NETWORK TROUBLESHOOTING AND MONITORING

Adapted from Linux Foundation

4.1 Learning Objectives

By the end of this session, you should be able to:

- **Explain network troubleshooting from a client perspective.**
- **Explain network troubleshooting from a server perspective.**

- **Test:**

192.168.0.5

4.2 Networking Troubleshooting

When dealing with network services, one often has to troubleshoot problems which crop up. One should keep in mind that there is a client-side, and a server-side to each network connection:

- Network services require a working network.
- Consider client and server connectivity.

4.3 Simple Client Troubleshooting

The basics of network troubleshooting usually deal with basic connectivity testing. You can use the tools **ping**, **traceroute**, and **nmap** to test connectivity. Remember to test both **DNS** hostnames and **IP** addresses to diagnose **DNS**-related issues.

Ping

```
NAME
       ping, ping6 - send ICMP ECHO_REQUEST to network hosts

SYNOPSIS
       ping  [-aAbBdDfhLnOqrRUvV]  [-c  count]  [-F  flowlabel] [-i interval] [-I interface] [-l preload]
[-m mark] [-M pmtudisc_option] [-N node-info_option] [-w deadline] [-W timeout] [-p pattern] [-Q tos] [-s
packetsize] [-S sndbuf] [-t ttl] [-T timestamp option] [hop ...] destination
```

```
vagrant@vagrant:~$ ping www.google.com
PING www.google.com (172.217.1.36) 56(84) bytes of data.
64 bytes from ord37s07-in-f36.1e100.net (172.217.1.36): icmp_seq=1 ttl=63 time=17.8 ms
64 bytes from ord37s07-in-f36.1e100.net (172.217.1.36): icmp_seq=2 ttl=63 time=19.6 ms
64 bytes from ord37s07-in-f36.1e100.net (172.217.1.36): icmp_seq=3 ttl=63 time=19.0 ms
64 bytes from ord37s07-in-f36.1e100.net (172.217.1.36): icmp_seq=4 ttl=63 time=18.2 ms
64 bytes from ord37s07-in-f36.1e100.net (172.217.1.36): icmp_seq=5 ttl=63 time=19.3 ms
64 bytes from ord37s07-in-f36.1e100.net (172.217.1.36): icmp_seq=6 ttl=63 time=19.8 ms
```

Traceroute

```
NAME
       traceroute - print the route packets trace to network host

SYNOPSIS
       traceroute [-46dFITUnreAV] [-f first_ttl] [-g gate,...]
                  [-i device] [-m max_ttl] [-p port] [-s src_addr]
                  [-q nqueries] [-N squeries] [-t tos]
                  [-l flow_label] [-w waittime] [-z sendwait] [-UL] [-D]
                  [-P proto] [--sport=port] [-M method] [-O mod_options]
                  [--mtu] [--back]
                  host [packet_len]
       traceroute6  [options]
       tcptraceroute  [options]
       lft  [options]
```

Traceroute

```
vagrant@vagrant:~$ traceroute www.google.com
traceroute to www.google.com (172.217.9.68), 30 hops max, 60 byte packets
 1  10.0.2.2 (10.0.2.2)  0.080 ms  0.128 ms  0.107 ms
 2  Nittany1-ve991.gw.psu.edu (104.39.0.3)  3.064 ms  3.445 ms  3.517 ms
 3  172.30.24.110 (172.30.24.110)  2.285 ms  2.636 ms  2.534 ms
 4  * 172.30.63.228 (172.30.63.228)  3.294 ms  3.218 ms
 5  172.30.5.102 (172.30.5.102)  3.127 ms  3.066 ms  2.859 ms
 6  et-11-0-5.2365.rtsw.chic.net.internet2.edu (64.57.20.16)  17.709 ms  17.587 ms
17.545ms
 7  lo-0.8.rtsw2.eqch.net.internet2.edu (64.57.29.130)  17.803 ms  17.826 ms  17.869 ms
 8  74.125.49.146 (74.125.49.146)  19.115 ms  18.121 ms 72.14.220.117 (72.14.220.117)
17.880 ms
 9  * 108.170.243.193 (108.170.243.193)  18.642 ms  18.827 ms
10  72.14.239.115 (72.14.239.115)  18.716 ms 72.14.239.123 (72.14.239.123)  18.828 ms
18.740 ms
11  ord38s09-in-f4.1e100.net (172.217.9.68)  18.846 ms  18.582 ms  18.516 ms
```

nmap

```
NAME
       nmap - Network exploration tool and security / port scanner
SYNOPSIS
       nmap [Scan Type...] [Options] {target specification}
```

```
Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery
and Security Scanning
           https://nmap.org/book/
```

```
Example 1. A representative Nmap scan
# nmap -A -T4 scanme.nmap.org
Nmap scan report for scanme.nmap.org (74.207.244.221)
Host is up (0.029s latency).
rDNS record for 74.207.244.221: li86-221.members.linode.com
Not shown: 995 closed ports
PORT     STATE    SERVICE     VERSION
22/tcp   open     ssh         OpenSSH 5.3p1 Debian 3ubuntu7 (protocol 2.0)
| ssh-hostkey: 1024 8d:60:f1:7c:ca:b7:3d:0a:d6:67:54:9d:69:d9:b9:dd (DSA)
|_2048 79:f8:09:ac:d4:e2:32:42:10:49:d3:bd:20:82:85:ec (RSA)
80/tcp   open     http        Apache httpd 2.2.14 ((Ubuntu))
|_http-title: Go ahead and ScanMe!
646/tcp  filtered ldp
1720/tcp filtered H.323/Q.931
9929/tcp open     nping-echo  Nping echo
Device type: general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6.39
OS details: Linux 2.6.39
Network Distance: 11 hops
Service Info: OS: Linux; CPE: cpe:/o:linux:kernel
TRACEROUTE (using port 53/tcp)
HOP RTT      ADDRESS
[Cut first 10 hops for brevity]
11  17.65 ms li86-221.members.linode.com (74.207.244.221)
Nmap done: 1 IP address (1 host up) scanned in 14.40 seconds
```

nmap

```
vagrant@vagrant:~$ nmap www.google.com
Starting Nmap 7.01 ( https://nmap.org ) at 2018-02-22 18:04 UTC
Nmap scan report for www.google.com (172.217.9.36)
Host is up (0.019s latency).
Other addresses for www.google.com (not scanned): 2607:f8b0:4009:815::2004
rDNS record for 172.217.9.36: ord38s08-in-f4.1e100.net
Not shown: 998 filtered ports
PORT    STATE SERVICE
80/tcp  open  http
443/tcp open  https
Nmap done: 1 IP address (1 host up) scanned in 4.54 seconds
```

4.4 Intermediate Client Troubleshooting

Test plain-text protocols by using the **telnet** command:

```
$ telnet example.com 80 Trying 192.0.43.10...

Connected to example.com.
Escape character is '^]'.
GET /
<html>
<head><title>welcome to example.com</title></head>
<body>
<h1>welcome to example.com</h1>
</body>
</html>
```

You can also do the same with **SSL** or **TLS** protocols.

```
$ openssl s_client -connect www.example.com:443
```

Use the **arp** command to check the link-layer connectivity.

Telnet

```
NAME
     telnet — user interface to the TELNET protocol
SYNOPSIS
     telnet [-468ELadr] [-S tos] [-b address] [-e escapechar] [-l user] [-n
tracefile] [host [port]]
DESCRIPTION
     The telnet command is used for interactive communication with another
host using the TELNET protocol. It begins in command mode, where it prints a
telnet prompt ("telnet> "). If telnet is invoked with a host argument, it
performs an open command implicitly; see the description below.
     Options:
     -4      Force IPv4 address resolution.
     -6      Force IPv6 address resolution.
     -8      Request 8-bit operation. This causes an attempt to negotiate the
TELNET BINARY option for both input and output. By default telnet is not 8-bit
clean.
     -E      Disables the escape character functionality; that is, sets the
escape character to ``no character''.
```

Telnet

```
vagrant@vagrant:~$ telnet www.google.com 80
Trying 216.58.216.228...
Connected to www.google.com.
Escape character is '^]'.
HTTP/1.0 400 Bad Request
Content-Type: text/html; charset=UTF-8
Referrer-Policy: no-referrer
Content-Length: 1555
Date: Thu, 22 Feb 2018 18:10:49 GMT
<!DOCTYPE html>
<html lang=en>
  <meta charset=utf-8>
  <meta name=viewport content="initial-scale=1, minimum-scale=1, width=device-width">
  <title>Error 400 (Bad Request)!!1</title>
  <style>
    *{margin:0;padding:0}html,code{font:15px/22px arial,sans-serif}html{background:#fff;color:#222;padding:
15px}body{margin:7% auto 0;max-width:390px;min-height:180px;padding:30px 0 15px}* > body{background:url(//
www.google.com/images/errors/robot.png) 100% 5px no-repeat;padding-right:205px}p{margin:11px 0
22px;overflow:hidden}ins{color:#777;text-decoration:none}a img{border:0}@media screen and (max-width:772px)
{body{background:none;margin-top:0;max-width:none;padding-right:0}}#logo{background:url(//www.google.com/
images/branding/googlelogo/1x/googlelogo_color_150x54dp.png) no-repeat;margin-left:-5px}@media only screen
and (min-resolution:192dpi){#logo{background:url(//www.google.com/images/branding/googlelogo/2x/
googlelogo_color_150x54dp.png) no-repeat 0% 0%/100% 100%;-moz-border-image:url(//www.google.com/images/
branding/googlelogo/2x/googlelogo_color_150x54dp.png) 0}}@media only screen and (-webkit-min-device-pixel-
ratio:2){#logo{background:url(//www.google.com/images/branding/googlelogo/2x/googlelogo_color_150x54dp.png)
no-repeat;-webkit-background-size:100% 100%}}#logo{display:inline-block;height:54px;width:150px}
  </style>
  <a href=//www.google.com/><span id=logo aria-label=Google></span></a>
  <p><b>400.</b> <ins>That’s an error.</ins>
  <p>Your client has issued a malformed or illegal request.  <ins>That’s all we know.</ins>
Connection closed by foreign host.
```

Openssl

Openssl

```
NAME
       openssl - OpenSSL command line tool
SYNOPSIS
       openssl command [ command_opts ] [ command_args ]
       openssl [ list-standard-commands | list-message-digest-commands | list-cipher-commands | list-
cipher-algorithms | list-message-digest-
       algorithms | list-public-key-algorithms]
       openssl no-XXX [ arbitrary options ]
DESCRIPTION
       OpenSSL is a cryptography toolkit implementing the Secure Sockets Layer (SSL v2/v3) and Transport
Layer Security (TLS v1) network protocols and related cryptography standards required by them.
       The openssl program is a command line tool for using the various cryptography functions of OpenSSL's
crypto library from the shell.  It can be used for
        o  Creation and management of private keys, public keys and parameters
        o  Public key cryptographic operations
        o  Creation of X.509 certificates, CSRs and CRLs
        o  Calculation of Message Digests
        o  Encryption and Decryption with Ciphers
        o  SSL/TLS Client and Server Tests
        o  Handling of S/MIME signed or encrypted mail
        o  Time Stamp requests, generation and verification
```

openssl

```
       rsa       RSA key management.

       rsautl    RSA utility for signing, verification, encryption, and decryption.
Superseded by  pkeyutl

       s_client  This implements a generic SSL/TLS client which can establish a
transparent connection to a remote server speaking SSL/TLS. It’s intended for testing
purposes only and provides only rudimentary interface functionality but internally uses
mostly all functionality of the OpenSSL ssl library.

       s_server  This implements a generic SSL/TLS server which accepts connections from
remote clients speaking SSL/TLS. It's intended for testing purposes only and provides only
rudimentary interface functionality but internally uses mostly all functionality of the
OpenSSL ssl library.  It provides both an own command line oriented protocol for testing
SSL functions and a simple HTTP response facility to emulate an SSL/TLS-aware webserver.
```

Openssl

```
vagrant@vagrant:~$ openssl s_client -connect www.google.com:443
CONNECTED(00000003)
depth=2 C = US, O = GeoTrust Inc., CN = GeoTrust Global CA
verify return:1
depth=1 C = US, O = Google Inc, CN = Google Internet Authority G2
verify return:1
depth=0 C = US, ST = California, L = Mountain View, O = Google Inc, CN =
www.google.com
verify return:1
---
Certificate chain
 0 s:/C=US/ST=California/L=Mountain View/O=Google Inc/CN=www.google.com
   i:/C=US/O=Google Inc/CN=Google Internet Authority G2
 1 s:/C=US/O=Google Inc/CN=Google Internet Authority G2
   i:/C=US/O=GeoTrust Inc./CN=GeoTrust Global CA
 2 s:/C=US/O=GeoTrust Inc./CN=GeoTrust Global CA
   i:/C=US/O=Equifax/OU=Equifax Secure Certificate Authority
```

Openssl

```
---
Server certificate
-----BEGIN CERTIFICATE-----
MIIEdjCCA16gAwIBAgIINC+Y7yLd9OswDQYJKoZIhvcNAQELBQAwSTELMAkGA1UE
BhMCVVMxEzARBgNVBAoTCkdvb2dsZSBJbmMxJTAjBgNVBAMTHEdvb2dsZSBJbnRl
cm5ldCBBdXRob3JpdHkgRzIwHhcNMTgwMjA3MjExMzI5WhcNMTgwNTAyMjExMTAw
WjBoMQswCQYDVQQGEwJVUzETMBEGA1UECAwKQ2FsaWZvcm5pYTEWMBQGA1UEBwwN
TW91bnRhaW4gVmlldzETMBEGA1UECgwKR29vZ2xlIEluYzEXMBUGA1UEAwwOd3d3
Lmdvb2dsZS5jb20wggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQC7lAOc
gsUECzoiJfpnAtq9qxAeTWBS8KYCd3ESvd7255YXW8FUiGTj9MYSSJ3OlYQvvU1I
NmnIXNU7BnhUBbY1kW4+GXc5RimwiIW5VsWftt1XOVZh5mR08DhYQjdQqI3IhK6r
FTS6/6BvFcjWMT/rVQv59XDaQLqWXSomEzOr1vDRXZSbAPr+YAGKUj+K0TjgZNW1
8xo8Lyp8kDjFxrWaThfwFMosbFw5HnnzpT1WSHfmXmF1mvvk4cJ+U2m3+K2pRki8
nNnWafLPdT408XoXrbWLVeEVSIQQH5z93uoj5lESal05pnOY5yYUJ+vmHdY7jOBh
sT9HaGzl3kD2J+1BAgMBAAGjggFBMIIBPTATBgNVHSUEDDAKBggrBgEFBQcDATAZ
BgNVHREEEjAQgg53d3cuZ29vZ2xlLmNvbTBoBggrBgEFBQcBAQRcMFowKwYIKwYB
BQUHMAKGH2h0dHA6Ly9wa2kuZ29vZ2xlLmNvbS9HSUFHMi5jcnQwKwYIKwYBBQUH
MAGGH2h0dHA6Ly9jbGllbnRzMS5nb29nbGUuY29tL29jc3AwHQYDVR0OBBYEFNGB
jzGWH9WkzeHj88QOo3gBTBs+MAwGA1UdEwEB/wQCMAAwHwYDVR0jBBgwFoAUSt0G
Fhu89mi1dvWBtrtiGrpagS8wIQYDVR0gBBowGDAMBgorBgEEAdZ5AgUBMAgGBmeB
```

Openssl

```
subject=/C=US/ST=California/L=Mountain View/O=Google Inc/CN=www.google.com
issuer=/C=US/O=Google Inc/CN=Google Internet Authority G2
---
No client certificate CA names sent
Peer signing digest: SHA256
Server Temp Key: ECDH, P-256, 256 bits
---
SSL handshake has read 3822 bytes and written 431 bytes
---
```

Openssl

```
New, TLSv1/SSLv3, Cipher is ECDHE-RSA-AES128-GCM-SHA256
Server public key is 2048 bit
Secure Renegotiation IS supported
Compression: NONE
Expansion: NONE
No ALPN negotiated
SSL-Session:
    Protocol  : TLSv1.2
    Cipher    : ECDHE-RSA-AES128-GCM-SHA256
    Session-ID:
E0153255AF5676E6DC0813733D322570D5C2EAF9807AA4A7D2AB0CD6992734C9
    Session-ID-ctx:
    Master-Key:
3D7A63ACCD0D2C5E82639D9B0D0015189FFA82E27C3E39766E96B1E706E6CA3DE0FDB66197CD9F
CB71FB804896018E88
    Key-Arg   : None
    PSK identity: None
    PSK identity hint: None
    SRP username: None
```

arp

```
NAME
       arp - manipulate the system ARP cache

SYNOPSIS
       arp [-vn] [-H type] [-i if] [-a] [hostname]
       arp [-v] [-i if] -d hostname [pub]
       arp [-v] [-H type] [-i if] -s hostname hw_addr [temp]
       arp [-v] [-H type] [-i if] -s hostname hw_addr [netmask nm] pub
       arp [-v] [-H type] [-i if] -Ds hostname ifname [netmask nm] pub
       arp [-vnD] [-H type] [-i if] -f [filename]
```

```
virsh dumpxml cse544_15 | grep address
arp -n | grep 52:54:00:f4:ff:14
```

ssh cse544@192.168.122.84

arp

```
vagrant@vagrant:~$ arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.2.2                 ether   52:54:00:12:35:02   C                     eth0
10.0.2.3                 ether   52:54:00:12:35:03   C                     eth0
```

4.5 Advanced Client Troubleshooting

The **tcpdump** command and **wireshark** tool are useful when you need to dig deeper into a protocol. The command line-based **tcpdump** truncates packets by default and generates **pcap** files.

wireshark uses the graphical interface to capture packets. It can capture and analyze packets in realtime. It is useful to analyze **pcap** files, but you may not want **wireshark** installed on the system you are troubleshooting.

To capture packets with **tcpdump** for use with **wireshark**, use:

```
# tcpdump -i eth0 -s 65535 -w capture.pcap port 22
```

Tcpdump

```
NAME
       tcpdump - dump traffic on a network

SYNOPSIS
       tcpdump [ -AbdDefhHIJKlLnNOpqStuUvxX# ] [ -B buffer_size ]
               [ -c count ]
               [ -C file_size ] [ -G rotate_seconds ] [ -F file ]
               [ -i interface ] [ -j tstamp_type ] [ -m module ] [ -M secret ]
               [ --number ] [ -Q in|out|inout ]
               [ -r file ] [ -V file ] [ -s snaplen ] [ -T type ] [ -w file ]
               [ -W filecount ]
               [ -E spi@ipaddr algo:secret,...  ]
               [ -y datalinktype ] [ -z postrotate-command ] [ -Z user ]
               [ --time-stamp-precision=tstamp_precision ]
               [ --immediate-mode ] [ --version ]
               [ expression ]
```

tcpdump

```
vagrant@vagrant:~$ sudo tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
18:30:50.787754 IP 10.0.2.15.ssh > 10.0.2.2.56208: Flags [P.], seq 872529436:872529512,
ack 848018319, win 42960, length 76
18:30:50.787957 IP 10.0.2.2.56208 > 10.0.2.15.ssh: Flags [.], ack 76, win 65535, length 0
18:30:50.788147 IP 10.0.2.15.ssh > 10.0.2.2.56208: Flags [P.], seq 76:112, ack 1, win
42960, length 36
18:30:50.788253 IP 10.0.2.2.56208 > 10.0.2.15.ssh: Flags [.], ack 112, win 65535, length 0
18:30:50.788523 IP 10.0.2.15.ssh > 10.0.2.2.56208: Flags [P.], seq 112:196, ack 1, win
42960, length 84
18:30:50.788605 IP 10.0.2.15.ssh > 10.0.2.2.56208: Flags [P.], seq 196:232, ack 1, win
42960, length 36
18:30:50.788660 IP 10.0.2.2.56208 > 10.0.2.15.ssh: Flags [.], ack 196, win 65535, length 0
```

4.6 Common Client Side Problems

Some common networking issues found at the client side include:

- **DNS** issues - Can you ping the IP address but not the hostname?
- Firewall issues - A firewall on the client side which is rejecting the return traffic from a network request will cause problems.
- Incorrect network settings.
- - Make sure the IP address is correct. Does it match the **DNS** host name?
- - If the route is wrong or missing, traffic will not get to the other network node.
- - Netmasks determine network routes, thus it is important to have the netmask of your host correct.

4.7 Basic Server Troubleshooting

To perform basic server troubleshooting, test the network connectivity from the server's point of view.

The **netstat** command lists which daemons are listening on which ports.

```
# netstat -taupe | grep httpd
```

```
tcp 0 0 *:http *:* LISTEN root 23390 5543/httpd
```

The **ss** command is another socket statistics utility. It may be a replacement to **netstat** although it is missing some socket types. A similar command to the **netstat** example shown would be:

```
# ss -ltp | grep httpd
```

Verify that the daemon is running, using the **chkconfig**, **service**, **ps** commands, or the **init** script.

One of the first steps in troubleshooting a server-side daemon should be to check the log files. Log messages will tell you exactly what is wrong, without having to do much debugging.

Netstat

```
vagrant@vagrant:~$ netstat -taupe
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       User       Inode      PID/
Program name
tcp        0      0 localhost:6379          *:*                     LISTEN      redis      289460     -
tcp        0      0 *:pop3                  *:*                     LISTEN      root       276708     -
tcp        0      0 *:imap2                 *:*                     LISTEN      root       276737     -
tcp        0      0 192.168.122.1:domain    *:*                     LISTEN      root       277380     -
tcp        0      0 *:ssh                   *:*                     LISTEN      root       270351     -
tcp        0      0 *:3000                  *:*                     LISTEN      root       298227     -
tcp        0      0 localhost:44688         localhost:6379          ESTABLISHED root       297897     -
tcp        0      0 localhost:44686         localhost:6379          ESTABLISHED root       297894     -
tcp        0      0 localhost:6379          localhost:44686         ESTABLISHED redis      297895     -
tcp        0      0 10.0.2.15:ssh           10.0.2.2:56208          ESTABLISHED root       175752     -
tcp        0      0 localhost:6379          localhost:44688         ESTABLISHED redis      297898     -
tcp6       0      0 [::]:pop3               [::]:*                  LISTEN      root       276709     -
tcp6       0      0 [::]:imap2              [::]:*                  LISTEN      root       276738     -
tcp6       0      0 [::]:http               [::]:*                  LISTEN      root       286597     -
tcp6       0      0 [::]:ftp                [::]:*                  LISTEN      root       15412      -
tcp6       0      0 [::]:ssh                [::]:*                  LISTEN      root       270360     -
tcp6       0      0 [::]:https              [::]:*                  LISTEN      root       290545     -
```

netstat

```
NAME
       netstat - Print network connections, routing tables, interface statistics, masquerade
connections, and multicast memberships
SYNOPSIS
       netstat  [address_family_options] [--tcp|-t] [--udp|-u] [--raw|-w] [--listening|-l] [--
all|-a] [--numeric|-n] [--numeric-hosts] [--numeric-
       ports] [--numeric-users] [--symbolic|-N] [--extend|-e[--extend|-e]] [--timers|-o] [--
program|-p] [--verbose|-v] [--continuous|-c]
       netstat {--route|-r} [address_family_options] [--extend|-e[--extend|-e]] [--verbose|-v] [--
numeric|-n] [--numeric-hosts]  [--numeric-ports]
       [--numeric-users] [--continuous|-c]
       netstat  {--interfaces|-i} [--all|-a] [--extend|-e[--extend|-e]] [--verbose|-v] [--
program|-p] [--numeric|-n] [--numeric-hosts] [--numeric-
       ports] [--numeric-users] [--continuous|-c]
       netstat {--groups|-g} [--numeric|-n] [--numeric-hosts] [--numeric-ports] [--numeric-users]
[--continuous|-c]
       netstat {--masquerade|-M} [--extend|-e] [--numeric|-n] [--numeric-hosts] [--numeric-ports]
[--numeric-users] [--continuous|-c]
       netstat {--statistics|-s} [--tcp|-t] [--udp|-u] [--raw|-w]
       netstat {--version|-V}
       netstat {--help|-h}
       address_family_options:
       [-4] [-6] [--protocol={inet,unix,ipx,ax25,netrom,ddp}[,...]]  [--unix|-x] [--inet|--ip] [--
ax25] [--ipx] [--netrom] [--ddp]
```

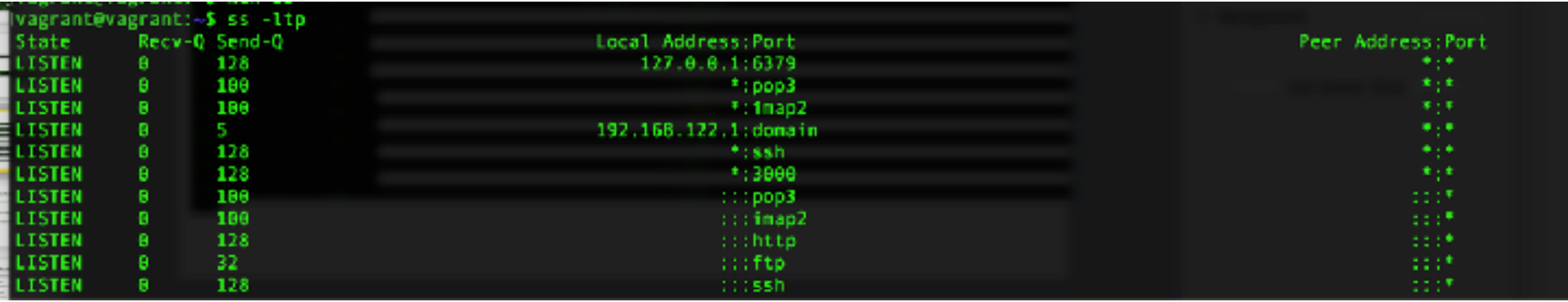
ss

```
NAME
       ss - another utility to investigate sockets

SYNOPSIS
       ss [options] [ FILTER ]

DESCRIPTION
       ss  is  used  to  dump socket statistics. It allows showing information
similar to netstat.  It can display more TCP and state informations
       than other tools.
```

SS


vagrant@vagrant:~$ ss -ltp
State Recv-Q Send-Q Local Address:Port Peer Address:Port
LISTEN 0 128 127.0.0.1:6379 *:*
LISTEN 0 100 *:pop3 *:*
LISTEN 0 100 *:imap2 *:*
LISTEN 0 5 192.168.122.1:domain *:*
LISTEN 0 128 *:ssh *:*
LISTEN 0 128 *:3000 *:*
LISTEN 0 100 :::pop3 :::*
LISTEN 0 100 :::imap2 :::*
LISTEN 0 128 :::http :::*
LISTEN 0 32 :::ftp :::*
LISTEN 0 128 :::ssh :::*

4.8 Intermediate Server Troubleshooting

With intermediate server troubleshooting, the server side firewall configuration needs to take into account that:

- New traffic is allowed in.
- Return traffic is allowed in.
- Unwanted traffic is filtered.

Access control systems may also cause troubleshooting problems. Check the settings of tools such as TCP wrappers. (`/etc/hosts.allow` and `/etc/hosts.deny`)

Consult **man 5 hosts_access** for additional details.

Application settings can also cause problems. Make sure there are not any restrictions, blacklists or other application configuration errors.

4.9 Advanced Server Troubleshooting

For advanced server troubleshooting, the `/proc` filesystem has settings that affect the network stack:

- `/proc/sys/net/ipv4/ip_forward`
- Allows for network traffic to be forwarded from one interface to another.
- `/proc/sys/net/ipv4/conf/*/accept_redirects`
- Accepting **ICMP** redirects from a router to find better routes. **This setting has the potential to be exploited by a malicious party to redirect your traffic.**
- `/proc/sys/net/ipv4/icmp_echo_ignore_all`
- Changing this setting will affect the host's visibility to **ICMP** ping packets.
- `/proc/sys/net/ipv4/icmp_echo_ignore_broadcasts`
- This setting will change the host's visibility to broadcast ICMP ping packets.
- `/proc/net/arp`
- Contains the current **arp table.**

These settings are not persistent across reboots. To make the changes persistent, edit the `/etc/sysctl.conf` configuration file, or a **.conf** file in the `/etc/sysctl.d` directory.

The syntax for `/etc/sysctl.conf` matches the path for the file in `/proc/sys` with the **.** character instead of */*.

netstat

```
/proc/net/dev -- device information
/proc/net/raw -- raw socket information
/proc/net/tcp -- TCP socket information
/proc/net/udp -- UDP socket information
/proc/net/igmp -- IGMP multicast information
/proc/net/unix -- Unix domain socket information
/proc/net/ipx -- IPX socket information
/proc/net/ax25 -- AX25 socket information
/proc/net/appletalk -- DDP (appletalk) socket information
/proc/net/nr -- NET/ROM socket information
/proc/net/route -- IP routing information
/proc/net/ax25_route -- AX25 routing information
/proc/net/ipx_route -- IPX routing information
/proc/net/nr_nodes -- NET/ROM nodelist
/proc/net/nr_neigh -- NET/ROM neighbours
/proc/net/ip_masquerade -- masqueraded connections
/proc/net/snmp -- statistics
```

4.10 Common Server Problems

Common server problems include broken **DNS**, overzealous firewall rules, incorrect network settings, and the daemon not listening on the right interface/port.

Some access control systems require that **Reverse DNS** be properly set up.

When enabling new traffic to pass through a firewall, pay attention to the type of protocol (**UDP** over **TCP** for example) used.

Some protocols break when return traffic comes back from a different **IP** address. Verify that your egress route is correct.

4.11 Network Monitoring

For network monitoring, the **iptraf** tool is a real-time network traffic analyzer. It recognizes the following protocols:

- **IP**
- **TCP**
- **UDP**
- **ICMP**
- **IGMP**
- **IGP**
- **IGRP**
- **OSPF**
- **ARP**
- **RARP.**

Snort is a network intrusion detection system. In addition to being a network monitor, it can help pinpoint unwanted traffic inside of a network.

ntop is an application and web app for monitoring network usage. It can pinpoint bandwidth use, display network statistics, and more.