Automating Security Mediation Placement in Legacy Code

Divya Muthukumaran, Dave King and Trent Jaeger

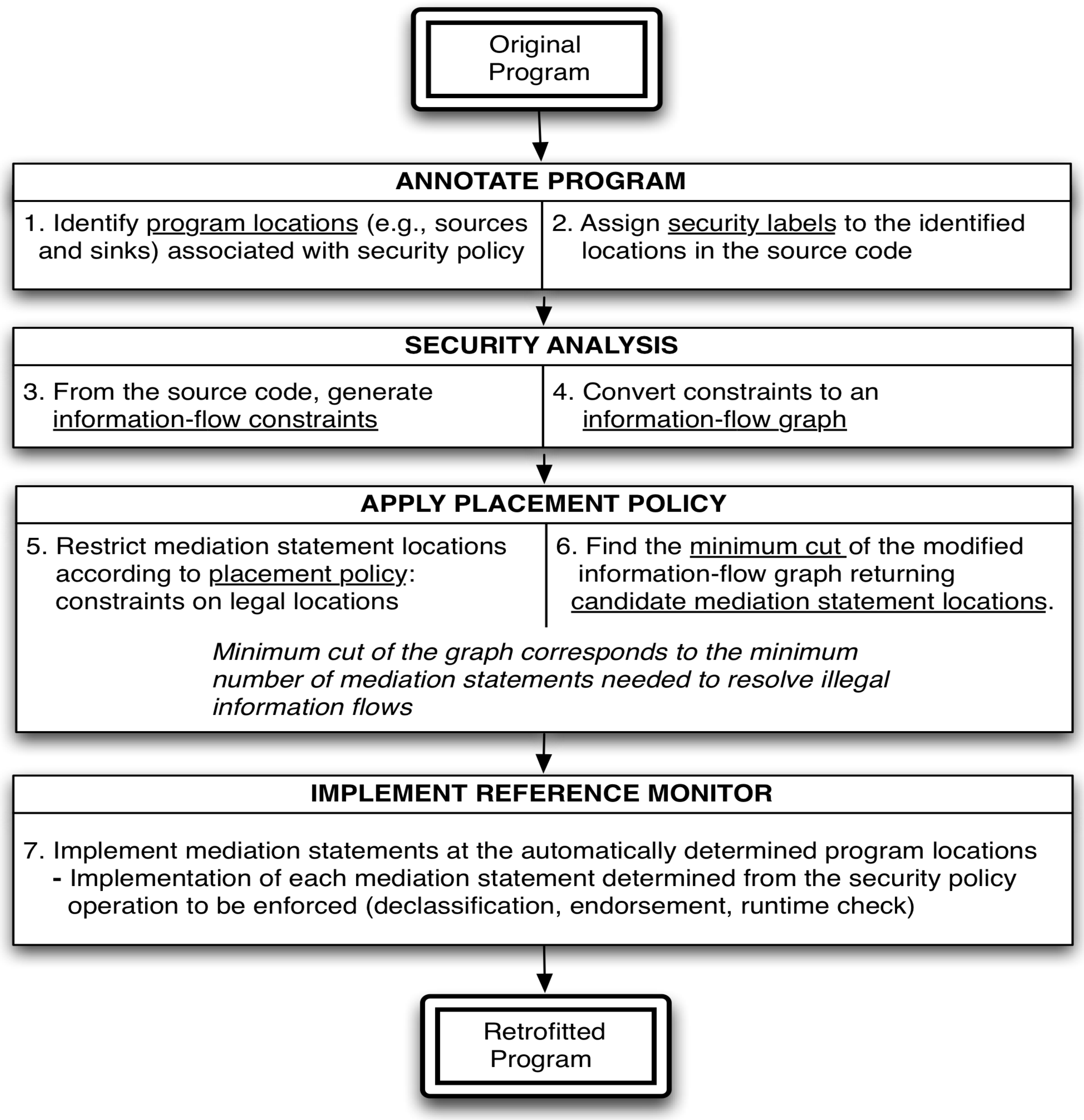
The Problem

Many programs make runtime security decisions about whether to accept untrusted inputs, provide access to program data, or perform operations on the behalf of request. Security type systems add security labels to data and statically verify that a program satisfies a security property based on these labels. For example, processes (servers and browsers) determine what operations an input request is authorized to access and what data can be returned.

- Programmers therefore need to **insert mediation statements** viz., de-classifiers, endorsers , authorization hooks to ensure that at runtime, the behavior of the program is in accordance with the **security policy** expressed by the type system and labels.
- Currently, these mediation statements are placed manually. This tiresome process involves the programmer examining large volumes of code and can be prone to errors.
- The goal of this work is *"to place such mediation statements **automatically"***.

The Retrofitting Process

- We model the mediation problem as a graph **min-cut problem** and the minimum cut corresponds to the **minimal set of mediation points** where mediation statements need to be placed.


Original Program

ANNOTATE PROGRAM

1. Identify program locations (e.g., sources and sinks) associated with security policy

2. Assign security labels to the identified locations in the source code

SECURITY ANALYSIS

3. From the source code, generate information-flow constraints

4. Convert constraints to an information-flow graph

APPLY PLACEMENT POLICY

5. Restrict mediation statement locations according to placement policy: constraints on legal locations

6. Find the minimum cut of the modified information-flow graph returning candidate mediation statement locations.

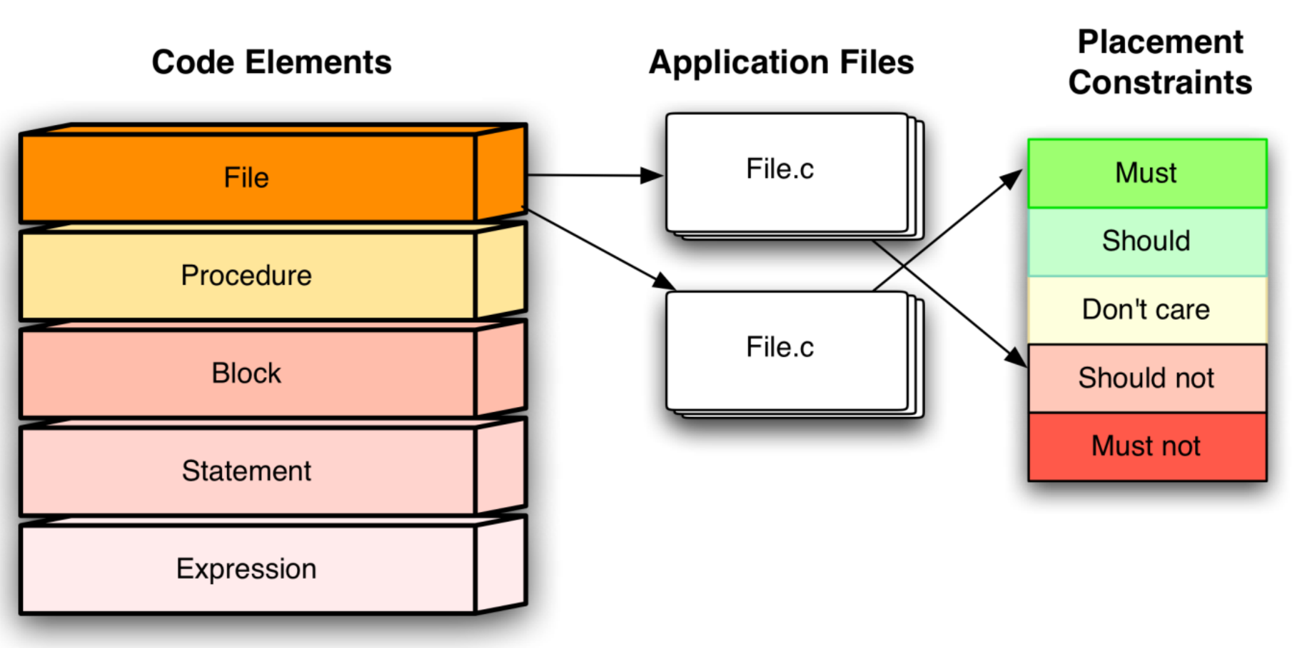
Minimum cut of the graph corresponds to the minimum number of mediation statements needed to resolve illegal information flows

IMPLEMENT REFERENCE MONITOR

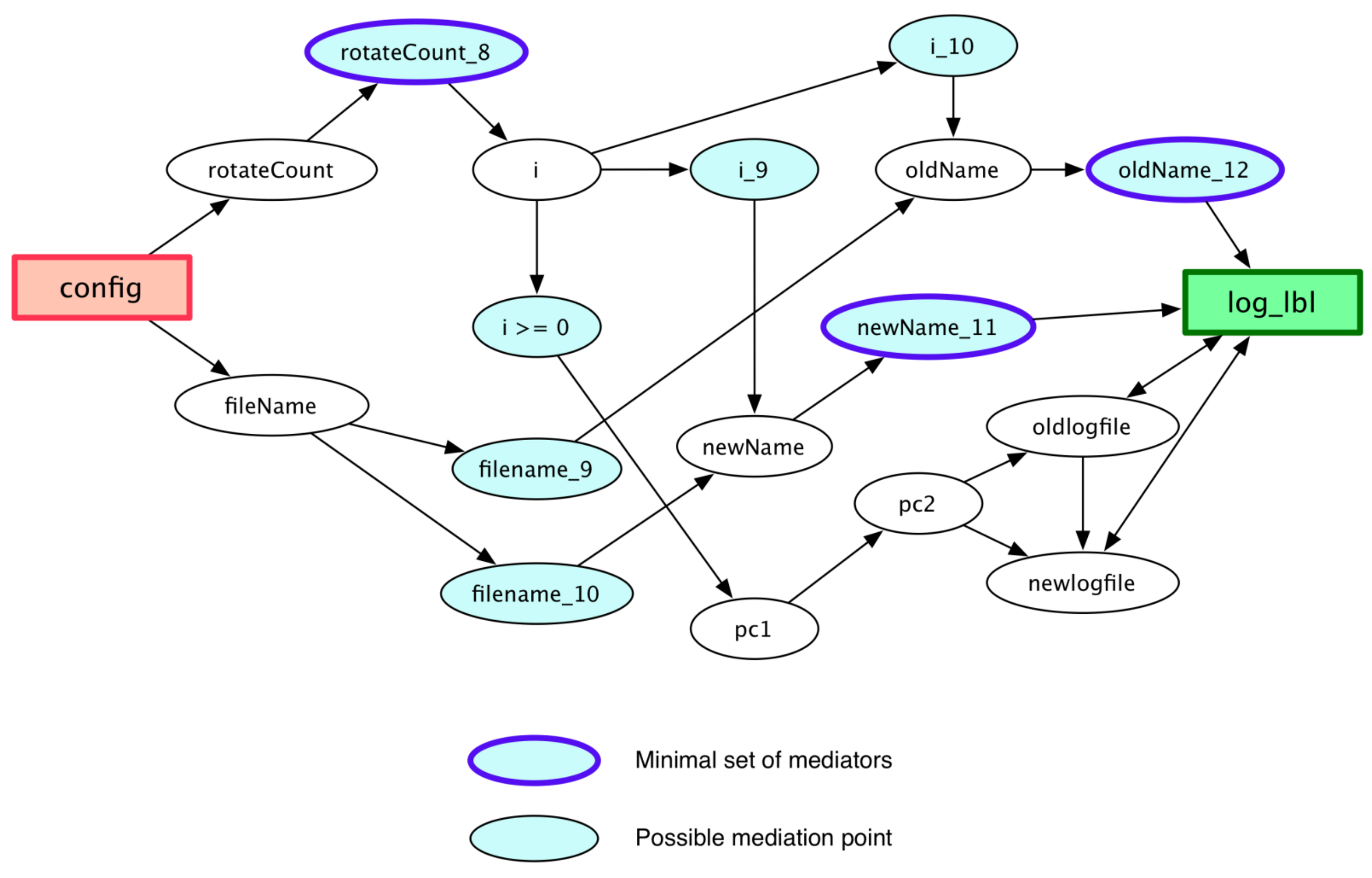
7. Implement mediation statements at the automatically determined program locations
- Implementation of each mediation statement determined from the security policy operation to be enforced (declassification, endorsement, runtime check)

Retrofitted Program

- We need to **customize the graph to account for programmer preferences**. The programmer will, for instance, be able to specify his like/dislike for a particular Code Element to bear a mediation point. Also runtime analysis like profiling will help in better mediation suggestions , for instance, avoid placing mediators in functions that are used very often. This will reduce the performance impact due to mediation statements.



- Example: The figure shows the information flow graph for the **logrotate** program and highlights possible mediation points. The suggestions will consist of a **minimal set** of mediation points.



Tool for C programs

- We currently have this tool working for JAVA programs. But we found that there are many legacy C programs that need this kind of mediation. SELinux/MLS identified 34 programs that are trusted by SELinux to enforce MLS on their execution.
- We intend to apply the retrofitting tool to these programs. Some applications like dbus, gconf, etc., already have mediation hooks. We can compare automated retrofitting with manual retrofitting.
- We use C Intermediate Language (CIL) to derive constraints from C code. CIL provides many powerful tools like points-to analysis, that we can use. The constraints are then fed to the mediation point suggestion tool (Ocaml and C++ versions), which outputs mediator suggestions.

Issues to address

- Converting mediation points to mediation statements: The programmer needs to place **mediation statements** based on the suggestions. This is program specific, e.g., adding filtering interfaces to transform low-integrity data to high-integrity data. Mediation points themselves have no security semantics, therefore, automating the actual placement of statements is difficult.
- Mediation can be done at different program levels – We can check policy at a higher program level e.g., instead of mediating a single expression, we can mediate a block of expressions together –> **Hoisting**
- We need to prune the candidate mediation points **to restrict results to legal sites,** for instance, to check whether the subject, object and operation are in scope. Program analysis can help with that.

NSF Funded project with Michael Hicks (Maryland), Ninghui Li (Purdue), and Somesh Jha