We want to protect the secrecy of confidential data.
Confidential data are managed at multiple layers: application, os, network.
The policies that rule the flows at different layers must be compliant.

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 \(\leq B\), if:

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

\[
\text{Flows}(A) \subseteq \text{Flows}(B)
\]

Challenge:
- Different names for security classes that are equivalent.
- Security classes that exist in A and do not exist in B.
- Security classes that exist in B and do not exist in A.
- Real applications need to re-label information in particular cases.

Solution:
- Mapping
-They do not introduce new flows in B
- They do not need to be considered
- Agreement about declassifiers

Results and Future Directions

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

Publications:

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.