Visualization of Context Sensitive Data Flow for Secure Object-Oriented Programming

Young Lee* , Jeong Yang

young.lee@tamusa.edu
Department of Computing and Cyber Security
Texas A&M University-San Antonio, San Antonio, TX, USA

Abstract – We propose a unified visualization of context-sensitive data flow of Java Program to help a programmer to locate the security data leak and to understand what caused the leak. There are a few efforts on how to visualize the security leak detection results even if those security tools used the Program Dependence Graph (PDG) internally. Proposed unified visualization technique is consisting of a static structure of the source code and dynamic behavior of the running Java program.

Keywords: Visualization, System Dependence Graph, Static Code Analysis, Control Flow, Data Flow

1 Introduction

Numerous studies are focused on generating precise PDG to detect security leak. The informative feedback to the programmers for the security tracks is as important as the security detection correctness such as less fault alarm. As for our best knowledge, little research has been done on how to visualize the security leak detection results even if those security tools used the PDG internally.

This paper proposes a united dependence visualization using multiple views for the target Java program for secure coding practices. Visualizing secure data flow helps programmers not only to locate the secure data leak but also to understand what caused the leak. Therefore, programmers can fix the program to remove the leaks using the visualized information flow.

Visualizing the interactions among objects is required to understand the data flow in object-oriented software. This visualization includes an object's state information in a collection of variables, and object's behaviors implemented by methods in the presence of inheritance, polymorphism, and dynamic binding [1]. Unified visualization aims to help students better understand how to develop a secure program by analyzing static source code and visualizing dynamic behavior using step-by-step synchronized control and data flow of a Java program.

2 Literature Review

A PDG describes the intra-procedural data dependencies on the control flow. A System Dependence Graph (SDG) extends the dependencies of PDG for a single procedure to the multiple procedures by analyzing their inter-procedural calls.

Both graphs are results from the static program source code analysis and contain nodes and edges representing each statement and dependencies among them. In recent years, researchers have become increasingly interested in information control flow using PDG to detect illegal flow, a violating path which might be a security leak.

Hammer and Snelting have applied static analysis with PDG to gain context and object-sensitive information flow control (See Figure 1 and 2) [2].

class PasswordFile {
    private String[] names;
    private String[] passwords;

    public boolean check(String user, String password) {
        boolean match = false;
        try {
            for (int i=0; i<names.length; i++) {
                if (names[i]==user && passwords[i]==password) {
                    match = true;
                    break;
                }
            }
        } catch (NullPointerException e) {} catch (IndexOutOfBoundsException e) {} return match;
    }
}

Figure 1. A Java Password checker. Adapted from [2]

Figure 2. PDG for check in Figure 1. Adapted from [2]
A PDG are internal structures for the static program analysis not for the visualization. A PDG is becoming more complex to analyze the source code precisely, therefore visualizing the PDG would be incomprehensive to the programmers.

3 Context Sensitive Data Flow

Visualizing context sensitive, object sensitive data flow in Java is difficult in the presence of inheritance, polymorphism, and dynamic binding. This visualization includes an object's state information in a collection of variables, and object's behaviors implemented by methods that use those variables.

Through the context sensitive, object sensitive data flow visualizing, programmers get a better perspective of the behavior, and interaction of the objects, and tracing the secure data along with control and object flows of the Java code.

It is anticipated that the proposed united dependence visualization helps programmers in detecting an explicit and implicit flow of secure data by synced visualizing of the control and data flow. United dependency visualization combines PDG and dynamic dependencies among program components by program execution tracking (See Figure 3).

4 Conclusion and Future Work

Our final expected result is a security detection and visualization tool with experimental test results of its usability. Experimental verification would be the final step of the project.

First, we will define more precisely relations between PDG and combination of united visualization views such as control flow, data flow, and object flow. Second, we will implement secure data flow tracing program. Third, we will integrate the data flow visualization to the existing static and dynamic visualization tool, JaguarCode [3]. Finally, we will analyze the united visualization technique to determine whether it is more effective than existing visualization techniques to locate and understand the security leak in Java programs.

References

