1 Trusted Execution

1.1 Background

Security of a system is multifaceted issue and requires different tools and frameworks to manage the threats. Some of the security mechanisms provide for protection against attacks (e.g.: network traffic monitoring for attacks and taking defensive actions), and some provide active monitoring against any successful attacks and the ability to denying the attacker of powers, privileges and data access (for example, Encrypting data, stop execution of unauthorised code, etc.).

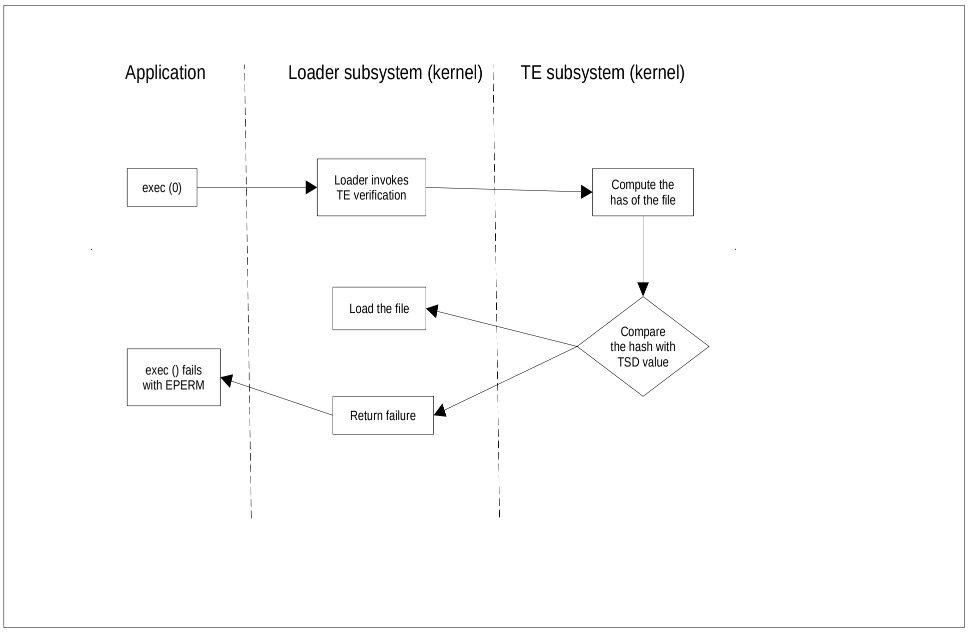
It is necessary that the system administrator be able to verify at any point in time that the system has not been compromised. Additionally it is required that system integrity mechanism provides means to spoil attempts by the attacker to compromise the baseline system integrity information itself. To summarise the requirements for a good system integrity function:

- Integrity Measurement: Provide administrator to detect changes to the system. The changes are identified by comparing the current state to a well known previous state (also called as the baseline).
- Lock down: Provide means to administrator to lock down the information established as baseline. This lock down will prevent an intruder from modifying the state of the system and the recorded Baseline so that the admin will not be able to detect these modifications.
- Monitor and Protect: Provide means to monitor executions of executables, libraries, kernel extensions etc.

Trusted Execution (TE) is an AIX in-built security feature for maintaining the integrity of files and executables on the system. By default, AIX ships a database of system files with their trusted signatures. TE checks any deviation from these signatures and reports it as an integrity breach.

The AIX Trusted Execution (TE) feature will be used for integrity check of critical files. TE can be used in offline mode or online mode or both. It is usually recommended to initially use TE in offline mode and after monitoring turn it online (See figure 1).

Figure 1


Application
Loader subsystem (kernel)
TE subsystem (kernel)
exec (0)
Loader invokes TE verification
Compute the has of the file
Compare the hash with TSD value
Load the file
exec () fails with EPERM
Return failure

To check the integrity of system against the Trusted Signature Database (TSD) use the `trustchk` command. This is also called running a system integrity check. The `trustchk` command can be used to audit the integrity state of the file definitions in the TSD against the actual files. It is recommended that this is done periodically (automated by cron, or run manually). It is recommended that the periodic audit will use the tree option (See section 1.2 - Trusted Execution – audit)

The following files will be audited TE:

Server	**File Names**
All	Default entries in TSD
Customer to provide	Examples: • BPM? • Web Server with DB2 • BigFix

The following TE policies will be enforced:

Policy	Description	Value (ON / OFF)
TE	Enables or disables Trusted Execution. All other policies can only be activated if TE is set to ON	ON (and only turned off for patching or upgrades)
CHKEXEC	Checks the integrity of executable files that belong to the TSD before starting them	ON
CHKSCRIPT	Checks the integrity of shell scripts that belong to the TSD before starting them	ON
CHKKERNEXT	Checks the integrity of the kernel extensions that belong to the TSD before loading them	ON
CHKSHLIB	Checks the integrity of shared libraries that belong to the TSD before loading them	ON
LOCK_KERN_POLICIES	If this policy is enabled, then all other policies will be locked. Reboot is required to change this policy	OFF
STOP_ON_CHKFAIL	Stops the loading of files whose integrity check fails	OFF until review of all application binaries/libraries/scripts completed
STOP_UNTRUSTD	Stops the loading of files that are not in the TSD. TROJAN: This will stop loading of files that are not in the TSD AND files that match the properties of a trojan file (See below)	OFF until review of all application binaries/libraries/scripts completed.
TEP	Sets the value of Trusted Execution path, and enables or disables it. When this policy is enabled, the files belonging to only these directory paths are allowed to be started	OFF

Policy	Description	Value (ON / OFF)
TLP	Sets the value of Trusted Library path, and enables or disables it. When this policy is enabled, the libraries belonging to only these directory paths can be loaded	OFF
TSD_LOCK	Disallows opening of a TSD file (/etc/security/tsd/tsd.dat) in write mode to disable editing of the TSD.	OFF
TSD_FILES_LOCK	Disables opening of files belonging to the TSD in write mode. WARNING: This means that files defined as VOLATILE cannot be modifed. EXVOL: Disables the opening of only the nonvolatile files that belong to the TSD in write mode. The volatile files can be changed.	OFF

For example

```
# trustchk -p tep
TEP=OFF
TEP=/usr/bin:/usr/sbin:/etc:/bin:/sbin:/sbin/helpers/jfs2:/usr/
lib/instl:/usr/ccs/bin:/usr/lib:/usr/lib/security:/etc/security
# trustchk -p tlp
TLP=OFF
TLP=/usr/lib:/usr/ccs/lib:/lib:/var/lib:/usr/lib/drivers:/usr/lib/
security:/usr/lib/nls/loc
```

1.2 Trusted Execution – audit

As before we suggest that a periodical integrity check (`trustchk -n ALL)` will be run as a cron job and the output will be stored in the log directory under [TBD] directory.

Note: -n Specifies the auditing mode, and indicates that the errors are to be reported. Any discrepancy between the attributes in the TSD and the actual file parameters are printed to the stderr. error file.
To check all of the entries in the TSD, use the “ALL” parameter.

To scan the entire system or directories for TROJAN detection, use with tree parameter.

The following can be configured in cron

```
0 15 0 0 0 /usr/sbin/trustchk -n all
```

Example of running an audit:

```
# trustchk -n ALL
trustchk: /etc/security/rtc/rtcd.conf: Verification of attributes
failed: mode
trustchk: /var/adm/cron/cron.deny: Verification of attributes
failed: owner group
trustchk: /usr/bin/rexec: Verification of attributes failed: mode
trustchk: /usr/bin/rdist: Verification of attributes failed: mode
```

Finding trojan horses
The trustchk command can also be used to scan the system for any executables that are not part of the TSD, but suspect from a privilege escalation point of view. Files will be reported as suspect if they:

- Have setuid or setgid set, owner root or group security, but not in the TSD
- Are owned by root and not in the TSD
- Are a privileged command (RBAC) and not in the TSD
- Are a symbolic link to a privileged command (RBAC) and not in the TSD

You can also block the execution of files that match these properties by setting STOP_UNTRUSTD=TROJAN.

1.3 How to configure TE policies:

To check TSD protection, run:

```
# trustchk -p
TE=ON
CHKEXEC=ON
CHKSHLIB=OFF
CHKSCRIPT=ON
CHKKERNEXT=ON
STOP_UNTRUSTD=OFF
STOP_ON_CHKFAIL=OFF
LOCK_KERN_POLICIES=OFF
TSD_FILES_LOCK=OFF
TSD_LOCK=OFF
TEP=OFF
TLP=OFF
```

To enable TSD protection, run:

```
trustchk -p TSD_LOCK=ON TE=ON
```

Note: trustchk -p tsd_lock=on te=on, works as well, ie case insensitive

Once the TSD has been configured it is vital that it is secured – this can be done by using a centralised copy (LDAP) or configuring TSD_LOCK=ON. A backup copy of the TSD can be made to a secure location and then used for integrity checking as follows:

```
trustchk -F <my_tsd_copy> -n ALL
```

1.4 How to add non IBM supplied binary to the TSD

You can add your own binaries to the TSD to extend the auditing and control of TE. All that is required is to set up a private key and a certificate, which can be used to generate the checksums for the TSD.

As openssl creates its keys and certificates in privacy enhanced mail security certificate (PEM) format, they need to converted into ASN.1/PKCS8/DER (distinguished encoding rules) format to become usable for TE. This can be quickly done by using the following commands:

First generate a 2048-bit private key in PEM format.

```
# openssl genrsa -out TEprivkey.pem 2048
```

Then create the public key and certificate that lasts approximately ten years:

```
# openssl req -new -x509 -key TEprivkey.pem -outform DER -out
TEcert.der -days 3650
```

Then convert the private key from PEM into DER format

```
# openssl pkcs8 -inform PEM -in TEprivkey.pem -topk8 -nocrypt -
outform DER -out TEprivkey.der
```

After the conversion, the private key in PEM format is no longer needed. You only need privkey.der and cert.der for the following examples. Now you can add any file you would like to the TSD by simply issuing:

```
# trustchk -s TEprivkey.der -v TEcert.der -a /path/to/binary
```

Note: It is not possible to modify an existing entry in the TSD if the binary is changed. The record needs to be deleted then recreated.

For example

```
# trustchk -d /path/to/binary
# trustchk -s privkey.der -v cert.der -a /path/to/binary
```

To add a file as a volatile file to the TSD using same pair of private key and certificate in the previous example, enter the following command:

```
# trustchk -s privkey.der -v cert.der -a /path/to/binary
size=VOLATILE
```

Or if file already in TSD

```
# trustchk –a my_textfile size=VOLATILE
```

Note: As the hash value of a volatile file is not checked, there is no need to provide keys

Warning: If TSD_FILES_LOCK policy is set on, then modifications to “VOLATILE” files is not allowed.

To check the integrity of a command (for example /usr/bin/mv)

```
trustchk -q /usr/bin/mv
/usr/bin/mv:
        owner = bin
        group = bin
        mode = 555
        type = FILE
        hardlinks =
        symlinks =
        size = 22332
        cert_tag = 49424d4149583a31324331342d33314332303a324b3a41
        signature =
7fbb9f9275d599d281cc54fbe2b46001621bf3c279f69cc353b631ee00c13f202a
aafc4e9619bfde25f5e416e9121ace222400c607ad61c5372b75dccfd91ae601c6
5a0c205fb57cdfb8e466f4ba9c79ad056fd7d6c813882cdb85759e91b8b9b37560
bd22efcd2dcde0c27d2e0061386906c44d622fdf851e73b6eaf8857de302ef22dd
ee32ea78f185d9d6d47a97819e5c5890fed90e86576356d8d5531f70d11d58b05b
f32b3edad4ef3b5c65089ceaa0e0264f17993205e57d07561c24c578b7b354c5b2
35750fbe371d1fb2b01ded811dc1c967d56b43d1de7c3f0985a21d7c63f51207ee
92754ab70d4f37534367c49bdffcb2b22f28d27b97b0fb4187
        hash_value =
fee1e43033ffd5d3e242f2059be8146c5ffdbe751335ffed1b71b815a1a15dd1
        minslabel =
        maxslabel =
        intlabel =
        accessauths =
        innateprivs =
        inheritprivs =
        authprivs =
        secflags =
```

1.5 Adding TE messages to syslog

Add the following entry to syslogd.conf, create the file and refresh syslogd

```
kern.debug      <te_out_file>
```

Example entries

```
Mar  5 03:55:01 aix-72 kern:info unix: Trusted Execution:
pid=11600228, euid=0, ruid=0: File not in TSD: /usr/bin/lsred
Mar  5 03:55:01 aix-72 kern:err|error unix: Trusted Execution:
pid=11600228, euid=0, ruid=0: Crypto hash verification failed:
/usr/bin/lsred
Mar  5 03:55:20 aix-72 kern:info unix: Trusted Execution:
pid=11600234, euid=0, ruid=0: File not in TSD: /usr/bin/lsred
```

1.6 Using the AIX Audit system

Auditing can be configured in either of two modes (bin or streams). For the purpose of this example, we will use streams.

The following events are defined in /etc/security/audit/events:

- TSDTPolicy
- TE_Untrusted
- TE_FileWrite

in /etc/security/config:
Create a new class

```
my_te = TSDTPolicy,TE_Untrusted,TE_FileWrite
```

Assign class to user

```
root = my_te
```

Examples of streams report

Base report - /usr/sbin/auditstream | auditpr

```
event           login    status      time                     command                        wpar name
--------------- -------- ----------- ------------------------ ------------------------------
-------------------------
S_PASSWD_READ   root     OK          Tue Jun 15 23:48:24 2021 sshd                           Global
TSDTPolicy      root     OK          Tue Jun 15 23:48:32 2021 trustchk                       Global
TSDTPolicy      root     OK          Tue Jun 15 23:48:32 2021 trustchk                       Global
TSDTPolicy      root     OK          Tue Jun 15 23:48:34 2021 trustchk                       Global
```

Base report with details - /usr/sbin/auditstream | auditpr -v

```
event           login    status      time                     command                        wpar name
```

```
-------------- -------- ---------- ----------------------- ------------------------------
------------------------
TSDTPolicy      root     OK          Tue Jun 15 23:53:01 2021 trustchk                        Global
        TE set
TSDTPolicy      root     OK          Tue Jun 15 23:53:01 2021 trustchk                        Global
        TE set
TSDTPolicy      root     OK          Tue Jun 15 23:53:03 2021 trustchk                        Global
        TE policy reset
```

Using /usr/sbin/auditstream | auditpr -h elrRtc -w

```
event           login    real     status      time                     command
--------------- -------- -------- ----------- ------------------------ ------------------------------
TSDTPolicy      root     root     OK          Tue Jun 15 23:14:41 2021 trustchk                          TE set
TSDTPolicy      root     root     OK          Tue Jun 15 23:14:41 2021 trustchk                          TE set
TSDTPolicy      root     root     OK          Tue Jun 15 23:14:43 2021 trustchk                          TE policy reset
TSDTPolicy      red      root     OK          Tue Jun 15 23:15:25 2021 trustchk                          TE set
TSDTPolicy      red      root     OK          Tue Jun 15 23:15:25 2021 trustchk                          TE set
TSDTPolicy      red      root     OK          Tue Jun 15 23:15:27 2021 trustchk                          TE policy reset
```

1.7 Steps to forward auditing to syslog daemon

Simple steps to redirect AIX Audit stream to syslog, which can either write to a local file, forward to a central syslog server, or both.
Steps

1. Configure the /etc/syslog.conf file:
   You will need to configure the facilty, priority and destination. In this example we will use the "local6" facility with priority of "notice" and send to a local file.
   For example:
   ```
   local6.notice /data/red/syslog_te.out rotate size 2m files 4 time 1w
   ```
   Which will sent the information to file /data/red/syslog_te.out, rotating every 2m or 1w (which ever comes first) across 4 files.
   If you wanted to send to central syslog server, you can use:
   ```
   local6.notice @my_syslog_server
   ```
   You will now need to create the file and refresh the syslogd daemon

2. Configure /etc/security/audit/config:
   As above confirm that streams mode is turned on
   Check the location of your streamcmds file (by default /etc/security/audit/streamcmds)
3. Modify the streamcmds file as:
   ```
   /usr/sbin/auditstream | /usr/sbin/auditselect -m -e "command !=
   logger && command != auditstream && command != auditpr && command !=
   auditselect"|auditpr -t0 -h eclrRdt -w | /usr/bin/logger -p
   local6.notice -r &
   ```
   Which will:
   - Read events from the audit subsystem and pass them to the auditselect command
   - The auditselect command will filter out commands generated by the streams pipeline itself
   - Run the auditpr command as seen in the example in section above, to format it they way we want
   - Send it to syslogd subsystem on the local6 facilty with an severity of notice
   - Uses -r flag to retry any messages dropped by syslog daemon until they are accepted.
   - And (&!) keep running in the background

1.8 Enabling TSD Protection

To enable TSD protection, run:

```
# trustchk -p tsd_lock=on
# trustchk -p te=on
```

The TSD is immediately protected against any kind of modification then. Neither trustchk Nor a manual edit of the file is possible:

```
# trustchk -d /usr/bin/ps
Error writing to database file
```

and

```
# echo >> /etc/security/tsd/tsd.dat
Operation not permitted.
```

To enable the TSD for write access again, you either need to switch off TE Completely or set tsd_lock to off. Either way, you need to reboot in order to have this change become active immediately:

```
# trustchk -p te=off
Policy in use. Changes applicable on next boot only
```

It is generally a good idea to save the TSD in a secure immutable place, so that regular checks against the system TSD can be run.

1.9 Trusting shell scripts

When blocking any untrusted shell scripts by using the CHKSCRIPT policy, make sure all scripts needed by your services are included in the TSD. For example, if you are using OpenSSH, make sure the Ssshd and Ksshd start and stop Scripts in /etc/rc.d/rc2.d are in the TSD. Otherwise, sshd will not get started upon reboot and not be shut down on a system shutdown:

```
# trustchk -p stop_untrustd=on
# trustchk -p chkscript=on
```

When trying to start a script with chkscript=on and that script is not included in the TSD, its execution will be denied, regardless of its permissions, even when root is invoking it:

```
# ./my_test
```

```
ksh: ./my_test: 0403-006 Execute permission denied.

# ls -l my_test
-rwx------- root system 17 May 10 11:51 my_test
```

1.10 Trusted Execution Path

The Trusted Execution Path (TEP) defines a list of directories that contain the trusted executables. Once TEP verification is enabled, the system loader allows only binaries in the specified paths to execute.

For example:

```
# trustchk -p tep
TEP=OFF
TEP=/usr/bin:/usr/sbin

# trustchk -p
tep=/usr/bin:/usr/sbin:/etc:/bin:/sbin:/usr/lib/instl:/usr/ccs/bin

# trustchk -p tep
TEP=OFF
TEP=/usr/bin:/usr/sbin:/etc:/bin:/sbin:/usr/lib/instl:/usr/ccs/bin

# trustchk -p tep=on

# trustchk -p tep
TEP=ON
TEP=/usr/bin:/usr/sbin:/etc:/bin:/sbin:/usr/lib/instl:/usr/ccs/bin
```

1.11 Trusted Library Path

The Trusted Library Path (TLP) has the same functionality as TEP with the only difference that it is used to define the directories that contain trusted libraries of the system. Once TLP is enabled, the system loader will allow only the libraries from this path to be linked to the binaries.

The trustchk command can be used to enable/disable the TEP/TLP as well as to set the colon-separated path list for oth using TEP and TLP command-line attributes of trustchk:

```
# trustchk -p tlp
TLP=OFF
TLP=/usr/lib:/usr/ccs/lib:/lib:/var/lib
```

TLP uses a flag to control its operations: FSF_TLIB. If the file has the FSF_TLIB flag set in its TSD stanza, then the process resulting from it will be set as a TLIB process.
Processes marked as TLIB processes can link only to *.so libraries that also have the TLIB flag set.

Note:	Note: Be careful when changing either TEP or TLP. We do not recommend removing paths from their Default settings, which are currently set to:
	TEP=/usr/bin:/usr/sbin:/etc:/bin:/sbin:/sbin/helpers/jfs2:/usr/lib/instl:/usr/ccs/bin
	TLP=/usr/lib:/usr/ccs/lib:/lib:/var/lib
	
	Doing so will most probably result in a system that will not reboot and function properly since it cannot access necessary files and data any longer.

1.12 Short examples

Perform a system check comparison with the TSD and report errors:

```
# trustchk -n ALL
```

Note:	-n Specifies the auditing mode, and indicates that the errors are to be reported. Any discrepancy between the attributes in the TSD and the actual file parameters are printed to the stderr. error file. To check all of the entries in the TSD, use the *ALL* parameter. To scan the entire system or directories for TROJAN detection, use with *tree* parameter.

Delete the entry for /usr/bin/ls in the TSD:

```
# trustchk -d /usr/bin/ls
```

Enable policy for checking executables listed in TSD on every load:

```
# trustchk -p CHKEXEC=ON
```

# Turn on Run-time TSD checking:

```
# trustchk -p TE=ON
```

Check the current run-time policy in effect:

```
# trustchk -p
```

Checking the hash

```
trustchk -q /usr/bin/mv | grep hash
```

```
        hash_value =
fee1e43033ffd5d3e242f2059be8146c5ffdbe751335ffed1b71b815a1a15dd1
# openssl dgst -sha256 /usr/bin/mv
SHA256(/usr/bin/mv)=
fee1e43033ffd5d3e242f2059be8146c5ffdbe751335ffed1b71b815a1a15dd1
```

Disabling a file if compromised

```
Test: Change a file for testing (/usr/bin/ls)
trustchk -y /usr/bin/ls
trustchk: Verification of attributes failed: owner
trustchk: Verification of attributes failed: group
trustchk: Verification of attributes failed: modetrustchk:
Verification of attributes failed: size
trustchk: Verification of attributes failed: hash
trustchk: Verification of attributes failed: signature
trustchk: Verification of stanza failed:
Check file:
# ls -l /usr/bin/ls
---------T    1 bin      bin       93 May 28 16:07 /usr/bin/ls
```

1.13 Files

The following files are in /etc/security/tsd:

tsd.dat and .tsd.bk	TSD file and backup
ldap/tspolicies.dat	Policy control with LDAP
lib/lib.tsd.dat and .lib.tsd.bk	Library TSD and backup
tepolicies.dat	Current TSD settings

Maintenance notes
Note:

1. Installing updates (non AIX) naturally changes the files as hashes in the TE database will be modified.
2. AIX updates come with new signatures, so need to update hashes for own files.

1.14 Third party tools - XYMON

https://xymon.sourceforge.io/

```
MSGS STATUS COLUMN SETTINGS
LOG logfilename pattern [COLOR=color] [IGNORE=excludepattern] [OPTIONAL]
```

The Xymon client extracts interesting lines from one or more logfiles - see the client-local.cfg(5) man-page for information about how to configure which logs a client should look at.

The LOG setting determine how these extracts of log entries are processed, and what warnings or alerts trigger as a result.

"logfilename" is the name of the logfile. Only logentries from this filename will be matched against this rule. Note that "logfilename" can be a regular expression (if prefixed with a '%' character).

"pattern" is a string or regular expression. If the logfile data matches "pattern", it will trigger the "msgs" column to change color. If no "color" parameter is present, the default is to go "red" when the pattern is matched. To match against a regular expression, "pattern" must begin with a '%' sign - e.g "%WARNING|NOTICE" will match any lines containing either of these two words. Note that Xymon defaults to case-insensitive pattern matching; if that is not what you want, put "(?-i)" between the "%" and the regular expression to turn this off. E.g. "%(?-i)WARNING" will match the word WARNING only when it is upper-case.

"excludepattern" is a string or regular expression that can be used to filter out any unwanted strings that happen to match "pattern".

The OPTIONAL keyword causes the check to be skipped if the logfile does not exist.

Example: Trigger a red alert when the string "ERROR" appears in the "/var/adm/syslog" file:

```
      LOG /var/adm/syslog ERROR
```

Example: Trigger a yellow warning on all occurrences of the word "WARNING" or "NOTICE" in the "daemon.log" file, except those from the "lpr" system:

```
      LOG /var/log/daemon.log %WARNING|NOTICE COLOR=yellow IGNORE=lpr
```

Defaults:

```
      color="red", no "excludepattern".
```

Note: That no logfiles are checked by default. Any log data reported by a client will just show up on the "msgs" column with status OK (green).

2 Trusted Signature Database

Similar to that of Trusted Computing Base (TCB) there exists a database which is used to store critical security parameters of trusted files present on the system. This database, called Trusted Signature Database (TSD), resides in the /etc/security/tsd/tsd.dat.

A trusted file is a file that is critical from the security perspective of the system, and if compromised, can jeopardise the security of the entire system. Typically the files that match this description are the following:

- Kernel (operating system)
- All setuid root programs
- All setgid root programs
- Any program that is exclusively run by the root user or by a member of the system group
- Any program that must be run by the administrator while on the trusted communication path (for example, the ls command)
- The configuration files that control system operation
- Any program that is run with the privilege or access rights to alter the kernel or the system configuration files

Every trusted file should ideally have an associated stanza or a file definition stored in the Trusted Signature Database (TSD). A file can be marked as trusted by adding its definition in the TSD using the trustchk command. The trustchk command can be used to add, delete, or list entries from the TSD.

2.1 Location of the TSD

Local
The Trusted Signature Database is a database that is used to store critical security parameters of trusted files present on the system. This database resides in the /etc/security/tsd/tsd.dat directory.
Remote
Centralised Trusted Signature Database (TSD) policies and Trusted Execution (TE) policies can be implemented in your system environment by storing them in LDAP.

2.2 Details of the TSD

The Trusted Signature Database is a database that is used to store critical security parameters of trusted files present on the system. This database resides in the /etc/security/tsd/tsd.dat directory.

Every trusted file must ideally have an associated stanza or a file definition stored in the Trusted Signature Database (TSD). Every trusted file is associated with a unique cryptographic hash and a digital signature. The cryptographic hash of the default set of trusted files is generated by using the

SHA-256 algorithm and the digital signature that is generated by using RSA by the AIX® build environment and packaged as part of AIX installation filesets. These hash values and the signatures are shipped as part of respective AIX installation images and stored in the Trusted Signature Database (/etc/security/tsd/tsd.dat) on the destination machine, in the sample stanza format that follows:

```
/usr/bin/ps:
           owner          = bin
           group          = system
           mode           = 555
           type           = FILE
           hardlinks      = /usr/sbin/ps
           symlinks       =
           size           = 1024
           cert_tag       = bbe21b795c550ab243
           signature      =
f7167eb9ba3b63478793c635fc991c7e9663365b2c238411d24c2a8a
           hash_value     = c550ab2436792256b4846a8d0dc448fc45
           minslabel      = SLSL
           maxslabel      = SLSL
           intlabel       = SHTL
           accessauths    = aix.mls.pdir, aix.mls.config
           innateprivs    = PV_LEF
           proxyprivs     = PV_DAC
           authprivs      =
aix.security.cmds:PV_DAC,aix.ras.audit:PV_AU_ADMIN
           secflags       = FSF_EPS
           t_accessauths  =
           t_innateprivs  =
           t_proxyprivs   =
           t_authprivs    =
           t_secflags     =
```

owner
: Owner of the file. This value is computed by the trustchk command when the file is being added to TSD.

group
: Group of the file. This value is computed by the trustchk command.

mode
: Comma-separated list of values. The permissible values are SUID (SUID set bit), SGID (SGID set bit), SVTX (SVTX set bit), and TCB (Trusted Computing Base). The file permissions must be the last value and can be specified as an octal value. For example, for a file that is set with uid and has permission bits as rwxr-xr-x, the value for mode is SUID, 755. The value is computed by the trustchk command.

type
: Type of the file. This value is computed by the trustchk command. The possible values are FILE, DIRECTORY, MPX_DEV, CHAR_DEV, BLK_DEV, and FIFO.

hardlinks
: List of hardlinks to the file. This value cannot be computed by the trustchk command. It must be supplied by the user when adding a file to the database.

symlinks
: List of symbolic links to the file. This value cannot be computed by the trustchk command. It must be supplied by the user when adding a file to the database.

size
: Defines size of the file. The VOLATILE value means that the file gets changed frequently.

cert_tag
: This field maps the digital signature of the file with the associated certificate that can be used to verify the signature of the file. This field stores the certificate ID and is computed by the trustchk command at the time of addition of the file to the TSD. The certificates are stored in /etc/security/certificates directory.

signature
: Digital signature of the file. The VOLATILE value means that the file gets changed frequently. This field is computed by the trustchk command.

hash_value
: Cryptographic hash of the file. The VOLATILE value means that the file gets changed frequently. This field is computed by the trustchk command.

minslabel
: Defines the minimum sensitivity label for the object.

maxslabel
: Defines the maximum sensitivity label for the object (valid on Trusted AIX system). This attribute is not applicable to regular files and fifo.

intlabel
: Defines the integrity label for the object (valid on Trusted AIX system).

accessauths
: Defines the access authorization on the object (valid on Trusted AIX system).

innateprivs
: Defines the innate privileges for the file.

proxyprivs
: Defines the proxy privileges for the file.

authprivs
: Defines the privileges that are assigned to the user after given authorizations.

secflags
: Defines the file security flags associated with the object.

t_accessauth
: Defines the additional Trusted AIX with Multi-Level Security (MLS) specific access authorizations (valid on Trusted AIX system).

t_innateprivs
: Defines the additional Trusted AIX with MLS-specific innate privileges for the file (valid on Trusted AIX system).

t_proxyprivs
: Defines the additional Trusted AIX with MLS-specific proxy privileges for the file (valid on Trusted AIX system).

t_authprivs
: Defines the additional Trusted AIX with MLS-specific privileges that are assigned to the user after given authorizations (valid on Trusted AIX system).

t_secflags
: Defines the additional Trusted AIX with MLS-specific file security flags associated with the object (valid on Trusted AIX system).

When you add a new entry to TSD, if a trusted file has some symbolic or hard links pointing to it, then these links can be added to the TSD by using symlinks and hardlinks attributes at the command line, along with the trustchk command. If the file being added is expected to change frequently, then use VOLATILE keyword at the command line. Then the trustchk command would not calculate the hash_value and signature fields when it generates the file definition for addition into the TSD. During integrity verification of this file, the hash_value and signature fields are ignored.

During addition of regular file definitions to the TSD, it is necessary to provide a private key (ASN.1/DER format). Use the -s flag and digital certificate with the corresponding public key by using the -v flag. The private key is used to generate the signature of the file and then discarded. It is up to the user to store this key securely. The certificate is stored into a certificate store in the/etc/security/certificates file for the signatures to be verified whenever you request integrity verification. Since signature calculation is not possible for non-regular files like directory and device files, it is not mandatory to supply the private key and certificate while adding such files to TSD.

You can also supply the pre-computed file definition through a file by using the -f option to be added to the TSD. In this case the trustchk command does not compute any of the values and stores the definitions into TSD without any verification. The user is responsible for sanity of the file definitions in this case. Using the -f filename flag, specifies that file definitions are to be read from the file specified with the filename parameter. The file (or stanza) name must end with a colon. There must be a blank line between each file name entry in the external file.

Supporting library verification
To support the library verification, the tsd.dat file is added in the /etc/security/tsd/lib/directory. The name of the database is /etc/security/tsd/lib/lib.tsd.dat. This database is specifically for libraries that include the stanzas for the .o files of a corresponding trusted library. The stanza for every object file (*.o) of a library is in the format as specified in the following example.

For library libc.a if the strcmp.o file is one of the.o file type, then the stanza for strcmp.o file in /etc/security/tsd/lib/lib.tsd.dat is similar to the following example:
/

```
usr/lib/libc.a/strcmp.o:
                  Type = OBJ
                  Size = 2345
```

```
                    Hash value
                    Signature =
                    Cert_tag =
```

This database has the entries corresponding to type, size hash, cert tag, and signature of the .o file. The hash of the library is updated in the /etc/security/tsd/tsd.dat file for the corresponding stanza. These attribute values are dynamically generated during the build, and the values are moved into the /etc/security/tsd/lib/lib.tsd.dat database during installation.

In the /etc/security/tsd/tsd.dat file, the stanzas for the libraries are modified to reflect the type attribute as LIB and the size and signature attributes are empty. Currently the values for the dynamic attributes size, hash, signature are maintained as a VOLATILE value. Therefore, the library verification is skipped during system boot. Beginning with the release of AIX 6.1.0, the size, hash, and signature of the trusted library stanzas are computed with the .o files of a library. During installation, the tsd.dat database is populated to reflect the computed values and the corresponding .o file stanza for a trusted library is stored in the /etc/security/tsd/lib/lib.tsd.dat database.

2.3 Remote TE data base access:

Centralised Trusted Signature Database (TSD) policies and Trusted Execution (TE) policies can be implemented in your system environment by storing them in LDAP.
The database that controls the TSD policies and TE policies are stored independently of each AIX system. The centralised TSD policies and TE policies are stored in LDAP so that they can be centrally managed. Using centralised TSD policies and TE policies allow you to verify that the policies in LDAP are the master copy, and that the policies can update the clients whenever the client is reinstalled, updated, or security is breached. Centralised TE policies allow one location to enforce the TE policies without needing to update each client separately. Centralised TSD policies are much easier to manage than TSD polices that are not centralised.

AIX Utilities can be used to export local TSD policies and TE policies data to LDAP, configure clients to use TSD policies and TE policies data in LDAP, control the lookup of TSD policies and TE policies data, and manage the LDAP data from a client system. The following sections provide more information about these features.

2.4 Exporting TSD policies and TE policies data to LDAP:

To use LDAP as a centralised repository for TSD policies and TE policies, the LDAP server must be populated with the policy data.

The LDAP server must have the TSD policies and the TE policies schema for LDAP installed, before LDAP clients can use the server for policy data. The TSD policies and the TE policies schema for LDAP is available on an AIX system in the /etc/security/ldap/sec.ldif file. The schema for the LDAP server must be updated with this file by using the ldapmodify command.

To identify a version the TE databases on the LDAP server and make LDAP clients aware of the particular version, you must set the databasename attribute in the /etc/nscontrol.conf file. The databasename attribute takes any name as the value, and it is used by the tetoldif command while generating the ldif format.

Use the tetoldif command to read the data in the local TSD policies and TE policies files, and output the policies in a format that can be used for LDAP. The output generated by the tetoldif command can be saved to a file in ldif format, and then used to populate the LDAP server with the data with the ldapadd command. The following databases on the local system are used by the tetoldif command to generate the TSD policies and TE policies data for LDAP:

- /etc/security/tsd/tsd.dat
- /etc/security/tsd/tepolicies.dat

2.5 LDAP client configuration for TSD policies and TE policies:

A system must be configured as an LDAP client to use TSD policies and TE policies data stored in LDAP.

Use the AIX /usr/sbin/mksecldap command to configure a system as an LDAP client. The mksecldap command dynamically searches the specified LDAP server to determine the location of the TSD policies and TE policies data, and saves the results to the /etc/security/ldap/ldap.cfg file.

After successfully configuring the system as an LDAP client with the mksecldap command, the system must be further configured to enable LDAP as a lookup domain for TSD policies and TE policies data by configuring the secorder of the /etc/nscontrol.conf file.

Once the system has been configured as a LDAP client and as a lookup domain for TSD policies and TE policies data, the /usr/sbin/secldapclntd client daemon retrieves the TSD policies and TE policies data from the LDAP server whenever any trustchk commands are performed on the LDAP client.

2.6 Enabling LDAP with the trustchk command:

All of the TSD policies and TE policies database management commands are enabled to use the LDAP TSD policies and TE policies database.

Use the trustchk command with the –R flag, to perform the initial setup of LDAP database. The initial setup involves the addition of TSD policies, TE policies, base DNs, and the creation of the local database /etc/security/tsd/ldap/tsd.dat file and /etc/security/tsd/ldap/tepolicies.dat file. If the trustchk command is run with the –R flag using the LDAP option, the operations are based on the

LDAP server data. If the trustchk command is run with the –R flag using the files option, the operations are based on the local database data. The default for the –R flag is to use the files option.

2.7 Auditing the integrity of Trusted Signature Database:

The trustchk command can be used to audit the integrity state of the file definitions in the Trusted Signature Database (TSD) against the actual files.

If the trustchk command identifies an anomaly, then it can be made to automatically correct it or prompt the user before attempting correction. If anomalies like size, signature, cert_tag or hash_value mismatch, the correction is not possible. In such cases, the trustchk command would make the file inaccessible, thereby rendering it useless and containing any damage.

2.8 Security policies configuration:

The Trusted Execution (TE) feature provides you with a run-time file integrity verification mechanism.

Using this mechanism, the system can be configured to check the integrity of the trusted files before every request to access those file, effectively allowing only the trusted files that pass the integrity check to be accessed on the system. When a file is marked as trusted (by adding its definition to Trusted Signature Database), the TE feature can be made to monitor its integrity on every access. TE can continuously monitor the system and is capable of detecting tampering of any trusted file (by a malicious user or application) present on the system at run-time (for example, at load time). If the file is found to be tampered, TE can take corrective actions based on pre-configured policies, such as disallow execution, access to the file, or logging error. If a file being opened or executed, and has an entry in the Trusted Signature Database (TSD), the TE performs as follows:

- Before loading the binary, the component responsible for loading the file (system loader) invokes the Trusted Execution subsystem, and calculates the hash value using the SHA-256 algorithm (configurable).
- This run-time calculated hash value is matched with the one stored in the TSD.
- If the values match, the file opening or execution is permitted.
- If the values do not match, either the binary is tampered, or somehow compromised. It is up to the user to decide the action to be taken. The TE mechanism provides options for users to configure their own policies for the actions to be taken if the hash values do not match.
- Based on these configured policies, a relevant action is taken.

The following policies can be configured:

CHKEXEC

Check hash value of only the trusted executables before loading them in memory for execution.

CHKSHLIBS

Check the hash value of only the trusted shared libraries before loading them in memory for execution.

CHKSCRIPTS

Check the hash value of only the trusted shell scripts before loading them in memory.

CHKKERNEXT

Check the hash value of only the kernel extension before loading it in memory.

STOP_UNTRUSTD

Stop loading of files that are not trusted. Only files belonging to TSD are loaded. This policy only works in combination with any of the CHK* policies mentioned above. For example, if CHKEXEC=ON and STOP_UNTRUSTD=ON, then any executable binary that does not belong to TSD is blocked from execution.

STOP_ON_CHKFAIL

Stop loading of trusted files that fail hash value check. This policy also works in combination with CHK* policies. For example, if CHKSHLIBS=ON and STOP_ON_CHKFAIL=ON, then any shared library not belonging to the TSD is blocked from being loaded into memory for use.

TSD_LOCK

Lock TSD so it is not available for editing.

TSD_FILES_LOCK

Lock trusted files. This does not allow opening of trusted files in write mode.

TE

Enable/Disable Trusted Execution functionality. Only when this is enabled, the above mentioned policies are in effect.

The following table gives the interaction between different CHK* policies and STOP* policies when
enabled:

Policy	STOP_UNTRUSTD	STOP_ON_CHKFAIL
CHKEXEC	Stop loading of executables that do not belong to TSD.	Stop loading of executables whose hash values do not match the TSD values.
CHKSHLIBS	Stop loading of shared libraries that do not belong to TSD.	Stop loading of shared libraries whose hash values do not match the TSD values.
CHKSCRIPTS	Stop loading of shell scripts that do not belong to TSD.	Stop loading of shell scripts whose hash values do not match the TSD values.
CHKKERNEXT	Stop loading of kernel extensions that do not belong to TSD.	Stop loading of kernel extensions whose hash values do not match the TSD values.

Note: A policy can be enabled or disabled at any time until the TE is turned on to bring the policies into effect. Once a policy is in effect, disabling that policy becomes effective only on next boot cycle. All the information messages are logged into syslog.

2.8.1 Trusted Execution Path and Trusted Library Path:

Trusted Execution Path (TEP) defines a list of directories that contain the trusted executables. Once TEP verification is enabled, the system loader allows only binaries in the specified paths to execute. Trusted Library Path (TLP) has the same functionality, except that it is used to define the directories that contain trusted libraries of the system.

Once TLP is enabled, the system loader allows only the libraries from this path to be linked to the binaries. The trustchk command can be used to enable or disable the TEP or TLP, as well as set the colon separated path list for both, using TEP and TLP command line attributes of the trustchk command.

2.8.2 Trusted Shell and Secure Attention Key:

Trusted Shell and Secure Attention Key (SAK) perform similarly to the Trusted Computing Base (TCB), except that if Trusted Execution is enabled on the system instead of TCB, the Trusted Shell executes files belonging only to the Trusted Signature Database.

For more information about TCB and SAK, see Trusted Computing Base, Using the Secure Attention Key, and Configuring the Secure Attention Key.

2.8.3 Trusted Execution (TE) policies Database:

The Trusted Execution (TE) policies are stored in the /etc/security/tsd/tepolicies.dat file. The path for the TE policies are listed with the TLP directories and TEP directories.