RESEARCH DIRECTIONS IN SECURITY METRICS

Wayne Jansen
National Institute of Standards and Technology

Abstract

More than 100 years ago, Lord Kelvin observed that measurement is vital to deep knowledge and understanding in physical science. During the last few decades, researchers have made various attempts to develop measures and systems of measurement for computer security with varying degrees of success. This paper provides an overview of the security metrics area and looks at possible avenues of research that could be pursued to advance the state of the art.

Keywords: *Security Metrics; Computer Security, Security Evaluation*

INTRODUCTION

Security metrics is an area of computer security that has been receiving a good deal of attention lately. It is not a new topic, but one which receives focused interest sporadically. Much of what has been written about security metrics is definitional, aimed at providing guidelines for defining a security metric and specifying criteria for which to strive. However, relatively little has been reported on actual metrics that have been proven useful in practice (Berinato, 2005; Center for Internet Security, 2008).

Information security metrics are seen as an important factor in making sound decisions about various aspects of security, ranging from the design of security architectures and controls to the effectiveness and efficiency of security operations. Security metrics strive to offer a quantitative and objective basis for security assurance to enable an organization to gauge how well it is meeting its security objectives.

Security metrics can be categorized various ways. One simple classification is to consider metrics that denote the maturity level of processes believed to contribute to the security of a system, versus those that denote the extent to which some security characteristic is present in a system (Jelen, 2000). The former apply to security processes, procedures, and training used when designing, configuring, maintaining, and operating a system. The latter apply to the security posture of a system and the inherent level of risk involved. More elaborate security metric taxonomies also exist (Savola, 2007; Vaughn et al., 2002).

Other disciplines, such as the field of finance, have proven quantitative methods for determining risk along with decision-making frameworks based on established measures and metrics. Such standardized measurements and decision-aid capabilities are just emerging for information system security, however, and as in any discipline, require realistic assumptions and inputs to attain reliable results. Advancing the state of scientifically sound, security measures and metrics (i.e., a metrology for information

system security) would greatly aid the design, implementation, and operation of secure information systems.[1]

BACKGROUND

A metric generally implies a system of measurement based on quantifiable measures. For information system security, the measures are concerned with aspects of the system that contribute to its security. That is, security metrics involve the application of a method of measurement to one or more entities of a system that possess an assessable security property to obtain a measured value.

A method of measurement used to determine the unit of a quantity can involve a measuring instrument, reference material, or measuring system. Measured values of security properties should be linearly ordered and the method of measurement well defined, including details of how specific factors are to be measured or assessed and explanations of sources of uncertainty.

To be of value, the method of measurement should be reproducible, capable of attaining the same result when performed independently by different competent evaluators. Results should also be repeatable, such that a second assessment by the same evaluators produces the same results. Relevance and timeliness are also implicit considerations, since it is of little benefit to have measures that are not meaningful or whose latency exceeds their usefulness.

When used in the context of Information Technology (IT), the term "metrics" is a bit of misnomer. It implies that traditional concepts in metrology, as used in physics and other areas of science and technology, apply equally to IT. That is not the case, however (Gray, 1999; Vaughn et al., 2002). For example, the concepts of fundamental units, scales, and uncertainty prevalent in scientific metrics have not traditionally been applied to IT or have been applied less rigorously.

While some movement toward quantitative metrics for IT system security exists, in practice, qualitative measures that reflect reasoned estimates of security by an evaluator are the norm. That is, measures of information system security properties are often based on an evaluator's expertise, intuition, and insight to induce an ordering, which is then quantified (e.g., 1=low, 2=medium, 3=high). Because of the subjectivity involved, some of the attributes sought in a good metric are not readily obtainable. For example, results in penetration testing or other methods of assessment that involve specialized skills are sometimes not repeatable, since they rely on the knowledge, talent, and experience of the individual evaluator.

ASPECTS OF SECURITY MEASUREMENT

Many major efforts to measure or assess security have been attempted. They include the Trusted Computer System Evaluation Criteria (TCSEC) (Department of Defense, 1985),

[1] Certain commercial products and trade names are identified in this paper to illustrate technical concepts. However, it does not imply a recommendation or an endorsement by NIST.

Information Technology Security Evaluation Criteria (ITSEC) (Commission of the European Communities, 1991), Systems Security Engineering Capability Maturity Model (SSE-CMM) (International Systems Security Engineering Association, 2008), and Common Criteria (Common Criteria Portal, 2006). Each attempt has obtained only limited success.

It is reasonable to infer from the experience to date that security measurement is a tough problem, not to be underestimated (Bellovin, 2006). Further evidence is that the topic, Enterprise-Level Security Metrics, was included in the most recent Hard Problem List prepared by the INFOSEC Research Council (IRC, 2005), which identifies key research problems from the perspective of its members, the major sponsors of information security research within the U.S. Government. The Institute for Information Infrastructure Protection (I3P) report also identified security metrics as one of its four research and development priorities for the next five to ten years (I3P, 2009).

Insights into some critical aspects of security measurement gleaned from past efforts are discussed below. The intent is not to give a list of common pitfalls, although some are mentioned in the discussion. Instead, the objective is to highlight those factors that are believed to be pertinent to a research effort in security metrics.

Correctness and Effectiveness

Security of an IT system comprises two interdependent aspects: correctness and effectiveness. Correctness denotes assurance that the security-enforcing mechanisms have been rightly implemented (i.e., they do exactly what they are intended to do, such as performing some calculation). Effectiveness denotes assurance that the security-enforcing mechanisms of the system meet the stated security objectives (i.e., they do not do anything other than what is intended for them to do, while satisfying expectations for resiliency).

The evaluation of correctness gauges the ability of the security-enforcing mechanisms to carry out their tasks precisely to the specifications. Correctness can be assessed with respect to the development process and the development environment during the construction of the system and also in terms of its operation. Emphasis is typically on substantiating how well the system exhibits the behavior expected of it.

The evaluation of effectiveness gauges the strength of the security-enforcing mechanisms to withstand attacks in carrying out their function. Effectiveness requires ascertaining how well the security-enforcing components tie together and work synergistically, the consequences of any known or discovered vulnerabilities, and the usability of the system. Emphasis is typically on substantiating whether the system can be induced to exhibit behavior that leads to or demonstrates a security vulnerability.

In practice, security evaluations of correctness and effectiveness are largely done through reasoning rather than direct measurement of actual hardware and software components. Often, simplifying assumptions are made. For example, it may be postulated that the system communicates only with other systems operating under the same management control and security policy constraints; the need to trust and communicate with external

systems under different management control would not be specifically addressed (Science Applications International Corporation, 2007). The emphasis on abstractions and simplifying assumptions dissociates the assessment results from actual operational use (Littlewood et al., 1993). Even with this emphasis, the correctness and effectiveness of significantly large systems cannot be accurately determined for higher levels of assurance. Because of the high degree of human element involved, the timeliness and reproducibility of results also come into question. Conventions and practices, such as standardized procedures and criteria, can be applied to help normalize results among evaluators and expedite the process. Technical refresher classes and conformance training are other useful practices. Nevertheless, unless more of the human element can be eliminated or replaced by automated means, the current conditions can be expected to persist.

Leading Versus Lagging Indicators

Analogous to economic indicators, security metrics may be potentially leading, coincident, or lagging indicators of the actual security state of the system. The distinction is significant. A coincident indicator reflects security conditions happening concurrently, while leading and lagging indicators reflect security conditions that exist respectively before or after a shift in security. If a lagging indicator is treated as a leading or coincident indicator, the consequences due to misinterpretation and reaction can be serious. The longer the latency period is for a lagging indicator, the greater the likelihood for problems. That is, a lagging security metric with a short latency period or lag time is preferred over one with a long latency period, since any needed response to an observed change can take place earlier. It is important to recognize lagging indicators and, if they are used, to be prepared to handle the intrinsic delay and associated limitations.

Simple counts, when used as a security measure, can be especially hard to classify and interpret. For example, does an increase in the number of viruses detected by antivirus software serve as a leading indicator, because the increased activity indicates an elevated threat level; serve as a lagging indicator, because the increased activity demonstrates a highly proficient leading-edge antivirus mechanism; or serve as a coincident indicator, because the increased activity acts as a notification that other security-enforcing mechanisms are failing? Similarly, decreased activity may be because the antivirus mechanism is losing its effectiveness, other security-enforcing mechanisms are increasingly successful, or the system is simply not being subjected to as many attacks.

Many security metrics can be viewed as lagging indicators. The perspective on the initial security assessments of a system, whether they were done by a human evaluator, automated means, or some combination of the two, is likely to change eventually to reflect a lower security standing and higher associated risk. The main reason is that over time, understanding of a system and its related weaknesses and vulnerabilities deepens, especially in light of successful attacks on the system or other systems sharing similar characteristics that reveal previously unimagined or unexpected avenues of attack. A greater understanding eventually leads to updated assessments (e.g., through additional tests). While some assessment programs have a process to refresh or maintain initial security assessments, a significant delay nevertheless occurs. Stated another way, no

metric exists that can denote the state of security of a system in an absolute sense (Torgerson, 2007a; Torgerson, 2007b).

Patches and software updates that are intended to repair or improve the security-enforcing mechanisms can lessen the significance of a lagging indicator. Not only does reestablishment of the baseline system occur with each change, potentially increasing risk (Brenner, 2007; Keizer, 2008; Lemos, 2008; Markoff, 2008; Nagel, 2008), but an exact picture of the initial baseline becomes obscured and more complicated to track over time (e.g., (Storms, 2008)), making revised assessments impractical. That is, the presumed security posture of the initial baseline is never updated in light of increased understanding, potentially giving a false impression of the actual state. Several recent examples of dramatic shifts in our understanding of vulnerabilities in commonly used technologies, such as the Flash Player-ActionScript virtual machine (Dowd, 2008), OpenSSL cryptography (Dougherty, 2008a; Garfinkel, 2008), MD5 algorithm in Web certificates (Poulsen, 2008), and DNS protocol implementations (Dougherty, 2008b), and also the prevalence of zero-day vulnerabilities indicate that with hindsight, unrevised values of lagging indicators could be highly misleading.

Organizational Security Objectives

Organizations exist for different purposes, hold different assets, have different exposure to the public, face different threats, and have different tolerances to risk. Because of these and other differences, their security objectives can vary significantly. For example, a government organization that mainly handles data about individual citizens of the country (e.g., taxes or social insurance) has different objectives than a government organization that does not (e.g., commerce or education). Similarly, the security objectives of a government organization that prepares and disseminates information for public consumption are different from one that deals mainly with classified information for its own use. In addition, practical considerations apply—most organizations cannot afford financially to protect all computational resources and assets at the highest degree possible and must prioritize based on criticality and sensitivity.

Security metrics are generally used to determine how well an organization is meeting its security objectives. Since the security objectives of organizations can vary widely, it is reasonable to expect that the metrics required to make such an assessment for one organization would also be very different from those used for another. In other words, security is risk and policy dependent from an organizational perspective; the same platform populated with data at the same level of sensitivity, but from two different organizations, could be deemed adequate for one and inadequate for the other. The implication is that establishing security metrics that could be used for meaningful system comparisons between organizations would be extremely difficult to achieve (i.e., if achieved, they would have little real significance for some organizations).

Although security objectives are unique and tied to the goals and purpose of an organization, similarities in high level security objectives do exist between organizations performing similar work and are sometimes captured as best practices. Some steps, such as standard security profiles of organizational security requirements and criteria, can be

taken to standardize common sets of core requirements needed by comparable organizations and allow reuse of engineered solutions. However, at best they cover only a common subset of the entire picture and may focus mainly on technical metrics.

Qualitative and Quantitative Properties

Measurement of software properties in general has been difficult to accomplish. Many desired properties such as complexity, usability, and scalability are qualities that can be expressed in general terms, but difficult to define in objective terms that are useful in practice and provide a fully complete picture. For example, two well known software complexity measures, the McCabe cyclomatic complexity metric and Halstead's software science metric, determine program complexity directly from source code (Marco, 1997). Critics claim that the number of control paths McCabe used to determine the complexity of code is only one part of the total picture. Similarly, some consider the emphasis placed on lexical and textual measures by Halstead to be weak, because the structural or logic flow is ignored, and consider the assumptions used to derive complexity equations to be faulty; furthermore, the equations can be difficult to compute and practical use is limited to comparisons between versions of the same code. At best, certain properties of software that are assessed are able to capture only some facets of any desired quality.

The distinction between quantitative and qualitative security metrics is easily obscured (Henning et al., 2001). Qualitative assignments can be used to represent quantitative measures of security properties (e.g., low means no vulnerabilities found; medium, between one and five found; and high, more than five found). More often, numeric values are used to represent rankings that are otherwise qualitative (e.g., 1, 2, and 3, versus low, medium, and high). While the numeric difference between ranked values may be significant for some metrics, it may not be for others, which is often the case with security metrics. For example, the numeric difference between security rankings assigned through the analysis and reasoning of evaluators would likely not have any particular significance, other than to impart an ordering. In other words, the common scales of measurement (i.e., nominal, ordinal, interval, and ratio) and associated principles of use apply.

Quantitative valuations of several security properties may also be weighted and combined to derive a composite value (e.g., rating = .25 × rankingA + .75 × rankingB). Such compositions can, however, yield undesired results. For example, the Common Vulnerability Scoring System (CVSS) v1 formula produced a lack of diversity in scoring, such that many cases of vulnerabilities with different characteristics received the same scores, but were clearly at significantly different levels of severity, which eventuated the formula's revision (Reid et al., 2007).

Measurements of the Large Versus the Small

Security measurements have proven to be much more successful when the target of assessment is small and simple rather than large and complex. For example, a FIPS 140 evaluation, which focuses exclusively on cryptographic modules, generally requires less cost and time than a Common Criteria evaluation of a product that incorporates such modules. This is not too surprising, since larger systems generally have greater complexity and functionality. As the number of components in a system increases, the number of

possible interactions increases with the square of the number of components. Greater complexity and functionality typically relate inversely to security and require more scrutiny to evaluate.

Physical analogies apply here. The strength of a single concrete block of uniform material is more readily determined than a wall or structure composed of these blocks. The latter involve binding and reinforcing materials as well as height, design, and other architectural considerations. In other words, the different materials involved and the way in which they are composed determine the overall strength.

The composability problem in security is a long-standing problem—two systems, both of which are judged to be secure, can be connected together such that the resulting composite system is not secure. In theory, evaluated systems could be designed to be predictably composable such that the properties of the resulting composite system are easily determined. In practice, this has not occurred. Composability is somewhat like the philosopher's stone, a legendary substance that could change cheap metals into gold. A solution for building meaningful, complex software systems from components may exist theoretically; however, progress to date in achieving this goal has been limited, suggesting that composability is an elusive problem that may be solvable only in limited situations.

A technological breakthrough in composability would be a way to have security measurements of small systems directly contribute to the measurements of larger systems of which they are a part. In the absence of sound security metrics that can be used to assess the security of composed systems made from composable components with measured properties, the current high latency and expense in evaluating large systems can be expected to continue and limit the ability to perform cross-system comparisons in security.

POSSIBLE RESEARCH AREAS

The implication from the previous section is that the present state of security metrics largely involves qualitative lagging indicators that require subjective evaluation and may not necessarily coincide with an organization's security objectives. While this is a bleak picture, it does not mean that efforts to secure systems are pointless or that existing security metrics are useless. In order to improve the state of the art of security metrics, it can be argued that research efforts need to be focused on areas that satisfy one or more of the following factors:

- Determine good estimators of system security.
- Reduce reliance on the human element in measurement and inherent subjectivity.
- Offer a more systematic and speedy means to obtain meaningful measurements.
- Provide understanding and insight into the composition of security mechanisms.

Based on the past attempts, it is clear that reaching this objective will likely not be simple. Nevertheless, the opportunity exists to build on previous work and find ways around or through present limitations. The remainder of this section considers some possible

research areas to explore as a starting point. Its intent is not to be prescriptive, but rather to suggest a range of plausible activities that may hold promise.

Formal Models of Security Measurement and Metrics

Security measurement and metrics efforts that are conceived at a high level of abstraction and formalism are often difficult to interpret and apply in practice. Existing formalisms also pose difficulties to reconcile with actual operational environments where software patches, version updates, and configuration setting changes take place regularly. The absence of formal security models and other formalisms needed to improve the relevance of security metrics to deployed systems have hampered progress. Having formal models that depict security properties of operational IT systems and incorporate relevant objects of significance to system security measurement would be a useful contribution. The research goal is to establish formal models with a level of detail sufficient to enable realistic predictions of operational system behavior and portray security measurements accurately.

An example of the type of work expected is the effort on attack surface metrics done at Carnegie Mellon University (Manadhata and Wing, 2005; Manadhata et al., 2007). A formal model is defined from an intuitive notion of a system's attack surface (i.e., the ways in which the system can be entered and successfully attacked). The formal model is characterized in terms of certain system resources—those methods (e.g., application programming interfaces), channels (e.g., sockets), and data items (e.g., input strings) that an attacker can use to cause damage to the system. A surface measurement model can then be applied to compare attack surface measurements along each of the three dimensions and used to determine whether one system is more secure than another. The measurements entail making estimates of the damage potential and effort required, which respectively are the level of harm the attacker can cause to the system when using the resource in an attack, and the amount of work needed by the attacker to obtain the necessary access rights to be able to use the resource in an attack.

Model checking, a method for formally verifying properties of systems, is another area where a formal technique may have potential for use in security metrics. Properties of interest are expressed as logic formulas. A representation of the system (e.g., a formal or program language description) is traversed and checked to determine whether a property holds, using efficient symbolic algorithms. Specifying security properties of interest to be automatically checked could automate certain types of evaluation. For example, model checking has been used to identify an important class of security vulnerabilities in the Red Hat Linux 9 distribution. Vulnerabilities were expressed as temporal safety properties that described the conditions under which the vulnerabilities were absent from the software programs that made up the distribution (Schwarz et al., 2005). Programs that violated a property were not verifiable and flagged for review. Model checking has also been used to verify security properties of access control policies (Guelev et al., 2004; Reith et al., 2007).

Research into formal models could also benefit the design of decision support systems that manage security infrastructure risks by strongly embracing security metrics in determining security investments. Developing decision support models that incorporate technical and organizational aspects of a system and also quantify the utility of a security investment

based on established principles would be the focus. The high level of detail and complexity of such models would likely depend strongly on the availability of essential empirical data. For instance, the QuERIES methodology and underlying attack-versus-protect economic model, which offers a means to determine investment levels and strategies for protecting intellectual property in complex systems, requires such data as asset valuations, the costs of protection and theft of assets, and theft probabilities (Carin et al., 2008).

Historical Data Collection and Analysis

Predictive estimates of the security of software components and applications under consideration should be able to be drawn from historical data collected about the characteristics of other similar types of software (e.g., code quality and complexity) and the vulnerabilities they experienced (e.g., number and severity). For example, models that define software reliability in terms of the number of faults in a body of code have been applied to the OpenBSD operating system to estimate characteristics about the number of faults remaining in the system and when those faults may cause failures (Ozment and Schechter, 2006). Empirical evidence also exists that features correlate with vulnerabilities and that components containing past vulnerabilities can be used to predict with reasonable accuracy vulnerable components (i.e., those having undiscovered vulnerabilities) based on their features (Neuhaus et al., 2007). At the very least, insight into security measurements would likely be gained by applying analytical techniques to such historical collections to identify trends and correlations (e.g., via statistical methods), to discover unexpected relationships (e.g., via data mining), and to reveal other predictive interactions that may exist (e.g., via visual analytics).

The research goal is to identify characteristics of software components and applications in the collection that can be extracted and used to predict the security condition of other software. Available open source software repositories can serve as a starting point for the data collection, but will require additional effort to incorporate vulnerability information and identify the points at which known vulnerabilities first appeared in the code set. With a large enough data collection, reliable estimators of the overall effectiveness of software might be derived that could be used in security assessments of software.

A historical data collection has other potential benefits and use. It can serve as a basis for confirming the validity of independently proposed security measurements and methods of measurement, identifying whether associated measures are leading, lagging, or coincident indicators, and establishing estimates of latency and uncertainty for useful indicators. It can also serve as a means to investigate new methods of detecting and discovering both expected and unexpected relationships for use as estimators, and to develop new and improved mathematical and computational methodologies to improve the visual analytics of the data collection. A selected subset of the historical data collection could also be used as reference materials for training or rating the proficiency of security evaluators.

Artificial Intelligence Assessment Techniques

The field of Artificial Intelligence (AI) involves the design and implementation of systems that exhibit capabilities of the human mind, such as reasoning, knowledge, perception,

planning, learning, and communication. AI encompasses a number of sub-disciplines including machine learning, constraint satisfaction, search, agents and multi-agent systems, reasoning, and natural language engineering and processing. While the use of AI has met with both successes and defeats, its application in aspects of security metrics might prove beneficial, particularly as a means for reducing subjectivity and human involvement in performing security assessments.

The research goal is to identify areas of security evaluations that could be performed using AI or AI-assisted techniques and demonstrate their use. Dealing with uncertainty and inconsistency has been a part of AI from its origins. More recently, AI systems are beginning to emerge that can independently formulate, refine, and test hypotheses from observed data to uncover fundamental properties (Figueroa, 2009). Technologies for managing uncertainty and inconsistency have already been used in areas such as the ranking algorithms used in web search engines and are being applied to the broader area of homeland security (Chen and Wang, 2005). The expectation is that AI technologies can play a similarly important role in the context of security assessments.

For example, fuzzy logic is a superset of conventional logic that has been extended to handle the uncertainty in data. Fuzzy logic is useful in situations where it is difficult to make a precise statement about system behavior and has been applied successfully to the area of risk management (Dondo, 2007; McGill and Ayyub, 2007; Shah, 2003). In these applications, qualitative risk descriptors, such as High, Medium, and Low, are able to be assigned to a range of values and calibrated as continuous quantitative input. In one case, fuzzy logic was used to assess the relative risk associated with computer network assets by ranking vulnerabilities with regard to the potential risk exposures of assets and networks (Dondo, 2007). In another case, using fuzzy logic provided risk analysts more information than qualitative approaches for ranking risks, to help them more effectively manage operational risks (Shah, 2003).

Practicable Concrete Measurement Methods

The current practice of security assessment, best illustrated by lower level evaluations under the Common Criteria, emphasizes the soundness of the evaluation evidence of the design and the process used in developing a product over the soundness of the product implementation. The rationale is that without a correct and effective design and development process, a correct and effective implementation is not possible. While this is true, the emphasis on design and process evidence versus actual product software largely overshadows practical security concerns involving the implementation and deployment of operational systems. The research goal is to devise methods of measurement that address vulnerabilities occurring in implementation and deployment and complementing existing security assessment practices that emphasize design and development process evidence. The intent is to be able to detect vulnerabilities that otherwise would escape detection.

Code analysis has been a traditional way to examine code automatically and identify software problems that stem from violations in standard coding practices. Code analysis tools that are capable of identifying different types of potentially exploitable vulnerabilities have also been developed for security (Chandra et al., 2006; Michael and Lavenhar, 2006).

They provide a good example of one type of concrete measurement method possible, which is reproducible, repeatable, relevant, and timely. Although such security analysis tools provide highly useful results, they detect vulnerabilities incompletely and generate significant numbers of false positives, allowing room for improvement (e.g., (Kirkland and Salem, 2006; Schwarz et al., 2005)).

Relatively little is known about software behavior or misbehavior after deployment (Bowring et al., 2002; Liblit, 2004). Traditional means of feedback such as error or vulnerability reporting often are imprecise, contain inaccuracies, and involve delay, making it difficult to form a complete picture representative of the situation. Cooperative Bug Isolation (CBI) is an innovative technique that uses software instrumentation and sparse random sampling to gather information from the actual operation of deployed software (Liblit, 2004). Algorithms have been developed for finding and fixing software errors based on statistical analysis of sparse feedback data. While developed for debugging purposes, the potential seemingly exists to extend the framework as a method of measurement for security through selective predicate instrumentation.

Various forms of black box security testing offer another example of a possible type of concrete measurement method. For example, fuzzing is a type of fault injection technique that involves sending various types of pseudorandom data to available interfaces to discover unknown flaws present in programs and systems (Juranić, 2006). Fuzzing techniques have been shown to be an effective means for detecting security vulnerabilities. Robustness testing is a closely related technique, which involves determining how well a software system or component functions when subjected to invalid inputs or other conditions that induce stress on the implementation. Incorrect behaviors exhibited under stressful conditions can often lead to potential security exploits such as denial of service (Kaksonen, 2001; Röning et al., 2002).

Intrinsically Measurable Components

Development of computing components that are inherently attuned to measurement would be a significant improvement in the state of the art of security metrics. The idea is to build components that clearly exhibit properties that matter to security. The research goal lies mainly on issues of mechanism and component design that facilitate or promote security measurement. For example, preparing strength of mechanism arguments in conjunction with the design and development of a security-enforcing component might be one method. Lower and upper bounds could be established, similar to the way performance bounds are calculated for sorting, matching, and other essential algorithms used in computing.

Cryptographic mechanisms are an area where research results exist and bounds on the effort required to breach components could be determined, under specific assumptions. Extending this type of analysis to trust mechanisms is a more challenging problem, but one with significant benefits, if achieved. Components that rely on certain surety mechanisms, such as authentication modules designed for passwords or biometric modules for fingerprints, lend themselves to certain types of strength analysis (e.g., average attack space), and results already exist that could be applied. Finally, techniques for strength

analysis might also be drawn from physical security, such as metrics used by industry to evaluate safe and vault security and identify weaknesses leading to failure (Blaze, 2004).

Another possible area involves the application of evaluation criteria to system design. One possibility is to determine whether a methodology can be formulated for stipulating how to compose individually evaluated components of systems, such that the security of the overall system is ensured. The Trusted Network Interpretation (TNI) served this purpose for the TCSEC, but it was tied to Department of Defense security policies and a small standardized subset of security profiles. Attempting to devise a similar type of methodology for the Common Criteria would be more difficult because of the vast range of protection profiles that are able to be specified and which already exist. Nevertheless, it may be possible to narrow that scope to a definable composable subset through the systematic application of rules, principles, and logic.

CONCLUSION

The security metrics area poses hard and multi-faceted problems for researchers. Quick resolution is not expected and the likelihood is that not all aspects of the problem are resolvable. Furthermore, only some of those aspects that are resolvable may be able to be done satisfactorily, meeting expectations of repeatability, reproducibility, relevance, timeliness, and cost. Several factors impede progress in security metrics:

- The lack of good estimators of system security.
- The entrenched reliance on subjective, human, qualitative input.
- The protracted and delusive means commonly used to obtain measurements.
- The dearth of understanding and insight into the composition of security mechanisms.

This paper proposes several lines of research that address these factors and could help to progress the state of the art in security metrics. The following research areas were identified:

- Formal Models of Security Measurement and Metrics
- Historical Data Collection and Analysis
- Artificial Intelligence Assessment Techniques
- Practicable Concrete Measurement Methods
- Intrinsically Measurable Components.

Each area in itself is quite extensive and requires a commitment to sustain the long-term research and development needed to be successful.

REFERENCES

Bellovin, S. (2006). On the Brittleness of Software and the Infeasibility of Security Metrics, IEEE Security and Privacy, Volume 4, Issue 4, July-August

Berinato, S. (2005). A Few Good Information Security Metrics, CSO Magazine, http://www.csoonline.com/article/220462/A_Few_Good_Information_Security_Metrics?contentId=220462&slug=&

Blaze, M. (2004). Safecracking for the Computer Scientist, Draft Document, http://www.crypto.com/papers/safelocks.pdf

Bowring, J., Orso, A., Harrold, M. (2002). Monitoring Deployed Software Using Software Tomography, ACM SIGPLAN-SIGSOFT Workshop on Program Analysis for Software Tools and Engineering, Charleston, South Carolina

Brenner, B. (2007). Windows Admins Feel Post-Patch Tuesday Pain, SearchSecurity.com, October 19, 2007, http://searchsecurity.techtarget.com/news/article/0,289142,sid14_gci1277683,00.html

Carin, L., Cybenko, G., Hughes, J. (2008). Cybersecurity Strategies: The QuERIES Methodology, IEEE Computer, Vol. 41, No. 8

Center for Internet Security (CIS) (2008). The CIS Security Metrics Service, http://securitymetrics.org/content/attach/Metricon3.0/metricon3-kreitner%20handout.pdf

Chandra, P., Chess, B., Steven, J. (2006). Putting the Tools to Work: How to Succeed with Source Code Analysis, IEEE Security & Privacy, vol. 4, no. 3, pp. 80-83

Chen, H., Wang, F. Y. (2005). Artificial Intelligence for Homeland Security, IEEE Intelligent Systems, vol. 20, no. 5, pp. 12-16

Commission of the European Communities (CEC) (1991). Information Technology Security Evaluation Criteria (ITSEC), Harmonised Criteria of France - Germany - the Netherlands - the United Kingdom, CEC Directorate XIII/F SOG-IS, http://www.iwar.org.uk/comsec/resources/standards/itsec.htm

Common Criteria Portal (2006). Common Criteria for Information Technology Security Evaluation Part 1: Introduction and general model, Version 3.1 Revision 1, http://www.commoncriteriaportal.org/files/ccfiles/CCPART1V3.1R1.pdf

Department of Defense (DoD) (1985). Trusted Computer System Evaluation Criteria, DoD 5200.28-STD, http://csrc.nist.gov/publications/history/dod85.pdf

Dondo, M. (2007). A Fuzzy Risk Calculations Approach for a Network Vulnerability Ranking System, Technical Memorandum 2007-090, Defence R&D Canada – Ottawa, http://www.ottawa.drdc-rddc.gc.ca/docs/e/TEO-TM-2007-090.pdf

Dougherty, C. (2008a). Debian and Ubuntu OpenSSL Packages Contain a Predictable Random Number Generator, Vulnerability Note VU#925211, U.S. Computer Emergency Readiness Team, https://www.kb.cert.org/vuls/id/925211

Dougherty, C. (2008b). Multiple DNS Implementations Vulnerable to Cache Poisoning, Vulnerability Note VU#800113, U.S. Computer Emergency Readiness Team, http://www.kb.cert.org/vuls/id/800113

Dowd, M. (2008). Application-Specific Attacks: Leveraging the ActionScript Virtual Machine, IBM Global Technology Services, http://documents.iss.net/whitepapers/IBM_X-Force_WP_final.pdf

Figueroa, J. (2009). Discovery Systems Check Their Own Facts, In the News, IEEE Intelligent Systems, Vol. 24, No. 3

Garfinkel, S. (2008). Alarming Open-Source Security Holes: How a programming error introduced profound security vulnerabilities in millions of computer systems, MIT Technology Review, http://www.technologyreview.com/Infotech/20801/?a=f

Gray, M. (1999). Applicability of Metrology to Information Technology, Journal of Research of the National Institute of Standards and Technology, Vol. 104, No. 6, http://nvl.nist.gov/pub/nistpubs/jres/104/6/j46gra.pdf

Guelev, D. P., Ryan, M., Schobbens, P. Y. (2004). Model Checking Access Control Policies, Proceedings of the 7th Information Security Conference, Palo Alto, CA

Henning, R., et al. (2001). Proceedings of the Workshop on Information Security System Scoring and Ranking, Applied Computer Security Associates, Williamsburg, Virginia, http://www.acsac.org/measurement/proceedings/wisssr1-proceedings.pdf

INFOSEC Research Council (2005). Hard Problem List, http://www.cyber.st.dhs.gov/docs/IRC_Hard_Problem_List.pdf

The Institute for Information Infrastructure Protection (I3P) (2009). National Cyber Security Research and Development Challenges Related to Economics, Physical Infrastructure and Human Behavior: An Industry, Academic and Government Perspective, http://www.thei3p.org/docs/publications/i3pnationalcybersecurity.pdf

International Systems Security Engineering Association (ISSEA) (2008). SSE-CMM: Systems Security Engineering Capability Maturity Model, http://www.sse-cmm.org/metric/metric.asp

Jelen, G. (2000). SSE-CMM Security Metrics, The National Institute of Standards and Technology (NIST) and Computer System Security and Privacy Advisory Board (CSSPAB) Workshop, Washington, D.C.

Juranić, L. (2006). Using fuzzing to Detect Security Vulnerabilities, INFIGO-TD-01-04-2006, Infigo Information Security, http://www.infigo.hr/files/INFIGO-TD-2006-04-01-Fuzzing-eng.pdf

Kaksonen, R. (2001). A Functional Method for Assessing Protocol Implementation Security, VTT Publications 448, Technical Research Centre of Finland, http://www.vtt.fi/inf/pdf/publications/2001/P448.pdf

Keizer, G. (2008). Hackers Attack Newest Windows Patch, PC World, http://www.pcworld.com/businesscenter/article/144486/hackers_attack_newest_windows_patch.html

Kirkland, D., Salem, L. (2006). BogoSec: Source Code Security Quality Calculator, IBM, http://download.boulder.ibm.com/ibmdl/pub/software/dw/linux/l-bogosec.pdf

Lemos, R. (2008). Patches Pose Significant Risk, Researchers Say, SecurityFocus, http://www.securityfocus.com/news/11514

Liblit, B. (2004). Cooperative Bug Isolation, PhD Thesis, University of California, Berkeley, http://pages.cs.wisc.edu/~liblit/dissertation/dissertation.pdf

Littlewood, B. et al. (1993). Towards Operational Measures of Computer Security, Journal of Computer Security, vol. 2, no. 2-3, pp. 211-230

Manadhata, P., Wing, J. M. (2005). An Attack Surface Metric, CMU-CS-05-155, Carnegie Mellon University, http://reports-archive.adm.cs.cmu.edu/anon/2005/CMU-CS-05-155.pdf

Manadhata, P., Tan, K., Maxion, R., Wing, J. (2007). An Approach to Measuring a System's Attack Surface, CMU-CS-07-146, Carnegie Mellon University, http://reports-archive.adm.cs.cmu.edu/anon/2007/CMU-CS-07-146.pdf

Marco, L. (1997). Measuring Software Complexity, Enterprise Systems Journal, http://cispom.boisestate.edu/cis320emaxson/metrics.htm

Markoff, J. (2008). Leaks in Patch for Web Security Hole, The New York Times, http://www.nytimes.com/2008/08/09/technology/09flaw.html?_r=1&oref=slogin

McGill, W., Ayyub, B. M. (2007). Multicriteria Security System Performance Assessment Using Fuzzy Logic, The Journal of Defense Modeling and Simulation (JDMS): Applications, Methodology, Technology, Special Issue: Homeland Security, vol. 4, no. 4, http://www.scs.org/pubs/jdms/vol4num4/McGill.pdf

Michael, C., Lavenhar, S. (2006). Source Code Analysis Tools – Overview, Cigital, Inc., https://buildsecurityin.us-cert.gov/daisy/bsi/articles/tools/code/263-BSI.html

Nagel, B. (2008). Excel Patch Causes Miscalculations, Government Computer News, http://www.gcn.com/online/vol1_no1/45992-1.html

Neuhaus, S., Zimmermann, T., Holler, C., Zeller, A. (2007). Predicting Vulnerable Software Components, ACM Conference on Computer and Communications Security (CCS '07), Alexandria, Virginia, http://www.st.cs.uni-sb.de/publications/files/neuhaus-ccs-2007.pdf

Ozment, A., Schechter, S. (2006). Milk or Wine: Does Software Security Improve with Age?, 15th USENIX Security Symposium, Vancouver, Canada, http://www.usenix.org/events/sec06/tech/full_papers/ozment/ozment.pdf

Poulsen, K. (2008). Researchers Use PlayStation Cluster to Forge a Web Skeleton Key, Wired Magazine, http://blog.wired.com/27bstroke6/2008/12/berlin.html

Reid, G., Mell, P., Scarfone, K. (2007). CVSS-SIG Version 2 History, Forum of Incident Response and Security Teams, http://www.first.org/cvss/history.html

Reith, M., Niu, J., Winsborough, W. (2007). Apply Model Checking To Security Analysis in Trust Management, C107-0030, University of Texas at San Antonio, http://stinet.dtic.mil/cgi-bin/GetTRDoc?AD=ADA462754&Location=U2&doc=GetTRDoc.pdf

Röning, J., Laakso, M., Takanen, A., Kaksonen, R. (2002). PROTOS - Systematic Approach to Eliminate Software Vulnerabilities, Invited presentation at Microsoft Research, Seattle, Washington, http://www.ee.oulu.fi/research/ouspg/protos/sota/MSR2002-protos/index.html

Savola, R. M. (2007). Towards a Taxonomy for Information Security Metrics, International Conference on Software Engineering Advances (ICSEA 2007), Cap Esterel, France

Schwarz, B., Chen, H., Wagner, D., Morrison, G., West, J. (2005). Model Checking an Entire Linux Distribution for Security Violations, 21st Annual Computer Security Applications Conference, Tucson, Arizona, http://www.acsac.org/2005/papers/165.pdf

Science Applications International Corporation (SIAC) (2007). Microsoft Windows Server 2003, XP Professional and XP Embedded Security Target, Version 3.0, SIAC

Common Criteria Testing Laboratory, http://www.commoncriteriaportal.org/files/epfiles/20080303_st_vid10184-st.pdf

Shah, S. (2003). Measuring Operational Risk Using Fuzzy Logic Modeling, International Risk Management Institute, Inc. (IRMI), http://www.irmi.com/Expert/Articles/2003/Shah09.aspx

Storms, A. (2008). Many Microsoft Bulletins Replaced; Bigger Set of Kill Bits Issued, nCircle, http://blog.ncircle.com/blogs/sync/archives/2008/08/many_microsoft_bulletins_repla.html

Torgerson, M. (2007). Security Metrics, 12th International Command and Control Research and Technology Symposium, Newport, Rhode Island, http://www.dodccrp.org/events/12th_ICCRTS/CD/html/presentations/108.pdf

Torgerson, M. (2007). Security Metrics for Communication Systems, 12th International Command and Control Research and Technology Symposium, Newport, Rhode Island, http://www.dodccrp.org/events/12th_ICCRTS/CD/html/papers/108.pdf

Vaughn Jr., R., Henning, R., Siraj, A. (2002). Information Assurance Measures and Metrics – State of Practice and Proposed Taxonomy, 30th Hawaii International Conference on System Sciences, Big Island, Hawaii, http://csdl2.computer.org/comp/proceedings/hicss/2003/1874/09/187490331c.pdf