5-8-2004

Derivation of metrics for effective evaluation of vulnerability assessment technology

Darwin Edward Ammala

DERIVATION OF METRICS FOR EFFECTIVE EVALUATION OF VULNERABILITY ASSESSMENT TECHNOLOGY

By

Darwin Edward Ammala

A Thesis
Submitted to the Faculty of
Mississippi State University
in Partial Fulfillment of the Requirements
for the Degree of Master of Science
in Computer Science
in the Department of Computer Science and Engineering

Mississippi State University

May 2004

DERIVATION OF METRICS FOR EFFECTIVE EVALUATION OF

VULNERABILITY ASSESSMENT TECHNOLOGY

By

Darwin Edward Ammala

Name: Darwin E. Ammala

Date of Degree: May 8, 2004

Institution: Mississippi State University

Major Field: Computer Science

Major Professor: Dr. Rayford B. Vaughn

Title of Study: DERIVATION OF METRICS FOR EFFECTIVE EVALUATION OF VULNERABILITY ASSESSMENT TECHNOLOGY

Pages in Study: 156

Candidate for Degree of Master of Science

Vulnerability in software receives constant attention in the media and in research. Yearly rates of disclosure of vulnerabilities in software have doubled. The discipline of Information Assurance lacks metrics that are useful in understanding vulnerability. In the problem of vulnerability assessment tool selection, users must make product choices based on results found in non-peer reviewed publications or subjective opinion. Users of vulnerability assessment tools must sift through volumes of data about their systems and are shown broad indications of the severity of the problems – often a high-medium-low ranking, which varies between tools. A need exists for metrics and a selection model for tool quality assessment. This study addresses these needs by analysis of the discipline of vulnerability assessment and remediation from first principles, and presents an organized approach and a best-fit metrics based model for selecting vulnerability assessment tools.

DEDICATION

This work is dedicated in love for my wife Patricia, loving memory of my father, Edward, and in honor of my mother, Mardell for giving me life, and to the glory of the Lord of all Creation for granting me Eternal life.

ACKNOWLEDGMENTS

I wish to thank my thesis advisor, Dr. Rayford Vaughn, for his diligence and commitment to Information Assurance education, and for providing advice and guidance to me as I progressed through my studies at Mississippi State. Thanks are also due to Mr. William Wall and Mr. John Farrell, Security Engineers at Harris Corporation and Ms. Margaret Knepper, Systems Engineer at Harris Corporation, for their help in the discipline of vulnerability assessment. Thanks are also due to the Harris Corp., STAT® project for allowing me the use of their testing laboratory and tools, which provided resources for the conduct of this study. Finally, thanks to Dr. Kevin Fox of Harris Corporation for his review and comment on drafts of this work.

TABLE OF CONTENTS

	Page
DEDICATION	ii
ACKNOWLEDGEMENTS	iii
LIST OF TABLES	vi
LIST OF FIGURES	viii
NOMENCLATURE	ix
CHAPTER	
I. INTRODUCTION	1
II. RELATED WORK	7
Vulnerability Taxonomies	8
Linde's Flaw Hypothesis Methodology (1975)	8
RISOS Project (1976)	9
Protection Analysis (1978)	11
IBM Orthogonal Defect Classification (1992)	12
Landwehr et al. (1993)	13
Aslam (1995)	14
Bishop (1995/6)	15
Du/Mathur (1997)	17
Krsul (1998)	18
Taxonomies of Faults	20
Taxonomy of Security Policy	23
Discussion of Metrics	24
The State of IA Metrics	25
The Properties of Useful Metrics	32
Applied Metrics and Measures	36
The Information Assurance Metrics Problem	42
Discussion of Vulnerability Assessment	45
Remediation Taxonomy	55
III. Methodology	57
IA Metrics Taxonomy	59

Collect Candidate Measures 62
IA Metrics Best-fit Model 74
Map Model to IA Metrics Taxonomy 76
Test Environment 78
VA Scanning Tools as Articles of Measure 78
Measurement Protocol 80

IV. Validation 81

Selection of VA Scanning Tools 82
Target System Selection and Setup 84
Measurement 85
Analysis 86
Depth Analysis 86
Breadth Analysis 98
Model Validation 104
Consistencies 115
Inconsistencies 116

V. Conclusion 118

Contributions of this Research 118
Future Research 120

REFERENCES 123

APPENDIX

A. Test Results 135

LIST OF TABLES

Table	Page
2.1 Flaw Hypothesis Methodology Taxonomy	9
2.2 RISOS Taxonomy	10
2.3 Protection Analysis Taxonomy	12
2.4 Landwehr Software Failure Taxonomy	13
2.5 Aslam Taxonomy	15
2.6 Bishop Taxonomy	16
2.7 Du/Mathur Taxonomy	18
2.8 Krsul Taxonomy	19
2.9 Rubey’s Software Fault Classification	21
2.10 Beizer Software Fault Taxonomy	21
2.11 Smith/Newton Security Policy Taxonomy	23
2.12 Schneidewind’s Metrics Validation	37
2.13 Comparison of System Compromise Models	45
3.1 IA Metrics Taxonomy within Organizations	59
3.2 IA Metrics Taxonomy Properties of Tools	60
3.3 Policy Management Measures	62
3.4 Support Measures	64
3.5 Operational Practices Measures	66
3.6 Operational Environment Measures	67
3.7 Readiness Measures	67
3.8 Tool Effectiveness Measures	69
3.9 Technical Target of Assessment Features in Normal Circumstances Measures	71
3.10 IA Metrics Best-fit VAST Model	77

Table Page

4.1 Vulnerabilities Found and the Portion of Highest Risk.. 87

4.2 False-positive Results on Patched Systems 89

4.3 Breadth Coverage of Tools... 99

4.4 IA Best-fit Metrics.. 106

4.5 Comparison of Best-fit Model with At-large Measures 114

A.1 Policy Management 137

A.2 Process Maturity 138

A.3 Personnel Support.. 138

A.4.Resource Support... 139

A.5 Operational Practices 140

A.6 Operational Environment 141

A.7 Management Readiness.. 142

A.8 Technical Readiness 143

A.9 Effectiveness 144

A.10 Effectiveness Case Study 145

A.11 TTOA Features in Normal Circumstances Part I 147

A.12 TTOA Features in Normal Circumstances Part II 148

A.13 TTOA Scan Performance 151

A.14 TTOA Adversary Work Factor 152

A.15 TTOA Survivability... 152

A.16 TTOA Risks... 153

A.17 TTOA Operational Limitations... 154

A.18 Summary 155

LIST OF FIGURES

Figure Page

1 IA Metrics Taxonomy 43

NOMENCLATURE

AIS - Automated Information System; any equipment of an interconnected system or subsystems of equipment that is used in the automatic acquisition, storage, manipulation, control, display, transmission, or reception of data and includes software, firmware, and hardware [44].

At-large set of measures - The union of all measures extracted from prior work, and derived as part of the IA metrics taxonomy [59].

Benchmark - a standard measure or point of reference for judging quality.

Information Assurance - Information operations that protect and defend information and information systems by ensuring their availability, integrity, authentication, confidentiality, and non-repudiation. This includes providing for restoration of information systems by incorporating protection, detection, and reaction capabilities [44].

Measure – a definite unit of capacity or extent in terms of which the size or capacity of things are ascertained e.g., distance.

Metric – a composite of measures established to assess the comparative quality or extent of similar things, e.g., distance traveled per unit time; speed.

Taxonomy - the study of the general principles of scientific classification.

VAST – vulnerability assessment scanning tool, designed to perform diagnostic tests on operating systems to identify areas of known vulnerability.

Vulnerability Assessment - systematic examination of an automated information system (AIS) or product to determine the adequacy of security measures, identify security deficiencies, provide data from which to predict the effectiveness of proposed security measures, and confirm the adequacy of such measures after implementation [44].

C H A P T E R I

INTRODUCTION

With the increased implementation of computing equipment and the interconnectedness of this equipment by communications networks, worldwide commerce is increasingly conducted in cyberspace. Threats to the computing infrastructure therefore pose threats to commerce. Reduction of these threats and vulnerabilities in computing is one of the primary objectives of Information Assurance (IA) and is the area of focus in this study. The definition of IA within this study is adopted from the NSA Glossary of Terms and is provided in the Nomenclature section along with other terms germane to this study. Protection includes detection, remediation, and prevention of incidents that pose threats to the data and its native processing environment. Such environments differ among each owning organization, and with this, an implication of divergent priorities emerges.

Users and implementers of IA technology have a wide range of products from which to choose their solution. The choice of solution is traditionally based on cost, appropriateness of solution offered, and the information technology (IT) staff familiarity with a specific solution. Frequently an incorrect or ineffective choice is made prior to finding the most appropriate solution. Correct decisions are possible when the acquirer has adequate means to compare and assess contending alternatives. Many disciplines have developed definitive benchmarks and metrics upon which comparisons are based. Examples from IT include the Transaction Processing Performance Council’s database query performance benchmarks [53], and the Text Retrieval Conferences (TREC) sponsored by the United States National Institute for Standards and Technology (NIST), and the Defense Advanced Projects Research Agency (DARPA) [57]. The TREC provides a reference set of documents by which organizations may evaluate text retrieval technology. A much more common example of a metric is miles per gallon; cited in automobile advertising as a uniform means of comparison for consumers. Thus, a consumer with fuel costs as a high priority has a strong predictor in his or her choice of an appropriate car. A purchaser relying only on a single benchmark is susceptible to making a less than optimal decision. A family of eight with a fuel economy priority would be disappointed in relying solely on fuel efficiency and ignoring seating capacity in purchasing a car. Yet, many purchasers of IT security products use this analog in deciding on available comparison metrics. In making such choices, deep differences between contending products can be overlooked and only detected well after the product has been deployed and expectations are unmet.

Competition for market share drives the demand for continual improvement by contending vendors. A critical area for commercial firms is that of customer retention; this motivates the development of higher quality products, with fewer defects, more features, and better support. Product quality follows from process quality. Process quality assessment is a reliable means to mark process improvement. Quality assessment practices such as the Software Engineering Institute's Capability Maturity Model (CMM), and international quality programs such as ISO-9000 provide valuable frameworks and standards, which serve to rate the proficiency of each organization. The improved quality of process yields improved product or service quality. These assessments are becoming mandatory for long-term marketplace success. For example, governments such as that in the United States have declared that large IT and defense contracts will be awarded only to organizations attaining or exceeding CMM level three. One could infer that process maturity ratings play a role in the overall quality of products.

The field of IA is new by comparison to other engineering or industrial disciplines such as civil engineering or shipbuilding. Yet increases in criticality of information systems and the quick dissemination of tools for exploitation of vulnerabilities in systems, make it increasingly important to develop quality IA assessment and repair tools. Quality in anything is more readily realized when a notion of measuring it exists, and regular the use of measurements takes place. Quality focused organizations install processes and metrics to assess them; as this is expected of CMM level three organizations. This same degree of discipline is needed in the IA tool development and assessment community. Analysis of the Carnegie-Mellon Computer Emergency Response Team (CERT) vulnerability incident data shows that attacks become widespread within three months of publication of a scripted exploit [4]. Yet the vendor identifying a patch consumes the first month of this time period, and vulnerability tool vendors developing checks, and releasing their next updates require time beyond the release of the patch. The Blaster worm released in the summer of 2003 showed that vulnerability assessment (VA) tool vendors have improved the time from detection to dissemination as is presented in the analysis phase of this study available in the Appendix. Hence, the user of vulnerability tools has a significantly reduced time window to detect and repair vulnerabilities. Attempts to decrease the time to product update are occurring, and over time, response should improve.

Increasing the energy and resources applied to the quality of IA solutions is critical. Comparisons of vulnerability assessment tools are driven by limited factors such as ability to detect or report on a small set of known vulnerabilities [3, 22, 38]. This single dimension is given heavy reliance by readers of comparison reports since most users and even professionals do not have the time, resources, expertise, or a frame of reference to study all dimensions rigorously. Development of representative and accurate vulnerability assessment tool performance metrics, with repeatable and reliable testing and measurement processes will greatly assist the entire IA and IT communities.

The problem of useful measurement and assessment of IA solutions deserves careful study. The scope of this study is the application of the knowledge and process of metric development to the IA discipline of vulnerability assessment. The specific IA sub discipline of vulnerability detection and removal is studied and discussed. This dimension was selected as a starting point because of the number of tools available, their familiarity within the IT and IA communities, the frequent comparisons found in IT trade magazines, and the potential of the results to provide insight to a large number of IA stakeholders enabling them to make choices that are more effective.

Another problem facing IT security professionals regarding use of vulnerability assessment tools is in prioritizing the solutions. The tools generally identify vulnerability severity via a qualitative high, medium, low, scale. These terms are ambiguous (or fuzzy), and have differing interpretations across vendors. In such cases where the state of an exploit of a vulnerability can change from unknown to published overnight, one may never do better than an ordinal severity scale, however finer grains of distinction may prove valuable. This comes to the fore, as one understands that vulnerabilities may exist at different points in their respective life cycles. Given two vulnerabilities, within the same classification of severity, and both having known patches or other remediation strategies, a vulnerability having a published exploitation script is of greater urgency to repair than one without a known, published exploit.

Results of this study show that IA solutions can be meaningfully assessable and measurable for each organization. Another product will be development of metrics to assess the quality, precision, speed, ownership cost, and repair cost for a given organization.

Chapter II addresses prior work in areas related to this study effort. The development of vulnerability classification systems is presented in search of a meaningful approach to classify vulnerabilities in a way that is useful for IT staffs. Other work on software defect classification and security policy classification is examined. Since the goal of this work is to produce metrics or measures for vulnerability assessment tool evaluation, we will also examine the topic of IA metrics and the properties of useful metrics in general. The topic of vulnerability remediation is examined to highlight areas of this crucial task that should be considered in vulnerability tool comparisons. Thus, this chapter illustrates that a framework upon which an IT organization can assess vulnerabilities as they relate to their own environments is needed. Currently intuitive sense and experience dictates the priority of treatment of vulnerability in systems. Chapter III presents the methodology used to develop a tool evaluation best-fit model, and the measurement protocol used in carrying out assessment of the tools. Chapter IV describes application of the methodology and validation of the methodology and model; providing comparisons between the model and a super set of measures across the categories of the IA metrics taxonomy. Chapter V presents the conclusions of the research and results of use of the model. An Appendix following the work provides explanations of all measures taken during the study and provides relevance to the IA metrics categories to which the measures are assigned.

CHAPTER II

RELATED WORK

The focus of this study is on objective comparisons of vulnerability assessment (VA) tool options by procuring organizations. This requires knowledge of vulnerabilities, measurements and metrics, vulnerability assessment, and security policies. This chapter explores previous work for contributions to this effort, whether directly or indirectly. Previous research exists in classification of system vulnerabilities, (as well as their life cycle and remediation), the classification of security policies, the proper selection of and construction of metrics, the overall discipline of information assurance, and the IA discipline of vulnerability identification and remediation. We begin by reviewing vulnerability classification work.

Classification of flaws in software, and their active manifestations as faults or vulnerability-faults with security consequences has been approached from many perspectives. Before discussing the related work in vulnerability classification, we mention the general properties that a taxonomy should possess. A taxonomy should have both an explanatory and a predictive value [30]. The taxonomy should be explanatory in the sense of sorting and organizing individual classes, and predictive in the sense that types not yet encountered may be accounted for, making prediction of their occurrence and their recognition easier. This investigation illustrates properties of vulnerabilities that vulnerability assessment tools may measure or present within their results.

2.1 Vulnerability Taxonomies

We now address the classification of vulnerability in computing systems through the past thirty-seven years.

In 1967, the Defense Science Board Task Force investigation [62], prompted the United States Federal Government to launch studies into the state of security protection within its computing centers. The results were unsettling to many and fueled the research thrust into security of computing systems. Penetration testing was studied intensively. Linde provided an early insight into classification of faults with the Flaw Hypothesis Methodology [32]. Two studies in the 1970’s centered on the then common computing environment consisting of mainframe systems. The paragraphs that follow present overviews of the contributing work in vulnerability classification.

2.1.1 Linde’s Flaw Hypothesis Methodology (1975)

Linde developed a systems penetration strategy that is useful for identifying weaknesses in the functional areas of operating system design. The hypothesis decomposes into four steps [32] as seen in Table 2.1. This work stimulated many later studies, and established the classic approach used by system penetration red teams to this day. The steps of obtaining knowledge, formulating a hypothesis of vulnerability within the system, testing the hypothesis, and finally generalizing about the flaw-discovered form the core of well-known strategies used in vulnerability assessment.

Table 2.1 Flaw Hypothesis Methodology Taxonomy

Step	Element
Obtain knowledge of system's control structures	inter-module knowledge, access control mechanisms, control object hierarchy, intra-module knowledge, and specific implementation
Formulate a flaw hypothesis	hypothesis of vulnerability through information from source code, design documents
Confirm the flaw hypothesis	writing software exploits against the hypothetical vulnerability
Construct a flaw generalization	generalize about vulnerability through knowledge of similar systems, and from study of other parts of the system under study).

2.1.2 RISOS Project (1976)

Abbott and his research group at the United States Department of Energy Lawrence Livermore Laboratory's Institute for Computer Science and Technology undertook the Research into Security of Operating Systems (RISOS) effort to understand system faults. This laboratory conducts research into nuclear power and weaponry, thus systems it maintains are critical. The project had the goal of understanding security issues in operating systems and exploration of approaches for addressing them by understanding faults and analyzing new faults as they were discovered. Seven broad categories resulted, that attempt to generalize software faults across several operating systems [1]. The RISOS classification is shown in Table 2.2.

Table 2.2 RISOS Taxonomy

Software fault	Nature of problem
Incomplete parameter validation	value validation of arguments of procedure calls
Inconsistent parameter validation	multiple sets of validation criteria exist in a system, and a 'wrong' set is used
Implicit sharing of confidential data	information from highly privileged processes is disclosed to lesser-privileged processes
Asynchronous validation/Inadequate serialization	serialization of data storage and access is not enforced
Inadequate identification, authentication and authorization	no uniquely distinguishing login session contexts, (this is the MS-DOS system model)
Violable prohibition and limits	enforced on accesses to data structures maintained by the system
Exploitable logic errors	incorrect error handling sequences, side effects of untested instructions and sporadic timing features

Note that incomplete parameter validation was discovered in 1976, and remains a frequent target of exploitation. Parameter validation is critical in requests by low-privileged user processes for services from highly privileged system processes. High-privileged processes must enforce input validation to ensure that correct input is received and correct actions will ensue from within the high privileged process. System kernels are the highest of privileged processes. Inconsistent parameter validation renders a function to execute with incorrectly validated parameters. Implicit sharing of confidential data is a consequence of improper input validation, and has been countered by the developments of mandatory access controls, such as labeling, and access decisions dictated by security policy. Asynchronous validation/inadequate serialization results in race conditions, deadlocks, and instantiation related coherency problems. Inadequate identification, authentication, authorization - without authorizations of a user or process, imply that the system has no means to enforce control of the integrity of its own components, or to protect the data of other users. Without identification and authentication, a system is incapable of distinguishing either between separate users, or in tracing actions and upholding of accountability policy. Lack of authorization control induces anarchy in system resource management. Violable prohibition and limits to system data structures is vital; if these structures and containers are overflowed, the system security state will transition from known stability and security maintenance to unknown stability and security postures. Exploitable logic errors lead to compromised security of a system [1].

2.1.3 Protection Analysis (1978)

Improvements in operating systems security were a concern in the 1970's. This study aimed to provide insight to operating systems developers to improve security mechanisms. The study group focused on various protection errors, and methods of identifying them. The classification was derived from a formulation of pattern matching techniques to examine source code for security faults. Four broad categories of syntactic structure were identified each having several types of security faults [6, 8]. The Protection Analysis classification is shown in Table 2.3. The protection domain and validation error classes are well understood and the vulnerabilities found by researchers and exploited by tools are heavily oriented toward these classes. The synchronization and operator/operand error classes are somewhat less common today mainly due to the difficulty in detecting them via external analysis and attack mechanisms.

Table 2.3 Protection Analysis Taxonomy

Error Class	Elements
Protection domain errors	initialization and enforcement issues addressing initial assignments, protection mechanism bypass, parameter change management, naming ambiguities, incomplete destruction of data, content and contexts
Validation errors	considering boundary and operand tests, allowing pointer boundary violations, and buffer overflows
Synchronization errors	improper protection of atomic operations, and lack of blocking or barriers on sequences of operations
Operator/operand errors	unfair process scheduling, use of incorrect operators and operands

Protection domain issues remain problematic in common operating systems to this day, although the “Orange Book” and Common Criteria address this as a requirement to gain evaluated status. Validation errors remain commonplace in software as input validation and buffer overflows dominate vulnerability categories in BugTraq and CVE descriptions. Synchronization errors are addressed in parallel computing middleware libraries, as well as system call libraries in modern programming languages. Operator/operand errors can still occur if inadequate verification and testing is performed on software.

2.1.4 IBM Orthogonal Defect Classification (1992)

Chillerege describes the intent of the orthogonal defect classification approach “*The goal is to provide an in-process measurement paradigm to extracting key information from* *defects and enable the metering of cause-effect relationships. Orthogonal Defect Classification (ODC) essentially means that we categorize a defect into classes that collectively point to the part of the process which needs attention, much like characterizing a point in a Cartesian system of orthogonal axes by its (x, y, z) coordinates*". This may be interpreted to mean that the categories chosen must provide coverage of the domain. Defects are grouped relative to the part of the process in which they are uncovered. The classifications take into consideration: cause, evoked by a classification set of triggers, and an effect class measured by severity. The effect classification used is IBM's CUPRMID—Capability, Usability, Performance, Reliability, Installability, Maintainability and Documentation [13].

2.1.5 Landwehr et al. (1993)

Landwehr and his team of researchers observed that the history of software failures was largely unpublished, yet system security was a rising concern. They worked toward the goal of helping system designers and implementers produce more stable and secure systems. The taxonomy was derived from analysis of the software development life cycle, and consists of three broad categories [31]. Landwehr's classification is shown in Table 2.4.

Table 2.4 Landwehr Software Failure Taxonomy

Axis	Element types
Genesis	malicious, non malicious
Location	considers the type of software involved
Time of introduction	pertains to product life cycle phases

Within the genesis dimension, a malicious cause in the form of worms, viruses, trap doors, time bombs, Trojan horses is possible. Non-malicious sources of vulnerability include implementation error, a lack of requirements comprehension or collection, or an implementer's or maintainer's misunderstanding of the design logic. Most non-malicious flaws fit into domain errors, validation errors, or serialization/aliasing errors, errors from identification/authentication problems, boundary condition errors, and logic errors. The location in the system software dimension includes system routines, system support utilities, or user programs. The time-of-introduction dimension includes requirements, specification, design, implementation, incomplete testing, or maintenance phases of the life cycle [31].

2.1.6 Aslam (1995)

Aslam's intent was to classify faults unambiguously in software on UNIX operating systems into non-overlapping categories, with the express purpose of populating a vulnerability database and to identify fault detection techniques. This would lead to systematic testing strategies that would improve success over that of the penetrate-and-patch paradigm. Vulnerabilities that led to system compromise were of primary interest. Sources used were the Computer Emergency Response Team (CERT) Security advisories, various computer security mailing lists such as BugTraq at SecurityFocus [47], Security Tracker [48], and literature surveys. Aslam's classification is shown in Table 2.5.

Table 2.5 Aslam Taxonomy

Fault Class	Element types
Coding	synchronization, condition, parameter handling,
Operational	installation
Environmental	resource constraints, faulty construction, component interaction, or event handling

The personal, communications, physical, and operations security related faults were acknowledged, and omitted to bring focus to software faults. The categories in more detail include operational faults concerning installation in the wrong place, installation using incorrect parameters, and installation with wrong permissions. The coding faults include synchronization issues of race conditions, event serialization, condition handing issues of missing conditions, unspecified conditions, and missing predicates. Other coding issues include limits being checked, access rights being checked, valid input values, correct syntax, parameter matching, missing delimiters, extraneous input fields, subject origin authentication, and checking of exception conditions. The environment faults include limitations of environment resources and capabilities, faulty compilation or builds of software products, interaction errors between functionally correct modules, and exception handlers that do not perform as expected [6].

2.1.7 Bishop (1995/6)

Bishop's intention was to develop a taxonomy through an examination of earlier work. In addition, through analysis of the results of the Protection Analysis work, illustrated how to improve the security of systems. A second goal was to show how to write programs with minimal security related flaws. The taxonomy used the UNIX operating system and supporting software and applications as the basis of study. Direct benefits of the work included: a description of vulnerabilities in a form readily useful to intrusion detection systems; an explanation of methods to identify vulnerabilities; and an explanation of an approach to prevent exploitation of vulnerabilities in the system. The taxonomy has six dimensions, each vulnerability being classified along all of them. The resulting classification from Bishop is shown in Table 2.6.

Table 2.6 Bishop Taxonomy

Dimension	Element types
Nature	class of error
Time of introduction	product life cycle phase
Exploitation domain	program, file, language
Effect domain	couplets {(user/system),(nothing, session, hardware)}
Cooperating components	number of separate parts necessary
Source of identification	source code, news lists, papers, or security advisories

The nature of the flaw dimension includes domain errors, validation errors, naming errors, and serialization errors. The time of introduction can occur during development, maintenance, or operation of the software. The exploitation domain considers the software program itself, related configuration files, or high-level command languages on the system such as interactive user shells or web content rendering languages. The effect domain considers the stakeholder and context relative to the system and is derived from couplets wherein one element is either {the system, or user}, while another is taken from {nothing, the session, applicable hardware, or a combination of session and hardware}. The minimum-components-to-enable addresses whether the vulnerability can be exploited by the single component or whether a group or chain of vulnerabilities must be exploited in order to produce the vulnerable condition. The source of identification serves to trace the understanding of the vulnerability to a known beginning, including software settings, news lists, articles, papers, and security advisories [9, 10].

2.1.8 Du/Mathur (1997)

Du and Mathur described their research of the study of vulnerabilities registered into the database built and maintained by the Purdue University COAST/CERIAS Laboratory. A taxonomy was constructed, based on the cause of a flaw, the impact of the flaw, and the repairs needed to render the system invulnerable to the exploitation of the vulnerability resulting from the flaw. Motivation for the taxonomy included:

1. Evaluation of code based coverage criteria (control flow, data flow, and mutation based criteria), in assessing the effectiveness of testing for vulnerabilities,
2. Motivation to develop a tool to assist developers and testers in assessing the effect of flaws in distributed software, e.g., Common Object Request Broker Architecture (CORBA) [40], interfaces to distributed objects written in Java.

The Du/Mathur scheme allows a vulnerability to have membership in several categories. They state that restriction of a vulnerability to one category induces information loss regarding the flaw [20]. The cause, impact, fix coupling is used [20, 40]. Du's and Mathur's resulting classification is shown in Table 2.7. The cause dimension was adapted from the work of Landwehr et al. including validation errors, authentication flaws, incorrect serialization, and incomplete boundary checking errors, domain application errors, incorrect designs, and other exploitable logic [31].

Table 2.7 Du/Mathur Taxonomy

Dimension	Elements
Cause	similar to Landwehr's genesis
Impact	consequence of successful exploit – unauthorized access, execution, modification, denial of service
Fix	spurious, missing, misplaced, or incorrect

The direct impact of the flaw dimension encompasses unauthorized execution of code, unauthorized modification of resources, unauthorized access of resource, and denial of service. The fix or remediation dimension utilizes DeMilo and Mathur's classification scheme since it is amenable to automation. This aspect examines source code for the presence of spurious entities and their removal, as would be the case for characteristic sub strings incorrectly specified. Additionally missing entities, misplaced entities and restoring to their proper place, and incorrect entities are used as a catchall case [40].

2.1.9 Krsul (1998)

This taxonomy was developed to build a vulnerability database. The taxonomy focuses on the assumptions made by programmers about the environment in which their software executes; often these assumptions fail to hold true. Krsul sought a deeper understanding of the nature of vulnerabilities, and presented his taxonomy as part of his Ph.D dissertation at Purdue University [30]. Three dimensions were identified. The threat dimension is concerned with direct impacts of vulnerabilities. Indirect impacts follow from these as advanced vulnerability state transitions. An environment dimension and a nature of vulnerability dimension were also included. Krsul's classification is seen in Table 2.8.

Table 2.8 Krsul Taxonomy

Dimension	Elements
Threat	unauthorized observation, creation, modification, destruction of data or objects
Environmental	assumptions about the properties of data and objects within the system
Nature of vulnerability	impact on relationships between the objects, environment

The threat dimension pertains to the security of data or objects and includes unauthorized observation of objects or data, unauthorized destruction of objects or data, unauthorized modification of data or objects, unauthorized creation of objects or data. The environmental dimension examines the surroundings of the data or objects. The nature of vulnerabilities dimension examines the relationship between object, environment, and affect.

There are four questions addressed and numerous decompositions arise from them. The four questions concern the object affected by a vulnerability, the effect on the object, the method used to affect the object, and the nature of the input resulting in the affectation of the object. Affectation includes command prompts, user or system files, stack codes, passwords, web sessions, net sessions, CPU time, and memory use. There are 34 elements in the nature of vulnerabilities dimension; a general sampling considers direct and indirect impact. Direct impacts involve immediate results of vulnerability exploitation and include change in availability, unauthorized disclosure, and misrepresentation of information, repudiation of information, change in integrity, loss of confidentiality. Indirect impacts refer to eventual effects of the exploitation, with intermediary steps or actions such as access to external systems, elevation of privilege, internal system disclosure, external system changes in availability, integrity, and loss of confidentiality.

2.2 Taxonomies of Faults

What follows is a review of work in efforts to classify faults in software; this work relates to vulnerabilities because many vulnerabilities proceed from faults in executing software. An understanding of faults can contribute to classification of vulnerabilities. In a study of a real-time software system by Rubey, ten divisions of faults were identified [43]. Four divisions of faults were outside software functionality; these include incorrect documentation that would lead to user-induced errors, variations of programming standards that could lead to unpredictable behavior or performance, erroneous specification, and deviation from specification where results may not be acceptable in customer acceptance procedures. The remaining six divisions pertain directly to software function and performance [43]. Rubey's classification is summarized in Table 2.9. Another study conducted on faults found in a compiler for the TLR language was reported by Portier, et al. The results were compared with two other studies. A cross section of the three studies was measured against eight categories, listed here in decreasing frequency of encounter:

logic errors, interface errors, data handling errors, data definition, I/O, computational, and database errors [41].

Table 2.9 Rubey's Software Fault Classification

Fault classification in software	Description	Functionality
Incorrect documentation	x	
Variance of programming standards	x	
Erroneous specification	x	
Deviation from specification	x	
Erroneous data access		x
Erroneous decision logic and sequencing		x
Erroneous algorithms		x
Invalid timing		x
Improper interrupt handling		x
Incorrect definition of constants		x

A similar study conducted by Beizer [7], who collected statements from programs, and analyzed results from four other researchers. He identified an aggregated sample space of 2,070 faults. Based on these results, Beizer developed a taxonomy of the faults, which is shown in Table 2.10.

Table 2.10 Beizer Software Fault Taxonomy

Dimension	Elements
Functional	specification, operation, test
System	interfaces, devices, software, sequences
Process	initialization, control, arithmetic, static logic
Data	type, initial value, structure
Code	source, libraries
Documentation	installation, user instructions, administration
Standards	language, protocol
Other	

The functional dimension includes the specification, test, and operation of the software. The system dimension encompasses internal interfaces, hardware (I/O devices), operating system, software architecture, system control and sequencing, and resources. The process dimension considers the software process, performance, arithmetic, initialization, control and sequencing, static logic, and allows for a catchall other element. The data dimension addresses aspects of data type, initial value, and structure. The code dimension examines faults (defects in this case) in the source software and included libraries. The documentation dimension covers installation, usage and administration guidance provided for the user. The standards dimension is also included since standards are often formulated by consensus (vendor-developer), sometimes a lack of clarity exists which results in differing interpretation, and implementation of a standard. The taxonomy also includes the other catchall category to include dispersed intangible causes of faults. The functional and process categories contributed to over half of the defects, with the specification sub category contributing the most [7].

These studies indicate that faults can be attributed to any aspect of system development. This shows that elimination of vulnerability in software products and systems is a long-term goal. The IT community must be prepared to address vulnerabilities in the products that they acquire. The various categories and dimensions can lead to an elaborate classification system; however, the truly relevant aspects for system owners need not span the full set of dimensions revealed in the literature.

2.3 Taxonomy of Security Policy

Sparse work exists in the area of classifying security policies. Smith and Newton developed a security policy taxonomy [50], and published it during the 23rd National Information Systems Security Conference in October 2000. This taxonomy is based on the ISO 15408 1999 Common Criteria for Information Technology Security Evaluation (CC) [16]. The taxonomy was developed as an aid to CC Protection Profile developers who must map the profiles to specific user community's organizational security policy. The taxonomy is derived from the Policy/Requirement Hierarchy (P/R) and follows from it. The Smith and Newton security policy classification is shown in Table 2.11.

Table 2.11 Smith/Newton Security Policy Taxonomy

Class	Element
Functionality	security service provision areas
Assurance	product, development,
Management	procedures, planning, training

Within the CC, the functionality dimension considers the security functional areas as opposed to product functionality. The areas include confidentiality, integrity, availability, authenticity, accountability, non repudiation, and generic system access. The assurance dimension takes into account the extent of maturity of product behavior, stability of the development environment and software development discipline that is exercised in producing the product; areas include system/product assurance and developmental assurance. The management/administrative dimension addresses the peripheral matters to the product itself, yet form a critical link to successful use and deployment of the product.

Such things as training, procedures, system use, system administration, and contingency planning are considered. Smith and Newton stated plans to include extension of the taxonomy to include the supporting Target of Evaluation (TOE) that addresses the Policy Requirements within the CC framework [50].

2.4 Discussion of Metrics

Organizations seeking improvement need to know how well they are performing each task, since this provides guidance for focus on refinement and in making investment related decisions. Profit motivated organizations favor investments that promise the greatest return or reduce the cost of ownership. Government and Defense organizations focus on compliance to numerous regulations and standards. Such regulations mandate metrics programs to measure compliance; many are included in Starret [52]. Selection of metrics requires care in that the choice of metrics may cause an organization to improve only in those areas for which they can obtain a measure or a metric [28]. When examined in another way, metrics do provide a basis of focus for the organization. This is true if an effort to identify the optimal set of metrics is made. Additional data from excessive metrics may distract, or even confuse, the decision makers by presenting too many options to address or improve IA posture or behavior. This latter phenomenon is a form of paralysis by analysis. Metrics programs ideally serve to guide the improvement process and measure progress toward improving aspects of a given program. The active use of metrics should follow a set of priorities that the organization can manage. They are most effective with regular and open communication, and all stakeholders should participate in the choice of metrics [63].

In the area of vulnerability assessment, with new discoveries occurring daily and the rate of new vulnerabilities doubling yearly for the past four years [17], one may conclude that attainment of complete security is impossible. Within IA, there are few published standards in these areas, although there is research ongoing in IA metrics. Work in this area is summarized below.

2.4.1 The State of IA Metrics

Leading researchers in the IA field convened the "First Workshop in Information System Security Scoring and Ranking" in 2001 to discuss the state and future of measurement and metrics as applied to the IA discipline. Those in attendance agreed that certain aspects are difficult to measure, but that measurement of those aspects that are measurable should be done [25]. The participants summarized the problems in IA metrics and measurement as non-uniformity in the understanding of metrics, metrics become ends in themselves and lose value with the user/consumer and, metrics are reported in contexts beyond their original intent. The purposes of metrics were divided into two general classifications:

1. To aid in decision support, and,
2. To show a measure of progress in support of mandated reporting of such progress.

The tracks of the workshop also illustrate aspects of metrics. Technically, metrics can be used to compare objects – algorithms, specifications, architectures, and other aspects of a system life cycle. Organizationally, metrics can describe and track the effectiveness of programs or initiatives. Operationally, metrics describe the risks to environments, and thus assist in management of those environments. Two other areas for the use of metrics include the ability to describe and assess expertise, and within the environment, metrics describe the security relevant aspects of threats. Observations by the participants included that in a technical sense, very little work has been done in metrics to describe and compare products or technologies. The most rigorous works in IA pay little attention to metrics. The US Department of Defense, Trusted Computer Systems Evaluation Criteria (TCSEC) - 'Orange Book' did not address metrics beyond the defined evaluation levels for products. Its successor the (ISO IS) 15408 1999, Common Criteria for Information Technology Security Evaluation standard succeeds the TCSEC and provides narrative descriptions, but does not incorporate defined metrics. The product development organization is free to define them in the respective Security Target (ST) specification. The workshop participants agreed on the following uses of technical metrics:

1. They establish goals and measure how well the available technologies or tools meet the goals,
2. They are most useful when the assigned value is valid for most of the product's lifetime,

3. When used for predictions of performance, they must be based on a model of IA in which values assigned are significant factors in system security; a counter example of this is the fact that within the TCSEC framework when combining a database and a system, each given a C2 rating, do not necessarily equate to full C2 protection,
4. When used as predictors, must be based on a model where the future resembles the past; a counter illustration is that a lack of discovery of vulnerabilities in a product to date does not guarantee that none will be found. Data must be collected without foreknowledge of how it would be used,
5. Respective communities use them differently. Government users are primarily concerned with product ratings such as the CC and compliance issues, while commercial users are concerned with implementing the overall risk management process with less concern for sanctioned evaluation results [25].

Organizational metrics track the progress of success of an organization and its implemented programs. Examples include the DOD's IA Vulnerability Alerts (IAVA), which are vulnerabilities encountered by DOD system administration personnel in the course of their work [58]. Another metric is the percentage of trained or certified IA personnel. Organizational questions include how well security concerns are identified, and measurement of the effectiveness of vulnerability management programs. How well are mandates being met [25]?

Operational metrics describe and help manage risks and threats in the environment. This study is concerned with this classification of metrics as they relate to the detection and removal of vulnerabilities. Examples of operational metrics include risk assessment metrics, and the associated component parts of asset value, impact severity, threats, vulnerability, and effectiveness of security measures. Realistic metrics here include the number of advisories or reported vulnerabilities repaired, the time spent in correction, and testing of corrections made, percentage of systems remediated. The controllable portion of the environment includes physical, procedural, and personnel related security measures. A quantitative approach is described in the literature. The measures include counts of vulnerabilities found, intrusions, or viruses detected. These measures do not address readiness, nor do they adequately inform managers of violations in system security policies. A need to monitor and track operational performance was also addressed at the workshop. Product vendors must address this matter in presenting trend analysis and supporting data for their products [25].

Conclusions from the workshop include:

1. No single metric is sufficient to quantify the assurance provided by a system,
2. Software and systems engineering are closely related to the assurance provided by a product,
3. Penetration testing is in wide use as a form of metric. However, it is seldom reproducible. It does provide value to organizations using it,

4. Governmental and commercial users have different motivations, being policy and profits focused respectively,
5. Measuring “defense in depth” is critical and warrants more research,
6. Metrics that incorporate processes, procedures, tools, and people will remain critical, as all four of these aspects combined constitute IA within an IT environment.

Metrics must also evolve with techniques and technology. Metrics must be updated or replaced as objectives are met, and new ones are enacted to replace them, as this implies that the nature and definition of progress changes. Better models of system behavior are needed to develop predictive technical metrics. Specific models of interrelationships between subsystems are needed [25].

Another recent measurement initiative, developed and supported by the MITRE Corp., concerns the unification of nomenclature for discovered vulnerabilities. The Common Vulnerabilities and Exposures (CVE) presents a cataloging system and basic guidelines that IT and security product vendors are encouraged to adopt. The CVE evolved from a vulnerability database forum at Purdue University. As of the April 2, 2003 release, over 6,400 vulnerabilities are either formally accepted by the voting membership, or were recognized as in candidate status – worthy of research and consideration. The CVE catalog can be used as a basis to compare vulnerability scanners on a common footing. The CVE catalog began in 1999, and thus shows that an average of over 1,300 vulnerabilities are added each year, yet the rate of introduction of vulnerabilities doubles yearly [35].

The Defense Information Systems Agency (DISA), with oversight and leadership responsibilities in the Defense Information Infrastructure (DII), instituted the IAVA program. Within this program, vulnerabilities of concern to the DOD are announced, and the respective IT staffs must remediate them to remain in compliance. Three severity levels are described with the critical ones requiring less than one-month mitigation. Vulnerability scanner vendors release updated signatures on a monthly, or more frequent, interval [55].

The Systems Administration, Networks, and Security (SANS) Institute is an organization of IA professionals that plays an active role in security education and awareness. SANS sponsors numerous conferences, provides web based training, offers certification in security and incident handling including their Global Information Assurance Certification (GIAC) track. GIAC certification is sought after by corporations hiring information systems security personnel. SANS provides links to many assessment and measurement tools. Along with the FBI and international organizations, they publish a yearly list of top vulnerabilities. Their 2003 Top 20 includes ten Windows and ten UNIX categories, and all examples in each category are cross-referenced to the CVE catalog. They co-sponsor the Center for Internet Security (CIS) [12], which also provides host based assessment-scoring tools. Also provided is the Security Consensus Operational Readiness Evaluation (SCORE) site, which includes numerous checklists and discussion forums with leading practitioners within many system specific disciplines – Cisco®, Personal Digital Assistants (PDA), Linux, and Windows® et al [45].

The Dartmouth College Institute for Security Technology Studies developed a certification suite for the 2000 SANS Top 10 vulnerabilities, and publishes results of those products that have demonstrated acceptable ability to correctly identify the vulnerabilities in systems against the SANS vulnerabilities [26].

We now address aspects of measurement in order to devise techniques for determining the strength of IA tools, and to provide a framework by which an organization is able to mark its progress toward a strong IA posture. There are several perspectives or dimensions by which tools can be measured. The dimension of assurance level of the tools including their life cycle management from requirements, through design, implementation, testing, vulnerability assessment, maintenance, configuration management and quality assurance practices of the vendor was difficult at best for a purchaser to determine until the advent of the ISO 15408 1999 Common Criteria. Vendors can seek Evaluated Assurance Levels (EAL) for their tools and developer capabilities in the areas of security function, assurance, and management [16]. A vendor subjecting a product to this analysis shows a desire to independently assess the product and demonstrate trustworthiness of the product, and of their ability to support and improve it. Since thorough research into vulnerability assessment, tool capability is costly for every potential stakeholder to undertake, many choose to trust the informed opinion of security research wherever they can find it. A review of research and literature related measurement and metrics of software product quality follows. From this work the foundations for the use of metrics within IA and specifically vulnerability assessment tools can be developed.

2.4.2 The Properties of Useful Metrics

When we speak of metrics, there is an implication of a measurement system and a baseline from which the measurements are taken, and evaluated. Metrics have quantitative, qualitative, and purpose dimensions. We examine each of these in this section. Quantitatively, there are four general types of measurement scales in use: categorical, ordinal, interval, and ratio. Categorical, (or nominal) scales are useful as classifiers (truth and falsehood in logic), and are sufficient when equivalences are available or derived; mathematically, category scales are equivalence relations [23]. Ordinal scales are useful in establishing orderings; mathematically, they are linear orderings [23]. Ordinal scales preserve the order of elements [64]. Interval scales are useful when comparing differences such as temperature changes; mathematically, interval scales are an ordered Abelian group [23]. Interval scales preserve ordering and differences between elements in the group [64]. The term Abelian refers to the results of Niels Henrik Abel, a Norwegian mathematician in the early 1800's, who independently developed the foundations of the branch of mathematics known as group theory. A group exists under a binary operation (+,-,*, /) when the four fundamental properties of closure, associativity, identity and inverse properties hold. Abelian groups also satisfy the commutativity property (ab=ba). When the group's members can be aligned based on a ranking, the group is ordered. The fourth measurement class is that of ratio scales, useful when comparing equality between ratios; length and weight are common examples; mathematically, ratio scales are Archimedean ordered fields [23]. Ratio scales preserve ordering, intervals, and ratios between elements in the group or field [64]. Fields are defined when the Field Axioms (commutativity, associativity, distributivity, identity, and inverse) apply. Archimedean ordered fields have the property of holding true for applications of Archimedes' Axiom across all members. Archimedes' Axiom states that for members within a field (a,b,c,d): and scalar values m, and n; for each of the inequality operators (<), >), and the equality operator (=), that if ma < nb; then mc <nb, when m < n This holds for (<) and (=) also [64]. The measurements of different IA properties can be mapped to one of these measurement scale types. Categorical scales are useful for describing the state of applied security among systems; we can describe each system as secured or unsecured based on our knowledge and testing that the systems security protection meets established standards. A system with rigid password enforcement, a full set of operating system and application patches, and minimal services running might be an organization's concept of a secured system. Ordinal scales are commonplace in vulnerability assessment. The description of vulnerability severity used by antivirus vendors, and security researchers in discussing severity of exploitations are examples of usage of the terms high, medium, and low. Each vendor applies its own criteria and descriptions to each of these terms. High might mean near certain likelihood of being exposed to a virus, or to the ability to obtain system level privileges immediately through exploitation of a flaw. Another example within ordinal classification is the rating system such as 1-10. McClure, Scambray, and Kurtz employ this scheme in their book "Hacking Exposed" to discuss the measures of risk rating based on popularity, simplicity, and impact of an exploit [34]. Interval scales within IA relate to observances of measured quantities such as cumulative intrusion attempts detected within a period when compared with a similar period in the past. Another example is the number of known patches not applied on a given day, versus a previous day in the past. Such measures would provide useful information on the relative security posture of the system or organization. Ratio scales in IA would be useful for assessing the relative security posture between peer departments or divisions, which may be of differing size and system compositions.

We must also reason about the definition of terms used in assessing security or assurance such as the meaning of “secure” – or “assurance” in determining when a level of sufficiency has been reached. This uncertainty determination is addressed in fuzzy logic [65], which attempts to address uncertain classifications through the creation of an ordinal or interval scale between categories. Fuzzy set membership is one such example. Intelligent, intrusion detection systems can be built upon fuzzy-ordered measurements of degree of attack or anomaly [11]. Fuzzy measures are used to provide finer grains of differentiation within ratio, interval, or in extreme cases, ordinal scales [28].

The science of measurement is consulted to provide direction in selecting or constructing useful measures and metrics. Properties of metrics have received much attention in the Software Engineering discipline. A set of properties that also applies to software products, such as vulnerability assessment tools, will help to provide the focus and evaluation criteria for solid metric selections. Kitchenham, Pfleeger, and Fenton have studied measurement in software engineering. Their work treats the aspects of measurement by creating models of the measurement process. Among the rules they suggest are these [29]:

- Use measures that you understand in the context of your own goals or situation,
- If you are concerned with multi-dimensional measures for attributes such as quality, or complexity, use different measures for each aspect of the attribute,
- Do not try to combine different aspects of an attribute into a single measure unless you have a model or theory to support the combination,
- If you measure different aspects of a multi-dimensional attribute and want to predict some other attribute, use step-wise multivariate linear regression to combine the attributes into a single predictive model. Using a stepwise algorithm ensures that only aspects that contribute to the prediction are used in the model,
- If you use a predictive model that is based on analyzing empirical data, be cautious, as the model is unlikely to reflect a truth of nature.

A model of measurement consists of a unit definition, instrumentation, attribute relationships, measurement protocols, and entity population. Entities are the subjects of observation, and attributes are observable properties of entities. A measurement maps an empirical property of an attribute to formal mathematical constructs. Units refer to denominations in measurement, e.g., temperature degrees can be in Kelvin, Fahrenheit, or Celsius. Values refer to the members of each group or field within a scale as discussed earlier. Values have permissible and non-permissible bounds, e.g., ordinal scale values must be non-decreasing integers beginning at one. Instrumentation can vary with the units or scale in use – measuring tape vs. triangulation by position and radar range finding, and can serve to classify an entity based on its mode of measurement, e.g., gender or Boolean truth-value.

The Kitchenham, Pfleeger, and Fenton model also stipulates indirect measurement as being valuable. An example is the use of size of a software product to reflect effort expended to develop it. In vulnerability assessment, false positive frequency could be used to reflect trustworthiness. Compound measures involve coupling simple measures into ratios for sake of comparison, e.g., vulnerabilities found per machine scanned. This also implies that indirect measures and compound measures must be based on a relationship model of how the indirect measure relates to the attribute. Magnitude comparisons of compound measures without a relational model would be meaningless.

Kitchenham, Pfleeger, and Fenton based their definitions of the components of their measurement model with models of each component, measurement models, instrumentation models, and entity models. They further developed validation methods for each of the component models [29]. Schneidewind had previously done work on metrics validation and related them to aspects of quality including association, consistency, discriminative power, tracking, predictability, and repeatability. Table 2.12 explains Schneidewind's research in the context of quality measurement [46].

2.4.3 Applied Metrics and Measures

Practical work on implementation of metrics can be found in journals such as CrossTalk, a journal of the defense software industry, and includes articles by leading software engineers working on defense projects. Articles on specific metrics, implementation of measurement programs, relationship to the Software Engineering Institute's Capability Maturity Model (SEI CMM), and discussion of tools are among those published [54].

Having examined the quantitative aspects of metrics, and the micro level qualitative aspects of metrics it is important to examine qualitative ones also. Qualitative properties can be applied to macro and micro considerations. In the macro scale, we consider combinations of metrics having different purposes to bring out a comprehensive assessment of a tool or process.

Table 2.12 Schneidewind's Metrics Validation

Quality Function	Scale	Method	Measurement Property	Validity Criterion	Purpose of Valid Metric	Statistical Method
Quality assessment	Interval	Parametric	Difference	Association	Assess differences in quality	1. Coefficient of determination 2. HO population correlation coefficient = 0. 3. HO: Population correlation coefficient 4. Linear partial correlation coefficient. (metric normalization. Accounting for size). 5. Population correlation coefficient confidence interval. 6. Factor analysis (tests of independence).
Quality assessment	Ordinal	Non parametric	Higher/lower	Consistency	Assess relative quality	1. Rank correlation coefficient
Quality control	Nominal	Non parametric	High/low	Discriminative Power	Control Quality (discriminate between high and low)	1. Mann-Whitney comparison of average ranks of Two Groups of components. 2. Chi-square contingency table for finding critical value of metric. 3. Short-cut technique for finding critical value of metric. 4. Sensitivity analysis of critical value of metric. 5. Rrusal-Wallis Test of average metric rank per given value of quality factor. 6. Discriminant analysis (use of a single metric's mean as discriminator).
Quality control	Nominal	Non parametric	Increment	Tracking	Control quality (track changes)	1. Binary Sequences Test and Wald-Wolfowitz Runs Test.
Quality prediction	Interval, ratio	Parametric	% Accuracy	Predictability	Predict quality	1. Scatter plot to investigate linearity. 2. Linear regression. a. Test assumptions b. Examine residuals 3. Find confidence and prediction intervals. 4. Test for predictability threshold, and repeatability thresholds 5. Non-linear regression. 6. Multiple-linear regression. . Test assumptions b. Examine residuals c. Test for predictability threshold and repeatability threshold.
All Quality functions	Ratio	Parametric	% Success	Repeatability	Ensure metric validated with specified success rate	Ratio of validations to total trials threshold.

Regarding macro properties and qualities that sound metrics should possess, there is a growing business in software metrics. Software Research, Inc. developed the TestWorks Quality Index to address the relative quality of software products independent of the process maturity of the software development process; static and dynamic performance considerations are addressed. The TestWorks Quality Index criteria describe the attributes of measures of quality. Among their observations is that no more than ten factors should be considered in developing a comparative metric. Of these ten, Software Research made the following recommendations for product evaluation metrics:

- At least half of the factors should be quantitative,
- At least three of them should address static measures,
- At least three factors should be dynamic measures of the product,
- At least one should address the product's development process maturity,
- At least one should address the quality need as addressed from outside – e.g., the cost of repairing defects.

The criteria may address more than one factor within each metric as otherwise; this list would easily exceed the ten prescribed [51].

Micro level qualitative properties are applied to individual metrics to understand the overall quality of the resulting measure. A good list of such properties was discussed during the workshop on Information Assurance Rating and Ranking discussed earlier. The workshop's list of attributes for metrics includes:

1. Scope – the portion of the IA problem domain for a given metric should be specified clearly,
2. Sound foundation – the metric should be based on a well-defined model of the portion of IA that it represents,
3. Assessment process – the process for assessing the metric should be well-defined, repeatable, and reproducible,
4. Relevance – the metric should be of value to IA stakeholders,
5. Effectiveness – it should be possible to evaluate against the metric quickly and with sufficiently low cost [25].

Standards oriented research into evaluation of software products also contributes to technical assessments of IA tools. With the increasing emphasis placed by the software consumer on its product providers following mature, repeatable, processes in order to deliver trustworthy products, many organizations are striving for ISO 9000, and 14000 certifications or SEI-CMM-I certification. Punter, van Solingen, and Trienkins discuss the state of this field in their paper, and apply the recommendations in ISO 14598, which addresses evaluation techniques, and ISO 9126, which addresses the evaluation process. Product evaluation supports investment proposals, when choosing among contending options. The quality characteristics outlined in ISO 9126 include functionality, reliability, usability, efficiency, maintainability, and portability. Each of these also has subclasses. Security is addressed in the functionality section, and pertains to the security of the product itself. ISO 14598 focuses on the evaluation process from several stakeholder viewpoints, with an entire document dedicated to each. The developer, acquirer, and evaluator viewpoints each are considered in software product evaluation. The standard identifies four phases of the evaluation: analysis, specification, design, and execution. Each phase has unique meaning for each stakeholder. For the evaluator, the four steps are to define the object of the evaluation, define the scope of the evaluation, document the evaluation process, and perform the evaluation and capture results. The authors also discuss the concept of levels of evaluation based on levels of capability or decomposition and detail of processes. Their argument is that based on function or consequences of success or failure of the software, some degree of choice can be made by the acquirer as to the extensiveness of the evaluation and on the definition of pass or failure. This would allow medical equipment control software to be evaluated with more rigor than would be needed for office productivity sotware. This all leads to the definition of a ‘Quality Profile,’ that is a specification of requirements to which an evaluation is coupled [42].

Many products are judged for acceptability on their ease of use, these judgments are qualitative and subjective in nature, and this is likely to remain true. However, much research has been done in usability evaluation methodologies. Zhang lists several [66]. Two such methods include an intuitive method, and heuristic evaluation [37], that can readily be constructed into an ease-of-use metric. An empirical approach on usability by Zhang, Basili, and Schneiderman also seems a reasonable approach [67].

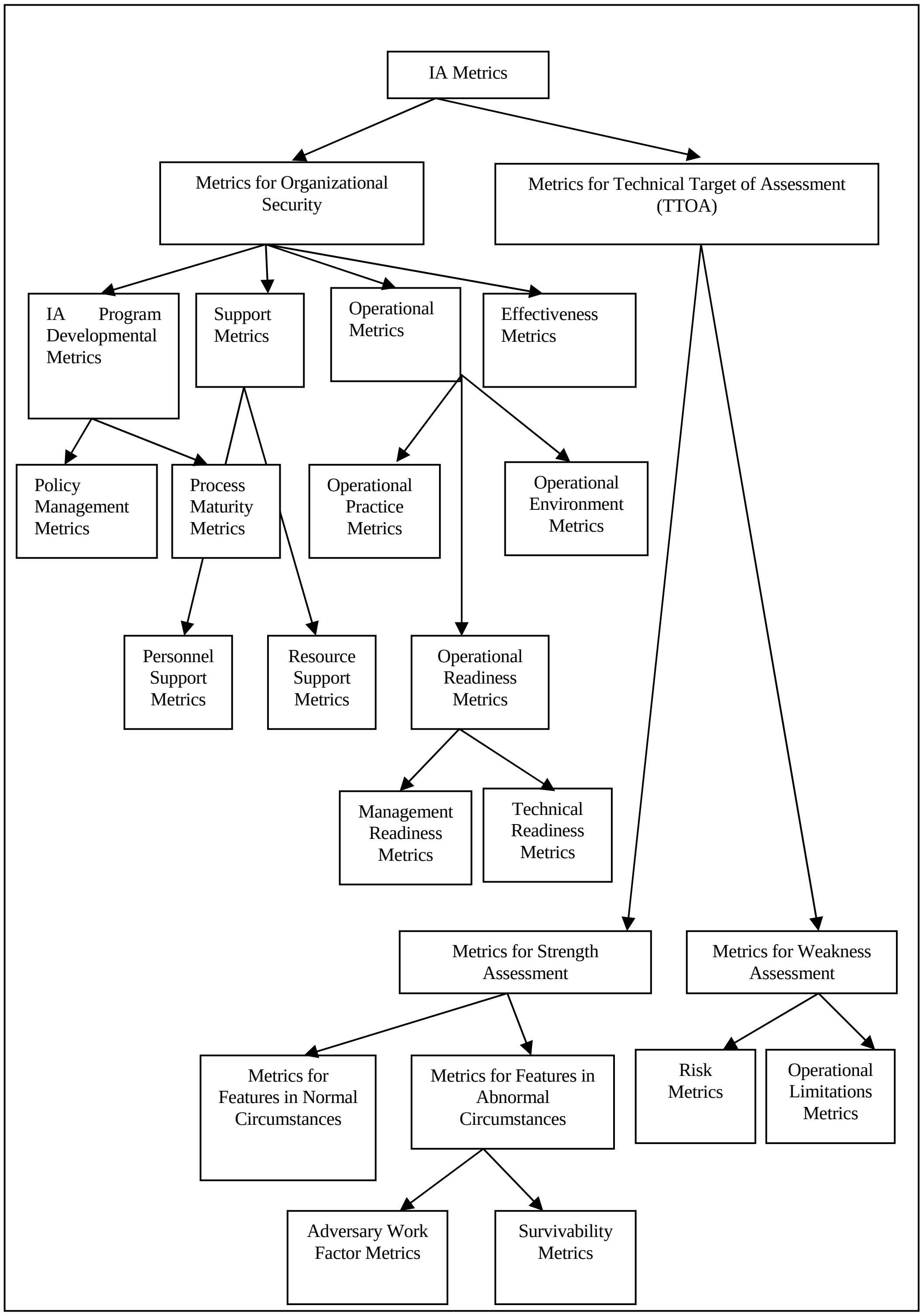
2.5 The Information Assurance Metrics Problem

The problem of IA metrics encompasses the security of the IT infrastructure from initial requirements to decommissioning and replacement of components, and considers policy, personnel, and regulatory, as well as technical aspects of performance of operations. The assessment of assurance provided must consider the full product development process, and such metrics should be developed, or adopted by each stake holding organization. The ISO 9000/14000 and ISO IS 17799 and the SEI's CMM and accompanying systems security engineering (SSE-CMM) are all worthy of consideration. The Common Criteria evaluation process also considers life cycle and product assurance measures in addition to security functional performance of the product. Personnel and regulatory measurement and metrics are also necessary. Metrics for the number of courses completed, certifications, degrees obtained, or hours of training per IA worker can be established. The technical components of IA include firewalls, Virtual Private Networking, anti-virus software, intrusion detection systems, authentication tokens, smart cards, disaster recovery, high-availability components, vulnerability assessment software, system and network management systems, and others. Each of these provides a measure of depth in the organization's IT defenses. The US Defense Department's defense-in-depth strategy is formulated upon this thinking [5, 58].

There are other areas in the operational and technical dimensions of IA, which have had less formalism or rigor invested in their measurements. We have few metrics for the strength or resistive ability of firewalls and authentication systems; other than password size, aging, and algorithm assessments. Just as each organization is different, and faces different quantities of threats, there is a notion of a best-fit capability for each product or technology within each organization. The study of metrics for goodness of fit for an organization and its security policy has not been studied and documented. Many measurements are of a qualitative nature, yet use of such measures is better than use of none, and experience with qualitative measures may lead to development of quantitative ones [25].

Developing metrics for technical capabilities in vulnerability assessment as relating to an organization's specific security policy is the focus of this study, and any metrics proposed will consider the full set of aspects within the IA domain. Vaughn, Henning and Siraj proposed a taxonomy, shown in Figure 1 of IA metrics building on work and discussions held at the earlier discussed "Workshop on Information Security System Rating and Ranking," and "Approaches to measuring security" conducted by the Computer System Security and Privacy Advisory Board (CSSPAB). This taxonomy addresses metrics based on the division of organizational security, and technical capabilities. In our study of vulnerability assessment tool evaluation metrics and models, the technical target of assessment would seem to provide the guidance for the development of metrics and a fitness model. However, we should not overlook the areas of operational and organizational assessment in the evaluation of tools. If a tool can distinguish itself as organization or operations friendly, this could provide a very strong advantage for its selection [59].

Figure 1 IA Metrics Taxonomy


IA Metrics
Metrics for Organizational Security
Metrics for Technical Target of Assessment (TTOA)
IA Program Developmental Metrics
Support Metrics
Operational Metrics
Effectiveness Metrics
Policy Management Metrics
Process Maturity Metrics
Operational Practice Metrics
Operational Environment Metrics
Personnel Support Metrics
Resource Support Metrics
Operational Readiness Metrics
Management Readiness Metrics
Technical Readiness Metrics
Metrics for Strength Assessment
Metrics for Weakness Assessment
Metrics for Features in Normal Circumstances
Metrics for Features in Abnormal Circumstances
Risk Metrics
Operational Limitations Metrics
Adversary Work Factor Metrics
Survivability Metrics

2.6 Discussion of Vulnerability Assessment

To understand vulnerability, it is important to understand the contexts in which vulnerability is discovered and exploited. Alford describes cyber warfare taxonomy [2]. McClure, Scambray, and Kurtz describe the taxonomy of a computer penetration [34]. Their classifications are similar. Cyber-warfare is comprised of four stages: cyber-infiltration, cyber-manipulation, cyber-assault, and cyber-raid. Generally, infiltration and manipulation occur in sequence producing a state wherein either an assault, or raid or both are possible [2]. The computer penetration taxonomy is more detailed and includes: 1. Foot printing, 2. Scanning, 3. Enumeration, 4. Gaining access, 5. Escalating privilege, 6. Pilfering, 7. Covering tracks, 8. Creating back doors, 9. Denial of service [34].

Table 2.13 compares system compromise models and is interpreted bottom-up. The lowest levels are necessary to ascend to higher goals. These two classification systems overlap as illustrated in the table, with cyber infiltration being comprised of foot printing, scanning, and enumeration. Cyber manipulation consists of gaining access and escalating privilege. From here, the cyber assault equals denial of service, while the cyber raid equates to pilfering and covering of tracks. The creation of back doors is a step in subsequent cyber manipulations. Cyber warfare pertains to incidents with a nationalistic agenda, while cyber-crime pertains to a lack of a nationalistic agenda and the presence of a financial or emotional agenda. Thus, exploitation of vulnerabilities in civilian systems is cyber-crime whenever circumstances or consequences of such violate applicable laws.

Table 2.13 Comparison of System Compromise Models

Cyber-warfare/cyber-crime	Computer penetration
	Create Backdoor
	Covering Tracks
Cyber-raid	Pilfering
Cyber-assault	Denial of Service
	Privilege Escalation
Cyber-manipulation	Gain Access
	Enumeration
	Scanning
Cyber-infiltration	Foot printing

Additional definitions are now discussed which will assist in the discussion of vulnerability and its remediation. Threat, risk, vulnerability, and exposure, are interrelated. Vulnerability in the software context is a manifestation of software defects and faults, wherein a fault with security compromising consequences is an active definition of vulnerability. Security related deficiencies occur throughout the software life cycle: in requirements or design, flaws are passive in nature and are referred to as defects within software engineering literature. A defect introduced during design through implementation is considered a fault if manifested at execution time. Faults with associated severe security policy compromises are vulnerabilities, while those with less harmful, security policy violations are exposures [35]. An exposure is a manifestation of software defects and faults, which may result in a security policy infraction, or in the use of unnecessary services that are exploitable to reveal information about the system. Stated another way, vulnerability is the property of a system forfeiting security when exposed to an attack, and exposure is the increased opportunity of an attack. Attacks can be intentional, or unintentional. Threat is knowledge of the existence of a potential attack against a perceived or known weakness (vulnerability) of a system. Risk is the estimated likelihood of an attack against a known vulnerability. Residual risk is the calculated decision to accept a less than complete remediation to a vulnerable condition.

The discipline of vulnerability assessment is growing. An article in Computer Finance in late 2000 [18] projected a quadrupling in the vulnerability scanning market size from 2000 – 2004. Reasons given for acquisition of vulnerability scanners include:

- Ensuring that devices on the network are configured securely,
- Checking for access privileges and excessive privileges,
- Checking for the latest vulnerabilities,
- Checking password strength,
- Checking open accounts and accounts without passwords,
- Testing unique environments,
- Insuring security within all devices all the time,
- Making sure that devices such as the firewall and IDS are secure,
- Conducting OS detection,
- Helping ensure security policy,
- Mapping vulnerabilities along the network, and
- Additionally, some vendor tools can be used in simulation mode, playing the role of an adversary and launching denial of service attacks on the network.

The preceding list illustrates user expectations of vulnerability assessment software. The previous models of system attack also highlight areas of concern and needs for coverage by IA technology including vulnerability assessment tools. The leading tools make claims of covering the critical vulnerabilities, and most indeed do address them in general. The question for a tool user and security stakeholder is one of making the wisest choice. The desire would be to have some concrete basis for making this choice, and a means of finding the tool that best suits requirements. This leads to an examination of the avenues by which measurements in vulnerability assessment can be performed. The process of vulnerability assessment measurement can be approached from several perspectives. The approaches for IA measurement, and thus for vulnerability assessment, can be divided along five sets of axes:

1. Objective or subjective: the average number of vulnerabilities found per month is measurable; the knowledge needed to operate a tool is more subjective. However, subjective measures are more common in IA,
2. Quantitative or qualitative: the number of vulnerabilities found by a tool is a preferable measure, while the opinion on a relative scale, such as 1-5 of a tool is less desirable,
3. Dynamic or static: dynamic metrics are preferred since the goals of an organization change as it matures, and the nature of IA itself changes with time. Many penetration-testing measures are dynamic, as is the percentage of known vulnerabilities fixed per month. Static metrics include those taken at regular

intervals pertaining to policy or compliance- e.g., security refresher course completion by IT staff,

4. Absolute or relative: absolute metrics need no basis for comparison – the number of nodes licensed for a product. Relative metrics rely on comparison e.g., the raw number of vulnerabilities says little of the IA posture, while the relative severity and number at each severity level adds much to the IA posture,
5. Direct or indirect: direct IA metrics can be generated from observing the property that they measure; the number of invalid packets rejected by a firewall over a certain period. Indirect IA metrics are derived by evaluation, as is the purpose of the Common Criteria, or assessments of attributes such as risk. Although direct metrics are preferred, it is not possible to measure directly. In these cases, indirect measures are useful [60].

. In principal, the mission of vulnerability reduction is to identify software with vulnerabilities, identify a repair process, and verify that the repair was successful. There are many steps that contribute to these goals. Given that we must identify vulnerabilities, we then determine a method to detect the evidence of a vulnerability. Accompanying this is the need to remove the likelihood of misidentification (avoid false-positives) and to ensure that no vulnerability is overlooked (prevent against false-negatives). We must present the severity of the vulnerabilities and illustrate potential abuses and consequences of successful exploitation. This dimension is also relevant to the output of the vulnerability assessment tool. Another aspect concerning vulnerability assessment is the criteria used in determining whether a vulnerability exists. One viewpoint, vulnerable software, is that the mere existence of software or settings known to be associated with a vulnerability renders the vulnerability as real. The second viewpoint, vulnerable configuration, concerns the actual runtime state of the system. In this view, the presence of known vulnerable software is not a concern if this software is inactive at the time of analysis. Thus, a question arises: is the severity rating of a vulnerability based on potential since the vulnerable software exists, and it could be exploited, or is it based on more exact measures, such as the given service or process associated with vulnerable software was found running at the time of the assessment. The acquirer of vulnerability assessment software should seek the answer to this question from each candidate vendor, to understand how vulnerability is treated, and whether this view is acceptable for the acquiring organization.

Other considerations in evaluating vulnerability assessment tools are considered here. It is also important to provide an identifiable catalog or cross-reference information in order to compare results of tools. In repairing a confirmed vulnerability, the tool must provide a well-defined and repeatable process for performing the repair and provide interim solutions, if patch availability is unknown. The repair must define sufficient steps to prevent ambiguity or improper repairs. For each repair made, it is important to ensure that the correct patch is applied. This is assured given that a means for verification that the repair has been successful exists. This can be done by retesting for the original vulnerability, or it can be an independent confirmation from an outside source, using a different tool, or in difficult cases, a hand-on-mouse, visual inspection of settings.

Contemporary information security publications conduct comparisons in the interest of educating IA consumers to make good choices. Several reviews have been published, with varying opinion of the best tool. In the reviews, there has been no consensus choice as best; this is evidence that the current popular measurement process for such tools is not reproducible. Examination of the details of the reviews also illustrates this fact. In addition, the content of reviews of vulnerability assessment tools shows that a more rigorous treatment of such IA metrics is needed. In the comparison of system vulnerability scanning tools, the results to date are high level and subjective in nature. Two recent examples are in a January 2001 product comparison by Network Computing magazine [22], and a November 2001 review by NetworkWorld Fusion magazine [3]. Each evaluation tested tools in assessing against pre-configured systems with a selection of vulnerabilities and assessed the competing products in their use and ability to identify the list of vulnerabilities, made other observations such as timing, and indicated whether repair information was provided. The NetworkWorld Fusion review emphasized 17 vulnerabilities by which products were scored, although their systems had many more than this. A scoring sample of 17 vulnerabilities is statistically small, and unreliable as a product quality indicator. It should be noted that the vulnerabilities selected were good choices that a worthy tool should identify. Measurements in these reviews included time to complete a scan and number of vulnerabilities found. The respective vulnerability signature databases were measured, as was ease of use, and number of false-positive returns. The false-positive count is a valuable consideration because elimination of these greatly enhances usability and value of a tool, as well as increasing user confidence in its accuracy. The Network Computing review was more representative of a professional environment, although their inclusion of antiquated Red Hat Linux version 5.2 diminished their results, as versions 6.2 and 7.0 were in widespread use as of the research and publication dates [22]. In December 2001, the NSS Group, a professional computing-security testing laboratory in the United Kingdom published comparative results of five commercial vulnerability scanners, and provided a much more rigorous testing regimen. Among the criteria accounted for by NSS is this partial list: tool architecture, installation, configuration, reporting, and, analysis. In June 2003, Network Computing published results of vulnerability scanner comparisons, which illustrated many vendor offerings over a wide range of vulnerabilities. Tool features were examined, and shortcomings noted. This survey utilized the Mitre CVE catalog [35], which makes it more repeatable. Categories of reporting, coverage, performance, management, and price were assigned arbitrary weights and tabulated. The results show that the popular press continues to monitor the VA tool quality [38].

In the NSS Group study, vendors wishing to have their tools evaluated completed a questionnaire. The supplied answers to questions were taken into a qualitative consideration. The NSS Group’s Question categories include:

- Architecture (client/server, use of remote agents),
- Whether a central control console is provided for distributing of scanning agents,

- Documentation,
- Minimum system requirements,
- Protocol layer at which it operates (TCP/IP, Ethernet),
- Whether specific packet drivers are necessary,
- Whether authentication is done,
- Management of policies,
- Number of vulnerability signatures provided,
- Systems supported by product,
- Whether the administrator can customize scan criteria,
- Nature of attacks performed,
- Frequency of updates,
- Can updates be scheduled,
- Are results immediately available during scans,
- Nature of solutions provided: advice, patches, automatic repairs,
- Integration with other products,
- Nature of reports provided,
- Can scans be scheduled,
- Integration with IDS tools,
- Licensing matters, and
- Pricing.

Given this information, the evaluation is conducted and accuracy is validated both against the provided answers, and against the known state of the targeted network of systems. As for results published, the focus was on raw number of vulnerabilities found, and on the time spent obtaining the solution [39].

All of these comparisons used a predominantly qualitative testing and relativistic comparison of product assessment. This can be useful only if there is also an absolute basis from which to measure product strength. The number of vulnerabilities found is not always an absolute measure due to frequent occurrences of false-positive results. A challenge exists in constructing an absolute basis. The real purpose of any product comparison and analysis metric is to help each stakeholder in IA obtain the most effective solution for their environment. Utilizing a vulnerability classification framework that considers all computable aspects of information systems security policy can serve to establish an absolute basis. No existing framework completely does this, as was shown in section 2.3. This study will build on work of Vaughn, Henning, and Siraj [59] to produce such a framework from which a baseline can be constructed and tailored to a specific environment and policy constraint. This baseline will provide each stakeholder the opportunity to view their security policy as a whole, and to select those parts that are essential for coverage by a candidate IA vulnerability scanner. This will assist in devising complete, well-reasoned standards and requirements for a candidate tool. Once the standards and requirements are established, candidate tools can be assessed for goodness of fit.

The question of the impact or severity of a vulnerability is also universally unanswered, however every vulnerability analysis vendor provides a very user friendly rating scheme: that of the high/medium/low rating. The CERT Coordination Center has also developed an ordinal metric for many common vulnerabilities. Their numerical rating is based on answers to these questions [17]:

- Is information about the vulnerability widely available or known?
- Is the vulnerability being exploited in the incidents reported to the CERT/CC?
- Is the Internet Infrastructure at risk because of this vulnerability?
- How many systems on the Internet are at risk from this vulnerability?
- What is the impact of exploiting the vulnerability?
- How easy is it to exploit the vulnerability?
- What are the preconditions required to exploit the vulnerability?

This list illustrates a big picture view of severity, which the CERT/CC must undertake. Their severity assessment should be a strong factor in the estimation of severity for a local site. To this factor, the local site must assess the degree of reliance they have in the system or software subject to the vulnerability. An obvious example of this is that a Microsoft-only organization would place a very low weight on Sun Solaris desktop tool vulnerabilities.

2.7 Remediation Taxonomy

Another area of study, which has relevance to the evaluation of vulnerability assessment tools, is the understanding of vulnerability life cycles and aspects of repairing vulnerabilities in systems. Remediation strategies for vulnerabilities and exposures are considered here. As discussed earlier, vulnerabilities and exposures are consequences of faults. Fault remediation can be viewed in the sense of prevention of their entering the system, removal during the design through integration testing phases, and corrective maintenance during the operational portion of the system's life. Vulnerability tools are used during the operational phase of the life cycle, thus are a subset of corrective maintenance. The Common Criteria refers to vulnerability related corrective maintenance as the flaw remediation process [16]. Operational remediation can take the form of an update, patch, workaround, or nothing. Additional considerations must be made for associated side effects or trade-offs in applying remediation measures. Impact can be none, partial loss of functionality, or full loss of some functionality, such as would be the case when disabling a vulnerable service. Another possible outcome of remediation is the introduction of another vulnerability, also of concern is whether the remediation effort succeeds in removal of the original vulnerability. The level of effort in cost, time, and impacts on service are also important variables [33]. A vulnerability assessment tool's solution set will provide valuable insight into these options and consequences.

CHAPTER III

METHODOLOGY

The hypothesis under investigation by this study is that it is possible to quantify strength of vulnerability assessment scanning tools (VAST) given that a sufficiently discriminating set of measurable attributes is found.

The collected set of discriminating measures will comprise a best-fit model for finding the most appropriate VAST. In this chapter, we explain the methodology used to construct the metrics based fitness model. We begin with the IA metrics taxonomy developed by Vaughn, Henning, and Siraj [59], which categorizes aspects of the IA problem for which measures are needed and possible. The goal for a best-fit metrics model is to include in it, a minimal set of measures spanning the categories defined in the IA metrics taxonomy that show distinctions in one tool over the field of contenders, and reduce common denominators in VAST properties. It is useful to note that metrics are made up of comparisons or dimensions of measures. A measure is a simple, single dimensional quantity, while metrics can be combinations of measures. Most available vulnerability assessment data has been compiled through measures. It also makes intuitive sense that some classification areas within the IA metrics taxonomy have more importance than others do. The most important functions of a VAST are the correct identification of vulnerabilities, and the presentation of effective remedies for them. Once tool common denominators are identified, it is imperative to an acquirer to find measures that when applied show distinction between the tools. All VASTs find hosts, illuminate open ports and list vulnerabilities in targeted systems. Discriminators of breadth, speed, and accuracy are important measures. Many other parts of the IA metrics taxonomy address concerns of the organization and overall IA discipline. The VAST must also be assessed in its coverage of these areas. These areas present a broad opportunity for finding discriminating measures.

Our model derivation methodology for the best-fit IA metrics model for VAST follows the steps below.

1. The IA metrics taxonomy by Vaughn, Henning, and Siraj [59] is used as the initial classifier for measurement requirements.
2. Existing measures and desired traits are collected.
3. The collected measures are mapped into the IA metrics taxonomy.
4. Frequency of occurrence of measures from all sources is noted as this gives us a broad based view of the importance of each measure; this also facilitates weighting in the model.
5. A representative measure is selected from each area from within the taxonomy – forming the best-fit model. The measure that best represents the scope of the category is selected for inclusion.

Next, having a candidate model, we must validate it in a lab setting. Broad capability and single-system focused testing is needed to address the user concerns of breadth, depth, and accuracy. The remainder of this chapter discusses the model construction. The model validation is discussed in chapter IV, and the Appendix presents all measures taken during the study. Steps taken in validating the candidate model are -

1. Representative VASTs are identified and configured.
2. A set of target systems are set up as prime subjects of the tools in the study.
3. All assembled measures are taken on representative VASTs.
4. In-depth assessment of the results of the tools is done to determine the tool's accuracy of detection. Accuracy is validated by independent analysis by systems and security experts, and by analysis resulting from running the Center for Internet Security's CIS Level 2 Benchmarks [12].
5. The metrics model is validated against the in-depth results.

3.1 IA Metrics Taxonomy

As mentioned earlier, IA metrics classification work has been undertaken as a result of findings of IA workshops on metrics [25]. Vaughn, Henning, and Siraj published an IA metrics taxonomy that decomposes the IA discipline into categories. The high-level decomposition was presented in Figure 1. The taxonomy makes highest-level divisions on Organizational security and Technical Target of Assessment (TTOA). Most studies of tools of any sort focus on the technical properties and base comparisons on measured observations of performance. Note that in the case of VASTs the number of vulnerabilities found is important, however this by itself deserves a closer look.

Table 3.1 IA Metrics Taxonomy within Organizations

Metric Class	Purpose for VA Tool in the Organization
Policy management	assess compliance to policy
Process maturity	security improvements with time (trend analysis) are certified tools employed, Common Criteria, ICSA, et al
Personnel support	required skill level of tool user needed.
Resource support	tool cost of ownership, cost per month
Operational practice	amount of password, audit, permissions, groups, extraneous services, found
Operational environment	extent of exposure to threat or attack, shows open ports, services, and discusses severity
Management readiness	system auditing configuration
Technical readiness	overall vulnerability analysis, and remediation, support for community vulnerability catalogs – CVE, DoE CIAC, CERT/CC Advisories, Military/DoD IAVA compliance reporting, SANS/FBI consensus vulnerabilities (Top 20)
Effectiveness	detection of malicious codes, evidence of system intrusions (worm, virus et al)
TTOA features in normal circumstances	vulnerabilities found, Presentation of vulnerability and its solution, Instant Fix support
TTOA adversary work factor	penetration tested, reported vulnerabilities in tool, Independent vulnerability testing of tools
TTOA survivability	performance of a tool under stress or during and after an attack against it
TTOA risk	availability of tool resources to unauthorized users, consequences of inadvertent erroneous use – scans or fixes
TTOA operational limitations	breadth of targets supported, scalability, performance, network impact of use – load imposition, disruptions

Table 3.1 summarizes the IA metrics taxonomy from the perspective of contributions that a VAST can make toward an organization using it. VAST’s contribute

greatly to organizational security policy whether by direct impact of detecting and reducing the vulnerability and threat of exploitation, or by indirect impact of assessing compliance to policy or regulations.

Table 3.2 IA Metrics Taxonomy Properties of Tools

Metric Class	Properties of VA Scanning Tool
Policy management	tool has checks for areas of policy, password, audit, permissions, groups, extraneous services
Process maturity	certifications tool has attained, ICSA, Common Criteria
Personnel support	steps from launch to scan
Resource support	extra database (SQL Server), Interoperable with other infrastructure components
Operational practice	tool can be tailored to operational practices – policy oriented settings available
Operational environment	tool explains meaning of results, Number of checks available for Exposures
Management readiness	tool can be tuned to audit practices
Technical readiness	extent of support for community vulnerability catalogs – CVE, DoE CIAC, CERT/CC advisories, Military/DoD IAVA compliance reporting, SANS/FBI consensus vulnerabilities (Top 20)
Effectiveness	tool finds intrusion evidence, offers solutions
TTOA features in normal circumstances	tool/developing organization provides guidance documents, wizards, knowledge base to users
TTOA adversary work factor	independent published results of penetration or vulnerability analysis exists
TTOA survivability	how does it handle loss of database
TTOA risk	does tool make host vulnerable
TTOA operational limitations	special requirements for hosts or targets

Table 3.2 uses the same set of metrics classification and depicts properties (examples) of characteristics that can be measured for each category.

3.2 Collect Candidate Measures

In this section, we present measures that have been used by product reviewers, acquiring organizations, and security experts. There is a good body of past work on which to draw these measures, and many measures are common across all groups. This will be illustrated by frequency counts for respective measures [3, 22, 38, 39, 56, 68]. The measures are presented in tables 3.3 – 3.9. The Frequency column in the tables reflects a count of the sources that included the respective measure. A range from 0 – 6 is possible. We added measures to map to IA metrics taxonomy categories which were not included in the prior 6 references. Such measures have a count of 0 as none of the cited sources used it. The union set of all measures in this study forms an 'at-large' set, we use this term to refer to all measures in the remainder of this document.

The emphasis here is on collecting properties and measures that the security community has used, and map them into the IA metrics taxonomy to determine whether there is room to broaden and improve upon the measures taken in the evaluation of VASTs. Measures from a broad spectrum of sources were collected and classified within the IA metrics taxonomy. In addition to popular product review measures, properties desired by customers are included. Customer assessment data from large government organizations are represented. A third source is from professionals within the security community, who have used a large number of tools in their work and who gave input as to the properties needed most. The full range of measures is mapped into the IA metrics taxonomy. Not surprisingly all groups cite some similar measures, while other measures are important to a few acquirers. This enabled the accretion of a large set of candidate measures. The tables 3.3 – 3.9 focus on each class of metrics from the larger IA metrics taxonomy and show frequency counts for measures that intersect among the groups of tool evaluators. The tables group the metrics by their respective higher-level category (Figure 1) where possible. The frequency column refers to number of occurrences of those cited above that included this measurement within their product comparison or requirements list. Few measures appear in all cited sources. Our intention is to select measures that represent the intention of the category from the IA metrics taxonomy.

Table 3.3 IA Program Development Measures

Metric Class	Frequency	Types of VA Tool Measures
Policy management	2	password guidelines – aging, lockout, audit
Process maturity (tool developer)	0	certification of tool (EAL)
	0	process maturity of vendor (CMM)

Policy management metrics (Table 3.3) are specific to development of a security, policy, implementation of policy, and compliance with policy. They can be supported by a VAST by enabling the given policy to be encoded within the tool to identify areas out of compliance. Identification and authentication of users is a common policy area, as is access control of resources by users. This includes permissions lists, and appropriate time of day or days to be using the resources. Other areas defined in the policy should be reviewed when considering a VA tool.

Process maturity metrics (Table 3.3) assess security-engineering activities that span the life cycle of secured systems deployed by organizations. Examples here include the Common Criteria, that measures process factors of systems by ranking them in one of the seven evaluation assurance levels (EAL's); primarily by examining the artifacts of the development process [59]. In our context, the VAST does not assess its owner's organization processes; we look at the properties and qualities of the tool developing organization. It applies to known development processes used to develop the tool, and any related certifications that consider process maturity of the vendor in the evaluation process. The Common Criteria [16] does make this consideration. The process maturity rating of a vendor gives the customer of the tool documented evidence that problems or requests of the vendor will be dealt with.

Table 3.4 Support Measures

Metric Class	Frequency	Types of VA Tool Measures
Personnel support	2	tool training offered
	1	documentation
	1	user certification
	1	technical support
	1	vendor responsiveness
Resource support	2	tool costs, maintenance
	2	price
	2	speed
	2	supplier health (P/E ratio, market capitalization, revenue growth trend)
	1	value (weighted vulnerabilities found)
	1	cost is not important

Personnel support (Table 3.4) addresses the people within the organization and their level of competence. Properly trained and certified IA professionals are better equipped to maintain and improve the IA health of their organizations. Our interest concerns the availability of training and certification of users of tools and the metrics address the impact that the introduction of this tool into the workforce would have upon its personnel, and the rest of the organization. The user certification category pertains to the issuance of certifications of competence by the vendor or group for use of the tool. Vendor responsiveness relates to the potential lag time involved in taking matters of personnel support or problem resolution to the supplier and obtaining redress for it.

Resource support issues (Table 3.4) are concerned with direct material, financial and indirect human impact of using the tool. Resource support metrics serve as indicators of financial support within the organization, and available resources for IA programs and processes. The tool's price regulates the quantity of the tool that can be acquired. Tool speed determines work output from it per unit time. Supplier health addresses the potential for a long term relationship and experience base to be built between the user and supplier. The value measure can take on many forms, looking at the vulnerabilities found (or solutions per critical problem offered) per unit of money may shed interesting perspectives on the tool suitability. Company accounting practices such as overhead charging practices, depreciations, labor rates, should be included when making this measure to ensure that all pertinent factors are included.

Operational practice (Table 3.5) addresses the typical behavior of the organization regarding IT security. It is important that members of an organization understand and comply with security policies. The tool should be able to assess compliance rates by users; examples include password compliance, and workstation security settings analysis.

Table 3.5 Operational Practices Measures

Metric Class	Frequency	Types of VA Tool Measures
Operational practice	1	data management/mining/navigation
	3	update mechanisms, updates (automated, monthly)
	2	update handling
	1	installation, operation, ease of use
	1	assets can be rated for risk
	2	support
	1	schedulable assessments
	2	notification of scan completion
	1	access to product knowledge base
	1	links provided for more info
	3	deployment scalability
	1	users/roles defined
	1	multiple organizations supportable
	1	policy distribution and changes
	1	scan progress monitoring
	1	supports ticketing/remediation management
	1	logging
	1	audit support

The ease and ability to distribute the tool and its updates is also considered in this measurement area, as this affects the ability to measure compliance in larger distributed organizations. If the organization performs regular audit trail reviews, the tool must be able to support this practice. A second dimension of operational practice is in aiding in the IT security day-to-day practices. Tool update handling, distribution, role definitions, scan progress monitoring are all useful features for evaluating IT operational practices.

Operational environment measures (Table 3.6) describe and measure the security relevant aspects of the operational environment (i.e., external threats, conditions, objects) that affect the organization's security operations directly or indirectly. An example might be number of systems susceptible to a specific penetration technique [59]. Therefore, the ability of a tool to summarize the list of machines susceptible to a given vulnerability would be valuable.

Table 3.6 Operational Environment Measures

Metric Class	Frequency	Types of VA Tool Measures
Operational environment	3	command line automation
	1	exposure and visibility of environment to attackers – inside and out: ports, services, related vulnerabilities
	3	host vulnerabilities
	0	platform options
	3	network mapping, asset identification, annotations
	2	GUI attributes/behavior/usefulness
	3	can harm network (DoS attacks), Impact on network and hosts (DoS, loading)
	3	architecture (host/network based, agents, consoles, database)
	3	integration with IDS/firewall/VPN, console, integrate with data management software

Operational environment (Table 3.6) has a second dimension addressing the impact to the IT infrastructure from use of the tool. Will agents be installed on each machine; can the tool integrate with other security tools in building a defense in depth strategy? Does the tool allow for asset identification and management? What types of vulnerabilities will the tool find? Will it disrupt operations in doing so? Will it trigger alarms in the intrusion detection system, rendering superfluous analytical time ferreting out false intrusion alarms?

Management readiness (Table 3.7) addresses the commitment of management to IA processes within the organization. Thus, its measurement addresses the ability of the tool to support organizational or domain-wide vulnerability remediation behaviors and provide reporting.

Table 3.7 Readiness Measures

Metric Class	Frequency	Types of VA Tool Measures
Management readiness	2	detect audit behavior aging, audit data review and practices
	2	storage, activity, are scans saved in a database
	4	reporting abilities Reporting (flexibility, diversity, ease of use, exporting,)
	1	trend analysis
	1	incident handling
	1	forensics
Technical readiness	4	support of organizational vulnerability management (IAVA, SANS, CERT, etc). CVE Support
	2	automatic signature updates
	2	automated remediation extent

Forensics support, vulnerability trend watching, and incident handling characteristics are also considered in this area. The tool can assess audit configuration, examine use or presence of forensics tools, and can be assessed for its support to remediation.

Technical readiness measurement (Table 3.7) addresses the readiness state of technical support that affects the organization’s ability to provide information assurance while performing operational missions. These measures can be static or dynamic. Risk assessment and vulnerability analysis are examples of static technical readiness measurements [59]. The US Defense Department and military services issue alerts, bulletins and technical tips under their respective information assurance vulnerability alert (IAVA) program. Compliance to these alerts and bulletins is mandatory. Health care professionals must be concerned with compliance to the health insurance portability accountability act (HIPAA). Along with compliance profiles, reporting must be flexible and diverse to support the compliance programs. The tools that support the well-known readiness assessment efforts of the private sector (CERT Advisories, SANS Top 20), and governmental (CIAC, Homeland Security FedCIRC, IAVA) provide ready value to users. Tool vendors who address public policy such as HIPAA and Gramm-Leach-Bliley financial behaviors demonstrate understanding of the complex impact of these laws on organizations subject to them. On the active side of technical readiness is the responsiveness of tool suppliers to provide updates for disclosed vulnerabilities.

Tool effectiveness (Table 3.8) is concerned by addressing IT threats relative to the managed environment. Basic areas of vulnerability detection and remediation along with detection features for malicious code agents are considered. Metrics for the efficacy of the organization’s IA program providing defense in-depth assurance are considered here. Examples include the number of malicious code incidents (measures protection), number of intrusions reported (measures detection), and percentage of data recovered after a security incident (response). This area is the stock and trade of IA professionals and is given great treatment in public and academic literature. This area is also among the most important to the organization as far as VAST technology is concerned.

Table 3.8 Tool Effectiveness Measures

Metric Class	Frequency	Types of VA Tool Measures
Effectiveness	3	password strength
	2	intrusion attempts, malware detection
	1	analyzes wireless devices
	5	vulnerabilities found vulnerabilities (found, unique, false-positive, false-negative)
	0	remediation features (retest, undo, automated,)
	2	vulnerabilities checked entries in database
	3	customizable checks
	0	quality of solutions given

During this study, a vulnerability in the Microsoft remote procedure call (RPC) functionality and its integration with the distributed common object model (DCOM) was disclosed. Microsoft® Windows® uses the RPC mechanism extensively for provision of network-based services. DCOM is one such network-based service allowing applications to share a common messaging and computing framework. The effectiveness of the vendors as they handled the release of patches from Microsoft, and new exploits from the security research and hacking community was tracked, and recorded in the Appendix. Such incidents give insight into the potential for a vendor to provide rapid response to critical vulnerabilities in software.

Table 3.9 Technical Target of Assessment Measures

Metric Class	Frequency	Types of VA Tool Measures
TTOA features in normal circumstances	0	certification level of tool
	0	features (number, usefulness)
	2	network (mapping, monitoring, port scanner, password checker, c2 security
	2	editable configurations
	4	service packs, shares, hotfix ordering, Trojan horse detection, web server and browser analysis
	2	result view can be sorted, views configurable,
	2	scalability and performance
	1	system tool access
	1	solutions provided really work
TTOA adversary work factor	4	the tool has been analyzed for vulnerabilities
	1	service on non standard location
TTOA survivability	0	resistance to corruption,
	0	data preservation.
TTOA risks	0	known or discovered vulnerabilities,
	5	tool supports vulnerability discovery and repair,
	2	crash network/devices, lockout users,
	1	scans complete,
	2	inter component authentication,
	3	vendor is known,
	3	evaluation copy available.
TTOA operational limitations	2	infrastructure coverage,
	1	requirements for use,
	2	alerting,
	2	report combinations,
	2	system privilege required,
	1	tool supports interaction while scanning,
	1	scans can be paused,
	1	source code available,
	3	Compliance to standards.

The measurement area of technical target of assessment (TTOA) features in normal circumstances (Table 3.9) is concerned with day-to-day use and maintenance of the tool, and using the tool for day-to-day information assurance preservation. These metrics measure the capabilities that the TTOA should have in order to provide information assurance under normal circumstances. They can be used for assessing the claimed features of a TTOA [59]. This area differs from effectiveness discussed earlier in that capabilities and potential of the tool are considered here.

TTOA adversary work factor (Table 3.9) considers the performance of the tool under more hostile conditions. Adversary work factor is the amount of effort an adversary spends in order to compromise protective measure(s) of a system. It not only incorporates technical factors, but also personnel and operational factors [59]. Questions measured here include has the tool been subjected to independent vulnerability analysis to point out possible avenues of exploitation or misuse? This area also addresses whether atypical configurations in the environment can be supported, such as scanning for web servers on non-standard TCP ports. This latter measure considers detection of vulnerabilities when the organization has taken steps to inhibit penetration from attackers. In addition to hiding well-known services, such defenses as disabling registries and limiting file shares are commonly done. Tools that require this resistance to be lowered may introduce a risk that the defenses will not be fully restored after use.

The measurement area of survivability (Table 3.9) considers the resiliency of the tool to tampering or inadvertent configuration errors. Questions asked in devising metrics include are resources such as registry keys or databases protected? Are instructions available for locking down the tool if the installation process does not perform this service? What happens if components of the tool such as databases are modified or moved prior to use?

TTOA risk measures (Table 3.9) are those that measure threats, vulnerabilities and associated risks to the TTOA. A full treatment of risks to products would be beyond the scope of this study, however much can be learned from independent analyses of the product in question. There is a secondary question on risk to the organization concerning the tool. Can the tool be evaluated with little or no cost of time or money? Have any vulnerabilities been published concerning the tool itself? This may indicate issues in software design and reliability. If several components are deployed how is their inter component authentication treated? The reporting of vulnerabilities in a tool does indicate a modest attempt at independent analysis, however, it also points toward potential additional problems.

The operational limitation of a tool (Table 3.9) is the last measurement area. Questions sought by such metrics include, how much of the infrastructure is covered by the tool? What is required of the user and host system prior to and during use? How are alerts handled during use? Can the tool be multi-tasked during scans? Is there evidence that the tool supports existing standards in software or security? Are there limitations in reporting from the tool, such as limited formats or report options?

3.3 IA Metrics Best-fit Model

Until now we have mentioned various measures used in the discussion and comparison of VASTs. In the goal to derive a metrics model, we must construct metrics from the available measures, where metrics are the composition functions of two or more measures. A metric of two measures such as bytes per second is a simple metric. Complex metrics are composed of three or more measures to assess a property that is meaningful in the context of one's goal. The goal in selection of a VAST is to find the best tool for the job when considering factors that are influenced by the tool's choice within the organization.

The derivation of a best-fit metrics model for evaluation of VAST was undertaken using the IA metrics taxonomy of Vaughn, Henning, and Siraj [59]. Additionally, intuitive guidelines from the problem of vulnerability assessment are introduced. Metrics properties following from the work of Kitchenham, Pfleeger and Fenton [29] below are consulted in defining the nature of metrics to include in the model. Their guidelines built on work by Schneidewind [46] who discusses validation of metrics and appropriate metric types in the areas of quality assessment, quality control, and quality prediction. This model contains quality assessment and quality prediction aspects. The list below summarizes the metrics selection and validation goals comprising the VAST IA best-fit metrics model hereafter known as the model.

- The model must contain only measures that are repeatable and reproducible,
- The model must contain measures that are not complex to apply,

- The model must contain measures that reveal distinguishing traits of tools, yet also allow a single tool to be distinct,
- The model must contain measures from each measurement area from the IA metrics taxonomy derived by Vaughn, Henning, and Siraj [59]. A secondary goal of this study is to validate this taxonomy within the IA discipline,
- The model must contain measures that are meaningful in the context of our goals,
- When plausible, measures in the model must also be consistent with concerns and dimensions measured by preceding comparisons and analysis,
- The model must place prudent weightings on each represented measure, and be amenable to variances of priorities of the user. This is so, since each user has distinct configurations, and remediation priorities,
- If a measure is valid in more than one IA metrics area, this measure should be given greater weight, and earliest consideration,
- The model must be practical and usable.

The aversion to complex metrics will encourage the model's use, and will facilitate reliable validation of results. Given the above criteria, we now present the list of measures to constitute the best-fit model. The table below maps the single measure and its IA Metrics classification, along with the weight given to this measure. The weights range from one to three, with three being most important. Rationale for the choices follows the table below. These metrics will illuminate crucial aspects of vulnerability discovery and remediation. This process will serve to place relative significance to the elements of vulnerability assessment.

3.4 Map Model to IA Metrics Taxonomy

In section 3.1, we summarized measures that have been used to measure VAST attributes and showed factors that large organizations have used in making purchase decisions. We then set out to study whether additional measures were possible and map them into the IA metrics taxonomy.

The union set of results of this search for metrics and measures resulted in the creation of an at-large set. This at-large set was compiled during this study and analysis of the respective tools and their features. Table 3.17 shows the IA metrics taxonomy categories along with questions that one would ask in seeking metrics for the category. The Weight column serves to distinguish between the categories in the extent of influence that each has over the selection of a tool. The range in values is from 1 – 3, with 1 being lesser in influence. After identifying the field of measures and the list of tools to apply them, we then identify the short list of measures that we will show is able to predict a tool’s performance. The performance data is then presented to validate the list of measures.

The purpose of VASTs is to identify vulnerabilities in targeted systems; this also includes finding evidence that the system has evidence of an intrusion, such as a worm. Note that this is a slight overlap with anti-virus tools, which track all known and detect properties of many unknown malicious codes.

Table 3.10 IA Metrics Best-fit VAST Model

IA Metrics Class	Measurement Questions	Weight
Policy management	find the extent that the tool is configurable for password, audit, accounts, user rights, security options, ACL settings, IP security, encrypted data recovery assessments	1
Process maturity	is the tool certified under the Common Criteria	1
Personnel support	Determine if the vendor teaches classes, and additionally certify user competence	1
Resource support	determine the cost per class C network	1
Operational practice	see if the tool perform an organizational or domain-based password compliance measurement	2
Operational environment	Determine if the tool allows for the likelihood of disruptions to environment in full feature usage	3
Management readiness	Determine if the tool assess audit trail support and review practices – age of audit log, logging settings	1
Technical readiness	Illuminate the extent of support for vulnerability cataloging (CVE, SANS, CERT, BugTraq, CIAC, FedCIRC, IAVA)	1
Effectiveness	determine the tool's measure of evidence of intrusion or system compromise	3
TTOA features in normal circumstances	analyze and record the vulnerabilities found. unique found in designated target systems. determine quality of solution by examining the steps to solution (patch and non patch solutions)	3
TTOA adversary work factor	Determine how much can non-privileged users do with the tool	1
TTOA survivability	Record what happens when one removes the vulnerability database before or during the scan	1
TTOA risk	find out if published vulnerabilities exist for tool in past year	1
TTOA operational limitations	count the range of targets covered	1

Many worms can be detected by superfluous files or registry keys and VA tools should be able to find these. The IA metrics taxonomy addresses the organizational, personnel, as well as technical component of information assurance. Since all are essential to a solid IA discipline, there is a need to consider impact to each of these components in

IA tool selection. The weights assigned are intended to guide the examination of additional factors.

3.5 Test Environment

The target environment consisted of a full range of systems from Microsoft®, Sun Microsystems, Hewlett-Packard, Red Hat, and Mandrake, plus a Cisco® router, and printers from HP® and Tektronix®. One hundred and one systems comprised the whole environment for the tests. Tool quality is also evident in the details presented, thus, a patched and un-patched system running the operating systems Windows® 2000, Solaris 8, and Red Hat Linux 8 were selected. The environment represents core systems used in most organizations. Note that each system was one version off from the latest series from each represented vendor. This was done to ensure higher vulnerability counts and show greater effects of keeping systems patched. There are many other device types to consider such as routers, switches, printers, and wireless access points. The study considers how many of these types are supported, but none of them are studied in depth. The test network however, contained a router, switch and 3 printers. Vulnerabilities found on these target types are noted in the test results.

3.6 The VA Scanning Tools as Articles of Measure

Having the candidate measures, a set of tools was chosen in order to validate this choice of measures. This research focuses on identifying useful measures that one can employ in choosing the most desirable tool for the environment. The emphasis is on setting expectations for use of the tools and finding measures covering the areas of IA, and measures for the most important performance factors of the tools under consideration. In the category of VASTs, the discovery and effective remediation of vulnerabilities are significant performance factors. Other factors are of less significance, but also must be considered. Four tools were chosen for this research due to the availability of these tools and due to the large number of measures to be taken and validated. Most acquiring organizations will narrow their options to three or four candidates and select the most desirable. Each selecting user will have differing viewpoints and needs to consider. Examples are:

- The number of unique vulnerabilities found. (this accounts for duplicate entries),
- Effectiveness of solution; is a solution given for vulnerabilities that are identified? Is sufficient information provided to apply a solution? Does the solution repair the problem? How much time or how many steps are necessary to apply the solutions presented, based only on information provided?
- Accuracy of vulnerabilities found. What is the false-positive rate? Is there a false-negative rate?
- Installation; is the tool easy to install? Is sufficient information provided for resolving problems encountered during installation?
- Performance, how quickly can the tool obtain results? What network load is imposed? Are any of the tests disruptive or dangerous?

- Costs, what are the tool and maintenance prices? What is the estimated IT overhead price? Calculate solutions per dollar, and vulnerabilities found per dollar.

3.7 Measurement Protocol

Each tool was run in a controlled environment, without other tools executing on the host machine. The network activity was measured with background data rates of about 1Kbit/second. The full set of output was captured in the most technical report offered by the tool. The results were evaluated for accuracy in representing the known vulnerabilities.

Static analysis of the tools involves examination of documentation to learn about features offered, extent of help available, restrictions of the tool, frequency of updates, available reports, size of vulnerability database, and others. The tool's database was queried for several words that indicate the type of vulnerabilities covered and use of words indicating an understanding of the IA discipline. One note is that three of the tools have readily searchable Microsoft® Access databases, and one used scripted plug-ins. The tool's interface is used for searches whenever possible.

CHAPTER IV

VALIDATION

In this chapter, we derive a best-fit model as a predictor of vulnerability assessment scanning tool (VAST) quality and suitability for the acquiring organization given that sufficient consideration is given to the whole IA posture of the organization. As stated earlier, IT staff resources in many organizations are stretched to extent that may not allow full and complete tool testing. The level of familiarity and expertise with vulnerability assessment varies widely. Thus, a model is needed that can quickly indicate reasonable choices of tools. Tool quality can be measured in range of capability testing and single-system focused testing to address the user concerns of breadth, depth, and accuracy. Work by Schneidewind [46] suggests that quality assessment measures and metrics would be most appropriate in tool quality comparisons. Ordinal and categorical measures are reasonable choices for differentiation assessments. The primary goal of this study is to select a minimal set of metrics that when combined, identify distinctions in contending options for VASTs. In this study, we selected four tools: three are commercial offerings and one is from the Open Source community. The tools are mentioned here once by name and afterward by a randomly chosen letter. This study will not endorse any given tool but, rather help to point out a suitable metrics model to assist VAST acquirers in obtaining the best available tool. We use the term 'available' since the organization may be unable to afford the best tool for their purposes.

4.1 Selection of VA Scanning Tools

The tools used in the pursuit of a best-fit metrics based model for VASTs include the eEye Digital Security Corp's Retina® Scanner version 4.9.x [22]. Many releases of this were used during the study period. A second tool from Internet Security Systems is ISS® Scanner version 6.2.1 with several Xpress® Updates installed during the study [27]. It should also be noted that ISS released a version 7.0; however, this version was not available during the study period. The third commercial scanner was that of Harris Corporation. Their STAT® Scanner version 5 was used with several updates during the study period [24]. The Open Source tool used was the Nessus Scanner, versions 2.0.5 and 2.0.7 with many plug-ins added during the study. Nessus also has a Windows® based client, NessusWx version 1.4.4 which was used [36]. This client was chosen since the commercial tools only ran on Microsoft® Windows® operating systems. From here onward in this study, the tools will be known as Tool A – Tool D, in no particular ordering. The anonymity of the tools is preserved from explicit mention. It should be noted that this work was done, in part using the above tools provided by the Harris Corporation STAT® testing laboratory.

Having selected tools, we must also define the stipulations under which the model is employed. The technique discussed in this study can be adjusted to fit any given environment and user community. The presumed VAST acquirer is responsible for proper security functionality of a general purpose IT infrastructure. The largest parts of the infrastructure are the workstation and server collections; however, printers, routers, and wireless access points normally are also present. The acquirer is responsible for all aspects of maintaining the network and therefore exercises care to minimize downtime in handling maintenance. The acquirer is assumed able to install and run the tools without outside assistance, however insufficient time exists for them to know the tools thoroughly. Thus, default tool configurations are employed unless noted during the discussion of measures and metrics. The acquirer will also hand the operation of the tool off to IT staff for continual use and infrastructure maintenance. Reporting to executive management on vulnerability status is also expected of the acquirer of the tool.

Each of the tools was configured to use all vulnerability checks that do not cause harm or disruption to the network. This was done to preserve the integrity of the lab environment and to observe the extent of safe checks in the tools. Efforts were made to level the tools as much as was feasible and to give each tool the optimal environment and opportunity to identify flaws safely. Background bandwidth measurements were taken in 90-second samples with the Ethereal protocol analyzer residing on the scanning tool host machine. Ethereal was chosen due to being available in win32 form for the Windows® based host platform, and in Linux for the open source tool scanning from a Red Hat® GNU Linux™ platform. Ethereal version 0.9.12 and 0.9.13a were used as we attempted to maintain currency and consistency in measurement. Most of the lab environment was switched at 100 M bits/second; however, some 10 M bit/second hubs also existed. The background bandwidth was consistent at .1 M bits/second before and after the tests. Each tool was tested on the same day and against the target machines configured identically. Reboots of the scanning hosts were also done between tool tests.

4.2 Target System Selection and Setup

In order to address as many features of the tools as possible tests were run against a collection of 101 machines representing HP® and TekTronix® printers, Cisco® router, Microsoft® Windows® systems from Windows® 95, 98, Me, NT 3.51, 4, 2000, XP, and Server 2003. Non Microsoft® systems included Sun Microsystems Solaris® version 2.5.1 – 9, Red Hat® GNU Linux™ versions from 6.2 – 9, and Mandrake® GNU Linux™ versions 7.2 – 9.1, and one HP-UX 10.20 system. Many environments also include wireless access points; however, the portion of the lab available for the tests did not include a wireless access point. Scans of the entire network were done with each tool. To assess concerns for accuracy and enable more fine-grained analysis, two systems each of three operating systems were set up. One of the pair was installed in a default configuration off original CD media, while the other was installed off original CD media and all current patches were installed through the day of testing. This test-bed allowed one to see the strengths and weaknesses of each tested product. The fine grain test environment consisted of Microsoft® Windows® 2000 Server systems, one installed in a default configuration, and a second system, containing the latest service packs and patches, along with other recommended settings such as security templates, and advice from the Windows® 2000 benchmark from the CIS organization [12]. In addition to the Windows 2000 systems, Sun Solaris® systems and Red Hat® GNU Linux™ systems were also configured following the steps for the Windows® systems above. The Solaris® and Red Hat® Linux™ systems were configured from default installations and a counterpart system was patched with vendor provided security patches. The Solaris® and Linux™ CIS benchmarks were run on one each these patched target systems. This configuration provided bounds for the candidate tools to illustrate the inventory of detections and solutions available from each tool, and provided an opportunity to determine false-identification rates against systems secured. This created a diverse test set.

4.3 Measurement

All assembled measures, both in the best-fit model and in the at-large set were taken on representative VASTs. Times were recorded from the tool output, available debugging trace logs, and from the system event logs as applicable to each tool. Searches for terms and checks were done through the tool's interface, as well as through the Microsoft® Access 'Find' feature. Three of the tools provided Access formatted databases, while another provided access to data that is not included in the tool's Access database. The same terms are used and are noted in the test descriptions. Wireless coverage was addressed through vulnerability check searches through the vulnerability check inventories. Note that none of the tools remained static during the test period, thus several tests were run on updated versions of the databases. The updates were done within a 15 minute window with random orderings as to which is updated first and last. This was to alleviate last minute; just in time, signature update queries from the tool update sites.

4.4 Analysis

Tools were run against the six designated machines discussed earlier, as well as against the full test lab environment. In devising measures, we decomposed the analysis by considering results obtained from scans of the designated targets and looked for broader capability indicators from looking at the full lab scans.

4.4.1 Depth Analysis

We consider the depth analysis first as this will reveal the number and quality of the vulnerabilities found. This also allows us to study the recommended solutions to determine if sufficient information exists to eliminate the vulnerability. Organizations evaluating the tools are likely to possess wide ranges of familiarity with vulnerabilities; some will need very little solution guidance, while others will need systematic advice.

The first aspect studied was the raw number of vulnerabilities returned on our test machines. We examined the number of these found that the tool identified in its highest risk category. The first consideration of an administrator is to remove the highest risks from the system. When seeing a large number of vulnerabilities one is faced with the task of prioritization. The results of running the tools on the Windows® 2000, Red Hat® Linux 8, and Sun Solaris® 8 systems individually appears below. Following this assessment, we will select a vulnerability from each operating system and assess the solution that the tool prescribes. Table 4.1 below summarizes the single machine scan findings, showing the number of vulnerabilities found, by each respective tool. The split horizontal accompanying cell shows the number of vulnerabilities rated as high using each tool's assessment criteria.

Table 4.1 Vulnerabilities Found and the Portion of Highest Risk

Total/High Vulnerabilities	A		B		C		D	
	Total	High	Total	High	Total	High	Total	High
Default Win 2000	236	13	90	12	67	6	146	76
Patched Win 2000 #	22	4	29	2	14	1	16	1
Default Red Hat 8	88	21	6	0	26	2	4	2
Patched Red Hat 8	10	0	14	0	59	4	14	2
Default Solaris 8	88	22	38	6	94	20	34	14
Patched Solaris 8	10	0	35	3	101	22	38	16

In the table 4.1 we observe some interesting trends in the data from the scan results. The measure shows Tool A finding the most vulnerabilities in default installed and non patched Windows 2000 systems, however Tool D finds a disproportionate number of them rated high; almost 2:3, the other tools all show ratios of 1:6 through 1:10. If the Windows machine is patched and the registry service turned off, the results show Tool B finding the most vulnerabilities, and Tool A indicating more high risks than do the others. The patched Windows 2000 system runs SQL Server 2000 Desktop Engine software, which was not patched; all of the tools identified this lack of patching SQL. The measures show Tool A to be patch oriented as is seen in the un-patched system scan results. Turning off the

Remote Registry showed a radical drop in ability of all tools in the list. Tool A found more vulnerabilities in un-patched vs. patched Solaris® and Linux systems, confirming it to be patch–centric. Tool C seems to excel in Solaris® and GNU Linux™, but also does well in Windows. Tool D would be of mild concern to a fastidious administrator by flagging numerous high's on patched machines.

We will now examine the criteria used by the tools in rating a vulnerability as high. We examine the output in the reports. Tool A provides a legend on its highest-level management report indicating that a high rating is assigned to granting system privileges to a successful attacker. Examining several descriptions in the other tools indicates that Tool B will rate the ability to execute code on a remote system, regardless of context, as high; even if the victim has low privileges. Tool C also rates remote system code execution as high regardless of victim privileges. Tool D adds susceptibility to remote denial of service of a system as a high vulnerability. This expanded definition of high probably explains its greatest number of high ratings in its scan results.

The preceding raises the point that tool suppliers do not have a uniform rating system, but all agree that remote code executions are of high concern. An administrator may want to sort out the consequences with more distinctive ratings. The enterprise or domain context may be a useful guide, or at a minimum, all tool suppliers should define their interpretations for what consequences entail high, medium, and low risk. This study indicates that this is not being done today.

Another concern for users of VAST technology is the possibility of false-positive indications from the tool. The table below indicates a summary of false-positive returns for the tools when run against patched systems. This analysis excludes indications for open ports and statements that a given service (correct or incorrect) was running, and concentrates on incorrect vulnerability renderings for correctly patched software.

Table 4.2 False-positive Results on Patched Systems

Total and High Risk False-positives	A		B		C		D	
	Total	High	Total	High	Total	High	Total	High
Win 2000	0	0	3	1	2	0	0	0
Red Hat 8	0	0	0	0	1	1	6	2
Solaris 8	5	0	2	1	8	6	7	5

Tools that make assessments based on reading of service banners ("banner grabbing") and general tests to see if services are running are susceptible to over generalizing on vulnerabilities. This is done in order to draw attention to patching and adjusting settings on the detected services. The numbers in table 4.2 were also included in table 4.1 earlier. Every false-positive result found would require time for the administrator to make a visit to the machine and examine it for actual settings.

An examination of the nature of false-positive output is also useful to understand areas of strength and weakness in vulnerability collection and analysis. In table 4.2 above, Tool A's results were found for superceded package versions of the Java environment on the system. Tools C, and D generalized on Sendmail vulnerabilities. Sun's Sendmail banner is not similar to the Sendmail.org variant, which was used as the assumption. Tool B was incorrectly assessing Apache services on Sun, and Internet Explorer version 6 service pack 1 on Windows systems. Understanding the areas where false-positives occur can help an acquirer decide if the occurrence of specific types of false-positives is acceptable.

We next examine solution guidance for the tools. The following example uses an SQL 2000 vulnerability reported by three of the tools. The vulnerability occurs in the Microsoft® SQL 2000 Server Resolution Service. This vulnerability was chosen due to the importance of database servers to organizations having them, and because patching of the SQL Server is a complicated process. Most Windows® patches are downloaded, and installed, followed by a system reboot. The problem description and solution texts have been sanitized to remove evidence of the supplying vendor.

Tool A:

> ***Description****: SQL Server 2000 and Microsoft Desktop Engine (MSDE) 2000 introduce the ability to install multiple copies of SQL Server on a single machine and have it appear that the copies are completely separate database servers. These copies, known as instances, run independently of each other. The default instance listens on TCP port 1433. Other instances cannot share this same port and require a port of their own. When a SQL client needs to connect to an additional instance on the SQL Server, it queries the SQL Server Resolution Service (which operates on UDP port 1434), which tells it which port the requested instance is using.*
> *A vulnerability result because a pair of functions offered by the SQL Server Resolution Service (SSRS) contain unchecked buffers. By sending a specially formatted request to UDP 1434 port, it could be possible to overrun the buffers associated with either of the functions. An attacker could cause portions of system memory (the heap in one case, the stack in*

the other) to be overwritten. Overwriting it with random data would likely result in the failure of the SQL Server service; overwriting it with carefully selected data could allow the attacker to run code in the security context of the SQL Server service. The Slammer (Sapphire) SQL worm takes advantage of this vulnerability. The worm sends 376 bytes to UDP port 1434. The worm continuously sends packets to randomly generated IP addresses, attempting to send itself to hosts that are running the Microsoft SQL Server Resolution Service. This causes a denial of service on the host on which it is running, as well as on the hosts to which it is attempting to connect. The worm resides in memory, and not on disk, so it can be eliminated using a system reboot or shutting down the SQL server and restarting it.
***Remediation**: Install the SQL Server 2000 Security Update for Service Pack 2 or SQL Server 2000 Service Pack 3.Using a web browser, navigate to http://support.microsoft.com/default.aspx?scid=kb;en-us;Q316333&sd=tech and download and install the 00.0686_enu_installer.exe file (MS02-061 patch). This will install a new version of sqlservr.exe and ssnetlib.dll. If you have ssnetlib.dll with a version of 2000.80.636.0 or later, you are patched. Note: You must have SQL Server 2000, Service Pack 2 installed. This service pack is available at http://microsoft.com/sql/downloads/2000/sp2.asp and you will need to reboot for the patch to take affect. SQL SERVER 2000 SP3 INSTALL Using a web browser, http://microsoft.com/sql/downloads/2000/sp3.asp and download and install the latest SQL Server 2000 Service Pack. To verify the service pack has been loaded, run the command SELECT @@VERSION from the SQL Server Query Analyzer window. The version should be 8.760 or greater. For more information, see Microsoft Knowledge Base articles Q316333, Q323875 and Microsoft Security Bulletins MS02 039 and MS02-061. Also see Mitre CAN-2002-0649.*

Tool B:

***Description**: Microsoft SQL Server 2000 is vulnerable to multiple vulnerabilities, which are addressed in the patch released with Microsoft Security Bulletin MS02-039. The most serious of these vulnerabilities would allow a remote attacker to execute code on the system*
***Remediation**: Apply the appropriate patch for your system, as listed in Microsoft Security Bulletin MS02-039. See References*

Tool C.

> ***Description***. *The remote host MS SQL server is vulnerable to several overflows which could be exploited by an attacker to gain SYSTEM access on that host. Note that a worm (sapphire) is exploiting this vulnerability in the wild.*
> ***Remediation***:
> *http://www.microsoft.com/technet/security/bulletin/ms02-061.asp*
> *Risk factor : High*
> *CVE : CAN-2002-1137, CAN-2002-1138, CAN-2002-0649, CVE-2002-0650, CAN-2002-1145, CAN-2002-0644, CAN-2002-0645, CAN-2002-0721*
> *BID : 5310, 5311*

Tool D:

> ***Description:*** *There are three security vulnerabilities here. The first two are buffer overflows. By sending a carefully crafted packet to the Resolution Service, an attacker could cause portions of system memory (the heap in one case, the stack in the other) to be overwritten. Overwriting it with carefully selected data could allow the attacker to run arbitary code. The third vulnerability is a remote DoS.*
> ***Remediation:*** *Install Service Pack 3 for SQL Server 2000 from Microsoft.*

Tool C lumped several vulnerabilities in one check, but contained a consistent CVE reference, as did the other tools. The other tools each found additional vulnerabilities in the server and treated them with the same respective levels of explanation.

We also looked at an Apache™ vulnerability on Red Hat® 8 GNU Linux systems. There were actually 3 vulnerable conditions reported by Red Hat.

Tool A:

> ***Description:*** *Multiple vulnerabilities have been discovered within the Apache web server.-ModSSL improperly negotiates for an upgraded SSLCipherSuite resulting in the use of the weaker version of the SSLCipherSuite.-Denial of Service condition can be invoked by client when a proxy ftp connect request is sent to an FTP server that is utilizing an IPv6 address.-A temporary Denial of Service conditions occurs during the handling of accept() errors. CVE's covered by this vulnerability: CAN-2003-0192, CAN-2003-0253, CAN-2003-0254*

Risk: Medium
Remediation: *Before applying this update, make sure all previously released errata relevant to your system have been applied. To update all RPMs for your particular architecture, run: rpm -Fvh [filenames] where [filenames] is a list of the RPMs you wish to upgrade. Only those RPMs which are currently installed will be updated. Those RPMs which are not installed but included in the list will not be updated. Note that you can also use wildcards (*.rpm) if your current directory *only* contains the desired RPMs. Please note that this update is also available via Red Hat Network. Many people find this an easier way to apply updates. To use Red Hat Network, launch the Red Hat Update Agent with the following command: up2date*
This will start an interactive process that will result in the appropriate RPMs being upgraded on your system. If up2date fails to connect to Red Hat Network due to SSL Certificate Errors, you need to install a version of the up2date client with an updated certificate. The latest version of up2date is available from the Red Hat FTP site and may also be downloaded directly from the RHN website: https://rhn.redhat.com/help/latest-up2date.pxt Bug IDs fixed (http://bugzilla.redhat.com/bugzilla for more info):
78019 - mod_ssl won't start up after unclean shutdown
82985 - Content-length miscalculated when using ssi include virtual with wrapped cgi
85022 - Apache fails to start: Cannot create SSLMutex file
97111 - httpd insists on always calculating Content-Length from CGI output.
98545 - Handle errors when starting piped logged processes
ftp://updates.redhat.com/9/en/os/i386/mod_ssl-2.0.40-21.5.i386.rpm

Tool B did not discover this vulnerability on the date tested.

Tool C:

Description: *The remote host appears to be running a version of Apache 2.x which is older than 2.0.47. This version is vulnerable to various flaws which may allow an attacker to disable this service remotely and/or locally.*
Remediation: *Upgrade to version 2.0.47*
See also : http://www.apache.org/dist/httpd/CHANGES_2.0
Risk : Medium
CVE : CAN-2003-0192, CAN-2003-0253, CAN-2003-0254
BID : 8134, 8135, 8137, 8138

Tool D:

> ***Description.*** *A vulnerability was reported in the Apache 2.0 web server. The server may apply a weaker encryption suite than intended.*
> ***Remediation:*** *The vendor has released a fixed version (2.0.47).*
> *(http://httpd.apache.org/download.cgi)*
> *(http://www.apache.org/dist/httpd/Announcement2.html)*
> *Risk: Medium*
> *CVE: CAN-2003-0192*
> *BID: 8134*

From the preceding, we find that Tool A gives the most detailed treatment of the vulnerability and indicates a denial of service is possible. The others indicate information leakage or weaker encryption. Tool A also gives a URL to obtain the corrected patch for a Red Hat® system. All three tools rated the problem a medium risk.

The third major platform within the study was Sun Solaris®. For Solaris® we chose a vulnerability in Sendmail. This application has been a mainstay on the internet for almost 20 years, yet vulnerabilities in it surface yearly. Addressing them should be a required test for any tool vendor. We chose an address parsing vulnerability that would grant root privileges or the privileges of the running sendmail daemon should it be successfully exploited on a vulnerable system.

Tool A

> ***Description:*** *Sendmail - Address Parsing - Solaris 2.6 - 9*
> *Sendmail versions prior to 8.12.8 are susceptible to root compromise. The address parser performs insufficient bounds checking in certain conditions due to a character to integer data type conversion, making it possible for an attacker to take control of the application. The attacker would have control of the length, offset and memory layout specifics, thus a message content based attack stands a chance to succeed.*
> *Risk: High*
> *Test Criteria: 105396-09*
> *Test Criteria: 5.6_x86*

Test Criteria: 105395-09
Test Criteria: 5.6
Test Criteria: 107685-09
Test Criteria: 5.7_x86
Test Criteria: 107684-09
Test Criteria: 5.7
Test Criteria: 110616-09
Test Criteria: 5.8_x86
Test Criteria: 110615-09
Test Criteria: 5.8
Test Criteria: 114137-03
Test Criteria: 5.9_x86
Test Criteria: 113575-04
Test Criteria: 5.9
Test Criteria: SUNWsndmr-11.6.0
Test Criteria: Solaris
***Remediation:** Install the latest sendmail patch for Solaris (AutoFix NOT Available) You should stop sendmail, apply the patch, then start sendmail again. Instructions will follow available patches. Before applying this update, make sure all previously released errata relevant to your system have been applied. Visit the patch finder page: http://sunsolve.sun.com/pub-cgi/show.pl?target=patches/patch-access*
Search for the 6 digit patch number - up to and NOT including the dash (-). This will take you to the latest released patch page, from which you can download the patch (Click on the patch for your Solaris version, then click on the HTTP or FTP download link) to install on your Solaris system. Use 'patchadd' to perform the installation, the command 'man patchadd' will provide further information.
Earliest non-vulnerable patches:
OS Version Patch ID
SPARC Platform
Solaris 2.6 with patch 105395-09 or later
Solaris 7 with patch 107684-09 or later
Solaris 8 with patch 110615-09 or later
Solaris 9 with patch 113575-04 or later

Tool B

***Description**: Sendmail address parser buffer overflow (CAN-2003-0161.) Sendmail is a Mail Transport Agent (MTA) used on many operating systems. Sendmail versions 8.12.8 and prior as well as other Sendmail-based commercial versions (see affected platforms) are vulnerable to a buffer overflow in the prescan() function. The prescan() function breaks down (or tokenizes) components of an email address prior to processing. The logic to prevent overly long strings is flawed, allowing a local attacker to construct a specifically long string that invokes a buffer overflow and executes arbitrary code at the privilege of the sendmail daemon (usually root). It is not certain whether remote compromise is possible using this exploit, although ______ has determined that significant barriers exist for reliable remote compromise.*

***Remediation:** Upgrade to the latest version of Sendmail (8.12.9 or later), or apply the appropriate patch for your system, available from the Sendmail Web site. See References.*

For Red Hat Linux:

Upgrade to the latest version of sendmail (8.11.6-26.72 or later), as listed in Red Hat Security Advisory RHSA-2003:121-06 for more information. See References.

For Red Hat Linux:

Upgrade to the latest sendmail package, as listed below. Refer to Red Hat Security Advisory RHSA-2003:120-07 for more information.

See References.

Red Hat 6.2: 8.11.6-1.62.3 or later
Red Hat 7.0: 8.11.6-25.70 or later
Red Hat 7.1: 8.11.6-25.71 or later
Red Hat 7.2: 8.11.6-25.72 or later
Red Hat 7.3: 8.11.6-25.73 or later
Red Hat 8.0: 8.12.8-5.80 or later
Red Hat 9: 8.12.8-5.90 or later

For Gentoo Linux:

Upgrade to the latest version of sendmail (8.12.9 or later), as listed in Gentoo Security Announcement 200303-27. See References.

For FreeBSD:

Upgrade to the latest version of FreeBSD (4-STABLE or 4.8-RELEASE or the latest security branch dated later than 2003-03-29), as listed in FreeBSD Security Advisory FreeBSD-SA-03:07.sendmail. See References.

--OR--

Apply the appropriate patch for your system, as listed in FreeBSD Security Advisory FreeBSD-SA-03:07.sendmail. See References.
For other distributions:
Contact your vendor for upgrade or patch information.

Tool C

Description. *The remote sendmail server, according to its version number, may be vulnerable to a remote buffer overflow allowing remote users to gain root privileges. Sendmail versions from 5.79 to 8.12.7 are vulnerable.*
Remediation: *Upgrade to Sendmail ver 8.12.8 or greater or if you cannot upgrade, apply patches for 8.10-12 here:*
http://www.sendmail.org/patchcr.html
NOTE: manual patches do not change the version numbers. Vendors who have released patched versions of sendmail may still falsely show vulnerabilty.
**** ______ reports this vulnerability using only *** the banner of the remote SMTP server. Therefore, *** this might be a false positive.*
See http://www.cert.org/advisories/CA-2003-07.html, http://www.kb.cert.org/vuls/id/398025,
Risk: High, CVE : CAN-2002-1337, BID : 6991

Tool D

Description: *Sendmail prescan() address buffer overflow*
Sendmail 8.12.8 and earlier contains a buffer overflow vulnerability in its handling of e-mail addresses that can be precipitated by the use of a special character value. An attacker can exploit this vulnerability to execute arbitrary code in the context of the mail server.
Risk: High
Remediation: *Upgrade to the most current version of Sendmail, or apply the appropriate vendor-supplied patch. Sendmail Consortium home page (http://www.sendmail.org/)*
CERT Advisory CA-2003-12 (http://www.cert.org/advisories/CA-2003-12.html)
CVE: CAN-2003-0161, BID: 7230

From the Solaris analysis, all tools found the vulnerability, and rated it high, since system compromise or loss to an attacker is the most serious setback to an organization. The tools do differ on CVE and BID citations for the vulnerability. Tools A and B give the most attention to the solution advice, however B does not include Solaris solutions even though this appeared on a Solaris system. Tool C warned of potential for false-positives due to use of service banner grabbing to obtain version information. This is an appropriate warning since skilled sendmail administrators can manipulate this banner text to echo arbitrary content. Tools C and D also found this vulnerability on the patched system, thus they produced false-positives. D did this also without warning of it being possible.

4.4.2 Breadth Analysis

. Breadth analysis reveals the ability of the candidate tool to span all systems of concern to the organization. In this analysis, we look at the ability to correctly identify the running systems and resolve the operating system. This measure addresses the ability of the tool to discover all present systems, and correctly identify them. This builds user confidence in the tool's results. We examine the number of vulnerabilities found on the full network of machines; this measure indicates the extent of the tool's ability to recognize vulnerabilities and bring them to the attention of the user. Following this, we examined the extent of coverage of vulnerabilities on peripheral systems such as network attached printers, routers, modems, and wireless access points. This assessment indicates the applicability of the tool beyond desktop and server vulnerability management. For the wireless and modem detection dimension, we counted the number of checks pertaining to wireless networking access points, SSID, and WEP settings. Table 4.3 also includes exposures such as opened ports, accessible network shares, and default accounts. The tools also reported vulnerabilities on the incorrectly guessed operating systems.

Table 4.3 Breadth Coverage of Tools

	Hosts found	Correctly identified hosts	Vulnerabilities Exposures	Printers/Routers/Modem/Wireless
Tool A	101	93	9550	10/1/1/0
Tool B	100	90	7595	48/13/6/6
Tool C	101	89	2776	4/39/3/1
Tool D	101	90	4335	14/4/2/4

Our determination of correctness requires that a tool distinguish between, for example Windows® 98 and NT, or between 'Unix®' and Linux™ or Solaris®. Being able to precisely identify the operating system of the target indicates ability or potential to diagnose problems correctly. To address how the tools offered support in the broader device support aspects, we illustrate with a Simple Network Management Protocol (SNMP) vulnerability. The printers, routers, and wireless devices – in some cases, are managed via SNMP, thus each device has an embedded agent included. Compromising the agent on a router would be a significant intrusion event, and the ability of a tool to cover this is vital. Tool A did not find the default 'Set' community string on the router interface tested.

Tool B.

> ***Description:*** *Snmp Set Guessable Community: SNMP_Set guessed Community Name and changedsystem information (CAN-1999-0516) Private The SNMP community name is guessable, and allows anyone who can guess the name the ability to set new system information. An attacker can use SNMP to obtain valuable information about the system, such as information on network devices and current open connections.*

Remediation: *Disable or remove the SNMP Service if it is not required. If your systems require SNMP, take steps to secure the SNMP community names. To disable or remove the SNMP Service:*
For Windows NT:
1. Open the Network control panel. (From the Start menu, select Settings, Control Panel, Network.)
2. Click the Services tab, and then select the SNMP service.
3. Click Remove, and then click OK to confirm the removal.
For Windows 2000:
1. Open the Control Panel. (From the Start menu, select Settings, Control Panel.)
2. Double-click Add/Remove Programs, and then double-click Add/Remove Windows Components in the left pane to open the Windows Components Wizard.
3. Select Management and Monitoring tools, and then click Details.
4. Clear the Simple Network Management Protocol checkbox, and then click OK to save the settings.
For Unix:
If SNMP is started from the rc script, comment it out as appropriate for your operating system. As an example, to disable SNMP under Solaris 2.6, execute the following commands:

```
# /etc/init.d/init.snmpdx stop
# mv /etc/rc3.d/S76snmpdx /etc/rc3.d/DISABLED_S76snmpdx
```

--OR--
If SNMP is required on your system, secure the SNMP community names. For Unix systems, refer to your SNMP documentation for information on securing SNMP community names. For Windows systems, secure SNMP community names using the Registry Editor and the control panel.
To edit the registry so that only approved users can access the SNMP Community Name:
CAUTION: Use Registry Editor at your own risk. Any change using Registry Editor may cause severe and irreparable damage and may require you to reinstall your operating system. Internet Security Systems cannot guarantee that problems caused by the use of Registry Editor can be solved.
1. Open Registry Editor. From the Windows Start menu, select Run, type regedt32, and click OK.
2. Select the HKLM\System\CurrentControlSet\Services\SNMP\Parameters\ValidCommunities registry key.

3. From the Security menu, select Permissions to display the Registry Key Permissions dialog box.
4. Set the permissions to permit only approved users access.
--AND--
To configure Windows SNMP security settings in the control panel:
1. Open the SNMP Service security settings, using the steps listed below, depending on your version of Windows.
2. Verify that your configuration contains the following security settings:
- At least one Accepted Community Name exists. Empty lists cause SNMP to accept requests from anyone. (This is discussed in Microsoft Knowledge Base Article Q99880. See References.)
- The Accepted Community Names are not default or easily guessed names, such as public.
- The Only Accept SNMP Packets from These Hosts option is selected, and one or more hosts, IP addresses, or IPX addresses are specified.
- Each host and community name in the lists is a valid, authorized destination.
To access the SNMP Service security settings:
For Windows NT:
1. Open the Network control panel. (From the Start menu, select Settings, Control Panel, Network.)
2. Click the Services tab, select the SNMP Service, and then click Properties.
Risk. High

Tool C.

***Description:** snmpwalk could get the open port list with the community name 'public'.*
***Remediation:** None given.*
Risk: Low

Tool D.

***Description**: SNMP Servers: private - SNMP default community name*
Risk Level: Medium
A default community name is enabled in this SNMP service. An attacker can exploit this vulnerability to gather a high degree of sensitive information about the remote device configuration. The information that can be retrieved varies between service configurations. Such information may include: installed software, running processes, installed patches, network configuration and network connections.
***Remediation:** Disable this community name, or password protect use of it.*

UCD-SNMP Home Page (http://ucd-snmp.ucdavis.edu/)
A Simple Network Management Protocol (SNMP) (ftp://ftp.isi.edu/in-notes/rfc1157.txt) CVE: CAN-1999-0517

Examining the above samples indicates that Tool B devotes superior attention to the solution of the Simple Network Management (SNMP) 'Set' community string problem, but does not explicitly include a solution for 3Com or Cisco, both of whom are leaders in SNMP enabled devices like the subject target router. Tool C found some data from the router, and explained that obtaining open ports was possible with the default string; however, no solution was provided. Tool D gave a concise statement of the problem, but lacked providing a solution. Tool A missed this common problem completely. Tool A uniquely allows one to define the correct community string for devices, however the choice was limited to one string, which was not the correct one in this test. This approach of Tool A did not induce guessing or penetration style behavior by attempting numerous community strings, however several well known strings exist and the testing of all well-known strings is reasonable. Tool C does such testing and documents the ones tried in its results.

Printers are in every modern environment, and can be used by an attacker to hide information. We look at how printer vulnerabilities are treated by each vendor and provide a sample of the most serious vulnerability that each tool found on printers in the test lab.

Tool A

Description: *Always keep firmware on HP Jetdirect print servers at the latest revision level. As firmware is revised, performance and security are improved. With older firmware, attackers can obtain sensitive information and gain unauthorized access to the printer. Jetdirect firmware can be upgraded using either Download Manager or HP Web Jetadmin software.*

Both of these applications are automatically able to download the latest firmware images from the Internet.
Risk Medium
CVE-2000-0636
Variable = system.sysDescr.0
Value = HP ETHERNET MULTI-ENVIRONMENT,ROM R.22.01,JETDIRECT,JD95,EEPROM R.22.09,CIDATE 01/17/2002
: http://www.hp.com/cposupport/swindexes/hpjetdirec4628_swen.html
***Remediation:** Install the latest HP Jetdirect printer firmware. (AutoFix NOT Available) Using a web browser, navigate to http://www.hp.com/cposupport/swindexes/hpjetdirec4628_swen.html and download the HP Jetdirect Download Manager to upgrade the HP Jetdirect firmware. For Jetdirect printer firmware A-J, the most recent firmware is x.08.40 (where x is A-J). For Jetdirect printer firmware L-U, the most recent firmware is x.24.08 (where x is L-U). For best security practice, use non-default community strings. Many SNMP agents are configured with "public" or "private" as the default community string when shipped from the factory. These "defaults" should be changed.*

Tool B

***Description:** SNMPv1Discovery: SNMP version 1 detected (CAN-1999-0615)*
Risk High
SNMP (Simple Network Management Protocol) is the primary standard for Internet network management. SNMP services are included in almost e ry operating system, router, switch, cable or DSL modem, and firewall. Various implementations of SNMPv1 are vulnerable to a wide range of attacks. Incorrectly formatted input in SNMP messages can crash the operating systems and devices that use SNMP. These vulnerabilities may be possible to exploit remotely, allowing an attacker to compromise remote systems and devices. SNMP packets containing invalid fields or data lengths can indicate an attack against SNMP.
***Remediation:** SNMP should be heavily filtered at your perimeter to minimize the threat of SNMP-based attacks. If SNMP is not needed in your environment, consider disabling SNMP completely. Contact your vendor for patch and upgrade information. CERT Advisory CA-2002-03 includes details about the vulnerabilities and updates for many SNMP vendors. See References.*

Tool C

> ***Description:*** *SNMP Agent responded as expected with community name: public SNMP Agent responded as expected with community name: internal*
> ***Remediation:*** *not offered*
> *Risk High.*
> *CVE : CAN-1999-0517, CAN-1999-0186, CAN-1999-0254, BID : 177, 7081, 7212, 7317*

Tool D

> *Risk Level: Medium*
> ***Description:*** *It is recommended that you disable anonymous FTP access if it is not needed. Anonymous FTP access can lead to an attacker gaining information about your system that can possibly lead to them gaining access to your system.*
> ***Remediation:*** *Follow your FTP server instructions on how to disable anonymous FTP.*
> *CVE: CAN-1999-0497*

The above examples illustrate that there is a wide range of opinion regarding printer vulnerabilities. Tool A finds firmware issues, Tool B tests for versions 1 and 2 of SNMP supported and calls version 1 a high risk. (SNMP version 1 has many known vulnerabilities). Tool C shows dexterity in attempting less used community names such as 'internal' which elicited a response from the printer. Tool D finds open anonymous ftp servers on the printer. None of the tools in the sample set does a full set of diagnostics, and each tool has specific strengths, with Tool B being the closest to full printer diagnosis according to our measures and judgment.

4.5 Model Validation

To validate the model, we first provide the list of measures used within each category of the IA metrics taxonomy. We select ordinal weights for each category to emphasize that the fundamental purposes of the tool have more weight. Results for the best-fit model measures are then given. The results of the full set of measures are summarized in Table A.18 in the Appendix, along with those for the best-fit model. We then provide rationale for choice of the representative measures in each category by explaining how they address the category's focus. In the at-large set of measures for each category, we include the scores for those metrics chosen in the best-fit model. We will look for correlation of results across all the areas of the IA metrics taxonomy in the model. Table 4.4 summarizes the measures comprising the best-fit model and shows the weights assigned to each category. Note that Table 3.10 provided the questions, and table 4.4 provides measures addressing the questions.

In Table 4.4, the Weight column serves to distinguish between the categories in the extent of influence that each has over the selection of a tool. The range in values is from 1 – 3, with 1 being lesser in influence. The higher weighted categories more directly reflect the purpose and function of the tool. Lower-weighted categories are not ignored, but have lesser bearing on the intended function of the tool.

The policy management category is concerned with the ability to measure compliance to the organization's policies. This requires that the tool have interfaces to allow customized settings for such things as password handling, auditing, and user rights. The list of policy settings in the measure is taken from the Windows® 2000 security policy editor. The measure looks at the extent of settings support in the tools.

Table 4.4 IA Best-fit Metrics

IA Metrics Class	Measure	Weight
Policy management	count of checks for settings of password, audit, accounts, user rights, security options, ACL settings, IP security, encrypted data recovery assessments	1
Process maturity	Common Criteria certified status	1
Personnel support	user skill certifications availability	1
Resource support	steps per patch	1
Operational practice	user password compliance measurement user keeps antivirus updated	2
Operational environment	Existence of DoS causing checks patch deployment tools supported – count. can scans be scheduled – ease/options break ties.	3
Management readiness	audit use detection forensics related checks in database	1
Technical readiness	the extent of support for vulnerability cataloging (CVE, SANS, CERT, BugTraq, CIAC, FedCIRC, IAVA) 1 for each catalog supported), unique CVE's included	1
Effectiveness	the tool's ability to measures evidence of intrusion or system compromise user defined checking count of checks with no solution/workaround offered	3
TTOA features in normal circumstances	vulnerabilities found in patched systems quality of solution by steps to solution (patch and non patch solutions), ease of updating tool	3
TTOA adversary work factor	the tool's installer protects the tool or give advice	1
TTOA survivability	characterize what happens when one removes the vulnerability database before or during the scan	1
TTOA risk	has the tool had reported vulnerabilities on BugTraq in the past year	1
TTOA operational limitations	can the user write and use her/his own checks?.	1

The process maturity category is difficult for a VAST to measure, however we can apply this category to the process maturity of the tool vendor; this maturity would manifest itself in the stability, support, and evolution of the tool over time. This can take the form of

ISO 9000 or 17799, or SEI CMM-I certifications and Common Criteria certifications. The Common Criteria certification is selected as it is relatively new, and most directly relates to Information Assurance.

A component of this certification is an Assurance class to address the software life cycle management, configuration management, quality management, distribution processes to name a few [16]. This certification also results in a document called the Security Target that is posted by the tool's certifying body or evaluation laboratory for public consumption. Vendor process maturity claims are stated in the Security Target, and ascertained in the Certification Report, which is publicly available. Thus, the buyer has a more solid notion of assurance provided by the tool. It should be stated also that Common Criteria or any other vendor process certification does not imply the best tool, but it does stipulate that the vendor's entire product process has been independently examined, documented and rated by a qualified outside source. The Common Criteria certification process also does not fully consider the tool's functions [16], which is left up to other areas of the IA metrics taxonomy.

The personnel support category concerns the impact of and on people in the organization. We chose the availability of user skill certification by the vendor as a measure in this category. User skills in tool use contribute to Information Assurance for the organization. A tool in the hands of a properly trained employee stands to be most effective. VAST tools vary in complexity and features, and the security discipline is sufficiently important to warrant user skill certifications.

The resource support category addresses the impact of the tool on company resources, mainly time and money. This dimension has a large number of potential measures. For illustration, we chose the cost for a Class C scanning license. Class C refers to the size of the network being scanned being limited to a single eight bit octet of cardinality 256, hence the cost of a 250-256 node license is used. Realistically the organization should also count time in solution application, average number of annual maintenance days to get a more solid grasp of resource impact. A tool with efficiently given solutions may save the company staff overtime payments as well. If a tool disrupts system resources, the staff would need to perform scans during non-work time. The larger the network, the greater care must be given to this computation.

The operational practice category considers the localized impact that a user or IT staff member has on the systems in the network. Users are free to tweak browser settings, possibly disable antivirus tools, use USB memory keys, PDA’s and other convenient devices. A VAST should be able to examine these settings. The second operational category is the environment. This is one of the primary functions for VASTs.

The operational environment category considers the ability to repair vulnerabilities found, and to preserve the normal functionality of the environment. Thus, we examine whether the tool contains checks or capabilities that would result in denial of service events besetting services or entire systems. A red team tester would choose tools with active exploitation techniques and denial of service causing tests. We also determine if the tool supports integration with remediation technology. Patch deployment technology addresses the concurrent deployment of corrective patches across IT systems in an entire management domain. The ability to schedule scans is also included here since it would entail off-peak analysis, which maximizes resource usage. A scheduled scan is practical when organizations do not require employees to shut down their systems at the end of the workday. This has indirect support to the Resource Support category as well.

We next looked at the tool support for readiness, both managerial and technical. The management readiness category is concerned with preserving intellectual or physical property and employing regular reviews of this protection. Audit trail reviews would support this to ensure that actions such as configuration changes are traceable to responsible parties. This category also is concerned with conducting risk assessments. This could include the regular and logged use of VAST technology. A measure of the number of checks supporting audit log management supports management readiness. An additional measure is the support that the tool lends to cyber-forensics activities. Forensics is a growing area for organizations and law enforcement as increasing numbers of computer related crimes are being reported. The ability of a tool to detect existence of forensics tools, possibly in use by unauthorized users would allow an organization to prepare for possible insider threat related activities. These measures are done by using the tool's search features and listing vulnerability checks matching forensics related terms such as log, cookie, or common tools used in forensics analysis including the Open Source TCPDump™ and Ethereal™ packet analysis tools.

The technical readiness category relates to the understanding of vulnerabilities in the organization. Understanding and prioritized remediation of vulnerabilities is possible when the tool can equate common descriptive references to a vulnerability to one that is disclosed in the media. Catalogs such as CERT Advisories, CVE catalog, US Department of Homeland Security FedCIRC, SANS Top 20, BugTraq and others assign designators for the most severe vulnerabilities. If an organization employs these catalogs, the tool's supporting them would make determination of technical assurance readiness easier. The CVE and BugTraq catalogs are common in tools, thus this is a minimal criteria. Searches of the tool for vulnerability catalog support beyond these, measures the technical assurance readiness determination potential by a VAST.

The effectiveness category measures examine the degree to which the organization's defenses are performing as desired. Recording intrusions or evidence of intrusion, as well as the preventive dimension of detecting vulnerabilities under active exploitation are included. We selected the ability of the tool to measure the detection of worms and other malware, and whether the tool allows the user to invoke custom defined checks. To address prevention further, we searched the tool's database to find the number of vulnerabilities reported for which no solution is given. We examined the vendor databases for this check and counted the number of solutions that indicated none was available. At a minimum the tool should provide a workaround in all cases.

The measures discussed and taken up to this point address the organization and its ability to maintain a secure environment. The final group of measures are concerned with properties of the tool itself which is referred to as the Technical Target of Assessment (TTOA) within the IA metrics taxonomy. This covers the features in normal and abnormal circumstances as well as risks and limitations of the tools.

The TTOA features in the normal circumstances category covers the intuitive and assumed functions of the tool. We selected the ability to find vulnerabilities in patched systems. Operating system vendors continually produce patches for their software in response to external and internal security research efforts. Thus, patching of systems is a basic IA practice. We considered the ability of a tool to detect vulnerabilities that extend beyond patching. This area is also conducive to a tool generating false-positive results. In addition to vulnerability identification, we subtracted values for false positives generated by tools claiming that fixed systems were still vulnerable. Another measure here would be to identify false-negative scores. False-negative analysis was not done in our study, but should be considered by tool users with substantial intellectual property to protect. We look at the typical number of steps given by the tool to apply solutions as well. A tool that simply instructs the user to go to the vendor to obtain the patch may omit critical details that would render a system residually vulnerable to another condition if specialized knowledge is omitted. A common example of this is in the rebooting or failure to reboot Windows® systems after they are patched. We examined twenty solutions and followed available web links to solutions. Tools that quickly provide the patch, and provide insight into installing it save their owners time in the remediation process.

For the behavior in abnormal circumstances, category we looked at the TTOA adversary work factor measures. This is a measure of the work needed to penetrate a given system. We address the quality of the tool's installer to provide protection for the tool. If this is not done, we examine the installation guide to determine if sufficient information is provided to secure the tool against an underprivileged insider. A higher ordinal priority is assigned to the installer security feature. Thus, this study did not perform penetration studies of the tools, but focused on the tool's ability to resist obvious penetrations. Full penetration analysis evidence should be sought out by tool acquiring organizations.

We also looked at the category of TTOA survivability in the event of a depredatory event in the tool's installation. We studied the tool's handling of the instance where its database was removed. We considered if the tool provides alerts to the user, and whether any functions can be performed without the database. These measures are categorical as we grouped tool behavior into defined behavior levels.

The TTOA risk category examines the potential risk to the tool or to the organization using it. VAST tools are powerful, and contain storehouses of information that are ripe for abuse and misuse. If installation of the tool requires an administrative user, and if the tool connects to a remote system it may be possible for a non-privileged user to obtain elevated privileges or obtain unauthorized access to information. To assess this risk we looked at the number of vulnerabilities of the tool within the past year reported on the SecurityFocus BugTraq site. The adversary work factor category measure of secure installation is a partial countermeasure to abuse of the tool's by incidental or malicious users. A rogue administrator is a risk to any organization. If a tool can mitigate the impact of a rogue, this would be another good risk countermeasure. Such a countermeasure would be the support of the least privilege principal of security through mechanisms such as role based access control.

The final category is TTOA operational limitations. Here we looked at the range of the tool in covering components of the organization's network. Other areas to consider are platform availability, whether any quick or instant remediation is possible from the tool or whether the tool reporting facilities can support the organization's needs. We did not measure this in this study since it varies with organizations. Rather, we looked at the number of reports and format combinations offered, which is an indirect way to gauge report limitations.

The results of tool examination for this model are given below. Table 4.5 shows the results for the best-fit model alongside the summarized results of the at-large collection of measures. The cell entries show the number of measures in which the tool placed with greatest positive distinction. The full set of results is provided in the Appendix as a set of tables. Tools that did not place best in a measure category are omitted from the respective cell. The numbers given are ordinal counts of best-place measures. Ties are broken if possible, and allowed to stand if not broken. For example, an indicator such as (D,7) implies that Tool D was best or tied for best in seven measures within the category. We observe the incidence of the same tool letter appearing in the best-fit and at-large columns in the table below. The weight column shows a comparative priority value assigned to each measure class. Note that tool technical performance is distributed through several categories in this assessment framework. This is not surprising in that the tools have capabilities to assess wide-ranging condition types, and many of these conditions relate to aspects of the organization's IA posture. Table A.18 in the Appendix, presents numerical scoring for the tools in the study, and includes weighted scores as would be obtained by the best-fit model. Table 4.5 shows the number of measures won by each of the tools along the categories in the IA metrics taxonomy.

Table 4.5 Comparison of Best-fit Model with At-large Measures

IA Metrics Class	Best-fit Model	At-large Measures	Weight
Policy management	(B,5), (A,2), (D,1)	(A,2),(B,9),(D,1)	1
Process maturity	(A,1)	(A,2), (B,1)	1
Personnel support	(B,1)	(A,3), (B,4), (D,2)	1
Resource support	(A,1)	(A,2),(B,2),(C,3),(D,3)	1
Operational practice	(A,2)	(A,6), (B,3), (C,1), (D,1)	2
Operational environment	(A,2), (D,1)	(A,8),(B,4),(C,4),(D,7)	3
Management readiness	(A,1),(B,1),(C,1)	(A,9),(B,3),(C,1),(D,3)	1
Technical readiness	(A,1)	(A,2)	1
Effectiveness	(A,2),(B,1)	(A,7),(B,6),(C,2),(D,3)	3
TTOA features in normal circumstance	(A,),(C,1),(D,1)	(A,19),(B,10),(C,5),(D,14)	3
TTOA adversary work factor	(A,1)	(A,1)	1
TTOA survivability	(B,1), (D,1)	(B,1),(D,1)	1
TTOA risks	(A,1),(D,1)	(A,6),(B,4),(C,4),(D,4)	1
TTOA operational limitations	(B,1),(C,1),(D,1)	(B,3),(C,3),(D,4)	1

In Table 4.5, the at-large measures column represents the union of all measures assembled from mass print literature, requirements lists from potential customers of Harris, independent security consultants, and test laboratories as they are mapped to the IA metrics taxonomy categories. The best-fit measures column represents the measures that most typify the intention of the category among those in the at-large measures set. The weight column is as explained earlier for Table 4.4 with higher weights being associated with intended tool functionality. Next, we discuss consistencies and inconsistencies in the results. The results here are considered along with the depth and breadth analyses as gauges of consistency in the measures taken.

4.5.1 Consistencies

We see in table 4.5, that there is strong agreement between the results among the sets of measures chosen as indicators between the best-fit and at-large sets in the categories. Policy management shows that one tool stood out in the best-fit and at-large measure groupings. Process maturity shows that the same tool stood out in the measures of this grouping. Personnel support showed that the tool standing out in the best-fit model also had a share of the best results in the at-large grouping. Operational practice showed that the same tool stood out in both best-fit and at-large groupings. Operational environment showed that the two tools indicated in the best-fit measures also showed as better in the at-large measure grouping. Management readiness showed a consistent ordering of tools between the best-fit and at-large measure groupings. Effectiveness shows that the tool indicated in best-fit also showed strongest in the at-large grouping. The TTOA features in normal circumstances are a large measure grouping as is expected. The best-fit measures here indicate one of the tools that ranked as best in the at-large grouping, and also selected lower placing tools in the at-large grouping. TTOA Risk indicated one tool placing as best in the at-large measure grouping and another that finished in the mid range. In each of these cases, the best-fit measures paralleled the at-large results in illustrating tool attributes.

4.5.2 Inconsistencies

In resource support measurement, we used a single dimensional metric, where as an organization having knowledge of remediation time and frequency and charge rates for staff would devise a complex metric to replace it. This may change the ordering in this category to favor a tool that provides fast and accurate remediation advice. In the technical readiness category, we found two measures, which comprised the set. In TTOA adversary work factor, we had a single measure: more measures are needed, however exploring this further exceeded the scope of this study, and tool vendors do not readily reveal shortcomings. An acquirer of VAST or any other tool types can learn about penetration tests or other tool vulnerability assessments by reading independent lab reports such as by the NSS Group [39], or in the tool certification reports and independent test reports for tools. The survivability measures are related to adversary work factor; our tests were limited due to scope of a serious effort such as tool survivability studies.

From the preceding we note that the best-fit model based on measures of the IA metrics taxonomy quite well represent the performance of a much larger set of measures.

This taxonomy addresses a much broader portion of the IA discipline than have previous reviews and comparisons of vulnerability assessment scanning tools.

Many tool reviews place emphasis on the quantity of vulnerabilities found in a tool and predict it as best based on this. This practice is contrary to Kitchenham, Pfleeger, and Fenton's [29] observation on basing predictive models on empirical data. In our analysis, Tool D often found the most risks, and found them faster, and placed high in many reviews, however applying measurement science and guided by the IA metrics taxonomy [59], we see a different outcome. Examination of the results obtained by using the best-fit model indicates that Tool A scores as being the best choice for an acquirer under the conditions that we outlined in using the model. Note that acquirers with goals in opposition to those used in this study may determine that a different tool is best. Thus, we offer evidence that the best-fit model methodology helps focus attention on the IA posture of an organization and applies holistic discipline to the problem of IA technology assessment toward meeting the goals of the tool user.

CHAPTER V

CONCLUSION

The hypothesis under investigation by this study is that it is possible to quantify IA suitability of vulnerability assessment scanning tools (VAST) in their utility given that a sufficiently discriminating set of measurable attributes is found.

Product comparison efforts have often focused on technical capabilities, with VAST being no exception. Within IA, there are personnel and organizational components accompanying the technical concerns. To choose a best tool for the organization requires the ability to develop measures and metrics that consider personnel, organizational, and technical elements.

5.1 Contributions of this Research

We have used the IA metrics taxonomy of Vaughn, Henning, and Siraj [59] as a framework to organize thinking about extra-technical measurements and their applicability toward VA scanning tool quality assessment. This framework exposed areas of vulnerability assessment scanning tools that have been ignored within prior tool reviews and comparisons. The areas exposed pertain to organizational and personnel performance enhancement within IA. Technical areas pertaining to assurance value and resistance to attack and survivability are also under-rated but considered here.

We applied results in metric selection and construction from software engineering, and reached a conclusions consistent with results stated in [29, 46]. We identified a subset of known and derived metrics to construct a best-fit measures model. Using the subset we evaluated four VA scanning tools to determine of any would stand out. Taking this result, we then compiled measures for the larger set of measures and looked for either validation or contradiction. We discovered that the measures chosen, and metrics derived did indicate strongly that a tool fairing well in the best-fit set of measures would do similarly well in a much larger set of measures. There were some areas where few good measures exist, indicating that more effort needs to be spent to derive measures for these areas. Examples are in the areas of adversary work factor and survivability of a tool, as well as in the ability of the tool to support organizational technical readiness assessment.

Applicability of this technique could serve as a guide in determination of tool suitability for an acquiring organization. The rankings can be changed; based on priorities and the intended user. For example, the needs of an attacking red team would differ from an IT department already having system management privileges. The fundamental concept behind this research was to show that given the groupings afforded by the IA metrics taxonomy, the most important or relevant measure from each grouping can be used as an indicator for the group in assessing a tool's ability to provide the required support to the given grouping. Note that one should always include key performance indicators of the tool category in any assessment. The IA metrics taxonomy provides two categories to consider the operational environment and tool effectiveness dimensions. This work also illustrated that tool rating based on too few organizational priorities and observation of subsets of empirical data may not serve in the acquiring organization's best interests. The research goal of validating the IA metrics taxonomy [59] via this research has also been satisfied. The taxonomy proved sufficiently flexible and well defined to enable placement of existing measures in it. There are some measures that serve more than one category in the taxonomy. For example ease of updates contributes to user skill support, current vulnerability detection, and a strong feature for a tool in normal circumstances. The taxonomy highlighted the contribution that automated updates can make to a tool.

5.2 Future Research

There is much work possible in follow-up to this research. One contribution possible is the proposal of a set of technical capabilities to be included in Common Criteria protection profile option packages [16]. The results can also be used to formulate security functional requirements for inclusion in vulnerability assessment protection profiles. This would contribute to discussion and development of standard capabilities by which tools could be assessed on essential technical merit. Further work may include a tool to score security policies on coverage of the sub classifications in the IA metrics taxonomy. This could be used to help organizations verify that their current policy covers all of their security concerns. It would be helpful to show which clauses of the security policy are violated upon exploitation of vulnerabilities in systems and user programs. Having this understanding will convey to the stakeholder the impact of exploitation of the vulnerability upon his or her security environment. Additionally, this could be constructed and intended to compliment the Smith, Newton Common Criteria policy taxonomy. Other tools could be developed to account for scope of a policy in assessing vulnerabilities; show classes and sub classes that exceed the coverage of the policy. This would illustrate any needed extensions to the taxonomy. Further work in the development of metrics for interpreting the results of scanning tools is possible, with a goal of showing the true effectiveness of a tool at identifying vulnerabilities, providing advice on repairing them, and measuring the actual cost of use –including metrics for learning time, and extraneous information filtering time. To address the dynamic nature of vulnerabilities that become more severe upon publication of an exploit, a severity metric in the form of a severity scale could be submitted for comment to the vulnerability assessment community. An approach that may prove useful in ascertaining the severity of a given vulnerability is in the use of fuzzy cognitive maps (FCM). They have been applied successfully in intrusion detection systems [49] and in development of business performance metrics [28]. The severity metric could be obtained from a vulnerability severity model based on an FCM determining the interrelations of a vulnerability to its environment, discovery, exploitation status, and degree of deployment. Training of the FCM could be conducted using data from common vulnerabilities from the CVE catalog for which all sufficient data relating to the above factors can be found. The CERT/CC organization's vulnerability severity scoring system could be used as a validation of the model's ability to indicate the severity of each vulnerability.

In further work, we could devise metrics that address the value to an organization in repair of vulnerabilities. Studies exist that illustrate the average costs of recovering from damage done by an exploited virus or vulnerability. Tools could be developed to assist stakeholders with estimating the impact of vulnerability assessment tools upon their organization. One method for understanding impact is to realize the cost of security incidents. Some work has been conducted in this area. One example is the "Incident Cost" formula based on the Internet Integrity and Critical Infrastructure Protection Act [19]. In his online document, Dittrich mentions the Committee for Institutional Cooperation (CIC) sponsored Incident Cost Analysis and Modeling Projects [14, 15]. Ideas from these studies could be incorporated into metrics to assess the potential loss from not owning and operating a vulnerability assessment tool. Once baseline cost metrics are established, an organization can regularly assess their security costs to other productivity measures to understand the value of diligent vulnerability assessment, and the returns on investment through reduced system downtime and IT administrator overtime due to security related incidents. Development of a metric to assess vulnerabilities on their relative severity or likelihood of their exploitation against security policy will contribute to this valuation computation.

To address a goal of integrated vulnerability assessment, and intrusion detection and prevention tools, a set of vulnerability primitives could be developed along with a representation algebra, which can be used to encode policy statements into a notation that can be used to build protection rules for active defense applications.

REFERENCES

[1] R. Abbott et al., "Security Analysis and Enhancements of Computer Operating Systems," technical report NBSIR 76-1041, Institute for Computer Science and Technology, National Bureau of Standards, Gaithersburg, Maryland, 1976.

[2] L. Alford, "Cyber-warfare: A New Doctrine and Taxonomy," *CrossTalk*, vol. 14, no. 4, April 2001, pp. 27-30.

[3] M. Andress, "Network Scanners Pinpoint Problems," *NetworkWorld Fusion*, February 4, 2002, http://www.nwfusion.com/reviews/2002/0204bgrev.html (current 14 April 2002).

[4] W. Arbaugh, W. Fitben, and J. McHugh, "Windows of Vulnerability: A Case Study Analysis," *IEEE Computer*, vol. 33, no. 12, December 2000, pp. 52-58.

[5] B. Ashley and G. Jackson, "Information Assurance Through Defense in Depth," *IANewsletter: The Newsletter for Information Assurance Technology Professionals*, Defense Information Systems Agency, vol. 3, no. 2, 1999, pp. 3-6.

[6] T. Aslam, *A Taxonomy of Security Faults in the Unix Operating System*, master's thesis, Department of Computer Science, Purdue University, West Lafayette, Indiana, 1995.

[7] B. Beizer, *Software Testing Techniques*, Van Nostrand, Reinhold, Company, New York, 1983.

[8] R. Bibsey and D. Hollingworth, *Protection Analysis Project Final Report*, technical report, ISI-RR-78-13, Institutue of Information Sciences, University of Southern California, Los Angeles, California, 1978.

[9] M. Bishop, *A Taxonomy of Unix System and Network Vulnerabilities*, technical report CSE-95-10, Department of Computer Science, University of California at Davis, California, 1995.

[10] M. Bishop and D. Bailey, *A Critical Analysis of Vulnerability Taxonomies*, technical report CSE-96-11, Department of Computer Science, University of California at Davis, California, 1996.

[11] S. Bridges and R. Vaughn, "Intrusion Detection via Fuzzy Data Mining," *Proceedings of the 12th Annual Canadian Information Technology Security Symposium*, Ottawa, Canada, June 19-23, 2000, pp. 109-122.

[12] Center for Internet Security, http://www.cisecurity.org/ (current 1 May 2003).

[13] R. Chillerege et al., "Orthogonal Defect Classification: A Concept for in-Process Measurement," *IEEE Transactions on Software Engineering*, vol. 18, no. 11, 1992, pp. 943-956.

[14] "Incident Cost Assessment and Modeling Project I: ICAMP 1 Executive Summary," Committee for Institutional Cooperation, http://www.cic.uiuc.edu/groups/CIC/archive/Report/ICAMP-I.htm (current 19 July 2002).

[15] "Incident Cost Assessment and Modeling Project II: ICAMP II Executive Summary," Committee for Institutional Cooperation, http://www.cic.uiuc.edu/groups/CIC/archive/Report/ICAMP-II.htm (current 19 July 2002).

[16] *ISO 15408 1999: Common Criteria for Information Technology Security Evaluations, version 2.1*, Common Criteria Implementation Management Board, International Organization for Standardization, Zurich, Switzerland, 1999.

[17] "CERT/CC Statistics: 1988-2002," Computer Emergency Response Team Coordination Center (CERT/CC), Carnegie-Mellon University, Pittsburgh, Pennsylvania, http://www.cert.org/stats/cert_stats.html (current 23 June 2002).

[18] "The Cost of Assessing Security," *Computer Wire*, 1 Nov. 2000, http://www.securitywatch.com/fr_icat1.html (current 23 March 2002).

[19] D. Dittrich, "Estimating the Cost of Damages Due to a Security Incident," http://staff.washington.edu/dittrich/misc/faqs/incidentcosts.faq (current 19 July 2002).

[20] W. Du and A. Mathur, *Categorization of Software Errors that Lead to Security Breaches*, COAST technical report 97-09, Department of Computer Sciences, Purdue University, West Lafayette, Indiana, 1997.

[21] "eEye Digital Security," eEye Digital Security, Inc. http://www.eeye.com (current 1 September 2003).

[22] J. Forristal and G. Shipley, "Vulnerability Assessment Scanners," *Network Computing*, January 8, 2001, http://www.networkcomputing.com/1201/1201f1b1.html (current 16 April 2002).

[23] L. Guttman, "A Basis for Scaling Qualitative Data," *American Sociological Review* vol. 9, 1944, pp 139-150.

[24] "STAT: Security Threat Avoidance Technology," Harris Corporation, http://www.stat.harris.com (current 1 September 2003).

[25] R. Henning, ed., *Proceedings of the 1st Workshop on Information Security System Scoring and Ranking: Information System Security Attribute Quantification or Ordering*, Applied Computer Security Associates, Williamsburg, VA, May 21-23, 2001, http://www.acsac.org/measurement/proceedings/wisssr1-proceedings.pdf (current 27 Mar 2002).

[26] "Investigative Research for Information Assurance," Institute for Security Technology Studies, Dartmouth University, Portsmouth, New Hampshire, http://www.ists.dartmouth.edu/IRIA/ (current 7 April 2002).

[27] "Internet Security Systems," Internet Security Systems, Inc. http://www.iss.net/ (current 1 September 2003).

[28] D. Kardaras and G. Mentzas, "Using Fuzzy Cognitive Maps to Model and Analyze Business Performance Assessment," *Advances in Industrial Engineering Applications and Practice II*, Jacob Chen and Anil Mital eds., El Paso, Texas, 1997, pp. 63-68.

[29] B. A. Kitchenham, S. L. Pfleeger, and N. E. Fenton, "Towards a Framework for Software Measurement Validation," *IEEE Transactions on Software Engineering*, vol. 21, no. 12, Dec. 1995, pp. 929--944.

[30] I. Krsul, *Software Vulnerability Analysis*, doctoral dissertation, Department of Computer Sciences, Purdue University, West Lafayette, Indiana, 1998.

[31] C. Landwehr et al., "A Taxonomy of Computer Program Security Flaws," *ACM Computing Surveys*, vol. 26, September 1993, pp. 211-254.

[32] R. Linde, Operating System Penetration," *National Computer Conference Proceedings: AFIPS Conference Proceedings*, vol. 44, 1975, pp. 361-368.

[33] U. Lindqvist, P. Kaijser, and E. Johnsson, "The Remedy Dimension of Vulnerability Analysis," *Proceedings of the 21st National Information Systems Security Conference*, Crystal City, Virginia, 6-9 October, 1998, National Institute for Standards and Technology, Gaithersburg, Maryland, http://csrc.nist.gov/nissc/1998/proceedings/paperD4.pdf (current 27 Mar. 2002).

[34] S. McClure, J. Scambray, and G. Kurtz, *Hacking exposed: Network security secrets and solutions, 3ed*, Osborne/McGraw-Hill, Berkley, California, 2001.

[35] "Common Vulnerabilities and Exposures," The MITRE Corp., http://cve.mitre.org/ (current 2 April, 2002).

[36] "Nessus," Nessus Project, http://www.nessus.org/ (current 1 September 2003).

[37] J. Nielsen, "Heuristic Evaluation," *Usability Inspection Methods*, Jakob Nielsen and Robert L. Mack, eds., John Wiley and Sons, Inc, New York, 1994, pp. 25-62.

[38] K. Novak, "VA Scanners Pinpoint Your Weak Spots," Network Computing, http://www.networkcomputing.com/1412/1412f2.html, 26 June 2003, (current 17 October 2003).

[39] "Vulnerability Assessment," NSS Group, http://www.nss.co.uk/va/va_edition_2.htm#Table (current 16 April 2002).

[40] "OMG's CORBA Website," Object Management Group, http://www.corba.org/ (current 16 July 2002).

[41] J. Portier et al., "Experiments with Computer Software Complexity and Reliability," *Proceedings of the 6th International Conference on Software Engineering*, Tokyo, Japan, 1982, IEEE Press, pp. 94-103.

[42] T. Punter, R. van Solingen, and J. Trienkins, "Software Product Evaluation: Current Status and Future Needs for Customers and Industry," *Proceedings of the 4th Conference on Evaluation of Information Technology*, Delft, Netherlands, October, 1997, pp. 30-31.

[43] Rubey, "Quantitative Aspects of Software Validation," *ACM SIGPLAN Notices*, ACM Press, New York, 1975, vol. SE-10, no. 6, pp. 246-251.

[44] "NSA Glossary of Terms used in Security and Intrusion Detection," SANS, http://www.sans.org/newlook/resources/glossary.htm (current10 November 2001).

[45] "Systems Administration and Network Security," SANS, http://www.sans.org/ (current 10 January 2003).

[46] N. Schneidewind, "Methodology for Validating Software Metrics," *IEEE Transactions on Software Engineering*, vol. 18, no. 5, pp. 410-422, May 1992.

[47] "Vulnerabilities," SecurityFocus, http://securityfocus.org/ (current 10 November 2001).

[48] "View Topics Page," SecurityTracker, http://www.securitytracker.com/topics/topics.html (current 12 November 2001).

[49] A. Siraj, S. Bridges, and R. Vaughn, "Fuzzy Cognitive Maps for Decision Support in Intrusion Detection Systems," *Proceedings of ISFA- NAFIPS-2001*, Vancouver, Canada, July 25 - 28, 2001, http://www.cs.msstate.edu/~security/index2.html (current 23 June 2002).

[50] G. Smith and R. Newton, "A Taxonomy of Organizational Security Policies," *Proceedings of the 23rd National Information Systems Security Conference*, Baltimore, Maryland, Oct. 2000, http://csrc.nist.gov/nissc/2000/proceedings/toc.html (current 24 November 2001).

[51] "Application Note: Testworks Quality Index," Software Research, Inc, http://www.soft.com/AppNotes/TestWorksIndex/index.html (current 23 March 2002).

[52] E. Starrett, "Metrics 101," *Crosstalk: The Defense Software Engineering Journal*, United States Air Force Software Technology Support Center, Ogden, Utah, August, 1998, vol. 11, no. 8, pp. 24-28.

[53] "TPC benchmarks," Transaction Processing Performance Council, http://www.tpc.org/information/benchmarks.asp (current 19 September 2002).

[54] "Measurement Crosstalk Articles," United States Air Force Software Technology Support Center, Ogden, Utah, http://www.stsc.hill.af.mil/metrics/MET_ART.asp (current 23 March 2002).

[55] "Audit Report: DoD Compliance with the Information Assurance Vulnerability Alert Policy," United States Department of Defense, http://www.dodig.osd.mil/audit/reports/fy01/01-013.pdf (current 1 November 2002).

[56] "Scanner Comparison," United States Defense Information Systems Agency, MS Excel features proposal spreadsheet submitted to responding tool vendors, provided by Harris Corporation, 22 March 2002.

[57] "Text Retrieval Conference (TREC)," United States National Institute of Standards and Technology, http://trec.nist.gov/ (current 19 September 2002).

[58] "Defense in Depth: A Practical Strategy for Achieving Information Assurance in Today's Highly-Networked Environments," United States National Security Agency, http://nsa2.www.conxion.com/support/guides/sd-1.pdf (current 18 May 2002).

[59] R. B. Vaughn, Jr., R. Henning, and A. Siraj, "Information Assurance Measures and Metrics – State of Practice and Proposed Taxonomy," *Proceedings of the 36th Annual Hawaii International Conference on System Sciences*, Big Island, Hawaii, January, 2003.

[60] R. Vaughn, A. Siraj, and D. Dampier, Information Security System Rating and Ranking," *Crosstalk: The Journal of Defense Software Engineering*, Ogden Utah, May 2002, vol. 15, no. 5, pp. 30-32.

[61] C. Wang and W. Wulf, "Towards a Framework for Security Measurement," *Proceedings of the 20th National Information System Security Conference*, Baltimore, Maryland, 1997, pp. 522-533.

[62] W. Ware, "Security Controls for Computer Systems: Report of Defense Science Board Task Force on Computer Security," Published for the Office of the Secretary of Defense, R-609-1, Rand Corp, Willis Ware ed., Santa Monica, California, 1970, http://www.rand.org/publications/R/R609.1/R609.1.html (current 18 November 2001).

[63] G. Wedberg, "Pro-Active Metrics," *Crosstalk: The Journal of Defense Software Engineering*, United States Air Force Software Technology Support Center, Ogden Utah, August, 1998, vol. 11, no. 8, pp. 11-3.

[64] E. Weinstein, "Eric Weinstein's World of Mathematics," Wolfram Research, http://mathworld.wolfram.com/ (current 20 November 2002).

[65] L. Zadeh, "Fuzzy Sets," *Information and Control*, MIT Press, Boston, Massachusetts, 1965, vol. 8, no. 3, pp.338-353.

[66] Z. Zhang, "List of Usability Evaluation Methods and Techniques," http://www.cs.umd.edu/~zzj/FramedLi.htm?Question.htm (current 27 May 2002).

[67] Z. Zhang, V. Basili, and B. Schneiderman, "Perspective-Based Usability Inspection: An Empirical Validation of Efficacy," *Empirical Software Engineering: An International Journal*, Kluwer Academic Publishers, New York, March, 1999, vol. 4, no. 1, p. 43.

[68] "Vulnerability Assessment Products," MS Excel spreadsheet market summary created by Paul Zimski, provided by Harris Corporation, 21 July 2001.

A P P E N D I X

TEST RESULTS

A. Test Results

Having used the best-fit model to search for and identify the tool best suited for the IT organization acquiring a VA scanning tool, we present the set of measures taken during the study, noting which of them are used in the best-fit model. The IA metrics taxonomy [59] categories were decomposed into VA scanning tool property measures relating to each category. The measures are presented in a set of tables below. The measures within the best-fit model are identified within the tables below by an 'at' symbol, (@), and in explanatory text following the listed measures for each category. The measures given also contain comments on the manner of measurement, with subjective ordinal ratings of 1- 5 for given capabilities of tools as they were measured. In a quality assessment study, many of the measurements are ordinal rankings, as Schneidewind indicates is reasonable [46]. Generally, a lower integer number when given is the more desirable capability. Measures of quantity are also included and noted to explain the nature of what is being measured. The measures and categories are similar to quality assessment measures as that is the primary goal of the best-fit model discussed within this study.

The best-fit model assigns greater scalar weight to four IA metrics taxonomy categories. These will have a notation of (2x) or (3x) in the summary row located at the bottom of each table. The operational practice category is weighted at 2x since the tool use will constantly affect the operations of its users in some tangible way, but overall this category is less important than actual performance in vulnerability assessment. The operational environment, effectiveness, and TTOA features in normal circumstances categories are assigned a weight of 3x since these categories are concerned with aspects of tool performance and effectiveness; and are directly measuring performance characteristics of the tool.

The measures in this study were taken from various sources including popular reviews in mass print literature, requirements lists from potential customers of Harris, independent security consultants, and test laboratories. Many sources converge on key performance related measures, while other measures were derived from a single source and stand as unique. This set of measures was then mapped into the IA metrics taxonomy along the intentions of each measure as this relates to the nature of a category within the taxonomy. The weightings used for the categories indicate the relative amount of influence of the category over the suitability of the tool. The weights were kept simple, yet attempted to place relative importance and influence on the measures in the model. The set of measures chosen for the best-fit model most directly address the intention of the category, and are repeatable to measure.

The concluding table presents a tally of the categories won by each tool. In many cases two or more tools tied for the best measure. The best-fit model measures are included in this table, and one can compare the best-fit model to the full set of measures. It can be observed that it is not necessary to perform all measures used in this study, if all of the best-fit model measures are taken. The best-fit model measures indicate relative trends and ordering of the tools.

Table A.1 Policy Management

Measure	Comments	Tool A	Tool B	Tool C	Tool D
Public policy	HIPAA		1		
Heavy infrastructure / process control safety	SCADA systems		1		
Public policy	Gramm-Leach-Bliley		1		
Public policy	Sarbanes-Oxley		1		
Security policy @	Password	11	9	9	9
Security policy @	Accounts	13	16	0	4
Security policy @	Audit	21	28	0	0
Security policy @	User rights	28	36	10	3
Security policy @	Security options	9	20	2	3
Security policy @	Encrypted data recovery	0	0	0	0
Security policy @	IP security policies	1	1	0	1
Security policy @	ACL settings				
Wins		**2**	**9**	**0**	1

In public policy matters, only Tool B mentioned this concern on its website at the time of the study. All security policy measures listed in the table above are used in the best-fit model. The counts refer to the number of different elements tested for by each tool. The concern for complying with regulations in the health care (HIPAA), finance (Gramm-Leach-Bliley), corporate accountability (Sarbanes-Oxley), and the public utilities which use supervisory control and data acquisition (SCADA) systems to monitor production of electricity or water treatment processes all will be compelled to protect their IT resources.

Table A.2 Process Maturity

Measure	Comments	Tool A	Tool B	Tool C	Tool D
Vendor tool certification @	Common Criteria certification	1	0	0	0
Tool vendor production process certifications	SEI-CMM, ISO 9000, ISO 17799 Certifications	2	0	0	0
Operations certifications	SysTrust certification	0	1	0	0
Vendor process maturity wins		**2**	**1**	**0**	**0**

The process certification status of the tool vendor is used in the best-fit model, as this is a predictor of the likely overall quality of the tool. The Common Criteria certification process considers aspects of the tool’s high-level design, functional specification, requirements traceability, standard production and delivery techniques as well as the security functions claimed. Systems acquisition beyond scanning and patching may benefit from systems certified as SysTrust compliant.

Table A.3 Personnel Support

Measure	Comments	Tool A	Tool B	Tool C	Tool D
Tool training	User class offered	1	1	0	1
Documentation	Form: 3=online help only, 2=3+web based, 1=2+hard copy	1	1	2	2
User skill certification @	1=Independent (e.g., GIAC), 2=vendor created (e.g. ISS-CA)), 3=completion cert.	3	2	0	3
Trouble support availability	Online FAQ, phone, e-mail, wizard – count 1 for each	3	3	2	3
Personnel support wins		**3**	**4**	**0**	**2**

User skill certification is the most rigorous indicator in the personnel support category, and is used as the best-fit measure for support of personnel. Certifying proficiency with a tool ensures both user and vendor that the tool will be used as intended and to the fullest potential, and minimizes support calls to the vendor.

The resource support category is presented next. This addresses costs to the organization because of choice of a tool, or in performing functions to use, or support the tool. The measures here address the relative ease of performing the basic steps in vulnerability assessment through use of a tool.

Table A.4 Resource Support

Measure	Comments	Tool A	Tool B	Tool C	Tool D
Price	Purchase cost per class C network	$4,370	$10,000	$0	$10,000
Setup	1=Installer, 2=manual/make 3=site visit needed	1	1	1	1
Update	1=Automatic, 2=automated test, with manual get, 3=manual test/get	2	3	3	1
Scan	1=Set range, go 2=setup range/OS/access	2	1	1	1
Patch @	Observed clicks to download link (mode statistic)	1	2	3	2
Wins		**2**	**2**	**3**	**3**

The most frequent operation of those listed is applying patches, since several patches will likely be applied per target machine per scan. Thus, the most impact in the lifespan of the tool comes from ease of applying patches; which saves time; this is the best-fit model measure from this category. For decisions where there is a large gap in acquisition or support costs between contending tools, attention should be paid to estimating the labor time involved in applying patches to targets and weigh this against the purchase price difference.

Table A.5 Operational Practices

Measure	Comments	Tool A	Tool B	Tool C	Tool D
Password strength tests @	1=Yes, no disruption 2=yes locks out accounts, 3=no	1	2	3	2
Checks for screen locking,	1=yes, 2=no	1	2	2	2
Antivirus out dated @	1=Yes, 2=no	1	2	2	2
Antivirus off	1=Yes, 2=no	2	2	2	2
Browser settings	1=Yes 2=related, 0=no	1	1	0	2
USB memory key used	1=Yes, 2=no	1	2	2	2
Modem in use	1=Yes, 2=no	2	1	2	2
Wins (2x)		**6**	**3**	**1**	**1**

Operational practice addresses normal routines and behaviors of users in the environment. Two measures are included within the best-fit model in this category. Password strength assessment and checking for updated anti-virus signature files were chosen due to the persistent vigilance needed to manage both of these concerns. Tools must perform this functionality often, without disrupting work of employees. If active password strength tests were done, a VA scanning tool user would cause all user accounts tested to become locked out. This results from policies to enforce account locking after a set number of consecutive failed login attempts. In high data value environments, the concern for employees carrying data in USB memory devices is likely to surface as well.

Within the operational environment category, presented in Table A.6, we consider measures address the capability of the tool to perform its functions without upsetting the normal operating conditions in the host environment.

Table A.6 Operational Environment

Measure	Comments	Tool A	Tool B	Tool C	Tool D
Port scan flexibility	1=Customizable, 2=all possible, 3=well-known only	1	2	1	1
Port related vulnerability presentation	1=remediation is given, 2=details of port, 3=simple listing	3	1	1	1
Latent vulnerability detection (non running software)	1=Solutions given, 2=detected only, 0=none.	1	0	0	0
Network discovery	Devices found	101	101	101	101
Discovery correctness	OS detection percentage	93	92	90	91
Device discovery diversity	Unique device types found	17	16	13	17
Network map presentation	1=Topology given, 2=chart, 3=text listing	3	3	3	2
Tool potential to disrupt environment @	Denial of service tests: 1=none, 2=exist	1	2	2	2
Presence of disruptive checks	# DoS checks-ordinal rank, (0 is highest score)	0	162	129	55
Disabled disruptive checks	DoS disabled/default 1=yes, 0=not applicable	0	1	1	1
Tool architecture	1=Agents, 2=remote control console, 3=single GUI	3	3	2	3
Integration with remediation products @	Remediation product interoperability count	2.	1	0	1.
Database analysis tool integration	Database scanner		1		
Intrusion prevention availability	1=yes available, 2=not available	1	1	2	1
Scheduling of future scans @	1=built-in, 2=tool instructions, 0=none	2	2	2	1
Wins (3x)		**8**	**4**	**4**	**7**

Table A.7 Management Readiness

Measure	Comments	Tool A	Tool B	Tool C	Tool D
Confidentiality, encryption)	Related checks	84	58	5	15
Integrity	Related checks	4	3	1	0
Availability, denial of service	Related checks	541	154	26	105
Authentication	Related checks	103	79	26	35
Access Control permission	Related checks	184	75	28	45
Non Repudiation	Related checks	0	0	0	1
Accountability, audit, logging	Related checks	109	26	3	23
Records		2530	1224	1010	1265
Audit configuration analysis @	1=customizable, 2=groupings select, 3=predefined	2,1	2,1	2,3	2,3
Reporting (flexibility, diversity, ease of use)	1=Custom; 2=selectable, 3=Fixed	2	2	3	2
Forensics support (login log, changes)	# Related checks or policies/configurations	90	8	110	44
Vendor incident handling	2=Email only, 1=Email+Phone	1	1	2	1
IA Term Frequency	Sum of related checks / all checks	0.405138	0.322712	0.088118	0.177075
Wins		**9**	**3**	**1**	**3**

The best-fit model measures here are the potential of the tool to upset the production environment. Active checks that cause denial of service are of concern. The support of patch deployment tools is another large issue, thus it is included. A third issue is the analysis of the environment while workers are away, such as weekends or overnight is important, this can be done if scheduling of scans is easier.

Audit reviews should happen regularly, thus this capability is included in the best-fit model. Forensics support will also prove to be valuable when an organization needs it.

Table A.8 Technical Readiness

Measure	Comments	Tool A	Tool B	Tool C	Tool D
CVE citations @	Unique CVE entries	1389	481	1050	823
Vulnerability checks	Total checks	2500	1275	1800	1600
Other catalogs	1 for each beyond BID, CVE (CERT, FedCIRC, SANS, CIAC, IAVA (4)	8	3	1	2
Catalog Rank Wins		**2**	**0**	**0**	**0**

The total checks in the database count above is not considered a competitive measure, since vendors create checks differently There is not consistent or standardized method among vendors for creating vulnerability checks. Some checks are for a single vulnerability on a single OS version, while others include multiple vulnerabilities on a single OS version, or on multiple versions. The unique CVE citation is a valid measure since CVE entries describe a single unique vulnerability. Note also that the vendor’s association of CVE names for each vulnerability was not examined for correctness. Close examination of the CVE catalog; including searches for keywords reveals that subtle differences distinguish between vulnerabilities.

Table A.9 Effectiveness

Measure	Comments	Tool A	Tool B	Tool C	Tool D
Intrusion evidence @	Count of backdoor/brute force/ports/sniffers/shares	3	5	4	4
Malicious code detection	Count of Trojans/worms/backdoors malware/trojan/backdoor/worm	85	88	7	22
Mis-detection	1=No false-positive / false negative results, 2=some false-positives found	2	2	2	2
Quick repair capability	1=Single click fixable solutions exist, 2=not so	1	2	2	2
Verification of remediation @	1=Instant remediation-retest 2=rescan 3=additional tools needed	1	2	2	3
Reversal of quick fix remediation	1=Remediation-undo capability, 2=lacing undo	1	2	2	2
Severity level is explained by the tool	1=In some reports, 2=no	1	1	2	2
Remediation process management	1= Provides process management / remediation oversight, 2=does not	2	1	2	2
Customizable/extensible vulnerability checks	1=User supplied data, 2=user groupings of checks possible, 3=pre-defined, 4=user script writing, 5=automated tool heuristic scanning (attack methods)	2,3	1,2,3	2,3,4	1,2,3,5
Solution detail/implementation	1=Workarounds given 2=not given	1	1	1	1
Problem without solution @	# checks with no-fix given	0	87 1232	155 1010	22 1275
Wins (3x)		**7**	**6**	**2**	**3**

The effectiveness category presented next addresses the impact of the tool within the organization. This is observed in the capability of the tool to help in assessing a suspected intrusion by a malicious software agent, or by isolating extraneous open ports or running services. The ease of applying solutions is addressed here, as is verification of repairs and reversal of a repair. Reversals are necessary when the patch induces an unacceptable degradation in a needed service, or if the patch is found to be otherwise defective, such as incomplete download. Additionally, there is little value in a tool with no solutions to problems found. The measures in best-fit model within this category are the detection of intrusion evidence, verification of remediation, and the extent to which solutions are provided for vulnerabilities.

Table A.10 Effectiveness Case Study

Measure	Comments	Tool A	Tool B	Tool C	Tool D
RPC DCOM Case					
MS03-026 detection available	Days past original ms03-026 disclosure	2	14	9	1
Exploit posted +14 days					
MSBlaster released +25 days					
Nachi/Welchia released +33 days					
Nachi/Welchia detection	Days past original mso3-026 disclosure	37	40+	40+	39
	Responsiveness Rank	1	3	2	1
MS03-039 Released	MS03-039 Check available	1	1	7	1
MS03-039 exploit published					
MS03-039 exploit detection	Days past mso3-039 release	1	1	2	1
Wins		**4**	**2**	**0**	**4**

Another indicator of effectiveness is to observe how the tool vendors respond to announced vulnerabilities. During this study, the Microsoft® Remote Procedure Call (RPC) vulnerability was disclosed, and patches were released by Microsoft. Table A.10

presents observed availability in days past the announcements of the vulnerability. There was a follow-on vulnerability discovered in the patch for the initial vulnerability. This case shows the ability of the tool vendors to respond in a real world example. None of these measures are included in the best-fit model, however the data is supplied as further measures of an organizations performance at meeting vulnerabilities and exploits in real life. Note also that past performance does not automatically assure that of the future.

To this point in the presentation of measures, the emphasis has been on the organization. The IA metrics taxonomy also considers properties about the tool itself referred to as the technical target of assessment. The remaining measures given examine the properties of the tool itself relating to static and dynamic properties. Included in this section are the results from using each of the tools against a collection of target machines of well-known configuration. An evaluating organization is likely to have good familiarity with the configurations of its machines.

The TTOA features in normal circumstances category is split into two tables for readability. Following the two parts is a table of results from scanning well-known and reproducible target systems. The features in normal circumstances category is assigned the highest weighting in the best-fit model, along with effectiveness. From this category, the measures included in the best-fit model include the number of vulnerabilities found in patched systems, the relative ease in which solutions can be applied, and the ease at which the tool can be updated. The first of these, vulnerabilities in patched systems considers aspects of system management. The ability to find solutions fast and apply them fast is useful to anyone having large numbers of systems to administer.

Table A.11 TTOA Features in Normal Circumstances Part I

Measure	Comments	Tool A	Tool B	Tool C	Tool D
Solution accessibility @	Steps to download patches	1	2	2	2
Patch preparation is given	Advice/prerequisites for patches given	1	1	0	0
Additional reference data	Background links	8	2	1	2
Speed of scan	Scan time – all safe checks	1:15	57:00:00	1:45	1:10
Unique feature	Orange book C2 checks	1	0	0	0
Attack modes	1=Yes, 2=no	2	1	1	1
Scan trend assessment	1=Yes, 2=no	1	1	2	2
Ease of use	Clicks, startup to scanning of 1 new host worst and best	13 1	9 3	16 10	3 1
Task Atomicity	Only port scan	1	1	1	1
Current security policy compliance	1=Editable, 2=predefined, 3=not supported	1	2	3	2
User access to system utilities	System tool access from within GUI	7	2	0	7
Command line / automation	CLI (API)	1	1	0	0
Integrates with data management	1=Yes, 2.=partial, 3=no	2	2	3	2
Database server configuration analysis	Published tools support SQL	0	1	1	1
Provides data mining	1=Yes, 2=no	2	2	2	2
Update mechanism @	1=Automated, 2-2=checks only, 3=manual means required	2	3	3	1
Installation	1=Installer, 2=Unix style make/config	1	1	1	1
Wins (3x)		**9**	**5**	**2**	**8**

The ease of updating the tool is important as this can ensure that the most recent set of checks is included, and this can be installed with little effort from the user. Organizations applying this model may also see priority in additional measures being included from this category.

Table A.12 TTOA Features in Normal Circumstances Part II

Measure	Comments	Tool A	Tool B	Tool C	Tool D
Product knowledge base	1=Publicly available, 2=private	2	1	1	1
Additional background sources provided	CVE/BugTraq ID are the baseline. Count of others (CERT,FedCIRC,CIAC,SANS)	5	1	0	2
Scalability	1=single copy/host	1	1	1	1
Deployable scanning	# distributed	0	many	0	many
Control console	1=Controling, 2=receive only	2	1	2	1
Policy distribution	1=Agents, 2=message, 0=none	0	0	0	0
Scan progress indicator	1=Active, 2=start/stop only, 3=no indication	1	2	1	1
Progress/debug logging	1=Event logs+text 2=one of these, 3=none	1	2	2	2
Audit trail support - Windows Event Logs etc.	1=Windows auditing used inside tool, 0=not used, rely on simple text logs.	1	0	0	0
Usability – scan configuration	Scan configuration change - # steps	4	3	7	3
Usability – scan execution	Scan # steps	6	4	2	3
Usability – repair clarity	Repair # steps	1-3	4	4	4
Report format * data format combinations	Reports * output forms	1258	87	3	3
Internationalization	Reports to foreign languages	0	1	0	0
Scanning process explained or facilitated	1=Wizard, 2=user guide, 3=none	2	2	3	1
Incorrect GUI functions	# malfunctions	0	0	1	1
Explanation of what was broken	Defective features found			pdf broken	miner not obv
Tool interface tool bar	# Icons	26	9	16	13
Tool interface menu bar	# Menus	13	9	5	6
Tool interface menus	# Picks	68	58	37	42
Wins (3x)		**10**	**5**	**3**	**6**

The use of attack techniques is listed here, but not included in scoring since users of VAST's have both attack and non-attack orientations. The patched system vulnerability count and the discussion on solution characteristics are given in Chapter IV of this study.

The features in normal circumstances category is the largest one for which measures are devised. Users acquiring IA tools have the most help from literature and peers, and requirements in devising measures that assess desired characteristics. Most of the product evaluation studies dwell on measures fitting this category. Finding vulnerabilities in fully patched systems is a best-fit model measure. Other measures are the ease of applying VA tool updates, and the ease of obtaining prescribed patches for vulnerabilities. To indicate the important performance properties of VA scanning tools, the Table A.13 has been provided.

Table A.13 shows the specific test timed in seconds, below this is a row for the vulnerabilities found by each tool. A third row shows bandwidth measured on the scanning machine as it assesses the target machine. The fourth and last row shows the packet count observed during the analysis. Below these six groupings of four rows are other measures taken for the tools. Memory used at startup of the tool, and during the scan are listed, as measured with the Task Manager in Windows®, and the *vmstat* utility in Linux. Other measures include the load introduced to the CPU during the scan, measured using Task Manager, and *vmstat* on respective operating systems. The measure 'Can give up on problem nodes' indicates the ability for the user to instruct the tool to not waste time on a non responsive address. Inactive addresses, and blocked ports are examples of such cases. The ping time measure examines the default ping time to wait between attempts. A setting of 3,000 milliseconds (3 seconds) is most common. The four ranking rows concluding the table present summaries of the comparative performance of each tool within the given measurement category. The raw simple measures based ranking considers the averaged ranking across the four measurement areas. There are many ways to deem the best tool. Efficiently finding vulnerabilities is the intention of the measurement ranking in this table. A tool may find the most vulnerabilities, but impose heavier network demands, or it may run faster than others, but miss some vulnerabilities. We measured the bandwidth used by a tool and packets sent during its scanning session. The vulnerabilities found per packet sent or per byte sent was not found to be a valuable measure unless the measurement can be made with identical lists of vulnerabilities on the same target. Memory used per node scanned can be an effective predictor of the node scanning capacity of a tool. This may indicate the number of licenses of a tool one needs to purchase given that the scanning host system has a known memory size. We did not have enough nodes at our disposal to push the tools to the point of space management performance. It may be valuable to stress the tool on a full disk in order to see how the tool handles deterioration in the environment. The raw simple measures ranking calculates the placement of the tool among the others in the study within the speed, bandwidth, vulnerabilities found, and memory size measures at the bottom of Table A.13. This set of measures indicates that users must prioritize the trade-offs between speed, accuracy, availability and clarity of solutions, support for integration with other tools in the enterprise among other factors.

Table A.13 TTOA Scan Performance

Test	Tool A	Tool B	Tool C	Tool D
full safe scan-of un-patched Windows 2000	146 secs.	717 secs.	714 secs.	63.50 secs.
Vulnerabilities found	236	90	67	146
bandwidth, bps	3,456,296	50,480	21,224	1,360,493
packets,	97,125	21,160	10,952	18,478
full safe scan Windows 2000 Service Pack 4 #	177 secs.	1,711 secs.	677 secs.	68 secs.
Vulnerabilities found #, @	22	29	14	16
bandwidth, bps #,	3,835,880	104,472	24,936	1,676,088
packets #,	97,125	18,000	8,500	9,400
full safe scan RedHat 8 un-patched	115 secs.	1,235 secs.	280 secs.	128 secs.
Vulnerabilities found,	88	6	26	4
bandwidth, bps	175,128	54,256	17,228	17,984
packets,	10,697	133,989	5,346	4,466
full safe scan single RedHat 8 fully patched	78 secs.	1,318 secs.	278 secs.	67 secs.
Vulnerabilities found, @	10	14	59	14
bandwidth, bps	196,264	18,648	29,704	189,480
packets,	9,388	32,966	7,505	11,270
full safe scan single Solaris 8 un-patched	75 secs.	2,274 secs.	250 secs.	60 secs.
Vulnerabilities found,	88	38	94	34
bandwidth, bps	92,176	51,200	18,016	44,032
packets,	7,935	23,348	4,883.	4,816
full safe scan single Solaris 8 fully patched	71 secs.	1,941 secs.	323 secs.	125 secs.
Vulnerabilities found, @	10	35	101	38
bandwidth, bps	82,088	20,576	32,040	115,040
packets,	4,151	70,553	9,204	15,037
Memory at startup MB	13.60	56	23	22.60
Memory used / node scanned MB	2.50	2	88	1.20
Memory OS +login MB	102	102	65-157	102
Average CPU load %	15	3	4	19
Tool installation size MB	90	250	44	27
Subnet scan time in minutes	185	131	360	57
execution threads	64	128	256	20
Can give up on problematic nodes	1	0	0	0
set ping time (ms)	1,500	3,000	3,000	3,000
Vulnerabilities found ranking	**1**	**4**	**2**	**3**
Patched vulnerabilities found ranking	**4**	**2**	**1**	**3**
Scan Time ranking	**2**	**4**	**3**	**1**
Bandwidth ranking	**4**	**3**	**1**	**2**
Memory usage ranking	**2**	**3**	**4**	**1**
Raw simple measure-based Ranking	**2**	**4**	**3**	**1**

Table A.14 TTOA Adversary Work Factor

Measure	Comments	Tool A	Tool B	Tool C	Tool D
Remove DB, What happens? @	1. Error dialogs, limited functions; 2. Error dialogs no function; 3. No indications	1	2	3	2
Wins	**If the database is missing, how useful is this tool?**	**1**	**0**	**0**	**0**

Databases are at the heart of VAST tools since scan results are preserved there for reporting and analysis. When the signatures were missing, none of the tools did well though port scanning was possible. All tools notified the user of a problem with the missing databases. This being the only measure present is also included in the best-fit model.

Table A.15 TTOA Survivability

Measure	Comments	Tool A	Tool B	Tool C	Tool D
Resistance to corruption of executables, configuration files, database,) @	1=Set during install, 2=instructions provided, 3=not set	2	1	3	1
Wins		**0**	**1**	**0**	**1**

The survivability category differs from adversary work factor. Survivability assesses the anticipation of adversarial activities against the tool, and assesses how well the tool is prepared, or helps the user prepare to protect it in the event of attack. This is the sole measure in this category and is used in the best-fit model.

Table A.16 TTOA Risks

Measure	Comments	Tool A	Tool B	Tool C	Tool D
Published vulnerabilities in BugTraq, prior 12 months @	1=No, 2=yes,	1	2	2	1
Vendor vulnerability response	1=Contact information exists, 2=no	1	1	1	1
Scanning disrupts network	1=No, 2=denial potential, 3=denial by default	1	2	2	2
Scans that start, complete	1=Yes,2=noo	1	1	1	1
Inter-component security	1=Encrypted, 2=Cleartext	1	1	1	1
Vendor is a known entity	1=Public, 2=Private, 3=Non profit	1	1	3	2
Evaluation copy available	1=Full function, 2=limited nodes	2	2	1	2
Wins		**6**	**4**	**4**	**4**

There are two measures here that best represent this category, the inter component security between the VA scanner, and any results or control console, ruling out any that are not protecting the connection. The best-fit model employs the existence of published vulnerabilities against the tool in the past 12 months. The SecurityFocus BugTraq site is used here since they track a high number of vulnerabilities. The CERT vulnerability notes database should be searched also when making this measurement.

Table A.17 TTOA Operational Limitations

Measure	Comments	Tool A	Tool B	Tool C	Tool D
Target device support	1=Non specific, 2=vendor specific	2	1	1	1
Alerts when analysis completes	1=External, 2=e-mail, 3=tool dialog messages	2	3	3	1
Host system prerequisites	Clearly stated – list here	NTSP3, MDAC	NT,MDAC	NT,MDAC	NT, MDAC
Customizable checks, configurations, @	1=User definable, 2=user configurable, 3=user selectable, 4=defaults only.	3.	1.2	1.2	1.2
Access levels on target systems needed	1=None, 2=administrator partial, 3=administrator only	3	1,2,3	1,2,3	1,2,3
Access level on host system needed	1=None, 2=administrator partial, 3=administrator only	2,3	3	1,3	2,3
Wins		**0**	**3**	**3**	**4**

The limitations of a tool are not readily apparent when reading literature from the vendors in most cases. Limitations should be identified to determine their acceptability in the environment. The first row on device support determines whether the tool claims to specialize or focus on a subset of all systems. Other tools are more general, and will often present solutions for only common systems or problems. The host system prerequisites for the tools are often similar, and bear examination for anything unreasonable within the environment. The customizable checks measure considers whether checks are extensible by the user, parametrically edited by the user, selected from the full set of checks, or whether no customization exists. The access level measure determines the level of privilege needed to use the tool. The tool should warn the user if insufficient privilege exists to run it. However, if some functionality can be done with less privilege, the vendor should document this. This is an example of the Least Privilege Principle of security. The best-fit measure here is the customizable check writing and usage capability. The trend is toward shorter times between vulnerability disclosure and exploitations against it. The user must be able to react to this, and the tool must allow for immediate reaction to new exploitations.

Table A.18 Summary

Points	Tool A	Tool B	Tool C	Tool D
All measures raw wins	71	52	24	46
Weighted wins via best-fit model	137	91	45	88
Best-fit measures points	14	11	2	5
Weighted best-fit model result	24	15	4	11

Table A.18 above presents the results of the best-fit measures model as applied to the set of measures taken during the evaluation of the tools. The all measures raw wins shows the number of wins, outright and ties for each tool. The weighted wins applies the weightings defined for the categories of greater importance to all measures. This weighting can be used to begin to discern suitability of a tool if the raw scores are close. The bottom two rows isolate the measures used in the best-fit model, and apply them in a raw count, and a weighted count, which is shown as the best-fit model result. We observe the best-fit model accurately tracks with the adjusted raw wins count for the sampling of measures and tools in this study, indicating that Tool A would be preferred.