DEMO:

Enforcing Security Policies on JVM

Paolo Mori

Istituto di Informatica e Telematica
CNR - Pisa - Italy

Outline

- Fine-grained & History-based access control
- Credential-based access control
- Security policy and examples
- Demo
- Examples of application
- Future work

Behavioural and Credential-based Access Control

The prototype enforces security policies on a Java Virtual Machine integrating:

1. Fine-grained and History-based access control
   - Monitoring of the application behaviour
2. Credential-based access control
   - Access decisions are based on the credentials submitted along with application

Fine-grained and History-based Access Control

- Fine-grained: all the actions performed by an application on the local resource are monitored
  - The application is not an atomic entity

- History-based: the right of an application to execute an action on the local resource depends on the actions previously executed
  - Application behaviour

Application Behaviour

Applications interact with the local resource through Operating System calls

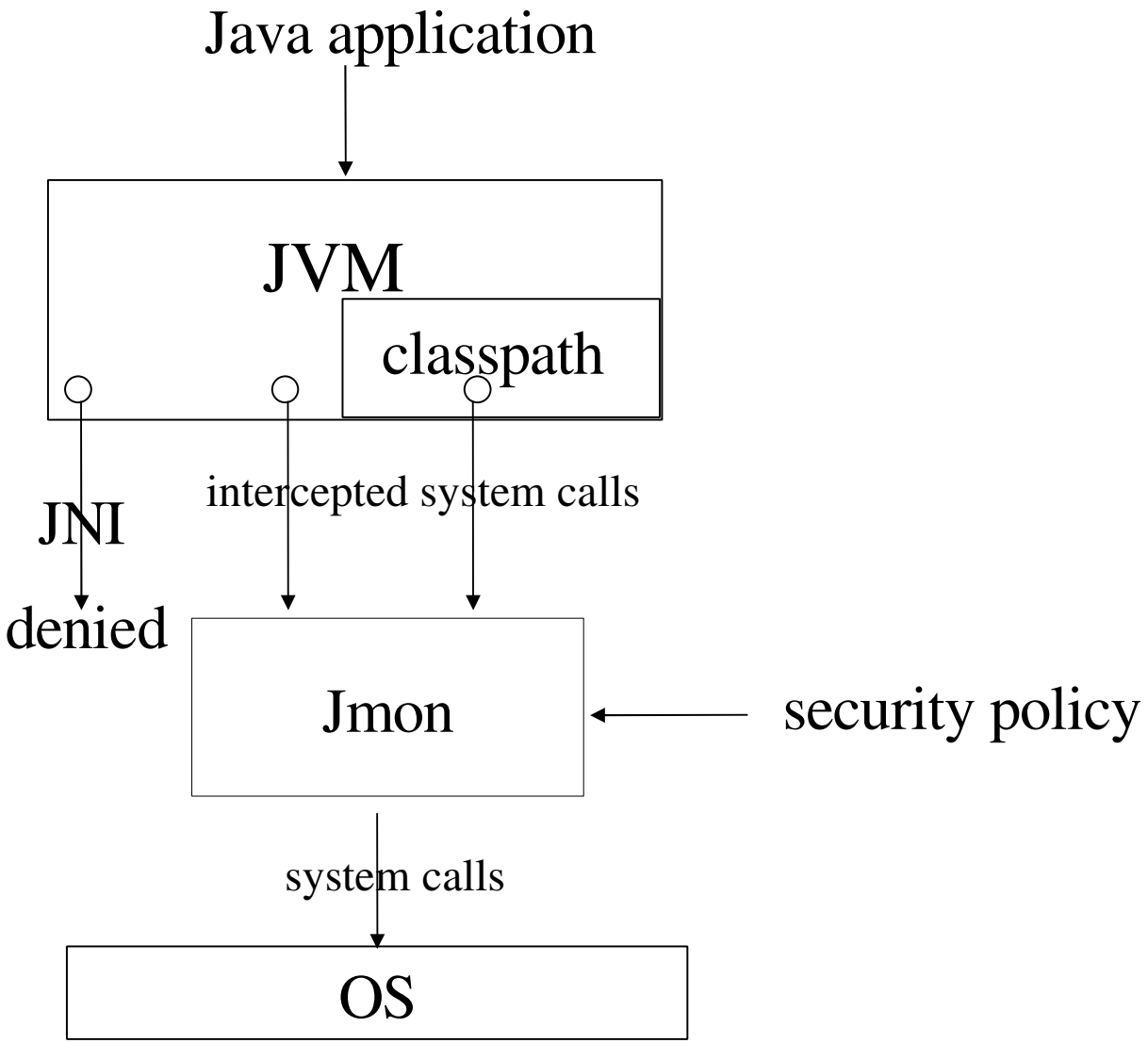
Application behaviour = sequence of OS calls

- parameters
- result

example:

```
open("fname.txt", O_RDONLY ) = 7
read(7, indbuf, 5 ) = 5
close(7)
```

Architecture



Behavioural Policy

Defines the admitted behaviour of applications in terms of:

- System calls
- Predicates that include:
  - Controls on system calls parameters and result
  - Environment conditions (e.g. time, workload, ..)
  - User's credentials evaluation request
- Composition operators
  - Define the sequences of system calls

Behavioural Policy

Expressed through a Process Algebra:

$P ::= \alpha.P || p(x).P || x:=e.P || P \text{ or } P || P \text{ par}_{\alpha 1,..,\alpha n} P || Z$

where:

- α is an action (System Call)
- x is a variable (vector)
- p(x) is a predicate on x
- Z is the constant process

Policy Example

....
OA:=false; OB:=false;
SetA:={/home/paolo/SetA/*}; SetB:={/home/paolo/SetB/*};

[eq(OB,false),in(x_1,SetA),eq(x_2,READ)].**open**(x_1,x_2,fd).
OA:=true.
i([eq(x_3,fd)].***read***(x_3,x_4,x_5)).
[eq(x_6,fd)].**close**(x_6,x_7)

[eq(OA,false),in(x_1,SetB),eq(x_2,READ)].**open**(x_1,x_2,fd).
OB:=true.
i([eq(x_3,fd)].***read***(x_3,x_4,x_5)).
[eq(x_6,fd)].**close**(x_6,x_7)

Policy Example

....
OA:=false; OB:=false;
SetA:={/home/paolo/SetA/*}; SetB:={/home/paolo/SetB/*};

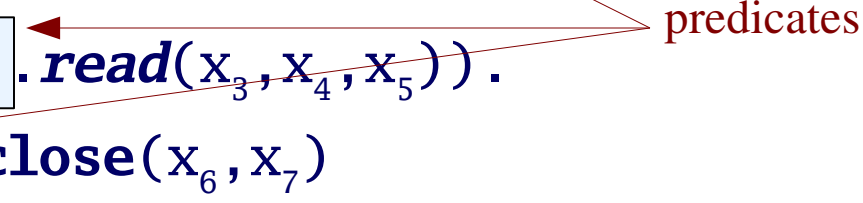
[eq(OB,false),in(x_1,SetA),eq(x_2,READ)].**open**(x_1,x_2,fd).
OA:=true.
i([eq(x_3,fd)].***read***(x_3,x_4,x_5)).
[eq(x_6,fd)].**close**(x_6,x_7)

actions

[eq(OA,false),in(x_1,SetB),eq(x_2,READ)].**open**(x_1,x_2,fd).
OB:=true.
i([eq(x_3,fd)].***read***(x_3,x_4,x_5)).
[eq(x_6,fd)].**close**(x_6,x_7)

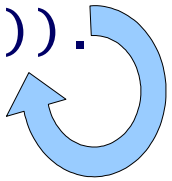
Policy Example

```
....
OA:=false; OB:=false;
SetA:={/home/paolo/SetA/*}; SetB:={/home/paolo/SetB/*};

[eq(OB,false),in(x1,SetA),eq(x2,READ)].open(x1,x2,fd).
OA:=true.
i([eq(x3,fd)].read(x3,x4,x5)).
[eq(x6,fd)].close(x6,x7)

[eq(OA,false),in(x1,SetB),eq(x2,READ)].open(x1,x2,fd).
OB:=true.
i([eq(x3,fd)].read(x3,x4,x5)).
[eq(x6,fd)].close(x6,x7)
```



Policy Example

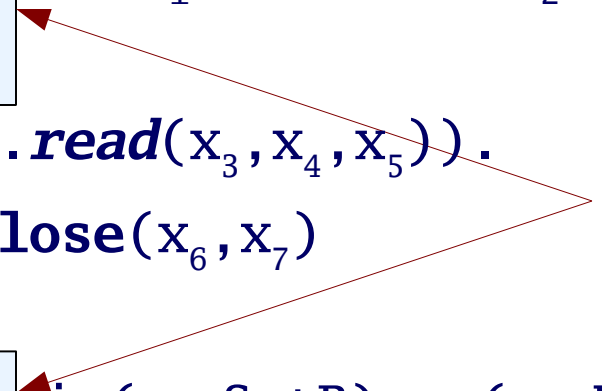
```
....
OA:=false; OB:=false;
SetA:={/home/paolo/SetA/*}; SetB:={/home/paolo/SetB/*};

[eq(OB,false),in(x1,SetA),eq(x2,READ)].open(x1,x2,fd).
OA:=true.
i([eq(x3,fd)].read(x3,x4,x5)).
[eq(x6,fd)].close(x6,x7)

[eq(OA,false),in(x1,SetB),eq(x2,READ)].open(x1,x2,fd).
OB:=true.
i([eq(x3,fd)].read(x3,x4,x5)).
[eq(x6,fd)].close(x6,x7)
```



iteration

Policy Example

```
....
OA:=false; OB:=false;
SetA:={/home/paolo/SetA/*}; SetB:={/home/paolo/SetB/*};

[eq(OB,false),in(x1,SetA),eq(x2,READ)].open(x1,x2,fd).
OA:=true.
i([eq(x3,fd)].read(x3,x4,x5)).
[eq(x6,fd)].close(x6,x7)

[eq(OA,false),in(x1,SetB),eq(x2,READ)].open(x1,x2,fd).
OB:=true.
i([eq(x3,fd)].read(x3,x4,x5)).
[eq(x6,fd)].close(x6,x7)
```



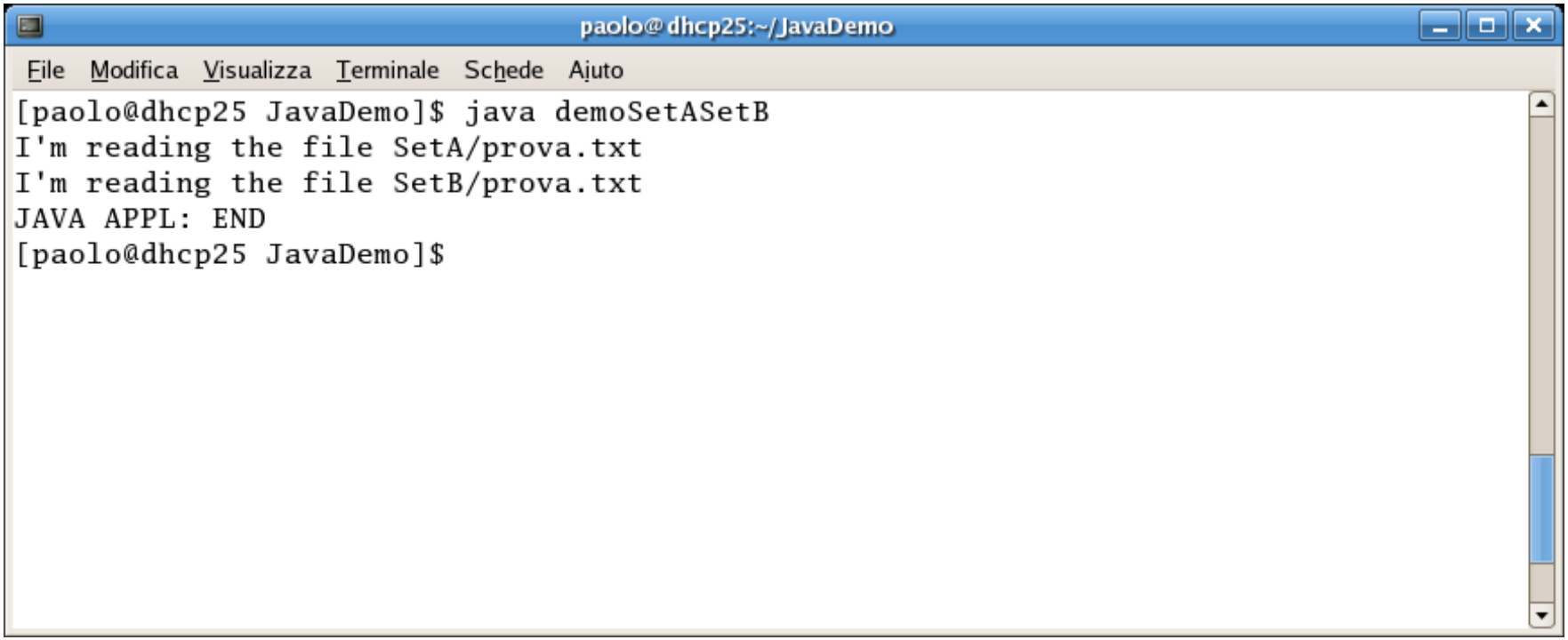
Java Application

```
InputStreamReader isrA =
                    new InputStreamReader(new  FileInputStream("SetA/prova.txt"));
System.out.println(“I'm reading the file SetA/prova.txt");
while(isrA.ready())   i = isrA.read();
isrA.close();

InputStreamReader isrB =
                    new InputStreamReader(new  FileInputStream("SetB/prova.txt"));
System.out.println(“I'm reading the file SetB/prova.txt");
while(isrB.ready())  i = isrB.read();
isrB.close();

System.out.println(“JAVA APPL: END");
```

DEMO 1

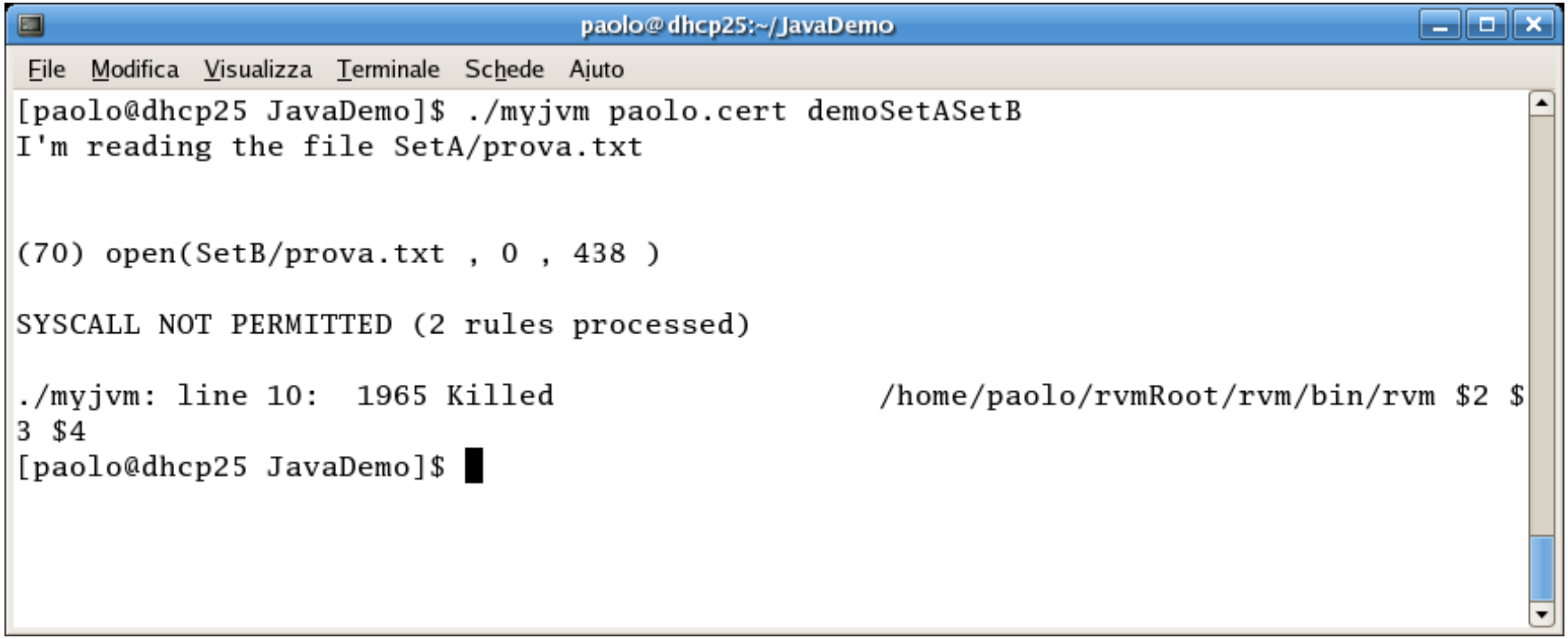
Demo 1

The application can read both files in SetA and SetB if the security policy is not enforced



Demo 1

The security policy does not allow to open a file in SetB if a file in SetA has been already opened



Policy Example II

.....previous policy.....

[eq(x_1,AF_INET),eq(x_2,STREAM),eq(x_3,TCP)].**socket**(x_1,x_2,x_3,sd).
[eq(x_5,sd),eq(x_6,localhost)].***connect***(x_5,x_6,x_7,x_8).
i(
  [eq(x_9,sd), eq(OA,false)].**send**(x_9,x_{10},x_{11},x_{12},x_{13})
  or
  [eq(x_{14},sd), eq(OA,true), tm(sendcrit)].**send**(x_{14},x_{15},x_{16},x_{17},x_{18})
  or
  [eq(x_{19}, sd)].**recv**(x_{19},x_{20},x_{21},x_{22},x_{23})
).
[eq(x_{24},sd)].**close**(x_{24},x_{25})

Policy Example II

Access Rules (RTML):

UniPi.sendcrit()<-IIT.researcher(name) ∩ UniPi.collab(name).

UniPi.collab(name)<-UniPi.university(uname).collab(name).

UniPi.university(uname)<-Miur.university(uname).

Java Application

```
Socket so = new Socket(host, port);

OutputStream sos = so.getOutputStream();

..... buffer initialization

sos.write(buffer, 0, iNumByte);
System.out.println("data sent BEFORE the critical file has been opened");

FileInputStream  fis = new  FileInputStream("SetA/prova.txt");

if ((iByteLetti = fis.read(buffer,0,buffer.length)) != -1)
    sos.write(buffer,0,iByteLetti);
System.out.println("data sent AFTER the critical file has been opened");
```

Application Provider Credentials and Certificate

Credentials:

```
UniGe.collab('CN=Paolo Mori, OU=IIT, O=CNR, L=Pisa, ST=Pisa, C=IT')<-Paolo.

Miur.university('CN=University of Genova, OU=Security Lab, O=CS Department,
   L=Genova, ST=Genova, C=IT')<- UniGe.

IIT.researcher('CN=Paolo Mori, OU=IIT, O=CNR, L=Pisa, ST=Pisa, C=IT')<-Paolo.
```

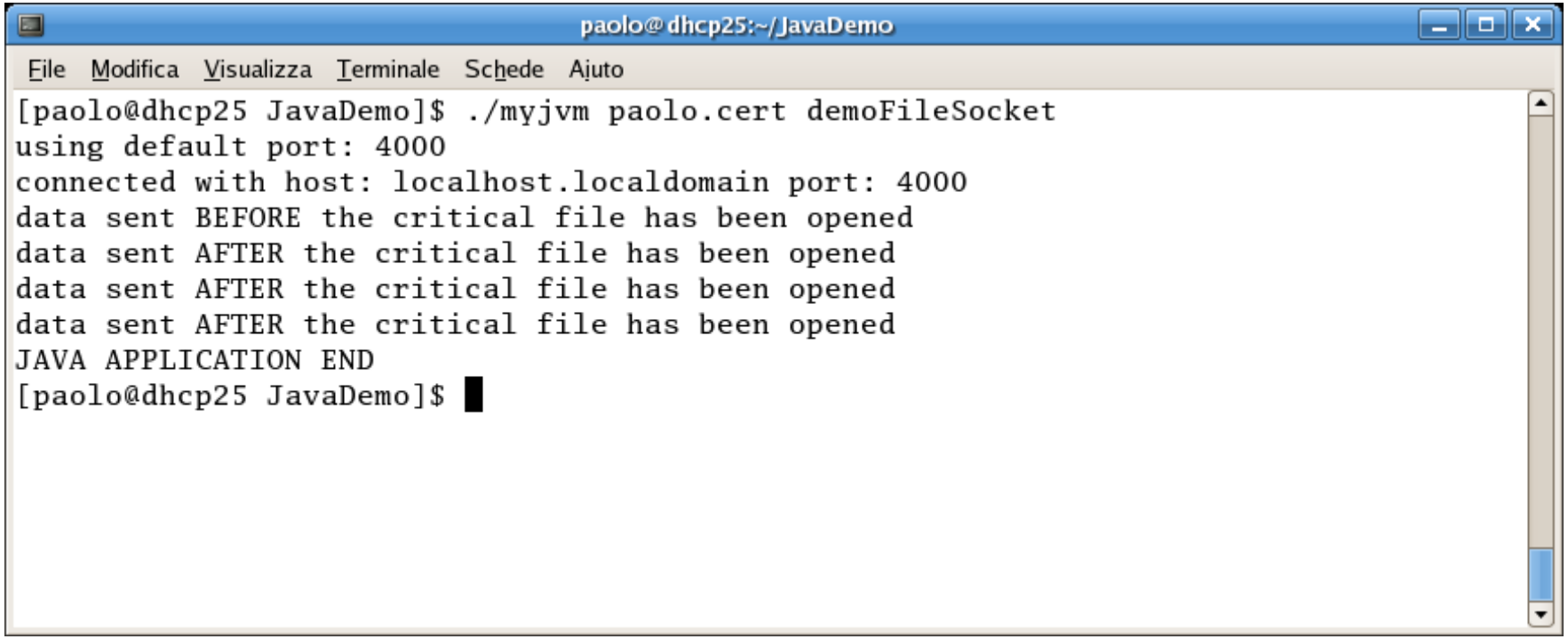
Certificate:

```
Owner: CN=Paolo Mori, OU=IIT, O=CNR, L=Pisa, ST=Pisa, C=IT
Issuer: CN=IIT-CA, OU=IIT, O=CNR, L=Pisa, ST=Pisa, C=IT
Serial number: 44f6fa55
Valid from: Thu Aug 31 17:03:49 CEST 2006 until: Wed Nov 29 16:03:49 CET 2006
Certificate fingerprints:
         MD5:  D3:8A:0D:2D:59:5B:E4:5B:4D:8C:1E:68:C3:DE:08:1D
         SHA1: B0:34:E3:27:C7:4B:43:A0:73:72:5A:7F:83:59:FC:83:D3:E9:7F:3E
```

DEMO 2

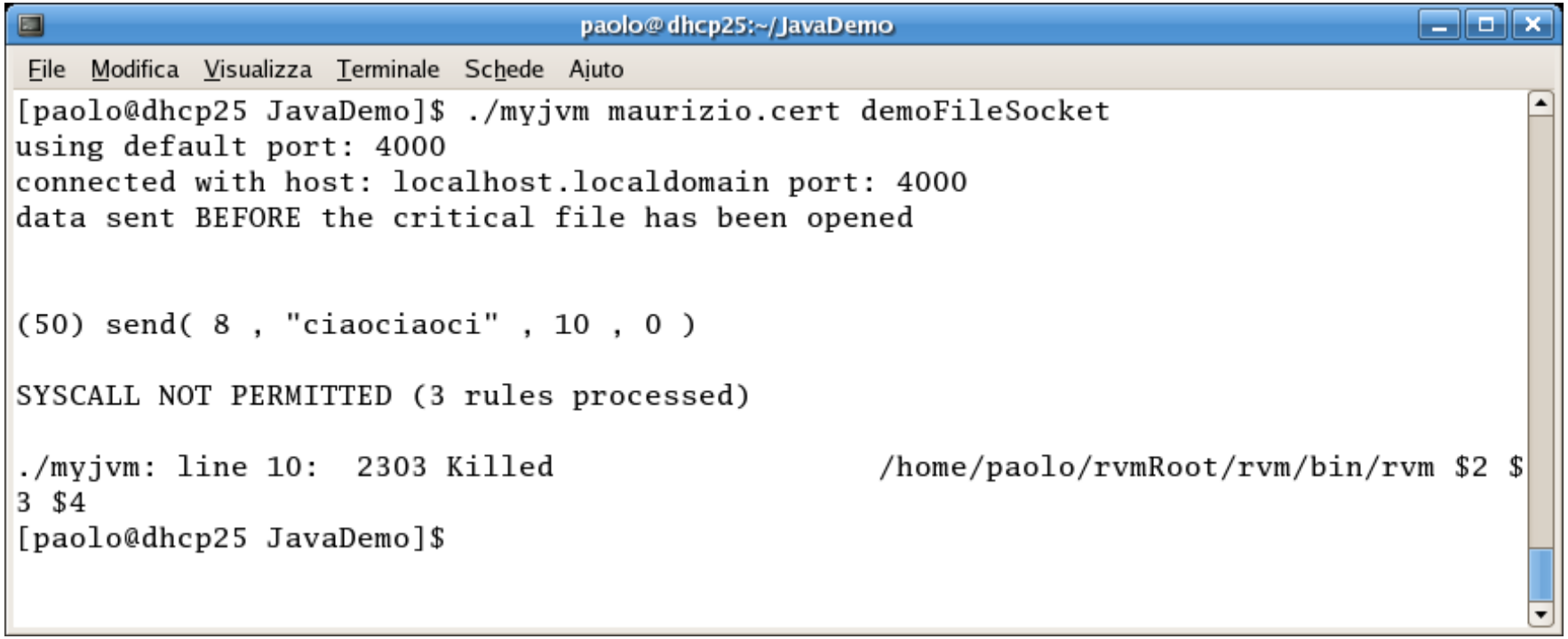
Demo 2

The credentials of the application provider “paolo” allow the application to send data to remote hosts after the critical file has been opened

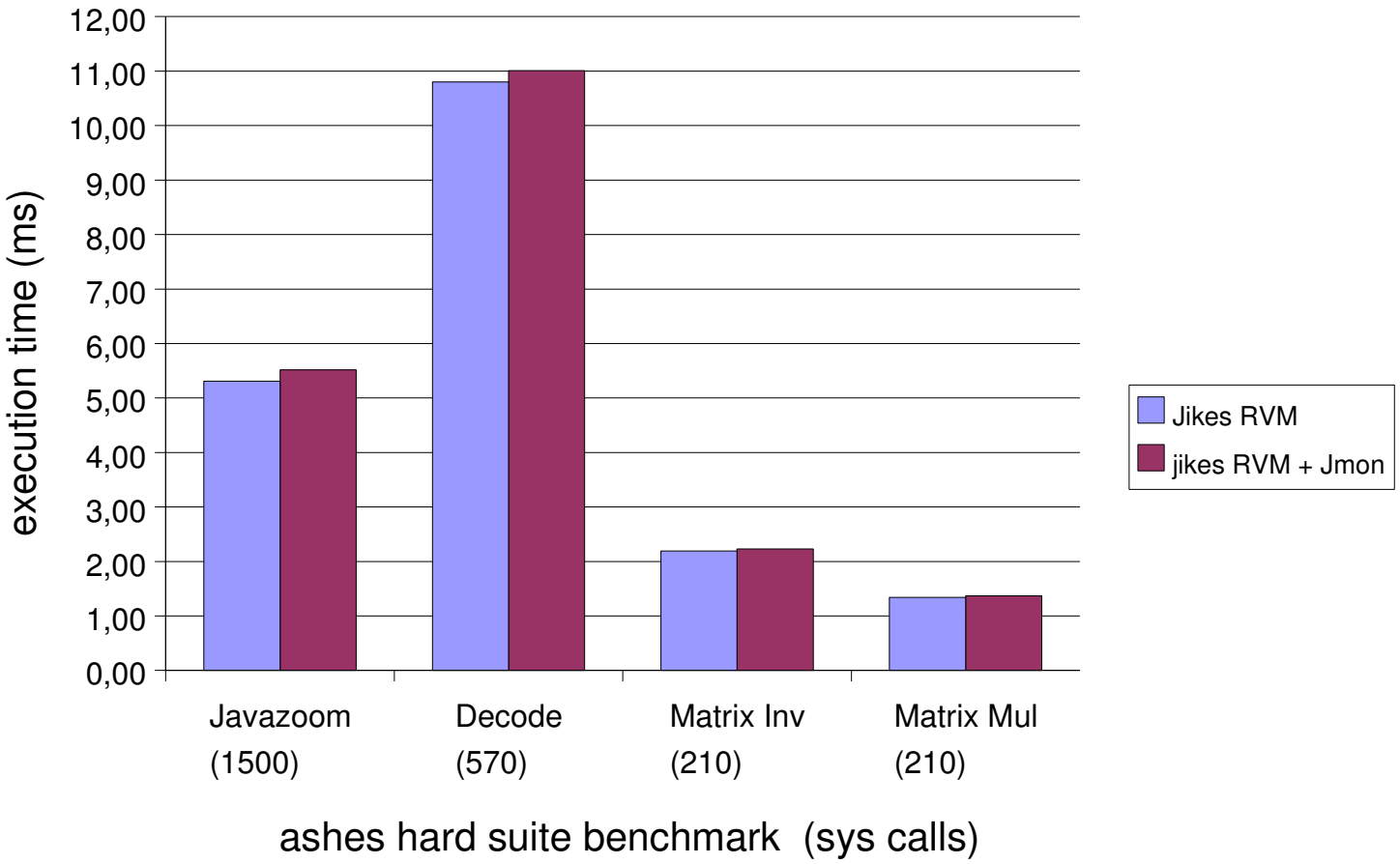


Demo 2

The credentials of the application provider “maurizio” DO NOT allow the application to send data to remote hosts after the critical file has been opened



Performances



Java Virtual Machine

- IBM Jikes Research Virtual Machine
  - Open source Java Virtual Machine
  - Research oriented
  - Follows:
    - The Java Language Specification
    - The Java Virtual Machine Specification
- GNU Classpath

Examples of Application

- Grid Computing
- Mobile systems

Grid Computing Environment

- Dynamic set of geographically dispersed users that share heterogeneous resources
  - Computers
  - Data Bases
  - Software repositories
  - Storage
  - .....
- Users belong to distinct administrative domains
  - Distinct security mechanisms
  - Distinct local security policies
  - No established trust relationships

Grid Computational Services

- The resource shared on the grid is a computer
- The grid computational service S executes the applications of any grid user U
  - Unknown applications
  - No direct trust relations between U and S

- The Grid Security Infrastructure must:
  - Protect the resources from the applications
  - Protect the applications from the resources

Protect Resources from Applications

- Several Access Control Systems have been adopted in grid environment
- Most of the systems have:
  - Coarse grained access control
    - Sometimes based on OS local account privileges
  - Static access control decision
    - Not history based

S3MS Security of Software and Service for Mobile Systems

S3MS project

Security of Software and Services for Mobile Systems

Coord: Fabio Massacci, University of Trento

- Motto: Security by contract.
- Goal: Providing a framework for managing contractual properties of mobile code as well as user/platform policy and corresponding matching technologies

S3MS Security of Software and Service for Mobile Systems

S&T Results

- Specification of S3MS contracts
- Architecture, Methods and Algorithms
- Prototype implementation for
  - Contractual support at development
  - Contractual enforcement at run-time
    - Java
    - .Net
  - Components for simulation environments
  - Case studies
- Validation and exploitation plan

Future Work

- Integration of reputation mechanisms
  - The enforced policy is determined by the reputation of the application provider
- Credential negotiation
  - Additional credentials are requested to the application provider when the submitted ones do not allow an action

References

H. Koshutanski, F. Martinelli, P. Mori and A. Vaccarelli: *Fine-grained and History-based Access Control with Trust Management for Autonomic Grid Services.* In Proceedings of International Conference on Autonomic and Autonomous Systems (ICAS06) IEEE Computer Society 2006 (Best Papers Award)

Thank You!

paolo.mori@iit.cnr.it