede, como roteadores na nossa rede por onde o pacote passa.**

# Firewalls

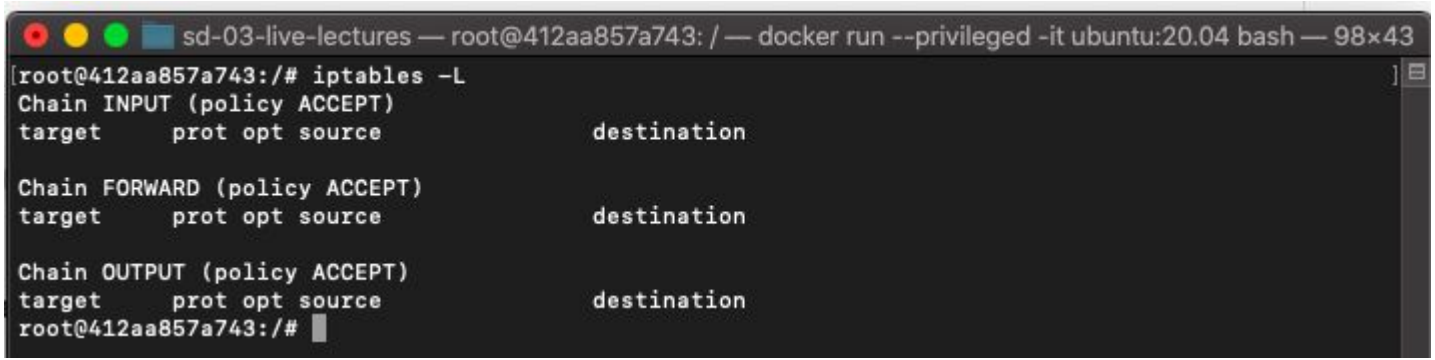
- **Lista de comandos**

```
$ brew cask install docker
$ docker ps
$ docker run --privileged -it ubuntu:20.04 bash
$ apt-get update && apt-get install iputils-ping
$ apt-get update && apt-get install traceroute
$ apt-get update && apt-get install iptables
```

# Mão na massa


- **Iptables**

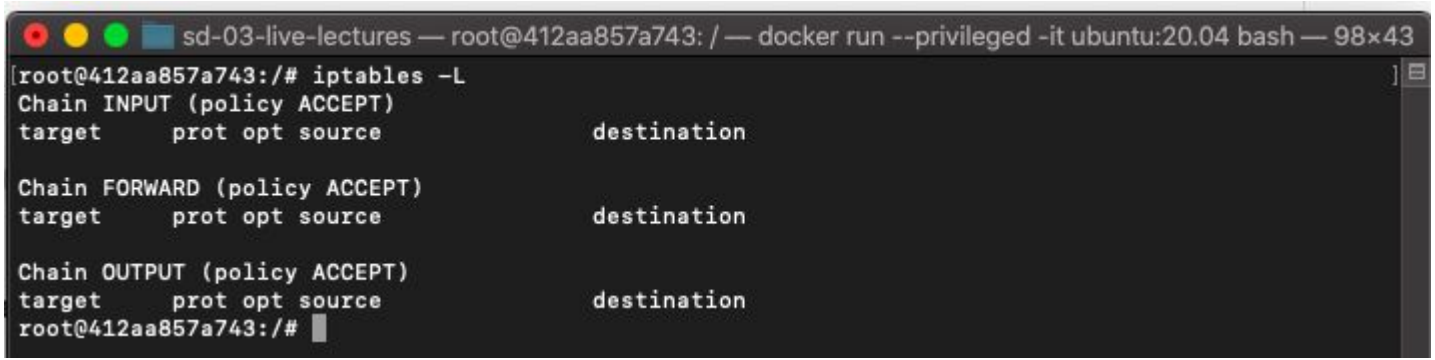
Para consultar as regras já configuradas, usaremos o comando **iptables -L**:



# Iptables

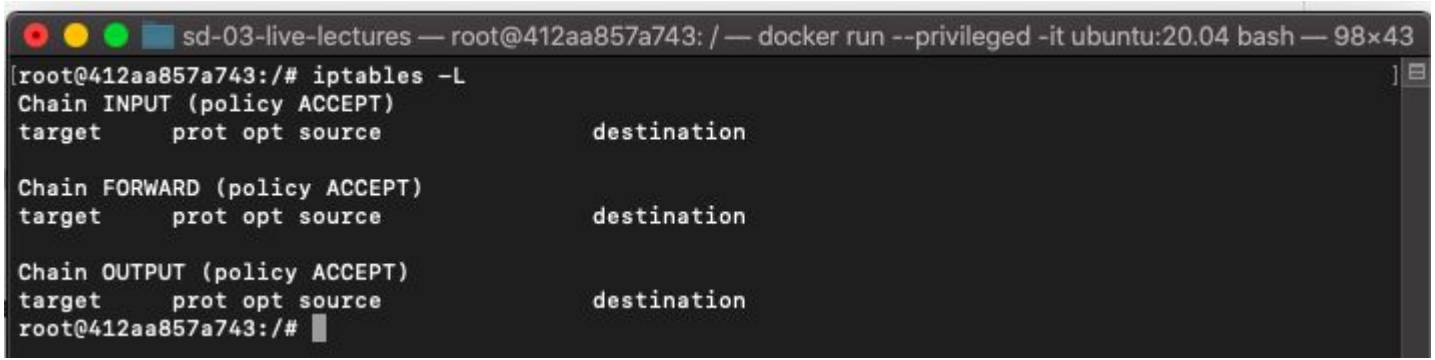
- Cada bloco desses representam uma "**cadeia**" de regras, em **INPUT** temos as regras de **entrada**, em **OUTPUT** temos as de **saídas** e em **FORWARD** as regras sobre os pacotes que são **encaminhados**.

  Basicamente, cada cadeia dessas controla as regras sobre os pacotes que entram, saem ou são encaminhados da nossa interface/placa de rede.



# Iptables

- Percebam que por padrão nossas máquinas não possuem **nenhuma regra configurada** e por padrão a política é de **aceitar qualquer pacote que não tenha uma regra definida para fazer o contrário**, conforme podemos ver em ***policy ACCEPT***.
- Normalmente, em servidores, a política padrão **é negar e então definimos quais as regras para permitir o tráfego**, dessa forma conseguimos ter mais segurança sobre elas.



# Iptables

- Vamos ver o processo de criação de uma regra:

  Sabemos que conseguimos acessar a nossa máquina localmente utilizando **localhost** ou **127.0.0.1**, vamos executar um ping:

```
$ ping 127.0.0.1
```

# Iptables

- Vamos ver o processo de criação de uma regra:

  Sabemos que conseguimos acessar a nossa máquina localmente utilizando **localhost** ou **127.0.0.1**, vamos executar um ping:

  $ ping 127.0.0.1

  Estamos tendo retorno dos pacotes **normalmente**. Vamos criar uma **regra** para **bloquear** esse tipo de requisição a nossa máquina, impedindo que nossa máquina receba "**pings**".

# Iptables

- Para criar uma regra com o "iptables" é bem simples, chamamos o comando **iptables**:

  I) Informamos que queremos acrescentar uma nova regra com a ***flag* -A** na cadeia de **INPUT**:

  **$ iptables -A INPUT ...**

  (Lembre-se que os comandos são *case-sensitive*, então temos que nos atentar se é em caixa alta ou não.)

# Iptables

- Para criar uma regra com o "iptables" é bem simples, chamamos o comando **iptables**:

  II) Em seguida definimos **qual o protocolo** que queremos realizar nossa **regra**, como vimos, o **ping** utiliza o protocolo **ICMP**, vamos usar o parâmetro **-p**:

  **$ iptables -A INPUT -p icmp ...**

  (Lembre-se que os comandos são *case-sensitive*, então temos que nos atentar se é em caixa alta ou não.)

# Iptables

- Para criar uma regra com o "iptables" é bem simples, chamamos o comando **iptables**:

  III) Outro parâmetro que podemos especificar é qual o tipo da mensagem **icmp** que queremos bloquear, vamos bloquear o **echo request**, que nada mais é que **o pacote enviado pelo comando ping**:

  **$ iptables -A INPUT -p icmp --icmp-type echo-request ...**

  (Lembre-se que os comandos são *case-sensitive*, então temos que nos atentar se é em caixa alta ou não.)

# Iptables

- Para criar uma regra com o "iptables" é bem simples, chamamos o comando **iptables**:

  IV) E, por último, vamos falar qual **ação** deverá ser realizada com os pacotes que "caírem" nesse filtro. Podemos aceitar: **ACCEPT**, rejeitar: **REJECT** ou "largá-lo"/"dropá-lo": **DROP**. No caso, não queremos aceitar, então vamos dar um **DROP**. O parâmetro para passar à ação é **"-j"**:

  **$ iptables -A INPUT -p icmp --icmp-type echo-request -j DROP**

  (Lembre-se que os comandos são *case-sensitive*, então temos que nos atentar se é em caixa alta ou não.)

# Iptables

- Para criar uma regra com o "iptables" é bem simples, chamamos o comando **iptables**:

  Poderíamos utilizar o **REJECT** também, a diferença é que o **REJECT** irá "**responder**" com um **erro** àquela requisição. Já com o **DROP** ele simplesmente irá descartar o pacote como se nada tivesse acontecido.

  **$ iptables -A INPUT -p icmp --icmp-type echo-request -j REJECT**

  (Lembre-se que os comandos são *case-sensitive*, então temos que nos atentar se é em caixa alta ou não.)

# Iptables

- Diferente do DROP, o REJECT irá responder com uma **falha** para àquele protocolo específico.

- Ele estará indicando que o pacote foi rejeitado pelo firewall numa resposta como "Oi, aqui é o firewall, estou rejeitando seu pacote, valeu", diferente do DROP que simplesmente irá descartar o pacote e o remetente ficará sem saber o que houve.

- Então o REJECT é melhor do que o DROP? ou DROP é melhor?

# Iptables

- Não tem um melhor, ambos têm diferentes consequências. Por exemplo, **se você quiser de fato dificultar a vida de um "atacante" na sua rede, o DROP pode ser melhor**, pois ele não saberá o porquê dele não estar recebendo a resposta, pode ser por N motivos. Agora com o REJECT, você entrega que é um bloqueio de propósito, provavelmente feito por um firewall, identificando inclusive o IP dele.

- Porém, **se for o bloqueio para impedir o tráfego interno, por exemplo, talvez seja mais interessante responder com uma falha**, de modo que o remetente não continue a tentar sem saber o que aconteceu com o pacote, pois como o DROP não dará uma resposta, poderá ter acontecido N coisas no caminho e o remetente pode inclusive achar que foi uma falha de conexão.

# Iptables

- As regras no **Iptables** são executadas de fato como uma "**cadeia**", isso significa que elas vão sendo **executadas em sequência** e de modo complementar.

- O próximo exemplo vai abordar esse conceito.

# Iptables

- Imagina que queremos **evitar** um ***DDoS*** (ataque de negação de serviço) em nossa máquina, porém, ainda quero que seja possível "**pingá-la**", para conseguir testar se ela está ok caso eu precise algum dia.

- Dessa forma, queremos criar a seguinte política para os protocolos "**icmp**":

  **"Aceitarei pacotes a cada 10 segundos, caso esteja dentro desse tempo eu aceito, caso contrário eu rejeito ou "dropo" esse pacote".**

# Iptables

- Vamos reescrever novamente um **filtro** para o protocolo **icmp**, mais especificamente para bloquear o ping, ou seja o echo-request do icmp:

```
$ iptables -A INPUT -p icmp --icmp-type echo-request ...
```

A ideia é: "**aceitar um pacote a cada 10 segundos e bloquear os demais**". Vamos criar a primeira regra, aceitar a cada 10 segundos.

# Iptables

- Vamos reescrever novamente um **filtro** para o protocolo **icmp**, mais especificamente para bloquear o ping, ou seja o echo-request do icmp:

```
$ iptables -A INPUT -p icmp --icmp-type echo-request ...
```

Para isso, conseguimos criar um "**limite**" para que pacotes caiam em determinada regra, vamos habilitar o limite com o parâmetro **-m**, passando **limit** e então definindo o **limit** com o parâmetro **--limit**:

```
... -m limit --limit ...
```

# Iptables

- Vamos reescrever novamente um **filtro** para o protocolo **icmp**, mais especificamente para bloquear o ping, ou seja o echo-request do icmp:

```
$ iptables -A INPUT -p icmp --icmp-type echo-request -m limit --limit ...
```

Para definir o limite, podemos utilizar algumas opções, como **minute**, **second** e **hour**. Como queremos **a cada 10 segundos**, vamos utilizar o parâmetro **minute**, dizendo que a cada minuto queremos aceitar 6 pacotes (60 segundos / 6 pacotes, 1 pacote a cada 10 segundos). O comando ficará assim:

```
$ iptables -A INPUT -p icmp --icmp-type echo-request -m limit --limit 6/minute -j ACCEPT
```

# Iptables

- Ok, mas esse comando não ta fazendo nada, né?! Todo os "**pings**" ainda estão respondendo.

- Isso porque a *"policy default"* é permitir. Vamos então "**encadear**" uma nova regra, essa nova regra será executada quando o pacote não cair na primeira, de fato um encadeamento. Sendo assim, chegarão nela todos os pacotes que estiverem fora do intervalo dos 10 segundos que definimos anteriormente.

- Para criá-la será simples, precisamos apenas de uma regra que rejeite ou descarte qualquer pacote vindo de um ping:

$ iptables -A INPUT -p icmp --icmp-type echo-request -j **REJECT**

# Iptables

- Vamos ver a nossa cadeia de regras criadas:

```
root@e4daa0940469:/# iptables -L                                Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     icmp --  anywhere             anywhere             icmp echo-request limit: avg 6
/min burst 5
REJECT     icmp --  anywhere             anywhere             icmp echo-request reject-with
icmp-port-unreachable

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
root@e4daa0940469:/#
```

# Iptables

- Percebam que elas ficaram na sequência em que criamos, de fato **encadeadas**. Quando damos **um attach com a flag -A**, falamos ao **iptables** para adicionar a **nova regra** na sequência do encadeamento.

- Agora vamos analisar o nosso **ping**. Percebam que vários deles são rejeitados até que um seja aceito, por padrão o ping é disparado a cada segundo, então temos uma proporção de 9 rejeitados para 1 permitido.

Vamos aumentar a taxa de aceitação para visualizar isso melhor.

# Iptables

- Como temos duas regra não podemos simplesmente apagar a de "ACCEPT" e recriá-la com a nova taxa, **se não iremos quebrar nosso encadeamento**, pois a sequência irá mudar e todos os pacotes serão rejeitados.

- Então devemos recriá-las na ordem. Existem comandos que permitem que façamos isso de uma maneira mais precisa, porém, não iremos entrar nessa complexidade nessa aula. Vamos então deletar todas as regras e recriá-las conforme nosso novo objetivo.

# Iptables

- Para deletar todas as regras podemos utilizar o comando **flush**:

```
$ iptables --flush
$ iptables -L
```

Percebam que não temos mais nenhuma regra criada, vamos recriar a de "**ACCEPT**". Vamos criar uma regra que seja capaz de aceitar um ping **a cada 2 segundos**. Para isso, precisaremos definir o **limit** para quanto?

```
$ iptables -A INPUT -p icmp --icmp-type echo-request -m limit --limit 30/minute -j ACCEPT
```

# Iptables

- Agora vamos criar nossa regra para rejeitar os demais pacotes.

```
$ iptables -A INPUT -p icmp --icmp-type echo-request -m limit --limit 30/minute -j ACCEPT

$ iptables -A INPUT -p icmp --icmp-type echo-request -j REJECT
```

Vamos ver o ping, percebam que como os pings são disparados a cada segundo, temos praticamente um **simetria de pings aceitos e rejeitados**.

# Fail2ban e Iptables

- Vimos no conteúdo sobre a ferramenta **fail2ban**, ela faz exatamente o que acabamos de fazer no **iptables**, porém, de acordo com **regras próprias**.

  Por default ela é capaz de fazer alguns bloqueios conforme vimos. Ela funciona analisando os logs dos pacotes e então realizando regras no iptables de acordo com essa análise.

- Podemos, por exemplo, a partir dos logs do nosso servidor como um Apache ou nginx, **bloquear um IP que esteja realizando um comportamento estranho**.

# Fail2ban e Iptables

- Um exemplo de comportamento estranho seria um mesmo IP estar recebendo vários **erros 404**, isso pode significar que na verdade trata-se de um **robô** varrendo nossas páginas em busca de uma falha de segurança.

- Com o **fail2ban**, podemos definir uma regra que se um mesmo IP receber **10 erros 404 em 10 segundos por exemplo**, bloquearemos aquele IP por 5 minutos.

# Fail2ban e Iptables

- Dessa forma o **fail2ban** irá pegar o endereço de IP e então criará uma **regra** no **iptables** especificamente para ele pelo tempo estabelecido e, então, removerá a regra posteriormente.

- Percebam o quão isso é poderoso e quantas medidas de **segurança** podemos implementar, tanto de maneira **proativa**, antes da situação ocorrer e até mesmo de maneira **reativa**, que não é o ideal mas estamos sujeitos, entretanto, conseguimos rapidamente agir em uma situação adversa assim que ela for identificada utilizando tais ferramentas.

TRYBE

betrybe.com

Obrigado!