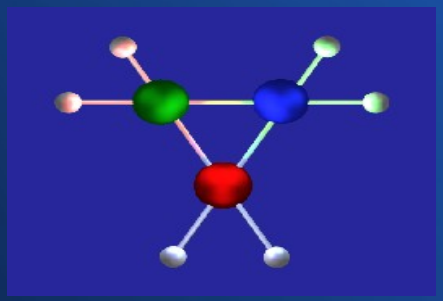
# LUARM AND ITPSL
# Sensing and specifying Insider Threats



*George Magklaras*
*Center for Security, Communications and Network Research*
*University of Plymouth*
*http://www.cscan.org*

***http://luarm.sourceforge.net/***

# Agenda

Point 1: Insider threat specification and its requirements

Point 2: Forensics to aid insider threat mitigation

Point 3: LUARM: A tool to create insider threat data repositories

Point 4: ITPSL: A tool to specify threats by mining insider threat data repositories

# Insiders (visually)


# Defining the “insider”

“An insider is a person that has been legitimately empowered  with the right to access, represent, or decide about one or more assets of the organization's infrastructure.”

http://www.dagstuhl.de/08302

# Defining “Insider Threat Specification”

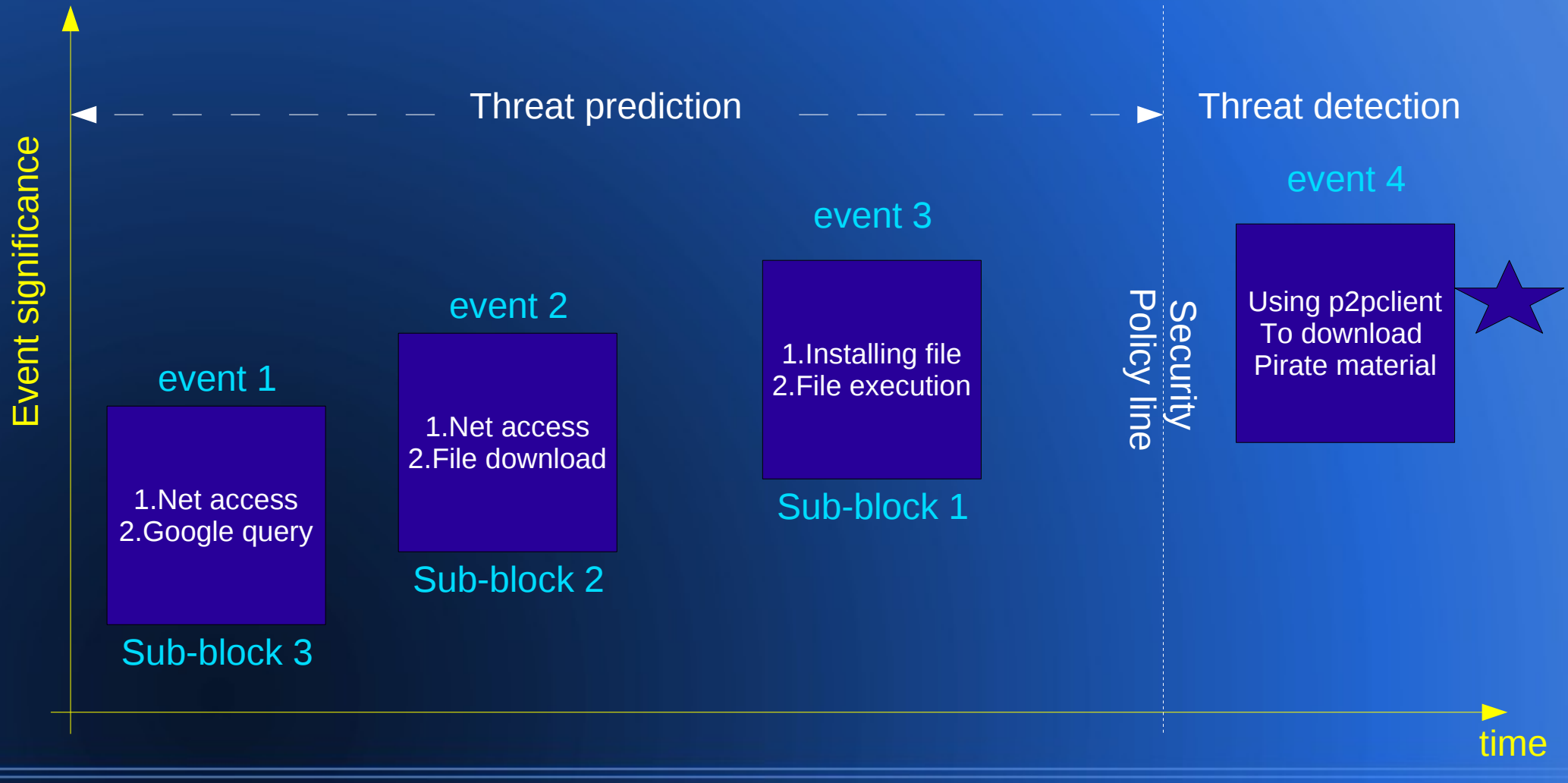
Insider Threat Specification is the process of using a standardized vocabulary to describe in an abstract way how the aspects and behavior of an insider relate to a security policy defined misuse scenario.

# Notes on the Insider Threat Specification Definition

- Standardized vocabulary: Taxonomies and ontologies of the research literature
- Aspects: character, personality, organizational role, financial status
- Behavior: The actions of an individual in relation to accessing, representing or deciding about organizational assets.
- Threat relation:

Concerns the execution of the threat (threat detection)
Concerns signs of the threat (threat prediction)

# Temporal dimension of an Insider Threat



# System-level Insider Threat Specification

- The previous definitions are wide in scope.
- To construct an automated threat detection/prediction system, we need to narrow down the observable aspect and behavioral data.
- Only system obtained data are considered at filesystem, process execution and network connection levels.

# System-level Insider Threat Logging wishlist

- We need a standardized way to monitor and deposit user actions: OS agnostic and log records should have a well defined format.

- Data should be stored away from the monitored host for security purposes (integrity and availability of log data)

- The record format should allow user entity accountability for each recorded insider action.

# System-level Insider monitoring and Forensics

Should a logging engine complement forensics: Yes. Why?

- The “observer effect”: No need to tamper with investigation source media [1].

-”Static” data forensic analysis can give a rather incomplete picture of an incident [1].

-”Dynamic” data forensic analysis (sequence of process events) can be built more easily in a logging engine rather than an OS forensic tool [2].

# Overview of existing logging engines

There are many logging engines/frameworks and Security Event Managers (SEMs) out there. A sample:
- Syslogd[3], WinSyslog[4], RFC 5424
- OpenXDAS [5], Cisco MARS[6]
- Event Data Warehouse [7], Arc Sight Logger 4 [], EDP

Most of these solutions are geared towards network and application security events and/or data audit compliance.

They do not really address the insider threat detection and prediction issues to a detailed extent.

# LUARM

-**L**og **U**ser **A**ctions in **R**elational **M**ode

-Written in Perl for rapid prototyping and Open Source.

-Uses MySQL to store the logs in a simple schema.

-Goal: Provide a prototype log engine for insider misuse researchers so that they are:

- able to log user actions in detail.
- able to use the logs to replay/study misuse incidents.
- cross reference logged user data to forensic procedures.

# LUARM architecture

netactivity.pl
fileactivity.pl
psactivity.pl
hwactivity.pl

Client 1

netactivity.pl
fileactivity.pl
psactivity.pl
hwactivity.pl

Client 2

Perl DBI

Perl DBI

mysqld
endpointresolver.pl
ITPSL Compiler and Client registration modules

Server

LUARM Client 1 Database

LUARM Client 2 Database

# LUARM relational schema

| fileaccessid | bigint |
|---|---|
| md5sum | text |
| filename | varchar |
| location | varchar |
| username | tinytext |
| application | text |
| fd | tinytext |
| pid | int |
| size | bigint |
| cyear | int |
| cmonth | tinyint |
| cday | tinyint |
| chour | tinyint |
| csec | tinytint |
| dyear | int |
| dmonth | tinyint |
| dday | tinyint |
| dhour | tinyint |
| dmin | tinyint |
| dsec | tinyint |

fileinfo table

| endpointinfo | bigint |
|---|---|
| md5sum | text |
| transport | tinytext |
| sourceip | tinytext |
| sourcefqdn | tinytext |
| destip | tinytext |
| destfqdn | tinytext |
| sourceport | smallint |
| destport | smallint |
| ipversion | smallint |
| cyear | int |
| cmonth | tinyint |
| cday | tinyint |
| chour | tinyint |
| csec | tinytint |
| dyear | int |
| dmonth | tinyint |
| dday | tinyint |
| dhour | tinyint |
| dmin | tinyint |
| dsec | Tinyint |
| usermame | tinytext |
| pid | int |
| application | text |

netinfo table

| psentity | bigint |
|---|---|
| md5sum | text |
| username | tinytext |
| pid | smallint |
| ppid | smallint |
| pcpu | decimal |
| pmem | decimal |
| command | text |
| arguments | mediumtext |
| cyear | int |
| cmonth | tinyint |
| cday | tinyint |
| chour | tinyint |
| csec | tinytint |
| dyear | int |
| dmonth | tinyint |
| dday | tinyint |
| dhour | tinyint |
| dmin | tinyint |
| dsec | Tinyint |
| usermame | tinytext |
| pid | int |

psinfo table

# LUARM relational schema (2)

| hwdevd | bigint |
|---|---|
| md5sum | text |
| devbus | tinytext |
| devstring | text |
| devvendor | text |
| application | text |
| userslogged | text |
| cyear | int |
| cmonth | tinyint |
| cday | tinyint |
| chour | tinyint |
| csec | tinytint |
| dyear | int |
| dmonth | tinyint |
| dday | tinyint |
| dhour | tinyint |
| dmin | tinyint |
| dsec | tinyint |

hwinfo table

| netintid | bigint |
|---|---|
| md5sum | text |
| ipversion | tinyint |
| ip | tinytext |
| subnet | tinytext |
| macaddr | tinytext |
| intname | text |
| cyear | int |
| cmonth | tinyint |
| cday | tinyint |
| chour | tinyint |
| csec | tinytint |
| dyear | int |
| dmonth | tinyint |
| dday | tinyint |
| dhour | tinyint |
| dmin | tinyint |
| dsec | tinyint |

netint table

| grouptid | bigint |
|---|---|
| md5sum | text |
| groupusers | text |
| myear | int |
| mmonth | tinyint |
| mday | tinyint |
| mhour | tinyint |
| msec | tinyint |
| cyear | int |
| cmonth | tinyint |
| cday | tinyint |
| chour | tinyint |
| csec | tinytint |
| dyear | int |
| dmonth | tinyint |
| dday | tinyint |
| dhour | tinyint |
| dmin | tinyint |
| dsec | tinyint |

groupinfo table

# LUARM query examples

**-Find all accesses of the file 'prototype.ppt' by users 'toms' OR 'georgem' between 9:00 and 14:00 hours on 23/10/2009.**

```
SELECT * FROM fileinfo WHERE filename='prototype.ppt' AND ((username='toms') OR (username='georgem')) AND cyear='2009' AND cmonth='10' AND cday='23' AND chour >= '9' AND chour <= '13' AND cmin >= '0' AND cmin >= '59';
```

**-Find all USB devices that were physically connected to the system when users 'toms' OR 'georgem' were logged on 23/10/2009.**

```
SELECT * from hwinfo WHERE devbus='usb' AND ((userslogged RLIKE 'toms') OR (userslogged RLIKE 'georgem')) AND cyear='2009' AND cmonth='10' AND cday='23' AND chour >= '9' AND chour <= '13' AND cmin >= '0' AND cmin >= '59';
```

# Audience alertness test 1

**-What does the following do (hint: psinfo is the process execution table) and what does the sequence of actions of the examples specify?**

```
select * FROM psinfo WHERE ((command='cp') OR (command='mv')) AND (arguments RLIKE 'prototype.ppt' AND arguments RLIKE '/media') AND ((username='georgem') OR (username='toms')) AND cyear='2009' AND cmonth='10' AND cday='23' AND chour >= '9' AND chour <= '13' AND cmin >= '0' AND cmin >= '59';
```

# LUARM deployment hardware specs

-MySQL LUARM server:

-4 Gbytes of RAM and 4 processing cores

-Disk space consumption in Gigabytes

$$D_{cons} = n_{clients} \times 18 \times d_{archive}$$

Example: 150 clients for 365 days of archiving ~ 1 Tbyte

-Data network: At least 100 Mbits/sec, maximum 20 Kbits/sec per client.

-LUARM client:

-2 processing cores and up to 300 Megs of RAM

-Up to 30% of a single core on a moderately busy system.

# LUARM issues/questions

-SQL is workable but not ideal (clarity, expressive compactness) for issuing event specific queries.

-How do we assemble queries together (temporal specification , correlation of events and combination of misuse and anomaly detection)?

-How do we increase the 'polymorphism' of the event expression schema?

-How do we relate the recorded events to decision theoretic information?

# Meet ITPSL

-**I**nsider **T**hreat **P**rediction and **S**pecification **L**anguage

-XML Domain Specific Language (DSL) construct made to address the LUARM issues/questions.

-LUARM collects the data and ITPSL mines the events.

-LUARM also facilitates threat signature repositories. Each signature specifies a threat scenario together with associated weight (confidence) data about the threat specifiers.

-Work in progress: Some of the specs mentioned here might change.

# ITPSL Header

**<itpslheader>**
**<signid>** md5sum (date and second, type of OS, current number of processes) **</signid>**
**<signdate>**
**<year>** dddd **</year> <month>** dd **</month> <day>** dd **</day>**
**</signdate>**
**<ontology>**
**<reason>** “intentional” | “accidental” **</reason>**
**<revision>** d.d **</revision>**
**<user_role>** “admins” | “advanced_users” | “ordinary_users” **</user_role>**
**<detectby>** “file” | “exec” | “network” | “multi” **</detectby>**
**<context>** detection | prediction **</context>**

**<weightmatrix>** $n_{events}$, $w_{event1}$,$w_{event2}$,...,$w_{eventn}$ **</weightmatrix>**

**<os>** “linux” | “windows” | “macosx” | “unix” **</os>**
**<osver>** “2.4” | ”2.6” | “2000” | “Vista” | “7” **</osver>**
**<threatkeywords>** keyword1 keyword2 ... keyword5 **</threatkeywords>**
[ **<synopsis>** “text that describes the signature’s purpose and function”
**</synopsis>**]
**</ontology>**
**</itpslheader>**

# ITPSL Header (2)

-Signature metadata.

-The ontology is the foundation for the signature taxonomy.

-An event is specified by an ITPSL sub-block (see latter slides)

-The **weightmatrix** tag facilitates decision theoretic information representation by means of event confidence weights:

$$\sum w_{eventn} = EPMO$$

EPTO -> Evaluated Potential Misuse Occurrence (0...1)
n-> number of specified events

# ITPSL body

```
<itpslbody>
    <mainblock>
        <mainop> AND | OR | XOR | as_a_result_of | justone </mainop>

        <subblock>
            <subop>  AND | OR | XOR | as_a_result_of | single  </subop>
            ITPSL directives
            ....
        </subblock>

        <subblock>
            <subop>  AND | OR | XOR | as_a_result_of | single  </subop>
            ITPSL directives
            ....
        </subblock>

    </mainblock>
</itpslbody>
```

# ITPSL runtime scopes

-Four language runtime scopes:

-**Header**: Concerns the header data.

-**Mainblock**: Concerns how the subblock data will be used.

-**Subblock**: How the ITPSL directives inside a subblock will be used.

-**ITPSL directive**: The specified file, network and process execution events.

-Runtime evaluation/parsing is performed on a bottom-up fashion (LR)
: ITPSL directive->Subblock->Mainblock.

# ITPSL 'mainop' operator

-**'mainop'** increases the language expressiveness/specificity for describing groups of actions (one action per subblock) :

-Marked by the **<mainop></mainop>** tags.
-Dictates how will the results of subblocks be combined/intepreted:
-AND | OR | XOR: Requires more than one subblock and combines them in terms of the binary operator (threat detection plus threat prediction). [9]
-as_a_result_of: Requires more than one subblock and is used to define a target set of actions (top subblock) and intermediate earlier stages (definition of abstract temporal sequence for threat detection plus threat prediction) . [9]
-justone: Requires just one subblock for the description of detecting a target state (threat detection).

# 'as_a_result_of' (mainblock scope)

```
<itpslbody>
    <mainblock>
        <mainop> as_a_result_of </mainop>

        <subblock>
            <subop> OR </subop>
            ITPSL directive1 ITPSL directive 2
        </subblock>

        <subblock>
            <subop> AND </subop>
            ITPSL directive1 ITPSL directive 2
        </subblock>

        <subblock>
            <subop> AND </subop>
            ITPSL directive1 ITPSL directive 2
        </subblock>

    </mainblock>
</itpslbody>
```

**TARGET (FINAL) CONDITION**

Middle temporal sequence

**INITIAL CONDITION**

# ITPSL 'subop' operator

-'**subop**' increases the language expressiveness/specificity for describing groups of file, network and process execution ITPSL directives within a subblock :

-Marked by the **<subop></subop>** tags.
-Dictates how will the ITPSL directives inside a subblock will be combined/intepreted:
-AND | OR | XOR | NOT: Requires more than one directive and combines them in terms of the binary operator. [9]
-as_a_result_of: Requires more than one directive in the block and is used to define a set of directives and intermediate in temporal sequence. [9]
-single: Requires a single directive in the subblock.

# 'as_a_result_of' (subblock scope)

```
<itpslbody>
    <mainblock>
        <mainop>  AND  </mainop>

        <subblock>
            <subop>  as_a_result_of   </subop>

            ITPSL directiven
            ITPSL directive n-1
            ....
            ITPSL directive 1
        </subblock>

        ...

        <subblock>
            <subop>  AND   </subop>
            ITPSL directive1 ITPSL directive 2
        </subblock>

    </mainblock>
</itpslbody>
```

TARGET (FINAL) CONDITION

INITIAL CONDITION

# The ITPSL directives

-Each ITPSL directive describes a discrete event related to a threat scenario. They can exist only inside an ITPSL subblock.

-Broadly divided into four categories:

- -**File directives**: Describe various file related events.
- -**Network directives**: Describe the presence of network endpoints and interfaces.
- -**Process Execution directives**: Express events related to program execution.
- -**Hardware operation statements**: Detect the addition or removal of hardware devices on the system.

# ITPSL file directives

-**File presence**: Detect files and dirs now.
  -fileexists
  -direxists

-**File access ability**: Examining the ability of users to access files
  -usercanaccessfile, usercanaccessdir
  -groupcanaccessfile, groupcanaccessdir

-**File access**: Examining the actual file access
  -fileaccess
  -diraccess

# ITPSL network directives

-**Network element detection**: Existence of interfaces and routes now.
    -netinterfaceexists
    -routeexists

-**Network access ability**: Can users access endpoints?
    -usercanaccessnet
    -groupcanaccessnet

-**Network access**: Checking for actual endpoint access.
    -netaccess

# ITPSL process execution directives

-**General process execution**: Running a program without reference to a user.

-procexec

-**User related process execution**: Associate process execution to users.

-userexec

-groupexec

-**In sequence user related process execution**: Associate sequences of process execution steps to users

-userexecsequence

-groupexecsequence

# Define the timing of single events with 'patterns'

- A pattern tag (<pattern></pattern) is used in many ITPSL directives to bind the event specification to a specific time period or a periodic occurrence specification (instance specifier) [9]:

**<pattern>**[AND/OR/XOR/NOT] (spec1,spec2,...,specn)**</pattern>**

Where each 'timespec' can have one of the following forms:

**from-now**
**hh-hh today**
**hh-hh (x | (0-999) ) days ago**
**[more-than | less-than] x times for the last (minute | hour | day | month | year)**
**[more than | less-than] x times every (Sunday...Saturday) for the last (month|year)**

# ITPSL pattern example

<hardwareop>
<operation>device-addition</operation>
<bus>usb</bus>
<deviceidstring> OR ('MuVo-X', 'MuVo NX', ) </deviceidstring>
<pattern> 08-17 6 days ago </pattern>
<userwasloggedon> chrisc </userwasloggedon>
</hardwareop>

Alternative pattern examples:

<pattern> 3 times every Monday for the last month </pattern>

<pattern> more than 3 times for the last hour </pattern>

# ITPSL signature polymorphism

-ITPSL directive specification tags employ binary operators:

- “Access on a file that could have a name like “this” OR “that” AND contents either likes “this” or “that”.
- Use of conjunction (AND), disjunction (OR), exclusive disjunction (XOR) and negation operator (NOT) simultaneous operator???.
- Signature reuse: repositories and semantics to apply

```
<?xml version="1.0"?>
<itpslsig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="/home/georgios/E6400backup/giorgos/Research2007/ITPSL/validate9.xsd">
<itpslheader>
      <signid> 69754c2b65627a098d02eb6244e40e69 </signid>
      <signdate>
            <year>2010</year>
            <month>8</month>
            <day>2</day>
      </signdate>
<ontology>
       <reason>intentional</reason>
       <revision>1.0</revision>
       <user_role>ordinary_users</user_role>
       <detectby>multi</detectby>
       <context> detection </context>
       <weightmatrix>1,1,1,10,20,30,20,10,20</weightmatrix>
       <os>linux</os>
       <version>2.6</version>
       <threatkeywords> ip theft portable media prototype surveillance </threatkeywords>
       <synopsis> “This signature detects the scenario of device prototypes being moved to
       USB keys ” </synopsis>
</ontology>
</itpslheader>
.....
```

```
<itpslbody>
    <mainblock>
        <mainop>as_a_result_of</mainop>
            <subblock>
                <subop>single</subop>
                <groupexec>
                    <groupname>engineering</groupname>
                    <name>OR (cp,mv) </name>
                    <path> OR (/usr/bin, /bin, /usr/sbin) </path>
                    <argumentlist> OR (prototype*, schem*, /media,) </argumentlist>
                    <singleprocess> yes </singleprocess>
                <groupexec>
            <subblock>
            <subblock>
                <subop>AND</subop>
                <groupcanaccessdir>
                    <groupid>engineering</groupid>
                    <dirname>OR (prototype,testdesign,schematics)</dirname>
                    <location> OR (/share/storage/pblk3000/,
                        /data/storage/prototypes)</location>
                    <ability>full</ability>
                    <singledir>yes</singledir>
                </groupcanaccessdir>
                <hardwareop>
                    <operation>device-addition</operation>
                    <bus>usb</bus>
                    <deviceidstring> NOT ('Creative MuVo-X USB player', 'MuVo NX', 'USB
                    Mass Storage') </deviceidstring>
                    <groupwasloggedon> engineering </groupwasloggedon>
                </hardwareop>
            </subblock>
        </mainblock>
    </itpslbody>
</itpslsig>
```

# References


[1] Hay B., Nance K., Bishop M. (2009), “Live Analysis Progress and Challenges”, IEEE Security & Privacy, Volume 7, Number 2, pages 30-37.

[2] Adelstein F. (2006), “Live Forensics: Diagnosing Your System without Killing it First”, Comm. ACM, vol.49, no.2, 2006, pages 63-66.

[3] http://en.wikipedia.org/wiki/Syslogd

[4] http://www.winsyslog.com/en/

[5] The OpenGroup's Distributed Audit System: http://openxdas.sourceforge.net/

[6]Cisco's Monitoring and Analysis Report System: http://www.cisco.com/en/US/products/ps6241/tsd_products_support_reference_guides.html

[7] Event Data Warehouse product: http://www.sensage.com/products/event-data-warehouse.php

[8] Arcsight logger applicance: http://www.arcsight.com/products/products-logger/

[9] Meier M. (2004), “A Model for the Semantics of Attack Signatures in Misuse Detection Systems”, 7th Information Security Conference, LNCS Volume 3225, Springer, Berlin/Heidelberg, pp. 158-169 .