

Security Policy Checker and Generator for Java Mobile Codes

Sep. 27, 2002

Haruhiko Kaiya,

H. Furukawa and K. Kaijiri

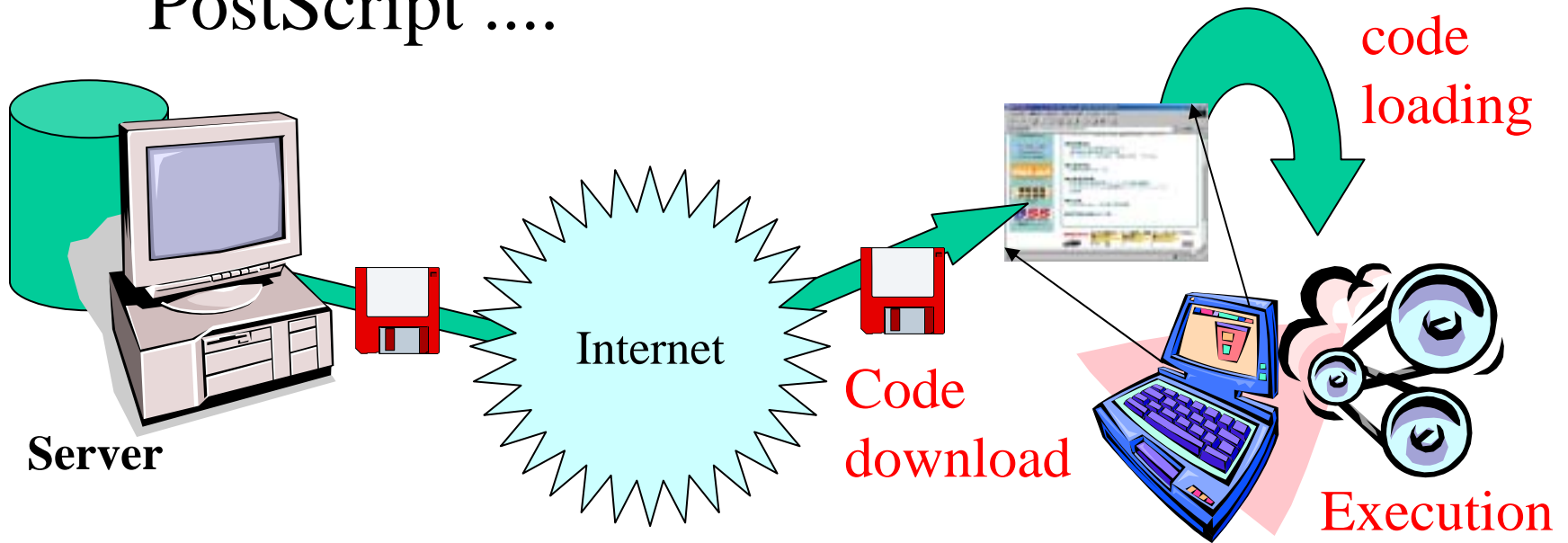
Shinshu University, JAPAN

Contents

- Mobile Codes, Expectation and Problems
 - Collaborative codes from different sites.
 - Security Policy.
- Requirements for the Tool
- How to generate Security Policy
- How to check it.
- Summary and Future Works

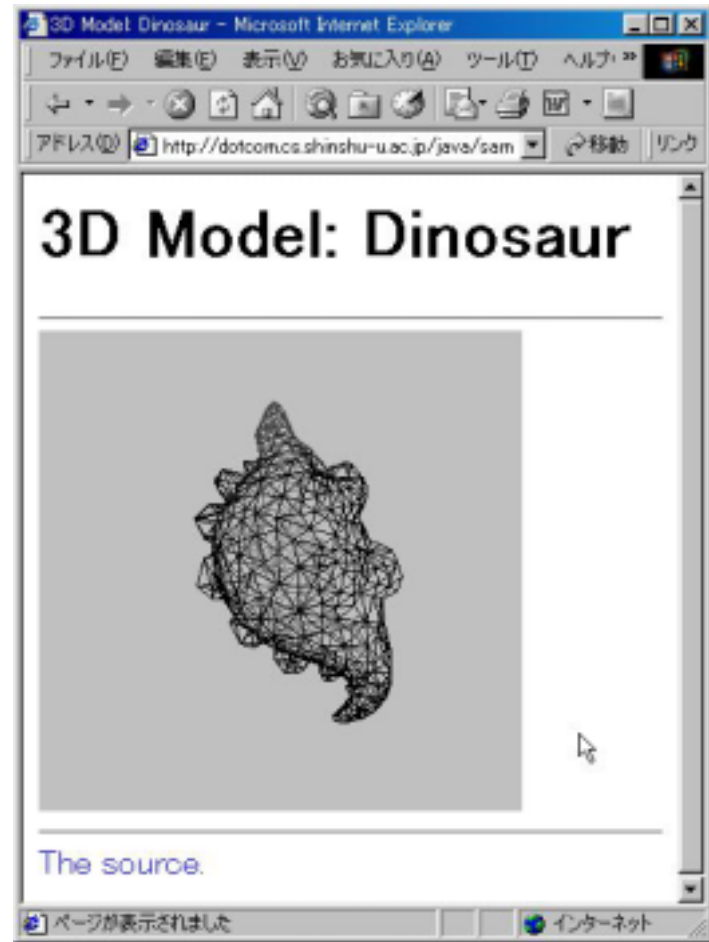
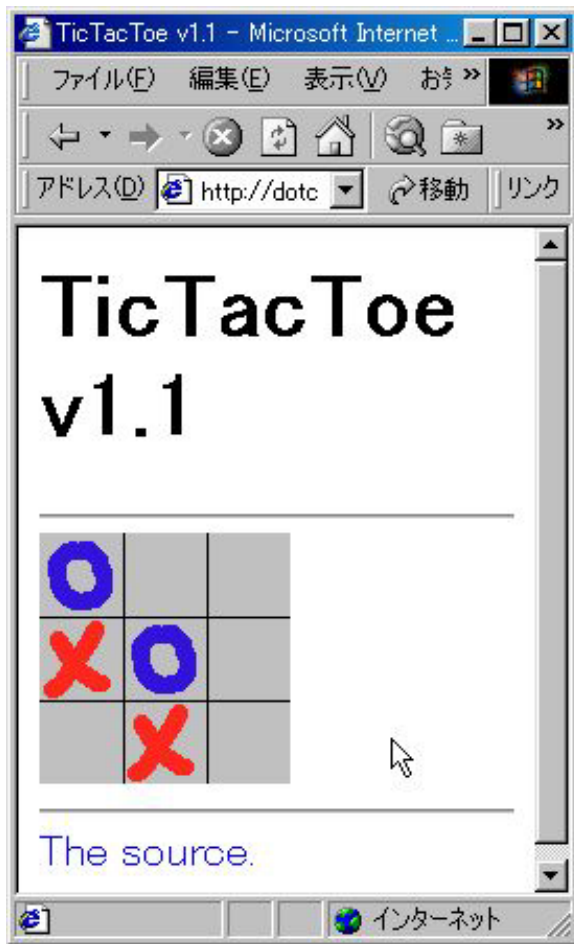
What is Mobile Code?

- Example: Java Applet, Javascript, VBscript, PostScript



- We do not deal with autonomous mobile codes in our current research.

Typical Examples: Applet

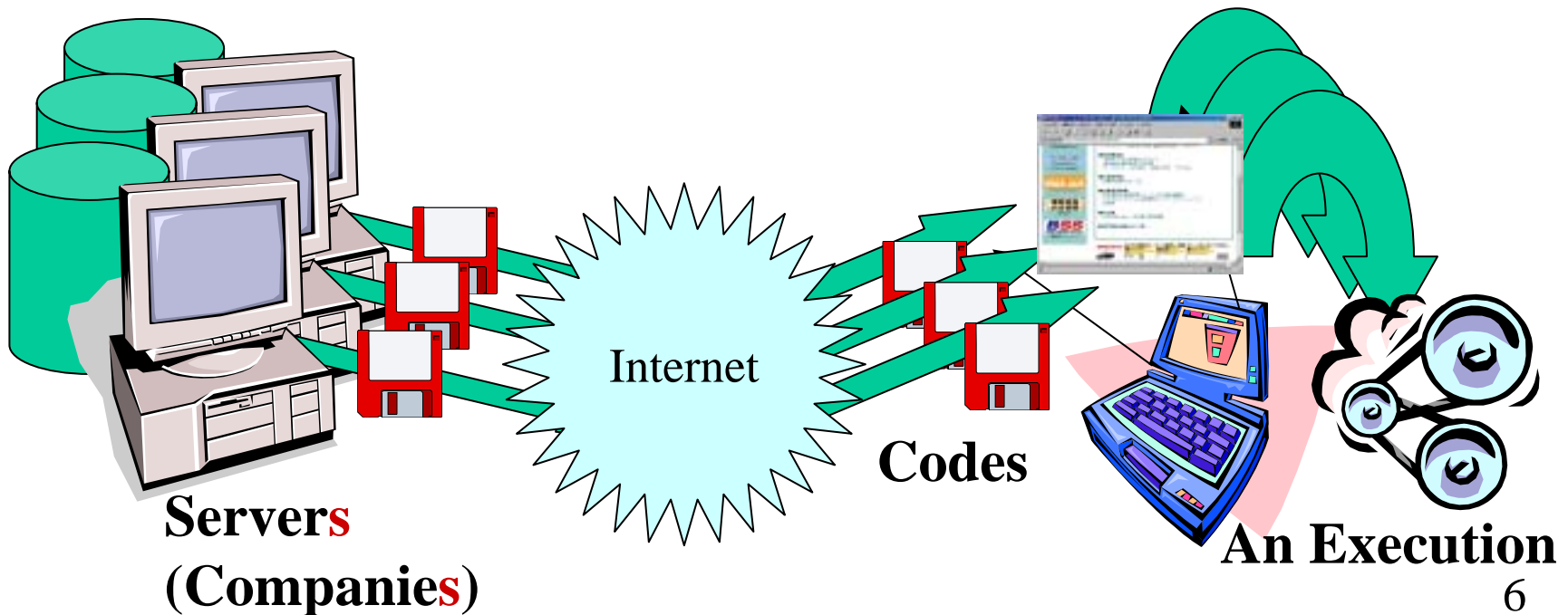


Advantages

- Users do not have to install software manually in their computers.
- Mobile codes decrease the traffic over the network.
- Response time is improved.
 - It is suitable for interactive system.

Optimistic Scenario

- Collaboration of mobile codes on a client.
 - Codes came from several servers (companies).
 - Effective B2B and B2C construction.

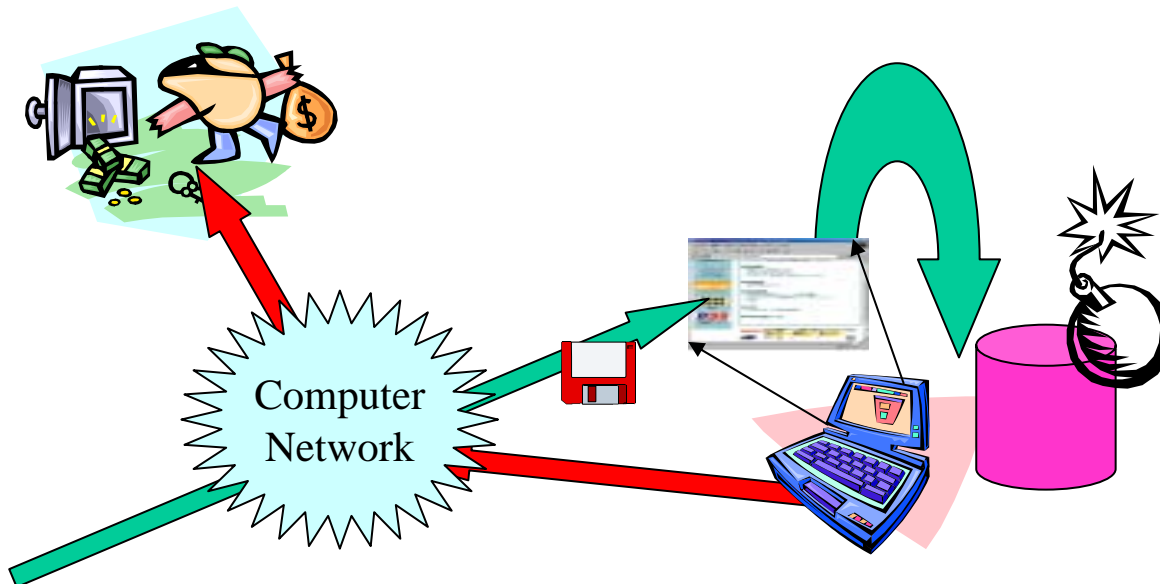


Meaning to Use Mobile Codes

- When you use mobile codes in your PC, the use is the same that authors of the codes directly operate your PC.
 - All authors are not always trustworthy.
 - So, Security Problems are occurred.

Problems

- If there are ill and/or defects in mobile codes,
 - Data in your PC can be **destroyed**.
 - Data in your PC can be **stolen**.
 - Your PC can do **wrong out of your control**.



Solutions in Java2

- Limit functionalities of a code, based on a place where the code was located, and on a sign on the code.
 - such place is called codebase which is represented in URL form.
 - digital signature.
- Security Policy: description of such limits.
- All functions can not work in a default policy.
- Sandbox model

Example of Java Policy

Codes download from here, and signed by this person

```
grant
  codeBase "http://java.sun.com/*",
  signedBy "Li" {
    permission java.io.FilePermission
      "/tmp/*", "read";
    permission java.io.SocketPermission
      "*", "connect";
  };
```

may read files under /tmp/, and

may make network connections to anywhere.

All other functions are not allowed!

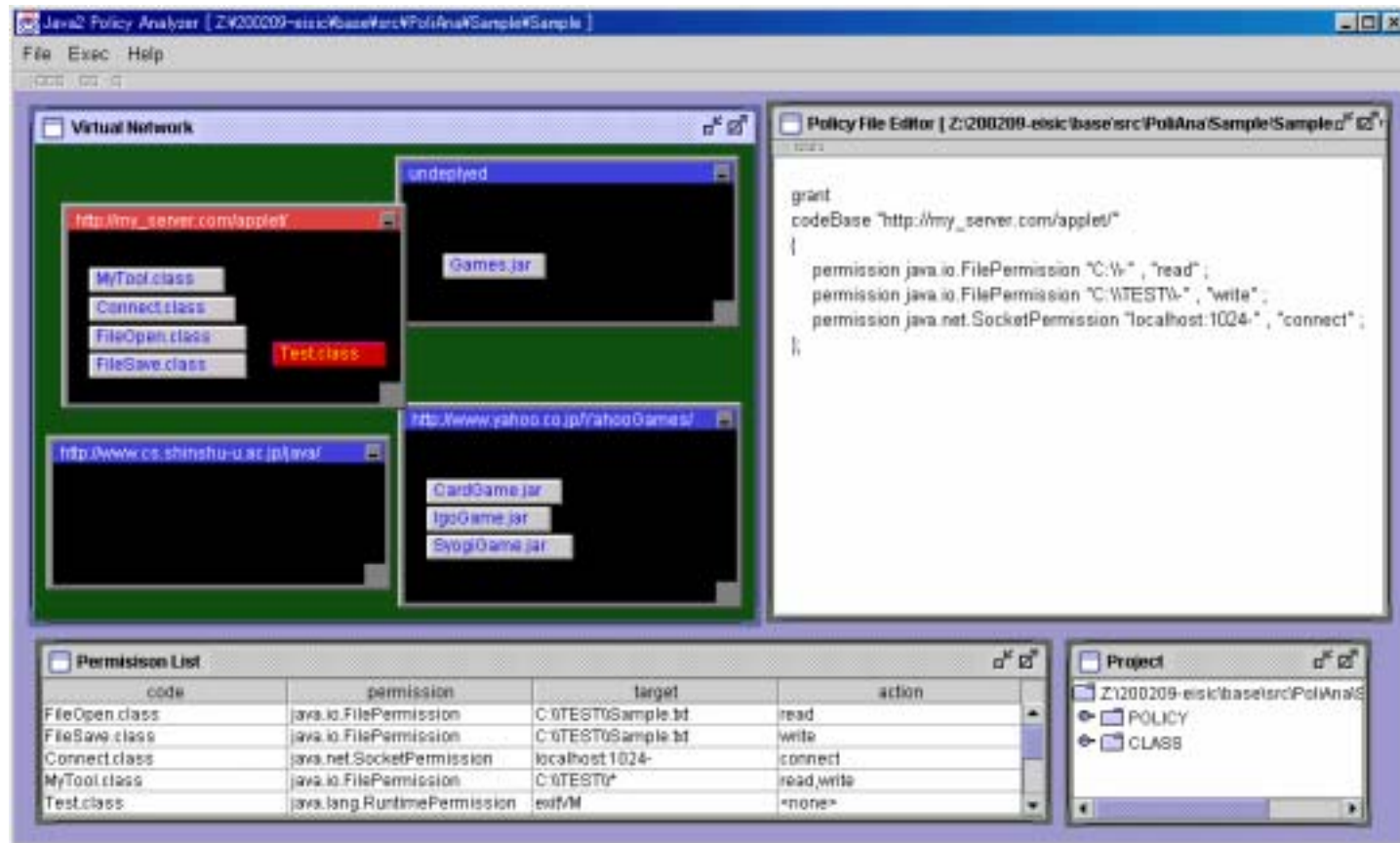
Problem Remains

- For developers and mobile code providers
 - Needs to check whether their codes can work under policies for codes provided by other companies.
 - If not, they should provide extended policies to their customers.
- For mobile code users
 - Needs to check that current policies provide just enough permissions for additional codes.
 - If too much permissions are provided to the codes, the codes can be executed unsuitably.

Requirements for our Tool

- Generate minimal set of policies, which enables collaborative codes to be executed.
- Check whether collaborative codes can be executed or not under a set of policies.

Overview of the Tool

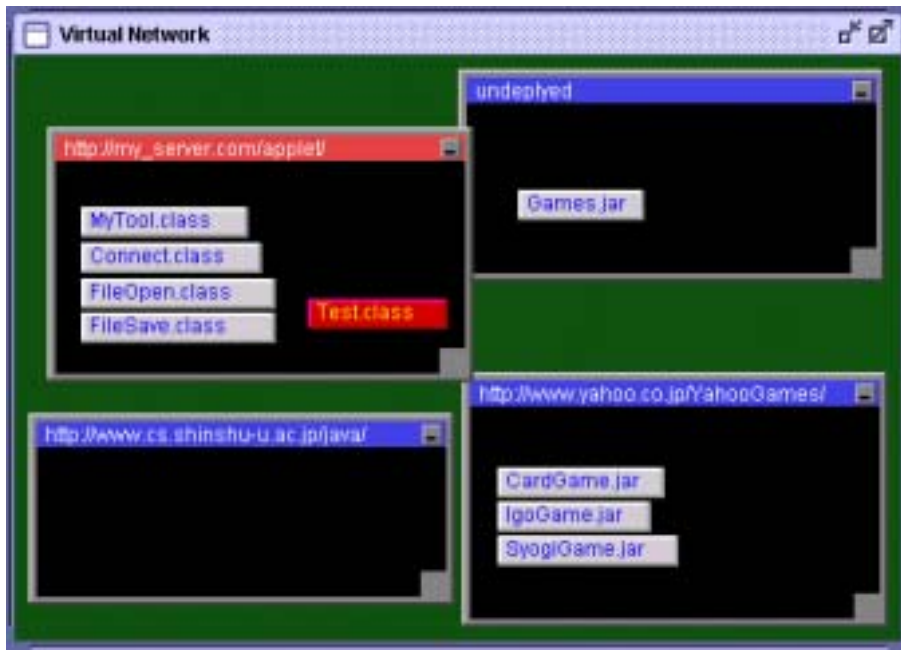


Generating Scenario

1. Input locations to deploy codes.
2. Input deployment of codes.
3. Input permissions that each code requires.
4. Select generate menu on our tool
 - Generate policies for the codes.

Virtual Network (VN) GUI

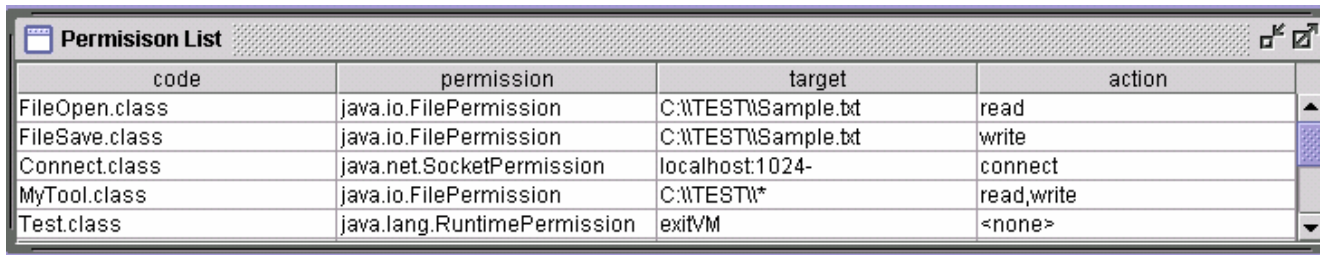
- Sub-Window: Location of codes.
- Rectangles in each sub-window: Codes



- Sub-windows and rectangles are created manually.

Permission List

- Each permission is input via GUI of VN.



code	permission	target	action
FileOpen.class	java.io.FilePermission	C:\TEST\Sample.txt	read
FileSave.class	java.io.FilePermission	C:\TEST\Sample.txt	write
Connect.class	java.net.SocketPermission	localhost:1024-	connect
MyTool.class	java.io.FilePermission	C:\TEST*	read,write
Test.class	java.lang.RuntimePermission	exitVM	<none>

- Each line shows a code and its required permission, when it is executed.
- There are more than one lines for a code, if the code needs several permissions.

Algorithm for minimal policies

For each *URL* in *VN*

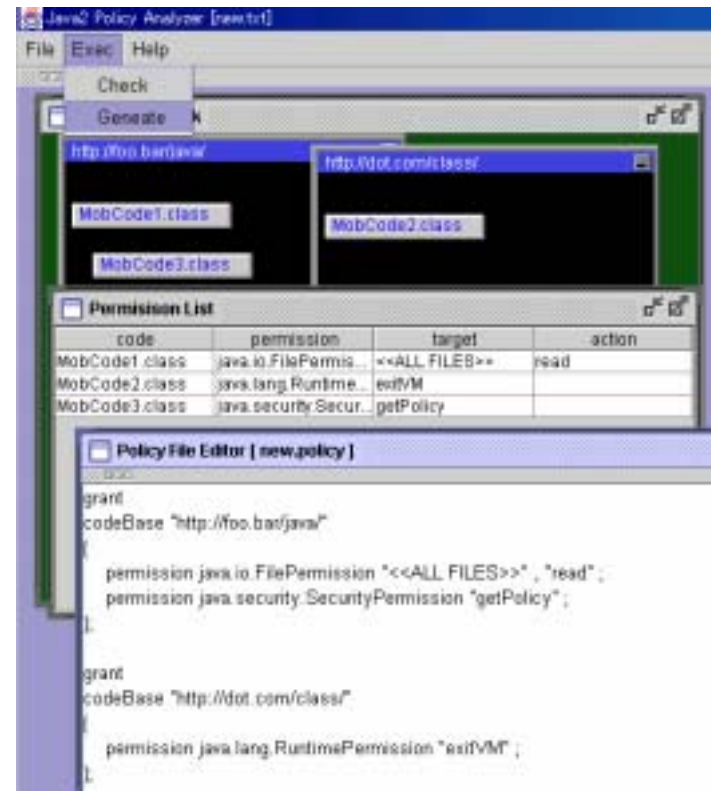
“grant codeBase *URL*{“

For each *code* in the *URL*

For each permission of the code

“permission action target”

“}”



Java2 Policy Analyzer [new.txt]

File Exec Help

Check

Generate

http://foo.bar/java/

MobCode1.class

MobCode3.class

http://dot.com/class/

MobCode2.class

Permission List

code	permission	target	action
MobCode1.class	java.io.FilePermis...	<<ALL FILES>>	read
MobCode2.class	java.lang.Runtime	exitVM	
MobCode3.class	java.security.Secur...	getPolicy	

Policy File Editor [new.policy]

```

grant
codeBase "http://foo.bar/java/"
{
  permission java.io.FilePermission "<<ALL FILES>>" , "read" ;
  permission java.security.SecurityPermission "getPolicy" ;
};

grant
codeBase "http://dot.com/class/"
{
  permission java.lang.RuntimePermission "exitVM" ;
};

```

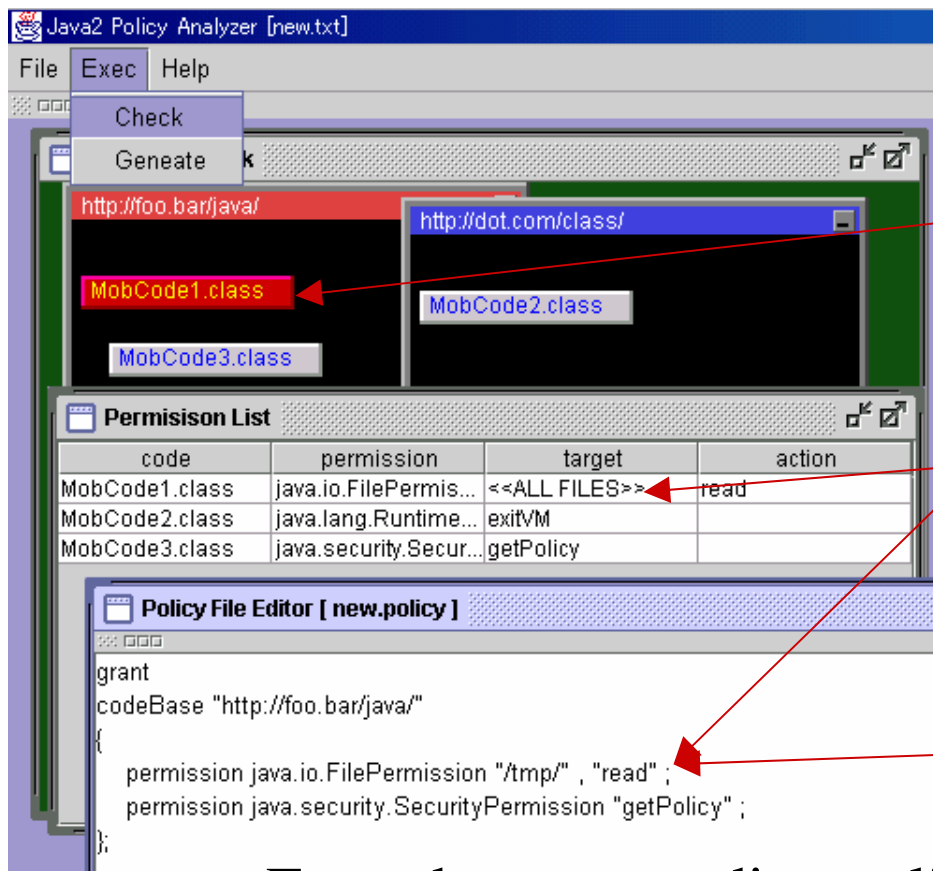
Example

- Two URLs
- Three Codes
- One permission for each code.

Checking Scenario

1. Add or modify a code and/or its permissions.
2. Select Exec.Check Menu to check.
3. Modify policies, permissions or deployment, if the code becomes red.

Example



- Under this policy, this code can not be executed,
- because this permission is not allowed under this policy.

- Extend corresponding policy.
- Reduce corresponding permission, if possible.
- Or change deployment of codes.

Summary

- Clarify advantages of collaborative mobile codes from several sites.
- Clarify its problem, how to provide just enough policies for the codes.
- Design and Implement a tool to solve them.

Future Work: Design & Impl.

- Support digital signature.
- Support hierarchy of URL.
 - Scalability of our tool
- Extract required permissions for each code automatically from codes.
 - Mandatory and optional permissions.

Future Work: General

- Find realistic examples of such kind of collaborative codes.
- Support autonomous mobile codes, agents.
- Support delegation of authorization.
- Bridge a gap between security requirements and our tool.
 - Link our tool to Goal Oriented Req. Analysis or AGORA (Attributed Goal Oriented RA).

That's All, Thank you.