NAME

sc_remoted — interact with a collection of remotely controlled scamper instances

SYNOPSIS

sc_remoted [**-?46D**] [**-O** *options*] [**-P** *[ip:]port*] [**-U** *directory*] [**-C** *tls-ca*] [**-c** *tls-certificate*] [**-p** *tls-privatekey*] [**-e** *pidfile*] [**-Z** *zombie-time*]

DESCRIPTION

The **sc_remoted** utility provides the ability to connect to a scamper(1) instance running remotely and interact with it by issuing commands and receiving results in warts format. The options are as follows:

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

-D causes **sc_remoted** to operate as a daemon.

-4 causes **sc_remoted** to only listen for IPv4-based connections.

-6 causes **sc_remoted** to only listen for IPv6-based connections.

-O *options*
allows the behavior of **sc_remoted** to be further tailored. The current choices for this option are:
- **allowgroup:** allow members of the unix domain socket's group to access to the unix domain sockets created by **sc_remoted**
- **allowother:** allow anyone on the system access to the unix domain sockets created by **sc_remoted**
- **debug:** print debugging messages
- **select:** use select(2) with all sockets, rather than epoll(2) or kqueue(2)
- **skipnameverification:** do not verify the monitor name, if presented, against the name in the certificate that the client presents if doing client TLS authentication.

-P *[ip:]port*
specifies the IP address and port on the local host where **sc_remoted** should listen for incoming connections. If an IP address is not specified, **sc_remoted** will listen on all available IP addresses for incoming connections.

-U *directory*
specifies the directory on the local host where unix domain sockets corresponding to remote hosts should be placed.

-C *tls-ca*
specifies the certificate authority certificate file in PEM format for **sc_remoted** to use to verify client certificates.

-c *tls-certificate*
specifies the server certificate file in PEM format to advertise to remote scamper(1) instances.

-p *tls-privatekey*
specifies the private key file in PEM format that corresponds to the certificate file. This key should have a passphrase. **sc_remoted** will prompt for the passphrase when starting up.

-e *pidfile*
specifies the name of a file to write the process ID to.

-Z *zombie-time*
specifies the length of time **sc_remoted** will retain state for a disconnected scamper(1) instance, allowing it to resume. By default **sc_remoted** retains state for 15 minutes.

EXAMPLES

The intended use of the remote control socket built into scamper(1) is as follows. A central server with IP addresses 192.0.2.1 and 2001:db8::1 runs a **sc_remoted** process listening on a port for remote scamper process, placing control sockets in a specified directory:

```
sc_remoted -P 31337 -U scamper-remote-sockets
```

Then, a remote host with IP address 198.51.100.55 runs scamper and connects to the remote controller:

```
scamper -R 192.0.2.1:31337
```

The **sc_remoted** process places a unix domain socket in the directory corresponding to the remote process. The name corresponds to the source IP address and port the remote scamper process connected to controller with. If the scamper process used source port 1025, then the unix domain socket's name will be

```
scamper-remote-sockets/198.51.100.55:1025
```

If a second remote host with IP address 2001:db8:1234::1 runs scamper and connects to the remote controller:

```
scamper -R [2001:db8::1]:31337
```

The same **sc_remoted** process will place another unix domain socket in the directory corresponding to the remote process. If the scamper process used source port 1026, then the unix domain socket's name will be

```
scamper-remote-sockets/2001:db8:1234::1.1026
```

If scamper is started with -M monitor-name, then it will pass the monitor name sc_remoted, which will use it in the unix domain socket's name. For example, if scamper is started as follows:

```
scamper -R [2001:db8::1]:31337 -M foo.bar
```

then the unix domain socket's name will be

```
scamper-remote-sockets/foo.bar-2001:db8:1234::1.1026
```

USING TRANSPORT LAYER SECURITY

sc_remoted and scamper support the use of transport layer security (TLS) using OpenSSL to authenticate and encrypt communications between **sc_remoted** and scamper. To use this support requires a public certificate signed by a certificate authority. Scamper will verify the certificate presented by **sc_remoted** and disconnect if the certificate presented by **sc_remoted** cannot be validated.

Generating a certificate that will be accepted by scamper requires you to create a certificate request and pass it for signing to a certificate authority. To generate a private key in file remotepriv.pem, and a request to sign the key in remotereq.pem:

```
openssl req -new -keyout remotepriv.pem -out remotereq.pem
```

and then send the remotereq.pem file to the certificate authority for signing. Do not send remotepriv.pem; that key must remain private to you. When openssl prompts for a passphrase, choose a passphrase that is unique and keep the passphrase secret. When your chosen certificate authority signs your private key, it will return a file which we will call remotecert.pem. Both remotecert.pem and remotepriv.pem are required parameters to **sc_remoted** to enable TLS support:

```
sc_remoted -P 31337 -U scamper-remote-sockets -c remotecert.pem -p
remotepriv.pem
```

and then run scamper as follows:

```
scamper -R example.com:31337
```

sc_remoted can also require that scamper present a certificate during the TLS handshake with the -C parameter:

```
sc_remoted -P 31337 -U scamper-remote-sockets -c remotecert.pem -p
remotepriv.pem -C remoteca.pem
```

In this case, **sc_remoted** requires that the scamper instance passes valid certificate signed by remoteca.pem, and that the certificate contains a monitor-name matching the monitor-name subsequently provided by scamper to **sc_remoted**. The scamper-side of this process looks like:

```
scamper -R example.com:31337 -O client-certfile=cert.pem -O
client-privfile=key.pem -M foo.bar
```

SIGNAL HANDLERS

sc_remoted installs handlers for two signals: SIGINT and SIGHUP. SIGINT causes **sc_remoted** to exit gracefully. SIGHUP causes **sc_remoted** to reload the TLS certificate and private key, without interrupting existing TLS connections.

SEE ALSO

scamper(1), sc_attach(1), sc_wartsdump(1), warts(5), openssl(1)

AUTHORS

sc_remoted was written by Matthew Luckie <mjl@luckie.org.nz>.