

Security Information Management

b-i branding. technology. integration.

Acronyms

Main acronyms used in this talk :

- **IDS** : Intrusion Detection System, commonly divided in
 - > **NIDS** : Network Intrusion Detection System works at network level.
 - > **HIDS** : Host-based Intrusion Detection System works at machine level by checking FS Integrity or monitoring logs
- **IDMEF** : Intrusion Detection Message Exchange Format, XML based format used by sensors to exchange data with the SIM (RFC 4765)
- **SIM** : Security Information Management, a software collecting events from security sensors to help users make sense of them.

Prelude SIM

- Open Source Project
- Can **interoperate** with **NIDS** and **HDIS** using **IDMEF**
- **Real-time correlation & visualization** of security events and attacks scenarii in a **centralized console**
- **Secured Architecture** : Redondancy, encrypted communications with sensors, works with networks restricted policy (DMZ..)



Prelude Components

- **Prelude Manager** : high availability server that collects events from deployed sensors
- **Prelude LML** : collects and normalize logs
- **Prelude Correlator** : multistream correlation engine written in python
- **Prewikka** : web vizualization interface written in python
- **libprelude** : provides features to inject events inside Prelude in various languages such a C++, Perl, Python, Ruby
- **libpreludedb** : for transparent DB access.

Prelude LML (Logs Monitoring Lackey)

- Prelude LML is compatible with a large numbers of log files :
 - > System Logs
 - > Syslog
 - > Apache
 - > Postfix
 - > PAM
 - > Cisco IPS, VPN, Routers
 - > Sudo
 - > Nagios
 - > Netfilter
 - > OpenSSH
 - > and a lot more ...

- Very extensible to custom applications by writing new rulesets

Prelude LML ruleset example

Identify requests that generates 404 errors in apache logs :

```
#LOG:[Sat Apr 16 14:30:12 2005] [error] [client :::1] File does not exist: /var/www/favicon.ico
regex=\[error\] \[client ([A-Za-f\d:]+\)] ((File|Premature|Directory|client|request)[\S+\s]+): (.+); \
classification.text=Web server error; \
id=4101; \
revision=1; \
analyzer(0).name=httpd; \
analyzer(0).manufacturer=www.apache.org; \
analyzer(0).class=Service; \
assessment.impact.severity=low; \
assessment.impact.completion=succeeded; \
assessment.impact.type=other; \
assessment.impact.description=Apache httpd '$2' error: '$4'; \
source(0).node.address(0).category=ipv4-addr; \
source(0).node.address(0).address=$1; \
source(0).service.iana_protocol_name=tcp; \
source(0).service.iana_protocol_number=6; \
target(0).service.iana_protocol_name=tcp; \
target(0).service.iana_protocol_number=6; \
target(0).service.name=http; \
last;
```

Prelude sensors

Some programs have a built-in Prelude compatibility :

- **Snort** : well-known lightweight IDS
- **Samhain** : provides file integrity checking
- **Nessus** : vulnerability scanner
- **Nepenthes** : honeypot
- **Auditd** : the Linux audit daemon
- **Prelude-LML** : the prelude log analyzer

libprelude provides an **easy** way to build **customs sensors** that can talk with an existing prelude architecture.

Secured sensors registration (OTP, SSL...) with the Prelude Manager

Prelude custom sensor example

```
#!/usr/bin/python
import PreludeEasy
Client = PreludeEasy.ClientEasy("My beautiful sensor")
client.Start()
idmef = PreludeEasy.IDMEF()
idmef.Set("alert.classification.text" , "There is a problem" )
idmef.Set("alert.source(0).node.address(0).address" , "192.168.1.65")
idmef.Set("alert.target(0).node.address(0).address" , "192.168.1.1")
client << idmef
print ( idmef )
```

Intrusion Detection Message Exchange Format

IDMEF message 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>
    <Alert>
      <Analyzer model="myids"/>
      <Target>
        <Node>
          <name>mynode</name>
        </Node>
      </Target>
      <AdditionalData meaning="data2" type="string">value2</AdditionalData>
      <AdditionalData meaning="data1" type="string">value1</AdditionalData>
    </Alert>
  </IDMEF-Message>
```

Prelude Correlator

- As it can be easy to fool one security system, it becomes very difficult to fool multiple systems.
- Prelude will help to identify an attack by correlating all the events in real-time
- Improve events presentation by reducing false-positives
- Can aggregate various events into a readable sequence (multiple connections on various ports ...)
- Can identify unknown attacks
- Plugins support to improve correlation by adding system independant rules (ie : connections hours, already identified IP by other systems...)

Correlator Plugin Example – Business Hours

```
#!/usr/bin/python

import time
from PreludeCorrelator.idmef import IDMEF
from PreludeCorrelator.pluginmanager import Plugin

# By default, alert only on saturday and sunday, and everyday from 6:00pm to 9:00am.

class BusinessHourPlugin(Plugin):
    BUSINESSHOUR_HWORKSTART = 9
    BUSINESSHOUR_HWORKEND = 17
    BUSINESSHOUR_OFFDAYS = '[5,6]'

    def __init__(self,env):
        Plugin.__init__(self, env)

        self.__hworkstart = int(self.getConfigValue("hworkstart", self.BUSINESSHOUR_HWORKSTART))
        self.__hworkend = int(self.getConfigValue("hworkend", self.BUSINESSHOUR_HWORKEND))
        self.__offdays = eval(self.getConfigValue("offdays", self.BUSINESSHOUR_OFFDAYS))

    def run(self, idmef):
        self.__t = time.localtime(int(idmef.Get("alert.create_time")))

        if not (self.__t.tm_wday in self.__offdays or self.__t.tm_hour < self.__hworkstart or self.__t.tm_hour > self.__hworkend):
            return

        if idmef.Get("alert.assessment.impact.completion") != "succeeded":
            return

        ca = IDMEF()
        ca.addAlertReference(idmef)
        ca.Set("alert.classification", idmef.Get("alert.classification"))
        ca.Set("alert.correlation_alert.name", "Critical system activity on day off")

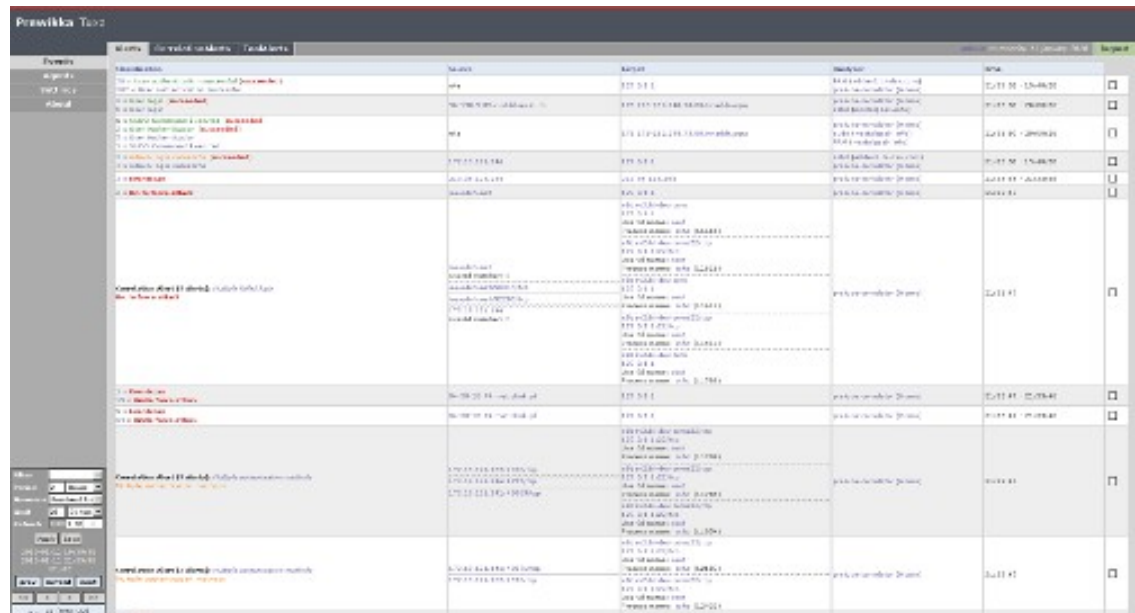
        ca.alert()
```

Prelude Manager

- Collects all events in real-time from all deployed sensors using IDMEF
- Support secured architectures such as DMZ, using reverse-relaying
- Redondancy
- Encrypted communications with sensors (SSL)
- Support of various databases (MySQL, PostgreSQL)

Prewikka

- Written in Python, improved UI using JQuery
- Events displayed in real-time
- Screen to monitor sensors status
- Asynchronous DNS resolution
- Advanced lists filtering



The screenshot displays the Prewikka web interface, which is a monitoring tool for sensors. The interface features a dark-themed header with the title 'Prewikka - Tab 2' and navigation tabs for 'Alarms', 'Devices', 'Sensors', and 'Tasks'. Below the header, there is a table listing various sensor events. The table has columns for 'Device', 'Status', 'Action', 'Priority', and 'Time'. The 'Status' column contains various sensor readings and error messages, such as 'Temperature sensor (1) alarm: multiple sensors multiple' and 'Temperature sensor (2) alarm: multiple sensors multiple'. The 'Action' column shows the type of event, such as 'Alarm triggered' or 'Sensor status change'. The 'Priority' column indicates the severity of the event, with values like 'High', 'Medium', and 'Low'. The 'Time' column shows the timestamp of the event, such as '2011-11-15 10:00:00'. On the left side of the interface, there is a sidebar with a search bar and several buttons for filtering and managing the data.

Device	Status	Action	Priority	Time
Temperature sensor (1)	Temperature sensor (1) alarm: multiple sensors multiple	Alarm triggered	High	2011-11-15 10:00:00
Temperature sensor (2)	Temperature sensor (2) alarm: multiple sensors multiple	Alarm triggered	High	2011-11-15 10:00:00
Temperature sensor (3)	Temperature sensor (3) alarm: multiple sensors multiple	Alarm triggered	High	2011-11-15 10:00:00
Temperature sensor (4)	Temperature sensor (4) alarm: multiple sensors multiple	Alarm triggered	High	2011-11-15 10:00:00
Temperature sensor (5)	Temperature sensor (5) alarm: multiple sensors multiple	Alarm triggered	High	2011-11-15 10:00:00
Temperature sensor (6)	Temperature sensor (6) alarm: multiple sensors multiple	Alarm triggered	High	2011-11-15 10:00:00
Temperature sensor (7)	Temperature sensor (7) alarm: multiple sensors multiple	Alarm triggered	High	2011-11-15 10:00:00
Temperature sensor (8)	Temperature sensor (8) alarm: multiple sensors multiple	Alarm triggered	High	2011-11-15 10:00:00
Temperature sensor (9)	Temperature sensor (9) alarm: multiple sensors multiple	Alarm triggered	High	2011-11-15 10:00:00
Temperature sensor (10)	Temperature sensor (10) alarm: multiple sensors multiple	Alarm triggered	High	2011-11-15 10:00:00

References

- Prelude IDS : <http://www.prelude-ids.org>
- Prelude Development Site : <https://dev.prelude-ids.org>
- IDMEF Standard : <http://www.rfc-editor.org/rfc/rfc4765.txt>
- Iptables / Netfilter : <http://www.netfilter.org>

Questions

?

?

thank you

alexandre.dedommelin@b-i.com
t: + 41.58.307.6603

b-i branding. technology. integration.