INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700 800/521-0600

Graduate School Form 9
(Revised 7/94)

PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis Acceptance

This is to certify that the thesis prepared

By Ivan Victor Krsul

Entitled Software Vulnerability Analysis

Complies with University regulations and meets the standards of the Graduate School for originality and quality

For the degree of Doctor of Philosophy

Signed by the final examining committee:

_____, chair

Approved by: _____ 5/1/98
Department Head Date

☐ is
This thesis ☒ is not to be regarded as confidential. _____
Major Professor

Format Approved by:

_____ or _____
Chair, Final Examining Committee Thesis Format Adviser

SOFTWARE VULNERABILITY ANALYSIS

A Thesis

Submitted to the Faculty

of

Purdue University

by

Ivan Victor Krsul

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

May 1998

UMI
300 North Zeeb Road
Ann Arbor, MI 48103

To Jacqueline M. Martinez sensei, for giving me the Dojo Kun, and to my family for giving me the life needed to understand it.

ACKNOWLEDGMENTS

There are many people that contributed significantly to my work and I would like to acknowledge their contributions.

Thanks to the members of my examination committee, Eugene Spafford, Aditya Mathur, Matthew Bishop, Carla Brodley, and Antony Hosking for their valuable suggestions and support. Carla Brodley's guidance in the fields of machine learning and data analysis, and her support with the tools and equipment needed for part of the analysis, is gratefully acknowledged.

I would like to thank Diego Zamboni, Tom Daniels, David Isacoff, Chapman Flack, and Eugene Spafford for their help with the reviews of this dissertation. Thanks to Diego Zamboni, Tom Daniels, Kevin Du, Mahesh Tripunitara, and Tugkan Tuglular for all their help and valuable suggestions during the vulnerability database weekly meetings. A significant portion of the ideas for this dissertation originated during our discussions there.

Tom Daniels and Adam Wilson helped enter data into the vulnerability database. I am grateful for their contributions. Thanks to Tugkan Tuglular for helping me develop most of the work relating to policies and the definitions of software vulnerability. Thanks for Edward Felten of Princeton University, and Michael Dilger for their contributions on software vulnerabilities. Thanks to Jeff Bradford for his help with the mineset classification tools.

Thanks to David Isacoff and Mahesh Tripunitara for their contributions on the nature of vulnerabilities and for clarifying many doubts with regards to the taxonomy of software vulnerabilities developed in section 6.1.

Portions of this work were supported by contract MDA904-97-6-0176 from the Maryland Procurement Office, and by the various sponsors of the COAST Laboratory — support that is gratefully acknowledged.

Diego Zamboni, Tugkan Tuglular, Tom Daniels and Mahesh Tripunitara deserve special recognition for tolerating my personality changes during the last few weeks of writing.

Last, but not least, my friends, Susana Soriano, Maria Khan, Elli Liassidou, Engin Uyan, Sidem Yavrucu, Pelin Aksit, Denis Lekic, Carlos Gonzales, Claudia Fajardo, Danielle Bolduc, Colin Brammer, Robert Ferguson, Carlos Ortiz, Charles Meyer, Juli Phillips, Pat Randolph, Lisa Anderson, and many others that I wish I could name, contributed significantly to my mental and spiritual well-being during the writing of this dissertation. Words are not sufficient to *thank* them.

TABLE OF CONTENTS

Page

Page

Page

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Figure Page

ABSTRACT

Krsul, Ivan Victor. Ph.D., Purdue University, May 1998. Software Vulnerability Analysis. Major Professor: Eugene H. Spafford.

The consequences of a class of system failures, commonly known as *software vulnerabilities*, violate security policies. They can cause the loss of information and reduce the value or usefulness of the system.

An increased understanding of the nature of vulnerabilities, their manifestations, and the mechanisms that can be used to eliminate and prevent them can be achieved by the development of a unified definition of software vulnerabilities, the development of a framework for the creation of taxonomies for vulnerabilities, and the application of learning, visualization, and statistical tools on a representative collection of software vulnerabilities.

This dissertation provides a unifying definition of software vulnerability based on the notion that it is security policies that define what is allowable or desirable in a system. It also includes a framework for the development of classifications and taxonomies for software vulnerabilities.

This dissertation presents a classification of software vulnerabilities that focuses on the assumptions that programmers make regarding the environment in which their application will be executed and that frequently do not hold during the execution of the program.

This dissertation concludes by showing that the unifying definition of software vulnerability, the framework for the development of classifications, and the application of learning and visualization tools can be used to improve security.

1 INTRODUCTION

1.1 Problem Statement

Software development can be complex. Added problem complexity, design complexity, or program complexity increases the difficulty that a programmer encounters in the design and coding of the software system [Conte et al. 1986]. Errors, faults, and failures are introduced in many stages of the software life-cycle [Beizer 1983; Myers 1979; DeMillo et al. 1987; Marick 1995].

The consequences of a class of system failures, commonly known as *software vulnerabilities*, violate security policies. They can cause the loss of information, and reduce the value or usefulness of the system [Leveson 1994; 1995; Amoroso 1994].

There is no single accepted definition of the term *software vulnerability*, and hence it is difficult to objectively measure the features of vulnerabilities, or make generalizations on this class of failures. Since the publication of [Linde 1975], software researchers have developed various programming guidebooks for the development of secure software and analyzed in detail various vulnerabilities [Weissman 1995; Garfinkel and Spafford 1996; Gavin 1998; Bishop 1986; Smith 1994; CERT Coordination Center 1998c; Spafford 1989; Kumar et al. 1995; Bishop 1995; Schuba et al. 1997; Carlstead et al. 1975; Bibsey et al. 1975; Abbott et al. 1976]. However, vulnerabilities that are the result of the problems listed in these programming guides continue to appear [CERT Coordination Center 1998a; 1998b; 1997a; 1997b; 1997c; Gavin 1998].

1.2 Thesis Statement

An increased understanding of the nature of vulnerabilities, their manifestations, and the mechanisms that can be used to eliminate them or prevent them can be achieved by the development of a unified definition of software vulnerabilities, the development of a

framework for the creation of taxonomies for software vulnerabilities, and the application of learning, visualization, and statistical tools on a representative collection of software vulnerabilities.

A unifying definition of software vulnerabilities can identify characteristics of vulnerabilities and allows researchers to agree on the object of study. A unifying definition can also be used to identify areas of focus for the development of taxonomies of software vulnerabilities.

An organizing framework can be used to generalize, abstract, and communicate findings within the research community. Taxonomies, or the theoretical study of classification, structure or organize the body of knowledge that constitutes a field. As such, they are an essential part of such a framework [Glass and Vessey 1995].

Researchers have attempted to develop such taxonomies and classifications for software vulnerabilities or related areas [Bishop 1995; Kumar and Spafford 1994; Kumar 1995; Kumar et al. 1995; Aslam 1995; Anderson 1994; Landwher et al. 1993; Cohen 1997a; 1997b]. However, as shown in Section 3.2, these classifications are ambiguous. The ambiguities are in part the result of conflictive definitions for software vulnerabilities, software faults, errors, etc.

A framework for the development of taxonomies according to generally accepted principles can be used to develop unambiguous classifications. These can result in an increased understanding of the nature of software vulnerabilities. An increased understanding of the nature of vulnerabilities can lead to improvements in the design and development of software.

The taxonomic characters developed for the classifications in taxonomies of software vulnerabilities can be used, in conjunction with the classifications themselves, to apply data mining and data visualization tools. These tools can reveal characteristics and properties of vulnerabilities that may not be apparent from the raw data.

1.3 Contributions

As shown in Section 2.1.3, the existing definitions of *software vulnerability* have one of the following forms: Access Control definitions, State Space definitions, and Fuzzy definitions. This dissertation provides a unifying definition based on the notion that it is security policies that define what is allowable or desirable in the system, and hence, the notion of software

vulnerability ultimately depends on our notion of policy. This dissertation also shows that existing classifications and taxonomies for software vulnerabilities, or related fields, do not satisfy all the desirable properties for classifications and taxonomies.

Section 3.1 defines the properties of measurements or observations necessary for the development of classifications; and provides a framework for the development of taxonomies for software vulnerabilities and related fields. This framework can be used as a basis for measuring features of vulnerabilities that can be used for the generation of classifications. These can be used to generalize, abstract, and communicate findings within the security research community, and contribute to our understanding of the nature of software vulnerabilities.

This dissertation presents an extension and revision of the classification of vulnerabilities presented by Aslam in [Aslam 1995]. Unlike its predecessor, this classification focuses on the assumptions that programmers make regarding the environment in which their application will execute, and that frequently do not hold in the execution of the program. Those vulnerabilities identified with this classification are not the result of software faults identified by common testing methods, because when tested in an environment that conforms to the assumptions made by programmers, the programs execute correctly.

This dissertation shows that machine learning and statistical analysis tools can reveal patterns and regularities that either reinforce our understanding of vulnerabilities, or provide new insights into the nature of vulnerabilities. Machine learning and statistical analysis tools can also influence the development of *a priori* classifications.

Finally, this dissertation describes how the development of taxonomies for software vulnerabilities can be used to build special domain-specific tools for the development of security-sensitive software.

1.4 Organization of the Dissertation

This dissertation is organized as follows: Chapter 2 introduces the terminology and the algorithmic conventions to be used throughout the dissertation. Chapter 3 presents the related work. Chapter 4 presents the development of new taxonomic characters for classifications and analysis. Chapter 5 shows how these taxonomic characters, measured for a collection of vulnerabilities, can be used with data mining and visualization tools.

Chapter 6 presents examples of *a priori* classifications and a taxonomy that focuses on vulnerabilities that result from mistaken environmental assumptions. Finally, Chapter 7 presents the conclusions, summarizes our findings, and discusses future directions.

2 NOTATION AND TERMINOLOGY

2.1 Terminology

In this section we introduce and define some of terms that will be used through the dissertation. Related terms are grouped by areas.

2.1.1 Error, Faults, and Failures

An **error** is a mistake made by a developer. It might be a typographical error, a misreading of a specifications, a misunderstanding of what a subroutine does, and so on [IEEE 1990]. An error might lead to one or more faults. Faults are located in the text of the program. More precisely, a **fault** is the difference between the incorrect program and the correct version [IEEE 1990].

The execution of faulty code may lead to zero or more failures, where a **failure** is the [non-empty] difference between the results of the incorrect and correct program [IEEE 1990].

2.1.2 Computer Policy

There are multiple definitions possible for computer policies, and the definitions presented in this dissertation have one of the following forms:

1. Policy helps to define what is considered valuable, and specifies what steps should be taken to safeguard those assets [Garfinkel and Spafford 1996].
2. Policy is defined as the set of laws, rules, practices, norms, and fashions that regulate how an organization manages, protects, and distributes sensitive information, and that regulates how an organization protects system services. [Longley and Shain 1990; DoDCSEC 1985; Sterne et al. 1991; Dijker 1996]