# Prelude 0.9 Handbook

24. Oktober 2007

## Inhaltsverzeichnis

# 1 Introduction

## 1.1 Introduction to Prelude

### What is Prelude ? Foreword

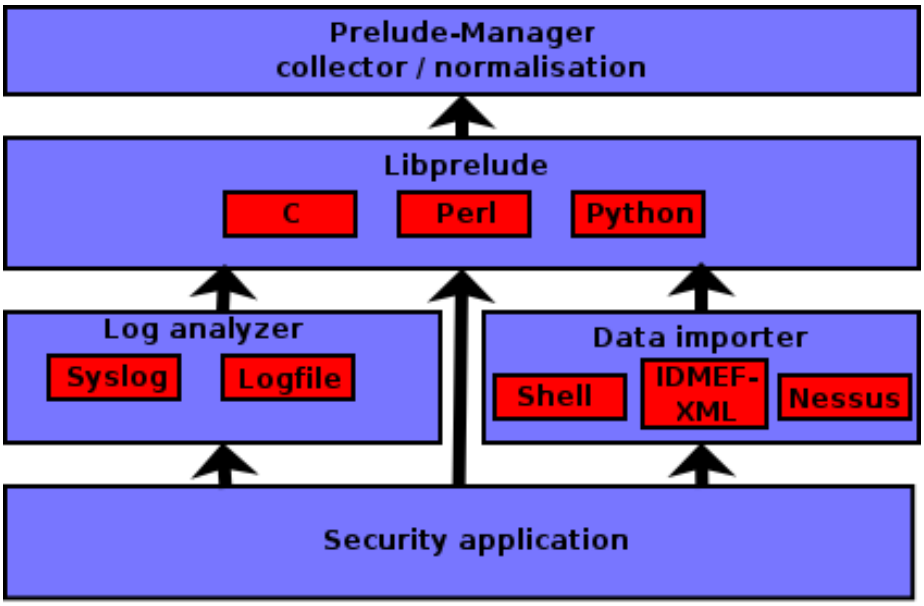
Prelude was born from the observation that more and more IDS systems each with their own specificity have been made available, but that no framework exists in order to unify information provided by these different systems in order to unify and centralize events.

We believe that relying on a single source of information in order to perform security analysis is not sufficient since different analysis methods have different avantages, and that unifying theses methods in a strong and powerful product is the only means to come to a stronger security analysis tool.

Prelude is a Hybrid IDS framework, that is, it is a product that enables all available security applications, be they open source or proprietary, to report to a centralized system. In order to achieve this task, Prelude relies on the IDMEF (Intrusion Detection Message Exchange Format) IETF standard, that enables different kinds of sensors to generate events using a unique language.

In this context, hybrid means that Host agent data is combined with network information to form a comprehensive view of the network. In the Prelude view, 'hybrid' means that you can combine events detected by any security applications (Network IDS, Log analyzer, Security application, Security shell script, or put simply, anything...provided that it reports security events).

Prelude provides a C, Python, and Perl framework so that you can convert existing security applications to use the Prelude system. It also provides some home-made sensors such as a log analyzer (PreludeLML). A Prelude sensor is a program which has the ability of using the Prelude framework.




**Prelude components**

- The Prelude library: Framework.

Libprelude is the library that provides the framework used to access the Prelude system. It handles secure communications with one or several prelude-manager collectors, and provides an API (Application Programming Interface) to create IDMEF (Intrusion Detection Message Exchange Format) based events.

It also provides important features like failover (saving to a local file for later retransmission, usage of a fallback route), in case one of the prelude-manager servers goes down.

Moreover, it gives you the ability to create sensors that read events received by one or a set of prelude-managers. You could for example write an interactive notification system using this feature.

- Prelude-Manager: Collects and normalize events.

The PreludeManager is a high-availability server which collects and normalizes information from distributed sensors and stores them into a database (or any kind of user-provided media). It also provides the ability to relay received events to one or several other prelude-manager servers. It also provides you the ability to filter received events so that you can provide specific actions for specific events.

- Prelude-LML: Log analyzer, Syslog events collector.

PreludeLML is a signature-based log analyzer monitoring your logfile and received syslog messages for suspicious activity. It handle events generated by a large set of components, including but not limited to: APC Emu, BigIP, Cisco PIX, Clamav, Dell-OM, Grsecurity, Honeyd, ipchains, Netfilter, ipfw, Nokia ipso, Apache ModSecurity?, Ms-SQL, Nagios, Norton Antivirus Corporate Edition, NTsyslog, Pam, Portsentry, Postfix, Proftpd, ssh, etc.

Prelude-LML was written in order to easily integrate third party products, most particularly products that can't be modified directly to use the Prelude library.

- The PreludeDB library : Easy access to the Prelude database.

The PreludeDB Library provides an abstraction layer based upon the type and the format of the database used to store IDMEF alerts.

It allows developers to use the Prelude IDMEF database easily and efficiently without worrying about SQL, and to access the database independently of the type/format of the database.

- Prewikka: The Prelude-IDS console.

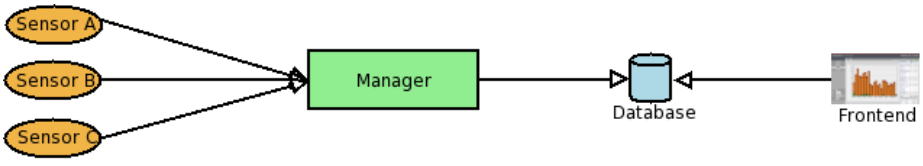
Prewikka is a professional application providing advanced features like contextual filtering, aggregation, etc. Prewikka is a large step forward compared to Piwi.

## 1.2 Prelude architecture

### Simplified architecture

Prelude is divided in several components. Sensors are responsible for intrusion detection, and report events in a centralized fashion using a TLS connection to a 'prelude-manager' server. The prelude-manager server can then process theses events and deliver them to an user specified media (mysql database, postgresql database, XML file, any format provided there is a report plugin for it).

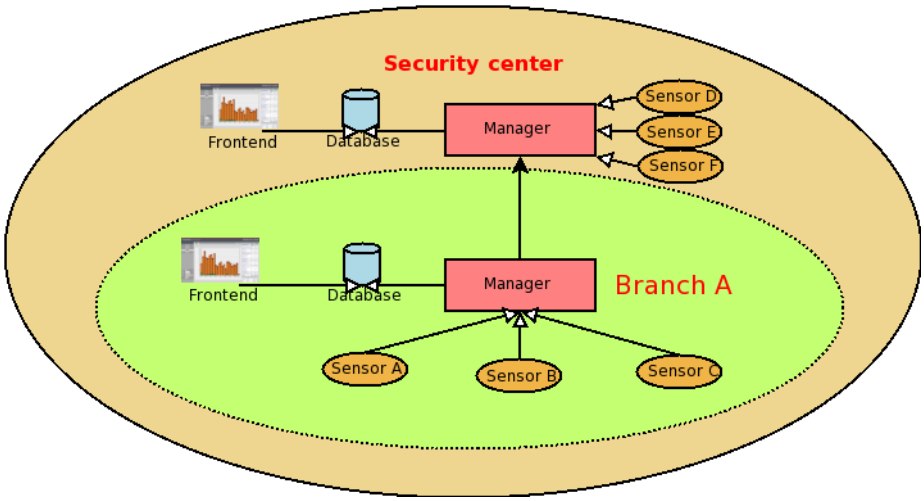
The Prelude console can then be used to view theses events. Here is a simple example of how the differents Prelude components interacts:



### Relaying

Relaying is a feature which allows the prelude-manager program to 'forward' received events to another 'prelude-manager' program.

The following ilustration shows this:

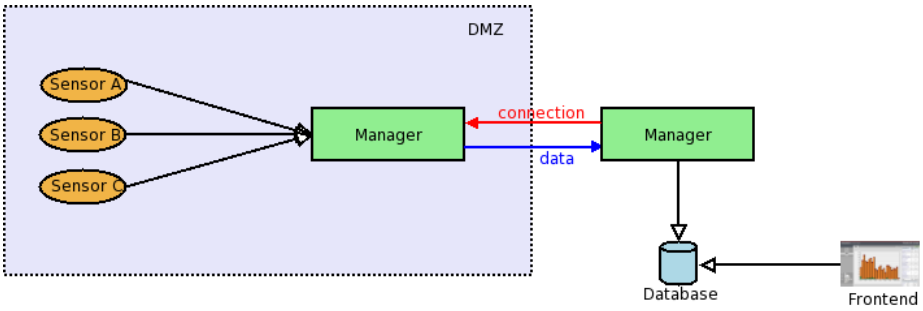



In the above example, 'Branch A' of the company only has access to events generated by Sensor A, B and C. However the company 'Security Center' can see events generated both by it's own sensors (D, E and F), as well as the events generated by the others company branch (like Branch A).

### Reverse relaying

On certain networks, it can sometimes be difficult or troublesome to arrange network permissions so that a program can connect to a server out of a given zone (for example, a firewall might not allow DMZ machines to connect outside of their own network).

In this specific case, you can configure the external 'prelude-manager' program to connect to another internal 'prelude-manager' located inside the DMZ network, and to read the event emited from it.




## 1.3 Prelude components

### Sensor: sending information

A Sensor is a program analysing a stream of information, and generating events when malicious activity is detected. Events are described using the IDMEF (Intrusion Detection Exchange Format) standard, although a binary version instead of XML of this standard is used for speed processing reasons in the Prelude implementation.

Prelude has a large number of sensors, including, but not limited to Snort, Samhain, PreludeLML (itself including hundred of differents sensors), libsafe, (...). You can check the list of available sensors on the Prelude website in the Download section.

If you want to know how to install a sensor, either refer to this handbook, in the section covering installation, or read the installation instructions given by sensors authors. Any sensor must be connected to one or several managers. This can be configured from the system wide Prelude configuration file (which will impact all sensors installed on the system), or on a per sensor basis through their own configuration file or using available command line options.

### Managers: collecting information

The prelude-manager is a high availability server which collects information from distributed sensors or prelude-manager and stores them into a database (or any kind of user provided media).

It also provides the ability to relay received events to one or several other prelude-manager servers, and to filter received events so that you can provide specific actions for specific events.

The communication between a Manager and its clients is encrypted using SSL. In order for a client to be able to communicate with a Prelude-Manager server, the client needs to be registered.

**Frontends: displaying human-readable information**

The frontend provides a means to query the Prelude database, aggregate and filter events, and provides useful statistics about what's going on. It provides a nice interface for the security analyst to see what's going on on the monitored system.

Prewikka is the official front-end.

### 1.4 IDMEF for reporting events

Since Prelude handles events from different kinds of sensors, a generic events description language had to be chosen. Prelude uses IDMEF as the common languages for reporting events. IDMEF started because of the lack of open standard for an Intrusion Detection Systems Exchange Format.

The purpose of the Intrusion Detection Message Exchange Format (IDMEF) is to define data formats and exchange procedures for sharing information of interest to intrusion detection and response systems, and to the management systems which may need to interact with them.

The Intrusion Detection Message Exchange Format (IDMEF) is intended to be a standard data format that automated intrusion detection systems can use to report alerts about events that they deem suspicious. The development of this standard format will enable interoperability among commercial, open source, and research systems, allowing users to mix-and-match the deployment of these systems according to their strong and weak points to obtain an optimal implementation.

The IDMEF Experimental RFC is available on IETF website :
http://tools.ietf.org/rfc/rfc4765.txt

IDMEF is originally intented to be an XML language. However since speed concerns arise when generating and using XML, or when converting events binary data to characters, the Prelude project has written a home-made implementation of the IDMEF specification, using binary structure, and preserving original datatype used to carry specific data.

Full IDMEF-XML compliance is preserved since components like Prelude-Manager and PreludeImport can output and import XML based IDMEF within the Prelude system.

## 2 Requirements

### 2.1 Installation Requirements

Here we will cover the various packages needed to get Prelude installed and working on your *nix system. Each dependency page explains where to get the package and how to install it. Along with other means of installation using yum, apt-get, ports, etc.

### 2.1.1 GnuTLS

**Installing from Source**

**Download Packages**

- libgpg-error
- libgcrypt
- gnutls

You will want to install the above packages in the same order to prevent dependency issues when configuring.

**Installing** First expand the package:

```
tar -xvzf libgpg-error-1.0.tar.gz
```

Then:

```
cd libgpg-error-1.0
```

Now configure, make, and install:

```
./configure && make && make install
```

Now follow the above steps but with the the other two package names.

**Other Installation Means**

Yum users:

```
yum install gnutls gnutls-devel
```

Apt-Get Users:

```
apt-get install gnutls gnutls-devel (some distros it is gnutls-dev)
```

(Please note that many apt-get archives have too old a version of gnutls prior to the version required for Prelude)

Ports Users:

Depending on your OS the same issue may apply as with the users of Apt-Get. The verion in your ports tree may be too old to fit the requirements of Prelude. It is best to just install gnutls from source if at all possible.

**Troubleshooting**

- You have an error when making libgpg-error on a NetBSD 2.0 machine with the default awk.

```
> make [...] gcc -E _mkerrcodes.h | grep GPG_ERR_ |
awk -f ./mkerrcodes.awk >mkerrcodes.h _mkerrcodes.h:57:8:
no macro name given in #ifdef directive _mkerrcodes.h:60:8:
no macro name given in #ifdef directive [...]
In file included from mkerrcodes.c:26:
mkerrcodes.h:3: error: parse error before '}' token [...]
To remedy this issue you should install gawk
(http://ftp.gnu.org/gnu/gawk/) and run again:
./configure && make && make install
```

- It is known that with gnutls-1.2.4 when configuring on an OpenBSD 3.6 machine it will error out when trying to configure OpenPGP support.

To remedy this issue run configure with the additional option of:

```
./configure --disable-openpgp-authentication
```

- You have an error on the libgnutls version it is too old : you should install the new library (http://www.gnupg.org).

- Your have an error after installing gnutls, the version from the config file is 1.0.xx but the library used is 1.0.yy :

```
=> this is a library link mistake you should try as root
>updatedb
>locate libgnutls.so.1.0.yy it will show you where the file is for
example ''/usr/local/lib'' and then you can do :
>ldconfig /usr/local/lib
Retry to ./configure libprelude.
```

### 2.1.2 Python

**Download Package**

- Python

**Installing** First expand the package

```
# tar -xvf python.version.tar.bz2
```

Then, go in the directory you just extracted the files in (within a terminal, and of course, as root)

type the command (==>you should read the README file since it might contain switches that are environment-critical ) :

```
# ./configure
```

As the lines stops scrolling at incredible speed to let you enter yet another command, type in (it's a bit long this time) :

```
# make
```

and to install Python in /usr/local enter this :

```
# make install
```

Python is installed!!!

### 2.1.3 PCRE

describe PCREInstall here please
Well, under CentOS (and others that support this command),
just do a

```
# yum install pcre
```

and it will install it by itself (for other distributions I don't know how thought..)
you can download it from : http://www.pcre.org/
For FreeBSD or OpenBSD

```
# cd /usr/port/devel/pcre && make install
```

For RedHat? ES 3 and 4

```
# up2date pcre;up2date pcre-devel
```

### 2.1.4 MySQL

Start by adding a user:

```
>groupadd mysql >useradd -g mysql mysql
Creation of folders:
>cd /usr/local >mkdir mysql >chown mysql mysql/ >chgrp mysql mysql/
>su - mysql >cd /usr/local/mysql
>mkdir data tmp var
```

Then:

```
>tar -zxvf mysql-4.1.11.tar.gz
>move mysql-4.1.11.tar.gz source
>cd source
>./configure --localstatedir=/usr/local/mysql/data
--with-unix-socket-path=/usr/local/mysql/tmp/mysql.sock
```

Compilation:

```
>make
>make test (optional)
>make install
Change folders right (just to be sure):
>chown -R mysql /usr/local/mysql
>chgrp -R mysql /usr/local/mysql
>chmod 700 /usr/local/mysql/data
>chmod 700 /usr/local/mysql/var
>chmod 755 /usr/local/mysql/tmp
```

Launch the mysql init script:

```
>scripts/mysql_install_db
>chown -R mysql.mysql /usr/local/mysql
Check if everything is all right:
>/usr/local/mysql/bin/mysqld_safe -user=mysql &
>/usr/local/mysql/bin/mysqladmin -u root password <password>
>/usr/local/mysql/bin/mysqladmin -p status
>/usr/local/mysql/bin/mysqladmin -p shutdown
```

**In order to launch mysql automatically there are various methods:**
In the directory where MySQL is uncompressed:

```
cp support-files/mysql.server /usr/local/sbin
```

Then modify /etc/rc .d/rc.local and add:

```
/usr/local/sbin/mysql.server start
chmod 744 /usr/local/sbin/mysql.server
```

Install help in French can be found:
http://www.interpc.fr/mapage/billaud/apmysphp.htm

#### 2.1.5 PostgreSQL

Required Packages

- Postgresql
- Postgresql-dev

# 3 Installation

## 3.1 Installing the Prelude framework

Prelude IDS can be downloaded from the official website:
http://www.prelude-ids.org.

### 3.1.1 Installing the Prelude Library

This section explains how to install the Prelude library libprelude from the tarball available from the Prelude website. However, libprelude might be included with your distribution as a package and it would be easier to install it this way.

**Resolve Dependencies** Ensure that you have GNUTLS installed. In Fedora Core, the gnutls package is crippled. For Fedora Core users, they will need to download and build gnutls from source. Instructions on installing gnutls can be found at https://trac.prelude-ids.org/wiki/GnuTLSInstall

Please be more specific: How is it crippled, in what versions, how will I know when their package is fixed?

**Get the sources** Download the Prelude library at
http://www.prelude-ids.org/download/releases/libprelude-latest.tar.gz

**Compile** In order to compile libprelude, please follow the following instructions:

```
as a user:
  $ ./configure
```

Note that the output should look like:

```
*** Dumping configuration ***
  - Generate documentation : yes
  - Perl binding : yes
  - Python binding : yes
```

In order to generate documentation, you need to use the –enable-gtk-doc command line switch.

If no languages bindings are activated, you will be unable to run sensors in the specified language. Note that Prewikka require Python bindings to be installed.

```
$ make
```

**Install**

```
as root:
  # make install
  # ldconfig
```

Now you have succesfully installed libprelude. You can check the version installed by executing:

```
$ libprelude-config --version
0.9.3
```

### 3.1.2 Installing the Prelude DB Library

This section explains how to install libpreludedb from the tarball available from the main website. However, libpreludedb might be included with your distribution as a package and it would be easier to install it this way.

**Get the sources** Download the file
http://prelude-ids.org/download/releases/libpreludedb-latest.tar.gz

**Compile** Then, compile and install the library using the following:

```
as a user:
  $ ./configure
```

Note that the output should look like:

```
*** Dumping configuration ***
  - Generate documentation : yes
  - Enable MySQL plugin : yes
  - Enable PostgreSQL plugin : no
  - Perl binding : yes
  - Python binding : yes
```

Because the PostgreSQL development files are not available on this system only MySQL was selected (in this case, mysql-devel package is already installed in the system).

If you have neither the MySQL and PostgreSQL plugin selected, you will not be able to use a frontend such as Prewikka. Note that Prewikka require Python bindings to be installed.

```
$ make
```

**Install**

```
as root:
  # make install
  # ldconfig
```

Now you have succesfully installed libpreludedb. You can check the version installed by executing:

```
$ libpreludedb-config --version
0.9.0
```

**Create Database** Once you have installed libpreludedb, the next step is to create a database that will be used by prelude-manager to store the IDMEF alerts gathered from your sensors. Two databases are currently supported by libpreludedb: MySQL and PostgreSQL.

### MySQL

**Database creation** Connect to your database server using the mysql client like this:

```
$ mysql -u root -p
Enter password:
```

where "root" is the name of the database administrator (this is the default account on mysql) and -p will prompt you for a password (by default the root account has no password on mysql and is only accessible from localhost).

Then, if everything is ok, you should see something like this:

```
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 303 to server version: 4.0.22-log
Type 'help;' or '\h' for help.
Type '\c' to clear the buffer.
mysql>
```

To create a new database named 'prelude' (for example):

```
mysql> CREATE database prelude;
Query OK, 1 row affected (0.05 sec)
```

**Create a dedicated user to access the database (optional)** You might want to access your database through a dedicated user (if you don't have already one). If you want to create a new user called 'prelude' with the password 'passwd' that we will have full access on a database called 'prelude' but only from localhost, use the following query:

```
GRANT ALL PRIVILEGES ON prelude.* TO prelude@'localhost'
IDENTIFIED BY 'passwd';
```

**Tables creation** The final step (supposing you have libpreludedb installed in /usr):

```
$ mysql -u prelude prelude -p <
/usr/local/share/libpreludedb/classic/mysql.sql
Enter password:
```

Enter your password, and the tables will be created.

For more details about MySQL databases/tables/users creation, please refer to http://dev.mysql.com/doc/mysql/en/tutorial.html

Updating tablesIf you already have created your tables but that a new libpreludedb version comes with an updated schema, you must update your schema this way, supposing that your current version of schema is 14 and the new one is 14.1:

```
$ mysql -u prelude prelude -p <
/usr/share/libpreludedb/classic/mysql-update-14-1.sql
Enter password:
```

**PostgreSQL** Database creation

Connect to your database server using the psql client like this:

```
$ PGPASSWORD=your_password psql -U postgres
```

where "postgres" is the name of the database administrator (this is the default account on postgresql) and the PGPASSWORD environnement variable is set to the correct password.

Then, if everything is ok, you should see something like this:

```
Welcome to psql 7.3.4, the PostgreSQL interactive terminal.
Type: \copyright for distribution terms
      \h for help with SQL commands
      \? for help on internal slash commands
      \g or terminate with semicolon to execute query
      \q to quit
postgres=#
```

To create a new database named 'prelude' (for example):

```
postgres=# CREATE database prelude; CREATE DATABASE
```

**Create a dedicated user to access the database (optional)** You might want to access your database through a dedicated user (if you don't have already one). If you want to create a new user called 'prelude' with the password 'prelude' that we will have full access on a database called 'prelude', use the following query:

```
CREATE USER prelude WITH ENCRYPTED PASSWORD 'prelude'
NOCREATEDB NOCREATEUSER;
```

**Tables creation** The final step (supposing you have libpreludedb installed in /usr):

```
$ psql -U prelude -d prelude -W -f
/usr/share/libpreludedb/classic/pgsql.sql
```

For more details about PostgreSQL databases/tables/users creation, please refer to http://www.postgresql.org/docs/8.0/interactive/index.html

**Updating tables** If you already have created your tables but that a new libpreludedb version comes with an updated schema, you must update your schema this way, supposing that your current version of schema is 14 and the new one is 14.1:

```
$ PGPASSWORD=prelude psql -U prelude -d prelude <
/usr/share/libpreludedb/classic/pgsql-update-14-1.sql
Enter password:
```

**SQLite3** SQLite is a very good choice if you don't expect lots of data to store. It is a convenient format that doesn't imply database user/password access, since it is a simple file in your system. You have to make sure no unprivileged users have read access to this file.

**Database ans tables creation** Once you have installed libpreludedb, simply run:

As root

```
# mkdir /usr/local/etc/prelude-db
# chown -R prelude:prelude /usr/local/etc/prelude-db
# su prelude
```

If /usr/local/ is your installation prefix.

As prelude

```
$ chmod 700 /usr/local/etc/prelude-db
$ sqlite3 /usr/local/etc/prelude-db/prelude-sqlite-IDMEF-classic.sql <
/usr/local/share/libpreludedb/classic/sqlite.sql
```

**Access to database configuration** Then, in your prelude manager configuration file, access to sqlite3 dabatase should be configured like this:

```
[db]
type = sqlite3
file = /usr/local/etc/prelude-db/prelude-sqlite-IDMEF-classic-14-5.sql
```

### 3.1.3 Installing the Prelude Manager

This section explains how to install prelude-manager from the tarball available from the main website. However, prelude-manager might be included with your distribution as a package and it would be easier to install it this way.

**Get the sources** Download the file

http://www.prelude-ids.org/download/releases/prelude-manager-latest.tar.gz

**Optional Dependencies**

- The database plugin require libpreludedb.
- The XML plugin require libxml2.

**Compile** Then, compile and install the library using the following:
as a user:

```
$ ./configure
```

Note that the output should look like:

```
*** Dumping configuration ***
  - Enable Xml plugin : yes
  - Enable database support: yes
$ make
```

**Install** as root:

```
# make install
```

**Create the Prelude-Manager database** If you want to use database reporting, then you must setup the database.

Install and setup libprelude-db as indicated in InstallingLibpreludedb. Then, edit the Prelude Manager configuration file (usually located in /etc/prelude-manager/prelude-manager.conf, FreeBSD users will find it under usr/local/etc/prelude-manager/prelude-manager.conf-dist and have to delete the trailing -dist) to match your previously setup database. I.e., look for the [db] section and provide the name of and location of the SQL database here.

You should now be able to run the manager with

```
# prelude-manager
```

#### 3.1.4 OS Specific Install Notes

...

## 3.2 Installing Sensors

### 3.2.1 Prelude-LML - the Prelude log analyzer

This section explains how to install PreludeLML from the tarball available from the main website. However, PreludeLML might be included with your distribution as a package and it would be easier to install it this way.

**Get the sources** Download the file
http://www.prelude-ids.org/download/releases/prelude-lml-latest.tar.gz
Then, compile and install the PreludeLML using the following:

```
$ ./configure
```

The output of the configure should be something like:

```
*** Dumping configuration ***
  - Enable FAM support : yes
  - Generate documentation : no
```

FAM support is a file change notification daemon. If you don't have the FAM support, PreludeLML will poll monitored files regularly.

```
$ make # make install
```

**Register the sensor**

```
# prelude-admin register prelude-lml "idmef:w admin:r" <manager address>
```

For more details, please follow the instructions given in the Registering a sensor section.

**Test your installation** An easy way to try your installation is to simulate a malicious activity matching a detection ruleset. Look into the file /usr/local/etc/prelude-lml/prelude-lml.conf which rulesets are loaded. You can add rulesets here from the directory /usr/local/etc/prelude-lml/ruleset .

Here, we will try the ssh ruleset, especially the following signature (this is an excerpt from the file ssh.rules):

```
regex=Failed (\S+) for root from (.+) port (\d+)\s*(ssh2)?; \
classification.text=SSH Remote root login failed; \
[rest of the rule removed for readability]
```

You can see here one of the pattern that PreludeLML will try to match against the received log entry. This specific entry is usually generated by ssh in case of a root login failure. You have to add the line

```
file = /var/log/auth.log
```

to /usr/local/etc/prelude-lml/prelude-lml.conf, so that PreludeLML knows which file it should monitor in order to capture the failed login. Run PreludeLML in a root shell with the command

```
# prelude-lml
```

and also run an instance of the prelude-manager in another root shell. Set the prelude-manager to text output, if you want to check the result in the /var/log/prelude.log.

Running PreludeLML and the prelude-manager should give confirmation messages that the communication between the two is set up correctly.

Let's try it:

```
$ ssh root@<prelude-lml sensor address>
Password: [type whatever is not the appropriate password for the root user]
```

Of course, replace <prelude-lml sensor address> by the address of the machine the sensor is installed in. For instance, if you have installed the sensor on your local machine this would be

```
$ ssh root@localhost
```

The alert should be reported to the PreludeManager collector, and should be readily available from the Prelude frontend. If you have directed your output to a text file like /var/log/prelude.log, and the alert was conducted correctly, you will see something like this:

```
*** Target information ***********************************************
* Target decoy: unknown
* Node[unknown]: name:HOSTNAME
* Addr[ipv4-addr]: 127.0.1.1
* Service: iana_protocol_number=6 iana_protocol_name=tcp port=22 (ssh)
* Process: pid=31095 name=sshd
* os-device user:
* name=root type=target-user
* *** Additional data within the alert*******************************
* Authentication method: password
* Log received from: /var/log/auth.log
* Original Log: Mar 7 16:54:04 HOSTNAME sshd[31095]:
* Failed password for root from 127.0.0. 1 port 59166 ssh2
*********************************************************************
```

### 3.2.2 3rd Party Sensors

- libsafe - middleware that intercepts all function calls made to library functions to prevent buffer overflows and format string vulnerabilities from being exploited.
- Nepenthes? - Nepenthes is a versatile tool to collect malware. It acts passively by emulating known vulnerabilities and downloading malware trying to exploit these vulnerabilities.

- NuFW? - NuFW lays on Netfilter, the state of the art IP filtering layer from the Linux kernel. It fully integrates with Netfilter and Iptables and adds authentication capabilities.
- Samhain - the file integrity checker
- InstallingSancp? Sancp] - designed to collect statistical information regarding network traffic, as well as, collect the traffic itself in pcap format, all for the purpose of: auditing, historical analysis, and network activity discovery.
- Shadow IDS? - is the result of a project that was originally called the Cooperative Intrusion Detection Evaluation and Response (CIDER) project. It was an effort of NSWC Dahlgren, NFR, NSA, the SANS community and other interested parties to locate, document, and improve security software.
- Snort - defacto standard open source intrusion detection system with full Prelude support.

## 3.3 Registering a sensor

*As of libprelude 0.9.15, prelude-adduser is deprecated. Please now use prelude-admin, as specified in the documentation.*

In order for a sensor to communicate with a 'prelude-manager', it needs to be registered. Registration involves several steps:

- Allocating an unique identity for the sensor
- Creating directory to be used by the sensor (example: failover purpose)
- Registering to a remote 'prelude-manager': get a signed X509 certificate that will allows communication between sensor and manager using the specified permissions.

All these informations are stored in a sensor 'profile'.

A sensor profile is identified by its name. When a sensor is started, it will try to load a profile of the same name as the program itself, that is, if your sensor is named "prelude-lml", the sensor will try to load a profile named "prelude-lml".

The name of the profile can be overriden using the '–prelude –profile name_of_my_profile' command line option. We provide the ability of defining the profile name so that you can have several instances of one sensor running with different permissions, which require different profiles.

Note that profile are not specific to sensor, but are used in all programs of the Prelude suite (sensors, managers, etc).

The registration process is driven by a single tool called prelude-admin.

```
$ prelude-admin register <profile name> <requested permission>
<manager address> --uid <uid> --gid <gid>
```

Replace <profile name> with the name of the sensor you are installing, or with your own defined name if you want more advanced sensor profile control. If you start your sensor without it being registered, it should show you a warning including the default profile name on how to register the sensor.

Remember to use the correct uid/gid when registering your sensor. For instance, if you want to register snort (running with snort euid / egid), use –uid snort –gid snort. If the sensor process cannot read libprelude information (key, cert...), it won't work.

Replace <requested permission> with the permission your sensor needs. There are several kind of permission:

- idmef
- admin

Both type can take permission r (read) and w (write). Usually, a sensor will need permission for writing IDMEF to a manager, and reading administrative command sent to it. That is : "idmef:w admin:r". If you are not sure which permission your sensor should get, just start the sensor, which should then provide you with the prelude-admin options to use for registering it.

You should replace the <manager address> argument by the address where the prelude-manager you wish to register to is running, this can either be its IP address or its hostname name. Typically, if you made a local installation, you can write localhost there.

You need to repeat this step for each manager you want to register the sensor. When you are not sure about how your sensor should be registered, just start the sensor, which should then provide you with the prelude-admin options to use for registering it:

```
prelude-client-profile: error creating prelude-client:
Could not open AnalyzerID file.
Basic file configuration does not exist. Please run :
prelude-admin register prelude-lml "idmef:w admin:r" <manager address>
--uid 1000 --gid 100 program to setup the analyzer.
Be aware that you should replace the "<manager address>" argument with
the server address this analyzer is reporting to as argument.
"prelude-admin" should be called for each configured server address.
```

The default is to create the sensor profile using the UID and GID of the user who launched the prelude-admin command. If you want the profile to be run by another set of permission, use the –uid and –gid options.

Here is an example on registering 'prelude-lml' to a prelude-manager running on the same machine:

```
$ prelude-admin register prelude-lml "idmef:w admin:r" localhost
```

prelude-admin will ask you to start another instance of 'prelude-admin' on the machine where the prelude-manager server is listening (localhost in this example).

```
You now need to start "prelude-admin" on the server host where you
need to register to:
use: "prelude-admin registration-server <analyzer profile>" example:
"prelude-admin registration-server prelude-manager"
This is used in order to register the 'sending' analyzer to the
'receiving' analyzer. <analyzer profile> should be set to the profile
name of the 'receiving' analyzer, the one where 'sending' analyzer
will register to.
Please remember that "prelude-admin" should be used to register every
server used by this analyzer.
```

Now start prelude-admin using the registration-server command, and the profile name used by the 'prelude-manager':

```
$ prelude-admin registration-server prelude-manager
```

This will display the output:

```
- Starting registration server.
  - generated one-shot password is "deadbeaf".

    This password will be requested by "prelude-admin" in order to
    connect. Please remove the first and last quote from this password
    before using it.
  - Waiting for install request from peer...
```

As you can see, the generated password is "deadbeaf". That is what you need to type in the session of prelude-admin on the prelude-lml side.

There, you will see:

```
- Enter registration one shot password: [you don't see this, but
deadbeaf is typed]
- Please confirm one shot password: [you don't see this, but deadbeaf
is typed]
- connecting to registration server (localhost:5553)...
- Sending certificate request.
- Receiving CA signed certificate.
- Receiving CA certificate.
- prelude-lml registration to localhost successful.
```

and on the server side:

```
- Waiting for install request from peer...
- Connection from 127.0.0.1:57232.
- Waiting for client certificate request.
- Analyzer with ID="1537698187535812" ask for registration with
permission="idmef:w admin:r". Approve registration [y/n]: y
Registering analyzer "1537698187535812" with permission "idmef:w
admin:r". - Generating signed certificate for client.
- Sending server certificate to client.
- 127.0.0.1:30098 successfully registered.
```

The operation was successful! congratulations, you now have a sensor up and running.

## 3.4 Installing the Prewikka console

This section explains how to install the Prelude frontend Prewikka using the tarball available from the Prelude website. However, Prewikka might be included with your distribution as a package and it would be easier to install it this way.

**Requirements** * Prewikka 0.9.9 depends on libpreludedb version 0.9.9 (database schema 0.9.1) * Prewikka 0.9.4 depends on libpreludedb version 0.9.7 (database schema 0.9.1) * Prewikka 0.9.1 and 0.9.2 depends on libpreludedb version 0.9.1 (database schema 0.9.1)

**Get the sources** Download Prewikka at http://www.prelude-ids.org/download/releases/prewikka-latest.tar.gz

**Install** You might install Prewikka as root:

```
# python setup.py install
```

Or as an user:

```
$ python setup.py install --prefix /prefix/where/prewikka/should/be/installed
```

You will need Python 2.3 from http://www.python.org/ and Cheetah templates for Python from http://cheetahtemplate.org/. Additionlally you might need python2.3-dev. Create the

**Prewikka database** Make sure the Prelude Framework (libprelude + libpreludedb with python bindings) is installed. Please read: https://trac.prelude-ids.org/wiki/InstallingPrelude.

**MySQL**

**Database creation** Connect to your database server using the mysql client like this:

```
$ mysql -u root -p
Enter password:
```

where "root" is the name of the database administrator (this is the default account on mysql) and -p will prompt you for a password (by default the root account has no password on mysql and is only accessible from localhost).

Then, if everything is ok, you should see something like this:

```
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 303 to server version: 4.0.22-log
Type 'help;' or '\h' for help.
Type '\c' to clear the buffer.
mysql>
```

To create a new database named 'prewikka' (for example):

```
mysql> CREATE database prewikka; Query OK, 1 row affected (0.05 sec)
```

**Create a dedicated user to access the database (optional)** You might want to access your database through a didacted user (if you don't have already one). If you want to create a new user called 'prewikka' with the password 'password' that we will have full access on a database called 'prewikka' but only from localhost, use the following query:

```
GRANT ALL PRIVILEGES ON prewikka.* TO prewikka@'localhost'
IDENTIFIED BY 'password';
```

**Tables creation** The final step (supposing you have prewikka installed in /usr):

```
$ mysql -u prewikka prewikka -p < /usr/share/prewikka/database/mysql.sql
Enter password:
```

Enter your password, and the tables will be created.

For more details about MySQL databases/tables/users creation, please refer to http://dev.mysql.com/doc/mysql/en/tutorial.html

**PostgreSQL**

**Database creation** Connect to your database server using the psql client like this:

```
$ PGPASSWORD=your_password psql -U postgres
```

where "postgres" is the name of the database administrator (this is the default account on postgresql) and the PGPASSWORD environnement variable is set to the correct password.

Then, if everything is ok, you should see something like this:

```
Welcome to psql 7.3.4, the PostgreSQL interactive terminal.
Type: \copyright for distribution terms
      \h for help with SQL commands
      \? for help on internal slash commands
      \g or terminate with semicolon to execute query
      \q to quit
postgres=#
```

To create a new database named 'prewikka' (for example):

```
postgres> CREATE database prewikka; CREATE DATABASE
```

**Create a dedicated user to access the database (optional)** You might want to access your database through a didacted user (if you don't have already one). If you want to create a new user called 'prewikka' with the password 'prewikka' that we will have full access on a database called 'prewikka', use the following query:

```
CREATE USER prewikka WITH ENCRYPTED PASSWORD 'prewikka'
NOCREATEDB NOCREATEUSER;
```

**Tables creation** The final step (supposing you have prewikka installed in /usr):

```
$ PGPASSWORD=prewikka psql -U prewikka -d prewikka < /usr/share/prewikka/database/pgsql
```

For more details about PostgreSQL databases/tables/users creation, please refer to http://www.postgresql.org/docs/8.0/interactive/index.html

**SQLite3**

**Database and tables creation** Make sure /usr/local/etc/prelude-db has the rights to read for and only for user prelude (or root if you have not created one).

```
$ sqlite3 /usr/local/etc/prelude-db/prelude-sqlite-PREWIKKA.sql <
/usr/share/prewikka/database/sqlite.sql
```

then, don't follow the following instructions concerning database:

```
[idmef_database] type: sqlite3 file:
/usr/local/etc/prelude-db/prelude-sqlite-IDMEF-classic.sql
[database] type: sqlite3 file:
/usr/local/etc/prelude-db/prelude-sqlite-PREWIKKA.sql
```

**Editing prewikka.conf** Once you have created the database for Prewikka you need to edit /etc/prewikka/prewikka.conf to fit your database settings prior to starting Prewikka.

```
[interface]
#This is the name at the top right and left of the Prewikka interface
#You can change it or leave as is software: Prewikka place: company
#ltd. title: Prelude management
#The following are the setting for your prelude database [idmef_database]
type: mysql
host: localhost
user: prelude
pass: prelude
name: prelude
#This is the database information for the prewikka DB you created
#above
[database]
type: mysql
host: localhost
user: prewikka
pass: prewikka
name: prewikka
#You can comment this out to stop logs from writing to stderr
[log stderr]
#No real need to edit this except to increase/decrease expiration time
[auth loginpassword]
expiration: 60
```

**Running Prewikka from the Apache web server (CGI/Apache)**

**Apache / CGI setup with VirtualHost?**

```
<VirtualHost *:80>
  ServerName my.server.org
  Setenv PREWIKKA_CONFIG "/etc/prewikka/prewikka.conf"
<Location "/">
  AllowOverride None
  Options ExecCGI
  <IfModule mod_mime.c>
    AddHandler cgi-script .cgi
  </IfModule>
  Order allow,deny
  Allow from all
</Location>
  Alias /prewikka/ /usr/share/prewikka/htdocs/
  ScriptAlias / /usr/share/prewikka/cgi-bin/prewikka.cgi
</VirtualHost>
```

Note that the PREWIKKA_CONFIG environment variable is optional, if unset the default configuration file relative to your installation path will be used.

**Apache / mod_python setup with VirtualHost?**

```
<VirtualHost *:80>
  ServerName my.server.org
<Location />
  SetHandler mod_python
  PythonHandler prewikka.ModPythonHandler
  PythonOption PrewikkaConfig /etc/prewikka/prewikka.conf
</Location>
<Location /prewikka>
  SetHandler None
</Location>
  Alias /prewikka /usr/share/prewikka/htdocs
</VirtualHost>
```

Note that the PrewikkaConfig? settings is optional, if unset the default configuration file relative to your installation path will be used.

**Running Prewikka from the lighttpd webserver**

**lighttpd setup with mod_cgi** First, make sure you enable mod_cgi in your lighttpd.conf, and add prewikka.cgi to your server.indexfiles:

```
server.modules = ("mod_cgi")
server.indexfiles = ("prewikka.cgi")
```

Then, add the following alias and cgi options:

```
$HTTP["url"] =~ "^/prewikka" {
  alias.url = (
    "/prewikka/" => "/usr/share/prewikka/htdocs",
    "/prewikka" => "/usr/share/prewikka/cgi-bin"
    )
  cgi.assign = ( ".cgi" => "" )
}
```

With newer versions of lighttpd like 1.4.11, you should use the following cgi options instead:

```
$HTTP["url"] =~ "^/prewikka" {
  alias.url = (
    "/prewikka/prewikka/images" => "/usr/share/prewikka/htdocs/images",
    "/prewikka/prewikka/css" => "/usr/share/prewikka/htdocs/css",
    "/prewikka/prewikka/js" => "/usr/share/prewikka/htdocs/js",
    "/prewikka" => "/usr/share/prewikka/cgi-bin/prewikka.cgi"
    )
  cgi.assign = ( ".cgi" => "" )
}
```

You should then be able to access prewikka through http://yourmachine/prewikka.

**Running Prewikka from the boa webserver** Edit /etc/boa/boa.conf and append the following lines:

```
# prewikka addon-lines BEGIN ###################################
Alias /prewikka/prewikka/images /usr/share/prewikka/htdocs/images
Alias /prewikka/prewikka/css /usr/share/prewikka/htdocs/css Alias
/prewikka/prewikka/js /usr/share/prewikka/htdocs/js
ScriptAlias /prewikka/ /usr/share/prewikka/cgi-bin/
# prewikka addon-lines END #####################################
```

You should then be able to access prewikka through http://yourmachine/prewikka/prewikka.cgi. This setup works fine at least with boa-0.94.14. With newer versions of prewikka you need a patched version of boa. The MAX_HEADER_LENGTH value (a compiled in parameter) is too small for prewikka, but this is very easy to fix if you are able to compile boa by yourself. Simply change the value of MAX_HEADER_LENGTH from 1024 to 2048 in file src/defines.h and compile the sources.

**Running Prewikka from the command line tool** If you didn't install Prewikka system wide (ie: you specified a prefix), use:

```
$ PYTHONPATH=$prefix/lib/python2.3/site-packages
$prefix/bin/prewikka-httpd
```

If you installed Prewikka system wide:

```
$ /usr/bin/prewikka-httpd
```

You can then use your browser to connect to your machine on port 8000. The default login/password is admin: please remember to change it.

**Initial login** Once everything is setup, you can use your browser to connect to the machine were Prewikka was installed. If you are not using Apache support, then remeber you should use the port 8000 to access Prewikka.

The default login/password is admin: please remember to change it.

**SELinux impact (Fedora Core 3)** If you find that prewikka is not able to connect to the local MySQL server then you may be encountering a problem with SELinux. Starting with Fedora Core 3, SELinux is turned on by default. SELinux is the Secure Edition that has extra security features. There are many ways to handle this, but a convenient way is from the Start menu on the desktop. Use System Settings > Security Settings > SELinux. You can disable SELinux completely or just for the mysqld daemon. (gdk - Telcordia 5/2/05)

**See also**

- InstallingPrewikka
- ExecutionPrewikka

# 4 Dig into prelude's world

## 4.1 Generic, System wide configuration

All Prelude sensors have a common set of options, provided through the Prelude framework. You might modify these options system wide or in the Prelude specific configuration file a client might provide.

All options defined system wide might be overriden in the client own Prelude configuration file. These are just template values that the client will use in case the values are not defined in the client own Prelude configuration file.

Once the Prelude library is installed, you can tune system wide options using the following configuration files (replace $PREFIX with your installation prefix, usually /usr or /usr/local):

```
$PREFIX/etc/prelude/default/client.conf
$PREFIX/etc/prelude/default/global.conf
$PREFIX/etc/prelude/default/idmef-client.conf
```

**$PREFIX/etc/prelude/default/global.conf** This is the common configuration file used by all Prelude programs (sensors, prelude-manager). It provides a system wide template for common IDMEF attributes used by sensors.

All of theses settings are optional, but keep in mind setting them will help you to keep track of where an event is coming from, especially in a distributed environment with a high number of sensors.

The heartbeat-interval option defines how often a Prelude client should send a heartbeat (the default is 600 seconds).

You can define IDMEF attributes to be carried by events emitted by the programs using the framework. All of these settings are optional, but keep in mind setting them will help you to keep track of where an event is coming from, especially in a distributed environment with a high number of sensors.

- node-name: Name of the equipment (usually the name of the machine this sensor is running on).
- node-location: Location of the equipment (could be a city, or a country).
- node-category: The type of node the clients are runing on (usually hosts).

You might also want to define one or several node-address section, containing the following option:

- address: The address of the equipment.
- netmask: Netmask for this address.
- vlan-name: Name of the Virtual Lan to which the address resides in.
- vlan-num: Number of the Virtual Lan to which the address resides in.
- category: Type of address represented (usually ipv4-addr or ipv6-addr).

**$PREFIX/etc/prelude/default/client.conf** In this configuration file, you can configure the connection string that clients, which need to connect to a Prelude Manager, will use. You can use boolean '\|\|' (OR) and '&&' (AND) to set up a redundant configuration environment.

Note that whatever you specify here, any events sent through the Prelude framework are saved in case the remote Manager goes down. In this case, all events will be saved and the client will periodically attempt to reconnect and flush saved events.

Here are a few configuration examples:

```
server-addr = x.x.x.x
```

Connect and send events to x.x.x.x.

```
server-addr = x.x.x.x && y.y.y.y
```

Connect and send events to both x.x.x.x and y.y.y.y.

```
server-addr = x.x.x.x || y.y.y.y
```

Connect and send events to x.x.x.x, or fallback to y.y.y.y if x.x.x.x failed.

**$PREFIX/etc/prelude/default/idmef-client.conf** This file includes both $PREFIX/etc/prelude/default/global.conf and *$PREFIX/etc/prelude/default/client.conf*. It is often used as the template by clients that need to connect to a prelude-manager.

You probably should not modify this file directly (use global.conf and client.conf).

## 4.2 Prelude Manager

Prelude-Manager is a high availability server that accepts secured connections from distributed sensors or other managers and saves received events to a media specified by the user (database, logfile, mail, etc).

The server is a high availability server capable of handling large number of connections, and processing large amounts of events. It uses a per client scheduling queues in order to process events by severity fairly accross clients.

Prelude Manager can listen on an UNIX domain socket, or on an IPv4 or IPv6 address. The default is to listen on an UNIX domain socket. You might change this using the listen command line option or configuration directive.

```
listen = unix
```

Will listen on /tmp/.prelude-unix UNIX domain socket (this is the default).

```
listen = unix:/tmp/myfilename
```

Will listen on /tmp/myfilename UNIX domain socket.

```
listen = x.x.x.x
```

Will listen on the specified IP address.

**Reporting** Once an event has been processed, the Manager uses Reporting Plugins to convert alerts from Prelude binary IDMEF format, to various output formats.

There are several Reporting Plugins:

- db - A database Plugin (MySQL and PostgreSQL).
- xmlmod - An XML Reporting Plugin.
- textmod - A text Reporting Plugin.
- relaying - A plugin relaying alert to another set of manager.
- smtp - A commercial plugin sending textual alert through your SMTP server.
- (see prelude-manager –help for others)

A reporting plugin might be loaded several times, so you could have two db reporting plugin loaded and addressing different database (this is true for most plugin categories).

In order for a plugin to be loaded several times, you need to provide it with the name of each instance. If you don't provide a name, then default is used.

Here is an example:

```
# prelude-manager --textmod --logfile stderr --textmod MyInstanceName
--logfile /var/log/prelude.log
```

Or from the configuration file:

```
[textmod] logfile = stderr
[textmod=MyInstanceName] logfile = /var/log/prelude.log
```

Will load an instance of the textmod reporting plugin named default, which will log to stderr; and another instance named MyInstanceName? logging to /var/log/prelude.log.

**Database** In order for prelude-manager to report to a database, you need to use the db reporting plugin. Additionaly, you need to provide this plugin with several arguments:

- type : The type of database (mysql or pgsql).
- host : The hostname where the database is listening (default is localhost).
- port : The port where the database is listening, if applicable.
- name : The name of the database to use.
- user : The username to use in order to connect to this database.
- pass : The password to use in order to connect to this database.

```
prelude-manager --db --type mysql --host localhost --name prelude
--user prelude --pass XXXXXX
```

Or from the configuration file:

```
[db]
type = mysql
host = localhost
name = prelude
user = prelude
pass = XXXXXX
```

**Relaying** Prelude Managers can also act as a relays to another Manager using the relaying plugin. Relaying Managers control a set of Analyzers, and relay alerts and messages received from theses analyzer to another Parent Manager. This is a very useful feature when a network is divided into sub-networks.

```
prelude-manager --relaying --parent-managers "x.x.x.x"
```

Started this way, the prelude-manager will connect to the specified parent manager (x.x.x.x in this example). Note that you can specify more complex connection information, like:

```
prelude-manager --relaying --parent-managers "x.x.x.x && y.y.y.y"
```

Relaying information to x.x.x.x and y.y.y.y, or even:

```
prelude-manager --relaying --parent-managers "x.x.x.x || y.y.y.y"
```

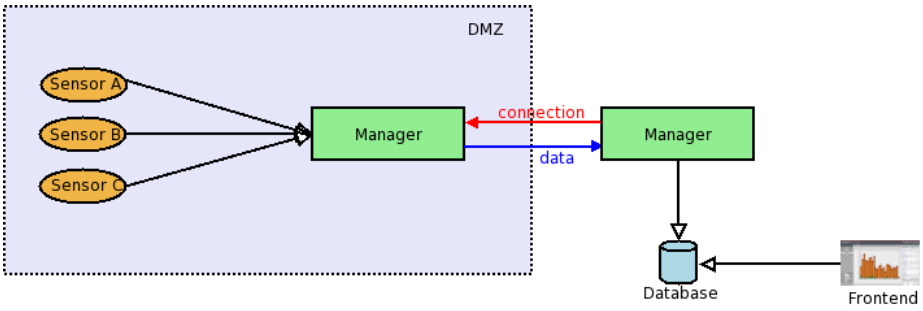
Relaying information to x.x.x.x or 'y.y.y.y' if x.x.x.x goes down.

This option might also be specified from the prelude-manager configuration file:

```
[relaying]
parent-managers = x.x.x.x && y.y.y.y
```

**Smtp** A commercial SMTP plugin is available from PreludeIDS Technologies. This plugin provide the ability to send textual alert as mail. Please check the Corporate Modules page for more information.

**Reverse relaying** Under certain conditions (an example of this are DMZ network), it might be difficult for an administrator to allow a set of analyzers on a given network to contact a manager located on another network. However, if connections to the analyzer network are permited, you might use Reverse Relaying to solve your problem. In this situation, the manager connects to the Analyzer in order to receive the information.



```
prelude-manager --child-managers "x.x.x.x"
```

from the command line option, or in the prelude-manager configuration file:

```
child-managers = "x.x.x.x"
```

will make prelude-manager connect to x.x.x.x and retrieve events from it. Connection strings are also supported, as specified in the Relaying section above.

Prelude Managers can also act as a relays to another Manager using the relaying plugin. Relaying Managers control a set of Analyzers, and relay alerts and messages received from theses analyzer to another Parent Manager. This is a very useful feature when a network is divided into sub-networks.

**Filtering events** The idmef-criteria filtering plugin allow you to filter events basing on specific IDMEF-Criteria. A filtering plugin might be used to decide whether a specific action should be taken.

```
prelude-manager --idmef-criteria --rule 'alert.classification.text ==
User login successful' --rule 'alert.assessment.impact.severity ==
high' --hook relaying[default]
```

or from the configuration file:

```
[idmef-criteria]
rule = alert.classification.text == User login successful
rule = alert.assessment.impact.severity == high
hook = relaying[default]
```

Will forward any events that match the defined criteria to the default instance of the relaying reporting plugin. The rule argument might also be a filename containing the rules.

In order to learn more about the available IDMEF Path, please have a look at the IDMEFPath. More documentation on how to setup IDMEF criteria is available on the IDMEFCriteria page.

**Event Suppression** The thresholding filtering plugin allow you to suppress events based on their content.

```
prelude-manager --thresholding --path
'alert.classification.text,alert.source.node.address.address' --limit
3600 --count 1 --hook relaying[default]
```

or from the configuration file:

```
[thresholding] path = alert.classification.text,
alert.source.node.address.address
limit = 3600
count = 1
hook = relaying[default]
```

Will forward one event with the unique alert.classification.text, alert.source.node.address.address value combination to the default instance of the relaying reporting plugin. Further events with the same value will be suppressed for 3600 seconds.

```
prelude-manager --thresholding --path
'alert.classification.text,alert.source.node.address.address'
--threshold 3600 --count 10 --hook relaying[default]
```

or from the configuration file:

```
[thresholding] path = alert.classification.text,
alert.source.node.address.address
threshold = 3600
count = 10
hook = relaying[default]
```

Will forward every tenth event within 3600 seconds with the unique alert.classification.text, alert.source.node.address.address value combination to the default instance of the relaying reporting plugin.

You can also combine threshold and limit, so that once the threshold is reached once, further events with the same value will be suppressed, until the limit expire.

In order to learn more about the available IDMEF Path, please have a look at the IDMEFPath. More documentation on how to write IDMEF criteria is available on the IDMEFCriteria page.

**Stacking Filtering Plugins Together** The ability to stack plugins, allows you to create more complex abilities in how you use the plugins. We'll use the example below, to show how idmef-criteria, thresholding and smtp can all be combined (ie. stacked) to enhance your reporting.

Example:

```
[smtp]
sender = prelude@mycompanyname.com
recipients = me@mycompanyname.com
smtp-server = mailserver.mycompanyname.com
[thresholding=sudotest]
path =
alert.classification.text,alert.target(0).node.address(0).address
threshold = 1
seconds = 3600
hook = smtp[default]
[idmef-criteria-filter=sudotest]
rule = alert.classification.text == 'SUDO Command Executed'
hook = thresholding[sudotest]
```

In the above example, we have created an instance of the idmef-criteria-filter called 'sudotest'. It looks for 'SUDO Command Executed' in an IDMEF alert, and when it matches, it passes this to the 'sudotest' instance of the thresholding plugin (this can be seen via the 'hook = thresholding[sudotest]).

The 'sudotest' instance of the thresholding plugin, will then keep track of this 'SUDO Command Executed' text, along with the target node's IP address. The threshold of '1', combined with the seconds of '3600', will suppress further of these messages with those two criteria from being passed to the smtp plugin for one hour after one has been seen. After that time period, messages with that text and target node address will generate another smtp message - before being suppressed for one hour again. Note: This does not affect other plugins, such as the database, as that is not hooked in our example.

Finally, the email is sent as the 'sudotest' instance of the thresholding plugin hooks the smtp[default] plugin.

Basically, the stacking of plugins in this example allowed us to look for certain IDMEF criteria, then both email on it matching, and suppress the number of emails generated for this match for a certain time period.

**Report plugins failover** Under certain conditions, you might want to activate failover for a given reporting plugin (as of the time of writing, only the db reporting plugin support it). Failover is a prelude-manager feature allowing to store alerts in case the reporting plugin fails. This is especially useful when the reporting plugin rely on another program (example: database).

If the protected reporting plugins fail, for any reason, to process the message, the failover subsystem will backup it as well as future messages and then will periodicaly try to re-initialize the plugin, and to process backed up message.

As an example, if you activate report plugins failover for the db plugin, and that you stop the underlying database. Message going to the plugin will be saved, and processed as soon as the db plugin is sucessfuly reinitialized (in this case, when the database goes up again).

```
--failover db --failover db[MyPluginInstance]
```

or from the configuration file:

```
failover = db
failover = db[MyPluginInstance]
```

Will both activate failover for the default instance of the db plugin, and for the MyPluginInstance? of this same plugin.

Note: It is not necessary to activate failover for the relaying plugins since the internal subsystem for this plugin will always failover in case of failure.

### Other configuration options

- daemon : Start prelude-manager as a daemon.

- pidfile : Write the prelude-manager PID to the specified file.
- config : Specify an alernate configuration file.

Please check the prelude-manager –help output, or have a look to the prelude-manager configuration file for an exhaustive list.

## 4.3 Prelude-LML

Prelude-LML, or Prelude Log Monitoring Lackey, is a part of the project that deals with host based intrusion detection aspects through log analysis. It can monitor files created by a syslog daemon coming from different hosts on heterogeneous platforms, other types of single-line event logs, or simulate a syslog server on its own. Thus any system generating logs can benefit from the Prelude-LML?s analysis engine.

- Some example platforms are:
  - Unix syssdtems
  - Switches and routers
  - Firewalls
  - Printers
  - Others systems which can log in Syslog format (like Windows NT/2K/XP with tools like Ntsyslog)

By placing Prelude-LML on a network and configuring other machines to send log messages to the Syslog daemon, it is possible to monitor an entire network of machines? logs.

- Prelude-LML has two modes of operation:
  - Watch log files on the host where it is running (syslog or any other).
  - Receive UDP syslog messages from other hosts on the network.

Prelude-LML?s primary function is log analysis. Logs on a local system or logs monitored over the network (if configured to accept syslog messages from other hosts) can be processed and analyzed in order to discover security anomalies.

Prelude-LML has a plugin system which actually performs all analysis and monitoring.

One of these plugins, called Pcre, is a regular expression engine powered by the PCRE (Perl Compatible Regular Expression) library. This plugin is used in Prelude LML to match a set of regular expressions (in common terms, signatures). Each ruleset provides regular expression matching for a particular purpose. Therefore, the Netfilter ruleset ?watches? for netfilter messages, and the GRSecurity ruleset ?watches? for GRSecurity messages. The configuration file tells Prelude-LML what analysis plugins to load and use to process logs.

Here is a non exhaustive list of logs that Prelude-LML Pcre plugin has rulesets for:

- APC Environmental Monitoring Unit
- arpwatch
- F5 Big-IP
- Cisco PIX
- Cisco Router
- Cisco VPN Concentrator
- Clam Antivirus
- Dell OpenManage
- GRSecurity
- Honeyd
- IPChains
- IPFW
- Checkpoint IPSO
- mod_security
- Norton Antivirus Corporate Edition
- NetApp? ONTAP
- Netfilter
- Windows NT/200x/XP
- PAM
- pcAnywhere
- SentryTools? Portsentry
- Postfix
- ProFTPd
- QPopper
- SELinux
- Sendmail
- GNU Shadow Utils
- Squid Proxy

- OpenSSH sshd
- GNU sudo
- Open Source Tripwire
- Vigor
- Vpopmail
- Linksys WAP11
- Webmin
- WU-FTPd
- Exim
- Oracle
- tftpd
- P3Scan
- D-Link Wireless Router

With the power of PCRE (the regexp engine that the signature engine uses), writing additional rules is an easy task.

**Multiple log files, different format** If you use multiple log file with different formatting, you can configure LML so that it know how to handle each log format. This is done using a format section, in the PreludeLML configuration file:

```
[format=syslog]
time-format = "%b %d %H:%M:%S"
prefix-regex = "^(?P<timestamp>.{15}) (?P<hostname>\S+)
(?:(?P<process>\S+?)(?:\[(?P<pid>[0-9]+)\])?: )?"
file = /var/log/messages
file = /var/log/auth.log
udp-server = 0.0.0.0:514
```

Additionally, LML can accept different format from a single log source (be it a file, or the UDP server). As an example, using the following configuration, LML will know how to parse any of the specified format from /var/log/mylogfile and UDP 0.0.0.0:514:

```
[format=syslog]
time-format = "%b %d %H:%M:%S"
prefix-regex = "^(?P<timestamp>.{15}) (?P<hostname>\S+)
(?:(?P<process>\S+?)(?:\[(?P<pid>[0-9]+)\])?: )?"
file = /var/log/mylogfile
udp-server = 0.0.0.0:514
[format=apache]
time-format = "%d/%b/%Y:%H:%M:%S"
prefix-regex ="^(?P<hostname>\S+) - - \[(?P<timestamp>.{20}) \+.{4}\]"
file = /var/log/mylogfile
udp-server = 0.0.0.0:514
```

The format section allow several option:

- prefix-regex

This tell LML how to handle the log header. With this option, you can bind variable used by LML to fill specific fields in the generated IDMEF Alert. Variable that you can use are:

| Variable | Usage |
|---|---|
| timestamp | Bind to the DetectTime? information of an IDMEF Alert |
| hostname | Bind to the Target node information in an IDMEF Alert |
| process | Bind to the Target process name in an IDMEF Alert |
| pid | Bind to the Target process pid in an IDMEF Alert |

Here is an example of how it work, using the following prefix-regex:

```
prefix-regex = "^(?P<timestamp>.{15}) (?P<hostname>\S+)
(?:(?P<process>\S+?)(?:\[(?P<pid>[0-9]+)\])?: )?"
```

Together with the following log entry:

```
Dec 30 20:09:03 hacklab honeyd[5711]: this is a log entry
```

When LML parse this log entry using the above prefix-regex, the timestamp, hostname, process, and pid variable will be set to the following value:

| Variable | Value |
|---|---|
| timestamp | Dec 30 20:09:03 |
| hostname | hacklab |
| process | honeyd |
| pid | 5711 |

Each of these value will be assigned in the relevant IDMEF fields of the generated alert. Please note that the timestamp variable is specific in that you have to specify an additional time-format option, so that LML is able to parse the time representation.

- time-format

Should be set so that LML know how to format the timestamp in your log entry. This will be used to match the timestamp vaiable content defined through the prefix-regex option.

| Sequence | Description |
|---|---|
| %% | The % character |
| %a or %A | The weekday name according to the current locale, in abbreviated form or the full name. |
| %b or %B or %h | The month name according to the current locale, in abbreviated form or the full name. %c The date and time representation for the current locale. |
| %C | The century number (0-99). |
| %d or %e | The day of month (1-31). |
| %D | Equivalent to %m/%d/%y. (This is the American style date, very confusing to non-Americans, especially since %d/%m/%y is widely used in Europe. The ISO 8601 standard format is %Y-%m-%d.) |
| %H | The hour (0-23). |
| %I | The hour on a 12-hour clock (1-12). |
| %j | The day number in the year (1-366). |
| %m | The month number (1-12). |
| %M | The minute (0-59). |
| %n | Arbitrary whitespace. |
| %p | The locale?s equivalent of AM or PM. (Note: there may be none.) |
| %r | The 12-hour clock time (using the locale?s AM or PM). In the POSIX locale equivalent to %I:%M:%S %p. If t_fmt_ampm is empty in the LC_TIME part of the current locale then the behaviour is undefined. |
| %R | Equivalent to %H:%M. |
| %S | The second (0-60; 60 may occur for leap seconds; earlier also 61 was allowed). |
| %t | Arbitrary whitespace. |
| %T | Equivalent to %H:%M:%S. |
| %U | The week number with Sunday the first day of the week (0-53). The first Sunday of January is the first day of week 1. |
| %w | The weekday number (0-6) with Sunday = 0. |
| %W | The week number with Monday the first day of the week (0-53). The first Monday of January is the first day of week 1. |
| %x | The date, using the locale?s date format. |
| %X | The time, using the locale?s time format. |
| %y | The year within century (0-99). When a century is not otherwise specified, values in the range 69-99 refer to years in the twentieth century (1969-1999); values in the range 00-68 refer to years in the twenty-first century (2000-2068). |
| %Y | The year, including century (for example, 1991). |
| | |

Some field descriptors can be modified by the E or O modifier characters to indicate that an alternative format or specification should be used. If the alternative format or specification does not exist in the current locale, the unmodified field descriptor is used.

The E modifier specifies that the input string may contain alternative locale-dependent versions of the date and time representation:

| Sequence | Description |
|---|---|
| %Ec | The locale?s alternative date and time representation. |
| %EC | The name of the base year (period) in the locale?s alternative representation. |
| %Ex | The locale?s alternative date representation. |
| %EX | The locale?s alternative time representation. |
| %Ey | The offset from %EC (year only) in the locale?s alternative representation. |
| %EY | The full alternative year representation. |

The O modifier specifies that the numerical input may be in an alternative locale-dependent format:

| Sequence | Description |
|---|---|
| %Od or %Oe | The day of the month using the locale?s alternative numeric symbols; leading zeros are permitted but not required. |
| %OH | The hour (24-hour clock) using the locale?s alternative numeric symbols. |
| %OI | The hour (12-hour clock) using the locale?s alternative numeric symbols. |
| %Om | The month using the locale?s alternative numeric symbols. |
| %OM | The minutes using the locale?s alternative numeric symbols. |
| %OS | The seconds using the locale?s alternative numeric symbols. |
| %OU | The week number of the year (Sunday as the first day of the week) using the locale?s alternative numeric symbols. |
| %Ow | The number of the weekday (Sunday=0) using the locale?s alternative numeric symbols. |
| %OW | The week number of the year (Monday as the first day of the week) using the locale?s alternative numeric symbols. |
| %Oy | The year (offset from %C) using the locale?s alternative numeric symbols. |

- file

A file to monitor, this option might be set several time if you want to monitor multiple files with this format.

- udp-server

Create an UDP server that is able to handle this format, and which listen to the specified address.

**Ruleset Tuning** In order to get the best performance out of Prelude-LML, you need to tune it post-install. The most important thing to realize about the default Prelude-LML rulesets is that there are a wide range of devices supported, most of which are probably not even present in your environment. Each of these device rulesets that you leave turned on sap CPU cycles from the LML parser.

The easiest way to ensure that LML runs efficiently is to turn off those rulesets you don't need. There are two places to do this:

- In pcre.rules, comment out the rulesets that don't apply.
- In single.rules, comment out the stand-alone rules that don't apply.

The largest performance gain you'll see will be from pruning single.rules. Each of the rulesets in pcre.rules has a required regex to allow events to be processed by them. single.rules rules, on the other hand, are individually evaluated for every incoming log line, and those regexes are larger and more complex than the pre-ruleset regexes on each of the other rulesets.

Another thing you can do to speed up LML is to study and understand your per-device event rates and rearrange the ordering of the rulesets in pcre.rules to give preference to high event-rate devices. For instance, in Linux-heavy environments, you might places the rulesets that match the regex=kernel and regex=sshd pre-ruleset regex at the top of your ruleset list. In an environment where you were mostly monitoring PIX firewalls, you'd put the regex=%PIX line at the top. This ensures that the high-rate events will hit a match and be removed from the parsing queue more quickly, resulting in better overall performance.

**Using more than one Pcre (modifying prelude-lml.conf and plugins.rules)** In some case you will try to split your logfile and match differents rules on the multiple log files. In order to do that you must change the prelude-lml.conf and plugins.rules. By default if you do not specify a Pcre name, there will be only one Pcre named default.

Example :

- You have two syslog files "first.log" and "second.log" and you have three rules files "1.rules" "2.rules" "3.rules"
- You need to create two different Pcre instance :

plugins.rules original :

```
# filename plugin-name-list pcre-option regex #
  *        Debug            -           .*
  *        Pcre             -           .*
```

plugins.rules new file :

```
# filename            plugin-name-list pcre-option regex
  /var/log/first.log  Pcre[first]      -           .*
  /var/log/second.log Pcre[second]     -           .*
```

And the new prelude-lml.conf :

```
include = /usr/local/etc/prelude/default/idmef-client.conf
#next you specify the files that you will check (if you do not put
#files in prelude-lml.conf nothing will be check)
#syslog time-format = "%b %d %H:%M:%S"
prefix-regex= "^(?P<timestamp>.{15}) (?P<hostname>\S+)
(?:(?<process>\S+?)(?:\[(?P<pid>[0-9]+)\])?: )?"
file= /var/log/first.log
file= /var/log/second.log
[Pcre=first]
ruleset= /usr/local/etc/prelude-lml/ruleser/1.rules
ruleset= /usr/local/etc/prelude-lml/ruleser/2.rules
[Pcre=second]
ruleset= /usr/local/etc/prelude-lml/ruleser/2.rules
ruleset= /usr/local/etc/prelude-lml/ruleser/3.rules
```

And when you lauch prelude-lml you should see :

```
Subscribing plugin pcre[first]
Monitoring /var/log/first.log
Subscribing plugin pcre[second]
Monitoring /var/log/second.log
```

**Filtering specific events** In order to filter certain events, you need to create a rule dedicated to matching the event you wish to filter out. As an example, if you want to skip all PAM events concerning a successful login by alex, triggered by the following log entry:

```
Apr 10 16:40:05 bigamd sshd(pam_unix)[10566]:
session opened for user alex by (uid=0)
```

You could add the following rule before the one triggering an alert:

```
regex=session opened for user alex by (\S*)\(uid=(\d*)\); last
```

The last keyword tell the engine that further processing should stop if this rule is matched. Please note that rules are evaluated from top to bottom, so this must be inserted before the rule that actually match.

**Creating and contributing rules** Rulesets that you contribute to the Prelude-LML maintainer should follow these guidelines:

- Avoid using .+ or .* in regex entries unless actually neccessary. Doing so will make your rule CPU-costly to implement.
- Avoid capturing variables which you don't use. This causes unneccessary memory consumption.
- At a minimum, include regex, classification().name, assessement.impact.severity, assessment.impact.type, impact.description.
- If it's correct for this application, use the last keyword.
- Put only a single field on each line of your rules.
- Include a sample log entry with each rule.
- Gather as many pieces of data, and fill as many IDMEF fields as possible from the log entry.
- If a similar rule exists in another ruleset (same function, different software), use the classification().name from the other rule.
- Use only the actual log message, none of the syslog headers (this generally includes timestamp, originating node, originating process, and pid).
- Submit new rulesets to the prelude-devel mailing list for consideration.

Below you will find a listing of most of the IDMEF fields Prelude-LML accepts. Where you see an item listed with (), it means that item is indexed. Indexing starts at 0, so, for example, an event with multiple targets would have the first target listed as target(0), followed by whatever IDMEF fields you use. See the existing rulesets for examples.

- regex: A PCRE regex that should be matched to trigger the alert.
- classification.text: The name of the alert, from one of the origins listed below.
- classification.reference().origin: The type of reference. The permitted values are: unknown, vendor-specific, user-specific, bugtraqid, cve, osvdb
- classification.reference().name: A exactly one string containing the name of the reference from the source. If classification.reference().name=bugtraqid This could be a bugtraqid

- classification.reference().meaning: A brief manager (or the human operator of the manager) description of the alert
- classification.reference().url: A URL at which the manager (or the human operator of the manager) can find additional information about the alert. The document pointed to by the URL may include an in-depth description of the attack, appropriate countermeasures, or other information deemed relevant by the vendor.
- assessment.impact.severity: An estimate of the relative severity of the event. Possible values are: info, low, medium, high.
- assessment.impact.completion: An indication of whether the analyzer believes the attempt that the event describes was successful or not. The permitted values are: failed, succeeded.
- assessment.impact.type: The type of attempt represented by this event, in relatively broad categories. The permitted values are: admin, dos, file, recon, user, other.
- assessment.impact.description: May contain a textual description of the impact, if the analyzer is able to provide additional details.
- source().node.address().address, target().node.address().address:

  Address that has been attacked/Address that issued the attack.

  There can be more than one.
- source().node.address().category, target().node.address().category:

  The type of address provided.

  Possible values: unknown, atm, e-mail, lotus-notes, mac, sna, vm, ipv4-addr, ipv4-addr-hex, ipv6-addr, ipv6-addr-hex, ipv6-net, ipv6-net-mask
- source().node.address().vlan_name, target().node.address().vlan_name:

  The name of the Virtual LAN to which the address belongs.
- source().node.address().vlan_num, target().node.address().vlan_num:

  The number of the Virtual LAN to which the address belongs.
- source().node.name, target().node.name:

  The name of the equipment. This information MUST be provided if no Address information is given.
- source().node.category, target().node.category:

  The domain from which the name information was obtained.

  Possible values are: unknown, ads, afs, coda, dfs, dns, hosts, kerberos, nds, nis, nisplus, nt, wfw

- source().node.location, target().node.location:

  The location of the equipment.

- source().spoofed, target().decoy:

  An indication of wheter the source/target is a decoy.

  The permitted values are: unknown, yes, no.

- source().interface, target().interface:

  May be used by a network-based analyzer with multiple interfaces to indicate which interfaces this source/target was seen on.

- source().service.name, target().service.name:

  The name of the service. Whenever possible, the name from the IANA list of well-known ports SHOULD be used.

- source().service.port, target().service.port:

  The port number being used.

- source().service.protocol, target().service.protocol:

  The protocol being used.

- source().service.portlist, target().service.portlist:

  A list of port numbers beeing used.

- source().user.category, target().user.category:

  The type of user represented (unknown, application, os-device).

- source().user.user_id(). target().user.user_id().

  Create a new UserId? inside an User object (that may contain several UserId?).

- source().user.user_id().type, target().user.user_id().type:

  The type of user information represented (current-user, original-user, target-user, user-privs, current-group, group-privs, other-privs).

- source().user.user_id().name, target().user.user_id().name:

  A user or group name.

- source().user.user_id().number, target().user.user_id().number:

  A user or group number.

- source().process.name, target().process.name:

  A process name

- source().process.pid, target().process.pid:

  A process number

## 4.4 Prelude-Import

Prelude-Import is a commercial extension available from PreludeIDS Technologies. Please check the Corporate Modules page for more information.

Prelude-Import is a tool whose purpose is to import data from applications that report events in a specific format. It can also be used to emit alert from a security shell script.

As of now, three different alerts format are supported:

- IDMEF XML: Import IDMEF-XML and convert it to the native Prelude-IDMEF format.
- Nessus XML: Import Nessus vulnerability scan XML report.
- IDMEF Object: A Prelude specific IDMEF format, very handy for textual representation.

**Importation options**

- dry-run - Print the result without sending the data.
- verbose - Print information regarding what is done.
- format - Force the input to be interpreted using the specified format.
- text-output - Dump the imported events to the specified file.

Please check prelude-import –help output for more options.

**Importing data** Prelude-Import will automatically probe the type of file you provide it on the command line, and use the appropriate plugin for importing each file. Here is the command to use in order to import a file, or a set of file:

```
prelude-import <file1> <file2> <fileN>
```

You might also specify - for stdin, but it then become mandatory to manually specify the input format using the format command line option.

**IDMEF XML file importation Example** Prelude-Import come with a set of test file. Here we're going to import idmef-example-12.xml, which contain an heartbeat. Since the -v (verbose) argument is provided, PreludeImport will print information on each imported IDMEF attribute. The generated event won't be sent since the dry-run option was specified.

```
$ prelude-import -v --dry-run tests/idmef-xml/idmef-example-12.xml
Using 'idmef-xml' to handle 'tests/idmef-xml/idmef-example-12.xml':
  Created path heartbeat.messageid=abc123456789
  Created path   heartbeat.analyzer(0).analyzerid=hq-dmz-analyzer01
```

```
  Created path heartbeat.analyzer(0).node.category=dns
  Created path heartbeat.analyzer(0).node.location=Headquarters DMZ
Network
  Created path heartbeat.analyzer(0).node.name=analyzer01.example.com
  Created path heartbeat.create_time=0xbc722ebe.0x00000000
  Created path heartbeat.additional_data(0).type=real
  Created path heartbeat.additional_data(0).meaning=%memused
  Created path heartbeat.additional_data(0).data=62.5
  Created path heartbeat.additional_data(1).type=real
  Created path heartbeat.additional_data(1).meaning=%diskused
  Created path heartbeat.additional_data(1).data=87.1
```

**Importing Nessus Vulnerability assessment** Using PreludeImport, you can also generate events for every vulnerability reported by the Nessus vulnerability scanner. Nessus data might be used to warn the analyst about a new machine property (new port opened/closed), and to regularly check and issue alert when new vulnerability are found by Nessus.

```
yoann@arwen ~/dev/prelude/svk/branches/private/prelude-import $
~/dev/prelude/bin/bin/prelude-import -v --dry-run
tests/nessus-xml/nessus.xml
Using 'nessus-xml' to handle 'tests/nessus-xml/nessus.xml':
Created path alert.analyzer(0).version
Created path alert.analyzer(0).node.name
Created path alert.analyzer(0).ostype
Created path alert.analyzer(0).osversion
Created path alert.detect_time
Created path alert.source(0).node.address(0).address
Created path alert.source(0).user.category
Created path alert.source(0).user.user_id(0).name
Created path alert.source(0).user.user_id(0).type
Created path alert.target(0).node.name
Created path alert.target(0).node.address(0).address
Created path alert.target(0).service.port
Created path alert.target(0).service.name
Created path alert.source(0).spoofed
Created path alert.assessment.confidence.rating
Created path alert.assessment.impact.completion
Created path alert.classification.text Sending IDMEF message.
Created path alert.target(0).service.port
Created path alert.target(0).service.name
Created path alert.source(0).spoofed
Created path alert.assessment.confidence.rating
Created path alert.assessment.impact.completion
Created path alert.classification.text
```

```
Sending IDMEF message.
Created path alert.target(0).service.port
Created path alert.target(0).service.name
Created path alert.source(0).spoofed
Created path alert.assessment.confidence.rating
Created path alert.assessment.impact.completion
Created path alert.classification.text
Sending IDMEF message.
Created path alert.target(0).service.port
Created path alert.target(0).service.name
Created path alert.source(0).spoofed
Created path alert.assessment.confidence.rating
Created path alert.assessment.impact.completion
Created path alert.classification.text
Sending IDMEF message.
[snip]
```

**Generating events from the shell** Using the IDMEF-object Prelude internal format, it is trivial to generate events from a security shell script. Here is an example:

```
$ echo "
> alert.messageid = blah
> alert.classification.text=This is an event generated from the shell
> " | ~/dev/prelude/bin/bin/prelude-import -v --dry-run --format
idmef-object -
Using 'idmef-object' to handle '-':
  Created object alert.messageid=blah
  Created object alert.classification.text=This is an event generated
from the shell
```

## 4.5 Prewikka

Once you have successfully installed Prewikka. You will need to do some initial configuration to get it going and customized as you want it.

You will first want to login as admin and set your password to something other than the default.

- Users permissions
- Alert listing
- Agents listing
- Browsable statistics

## 4.6 High Availability Prelude Central Services

This is an example configuration utilizing two machines to provide a high availability pair for the central Prelude services, which include: PreludeManager, PreludeCorrelator, Prewikka, and Mysql.

Heartbeat will control the failing over of hard failures, such as a machine going down or total loss of connectivity. Both servers will maintain an up-to-date set of databases, and either can take over at a moments notice as the primary. You will need to use some form of monitoring, such as Nagios to handle service failures - thus alerting you when a manual failover is required (such as if a particular service dies, but the hardware remains operational and reachable).

Things that are assumed:

- You will need at least two ethernet interfaces per server (and two servers).
- Assumes you have eth0 configured with IP/hostnames on both boxes. This example uses preludecentral_1 and preludecentral_2, and a VIP (virtual IP) with associated hostname, such as preludecentral. Make sure you also add the VIP/hostname to DNS/host/etc., as this is what you expose to your clients/relays/agents/etc.
- This document assumes high availability for fault tolerance, not for performance. Although you could stagger which services are offered where, etc.

**MySQL Multi-Master Replication Configuration** ***You must have installed MySQL v5.x or above. Some features to avoid collisions in multi-master replication are only available in MySQL v5.x***

- Setup secondary ethernet interfaces
  - Setup eth1 on each server to be an unused private network for use by your HA pair.
  - Connect with a crossover cable between them.
- Make the following additions on both servers in /etc/my.cnf, under [mysqld] section:

```
wait_timeout=259200
interactive_timeout=259200
max_connections=200
log-bin=<$HOSTNAME>-mysql-bin #change hostname to be each machines hostname
server-id={1,2} #set one server for 1 and the other for 2
auto_increment_increment=2
auto_increment_offset={1,2} #set one server for 1 and the other for 2
# Timeouts and max connections have been increased to handle a client
#disconnect issue.
# Various other settings can be adjusted to the specs of your machine,
#such as buffer sizes, log sizes, etc.
```

- Change datadir in /etc/my.cnf to be on a partition of its own of acceptable size, edit /etc/fstab, and add "async,noatime" as options for the partition used for the MySQL database directory.
- Run mysql on each server:
  - Enter at prompt: GRANT REPLICATION SLAVE on *.* TO some_replication_user IDENTIFIED by 'putreplicationpasswordhere';
- Add to the end of the mysql binary line in the start section of /etc/init.d/mysqld
  - –relay-log=<$HOSTNAME>-relay-bin #make sure to specify the correct hostname
- From each server's mysql command-line: show master status;
- From each server's mysql command-line: change master to master_host='<other hosts eth1 private ip address>', master_user='<username setup earlier for replication>', master_password='<password used earlier for replication>', master_log_file='<output of 1st column from above step on other server', master_log_pos=<output of 2nd column from above step on other server>;
- Restart mysqld on each of the servers
- Configure a root mysql password from the mysql command-line (only needs to be done on one server):
  - SET password for root@localhost=password('putmysqlrootpasswordhere');
  - FLUSH PRIVILEGES;

**Prelude Central Components**

- You must install libprelude, libprelude-db, PreludeManager, PreludeCorrelator and Prewikka on both machines.
- All of the central Prelude services should have the same configuration files on both servers (ie. prelude-manager, prelude-correlator, apache, etc.)
- When importing the prelude and prewikka schemas to their respective databases, it only needs to be done on one of the servers, it will be replicated to the other automatically.
- Copy over profiles (if you have a Prelude Central already setup) or register for all profiles needed: correlator, manager, prewikka. Do this on one server, then copy those profiles/directories under /usr/local/etc/prelude/profile/ of the other machine in the HA pair.
- Edit /usr/local/etc/prelude/default/global.conf, make 'Node Name' the name of the VIPs hostname on both. Copy over to other HA server.

- Your web server and MySQL should be set to start automatically on boot. Prelude Manager and Correlator should NOT be set to start automatically on boot, as Heartbeat will handle these two services.

**Setting up the Heartbeat Pair**

- Heartbeat Configuration
  - Install heartbeat, and all packages necessary such as stonih, pils, etc. that are required by your package manager.
  - Configure heartbeat to start on boot.
  - Create /etc/ha.d/ha.cf debugfile /var/log/ha-debug logfile /var/log/ha-log logfacility local0 keepalive 2 auto_failback off deadtime 30 bcast eth1 eth0 node preludecentral_1 node preludecentral_2
  - Create /etc/ha.d/haresources preludecentral_1 IPaddr::192.168.1.25 prelude-manager prelude-correlator
    - Use your primary hostname, assumes 192.168.1.25 is your VIP, and that you called your init scripts prelude-manager and prelude-correlator
  - Create /etc/ha.d/authkeys; make sure to chmod 600 it auth 1 1 sha1 some_password
- Set the Heartbeat service to start on boot automatically.

**Service Monitoring**

- Setup SNMP monitoring with something such as Nagios
  - For the VIP hostname/IP
    - Ping
    - Apache
    - Prelude Manager
    - Prelude Correlator
    - Prewikka Ctl
    - TCP Port 4690
  - For each hostname/IP of the pair
    - Ping
    - Diskspace
    - MySQL
    - Heartbeat
- Set the snmpd service to start on boot automatically.

**Clients / Relays / Agents**

- Make sure all clients, agents, relays, etc. connect to your VIP hostname/IP.

# 5 Troubleshooting

## 5.1 Libprelude

- I get "TLS server certificate is NOT trusted"

Answer: You most probably used the "sensor analyzer profile" instead of the "receiving analyzer profile" on the registration-server command line. You can find more information about your problem here:
http://thread.gmane.org/gmane.comp.security.ids.prelude.user/1926/focus=1927
Typical case: When you register a sensor, don't register it to the sensor itself!

```
sudo prelude-admin register NAME ...
sudo prelude-admin registration-server NAME
```

But register it to your prelude-manager, example:

```
sudo prelude-admin register NAME ...
sudo prelude-admin registration-server prelude-manager
```

- How can I validate a IDMEF message ?

Answer: First run prelude-manager with XMLmod and activate the DTD validation option. Then, generate an exhaustive list of alerts supported by the sensor and correct any error. Open a bug on the ticket system, and ask a review of the alert dumped by the debug plugin.

## 5.2 Libpreludedb

libpreludedb has a problem with prelude-manager. in start time of run of prelude-manager you see this message "error initializing libpreludedb" in compiling time you should enable perl , mysql , python option such as this: make WITH_PERL=1 WITH_PYTHON=1 WITH_MYSQL=1 and for libprelude you should compile with: make WITH_PERL=1 WITH_PYTHON=1

## 5.3 PreludeManager?

...

## 5.4 Prelude-LML

- I get an error concerning auth.log when starting prelude-lml?

Answer: This is due to some systems using authlog or secure for the same purpose. To remedy this issue just edit prelude-lml.conf to point to the correct auth log depending on your system.

- I get an error "End from FAM server connection" after some time, and no more alert are sent?

Answer: This is due to FAM, the best way to get prelude-lml working correctly is by recompiling the prelude-lml without FAM (./configure –disable-fam). FAM just help getting files checked when there are modifications, without FAM, prelude-lml will check files modification every second.

- Prelude-LML warn that it could not match prefix against log entry or that there is no appropriate format defined for log entry

Answer: This warning mean that wiki:PreludeLML was unable to find a matching format (defined in the prelude-lml configuration file) for the input log.

The log entry will still be processed by the signature engine, but wiki:PreludeLML won't have access to the following information:

- The log entry timestamp (Bound to the DetectTime? information of an IDMEF Alert).
- The log entry hostname (Bound to the Target node information in an IDMEF Alert).
- The log entry process (Bound to the Target process name in an IDMEF Alert).
- The log entry pid (Bound to the Target process pid in an IDMEF Alert).

In order to fix this problem, you need to configure an appropriate format section in the configuration file that instruct PreludeLML how to parse a given log format.

## 5.5 Prewikka

- Prelude Database shema version too old error message?

Answer: You must change the mysql shema using a pre-made script :

```
>mysql -u prelude prelude -p <
/usr/share/libpreludedb/classic/mysql-update-14-1.sql syntaxe : mysql
-u <login> <password> -p <
/usr/share/libpreludedb/classic/mysql-update-14-1.sql
```

NOTE: Sometimes the path would be /usr/local/share/. . .

- I have issues with Prelude and apache configuration since I want Prewikka in a sub-directory.

Answer: You are already using your web server for something else and you want to have an access using http://www.mycompany.xx/prelude :

```
Alias /prelude/prewikka /usr/share/prewikka/htdocs/ ScriptAlias
/prelude/ /usr/share/prewikka/cgi-bin/prewikka.cgi
<Directory "/usr/share/prewikka/htdocs/">
  AllowOverride None
  Options ExecCGI
  <IfModule mod_mime.c>
    AddHandler cgi-script .cgi
  </IfModule>
  Order allow,deny
  Allow from all
</Directory>
```

- Abnormal Offline Agents and MySQL table 'Prelude_Analyzer' is full

Answer: If checking the mysql log you find that :

```
MySQL Query error: The table 'Prelude_Analyzer' is full
```

Then you may need to alter a setting in my.cnf (maybe back it up first) So find the line

```
innodb_data_file_path = ibdata1:10M:autoextend:max:128M
```

and try changing it to read:

```
innodb_data_file_path = ibdata1:10M:autoextend
```

After a restart of mysql, prelude-manager, lml etc. hopefully this will bring your prewikka back online.

(With thanks to Yoann on prelude-user mailing list)

# 6 Programming with prelude

Please see https://trac.prelude-ids.org/wiki/PreludeHandbook for more details.

- libprelude API Documentation
- libpreludedb API Documentation
- Building your first sensor

  - In C
  - In python
  - In perl
- Writing plugins for prelude manager
- Debugging your sensor