

An Overview of the Research Activities of the CIDRE Inria Team (with a Focus on Security Events Management)

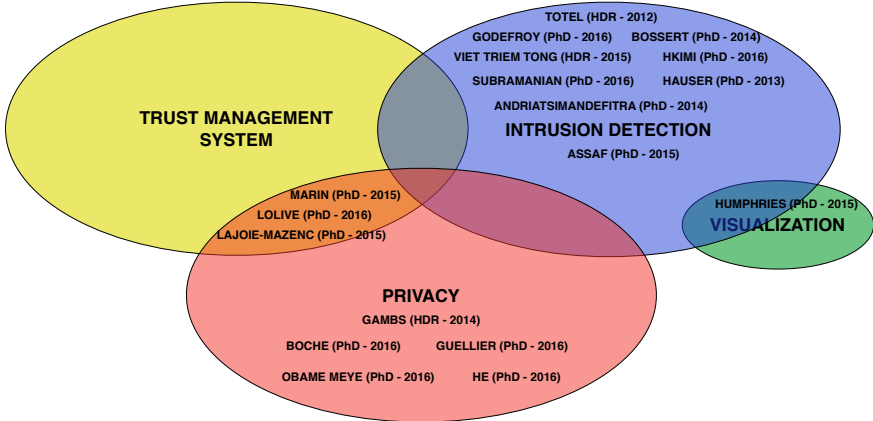
January 2017

Ludovic Mé
Inria and CentraleSupélec

Team Motto

Contribute building **secure** distributed systems
respectful of users **privacy**
even in the presence of **compromised** components

Research areas



Permanent Members



INRIA and CNRS researchers



E. Anceaume 



M. Hurfin 

Faculty members



C. Bidan 



G. Guette 



G. Hiet 



L. Mé 



G. Piolle 



E. Total 



F. tronel 



V. Viet Triem Tong 

To Follow in this Presentation

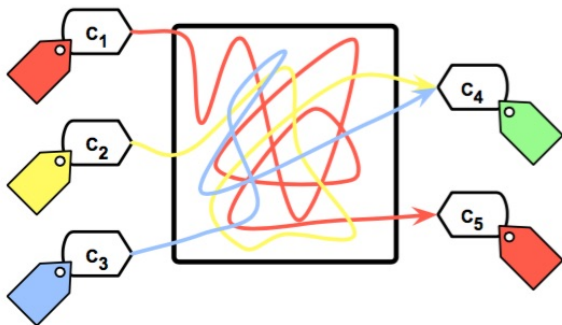
1. Intrusion Detection : through Information Flow Control
2. Alert Correlation :
 - ▶ Taking the context into account
 - ▶ Generation of correlation rules
3. Security Event Visualisation
4. ICS Monitoring

1. Intrusion Detection through Information Flow Control

Rather than specifying abnormal (misuse detection) or normal behaviors (anomaly detection), we observe the system to detect any **information flows** that violate the security policy :

- ▶ Objects of the supervised system (or inside supervised programs) are considered as containers of information (i.e. files, variables, etc.)
- ▶ The security policy specifies the type of information that each objects is allowed to contain
- ▶ Tags attached to the objects contains taint information (meta-information) describing their content

Intrusion Detection through Information Flow Control



Intrusion Detection through Information Flow Control

Information flow and intrusion

- ▶ System calls modify the tags of the objects depending of the resulting information flow (propagation of taint data)
- ▶ A intrusion occurs when the tag of a given object specifies a type of information that is irrelevant with respect to the security policy
- ▶ The security policy can :
 - ▶ be defined “from scratch”, through a language or a GUI
 - ▶ inferred from access control rights (i.e., rw rights to files)

Intrusion Detection through Information Flow Control

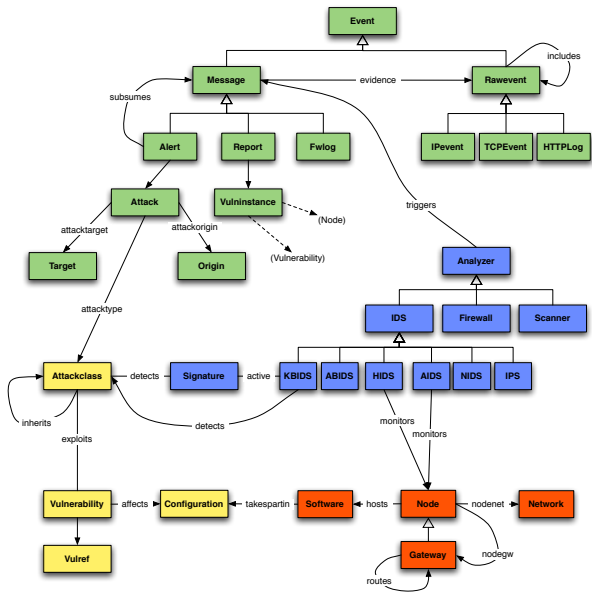
Blare family tools



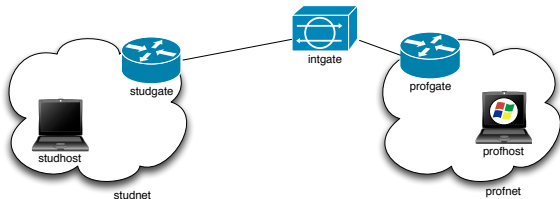
- ▶ KBlare : a Host-based IDS implemented as a Linux Security Module (LSM)
- ▶ JBlare : a Java Virtual Machine (JVM) hypervisor, able to track information flows inside Java programs
- ▶ AndroBlare : a port of KBlare to Android smartphones (actually used to analyse Android malware)
- ▶ HardBlare : hardware enhanced information flow monitoring (work in progress)

2-1. Alert Correlation : Taking the Context into Account

- ▶ Alarm correlation requires additional data which are not found in alerts
- ▶ We propose a structured data model which provides the required information
 - ▶ Contextual information
 - ▶ about the monitored information system
 - ▶ about external activities and trends
 - ▶ Background knowledge about attacks and vulnerabilities
 - ▶ Analyzers (IDSes, FW, scanners) capabilities
 - ▶ Events and alerts generated by the analyzers

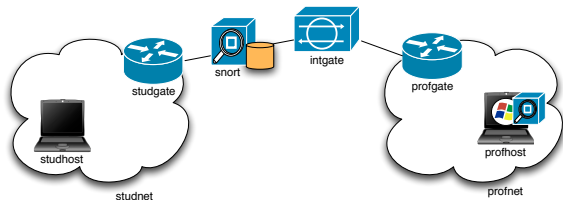


Description of the System to Monitor



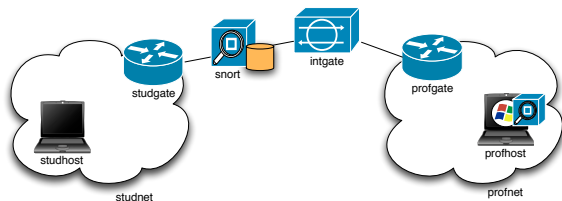
```
network(profnost), network(studnet),network(intnet)  
gateway(profgate), gateway(studgate), gateway(intgate)  
routes(studgate, intgate, profnet)  
node(profnost), node(studhost)  
nodeaddress(profnost, '192.168.2.38'), nodeaddress(studhost, '192.168.1.12')  
hosts(profnost, software('WindowsXP', -, os, -))
```

Description of the Monitoring System



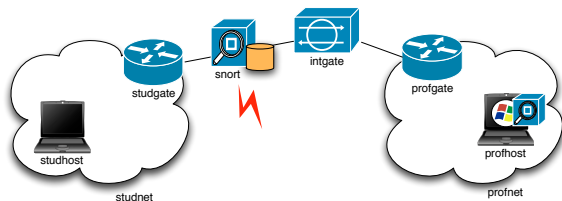
```
nids(snort), kbids(snort)  
monitors(snort, studgate, intgate)  
signature('s2351')  
detects('s2351', attackclass('blaster'))  
active(snort, s2351)  
hids(av)  
abids(av)  
monitors(av, profhost)  
detects(av, attackclass(bufferoverflow))  
firewall(intgate)
```

Attacks and vulnerabilities



```
vulnerability(blastervuln)  
refersto(blastervuln, 'cve', '2003-0352')  
affects(blastervuln, blasterconf)  
takespartin(software('WindowsXP', -, os, -), blasterconf)  
exploits(attackclass('blaster'), blastervuln)  
inherits(attackclass('blaster'), attackclass('worm'))  
inherits(attackclass('blaster'), attackclass('bufferoverflow'))
```

Alerts and events

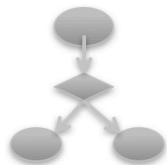


```
attack(blasterattack, db)  
attacktype(blasterattack, attackclass(blaster))  
origin(o1, ipaddress('192.168.1.12'))    attackorigin(blasterattack, o1)  
target(t1, ipaddress('192.168.2.38'))    attacktarget(blasterattack, t1)  
event(snortalert, ds)  
message(snortalert, snort)  
alert(snortalert, blasterattack)
```

Data can be used to...

1. Correlate alerts and investigate security events : sysadmin submits queries to the knowledge management system, for example to eliminate false positives
 - ▶ Is the host targeted (according to the alert) immune to the attack ?
 - ▶ Is a given analyzer able to detect a given attack ?
 - ▶ etc.
2. Generate correlation rules from attack tree, using
 - ▶ the description of the system to monitor
 - ▶ the description of configuration of the monitoring system

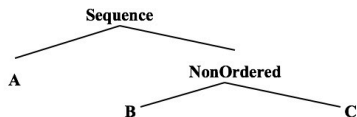
2-2. Alert Correlation : Rule Generation



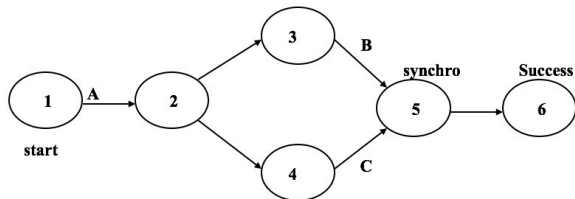
```
<ENCHAIN>  
  Sequence(A, NonOrdered(B, C))  
</ENCHAIN>
```



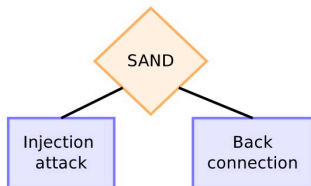
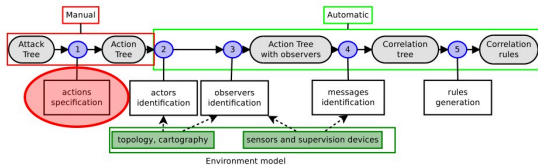
Compilation



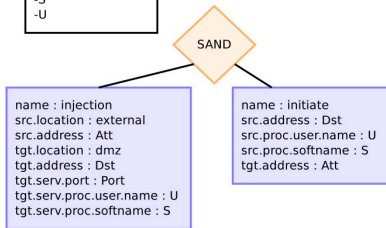
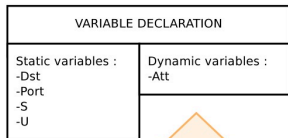
Compilation



Alert Correlation : Rule Generation



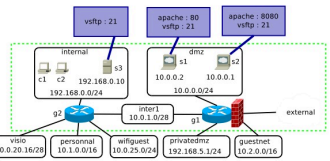
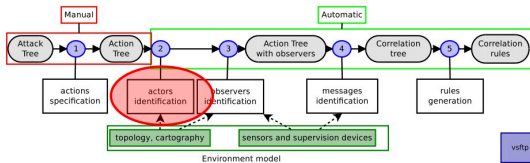
CAPEC
CEE



Action d'attaque: **CAPEC** : Common Attack Pattern Enumeration and Classification

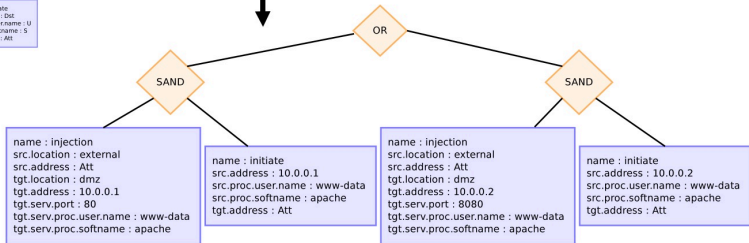
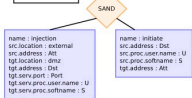
Action normale: **CEE** : Common Event Expression

Alert Correlation : Rule Generation

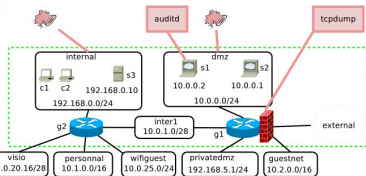
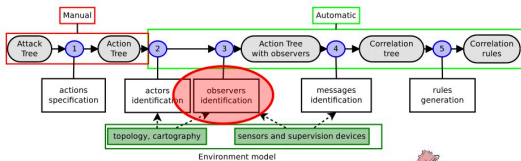


Instanciation des var. statiques

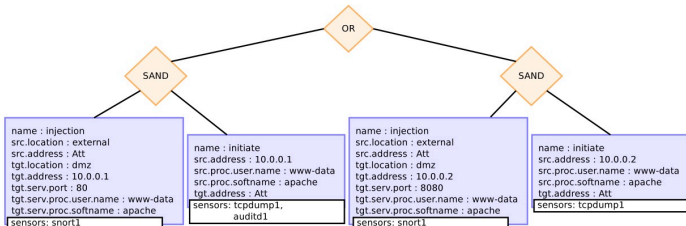
VARIABLE DECLARATION	
Static variables :	Dynamic variables :
-Dst	-Att
-Port	-S
-U	



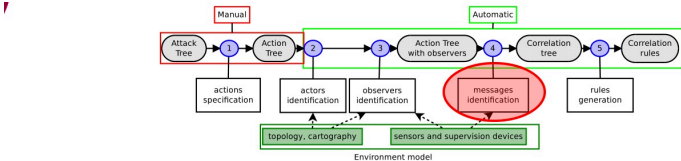
Alert Correlation : Rule Generation



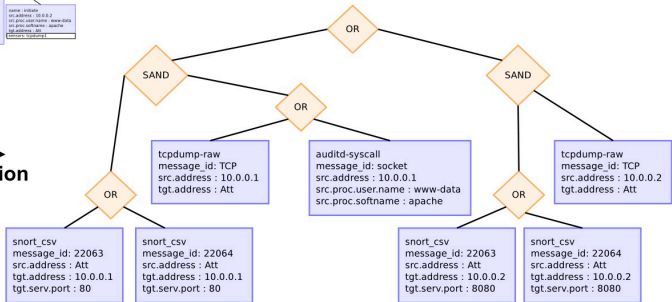
Observateur



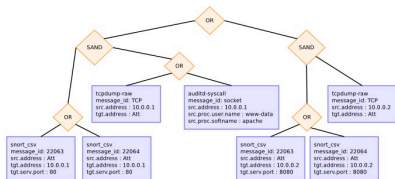
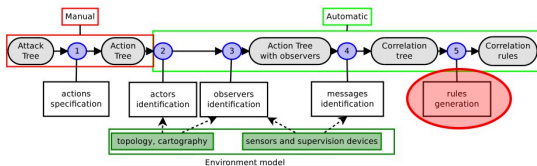
Alert Correlation : Rule Generation



Action → observation



Alert Correlation : Rule Generation



OneAmong (

Sequence(OneAmong(A1, A2), OneAmong(A3,A4)),
Sequence(OneAmong(A5,A6), A7)

)

3. Security Events Visualization

Goal

Helping the security analyst for visualizing the (huge amount of) security events

- ▶ Analysts are opportunistic : they use anything that can be useful, regardless of the format
- ▶ Helping them ?
 - ▶ Extensible tool to analyze multiple log file types consistently
 - ▶ By offering them appropriate representations
 - ▶ By seeking simplicity
 - ▶ By making it possible to access the details during an iterative visualization process

Vizualization Tools : Requirements

- ▶ Accept logs of any format
- ▶ Avoid the syndrome of the blank page by proposing a summary view
- ▶ Allow the analyst to interactively choose the fields it wants to see applied and automatically select a suitable representation based on the selected data
- ▶ Allow filtering of alerts
- ▶ Allow to cross logs

Example



Intrusion detection in Industrial control systems (ICS)

ICS require adequate intrusion detection approaches

- ▶ Limited resources and real time constraints \Rightarrow popularity of network-based approaches
- ▶ Regularity assumptions on the network's behavior \Rightarrow popularity of anomaly-based approaches
- ▶ Need knowledge about the physical process to detect process-aware attacks

In practice. . .

- ▶ Regularity assumptions often fail when confronted with real data from industrial plants
 - ▶ Manual interventions of operators [Caselli, 2015]
 - ▶ Features such as dynamical port allocation (RPC) [Barbosa, 2013]
- ▶ Limited knowledge about the physical process for certain types of systems, for instance sequential control systems

Detecting sequence attacks

Examples of sequence attacks

- ▶ Exclusion attacks
 - ▶ Simultaneously opening a valve and activating a motor
- ▶ Wear attacks
 - ▶ Repeatedly opening and closing a valve

State of the art

- ▶ Few attempts at detecting attacks against sequential control systems
- ▶ Existing work is inadequate to detect this type of attacks
 - ▶ Formalism not expressive enough [Mitchell, 2014]
 - ▶ Indirect detection based on network exchanges [Caselli, 2015]

Detecting sequence attacks

Work in Progress (PhD thesis, coop. with Gipsa-Lab in Grenoble)

- ▶ The specifications are safety properties expressed in linear temporal logic (LTL)
- ▶ The properties represent ordering constraints on the state of actuators and sensors
- ▶ Approach in two steps :
 - ▶ **Learning step** : inference of specifications from execution traces of the physical process
 - ▶ **Detection step** : detection of specification violations
- ▶ Notion of activity
 - ▶ An activity corresponds to a sub process or a functioning mode (normal, degraded) of a sequential control system
 - ▶ Specifications differ for each activity
 - ▶ Learning and detection steps are performed per activity

Tanks.
Questions ?