[security -> policy modifies]

The system developers often separated security policies from the actual functionality of the system. Separation access control policies from the actual functionality of the system. of software and security concerns is a popular approach for developmenting and maintaining access control policies, which simplifies policy maintenance and evolution. The access control mechanism is based on a simple yet powerful model for the interaction between a policy-neutral enforcement manager and a given security policy, and it is transparent to extensions and the core system services in the absence of security violations.

When security requirements of software are modified during development and maintenance, system developers change security policies, which used to govern which subjects can be permitted or denied to access which resources in which conditions. For example, new security concerns are added to security policies, or when an existing security policies is modified without changing program code according to the modified security requirements.

Since program code is interacting with policies through policy decision point (PDP), the program needs to be regression tested. Access control policies can interacting with application code and impact the behavior of the code.

For example, without any changes to the origianl code, adding a single rule in access contol polciies change the pre- and post-conditions of methodes. Regression testing is important for security policies due to the its effects of small code chagnes. This paper focuses on on the problems what are safe and precise regression-test-selection techniques for systems built with Java and security policies? In order to increase confidence that the modification is correct and not introduce unexpected behavior, developers often periodically conduct regression testing.

[system test cases][policy changes]

For regression testing, developers often write an initial test suite and reuse a test suite. The na?ve regression testing strategy is to rerun all test cases in the initial test suite. However, this strategy is costly and time-consuming, especially for a large system. Moreover, if the size of the initial test suite is large, this strategy may require a significant time for system developers to conduct testing. In order to reduce the cost of regression testing, system developers often adopt regression test selection, which selects and execute only test cases to expose different behaviors across versions of software systems. This approach also requires substantial costs to analyze software and select such system tests from the initial test suite that could reveal faults introduced by the changes. A precise and safe test selection technique helps reduce time-consuming task needed to valiedate whether the modification are correctly aplied. Our paper focus on a safe regressiontest-selection, which selects every test case that may reveal a fault in the modified software.

Of particular interest for our work is the technique proposed by Harrold et al. [8] for regression test selection for Java based on comparisons of control-?ow graphs (CFGs). Given a program P , regression tests are executed to build an dge-coverage matrix which maps each test case to the set of CFG edges exercised by that test case. For a subsequent modi?ed program version P 0 , the CFGs of P and P 0 are compared to identify “dangerous” edges in the CFG for P . These edges represent program points at which P and P 0 differ. All and only test cases for P which cover dangerous edges are selected for testing of P 0

The execution of functions in the business logic should thus include calls to the PDP, which grants or denies the access to the protected resources/ functionalities of the system. From a security testing point of view, the issue is to determine the correctness of the interactions between the business logic and the PDP. The generation of test cases targeting security thus consists in generating a test sequence ending with a given access control call, and also relies on the creation of the oracle function.

the recommended architecture consists of designing a security component, called the Policy Decision Point (PDP), which can be configured independently from the rest of the application. The PDP is configured with respect to a security policy, modeled using an access control modeling language such as OrBAC or RBAC [2, 3] A security policy is composed of a set of access control rules, which specify the conditions for granting or denying access to protected elements. The execution of functions in the business logic should thus include calls to the PDP, which grants or denies the access to the protected resources/ functionalities of the system. From a security testing point of view, the issue is to determine the correctness of the interactions between the business logic and the PDP. The generation of test cases targeting security thus consists in generating a test sequence ending with a given access control call, and also relies on the creation of the oracle function.