Misconfiguration Injection in Systems for Reaction Evaluation

Michael Zronek, MSc[a]

a Institute for Information Systems Engineering, TU Wien, Vienna

Abstract Among the most costly mistakes in software engineering are those which come from system misconfiguration. Compared to software bugs, misconfiguration is more vulnerable to user mistakes.

Therefore it is desirable to make diagnosing easier for users by testing the system's reaction ability towards misconfiguration with error injection methods in advance. Few studies achieved this goal of testing system's reaction ability. The reason behind this is that mainly error types are not used which categorize kind of errors. Just a generic approach like simple alternations is used in other papers.

The extensively studied paper [3] of this course examined eight mature open source and commercial software and classified the option types. Using the fine-grained classification, syntactic and semantic constraints are extracted for each option type in order to generate misconfigurations. Misconfiguration injection is executed and afterwards the system's reaction is analyzed. The authors developed a tool called ConfTest which automatically executes such injections.

The analysis is carried out upon 4 open-source software systems: Httpd, Yum, PostgreSQL and MySQL. The evaluation shows that out of 1582 options, 96% were covered by the classification. Compared to the well-known pioneer ConfErr tool [2], ConfTest detects nearly 3 times more bad reactions.

Keywords misconfiguration, configuration, injection

The Art, Science, and Engineering of Programming

Perspective The Engineering of Programming

Area of Submission Configuration Management

1 Introduction

Software systems are getting more and more important in the century of automation. Most systems are highly configurable to fit in the respective environment or have a desired behavior. Unfortunately, misconfiguration, is one of the major causes for worsening software reliability and several studies have shown that. For example, Yin, Ma, Zheng, Zhou, Bairavasundaram, and Pasupathy[8] report that 27% of customer cases from a commercial storage system come from configuration errors. One method to harden systems to misconfiguration is to purposefully try to misconfigure a system and see its reaction. This method is called error injection and is done by several studies before such as ConfErr [2].

ConfTest improves the error-injection method by extracting more fine-grained option constraints compared to previous works. The main contributions of the paper can be structured into 3 parts:

(1) The classification of configuration options were extracted from 8 mature open-source and commercial software. 96% of 1582 options are covered from Httpd, Yum, PostgreSQL and MySQL which is more fine-grained compared to other constraints proposed.

(2) Misconfiguration injection and evaluation of the system reaction via the tool ConfTest is contributed. ConfTest can reveal design problems and bad reactions.

(3) 6 types of system reactions to misconfigurations are proposed. Misconfiguration of a path turned out to be the hardest misconfiguration to diagnose as it is difficult to syntactically and semantically check.

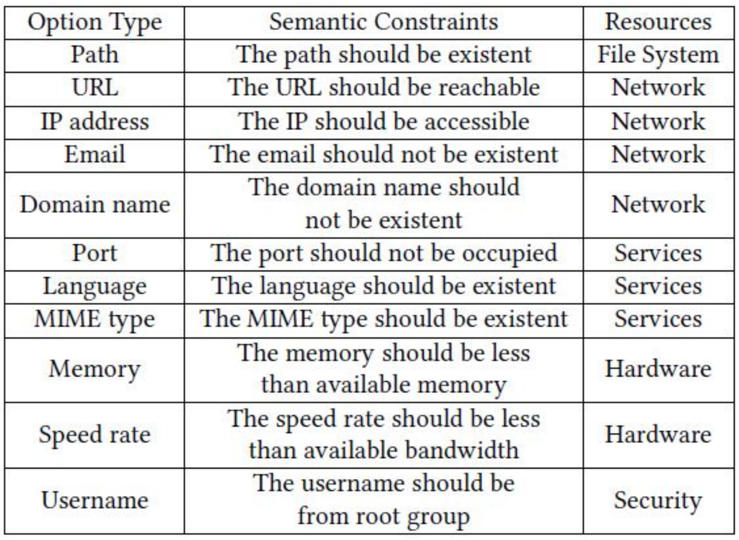
The paper is structured into constraint generation (Section 2), misconfiguration injection process (Section 3), evaluation (Section 4), related work (Section 5) and conclusion (Section 6).

2 Constraints Generation

Configuration options must satisfy certain configuration constraints in order to be valid, e.g. boolean options or file path options. [3] studied over 1500 configuration options from Squid, Nginx, Redis, Nagios, Lighttpd (core), Puppet, SeaFile, Vsftpd which are representative in their field. They manually investigated the official documentation as well as the configuration files to extract detailed information about each configuration option.

The reader is encouraged to look into the paper of [3, p. 3] for a detailed figure of the option type classification. In contrast to other type taxonomies which were studied, the fine-grained classification has its focus on generating option constraints for misconfiguration injection.

Applying the classification tree to the studied software systems Httpd, Yum, PostgreSQL and MySQL yields a coverage rate of 96,5% when excluding the types "Others". The type "Others" includes hybrid configuration options such as the option "LogFile" in MySQL which can either be a path or a network address. The classification can



Option Type	Semantic Constraints	Resources
Path	The path should be existent	File System
URL	The URL should be reachable	Network
IP address	The IP should be accessible	Network
Email	The email should not be existent	Network
Domain name	The domain name should not be existent	Network
Port	The port should not be occupied	Services
Language	The language should be existent	Services
MIME type	The MIME type should be existent	Services
Memory	The memory should be less than available memory	Hardware
Speed rate	The speed rate should be less than available bandwidth	Hardware
Username	The username should be from root group	Security

Figure 1 Semantic constraints

easily be supplemented with new types if it is difficult to classify a particular type of configuration option.

2.1 Type Constraints

To study the system reaction ability, for each configuration type, fine-grained constraints were developed. This is done by inherent constraints or from domain knowledge. Inferring constraints from program analysis, ie. by studying source code would be too difficult, if not impossible. Semantic as well as syntactic aspects are considered. As an example, type PORT should not use the same number as a used port and should be syntactically an integer between 0 and 65535.

Li, Li, Liao, Xu, Zhou, and Jia use syntactic constraints in the form of string patterns as standardization which was also done by EnCore [9]. As a short example take an Email Address option which should fulfill the pattern (\w)+(\.\w+)*@(\w)+((\.\w+)+) in order to be valid. The proposed syntactic constraints are consistent with over 90% of the options of Httpd, MySQL, PostgreSQL and Yum. For more detailed information about these string patterns, the user is encouraged to look into the paper of Li, Li, Liao, Xu, Zhou, and Jia[3, p. 5]. Semantic constraints are listed and explained in Figure 1. These constraints reflect complex relationships between software end environment. Please note that every Figure from this paper is taken from the original ConfTest [3] paper and will be omitted in the captures.

3 Misconfiguration Injection

After clarifying constraints for each option type, [3] show how to use these and how to test a systems' reaction with misconfiguration. The tool ConfTest is proposed to conduct misconfiguration injection.

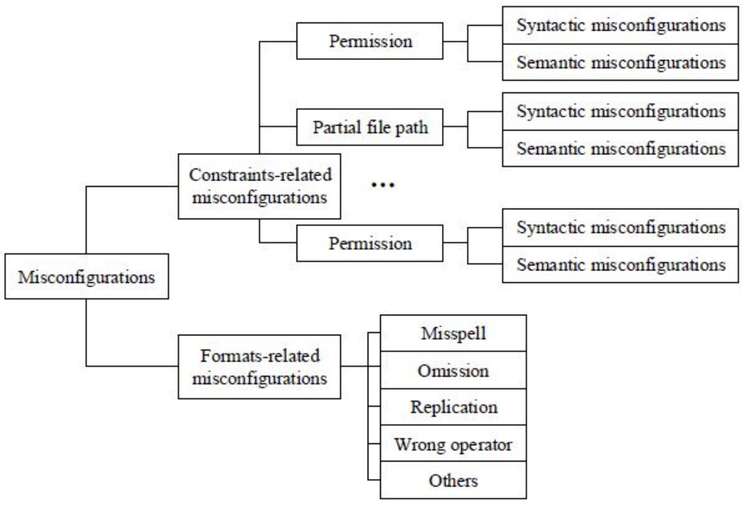

Misconfigurations
Constraints-related misconfigurations
Permission
Syntactic misconfigurations
Semantic misconfigurations
Partial file path
Syntactic misconfigurations
Semantic misconfigurations
...
Permission
Syntactic misconfigurations
Semantic misconfigurations
Formats-related misconfigurations
Misspell
Omission
Replication
Wrong operator
Others

Figure 2 Misconfiguration generation rules

ConfTest parses configuration files into structured data and then modifies the original data to generate misconfigurations. The newly modified configuration is then used for the target system. Figure 2 depicts the rules which were used to classify the injections: constraints-related and formats-related.

Constraint-related rules are used to violate constraints in the configuration. The 'Listen' configuration option in Httpd is identified as PORT with the typical syntactic constraints a port should have: (1) it should be an integer, (2) higher than 0 and (3) lower than 65535. Semantic misconfiguration is injected by actually using an occupied port.

Format-related rules are those which require a specific format and ConfTest injects typical user mistakes such as omission, misspelling, etc. when editing a complex configuration file. A similar approach is also done in ConfErr [2].

3.1 Testing

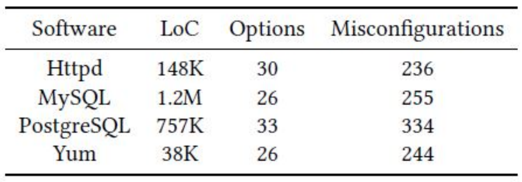
ConfTest uses test cases and test oracles which come from the software's own infrastructure (for example [5]) to mitigate the fact that the same misconfiguration can cause different systems reaction due to different states of program. ConfTest executes sequential steps to simulate administrators' behavior such as launching system, functional tests, etc. The faulty configuration replaces the old one and the target system is started up. If the system is running, functional test cases are executed until the system either fails or all test cases are executed. In the process all logs and outputs are gathered and analyzed by ConfTest to evaluate system reaction towards misconfiguration.

4 Evaluation

This section evaluates the reaction ability based on ConfTest's result. Figure 3 shows all generated 1069 misconfigurations. Only a sample of all possible misconfigurations is taken as otherwise it would lead to explosive exponential growth.

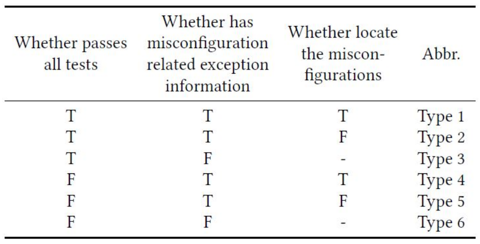
Another classification is proposed in order to structure the systems reaction of misconfiguration. Figure 4 shows all 6 error types.

For every misconfiguration, it is checked if it passes all system tests. After this phase, log messages from the test suites are manually checked for exception information related to the misconfiguration. If an exception can be found, it is checked if it can lead to the misconfiguration. As an example, in Type 1 all tests passed without any failure but an exception in the logs occurred which explicitly locates the injected misconfiguration. In Figure 5 the overall results can be seen.



Software	LoC	Options	Misconfigurations
Httpd	148K	30	236
MySQL	1.2M	26	255
PostgreSQL	757K	33	334
Yum	38K	26	244

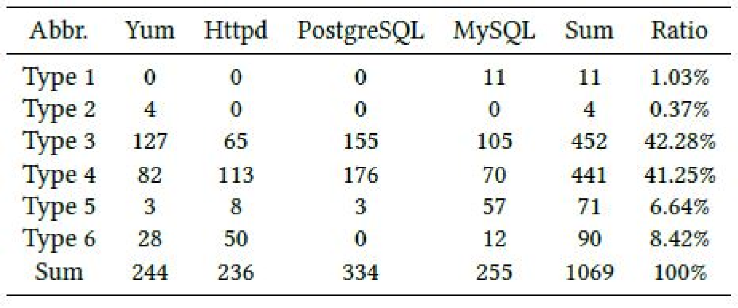
Figure 3 Systems in evaluation



Whether passes all tests	Whether has misconfiguration related exception information	Whether locate the miscon-figurations	Abbr.
T	T	T	Type 1
T	T	F	Type 2
T	F	-	Type 3
F	T	T	Type 4
F	T	F	Type 5
F	F	-	Type 6

Figure 4 System reactions classification

The root causes are analyzed per reaction type and yield interesting insights. For example, only Type 1 (1.03%) and Type 2 (0.37%) occurred in Yum and MySQL. In these applications invalid assignments are overruled which lead to Type 1 reactions. In MySQL for example, the configuration option "table_open_cache" would fallback to a normal number if misconfigured with an invalid number. Type 2 reactions throw exceptions, pass all tests but do not locate the concrete misconfigured location. This happens in Yum for example which prints out logs that a particular file or directory could not opened, but not which actual configuration setting caused the exception. Type 3 reactions are surprisingly common (42.28%) which has the root cause in two occasions. The "good" one comes from system design that is resilient to misconfiguration and legalizes it such as in PostgreSQL which allows both "key value" and "key = value" formats to counteract formatting errors. The "bad" case would be latent configuration errors which are not covered by the test cases. Type 4 reactions (41.25%) happen mostly during startup and can locate the exact line or name of the misconfiguration. Type 5 reactions (6.64%) failed to locate the misconfiguration which was triggered by an exception. This happens if an option is not checked or if the condition which checks the option does not capture the error. This case can be very difficult to debug in practice as reactions may obscure or even mislead users diagnosis. One example is Httpd: if you add "Listen 80" twice in the same configuration, the logs show "Address already in use" and "Could not bind to address" which will most likely mislead users. Type 6 reactions (8.42%) are caused by improper exception handling of configuration checking which results in reactions such as crashes, hangs or silent failures.



Abbr.	Yum	Httpd	PostgreSQL	MySQL	Sum	Ratio
Type 1	0	0	0	11	11	1.03%
Type 2	4	0	0	0	4	0.37%
Type 3	127	65	155	105	452	42.28%
Type 4	82	113	176	70	441	41.25%
Type 5	3	8	3	57	71	6.64%
Type 6	28	50	0	12	90	8.42%
Sum	244	236	334	255	1069	100%

Figure 5 Results of system reactions

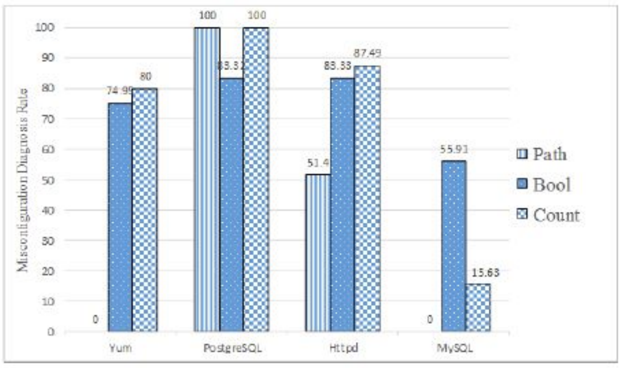


Figure 6 Misconfiguration diagnosis rate of different systems with 3 kinds of misconfiguration

Figure 6 illustrates the Type 4 misconfiguration diagnosis rate for the most widely used options boolean, path and count. MySQL and Yum completely failed to locate path related problems. MySQL overall reaches a worse diagnosis rate since exceptions mostly do not give the location of the misconfiguration. The diagnosis rate of count related errors are high with PostgreSQL even reaching 100%. Boolean misconfigured options also reach a high diagnosis rate with PostgreSQL again leading the board.

Path related misconfigurations are the hardest to diagnose, even for experienced developers. However, checks for simple constraints such as boolean, counts, modes, etc. are easy to implement and therefore have a high diagnosis rate. Li, Li, Liao, Xu, Zhou, and Jia thus highly recommend to use simple constraint options to reduce potential misconfiguration. Furthermore, the authors point out that more reasons are required in messages and the message should point to the root cause rather than only printing out a console log message on what is going on in the system.

5 Related work

A lot of papers deal with misconfiguration detection and troubleshooting or try to find a classification of options or configuration constraints. In a very recent paper of Liao, Zhou, Li, Jia, Liu, and He[4] for example, the authors find configuration constraints via a comprehensive manual study of five widely used open source software. The aim of their paper was to improve automatic configuration constraints extraction out of software.

Some papers have a different approach to finding misconfiguration such as ConfSuggester [10]. This tool developed by Zhang and Ernst uses dynamic profiling, execution trace comparison and static analysis to link an undesired behavior to its root cause in case of a misconfiguration. The paper complements ConfTest as ConfSuggester would reduce type 2 errors for example. Another Log-based Configuration Testing for misconfiguration diagnosing comes from MisconfDoctor[7]. It extracts log features for every misconfiguration and builds a database. When a misconfiguration then occurs, MisconfDoctor does a similarity calculation of the new log compared to the database and suggests a potential misconfiguration.

Another detection method comes from Uchiumi, Kikuchi, and Matsumoto[6] which use decision tree analysis for misconfiguration detection in cloud datacenters. This method identifies the relations among the majority of the parameters via a statistical decision tree analysis. This method is further enhanced by pattern modification and achieves a high accuracy (78,6%) in misconfiguration detection. Another approach to reduce configuration errors comes from ConfValley [1]. Its authors introduce a declarative language to express configuration specification. The language is then used for configuration validation and detected a number of configuration errors in the latest configuration deployed in Microsoft Azure.

Interestingly, the same authors of ConfTest, recently published a paper ConfVD which does exactly the same as ConfTest but is a reworked and polished up version with slightly different out comings. While MySQL still has the worst diagnosis rate, Httpd and Yum were drastically better (0% -> 50% Path diagnosis rate for Yum for example). Unfortunately this paper was not yet available once we wrote about ConfTest in this paper. If we would have seen ConfVD, we would obviously have taken the paper as it is much more recent and overworked.

6 Conclusion

In the paper of Li, Li, Liao, Xu, Zhou, and Jia we have seen how the system reaction ability of Httpd, Yum, PostgreSQL and MySQL. The authors proposed a fine-grained configuration option taxonomy which covered 96,5% of all configuration options and applied constrained-based injections into the four software systems. Those injections where either constraints-related or formats-related. The studied softwares were misconfigured and afterwards extensively tested with an appropriate test suite. 1069 misconfigurations were generated and the system reaction ability was classified in 6 types. The reactions of every system differ in the quality and user feedback but what is a common anchor, is that path related errors are the most difficult to diagnose and handle in the system itself. Avoiding complex configuration options and sticking to simple ones such as boolean, count and modes yields the best outcome concerning diagnosing and detecting as implementing checks for these option types are much easier.

References

[1] Peng Huang, William J. Bolosky, Abhishek Singh, and Yuanyuan Zhou. "ConfValley: A Systematic Configuration Validation Framework for Cloud Services". In: *Proceedings of the Tenth European Conference on Computer Systems*. EuroSys '15. Bordeaux, France: ACM, 2015, 19:1–19:16. ISBN: 978-1-4503-3238-5. DOI: 10.1145/2741948.2741963. URL: http://doi.acm.org/10.1145/2741948.2741963.

[2] L. Keller, P. Upadhyaya, and G. Candea. "ConfErr: A tool for assessing resilience to human configuration errors". In: *2008 IEEE International Conference on Dependable Systems and Networks With FTCS and DCC (DSN)*. June 2008, pages 157–166. DOI: 10.1109/DSN.2008.4630084.

[3] Wang Li, Shanshan Li, Xiangke Liao, Xiangyang Xu, Shulin Zhou, and Zhouyang Jia. "ConfTest: Generating Comprehensive Misconfiguration for System Reaction Ability Evaluation". In: *Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering*. EASE'17. Karlskrona, Sweden: ACM, 2017, pages 88–97. ISBN: 978-1-4503-4804-1. DOI: 10.1145/3084226.3084244. URL: http://doi.acm.org/10.1145/3084226.3084244.

[4] X. Liao, S. Zhou, S. Li, Z. Jia, X. Liu, and H. He. "Do You Really Know How to Configure Your Software? Configuration Constraints in Source Code May Help". In: *IEEE Transactions on Reliability* 67.3 (Sept. 2018), pages 832–846. ISSN: 0018-9529. DOI: 10.1109/TR.2018.2834419.

[5] *The MySQL Test Framework*. Oct. 2018. URL: https://dev.mysql.com/doc/dev/mysql-server/latest/PAGE_MYSQL_TEST_RUN.html (visited on 2018-11-17).

[6] T. Uchiumi, S. Kikuchi, and Y. Matsumoto. "Misconfiguration detection for cloud datacenters using decision tree analysis". In: *2012 14th Asia-Pacific Network Operations and Management Symposium (APNOMS)*. Sept. 2012, pages 1–4. DOI: 10.1109/APNOMS.2012.6356072.

[7] T. Wang, X. Liu, S. Li, X. Liao, W. Li, and Q. Liao. "MisconfDoctor: Diagnosing Misconfiguration via Log-Based Configuration Testing". In: *2018 IEEE International Conference on Software Quality, Reliability and Security (QRS)*. July 2018, pages 1–12. DOI: 10.1109/QRS.2018.00014.

[8] Zuoning Yin, Xiao Ma, Jing Zheng, Yuanyuan Zhou, Lakshmi N. Bairavasundaram, and Shankar Pasupathy. "An Empirical Study on Configuration Errors in Commercial and Open Source Systems". In: *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*. SOSP '11. Cascais, Portugal: ACM, 2011, pages 159–172. ISBN: 978-1-4503-0977-6. DOI: 10.1145/2043556.2043572. URL: http://doi.acm.org/10.1145/2043556.2043572.

[9] Jiaqi Zhang, Lakshminarayanan Renganarayana, Xiaolan Zhang, Niyu Ge, Vasanth Bala, Tianyin Xu, and Yuanyuan Zhou. "EnCore: Exploiting System Environment and Correlation Information for Misconfiguration Detection". In: *SIGPLAN Not.* 49.4 (Feb. 2014), pages 687–700. ISSN: 0362-1340. DOI: 10.1145/2644865.2541983. URL: http://doi.acm.org/10.1145/2644865.2541983.

[10] Sai Zhang and Michael D. Ernst. “Which Configuration Option Should I Change?” In: *Proceedings of the 36th International Conference on Software Engineering*. ICSE 2014. Hyderabad, India: ACM, 2014, pages 152–163. ISBN: 978-1-4503-2756-5. DOI: 10.1145/2568225.2568251. URL: http://doi.acm.org/10.1145/2568225.2568251.

About the author

Michael Zronek, MSc is the author of this LaTeX class. Contact him at e0951864@student.tuwien.ac.at.