Preparação ambiente para teste...

Atenção: Caso você utilize o sistema MacOS para dar aula, será estritamente necessário que você tenha o docker instalado na sua máquina. Caso você utilize Linux, não se preocupe com essa preparação, certifique-se apenas de que você tem o iptables instalado corretamente na sua máquina.

Caso você não tenha o docker instalado, execute o comando abaixo:

$ brew install --cask docker
Em seguida, execute o aplicativo do docker, que foi instalado na sua máquina através do brew cask, dê privilegio para o docker passando a sua senha e pronto! O docker já está rodando, para testar execute o comando abaixo:

$ docker ps
Tudo certo até então. Agora rode o comando abaixo para subir uma máquina Linux com a distro ubuntu 20.4. Estamos passando a tag --privileged para que você consiga executar os comando da aula sem problemas de permissão de host:

$ docker run --privileged -it ubuntu:20.04 bash
Beleza, com a máquina Linux rodando, execute o comando abaixo para instalar o iptables, o ping e o traceroute:

$ apt-get update && apt-get install iputils-ping traceroute iptables
Antes de iniciar a aula, certifique-se que você possui funcionando as seguintes tecnologias/ferramentas:

ping;

traceroute;

iptables.

Para certificar, rode os comandos abaixo e verifique se os resultados estão ok:

$ ping google.com

$ traceroute google.com

$ iptables -L
Atenção: Se o ping tiver dado 100% de packet loss, reinicie o docker (confia em mim 😉).

Além disso, prepare duas abas de terminal com acesso a esse container criado. Para abrir a segunda aba no mesmo container, colete a resposta de:

docker ps -aq
A cadeia de caracteres que mostrou na tela é o id do container. Em seguida, rode o seguinte comando, substituindo [CONTAINER_ID] pelo id achado anteriormente:

docker exec -it [CONTAINER_ID] bash
Teste os comandos listados acima para garantir que está no mesmo container.

Nessa aula faremos a demonstração do uso de um firewall bastante comum, o iptables, nativo na maioria das distros Linux. Por isso a necessidade de subir uma máquina Linux no docker para usuários de MacOS.

Como o uso mais avançado de firewalls e suas regras são mais comuns em servidores do que em máquinas pessoais, sendo estas mais comum o uso de Linux ou Windows. Vale ressaltar que no dia-a-dia, provavelmente, eles precisarão saber lidar com firewalls Linux ou até mesmo Windows, por serem SOs mais utilizados em servidores.

Prática!

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.

Então vamos lá.

Vamos fazer alguns exemplos, primeiramente vamos aprender como consultar as regras já configuradas, para isso usaremos o comando iptables -L:

$ iptables -L
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.

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.

Vamos ver o processo de criação de uma regra:

Acabamos de aprender sobre o comando ping, vamos utilizá-lo aqui.

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".

Para criar uma regra com o "iptables" é bem simples, chamamos o comandoiptables:

OE Mantenha um terminal executando o ping e abra outro para digitar os próximos comandos. Digite, sem executar:

$ iptables
Então informamos que queremos acrescentar uma nova regra com a flag -A na cadeia de INPUT:

OE Complete o comando digitado anteriormente com o abaixo, sem executar:

$ ... -A INPUT
Lembre-se que os comandos são case-sensitive, então temos que nos atentar se é em caixa alta ou não.

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

OE Complete o comando digitado anteriormente com o abaixo, sem executar:

$ ... -p icmp
Um outro parâmetro que podemos utilizar é especificar 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:

OE Complete o comando digitado anteriormente com o abaixo, sem executar:

$ ... --icmp-type echo-request
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":

OE Complete o comando digitado anteriormente com o abaixo, sem executar:

$ ... -j DROP
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.

Vamos executar nosso comando para criação da regra.

OE Execute o comando. O iptables requer permissões especiais para serem executados, caso seu usuário não possua, adicione sudo aos comandos e explique para turma a importância dele. O comando completo deverá ficar assim:

$ iptables -A INPUT -p icmp --icmp-type echo-request -j DROP
Percebam que ao criar a regra o nosso ping para de responder, agora todos os pacotes estão sendo descartados pelo nosso firewall.

OE Pare o ping e mostre a quantidade de pacotes transmitidos x quantidade de pacotes recebidos. Mostre também o packet loss. Explique para os estudantes que os pacotes foram perdidos devido à regra que criamos.

Vamos listar as regras com o parâmetro -L, assim podemos ver a regra que acabamos de criar:

$ iptables -L
Esse tipo de filtro pode dificultar a descoberta da nossa máquina na rede, por exemplo, e até um DDoS (que utilize o comando ping no ataque), utilizando esse protocolo.

Vamos ver a diferença do DROP e do REJECT.

Primeiramente vamos adicionar um parâmetro ao nosso ping para melhorar nosso output, o parâmetro -O, com ele conseguiremos ver também os outputs dos pacotes que "falharem".

OE Rode o comando (mantenha o comando rodando durante todo o experimento com o iptables para a experiencia ser mais legal):

$ ping -O 127.0.0.1
Nota: Esse comando, com a flag -O só é necessário no Linux, no MacOS esse comportamento é padrão.

Agora, vamos deletar a regra criada. Para isso podemos executar o mesmo comando, porém alterando o -A (attach) por -D (delete).

OE Execute no terminal:

$ iptables -D INPUT -p icmp --icmp-type echo-request -j DROP
Perceba que nosso ping voltou a responder. Vamos listar as regras novamente:

OE Execute o comando:

$ iptables -L
Vamos então recriar a nossa regra mas dessa vez utilizando REJECT.

OE Execute no terminal:

$ iptables -A INPUT -p icmp --icmp-type echo-request -j REJECT
Agora reparem no nosso ping como recebemos uma resposta de falha, com o parâmetro -O ela é exibida.

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?

Não tem um melhor, ambos tem 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.

Vamos ver o que mais conseguimos fazer com as regras mas, antes, vamos excluir a regra criada e fazer uma pausa para tirarmos dúvidas também.

OE Execute no terminal iptables -D INPUT -p icmp --icmp-type echo-request -j REJECT

Vamos listar as regras novamente:

OE Execute o comando iptables -L.

OE Faça uma pausa para dúvidas.

Lembram que eu falei que cada tabela dessa é uma "cadeia" de regras?!

Então, as regras 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.

Imagina que queremos evitar um DDoS em nossa máquina, porém, ainda quero que seja possível "pingá-la", para eu conseguir testar se ela está ok caso eu precise algum dia. Dessa forma, eu quero 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".

Vamos ao comando!

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

OE Digite, sem executar:

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