# Paper Review: Do Not Blame Users for Misconfigurations

Sohail Ahmed Shaikh

February 18, 2019

## 1 Summary:

Through this paper the authors describe SPEX, a tool which can aid developers in mitigating configuration errors in the system, which are otherwise regarded as user issues. SPEX does this by source code analysis to infer configuration constraints. The tool is also evaluated through SPEX-INJ which injects configuration errors in the system to expose problems.

## 2 Description:

Misconfiguration errors are a significant cause of system failures and affect both end users as well as developers. The reason for this is that the symptoms of a system in case of failures may be similar despite of root causes. This prohibits users who due to their limited exposure of the application are unable to diagnose the problem as a configuration issue. These problems when reported to the developers may lead to wastage of time in debugging the issue.

The authors deviate from the measures taken in previous research in handling misconfigurations, which focus on detection and then taking measures. Instead they attempt to make the system more resilient to misconfigurations. The paper aims to enable developers to play an active role in handling misconfigurations by: Making the system less prone to system misconfigurations and detecting possible configuration errors.

The authors have implemented SPEX, which is a tool to achieve these goals. At the heart of this tool is the automatic inference of configuration constraints which is achieved by analysis of source code which has mapping information about which variables map to configuration parameters. This mapping is provided by developers by annotating the source code.

The configuration constraints could be attributes themselves or a relationship between attributes and are identified by two scans of the source code. The datatype constraints (essentially what must be the datatypes of configuration parameters) are detected by inferring the datatype of configuration parameters. The data type that is actually specified in the code may just be a placeholder, i.e. a data type which was used for initial assignment of the parameter (which might be a string in most cases) and later might be converted to or required to be represented in some other format, IP address for instance. Hence, SPEX performs a static code analysis to infer the semantic datatype of configuration parameters.

Further, SPEX can also detect the possible value ranges the parameters can take.

Several configuration parameters might have a dependency i.e. a configuration parameter might be dependent on the value of another configuration parameter. SPEX makes it possible to detect the correlations

between them.

The working of SPEX to infer configuration constraints is quite simple: First scan the source code to identify configuration variables, then track dataflow of each of these variables and third, record the constraints detected along the path. SPEX avoids symbolic execution because it leads to path explosion and instead it is based on LLVM compiler.

The authors also describe SPEX-INJ a misconfiguration injection testing tool to evaluate SPEX. It works by injecting configuration errors violating the constraints inferred by SPEX to test how the system reacts. If this leads to performance degradation, the developers are notified to better handle the misconfiguration issue so that it can be avoided.

Finally, the authors evaluate the performance of the system on several open source systems and Squid a web proxy tool.

# 3 Strong points:

1) It's a new approach towards handling configuration issues by early detection - almost a proactive approach that might be undertaken and mitigated by the developers.
2) Trivial annotations in the source code are required to infer constraints.
3) It makes the handling of configuration errors more user friendly.

# 4 Weak points:

1) Not all configuration errors can be detected.
2) It is intrusive at it requires the analysis of source code.
3) Requires annotation of source code so it's not completely automated.
4) Not language agnostic.
5) Although it has 90% accuracy, 10% of them are false positives. If the number of errors detected are large (as self confessed in the paper, it detected around 700 errors), it would mean that a large number of them are inaccurate and development time may get wasted in trying to debug and fix those errors.
6) Cannot detect cross software configuration problems.
7) Program specific constraints cannot be detected.
8) SPEX-INJ relies on the test cases shipped by the product. If test cases were poorly designed the resulting evaluations that the authors performed might not hold true.

# 5 Improvements:

1) Cross program configurations have not been inferred in this paper. These configurations may be inferred through additional annotations by the developers.
2) SPEX cannot identify program specific datatypes of the constraints which might be complex. To solve this, SPEX could have been designed to be more modular so that custom metrics for configuration parameters can be added by the developer.