**Static v/s Dynamic Analysis & Secure static Analysis Compiler**

Mr. Jagadish. S. Kallimani

Associate Professor

M S Ramaiah Institute of Technology

Bengaluru-560054

jsk\_msrit@rediffmail.com

Manoj More S

M S Ramaiah Institute of Technology

Manoj8894s@gmail.com

***Abstract***

*A program can be compiled both using static and dynamic methodology, each of the method has its own advantage. Many patterns of unsafe usage can be discovered through static analysis. Dynamic analysis help during run time .Combination of both methodologies gives effective compilation result. This paper aims at providing the clear picture on static analysis over dynamic analysis, main advantage of static analysis over dynamic analysis, how aspect of static analysis is especially valuable? And what is “Secure Static Analysis”.*

1. **Introduction**

To illustrate the power of static code analysis we will use sample code involving memory management and the use of pointer constructs in various languages. Since these are arguably one of the most error prone and hard to debug features in wide-spread programming languages. While garbage collected languages have become a large part of the programming marketplace. Even when using a garbage collected language, they are often based on a Virtual Machine. Which may itself have memory leaks or other errors? By making use of static analysis you can now automatically find those mistakes. A static analysis tool is a program which parses then analyses you source code. This means that in your compiler the static analyzer acts has an additional small sub compiler. It does not produce binaries as such but produces an intermediate representation which is more suited than the source code to be analyzed.

1. **Static Analysis**

Static analysis helps to figure out code sequences that, when executed, could result in various errors like buffer overﬂows, resource leaks, or many other security related problem. These analyzers are effective at identifying a signiﬁcant class of defects that are not detected by compilers during standard builds and dynamic analysis. General static analyzers use the universal front end that is used to compile code. Logically it would be effective if static analyzers are integrated with the most commonly used compiler to maximize use and reduce complexity at the run time. Static analysis enables source code parsing to be performed only once instead of twice. The integrated front end takes advantage of pre-existing dataﬂow algorithms to perform its bug ﬁnding mission. The analyzer determines execution paths through code, including paths into and across subroutine calls, and how the values of program objects could change across these paths. The objects could reside in memory or in machine registers. Static analysis provides model which is straightforward for manipulation.

**2.1 Benefits of Static Analysis**

Warning about anti-patterns in code can be raised.

McCabe's Cyclomatic Complexity tells useful things about source code.

Memory leak and logical error are found before dynamic analysis.

Static analysis helps in determining cleanliness in code.

1. **Dynamic Analysis**

Dynamic analysis is performed by executing the program on target machine and observed for its behaviour. Effectiveness of the dynamic analysis can be obtained by making use of the sufficient test input on the target program; this process helps in obtaining exact runtime behaviour of the program during execution. Dynamic analysis takes place as a part of program execution hence dynamic analysis is as fast as program execution, although dynamic analysis runs quite faster it ends up with accurate result. Dynamic analysis lacks when large programs are analyzed then its result would not be applicable for future execution. Dynamic analysis will not guarantee that it covers all program execution based upon the test suites. Dynamic analysis blocks potential faults during execution. Reason for dynamic analysis is some information is available at the time of runtime; observe red behaviour of the system depends upon the execution platform.

**3.1 Benefits of Dynamic Analysis**

Provides accurate analysis results. Detect bug during runtime on target environment and also focus on things that were ignored by static analysis.

**3.2 Limitation of Dynamic Analysis**

Difficult to trace the problem back to the exact location in the code.

Analysis is only as good as the tests you write\implement and run.

1. **Static v/s Dynamic Analysis**

The difference between static and dynamic analysis is static analysis takes place before runtime and dynamic analysis takes place during runtime. Static analysis is used to perform pattern check and verify a program for all possible execution paths before the actual execution. Dynamic analysis includes the technique of verifying the system properties for particular execution trace. Static and dynamic analysis complement each other for test case generation i.e. static analysis smartly generate test case and dynamic analysis tool finds the errors using those test cases.

The difference between static analysis and dynamic analysis is over-emphasized. Complementarily can be achieved by applying both approaches and taking advantage of the “soundness” of static analysis while also benefiting of the “efficiency and precision” of dynamic analysis [2].

Both static and dynamic analysis is very important for runtime monitoring .Most of the current compiler include both static and dynamic analysis technique in them i.e. they incorporates the advantages of static and dynamic together to obtain the optimal result.

1. **Secure Static Analysis(SSA)**

Static security analysis is performed by the Intel C++ and the Intel FORTRAN Compilers operating in a special mode “Static security analysis (SSA) attempts to identify errors and security weaknesses through deep analysis of source code.” [1]

SSA is an effective way to detect defects for the code where test suites are not clearly defined. SSA also detects race around condition in parallel program methodology.

Unsafe patterns are discovered through static analysis. The main advantage of static analysis over dynamic analysis is that it examines all possible execution paths and variable values. This aspect of static analysis is especially valuable in security assurance; SSA can detect over 250 different error conditions.

Most of the time is spent in error analysis and instruction generation process by the modern compilers, SSA requires our code to go under Intel compiler for avoiding serious errors Intel compiler to create your production binaries in order to take advantage of SSA.SSA can be done on practical program but to obtain best results it better to run on entire program.

Once the program is processed under SSA technique the result is obtained on the Intel inspector which also stores the state information in the result file. By Intel Inspector XE GUI you can decide whether the bug fixing is necessary .once the bug are fixed the source changes and again the analysis is done then the result between previous and current run in Intel inspector.

1. **Conclusion**

As of above fact both static and dynamic analysis is necessary for runtime monitoring. Both approaches has its own merits and demerits, importance of each is combined and effective results are obtained by the advanced compilers, static analysis works before execution of the program and one works at the time of the runtime.

Dynamic analysis works on the different paths that are being tracked by static analysis before runtime. Static analysis tracks all possible before runtime but dynamic analysis monitors the program using single path which makes dynamic analysis more efficient compared to static.

SSA is a new way of compiler implemented by Intel. Where the compilation process is given the graphical touch. This technique gives a clear picture of result in GUI manner and comparison of different interval of results.SSA is effective technique of compilation.

**Reference**

[1] INTEL PAR ALLEL STUDIO XE EVALUATION GUIDE, Static Security Analysis (SSA).

[2] Mohd. Ishrat1, Manish Saxena and Dr. Mohd. Alamgir “Comparison of Static and Dynamic Analysis for Runtime Monitoring”, ISSN: 2249-5789, Oct-Nov2012

[3] Marc de Kruijf, Karthikeyan

Sankaralingam, Somesh Jha

University of Wisconsin – Madison

{Dekruijf, karu, jha}@cs.wisc.edu “Static Analysis and Compiler Design for Idempotent Processing”