Review “Share Mobile Code”

Name: Wenhui Zhang

Paper Name：

Arden, O., George, M. D., Liu, J., Vikram, K., Askarov, A., & Myers, A. C. (2012, May). Sharing mobile code securely with information flow control. In *Security and Privacy (SP), 2012 IEEE Symposium on* (pp. 191-205). IEEE.

Contribution: https://www.cs.cornell.edu/projects/fabric/

This paper proposes a new architecture for secure mobile code in both compile time and run time. Through this architecture, code could be secure-shared across different domains. In this paper a language for remote calls, nested transactions and security annotations is developed. Also, a system with secure transparent data shipping, secure remote calls, secure federated transactions, and enforcement of security labels is implemented.

Motivation:

To balance security and functionality, people developed distributed information control flow for application policy oriented enforcement across different domains, such as Fabric. Fabric is a language and runtime system that supports secure federated distributed computing. However it does not handle code provided by adversaries with adversaries controlled tags.

Related works:

Fabric does not supports secure mobile code. DStar does not support code integrity or secrecy, also publishing and installation of code does not work in this system. CORS allows websites to have their policies, and enforce policies across domains, however users have no control over this process. Caja assumes a static analysis of information flow, and could not afford dynamic updates of policies.

Methodology:

Nested transactions ensure that computations observe and update objects consistently, and provide clean recovery from failures. – Remote method calls (remote procedure calls to methods) allow computations that are distributed. – Remote objects are accessed transparently, as if they are local objects. – Mobile code allows program components to be dynamically downloaded and executed.

Results:

The implementation is secure based on the author’s assumptions. The implementation is expressive and could be applied to various of applications. The implementation is performance tolerance in some situations.

Take away:

This paper assumes that adversaries could not mess up with network messages and change tags etc. from there, which is not always the case. Also, there might be some time of check at time of write security issues brought up by race conditions between threads.