NAME

sudo_logsrvd **-** sudo event and I/O log server

SYNOPSIS

sudo_logsrvd [**-hnV**] [**-f** *file*] [**-R** *percentage*]

DESCRIPTION

sudo_logsrvd is a high-performance log server that accepts event and I/O logs from **sudo**. It can be used to implement centralized logging of **sudo** logs. The server has two modes of operation: local and relay. By default, **sudo_logsrvd** stores the logs locally but it can also be configured to relay them to another server that supports the sudo_logsrv.proto(5) protocol.

When not relaying, event log entries may be logged either via syslog(3) or to a local file. I/O Logs stored locally by **sudo_logsrvd** can be replayed via the sudoreplay(8) utility in the same way as logs generated directly by the **sudoers** plugin.

The server also supports restarting interrupted log transfers. To distinguish completed I/O logs from incomplete ones, the I/O log timing file is set to be read-only when the log is complete.

Configuration parameters for **sudo_logsrvd** may be specified in the sudo_logsrvd.conf(5) file or the file specified via the **-f** option.

sudo_logsrvd rereads its configuration file when it receives SIGHUP and writes server state to the debug file (if one is configured) when it receives SIGUSR1.

The options are as follows:

-f *file*, **--file**=*file*
Read configuration from *file* instead of the default, */etc/sudo_logsrvd.conf*.

-h, **--help** Display a short help message to the standard output and exit.

-n, **--no-fork** Run **sudo_logsrvd** in the foreground instead of detaching from the terminal and becoming a daemon.

-R *percentage*, **--random-drop**=*percentage*
For each message, there is a *percentage* chance that the server will drop the connection. This is only intended for debugging the ability of a client to restart a connection.

-V, **--version**

Print the **sudo_logsrvd** version and exit.

Securing server connections

The I/O log data sent to **sudo_logsrvd** may contain sensitive information such as passwords and should be secured using Transport Layer Security (TLS). Doing so requires having a signed certificate on the server and, if *tls_checkpeer* is enabled in sudo_logsrvd.conf(5), a signed certificate on the client as well.

The certificates can either be signed by a well-known Certificate Authority (CA), or a private CA can be used. Instructions for creating a private CA are included below in the *EXAMPLES* section.

Debugging sudo_logsrvd

sudo_logsrvd supports a flexible debugging framework that is configured via Debug lines in the sudo.conf(5) file.

For more information on configuring sudo.conf(5), refer to its manual.

FILES

/etc/sudo.conf	Sudo front-end configuration
/etc/sudo_logsrvd.conf	Sudo log server configuration file
/var/log/sudo_logsrvd/incoming	Directory where new journals are stored when the *store_first relay* setting is enabled.
/var/log/sudo_logsrvd/outgoing	Directory where completed journals are stored when the *store_first relay* setting is enabled.
/var/log/sudo-io	Default I/O log file location
/var/run/sudo/sudo_logsrvd.pid	Process ID file for **sudo_logsrvd**

EXAMPLES

Creating self-signed certificates

Unless you are using certificates signed by a well-known Certificate Authority (or a local enterprise CA), you will need to create your own CA that can sign the certificates used by **sudo_logsrvd**, **sudo_sendlog**, and the **sudoers** plugin. The following steps use the openssl(1) command to create keys and certificates.

Initial setup

First, we need to create a directory structure to store the files for the CA. We'll create a new directory hierarchy in */etc/ssl/sudo* for this purpose.

```
# mkdir /etc/ssl/sudo
# cd /etc/ssl/sudo
# mkdir certs csr newcerts private
# chmod 700 private
# touch index.txt
# echo 1000 > serial
```

The serial and index.txt files are used to keep track of signed certificates.

Next, we need to make a copy of the openssl.conf file and customize it for our new CA. The path to openssl.cnf is system-dependent but */etc/ssl/openssl.cnf* is the most common location. You will need to adjust the example below if it has a different location on your system.

```
# cp /etc/ssl/openssl.cnf .
```

Now edit the *openssl.cnf* file in the current directory and make sure it contains "ca" and "CA_default" sections. Those sections should include the following settings:

```
[ ca ]
default_ca      = CA_default

[ CA_default ]
dir             = /etc/ssl/sudo
certs           = $dir/certs
database        = $dir/index.txt
certificate     = $dir/cacert.pem
serial          = $dir/serial
```

If your *openssl.conf* file already has a "CA_default" section, you may only need to modify the "dir" setting.

Creating the CA key and certificate

In order to create and sign our own certificates, we need to create a private key and a certificate for the root of the CA. First, create the private key and protect it with a pass phrase:

```
# openssl genrsa -aes256 -out private/cakey.pem 4096
# chmod 400 private/cakey.pem
```

Next, generate the root certificate, using appropriate values for the site-specific fields:

```
# openssl req -config openssl.cnf -key private/cakey.pem \
    -new -x509 -days 7300 -sha256 -extensions v3_ca \
    -out cacert.pem

Enter pass phrase for private/cakey.pem:
You are about to be asked to enter information that will be
incorporated into your certificate request.
What you are about to enter is what is called a Distinguished Name
or a DN.
There are quite a few fields but you can leave some blank.
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:Colorado
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:sudo
Organizational Unit Name (eg, section) []:sudo Certificate Authority
Common Name (e.g., server FQDN or YOUR name) []:sudo Root CA
Email Address []:

# chmod 444 cacert.pem
```

Finally, verify the root certificate:

```
# openssl x509 -noout -text -in cacert.pem
```

Creating and signing certificates

The server and client certificates will be signed by the previously created root CA. Usually, the root CA is not used to sign server/client certificates directly. Instead, intermediate certificates are created and signed with the root CA and the intermediate certs are used to sign CSRs (Certificate Signing Request). In this example we'll skip this part for simplicity's sake and sign the CSRs with the root CA.

First, generate the private key without a pass phrase.

```
# openssl genrsa -out private/logsrvd_key.pem 2048
# chmod 400 private/logsrvd_key.pem
```

Next, create a certificate signing request (CSR) for the server's certificate. The organization name must match the name given in the root certificate. The common name should be either the server's IP address or a fully qualified domain name.

```
# openssl req -config openssl.cnf -key private/logsrvd_key.pem -new \
    -sha256 -out csr/logsrvd_csr.pem

Enter pass phrase for private/logsrvd_key.pem:
You are about to be asked to enter information that will be
incorporated into your certificate request.
What you are about to enter is what is called a Distinguished Name
or a DN.
There are quite a few fields but you can leave some blank.
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:Colorado
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:sudo
Organizational Unit Name (eg, section) []:sudo log server
Common Name (e.g., server FQDN or YOUR name) []:logserver.example.com
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

Now sign the CSR that was just created:

```
# openssl ca -config openssl.cnf -days 375 -notext -md sha256 \
    -in csr/logsrvd_csr.pem -out certs/logsrvd_cert.pem

Using configuration from openssl.cnf
Enter pass phrase for ./private/cakey.pem:
Check that the request matches the signature
Signature ok
```

```
Certificate Details:
        Serial Number: 4096 (0x1000)
        Validity
            Not Before: Nov 11 14:05:05 2019 GMT
            Not After : Nov 20 14:05:05 2020 GMT
        Subject:
            countryName               = US
            stateOrProvinceName       = Colorado
            organizationName          = sudo
            organizationalUnitName    = sudo log server
            commonName                = logserve.example.com
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Comment:
                OpenSSL Generated Certificate
            X509v3 Subject Key Identifier:
                4C:50:F9:D0:BE:1A:4C:B2:AC:90:76:56:C7:9E:16:AE:E6:9E:E5:B5
            X509v3 Authority Key Identifier:
                keyid:D7:91:24:16:B1:03:06:65:1A:7A:6E:CF:51:E9:5C:CB:7A:95:3E:0C

Certificate is to be certified until Nov 20 14:05:05 2020 GMT (375 days)
Sign the certificate? [y/n]:y

1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
```

Finally, verify the new certificate:

```
# openssl verify -CAfile cacert.pem certs/logsrvd_cert.pem
certs/logsrvd_cert.pem: OK
```

The */etc/ssl/sudo/certs* directory now contains a signed and verified certificate for use with **sudo_logsrvd**.

To generate a client certificate, repeat the process above using a different file name.

Configuring sudo_logsrvd to use TLS

To use TLS for client/server communication, both **sudo_logsrvd** and the **sudoers** plugin need to be configured to use TLS. Configuring **sudo_logsrvd** for TLS requires the following settings, assuming the same path names used earlier:

```
# Listen on port 30344 for TLS connections to any address.
listen_address = *:30344(tls)

# Path to the certificate authority bundle file in PEM format.
tls_cacert = /etc/ssl/sudo/cacert.pem

# Path to the server’s certificate file in PEM format.
tls_cert = /etc/ssl/sudo/certs/logsrvd_cert.pem

# Path to the server’s private key file in PEM format.
tls_key = /etc/ssl/sudo/private/logsrvd_key.pem
```

The root CA cert (*cacert.pem*) must be installed on the system running **sudo_logsrvd**. If peer authentication is enabled on the client, a copy of *cacert.pem* must be present on the client system too.

SEE ALSO

sudo.conf(5), sudo_logsrvd.conf(5), sudoers(5), sudo(8), sudo_sendlog(8), sudoreplay(8)

AUTHORS

Many people have worked on **sudo** over the years; this version consists of code written primarily by:

Todd C. Miller

See the CONTRIBUTORS.md file in the **sudo** distribution (https://www.sudo.ws/about/contributors/) for an exhaustive list of people who have contributed to **sudo**.

BUGS

If you believe you have found a bug in **sudo_logsrvd**, you can submit a bug report at https://bugzilla.sudo.ws/

SUPPORT

Limited free support is available via the sudo-users mailing list, see https://www.sudo.ws/mailman/listinfo/sudo-users to subscribe or search the archives.

DISCLAIMER

sudo_logsrvd is provided "AS IS" and any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. See the