Laboratório - Servidores Linux

# Objetivos

Neste laboratório, você usará a linha de comando do Linux para identificar servidores em execução em um determinado computador.

Parte 1: Servidores

Parte 2: Usando Telnet para testar serviços TCP

# Equipamentos recomendados

=   Máquina virtual Security Workstation.

# Instruções

## Parte 1: Servidores

Servidores são essencialmente programas escritos para fornecer informações específicas mediante solicitação. Os clientes, que também são programas, chegam ao servidor, colocam a solicitação e aguardam a resposta do servidor. Muitas tecnologias diferentes de comunicação cliente-servidor podem ser usadas, sendo as mais comuns as redes IP. Este laboratório concentra-se em servidores e clientes baseados em rede IP.

### Etapa 1: Acesse a linha de comando

a.     Faça logon na Security Workstation VM como **analyst**, usando a senha **cyberops**. A conta **analyst** é usada como exemplo de conta de usuário neste laboratório.

b.     Para acessar a linha de comando, clique no ícone de **terminal** localizado no Dock, na parte inferior da tela da VM. O emulador de terminal é aberto.



### Etapa 2: Exibir os serviços em execução no momento.

Muitos programas diferentes podem ser executados em um determinado computador, especialmente em um computador executando um sistema operacional Linux. Muitos programas são executados em segundo plano para que os usuários não detectem imediatamente quais programas estão sendo executados em um determinado computador. No Linux, os programas em execução também são chamados de *processos*.

**Observação**: a saída do comando **ps** será diferente porque será baseada no estado da VM Security Workstation.

a.     Use o comando **ps** para exibir todos os programas em execução em segundo plano:

[analyst@secOps ~]$ **sudo ps –elf**

[sudo] password for analyst:

F S UID        PID  PPID  C PRI  NI ADDR SZ WCHAN  STIME TTY          TIME CMD

4 S root         1     0  0  80   0 -  2250 SyS\_ep Feb27 ?        00:00:00 /sbin/init

1 S root         2     0  0  80   0 -     0 kthrea Feb27 ?        00:00:00 [kthreadd]

1 S root         3     2  0  80   0 -     0 smpboo Feb27 ?        00:00:00 [ksoftirqd/0]

1 S root         5     2  0  60 -20 -     0 worker Feb27 ?        00:00:00 [kworker/0:0H]

1 S root         7     2  0  80   0 -     0 rcu\_gp Feb27 ?        00:00:00 [rcu\_preempt]

1 S root         8     2  0  80   0 -     0 rcu\_gp Feb27 ?        00:00:00 [rcu\_sched]

1 S root         9     2  0  80   0 -     0 rcu\_gp Feb27 ?        00:00:00 [rcu\_bh]

1 S root        10     2  0 -40   - -     0 smpboo Feb27 ?        00:00:00 [migration/0]

1 S root        11     2  0  60 -20 -     0 rescue Feb27 ?        00:00:00 [lru-add-drain]

5 S root        12     2  0 -40   - -     0 smpboo Feb27 ?        00:00:00 [watchdog/0]

1 S root        13     2  0  80   0 -     0 smpboo Feb27 ?        00:00:00 [cpuhp/0]

5 S root        14     2  0  80   0 -     0 devtmp Feb27 ?        00:00:00 [kdevtmpfs]

1 S root        15     2  0  60 -20 -     0 rescue Feb27 ?        00:00:00 [netns]

1 S root        16     2  0  80   0 -     0 watchd Feb27 ?        00:00:00 [khungtaskd]

1 S root        17     2  0  80   0 -     0 oom\_re Feb27 ?        00:00:00 [oom\_reaper]

<some output omitted>

#### Pergunta:

Por que foi necessário executar o **ps** como root (precedendo o comando com **sudo**)?

***Área de Resposta***

b.     No Linux, os programas também podem chamar outros programas. O comando **ps** também pode ser usado para exibir essa hierarquia de processos. Use as opções **--ejH** para exibir a árvore de processos em execução no momento após iniciar o servidor web nginx com privilégios elevados.

**Observação**: As informações do processo para o serviço nginx são realçadas. Seus valores de PID serão diferentes.

[analyst@secOps ~]$ **sudo /usr/sbin/nginx**

[analyst@secOps ~]$ **sudo ps –ejH**

[sudo] password for analyst:

 PID  PGID   SID TTY          TIME CMD

    1     1     1 ?00:00:00 systemd

  167   167   167 ?        00:00:01   systemd-journal

  193   193   193 ?        00:00:00   systemd-udevd

  209   209   209 ?        00:00:00   rsyslogd

  210   210   210 ?        00:01:41   java

  212   212   212 ?        00:00:01   ovsdb-server

  213   213   213 ?        00:00:00   start\_pox.sh

  224   213   213 ?        00:01:18     python2.7

  214   214   214 ?        00:00:00   systemd-logind

  216   216   216 ?        00:00:01   dbus-daemon

  221   221   221 ?        00:00:05   filebeat

  239   239   239 ?        00:00:05   VBoxService

  287   287   287 ?        00:00:00   ovs-vswitchd

  382   382   382 ?        00:00:00   dhcpcd

  387   387   387 ?        00:00:00   lightdm

  410   410   410 tty7     00:00:10     Xorg

  460   387   387 ?        00:00:00     lightdm

  492   492   492 ?        00:00:00       sh

  503   492   492 ?        00:00:00         xfce4-session

  513   492   492 ?        00:00:00           xfwm4

  517   492   492 ?        00:00:00           Thunar

 1592   492   492 ?        00:00:00             thunar-volman

  519   492   492 ?        00:00:00           xfce4-panel

  554   492   492 ?        00:00:00             panel-6-systray

  559   492   492 ?        00:00:00             panel-2-actions

  523   492   492 ?        00:00:01           xfdesktop

  530   492   492 ?        00:00:00           polkit-gnome-au

  395   395   395 ?        00:00:00   nginx

  396   395   395 ?        00:00:00     nginx

  408   384   384 ?        00:01:58   java

  414   414   414 ?        00:00:00   accounts-daemon

  418   418   418 ?        00:00:00   polkitd

<alguma saída omitida >

#### Pergunta

Como a hierarquia de processos é representada por **ps**?

***Área de Resposta***

c.     Como mencionado anteriormente, os servidores são essencialmente programas, muitas vezes iniciados pelo próprio sistema no momento da inicialização. A tarefa executada por um servidor é chamada de *serviço.* De tal forma, um servidor web fornece serviços web.

O comando **netstat** é uma ótima ferramenta para ajudar a identificar os servidores de rede em execução em um computador. O poder do **netstat** reside na sua capacidade de exibir conexões de rede.

**Observação**: sua saída podeser diferente dependendo do número de conexões de rede abertas em sua VM.

Na janela do terminal, digite **netstat**.

[analyst@secOps ~]$ **netstat**

Conexões de Internet ativas (sem servidores)

Proto Recv-Q Send-Q Local Address           Foreign Address         State

tcp        0      0 localhost.localdo:48746 localhost.local:wap-wsp ESTABLISHED

tcp        0      0 localhost.localdo:48748 localhost.local:wap-wsp ESTABLISHED

tcp6       0      0 localhost.local:wap-wsp localhost.localdo:48748 ESTABLISHED

tcp6       0      0 localhost.local:wap-wsp localhost.localdo:48746 ESTABLISHED

tcp6       0      0 localhost.local:wap-wsp localhost.localdo:48744 ESTABLISHED

tcp6       0      0 localhost.localdo:48744 localhost.local:wap-wsp ESTABLISHED

Soquetes de domínio UNIX ativos (sem servidores)

Proto RefCnt Flags       Type       State         I-Node   Path

unix  3      [ ]         DGRAM                    8472     /run/systemd/notify

unix  2      [ ]         DGRAM                    8474     /run/systemd/cgroups-agent<some output omitted>

Como visto acima, **netstat** retorna muitas informações quando usado sem opções. Muitas opções podem ser usadas para filtrar e formatar a saída do **netstat**, tornando-a mais útil.

d.     Use **netstat** com as **opções** —tunap para ajustar a saída do **netstat**.Observe que **netstat** permite que várias opções sejam agrupadas sob o mesmo sinal “**-**“.

As informações para o servidor nginx são realçadas.

[analyst@secOps ~]$ **sudo netstat -tunap**

[sudo] password for analyst:

Conexões de Internet ativas (servidores e estabelecidos)

Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name

tcp        0      0 0.0.0.0:6633            0.0.0.0:\*               LISTEN      257/python2.7

tcp        0      0 0.0.0.0:80              0.0.0.0:\*               LISTEN      395/nginx: master

tcp        0      0 0.0.0.0:21              0.0.0.0:\*               LISTEN      279/vsftpd

tcp        0      0 0.0.0.0:22              0.0.0.0:\*               LISTEN      277/sshd: /usr/bin

tcp6       0      0 :::22                   :::\*                    LISTEN      277/sshd: /usr/bin

udp        0      0 192.168.1.15:68         0.0.0.0:\*                           237/systemd-network

#### Pergunta:

Qual é o significado das opções **—t**, **-u**, **—n**, **—a** e **—p** em **netstat**? (use **man netstat** para responder)

***Área de Resposta***

A ordem das opções é importante para **netstat**?

***Área de Resposta***

Os clientes se conectarão a uma porta e, usando o protocolo correto, solicitarão informações de um servidor. A saída **netstat** acima exibe uma série de serviços que estão atualmente escutando em portas específicas. Colunas interessantes são:

o      A primeira coluna mostra o protocolo de Camada 4 em uso (UDP ou TCP, neste caso).

o      A terceira coluna usa o formato **<ADDRESS:PORT>** para exibir o endereço IP local e a porta em que um servidor específico está acessível. O endereço IP 0.0.0.0 significa que o servidor está atualmente escutando todos os endereços IP configurados no computador.

o      A quarta coluna usa o mesmo formato de soquete **<ADDRESS:PORT>** para exibir o endereço e a porta do dispositivo na extremidade remota da conexão. 0.0.0.0: \* significa que nenhum dispositivo remoto está usando a conexão no momento.

o      A quinta coluna exibe o estado da conexão.

o      A sexta coluna exibe o ID do processo (PID) do processo responsável pela conexão. Ele também exibe um nome curto associado ao processo.

#### Pergunta:

Com base na saída **netstat** mostrada no item (d), qual é o protocolo da Camada 4, o status da conexão e o PID do processo em execução na porta 80?

***Área de Resposta***

Embora os números de porta sejam apenas uma convenção, você pode adivinhar que tipo de serviço está sendo executado na porta 80 TCP?

***Área de Resposta***

e.     Às vezes, é útil cruzar as informações fornecidas pelo **netstat** com **ps**. Com base na saída do item (d), sabe-se que um processo com **PID 395** está vinculado à porta TCP 80. A porta 395 é usada neste exemplo. Use **ps** e **grep** para listar todas as linhas da saída **ps** que contêm **PID 395**. Substitua 395 pelo número PID da sua instância em execução específica do nginx:

[analyst@secOps ~]$ **sudo ps -elf | grep 395**

[sudo] password for analyst:

1 S root       395     1  0  80   0 -  1829    19:33 ?        00:00:00 nginx: master process /usr/bin/nginx

5 S http       396   395  0  80   0 -  1866    19:33 ?        00:00:00 nginx: worker process

0 S analyst   3789  1872  0  80   0 -  1190    19:53 pts/0    00:00:00 grep 395

Na saída acima, o comando **ps** é canalizado através do comando **grep** para filtrar apenas as linhas que contêm o número 395. O resultado é três linhas com quebra de texto.

A primeira linha mostra um processo de propriedade do usuário **root** (terceira coluna), iniciado por outro processo com PID 1 (quinta coluna), em 19:33 (décima segunda coluna)

A segunda linha mostra um processo com PID 396, de propriedade do usuário **http** , iniciado pelo processo 395, às 19:33.

A terceira linha mostra um processo de propriedade do usuário **analista**, com PID 3789, iniciado por um processo com PID 1872, como o comando **grep 395**.

#### Pergunta:

O processo PID 395 é **nginx**. Como isso poderia ser concluído a partir da produção acima?

***Área de Resposta***

O que é **nginx**? Qual é a sua função? (Use o google para saber mais sobre o nginx)

***Área de Resposta***

A segunda linha mostra que o processo 396 pertence a um usuário chamado http e tem o número de processo 395 como seu processo pai. O que isso quer dizer? Isto é um comportamento comum?

***Área de Resposta***

Isso significa que nginx iniciou o processo 396 sob o nome de usuário http. Isso é normal, pois o nginx é executado para cada cliente que se conecta à porta 80 TCP.

Por que a última linha mostra grep 395?

***Área de Resposta***

## Parte 2: Usando Telnet para testar serviços TCP

Telnet é um simples aplicativo de shell remoto. Telnet é considerado inseguro porque não fornece criptografia. Os administradores que optarem por usar o Telnet para gerenciar remotamente dispositivos e servidores de rede exporão as credenciais de login para esse servidor, já que o Telnet transmitirá dados de sessão em texto não criptografado. Embora o Telnet não seja recomendado como um aplicativo de shell remoto, ele pode ser muito útil para testar rapidamente ou coletar informações sobre serviços TCP.

O protocolo Telnet opera na porta 23 usando TCP por padrão. No entanto, o cliente **telnet** permite que uma porta diferente seja especificada. Ao alterar a porta e conectar-se a um servidor, o cliente **telnet** permite que um analista de rede avalie rapidamente a natureza de um servidor específico, comunicando-se diretamente com ele.

**Observação**: É altamente recomendável que o **ssh** seja usado como aplicativo de shell remoto em vez de **telnet**.

a.     Na Parte 1, o **nginx** foi encontrado para ser executado e atribuído à porta 80 TCP. Embora uma pesquisa rápida na internet revelou que o **nginx** é um servidor web leve, como um analista teria certeza disso? E se um invasor mudou o nome de um programa de malware para **nginx**, apenas para que ele pareça o popular servidor web? Use o **telnet** para se conectar ao host local na porta 80 TCP:

[analyst@secOps ~]$ **telnet 127.0.0.1 80**

Trying 127.0.0.1...

Connected to 127.0.0.1.

Escape character is '^]'.

b.     Pressione algumas letras no teclado. Qualquer tecla funcionará. Após algumas teclas serem pressionadas, pressione ENTER. Abaixo está a saída completa, incluindo o estabelecimento de conexão Telnet e as teclas aleatórias pressionadas (fdsafsdaf, neste caso):

fdsafsdaf

HTTP/1.1 400 Bad Request

Server: nginx/1.16.1

Date: Tue, 28 Apr 2020 20:09:37 GMT

Content-Type: text/html

Content-Length: 173

Connection: close

<html>

<head><title>400 Bad Request</title></head>

<body bgcolor="white">

<center><h1>400 Bad Request</h1></center>

nginx / 1.16.1

</body>

</html>

Connection closed by foreign host.

Graças ao protocolo Telnet, uma conexão TCP de texto claro foi estabelecida, pelo cliente Telnet, diretamente ao servidor nginx, ouvindo na porta 127.0.0.1 80 TCP. Esta conexão nos permite enviar dados diretamente para o servidor. Como o nginx é um servidor web, ele não entende a seqüência de letras aleatórias enviadas a ele e retorna um erro no formato de uma página web.

Por que o erro foi enviado como uma página da Web?

***Área de Resposta***

Enquanto o servidor relatou um erro e encerrou a conexão, fomos capazes de aprender muito. Aprendemos que:

1)    O **nginx** com PID 395 é de fato um servidor web.

2)    A versão do **nginx** é 1.16.1.

3)    A pilha de rede da nossa VM CyberOps Workstation é totalmente funcional até a camada 7.

Nem todos os serviços são iguais. Alguns serviços são projetados para aceitar dados não formatados e não serão encerrados se o lixo for inserido via teclado. Abaixo está um exemplo de tal serviço:

c.     Olhando para a saída **netstat** apresentada anteriormente, é possível ver um processo anexado à porta 22. Use Telnet para se conectar a ele.

Porta 22 TCP é atribuído ao serviço SSH. O SSH permite que um administrador se conecte a um computador remoto de forma segura.

Abaixo está a saída:

[analyst@secOps ~]$ **telnet 127.0.0.1 22**

Trying 127.0.0.1...

Connected to 127.0.0.1.

Escape character is '^]'.

SSH-2.0-OpenSSH\_8.2

sdfjlskj

Invalid SSH identification string.

Connection closed by foreign host.

Use Telnet para se conectar à porta 68. O que acontece? Explique.

***Área de Resposta***

# Questões para Reflexão

1.     Quais são as vantagens de usar netstat?

***Área de Resposta***

2.     Quais são as vantagens de usar o Telnet? É seguro?

***Área de Resposta***