FAKULTÄT FÜR INFORMATIK

DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Automated Detection, Localization and Removal of Information Exposure Errors

Kommanapalli Vasantha

FAKULTÄT FÜR INFORMATIK

DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Automated Detection, Localization and Removal of Information Exposure Errors

Automatisches Erkennen, Lokalisieren und Entfernen von ungewollter Informationsfreigabe

Author:	Kommanapalli Vasantha
Supervisor:	Prof. Dr. Claudia Eckert
Advisor:	M. Sc. Paul Muntean
Date:	November 15, 2015

Ich versichere, dass ich diese Diplomarbeit selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den 5. August 2015 Kommanapalli Vasantha

Acknowledgments

If someone contributed to the thesis... might be good to thank them here.

Abstract

Automatically detecting, localizing and fixing information errors is very much needed in current generation as people are not ready to spend much on their time in debugging and fixing the errors. In general debugging requires a lot of time and effort. Even if the bug's root cause is known, finding the bug and fixing it is a tedious task. The information that is exposed may be very valuable information like passwords or any information that is used for launching many deadly attacks. In order to fix the information exposure bug we should refactor the code in such a way that the attacks or information exposure can be restricted.

The main objective of this master thesis is to develop a quick fix generation tool for information exposure bugs. Based on the available information exposure checker which detects errors in the open source Juliet test cases: CWE-526, CWE-534 and CWE-535 a new quick fix tool for the removal of these bugs should be developed. The bug location and the bug fix location can be different (code lines) in a buggy program. Thus, there is a need for developing a bug quick fix localization algorithm based on software bug fix localization techniques. It should help to determine the code location where the quick fix should be inserted. Based on the bug location the developed algorithm should indicate where the quick fix should be inserted in the program. We can consider a bug fix to be a valid fix if it is able to remove the confidential parameter inside a function call to a system trust boundary. After the quick fix location was determined and the format of the quick fix was chosen then the quick fix will be inserted in the program with the help of the Eclipse CDT/LTK API.

The effectiveness of the implemented code refactoring is checked by re-running the information flow checker on the above mentioned open source Juliet test cases. If the checker detects no bug then the code patches are considered to be valid. The generated patches are syntactically correct, can be semi-automatically inserted into code and do not need additional human refinement. The generated patches should be correct and sound.

Contents

Outline of the Thesis

Part I: Introduction and Theory

CHAPTER 1: INTRODUCTION
This chapter presents an overview of the thesis and it purpose. Furthermore, it will discuss the sense of life in a very general approach.

CHAPTER 2: TECHNICAL AND SCIENTIFIC FUNDAMENTALS
No thesis without theory.

CHAPTER 3: RELATED WORK
write about related work.

Part II: Design and Development

CHAPTER 4: OVERVIEW
This chapter presents the requirements for the process.

Part III: Evaluation and Discussion

CHAPTER 5: EVALUATION
This chapter presents the requirements for the process.

CHAPTER 6: DISCUSSION
This chapter presents the requirements for the process.

CHAPTER 7: CONCLUSION
This chapter presents the requirements for the process.

Part I.

Introduction and Theory

1. Introduction

"The only truly secure system is one that is powered off, cast in a block of concrete and sealed in a lead-lined room with armed guards." — ***Gene Spafford***

According to Forbes 2015 [2], one among the topmost data breaches occurred in the previous years is Neiman Marcus hack. In January 2013, many debit and credit card information of almost 350,000 customers have been hacked. It is believed that the breach happened because of a malicious software that was installed onto the Neiman Marcus system. This software collected all the payment card information from customers who purchased. This proves that the software has helped the attackers to leak all the sensitive information through which they could get access to the system without leaving any trace of hacking. Sensitive information can be leaked in many ways [1]. It can be through the environmental variables which contain the sensitive information about any remote server, through a log file that was used while debugging the application where access to that file was not restricted or through a command shell error message which indicates that the web application code has some unhandled exception. In the last case, the attacker can take advantage of that error causing condition in order to gain access to the system without authorisation.

1.1. Information Exposure Bug

about info expo bug.

1.2. Motivation with Example

Write about motivating example.

1.3. Security Hazards

securitx hazards of info expo bugs

1.4. Basic Terminologies

Terminologies

1.5. Contribution

Contribution

2. Technical and Scientific Fundamentals

2.1. Information Exposure Bug with examples

about info expo bug.

2.2. Different Attacks

Write about motivating example.

2.3. Analysis Techniques

like data flow and control flow

2.4. Program Representation

Terminologies

2.5. Code Transformation

Contribution

2.6. Analysis Methods

like static, dynamic or combined

3. Related Work

Part II.

Design and Development

4. Localization Algorithm

5. Approach

5.1. About the Tool

Contribution

5.2. Bug Detection with SMT

Contribution

5.3. Semi Automated Patch insertion Wizard

about info expo bug.

6. Implementation

Part III.

Evaluation and Discussion

7. Evaluation

7.1. Test Setup

Setup configurations

7.2. Methodology

Methodology followed

7.3. Correctness validation

all the validations

7.4. Efficiency and Overhead

about the tool Efficiency.

7.5. Program Behaviour

about the program Behaviour

7.6. Usefulness of the generated Patches

Usefulness of the generated patches.

8. Discussion

8.1. Limitations

limitations of the tool and approach.

8.2. Applications

Where can this be applied.

8.3. Future Work

about the Future work.

9. Summary and Conclusion

Appendix

A. Detailed Descriptions

Here come the details that are not supposed to be in the regular text.

Bibliography

[1] CWE. *https://cwe.mitre.org/*.

[2] H. Bill. *http://www.forbes.com/sites/moneybuilder/2015/01/13/the-big-data-breaches-of-2014/*, jan 2015.