Review "DIFC"

Name: Wenhui Zhang

Paper Name：

Harris, W. R., Jha, S., & Reps, T. (2010, October). DIFC programs by automatic instrumentation. In *Proceedings of the 17th ACM conference on Computer and communications security* (pp. 284-296). ACM.

Contribution:

In this paper proposes a method of Decentralized Information Flow Control with implementation in Operating System, which allows programs to control flow of their data throughout the entire system. In this paper, there are 3 main contributions: (1) from high-level policies to DIFC code; (2) efficiently generate DIFC code; (3) Provide useful debugging information.

Motivation:

Application runs on a system that supports Decentralized Information Flow Control, or DIFC, then the application can use the system to support its policy. The challenge now though is that the correspondence between policy and the code the defines it is less direct. We may have the code the defines a policy spread out over application, possibly in code that runs in multiple processes. Moreover, this code makes use of a low-level API to try to implement a policy that's most naturally thought of in high-level terms of information flowing between processes.

Related works:

Asbestos, HiStar and Flume are all related works, however they do not provide constraint solver and sanitization of policies, policy generating are depends on individual programmers.

Methodology:

A DIFC system works by mapping every process in the system to label. In general, a label is an element in some partially ordered set. This paper assumes that labels look they do in the Flume DIFC OS. In Flume, every label is a set tags, where a tag is just an atomic element that can be created whenever by any process.

The DIFC OS then enforces an information flow policy by checking communications based on labels, as well as checking how each process changes its label. In the case of Flume, the OS requires that if one process tries to send information to another, then the label of the sender must be a subset of the label of the receiver.

Note that a process may change its label over its execution, in a restricted way by adding a removing tags to the label. Intuitively, the ability to add tags corresponds to the ability to read sensitive data.

Results:

It performs Swim (for Secure What I Mean) on a set of real-world programs and policies.

Take away:

There is a semantic gap from policy to DIFC code, and instrumenting legacy code without conflicts between policies across different programs is a challenge.