Compliance of Multilayer Information Flow Policies

B. Hicks, S. Rueda, Y. Sreenivasan, T. Jaeger, P. McDaniel

We want to protect the secrecy of confidential data → Information Flow Security

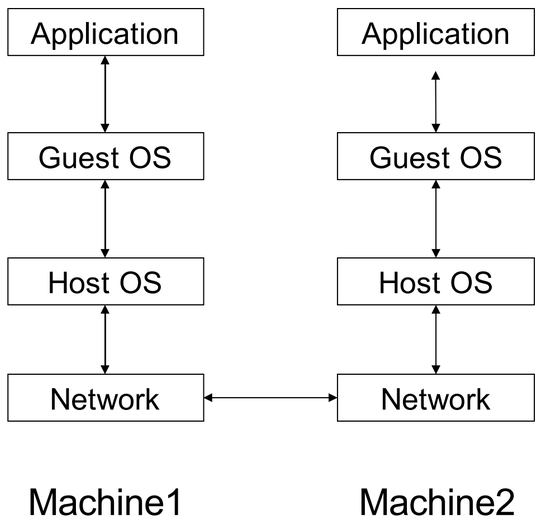
Confidential data are managed at multiple layers: application, os, network → Single level Flow Controls are not enough

The policies that rule the flows at different layers must be compliant → Information Flow Policy Compliance

Multilayer Checker

We developed an infrastructure to evaluate compliance in an architecture that enforces end-to-end information flow policies.

There are information flow policies at every layer: application, operating system and network. If virtual machines are included we also need to consider guest and host operating systems.



Compliance

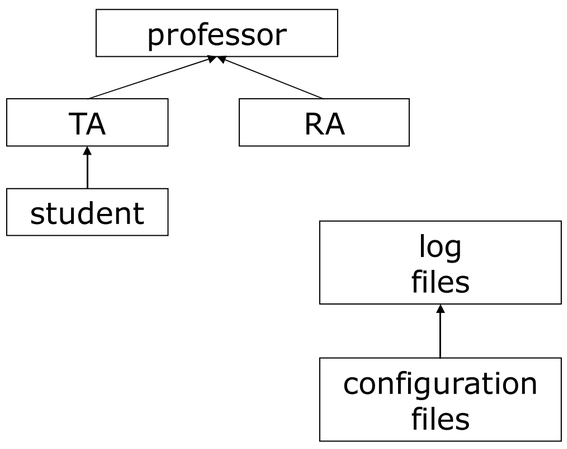
Given two information flow policies A and B. A is compliant with B, A B, if:

The set of information flows allowed by A is also allowed by B:
Flows(A) Flows(B)

Challenge:	Solution:
Different names for security classes that are equivalent.	Mapping
Security classes that exist in A and do not exist in B.	They do not introduce new flows in B
Security classes that exist in B and do not exist in A.	They do not need to be considered
Real applications need to re-label information in particular cases.	Agreement about declassifiers

IF Policies

Lattice models are used as a way of expressing Information Flow policies.



Challenges to evaluate compliance at multiple layers:

- Difference in semantics between policies.

- Difference in granularity (objects and permissions).

Translation to a common representation

Application & OS Policies

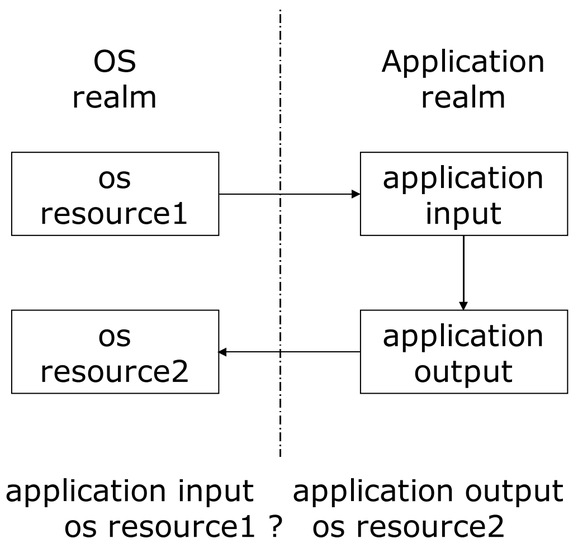
At the application layer we have Jif (Java Information Flow). Jif allows a programmer to incorporate Information Flow Policy Enforcement into an application.

At the os layer we have SELinux. SELinux uses Mandatory Access Control to enforce Information Flow Policies in the OS.

SIESTA

We developed SIESTA: System for Inspection and Execution of Security-typed Applications.

SIESTA evaluates compliance between Jif Policies (application layer) and SELinux policies (operating system layer). SIESTA authorizes an application execution only if the involved policies are compliant.



challenges:

- Verification that an application honors a given policy
- Compliance verification in case of partial mappings
- Execution of the application in the proper security range

Results and Future Directions

Publications:

B. Hicks, S. Rueda, L. StClair, T. Jaeger, P. McDaniel. A Logical Specification and Analysis for SELinux MLS Policy. ACM Symposium on Access Control Models and Technologies, SACMAT, 2007.

B. Hicks, S. Rueda, T. Jaeger, P. McDaniel. From Trusted to Secure: Building and Executing Applications that Enforce System Security. USENIX Annual Technical Conference, 2007.

B. Hicks, S. Rueda, T. Jaeger, P. McDaniel. Integrating SELinux with Security-typed Languages. Third Annual Security Enhanced Linux Symposium, 2007.

Future Directions:

Extension: Implement compliance evaluation for remaining layers.

Virtual Machines: Include VMs in the design and implementation.

Dynamic Tracing: Specific applications are statically authorized to assume various security classes. At run time the application must assume a single class thus constraining the set of resources the application is allowed to access later.

Sponsored by DTO