## ASIST – PL7

## Na máquina virtual Linux:

1. Filtragem
2. Criar um script para definir regras de filtragem na cadeia que analisa o tráfego que chega ao sistema (sugestão: man iptables). Deve bloquear todo o tráfego com exceção daqueles que necessita para aceder ao sistema (NOTA: não se encontra incluída neste aceder o cliente vSphere;
3. Qual o ou os comandos que pode utilizar para o efeito?

sudo mkdir /etc/network/firewall

sudo nano /etc/network/firewall/firewall.sh

sudo chmod +x /caminho/para/seu/script/firewall.sh

Crie um script de shell (por exemplo, firewall.sh) e adicione as seguintes linhas para bloquear todo o tráfego, exceto aqueles necessários para acessar o sistema:

#!/bin/bash

# Limpar regras existentes

iptables -F

# Bloquear todo o tráfego por padrão

iptables -P INPUT DROP

iptables -P FORWARD DROP

iptables -P OUTPUT ACCEPT

# Permitir tráfego de loopback

iptables -A INPUT -i lo -j ACCEPT

iptables -A OUTPUT -o lo -j ACCEPT

# Permitir tráfego necessário para acessar o sistema (substitua os protocolos e portas conforme necessário)

iptables -A INPUT -p tcp --dport 22 -j ACCEPT # Permitir SSH

iptables -A INPUT -p icmp -j ACCEPT # Permitir ICMP

# Adicione outras regras conforme necessário para permitir ou bloquear tráfego adicional

# Salvar as regras

service iptables save

service iptables restart

1. Valide que consegue aceder através do ou dos protocolos que colocou como exceção.

De forma a garantir que as regras foram definidas corretamente podemos testar a conectividade através dos comandos:

ssh admin@ 10.9.10.76

ping 10.9.10.76

1. Quais as ações que poderão ser indicadas para o parâmetro próprio quando se trata de bloquear? E quais as consequências práticas delas?

Algumas das ações comuns para bloquear são -j DROP e -j REJECT. Aqui estão as explicações para cada uma delas:

-j DROP:

Ação: O pacote é silenciosamente descartado, sem enviar nenhuma resposta de volta ao remetente.

Consequências Práticas:

O remetente não recebe nenhuma confirmação de que o pacote foi descartado.

Pode levar a atrasos ou timeouts no lado do remetente, pois não há indicação imediata de que o pacote foi bloqueado.

Exemplo:

# Criar uma regra para bloquear tráfego na porta 22 (SSH)

iptables -A INPUT -p tcp --dport 22 -j DROP

-j REJECT:

Ação: O pacote é descartado, e é enviada uma resposta ICMP ao remetente indicando que o acesso foi proibido.

Consequências Práticas:

O remetente recebe uma notificação de que o pacote foi rejeitado.

Pode resultar em tempos de resposta mais rápidos para o remetente, pois ele é informado de imediato que o acesso foi negado.

Exemplo:

# Criar uma regra para bloquear tráfego na porta 80 (HTTP) e enviar uma resposta ICMP de rejeição

iptables -A INPUT -p tcp --dport 80 -j REJECT

1. O acesso fica bloqueado para o IPv4, para o IPv6 ou para ambos?

Os exemplos fornecidos anteriormente são destinados ao IPv4, pois utilizam o comando iptables para configurar as regras de filtragem. Para lidar com o IPv6, você precisaria utilizar o comando ip6tables e criar regras específicas para o IPv6.

1. Qual é o comportamento por omissão?

No iptables, as políticas padrão são instruções que determinam o que acontece com o tráfego que não é afetado por nenhuma regra explícita. Existem três cadeias principais: INPUT, FORWARD e OUTPUT.

INPUT:

A cadeia INPUT lida com pacotes que chegam à própria máquina.

A política padrão para a cadeia INPUT determina o que acontece com pacotes que não correspondem a nenhuma regra explícita nesta cadeia.

FORWARD:

A cadeia FORWARD lida com pacotes que são roteados através da máquina, mas não são destinados à própria máquina.

Se a máquina está configurada como um roteador, a política padrão para a cadeia FORWARD é relevante.

OUTPUT:

A cadeia OUTPUT lida com pacotes gerados pela própria máquina e enviados para fora.

A política padrão para a cadeia OUTPUT determina o que acontece com pacotes gerados pela própria máquina que não correspondem a nenhuma regra explícita nesta cadeia.

No exemplo que forneci:

Para a cadeia INPUT e FORWARD:

A política padrão é DROP. Isso significa que, se um pacote não corresponder a nenhuma regra explícita na cadeia INPUT ou FORWARD, ele será descartado silenciosamente (DROP).

Para a cadeia OUTPUT:

A política padrão é ACCEPT. Isso significa que, se um pacote gerado pela própria máquina não corresponder a nenhuma regra explícita na cadeia OUTPUT, ele será aceito (ACCEPT).

Essa configuração segue uma abordagem padrão de "negar por padrão" para o tráfego de entrada e encaminhamento, o que significa que todo o tráfego é bloqueado a menos que haja uma regra específica permitindo-o. Para o tráfego de saída, a abordagem é "permitir por padrão", o que significa que o tráfego gerado pela própria máquina é permitido, a menos que haja uma regra específica bloqueando-o.

1. Qual será o comportamento por omissão mais correto? Porquê?

O comportamento por omissão mais correto depende dos requisitos de segurança, mas bloquear por padrão (DROP) é uma prática comum para criar um ambiente mais seguro.

1. Acrescente mais algumas regras para permitir ou bloquear alguns pedidos, como por exemplo, aceitar ligações por SSH apenas ao endereço IP dos elementos do grupo, responder a pedidos ICMP apenas se for proveniente dos computadores dos elementos do grupo, só permitir duas sessões SSH em simultâneo por hora, etc.

#!/bin/bash

# Limpar regras existentes

iptables -F

# Definir políticas padrão (DROP para INPUT e FORWARD, ACCEPT para OUTPUT)

iptables -P INPUT DROP

iptables -P FORWARD DROP

iptables -P OUTPUT ACCEPT

# Permitir tráfego de loopback

iptables -A INPUT -i lo -j ACCEPT

iptables -A OUTPUT -o lo -j ACCEPT

# Exceção para SSH (porta 22) apenas a partir de endereços IP específicos

iptables -A INPUT -p tcp --dport 22 -m iprange --src-range IP\_INICIAL\_GRUPO:IP\_FINAL\_GRUPO -j ACCEPT

# Exceção para ICMP (ping) apenas a partir de endereços IP específicos

iptables -A INPUT -p icmp -m iprange --src-range IP\_INICIAL\_GRUPO:IP\_FINAL\_GRUPO -j ACCEPT

# Bloquear o cliente vSphere (substituir pelo IP real do cliente)

iptables -A INPUT -s IP\_DO\_CLIENTE\_VSPHERE -j DROP

# Limitar a duas sessões SSH simultâneas por hora

iptables -A INPUT -p tcp --syn --dport 22 -m recent --name ssh --set

iptables -A INPUT -p tcp --syn --dport 22 -m recent --name ssh --update --seconds 3600 --hitcount 3 -j DROP

iptables -A INPUT -p tcp --dport 22 -j ACCEPT

# Salvar as regras

service iptables save

service iptables restart

1. Scripting
2. Desenvolva um script para avisar o utilizador asist de alterações ao número de objetos existentes na pasta /etc.

sudo nano /usr/local/bin/verificar\_alteracoes.sh

sudo chmod +x /usr/local/bin/verificar\_alteracoes.sh

#!/bin/bash

# Definir o caminho da pasta

pasta="/etc"

# Obter o número inicial de objetos na pasta

num\_objetos\_inicial=$(ls -A "$pasta" | wc -l)

# Função para verificar alterações

verificar\_alteracoes() {

# Obter o número atual de objetos na pasta

num\_objetos\_atual=$(ls -A "$pasta" | wc -l)

# Verificar se houve alterações

if [ "$num\_objetos\_atual" -ne "$num\_objetos\_inicial" ]; then

echo "Atenção: O número de objetos em $pasta foi alterado."

echo "Número inicial: $num\_objetos\_inicial"

echo "Número atual: $num\_objetos\_atual"

else

echo "Não foram encontradas alterações na pasta $pasta."

fi

}

# Executar a função

verificar\_alteracoes

Explicação:

* O script define a variável pasta como o caminho da pasta /etc.
* A variável num\_objetos\_inicial é inicializada com o número de objetos (arquivos e diretórios) existentes na pasta no início.
* A função verificar\_alteracoes compara o número atual de objetos na pasta com o número inicial e exibe uma mensagem se houver alterações.
* O script executa a função verificar\_alteracoes.

1. Desenvolva um script para verificar se:
2. Todos os utilizadores registados na máquina local possuem uma homedir que exista; caso não exista, o script deve criá-la.

sudo nano /usr/local/bin/verificar\_homedirs.sh

sudo chmod +x /usr/local/bin/verificar\_homedirs.sh

#!/bin/bash

# Lista todos os usuários no sistema e itera sobre eles

for usuario in $(cut -f1 -d: /etc/passwd); do

# Obtém o caminho da pasta home do usuário

homedir=$(eval echo ~$usuario)

# Verifica se a pasta home existe

if [ ! -d "$homedir" ]; then

echo "A pasta home do usuário $usuario não existe. Criando..."

mkdir -p "$homedir"

chown $usuario:$usuario "$homedir"

echo "Pasta home criada para o usuário $usuario."

else

echo "A pasta home do usuário $usuario já existe."

fi

done

Explicação:

* O script utiliza um loop para iterar sobre todos os usuários encontrados no arquivo /etc/passwd.
* Para cada usuário, ele obtém o caminho da pasta home usando eval echo ~$usuario.
* Verifica se a pasta home existe (-d "$homedir").
* Se a pasta home não existe, ela é criada usando mkdir -p "$homedir", e as permissões são ajustadas com chown.
* O script exibe mensagens informativas sobre suas ações.

1. Todas as pastas existentes na partição das homedir possuem pelo menos um utilizador registado na máquina local; caso alguma não tenha, deve ser eliminada.

sudo nano /usr/local/bin/verificar\_user\_homedirs.sh

sudo chmod +x /usr/local/bin/verificar\_user\_homedirs.sh

#!/bin/bash

# Diretório da partição das homedirs

particao\_homedirs="/home"

# Itera sobre todas as pastas no diretório de homedirs

for homedir in $particao\_homedirs/\*/; do

# Obtém o nome da pasta (nome do usuário)

usuario=$(basename "$homedir")

# Verifica se o usuário existe no sistema

if id "$usuario" &>/dev/null; then

echo "O usuário $usuario existe. Nada será feito."

else

echo "O usuário $usuario não existe. A pasta será removida."

rm -r "$homedir"

fi

done

Explicação:

* O script utiliza um loop para iterar sobre todas as pastas no diretório de homedirs (/home por padrão).
* Para cada pasta, obtém o nome do usuário a partir do nome da pasta.
* Verifica se o usuário existe no sistema usando o comando id.
* Se o usuário não existe, a pasta é removida usando rm -r.

1. O dono e grupo de todas as pastas existentes na partição das homedir estão atualizados; caso não estejam, devem ser corrigidos.

sudo nano /usr/local/bin/atualizar\_donos\_grupos.sh

sudo chmod +x /usr/local/bin/atualizar\_donos\_grupos.sh

#!/bin/bash

# Diretório da partição das homedirs

particao\_homedirs="/home"

# Dono e grupo desejados

dono\_desejado="usuario"

grupo\_desejado="grupo"

# Itera sobre todas as pastas no diretório de homedirs

for homedir in "$particao\_homedirs"/\*/; do

# Obtém o nome da pasta (nome do usuário)

usuario=$(basename "$homedir")

# Atualiza o dono e grupo da pasta, se necessário

chown -R "$dono\_desejado:$grupo\_desejado" "$homedir"

echo "Dono e grupo atualizados para $dono\_desejado:$grupo\_desejado em $homedir"

done

Explicação:

* script utiliza um loop para iterar sobre todas as pastas no diretório de homedirs (/home por padrão).
* Para cada pasta, obtém o nome do usuário a partir do nome da pasta.
* Utiliza o comando chown para atualizar o dono e grupo da pasta para os valores desejados.

1. Configure o sistema para executar o script todos os dias a uma hora determinada e registar num ficheiro a data de execução e as ações que teve que desencadear.

Primeiramente editamos o arquivo **crontab:**

**crontab -e**

Seguidamente, adicionamos a seguinte linha ao final do arquivo para executar o script todos os dias às 3 da manhã:

0 3 \* \* \* /usr/local/bin/verificar\_alteracoes.sh >>/var/log/verificacao\_homedirs.log2>&1

0 3 \* \* \* /usr/local/bin/verificar\_user\_homedirs.sh >>/var/log/verificacao\_homedirs.log2>&1

0 3 \* \* \* /usr/local/bin/atualizar\_donos\_grupos.sh >>/var/log/verificacao\_homedirs.log2>&1

Explicação:

* 0 3 \* \* \*: Define a hora de execução para as 3:00 AM todos os dias.
* >> /caminho/para/o/seu/arquivo\_de\_log.txt 2>&1: Redireciona a saída padrão e de erro para o arquivo de log especificado.

Reinicie o serviço cron para aplicar as mudanças:

**service cron restart**

Finalmente verificamos o arquivo:

**cat /caminho/para/o/seu/arquivo\_de\_log.txt**