Prelude User Manual

[[PreludeForeword|Foreword]]

1. **Introduction**
   1. [[PreludeGlossary|Glossary]]
   2. [[PreludeComponents|Prelude Components]]
   3. [[PreludeArchitecture|Prelude Architecture]]
   4. [[PreludeStandards|Prelude Standards]]
   5. [[PreludeCompatibility|Prelude Compatibility]]
2. **Installation**
   1. [[InstallingPreludeRequirement|Installation Requirements]]
   2. [[InstallingPrelude|Installation from sources]]
      1. [[InstallingPreludeLibrary|Libprelude]]
      2. [[InstallingPreludeDbLibrary|LibpreludeDB]]
      3. [[InstallingPreludeManager|Prelude Manager]]
      4. [[InstallingPreludeCorrelator|Prelude Correlator]]
      5. [[InstallingPreludeLml|Prelude LML]]
      6. [[InstallingPreludePrewikka|Prewikka]]
   3. [[InstallingPackage|Installation from packages]]
   4. [[AgentsInstallation|Agents Installation]]
      1. [[InstallingAgentRegistration|Agents Registration]]
      2. [[InstallingAgentThirdparty|3rd Party Agents Installation]]
3. **Configuration**
   1. [[ConfigurationGeneral|General Configuration]]
   2. [[PreludeComponentsConfiguration|Prelude Components Configuration]]
      1. [[PreludeManager|Prelude-Manager]]
      2. [[PreludeCorrelator|Prelude-Correlator]]
      3. [[PreludeLml|Prelude-LML]]
      4. [[PreludeImport|Prelude-Import]]
   3. [[HowtoHACentralServices|Howto Configure High Availability Central Services]]
4. **Optimisation**
   1. [[DatabaseOptimisation|Database Optimisation]]
5. **User Manuals**
   1. [[ManualPrewikka|Prewikka Manual]]
   2. [[ManualPreludeAdmin|Prelude-Admin Manual]]
   3. [[ManualPreludeDbAdmin|PreludeDB-Admin Manual]]
6. **Development**
   1. [[DevelopmentGuidelines|Development guidelines]]
   2. [[SourceOrganization|Source organization]]
   3. [[PrewikkaApps|Prewikka Apps]]
   4. [[DevelAgentQuickly|Prelude Agent]]
   5. [[PreludeAgentContribution|Prelude Agent Contribution program]]
   6. [[libpreludeAPI|Libprelude API]]
   7. [[libpreludedbAPI|Libpreludedb API]]
   8. [[ContinuousIntegration|Prelude CI]]

General Configuration

All Prelude agents 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.

- **analyzer-name**: Name for the analyzer (By default, this is set to the profile name used by the sensor).
- **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 options:

- **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).

Howto Configure 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|Prelude-Manager]] [[PreludeCorrelator|Prelude-Correlator]], [[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

1. 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.

2. 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.
```

3. 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.
4. Run mysql on each server:
   - Enter at prompt: GRANT REPLICATION SLAVE on *.* TO some_replication_user IDENTIFIED by 'putreplicationpasswordhere';
5. 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

6. From each server's mysql command-line: show master status;

7. 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>;

8. Restart mysqld on each of the servers
9. 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

1. You must install [[InstallingPreludeLibrary|Libprelude]] [[InstallingPreludeDbLibrary|LibpreludeDB]] [[InstallingPreludeManager|Prelude-Manager]] [[InstallingPreludeCorrelator|Prelude-Correlator]] and [[InstallingPreludePrewikka|Prewikka]] on both machines.

2. All of the central Prelude services should have the same configuration files on both servers (ie. prelude-manager,

prelude-correlator, apache, etc.)

3. 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.

4. 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.

5. 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.

6. 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

1. 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
```

2. Set the Heartbeat service to start on boot automatically.

Service Monitoring

1. 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

2. 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.

Packages Installation

Prelude is available through a number of OS Packages:

- [[InstallingPackageRHEL|RHEL/CentOS]]
- [[InstallingPackageFedora|Fedora]]
- [[InstallingPackageDebian|Debian]]
- [[InstallingPackageMandriva|Mandriva]]
- [[InstallingPackageSuse|SuSE]]
- [[InstallingPackageUbuntu|Ubuntu]]
- [[InstallingPackageGentoo|Gentoo]]
- [[InstallingPackageFreebsd|FreeBSD]]
- [[InstallingPackageMacosx|Mac OSX]]
- [[InstallingPackageNetbsd|NetBSD]]
- [[InstallingPackageOpenbsd|OpenBSD]]
- [[InstallingPackageSolaris|Solaris]]
- [[InstallingPackageSourceMage|Source Mage GNULinux]]

If you need more information regarding a Prelude package, contact the package maintainer:
See the Prelude Community Page for email contact information.

Prelude Components Installation

1. Download
   - Prelude SIEM Official website: Download Page
2. Get the Framework Packages:
   - *[[InstallingPreludeLibrary|Libprelude]]*: The main library, used in every program using the Prelude architecture.
   - *[[InstallingPreludeDbLibrary|Libpreludedb]]*: The library which retrieves IDMEF information from the database.
   - *[[InstallingPreludeManager|Prelude-Manager]]*: The program sensors are connected to, and delivering alerts to.
3. Get the Interface:
   - *[[InstallingPreludePrewikka|Prewikka]]*: The official frontend. It collects information from the database and represents it in tables and graphs.
4. Get the Correlation Engine:
   - *[[InstallingPreludeCorrelator|Prelude Correlator]]*: The Prelude correlation engine. It allows conducting multistream correlations thanks to a powerful programming language for writing correlation rules.
5. Get the Sensors:
   - *[[InstallingPreludeLml|Prelude-LML]]*: A sensor to monitor logfiles using predefined rulesets.
   - *[[InstallingAgentThirdpartyAuditd|Auditd]]*: The Linux Audit Daemon.
   - *[[InstallingAgentThirdpartyNepenthes|Nepenthes]]*: A versatile tool to collect malware.
   - *[[InstallingAgentThirdpartyNufw|NuFW]]*: An identity access management solution at the network level.
   - *[[InstallingAgentThirdpartyOssec|OSSEC]]*: An Open Source Host-based Intrusion Detection System.
   - *[[InstallingAgentThirdpartyLinuxpam|Linux-PAM]]*: Linux Pluggable Authentication Modules.
   - *[[InstallingAgentThirdpartySamhain|Samhain]]*: A file integrity checker.
   - *[[InstallingAgentThirdpartySancp|SanCP]]*: A network traffic statistical information collector
   - *[[InstallingAgentThirdpartySnort|Snort]]*: The Defacto Standard Open Source IDS.

Requirements:

- *Libprelude* is required on every systems.
- *Libpreludedb* and *prelude-manager* are required on a prelude-manager system.
- *Prelude-LML* and other sensors have only one requirement, *libprelude*.
- *Prewikka* requires Python bindings of both *Libprelude* and *Libpreludedb* libraries.

This chapter only talks about how to install all the modules on the same machine. If you wish to seperate components over multiple hosts, **remember to always install libprelude on every host**.

If you want to see how to configure a sensor to report to one or several Prelude managers, please refer to the *[[InstallingAgentRegistration|Agent Registration]]* page of this Manual.

Targeted platforms

Targeted plateforms are :

- CentOS 6 and 7, 32 and 64 bits
- Debian 7 and 8, 32 and 64 bits

But it should work on every unix ! Also, it should work on Windows.

Installing Prelude-Correlator

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

Table of Contents

Install Dependencies

- Ensure that you have the Prelude Library installed. Instructions on installing libprelude can be found on the [[InstallingPreludeLibrary|Prelude Library Installation Page]]
- Prelude-Correlator requires Python 2.4 or later.

Get the sources

Download the latest Prelude-Correlator

Install

You might install Prelude-Correlator as root:

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

Or as an user:

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

This will install files into the directories that were specified in step 1. Make sure that you have appropriate permissions to write into that area. Normally you need to do this step as root. Alternatively, you could create the target directories in advance and arrange for appropriate permissions to be granted.

More information on configuration and writing correlation rules can be found on the [[PreludeCorrelator]] page.

Installing the [[PreludeDB]] Library

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

Table of Contents

Install Dependencies

Ensure that you have libprelude installed. Instructions on installing libprelude can be found on the [[InstallingPreludeLibrary|Prelude Library Installation Page]].
Additionally, install any of the database software you would like the [[PreludeDB]] Library to support: [[MySQL]], [[PostgreSQL]], SQLite3.

Get the sources

Download the latest PreludeDB library

Configuration

The first step of the installation procedure is to configure the source tree for your system and choose the options you would like. This is done by running the configure script. For a default installation simply enter

```
./configure
```

This script will run a number of tests to guess values for various system dependent variables and detect some quirks of your operating system, and finally will create several files in the build tree to record what it found. (You can also run configure in a directory outside the source tree if you want to keep the build directory separate.)

You can customize the build and installation process by supplying one or more of the following command line options to configure:

Installation directories

- --*prefix*=_PREFIX_
  Install all files under the directory PREFIX instead of /usr/local. The actual files will be installed into various subdirectories; no files will ever be installed directly into the PREFIX directory.
If you have special needs, you can also customize the individual subdirectories with the following options.

- --*exec-prefix*=_EXEC-PREFIX_
You can install architecture-dependent files under a different prefix, EXEC-PREFIX, than what PREFIX was set to. This can be useful to share architecture-independent files between hosts. If you omit this, then EXEC-PREFIX is set equal to PREFIX and both architecture-dependent and independent files will be installed under the same tree, which is probably what you want.

- --*bindir*=_DIRECTORY_
Specifies the directory for executable programs. The default is EXEC-PREFIX/bin, which normally means /usr/local/bin.

- --*datadir*=_DIRECTORY_
Sets the directory for read-only data files used by the installed programs. The default is PREFIX/share.

- --*sysconfdir*=_DIRECTORY_
The directory for various configuration files, PREFIX/etc by default.

- --*libdir*=_DIRECTORY_
The location to install libraries and dynamically loadable modules. The default is EXEC-PREFIX/lib.

- --*includedir*=_DIRECTORY_
The directory for installing C and C++ header files. The default is PREFIX/include.

- --*mandir*=_DIRECTORY_
The man pages that come with the [[PreludeDB]] Library will be installed under this directory, in their respective man_N_ subdirectories. The default is PREFIX/man.

Path to dependencies

- --*with-libprelude-prefix*=_PFX_
The [[PreludeDB]] Library require libprelude in order to build. Using this option you can tell the configure script in which prefix the Prelude Library has been installed.

- --*with-mysql-prefix*=_PFX_
The [[PreludeDB]] Library can interoperate with [[MySQL]] database. Using this option you can tell the configure script in which prefix [[MySQL]] has been installed.

- --*with-pgsql-prefix*=_PATH_
The [[PreludeDB]] Library can interoperate with [[PostgreSQL]] database. Using this option you can tell the configure script in which prefix [[PostgreSQL]] has been installed.

- --*with-sqlite3-prefix*=_PATH_
The [[PreludeDB]] Library can interoperate with SQLite database. Using this option you can tell the configure script in which prefix SQLite has been installed.

- --*with-perl*=_PATH_

The *Perl* interpreter is necessary to enable support for *Perl* language bindings and is normally automatically detected. Using this option you can specify a specific path to *Perl* or disable its support using --*without-perl*.

- --*with-python*=_PATH_

The *Python* interpreter is necessary to enable support for *Python* language bindings and is normally automatically detected. Using this option you can specify a specific path to *Pythonl* or disable its support using --*without-python*.

Optional Features

- --*enable-gtk-doc*

Generate code documentation (note that distribution come with documentation generated).

Once finished, the *configure* script will dump a summary of the enabled features:

```
*** Dumping configuration ***
   - Generate documentation      : no
   - Enable [[MySQL]] plugin         : yes
   - Enable [[PostgreSQL]] plugin    : yes
   - Enable SQLite3 plugin       : yes
   - Perl binding                : yes
   - Python binding              : yes
```

If no languages bindings are activated, you will be unable to run agent using [[LibpreludeDB]] from the specified language.

Note: Prewikka require Python bindings to be available.

Note: You should always have one of the [[MySQL]], [[PostgreSQL]], or SQLite3 plugin enabled.

Build

To start the build, type

```
make
```

The build will take a few minutes depending on your hardware.

Regression Tests

If you want to test the newly built library before you install it, you can run the regression tests at this point. The regression tests are a test suite to verify that libpreludedb runs on your machine in the way the developers expected it to. Type

```
make check
```

Install

To install Libpreludedb enter

```
make install
```

This will install files into the directories that were specified in step 1. Make sure that you have appropriate permissions to write into that area. Normally you need to do this step as root. Alternatively, you could create the target directories in advance and arrange for appropriate permissions to be granted.

You can use *make install-strip* instead of *make* install to strip the executable files and libraries as they are installed. This will save

some space. If you built with debugging support, stripping will effectively remove the debugging support, so it should only be done if debugging is no longer needed. install-strip tries to do a reasonable job saving space, but it does not have perfect knowledge of how to strip every unneeded byte from an executable file, so if you want to save all the disk space you possibly can, you will have to do manual work.

The standard installation provides all the header files needed for Prelude application development, such as custom functions or data types written in C.

Post-Installation Setup

On systems that have shared libraries (which most systems do) you need to tell your system how to find the newly installed shared libraries. The systems on which this is not necessary include BSD/OS, [[FreeBSD]], HP-UX, IRIX, Linux, [[NetBSD]], [[OpenBSD]], Tru64 UNIX (formerly Digital UNIX), and Solaris.

The following step are necessary if you get the above message while running a program using libprelude:

```
preludedb-admin: error in loading shared libraries
libpreludedb.so.2.1: cannot open shared object file: No such file or directory
```

If you are on BSD/OS, Linux, or [[SunOS]] 4 and you have root access you can run the following command after installation to enable the run-time linker to find the shared libraries faster. Refer to the manual page of ldconfig for more information:

```
/sbin/ldconfig /usr/local/lib
```

On [[FreeBSD]], [[NetBSD]], and [[OpenBSD]] the command is:

```
/sbin/ldconfig -m /usr/local/lib
```

The method to set the shared library search path varies between platforms, but the most widely usable method is to set the environment variable LD_LIBRARY_PATH like so: In Bourne shells (sh, ksh, bash, zsh):

```
LD_LIBRARY_PATH=/usr/local/lib
export LD_LIBRARY_PATH
```

Or in csh or tcsh:

```
setenv LD_LIBRARY_PATH /usr/local/lib
```

Note: Replace /usr/local/lib with whatever you set --libdir to in step 1. On some systems it might be preferable to set the environment variable LD_RUN_PATH before building. If in doubt, refer to the manual pages of your system (perhaps ld.so or rld).

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. Three databases are currently supported by libpreludedb: [[MySQL]], [[PostgreSQL]] and SQLite:

[[InstallingPreludeDbLibrary#MySQL|MySQL]]
[[InstallingPreludeDbLibrary#PostgreSQL|PostgreSQL]]
[[InstallingPreludeDbLibrary#SQLite3|Sqlite]]

[[MySQL]]

Note : MySQL >= 4.0.18 is required.

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 [[BR]]
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:

```
$ 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 [[BR]]
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:
```

PostGreSQL 9 compatibility

Beginning with PostGreSQL 9, bytea are returned in hex format by default (see http://www.postgresql.org/docs/devel/static/release-9-0.html).
To work with Prelude, the following parameter must be specified in postgresql.conf.

```
bytea_output='escape'
```

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 and tables creation

Once you have installed libpreludedb, simply create the SQLite3 database using the following:

```
$ sqlite3 /path/to/your/sqlite-prelude.db < /usr/share/libpreludedb/classic/sqlite.sql
```

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.

Table of Contents

Install Dependencies

Ensure that you have GNUTLS installed. Instructions on installing gnutls can be found on the [[InstallingPreludeRequirementGnutls|GnuTLS Installation Page]]

Get the sources

Download the latest Prelude library

Configuration

The first step of the installation procedure is to configure the source tree for your system and choose the options you would like. This is done by running the configure script. For a default installation simply enter

```
./configure
```

This script will run a number of tests to guess values for various system dependent variables and detect some quirks of your operating system, and finally will create several files in the build tree to record what it found. (You can also run configure in a directory outside the source tree if you want to keep the build directory separate.)

You can customize the build and installation process by supplying one or more of the following command line options to configure:

Installation directories

- --*prefix*=_PREFIX_
  Install all files under the directory PREFIX instead of /usr/local. The actual files will be installed into various subdirectories; no files will ever be installed directly into the PREFIX directory.
If you have special needs, you can also customize the individual subdirectories with the following options.

- --*exec-prefix*=_EXEC-PREFIX_
You can install architecture-dependent files under a different prefix, EXEC-PREFIX, than what PREFIX was set to. This can be useful to share architecture-independent files between hosts. If you omit this, then EXEC-PREFIX is set equal to PREFIX and both architecture-dependent and independent files will be installed under the same tree, which is probably what you want.

- --*bindir*=_DIRECTORY_
Specifies the directory for executable programs. The default is EXEC-PREFIX/bin, which normally means /usr/local/bin.

- --*datadir*=_DIRECTORY_
Sets the directory for read-only data files used by the installed programs. The default is PREFIX/share.

- --*sysconfdir*=_DIRECTORY_
The directory for various configuration files, PREFIX/etc by default.

- --*libdir*=_DIRECTORY_
The location to install libraries and dynamically loadable modules. The default is EXEC-PREFIX/lib.

- --*includedir*=_DIRECTORY_
The directory for installing C and C++ header files. The default is PREFIX/include.

- --*mandir*=_DIRECTORY_

The man pages that come with the Prelude Library will be installed under this directory, in their respective man_N_ subdirectories. The default is PREFIX/man.

Path to dependencies

- --*with-libgnutls-prefix*=_PFX_

The Prelude Library require [[GnuTLS]] in order to build. Using this option you can tell the configure script in which prefix [[GnuTLS]] has been installed.

- --*with-libgnutls-extra-prefix*=_PFX_

The Prelude Library can make use of libgnutls-extra to provides additional authentication protocols (example: SRP). Using this option you can tell the configure script in which prefix libgnutls-extra has been installed.

- --*with-libgcrypt-prefix*=_PFX_

The Prelude Library require *libgcrypt* in order to build. Using this option you can tell the configure script in which prefix libgcrypt has been installed.

- --*with-perl*=_PATH_

The *Perl* interpreter is necessary to enable support for *Perl* language bindings and is normally automatically detected. Using this option you can specify a specific path to *Perl* or disable its support using --*without-perl*.

- --*with-python*=_PATH_

The *Python* interpreter is necessary to enable support for *Python* language bindings and is normally automatically detected. Using this option you can specify a specific path to *Pythonl* or disable its support using --*without-python*.

- --*with-ruby*=_PATH_

The *Ruby* interpreter is necessary to enable support for *Ruby* language bindings and is normally automatically detected. Using this option you can specify a specific path to *Ruby* or disable its support using --*without-ruby*.

- --*with-lua-config*=_PATH_

The *Lua* interpreter is necessary to enable support for *Lua* language bindings and is normally automatically detected. Using this option you can specify a specific path to *Lua* or disable its support using --*without-lua-config*.

Optional Features

- --*enable-gtk-doc*

Generate code documentation (note that distribution come with documentation generated).

- --*enable-easy-bindings*

Enable support for high level bindings (available for *C++*, *Perl*, *Python*, *Ruby*, *Lua*).

Once finished, the *configure* script will dump a summary of the enabled features:

```
*** Dumping configuration ***
   - Generate documentation   : no
   - Libtool dynamic loader   : System
   - LUA binding              : yes
   - Perl binding             : yes
   - Python binding           : yes
   - Ruby binding             : yes
   - Easy bindings            : yes
```

If no languages bindings are activated, you will be unable to run agent using Libprelude from the specified language.

Note: Prewikka require Python bindings to be available.

Build

To start the build, type

```
make
```

The build will take a few minutes depending on your hardware.

Regression Tests

If you want to test the newly built library before you install it, you can run the regression tests at this point. The regression tests are a test suite to verify that libprelude runs on your machine in the way the developers expected it to. Type

```
make check
```

Install

To install Libprelude enter

```
make install
```

This will install files into the directories that were specified in step 1. Make sure that you have appropriate permissions to write into that area. Normally you need to do this step as root. Alternatively, you could create the target directories in advance and arrange for appropriate permissions to be granted.

You can use *make install-strip* instead of *make* install to strip the executable files and libraries as they are installed. This will save some space. If you built with debugging support, stripping will effectively remove the debugging support, so it should only be done if debugging is no longer needed. install-strip tries to do a reasonable job saving space, but it does not have perfect knowledge of how to strip every unneeded byte from an executable file, so if you want to save all the disk space you possibly can, you will have to do manual work.

The standard installation provides all the header files needed for Prelude application development, such as custom functions or data types written in C.

Post-Installation Setup

On systems that have shared libraries (which most systems do) you need to tell your system how to find the newly installed shared libraries. The systems on which this is not necessary include BSD/OS, [[FreeBSD]], HP-UX, IRIX, Linux, [[NetBSD]], [[OpenBSD]], Tru64 UNIX (formerly Digital UNIX), and Solaris.

The following step are necessary if you get the above message while running a program using libprelude:

```
prelude-admin: error in loading shared libraries
libprelude.so.2.1: cannot open shared object file: No such file or directory
```

If you are on BSD/OS, Linux, or [[SunOS]] 4 and you have root access you can run the following command after installation to enable the run-time linker to find the shared libraries faster. Refer to the manual page of ldconfig for more information:

```
/sbin/ldconfig /usr/local/lib
```

On [[FreeBSD]], [[NetBSD]], and [[OpenBSD]] the command is:

```
/sbin/ldconfig -m /usr/local/lib
```

The method to set the shared library search path varies between platforms, but the most widely usable method is to set the environment variable LD_LIBRARY_PATH like so: In Bourne shells (sh, ksh, bash, zsh):

```
LD_LIBRARY_PATH=/usr/local/lib
export LD_LIBRARY_PATH
```

Or in csh or tcsh:

```
setenv LD_LIBRARY_PATH /usr/local/lib
```

Note: Replace /usr/local/lib with whatever you set --libdir to in step 1. On some systems it might be preferable to set the environment variable LD_RUN_PATH before building. If in doubt, refer to the manual pages of your system (perhaps ld.so or rld).

Installing Prelude-LML

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

Table of Contents

Install Dependencies

Ensure that you have the Prelude Library installed. Instructions on installing libprelude can be found on the [[InstallingPreludeLibrary|PreludeDB Library Installation Page]]

Get the sources

Download the latest Prelude-LML

Configuration

The first step of the installation procedure is to configure the source tree for your system and choose the options you would like. This is done by running the configure script. For a default installation simply enter

```
./configure
```

This script will run a number of tests to guess values for various system dependent variables and detect some quirks of your operating system, and finally will create several files in the build tree to record what it found. (You can also run configure in a directory outside the source tree if you want to keep the build directory separate.)

You can customize the build and installation process by supplying one or more of the following command line options to configure:

Installation directories

- --*prefix*=_PREFIX_
  Install all files under the directory PREFIX instead of /usr/local. The actual files will be installed into various subdirectories; no files will ever be installed directly into the PREFIX directory.
If you have special needs, you can also customize the individual subdirectories with the following options.

- --*exec-prefix*=_EXEC-PREFIX_
You can install architecture-dependent files under a different prefix, EXEC-PREFIX, than what PREFIX was set to. This can be useful to share architecture-independent files between hosts. If you omit this, then EXEC-PREFIX is set equal to PREFIX and both architecture-dependent and independent files will be installed under the same tree, which is probably what you want.

- --*bindir*=_DIRECTORY_
Specifies the directory for executable programs. The default is EXEC-PREFIX/bin, which normally means /usr/local/bin.

- --*datadir*=_DIRECTORY_

Sets the directory for read-only data files used by the installed programs. The default is PREFIX/share.

- --*sysconfdir*=_DIRECTORY_

The directory for various configuration files, PREFIX/etc by default.

- --*libdir*=_DIRECTORY_

The location to install libraries and dynamically loadable modules. The default is EXEC-PREFIX/lib.

- --*includedir*=_DIRECTORY_

The directory for installing C and C++ header files. The default is PREFIX/include.

- --*mandir*=_DIRECTORY_

The man pages that come with Prelude-LML will be installed under this directory, in their respective man_N_ subdirectories. The default is PREFIX/man.

Path to dependencies

- --*with-libprelude-prefix*=_PFX_

Prelude-LML require the Prelude Library in order to build. Using this option you can tell the configure script in which prefix libprelude has been installed.

- --*with-fam-prefix*=_PFX_

Prelude-LML can make use of FAM (or Gamin) to get real time notification for files changes. Using this option you can tell the configure script in which prefix FAM has been installed.

Once finished, the *configure* script will dump a summary of the enabled features:

```
*** Dumping configuration ***
   - Enable FAM support               : yes
   - Enable unsupported rulesets:     : yes
```

Build

To start the build, type

```
make
```

The build will take a few minutes depending on your hardware.

Regression Tests

If you want to test the newly built program before you install it, you can run the regression tests at this point. The regression tests are a test suite to verify that Prelude-LML runs on your machine in the way the developers expected it to. Type

```
make check
```

Install

To install Prelude-LML enter

```
make install
```

This will install files into the directories that were specified in step 1. Make sure that you have appropriate permissions to write into

that area. Normally you need to do this step as root. Alternatively, you could create the target directories in advance and arrange for appropriate permissions to be granted.

You can use *make install-strip* instead of *make* install to strip the executable files and libraries as they are installed. This will save some space. If you built with debugging support, stripping will effectively remove the debugging support, so it should only be done if debugging is no longer needed. install-strip tries to do a reasonable job saving space, but it does not have perfect knowledge of how to strip every unneeded byte from an executable file, so if you want to save all the disk space you possibly can, you will have to do manual work.

The standard installation provides all the header files needed for Prelude-LML plugin development, such as custom functions or data types written in C.

Installing Prelude-Manager

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

Table of Contents

Install Dependencies

Ensure that you have [[GnuTLS]] installed. Instructions on installing [[GnuTLS]] can be found on the [[InstallingPreludeRequirementGnutls|GnuTLS Installation Page]].

Ensure that you have the Prelude Library installed. Instructions on installing libprelude can be found on the [[InstallingPreludeLibrary|PreludeDB Library Installation Page]]

In order for Prelude-Manager to write to a database, the [[PreludeDB]] Library should be installed. Instructions on installing libpreludedb can be found on the [[InstallingPreludeDbLibrary|PreludeDB Library Installation Page]].

If you'd like Prelude-Manager to be able to report incoming events using IDMEF XML, install the libxml2 library.

Get the sources

Download the latest Prelude-Manager

Configuration

The first step of the installation procedure is to configure the source tree for your system and choose the options you would like. This is done by running the configure script. For a default installation simply enter

```
./configure
```

This script will run a number of tests to guess values for various system dependent variables and detect some quirks of your operating system, and finally will create several files in the build tree to record what it found. (You can also run configure in a directory outside the source tree if you want to keep the build directory separate.)

You can customize the build and installation process by supplying one or more of the following command line options to configure:

Installation directories

- --*prefix*=_PREFIX_
  Install all files under the directory PREFIX instead of /usr/local. The actual files will be installed into various subdirectories; no files will ever be installed directly into the PREFIX directory.

If you have special needs, you can also customize the individual subdirectories with the following options.

- --*exec-prefix*=_EXEC-PREFIX_

You can install architecture-dependent files under a different prefix, EXEC-PREFIX, than what PREFIX was set to. This can be useful to share architecture-independent files between hosts. If you omit this, then EXEC-PREFIX is set equal to PREFIX and both architecture-dependent and independent files will be installed under the same tree, which is probably what you want.

- --*bindir*=_DIRECTORY_

Specifies the directory for executable programs. The default is EXEC-PREFIX/bin, which normally means /usr/local/bin.

- --*datadir*=_DIRECTORY_

Sets the directory for read-only data files used by the installed programs. The default is PREFIX/share.

- --*sysconfdir*=_DIRECTORY_

The directory for various configuration files, PREFIX/etc by default.

- --*libdir*=_DIRECTORY_

The location to install libraries and dynamically loadable modules. The default is EXEC-PREFIX/lib.

- --*includedir*=_DIRECTORY_

The directory for installing C and C++ header files. The default is PREFIX/include.

- --*mandir*=_DIRECTORY_

The man pages that come with the Prelude Library will be installed under this directory, in their respective man_N_ subdirectories. The default is PREFIX/man.

Path to dependencies

- --*with-libgnutls-prefix*=_PFX_

Prelude-Manager require [[GnuTLS]] in order to build. Using this option you can tell the configure script in which prefix [[GnuTLS]] has been installed.

- --*with-libprelude-prefix*=_PFX_

Prelude-Manager require the Prelude Library in order to build. Using this option you can tell the configure script in which prefix libprelude has been installed.

- --*with-libpreludedb-prefix*=_PFX_

Prelude-Manager can make use of libpreludedb to provides database storage support (recommended, but not mandatory). Using this option you can tell the configure script in which prefix libpreludedb has been installed.

- --*with-xml-prefix*=_PFX_

Prelude-Manager can make use of libxml2 to provides IDMEF XML compatible reporting. Using this option you can tell the configure script in which prefix libxml2 has been installed.

- --*with-libwrap-prefix*=_PFX_

Prelude-Manager can make use of libwrap to provides additional host access control scheme. Using this option you can tell the configure script in which prefix libwrap has been installed. libwrap is a free software program library that implements generic TCP Wrapper functionality for network service daemons to use (rather than, or in addition to, their own host access control schemes).

Optional Features

- --*enable-thread={posix|solaris|pth|win32}*

Using this option, you can specify which multithreading API should be used by Prelude-Manager. Note that this value is normally auto-detected.

Once finished, the *configure* script will dump a summary of the enabled features:

```
*** Dumping configuration ***
   - TCP wrapper support    : yes
   - XML plugin support     : yes
   - Database plugin support: yes
```

Note: *Database plugin support* (libpreludedb) is required for database access.

Build

To start the build, type

```
make
```

The build will take a few minutes depending on your hardware.

Regression Tests

If you want to test the newly built program before you install it, you can run the regression tests at this point. The regression tests are a test suite to verify that Prelude-Manager runs on your machine in the way the developers expected it to. Type

```
make check
```

Install

To install Prelude-Manager enter

```
make install
```

This will install files into the directories that were specified in step 1. Make sure that you have appropriate permissions to write into that area. Normally you need to do this step as root. Alternatively, you could create the target directories in advance and arrange for appropriate permissions to be granted.

You can use *make install-strip* instead of *make* install to strip the executable files and libraries as they are installed. This will save some space. If you built with debugging support, stripping will effectively remove the debugging support, so it should only be done if debugging is no longer needed. install-strip tries to do a reasonable job saving space, but it does not have perfect knowledge of how to strip every unneeded byte from an executable file, so if you want to save all the disk space you possibly can, you will have to do manual work.

The standard installation provides all the header files needed for Prelude-Manager plugin development, such as custom functions or data types written in C.

Installing Prewikka

This section explains how to install the Prelude Interface 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.

Table of Contents

Requirements

Prewikka 1.2.6 and higher

- libpreludedb version >= 1.2.6 (database schema 14.7) with python bindings
- Python >= 2.6 (not Python 3.x)
- python-babel
- python-lesscpy
- Cheetah templates for Python from http://cheetahtemplate.org/
- One of the following DBMS:
  - MySQL >= 4.0.18 or
  - PostgreSQL >= 8 or
  - SQLite

Get the sources

Download the latest Prewikka source distribution.

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
```

Create the Prewikka database

Make sure the Prelude Framework (libprelude + libpreludedb with python bindings) is installed.

Three databases are available:

- [[InstallingPreludePrewikka#MySQL|MySQL]]
- [[InstallingPreludePrewikka#PostgreSQL|PostgreSQL]]
- [[InstallingPreludePrewikka#SQLite3|SQLite3]]

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 2
Server version: 5.1.73 Source distribution

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

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 dedicated 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';
```

PostgreSQL

Database creation

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

```
$ psql -U postgres -W
```

where "postgres" is the name of the database administrator (this is the default account on postgresql) and -W will prompt you for a password.

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

```
psql (8.4.20)
Type "help" for help.

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 dedicated 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;
```

SQLite3

Database creation

Make sure the DB file has read permissions for and only for user prelude (or root if you have not created one).

```
$ sqlite3 /path/to/your/sqlite-prewikka.db
```

then, use the following instructions concerning database in /etc/prewikka/prewikka.conf instead of those in the next paragraph:

```
[idmef_database]
type: sqlite3
file: /path/to/your/sqlite-prelude.db

[database]
type: sqlite3
file: /path/to/your/sqlite-prewikka.db
```

Edit 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.
browser_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]
```

Run Prewikka

Apache / WSGI setup with VirtualHost

```
<VirtualHost *:80>
        # This is optional: handling is faster if this is done by Apache, but
        # Prewikka WSGI module will handle it if not.
        #
        #<Location /prewikka>
        #       SetHandler None
        #</Location>
        #
        # Alias /prewikka /usr/share/prewikka/htdocs

        # For inclusion of a non-default configuration file
        # SetEnv PREWIKKA_CONFIG /etc/prewikka/prewikka.conf

        WSGIApplicationGroup %{GLOBAL}
        WSGIScriptAlias / /var/www/html/prewikka.wsgi
</VirtualHost>
```

with a prewikka.wsgi file that should look like:

```
from prewikka.web import wsgi
application = wsgi.application
```

From the command line tool

If you didn't install Prewikka system wide (ie: you specified a prefix), use:

```
$ PYTHONPATH=$prefix/lib/python2.6/site-packages:$prefix/bin/prewikka-httpd
```

Note: Check your install, python2.6 above may be another version.

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 set up, you can use your browser to connect to the machine where Prewikka was installed. If you are not using Apache support, then remember you should use the port 8000 to access Prewikka.
The default login/password is **admin**: please remember to change it.

Depending on the Prewikka plugins you have installed, you might need to activate them through the interface : go to Settings > Plugins > Install update.

SELinux impact

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)

Prelude 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.

Installing from Source

- [[InstallingPreludeRequirementGnutls|GnuTLS]]
  Required by Libprelude

- [[InstallingPreludeRequirementPython|Python]]
  Required for Libprelude and Libpreludedb Python bindings, Prewikka and Prelude-Correlator

- [[InstallingPreludeRequirementPcre|PCRE]]
  Required by Prelude-LML

Databases

- [[InstallingPreludeRequirementMysql|MySQL]]
- MySQL Official Installation Manual

- [[InstallingPreludeRequirementPostgresql|PostgreSQL]]
- PosgreSQL Official Documentation

- SQLite
- SQLite Official Website

Prelude-Admin Manual

Table of Contents

Presentation

prelude-admin can be used to perform various operation involving agent profile.
Profile are required for a Prelude agent to run and communicate correctly with other agents. A profile include:

- An unique analyzer identity.
- A default, template configuration file.
- A private key, used for certificate request generation, and communication encryption.
- X509 certificates used for communication with remote agents.

Creating profile

The **add** command is used to create a new profile. The profile will be created using the permission of the user running the

prelude-admin command, unless specific options are provided. The created profile include a failover spool directory, default configuration files, and a private key used for communication encryption.

Note: Private key generation might take a very long time. Information on speeding up the process are available in the [[MiscEntropy|Entropy Page]].

```
prelude-admin add my-sensor
```

Available options

- --*uid*

UID or user used to create analyzer profile.

- --*gid*

GID or group used to create analyzer profile.

TLS specific options (default defined in the /etc/prelude/default/tls.conf template)

- --*key-len*

Profile private key length (default: 1024 bits).

Changing profile permission

The **chown** command allow to change a profile ownership.

```
prelude-admin chown my-sensor --uid username --gid 1500
```

Available options

- --*uid*

UID or user used as new profile permission.

- --*gid*

GID or group used as new profile permission.

Removing a profile

The **del** command will delete the specified profile.

```
prelude-admin del my-agent
```

Listing profile

The **list** command allow listing profiles available on the system.

Available options

- ~~l~~ --*long*

Print detailed listing (include uid/gid, profile analyzerID).

```
prelude-admin list -l
Profile            UID   GID   [[AnalyzerID]]       Permission       Issuer [[AnalyzerID]]
-------------------------------------------------------------------------------
prelude-lml        yoann yoann 383026397799329  idmef:w          2038315305918460
prelude-correlator  yoann yoann 2052969743121519 idmef:rw admin:r 2038315305918460
                                    idmef:rw         3952006679182968
prelude-manager     yoann yoann 2038315305918460 idmef:r          3952006679182968
                                    idmef:w admin:rw 522518306647497
                                    idmef:w          2038315305918460
                                    idmef:w          1426351250159626
                                    idmef:w          1489495859346286
```

Renaming a profile

The **rename** command can be used to rename a profile.

```
prelude-admin rename my-agent-old my-agent-new
```

Profile registration

The **register** command is used to register the specified analyzer profile to a remote registration server. The analyzer profile will be created if it does not exist.

See the [[InstallingAgentRegistration|Agents Registration Page]] for details on registering agents.

```
prelude-admin register prelude-lml "idmef:w" 192.168.0.1
```

Available options

--uid=UID
  UID or user used to create analyzer profile.
--gid=GID
  GID or group used to create analyzer profile.
--passwd=PASSWD
  Use provided password instead of prompting it.
--passwd-file=-|FILE
  Read password from file (- for stdin).

TLS specific options (default defined in the /etc/prelude/default/tls.conf template)

- --*key-len*
  Profile private key length (default: 1024 bits).

- --*cert-lifetime*=DAYS
  Profile certificate lifetime (default: unlimited).

- --*ca-cert-lifetime*=DAYS
  Authority certificate lifetime (default: unlimited).

Registration server

The **registration-server** command launch a registration server for the specified Prelude-Manager profile. The profile will be created if it does not exist. Registered analyzers will be able to communicate with Prelude-Manager instance using this profile.

See the [[InstallingAgentRegistration|Agents Registration Page]] for details on registering agents.

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

Available options

--uid=UID
  UID or user used to create analyzer profile.
--gid=GID
  GID or group used to create analyzer profile.
--passwd=PASSWD
  Use provided password instead of prompting it.
--passwd-file=-|FILE
  Read password from file (- for stdin).

TLS specific options (default defined in the /etc/prelude/default/tls.conf template)

- --*key-len*
  Profile private key length (default: 1024 bits).

- --*cert-lifetime*=DAYS
  Profile certificate lifetime (default: unlimited).

- --*ca-cert-lifetime*=DAYS
  Authority certificate lifetime (default: unlimited).

Revoking profile

The **revoke** command will revoke the analyzer using the given analyzerID from the specified profile. Analyzer using the revoked analyzerID won't be able to communicate with the profile it was revoked from anymore.

```
prelude-admin revoke prelude-manager 227879253605921
```

Miscellaneous usage

Sending messages

The **send** command allow to send the messages contained within a Prelude IDMEF binary file (example: failover file) to the specified Prelude-Manager address. The specified profile is used for authentication.

```
prelude-admin send prelude-lml 192.168.0.1 /path/to/file1 /path/to/file2 /path/to/fileN
```

Available options

--*offset*=OFFSET
  Skip processing until 'offset' events.
--*count*=COUNT
  Process at most 'count' events.

Printing failover messages

Using the **print** command, you can print the messages within a Prelude IDMEF binary file (example: failover file) to stdout using an human readable format.

```
prelude-admin print /path/to/file1 /path/to/file2 /path/to/fileN
```

Available options

--*offset*=OFFSET
  Skip processing until 'offset' events.
--*count*=COUNT
  Process at most 'count' events.

PreludeDB-Admin Manual

Table of Contents

Presentation

preludedb-admin can be used to perform various operation on the Prelude IDMEF database. It support moving/copying data between database, exporting event from a database, restoring event in a database, events deletion and printing.

Run preludedb-admin with no argument to get a listing of available commands. By specifying a command with no argument, you can get a detailed help and example concerning the command.

Database settings

All preludedb-admin command require database settings to be provided, in order for the tool to access the database. Database settings are a string of key=value pair, made of the following key:

- type

Type of the database (mysql, pgsql, sqlite).

- name

Name of the database

- user

Username to use to access the database

- pass

Password to use to access the database

- host

Optional host where the database should be contacted (default is localhost).

- port

Optional port where the database should be contacted (default depend on the selected database *type*).

Database settings example, to access a [[MySQL]] database named prelude:

```
"type=mysql name=prelude user=prelude pass=prelude"
```

Various options

A number of options, compatible with all preludedb-admin command, are available:

- --*offset*

Skip processing until 'offset' events.

- --*count*

Process at most this number of events.

- --*query-logging*

Log SQL query to the specified file.

- --*criteria*

Only process events that match the provided [[IDMEFCriteria|criteria]].

- --*events-per-transaction*

By default, preludedb-admin will surround multiples write/deletion command to the database by a transaction, allowing to rollback in case of error. The number of events to process before flushing the transaction can be controlled using this option. Higher value provide improved performance, at the price of an higher memory pressure. The default is to process 1000 events per transaction.

Removing events

In order to delete events from a specific database, you might use the **delete** command. Note that the criteria argument is mandatory for the delete command, since providing no criteria would result in the whole database being deleted.

Example for flushing any event created before a specific date:

```
preludedb-admin delete alert --criteria "alert.create_time < YYYY-MM-DD" "type=mysql name=prelude user=prelude
pass=prelude"
```

Note: If you are using the [[MySQL]] InnoDB database engine, keep in mind that deletion of old events does not shrink the database size. In order to shrink an [[InnoDB]] database, you can save the whole database using the mysqldump tool, drop the database, and rebuild it using the previously generated backup.

Automatically removing events older than ..

As explained above, you to use preludedb-admin with a 'string' of parameters that perform the wanted actions, amongst other delete given alerts from the database. This is a manual task, and the following bash script allows to automate these tasks for the given context: Remove alerts older than ...

Please note that the bash script is only an example for deleting alerts older than 30 days. When added to a cron job running daily for example, your alerts dataase will only contain alerts younger than 30 days. and is only valid for Linux based systems that can use the date command.

```
#!/bin/sh
DATE=$(date -d "last month" +"%Y-%m-%d")
preludedb-admin delete alert "type=pgsql name=prelude user=prelude pass=prelude" --criteria "alert.create_time < $DATE"
```

How it works: date ~~d "last month" will take the current date and substract 30 days, the +"%Y~~%m-%d" formats the output of the date command in the right format for libpreludedb-admin.

Date examples: -d "10 days", -d "2 weeks", -d "last year", ...

Adjust the prelude-db admin command to suit your needs and your configuration.

Copy or Move events to another database

Using preludedb-admin, it is possible to easily copy or move data between database. Additional option might be specified, for example, you might want to provide an IDMEFCriteria using the --*criteria* option, or define a specific limit or offset.

```
preludedb-admin copy alert "type=mysql name=prelude user=prelude pass=prelude" "host=pgsql.host type=pgsql
name=prelude pass=prelude"
preludedb-admin copy heartbeat "type=mysql name=prelude user=prelude pass=prelude" "host=pgsql.host type=pgsql
name=prelude pass=prelude"
```

The command used to move data to another database is mostly the same:

```
preludedb-admin move alert "type=mysql name=prelude user=prelude pass=prelude" "host=pgsql.host type=pgsql
name=prelude pass=prelude"
preludedb-admin move heartbeat "type=mysql name=prelude user=prelude pass=prelude" "host=pgsql.host type=pgsql
name=prelude pass=prelude"
```

Save event to a file

Using the preludedb-admin **save** command, you can retrieve events from a database and store them in a file. The format is a binary, machine independant Prelude format, thus the generated file can be transfered on another machine without compatibility issue. If you do not specify an output filename on the command line, then then data will be directly written to the standard output.

Example to retrieve every database alerts, and store them in the *alert.out* file:

```
$ preludedb-admin save alert "type=mysql name=prelude user=prelude pass=prelude" alert.out
```

Load event from a file

Event saved through the *preludedb-admin save* command can be imported into the Prelude database again. If no filename is provided, preludedb-admin will attempt to read from standard input.

```
$ preludedb-admin load "type=mysql user=root pass=root name=xlr" alert.out
```

Print database events

```
$ preludedb-admin print alert "type=mysql user=prelude pass=prelude name=prelude" --count 1
alert:
        messageid: 898ac484-4bd2-11dd-a208
        analyzer(0):
                analyzerid: 2012219155818466
                name: prelude-manager
                manufacturer: http://www.prelude-ids.com
                model: Prelude Manager
                version: 0.9.12.1
                class: Concentrator
                ostype: Linux
                osversion: 2.6.24.5-85.fc8
                node:
                        category: hosts (6)
                        location: Paris
...
```

Prewikka Manual

Table of Contents

Overview

Prewikka is the official Prelude User Interface. The Prewikka interface is a web GUI compatible with IE >= 9, Firefox >= 18, Chrome >= 26.
Prewikka is open source and is released under GPLv2 license. Prewikka has been developped in **Python language**.
Prelude supports real-time visualization of data thanks to Prewikka which provides automatic reloading of the alert listing.
PrewikkaPro, the commercial version of Prewikka provides additional functionalities. It is available through the Prelude SIEM website.

Caution: All Prewikka and PrewikkaPro functionalities are listed in this manual. So if you don't see all these functionalities on your own system, this is because either you use Prewikka and you are reading about a PrewikkaPro functionality, or you don't have the necessary permissions in Prewikka to see / use it.

Core Features

Below are listed the Prewikka and PrewikkaPro main functionalities. More details and pictures are available on the Prelude SIEM website.

Prewikka functionality

- Advanced Aggregation System
- Filter creation
- Sensor monitoring
- Alert listing automatic refresh
- Plugins architecture
- Themes

PrewikkaPro functionality

PrewikkaPro is the commercial version of Prewikka.

- Permission management
- Advanced Ticket System
- Graphical Fully Interactive Statistics
- Graphical Fully Interactive Forensic
- Reporting (PCI DSS, vulnerabilities, ...)
- Ability to Create Virtual Alert "Views"

- Expert alert listing
- Alert Listing PDF Export
- Users and groups management
- Secured Authentication from LDAP server
- System command
- Graphical LML and Correlator edition

Getting Started

Technical Requirements

Before you begin using the Prewikka interface, ensure that you have the required software installed and configured on your system as follows:

1. **A current Web browser on your computer**

Prewikka is compatible with:

- **Microsoft IE** - www.microsoft.com/ie
- **Firefox** - www.mozilla.org/firefox
- **Google Chrome** - www.google.com/chrome

You may encounter problems if you try to access Prewikka using old Web browser versions.

2. **Enable Java Script and cookie support on your Web browser**

Both Java Script and cookie support must be enabled in the security settings of your browser and is usually turned on by default. If you encounter problems accessing the system, check your browser configuration to ensure both Java Script support and cookie support are enabled as follows:

- **IE:** Click *Tools* > *Internet Options* > *Privacy* and *Security* tabs
- **Firefox:** Click *Tools* > *Options* > *Privacy* and *Web Features* tabs

3. **Network access to a server that is running the Prewikka software**

Your system or network administrator can provide you with a Web address (URL) from which the system can be accessed.

Accessing Prewikka

You access Prewikka through a Web browser.
To log into the system:

- Enter the Prewikka URL in the address bar of your Web browser.

Prewikka does not support users management so you are logged as *Anonymous* user.

Setting Your Preferences

On your *My Account* page (click on the user on the top right side of the screen), you can view the settings.
As a user, you can edit some of these settings, such as your preferred language and your theme.

To set your language, click your User Name link located at the top right side of the page.
1. Language setting

Choose the appropriate language:

German, English, Spanish, French, Italian, Polish, Portuguese (Brazilian), Russian

Session Timeout

For security reasons, the system automatically logs you out of the interface if you don't perform any tasks during one hour (default configuration).

This doesn't happen if the alert listing automatic refresh is activated and the refresh time is less than one hour.

Overview of the Prewikka User Interface

prewikka-overview.png

Using Prewikka

Alerts section

The *Alerts* section allows you to see and manage your security alerts.

In order to learn how to use the *Alerts* section, **see the [[ManualPrewikkaEvents|Alerts Section Page]]**

Agents section

The *Agents* section allows you to manage and monitor your agents.

In order to learn how to use the *Agents* section, **see the [[ManualPrewikkaAgents|Agents Section Page]]**

Tickets section (!PrewikkaPro only)

The *Tickets* section allows you to create and manage your tickets.

In order to learn how to use the *Tickets* section, **see the [[ManualPrewikkaTickets|Tickets Section Page]]**

Statistics section (!PrewikkaPro only)

The *Statistics* section enables you to navigate inside of alerts in a transversal manner.

In order to learn how to use the *Statistics* section, **see the [[ManualPrewikkaStatistics|Statistics Section Page]]**

Settings section

The *Settings* section allows you to manage your virtual views, your filters, your preferences and to create or delete users.

In order to learn how to use the *Settings* section, **see the [[ManualPrewikkaSettings|Settings Section Page]]**

About section

In the *About* section, you will find the version of your Prewikka software, a description of the services provided by the [[CS]] company, and the company contact details.

Architecture overview

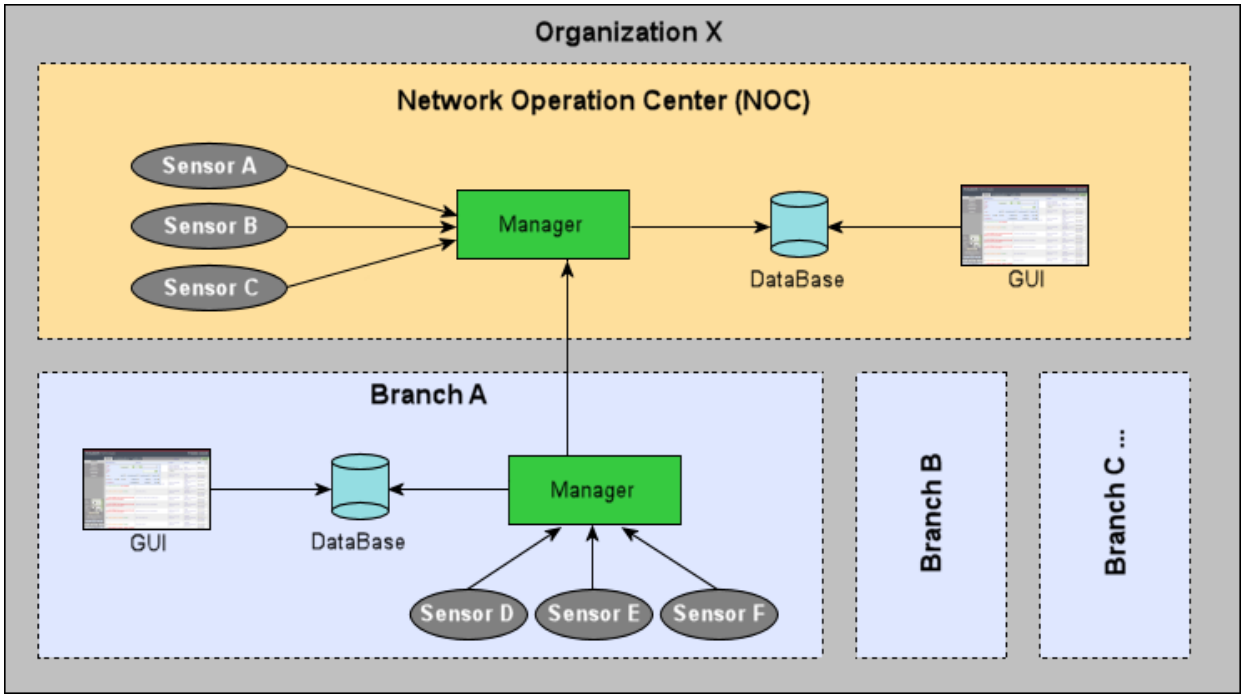
Prelude is divided in several components. Sensors are responsible for intrusion detection, and report alerts in a centralized fashion using a TLS connection to a 'prelude-manager' server. The prelude-manager server can then process these alerts and deliver them to an user-specified media (mysql database,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

The Prelude console can then be used to view these alerts.
Here is a simple example of how the different Prelude components interact:

simple-architecture.png

Architecture overview with commercial extension

You can do decentralized architecture with the commercial extension available from CS-SI like this :



or this

reverse-relaying.png
Please check the Corporate Modules page for more information.

Prelude Compatibility

Prelude is a SEM system capable of interoperating with all the systems available on the market.

Prelude provides a C, C++, Python, Ruby, Lua and Perl framework so that you can convert existing security applications to use the Prelude system (Native compatibility). It also provides some home-made sensors such as the Prelude-LML log analyzer (Log Compatibility) or the [[PreludeImport|Prelude-Import]] data importer which is a commercial module. A Prelude sensor is a program which has the ability of using the Prelude framework.

framework.png

Native Compatibility

The native compatibility existing between Prelude and certain security solutions improves the quality of the data collection and configuration possibilities.

	AuditD		The Linux Audit Daemon	
	Nepenthes		A versatile tool to collect malware	
	ufwi-filterd		An identity access management solution at the network level	
	OSSEC		An Open Source Host-based Intrusion Detection System	
	Pam		Linux Pluggable Authentication Modules	
	Samhain		A file integrity checker	
	Sancp		A network traffic statistical information collector	
	Snort		The Defacto Standard Open Source IDS	

	Suricata (new)		Maybe the future Defacto Standard Open Source IDS	

Learn more and install the Prelude native sensors with the [[InstallingAgentThirdparty|3rd Party Agent Installation Page]]

Log Compatibility

Prelude depends on no single brand or format and is capable of analyzing any type of log (system logs, syslog, flat files, etc.)
Example of supported logs:

	Firewall, Routers & VPN		BIG-IP, Check Point, CISCO ASA, CISCO IOS, CISCO Router, CISCO VPN, D-Link, Ipchains, IpFw, Juniper Networks NetScreen, Linksys WAP11, ModSecurity v2, Netfilter, SonicGuard SonicWall, Symantec, xg45-datapower	
	Switchs		CISCO CSS	
	IDS		CISCO IPS, Portsentry, Shadow, Tripwire	
	Monitoring		APC-EMU, ArpWatch, Dell OpenManage, Nagios	
	AntiVirus/AntiSpam		ClamAV, P3Scan, SpamAssassin, Navce	
	Database		Microsoft SQL Server, Oracle	
	SMTP/POP Server		Exim, Postfix, Qpopper, Sendmail, Vpopmail	
	FTP Server		ProFTPD, WU-FTPD	
	Web Server		Apache	
	Vulnerability Scanner		Nessus	
	Honeypots		Honeyd, Honeytrap, Kojoney	
	Authentication		OpenSSH, Su, Radius	
	Applications		Asterisk, Cacti, Libsafe, Shadow Utils, Squid, Sudo, Cisco ACE, Pam, Pcanywhere	
	OS (security tools)		GrSecurity, PaX, SELinux	
	Miscellaneous		Unix specific logs, Webmin, Windows Server, Arbor, Linux bonding, Microsoft Cluster Service, NetApp ONTAP, NTSyslog, OpenHostAPD, Rishi, Suhosin, Vigor	

Prelude Components

The Prelude Universal SEM system is distributed, meaning it comprises multiple modular elements.

Prelude Manager

Prelude-Manager is a high availability server that accepts secured connections from distributed sensors and/or other Managers and saves received alerts to a media specified by the user (database, log file, mail etc.). The server schedules and establishes the priorities of treatment according to the critical character and the source of the alerts.
The Prelude Manager is a concentrator capable of handling large number of connections, and processing large amounts of alerts. It uses a per client scheduling queues in order to process alerts by severity fairly across clients.

The Prelude Manager comes with multiple plugins like filtering plugins (idmef-criteria, thresholding, etc.) or reporting plugins like the SMTP plugin which automatically sends emails containing a textual description of alerts to a configured list of recipients.

See the [[PreludeManager|Prelude Manager Page]] to learn more about Prelude Manager Configuration.

Libprelude

Libprelude is a library that guarantees secure connections between all sensors and the Prelude Manager. Libprelude provides an Application Programming Interface (API) for the communication with Prelude sub-systems, it supplies the necessary functionality for generating and emitting IDMEF alerts with Prelude and automates the saving and re-transmission of data in times of temporary interruption of one of the components of the system.
Libprelude also makes it easy for third party software to be made "Prelude Aware" (able to communicate with Prelude components). This library provides common, useful features used by every sensor.

LibpreludeDB

The PreludeDB Library provides an abstraction layer 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.

Prelude-LML

Prelude-LML is a log analyser that allows Prelude to collect and analyze information from all kind of applications emitting logs or syslog messages in order to detect suspicious activities and transform them into Prelude-IDMEF alerts. Prelude-LML handles alerts generated by a large set of applications, see the [[PreludeCompatibility|Compatibility Page]] to learn more.

See the [[PreludeLml|Prelude-LML Page]] to learn more about Prelude-LML Configuration.

Prelude-Correlator

Prelude-Correlator allows conducting multistream correlations thanks to a powerful programming language for writing correlation rules. Prelude-Correlator is a Python rules based correlation engine. It has the ability to connect and fetch alerts from a remote Prelude-Manager server, and correlate incoming alerts based on the provided ruleset. Upon successful correlation, IDMEF correlation alerts are raised.

See the [[PreludeCorrelator|Prelude-Correlator Page]] to learn more about Prelude-Correlator.

Prewikka

Prewikka is the official web Graphical User Interface (GUI) for the Prelude Universal SEM system. Providing numerous features, Prewikka facilitates the work of users and analysts. Prewikka also provides access to external tools such as whois and traceroute.

See the [[ManualPrewikka|Prewikka Manual]] to learn more about Prewikka.

Foreword

The Prelude Intrusion Detection System (IDS) was created in 1998 by information security expert Yoann Vandoorselaere. Prelude was born from the observation that while a growing number of Intrusion Detection Systems (Network based IDS - NIDS, File Integrity Checkers, Vulnerability scanners, etc.) were arriving on the market, there was no framework allowing them to work in concert and thereby boost security exponentially.

The project, called Prelude-IDS, boasts an ant as its mascot. Ants live in colonies, which has enabled them to develop excellent abilities in communication and cooperation. This, coupled with their warrior nature, makes ants the perfect symbol for a project to develop a security platform composed of distributed defence systems.

At this initial stage, Prelude comprised a NIDS (Prelude-NIDS) and a decentralised server. The licensing model selected for Prelude was a natural choice: **open source (Gnu General Public license – GPL)**. Its author is already very active in the open source milieu, which corresponds to his ideals and allowed him to learn programming.

In 1999, MandrakeSoft (a company renamed to Mandriva then forked to OpenMandriva, https://www.openmandriva.org) took an interest in the project and came aboard as its sponsor. This sponsorship over a three year period accelerated development of the software, which became a hybrid IDS.
This meant Prelude was now capable of processing data from both NIDS and HIDS. Once an innovative concept, Prelude had attained the status of an innovative solution.

In order to allow different security tools on the market to interact with each other, it became necessary to create a single communication format. **At the initiative of IETF and with the participation of the Prelude team, the "Intrusion Detection Message Exchange Format" (IDMEF) project was launched in 2001.**

In 2004, the Switzerland-based company Dreamlab (http://www.dreamlab.net) became a supporting partner in the project.

In 2005, the growing success of Prelude led to the founding of a new company – [[PreludeIDS]] Technologies. This enterprise made it possible to meet new demands for related services (support, development, etc.) and to spur development of the software. Dreamlab was the company's first commercial partner.

Given the major advances made on the IDMEF project, universal interoperability became the priority. The Prelude team decided to depreciate Prelude-NIDS in favor of an external NIDS: Snort.

In 2007, IDMEF was officially ratified and became the international standard in intrusion detection message exchange. Prelude became an IDS Framework.

The release of its multistream correlation engine has brought new impetus to Prelude's development in 2008, paving the way for the release of version 1.0 and marking the 10th anniversary of innovation and cooperation on the Prelude project.

Today, Prelude has become a universal Security Information Management (SIM) system. Prelude collects, normalizes, sorts, aggregates, correlates and reports all security-related events independently of the product brand or license giving rise to such events; Prelude is "agentless". As well as being capable of recovering any type of log (system logs, syslog, flat files, etc.), Prelude benefits from a native support with a number of systems dedicated to enriching information even further (snort, samhain, ossec, auditd, etc.).

Since its creation, security engineers and specialists have enthusiastically contributed to the Prelude project in the spirit of open source. Powerful developments lie ahead in order to continue revolutionizing the world of information security.

PreludeAnt.png

Prelude Standards

IDMEF Standard

Since Prelude handles events from different kinds of sensors, a generic events description language had to be chosen. Prelude uses Intrusion Detection Message Exchange Format (IDMEF) as the common languages for reporting events.

The purpose of 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.

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 enables 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 [[PreludeManager]] and [[PreludeImport]] can output and import XML based IDMEF within the Prelude system.

Creating filter using IDMEF Criteria

IDMEF Criteria are filters on IDMEF fields.
Several Prelude components use IDMEF Criteria in order to provide IDMEF filtering capability.

- [[PreludeManager|Prelude-Manager]] comes with the *idmef-criteria-filter* plugin, that can be hooked and use IDMEF Criteria, defining specific actions to take for an incoming events (relaying it, storing it in a specific database, etc).

- [[ManualPrewikka|The Prewikka interface]] allows the user to setup criteria in order to filter query results.

IDMEF Criteria syntax

<IDMEF Path> <operator> <value>

Simple example:

alert.analyzer(-1).name = 'MySensor'

Note that you can use boolean AND / OR:

alert.analyzer(-1).name = 'MySensor' && ('alert.assessment.impact.severity = 'high' || alert.assessment.impact.completion = 'succeeded')

Available IDMEF path

In order to know more about the IDMEF Path you can use, please have a look at the [[IDMEFPath]] documentation

Available operator

= -> Case sensitive equal.
=* -> Case insensitive equal.
!= -> Case sensitive not equal.
!=* -> Case insensitive not equal.
~ -> Case sensitive Regexp.
~* -> Case insensitive regexp.
!~ -> Case sensitive, not matched regexp.
!~* -> Case insensitive, not matched regexp.
< -> Lower than.
<= -> Lower or equal than
> -> Higher than.
>= -> Higher or equal than.
<> -> Case sensitive sub-string.
<>* -> Case insensitive sub-string.
!<> -> Case sensitive not sub-string.
!<>* -> Case insensitive not sub-string.

Prelude IDMEF Path

In Prelude, an IDMEF Path is a pointer to a specific value in an IDMEF message.
Using this pointer, you can update or retrieve the value pointed by that path.

Any class specified in the IDMEF RFC can be converted to a Prelude IDMEF path.

Mapping an IDMEF class to a Prelude IDMEF Path

When mapping an IDMEF XML class to Prelude, you should obey the following rules:

- Prelude IDMEF Path are lower case.
- Subsequent member of a path should be separated using "*.*"
- Where IDMEF XML class use upper case for word separation, Prelude use a dash "*_*".
- In case a path member is a list, you can use a specific index to access the path, for example: alert.source*(0)*.node.name

Indexed member

When accessing listed object (example: alert.source, alert.additional_data), the index can be specified as a number, which should be a currently existing index, or, in the context of assigning an object, the first unassigned index of the list (example, if source(0) and source(1) currently exist, you might use index 0, 1, or 2 to assign data, but not 3).

Note that you might also use negatives indexes to access listed objects: -1, -2, -n...

-1 will always point to the end of the list, -2 to the object next to last, etc.

Additionally, in the context of an assignement, you might use the **>>append** or **prepend** operator:

```
alert.source(>>).interface = eth1
alert.source(-1).node.name = myNode
```

The above example append a source object to the current 'alert.source' list, which *interface* value is eth1. It then set the *node.name* attribute for this added source object to *myNode*.

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE IDMEF-Message PUBLIC "-//IETF//DTD RFC XXXX IDMEF v1.0//EN"
"idmef-message.dtd">

<IDMEF-Message version="1.0" xmlns="urn:iana:xml:ns:idmef">
 <Alert messageid="abc123456789">
  <Analyzer analyzerid="hq-dmz-analyzer01">
   <Node category="dns">
    <location>Headquarters DMZ Network</location>
    <name>analyzer01.example.com</name>
   </Node>
  </Analyzer>

  <CreateTime ntpstamp="0xbc723b45.0xef449129">
   2000-03-09T10:01:25.93464-05:00
  </CreateTime>

  <Source ident="a1b2c3d4">
   <Node ident="a1b2c3d4-001" category="dns">
    <name>badguy.example.net</name>
    <Address ident="a1b2c3d4-002" category="ipv4-net-mask">
     <address>192.0.2.50</address>
     <netmask>255.255.255.255</netmask>
    </Address>
   </Node>
  </Source>

  <Target ident="d1c2b3a4">
   <Node ident="d1c2b3a4-001" category="dns">
    <Address category="ipv4-addr-hex">
     <address>0xde796f70</address>
    </Address>
   </Node>
  </Target>

  <Classification text="Teardrop detected">
   <Reference origin="bugtraqid">
    <name>124</name>
    <url>http://www.securityfocus.com/bid/124</url>
   </Reference>
  </Classification>
 </Alert>
</IDMEF-Message>


alert.messageid=abc123456789
alert.analyzer(0).analyzerid=hq-dmz-analyzer01
alert.analyzer(0).node.category=dns
alert.analyzer(0).node.location=Headquarters DMZ Network
alert.analyzer(0).node.name=analyzer01.example.com
alert.create_time=0xbc723b45.0xef449129
alert.source(0).ident=a1b2c3d4
```

```
alert.source(0).node.ident=a1b2c3d4-001
alert.source(0).node.category=dns
alert.source(0).node.name=badguy.example.net
alert.source(0).node.address(0).ident=a1b2c3d4-002
alert.source(0).node.address(0).category=ipv4-net-mask
alert.source(0).node.address(0).address=192.0.2.50
alert.source(0).node.address(0).netmask=255.255.255.255
alert.target(0).ident=d1c2b3a4
alert.target(0).node.ident=d1c2b3a4-001
alert.target(0).node.category=dns
alert.target(0).node.address(0).category=ipv4-addr-hex
alert.target(0).node.address(0).address=0xde796f70
alert.classification.text=Teardrop detected
alert.classification.reference(0).origin=bugtraqid
alert.classification.reference(0).name=124
alert.classification.reference(0).url=http://www.securityfocus.com/bid/124
```

Database Optimisation

[[MySQL]] performance

First, ensure yourself that */var/lib/mysql* is a separate partition, mounted with **noatime**.

On many linux distributions, the default [[MySQL]] config is suited for a very low-end machine (32 MB of RAM). You have to configure it in */etc/mysql/my.cnf*, for ex:

```
innodb_data_home_dir=/db/disk2/data
innodb_data_file_path=ibdata1:10240M;ibdata2:10240M;ibdata3:10240M;ibdata4:10240M;ibdata5:10M:autoextend
innodb_log_group_home_dir=/var/log/mysql
innodb_log_arch_dir=/var/log/mysql
innodb_table_locks=0
innodb_buffer_pool_size=1800M # USE ALL MEMORY AVAILABLE
#innodb_log_buffer_size=8M # Lowered from 32M according to [[MySQL]]
innodb_additional_mem_pool_size=20M
```

Disable binary logs, unless you have a good reason to keep them (like using replication or PITR). Just comment the line in the config file:

```
log_bin = /var/log/mysql/mysql-bin.log
```

Important things:
- When removing data, remember that [[MySQL]] **does not delete anything from disk** from an [[InnoDB]] partition, and that you can't reclaim free space ! Problem is known, but nothing was done to fix it (See comments in http://bugs.mysql.com/bug.php?id=1287 and http://bugs.mysql.com/bug.php?id=1341)
- There is no vacuum in [[MySQL]], but you can use the *OPTIMIZE TABLE* command to do the same. However, remember this command **will lock the table** during its execution, and can take some time.

See
- http://www.mysqlperformanceblog.com/2006/05/30/innodb-memory-usage/
- http://www.mysqlperformanceblog.com/2007/11/03/choosing-innodb_buffer_pool_size/

[[PostgreSQL]] performance

First, ensure yourself that */var/lib/postgresql* is a separate partition, mounted with **noatime**.
Here's some tips for performance:
- **shared_buffers**: this is the most important parameter. On a dedicated server, raise it to 75-80% of the total RAM (you may need to change the *shmmax* sysctl value).
- **work_mem**: this is the amount of memory used for internal operations like sorting and hashing. The default is *1MB*, you can raise it a bit (between 32MB and 64MB). Note that each connexion can use this amount of memory, so do not set the value too high.
- Raise **default_statistics_target** to something like 100 (it must be between 1 and 1000, default is 10). This parameter controls how the query planner will decide to choose a plan or another, and can lead to serious performance problems if too low
- **join_collapse_limit**: this parameters controls how deep the query planner will search and try to reorder JOIN instructions.

Prelude uses many JOINs, so you should raise this to 12 or 15 (default is 8)
- remember to vacuum regularly !
- Depending on your data, you may need to change **max_fsm_relations** and **max_fsm_pages**

See
- http://www.postgresql.org/docs/8.3/static/runtime-config-resource.html
- http://www.desknow.com/kb/idx/0/061/article/
- http://wiki.postgresql.org/wiki/Performance_Optimization

Remove heartbeats

Heartbeats are logged for every sensor, and there are *tons* of them. This can completely cripple performances (especially when using Prewikka) of the database, so you should remove them. Writing a simple shell script, executed by a cron job every day, is a trivial task:

```
#!/bin/sh

set -e

DB_TYPE="pgsql"
DB_HOST="localhost"
DB_USER="prelude"
DB_PASS="xxxxxx"

KEEP_INTERVAL="2 month"

DATE=$(date -d "now - $KEEP_INTERVAL" +%Y-%m-%d)

preludedb-admin delete heartbeat --criteria "heartbeat.create_time <= $DATE" "type=$DB_TYPE host=$DB_HOST
user=$DB_USER pass=$DB_PASS"
```

Clean up old alerts

Note: Be careful ! Deleting old entries may remove important data.

The *preludedb-admin* command can be used to suppress alerts, based on some conditions (called criteria).

```
# preludedb-admin delete --alert-criteria "alert.create_time <= 2007-06-15" "type=pgsql user=prelude pass=<db_password>"
```

With recent versions, the syntax has changed:

```
# preludedb-admin delete alert --criteria "alert.create_time <= 2007-08-15" "type=mysql user=prelude pass=<db_password>"
```

This command will delete all alerts prior to 2007-06-15

If you are using [[MySQL]] and got the following error:

```
retrieving alert ident failed: The total number of locks exceeds the lock table size.
```

Then edit [[MySQL]] configuration file and try increasing *innodb_buffer_pool_size*.

See http://mrothouse.wordpress.com/2006/10/20/mysql-error-1206/ for some details.

Replicating a database to another

The *preludedb-admin* tool has a *copy* mode.

```
# preludedb-admin copy alert "type=pgsql name=prelude user=prelude pass=***** host=192.168.1.101" "type=mysql
name=prelude user=prelude pass=****** host=192.168.1.104"
# preludedb-admin copy heartbeat "type=pgsql name=prelude user=prelude pass=***** host=192.168.1.101" "type=mysql
```

name=prelude user=prelude pass=****** host=192.168.1.104"

Glossary

The Prelude system, and its graphical interface Prewikka in particular, uses several terms that need to be defined. Most of them derive from the [[PreludeStandards|IDMEF standard]].

Alert (IDMEF)

A data structure describing a security incident. It contains information about :

1. its classification
2. the sensor from which it originates
3. the time of detection/creation
4. the source and the target
5. its assessment (impact of the event)

It can be a simple alert or the grouping of related alerts into a "correlation alert".

Heartbeat (IDMEF)

A message sent in a regular period by analyzers to their attributed manager(s), in order to indicate that they are running.
The lack of some number of consecutive heartbeats means the failure of either the analyzer or its network connection.

Analyzer/Sensor (IDMEF)

The emission source of an alert or heartbeat message.
This definition can be extended to cover not only Prelude agents (Prelude-LML, Prelude-Correlator, Snort, Samhain...) but also services (web server, PAM, ssh, antivirus...) that write in logs instead of emitting alerts.

Manager (IDMEF?)

The destination of an alert or heartbeat message.
The manager can process the received data and for example deliver it to a database. It can also act as a relay, i.e. forward the messages to another manager.
Example : Prelude-Manager

Agent (Prelude)

A Prelude client, i.e. a program using libprelude. It can be an analyzer or a manager.
Examples : Prelude-LML, Prelude-Manager...

Node (IDMEF)

An equipment that hosts one or several analyzers/managers, identified by a network address or a name.
Example : 192.0.2.1

Location (IDMEF)

The physical site of one or several nodes.
Example : Paris

Agents Installation

[[InstallingAgentRegistration|Agents Registration]]

[[InstallingAgentThirdparty|3rd Party Agents Installation]]

Agent Registration

Table of Contents

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

Registration

In order for an agent to connect and 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'.

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 multiples instances of one sensor running with different permissions, which require different profiles.

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

Agent Registration / Profile Creation

The agent 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 will show you a warning including the default profile name to be used for registering 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 the created profiles information (key, cert...), you will get an error and the sensor will refuse to start.

The first time an agent is registered, **prelude-admin** will need to create a private key for the agent. Under Linux, the operation can take a very long time due to the entropy generation system: advises on fixing this problem are available on the [[Misc/Entropy]] page.

Requested Permission

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

- idmef
- admin

Both *idmef* and *admin* type can take **read** (*r*) and **write** (*w*) permission. Usually, a sensor need permission of writing IDMEF messages to a manager, and optionally accept administrative command sent to the sensors. That is : **idmef:w admin:r**.

Note: 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 registration.

Manager Addresses

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.

Prelude-LML Registration example

This is an example on registering *Prelude-LML* on host *lmlhost* to a *Prelude-Manager* running on host *managerhost*:

$ prelude-admin register prelude-lml "idmef:w admin:r" *managerhost*
Generating 1024 bits RSA private key... This might take a very long time.
[Increasing system activity will speed-up the process].
Generation in progress... X

Note: The first time an agent is registered, **prelude-admin** will need to create a private key for the agent. Under Linux, the operation can take a very long time due to the entropy generation system: advises on fixing this problem are available on the [[Misc/Entropy]] page.

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

You now need to start "prelude-admin" registration-server on managerhost:
example: "prelude-admin registration-server prelude-manager"

Enter the one-shot password provided on managerhost:

On **managerhost**, you now need to start *prelude-admin registration-server* command using the profile name used by your *Prelude-Manager*:

$ prelude-admin registration-server prelude-manager

The "deadbeaf" password will be requested by "prelude-admin register"
in order to connect. Please remove the quotes before using it.

Generating 1024 bits Diffie-Hellman key for anonymous authentication...++++++++++.++++++..++++++++++
Waiting for peers install request on 0.0.0.0:5553...

As you can see, the generated password is **deadbeaf**. You need to enter this password in the **lmlhost***prelude-admin* session:

Enter the one-shot password provided on localhost: [you don't see this, but deadbeaf is typed]
Confirm the one-shot password provided on localhost: [you don't see this, but deadbeaf is typed]

Connecting to registration server (localhost:5553)... Authentication succeeded.
Successful registration to localhost:5553.

This is what you'll get on the server side:

Connection from 127.0.0.1:52507...

Registration request for analyzerID="229348179011709" permission="idmef:w admin:r".
Approve registration? [y/n]: y
127.0.0.1:52507 successfully registered.

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

3rd Party Agents Installation

- **[[InstallingAgentThirdpartyAuditd|Auditd]]** (Homepage) - The Linux Audit Daemon.
  Auditd provides user-space utilities for creating audit rules, as well as for storing and searching audit records generated by the audit subsystem in the Linux 2.6 kernel. It features an Intrusion Detection plugin that analyses the audit stream in realtime for suspicious events and alerts via IDMEF using Prelude.

- **[[InstallingAgentThirdpartyNufw|ufwi-filterd]]** (Homepage)
  ufwi-filterd adds user-based filtering to Netfilter, the state of the art IP filtering layer from the Linux kernel. Its exclusive algorithm allows authenticated filtering even on multiuser computers. ufwi-filterd can be seen as an Identity access management solution, at the network level.

- **[[InstallingAgentThirdpartyLinuxpam|LinuxPAM]]** (Homepage)
  Linux-PAM is a system of libraries that handle the authentication tasks of applications on the system. The library provides a stable general interface that privilege granting programs (such as login and su) defer to perform standard authentication tasks.

- **[[InstallingAgentThirdpartyNepenthes|Nepenthes]]** (Homepage)
  Nepenthes is a versatile tool to collect malware. It acts passively by emulating known vulnerabilities and downloading malware trying to exploit these vulnerabilities.

- **[[InstallingAgentThirdpartyOssec|OSSEC]]** (Homepage)
  OSSEC is an Open Source Host-based Intrusion Detection System. It performs log analysis, integrity checking, Windows registry monitoring, rootkit detection, real-time alerting and active response.

- **[[InstallingAgentThirdpartySamhain|Samhain]]** (Homepage) - The File Integrity Checker
  Samhain is a multiplatform, open source host-based intrusion detection system (HIDS) for POSIX (Unix, Linux, Cygwin/Windows). Samhain provides file integrity checking, as well as rootkit detection, port monitoring, detection of rogue SUID executables, and hidden processes.

- **[[InstallingAgentThirdpartySancp|SanCP]]** (Homepage)
  SanCP is a network security tool designed to collect statistical information regarding network traffic, as well as, record the traffic itself to file in pcap format for the purpose of: auditing, historical analysis, and network activity discovery.

- **[[InstallingAgentThirdpartySnort|Snort]]** (Homepage) - The Defacto Standard Open Source IDS.
  Snort is a network intrusion prevention and detection system utilizing a rule-driven language, which combines the benefits of signature, protocol and anomaly based inspection methods.

- **[[InstallingAgentThirdpartySuricata|Suricata]]** (Homepage) - Open Source IDS / IPS / NSM engine
  Suricata is a high performance Network IDS, IPS and Network Security Monitoring engine.

Prelude Components Configuration

[[PreludeManager|Prelude-Manager]]

[[PreludeCorrelator|Prelude-Correlator]]

[[PreludeLml|Prelude-LML]]

[[PreludeImport|Prelude-Import]]

Prelude Correlator

Prelude-Correlator is a Python rules based correlation engine. It has the ability to connect and fetch alerts from a remote Prelude-Manager server, and correlate incoming alerts based on the provided ruleset. Upon successful correlation, IDMEF correlation alerts are raised.

Initially, the Prelude-Correlator rule language was inspired by SEC, evolving to use a real programming language. At this point, we decided to switch to a Python based rules engine, which provides the great flexibility required for writing correlation rules.

Python-based rule format example

```
from preludecorrelator.context import Context
from preludecorrelator.pluginmanager import Plugin
```

```
class EventScanPlugin(Plugin):
    def run(self, idmef):
        source = idmef.get("alert.source(*).node.address(*).address")
        target = idmef.get("alert.target(*).node.address(*).address")

        if not source or not target:
            return

        for saddr in source:
            for daddr in target:
                ctx = Context(("SCAN EVENTSCAN", saddr, daddr), { "expire": 60, "threshold": 30, "alert_on_expire": True }, update
= True, idmef=idmef)
                if ctx.getUpdateCount() == 0:
                    ctx.set("alert.correlation_alert.name", "A single host has played many events against a single target. This may be a
vulnerability scan")
                    ctx.set("alert.classification.text", "Eventscan")
                    ctx.set("alert.assessment.impact.severity", "high")
```

Writing Python rules

In order to write Prelude-Correlator rules, you will need to learn basics of the Python language. Here are some good starting points for learning Python:

- Python website
- Python documentation
- Dive into Python

Each Python plugin consists of a primary Python class which will run upon reception of an IDMEF event.

```
from preludecorrelator.pluginmanager import Plugin

print("*** Any global initialization code goes here")

class MyPlugin(Plugin):
    def run(self, idmef):
        print("*** This function is going to be called when Prelude-Correlator receives an IDMEF event")
```

The primary run() function takes a single argument: the IDMEF message received by Prelude-Correlator.

When running Prelude-Correlator using this simple ruleset, here is what you will get:

```
$ prelude-correlator
*** Any global initialization code goes here
09 Feb 09:53:32 (process:14076) INFO: 10 plugin have been loaded.
09 Feb 09:53:32 (process:14076) INFO: Connecting to 127.0.0.1:4690 prelude Manager server.
09 Feb 09:53:32 (process:14076) INFO: TLS authentication succeed with Prelude Manager.

[And when Prelude-Correlator receive an IDMEF message]:

*** This function is going to be called when Prelude-Correlator receive an IDMEF event
```

Playing with received IDMEF messages

The IDMEF message object ('idmef' in the example above), passed to the primary plugin run() method, provides a number of methods:

The get() method

The get() method allow to retrieve a given field from the IDMEF message using an [[IDMEFPath|IDMEF Path]]. Example:

```
classification = idmef.get("alert.classification.text")
print("*** %s" % classification)
```

This will retrieve the **alert.classification.text** [[IDMEFPath|IDMEF Path]], and print it afterwise:

```
*** Portscan
```

Now let see how results are retrieved for more complex object (example: listed object):

```
sources = idmef.get("alert.source(*).node.address(*).address")
for addr in sources:
   print("*** %s" % addr)
```

This will retrieve the **alert.source(*).node.address(*).address** [[IDMEFPath|IDMEF Path]], iterate the returned value and print them one by one:

```
*** 192.0.2.50
*** 192.0.2.51
*** 192.0.2.52
```

The match() method

The *match* method is similar to the *Get* method described above, except it allows matching the value against a regular expression:

```
is_failed_auth = idmef.match("alert.classification.text", "(.+)")
```

The set() and **alert()** methods

The *set* method allows you to set [[IDMEFPath|IDMEF Path]] to a certain value within an IDMEF object. The *alert* method allow to send an IDMEF object to the Prelude-Manager concentrator.
This is mostly useful when generating CorrelationAlert. Here we will make a simple demonstration of how to use these methods:

```
from preludecorrelator.idmef import IDMEF

# create a new IDMEF object
idmef = IDMEF()
idmef.set("alert.classification.text", "My Classification")
idmef.alert()
```

If you run the above code in a Prelude-Correlator plugin, here is what you will get when Prelude-Correlator receives an IDMEF message:

```
$ prelude-correlator --print-output
25 Jun 14:00:38 (process:13208) INFO: 1 plugins have been loaded.
25 Jun 14:00:38 (process:13208) INFO: Connecting to 127.0.0.1:4690 prelude Manager server.
25 Jun 14:00:38 (process:13208) INFO: TLS authentication succeed with Prelude Manager.

[When Prelude-Correlator receive an IDMEF message]:

version: <empty>
alert:
        messageid: bd32c2a8-1559-11df-8f1a
        analyzer(0):
                analyzerid: 952281412762694
                name: prelude-correlator
                manufacturer: CS-SI
                model: Prelude-Correlator
                version: 1.2.6
                class: Correlator
                ostype: Linux
                osversion: 2.6.32-573.3.1.el6.x86_64
                process:
                        name: prelude-correlator
                        pid: 14286
                        path: /usr/bin/prelude-correlator
        create_time: 09/11/2015 18:26:39.66467 +01:00
        classification:
                text: My Classification
```

Python Context

A **Context** class is available, that greatly eases the writing of Python correlation rules. A Context is a named variable, which you can use to keep track of a given plugin state. They consist of:

- The context name, used to identify the context
- An embedded IDMEF message, that might be used to generate a CorrelationAlert
- An optional expiry timer (the context is destroyed on timer expiration)
- An optional threshold.

Creating a context:

```
ctx = Context("MY_CONTEXT_NAME")
```

Updating a context (create the context if it does not exist, or reset it's expiration timer if it does):

```
ctx = Context("MY_CONTEXT_NAME", update=True)
```

Additionally, several context creation options might be specified:

- **expire** : number of seconds after which the context should be destroyed.
- **alert_on_expire** : If a context expires and this option is set, the embedded IDMEF message will be sent.
- **threshold** : Internal threshold counter, that might be checked using the **getUpdateCount** method.

Example, creating (or updating) a context that will expire after 60 seconds, and with the threshold counter set to 30:

```
ctx = Context("MY_CONTEXT_NAME", { expire = 60, threshold = 30 }, update=True)
```

Setting values within the context embedded IDMEF object:

```
ctx.set("alert.classification.text", "My Correlation Alert")
```

Retrieving a context:

```
ctx = context.search("MY_CONTEXT_NAME")
```

Example: CorrelationAlert on EventStorm (playing excessive events by a single host)

Here is how to write a Python correlation plugin for EventStorm detection: we want to generate a CorrelationAlert if a single source is generating an excessive amount of events.

We initially start by retrieving a list of all the source addresses within the received alert:

```
source = IDMEF.get("alert.source(*).node.address(*).address")
```

We then iterate the sources list, and create (or retrieve - if it already exists) the *Context* associated with each source address.
The *Context* is set to expire after 120 seconds, and the internal threshold counter is set to 150.

If receiving an IDMEF message with the "x.x.x.x" source address, the created context will be named **SCAN_EVENTSTORM_x.x.x.x**.

```
for saddr in source:
    ctx = Context(("SCAN_EVENTSTORM", saddr), { "expire": 120, "threshold": 150, "alert_on_expire": True }, update = True,
idmef = idmef)
```

Notice that we provides the received IDMEF message to the Context creation/update method. This will append the received alert source and target to the Context embedded IDMEF message (which will become the generated CorrelationAlert).

This is equivalent to calling:

```
created_idmef_message = IDMEF()
created_idmef_message.addAlertReference(received_idmef_message)
```

After creating/updating the context, we check whether the context has been created (and not updated), and we initialize portion of the alert to be generated if this is the case :

```
if ctx.getUpdateCount() == 0:
```

```
ctx.set("alert.correlation_alert.name", "A single host is producing an unusual amount of events")
ctx.set("alert.classification.text", "Eventstorm")
ctx.set("alert.assessment.impact.severity", "high")
```

That's it. Here is what will happen with the above plugin (with a threshold of 3, so that the example is readable):

- Prelude-Correlator receives the first alert: [source address: 1.1.1.1 -> target address: 2.2.2.2]
  - The **SCAN_EVENTSTORM_1.1.1.1** context is created, setup to expire in 120 seconds, with an internal threshold of 3.
  - The **!getUpdateCount()** method returns 0, so we copy some of the current alert data into the Context embedded correlation alert.

- Prelude-Correlator receives the second alert: [source address: 1.1.1.1 -> target address: 3.3.3.3]
  - The **SCAN_EVENTSTORM_1.1.1.1** context is updated, the 120 seconds timer is reset.
  - The **!getUpdateCount()** method returns 1.

- Prelude-Correlator receives the third alert: [source address: 1.1.1.1 -> target address: 4.4.4.4]
  - The **SCAN_EVENTSTORM_1.1.1.1** context is updated, the 120 seconds timer is reset.
  - The **!getUpdateCount()** method returns 2.

At this point, we know a CorrelationAlert is going to be generated as soon as the internal context Timer will expire.

Here is the alert generated:

```
version: <empty>
alert:
        messageid: 12e8fe68-42bd-11dd-9544
        analyzer(0):
                analyzerid: 2052969743121519
                name: prelude-correlator
                model: prelude-correlator
                version: 1.2.6
                class: Correlator
                ostype: Linux
                osversion: 2.6.32-573.3.1.el6.x86_64

        create_time: 09/11/2015 18:26:39.66467 +01:00
        classification:
                text: Eventstorm
        source(0):
                spoofed: unknown (0)
                node:
                        category: unknown (0)
                        address(0):
                                category: ipv4-addr (7)
                                address: 1.1.1.1
        target(0):
                decoy: unknown (0)
                node:
                        category: unknown (0)
                        address(0):
                                category: ipv4-addr (7)
                                address: 2.2.2.2
        target(1):
                decoy: unknown (0)
                node:
                        category: unknown (0)
                        address(0):
                                category: ipv4-addr (7)
                                address: 3.3.3.3
        target(2):
                decoy: unknown (0)
                node:
                        category: unknown (0)
                        address(0):
                                category: ipv4-addr (7)
                                address: 4.4.4.4
```

```
assessment:
        impact:
                severity: high (4)
                type: other (0)
correlation_alert:
        name: A single host is producing an unusual amount of events
        alertident(0):
                alertident: 90704f76-42bd-11dd-8410
                analyzerid: bc-fs-sensor13
        alertident(1):
                alertident: 94a7b25a-42bd-11dd-ba64
                analyzerid: bc-fs-sensor13
        alertident(2):
                alertident: 9bc0e962-42bd-11dd-b088
                analyzerid: bc-fs-sensor13
```

Prelude Import

Prelude-Import is a commercial extension available from CS-SI. 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, Prelude-Import 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 Prelude-Import, 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
```

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 systems
  - 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 honeydr5711: 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.

- **idmef-alter** and **idmef-alter-force**

Recent PreludeLML version allow you to include static values into events generated using a given format:

```
[format=syslog]
idmef-alter = alert.analyzer(-1).node.location = My Location
```

Using the above example, any events generated from a syslog format source will have the *alert.analyzer(-1).node.location* [[IDMEFPath]] set to *My Location*, unless the path is already set. You might use the **idmef-alter-force** option in case you want to overwrite a path that is already set.

Metadata

In the default mode of operation, Prelude-LML will keep track of the last offset analyzed in a given log file. If Prelude-LML is restarted, it will start analyzing log files at their last known analyzed position.

You can overcome this behavior by using the 'metadata' option (available from the command line or the configuration file). This option might
take multiple arguments:

	Value		Description	
	tail		Analyze from the tail of the file	
	head		Analyze from the head of the file	
	last		Analyze from the last known file position (default)	
	nowrite		Do not write file metadata (prevent last keyword from working)	

Example: if you want to analyze a log from the beginning of the file, without writing any metadata to be used for resuming operation, you could start Prelude-LML with the following options:

```
prelude-lml --metadata=head,nowrite
```

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 different 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)!r10566: 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 necessary. Doing so will make your rule CPU-costly to implement.
- Avoid capturing variables which you don't use. This causes unnecessary memory consumption.
- At a minimum, include regex, **classification.text**, **assessement.impact.severity**, **assessment.impact.type**, **assessment.impact.description**.
- If it's correct for this application, use the *last* keyword.
- For readability, put only a single field on each line of your rules.
- **Include a sample log entry with each rule. This is very important for regression testing.**
- 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.text 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.

Rule format

An LML rules consist of a regular expression, used to match a given log entry, followed by [[IDMEFPath]] assignement (that might use value captured from the regular expression) in order to create an alert:

```
regex=your_regex; idmef.path = value; idmef.path2 = value2;
```

Here is a working example:

```
#LOG:Dec  8 14:45:17 itguxweb1 sshdr32112: Accepted publickey for root from 12.34.56.78 port 56634 ssh2
#LOG:Jan 14 03:30:44 mail sshdr20298: Accepted publickey for root from fec0:0:201::3 port 63018 ssh2

regex=Accepted (\S+) for root from (\S+) port (\d+); \
 classification.text=Admin login; \
 id=1908; \
 revision=3; \
 analyzer(0).name=sshd; \
 analyzer(0).manufacturer=OpenSSH; \
 analyzer(0).class=Authentication; \
 assessment.impact.severity=medium; \
 assessment.impact.completion=succeeded; \
 assessment.impact.type=admin; \
 assessment.impact.description=Root logged in from $2 port $3 using the $1 method; \
 source(0).node.address(0).address=$2; \
 source(0).service.port=$3; \
 source(0).service.iana_protocol_name=tcp; \
 source(0).service.iana_protocol_number=6; \
 target(0).service.port=22; \
 target(0).service.name=ssh; \
 target(0).service.iana_protocol_name=tcp; \
 target(0).service.iana_protocol_number=6; \
 target(0).user.category=os-device; \
 target(0).user.user_id(0).type=target-user; \
 target(0).user.user_id(0).name=root; \
 additional_data(0).type=string; \
 additional_data(0).meaning=Authentication method; \
 additional_data(0).data=$1; \
 last;
```

Prelude-LML can accept any IDMEF field in the form of an [[IDMEFPath]]. Below, you will find some example of IDMEF fields.

For listed IDMEF field (annotated /plugin_assets/redmine_wiki_extensions/images/x_mark.png above), 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(x).origin**: The type of reference, permitted values are: unknown, vendor-specific, user-specific, bugtraqid, cve, osvdb
- **classification.reference(x).name**: Exactly one string containing the name of the reference from the source.
- **classification.reference(x).meaning**: A brief manager (or the human operator of the manager) description of the alert
- **classification.reference(x).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(x).node.address(y).address**, **target(x).node.address(y).address**: Address that has been attacked/Address that issued the attack. There can be more than one.
- **source(x).node.address(y).category**, **target(x).node.address(y).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(x).node.address(y).vlan_name**, **target(x).node.address(y).vlan_name**: The name of the Virtual LAN to which the address belongs.
- **source(x).node.address(y).vlan_num**, **target(x).node.address(y).vlan_num**: The number of the Virtual LAN to which the address belongs.
- **source(x).node.name**, **target(x).node.name**: The name of the equipment. This information MUST be provided if no Address information is given.
- **source(x).node.category**, **target(x).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(x).node.location**, **target().node.location**: The location of the equipment.
- **source(x).spoofed**, **target(x).decoy**: An indication of whether the source/target is a decoy. The permitted values are: *unknown*, *yes*, *no*.
- **source(x).interface**, **target(x).interface**: May be used by a network-based analyzer with multiple interfaces to indicate which interfaces this source/target was seen on.
- **source(x).service.name**, **target(y).service.name**: The name of the service. Whenever possible, the name from the IANA list of well-known ports SHOULD be used.
- **source(x).service.port**, **target(x).service.port**: The port number being used.
- **source(x).service.iana_protocol_name**, **target(x).service.ianal_protocol_name**: The protocol being used.
- **source(x).service.portlist**, **target(x).service.portlist**: A list of port numbers being used.
- **source(x).user.category**, **target(y).user.category**: The type of user represented (*unknown*, *application*, *os-device*).
- **source(x).user.user_id(y).type**, **target(x).user.user_id(y).type**: The type of user information represented (*current-user*, *original-user*, *target-user*, *user-privs*, *current-group*, *group-privs*, *other-privs*).
- **source(x).user.user_id(y).name**, **target(x).user.user_id(y).name**: A user or group name.
- **source(x).user.user_id(y).number**, **target(x).user.user_id(y).number**: A user or group number.
- **source(x).process.name**, **target(x).process.name**: A process name
- **source(x).process.pid**, **target(x).process.pid**: A process PID.

Rule flow control

- **last**: if a rule using the *last* keyword is matched, Prelude-LML won't process further rules.
- **silent**: if a rule using the *silent* keyword is matched, Prelude-LML will remain silent and won't generate an alert.
- **chained**: Rule marked as chained can only be called from other rules, using the **goto** or **optgoto** keyword.
- **goto**: allow to call a specific rule from the currently matched rule: **goto** = *rule_id*. If the rule that is called doesn't match, the current rule will fail.
- **optgoto**: Same as goto, but the called rule remain optional.
- **optregex**: Allow to specify an optional regex.

- **min-optgoto-match**: Require that at least the specified number of **optgoto** match for the current rule to be considered as a match.
- **min-optregex-match**: Same as *min-optgoto-match*, but for **optregex**.

Prelude Manager

Table of Contents

Introduction

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). It is **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.

You can customize the number of second Prelude-Manager wait for an incoming client to successfully authenticate before dropping a connection. The default value is 10 seconds.

```
connection-timeout = 10
```

Prelude-Manager Plugins

Prelude-Manager provides **three Plugin Categories** in order to carry out different actions:

Reporting Plugins

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

A number of Reporting Plugins are available:

- **[[PreludeManagerReportingDb|db]]** - A database Plugin (MySQL and PostgreSQL).
- **xmlmod** - An XML Reporting Plugin.
- **textmod** - A text Reporting Plugin.
- **[[PreludeManagerReportingRelaying|relaying]]** - A plugin relaying alert to another set of manager.
- **[[PreludeManagerReportingSmtp|smtp]]** - Send textual alert through your SMTP server.
- (check the *prelude-manager --help* output for others)

In order to learn how to use Reporting Plugins, **see the [[PreludeManagerReportingPlugins|Reporting Plugins Page]]**

Filtering Plugins

Filtering events, event suppression and Thresholding

Two filtering plugins are available:

- **IDMEF Criteria Filtering Plugin** - Filtering events
- **Thresholding Filtering Plugin** - Event suppression and thresholding

In order to learn how to use Filtering Plugins, **see the [[PreludeManagerFilteringPlugins|Filtering Plugins Page]]**

Normalization Plugins

For each incoming events, Prelude-Manager will run a number of normalization routine: sanitize address, services information, etc.

In order to customize the normalization process, **see the [[PreludeManagerNormalizationPlugins|Normalization Plugins Page]]**

Scheduler options

On systems with many concurrent sensors sending events to Prelude-Manager, Prelude-Manager might have an hard time keeping up with the demand for events reporting.

The Prelude Manager scheduler allocate reporting time slot per sensor, allowing to define the maximum number of events processed for one sensor before processing others sensors events (in case a sensor is sending a continuous events burst, this prevent other sensors starvation).

By default, for each sensor connected, a maximum of 100 events will be processed before processing others sensors events.

Additionally, priority will be given to events depending on their priority. Assuming there is enough events of each priority, 50 high priority message will be processed, 30 medium, and 20 low (totalling the maximum of 100 described above).

```
sched-priority = high:50 medium:30 low:20
```

You can define the maximum amount of reserved memory for storing incoming events before they are processed. When the number of events waiting to be processed exceed this amount of reserved memory (default is 1 Megabyte), Prelude-Manager will start storing events on disk. You can customize the amount of reserved memory using the **sched-buffer-size** option.

```
sched-buffer-size = 1M
```

TLS options

On system using GnuTLS 2.2.0 or later, you might customize a number of TLS options, including setting available ciphers, key exchange methods, macs and compression methods.
Predefined sets of ciphersuites:

- **NORMAL** option enables all "secure" ciphersuites, 256-bit ciphers included.
- **SECURE128** flag enables all "secure" ciphersuites with ciphers up to 128 bits.
- **SECURE256** flag enables all "secure" ciphersuites including the 256 bit ciphers.
- **EXPORT** all the ciphersuites are enabled, including the low-security 40 bit ciphers.
- **NONE** nothing is enabled. This disables even protocols and compression methods.

Note that much more options might be enabled or disabled using this setting: please see gnutls_priority_init(3) for more details. The default value for **tls-option** is **NORMAL**.

```
tls-options = NORMAL
```

You can customize the Diffie-Hellman parameters, for use with the Prelude-Manager server.

The number of bits of the prime might be defined using the **dh-prime-length** option. Note that the value should be one of 768, 1024, 2048, 3072 or 4096. The default is 1024.

```
dh-prime-length = 1024
```

How often to regenerate the parameters can be defined using the **dh-parameters-regenerate** option. These should be discarded and regenerated once a day, once a week or once a month, depending on the security requirements.

Generation is a CPU intensive operation. The value is in hours, 0 disables regeneration entirely. The default is 24 hours.

```
dh-parameters-regenerate = 24
```

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.

TCP/IP related options

Using the global prelude section, where you can define Prelude related options, you can define option of matter for Prelude-Manager, and most specifically in the the context of relaying, TCP/IP options that influence the behavior of when the operating system should consider a connection dead in case sent data is left unacknowledged.

Theses option are operating system specific, and might not work on certain platform. In case you modify these settings on an unsupported system, a warning message will be issued when the agent starts.

- **tcp-keepalive-time** represents the number of seconds the connection needs to be idle before TCP begins sending out keep-alive probes.
- **tcp-keepalive-probes** represent the number of not acknowledged probes to send before considering the connection dead.
- **tcp-keepalive-intvl** represents the interval between subsequent keepalive probes.

Under Linux, the default system wide configuration is:

```
[prelude]
tcp-keepalive-time   = 7200
tcp-keepalive-probes = 9
tcp-keepalive-intvl  = 75
```

The average time to notice a dead connection can be calculated using: *tcp-keepalive-time* + (*tcp-keepalive-probes* * *tcp-keepalive-intvl*).
Under Linux, the default settings thus instruct that a dead connection will be dropped after **7875** seconds.

This might be a problem if a network outage prevent relayed events from reaching their target, in this case, the Operating System won't report the dead connection before 7875 seconds elapse, and all data transmitted during this period will be lost because the failover is not yet activated.

You can improve this behavior using your own customized settings:

```
[prelude]
tcp-keepalive-time   = 60
tcp-keepalive-probes = 3
tcp-keepalive-intvl  = 10
```

Using the above settings, a dead connection will be detected within 90 seconds.

Files

prelude_handbook_09.pdf.tar.gz	243 KB	10/24/2007	sebastian-roschke-hpi-uni-pots -
prewikka-overview.png	248 KB	12/21/2008	anonymous -
Prewikka_User_Manual.pdf	256 KB	01/22/2009	anonymous -
prewikka-overview.png	197 KB	09/08/2015	Thomas ANDREJAK
simple-architecture.png	20.9 KB	08/13/2008	anonymous -
architecture.png	46.7 KB	09/08/2015	Thomas ANDREJAK
reverse-relaying.png	25.5 KB	09/08/2015	Thomas ANDREJAK
framework.png	12.1 KB	09/11/2008	anonymous -
PreludeAnt.png	69 KB	09/11/2008	anonymous -