NAME

sc_filterpolicy — scamper driver to test systems for congruent filtering policy

SYNOPSIS

sc_filterpolicy [**-D**] [**-a** *input-file*] [**-l** *log-file*] [**-o** *output-file*]
[**-O** *options*] [**-p** *scamper-port*] [**-t** *host-type*] [**-T** *test*]
[**-U** *scamper-unix*]

sc_filterpolicy [**-r** *data-file*]

DESCRIPTION

The **sc_filterpolicy** utility provides the ability to connect to a running scamper(1) instance and use that instance to test systems for congruent filtering policy. The utility tests each system specified in the input file by probing for application reachability with ICMP, UDP, and TCP probes, using both IPv4 and IPv6 where applicable. Each system in the input file should have multiple IP addresses specified; the driver probes each IP address on each system one at a time to avoid causing the remote system to rate-limit responses. **sc_filterpolicy** obtains speed by probing systems in parallel, though it may appear to operate slowly because no progress is reported until all addresses belonging to a device have been tested one at a time.

The applications supported by **sc_filterpolicy** to test filtering policy are:

- **ICMP:** test responsiveness to ICMP echo request packets. We classify the IP address as responsive to ICMP echo requests if it sends an ICMP echo reply.
- **NetBIOS:** test responsiveness to TCP SYN packets sent to port 139 (the NetBIOS port). We classify the IP address as responsive if it sends a SYN/ACK.
- **MSSQL:** test responsiveness to TCP SYN packets sent to port 1433 (the Microsoft SQL server default port). We classify the IP address as responsive if it sends a SYN/ACK.
- **FTP:** test responsiveness to TCP SYN packets sent to port 21 (the default port for FTP control connections). We classify the IP address as responsive if it sends a SYN/ACK.
- **SSH:** test responsiveness to TCP SYN packets sent to port 22 (the default port for SSH). We classify the IP address as responsive if it sends a SYN/ACK.
- **Telnet:** test responsiveness to TCP SYN packets sent to port 23 (the default port for telnet). We classify the IP address as responsive if it sends a SYN/ACK.
- **MySQL:** test responsiveness to TCP SYN packets sent to port 3306 (the default port for MySQL). We classify the IP address as responsive if it sends a SYN/ACK.
- **RDP:** test responsiveness to TCP SYN packets sent to port 3389 (the default port for RDP). We classify the IP address as responsive if it sends a SYN/ACK.
- **HTTPS:** test responsiveness to TCP SYN packets sent to port 443 (the default port for HTTPS). We classify the IP address as responsive if it sends a SYN/ACK.
- **SMB:** test responsiveness to TCP SYN packets sent to port 445 (the default port for SMB). We classify the IP address as responsive if it sends a SYN/ACK.
- **HTTP:** test responsiveness to TCP SYN packets sent to port 80 (the default port for HTTP). We classify the IP address as responsive if it sends a SYN/ACK.
- **BGP:** test responsiveness to TCP SYN packets sent to port 179 (the default port for BGP). We classify the IP address as responsive if it sends a SYN/ACK.
- **NTP:** test responsiveness to UDP packets sent to port 123 (the default port for NTP) with an NTP version request payload. We classify the IP address as responsive if it sends a UDP response.
- **DNS:** test responsiveness to UDP packets sent to port 53 (the default port for DNS) with a query for www.google.com. We classify the IP address as responsive if it sends a UDP response.
- **SNMP:** test responsiveness to UDP packets sent to port 161 (the default port for SNMP) with a get for sysDescr via the public community using the SNMPv2c protocol. We classify the IP address as responsive if it sends a UDP response.

- **VNC:** test responsiveness to TCP SYN packets sent to port 5900 (the default port for VNC). We classify the IP address as responsive if it sends a SYN/ACK.

The options supported by **sc_filterpolicy** are as follows:

-?
prints a list of command line options and a synopsis of each.

-a *input-file*
specifies the name of the input file which consists of a sequence of systems to test. See the examples section for input-file formatting examples.

-D
with this option set, **sc_filterpolicy** will detach and become a daemon.

-l *log-file*
specifies the name of a file to log progress output from **sc_filterpolicy** generated at run time.

-o *output-file*
specifies the name of the file to be written. The output file will use the warts(5) format.

-O *options*
allows the behavior of **sc_filterpolicy** to be further tailored. The current choices for this option are:
- **impatient:** order the systems found in the input-file so that those with the most addresses are probed first, so that probing will complete as fast as possible.
- **incongruent:** only report systems which are inferred to have an incongruent filtering policy.
- **trace:** probe the addresses found in the input-file using traceroute, rather than ping.
- **tuples:** signals that the input-file is formatted as tuples, rather than rows. See the examples section for more information.

-p *scamper-port*
specifies the port on the local host where scamper(1) is accepting control socket connections.

-r *data-file*
specifies the name of a previously collected filter policy data file, in warts(5) format, to read and analyse.

-t *probe-class*
specifies the class of probes to send for each IP address in the input file. The current choices for this option are:
- **router:** test ICMP, SSH, Telnet, HTTPS, HTTP, BGP, NTP, DNS, and SNMP.
- **server:** test ICMP, FTP, SSH, Telnet, MySQL, RDP, HTTPS, SMB, HTTP, NTP, DNS, and SNMP.
- **all:** test ICMP, NetBIOS, MSSQL, FTP, SSH, Telnet, MySQL, RDP, HTTPS, SMB, VNC, HTTP, BGP, NTP, DNS, and SNMP.

-T *test*
specifies adjustments to the test schedule from the supported application types. Prefacing an application with + causes the application type to be added to the test schedule, and prefacing an application with - causes the application type to be removed from the test schedule.

-U *scamper-unix*
specifies the unix domain socket on the local host where scamper(1) is accepting control socket connections.

EXAMPLES

sc_filterpolicy requires a scamper(1) instance listening on a port or unix domain socket for commands in order to collect data:

scamper -P 31337

will start a `scamper`(1) instance listening on port 31337 on the loopback interface. To use **`sc_filterpolicy`** to test the filtering policy of a set of routers specified in a file named routers.txt and formatted as rows as follows:

foo.example.com 192.0.2.1 2001:DB8::1
bar.example.com 192.0.2.2 2001:DB8::2

the following command will test these routers for responsiveness to ICMP, SSH, Telnet, HTTPS, HTTP, BGP, NTP, DNS, and SNMP probes, recording raw data into example-routers.warts:

sc_filterpolicy -p 31337 -a routers.txt -t router -o example-routers.warts

Including the name of each device in the input file is optional.

The following command will only test the routers for responsiveness to SSH:

sc_filterpolicy -p 31337 -a routers.txt -T +ssh -o example-ssh.warts

To use **`sc_filterpolicy`** to test the filtering policy of a set of servers specified in a file named servers.txt and formatted as tuples as follows:

db.example.com 192.0.2.3
db.example.com 2001::DB8::3
corp.example.com 192.0.2.4
corp.example.com 2001::DB8::4

the following command will test these servers for responsiveness to ICMP, FTP, SSH, Telnet, MySQL, RDP, HTTPS, SMB, HTTP, NTP, DNS, and SNMP probes, recording raw data into example-servers.warts:

sc_filterpolicy -p 31337 -a servers.txt -t server -o example-servers.warts -O tuples

In an input file formatted as tuples, the name (or an identifier) for each device is mandatory, and is used to ensure only one probe is sent to any one device at a time, and to collate responses from different addresses to the same device for reporting.

Once raw data has been collected, **`sc_filterpolicy`** can be used to analyse the collected data. For the example-routers.warts file, the following command dumps a summary of the data collected for each router:

```
sc_filterpolicy -r example-routers.warts
```

```
            :        T
            :        e  H
            :  I     l  T  H           S
            :  C  S  n  T  T  B  N  D  N
            :  M  S  e  P  T  G  T  N  M
            :  P  H  t  S  P  P  P  S  P
=======================================
192.0.2.1   :  O  O        O        O  O
2001:DB8::1 :  O  O        O        O  O

192.0.2.2   :  O  X
2001:DB8::2 :  O  O
```

The first router is responsive (O) for ICMP, SSH, HTTP, DNS, and SNMP probes on all addresses. The second router is responsive (O) to ICMP probes on both addresses is unresponsive (X) to SSH on the IPv4 address, but is responsive (O) to SSH on the IPv6 address and possibly represents a filtering policy that is incongruent and requires attention. Note that the empty cells in the table represent a router that was unresponsive (X) to that protocol for all addresses tested; the cells are left empty to allow the user to focus on

open and incongruent application services.

The command:

sc_filterpolicy -O incongruent -r example-routers.warts

will only show routers with an incongruent filtering policy.

SEE ALSO

J. Czyz, M. Luckie, M. Allman, and M. Bailey, *Don't Forget to Lock the Back Door! A Characterization of IPv6 Network Security Policy*, Proc. Network and Distributed Systems Security (NDSS) Conference 2016.
`scamper`(1), `sc_wartsdump`(1), `sc_warts2json`(1), `warts`(5)

AUTHORS

`sc_filterpolicy` was written by Matthew Luckie <mjl@luckie.org.nz> and Jakub Czyz.