Linux Incident Response - using ss for network analysis

Taz Wake | Halkyn Consulting Ltd | Originally published 2023-11-14

Introduction to the ss Command

The ss (socket statistics) command is a powerful tool in Linux used for examining sockets. As an incident responder, understanding the ss command is crucial for analyzing network connections and traffic, particularly in identifying and investigating potentially malicious activities. Unlike the older netstat command, ss is faster and displays more detailed information, making it invaluable in modern incident response scenarios.

Basic Usage of ss

Overview of Basic Commands

The basic usage of ss is straightforward. When executed without any options, ss lists all open sockets. To refine this, various options can be applied. For example, **ss -t** displays TCP sockets, **ss -u** shows UDP sockets, **ss -l** lists listening sockets, and **ss -a** shows both listening and non-listening sockets.

```
$ ss -t
State      Recv-Q      Send-Q                  Local Address:Port                   Peer Address:Port      Process
ESTAB      0           0                     192.168.182.128:54504                185.125.190.27:https
ESTAB      0           0                     192.168.182.128:58804                  91.189.91.42:https
ESTAB      0           0                [::ffff:192.168.182.128]:1716          [::ffff:192.168.182.138]:58222
```

Example output from ss -t

Filtering Sockets

Filtering is a key feature in ss. Incident responders can use ss to filter sockets based on states like *established, listen, syn-sent, syn-recv, fin-wait-1*, etc. For instance, **ss state established** will list all established connections.

```
$ ss state established
Netid   Recv-Q   Send-Q                                  Local Address:Port                   Peer Address:Port   Process
u_dgr   0        0                    /var/lib/samba/private/msg.sock/1470 25497                          * 0
u_dgr   0        0                    /var/lib/samba/private/msg.sock/1486 24501                          * 0
u_dgr   0        0                    /var/lib/samba/private/msg.sock/1492 28674                          * 0
u_dgr   0        0                    /var/lib/samba/private/msg.sock/1510 24573                          * 0
u_dgr   0        0                    /var/lib/samba/private/msg.sock/1511 26913                          * 0
u_dgr   0        0                    /var/lib/samba/private/msg.sock/1514 29701                          * 0
u_dgr   0        0                    /var/lib/samba/private/msg.sock/1346 27668                          * 0
u_dgr   0        0                                      /run/systemd/notify 20751                         * 0
u_dgr   0        0                             /run/systemd/journal/dev-log 20777                         * 0
u_dgr   0        0                              /run/systemd/journal/socket 20779                         * 0
u_str   0        0                                                        * 33455                         * 33456
u_str   0        0                               /run/systemd/journal/stdout 31260                        * 33362
u_str   0        0                                                        * 32348                         * 34252
u_str   0        0                                      /run/user/1000/bus 31470                          * 32327
u_str   0        0                                 /run/user/1000/wayland-0 33393                         * 33971
u_str   0        0                                                        * 31030                         * 32905
```

Example output from ss state established. Note: This is very noisy.

Advanced Usage and Options

Displaying Process Information

One of the most powerful features of ss is its ability to display process information linked to sockets. Using **ss -tp**, you can view the process ID (PID) and process name associated with each socket. This is particularly useful for tracing back network activity to specific processes.

```
$ ss -tp
State    Recv-Q    Send-Q        Local Address:Port          Peer Address:Port     Process
ESTAB    0         0          192.168.182.128:57856       152.195.38.76:http      users:(("firefox",pid=3764,fd=92))
ESTAB    0         0          192.168.182.128:33422       142.251.221.78:http     users:(("firefox",pid=3764,fd=124))
ESTAB    0         0          192.168.182.128:44478       172.217.167.67:http     users:(("firefox",pid=3764,fd=123))
ESTAB    0         0          192.168.182.128:51206          34.98.75.36:https    users:(("firefox",pid=3764,fd=130))
ESTAB    0         0          192.168.182.128:49524       34.160.90.233:https     users:(("firefox",pid=3764,fd=176))
ESTAB    0         0          192.168.182.128:38378       34.117.237.239:https    users:(("firefox",pid=3764,fd=93))
ESTAB    0         0          192.168.182.128:54082       152.195.38.76:http      users:(("firefox",pid=3764,fd=116))
ESTAB    0         0          192.168.182.128:43352       34.120.208.123:https    users:(("firefox",pid=3764,fd=111))
ESTAB    0         0          192.168.182.128:42810       34.160.144.191:https    users:(("firefox",pid=3764,fd=61))
ESTAB    0         0          192.168.182.128:58480       192.168.182.2:domain
ESTAB    0         0          192.168.182.128:59148       23.214.88.58:http       users:(("firefox",pid=3764,fd=98))
ESTAB    0         0          192.168.182.128:52146       34.107.221.82:http      users:(("firefox",pid=3764,fd=87))
ESTAB    0         0          192.168.182.128:39558       34.120.115.102:https    users:(("firefox",pid=3764,fd=155))
ESTAB    0         0          192.168.182.128:36178       34.117.65.55:https      users:(("firefox",pid=3764,fd=139))
ESTAB    0         0          192.168.182.128:58414       142.251.221.65:https    users:(("firefox",pid=3764,fd=175))
ESTAB    0         0          192.168.182.128:38096       18.67.98.168:http       users:(("firefox",pid=3764,fd=128))
ESTAB    0         0          192.168.182.128:41480       34.160.144.191:https    users:(("firefox",pid=3764,fd=143))
```

Example ss -tp output showing a lot of browser activity.

Extended Information and Statistics

For more detailed analysis, **ss -ie** displays extended information like TCP congestion control algorithm used and **ss -m** shows socket memory usage statistics. These extended details can be crucial in diagnosing performance issues or detecting unusual patterns indicative of malicious activities.

```
$ ss -m
Netid  State   Recv-Q  Send-Q                                  Local Address:Port                  Peer Address:Port   Process
u_dgr  ESTAB   0       0                 /var/lib/samba/private/msg.sock/1470 25497                         * 0       skmem:(r0,rb212992,t0,tb212992,f0,w0
u_dgr  ESTAB   0       0                 /var/lib/samba/private/msg.sock/1486 24501                         * 0       skmem:(r0,rb212992,t0,tb212992,f0,w0
u_dgr  ESTAB   0       0                 /var/lib/samba/private/msg.sock/1492 28674                         * 0       skmem:(r0,rb212992,t0,tb212992,f0,w0
u_dgr  ESTAB   0       0                 /var/lib/samba/private/msg.sock/1510 24573                         * 0       skmem:(r0,rb212992,t0,tb212992,f0,w0
u_dgr  ESTAB   0       0                 /var/lib/samba/private/msg.sock/1511 26913                         * 0       skmem:(r0,rb212992,t0,tb212992,f0,w0
u_dgr  ESTAB   0       0                 /var/lib/samba/private/msg.sock/1514 29701                         * 0       skmem:(r0,rb212992,t0,tb212992,f0,w0
u_dgr  ESTAB   0       0                 /var/lib/samba/private/msg.sock/1346 27668                         * 0       skmem:(r0,rb212992,t0,tb212992,f0,w0
u_dgr  ESTAB   0       0                                  /run/systemd/notify 20751                         * 0       skmem:(r0,rb16777216,t0,tb212992,f0,
u_dgr  ESTAB   0       0                         /run/systemd/journal/dev-log 20777                         * 0       skmem:(r0,rb16777216,t0,tb16777216,f
u_dgr  ESTAB   0       0                          /run/systemd/journal/socket 20779                         * 0       skmem:(r0,rb16777216,t0,tb212992,f0,
u_str  ESTAB   0       0                                                    * 33455                         * 33456   skmem:(r0,rb212992,t0,tb212992,f0,w0
u_str  ESTAB   0       0                           /run/systemd/journal/stdout 31260                         * 33362   skmem:(r0,rb212992,t0,tb212992,f0,w0
```

Truncated output from ss -m

Analyzing Network Traffic with ss

Identifying Suspicious Connections

Incident responders often need to identify outbound connections that could be indicative of exfiltration or C2 (Command and Control) activities. Using ss, one can filter for established connections on unusual ports or to unfamiliar IP addresses. For example, **ss -tp state established dport = :[PORT_NUMBER]** lists all established TCP connections on a specific port and shows the process involved.

```
$ ss -tp state established dport = :443
Recv-Q         Send-Q                  Local Address:Port                  Peer Address:Port          Process
0              0                    192.168.182.128:36186                 34.117.65.55:https          users:(("firefox",pid=3764,fd=140))
0              0                    192.168.182.128:55558               142.250.76.100:https          users:(("firefox",pid=3764,fd=35))
```

Example output from ss -tp looking for established connections to destination port 443

Monitoring Listening Ports

Monitoring for unexpected listening ports is another critical activity. **ss -tuln** lists all TCP and UDP ports on which the system is listening, helping to identify unauthorized services or backdoors.

```
$ ss -tuln
Netid          State           Recv-Q          Send-Q                    Local Address:Port                  Peer Address:Port         Process
udp            UNCONN          0               0                               0.0.0.0:58300                      0.0.0.0:*
udp            UNCONN          0               0                               0.0.0.0:5353                       0.0.0.0:*
udp            UNCONN          0               0                               0.0.0.0:2055                       0.0.0.0:*
udp            UNCONN          0               0                               0.0.0.0:60996                      0.0.0.0:*
udp            UNCONN          0               0                          127.0.0.53%lo:53                        0.0.0.0:*
udp            UNCONN          0               0                  192.168.182.128%ens33:68                        0.0.0.0:*
udp            UNCONN          0               0                        192.168.182.255:137                       0.0.0.0:*
udp            UNCONN          0               0                        192.168.182.128:137                       0.0.0.0:*
udp            UNCONN          0               0                               0.0.0.0:137                        0.0.0.0:*
udp            UNCONN          0               0                        192.168.182.255:138                       0.0.0.0:*
udp            UNCONN          0               0                        192.168.182.128:138                       0.0.0.0:*
udp            UNCONN          0               0                               0.0.0.0:138                        0.0.0.0:*
udp            UNCONN          0               0                               0.0.0.0:631                        0.0.0.0:*
udp            UNCONN          0               0                                  [::]:5353                          [::]:*
udp            UNCONN          0               0                                     *:1716                             *:*
udp            UNCONN          0               0                                  [::]:45434                         [::]:*
tcp            LISTEN          0               4096                       127.0.0.53%lo:53                        0.0.0.0:*
tcp            LISTEN          0               128                             0.0.0.0:22                         0.0.0.0:*
tcp            LISTEN          0               128                           127.0.0.1:631                        0.0.0.0:*
tcp            LISTEN          0               50                              0.0.0.0:445                        0.0.0.0:*
```

Truncated ss -tuln output

Investigating Malicious C2 and Exfiltration Activity

Detecting Unusual Traffic Patterns

In the context of C2 or data exfiltration, unusual traffic patterns are a red flag. Commands like **ss -tan** state established can be used to view all (listening and non-listening) TCP connections, which can then be further analyzed for suspicious activity.

```
$ ss -tan
State          Recv-Q          Send-Q                    Local Address:Port                  Peer Address:Port         Process
LISTEN         0               4096                       127.0.0.53%lo:53                        0.0.0.0:*
LISTEN         0               128                             0.0.0.0:22                         0.0.0.0:*
LISTEN         0               128                           127.0.0.1:631                        0.0.0.0:*
LISTEN         0               50                              0.0.0.0:445                        0.0.0.0:*
LISTEN         0               50                              0.0.0.0:139                        0.0.0.0:*
ESTAB          0               0                  192.168.182.128:36632                142.250.76.99:80
ESTAB          0               0                  192.168.182.128:44178                172.217.167.67:443
ESTAB          0               0                  192.168.182.128:36636                142.250.76.99:80
ESTAB          0               0                  192.168.182.128:36628                142.250.76.99:80
ESTAB          0               0                  192.168.182.128:36614                142.250.76.99:80
ESTAB          0               0                  192.168.182.128:36630                142.250.76.99:80
ESTAB          0               0                  192.168.182.128:36186                34.117.65.55:443
```

Example of ss -tan output

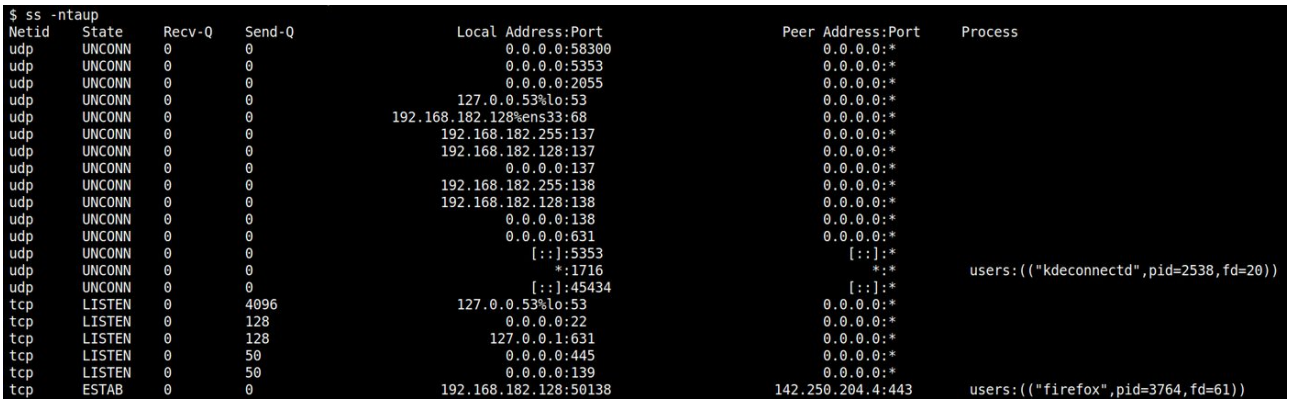
Tracing Process IDs

When a suspicious connection is identified, using **ss -tp** to find the PID and process name can lead to the source of the malicious activity. This can be critical in understanding the scope and impact of a security incident.

Converting from netstat

If you are normally a netstat user, you can easily move over to using ss on Linux platforms. For example, the typical netstat command used for DFIR is **netstat -anop**. This means: show all sockets (listening and non-listening), show numerical addresses, show timer information and show process ID.

You can replicate this with ss by using



```
$ ss -ntaup
Netid     State      Recv-Q     Send-Q                 Local Address:Port                  Peer Address:Port     Process
udp       UNCONN     0          0                            0.0.0.0:58300                      0.0.0.0:*
udp       UNCONN     0          0                            0.0.0.0:5353                       0.0.0.0:*
udp       UNCONN     0          0                            0.0.0.0:2055                       0.0.0.0:*
udp       UNCONN     0          0                          127.0.0.53%lo:53                     0.0.0.0:*
udp       UNCONN     0          0                  192.168.182.128%ens33:68                     0.0.0.0:*
udp       UNCONN     0          0                      192.168.182.255:137                      0.0.0.0:*
udp       UNCONN     0          0                      192.168.182.128:137                      0.0.0.0:*
udp       UNCONN     0          0                            0.0.0.0:137                        0.0.0.0:*
udp       UNCONN     0          0                      192.168.182.255:138                      0.0.0.0:*
udp       UNCONN     0          0                      192.168.182.128:138                      0.0.0.0:*
udp       UNCONN     0          0                            0.0.0.0:138                        0.0.0.0:*
udp       UNCONN     0          0                            0.0.0.0:631                        0.0.0.0:*
udp       UNCONN     0          0                               [::]:5353                          [::]:*
udp       UNCONN     0          0                                  *:1716                             *:*         users:(("kdeconnectd",pid=2538,fd=20))
udp       UNCONN     0          0                              [::]:45434                         [::]:*
tcp       LISTEN     0          4096                       127.0.0.53%lo:53                     0.0.0.0:*
tcp       LISTEN     0          128                          0.0.0.0:22                         0.0.0.0:*
tcp       LISTEN     0          128                        127.0.0.1:631                        0.0.0.0:*
tcp       LISTEN     0          50                           0.0.0.0:445                        0.0.0.0:*
tcp       LISTEN     0          50                           0.0.0.0:139                        0.0.0.0:*
tcp       ESTAB      0          0                      192.168.182.128:50138                142.250.204.4:443       users:(("firefox",pid=3764,fd=61))
```

Example ss -ntaup output

Summary

The ss command is a versatile and powerful tool for incident responders. Its ability to provide detailed socket information, coupled with advanced filtering and process association capabilities, makes it essential for modern network traffic analysis and incident investigation. By mastering ss, incident responders can efficiently identify, analyze, and respond to various network-related security incidents, especially in the realms of unauthorized access, C2 communications, and data exfiltration attempts.

If you would like to know more about this, and Linux Incident Response in general, have a look at the SANS Institute course FOR577 Linux Incident Response and Threat Hunting. You can find out more about this course at https://sans.org/for577

About Halkyn Consulting

Halkyn Consulting Ltd is a UK-based security and risk management consultancy providing 24/7 incident response, digital forensics, threat hunting, virtual security leadership (vCISO) and security compliance support to organisations worldwide. We deliver clear, practical, vendor-independent advice based on real-world threats — no hype, no upsell.

More free security resources are available at www.halkynconsulting.co.uk/security-resources.

Web: www.halkynconsulting.co.uk | **Email:** info@halkynconsulting.co.uk | **Tel:** +44 1244 940858