`libsklog`
User Guide

Paolo Smiraglia
paolo.smiraglia@polito.it

Last Update: July 28, 2011

1 Introduction

`libsklog` is a library for C language which allows to perform secure remote logging following the schema defined by B.Schneier and J.Kelsey in *Secure Audit Logs to Support Computer Forensics*. This document illustrates how to install `libsklog` under Linux operating systems and how to configure the system enviroment to use it. To get more information, to notify a bug, or generally to contact me write at `paolo.smiraglia@polito.it`.

1.1 The Schneier and Kelsey's model

In Schneier-Kelsey's model, three entities are defined. They consider a trusted machine $\mathcal{T}$ (e.g. a server in a secure location), an untrusted machine $\mathcal{U}$ (potentially the victim of an attack) on which the log entries are to be temporarily kept and a moderately-trusted external verifier $\mathcal{V}$ (e.g. an external auditor). Moreover, a log entry creation scheme is well-defined (Figure 1). With this type of scheme, the immediate identification of log tampering (e.g. deletion of a log entry) becomes possible because the log entries are linked in an *hash-chain* by the element Y_j. The integrity of logs is ensured by including an HMAC field (Z_j) within the log entries and the confidentiality of the logged data ($E_{K_j}(D_j)$) is guaranteed thanks to the usage of a symmetric cryptography mechanism. In SK when $\mathcal{V}$ wants to read the logs she requests and obtains the log entries from $\mathcal{U}$ and successively sends them to $\mathcal{T}$ for validation and decryption.

2 Installation

`libsklog` allows to write applications which act as the actors $\mathcal{U}$, $\mathcal{T}$ and $\mathcal{V}$ defined by Schneier and Kelsey. In this section is described the `libsklog` installation procedure and how to configure the environment for each component of Scneier-Kelsey's schema. All steps are described assuming that `libsklog` installation prefix is `/usr/local` and that a SQLite database is used to store data locally.

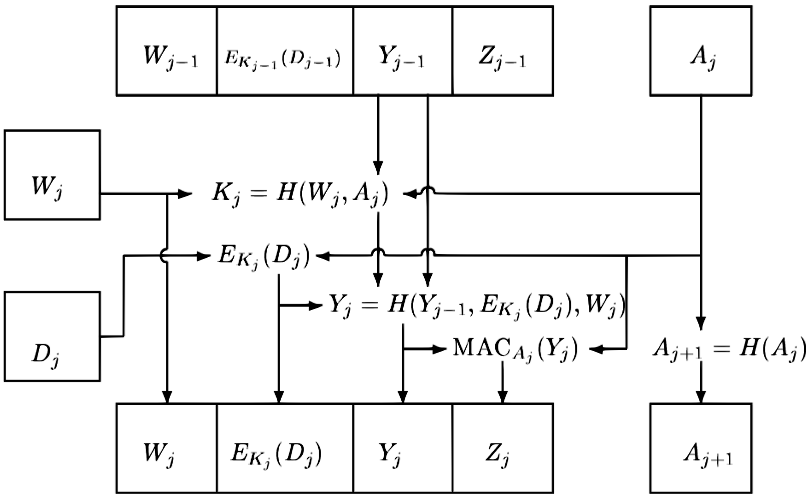

Figure 1: Schneier and Kelsey's log entry creation scheme.

2.1 Get Sources and Install Library

Before proceeding with the installation, the dependencies listed below need to be resolved:

- `Libtool`
- `Autoconf`
- `OpenSSL` $\geq$ `0.9.8`
- `SQLite 3.x`
- `libuuid-dev`
- `libconfuse`

Installing `libsklog` is rather painless through the use of the GNU autoconf package. Simply get the sources from the Git[1] repository, generate the `configure` script and finally run it. In most cases, `configure` will automatically determine everything it needs to know in order to compile. However, there are a few options to "configure" to help it out, or change the default behavior:

```
--enable-trace      Enable high verbosity mode for libsklog library
                    [default=no]

--enable-notify     Enable notify messages for libsklog library
                    [default=no]

--enable-debug      Enable debug support [default=no]
```

[1] Git Usage

The commands listed below, show you how to get the `libsklog` sources, how to generate the `configure` script and finally how to install the library.

```
mkdir ˜/temp
cd ˜/temp
git clone https://github.com/psmiraglia/Libsklog.git libsklog

cd libsklog
mkdir m4
autoreconf --install --force --verbose

./configure --prefix=/usr/local [other options]
make
make check
make install (as root)
```

At this point the library should be correctly installed. Below is reported the installation result. Now you can proceed with the configuration of the components.

```
/usr/local
/usr/local/bin
/usr/local/bin/tnode
/usr/local/bin/unode
/usr/local/etc
/usr/local/etc/libsklog
/usr/local/etc/libsklog/certs
/usr/local/etc/libsklog/certs/ca
/usr/local/etc/libsklog/certs/ca/ca_cert.pem
/usr/local/etc/libsklog/certs/private
/usr/local/etc/libsklog/certs/private/ca_key.pem
/usr/local/etc/libsklog/certs/private/u1_key.pem
/usr/local/etc/libsklog/certs/u1_cert.pem
/usr/local/etc/libsklog/libsklog-t.conf.example
/usr/local/etc/libsklog/libsklog-u.conf.example
/usr/local/etc/libsklog/sql
/usr/local/etc/libsklog/sql/t_database.sql
/usr/local/etc/libsklog/sql/u_database.sql
/usr/local/include
/usr/local/include/libsklog
/usr/local/include/libsklog/sklog_commons.h
/usr/local/include/libsklog/sklog_err.h
/usr/local/include/libsklog/sklog_internal.h
/usr/local/include/libsklog/sklog_t.h
/usr/local/include/libsklog/sklog_u.h
/usr/local/include/libsklog/sklog_utils.h
/usr/local/include/libsklog/sklog_v.h
/usr/local/lib
/usr/local/lib/libsklog.a
/usr/local/lib/libsklog.la
/usr/local/lib/libsklog.so -> libsklog.so.0.0.0
/usr/local/lib/libsklog.so.0 -> libsklog.so.0.0.0
/usr/local/lib/libsklog.so.0.0.0
```

```
/usr/local/var
/usr/local/var/libsklog
/usr/local/var/libsklog/db
/usr/local/var/libsklog/db/t.db
/usr/local/var/libsklog/db/u.db
```

2.2 Setup $\mathcal{U}$ Component

2.2.1 Configuration File

To configure a $\mathcal{U}$ component it's necessary to create a file called `libsklog-u.conf` in `/usr/local/etc/libsklog` which will contains all required settings. Below all settable parameters:

`t_cert`	Specifies the path where the certificate of $\mathcal{T}$ is installed. $\mathcal{T}$ acts also as certification authority.
`t_address`	Specifies the IP address of $\mathcal{T}$.
`t_port`	Specifies the port on where $\mathcal{T}$ is listening.
`u_cert`	Specifies the path where the certificate of $\mathcal{U}$, issued by $\mathcal{T}$, is installed.
`u_id`	Specifies the identifier (common name) of $\mathcal{U}$.
`u_privkey`	Specifies the path where the private key of $\mathcal{U}$ is installed.
`u_timeout`	Sets the timeout for the logfile initialization procedure.
`logfile_size`	Sets the number of log entries which can be collected into the logfile.

The file `libsklog-u.conf.example` is a template of a configuration file for $\mathcal{U}$ component. You can use it as staring point for the definition of a new file:

```
cd /usr/local/etc/libsklog
cp libsklog-u.conf.example libsklog-u.conf
vim libsklog-u.conf
(edit your file)
```

2.2.2 Database Initialization

```
cd /usr/local/var/libsklog/db
sqlite3 u.db < /usr/local/etc/libsklog/sql/u_database.sql
```

2.3 Setup $\mathcal{T}$ Component

2.3.1 Configuration File

To configure a $\mathcal{T}$ component it's necessary to create a file called `libsklog-t.conf` in `/usr/local/etc/libsklog` which will contains all required settings. Below all settable parameters:

`t_cert`	Specifies the path where the certificate of $\mathcal{T}$ is installed. $\mathcal{T}$ acts also as certification authority.
`t_privkey`	Specifies the path where the private key of $\mathcal{T}$ is installed.
`t_id`	Specifies the identifier (common name) of $\mathcal{T}$.
`t_address`	Specifies the IP address of $\mathcal{T}$.
`t_port`	Specifies the port on where $\mathcal{T}$ is listening.

The file `libsklog-t.conf.example` is a template of a configuration file for $\mathcal{T}$ component. You can use it as staring point for the definition of a new file:

```
cd /usr/local/etc/libsklog
cp libsklog-t.conf.example libsklog-t.conf
vim libsklog-t.conf
(edit your file)
```

2.3.2 Database Initialization

```
cd /usr/local/var/libsklog/db
sqlite3 t.db < /usr/local/etc/libsklog/sql/t_database.sql
```

2.4 Setup $\mathcal{V}$ Component

Not yet implemented. Do you want to help me?

3 Usage

3.1 $\mathcal{U}$ component

```
/*
** This is a really simple application
** which acts as U component
*/

#include <stdio.h>
#include <string.h>
#include <libsklog/sklog_u.h>

#define BUFLEN 1024

int main (void) {

    SKLOG_U_Ctx *ctx = 0;
    SKLOG_DATA_TYPE e_type = 0;
    char event[BUFLEN] = { 0 };

    ...

    ctx = SKLOG_U_NewCtx();

    if ( ctx == NULL ) {
        fprintf(stderr,"SKLOG_U_NewCtx() failure");
        exit(1);
    }

    /* something happens */
    SKLOG_U_LogEvent(ctx,e_type,event,strlen(event));

    ...

    /* something happens */
    SKLOG_U_LogEvent(ctx,e_type,event,strlen(event));

    ...

    /* something happens */
    SKLOG_U_LogEvent(ctx,e_type,event,strlen(event));

    ...

    SKLOG_U_FreeCtx(&ctx);

    return 0;
}
```

```
gcc -I/usr/local/include -L/usr/local/lib \
    u_app.c -o u_app -lsklog
```

3.2 $\mathcal{T}$ component

```c
/*
** This is a really simple application
** which acts as T component
*/

#include <stdio.h>
#include <libsklog/sklog_t.h>

#define BUFLEN 1024

int main (void) {

    SKLOG_T_Ctx *ctx = 0;

    ...

    ctx = SKLOG_T_NewCtx();

    if ( ctx == NULL ) {
        fprintf(stderr,"SKLOG_T_NewCtx() failure");
        exit(1);
    }

    if ( SKLOG_T_InitCtx(ctx) == SKLOG_FAILURE ) {
        fprintf(stderr,"SKLOG_T_InitCtx() failure");
        exit(1);
    }

    ...

    SKLOG_T_Run(ctx);

    ...

    SKLOG_T_FreeCtx(&ctx);

    return 0;
}
```

```
gcc -I/usr/local/include -L/usr/local/lib \
    t_app.c -o t_app -lsklog
```

4 TODO List

4.1 $\mathcal{U}$

- Maybe nothing...

4.2 $\mathcal{T}$

- Implement function which parse the configuration file
- Implement function which verify the M0 message
- Implement function which verify the integrity of collected data

4.3 $\mathcal{V}$

- All! Do you want to help me?