

# Blare Tools: A Policy-Based Intrusion Detection System Automatically Set by the Security Policy

Laurent George Valérie Viet Triem Tong Ludovic Mé  
 SUPELEC, SSIR Group (EA 4039), Rennes, France  
 Mail: [firstname.lastname@supelec.fr](mailto:firstname.lastname@supelec.fr)  
 WWW: <http://www.rennes.supelec.fr/ren/rd/ssir/>



## Monitoring flows to detect those not allowed by the current security policy

Intrusion  $\equiv$  Illegal information flow

IDS  $\equiv$  Monitors information flow  $\left\{ \begin{array}{l} \text{At OS Level : Linux Kernel Patch (BLARE)} \\ \text{At Application Level : In Java (JBLARE)} \end{array} \right.$

Alert  $\equiv$  Detection of a flow not allowed by the security policy

How  $\equiv$  Using metadata (Tags) and monitoring information flows using hooks in the kernel and in the JVM

### Access Control is a reference for the IDS

We use an access control policy to infer a specification of the expected behavior (in terms of information flows) for our IDS.

### Access control update

The access control permission can be updated according to the needs of the organization; in this work we focus on this kind of updates

### Updating the flow policy : Aim of the Controller Daemon

- ▶ monitors the access policy
- ▶ translates every update into an update of the flow policy ( $\Rightarrow$  update of the tags)
  - ▶ For example, adding a reading permission (+read,object,user) to the access control policy implies that the information initially contained in the object can now flow to the user.
- ▶ Update the tags of objects that are concerned : all objects containing information initially in the object

### Example of an access control update (+read, file1, Bob)

	file1	file2	file3
Alice	read	read, write	-
Bob	<i>read</i>	read	read, write

### The corresponding tag updating procedure

```
O ← find objects containing information (i1)
foreach o in set O do
  update (Read Tag, o)
end
```

$i_1$  represents the information initially contained in *file1*

## Implementation : Blare, JBlare and Blare Controller Daemon

### Blare

- ▶ hooks in the kernel
- ▶ check flows using binary operations on tags
- ▶ tags are located in kernel memory
- ▶ tags are accessible to userspace through an API

### JBlare

- ▶ Translator performs bytecode instrumentation
- ▶ BlareMonitor classes manage security tags
- ▶ BlareSecManager handles the cooperation with Blare
- ▶ a modified JRE

### Blare Controller Daemon

- ▶ Controller Daemon watches the access control Policy
- ▶ Update of the flow policy is done using the Blare API

## Example: Illegal flow detection and Policy Update

