Name: Prashanth Mallyampatti
Unity ID: pmallya
Student ID: 200250501

CSC724- Advanced Distributed Systems
Paper Review

Do Not Blame Users for Misconfigurations

Summary:

This paper addresses the problems associated with configuration errors which cause a system to fail. Configuration errors are bugs in a system wherein the programmer hasn't handled it or not considered it to happen. These errors manifest slowly and contribute for crashes, hangs, silent failures. Due to lack of proper error messages users know very little about these errors, its the programmer who has to handle these. The paper proposes a tool- SPEX which automatically infer constraints of configuration and then detect & expose their vulnerabilities.

SPEX analyses the source code to infer configuration constraints- which for a parameter refers to data type, format, value range, dependencies, etc. The paper discusses various types of constraints in detail. These constraints are inferred by identifying the configuration variables in source code. These variables data flow is tracked to record any constraint along the data-flow path. SPEX provides toolkits to map configuration parameters into program variables- namely, Structure, Comparison, and Container interfaces in the form of key-value pairs. Each of these follow a different mapping convention with various levels of developer involvement. These parameter's basic data type, data range, control dependency, and value relationship is inferred using various inferences and comparisons along the data-flow path. To test and harden systems against Configuration errors, SPEX-INJ tool is used to generate errors which violates the inferred constraints by SPEX. Thereby reporting bad reactions back to the developer for adding checks and log messages. After injecting misconfigurations the system has to point to a location or parameter's name-value, otherwise into as a misconfiguration vulnerability. It uses software's own test infrastructure and records the logs. Then the report is used to analyze and take actions. This paper also discusses about how SPEX detects error prone system designs.

Tools discussed are evaluated both in commercial and open source softwares. Misconfiguration vulnerabilities, Error-prone design exposed by SPEC-INJ shows good numbers and examples shown gives a clear idea on what were detected. Real world misconfigurations have been used to compare the proposed method, with 24%-38% of them could have been avoided if SPEX had been used. A total of 3800 constraints have been inferred and 90% of them are accurate. The later part of the paper also discusses the experience of the authors when reported to developers and best practices to prevent configuration errors.

Strong Points:

1) The tool proposed is an "Automated" configuration error inference tool, which doesn't require any outside interference to *detect* the error.
2) Given 1/3rd of system failures are due to configuration errors, a research and proposing methods/tools to address them (which are less commonly looked into) is very much appreciated.
3) The best practices and the experiences discussed in the paper gives us a better understanding of configuration errors in real world, scope of it, and their impact.

Weak Points:

1) Overheads of inference haven't been discussed (except for 10 hrs of SPEX-INJ testing), which is one of the major drawbacks.
2) SPEX being able to detect only on single program, limits the scope of this paper. As mentioned in paper itself, SPEX doesn't infer all constraints in a single program.
3) Not exploring all code paths might leave out many parameters, which could pose serious issues.
4) SPEX looks for "concrete" code patterns. What does concrete refer to hasn't been discussed.
5) The authors could have proposed custom test infrastructures along with software's own test infrastructure.