Research on the Static Analysis Method of the Localization Embedded Platform Software Code

Zhijie Gao[a], Ling Lu, Wen Jiao

Software Testing Center of the 96658 Troops, Beijing, 100094, China

[a]email: zjiegao@163.com

Keywords: Static Analysis; Formalization; RELAY Algorithm;Testing Framework

Abstract. Localization of embedded system software usually has the characteristics of component, multi thread. Dynamic test is difficult to achieve full coverage, through static analysis techniques for automatically check source code, source code in the presence of common software defects found, can thoroughly solve specific types of software defects, effectively improve the localization platform software system reliability. In this paper, based on the formal method of static analysis, the algorithm of solve pointer alias analysis problem, puts forward the static analysis to test the framework for localization platform software code to carry out the static test and provide an effective way.

Introduction

At present, the software testing technology which is applied to the localization platform is still in a backward state. How to make the full test of the embedded platform software is an urgent task in the field of software testing. With the increasing complexity of embedded system software, dynamic testing is difficult to meet the requirements of test adequacy.

Static analysis can be in an early stage of development is found effective to all kinds of common defects exist in the source code, can also in the completion of the development of the source code of test system, through static analysis techniques for automatically check source code, source code in the presence of common software defects found, to thoroughly solve the specific type of software problems, effectively improve the software reliability.

However, static analysis of effectiveness and software source code characteristics have direct relation, serious affected the encoding characteristics, existing mature testing tools cannot be used in localization of embedded platform, so we need to research on localization of embedded system software code features of efficient and feasible method for the static analysis.

Static Analysis Method of Software Code

Static analysis is a static test method for the direct object of the source program, which can verify whether the program source code meets the code programming code and the implementation rules, and can directly map the nature of the program structure. The static analysis method does not need the actual operation procedure, can quickly discover the software potential problem and each kind of flaw.

Static analysis can usually be at compile time analysis of program structure to detect potential errors and do not meet the requirements of the code structure, in the development process can be synchronous reported a series of errors, to facilitate defect rapid correction, thereby reducing the development costs. Commonly used static analysis techniques include data flow analysis, control flow analysis, abstract interpretation [1], and analysis of types and effects [2].

Static analysis can detect the following typical software defects:

1) read the initialization variable: reading the data that is not initialized will lead to uncertainty. Static analysis tools are able to locate the code positions that use the non initialized data.

2) access violation without protecting shared variables.

3) the reference pointer NULL or out of bounds.

4) illegal arithmetic operations (negative open square).

5) integer and floating point operations overflow: in the integer calculation, when the results of the calculation and storage variables can not be compatible, it will occur overflow, can not be expressed in memory. Static analysis can be used to locate and find the problem.

6) array bounds, except the zero error, unreachable code and illegal type conversion.

Static analysis method is divided into many types, according to the analytical methods are used widely and analysis method between similar, which can be divided into three categories: basic analysis, based on the analysis of the formal analysis and other auxiliary. Among them, the basic analysis including syntax analysis, type analysis, control flow analysis, data flow analysis, is most compilers are contained in the analysis process, based on the formal analysis is mainly on the analysis of the process by some mathematical mature formal methods, to get close to the code more accurate or more extensive properties; other auxiliary analysis contains a number of separate analysis for the basic analysis, based on the form of analysis to provide support.

(a)Basic Analytical Method

The basic analysis, including syntax analysis, type analysis, control flow analysis, data flow analysis, and so on, all kinds of analysis of different concerns, can be mutually verified.

1) Syntax Analysis

Syntax analysis is an important step in the compilation process, as well as the premise of most other analysis. Grammar analysis is the process of analyzing and processing the results generated by the lexical analysis program according to the grammar rules of the specific programming language and the process of generating the parse tree. It can be judged that the program is consistent with the pre-defined pattern in the structure, that is, whether there are any grammatical errors in the program. Support syntax analysis technology mainly includes operator priority analysis, recursive descent analysis and LR analysis, etc..

2) Type Analysis

Type analysis is mainly refers to type checking. The purpose of type checking is to analyze the type of errors in the program. Wrong type usually refers to the violation of the type constraints, such as two string multiplication, an array of cross-border access etc.. Type checking can be static, can also be carried out dynamically. Compile time type checking is a static check. For a programming language, if all of its expression types can be determined by static analysis, and then eliminate the type error, the language is a static type language, or a strong type language. Programs that are developed using the static type language can eliminate many errors before running the program.

3) Control Flow Analysis

Control flow chart is a graphical representation of all the paths that may be passed during the execution of a program. The goal is to obtain a control flow graph of the program. According to the relationship between different statements, especially considering introduced by "transfer", "Circular" branch relations, the statements are merged, some results about the structure of the program can be obtained. The control flow graph is a directed graph, the nodes in the graph correspond to the basic block of the program, and the edge of the graph corresponds to the possible branch direction.

4) Data Flow Analysis

Data flow analysis is mainly to determine the use of a certain statement in the program on the use of the various variables or possible values, the general flow of control from the beginning of the program. Data stream analysis includes forward analysis and backward analysis.

(b) Based on the Formal Analysis Method

Analysis based on formal methods is an extension of the basic analysis techniques, mainly to improve the accuracy of the analysis, to obtain more properties on the program. The representation of the formal analysis method mainly has the constraint solving and abstract interpretation.

1) Constraint Solving

The main idea of program analysis technique based on constraint solving is to translate the program code into a set of constraints, and to obtain the solution of satisfying constraints by the constraint solver [3]. For constraint solving, studies show that for path test data generation can be well attributed to constraint solving problem, and analysis of program invariants can also be attributed to constraint solving. Program analysis constraint problem is formulated as constraint solving problem, that is, from the program obtained usually use first-order or second-order said further convert it into a form of constraint solvers can handle, support constraint solving software analysis tool for SAT/SMT solver [4].

2) Abstract Interpretation

For abstract interpretation of the program code eventually program established an abstract model, is to use abstract domain object approximation calculation program object field calculation, the abstract program execution results can reflect the information of real programs running. Abstract interpretation is to calculate the accuracy of the calculation of the accuracy of the calculation of the feasibility, in the calculation of efficiency and accuracy to achieve a balance between the iterative calculation of the accuracy of the method to enhance the accuracy of the method. If there is no error in the abstract model, it is proved that there is no error in the source program.

Formal analysis method in the analysis of a software and need to first formalized description of the nature, then the description and software code together as input provides to the analysis tool, the software is given the nature of the analysis results. The formal method based on abstract interpretation theory has been widely applied to large scale software and hardware system verification. It is an effective way to analyze and verify the large-scale software and hardware system.

(c) Other Auxiliary Analysis Method

Other auxiliary analysis method mainly includes the analysis method, which is based on the analysis of the execution and the slice analysis, which provides the support for the basic analysis and the formal analysis.

1) Symbolic Execution

Symbolic execution is performed by using an abstract symbolic representation of the value of a variable in a program to simulate the execution of a program [5]. Consistent with the implementation of more close to the actual implementation of the program, to provide accurate, and the defect related to the context of information. Because the implementation is through an exhaustive search of all possible execution path, the need to deal with the workload and exponentially with the level of growth with the increase in the size of the procedure.

2) Slicing Analysis

Slice analysis for the extraction of interest points in the program from the source of the specific variables have influence on the statements and predicates, the composition of the new program, known as the slice, through the analysis of the behavior of the source program [6]. There are two main methods of slice analysis. According to the calculation of the data flow equation and the calculation of the dependence graph. Slice analysis technology has been widely used in the process of program debugging, analysis, testing and software maintenance.

Program Pointer Alias Analysis Based on RELAY Algorithm

Different static analysis methods are needed to deal with the localization of embedded system software which has the characteristic of component and multi thread. RELAY algorithm can effectively solve the problem of pointer alias analysis.

(a) Pointer Alias Analysis Problem

Because of the increasing complexity of the embedded system software, the core of the data competition detection in the concurrent multi thread program is the lock collection algorithm, which is mainly through the context sensitive pointer alias analysis. At present, most of the lock collection algorithms in most of the existing tools are also realized, but there are also the problems of determining whether the memory cells accessed by multiple threads are the same storage unit.

This problem is difficult to solve in the static analysis, even in the absence of dynamic memory allocation, the only pointer to the existing memory unit, there is still the problem. The problem is mainly due to the flexibility of the application of pointer in C language. For example, even if the pointer P1 and P2 does not exist alias, but does not guarantee P1->data and P2->data does not exist between alias, multilevel pointer will make the alias analysis problem is more complicated, the similar problem is data race detection in the key. For general data competition, the program designers will use the lock mechanism to protect the data to avoid competition. But this through the alias data competition, sometimes programmers is not clear what pointers is lead to alias data competition, the competition of this kind of data may cause some program point data inconsistency, which lead to software failure or error.

(b) RELAY Algorithm Implementation Pointer Alias Analysis Steps

RELAY algorithm is the United States W. Voung J. and other research points to the relationship between the analysis method [7], can be analyzed to get the basis of pointer information and alias information. For any function, the calculation of the relevant lock set can be realized by five steps:

1) Code Compilation: mainly for syntax analysis, the compiler process is an important step in the implementation of static analysis, but also the basis for the subsequent analysis.

2) Symbolic Execution: the aim is to acquired two analysis of relevant data, which includes preserved through the implementation of specific path information to get memory access protection lock set situation, and lock application and release.

3) Lock Eet Analysis: using symbolic execution results, the relevant set of locks changes are calculated, even order lock is used to represent the relevant set of locks L, L_+ represents the lock added on the application in a function and need to in the global lock set to add the new application lock, the L_- represents the lock release operation in the function, needs on a global lock set will remove the lock.

Example: In a function, lock(&l) corresponding summary information is $\langle L_+, L_- \rangle = \langle \{l\}, \emptyset \rangle$, unlock(&l) corresponding summary information is $\langle L_+, L_- \rangle = \langle \emptyset, \{l\} \rangle$. L_f represents a set of functions, At the end of the function, the information of the lock set information is the information of the whole function f.

The specific condition of the lock set at the end of the execution of the function can be calculated by the L_f when the corresponding different input parameters are calculated. The relevant change information of the lock set can be obtained by using the summary information of the function f. Function f call $f()$, Summary information for the existing function f is represented as $\langle L^{0}_{+}, L^{0}_{-} \rangle$, The lock set for the current function call point is $\langle L_+, L_- \rangle$, at the end of the function call, the corresponding lock set becomes:

$$\langle L_{f+}, L_{f-} \rangle = \langle (L_+ \cup L^{0}_{+}) \setminus L^{0}_{-}, (L_- \cup L^{0}_{-}) \setminus L^{0}_{+} \rangle \quad (1)$$

Through the analysis of the lock set expression, first call function f in the application of the lock set to join the total application of the lock collection, and then in the total application lock set to remove the lock in the function f in the release of the lock. Another condition is that the lock is released in the function f to the current total release lock set, and then the lock set in function f is removed from the total release lock set. In addition to the function of the unlock process, all of the process information is hidden, can only see the corresponding results information, that is, the last lock is maintained or application, which lock has been released.

4) Shared Variable Protection Lock Set Analysis: Calculation of the various shared variables access to the protection of the lock, Using three tuple $g = \langle o, L, rw \rangle$ to represent the protection of a lock on a shared variable access. Among them, o represents the identity of the shared variable that is accessed, and the shared variable has been analyzed by an alias. L represents the specific lock collection information when accessing the o. rw indicates the nature of the access to the shared variable o, that is, read or write. The calculation of the information of the shared variable lock set can also be carried out using the method of the abstract.

5) Results Analysis and Rreatment: according to the set of rules and the protection of the lock set analysis results, the results of static analysis.

In the above five steps, lock set analysis and shared variable protection lock set analysis is based on the results of symbolic execution, implementing procedure function pointer alias analysis through five steps can be, so as to ensure the accuracy of the results of static analysis.

Software Code Static Analysis Framework

According to the characteristics of the embedded system software source code, the use of formal static analysis method to find the software in the numerical related defects and memory related defects. Numerical related defects include the addition of zero error, integer overflow, memory related defects such as buffer overflow, empty pointer references, illegal address access, etc.. Localization of the embedded platform software code static analysis, including abstract interpretation, pointer analysis, structure diagram figure 1.

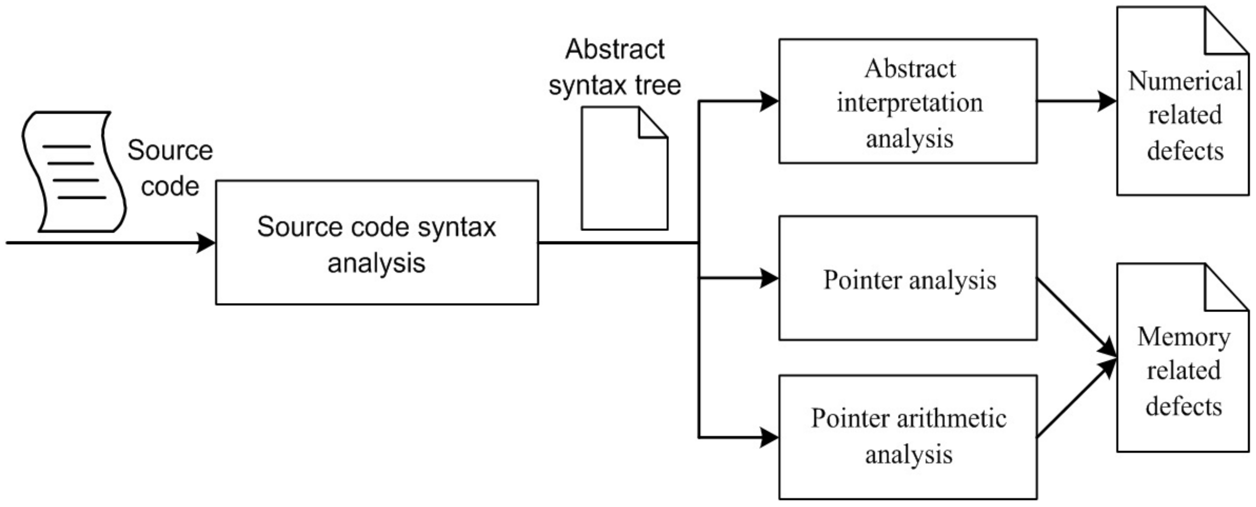
Source code
Abstract syntax tree
Source code syntax analysis
Abstract interpretation analysis
Pointer analysis
Pointer arithmetic analysis
Numerical related defects
Memory related defects

Fig.1. The Software Code Static Analysis Framework

Static analysis methods based on abstract interpretation, including machine integer programs and automatic analysis methods of floating point programs, are mainly based on the analysis of. The discrete uniform distribution of machine integers program machine integers is, there is a limited range, the value range over fixed binary digits of the integer will lead to a machine integer overflow. Floating point numbers in floating point programs are non uniformly distributed, and floating point operations are introduced into floating point error. Abstract interpretation can be effective in the analysis of machine integer and program floating point program, and the analysis results are of high reliability.

Memory related defect testing is mainly based on the formal method of pointer analysis, including pointer analysis method and pointer arithmetic analysis method. In C/C++ program are generally widely use pointer to memory value directly operation, involving a large number of pointer dereference, and pointer arithmetic operation, so there is null pointer references, illegal access address and memory defects. Aiming at the localization of embedded system software, the establishment of a pointer to a memory model, usually the "base + relative offset + base type" memory model to describe the pointer information. Through the relay algorithm get a pointer to the base address information and alias information.

Conclusion

The advantage of static analysis technique to analyze the essence and defect detection analysis, expounds the basic analysis, characteristics and differences of three kinds of static analysis method based on formal analysis and other auxiliary analysis, for data race detection issue at the core program. Use of numerical method is used to establish the pointer alias analysis solutions, the proposed static analysis to test the framework for development of static analysis tools provide solutions, help to improve the software static analysis accuracy.

References

[1] Cousot P, Cousot R. Abstract Interpretation: AUnified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints[C]. Proceedings of the 4th ACM SIGACT-SIGPLAN, 1977:238-252.

[2] F. Nielson, H. R. Nielson, C. Hankin. Principles of Program Analysis. Springer, 1999.

[3] Gulwani S, Srivastava S, Venkatesan R. Program analysis as constraint solving. Proceedings of the SIGPLAN Conference on Programming Language Design and Implementation. Tueson, Arizona, 2008: 281-292.

[4] Een N，Sorensson N．An extensible SAT-solver. Proceedings of the 6th International Conference on Theory and Applieations of Satisfiability Testing．LNCS 2919．Santa Margherita Ligure，Italy, 2003: 502-518.

[5] J. C. King. Symbolic execution and program testing. Communications of the ACM, 19(7):385-394, 1976.

[6] Weiser Mark．Program slicing．IEEE Transactions on Software Engineering，1984, 10(4)：352-357.

[7] J. W. Voung, R. Jhala, S. Lerner. RELAY: static race detection on millions of lines of code. In Proceedings of the the 6th ACM SIGSOFT (ESEC-FSE '07), ACM, New York, NY, USA, pp.205-214, 2007.