Site: **EJBCA Wiki** at http://wiki.ejbca.org
Source page: **Logsigning** at http://wiki.ejbca.org/logsigning

Logsigning

Fold

Table of Contents

Log signing
Signing log files
Exporting signed logs in CSV format
Enabling log signing using logrotate and openssl
Enabling log signing using logrotate and TSA
Enabling log signing using TSA in JBoss
Enabling log signing using external script in JBoss
Verifying timestamps
Internal log signing in EJBCA

Log signing

Some suggestions are given here for the signing of log-files.

Signing log files

There are several ways to sign log files.
The most common is to sign the logs as they are beeing archived (rotated).

Some examples:

- Export signed logs in CSV format using the admin GUI
- If logs are sent to syslog, the syslog rotation can sign files when they are rotated
  - Using openssl
  - Using a time stamp server (TSA)
- If logs are stored by JBoss, a special log4j appender can be used to sign files when they are rotated
  - Using a time stamp server (TSA)

These are general ways, which works for any application that either uses syslog or JBoss.

This document describes these three options.

Exporting signed logs in CSV format

In the 'View Log' functionality you can export the logs you view in the admin-GUI.
You export logs by clicking the 'Export as csv' button. If you want the logs signed, you can select a CA in the drop down list besides the button. If you don't select a CA, the exported file is in plain csv format. If you select a CA, the exported file is a signed CMS/PKCS#7 file returned as 'logexport.p7m'.

The signing certificate used is a special CMS signer certificate signed by the selected CA. This signing service is created when the CA is created, but it is by default disabled.
You can enable the CMS Signing service in the 'Edit Certificate Authorities' view by editing a CA and activating the 'CMS Service'.

A signed 'logexport.p7m' can be verified with for example:
openssl smime -verify -inform DER -in logexport.p7m -CAfile adminca1.pem
where adminca1.pem is the certificate of the CA that has the CMS Service used to sign the file.

Enabling log signing using logrotate and openssl

Save and enable the below logrotate configuration for syslog, is should normaly be stored as the file syslog in the directory /etc/logrotate.d.
If there is a previous configuration for syslog, you should remove this, perhaps merge some changes if you like. Syslog is normally not rotated in most linux distributions.

After enabling the configuration you should make sure that logrotate is run as often as you like. Normally is is run every night. If you want to run it more often for syslog, you can add a crontab entry to run it every hour for example.
In that cas call logrotate with the specified configuration file:
logrotate syslog.conf

Note, that when stored in logrotate.d, the syslog configuration will be called when logrotate is normally run.

```
/var/log/syslog {
 rotate 5
 postrotate
 /usr/bin/killall -HUP syslogd
 endscript
 lastaction
 OPENSSL=/usr/bin/openssl
 LOGFILE=/var/log/syslog
 FILE="$LOGFILE.`date +%F.%H:%M:%S`.log"
 SIGNATUREFILE="$FILE.sign"
 cp $LOGFILE.1 "$FILE"
 PRIVATEKEY="/etc/logsigner/qc1.priv"
 $OPENSSL dgst -sign $PRIVATEKEY -sha1 $FILE > $SIGNATUREFILE
 endscript
 }
```

qc1.priv is the private key used for signing the logfile.

See the simple shell script below for
openssl command to convert a pkcs12 file to the public key and private key files that openssl uses for signing and verifying.

To verify, you can issue the command:
log-sign.sh verify qc1.pub syslog.2006-07-29.11:06:27.log syslog.2006-07-29.11:06:27.log.sign
Where the dates and time correspond to the file you are verifying off-course.

log.sign.sh is a simple shell script use to manually sign and verify files:

```
#!/bin/bash

  # Openssl command for signing and verifying
  #openssl dgst -sign superadmin.key -sha1 test.txt > test.txt.sign
  #openssl dgst -verify superadmin.pubkey -signature test.txt.sign -sha1 test.txt

  # Openssl command to convert a p12 file to cert and key files in pem
  # First cert:
  #openssl pkcs12 -in qc1.p12 -nodes -nokeys -clcerts -out qc1.pem
  # Then public key
  #openssl x509 -in qc1.pem -pubkey -noout > qc1.pub
  # Then private key:
  #openssl pkcs12 -in qc1.p12 -nodes -nocerts -out qc1.priv

  OPENSSL=/usr/bin/openssl
  SCRIPTNAME=`basename $0`
  OPTION=$1
  DATE=`date +"%Y-%m-%d"`

  if [ "$OPTION" = "sign" ]; then
  PRIVATEKEY="$2"
  FILE="$3"
  SIGNATUREFILE="$4"
  $OPENSSL dgst -sign $PRIVATEKEY -sha1 $FILE > $SIGNATUREFILE

  exit 0

  elif [ "$OPTION" = "verify" ]; then
  PUBLICKEY="$2"
  FILE="$3"
  SIGNATUREFILE="$4"

  $OPENSSL dgst -verify $PUBLICKEY -signature $SIGNATUREFILE -sha1 $FILE
  exit 0

  else
  echo "Usage:"
  echo "To sign files you have to edit the script and add the files you want signed."
  echo "$SCRIPTNAME sign   "
  echo "$SCRIPTNAME verify   "
  exit 0
  fi
```

Enabling log signing using logrotate and TSA

The same approach is taken for signing using the TSA server instead of openssl.
In this example the timeStampClient and TSA from PrimeKey is used. After building 'signserver' the timeStampClient.jar and other needed jar files is in the dist-client directory.

The new logrotate configuration:

```
/var/log/syslog {
  rotate 5
  postrotate
  /usr/bin/killall -HUP syslogd
  endscript
  lastaction
  JAVA_HOME=/usr/local/java
  TSACLIENTDIR=/home/tomas/dev/workspace/signserver/dist-client
  LOGFILE=/var/log/syslog
  FILE="$LOGFILE.`date +%F.%H:%M:%S`.log"
  SIGNATUREFILE="$FILE.sign"
  cp $LOGFILE.1 "$FILE"
  $JAVA_HOME/bin/java -jar "$TSACLIENTDIR/timeStampClient.jar" -url "http://127.0.0.1:8080/s
  endscript
  }
```

Replace the ip-address and port (127.0.0.1:8080) with the ip and port of your actual TSA.

And the new shell script for verification (or manual signing).
log-sign-tsa.sh

```
#!/bin/bash

  TSACLIENTDIR=/usr/local/signserver/dist-client
  SCRIPTNAME=`basename $0`
  OPTION=$1
  DATE=`date +"%Y-%m-%d"`

  if [ "$OPTION" = "sign" ]; then
  FILE="$2"
  SIGNATUREFILE="$3"
  TSAURL="$4"
  java -jar "$TSACLIENTDIR/timeStampClient.jar" -url $TSAURL -infile $FILE -outrep $SIGNATUR

  exit 0

  elif [ "$OPTION" = "verify" ]; then
  PUBLICKEY="$2"
  FILE="$3"
  SIGNATUREFILE="$4"
  java -jar "$TSACLIENTDIR/timeStampClient.jar" -verify -inrep $SIGNATUREFILE -signerfile $P
  sha1sum $FILE

  exit 0

  else
  echo "Usage:"
  echo "To sign files you have to edit the script and add the files you want signed."
  echo "$SCRIPTNAME sign   "
  echo "$SCRIPTNAME verify   "
  exit 0
  fi
```

When verifying the time stamp token, the sha1 hash from the time stamp token (signed) and the calculated sha1 hash of the file is printed.
You must compare them manually.

Enabling log signing using TSA in JBoss

There is an implementation that creates a signed time stamp, from a TSA, on JBoss log files when they are rolled-over. This can be every hour for example.

- Stop JBoss
- Issue the command: ant jbosslogsigning
- Copy dist/ejbcalogsigning.jar to jboss.home/server/default/lib
- Copy lib/bctsp-jdk<your java version> to jboss.home/server/default/lib
- Copy lib/bcprov-jdk<your java version> to jboss.home/server/default/lib
- Configure jboss.home/server/default/conf/log4j.xml, according to below
- Start JBoss

Logs will be rolled over at the selected interval, and a file with the same name as the log file, with ending .tsp will be created. The tsp file contains the complete base64 encoded response from the TSA.

NOTE!!!
Unforturnately you have to add bcprov to JBoss's lib directory.
This means that you have to keep this jar in sync with the EJBCA release, if you upgrade EJBCA.
When upgrading EJBCA, you should also copy a new version of bcprov from EJBCA to JBoss's lib directory.

If you happen to start up JBoss at exactly the time when a log file rollover occurs, it is normal in the startup log with an error message like:
SigningDailyRollingFileAppender: No reply bytes received, is TSA down?
The log appender should recover from this and resume normal opperations.

The section:

```
<!-- A time/date based rolling appender -->
     <appender name="FILE" class="org.jboss.logging.appender.DailyRollingFileAppender">
        <errorHandler class="org.jboss.logging.util.OnlyOnceErrorHandler"/>
        <param name="File" value="${jboss.server.log.dir}/server.log"/>
        <param name="Append" value="false"/>

        <!-- Rollover at midnight each day -->
        <param name="DatePattern" value="'.'yyyy-MM-dd"/>

        <!-- Rollover at the top of each hour
        <param name="DatePattern" value="'.'yyyy-MM-dd-HH"/>
        -->

        <layout class="org.apache.log4j.PatternLayout">
           <!-- The default pattern: Date Priority [Category] Message\n -->
           <param name="ConversionPattern" value="%d %-5p [%c] %m%n"/>

           <!-- The full pattern: Date MS Priority [Category] (Thread:NDC) Message\n
           <param name="ConversionPattern" value="%d %-5r %-5p [%c] (%t:%x) %m%n"/>
            -->
        </layout>
     </appender>
```

should be replaced with:

```
<!-- A time/date based rolling appender that signs rollev over log files using a time stamp
  <appender name="FILE" class="org.ejbca.appserver.jboss.SigningDailyRollingFileAppender">
  <errorHandler class="org.jboss.logging.util.OnlyOnceErrorHandler"/>
  <param name="File" value="${jboss.server.log.dir}/server.log"/>
  <param name="Append" value="false"/>
  <param name="SignMethod" value="tsa"/>
  <param name="TsaUrl" value="http://127.0.0.1:8080/signserver/tsa?signerId=1"/>

  <!-- Rollover at midnight each day -->
  <param name="DatePattern" value="'.'yyyy-MM-dd"/>

  <!-- Rollover at the top of each hour
  <param name="DatePattern" value="'.'yyyy-MM-dd-HH"/>
  -->

  <!-- Rollover at the beginning of every minute
  <param name="DatePattern" value="'.'yyyy-MM-dd-HH-mm"/>
```

```
   -->

   <layout class="org.apache.log4j.PatternLayout">
   <!-- The default pattern: Date Priority [Category] Message
   -->
   <param name="ConversionPattern" value="%d %-5p [%c] %m%n"/>

   <!-- The full pattern: Date MS Priority [Category] (Thread:NDC) Message

   <param name="ConversionPattern" value="%d %-5r %-5p [%c] (%t:%x) %m%n"/>
   -->
   </layout>
   </appender>
```

replace the ip-address and port '127.0.0.1:8080' with the IP and port of your actual TSA.
The TSA should listen on the HTTP protocol as defined in RFC 3161.

Enabling log signing using external script in JBoss

There is an implementation that runs an external script when JBoss log files
when they are rolled-over. This can be every hour for example.

- Stop JBoss
- Issue the command: ant jbosslogsigning
- Copy dist/ejbcalogsigning.jar to jboss.home/server/default/lib
- Configure jboss.home/server/default/conf/log4j.xml, according to below
- Create the script that is to be run (see example below)
- Start JBoss

As for the SigningDailyRollingFileAppender above, replace the default section in log4j.xml with:

```
<!-- A time/date based rolling appender that signs rollev over log files using a time stamp
   <appender name="FILE" class="org.ejbca.appserver.jboss.ScriptrunningDailyRollingFileAppen
      <errorHandler class="org.jboss.logging.util.OnlyOnceErrorHandler"/>
      <param name="File" value="${jboss.server.log.dir}/server.log"/>
      <param name="Append" value="false"/>
      <param name="Script" value="/home/jboss/log-sign-tsa-jboss.sh"/>

      <!-- Rollover at midnight each day -->
      <param name="DatePattern" value="'.'yyyy-MM-dd"/>

      <!-- Rollover at the top of each hour
      <param name="DatePattern" value="'.'yyyy-MM-dd-HH"/>
      -->

      <!-- Rollover at the beginning of every minute
      <param name="DatePattern" value="'.'yyyy-MM-dd-HH-mm"/>
      -->

      <layout class="org.apache.log4j.PatternLayout">
         <!-- The default pattern: Date Priority [Category] Message\n -->
         <param name="ConversionPattern" value="%d %-5p [%c] %m%n"/>

         <!-- The full pattern: Date MS Priority [Category] (Thread:NDC) Message\n
         <param name="ConversionPattern" value="%d %-5r %-5p [%c] (%t:%x) %m%n"/>
          -->
      </layout>
   </appender>
```

Where /home/jboss/log-sign-tsa-jboss.sh is your script that should be run to do the signing.
signlog.sh must take one argument, which is the file to sign.

And the new shell script for verification (or manual signing).
log-sign-tsa-jboss.sh

```
#!/bin/bash
```

```
TSACLIENTDIR=/usr/local/signserver/dist-client
TSAURL="http://127.0.0.1:8080/signserver/tsa?signerId=1"
SCRIPTNAME=`basename $0`

FILE="$1"
SIGNATUREFILE="$FILE.tsa"
java -jar "$TSACLIENTDIR/timeStampClient.jar" -url $TSAURL -infile $FILE -outrep $SIGNATUREI

exit 0
```

In the script you must replace TSACLIENTDIR with the path to your TSA client and TSAURL with the url to your TSA.

Verifying timestamps

You can verify time stamps with a time stamp client, for example the one that comes with PrimeKeys TSA. After building 'signserver' the timeStampClient.jar and other needed jar files is in the dist-client directory.

```
java -jar timeStampClient.jar -verify -signerfile tsa1.pem -inrep server.log.2006-07-28-18-5
Token was validated successfully.
Token was generated on: Fri Jul 28 18:59:00 CEST 2006
Token hash alg: SHA1
MessageDigest=1a02dc7e05d06df45d2e0f74da502513852064d5
```

```
sha1sum server.log.2006-07-28-18-58
1a02dc7e05d06df45d2e0f74da502513852064d5 *server.log.2006-07-28-18-58
```

If the hashes from the time stamp token (MessageDigest) matches the hash from sha1sum, the file is verified, i.e. has not been changed since the time stamp token was generated.

Internal log signing in EJBCA

See EJBCA documentation at http://www.ejbca.org/

page revision: 3, last edited: 29 May 2013, 14:48 (650 days ago)