

NAME

logksi verify - Verify log file using its log signature.

SYNOPSIS

logksi verify *logfile* [*logfile.logsig*] [*more_options*]

logksi verify --ver-int *logfile* [*logfile.logsig*] [*more_options*]

logksi verify --ver-cal *logfile* [*logfile.logsig*] **-X** *URL* [--ext-user *user* --ext-key *key*] [*more_options*]

logksi verify --ver-key *logfile* [*logfile.logsig*] **-P** *URL* [--cnstr *oid=value*]... [*more_options*]

logksi verify --ver-pub *logfile* [*logfile.logsig*] **--pub-str** *pubstr* [-x **-X** *URL* [--ext-user *user* --ext-key *key*]] [*more_options*]

logksi verify --ver-pub *logfile* [*logfile.logsig*] **-P** *URL* [--cnstr *oid=value*]... [-x **-X** *URL* [--ext-user *user* --ext-key *key*]] [*more_options*]

DESCRIPTION

Verifies log file *logfile*. The log signature is by default expected to be found in *logfile.logsig*. If this is not the case, the correct log signature file name must be given after the *logfile*.

For each signed log block the root hash of the block is recomputed and then verified using the KSI signature of that block.

KSI signature can be verified using any of its standard verification policies:

- Internal verification (**--ver-int**). Only internal consistency of the signature is checked and no trust anchor is used and no external resources are needed. This check is also performed as the first step in all other policies.
- Calendar-based verification (**--ver-cal**). Signature is verified against calendar blockchain database at the KSI Extender. Verification is done by checking that the output hash value computed from the aggregation hash chain matches the corresponding entry in the calendar blockchain. Access to KSI Extender is needed.
- Key-based verification (**--ver-key**). Signature must contain a calendar hash chain and a calendar authentication record that can be verified against the signing certificates. To be able to perform key-based verification user must have an access to a trusted KSI publications file with signing certificates in it.
- Publication-based verification (**--ver-pub**). Signature must be extended to a time of publication and contain a publication record unless automatic extension of the signature is enabled with **-x**. Verification is done by checking that the publication record in the signature matches a publication in the publications file or the publication string given on the command line. Publications file or publication string retrieved from printed media is needed.

It must be noted that only publication-based verification should be preferred in long term as it does not rely on any keys and trusted services. The other policies can be used temporarily when the signature is created and there is not yet a publication to extend the signature to.

OPTIONS**--ver-int**

Perform internal verification.

--ver-cal

Perform calendar-based verification (use extending service).

--ver-key

Perform key-based verification.

--ver-pub

Perform publication-based verification (use with **-x** to permit extending).

-x

Permit to use extender for publication-based verification. See **logksi-extended**(1) for details.

- X URL**
Specify the extending service (KSI Extender) URL.
- ext-user *user***
Specify the username for extending service.
- ext-key *key***
Specify the HMAC key for extending service.
- pub-str *str***
Specify the publication string to verify with.
- P URL** Specify the publications file URL (or file with URI scheme 'file://').
- cnstr *oid=value***
Specify the OID of the PKI certificate field (e.g. e-mail address) and the expected value to qualify the certificate for verification of publications file's PKI signature. At least one constraint must be defined. All values from lower priority source are ignored (see **logksi-conf(5)** for more information).
- For more common OIDs there are convenience names defined:
- **E** or **email** for OID 1.2.840.113549.1.9.1
 - **CN** or **cname** for OID 2.5.4.3
 - **C** or **country** for OID 2.5.4.6
 - **O** or **org** for OID 2.5.4.10
- V file** Specify the certificate file in PEM format for publications file verification. All values from lower priority source are ignored (see **logksi-conf(5)**).
- d** Print detailed information about processes and errors to *stderr*.
- conf file**
Read configuration options from given file. It must be noted that configuration options given explicitly on command line will override the ones in the configuration file (see **logksi-conf(5)** for more information).
- log file**
Write libksi log to given file. Use '-' as file name to redirect log to *stdout*.

EXIT STATUS

See **logksi(1)** for more information.

EXAMPLES

In the following examples it is assumed that KSI service configuration options (URLs, access credentials) are defined. See **logksi-conf(5)** for more information.

- 1 To verify */var/log/secure* using only internal verification of KSI signatures:

```
logksi verify --ver-int /var/log/secure
```

- 2 To verify */var/log/secure* using publication-based verification of the KSI signatures with specified publication string:

```
logksi verify --ver-pub /var/log/secure --pub-str AAAAAA-CWYEKQ-AAIYPA-UJ4GRT-HXMFBE-OTB4AB-XH3PT3-KNIKGV-PYCJXU-HL2TN4-RG6SCC-3ZGSBM
```

- 3 To verify */var/log/secure* using publication-based verification of the KSI signatures and publications file which is auto-downloaded and verified based on the default configuration options:

```
logksi verify --ver-pub /var/log/secure
```

- 4 To verify */var/log/secure* using publication-based verification of the KSI signatures and possibly extending them on the fly:

```
logksi verify --ver-pub /varlog/secure -x
```

5 To verify */var/log/secure* using any policy possible, depending on the current state of the signatures:

logksi verify */var/log/secure*

ENVIRONMENT

Use the environment variable **KSI_CONF** to define the default configuration file. See **logksi-conf(5)** for more information.

AUTHOR

Guardtime AS, <http://www.guardtime.com/>

SEE ALSO

logksi(1), **logksi-sign(1)**, **logksi-integrate(1)**, **logksi-extend(1)**, **logksi-conf(5)**