Quantitative Metrics for Network Security Evaluation

by

Fadi El-Hassan

A Thesis submitted to

the Faculty of Graduate Studies and Research

in partial fulfilment of

the requirements for the degree of

Master of Applied Science

Ottawa-Carleton Institute for Electrical and Computer Engineering

Department of Systems and Computer Engineering

Carleton University

Ottawa, Ontario, Canada

December 2006

Your file Votre référence
ISBN: 978-0-494-23335-1
Our file Notre référence
ISBN: 978-0-494-23335-1

Abstract

Quantitative security metrics have become an important research topic recently. In this thesis, a Hierarchical Quantitative Metrics (HQM) model is proposed and applied to enable the representation of important aspects of network security using quantitative metrics.

Since Intrusion Detection Systems (IDSs) play a vital role in protecting networks, the well-known Snort IDS is utilized to explore methods to compress the large amount of IDS generated information into a few but meaningful metrics. Based on three different ways of categorizing Snort alerts, three different sets of metrics are extracted based on alerts' priority, protocol type, and attack classification. Then, the effect of selecting different intrusion metrics on the evaluation approach is experimentally analyzed. A prototype security evaluator is implemented to combine selected IDS metrics and deliver an overall intrusion metric which serves as an external threat indicator. All experiments are conducted using real network traffic traces from the WIDE network.

To my dear parents Taleb and Hind

to whom I am very grateful

and of whom I am very proud.

To my great wife Inaam

who endured the considerable busy times I had.

To my little two-year-and-half daughter Meera

who always brings wide smiles to my face.

Acknowledgments

I would like to deeply thank my supervisor Dr. Ashraf Matrawy for his great and continuous support throughout this research. I highly appreciate as well Solana Networks team (Mr. Nabil Seddigh and Dr. Biswajit Nandy) for their invaluable ideas and constructive comments.

In addition, the role of Solana Networks, Communications and Information Technology Ontario (CITO), and Natural Sciences and Engineering Research Council (NSERC) of Canada in funding this research facilitated greatly my work and made it possible.

I would like to thankfully acknowledge that at Carleton University, the Department of Systems and Computer Engineering and the Faculty of Graduate Studies and Research supported me in many circumstances, including the Dr. Roger Kaye Memorial Award and the Graduate Student Research Bursary.

Finally, I would like to thank sincerely my wife Inaam for her patience, encouragement, and distinguished support.

Table of Contents

List of Tables

List of Figures

List of Acronyms

AHP	Analytic Hierarchy Process
ALE	Annual Loss Expectancy
ANP	Analytic Network Process
arachnids	Advanced Reference Archive of Current Heuristics for Network IDSs
ARO	Annual Rate of Occurrence
BAR	Business-Adjusted Risk
CAP	Composed Assurance Package
CAR	Committed Access Rate
CC	Common Criteria
CFI	Composite Financial Index
CMM	Capability Maturity Model
cve	Common Vulnerabilities and Exposures
DDoS	Distributed Denial-of-Service
DF	Don't Fragment
DJIA	Dow Jones Industrial Average
DoS	Denial-of-Service

EAL	Evaluation Assurance Level
EF	Exposure Factor
ETA	Event Tree Analysis
FCM	Factor Criteria Metric
FTA	Fault Tree Analysis
GQM	Goal-Question-Metric
GRE	Generic Routing Encapsulation
HQM	Hierarchical Quantitative Metrics
IA	Information Assurance
IAM	Information Assurance Metric
ICMP	Internet Control Message Protocol
icode	ICMP Code Number
IDM	Intrusion Detection Metric
IDS	Intrusion Detection System
IDSs	Intrusion Detection Systems
IP	Internet Protocol
IRR	Internal Rate of Return
IT	Information Technology
itype	ICMP Type Number
msg	Message
MTU	Maximum Transfer Unit

PA	Process Area
PKI	Public Key Infrastructure
PP	Protection Profile
QoS	Quality of Service
rev	Revision Number
ROI	Return On Investment
ROSI	Return On Security Investment
RPC	Remote Procedure Call
RST	Reset
sid	Snort ID
SLE	Single Loss Expectancy
SSE-CMM	Systems Security Engineering Capability Maturity Model
ST	Security Target
TCP	Transport Control Protocol
TOE	Target Of Evaluation
TTL	Time To Live
UDP	User Datagram Protocol
VI	Vulnerability Index
VLAN	Virtual Local Area Network
WIDE	Widely Integrated Distributed Environment

Chapter 1

Introduction

There is pressing need to be able to represent the status of network security (and information security in general) using metrics, preferably quantitative metrics. Effective quantitative evaluation of the security status for computer networks and information systems relies on many factors. In order to select security metrics that can contribute in the evaluation process, some questions arise such as: Are those metrics meaningful? measurable? reproducible? Are they standard for all enterprises' networks or do depend on the specific objectives of each enterprise? Are those metrics objective or subjective? What weights should be assigned to each metric? All these questions are valid. At the present time the body of research is not comprehensive enough to suggest strong answers to the above questions. There remains much uncertainty in regards to viable approaches to producing quantitative security metrics. This uncertainty can be mitigated if more real-world experiments are conducted in the area of security metrics. In other words, apart from proposing

theoretical models and frameworks, researchers may find better answers after they experimentally study metrics and analyze their effect on network security evaluation.

1.1 The Need for Security Metrics

Network and Information security attacks are constantly growing and threatening communications and information production. The threat can be from insiders as well as from external sources. A survey conducted in 2006 with the world's largest financial institutions showed that 78% of respondents (up from 26% in 2005) confirmed the occurrence of attacks from external sources, and 49% (up from 35% in 2005) stated they encountered breaches from inside sources[1]. The reported attacks occurred even in the presence of security measures that were implemented by the financial institutions. Therefore, evaluation of the security in networks is vital to gauge the need for additional security measures. Evaluation of a network security would lead to more understanding of how secure the network is, so that decision-makers would be able to see whether the employed security measures are still effective while evolving over time in reducing the threat. In the event that the security threat is increasing, decision-makers would be able to make decisions on investing more in security measures.

Combining security evaluations from different aspects of information security into a single overall security score, such as Information Assurance (IA) rating or index

[1]The third annual "Global Security Survey" conducted in 2006 by the "Financial Services Industry practices of the member firms of Deloitte Touche Tohmatsu (DTT)"

[Nandy04], will be of considerable importance. This is reminiscent of assigning an overall financial index for an enterprise, as known in the financial market.

In the stock market, Dow Jones Industrial Average (DJIA)[2], which is the oldest among more than 3000 indexes maintained by Dow Jones[3], consists of watching the changes in stock prices of the largest 30 public companies in the USA representing about 23.8% of the whole U.S. market. This index is viewed as an important measure of the performance of the American industrial stock market. While many stock indexes are calculated based on both stocks' prices and corresponding number of shares, the DJIA is based on only stocks' prices. Thus, the DJIA is calculated by adding the prices of its 30 stocks and dividing the sum by a divisor called "Dow Divisor". Initially more than a hundred years ago, the divisor's value was exactly the number of stocks i.e. the number of companies[4] selected in the DJIA. To face the effects of stock splits, and the change in the list of 30 companies, the divisor had to be adjusted. Currently, the Dow Divisor value[5] is 0.12493117, and the DJIA index value is above 12000. Therefore, the DJIA is considered price-weighted rather than market-capitalization-weighted, because highly-priced stocks have more influence in the index.

While the DJIA is a price-weighted index, many other Dow Jones indexes are weighted

[2]http://www.djindexes.com/mdsidx/index.cfm?event=showAvgOverview&averageSelection=I

[3]http://www.djindexes.com/mdsidx/?event=showAboutUsOverview

[4]Charles Dow started calculating his average with 11 stocks in 1884, but published it with 12 stocks in 1896. In 1916, the average was expanded to 20 stocks, and to finally 30 stocks in 1928.

[5]http://www.djindexes.com/mdsidx/index.cfm?event=showAvgOverview&averageSelection=G

by market capitalization[6]. In such indexes, the companies of larger market capitalization have more influence.

A financial tool called Composite Financial Index (CFI) is proposed by Salluzzo and Prager to indicate the overall level of financial health of an institution [Hudack03]. The proposed CFI is the result of combining four measure ratios: Primary Reserve Ratio, Net Income Ratio, Return on Net Assets Ratio, and Viability Ratio [Hudack03]. These four measurements are considered key performance metrics to assess the financial health of an institution.

While the procedure of CFI is useful to assess the financial health of an institution, the weights assigned to the metrics may arguably vary between organizations. Since the main purpose of the CFI is to compare with different institutions, Salluzzo and Prager suggest that these same weights be used for all similar institutions, and different weights be used for long-term debt institutions [Hudack03].

In the case of Network security evaluation, there is a need to assign metrics to know how much an organization is immune to attacks and whether confidentiality, data integrity, and network availability are always ensured.

[6]The market capitalization is usually referred to "the number of common shares multiplied by the current price of those shares" [http://en.wikipedia.org/wiki/Market_Capitalization]. The "full" market capitalization is calculated using the total number of shares, while the "free float" market capitalization takes into account only the portion of shares available in the market.

1.2 Objectives of Evaluation

The problem of assigning a security score to the security of a network, can be viewed from two different points depending on the objectives of evaluation. If the aim of evaluating the security of a network is to compare the status of security measures between multiple institutions, standardized metrics should be employed in order to come up with a comparable score. Since enterprises have different objectives and implement different security measures, finding standardized metrics that can be industry-wide and universally-accepted seems very difficult. This is even more obvious when an asset is considered important for an enterprise but marginally important for another.

However, if the objective of evaluating security is to monitor the security status over time within an institution, then the metrics used in the evaluation would be specific to this particular institution according to its own objectives and own perception of important assets.

The work of this thesis falls in the objectives of evaluating security over time within an institution, rather than comparing security between institutions.

1.3 Quantitative versus Qualitative Metrics

Whether the metrics used in evaluating the security of networks are standardized or not, there is an issue of which metrics are relevant and whether they should be qualitative or

quantitative. In risk assessment, the advantage of qualitative risk metrics resides in the ability to "prioritize the risks" and to "identify areas for immediate action and improvement" in a subjective manner [Peltier05]. In other words, if a certain identified risk is very likely to occur, it is assigned a high priority value, so that an immediate action should be considered. But in this case, there is no ability to measure the "magnitude" of the risk impact. On the other hand, the quantitative risk metrics measure the "magnitude" of the risk impact, but it might not be easy to "conduct" such measurements. Peltier states that some "subjective" factors may still be needed in quantitative risk assessment, such as [Peltier05]:

- An estimate of the threat occurrence rate over a period of time
- An estimate of the cost of each threat occurrence
- An estimate of the weight of the "relative impact" of each threat occurrence

The main focus of this thesis is on the selection of "quantitative" metrics to evaluate the security of a network. Then, the impact of this selection on the security evaluation is experimentally studied.

Building on the framework proposed in [Nandy04] and [Seddigh04] that consists of a general Information Assurance (IA) quantitative metrics taxonomy, the thesis extends that proposal while focusing on IDS metrics for network security evaluation rather than considering the general problem of IA. The experimental study conducted in this thesis on IDS metrics is one important step towards evaluation of network security.

A Hierarchical Quantitative Metrics (HQM) model, which can be a part of the general framework proposed in [Nandy04, Seddigh04], is presented. While the framework in [Nandy04, Seddigh04] considers Information Assurance (IA) in general, and argues that Quality of Service (QoS), Availability, and Security are essential to the evaluation of IA, this thesis only considers the technical aspects of network security evaluation, based on network IDS metrics, rather than the general IA evaluation problem that would include operational and organizational factors. To experimentally demonstrate the use of the HQM model, a prototype network security evaluator is implemented based on the HQM model and tested using real network traffic traces.

1.4 Thesis Contributions

The main contributions of this thesis consist of the following:

- State-of-art of information security evaluation approaches are classified (Chapter 2).
- A Hierarchical Quantitative Metrics (HQM) model for network security evaluation is proposed, building on the security branch of a general tree-based metrics framework.
- A prototype tool that automates the use of this model is implemented.
- While the HQM is general, the model used in experiments focuses on one aspect of network security through selected sets of Intrusion Detection System (IDS) metrics.
- Using Snort as a case study of IDS, three sets of intrusion metrics are proposed. Rationale is given behind selection of metrics from different perspectives.

- Experimentations using traces of real network traffic from the WIDE network are conducted.

Some of the results mentioned above are published in the paper titled "Experimental Evaluation of Network Security Through a Hierarchical Quantitative Metrics Model" [El-Hassan06].

1.5 Thesis Organization

The rest of this thesis is organized as follows.

Chapter 2 examines prior approaches to evaluating security in general, followed by a discussion on how metrics of each approach may contribute in the evaluation process.

In Chapter 3, details of the HQM are presented focusing on one aspect of network security, namely the status of threat due to network intrusions. Then, a way of normalizing and combining intrusion metrics is presented with corresponding formulas.

The Chapter 4 starts from a background on IDSs, then states the factors that may affect the IDS metrics used in the evaluation. Exploring the Snort IDS as a case study, the challenges of selecting three different sets of meaningful metrics from Snort are discussed.

The experimental evaluation procedure appears in Chapter 5, where two network

setups are considered to test different perspectives of network security evaluation using a security evaluator prototype designed and implemented for that purpose. Real network traffic traces are applied in both setups needed for the experimentations.

In Chapter 6, the actual experiments on one set of IDS metrics are detailed aiming at the study of the effect of metrics' weights and Snort rules on the evaluation results. The experiments of Chapter 7 are led using different sets of IDS metrics followed with comparative results.

Conclusions and future work are pointed to in Chapter 8.

Chapter 2

Methodologies to Evaluate Information Security

Developing a methodology to measure security is not a trivial task. To know which methodologies were studied in the literature to evaluate information security in general, and network security in particular, a survey is conducted. In this chapter, the methodologies to be addressed are then identified as follows:

- Iterative Optimization
- Kolmogorov Complexity
- Directed Graph for Authentication Metrics
- Fault and Risk Assessment
- Goal-Driven and Relevant Software Metrics
- Product Evaluation: Common Criteria (CC)
- Process Evaluation: Capability Maturity Model

- Economic Security Metrics
- Tree-based Metrics

2.1 Iterative Optimization

If the inputs to an evaluation system are simple and subjective, the evaluation output will be probably simple and subjective. To have an "objective" evaluation output, it is very useful to have "objective" inputs. But if accurate and objective input knowledge is not available a priori, it is possible then, by experimenting and learning, to refine knowledge in order to achieve more accurate output. This iterative optimization methodology is the essence of the idea of "Bayesian statistics" which consists of gradually refining knowledge and moving from a situation with "imperfect information" to one with "perfect information" [Burgess04]. This assumes that some data may be missing in the beginning of an estimating process, and from subsequently experimenting and revising the original assumptions the knowledge of a system can be built up. Burgess states that even though uncertainty remains to some extent, the process "can be quantified and allowances can be made for the imperfection" until convergence occurs [Burgess04]. To apply this approach in network security evaluation, one would start from the fact that uncertainty exists in the beginning of a network security evaluation process, and by some experiments and statistics the process can be quantified in a more certain way. The term "Bayesian" comes from the name of a theorem called "Bayes' theorem" for conditional probability. Even though clear metrics have not been found in the literature using Bayesian statistics, it is possible that

any subjective metrics would work as inputs, and over time the experimentations would refine the metrics and turn them if possible into objective ones.

Not far from the concept of Bayesian statistics is the essence of "game theory." Burgess states that game theory is a method to set a "pre-emptive and defensive strategies against one another," and then find their point of balance in order to "maximize gain and minimize loss" [Burgess04]. In network security, this theory seems available in mind when the evaluation process tries to maximize security strength of a network, and minimize vulnerabilities and intrusion possibilities. This same "game theory" can be applied after evaluation and during ranking a network in different and subsequent times. Burgess makes the statement that game theory is "applicable in all cases where it is difficult to evaluate the gains generated by following particular policies" [Burgess04]. For example, the authors of [Cavusoglu04] proposed a model based on a "game tree" to analyze IT security investment problems. The metric used in [Cavusoglu04] consists of an incurred cost if any intrusion is detected.

Data fusion is "the process of collecting information from multiple and possibly heterogeneous sources and combining it in order to get a more descriptive, intuitive and meaningful result" [Siaterlis04]. Having known the "game theory" can be applied in all policies that don't facilitate the evaluation of generated gains, data fusion principles assume also that it is difficult to evaluate a system having many diverse and "disparate" sources. Thus, data fusion addresses the "correlation" between these "disparate" sources [Fox03]. Similar to

Bayesian statistics, some researchers apply the notions of "uncertainty" and "ignorance" in their data fusion models as in [Siaterlis04]. Note that data fusion may be applied in both detection network anomalies [Siaterlis04] and evaluation of information assurance [Fox03]. The data fusion model of [Fox03] generates a single network security assessment as a result of fusion of multiple Information Assurance (IA) tools outputs, such as vulnerability assessment and risk analysis tools. Fuzzy Logic may be used as well to apply the fusion concepts in the IA domain [Fox03]. However, it is not clear how metrics are defined and extracted from the fusion process.

2.2 Kolmogorov Complexity

The concept of Kolmogorov complexity addresses a way to measure a complexity of a system. This means a system can be measured how complex it is by referring to the "minimum length of a program" needed to generate a random string or sequence by a universal computer [Evans01]. Kolmogorov complexity is then "a measure of descriptive complexity contained in an object" [Evans01]. Interestingly enough, Kolmogorov complexity cannot be computed [Evans01]. Instead, it can be estimated by its upper bound only, while its lower bound cannot be computed as well [Li97, Evans01]. In other words, a program, that happens to be the minimum length needed to generate a random string, indicates the best estimate of the upper bound of Kolmogorov complexity. If we want to estimate Kolmogorov complexity of a string, given some input data, the knowledge of this input data may reduce the size of the program needed to generate the string, and thus,

reduces the complexity needed to produce the string. If there is no such a program that can generate the string, the Kolmogorov complexity of that string is estimated to be infinite. But how this relates to information assurance and network security? The relation comes from the fact that a system which is "apparently complex" is less vulnerable to attacks [Evans01]. Similarly, the lower the "apparent complexity" is, the easier for an attacker to read and understand data of a system.

A proposed metric consists of measuring the apparent "joint complexity" between input and output of a system [Evans01]. That is if the input and the output are concatenated, the complexity of this concatenated data, identified as joint complexity, would be very high if there is no correlation at all between the input and the output, or very low if some significant correlation between the input and the output exists. Thus, the joint complexity metric could be an estimated measure of how vulnerable the data are. For example, the correlation that may exist between Distributed Denial-of-Service (DDoS) attack flows can be identified by an estimation technique of Kolmogorov complexity [Kulkarni06]. An alternative metric can be the measure of "relative complexity" of a process [Evans01]. That is if some input given data do not reduce the complexity of a string, but a certain process has modified the given data somehow and led to reduction in the program size needed to generate the string, thus, the relative complexity of this process has been reduced. This metric measures the process vulnerability. Both data and process vulnerabilities can be detected "without *a priori* knowledge of vulnerability types" [Evans01]. However, the inability to compute Kolmogorov complexity gives impression that the estimated metrics

mentioned above may not be enough to rank the security of a network accurately.

Though the above-description of Kolmogorov complexity may have some relationship with the general ways to measure software complexity, the software complexity usually points out to something different from what Kolmogorov complexity does. A high value of software complexity can be an indication of "low comprehensibility" and "low reliability" even though it is not necessary true in all cases [Fenton91]. Thus, the complexity of a software problem is considered as "the amount of resources required for its solution" in terms of time (performance) and space (memory) [Fenton91]. Fenton makes the point that measuring software complexity means measuring a number of internal (structural) attributes, rather than external attributes that are more related to software quality such as reliability, maintainability, and usability [Fenton91]. Fenton assures, though, that internal measures can improve the overall quality and is possible to use them for "quality control and assurance" [Fenton91]. Internal attributes examples include size measures, such as length and functionality, modularity measures, such as coupling and information flow, and test coverage measures [Fenton91]. Some of the measures can be collected by "non-commercial" static analysis tools [Fenton91], such as QUALMS (Quality Analysis Measurement) and METKIT (Metrics Educational Toolkit) [Bush93]. It is useful to mention some software complexity metrics such as recursive complexity measures (Basili-Hutchens and Prather), VINAP measures, McCabe's cyclomatic complexity measure, and Henry-Kafura measure [Fenton91]. These measures are specifically designed for software products, and their validity is not definite. For example, McCabe's cyclomatic measure counts the number of

linearly independent paths in a flowgraph of a program. This number equals the number of edges in the graph - the number of nodes + 2. Thus, high values of the measure indicate that the number of edges in the flowgraph are much more than the number of nodes, which shows high complexity of the program, according to McCabe, with a possibility of error-prone and lack of maintainability in the code. Fenton and Pfleeger comment on McCabe's measure as "misleading", because it measures only the number of decision nodes + 1, and many programs with high number of decision nodes are still understandable and maintainable [Fenton97].

2.3 Directed Graph for Authentication Metrics

In Public Key Infrastructure (PKI), there is a need of authenticating entities in communication paths with legitimate certificates. Since some authentication ways are less trustworthy than others, some researchers proposed metrics to evaluate the "confidence afforded by a set of paths", while a single path is a chain of authorities [Reiter99]. The authors of [Reiter99] indicate that a single path is weak because if an authority in the path has falsely authenticated the next authority, misleading occurs. Though a set of multiple paths may avoid the misleading possibility, the possible existence of correlated authorities may not provide enough confidence in the authentication. That's because a misbehaving authority that may exist in multiple paths, would cause authentication ambiguity. Reiter and Stubblebine criticize some previously proposed numerical metrics that measure the assurance provided by multiple paths, because they seem work for "specific goals" at

the expense of other properties [Reiter99]. These "other properties", that may have been slightly ignored previously, are further explained in [Reiter99] in terms of 8 principles which are requirements for good authentication metrics.

Principle 1 states that the user is not required to infer any binding between a key and its owner. Instead, the metric should be employed to determine "name-to-key" binding.

Principle 2 points out to the importance of having unambiguous parameters, especially where probabilities and trust values are involved.

Principle 3 requires the metric output be based on as much as possible of relevant information, not just limited information, that helps make the authentication decision easier.

Principle 4 indicates that the user should be consulted for non-automated decisions.

Principle 5 requires that metric output be intuitive.

Principle 6 asserts that the metric should be designed to be resilient to manipulations that may be led by "misbehaving" entities.

Principle 7 requires that metric be computed efficiently.

Principle 8 makes the point that if the metric output is based on partial information it has to be meaningful enough to draw useful conclusions.

The proposed metric in [Reiter99] respects the above-mentioned 8 principles. The authors' model consists of a directed graph, whose nodes are public keys, and edges represent certificates. Each edge is labelled with a numerical value, called insurance label, which represents an amount of money that has to be used to "insure" the correctness of the attributes

in the certificate. For example, if the edge K_1 —> K_2 exists in the directed graph model, then if the private key corresponding to K_2 is compromised (and used to mislead the user) the owner of K_1 is "liable" to the user for an amount of money which is the insurance label for edge K_1 —> K_2. Then the metric used by [Reiter99] is the "minimum insured amount" of the "name-to-key" binding for the target key, which is the last node in a path. This minimum amount can be computed by a graph theory tool such as minimum capacity cut or any maximum flow algorithm [Cormen01].

The description of the insurance metric above is a method that keeps in mind the way the insurance industry works. However, certificates revocation may be a difficult problem and a limitation of the proposed model [Reiter99].

2.4 Fault and Risk Assessment

While some IA papers address the complexity of a system, some others assess the reliability of a system. Therefore, a system's reliability can be measured by assigning metrics to the outcome of a series of either events (that have been occurred) analyzed by a probabilistic approach through Event Tree Analysis (ETA), or faults (that have or have not yet occurred) analyzed by another (or similar) probabilistic approach through Fault Tree Analysis (FTA) and fault localization techniques. Details of these methods have been studied in [Apthorpe01][Hallberg05].

In [Hallberg05], a framework for system security assessment is proposed, not as a

solution to system security values calculations, but to help in categorization of system security assessment approaches such as CAESAR [Peterson04] approach described in a master's thesis. CAESAR calculates an Overall Security Level (OSL) of a system, by using scalar security values for system components modeled either traffic generators (computers and networks) or traffic mediators (firewalls, routers, proxies, and hubs) [Hallberg05, Peterson04]. However, CAESAR does not take into account processes, then, "security aspects of system operation are not captured" [Hallberg05].

The impact of faults on routers in large networks has been studied by [Hariri03], where online monitoring of vulnerabilities is led to measure whether the operation of the network exceeds a certain threshold using a Vulnerability Index (VI) metric, in order to determine the network health in terms of 3 states: normal, uncertain, and vulnerable. Then, an agent-based engine calculates 2 impact factors to determine the impact of faults on routers [Hariri03].

2.5 Goal-Driven and Relevant Software Metrics

Software metrics are a "diverse collection of fringe topics" which range from predicting software costs at the specification stage to measuring of program structures [Fenton97]. A software metric may be defined in terms of "process" and "product", so that it represents "the degree to which a process or product possesses a given attribute" [*Thayer, Editor*97]. It may also be defined in terms of software quality, quantity, and

management [*Thayer, Editor*97]. Some quantity metric examples include "number of function points", "lines of code", "scale of software failures", "number of decisions", "number of operators in a program", "number of bugs", and so on [Fenton91]. Examples of software management metrics include "software size", "personnel", "volatility", "computer resource utilization", and "schedule progress" [Schultz97].

Fenton states that setting up a "metrics program" is necessary in a way that must have "goal-driven" approach. However, it has to have "long-term benefits" and be designed carefully in order to avoid "pitfalls" [Fenton91]. The Goal/Question/Metric (GQM) paradigm of Basili and Rombach [Basili88] [Basili94] is consistent with the approach of Fenton in setting up software metrics [Fenton97]. Thus, quality can be known a priori, while improvement can be achieved by learning. Some factors in software quality models include reliability, reusability, maintainability, and testability [Fenton91]. The criteria for setting up metrics depend in the case of, for example, reliability, on accuracy, completeness, and consistency [Fenton91]. In terms of GQM, the approach consists of 3 steps. In the first step, a number of goals are listed. From each goal, a number of questions may be derived in the second step in order to be answered. And to answer each question, a number of metrics must be decided in the third step. Thus, a goal can be "evaluating new design using specific method", a question can be "what productivity improvements?", and metrics can be "lines of code", "number of function points", "number of debugged executable statements per programmer-month", and so on [Fenton91]. However, there are some limitations in GQM. For example, some measures could be "subjective" that

may lead to "subjective" ratings [Fenton91] unless adjustment of metrics occurs by learning. For that, GQM methodology is sometimes combined with "process maturity" [Fenton97] [SSE-CMM03]. The GQM paradigm appears in some IA papers such as in [Chaula03], where the problem of interoperability between PKI and security metrics has been addressed.

The author of [Chaula03] used also Common Criteria (CC) (described in section 2.6) and Systems Security Engineering - Capability Maturity Models (SSE-CMM) (described in section 2.7) levels to set up security metrics. However, the security metrics in [Chaula03] are more specific to PKI application testing. For example, a goal can be: "To test if there is a capability to provide support to user who request certificate revocation." The associated question would be: "Is certificate revocation incident response procedure available and well-defined?" The assigned metric is: "Percentage of certificates revoked within defined time." From this metric a formula can be derived, which is: (Number of certificates whose private keys has been compromised but the revocation could not be done timely) divided by (Total number of certificate revocation requests) [Chaula03].

Another goal-driven approach is denoted by Hoo in the context of "network integrity" looking at it as a simplified "tractable subset" of what is measurable, instead of looking at the whole network information and involved cost [Hoone]. Since some metrics are, according to Hoo, biased, subjective, and not repeatable, there is a need for "good" metrics that are "goal-oriented" and possess the characteristics of SMART, the abbreviated word

formed by the terms Specific, Measurable, Attainable, Repeatable, and Time-dependent [Hoone]. Hoo gives examples of subjective and non-goal-oriented metrics that are based on data collection such as "the numbers of vulnerabilities found in network scans, known incidents reported, estimated losses from security events, security bug discovery rate in a new software application, intrusion detection system alerts, number of virus infected e-mails intercepted, and others" [Hoone]. As for good metrics for network integrity, Hoo provides some examples: "Numbers of vulnerabilities found on the network, broken out by those on policy-compliant devices vs. those found on devices that are not," "Numbers of users and devices compliant with each element of the security policy," "numbers of incidents attributable to policy failures vs. policy compliance failures," "impact of compromise in terms of users affected, data lost or modified, number of devices participating in compromise, decrease in network performance, increase in network utilization, and increases in wait times during a network compromise, and time between compromise discovery and completion of system remediation" [Hoone].

The thesis of [Wold04] includes a survey in which respondents (mainly security officers) answered predefined security-related questionnaire that is distributed to many organizations in Norway. There is another thesis [Simonsen04] that develops a process for security metrics program to "measure and estimate a network's trustworthiness", and many metrics are proposed as well. For example, Simonsen's proposed metrics include checking the implementation of the 3 layers of protection: prevention, detection and response.

Prevention includes "firewalls, router filters, switch-configuration (VLANs etc.), and out-of-band admin of elements exposed to external networks" [Simonsen04].

2.6 Product Evaluation: Common Criteria (CC)

The Common Criteria (CC) project [CC005] is an interesting tool to evaluate the security of IT products. An IT product can be, for example, an application-software, a piece of hardware, an operating system, a local area network, and so on. Since a non-configured IT product is passive, its evaluation has nothing to do with IT security. Once a selected product is configured, it is seen as a "target" and its configuration as a "Target Of Evaluation" (TOE) [CC005]. Since an IT product can have multiple configurations, it may have multiple TOEs because each configuration of the same product can contribute to IT security differently. Thus, the CC evaluation process is seen as an "active investigation" of a certain IT product [CC005].

Consumers and developers can use CC evaluations to see if their security needs are met. In order to give consumers the opportunity to tell their security needs, resulted of some sort of "risk analysis" and "policy direction," for a certain product type, the CC provides an "implementation-independent" structure called Protection Profile (PP) [CC005]. As for developers, the CC provides an "implementation-dependent" structure called Security Target (ST), that may be based on one or more PPs, in order to analyze how sufficient the TOE was in terms of assets, threats to assets, security

policies, and operational environment assumptions [CC005]. In order to let evaluation process work properly, the CC defines evaluation criteria for PPs and STs, and provides seven predefined assurance packages called Evaluation Assurance Levels (EALs) [CC005].

EALs define a scale for rating assurance for TOEs, and are ranged from EAL1 to EAL7 [CC005]. Each EAL contains a set of assurance components, and each component includes objectives, dependencies, and assurance elements. For example EAL1 is labelled "Functionally tested", and has components such as "Vulnerability survey" and "Stated security requirements." EAL2 is labelled "Structurally tested", and has additional components such as "Vulnerability analysis" and "Derived security requirements." EAL3 is labelled "methodically tested and checked" and has more additional components, and so on. EAL1 indicates that the tested TOE has the least security level, and is "functionally tested." EAL7 indicates that the TOE has the highest security level, and is "formally verified and tested" [CC005].

If components depend on other components, the dependent component may not reach a level higher than EAL2 because it does not have access to necessary information from the other component. That's why the CC define three Composed Assurance Packages (CAPs) (abbreviated CAP-A, CAP-B, and CAP-C) which provide also an increasing scale for rating assurance but for "composed" TOEs. Composed TOEs are those who contain "base components" (entities providing services) and "dependent components" (entities receiving services). For example, "an application (dependent component) may use services provided

by an operating system (base component)" [CC005].

The CC becomes an ISO standard (ISO15408) [ISO1540805]. However, after years of CC evaluation efforts of IT products, particularly in 2004, the future of CC seemed questionable [Hearn04]. The reasons reside in some key observations, pointed out by Hearn, such as "sellers do not see CC as a product-improvement process." Why don't they!? Hearn wrote that there is little commercial interest driving the CC market providing "most evaluations and certifications result from government regulation or government purchase" [Hearn04]. An important question is asked by any user: "How has this CC-evaluated product improved my IT system's security?" In fact, there is often no answer to this question [Hearn04]. Hearn stated that some IT professionals, who have experience in CC evaluations, suggest that there is urgent need to improve the security of a system composed of CC-evaluated products. There were concerns commented by [Lee03] that CC lacks "development process" evaluation. Even though the version 3 (issued in 2005) of CC [CC005] has improvements, the concerns stated in [Hearn04] and [Lee03] seem still valid.

2.7 Process Evaluation: Systems Security Engineering Capability Maturity Model (SSE-CMM)

The project SSE-CMM (Systems Security Engineering Capability Maturity Model) [SSE-CMM03] is a collaborative project from more than 40 industrial companies, in

addition to governmental organizations and Carnegie Mellon University (CMU) [Paulk93]. The SSE-CMM was developed "to advance the state of the practice of security engineering with the goal of improving the quality and availability" and "reducing the cost of delivering secure systems, trusted products, and security engineering services" [SSE-CMM03].

The SSE-CMM defines five capability levels, where level 5 is the highest improvement one: 1- Performed Informally: focuses on whether an organization performs "base practices." 2- Planned and Tracked: focuses on project-level definition, planning, and performance issues. 3- Well-Defined: checks whether organizational-level process is well-defined and disciplined. 4- Quantitatively Controlled: focuses on whether measurements are tied to organization goals. 5- Continuously Improving: the gain obtained from earlier levels is sustainable and facilitates improving.

The SSE-CMM distinguishes between process metrics and security metrics. Process metrics serve as "quantitative or qualitative evidence of the level of maturity" for a particular SSE-CMM process area. Security metrics are measurable attributes of "the result of an SSE-CMM security engineering process," and serve as evidence of the process effectiveness. Security metrics program can be built to measure the progress against security objectives.

The SSE-CMM process areas fit into three categories: Project, Organization, and

Engineering. The "project" process area focuses on one specific product, while "organization" process area focuses on multiple projects. The "engineering" process area includes administering security controls, assessing threats and vulnerabilities, assessing security risks, monitoring security activities, verifying and validating security [SSE-CMM03].

There are 22 Process Areas (PAs) for the development of metrics ranged from PA01 to PA22 [SSE-CMM03]. For example, PA09 is to "Provide Security Input", PA10 is to "Specify Security Needs", PA11 to "Verify and Validate Security." Some of those PAs are adopted by other IA evaluation tools. For example, PA09, PA10, and PA11, which are mentioned above, are used by [Chaula03].

2.8 Economic Security Metrics

Some researchers look at the security problem from an economic point of view. For example, a cost will be incurred if the firewall installed in a network drops an authorized user [Cavusoglu04]. As another example, a cost will be incurred each time "the audit trail of an internal user" is monitored for a possible intrusion [Cavusoglu04]. Schechter made a distinction between "security risk metrics" and "security strength metrics." Security risk metrics measure the level of risk in a certain system from a defender's perspective. Security strength metrics "quantify the time, effort, and other resources" required from an attacker's perspective to bypass the implemented safeguards that belong to a system in order to perform the attack. Therefore, Schechter views that the "measure of resources

needed to breach a system is fundamentally an economic one" rather than based only on the known computational approaches [Schechter04, Schechter05]. Then, if the cost to breach a system is too high, or in other words higher than the expected return, the attacker will not perform any search for vulnerabilities. To maximize their productivity, attackers may start with the tasks that seem more likely to be profitable and "have the greatest chance of success per unit cost" [Schechter05]. Then, the chance of finding a vulnerability increases as total investment increases, but the chance of success for each additional dollar invested is smaller than for the previous dollar [Schechter05]. This leads Schechter to say that the "perceived expectation of the cost," rather than the true cost, that determines how many resources attackers are willing to spend to find vulnerabilities in a system. Schechter ends up by determining "a metric of security strength" as it is the "perceived cost-to-break," that can be measured by "determining the price that someone would have to pay to acquire a vulnerability" [Schechter05].

There is a technique, called Business-Adjusted Risk (BAR), to classify security problems according to vulnerability type, risk of exploit, and business impact [Geer03]. The risk of exploit can have a high score if an attacker can exploit certain vulnerability easily. The business impact would have a high score if the successful exploit led to significant damages to a certain business enterprise either financially or in its reputation. An economic metric using BAR would be a tool for measuring business risk.

To identify the usefulness of implementing security protection measures, an enterprise

can assess the corresponding Return On Investment (ROI). To do so, it is useful first to quantify risks by evaluating the costs due to potential losses [Proctor01]. For example, if a specific asset such as a web server is compromised somehow and becomes unavailable, what is the cost expected from the occurrence of this single event? This leads to defining the Single Loss Expectancy (SLE) as the specific amount of money resulted from the occurrence of that event, which is equal to the asset value multiplied by an estimated percentage of loss called Exposure Factor (EF). Thus, `SLE = Asset Value x EF`. If this event is likely to occur more than once a year, the Annual Loss Expectancy (ALE) for the same asset is the SLE multiplied by the estimated Annual Rate of Occurrence (ARO). Thus, `ALE = SLE x ARO`. Security statistical numbers[1] show that 60 percent of respondents to a worldwide survey of 7,500 companies indicated that either "revenue impact" or "economic ROI" were used to justify security expenditures, according to October 2003 issue of CIO Magazine. More accurate estimations of ALE can be obtained after applying new safeguards as explained in [Hoo00, Schechter04]. Then, based on assumed reduction in ALE, a benefit can be calculated, and subsequently the ROI which is the benefit divided by the cost of the safeguards [Hoo00]. Since ROI is gaining more attention in security context, this metric is sometimes referred to Return On Security Investment (ROSI). The authors of an online article[2] use another economic rate of return, instead of ROI, called the Internal Rate of Return (IRR), which takes into account the differences in costs and benefits in different years. In his Ph.D thesis, Schechter wrote that both ROI and IRR are in fact "best

[1] http://www.cio.com/archive/101503/state.html [Accessed December 6, 2006]

[2] The article appeared in *Strategic Finance* in November 2002, and is citepd by [Schechter04]. It can be purchased from: http://www.findarticles.com/p/articles/mi_go1908/is_200211/ai_n7319926

guesses rather than quantitative models" [Schechter04]. These "best guesses" appear in the calculation of ALE and the assumption of ARO. However, since risk assessment is a "black art", risk mitigation with ALE calculations is possible [Proctor01].

2.9 Tree-based Metrics

To measure software quality, some proposed software quality models are basically tree-based. Boehm's model and McCall's model are among these tree-based quality models. The model of Boehm was originally the COCOMO cost estimation model [Fenton97], but is "tied" later to a quality model that consists, at a high level, of "intermediate constructs", such as reliability and portability, and at a lower level in the tree, of "primitive constructs", such as accuracy and device independence. At the lowest level of the tree are located the metrics needed to measure the "primitive constructs". The other tree-based model is McCall's quality model, which consists of Factor Criteria Metric (FCM) tree [Fenton97].

In the FCM model, the quality factors include mainly external attributes at a high-level, and are decomposed into quality criteria at a lower level. The metrics are measurable attributes for both product and process, and are at the lowest level of the tree. The metrics belonging to quality criteria are linearly combined to produce one metric for each quality criteria. The criteria belonging to a quality factor are linearly combined as well to produce one metric for each quality factor. There is no further combination of metrics, so that the metrics associated with quality factors would be indicative of the product uses whether in

operation, revision, or transition mode. For example [Fenton97], the quality criteria "Completeness" can have a metric to fulfill 6 conditions for requirements, and a second metric to fulfill 8 conditions for design, and a third metric to fulfill 8 conditions for implementation. Therefore the combined metric associated with the quality criteria "completeness" is calculated as:

$$\frac{1}{3}(\frac{R}{6}+\frac{D}{8}+\frac{I}{8})$$

where R stands for the number of fulfilled conditions for requirements, D for the number of fulfilled conditions for design, and I for the number of fulfilled conditions for implementation. Then, the combined metric associated with the quality factor "Correctness" will be the mean value of the metrics calculated for the three quality criteria that the factor depends on, "Completeness" (calculated above), "Traceability", and "Consistency".

Fenton mentions that the FCM model of McCall was developed for the US Air Force, but it is still not an adopted standard [Fenton97].

To determine security "risk metrics", the authors of [Clark05] make use of "mission trees" and "risk trees" based on enterprise objectives. A mission tree consists of root, which is the ultimate goal mission, and lower levels ranging from up to bottom as objectives level, sub-goals level, and assets level. Each objective, sub-goal, or asset is given a certain metric according to its importance in the enterprise. Since enterprise needs and objectives differ, the assigned metrics differ for different mission trees characterizing different enterprises. Then, with the help of vulnerabilities scanning tool, the "critical

states" of hosts are determined to evaluate the risk the hosts are at, where each vulnerability found will cause an impact value equal to the metric assigned by the mission model. By adding these risk states to the mission tree, the expanded tree becomes the risk tree that is able to show the degree of risk for the mission root. This allows, instead of manually patching vulnerabilities, to derive "host trees", where the host is the root for each tree and associated objectives and vulnerabilities are the leaves. Furthermore, a "vulnerability tree" can be built for each found vulnerability to visualize its impact, where the root is a certain detected vulnerability and the leaves are the hosts that are vulnerable to the specific vulnerability indicated by the root. Throughout the trees analysis, the metrics are just added from a low level to an upper level.

There were attempts to build taxonomies of IA metrics [Seddigh04, Nandy04, Vaughn03]. The metrics in [Nandy04] were chosen to be applied in the IT government sector (and similar large sectors) where metrics can be grouped into technical, organizational, and operational groups. Since the tree-based taxonomy includes comprehensively all aspects of information security, the model proposed in this thesis starts from that taxonomy framework to evaluate network security. More information about the taxonomy and the proposed model is presented in Chapter 3.

2.10 Discussion

From the descriptions of methodologies above, one may think about which methodology would be applicable in evaluating network security, and which metrics can be used effectively to achieve the goal of evaluation. In the following sections, each of those methodologies are discussed.

In the methodologies that take subjective or uncertain inputs and need optimization over time, such as in Bayesian and Fusion approaches, their importance resides more in detecting vulnerabilities or anomalies in networks [D'Ambrosio01, Laskey04] rather than evaluating and/or ranking a network. However, an efficient network security tool would take the outputs, resulted from any Bayesian or Fusion tool if exists, as inputs among others. Since in the beginning, the outputs of Bayesian and Fusion tools would be subjective providing their inputs are subjective, the network security evaluation tool would exclude them as inputs for some time giving them a chance to optimize over time. Then later on, their outputs would be applied to the tool. On-going adjustment of subjective inputs should lead to more accurately evaluate the overall network security health.

As for the Kolmogorov complexity measurement, described in section 2.2, it might be much more useful in detecting attacks [Kulkarni06] and in evaluating products rather than in evaluating the security of a network. If any tool that uses Kolmogorov complexity already exists, its output can be inputs to a network security evaluation tool for the

metrics that are related to the product measured by Kolmogorov complexity. However, the joint-complexity metric used [Evans01] points out to an important topic, which is the effect of correlated data on the vulnerability of a product or a system. In terms of security metrics, this leads us to question whether we can apply Kolmogorov complexity on possibly correlated metrics.

The CC project described in section 2.6 can also be run on any product and its output be applied to a network security evaluation tool. It might be useful for such a tool to contain a library of product evaluation and vulnerability assessment tools, so that any enterprise that have some of tools in the library may be applied to the network security evaluation tool. Further discussion is needed for this case.

The directed graph for authentication metrics has been discussed in section 2.3. The authentication metric used was derived from the combination of edges representing the minimum capacity cut of a directed graph whose edges are labelled with "insurance labels". In other words, the combined metric is just a summation of minimum insured edges to indicate the amount of trust in the certificates. This metric is used in the context of certificate authentication by analogy of the real insurance market. There is also a metric, derived from an economic model of security risks, called the "perceived cost-to-break" discussed in section 2.8, which measures the security strength by evaluating the cost that an attacker would afford to exploit a vulnerability. These are important metrics that are desirable in an effective network security evaluation tool. For example, it is possible for

such a tool to evaluate some measures in terms of cost or insured amount, and convert them later to some values to be combined to produce one global metric. Then for highly sensitive places in the network, high cost values (or high insurance) are applied on them if an attack occurs. If after some time (e.g. a year) no attack occurs on these sensitive places, the insurance on them gets lower.

Regarding the GQM paradigm, discussed in section 2.5, it is an important approach that can be applied in a network security evaluation tool. The metrics would be goal-driven, and if they are based on measurements, they will be objective metrics. The GQM approach can be combined with tree-based security taxonomy, such as [Seddigh04] and [Nandy04], to assess a network security. However, it is important to make sure that metrics are goal-driven and based on measurements to be meaningful and objective.

The SSE-CMM process maturity evaluation, described in section 2.7, is important but may be replaced by a less complicated procedure like GQM or other goal-driven and tree-based approaches.

The tree-based taxonomy approaches, described in section 2.9, are very important to be part of an effective network security evaluation tool. Therefore, metrics derived from well-studied tree-based taxonomy combined with goal-driven approaches would be a good factor in building such a tool. The questionnaire used by [Wold04], mentioned in section 2.5, would be also applied as a subjective input to that tool and be refined over time by

experimentation.

2.11 Examples of Commercial Tools

In the security market, McAfee security center provides, in its Internet Security Suite 2005, a security index from 1 to 10 for evaluating the security of a computer, where 10 indicates that the computer under test is very-well secure, while 1 indicates a serious security problem[3]. This index is an average based on AntiVirus, AntiHacker, AntiAbuse, and AntiSpam indices, which each of them is in turn rated from 1 to 10. However, there are no available documents that explain how every index is calculated.

Egan and Mather from Symantec[4], proposed recently a security evaluation framework that includes scored templates in terms of 3 categories: People, Process, and Technology [Egan05]. Adding the scores of the 3 categories result in an overall security assessment number (between 0 and 100) and rated accordingly as Good, Average, or Poor. Since companies have different levels of "dependency" on information security, the score ranges needed to rate the security system differ accordingly. Thus, companies having high "business dependency" on information security need to achieve higher overall assessment to be rated as "Good" [Egan05].

[3] us.mcafee.com/common/en-us/redirects/msc/datasheet.asp
[4] www.symantec.com

Chapter 3

HQM: The Hierarchical Quantitative Metrics Model

As indicated earlier, in [Nandy04] and [Seddigh04] a tree-based Information Assurance metric taxonomy was proposed. Metric groups can be applied to each of three portions of the Nandy et al proposed IA taxonomy: Security, QoS, and Availability. In addition to be measurable and repeatable, the defined metrics are required to be also reproducible, usable, and robust. In their work, the authors proposed that the system should be able to accept "flexible" metrics by including new metrics "on an on-going basis" without obsoleting previous evaluations [Nandy04]. However, the authors acknowledge that the proposed taxonomy is not validated yet and some metrics in the taxonomy may have to evolve over time. Each proposed metric is assigned a score value and a "formula" based on four elaborated fields: "What to measure", "How to measure", "How often to measure", and "Data source". As examples of proposed metrics in [Nandy04], are the following metrics: "Percentage of networking components with a certain security rating",

"Percentage of software components with a certain security rating", "Percentage of IT budget allocated to security technical resources", "Number of detected virus attacks per month", "Percentage of unsuccessful service denial attacks during a service denial test", "Percentage of protected network access points connected directly to the Internet", and so on.

On the top of the tree-based IA metric taxonomy, an IA grand metric (i.e. IAM) is dedicated to comprise the three categories of metrics mentioned above. The IAM would be a health indicator reflecting the information assurance posture of an enterprise IT network. Each one of these three categories is further subdivided into technical, operational, and organizational elements. Figure 3.1 shows an overview of the general taxonomy framework [Seddigh04], emphasizing on its Security Branch.

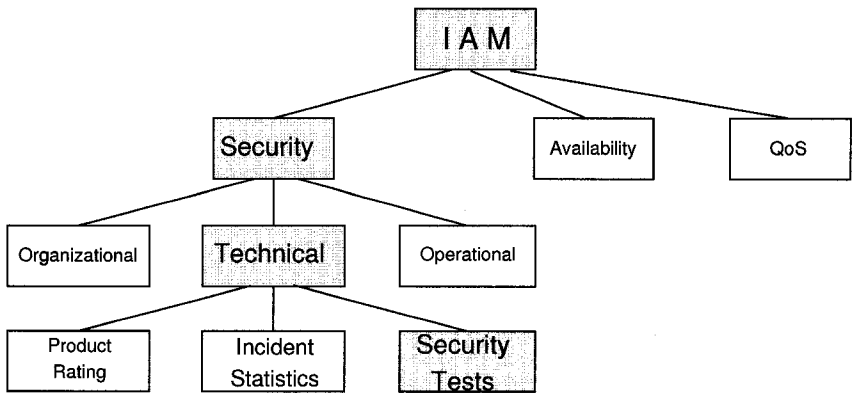


Figure 3.1: Overview of General Taxonomy Framework [Seddigh04].

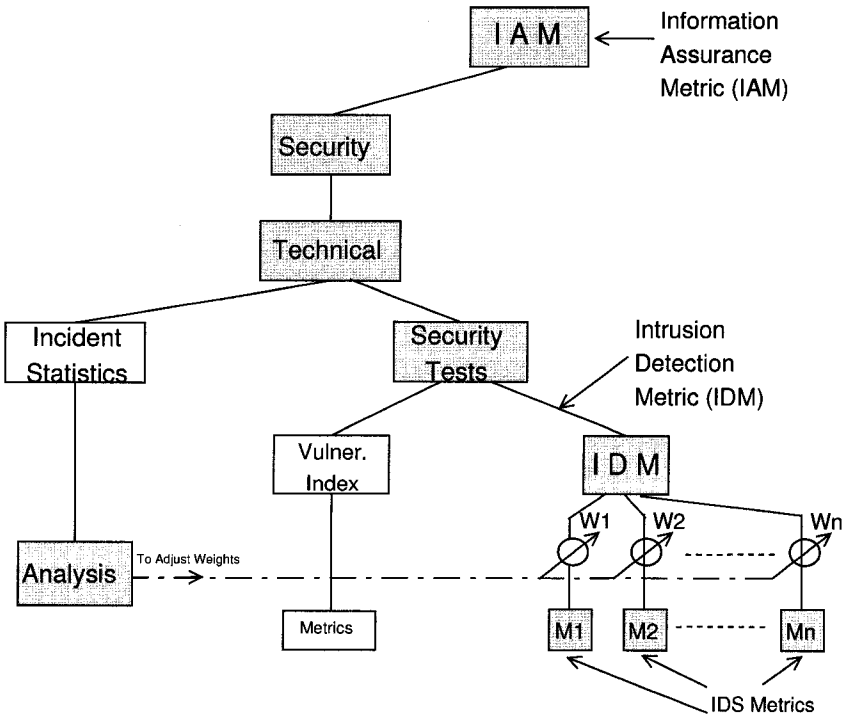


Figure 3.2: The Hierarchical Quantitative Metrics (HQM) Model.

3.1 Model

For the purpose of producing one global metric to depict a network health, as an index based on metrics derived from many aspects of network security, the tree-based taxonomy proposed by [Seddigh04, Nandy04] is considered. Starting from this taxonomy, and benefiting from the understanding of other approaches, an efficient network security evaluation model is sought.

In Figure 3.1, the "Organizational" category involves management policies, and the "Operational" category involves personnel practices. Focusing on the "Technical" category and its children in the tree excluding "Product Rating" (Figure 3.1), a Hierarchical

Quantitative Metrics (HQM) model is proposed (Figure 3.2). The technical category of the tree is expanded, as shown in Figure 3.2, into two sub-categories of metrics that contribute in the delivery of the technical overall metric for the "Security" branch. These two sub-categories are Incident Statistics, and Security Tests. The Incident Statistics are supposed to indicate, based on real traffic, the rate of attack incidents that could have penetrated the network. The Security Tests are done using known security tools, to scan the network for vulnerabilities and to alert on possible intrusions. The Security Tests sub-tree further consists of two sub-categories: vulnerabilities and intrusion detection (Figure 3.2). A variety of metrics can be utilized to reflect the status for each of the above sub-categories. In the case of vulnerabilities and intrusion detection, actual quantitative metrics can be extracted from the logs of various security tools.

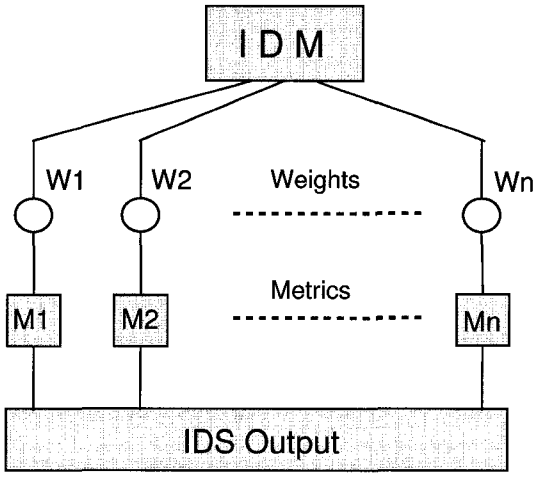


Figure 3.3: The Intrusion Detection Metric (IDM) Sub-Tree.

As indicated above, there are two sub-trees under the "Security Tests" of the HQM - one test is performed using a network-based IDS and the other is done using a Vulnerability

scanner (Figure 3.2). The IDS is capable of alerting on potential external intrusions, while the Vulnerability scanner would identify the vulnerabilities that may exist in the network hosts. These two components should be enough to evaluate the "Security Tests" sub-tree of the HQM even though additional possible tools could be used such as host-based IDSs and custom penetration test tools. In this thesis, the focus is on evaluating the Intrusion Detection branch of the "Security Tests" sub-tree, as illustrated in Figure 3.3. Therefore, The proposed model is not intended to detect unknown attacks unless the IDS detects suspicious activities and generates alerts.

An overall IDM (Intrusion Detection Metric) is calculated to reflect the Intrusion Detection posture of the IT infrastructure. The IDM contributes along with other security measures to provide the overall IAM metric shown on the top of the HQM tree (Figure 3.2).

For the purposes of this thesis, the Snort IDS [Roesch99] - a well-known open source network-based intrusion detection system - is utilized.

A Vulnerability Index (VI) would be on the top of a vulnerabilities sub-tree as shown in Figure 3.4. Experiments with the vulnerabilities sub-tree should be part of future research.

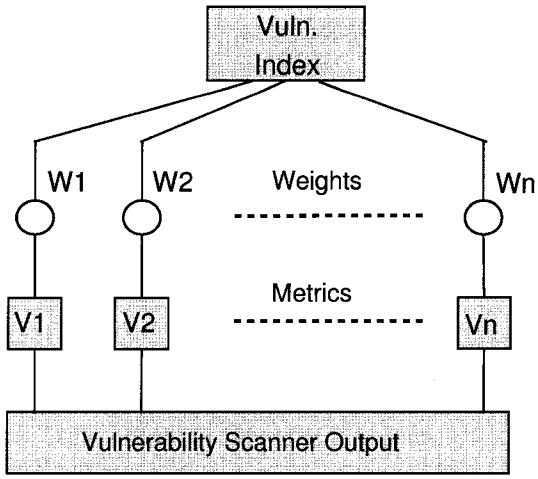


Figure 3.4: The Vulnerability Index (VI) Sub-Tree.

3.2 Aggregation of Metrics

The selected IDS metrics must be combined together to produce an overall IDM as shown in Figure 3.2. There are a few steps in this procedure. Firstly, since each metric is usually not equivalent to the other in terms of importance, different weights have to be assigned to selected Snort metrics, reflecting their relative importance to network security. Secondly, since the raw count of Snort alerts for a particular priority level may differ greatly, some kind of normalization must be applied on the metrics in order to obtain meaningful results. Finally, some combination of the normalized metrics with their relative assigned weights is needed to produce the IDM.

3.2.1 Calculation of IDM

At the bottom of the HQM model tree, in Figure 3.2, every measurement m_i is first normalized to the metric value (M_i) then multiplied with its assigned weight W_i. Most of the

normalized values (greater than 90%) fall in the range $[0, 1]$.

The summation of all W_iM_i terms is then performed. The ultimate result produces an overall intrusion detection metric IDM.

Accordingly the IDM is formulated as follows,

$$IDM = \sum_{i=1}^{N} W_i M_i \tag{3.1}$$

where IDM is the resultant grand metric for the IDS branch of the sub-tree, and N represents the number of metrics under the same branch of the sub-tree. It is important to emphasize the point that the term IDM should be not confused with the term IAM - the overall grand metric at the top of the tree. Figure 3.2 makes this distinction very clear.

In Equation 3.1, the metrics M_i represent the normalized values of m_i measured directly from the network. This normalization is done by relating the measured values m_i to a threshold value that represents a percentage of the maximum value observed for the same quantity over a long time in the network. The threshold value is determined such that at least 90% of the m_i values, recorded for each metric, fall below the threshold. The reason for this is to ensure that at least 90% of the m_i values of each metric are normalized in the range $[0, 1]$, and that at most 10% of the m_i values exceed the threshold. It is necessary to utilize a 90% (or more) max threshold as opposed to the absolute 100% max threshold, because otherwise a few extremely high values may skew the results. Therefore,

$$M_i = \frac{m_i}{max\_thr(m_i)} \tag{3.2}$$

where $max\_thr(m_i)$ is the mentioned thresholds for metric values. Note that each threshold may vary when more metric values are collected over long periods of time.

The resultant IDM will have a value that reflects the weighted combined metrics. The combination is assumed linear, for simplicity and because it is an intuitive way of aggregation [Nandy04]. The assigned weights for the selected metrics must have a summation value of 1 [Nandy04], resulting in the calculation of the weighted-average of the metrics. The weights were selected to reflect the relative importance of the metrics as a starter in a way similar to the "swing-weighting approach" described in [Clemen01]. This weighting method consists of rating each objective (or metric) according to its importance to decision-makers, with some percentage number, then inferring a relative weight for each objective.

3.2.2 Generalized Equation for the Overall IAM

Equation 3.1 represents only the bottom level of the tree, namely the IDM level. All levels of the tree from bottom up can be represented in a general equation as in [Nandy04]. The authors of [Nandy04] show a "weight map" for multi-level tree-taxonomy. For example, a grand information assurance metric IAM is represented in a 3-level tree as

$$IAM = \sum_{i=1}^{N} W_i \left(\sum_{j=1}^{G_i} \left(K_{ij} \sum_{l=1}^{D_j} S_{ijl} M_{ijl} \right) \right) \tag{3.3}$$

where W_i represents the weights of the top-level branches of the tree having N metrics (i.e. IAM is connected at the top of the tree to N nodes), K_{ij} represents the weights of the middle-level branches of the tree having G_i metrics, and S_{ijl} represents the weights of the bottom-level branches of the treehaving D_j (i.e. M_{ijl}) metrics. Equation 3.3 is actually a 3-level generalized form that can be extended to many levels of the tree.

3.3 Model Refinement

It is mentioned above that the "Incident Statistics" are supposed to indicate, based on real traffic, the rate of attack incidents that could have penetrated the network. Based on analysis of these incident statistics, weights can be assigned to IDS metrics. With changes of statistics over a long period of time, the assigned weights can be adjusted. Next, a method of incidents analysis, that is helpful in determining the weight to be adjusted for each metric, is presented.

Incidents can occur due to internal attacks, where external activities did not contribute in causing of these attacks. In addition, incidents may occur due to external (or combination between external and internal) activities that were not alerted on by the IDS. Analysis of incidents is then needed to know the number of attacks that occurred due to external activities that were alerted on or not by the IDS. Therefore, four metrics would be sought

in incident statistics:

- Number of incidents due to external activities alerted on by the IDS.
- Number of incidents due to external (or combination between external and internal) activities that were not alerted on by the IDS (false negatives).
- Number of incidents due to internal activities.
- Number of incidents due to unknown sources.

The four metrics above appear in Figure 3.5 as S1, S2, S3, and S4 respectively.

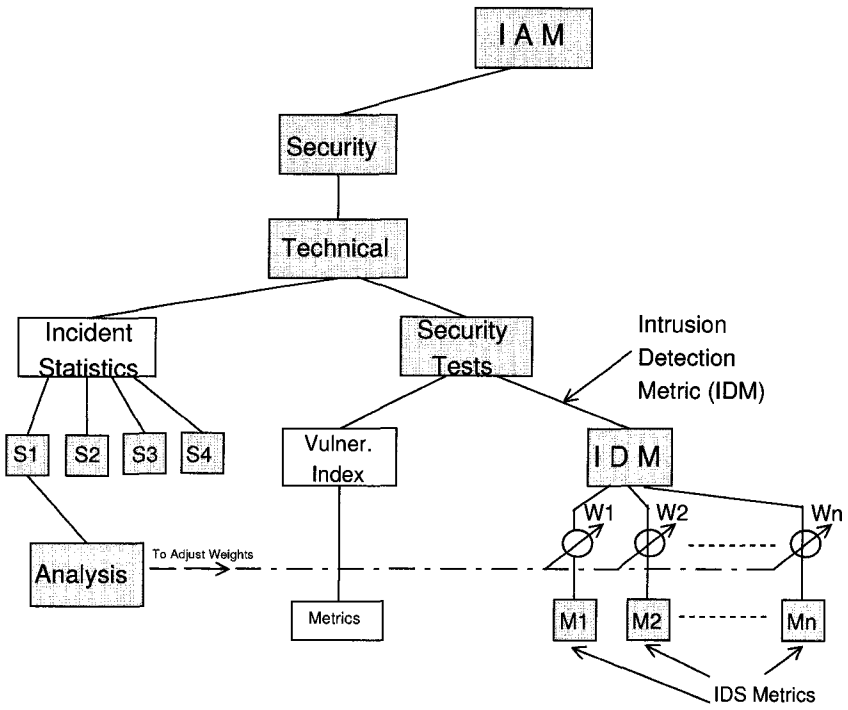


Figure 3.5: Adjustment of Weights Assigned to IDS Metrics in The HQM Model.

Obviously, the first metric (S1) - that is due to external activities alerted on by the IDS - must be the one to be known to adjust the weights of IDS metrics. The rate of false negatives

(second metric) is important to update the signatures stored in the IDS database, or better tune the IDS rules and string-matching algorithms to catch the missed attacks, but it should not have any role in adjusting weights of IDS metrics. However, the rates of both false negatives and incidents due to internal activities contribute, along with the first and fourth metrics (S1) and (S4), in the calculation of an overall metric for incident statistics. Up in the HQM tree, an overall "incident statistics" metric combined with an overall "security tests" metric are combined to produce an overall "technical" metric (Figure 3.2 and Figure 3.5).

Chapter 4

Intrusion Detection Metrics

Prevention, detection, and recovery mechanisms are among the range of elements of network security protection as recommended by various authorities in the security domain [Stoneburner02]. These mechanisms should be employed whether the threats originate from external or internal sources. Operational and organizational security measures are mostly implemented to protect institutions from insiders or from cooperative activities of both insiders and outsiders. These are considered preventive measures. A firewall is a technical example of preventive measures that reduces the amount of threat to networks from external sources.

4.1 Intrusion Detection Systems

Since preventive measures alone are insufficient arsenal for network security, a detection-based mechanism is important to identify the external and internal threats that occur in the network. A network-based Intrusion Detection System (IDS) is a good example for

detecting external suspicious activities, while a host-based IDS can be an additional tool to detect both external and internal threats.

4.1.1 IDS Functionality

IDSs are typically characterized as either anomaly or signature-based. The anomaly based IDS detects anomalies in the incoming traffic providing regular activities were already statistically collected. This is useful in detecting new attacks that look abnormal but may miss well-stealthed attacks. The signature-based IDS stores in its database signatures of already-known attacks, and employs pattern-matching algorithms to detect attacks - embedded in the coming packets - whose signatures are matched. However, it may miss a new attack if its signature is unknown and therefore does not have a corresponding signature stored in its database. Sometimes hybrid IDS systems are used in order to detect both new anomalies as well as known attack signatures.

On detecting suspicious activities, the IDS produces alerts which can be analyzed further by security experts to identify true intrusions or false alarms. This analysis is vital in an effective recovery mechanism. The generated IDS alerts can be very helpful if they can be analyzed quickly. Today however, the volume of alerts generated by IDS systems are so high that it is challenging to effectively undertake rapid analysis. Furthermore, successful intrusions remain undetected, even though some IDS systems classify alerts into categories and use viewing displays of the classified alerts to facilitate analysis. Some

studies indicate that 98% of intrusions go undetected, and of the remaining small number of detected intrusions, only 5% get reported [Killmeyer06]. Clearly then, the impact of network attacks including intrusions is costly for many institutions.

To illustrate more the IDS functionality, the flow diagram shown in Figure 4.1 points to four possible states in which an incoming packet can be after it has been examined by the IDS. These states are due to either correct or incorrect behavior/configuration of the IDS.

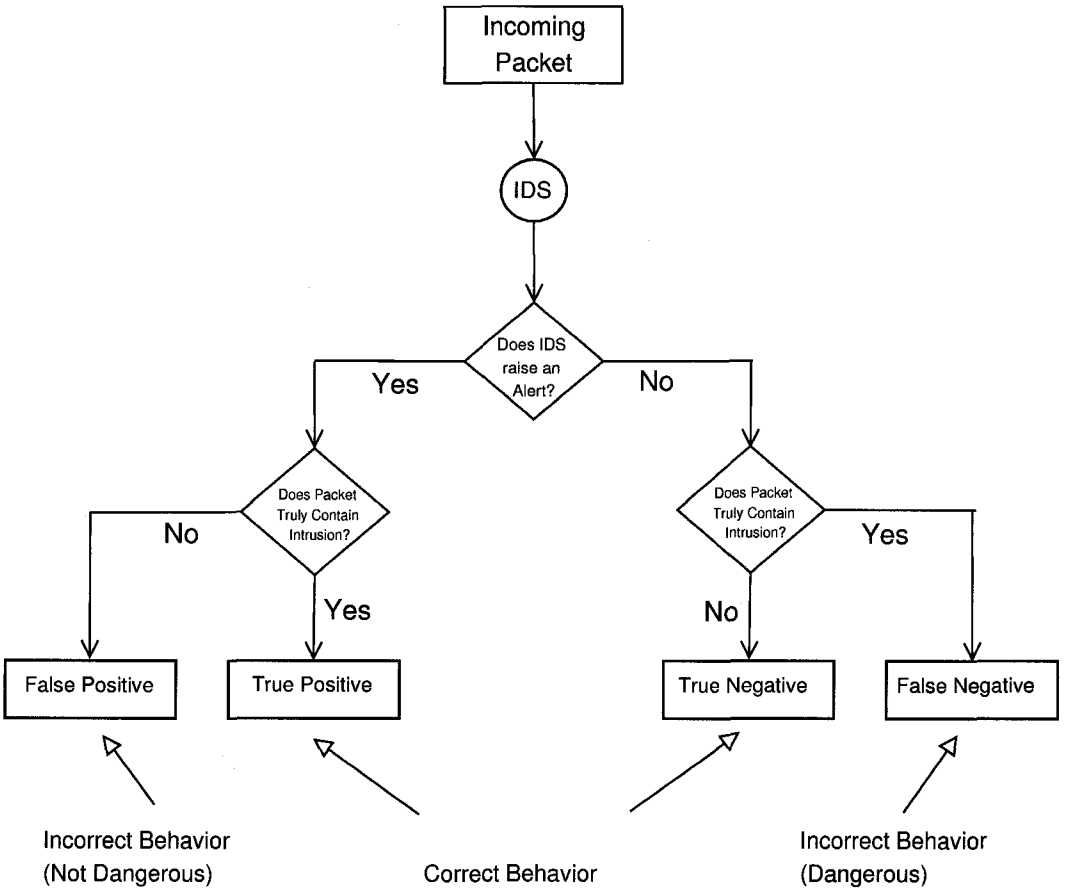


Figure 4.1: Flow Diagram that Illustrates the Decision Behavior of an IDS.

The correct behavior of an IDS consists of the two states: *True Positive* and *True Negative*. When the IDS flags an alert on a packet carrying truly an intrusion, the alert is a true positive. And when the IDS does not flag an alert on a packet that does not carry an intrusion, the no-alert IDS decision is a true negative.

The incorrect behavior of an IDS consists of the two states: *False Positive* and *False Negative*. When the IDS flags an alert on a packet that is legitimate and does not carry an intrusion, the alert is a false positive. However, when the IDS does not flag an alert on a packet that actually carry an intrusion, the no-alert IDS decision is a false negative.

The false negatives, if discovered, indicate that the IDS has dangerously failed to detect the intrusions embedded in the incoming packets. If this happened because of attacks' signatures that were not present in the IDS database, the new discovered signatures must be updated. However, false negatives may occur due to misconfiguration of the IDS [Cavusoglu05]. In this case, a special attention should be led in order to correctly configure the IDS to avoid such a dangerous situation.

In the event that a false positive is produced by the IDS, this incorrect noisy behavior is annoying but eventually is not dangerous. The false positives may be reduced by either disabling some "noisy" IDS rules or adding pass rules to allow legitimate traffic. However, disabling or adding many rules in order to reducing false positives may not be desirable, since this would allow the occurrence of false negatives. Thus, careful analysis of alerts is

needed to ensure that any modification in rules' configuration is safe. In this regard, there was a study that aimed at the reduction of false positives by proposing fusion-based IDSs [deBoer02], but it was not accompanied with an actual implementation.

By carefully selecting IDS metrics, out of IDS alerts, and combining the selected metrics to produce a threat index in an automated way, enterprises managers can see at a glance an overall indication of the security health of their network.

4.1.2 IDS Comparison

In research literature, IDS metrics have been examined in the context of evaluating specific IDS systems against other IDS systems [Fink02, Mell03]. Comparison of IDS systems was done using criteria such as logistics, architecture, and performance. For example, the ease of IDS configuration is a logistical metric [Fink02], while the system throughput is an architectural metric [Fink02] and the observed false positive ratio is a performance metric [Fink02, Mell03]. Other examples of IDS performance metrics include the coverage number of attacks that an IDS is able to detect, and its ability to determine attack success [Mell03]. Most of logistical and architectural metrics (and some performance metrics) are only useful for IDS system comparison purposes. We seek in this paper to identify the metrics that are useful for evaluation of IP network security health.

Even though the IDS was utilized as part of previously proposed tree-based metrics taxonomies [Seddigh04], IDS metrics were not focused on in those papers. In [El-Hassan06], an experimental study was done on the IDS branch of the technical group of the Information assurance (IA) taxonomy presented in [Seddigh04], where a few IDS metrics were selected and the effect of their assigned different weights on the security evaluation was analyzed. In this same study, the effect of disabling some IDS rules on the evaluation was included as well [El-Hassan06]. In this thesis, most of the contents and the results presented in [El-Hassan06] are included, and then expanded to include more IDS metrics.

4.2 Factors Affecting IDS Metrics - Snort: a Case Study

To determine the IDS metrics that may contribute in the evaluation of network security, Snort [Roesch99] is utilized as a case study. Snort is a widely-adopted open-source network-based IDS. Since Snort generates alerts on suspicious packets, metrics can be extracted from the Snort output alert file. These metrics, once identified, can be combined together to produce an intrusion index, called Intrusion Detection Metric (IDM). The total number of alerts generated by Snort is affected by the way Snort is configured - there are over 3000 rules which can be enabled or disabled. In the following sections, the effect of Snort configuration on the produced alerts is studied.

Snort consists of decoders and preprocessors which inspect incoming packets before

they are matched against stored rules [Roesch05]. Network inbound packets are first captured by libpcap, a network sniffer library. Received packets are captured and then put into a queue in order to be read by Snort. The captured raw packets then pass through three main sequential phases in Snort:

- Decoding.
- Preprocessing.
- Detecting.

Since some of these phases consist of components that may be either disabled or enabled via the Snort configuration file, the total generated number of alerts is affected. In addition, there are ways to reduce or stop alerts that might generate potential false positives. These issues are discussed in greater detail in the following sections.

Decoding

Once Snort is initialized, it checks for the enabled parameters (such as decoders and preprocessors) in its configuration file. Upon reception of new packets, the decoding phase starts. In this phase, each raw packet is tested for specific elements in headers of its embedded protocol layers - namely data-link, network, and transport layers that make up the raw packet. Accordingly, alerts may be triggered on header truncation or incorrect packet length.

Preprocessing

Before passing packets to Snort detection engine, enabled preprocessors look for abnormal activities. For instance, the "frag3" preprocessor issues alerts on abnormal IP fragments that can be exploited by attackers to evade IDSs, while the "stream4" preprocessor generates alerts on detected stealth portscans and IDS evasion as well [Roesch05].

Detecting

In this phase, packets are matched against signatures figured in Snort rules. Once a signature of an attack is found in a packet, an action is performed according to the first word that appears in the matched rule's header, which is commonly "alert". The issued alert is then logged in an output alert file, with the "priority" number and the "classification" type that exist in the matched rule. The alerts issued by Snort decoding and preprocessing phases are also logged in the same output alert file, but mostly without any assigned priority number and classification type.

The alerts triggered due to matched rules are mainly assigned one of three priority numbers, where 1 is high, 2 is medium, and 3 is low. The three assigned priority numbers are based on 33 classification attack groups figured in version 2.4.1 of Snort [Roesch05]. Tables 4.1, 4.2, and 4.3 detail the attack classes with their assigned priority.

Table 4.1: Snort Classes of Attacks having a default priority of 1.

Snort Class Number	Snort Class Description	Snort Default Priority
1	Attempted User Privilege Gain	1
2	Unsuccessful User Privilege Gain	1
3	Successful User Privilege Gain	1
4	Attempted Administrator Privilege Gain	1
5	Successful Administrator Privilege Gain	1
6	Executable Code was Detected	1
7	A Network Trojan was Detected	1
8	Web Application Attack	1
9	Porn	1
10	Potential Corporate Privacy Violation	1

False Positives

Some rules in Snort may happen to be too "noisy" by triggering many alerts. If it was determined that these alerts are either completely or partially false positives, Snort can be configured through "thresholding" to limit the number of times a particular alert is logged during a certain time interval. In certain circumstances, event "suppression" can be used in Snort configuration to stop completely any alerts triggered by a specific rule.

4.3 Selection of Metrics

The IDS was viewed theoretically as part of the security penetration tests in tree-based Information Assurance (IA) taxonomy presented in [Seddigh04]. Extending from theory to experimentation, an experimental study (which is part of this thesis) was done on the

Table 4.2: Snort Classes of Attacks having a default priority of 2.

Snort Class Number	Snort Class Description	Snort Default Priority
1	Potentially Bad Unknown Traffic	2
2	Attempted Information Leak	2
3	Limited Information Leak	2
4	Large-scale Information Leak	2
5	Attempted DoS	2
6	Successful DoS	2
7	Detection of a DoS Attack	2
8	Decode of an RPC Query	2
9	A Suspicious Filename was Detected	2
10	An Attempted Login using a Suspicious Username was detected	2
11	Attempt to Login by a Default Username and Password	2
12	A System Call was Detected	2
13	A Client was using an Unusual Port	2
14	Detection of a non-standard Protocol or Event	2
15	Access to a Potentially Vulnerable Web Application	2
16	Misc Attack	2

Table 4.3: Snort Classes of Attacks having a default priority of 3.

Snort Class Number	Snort Class Description	Snort Default Priority
1	Not Suspicious Traffic	3
2	Unknown Traffic	3
3	A Suspicious String was Detected	3
4	Detection of a Network Scan	3
5	Generic Protocol Command Decode	3
6	Generic ICMP Event	3
7	Misc Activity	3

IDS branch of the technical group of the taxonomy, where a few IDS metrics were selected and the effect of their assigned different weights on the security evaluation was analyzed [El-Hassan06].

In an attempt to select metrics from Snort output alerts, some criteria is used. However, regardless of the chosen criteria, Snort is assumed to have been well-tuned and the alerts produced by Snort were due to the configuration deemed acceptable by decision-makers of the network. The Snort output alerts are grouped based on either one of three parameters: Priority, Protocol, or Classification. In the following sections, the benefits of grouping alerts based on each parameter are explored.

4.3.1 Priority-based Metrics

As discussed earlier, alerts are assigned one of three priority numbers according to the corresponding class of suspicious activities as shown in Tables 4.1, 4.2, and 4.3. Therefore, selecting metrics as the number of alerts based on priority seems a simple and obvious choice. Then, a network security evaluator - operating in the IDS branch of tree metrics taxonomy [Seddigh04, Nandy04] - would extract these three priority metrics from the logs of Snort, combine the metrics together in order to result an overall intrusion index. Design and implementation of the prototype security evaluator is discussed in Chapter 5 of this thesis, and experimentation with these three metrics - illustrated in Table 4.4 - is detailed in Chapter 6.

However, some alerts are not assigned by Snort any priority number if they are mostly flagged in the decoding and preprocessing phases described in section 4.2. These "unclassified" alerts are eventually not included in the evaluation in the case of priority-based metrics.

Table 4.4: Selected Snort Metrics based on Priority.

Number	Metric	Relative Weight
1	Number of Snort alerts with Priority 1	0.5
2	Number of Snort alerts with Priority 2	0.3
3	Number of Snort alerts with Priority 3	0.2

4.3.2 Protocol-based Metrics

All alerts are triggered by Snort on suspicious packets mainly in network and transport layers. When any of the Snort rules is matched, an alert is triggered carrying the rule's classification group, priority number, and protocol type. When preprocessors detect in the packets what needs to be flagged, the corresponding alerts carry only a protocol type. Thus, extracting metrics based on alerts' protocol type allows the inclusion of all alerts - having the same protocol type - that were triggered in different phases. In all Snort alerts raised on WIDE traces, four types of protocol exist in almost all packets on which alerts are issued. These protocol types are: TCP, UDP, ICMP, and IP. Then, four protocol-based metrics are sought as Table 4.5 shows. These metrics may be ranked differently. However, a simple way of ranking them would treat TCP alerts as more important than UDP and ICMP alerts as shown in Table 4.5.

Table 4.5: Selected Snort Metrics based on Protocol Type.

Number	Metric	Relative Weight
1	Number of Snort TCP alerts	0.5
2	Number of Snort UDP alerts	0.3
3	Number of Snort ICMP alerts	0.1
4	Number of Snort IP alerts	0.1

4.3.3 Classification-attacks-based Metrics

A third set of metrics can be selected based on analysis of Snort classifications figured in Tables 4.1, 4.2, and 4.3. For example, in Table 4.2, class number 5 that consists of

Attempted Denial of Service (DoS) and class number 6 that consists of Successful DoS are both assigned by Snort a priority of 2. The alerts triggered on matched signatures of successful DoS should be viewed as the most important. Therefore, it is placed in the highest rank in Table 4.6. Another example in Table 4.2 is class number 1 that consists of Potentially Bad Unknown Traffic. The alerts of this class may be caused by a network misconfiguration that an attacker may exploit. Thus, ranking this class as second in importance is possible, as in Table 4.6. However, the Attempted DoS would take the third rank because it is not successful even though it must draw attention. Finally miscellaneous activities get the fourth rank.

Table 4.6: Selected Snort Metrics based on Classification.

Number	Metric	Relative Weight
1	Num. of Successful attacks alerts	0.5
2	Num. of Potentially Bad Traffic alerts	0.3
3	Num. of Attempted attacks alerts	0.1
4	Num. of Misc attacks alerts	0.1

4.4 Ranking of Metrics

The ranking order of the selected metrics, whether they are based on priority, protocol, or classification, is intuitive but not definite. The weights given to the metrics are assigned according to their relative importance in a way similar to the "swing-weighting approach"

described in [Clemen01]. This weighting approach rates each enterprise objective according to its relative importance to decision-makers and infers a relative weight for each objective. In the experiments done on priority-based metrics in Chapter 6 of this thesis, the relative weights are 0.6, 0.3, and 0.1 for Priority 1, 2, and 3 metrics respectively. In the experiments done on all three sets of metrics in Chapter 7, the relative weights of the highest-ranked and second-ranked metrics are 0.5 and 0.3 respectively in each set of metrics, while lower-ranked metrics are assumed to get equal relative weights (Tables 4.4,4.5,4.6).

4.4.1 Examples of Snort Alerts

The structure of most flagged Snort alerts is similar. Examples of alerts having a priority of 1, 2, and 3 are detailed below.

This is an example of a priority 1 alert:

"06/21-00:09:25.004085 [**] [1:2182:8] BACKDOOR typot trojan traffic [**] [Classification: A Network Trojan was detected] [Priority: 1] TCP 181.101.166.75:35437 -> 166.69.215.204:64896"

The alert reads that a trojan was detected. The Snort rule that causes this alert is one of the backdoor rules stored in Snort database. Backdoor alerts are assigned by Snort a default priority of 1 (highest) to emphasize the importance of the suspicious matter found in the corresponding packet. The rule itself as stored in the database is as follows:

"alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"BACKDOOR typot trojan traffic"; flow:stateless; flags:S,12; window:55808; reference:mcafee,100406; classtype:trojan-activity; sid:2182; rev:8;)"

Coordinating the rule with the triggered alert, one understands that a backdoor "typot" trojan was detected due to some conditions including a window size of 55808. The rule provides "McAfee" as a reference to the suspicious matter. Investigating more this trojan in the web site of McAfee[1], one finds that the trojan may run only on Linux where it decrypts itself using a "password passed to it via the command line." Then, it sends out a SYN packet every second with a window size of 55808 bytes, providing that these SYN packets are sent to "randomly generated IP addresses with spoofed randomly generated source addresses" with randomly generated port numbers. At the end of the description, McAfee makes the point that "the exact purpose of this trojan is not really known but could conceivably be part of an attempt to 'map' systems with Internet connectivity."

McAfee provides a "low" risk assessment to this trojan even though it has a suspicious behavior and its exact purpose is unknown. However, Snort assigns a high priority for this trojan stating that even though "any application that generates a TCP SYN packet with a window size of 55808 bytes" will trigger this alert, there is no known deployed application generating these specific packets[2]. Thus, it is very useful to admit a high threat level

[1]http://vil.nai.com/vil/content/v_100406.htm
[2]http://www.snort.org/pub-bin/sigs.cgi?sid=1-2182

assessment for a similar SYN packet in the case of a network having Linux in some of its workstations. The threat level may be changed for enterprises that do not use Linux at all.

Notice that the protocol type indicated in the alert is TCP. In Table 4.5, TCP alerts are ranked the highest. Based on classification, since the alert is classified by Snort in a category of trojan detection, its threat is ranked the highest as well in Table 4.6 and considered in the category of "successful" attacks. The term "successful" is given to illustrate the threat from external point of view. Whether it is really successful depends on the analysis that should be done to the network.

This is an example of a priority 2 alert:

"06/21-00:08:06.833069 [**] [1:3626:1] ICMP PATH MTU denial of service [**] [Classification: Attempted Denial of Service] [Priority: 2] ICMP 163.122.233.66 -> 219.133.112.86"

From the alert body, one can see that some suspicious activity that may lead to a DoS was detected. The Snort rule that causes this alert states as follows:

"alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP PATH MTU denial of service"; itype:3; icode:4; byte_test:2,<,576,1; reference:cve,2004-1060; classtype:attempted-dos; sid:3626; rev:1;) sid-msg.map:3626 | | ICMP PATH MTU denial of service | | cve,2004-1060"

This alert is generated when an ICMP path MTU message (spoofed) is received asking to fragment packets to sizes smaller than 576 bytes. This might lead to a DoS if the receiving host responds by sending unnecessarily small packets to destination hosts[3].

Snort assigns a priority of 2 to this alert stating that the host should be patched to ignore responses with packets smaller than 576 bytes. Based on classification, this is considered an attempted attack which is ranked third in Table 4.6 because it is not successful as long as the receiving host has not responded with packets smaller than 576 bytes. The protocol type in the alert is ICMP, which is ranked third in Table 4.5.

This is an example of a priority 3 alert:

"06/21-00:08:06.833069 [**] [1:396:6] ICMP Destination Unreachable Fragmentation Needed and DF bit was set [**] [Classification: Misc activity] [Priority: 3] ICMP 163.122.233.66 -> 219.133.112.86"

The alert was caused by the following Snort rule:

"alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP Destination Unreachable Fragmentation Needed and DF bit was set"; icode:4; itype:3; classtype:misc-activity; sid:396; rev:6;) sid-msg.map:396 | | ICMP Destination Unreachable Fragmentation Needed and DF bit was set"

The alert is generated when an "ICMP Destination Unreachable Fragmentation

[3]http://www.snort.org/pub-bin/sigs.cgi?sid=1-3626

Needed" packet is detected, while the Don't Fragment (DF) bit was set. Snort states that "gateway devices normally generate these ICMP messages when the destination network requires fragmentation before the datagram can be forwarded by a gateway" which might be "an indication of improperly configured network hosts"[4]. Snort assigns a priority of 3 to this alert classified as miscellaneous activity. Thus, it is ranked fourth in Table 4.6. In Table 4.5, it is ranked third because its protocol type is ICMP. More information about miscellaneous activity category in Snort exists in Appendix A.

4.5 Weights Adjustment

As mentioned above in Tables 4.4, 4.5, and 4.6, the relative weights shown for the first two metrics in each set are 0.5 and 0.3 respectively. The remaining metrics in each set get equal relative weights, providing that the sum of all relative weights in each set of metrics is 1. Assuming that these relative weights are assigned to IDS metrics, I examine through an example how analysis of incidents over a long period of time would lead to adjusting the assigned weights (see Figure 3.5).

4.5.1 Example

As an example, let's assume that the number of total incidents occurred in a network during 2005 is 100, distributed as follows:

[4]http://www.snort.org/pub-bin/sigs.cgi?sid=1-396

- 30% of incidents were due to external activities alerted on by the IDS (S1 in Figure 3.5).
- 10% of incidents were due to false negatives (S2 in Figure 3.5).
- 40% of incidents were due to internal activities (S3 in Figure 3.5).
- 20% of incidents were due to unknown activities (S4 in Figure 3.5).

The number of incidents that were due to external activities alerted on by the IDS (i.e. S1) will then be 30. As stated in section 3.3, only the metric S1 must be analyzed to adjust the IDS metrics. If further analysis of S1 metric shows that 18 incidents were due to packets that cause alerts of priority 1, 9 incidents were due to packets that cause alerts of priority 2, and 3 incidents were due to packets that cause alerts of priority 3, then the relative impact of incidents in S1 metric will be as follows:

- Relative impact of priority 1 alerts in S1 metric incidents is 18 divided by 30, i.e. 0.6.
- Relative impact of priority 2 alerts in S1 metric incidents is 9 divided by 30, i.e. 0.3.
- Relative impact of priority 3 alerts in S1 metric incidents is 3 divided by 30, i.e. 0.1.

Considering the incident impact of each IDS metric as equivalent to its relative weight, the assigned weights would then be adjusted accordingly. Thus, the priority 1 metric would get a relative weight of 0.6 instead of 0.5 figured in Table 4.4, while priority 3 metric relative weight would become 0.1 instead of 0.2. Since the impact of priority 2 in S1 incidents is still 0.3, there will be no adjustment for priority 2 relative weight. Table 4.7

summarizes the adjustment process.

Table 4.7: Adjustment of Weights of Snort Priority Metrics.

	S1 incidents	IDS Metric	Relative Impact	Adjusted Weight
	18	Priority 1 Alerts	18 / 30 = 0.6	0.6 instead of 0.5
	9	Priority 2 Alerts	9 / 30 = 0.3	0.3 instead of 0.3
	3	Priority 3 Alerts	3 / 30 = 0.1	0.1 instead of 0.2
Total	30	Priority Metrics	30 / 30 = 1.0	1.0

This same example can be analyzed alternatively with a "pairwise comparison matrix" as detailed in the Analytic Hierarchy Process (AHP) [Saaty89] (or its generalized form: the Analytic Network Process (ANP) [Saaty06]). In such a matrix, elements are compared in a pairwise way according to each element relevant impact, then normalized to calculate relative weights. The matrix elements a_{ij} are then represented in a way that satisfies the relation $a_{ji} = \frac{1}{a_{ij}}$, where i is the row number and j the column number.

Table 4.8: Adjustment of Weights with the Pairwise Comparison Matrix of AHP before Normalization.

	SP1	SP2	SP3
SP1	$\frac{18}{18} = 1$	$\frac{18}{9} = 2$	$\frac{18}{3} = 6$
SP2	$\frac{9}{18} = \frac{1}{2}$	$\frac{9}{9} = 1$	$\frac{9}{3} = 3$
SP3	$\frac{3}{18} = \frac{1}{6}$	$\frac{3}{9} = \frac{1}{3}$	$\frac{3}{3} = 1$
Total	$\frac{5}{3}$	$\frac{10}{3}$	**10**

Table 4.8 illustrates the corresponding comparison matrix, where:

- SP1 = S1 incidents due to Priority 1 Alerts,

- SP2 = S1 incidents due to Priority 2 Alerts,
- SP3 = S1 incidents due to Priority 3 Alerts.

In the AHP matrix, elements are further normalized for each column, as depicted in Table 4.9. The relative weights for SP1, SP2, and SP3 are 0.6, 0.3, and 0.1 respectively. In this example, the normalized weights for each column are consistent. In the cases that the normalized weights for each column are not consistent, the average values across the rows will be the relative weights.

Table 4.9: Adjustment of Weights with the Pairwise Comparison Matrix of AHP after Normalization.

	Normalized 1st Column	**Normalized 2nd Column**	**Normalized 3rd Column**
SP1	$\frac{1}{\frac{5}{3}} = 0.6$	$\frac{2}{\frac{10}{3}} = 0.6$	$\frac{6}{10} = 0.6$
SP2	$\frac{\frac{2}{5}}{\frac{5}{3}} = 0.3$	$\frac{1}{\frac{10}{3}} = 0.3$	$\frac{3}{10} = 0.3$
SP3	$\frac{\frac{2}{6}}{\frac{5}{3}} = 0.1$	$\frac{\frac{1}{3}}{\frac{10}{3}} = 0.1$	$\frac{1}{10} = 0.1$

Chapter 5

Prototype and Experimental Evaluation

In this chapter, the applicability of HQM and the corresponding setup needed to conduct experiments are demonstrated. The following items are detailed next:

- Design and implementation of a prototype security evaluator,
- Use of real network traffic traces,
- Setup needed for experiments.

5.1 Prototype Evaluator

A Prototype Evaluator is used to study the effect of selected IDS metrics on the HQM model, evaluating network security. Basically, the prototype is required to extract the metrics intended for evaluation, normalize and aggregate them based on three parameters, organize them into three sets, and report an overall IDM for each set to a web server. The three sets of metrics, and their parameters, were discussed in Chapter 4. The prototype

is designed to be automated to operate in an industry-like fashion. Next, the evaluator's architecture and components are described.

5.1.1 Architecture

The current role of the Prototype Evaluator is to produce the IDM in one node of the tree. The architecture of the engine is designed to accept additional parsers from other branches of the tree, to be able ultimately to produce an overall IAM to the entire tree i.e. to depict the security health of the network from all aspects designated in the metric tree. Figure 5.1 illustrates the generalized architecture of the IAM engine.

The N additional parsers shown in Figure 5.1 are included for future development to extract metrics from other leaves and nodes of the tree. For instance, one of these additional parsers gets metrics to produce a vulnerability index from the output scans performed by a vulnerability scanner. The vulnerability index is located at the same level of the IDM as illustrated in the HQM model (Figure 3.2).

At the core of this evaluator is an engine that communicates with an IDS parser component that extracts the few selected metrics from the logs of the Snort network-based IDS [Roesch99]. Combining those metrics together depicts the IDM. As more parsers can be added (to extract more metrics), the architecture of the engine in its generalized form is illustrated in Figure 5.1.

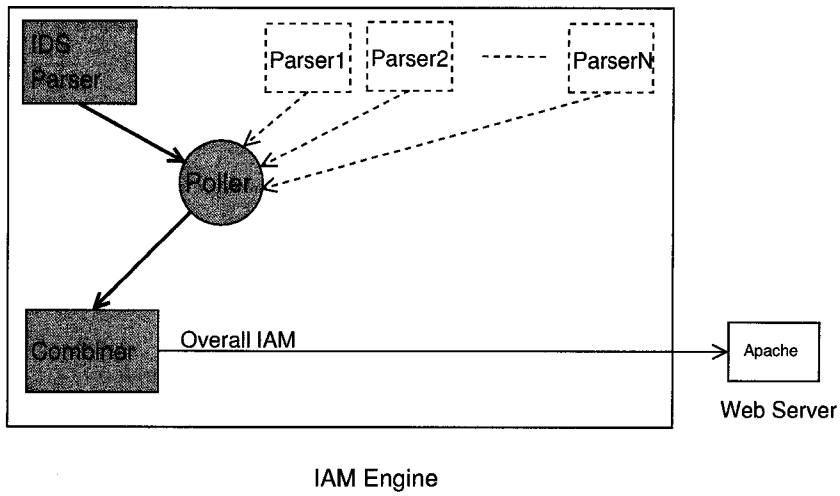


Figure 5.1: Generalized Architecture of the IAM Engine. (The shaded areas are implemented)

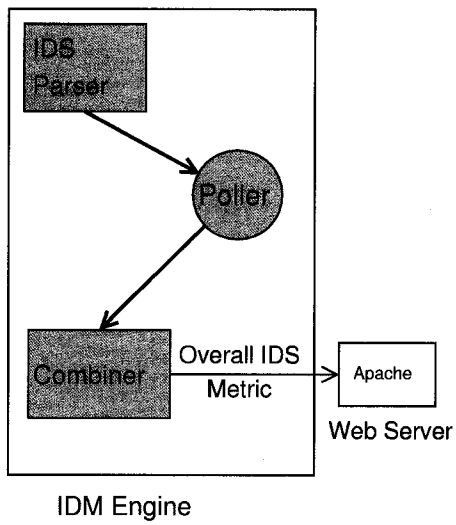


Figure 5.2: The IDM Engine as a Special Case of the Generalized IAM Engine.

The current implementation of the engine consists of the following (Figure 5.2 or shaded areas of Figure 5.1):

- A parser to extract metrics from the IDS outputs.

- A poller that polls the metrics parsed by the parser.
- A combiner that applies preassigned weights to all metrics, according to the computed relative weights, combines them according to Equation 3.1 (see section 3.2.1), and outputs an overall intrusion detection metric (i.e. IDM). The IDM and the metrics are reported to a web server, which allows system administrators and managers to remotely view the results of the engine.

Parser

The implemented IDS Parser in Figure 5.1 extracts selected Snort metrics from the alert file of Snort, and stores them in a file to be polled by the Poller of the engine. The alert Snort file contains all alerts triggered on incoming packets carrying suspicious activities. The extracted Snort metrics are mentioned earlier in Tables 4.4, 4.5, and 4.6 (see section 4.3), and based on three different parameters: Priority, Protocol type, and Classification of attacks. In order to behave efficiently, the parser is installed in the same machine the IDS Snort operates.

Poller

The poller is designed to communicate with the machine where the IDS parser operates to get the extracted metrics. For automation purposes, the poller periodically communicates through a TCP socket with the IDS parser. Therefore, in the parser side a "listener" program runs to be ready to receive data through the socket on a specific port. In the poller program, installed in a machine in the network, the IDS machine identity and port are hard-coded.

Combiner

Once the Poller obtains the parsed metrics from the parser, it sends this data to the Combiner located with the poller in the same machine. The Combiner treats data in the following steps (Figure 5.2):

- The received parsed metrics are normalized according to the description stated in section 3.2.
- The normalized metrics are multiplied with their assigned relative weights stored in a configuration file.
- The IDM is computed according to Equation 3.1 (section 4.3).
- The IDM is reported to the web server. This "intrusion index" can thus be viewed using any standard web browser.

5.1.2 Implementation Issues

Different components of the prototype including the IDS parser and the engine are implemented in C++ in Linux environment. There was no particular reason for having chosen C++ in the implementation, other than it is one of the most popular languages of development. The web server used to view the results is Apache, which is available for free under Linux.

Several implementation issues were raised. For example, a main issue consisted of

dealing with large files. Since Snort may generate hundreds of thousands of alerts every day, the output Snort alert file - from which metrics are extracted - can become so large quickly. There is an option in Snort to run without logging alerts into a text file. However, the textual output alert file is needed to consult the type of the generated alerts and to extract metrics from it. Therefore, in order to deal effectively with a large file, the prototype program is designed to break the output Snort alert file into as many as 64 files. After parsing metrics from the split files, the implemented program rearranges the parsed metrics based on each parameter under the day on which alerts were triggered. In other words, the metrics are arranged time-wise in each day. The date and time information exist in the beginning of the body of each flagged alert as shown in the alert-example below:

"06/21-00:08:06.884286 [**] [1:3626:1] ICMP PATH MTU denial of service [**] [Classification: Attempted Denial of Service] [Priority: 2] ICMP 203.16.240.178 -> 6.177.16.4"

To effectively implement the automation of the tool, and report the results to a web server, shell scripts were used. The web interface itself has been developed using PHP.

Next, a context for the experiments that were executed is described.

5.2 Challenges in Experimental Setup

Two different strategies were available for experiments to test the Evaluator as part of the HQM Model. The first strategy was to obtain a set of publicly available network traces

and use that as input to the Evaluator. The second strategy was to connect the Evaluator to a live network. Both strategies have merits in any experimentation. In the actual experiments, the first strategy is pursued as it allowed access to much larger data sets than with the second strategy. The second strategy is more "real-world," but it was not possible to gain access to a real network for administrative and privacy concerns. Thus, the online live network experiments were left for future endeavors. However, the architecture of the prototype evaluator was designed to operate using either strategy.

Snort has multiple options available for getting its input. In the most obvious model, it can be connected to a live network. In another one of its configuration options, it can read traffic traces stored in tcpdump format [Roesch05]. Using the latter approach, the prototype security Evaluator and Snort are effectively utilized for the first strategy mentioned above, where Snort reads WIDE network traces and outputs alerts upon identification of suspicious activities embedded in the traces.

Snort version 2.4.1 [Roesch05] was utilized for this work, where a total of 3191 Snort security rules were present in its database as of this version. In some experiments, all the rules were enabled while in other experiments, only a subset of the rules was enabled.

5.3 Selection and Preparation of Real Network Traces

After careful evaluation, we selected traffic traces from the WIDE[1] Network as the basis for our experiments. These traffic traces are publicly available and can be downloaded from the MAWI[2] Working Group Traffic Archive [wid05]. The WIDE Network is a nation-wide R&D network in Japan with sufficiently large user traffic for the desired experiments. Currently, publicly available WIDE traffic traces are collected at five sampling points. Due to the volume of data, WIDE provides traffic traces for a brief period every day, typically a 15-minute period starting at 2pm in the afternoon. We utilize traffic traces from sampling point-B (18Mbps CAR link) for our experiments. The traces are published in the well-known tcpdump format. A 15-minute sample of network traffic from the sampled WIDE network link typically includes anywhere between 3 to 6 million packets for hundreds of thousands of unique five-tuple flows. Tables 5.1 and 5.2 provide statistical information for sample traces collected in the first five days of January 2005. Most of the sample traces used in our experiments resemble the profile of the traces in both tables.

Obtaining large traffic traces for implementation is a difficult task. The WIDE Traces were a suitable choice for this experimental study despite a number of concerns:

- Scrambling - Public traffic traces typically scramble header information in order to

[1]Widely Integrated Distributed Environment (WIDE) project, available at: http://www.wide.ad.jp/about/foreword.html

[2]Measurement and Analysis on the WIDE Internet (MAWI)

Table 5.1: Time Information of WIDE Traces collected in the first five days of January 2005.

	Collection Date	**Collection Start Time**	**Collection End Time**	**Duration**
Data Set 1	01/01/05	02:00:00 pm	02:15:00 pm	15:00 mn
Data Set 2	01/02/05	02:00:00 pm	02:15:00 pm	15:00 mn
Data Set 3	01/03/05	02:00:00 pm	02:15:00 pm	15:00 mn
Data Set 4	01/04/05	02:00:00 pm	02:15:00 pm	15:00 mn
Data Set 5	01/05/05	02:00:01 pm	02:15:00 pm	14:59 mn

Table 5.2: Statistical Information of WIDE Traces collected in the first five days of January 2005.

	Collection Date	**Av. Link Speed**	**Nb. of Flows**	**Nb. of Packets**
Data Set 1	01/01/05	13.89Mbps	279098	3103414
Data Set 2	01/02/05	13.34Mbps	311859	3146322
Data Set 3	01/03/05	18.00Mbps	394034	4633664
Data Set 4	01/04/05	19.26Mbps	421027	5115338
Data Set 5	01/05/05	19.48Mbps	373785	5169218

anonymize their data - in some cases losing any pre-existing flow relationships between packets. Scrambling of the WIDE traces is done in such a way that all occurrences of an IP address were mapped to a new single IP address within the s ame daily trace. As a result, flow relationships are maintained.

- Data Volume - The large-volume WIDE traces were collected for 15 minutes daily. While traces for such a short duration do not reflect the traffic status of the network during the whole day, its impact is mitigated due to the fact that they are largely

diversified and collected from a Trans-Pacific link during regular office hours.

- Missing payload - The packet payloads of the traces were removed for reasons related to privacy protection. In order to ensure that the Snort decoder does not raise false alerts on truncated packets, the decoder alerts were disabled during the experiments as were as some Snort preprocessors.

- Missing data sets - The WIDE traces for some days in 2005 are missing, especially between August 21 and September 12 2005. The total number of days missing in the traces is 26 out of 365 days. Thus, the percentage of 2005 data sets used in experiments is 93%, which is still very useful in concluding meaningful results.

5.4 Network Setup

Snort runs on network WIDE traces stored in tcpdump format. In order to study metrics' weights sensitivity, it is enough to extract a set of metrics based on their default priority. The priority metrics are to be extracted then by the parser of the prototype Security Evaluator and combined by the prototype combiner. The resulting overall intrusion metric IDM is reported to an Apache web server. The corresponding network setup is illustrated in Figure 5.3.

This network setup is used in the experiments of Chapter 6. Slightly modifying this setup, the parser is expanded to extract three sets of metrics based on either priority, protocol type, or classification. Each set of metrics is then combined separately by the Security

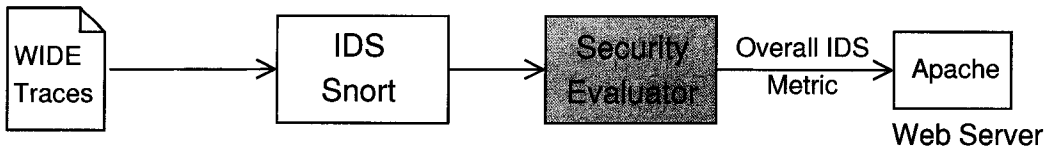


Figure 5.3: First Experimental Setup.

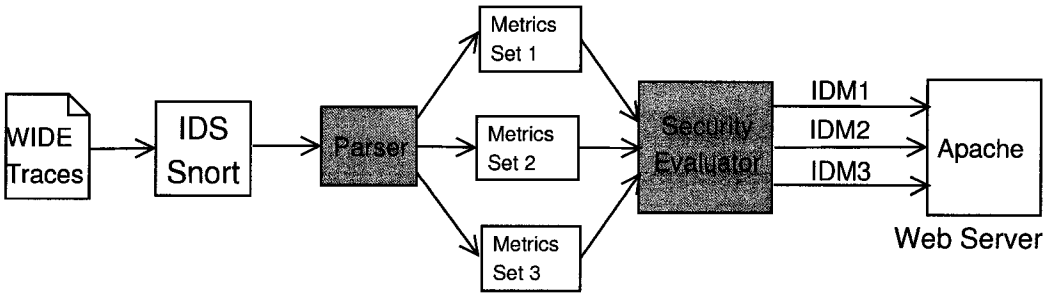


Figure 5.4: Second Experimental Setup.

Evaluator, as illustrated in the second experimental setup of Figure 5.4. The resulting overall intrusion metric IDM1 represents the combined set of metrics based on priority, IDM2 represents the combined set of metrics based on protocol, and IDM3 represents the combined set of metrics based on classification. This network setup is to be used in the experiments described in Chapter 7.

Chapter 6

Calculation of the IDM using Priority Metrics

The ultimate purpose of the experiments is to study the use of the proposed HQM model and the IDM as a security threat indicator over a long period of time. This experimentally-delivered IDM would have to contribute in the future with other metrics at different nodes of a generalized metric tree, such as the taxonomy of [Seddigh04, Nandy04], to produce at the top of the tree an overall IAM to depict a comprehensive evaluation of the network security.

All the experiments in this chapter and the next one utilize the two experimental setups described in Chapter 5 and illustrated in Figure 5.3 and Figure 5.4. WIDE network traces collected in 2005 were utilized as inputs to Snort. The number of packets and flows in each daily trace resembles the sample traces shown in Table 5.1 and Table 5.2. For example, on January 3rd the number of packets was more than 4.6 million, and the number of flows was slightly less than 4 million.

In this chapter, the sensitivity of metrics' weights and Snort rules is studied. It is enough to use one set of metrics to test the impact of weights and Snort rules on metrics. Therefore, the network setup illustrated in Figure 5.3, where metrics are extracted based on their default priority, is the considered setup in all experiments of this chapter. The metrics extracted from Snort, chosen as described in section 4.3.1, are the three metrics stated in Table 4.4 - namely Priority 1, Priority 2, and Priority 3 alerts. A supposed fourth metric, number of priority 4 (and greater) alerts, is not included because no alerts at all having this priority were flagged.

In the next experiments, two key issues are examined:

- The effect of weights on the IDM when traces are collected in one year.
- The effect of Snort rules on the IDM.

Before running experiments, Snort should be configured and tuned to avoid as much as possible false positives. Even though avoiding false positives is not focused on in this research, some configuration steps must be done to avoid unnecessary alerts on WIDE traces whose packets' payloads were removed. Thus, Snort decoder and some preprocessors were disabled.

6.1 Production of Normalized Metrics

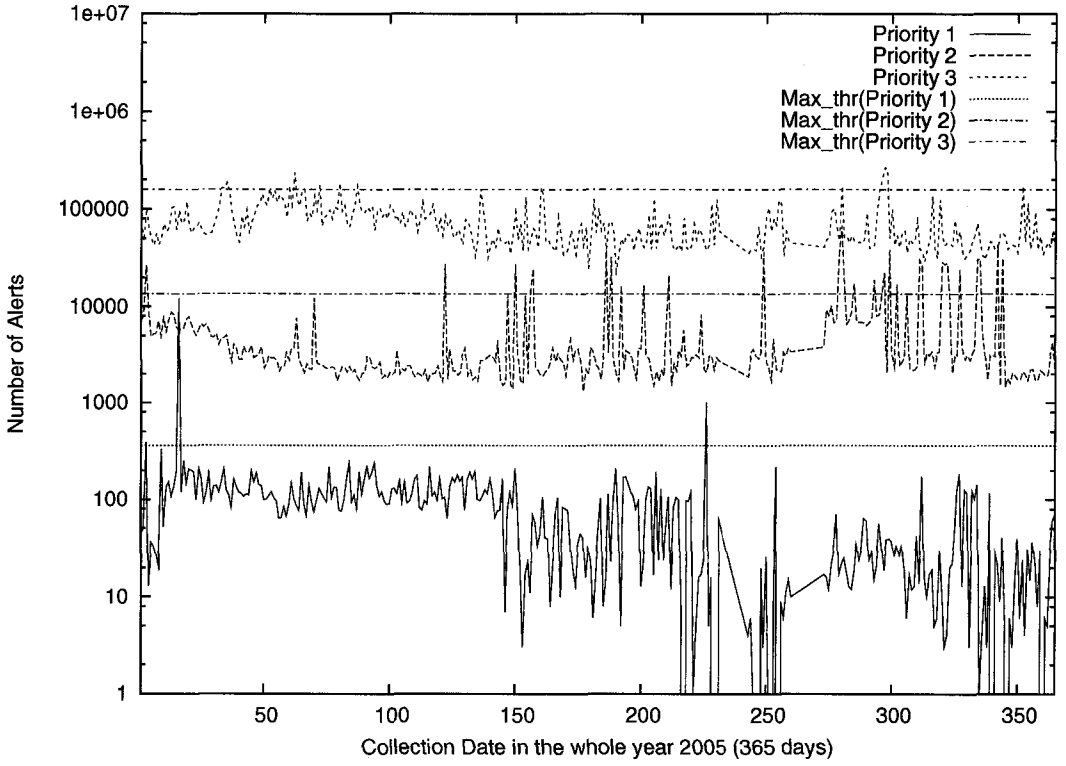
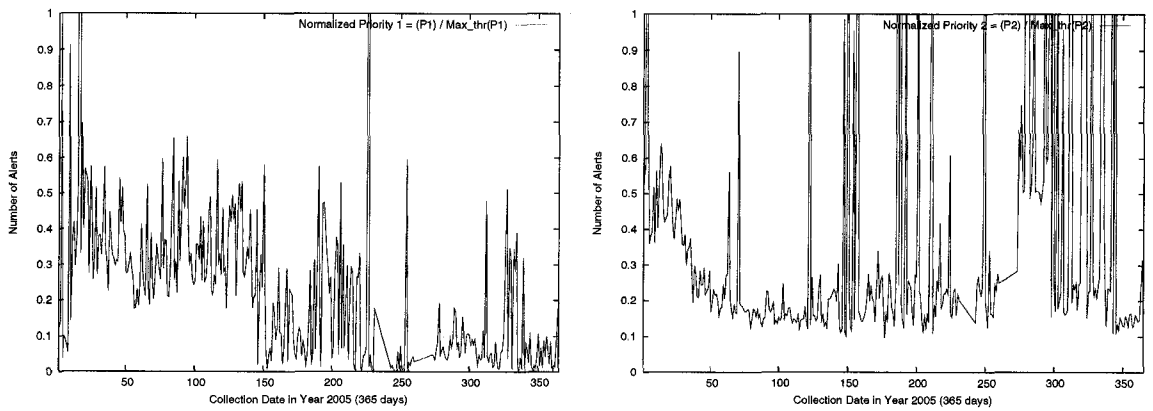


Figure 6.1: Priority 1, 2, and 3 Snort Alerts flagged on all 2005 WIDE Traces.

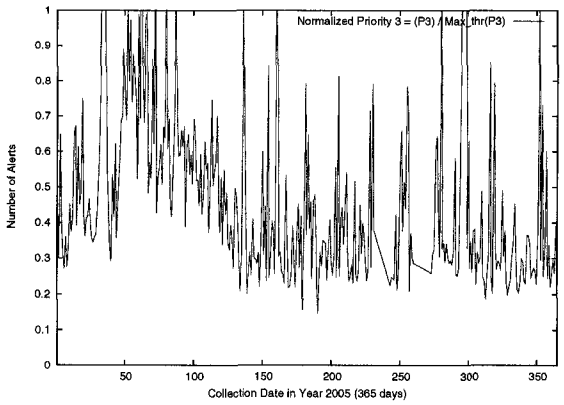
A graphical view of the three Snort metrics, as parsed and counted by the parser component of the evaluator, is shown in Figure 6.1 with log-scale y-axis. The same metrics are illustrated in Figure 6.2 after they are normalized according to the normalization approach described in section 3.2.

The Priority 1 metric fluctuates between 0 and 12116 alerts. The Priority 2 metric fluctuates between 1323 and 68013 alerts. The Priority 3 metric fluctuates between 23014 and 262622 alerts. Whether many of these alerts are false positives or not is something



(a) Priority 1

(b) Priority 2



(c) Priority 3

Figure 6.2: Normalized Metrics based on 'Priority' for all 2005 WIDE Traces.

to be analyzed by the network's security experts who may tune Snort rules to avoid false positives. In addition network monitoring tools would help identify false positives. The role of the IDM evaluator is to evaluate the threat as reported by the IDS. As shown in Figure 6.1, the normalization threshold (Max_thr(*Metric*)) for each metric is drawn as a horizontal line. For example, the Max_thr(Priority 1) value is 242.32 which is 20% of the maximum number of Priority 1 alerts. More than 90% of Priority 1 values fall below that

threshold.

6.2 Effect of Weights

As mentioned in section 3.2, as part of the IDM and the generalized IAM calculations, weights are applied against the normalized metrics. Clearly then, the choice of weights affects the IDM value. In the following three experiments, the purpose is to understand how much impact the weights might have in determining the IDM value for the large traffic traces used in the experimentations.

Similar to the "swing-weighting approach", mentioned in section 3.2, the Snort metrics are assigned weights first according to their intuitive relative importance. Then, the first set of weights 0.6, 0.3, and 0.1 is selected to be tested against the metrics Priority 1, Priority 2, and Priority 3 respectively. Since the relative importance of these metrics can change according to the objectives of decision-makers, two additional sets of weights are tested assuming they were produced by a method similar to the swing-weighting approach. Table 6.1 shows the three sets of weights to be applied on the metrics to produce the corresponding IDM for each set.

6.2.1 First Experiment

The first test utilizes the most-intuitive set of weights, where the objectives of the decision-makers are to rank metrics according to their most obvious relative importance.

Table 6.1: Assigned Weights to Snort Metrics.

Number	Metric	Set1 of Weights	Set2 of Weights	Set3 of Weights
1	Priority 1	0.6	0.3	0.1
2	Priority 2	0.3	0.6	0.3
3	Priority 3	0.1	0.1	0.6

These weights are labeled Set1 in Table 6.1.

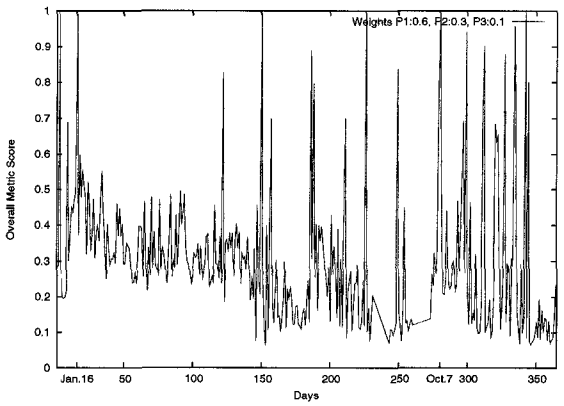
The results of the experiment are captured in Figure 6.3-a. The results show that the network faces generally moderate external threats. However, the high IDM score computed for a few days of the year is a cause of concern. Detailed investigation of the packet traces for those few days revealed some interesting results. For example, January 16th yielded the maximum number of Priority 1 alerts recorded during the whole year, while on October 7th Snort raised the maximum number of Priority 2 alerts for 2005.

6.2.2 Second Experiment

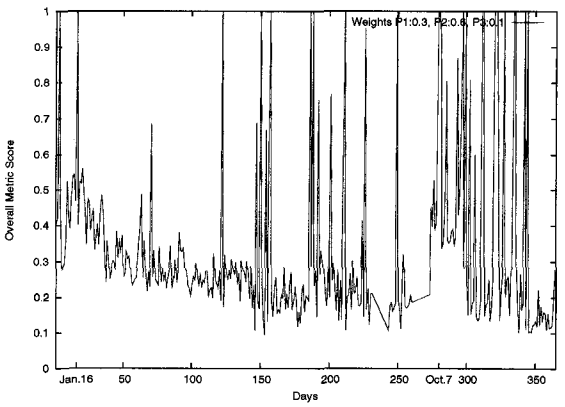
The weights under this test are labeled Set2 in Table 6.1. The rank order of the metrics in this test indicates that Priority 2 metric is viewed by decision-makers as the most important relatively.

The results of this experiment are depicted in Figure 6.3-b. The test results clearly indicate more variations of external threat than do the first test results. A high IDM score

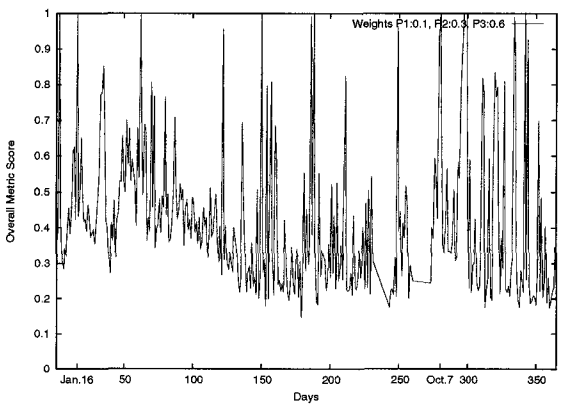
was again computed for January 16th and October 7th, as was the case in the first test, but more than 20 additional days had very high IDM values as well. In November, for example, there were 7 days with very high IDM values. These high scores would draw attention to more investigating the external activities.



(a) IDM for Set1 of Weights



(b) IDM for Set2 of Weights



(c) IDM for Set3 of Weights

Figure 6.3: Overall IDM for all 2005 WIDE Traces, with Sets of Weights assigned in Table 6.1.

6.2.3 Third Experiment

The third set of weights (labeled Set3 in Table 6.1) is applied against the same WIDE traces. In this set, the Priority 3 metric is considered as being the most important to the decision-makers. In general, throughout the year 2005, the number of Priority 3 alerts flagged by Snort on WIDE traces is higher than the number of Priority 1 and 2 alerts.

The results of this experiment, illustrated in Figure 6.3-c, show higher IDM values than those recorded in the two previous tests for many days in the first three months. Furthermore, most IDM values are greater than 0.2, while in the previous two tests a significant number of IDM values were less than 0.2.

6.2.4 Discussion

To better analyze the IDM results in the three experiments illustrated in Figure 6.3, the following points are discussed.

- The metrics were ranked differently in each of the three experiments to point to the importance of the impact of assigning different weights to metrics. In an enterprise, decision-makers would face similar weights assignment decisions in every aspect of network security, even in non-technical branches of a tree-based information security. Analysis of the impact of risk that may occur from elements of each metric helps in adjusting the weights, as would do the swing-weighting approach.

- After long time of operation, security experts are able to identify most false positives and correctly analyze the impact of risk that would occur from real threats. Consequently, "correct" adjustment of weights depends on how solid false positives identification and risk impact analysis are. An example on weights adjustment after analysis of long-time collected incidents was given in section 4.5.1.

- The IDM results are based on external intrusions identified from Snort running on WIDE traces, regardless of the vulnerabilities that may exist in a specific network. Therefore, if these traces are applied to a highly-vulnerable network, the difference in IDM values resulting from either test of the three experiments would not be considered important. In other words, whatever weights are applied to metrics the IDM value should give decision-makers just a general idea of how much they should invest to strengthen security. Once vulnerability patches are applied to a network, the difference in IDM values becomes important because decision-makers want to evaluate the external threat from different perspectives in the presence of vulnerability patches. A vulnerability index as suggested in the HQM model (Chapter 3) and its vulnerability sub-tree illustrated in Figure 3.4 is certainly very useful.

6.3 Effect of Snort Rules on the IDM

Snort makes use of a variety of pattern-matching algorithms where it inspects incoming packets against signatures of known attacks or suspicious activities stored in the Snort rules database. Packets, upon matching a signature pattern, cause an alert to be generated. With

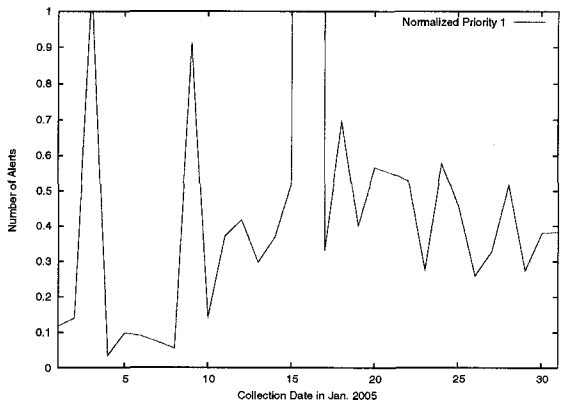
the rapid proliferation of new applications and the dynamic nature of malicious attacks, Snort - like other intrusion detection tools - is susceptible to false positives. Snort allows the administrator to disable certain rules, or to specify new rules which ensure that alerts are not raised for legitimate traffic. To learn the impact of disabling certain rules on an IDM score, new experiments are conducted under new conditions.

6.3.1 Experiments

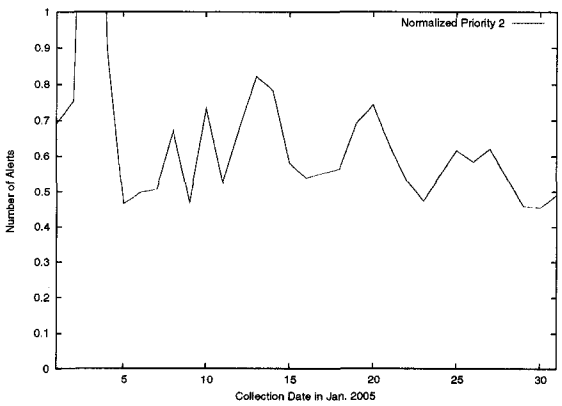
The experiments in this section are conducted on WIDE network traffic traces for January 2005. Before disabling any of Snort rules, observations of priority metrics resulted from WIDE traces indicate that most of the alerts are due to three main categories of Snort rules (i) backdoor, (ii) ICMP, and (iii) ICMP information rules. In Snort, these rule categories are named backdoor.rules, icmp.rules, and icmp-info.rules respectively. Backdoor rules are classified by Snort as Priority 1 alerts. Alerts triggered by ICMP and ICMP information rules are classified as either Priority 2 or 3 alerts. Examples of Snort alerts and rules were given in section 4.4.1. For further illustration of Snort rules, examples related to ICMP information rules are described in Appendix A.

To determine the effect of disabling certain rules on the IDM score, the following two tests are undertaken:

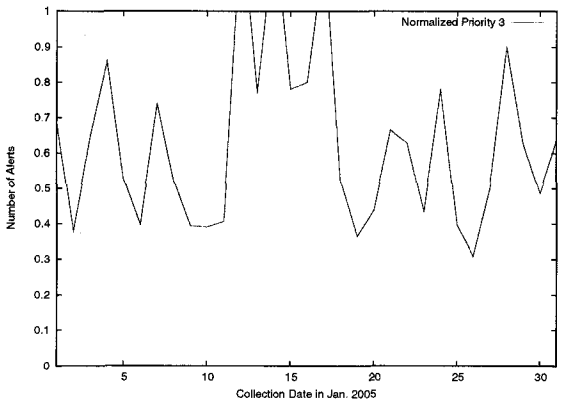
- (a) The "icmp-info.rules" category is disabled in Snort. The corresponding normalized Priority 1, 2, and 3 alerts metrics are shown in Figure 6.4.



(a) Priority 1



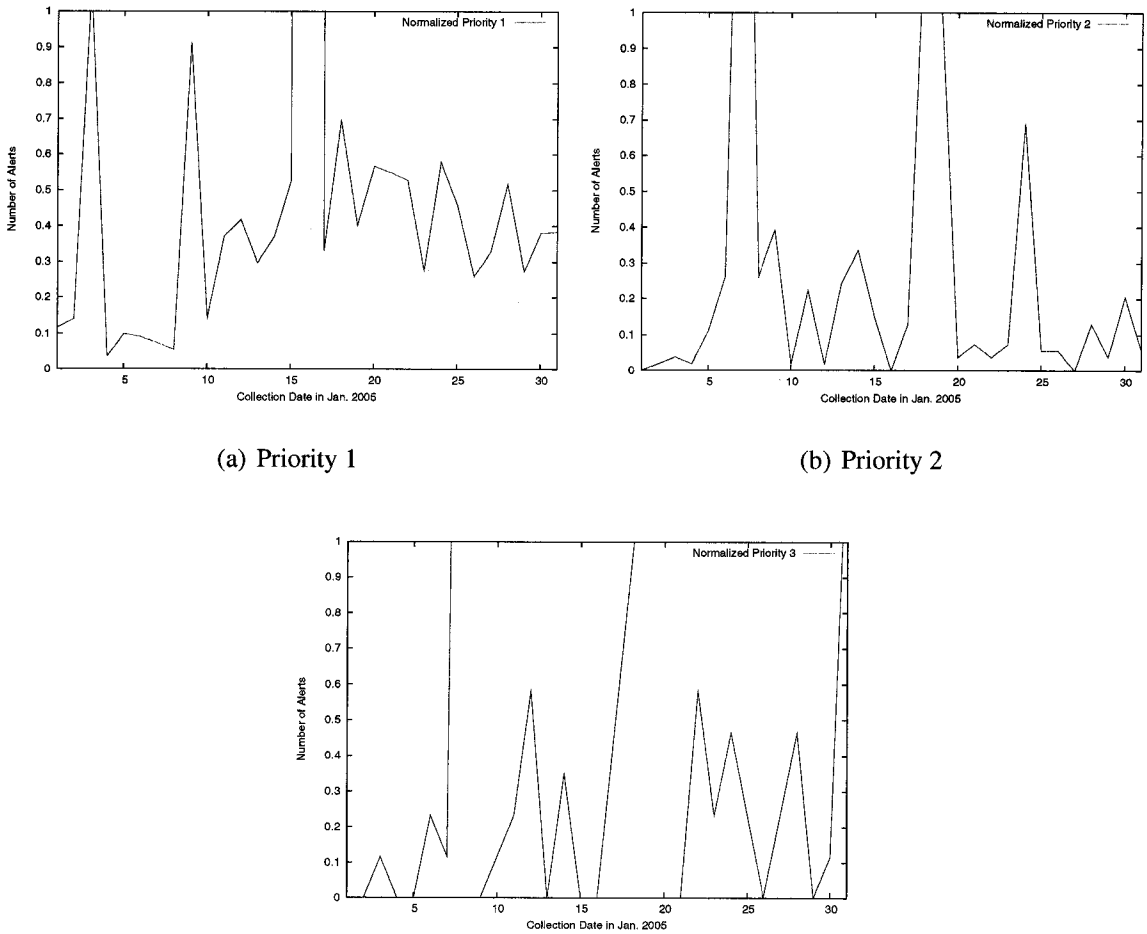
(b) Priority 2



(c) Priority 3

Figure 6.4: Normalized Priority 1, 2, and 3 Snort Alerts flagged, on January 2005 WIDE Traces, after Disabling "icmp-info.rules" Category in Snort Rule Database.

- (b) Both "icmp-info.rules" and "icmp.rules" categories are disabled in Snort. The corresponding normalized Priority 1, 2, and 3 alerts metrics are shown in Figure 6.5.



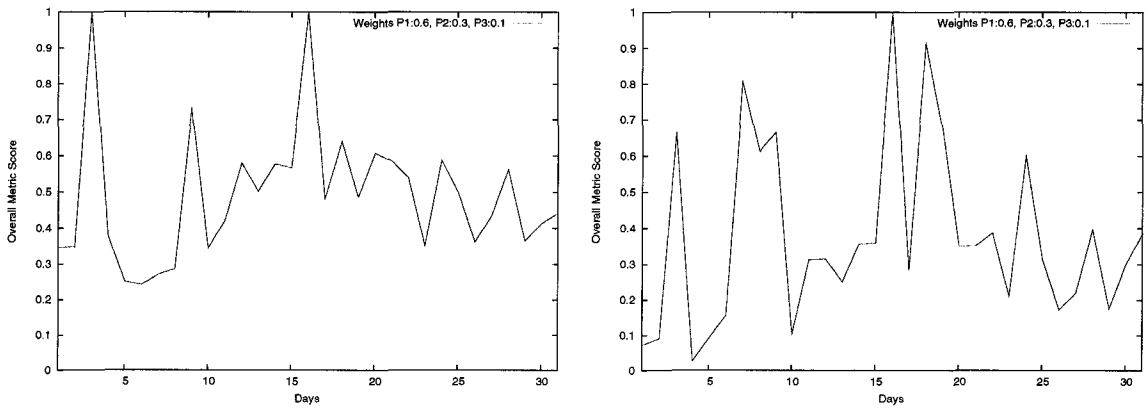
(a) Priority 1

(b) Priority 2

(c) Priority 3

Figure 6.5: Normalized Priority 1, 2, and 3 Snort Alerts flagged, on January 2005 WIDE Traces, after Disabling both "icmp-info.rules" and "icmp.rules" Categories in Snort Rule Database.

The actual configuration of Snort does not change except that one or both categories of rules mentioned above are disabled.



(a) "icmp-info.rules" Disabled

(b) "icmp-info.rules" & "icmp.rules" Disabled

Figure 6.6: IDM for January 2005 WIDE Traces with Normalized Metrics after (a) Disabling "icmp-info.rules" Category in Snort Rule Database (b) Disabling both "icmp-info.rules" and "icmp.rules" Categories in Snort Rule Database.

The experiments was carried out using the weights from Set1 of Table 6.1. The corresponding IDM, for both cases above, are illustrated in Figure 6.6. From Figure 6.6, one may observe that the choice of which set of rules to enable causes as much as 30% change in the IDM score. On some days, there is 0.5 difference in the IDM score depending on which rule set is enabled.

6.3.2 Discussion

Discussions of the IDM results after disabling rules in Snort are presented next.

- Disabling some rules in Snort, based on rational analysis, may detect some serious activities in the traffic - such as the IDM increase in January 7th traces - detected in the case of disabling both "icmp-info.rules" and "icmp.rules" Snort categories (Figure 6.6). In this day, normalized priority 2 alerts exceed 1 as shown in Figure 6.5.

Analysis for this day alerts indicates that a lot of these alerts are due to a detection of a "non-standard protocol or event" considered in Snort as "BAD-TRAFFIC Unassigned/Reserved IP protocol" used in a number of packets. Further analysis of the same alerts shows that they are due to packets belonging to the same flow. This fact will lead the administrator, after inspecting payloads (assuming they are not removed), to determine whether false positives or real attacks are carried inside the packets.

- When rules are disabled in both experiments above, a whole category of rules is disabled rather than selected rules within the category. Identifying of specific rules to be disabled depends on rationale analysis of triggered alerts as well as alerts impact on the network. The impact analysis can be particularly done in the presence of vulnerability information after vulnerability patches are applied. For highly-vulnerable networks, there is no need to disable any rule since alerts due to false positives are difficult to be identified.

- In both experiments, the IDM values were computed for one month (January) in 2005 WIDE traces. Even though computing IDM for all 2005 WIDE traces is possible, the results for one month are sufficient to learn the impact of disabling rules on the IDM values.

- Instead of disabling rules, adding pass rules to Snort rule database is possible to reduce the number of alerts raised due to false positives. This should be carefully done based on alerts impact analysis that allows identification of false positives. Fearing

from causing false negatives, one may not add pass rules.

Chapter 7

Different Perspectives of Network Security using Different IDS Metrics

7.1 Snort Configuration and Setup

In the following experimentations, the main goal is to compare each set of selected Snort metrics with the other two sets, and to study the impact of the selection on the resultant overall Intrusion Detection Metric (IDM). The IDM is produced as a result of linearly combining each block of normalized Snort metrics, according to the relative weight of each metric. Again, normalization and combination of metrics are led as was the case in the experiments of Chapter 6. To extract three sets of metrics from the logs of Snort, same real and large WIDE network traces - sampled in 2005 - are used as input to Snort.

Since data payloads of the WIDE traces were removed and their headers information was scrambled to protect privacy of the traces, Snort is tuned as before by disabling

the decoding phase (see section 4.2) and some preprocessors. However, two of Snort preprocessors are enabled: "frag3" and "stream4" (see section 4.2). Enabling these preprocessors is useful in identifying fragmented and evasion packets with alerts that do not contain any priority or classification assignment. The following is an example of alert triggered by the "frag3" Snort preprocessor:

"04/01-00:05:42.509139 [**] [123:5:1] (spp_frag3) Zero-byte fragment packet [**] UDP 193.192.19.132 -> 180.11.232.197".

Note that the alert above was raised due to a detection of a "zero-byte fragment" packet whose protocol type is UDP. As another example, the following is an alert triggered by the "stream4" preprocessor:

"04/01-00:05:22.739604 [**] [111:2:1] (spp_stream4) possible EVASIVE RST detection [**] TCP 173.226.1.168:3332 -> 197.109.1.64:80".

This alert was due to a possible attack to evade the IDS itself detected by the "stream4" preprocessor on a TCP packet.

The parser of the prototype Security Evaluator is used to extract the three sets of Snort metrics illustrated in Tables 4.4, 4.5, and 4.6. Once extracted, the metrics are polled,

normalized, combined, and delivered to an Apache web server for display, by the prototype components as depicted in Figure 5.4. The setup of Figure 5.4 is considered in all subsequent experiments running in a Linux environment.

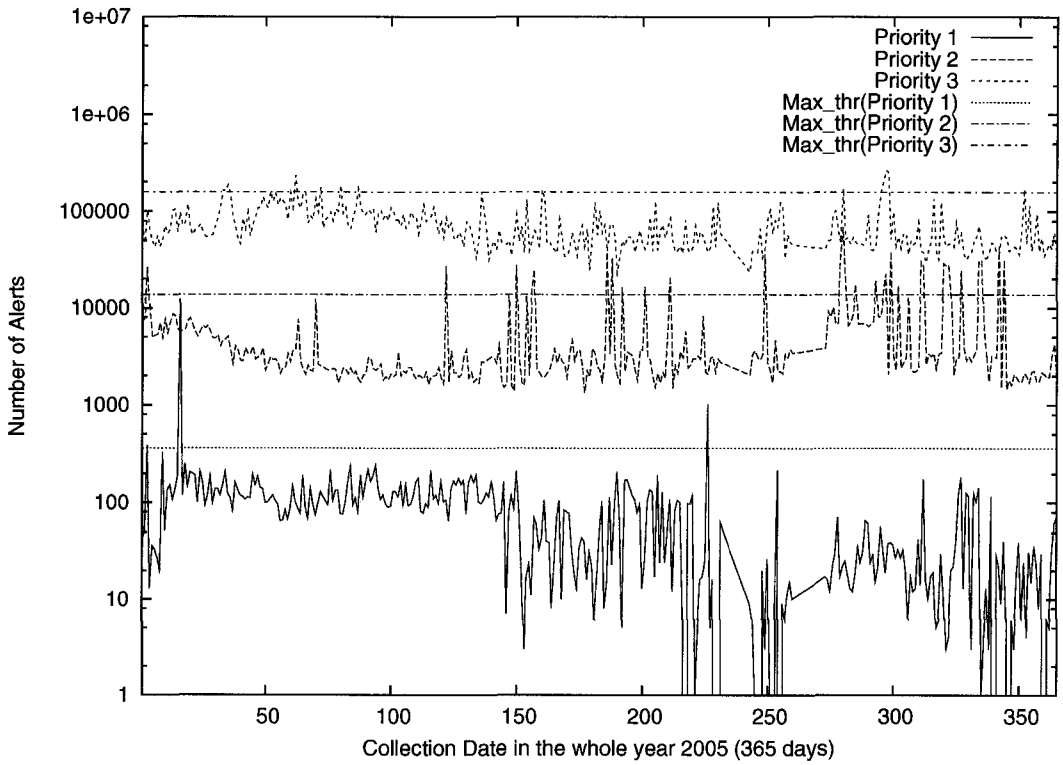
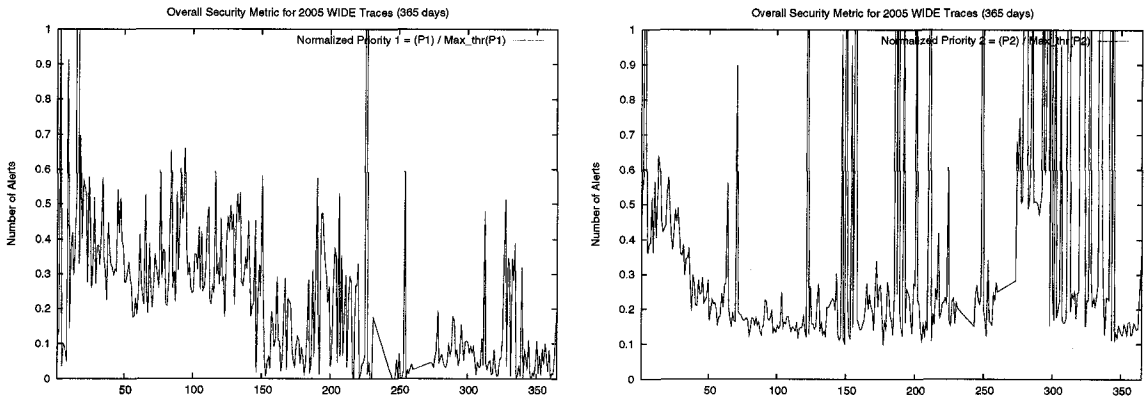


Figure 7.1: Priority 1, 2, and 3 Snort Alerts flagged on all 2005 WIDE Traces.

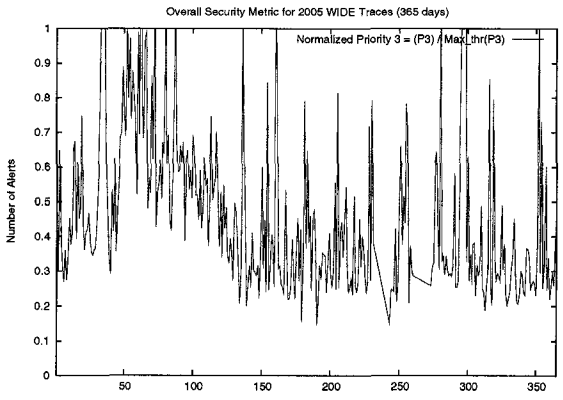
7.2 Priority Metrics

The first set of metrics is extracted based on their default priority designated by Snort. Then, the Priority 1, 2, and 3 Snort alerts metrics flagged on all 2005 WIDE Traces are plotted in Figure 7.1. Some examples of alerts with priority 1, 2, or 3 are mentioned in



(a) Priority 1

(b) Priority 2



(c) Priority 3

Figure 7.2: Normalized Metrics based on 'Priority' for all 2005 WIDE Traces.

section 4.3. In this figure, the normalization threshold of each metric is shown as a horizontal line. The shapes of the three metrics after the normalization process are illustrated in Figure 7.2.

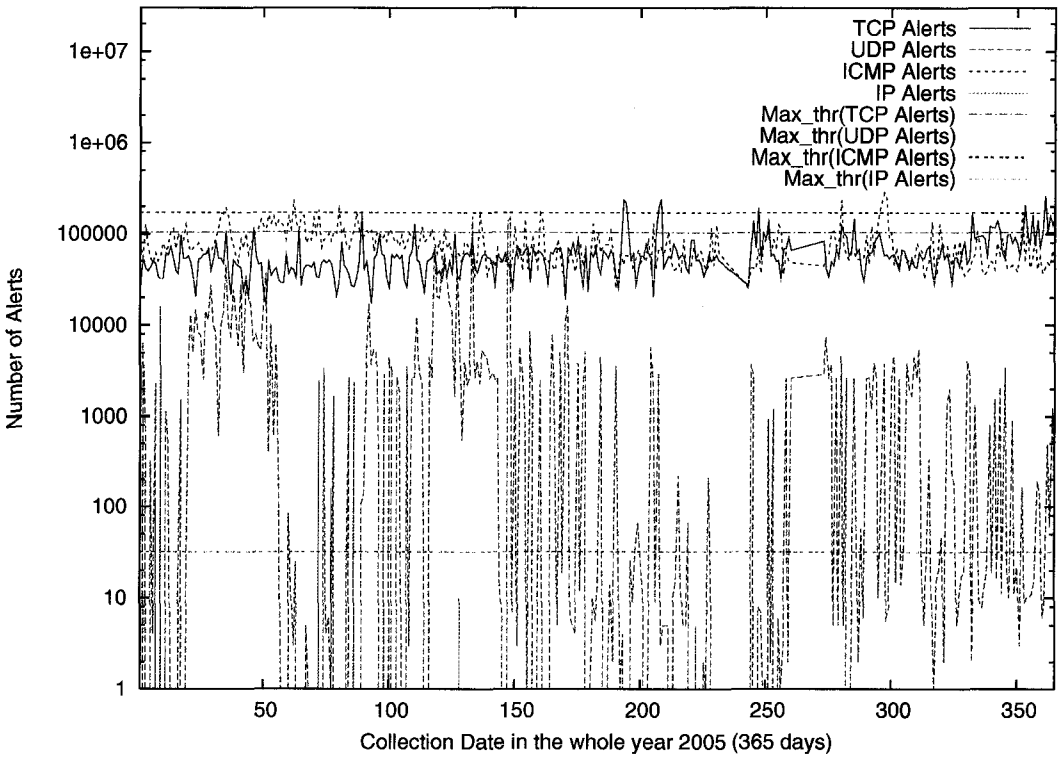
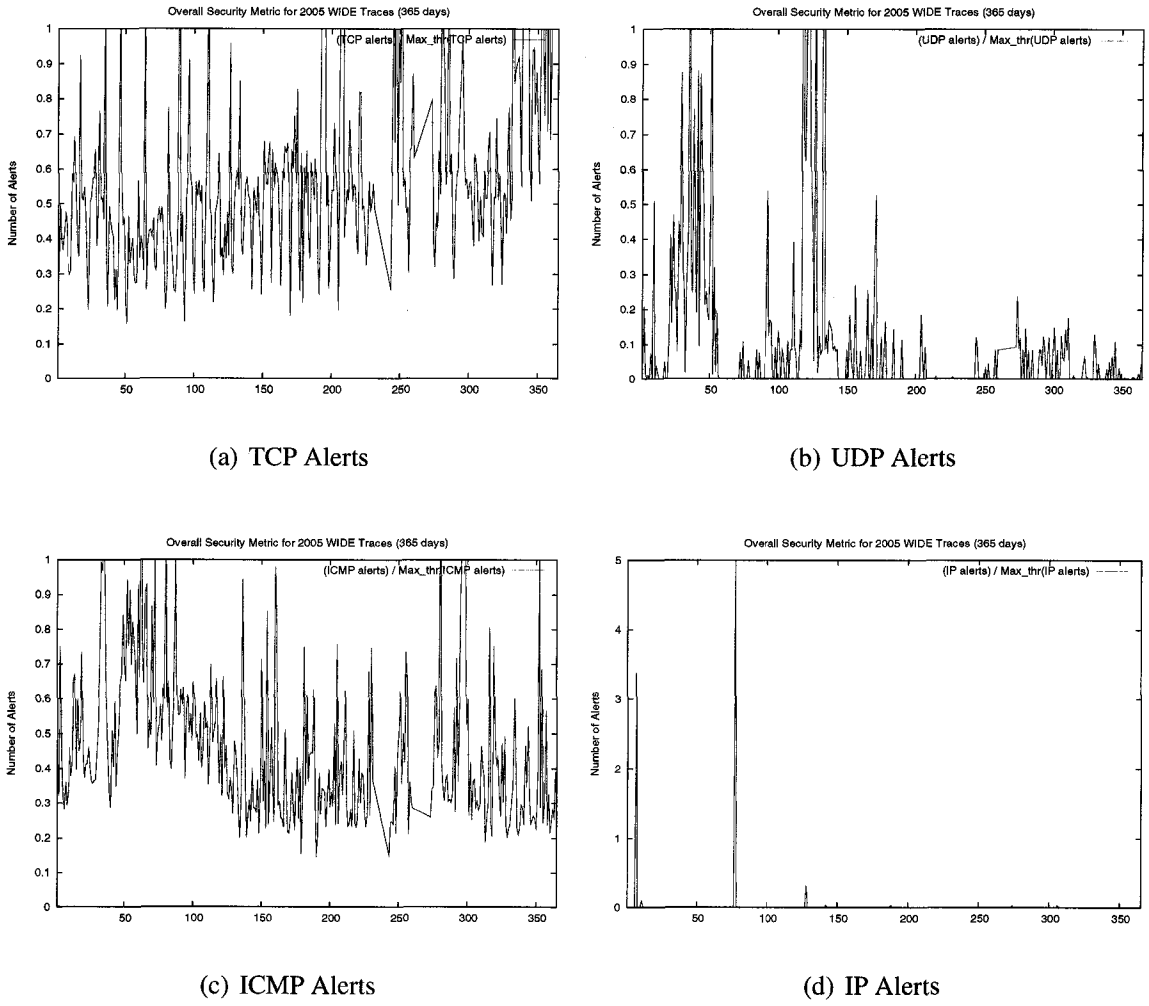


Figure 7.3: TCP, UDP, ICMP, and IP Snort Alerts flagged on all 2005 WIDE Traces.

7.3 Protocol Metrics

The Second set of metrics is extracted based on the protocol type of the packets on which the alerts were triggered by Snort. The four extracted metrics are distributed among four main protocol types: TCP, UDP, ICMP. and IP as illustrated in Chapter 4 and Table 4.5. Then, the TCP, UDP, ICMP, and IP Snort alerts metrics flagged on all 2005 WIDE Traces are plotted in Figure 7.3. Examples of alerts falling under each of these four metrics are stated in section 4.3. In this figure, the normalization threshold of each metric is shown as a horizontal line. The shapes of the four metrics after the normalization process are illustrated in Figure 7.4.



(a) TCP Alerts

(b) UDP Alerts

(c) ICMP Alerts

(d) IP Alerts

Figure 7.4: Normalized Metrics based on 'Protocol' for all 2005 WIDE Traces.

7.4 Classification Metrics

The third set of metrics is extracted based on the classification of the attacks or suspicious contents carried by the packets on which the alerts were raised by Snort. In this set of metrics, the Snort alerts are grouped into four categories: Successful attacks alerts, Potentially bad traffic, attempted attacks alerts. and miscellaneous activity alerts as illustrated in

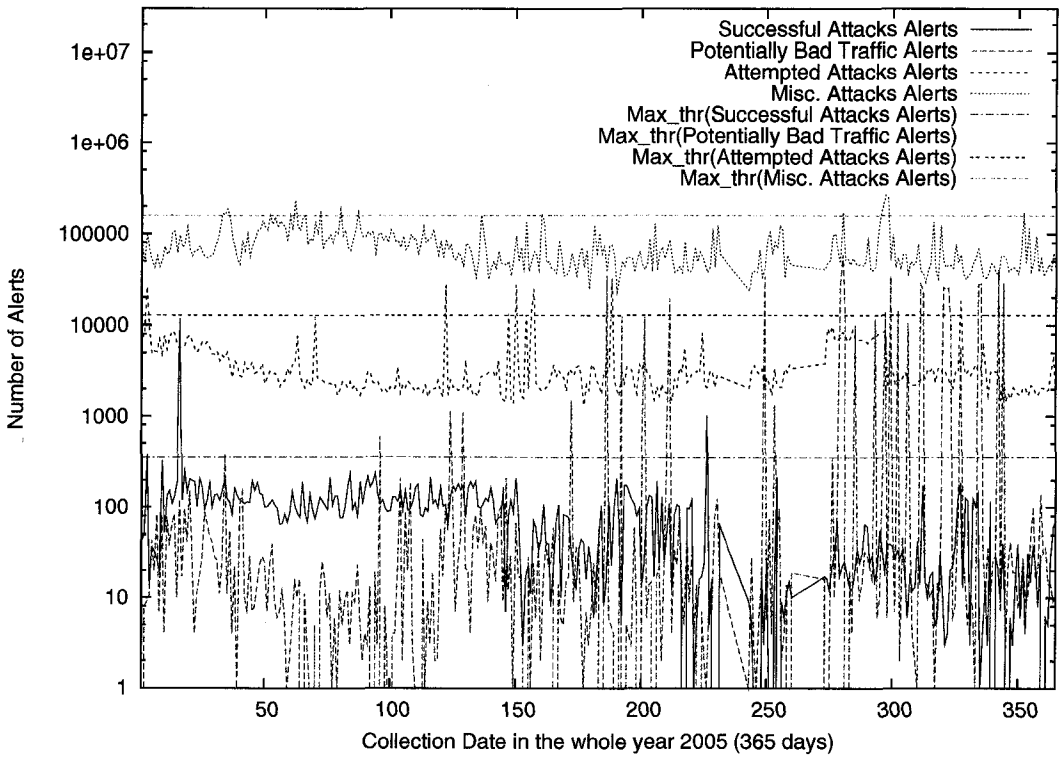
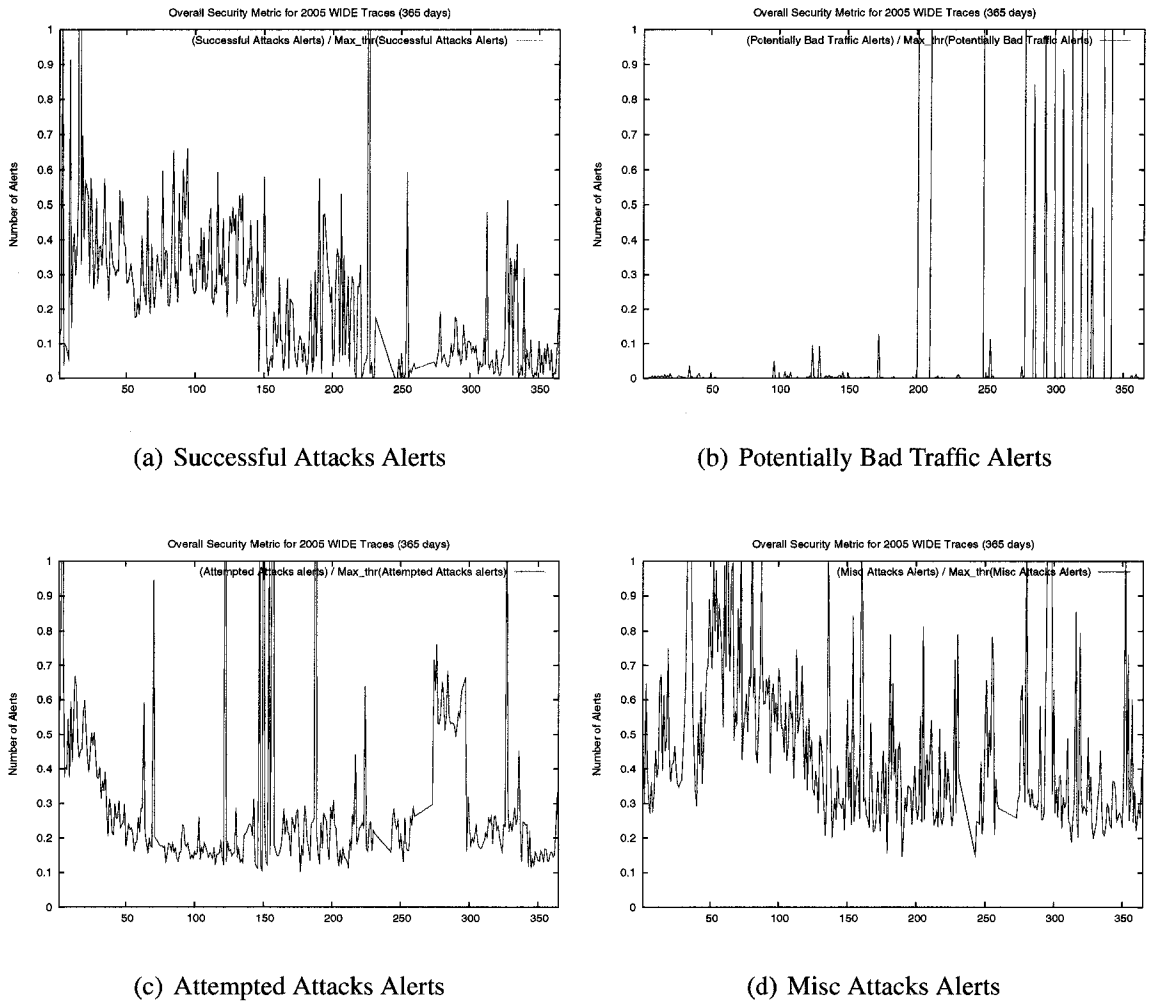


Figure 7.5: Successful Attacks, Potentially Bad Traffic, Attempted Attacks, and Miscellaneous Activities Snort Alerts flagged on all 2005 WIDE Traces.

Chapter 4 and Table 4.6. The metric "Successful attacks alerts" consists of alerts flagged on packets carrying apparent "backdoor trojan horses", a DoS deemed "Successful" by Snort, or even the presence of some system calls. The "Potentially bad traffic" metric is just the number of alerts raised on packets deemed potentially bad by Snort. The alerts flagged on packets that carry what is apparently considered by Snort as attempted attacks are classified as a separate metric. The alerts of this metric may include an "attempted DoS" found in the corresponding packets. Snort may flag alerts on packets having some suspicious content deemed as "Miscellaneous activity," which makes up the fourth metric in this set. More information about the classifications of these four metrics are discussed



(a) Successful Attacks Alerts

(b) Potentially Bad Traffic Alerts

(c) Attempted Attacks Alerts

(d) Misc Attacks Alerts

Figure 7.6: Normalized Metrics based on 'Classification' for all 2005 WIDE Traces.

in sections 4.3 and 4.4.1 supported by examples. These four Snort metrics flagged on all 2005 WIDE Traces are plotted in Figure 7.5, where the normalization threshold of each metric is shown as a horizontal line. The shapes of each of the four normalized metrics are illustrated in Figure 7.6.

7.5 Metrics Comparison

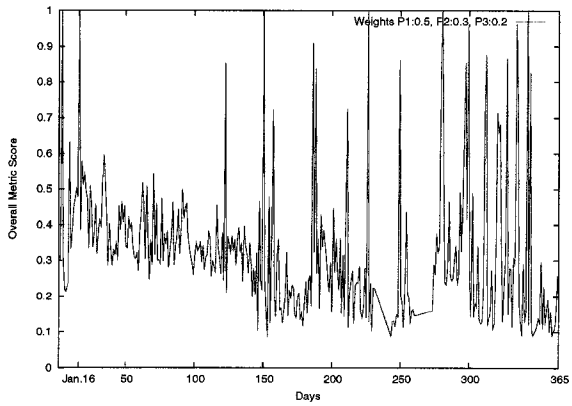
The purpose of comparing a set of metrics with other sets is to highlight their differences and similarities providing that each set is based on a different parameter. The metrics based on their default priority look in general close to the metrics based on the classification of Table 4.5, except in what is related to potentially bad traffic. However, the metrics based on protocol type are very different except in the case of 'ICMP alerts' metric, where a clear similarity appears between 'ICMP alerts', 'Priority 3', and 'Misc Attacks Alerts' metrics. This reads that the alerts assigned by Snort as 'Priority 3' are mainly classified as miscellaneous activities having a protocol type of ICMP. Examples of these similar-alerts are given in section 4.4.1.

Note that the priority metrics have slightly different values from those of the priority metrics extracted in experiments of Chapter 6. That is due to the fact that additional preprocessors are enabled in the Snort configuration under which the experiments of this chapter run. Consequently, a slightly-different normalization threshold is applied to the priority metrics of Figure 7.1.

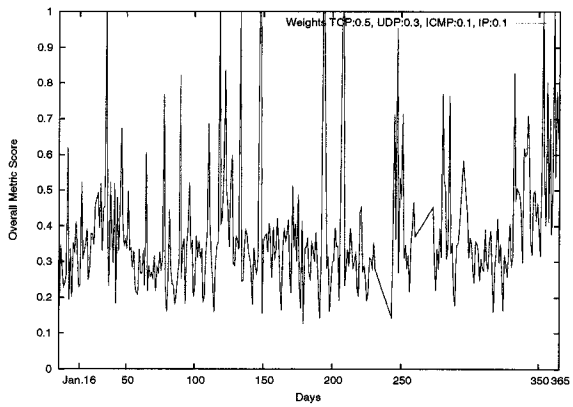
7.6 IDM Comparison

In Fig. 5.4, the intrusion detection metric IDM1 is the resultant overall intrusion index, where metrics based on default priority are combined. The IDM2 is the resultant of combined metrics based on protocol type, while IDM3 is the one that resulted from combined

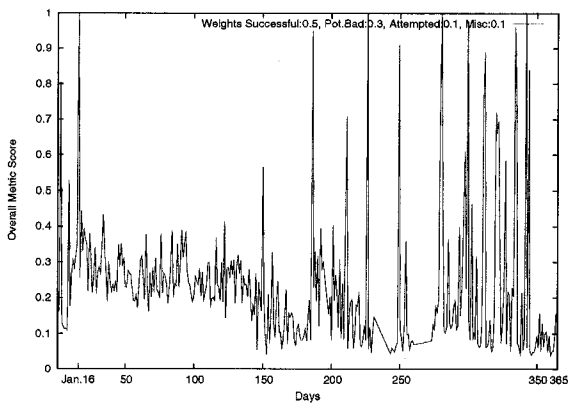
metrics based on classification of Table 4.6. The plots of these 3 IDMs are illustrated in Fig. 7.7.



(a) IDM1 based on 'Priority'



(b) IDM2 based on 'Protocol'



(c) IDM3 based on 'Classification'

Figure 7.7: Overall IDM for 2005 WIDE Traces based on (a) Priority, (b) Protocol, and (c) Classification.

The first notice is that IDM1 is not very different from IDM3 except in days between 180 and 220 approximately. In these days, IDM3 is affected by the potentially bad traffic shown in Fig. 7.6-b. IDM2 looks different than other two IDMs especially in days between

150 and 180, where a lot of TCP and UDP alerts were flagged.

7.7 Discussion

The results of the experiments in this chapter are important for comparison purposes either between the three sets of metrics or between the three corresponding IDMs. Further discussions are presented next.

- It is clear that the metrics extracted based on protocol type are higher than those based on priority or classification. This is very useful in uncovering hidden attacks that might not be identified from analysis of other two sets of metrics.

- Among metrics based on protocol type, the fourth metric is really two low. This is the number of Snort alerts raised on 2005 WIDE traces having IP as a type of protocol. The reason of getting such a low value of alerts on IP packets resides in the fact that Snort inspects packets of transport layer (TCP and UDP) before IP packets. All snort alerts of IP protocol type were due to "BAD-TRAFFIC Unassigned/Reserved IP protocol" which reads a bad number found in the field of IP protocol in packets. This is a priority 2 alert classified as part of a miscellaneous activity metric having the classification of "Detection of a non-standard Protocol or Event" (Table 4.2).

- There were few alerts triggered due to fragmentation problems occurred in the Generic Routing Encapsulation (GRE) protocol [Farinacci00]. GRE is an IP tunneling protocol. Since all alerts on GRE are related to fragmentation problems, they

are treated as fragmentation alerts triggered by Snort "frag3" preprocessor. Thus, the few GRE alerts are affiliated to UDP alerts and ranked second in the set of metrics based on protocol type.

- The ranking of metrics based on protocol type is intuitive. However, different ranking and weights to these metrics are possible. Weights adjustment depends on analysis of the impact of each protocol type metric over time.

- The rationale behind the ranking of metrics based on classification was given in Chapter 4. Examples of alerts and their corresponding Snort rules were discussed in section 4.4.1. Again different ranking and weights adjustment can be done after analysis of the impact of each class of metrics over time.

Chapter 8

Conclusions and Future Work

In this thesis, an experimental study to evaluate network security is presented, proposing a hierarchical quantitative metrics model and using a prototype network security evaluator. The model takes advantage of a known security tree taxonomy, and relies on metrics extracted from the generated output of the Snort IDS tool. Experiments are conducted using network traffic traces from the WIDE network. These experiments analyzed the use of the HQM model for measuring network security posture, and assessed the effect of adjustable weights and IDS rules on the resultant overall metric.

Further experiments involve three sets of IDS metrics in this study. One set of three metrics is based on Snort alerts default priority, another set of four metrics is based on the protocol types of generated Snort alerts, and a final set of four metrics based on Snort classification of triggered alerts. An overall security threat indicator is calculated for each set of metrics applying normalization and combination methods. Comparison of

the resulting three overall security indicators show the modifications that may occur if a different parameter is used in the selection of IDS metrics.

In this experimental study, the following key observations and learning are discussed:

- The proposed HQM is a tree-based security metrics model, where metrics are extracted quantitatively from different branches of the tree. The way of extracting IDS metrics from one branch (IDM sub-tree) can be applied in other branches of the tree, such as in the Vulnerability Index sub-tree.

- The Snort tool is an example of an IDS used in this metrics study. The choice of Snort was due to its widely-deployed open-source tool status. Even though the quantitative metrics extracted from Snort alerts are specific to Snort, similar way of metrics extraction can be applied in other IDS tools. The use of three different parameters (priority, protocol type, and alerts classification) suggests the possibility of extracting metrics from other IDSs based on one or more of these parameters.

- While metrics selection based on default priority leads to an overall intrusion metric IDM, the selection of metrics based on different perspectives (protocol type and alerts classification) gives more opportunity to security experts to analyze the resulting IDMs, compare between them, and explore hidden suspicious activities.

- The experiments conducted in this study reveal the importance of real network traffic traces usage, the effect of different metrics' weights on the IDM, and the impact of Snort rules modification on the evaluation.

- The automated security evaluator prototype serves as a good means to ensure that security metrics are captured, combined, and then reported to a web server for viewing by decision-makers. The reported result would be a good indicator, while deep investigation into analysis of incoming packets is needed. Once reported, the IDM as a security threat indicator helps decision-makers in the planning and investment into enforcing the security of their network.

- The ultimate purpose of evaluating the security of a network, is to identify the status of the network while considering a variety of aspects that may affect network security. The experiments led in this research show the feasibility of employing a network security evaluator, starting from the bottom level of the HQM model. At the top of the model, as in [Seddigh04, Nandy04], an overall assurance metric, namely IAM, would combine the IDM and other metrics to obtain a more comprehensive picture of network security. Combining all levels of the tree is a challenge that requires significantly further study and research.

- The normalization procedure used on metrics is very important to be able to compare metrics given the different ranges of raw values. Using a "maximum threshold" for each metric, as stated in section 3.2, is a necessity to obtain meaningful results.

- The weighting approach used against selected metrics is of great importance. In the experiments of this study, a method similar to the "swing-weighting approach"

[Clemen01] is used where the order of metrics importance is alternated to show experimentally how this would affect the evaluation of network security (see Chapter 6). Applying this method in assigning weights to all levels of the tree is an area for potential future investigation.

- Even though the traces applied in the experiments are very useful in emulating live networks, the fact that no payloads exist in the traces makes Snort unable to deliver alerts that might have been caused by those missing payloads. Therefore, it would be very useful to do experiments in online live networks.

This research is particularly important for business and governmental enterprises who want to evaluate their networks. In addition. this work is a step to move from theory into the practice and experimentation of network security evaluation. Expansion of this work is possible to address greater parts of the HQM tree, that allows the inclusion of more quantitative metrics. There is an open research in the use of quantitative metrics in network security evaluation that is not limited to technical elements; it includes operational and organizational elements as well to form the broad area of Information Assurance research.

References

[Apthorpe01] Apthorpe, R., "A Probabilistic Approach to Estimating Computer System Reliability," *In Proceedings of the LISA 2001 15th Systems Administration Conference, San Diego, CA, USA*, December 2001.

[Basili88] Basili, V. R. and H. D. Rombach, "The TAME project: Towards improvement-oriented software environments," *IEEE Transactions on Software Engineering*, volume 14, no. 6, June 1988, pp. 758–773.

[Basili94] Basili, V. R., G. Caldiera, and H. D. Rombach, "The Goal Question Metric Paradigm," *Encyclopedia of Software Engineering*, volume 2, John Wiley & Sons, 1994, pp. 528–532.

[Burgess04] Burgess, M., Analytical Network and System Administration - Managing Human-Computer Systems, *John Wiley & Sons*, 2004, ISBN 0-470-86100-2.

[Bush93] Bush, M. and N. Ashley, "Metkit: Toolkit for Metrics Education," *IEEE Software*, volume 10, no. 6, November 1993, pp. 52–54.

[Cavusoglu04] Cavusoglu, H., B. Mishra, and S. Raghunathan, "A Model for Evaluating IT Security Investments," *Communications of the ACM*, volume 47, no. 7, July 2004.

[Cavusoglu05] Cavusoglu, H., B. Mishra, and S. Raghunathan, "The Value of Intrusion Detection Systems in Information Technology Security Architecture," *Information Systems Research*, volume 16, no. 1, March 2005, pp. 28–46.

[CC005] "Common Criteria for Information Technology Security Evaluation - Part 1: Introduction and general model, CCMB-2005-07-001; Part 2: Functional security components, CCMB-2005-07-002; Part 3: Security assurance components, CCMB-2005-07-003; Part 4: Evaluation methodology, CCMB-2005-07-004, Version 3.0, Revision 2," , July 2005, `http://www.commoncriteriaportal.org/public/expert/index.php?menu=3`.

[Chaula03] Chaula, J. A., "Security Metrics and Public Key Infrastructure Interoperability Testing," Ph.D. thesis, *Stockholm University*, December 2003.

[Clark05] Clark, K., J. Dawkins, and J. Hale, "Security Risk Metrics: Fusing Enterprise Objectives and Vulnerabilities," *In Proceedings of the 2005 IEEE Workshop on Information Assurance and Security, US Military Academy, West Point, NY*, June 2005.

[Clemen01] Clemen, R. T. and T. Reilly, Making Hard Decisions with Decision-Tools, Duxbury/Thomson Learning, 2001, ISBN 0-534-36597-3.

[Cormen01] Cormen, T. H., C. E. Leiserson, R. L. Rivest, and C. Stein, Introduction to Algorithms, *MIT Press and McGraw-Hill*, Chapter 26, Section 26.2: The Ford-Fulkerson method, pp. 651-664, 2001, 2nd edition, ISBN 0-262-03293-7.

[D'Ambrosio01] D'Ambrosio, B., M. Takikawa, J. Fitzgerald, D. Upper, and S. Mahoney, "Security Situation Assessment and Response Evaluation (SSARE)," *Proceedings of the DARPA Information Survivability Conference & Exposition II, DISCEX'01*, volume 1, June 2001, pp. 387–394.

[deBoer02] de Boer, R. C., "A Generic Architecture for Fusion-Based Intrusion Detection Systems," Master's thesis, *Erasmus University, Rotterdam*, October 2002.

[Egan05] Egan, M. and T. Mather, The Executive Guide to Information Security: Threats, Challenges, and Solutions, *Symantec Corporation*, 2005, ISBN 0-32-130451-9.

[El-Hassan06] El-Hassan, F., A. Matrawy, N. Seddigh, and B. Nandy, "Experimental Evaluation of Network Security through a Hierarchical Quantitative

Metrics Model," *Proceedings of the Third IASTED International Conference on Communication , Network, and Information Security (CNIS 2006), MIT, Cambridge, MA, USA*, October 2006.

[Evans01] Evans, S., S. Bush, and J. Hershey, "Information Assurance through Kolmogorov Complexity," *DARPA Information Survivability Conference and Exposition II, DISCEX'01, Proceedings*, volume 2, 12-14 June 2001, pp. 322–331.

[Farinacci00] Farinacci, D., T. Li, S. Hanks, D. Meyer, and P. Traina, "Generic Routing Encapsulation (GRE)," *RFC 2784*, March 2000, `http://www.faqs.org/rfcs/rfc2784.html`.

[Fenton91] Fenton, N. E., Software Metrics - A Rigorous Approach, Chapman and Hall, 1991, ISBN 0-412-40440-0.

[Fenton97] Fenton, N. E. and S. L. Pfleeger, Software Metrics - A Rigorous and Practical Approach, PWS Publishing Co., 1997, ISBN 0-534-95600-9.

[Fink02] Fink, G. A., B. L. Chappell, T. Turner, and K. O'Donoghue, "A Metrics-Based Approach to Intrusion Detection System Evaluation for Distributed Real-Time Systems," *Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS'02)*, 2002.

[Fox03] Fox, K., R. Henning, J. Farrell, and R. Vaughn, "A Prototype Next Generation Information Assurance Tool - A Data Fusion Model for Information Systems Defense," *In Proceeding of the 7th World Multiconference on Systemics, Cybernetics and Informatics (SCI 2003), Orlando, FL*, volume 1, July 2003, pp. 284–289.

[Geer03] Geer, D., K. Hoo, and A. Jaquith, "Information security: Why the Future Belongs to the Quants," *IEEE Security and Privacy*, volume 1, no. 4, July/August 2003.

[Hallberg05] Hallberg, J., A. Hunstad, and M. Peterson, "A Framework for System security Assessment," *In Proceedings of the 2005 IEEE Workshop on Information Assurance and Security, US Military Academy, West Point, NY*, June 2005.

[Hariri03] Hariri, S., G. Qu, T. Dharmagadda, M. Ramkishore, and C. S. Raghavendra, "Impact Analysis of Faults and Attacks in Large-Scale Networks," *IEEE Security and Privacy*, volume 1, no. 5, September-October 2003, pp. 49–54.

[Hearn04] Hearn, J., "Does the Common Criteria Paradigm Have a Future?" *IEEE Security and Privacy*, volume 2, no. 1, January/February 2004.

[Hoo00] Hoo, K. S., "How Much Is Enough? A Risk-Management Approach to Computer Security," Ph.D. thesis, *Stanford University*, June 2000.

[Hoone] Hoo, K. S., "Metrics of Network Integrity," , online, http://www.bizforum.org/whitepapers/sygate-2.htm.

[Hudack03] Hudack, L. R., L. L. Orsini, and B. M. Snow, "How To Assess And Enhance Financial Health," *National Association of College and University Business Officers (NACUBO) Business Officer Magazine*, April 2003, http://www.nacubo.org/documents/bom/2003_04_financial_health.pdf.

[ISO1540805] ISO15408, "Information technology – Security techniques – Evaluation criteria for IT security – Part 1: Introduction and general model(2005) – Part 2: Security functional requirements(1999) – Part 3: Security assurance requirements(1999)," , 2005, http://www.iso.org/iso/en/CombinedQueryResult.CombinedQueryResult?queryString=15408.

[Killmeyer06] Killmeyer, M., The Executive Guide to Information Security: Threats, Challenges, and Solutions, Auerbach, 2006, ISBN 0-32-130451-9.

[Kulkarni06] Kulkarni, A. and S. Bush, "Detecting Distributed Denial-of-Service Attacks Using Kolmogorov Complexity Metrics," *Journal of Network and Systems Management*, volume 14, no. 1, March 2006.

[Laskey04] Laskey, K., G. Alghamdi, X. Wang, D. Barbar, T. Shackelford, E. Wright, and J. Fitzgerald, "Detecting Threatening Behavior Using

Bayesian Networks," *Proceedings of the 13th Conference on Behavioral Representation in Modeling and Simulation (BRIMS), Arlington, Virginia, USA*, May 2004.

[Lee03] Lee, J., J. Lee, S. Lee, and B. Choi, "A CC-based Security Engineering Process Evaluation Model," *In Proceedings of the 27th Annual International Computer Software and Applications Conference (COMPSAC.03)*, November 2003.

[Li97] Li, M. and P. Vitanyi, "An Introduction to Kolmogorov Complexity and Its Applications," *Springer-Verlag, NY*, 1997.

[Mell03] Mell, P., V. Hu, R. Lipmann, J. Haines, and M. Zissman, "An Overview of Issues in Testing Intrusion Detection Systems," *Technical Report NIST IR 7007, National Institute of Standard and Technology.*, 2003, `citeseer.ist.psu.edu/621355.html`.

[Nandy04] Nandy, B., P. Pieda, N. Seddigh, I. Lambadaris, A. Matrawy, and A. Hatfield, "Information Assurance Metrics," *Technical Report prepared for the Canadian Federal Government Department Office of Critical Infrastructure Protection and Emergency Preparedness (OCIPEP)*, March 2004.

[Paulk93] Paulk, M. C., C. V. Weber, S. M. Garcia, M. B. Chrissis, and M. Bush,

"Key Practices of the Capability Maturity Model, Version 1.1," , February 1993, http://www.sei.cmu.edu/pub/documents/93.reports/pdf/tr25.93.pdf.

[Peltier05] Peltier, T. R., Information Security Risk Analysis, *Auerbach*, 2005, 2nd edition, ISBN 0-8493-3346-6.

[Peterson04] Peterson, M., "CAESAR - A Proposed Method for evaluating security in component-based distributed information systems," Master's thesis, *Linkoping University, Sweden*, 2004.

[Proctor01] Proctor, P. E., The Practical Intrusion Detection Handbook, *Prentice Hall PTR*, 2001, ISBN 0-13-025960-8.

[Reiter99] Reiter, M. and S. Stubblebine, "Authentication Metric analysis and Design," *ACM Transactions on Information and System Security*, volume 2, no. 2, May 1999.

[Roesch99] Roesch, M., "Snort: Lightweight Intrusion Detection for Networks," *In Proceedings of USENIX LISA99, the 13th Systems Administration Conference*, November 1999, http://www.snort.org.

[Roesch05] Roesch, M., C. Green, and Sourcefire., "Snort Users Manual 2.4.0," *The Snort Project*, August 2005, http://www.snort.org/docs/snort_manual.pdf.

[Saaty89] Saaty, T. L. and J. M. Alexander, Conflict Resolution - The Analytic Hierarchy Approach, *Praeger*, 1989, ISBN 0-275-93229-X.

[Saaty06] Saaty, T. L. and L. G. Vargas, Decision Making with the Analytic Network Process - Economic, Political, Social and Technological Applications with Benefits, Opportunities, Costs and Risks, *Springer Science + Business Media*, 2006, ISBN 0-387-33859-4.

[Schechter04] Schechter, S. E., "Computer Security Strength & Risk: A Quantitative Approach," Ph.D. thesis, *Harvard University*, May 2004.

[Schechter05] Schechter, S. E., "Toward Econometric Models of the Security Risk from Remote Attacks," *IEEE Security and Privacy*, January/February 2005.

[Schultz97] Schultz, H. P., "Software Management Metrics," *In book: Software Engineering Project Management, Edited by R.H. Thayer, Second Edition, IEEE, ISBN: 0-8186-8000-8*, 1997, pp. 488–502.

[Seddigh04] Seddigh, N., P. Pieda, A. Matrawy, B. Nandy, I. Lambadaris, and A. Hatfield, "Current Trends and Advances in Information Assurance Metrics," *In Proc. of the Second Annual Conference on Privacy, Security, and Trust, PST2004, Fredericton, NB*, October 2004.

[Siaterlis04] Siaterlis, C. and B. Maglaris, "Towards Multisensor Data Fusion for DoS detection," *ACM Symposium on Applied Computing*, 2004.

[Simonsen04] Simonsen, G., "En prosess for Sikkerhets Metrikk Program (SMP) (A process for Security Metrics Program (SMP))," Master's thesis, *Gjovik University College, Norway*, June 2004.

[SSE-CMM03] SSE-CMM, Systems Security Engineering Capability Maturity Model (SSE-CMM) Project, Version 3 (Final Version), *Carnegie Mellon University and Organizations*, June 2003, `http://www.sse-cmm.org/docs/ssecmmv3final.pdf`.

[Stoneburner02] Stoneburner, G., A. Goguen, and A. Feringa, "Risk Management Guide for Information Technology Systems," *Recommendations of the National Institute of Standards and Technology*, July 2002, `http://csrc.nist.gov/publications/nistpubs/800-30/sp800-30.pdf`.

[*Thayer, Editor*97] *Thayer, Editor*, R. H., Software Engineering Project Management, IEEE, 1997, 2nd edition, ISBN 0-8186-8000-8.

[Vaughn03] Vaughn, R. B., R. Henning, and A. Siraj, "Information Assurance Measures and Metrics - State of Practice and Proposed Taxonomy," *In proceedings of the Frameworks and Methods for the Study and Analysis of Trust in Information Systems: Minitrack in the Software Technology Track of the 36th Hawaii International Conference on System Sciences (HICSS-36)*, January 2003.

[wid05] "Packet Traces from WIDE Backbone stored in the MAWI Working Group Traffic Archive," , Collected in 2005, `http://tracer.csl.sony.co.jp/mawi`.

[Wold04] Wold, G., "Key Factors in Making Information Security Policies Effective," Master's thesis, *Gjovik University College, Norway*, June 2004.

Appendix A

Example of Snort Rules

One collection of rules is stated as an example of Snort rules: The ICMP Information rules.

A.1 ICMP Information Rules

In the Snort configuration file, the "icmp-info.rules" are included as follows.

```
include $RULE_PATH/icmp-info.rules
```

The total number of ICMP information rules in the version 2.4.1 of Snort database is 93 rules categorized and prioritized as illustrated in Table A.1.

Table A.1: Snort ICMP Information Rules.

Snort Category	**icmp-info.rules**	**Priority**
misc-activity	92 rules	3
attempted-recon	1 rule	2
	Total: 93 rules	

A.1.1 Examples of "misc-activity" Category

Examples of rules in the "misc-activity" category of Snort database rules are stated below. Note that the first word of each rule "alert" determines the action that Snort would execute once a match is identified between a packet and the rule.

- alert icmp $EXTERNAL\_NET$ *any* $->$ $HOME\_NET$ any (msg:"ICMP PING Windows"; itype:8; content:"abcdefghijklmnop"; depth:16; reference:arachnids,169; classtype:misc-activity; sid:382; rev:7;)
- alert icmp $EXTERNAL\_NET$ *any* $->$ $HOME\_NET$ any (msg:"ICMP Destination Unreachable Host Unreachable"; icode:1; itype:3; classtype:misc-activity; sid:399; rev:6;)
- alert icmp $EXTERNAL\_NET$ *any* $->$ $HOME\_NET$ any (msg:"ICMP Mobile Host Redirect undefined code"; icode:>0; itype:32; classtype:misc-activity; sid:420; rev:7;)
- alert icmp $EXTERNAL\_NET$ *any* $->$ $HOME\_NET$ any (msg:"ICMP Parameter Problem undefined Code"; icode:>2; itype:12; classtype:misc-activity; sid:428; rev:7;)

For example, the first rule above causes an alert if the ICMP packet's content "abcdefghijklmnop" is matched. The flagged alert identifies the external network (source IP address) on any port and the home network (destination IP address) on any port, with ICMP

type value 8 (Echo request). The text message accompanied with the flagged alert is "ICMP PING Windows". More information about this alert can be known from arachnids with 169 as an id. Then, Snort includes in the alert its category "misc-activity" with a Snort ID 382 and a revision number 7 (since the rule can be updated to a new revision).

A.1.2 Example of "attempted-recon" Category

A rule-example that belongs to "attempted-recon" category is stated below.

- alert icmp $EXTERNAL\_NET$ *any* $-> $HOME\_NET$ any (msg:"ICMP traceroute"; itype:8; ttl:1; reference:arachnids,118; classtype:attempted-recon; sid:385; rev:4;)

This category includes detected attempts of information leaks. There is only one rule in icmp-info.rules that is classified as "attempted-recon". Snort inspects incoming ICMP packets with this rule, and flags an alert if the rule conditions are matched. Therefore, if an incoming packet is an ICMP one coming from a specific network with (source IP address) on any port to a specific home network (destination IP address) on any port, with ICMP type value 8 (Echo request), and TTL value 1, then an alert will be flagged with a text message "ICMP traceroute". More information about this alert can be known from arachnids with 118 as an id. Snort information about this alert indicates alert's category "attempted-recon" with a Snort ID 385 and a revision number 4. The "attempted-recon" means attempted reconnaissane.