Regular Paper

Measuring Systems Security

Jennifer Bayuk* and Ali Mostashari

Stevens Institute of Technology, Hoboken, NJ 07030

Received 27 June 2011; Revised 2 October 2011; Accepted 6 January 2012, after one or more revisions
Published online 23 October 2012 in Wiley Online Library (wileyonlinelibrary.com).
DOI 10.1002/sys.21211

ABSTRACT

Security metrics have evolved side by side with the advent of security tools and techniques. They have been derived from the techniques rather than specified as system requirements. This paper surveys the evolution and state of the practice of security metrics from both a technical and historical perspective. It describes the evolution of currently popular security metrics, and classifies them to illustrate their utility in systems engineering verification and validation activities. It provides criteria with which to evaluate security metrics based on system purpose and architecture. The criteria are illustrated using a case study of Cloud System security. © 2012 Wiley Periodicals, Inc. Syst Eng 16: 1–14, 2013

Key words: systems security; systems engineering; computer security; security metrics

1. CONCEPTS IN SYSTEMS SECURITY

Though most systems engineers have some experience with security metrics, security is not a focus for the profession. There is no set of security metrics that has repeatedly demonstrated utility in an engineering context. A recent issue of *INCOSE Insight* dedicated to security expressed concern that the security metrics community "does not want to address any fundamental challenges like developing a theoretical basis for measurement, identifying basic underlying things that might be worth measuring, or providing any sort of mathematical basis for doing anything with the measurements they provide" [Cohen, 2011, p. 31] The reasons why the utility of traditional security metrics may have been dismissed by any given systems engineer will vary with the experience of the individual. Experience matters because different systems engineers may have been presented with entirely different sets of measures under the name of "Security Metrics." These sets may have been compiled in support of decisions concerning different aspects of security.

*Author to whom all correspondence should be addressed (e-mail: jennifer@bayuk.com).

Systems Engineering Vol. 16, No. 1, 2013

To illustrate different aspects of security that may be measured differently, we use a systems engineering job aid used to make sense of complex concepts, a systemigram [Boardman and Sauser, 2008]. A systemigram is read from left to right, top to bottom. The system to be defined is placed in the top left, the mission or purpose of the system is placed in the bottom right. Concepts that assist in understanding the system are placed in between and linked using verbs. The systemigram in Figure 1 was composed by a team of security systems engineers collaborating on a research roadmap. They were challenged to come up with a shared definition of security, one that would be recognized within the security community. Hence the criteria for the definition of security was that it be "face-valid," as the measurement of face validity reflects general agreement in the layperson's opinion that a given measurement technique is suitable or unsuitable for its expected use [Nevo, 1985]. The line that reads from the top left to the bottom right is that definition, and, in the context of a systemigram, it is called the "mainstay" thread. The mainstay is a high-level concept definition that is generally agreed by those who best understand the system to be defined. However, a complex system will typically be understood in specialized contexts, and the systemigram tool supports that. Figure 2 shows different perspectives on system security that enrich the skeletal concept outlined in the mainstay. The portion of Figure 2 that links to the circle labeled "infrastruc-

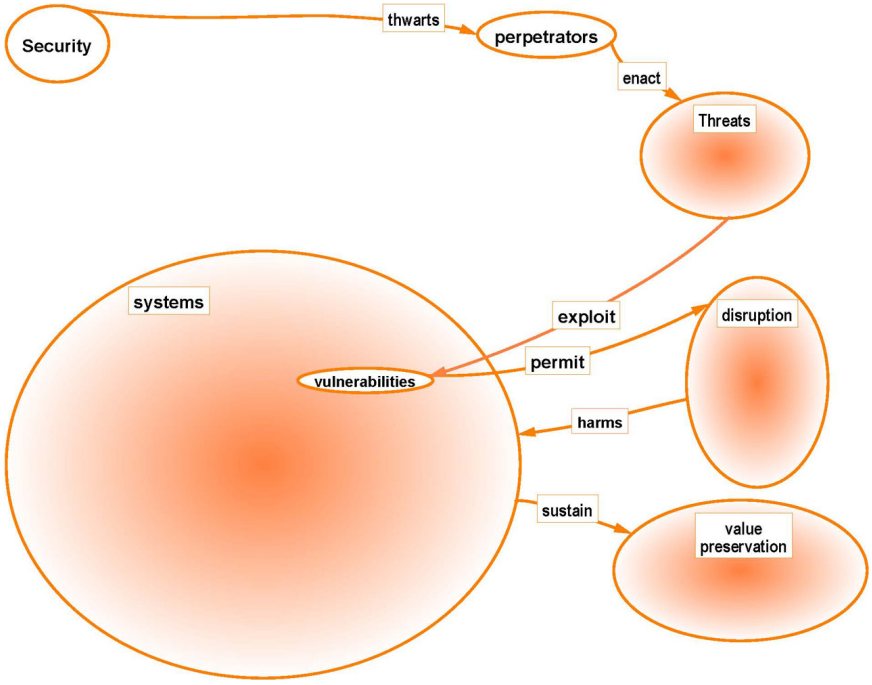

Figure 1. A systemigram mainstay for the concept of systems security. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

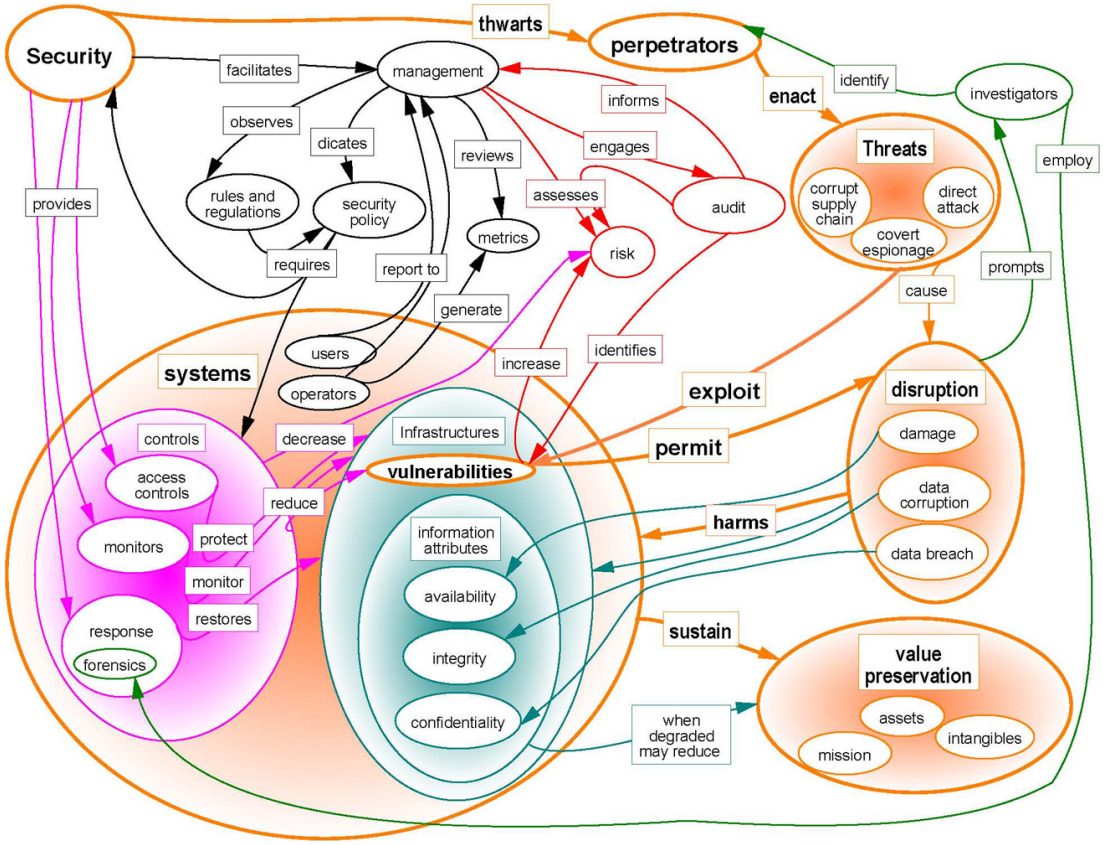

Figure 2. Systems security systemigram with multiple perspectives. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

tures" demonstrates that those who work in information security tend to conceptualize security in terms of information attributes that are subject to similar types of threats and disruptions. The portion labeled "controls" adopts the point of view of a network security engineer, who typically views security in terms of preventive, detective, and corrective controls designed to minimize vulnerabilities and reduce risk. The "management" portion adopts the perspective of security from the point of view of technology governance. The "audit" and "investigators" portions similarly represent the corresponding viewpoints. The full systemigram provides a holistic view of security that depicts it as having multiple stakeholders.

Each of these systems perspectives has an associated set of security metrics of interest to the faction of the security community who created them. Figure 3 provides some examples of security standards that have been used as a basis for metrics by each community of stakeholders that correspond to systemigram threads. Originally conceived as job aids, these collected best practices have become the defacto basis for security audit and assessment, and in fact are criticized by auditors when they cannot be readily adopted for that purpose [Ross, 2006; Oud, 2005]. Thus systems engineers are often presented with one or more of these standards documents as if they were systems engineering requirements. For example, a recent systems engineering textbook with multiple respected systems engineer authors describes security as "related to system attributes that enable it to comply with regulations and standards" (e.g., in Larson et al. [2009: 114], security requirements are said to originate from industry standards). As standards are assumed to be verifiable, they have *ipso facto* become requirements, and by extension, a basis for security metrics.

Each of the communities of security professionals who created the standards documents drew on considerable successful professional practices, and these communities include prestigious security subject matter experts. It is not the intention of this article to question the experience or judgment

Systemigram Thread	Example Standard	Stakeholder Communities
Information Assurance	• Recommended Security Controls for Federal Systems, NIST 800-53	Technology and data center security managers
Network Security	• International Telecommunications Union X.805 • National Vulnerability Database	Network engineers and security architects
Security Management	• ISO/IEC 27000 Series	Security management consultants and assessment organizations
Information Systems Audit	• Control Objectives for Information Technology (COBIT)	Information Systems Audit and Control Association
Forensics	• Handbook for Computer Security Incident Response Teams (CSIRTs) • NIST 800-61	Computer Incident Response Teams

Figure 3. Systemigram map to security standards.

invested in creating security standards. Rather, we try to help systems engineers see where there may be value in evaluating the security metrics that these experts have produced. To provide such a perspective, we introduce the provenance of some of these standards.

2. HISTORY OF SECURITY METRICS

Even before electronic systems, physical security standards existed that concerned the strength of materials required to thwart a design-basis threat (DBT), and these continue to support the physical threat surfaces of our systems [Garcia, 2008]. The first computer security metrics did not actually measure any attribute of a system itself, but instead confirmed its correct operation by comparing physical batches of input to output and manually performing electronic calculations to ensure that the system was functioning properly. As system security auditors became more sophisticated at designing tests of correct operation, this input-to-output comparison was deprecated, and derisively referred to as "auditing around the computer" [Bayuk, 2005]. Since the early 1970s, members of the Electronic Data Processing Audit Association, which has evolved into the Information Systems Audit and Control Association, have shared tests and practices for determining whether management properly controls the deployment and operation of automated systems. These include both physical and electronic security measures. As security is essential to ensure that systems are restricted to authorized use, security testing has always played a significant role in systems audit practices [Bayuk, 2005]. As security management practices evolved, system security audit standards followed, because the audit community continually updates audit best practices standards in correspondence with security management practices.

The first document that was considered a systems security technology standard was the Orange Book, and undertaking of the then US National Bureau of Standards (NBS), the organization that later became the National Institute of Standards and Technology (NIST). The Orange Book standardized many of the security terms we take for granted today, like identification, authentication, and authorization. It introduced the notion of a trusted path that enforces integrity of communication between user commands and systems processing of those commands. It specified a hierarchy of system security features. The lowest was minimal, or none. The most commonly achieved was discretionary; that is, users had to be granted access to data and functions but could share them. The highest was mandatory access control, a requirement that subject classification level match labels on the data to which they have access, and also that all access was controlled and auditable. In addition, the design had to be validated using formal methods. Before it was even published, people were designing tests to check for the different types of technology features that are recommended by the Orange Book, so the Orange Book became the original systems security certification and accreditation standard. However, there were drawbacks to using the Orange Book as a system security certification because the scope of Orange Book testing was a single device, a technical target of evaluation or assessment (TTOE or TTOA). By the time the Orange Book was widely known, nearly all computing devices of interest were net-

worked, but many of the Orange-Book-certified systems did not include security features rich enough to maintain their Orange Book compliance ratings once they were augmented to accept TCP/IP input. Firewalls external to the system became an acceptable method of maintaining network security, and these devices incorporated only a subset of Orange Book requirements.

By the early 1990s, the recognition that security features requirements were dependent on operational context prompted NIST to follow the Orange Book with a project called "Common Criteria" [Merkow and Breithaupt, 2005]. The Common Criteria allowed product owners to be selective in identifying a set of security functions, allowed the system designers to specify their own "Security Target," and specifies an evaluation mechanism designed to ensure that the Security Target is met. As in the Orange Book, the Common Criteria specifies hierarchical levels of validation, and the resulting security rating reflects the level of testing that was successful. The lowest level is functional, followed by structural, and then methodical. Higher ratings may be obtained if it can be demonstrated that the product was methodically designed, and even higher by being semiformally or formally verified. The Common Criteria has since evolved into an international standard [ISO/IEC, 2009a].

Also in the early 1990s, organizations attempting to use the Orange Book and Common Criteria as a design guide determined that it was not enough for a computer to have security features, the features had to be configured and operated properly in order for the organization to achieve the desired level of security from the systems that had the features. Simply to know devices had passed tests was not an appropriate measure of whether they have been integrated properly to achieve system security. This recognition led a diverse set of volunteers from a wide variety of organizations to participate in the development of the System Security Engineering Capability Maturing Model (SSE-CMM). Also now an International Standard, this model is a system development lifecycle (SDLC) planning exercise designed to ensure that security requirements have been properly addressed in the design phase, and thereby to ensure that successful testing against security requirements would be included as both quality assurance and operational deployment tollgates [ISO/IEC, 2002]. In addition to measuring whether security planning processes are reliably executed and successful, the SSE-CMM provides systems engineers with a checklist of security controls to consider at each stage of the development lifecycle, from requirements to disposal. The process measurements and associated control checklists are used to derive security metrics.

Though the SSE-CMM does include guidance concerning considerations for secure system operations, in practice, the root cause of many system security failures are due to lapses in executing operational process as opposed to planning them. In an attempt to provide guidance on how to avoid these security management lapses, NIST proscribes ways that security should be managed in computer systems. Most of the security management metrics we have today follow the general strategy of the 1995 NIST Computer Security Handbook [NIST, 1995]. The handbook addressed organizational accountability for security risk management as well as management controls such as policy, education, and incident response. This handbook evolved into the Recommended Security Controls for Federal Information Systems [Ross et al., 2007] that has become the basis for Federal Information Security Management Act (FISMA) regulatory guidance [US Congress, 2002].

While NIST was composing the Computer Security Handbook, in the UK, security professionals were converging on British Standards for security management. The UK approach aimed for a higher level of management accountability, and placed more focus on management planning for security, but otherwise generally recommended the same types of controls as the NIST version [DTI/CCSC, 1995]. The UK security standard has evolved into the International Standards (ISO/IEC) 27000 Series [ISO/IEC, 2005a, 2005b, 2008, 2009b]. This series has become the basis for security management metrics as well as for ISO Security Certification exercises.

These security management practices evolved in an era of recurring destructive computer worms and viruses. Since the first such program was unleashed in 1988, a US government funded Computer Emergency Response Team (CERT) had been steadily working to develop incident response procedures, which have been since codified as standards [Allen, 2001]. Prevent, detect, respond has always been a mantra in the physical first responder community, and its systems corollary is prevent, detect, recover. As the ability to effectively recover relies on root cause analysis, this type of standard introduced metrics for log collection and automated change detection.

By the 1990s, the best method of thwarting viruses was to rely on antivirus software to identify and eradicate each virus. However, because multiple viruses would be written to exploit the same system software vulnerability, it was more efficient to continually update operating systems to eliminate known the software vulnerabilities that the viruses exploited. To keep track of all these vulnerabilities, NIST proposed a dictionary of publicly known security vulnerabilities and exposures using common terms and names, and this has been realized under a project managed by The MITRE Corporation, the Common Vulnerability Enumeration (CVE) [MITRE, 2011a]. The CVE is one of several software vulnerability-related enumeration categories under a more general NIST effort known as the National Vulnerability Database [MITRE, 2011b]. Another example of an NVD enumeration is a Common Weakness Enumerator (CWE). This lists common mistakes that software developers make that expose systems to hacking attempts, such as the lack of input validation. Security software developers use NVD efforts to create security metrics by scanning system to see if they have any vulnerability on the list, and counting them. These security testing tools have been aptly christened "badness-o-meters" [McGraw, 2006]. The word badness-o-meter is appropriate because, if you have NVD vulnerabilities, you can tell your system security is bad, but the absence of an NVD list of vulnerabilities in your system does not necessarily mean that your security is good. There could be software vulnerabilities that are not yet catalogued by NVD, or there could be design flaws that expose systems despite their use of secure software. The NVD community has established met-

rics that correspond to its vulnerability lists in the form of a Common Vulnerability Scoring System [Mell, Scarfone, and Romanosky, 2007]. These metrics provide a scale that only has bad security on it, but no measure of good security.

3. TYPES OF SECURITY METRICS

As metrics are fundamentally tools for decision-making, the perspective of the decision-maker is important. However, metrics that are useful for analyzing one aspect of security may not be useful for analyzing another aspect. Even within a single perspective, such as that of a network security engineer, security metrics that assist in determining whether a given system is secure may not be useful in determining whether a different system is secure. For example, a set of security metrics that are valid in ascertaining the security of a news media website will appear woefully inadequate in ascertaining the security of an electronic banking website.

Nevertheless, today's security standards adopt a one-size-fits-all approach in both perspective and content. The system of interest is assumed to be well understood by the standards-reader. It is assumed that there is a decision-maker who is qualified to judge the potential impact of security weakness and also is in a position to accept responsibility for security risks. They generally provide a process for examining the system of interest with the goal of minimizing vulnerabilities. Figure 4 is a systemigram that depicts security standards generically. Figure 5 supplements the systemigram elucidation of how security standards work. They generally recommend reducing a system into inventory components that can be assessed independently for risk, and then these risk assessments are composed into an overall metric that dictates some recommended controls. Figures 4 and 5 simplify but do not exaggerate the simplicity of the structure of today's security standards. Today's standards contain little guidance by way of methods, processes, or tools with which to analyze systems or system components, nor to create relevant systems-level security requirements. They do not even provide guidance on how to question a system owner or operator as to the judgment on whether a control is necessary. Hence, they are reducible to instructions in the technical implementation of best practices.

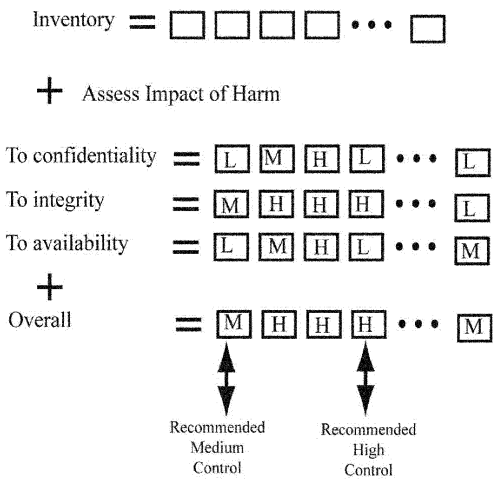

Figure 5. Security standards methodology.

Note that standards documents have evolved for each perspective corresponding to the threads of our security systemigram, and these have all been codified by standards

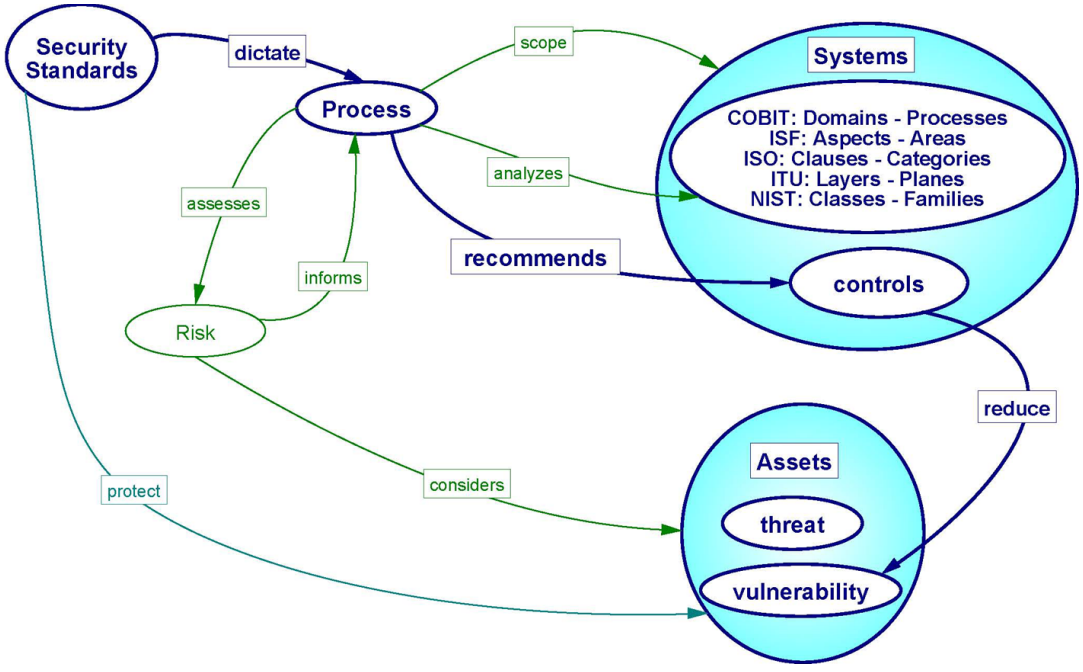

Figure 4. Systemigram of security standards. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

compliance organizations as security metrics. So, given the way security metrics have evolved, they can be categorized generally into things that measure TTOA, SDLC, Security Management, or Security Vulnerabilities.

A TTOA metric is typically quantified with respect to a set of security targets, a set of tests for criteria, and a percentage of targets that have passed the tests. These measure the extent to which technical controls identified by some standard have been implemented. The basis for the metric may be any kind of standard that includes technology recommendations, ranging from the Orange Book to an enterprise department-specific security technology configuration specification.

An SLDC metric is normally quantified by listing activities and monitoring the execution of that activity. Also called a process metric, its goal is to determine whether process is followed. Those developing process metrics find ways to collect data as information flows through the process in the course of its execution, and report on whether all process steps were correctly followed for each case, and in aggregate.

Security metrics based on the presence of vulnerabilities are typically quantified by classifying the vulnerabilities identified in a scan by potential impact. That is, a scenario is envisioned where a successful exploit takes advantage of each vulnerability, and a judgment is made on the extent to which the successful exploit would negatively impact the organization (typically using the ordinal scale: low, medium, high). The number of vulnerabilities in each class is counted and presented in graphical format. A vulnerability metric would normally be presented in conjunction with a vulnerability remediation plan, and hence may be thought of as Security Remediation Metrics.

A Security Management metric is typically quantified as a dashboard of departments or organizations, with some quantification indicating the extent to which both target, process, and remediation metrics are met. Security management metrics typically also include measures of activity such as incidents investigated or access control administrative tasks performed. These do not always necessarily always measure the security environment, but typically also measure resources consumed in maintaining security. There are no standards for what types of metrics should be included in these dashboards.

Security metrics may therefore generally be classified as target, process, remediation, or management activity metrics. Figures 6–9 illustrate the approaches.

The type of metric that most closely aligns with systems engineering activities is a process metric. Figure 10 is an example of a systems engineering process that is designed to produce system security requirements. Where such processes are formalized, they are typically performed as a specialty engineering task in systems security engineering led by a systems security engineer [INCOSE, 2011]. The output of the process is a set of requirements that is then provided back to the lead systems engineer. Where metrics are collected on the process itself, they are maintained as a collection of tollgates met and milestones achieved. The steps in a security requirements process are often very similar to the process of gathering security requirements to be incorporated into a software development project [McGraw, 2006]. As the author of Figure 10 put it, the process model provides "a structure within which a security framework is defined for a system; a placeholder for metrics as the assessment function is refined" [Oren, 2012, p. 25]. This refers to the security engineering function, not to the security metrics that will eventually be derived from the requirements that the security function produces.

The discussion above concerns security standards that have evolved into associated security metrics. These do not include the full set of candidates in the literature on security metrics. The full suite includes security metrics for mathematical modeling of security management processes [Berres et al., 2009], weighting network forensics evidence to increase probabilities of conviction [Amran, Phan, and Parish, 2009], quantifying threat surface using hidden Markov models [Wang et al., 2010], using game theory to determine security investment strategies [Carin, Cybenko, and Hughes, 2008], and complex mathematical models for assessing software security [Alshammari, Fidge, and Corney, 2009]. Most of these are the subject of one or two papers by the same group of authors, and rely on data that are not completely described (and also usually includes subjective measures of probability). A practitioner reading these types of hypothetical usefulness cases recognizes that there are more straightforward ways to assess the security of the target environment. So, although they are included in the security metrics literature, they will not be included herein.

There are at least 900 measures/metrics that exist in the literature for measuring security. This can be stated with certainty because more than 900 are listed in Herrmann's 2007 book, *A Complete Guide to Security and Privacy Metrics* [Herrmann, 2007]. Herrmann only included metrics that she considered appropriate for use in decision-making by practicing auditors, engineers, and managers. Herrmann's intent was to create a useful menu for security practitioners, and so she purposely excluded metrics that were abstruse or that relied heavily on an intuitive understanding of complex mathematical models. This idea is echoed in security literature: that metrics which form the basis for decisions should be well understood. As Jaquith put it, "transparency facilitates adoption by management" [Jacquith, 2007: 20]. As Pironti put it, "keep it simple" [Pironti, 2007].

Validation of any measurement is strengthened by a number of factors that contribute to its credibility. Transparency

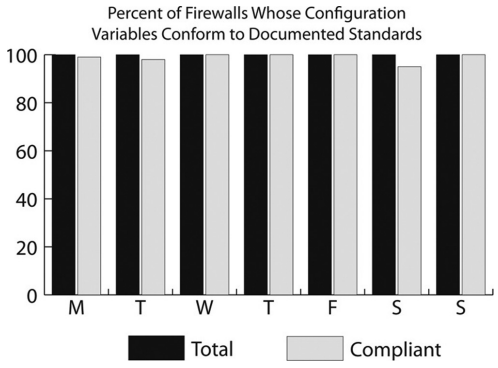

Figure 6. Example security target metrics.

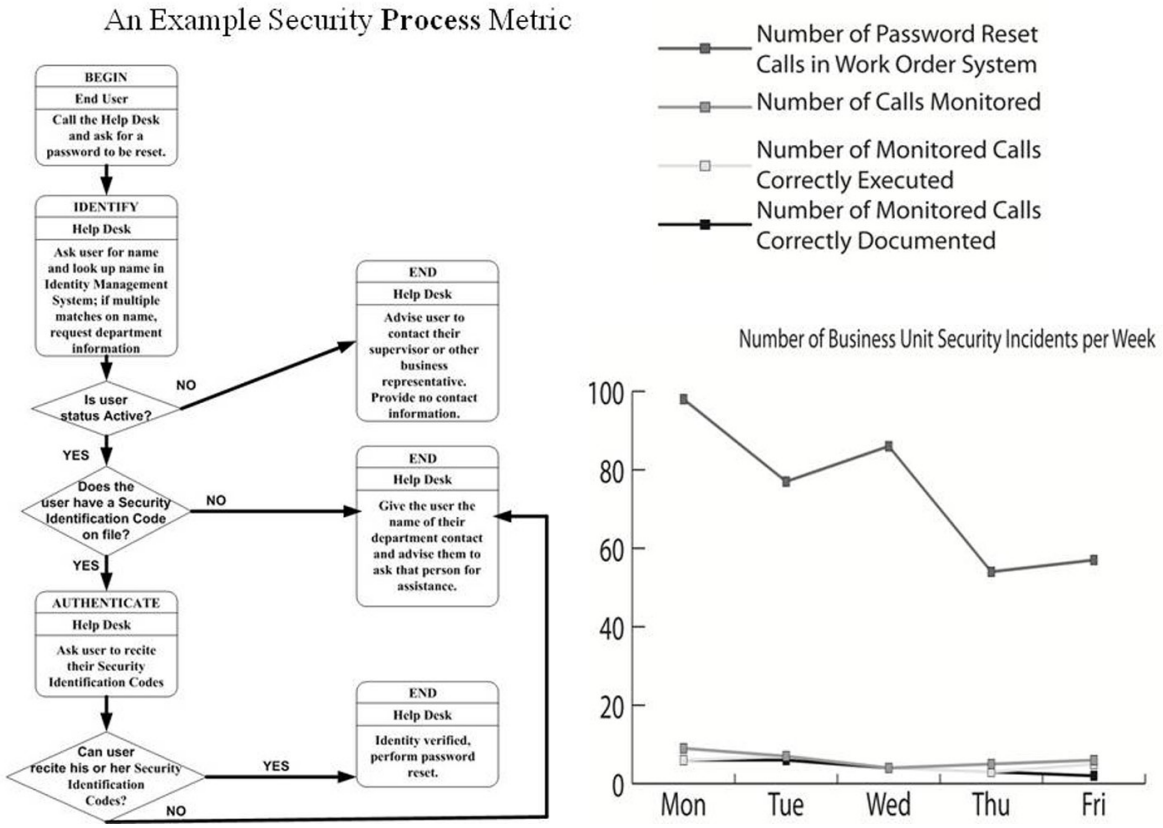

Figure 7. Example security process metrics.

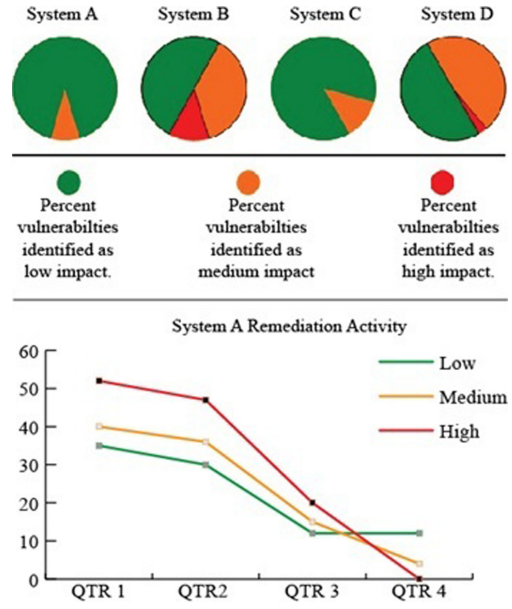

Figure 8. Example security remediation metrics. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

is one of them. Other attributes of measurement that have been proposed to contribute to metrics validity are:

Accurate: Data reflect the content of measurement as it was envisioned.
Correct: Data are collected according to specifications.
Consistent: measure is independent of measurer.
Informative: Data provide information without additional context.
Numeric: Data can be precisely quantified.
Ordinal: Data samples can be ordered.
Replicable: Measurement repeated in same manner in same environment will yield the same result.

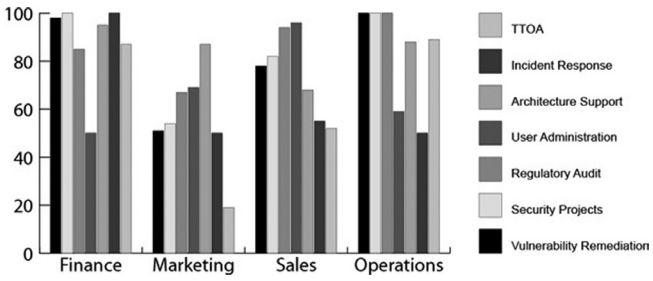

Figure 9. Example security management metrics.

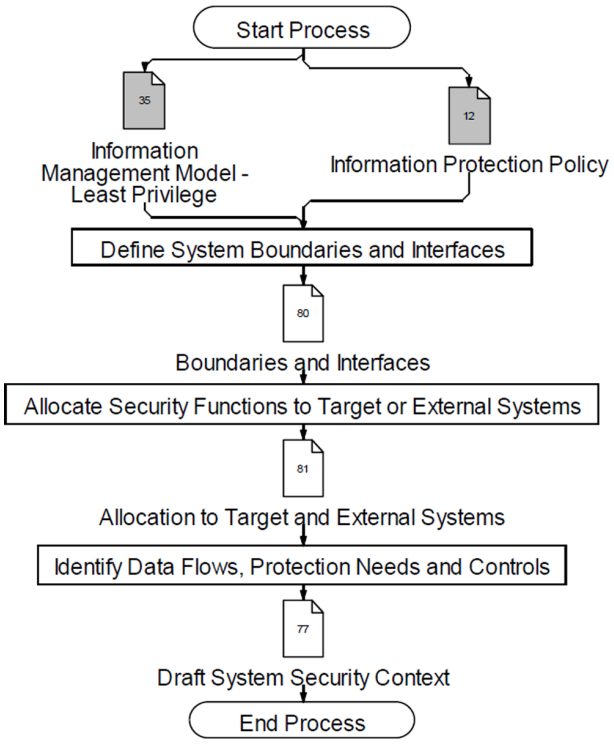

Figure 10. Example systems engineering security process [Oren, 2012, p. 28].

Time-based: There is a fixed reference point of data collection.
Unit-based: Data may be expressed in terms of a unit.

Where a proposed security metric has face validity, and meets many of these criteria, it will be deemed more reliable an indicator of security than metrics that do not. For any given exercise in security metrics, a systems engineer may adopt rules for strength of validity. For example: "*In order to considered a valid metric, the underlying measure must be numeric, unit-based, correct, time-based, and replicable.*" These rules of thumb may be adopted for both verification and validation metrics.

For a systems engineer to find security metrics useful, the decisions for which they are relevant must be framed by a context in which security will serve the mission or purpose of the system of interest [Bayuk et al. 2010]. Hence, the dimensions of security measurement (target, process, activity, or remediation) must be focused on the features of the system of interest that constitute its security profile. This is not the same as a Security Target, in the language of Common Criteria, because the target functionality of the system of interest is not security functionality, per se, but secure system operations, which may require security features to be customized and integrated with system operational capabilities.

4. SECURITY VERIFICATION AND VALIDATION

From the above discussion, it should be clear that it is possible to use existing TTOA security target metrics in combination with security process metrics to *verify* that build-to components specifications meet and are operated in compliance with security requirements. However, for security requirements themselves to be *validated*, validity measures must be determined not only for build-to specification but for security goal achievement. As illustrated in Figure 11, security requirements exist both above and below the horizontal line in the Vee model that separates systems engineering tasks from components engineering. We have so far discussed, and the history of security metrics has so far been concentrated on, verification of security requirements in build-to specifications

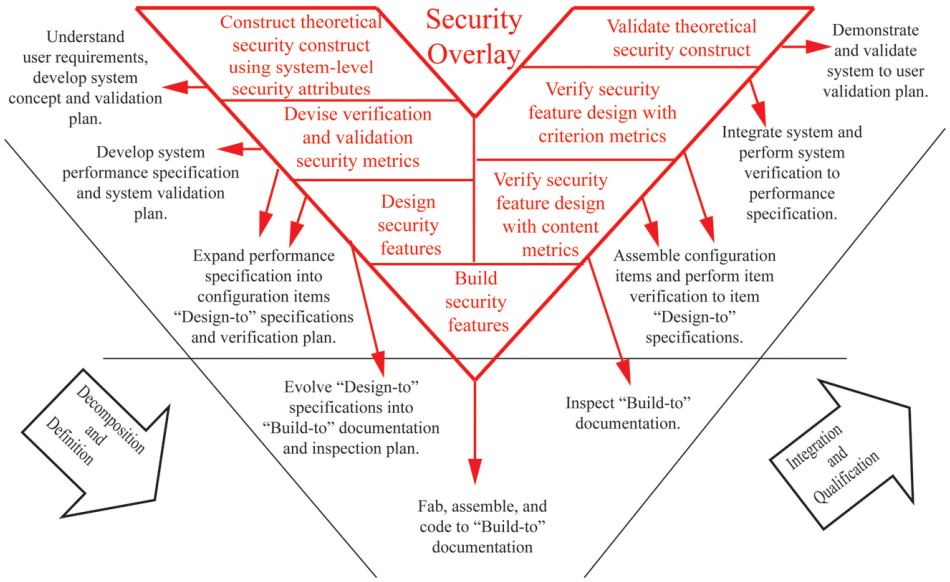

Figure 11. Security metrics in the Vee Model, based on INCOSE [2011]. Adapted from Bayuk and Horowitz [2011]. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

and process operation. Target and process metrics are used to ensure that systems are configured and operated as designed. If these metrics reveal an inconsistency between design and implementation, activity metrics may be devised to catalogue incidents of noncompliance, and remediation metrics may be designed to chart progress back in line. These are all verification metrics, and stand in contrast to metrics that would show whether or not the system security design actually achieves system security goals. This is the province of validation. Though the term "valid" has been applied to metrics as an adjective, none of the metrics discussed so far address validation that security goals are achieved.

Techniques for devising security validation tests do not differ from validation tests for other important system features or characteristics. For example, a Goal-Question-Metric (GQM) approach is a common way to identify metrics that correspond to system goals and it may productively be applied to security [Basili, Caldiera, and Rombach, 1994]. If system stakeholders want to know whether their goals for system security are met, what questions would they ask to so determine? As they do not have the insight into technical or process design, their questions would be higher level, like, "Am I the only one who can see my data?" These validation questions can only be answered once the system is in operation, and creative approaches to measurement are often required to answer them. Nevertheless, metrics devised for validation measures should be subject to the same standards for validity of measurement as metrics devised for verification measures. That is, in order to be considered a valid metric, the underlying measure must be numeric, unit-based, correct, time-based, and replicable.

Another useful tool for devising security validation tests is Quality Function Deployment (QFD) [QFDI, 2011]. QFD stresses quality targets, competitive analyses, and selling features achieved through the use of cross-functional teams from marketing, design engineering, and manufacturing to integrate customers' demands with the technical aspects of the solution. Using QFD for security requires that customer security requirements be stated positively, not as constraints, and that security products and services have distinguishable, thus measureable, security attributes that may be used as validation metrics.

The primary concept to internalize when establishing security validation metrics is that how you measure security depends on how you define security. This definition will vary by system and stakeholder community. The question is the same faced by every systems engineer in the course of daily work: *Did we build the right system?*

5. CLOUD SYSTEM CASE STUDY

In no scenario is the difference between verification and validation more obvious than in cloud computing. This is because verification requires a hands-on evaluation technique of actual implementation versus requirements, while validation corresponds to the customer perspective. Cloud services are by nature shared. Though they may have been originally designed with the help of a customer with special requirements, cloud services generally build on that initial platform to offer similar services to other customers who were not involved in the design of the control environment, and also not allowed to perform verification tests. Where customers clearly specify goals and objectives for security to which a cloud vendor is contractually bound, these are typically stated with reference to a published security standard or a list of control mechanisms attached to the contract. It is the responsibility of the cloud vendor to verify that all internal processes that handle information have been evaluated for compliance with customer security requirements. However, a cloud vendor may not be obligated to release that evaluation to the customer. Even in situations where it has been agreed that external auditors will perform audits against customer requirements, as in the case where customers must demonstrate due diligence in meeting regulatory assurance requirements, the audit does not typically approach true security feature verification. For example, verification requires that each TTOA undergo system readiness tests while an auditor is generally satisfied with evidence of process designed to accomplish TTOA security readiness and metrics that demonstrate that the process is followed. Even where verification testing is done by auditors, they tend to use statistically valid samples of configurations rather than aim for full verification. In most cases, the only verification that cloud components are verifiably secure is performed by cloud vendors themselves.

Validation, on the other hand, is a test of fitness for purpose. Security validation requires that system operation achieves enterprise security goals for prevention of harm, as well as detection and appropriate response and recovery activities. Security cannot be validated in test environment, and so it can only be done once the service is in the process of being used by a cloud customer. Of course, as in any systems engineering endeavor, validation tests may necessarily not completely cover all verification criteria. However, customers that have the foresight to include security features required for their own validation testing in service contracts will be at an advantage when attempting to perform validation tests. These may be, for example, software change planning announcements and automated software change detection alerts, the correlation of which would allow them to judge the extent to which the cloud vendor controls software integrity.

Assume a cloud customer is considering storing a customer database on a site that advertises client relationship management software as a service. Information stored in the cloud may be client contact information, order history, credit line, and various reminders such as children's names, birthdays, and wine preferences. The cloud customer, of course, does not want to lose control over the confidentiality, integrity, or availability of the information, but also does not have the in-house expertise to build all the features that the cloud software already has.

A precisely stated goal for cloud security is sometimes elusive because many stakeholders have no articulated goal for security other than "the system should be secure." They provide this directive to cloud vendors via legal agreements to secure systems to "industry standards" with the result that cloud vendors either create, or are provided with, checklists of accumulated security standards recommendations rephrased to refer to cloud security architecture components

[Cloud Security Alliance, 2009]. Though these checklists may form the basis for verification requirements, as they are not composed or selected based on the requirements of the system of interest or its stakeholders, so using them as verification requirements does not follow best practice in systems engineering methodology, and their use should never be confused with an attempt at validation.

To apply GQM to cloud security, the stakeholder must articulate security requirements at a level of detail wherein they themselves can recognize when security requirements are met. To demonstrate how to use the GQM approach to identify security validation metrics, we formulate goals for confidentiality, integrity, and availability, respectively. For each, we proposed a candidate metrics based on current industry approaches to answering the questions.

Goal: Maintain client data confidentiality.
Question: Can others see client data?
Candidate Metric: The time it takes a skilled penetration test team to access client data. Longer time periods indicate that security is better. Management estimation of potential adversarial efforts with which to compare the time period of the test result (and associated risk tolerance levels) will determine the amount of time that indicates poor security. The estimation may be supplemented with data from known security incidents within the industry as recorded by penetration test team practitioners [Verizon Business, 2010].
Goal: Maintain client data integrity.
Question: Is client data accurate?
Candidate Metric: Survey customer cloud users to determine the extent to which they find the client data reliable via an ordinal measure from "not reliable" to "highly reliable."
Goal: Maintain client data availability.
Question: Are client data always accessible?
Candidate Metric: Automate continuous data access routines and measure amount of time that client data are inaccessible. Any inaccessibility period that exceeds service level agreements indicates a lack of security. Shorter inaccessibility period indicate security is better.

Some of these candidate metrics may be better security indicators that others, and some are more difficult than others to measure; but as a whole, they seem to scratch the surface of the answer to the questions rather than to provide hard validation criteria. Though they may have face-validity in answering the question, their use in a scientific validation of a construct theory of security is not evident. This introduces an opportunity for QFD to supplement GQM in order to supply some needed constructs. To apply QFD to security, it must have a quality target based on competitive analyses, and a description of security features that are integrated with the technical aspects of the solution and also meet customer demand for security. The GQM question may provide that target, as long as the metric is positively stated. Creating a QFD metric requires the question to be mapped to the customer goal evaluation criteria and also to the system of interest. For example:

Goal: Data should be safe from theft.
Question: Can data be stolen from the cloud?
Metric: Time and effort estimates to accomplish each branch in an attack trees scenario wherein data theft is the goal, as in physical adversary path analysis [Hester, 2007]. Higher time and effort indicates better security. As in the confidentiality metric using GQM above, management estimation of potential adversarial efforts or known incident analysis may be used for comparison with the time and effort data gleaned from the attack tree analysis.

Though not validated in the scientific sense, experienced security analysts do use such metrics to gage the security posture of a system of interest, and these are candidates for QFD analysis because they measure technical system features to validate that stakeholder requirements are met in a fielded system [Garcia, 2008]. Arrott described one such possible measure as a Work Factor Ratio ("WFR") between "time to protect" and "time to attack" [A. Arrott, personal communication, 2011]. Although he did not formally publish this theory, it may be described as follows:

- The *time to protect* (TTP) is the average interval between when a target is first aware of the existence of a new threat and when it successfully deflects it. This measure depends mainly on the speed and effectiveness of a target's response capability.
- The *time to attack* (TTA) is measured as the median lifetime of malicious activity emanating from a specific source. This is useful to measure in situations where attackers must constantly create and abandon original points to evade detection. The shorter this median lifetime, the heavier is the burden on the attacker to continuously change its location to evade detection.
- To the extent the ratio TTP/TTA is minimized, the defenders are successfully thwarting attacks. To the extent it increases, the attackers are more successful. The goal of absolute security would be measured with a TTP/TTA metric that is better as the ratio approached 0.

To measure whether the goal of preventing data theft is met in a cloud system, the TTP may be derived from a combination of the vulnerable components that need to be compromised for the data to be stolen and the existence (or not) of security controls that compensate for the vulnerable components. This requires modeling of attack paths and identification of defenses in place to delay or stop each path. For example, Figure 12 is an attack tree corresponding to the GQM example of: "Data should be safe from theft." Attack trees are formulated by placing an adversary goal at the top, and decomposing that goal into subgoals joined by "and" or "or" branches that signify whether subgoals must all be achieved for the higher level goal to be met, or whether there are alternative subgoals that would independently suffice to achieve the higher level goal. The leaves of the tree indicate the actual adversary activity that, when combined according to the logical constructs which dictate whether they must be used in combination (the and gates in the diagrams), form one

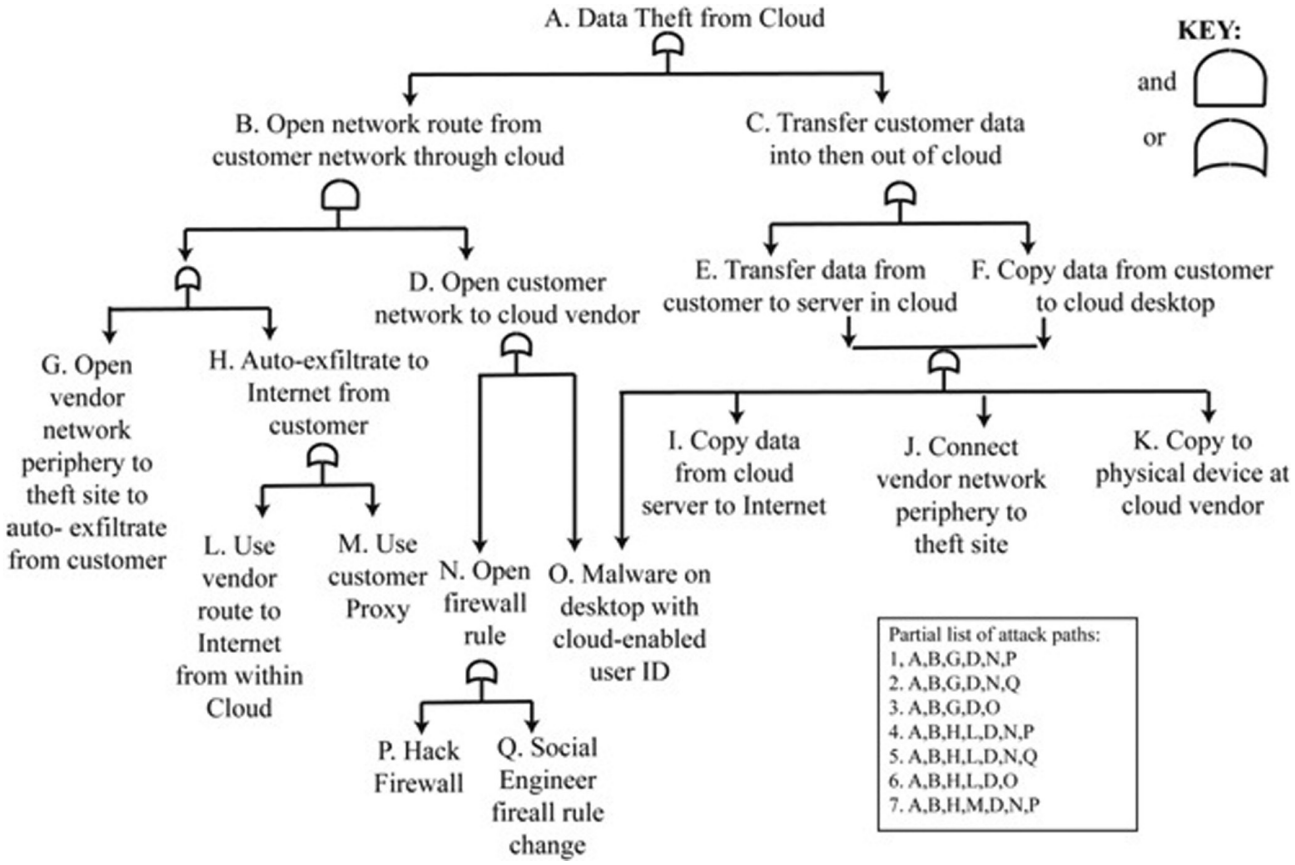

Figure 12. Attack tree for a cloud system. Adapted from Bayuk [2011].

"attack path." Attack path 1 in Figure 12 indicates that activities on leaves G and P set the stage for the attack path to be utilized. The vendor network periphery must be opened to an attacker site and the customer firewall must be hacked in order for the exploit to occur via attack path 1.

To assign a time to attack value to the path, the length of time that an attack is available to the attacker would be calculated for each leaf activity. Assume that there is no control against the vendor periphery connecting to the attacker site (as would be the case if outbound Internet access was allowed from within the vendor network, which is common). Then the time assigned to the leaf is infinity (INF). The time to attack is then bounded only by the time an attacker determines that there is opportunity to hack a customer firewall to gain administrative access. The opportunity determination will depend on what vulnerabilities are known to exist in firewalls in general at the time, and the prevalence of hacker tools that efficiently execute that firewall attack. Such measures can be estimated using publicly available historical data concerning similar attacks [Verizon Business, 2010]. As cloud systems are subject to the same attacks as any network on the public Internet, comparative attack data may also be captured using the time that URLs used to distribute malicious software and/or collect data from infected hosts are active before they are detected by security services companies that investigate and filter such URLs ("web reputation services").

To assign a time to protect, available corrective controls must be reviewed. For a firewall whose rules and configurations are checked daily via automated mechanisms and response is immediate configuration correction, this attack may be available for 1 day. For environments where firewall rules are checked once annually by external auditors, this time period is 1 year. For firewalls with known exploitable vulnerabilities due to software flaws, this time period is the average time between firewall software vulnerability announcements and the time customers install patches. Consideration of more detailed alternatives may prompt a systems engineer to add levels to the attack tree in an iterative requirements process.

Where there is a single point of security failure on any one attack path, then the time to defend is the time to correct that situation. Assume that the control failure that allows a firewall hack is a software vulnerability in firewall access control. In this case, the time to fix that component includes not only the time the firewall vendor takes to offer a patch for the component, but also the time the cloud vendor takes to apply the patch. Although the vendor may have signed a service level agreement to apply patches "as soon as possible," the vendor's historical time to repair can only be measured by monitoring the system in operation. If another mechanism may compensate for that failure, but is a detection rather than a prevention mechanism, then the time to protect is the interval between the detection and the response that thwarts the threat. Hence, the time to protect a given path will depend on the

controls preventing exploit on that path, and is measured as the minimum time required to establish compensating or corrective controls.

If each attack path can be assigned a WFR based on the minimum TTP/TTA for attack recovery, then for any given cloud system C, the maximum time to protect against all identified threats to the cloud is formulated as follows: Assume P_1 through P_n are the paths on a rigorously devised attack tree for cloud system C, and $P_{1\mathrm{WFR}}$ through $P_{n\mathrm{WFR}}$ are the corresponding WFR ratios that an attack of depth d would take on each path. C_{WFR} is the longest of those minimum values, calculated as

$$C_{\mathrm{WFR}} = max(P_{1t} \ldots P_{nt}).$$

The median of a cloud attack is measured using a moving historical sample of active attacks. Trends will, of course, change continuously, so any cloud security validation metric based on it will have to be continuously monitored to ensure that any validation that stakeholder expectations for security are met evolve in conjunction with changes in the threat environment. But, in general, assuming equivalently thorough attack trees, the higher the C_{WFR}, the stronger the security metric. Given two cloud environments with roughly equivalent services, a cloud with a lower WFR will be more secure than one in which it is higher.

In this metric calculation, the time to thwart the attack is taken as a constant. In practice, however, security best practices dictate that there be multiple controls layered on each attack path to ensure that there is no single point of failure that would allow attacks to be successful. This is a "defense in depth" approach. In such cases, the WFR would not be a single number, but an upper and lower bound. Different combinations of controls may be compared to achieve the longest time period as the upper bound, while minimizing the range between the upper and lower bound.

Where redundant protective controls have been designed into a path, the path is said to have defense in depth, and, unless the same vulnerability applies to both controls, the TTP for redundant controls is zero. This suggests that another metric may be the percentage of attack paths for which the TTP is zero due to the presence of compensating controls of diverse technology.

6. CONCLUSION

This paper has described security metrics as typically applied in current systems engineering activities. It has compared such use to requirements for security verification and validation. These requirements were illustrated via a case study.

We conclude that state-of-the-art security metrics are adequate for security verification because verification only requires that requirements for build-to components are met. However, as currently applied, these are blanket requirements for every system component and often do not get the scrutiny that should correspond to component criticality in a given system of interest. Even if they were artfully applied, they would still not meet requirements for security validation.

Security validation metrics are less mature and by nature will emerge as unique for each system of interest. However, systems of similar architecture patterns should have enough similar stakeholder expectations to make even validation measures reusable to some extent. The primary points to keep in mind when establishing security validation metrics are:

- How you measure security depends on how you define it.
- The definition must include the context of operations.
- The context of operations includes the threat environment.

The security overlay for the Vee model equips a systems engineer to construct a theory of security for a given system of interest that can be tested for validity. This theoretical construct focuses on system security validation and so is comprehensible to executive decision-makers faced with trade-space decisions that affect system security. That is, where security requirements are integrated into the systems engineering process, resulting measures of system security validity are both construct and face valid.

The goal of this work is to dispel the belief that compliance with security standards provides assurance that system security goals are met. It does not make today's certification and accreditation programs obsolete, but it should raise awareness within the engineering profession of the relative contribution of standards compliance in the context of system security goals and objectives. Future research in this area may be expected to ultimately result in a pattern catalogue of systems security models suitable for a given type of system of interest [Jones and Horowitz, 2012]. This catalogue would not compete with security standards, but provide an alternative view on system security requirements that would enhance stakeholder appreciation for systemic security features.

REFERENCES

J. Allen, The CERT guide to system and network security practices, Addison-Wesley, Reading, MA, 2001.

B. Alshammari, C. Fidge, and D. Corney, Security metrics for object-oriented class designs, Ninth Int Conf Qual Software, IEEE, 2009, pp. 1550–6002.

A.R. Amran, R. Phan, and D.J. Parish, Metrics for network forensics conviction evidence, Int Conf Internet Tech Secured Trans, 2009, pp. 1–8.

V.R. Basili, G.H. Caldiera and D. Rombach, The goal question metric approach, University of Maryland, College Park, 1994.

J. Bayuk, Stepping through the IS audit, a guide for information systems managers, 2nd edition, Information Systems Audit and Control Association, Rolling Meadows, IL, 2005.

J.L. Bayuk, Cloud security metrics, IEEE Int Conf Syst Syst Eng, 2011, pp. 341–345.

J.L. Bayuk and B.M. Horowitz, An architectural systems engineering methodology for addressing cyber security, Syst Eng 14(3) (2011), 294–304.

J. Bayuk, D. Barnabe, J. Goodnight, D. Hamilton, B. Horowitz, C. Neuman, and S. Tarchalski, Systems security engineering, a research roadmap, final technical report, Systems Engineering Research Center, Washington, DC, 2010, www.sercuarc.org.

Y. Beres, M.C. Mont, J. Griffin, and S. Shiu, Using security metrics coupled with predictive modeling and simulation to assess secu-

rity processes, Third Int Symp Empirical Software Eng Measure, IEEE, 2009, pp. 564–573.

J. Boardman and B. Sauser, Systems thinking: Coping with 21st century problems, Taylor & Francis, New York, 2008.

L. Carin, G. Cybenko, and J. Hughes, Quantitative evaluation of risk for investment efficient strategies in cybersecurity: The queries methodology, Metricon 3.0, San Jose, CA, 2008.

Cloud Security Alliance, Security guidance for critical areas of focus in cloud computing v2.1, http://www.Cloudsecurityalliance.Org, Las Vegas, NV, 2009.

F. Cohen, How do we measure security? INCOSE Insight 14(4) (2011), 30–32.

DTI/CCSC [UK Department of Trade and Industry's (DTI) Commercial Computer Security Centre (CCSC)], A code of practice for information security management, British Standard BS7799:1995, London, 1995.

M.L. Garcia, The design and analysis of physical protection systems, Butterworth-Heinemann, Amsterdam, 2008.

D. Herrmann, The complete guide to security and privacy metrics, Auerbach, Boca Raton, FL, 2007.

P.T. Hester, Facility protection optimization under uncertainty, Ph.D. Dissertation, Vanderbilt University, Nashville, TN, 2007.

INCOSE, INCOSE systems engineering handbook, version 3.2.1, Seattle, WA, 2011.

ISO/IEC, Information technology—systems security engineering—capability maturity model, SSE-CMM, ISO/IEC 28127, International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC), Geneva, Switzerland, 2002.

ISO/IEC, Information technology—security techniques—information security management systems—requirements, ISO/IEC 27001, International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC), Geneva, Switzerland, 2005a.

ISO/IEC, Information technology—security techniques—code of practice for information security management, ISO/IEC 27002, International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC), Geneva, Switzerland, 2005b.

ISO/IEC, Information technology—security techniques—information security risk management, ISO/IEC 27005, International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC), Geneva, Switzerland, 2008.

ISO/IEC, Information technology—security techniques—evaluation criteria for it security—part 1: Introduction and general model, ISO/IEC 15408, International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC), Geneva, Switzerland, 2009a.

ISO/IEC, Information technology—security techniques—information security management—measurement, ISO/IEC 27004, International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC), Geneva, Switzerland, 2009b.

A. Jaquith, Security metrics, Pearson Education, Upper Saddle River, NJ, 2007.

R.A. Jones and B.M. Horowitz, A system-aware cyber security architecture, Syst Eng (2012), to appear.

W. Larson, D. Kirkpatrick, J.J. Sellers, D. Thomas, and D. Verma, Applied space systems engineering, McGraw-Hill, New York, 2009.

G. McGraw, Software security, Addison-Wesley, Reading, MA, 2006.

P. Mell, K. Scarfone, and S. Romanosky, A complete guide to the common vulnerability scoring system version 2.0, Forum of Incident Response and Security Teams (FIRST) (sponsored by NIST), Washington, DC, 2007.

M.S. Merkow and J. Breithaupt, Computer security assurance, using the common criteria, Thomson Delmar Learning, Clifton Park, NY, 2005.

MITRE, Common vulnerabilities and exposures, dictionary of common names for publicly known information security vulnerabilities, McLean, VA, http://cve.mitre.org, 2011a.

MITRE, National vulnerability database, US National Institute of Standards and Technology, McLean, VA, nvd.nist.gov, 2011b.

B. Nevo, Face validlity revisited, J Educ Measure 22(4) (1985), 287–293.

NIST, An introduction to computer security: The NIST handbook, National Institute of Standards and Technology, 1995.

J.C. Oren, Establishing a framework for an information systems security engineering process. Ph.D. Thesis in Systems Engineering, Stevens Institute of Technology, Hoboken, NJ, 2012.

E.J. Oud, The value to it of using international standards, ISACA Journal 3 (2005).

J.P. Pironti, Developing metrics for effective information security governance, ISACA Journal 2 (2007).

QFDI, www.qfdi.org, Quality Function Deployment Institute, Ann Arbor, MI, 2011.

R. Ross, S. Katzke, A. Johnson, M. Swanson, G. Stoneburner, and G. Rogers, Recommended security controls for federal information systems, SP 800-53 rev 2, National Institute of Standards and Technology, Washington, DC, 2007.

S.J. Ross, Standard deviation, Inform Syst Control J 6 (2006).

US Congress, Federal Information Security Management Act, US 107-347, Vol. 3541, Washington, DC, 2002.

Verizon Business, Verizon incident sharing metrics framework, http://securityblog.Verizonbusiness.Com/2010/02/19/veris-framework, Washington, DC, 2010.

H. Wang, S. Roy, A. Das, and S. Paul, A framework for security quantification of networked machines, 2nd Int Conf COMmunication Syst NETworks, (COMSNETS), 2010, pp. 5–9.

Jennifer Bayuk is the Industry Professor and Director of Cyber Security Programs for the School of Systems and Enterprises at Stevens Institute of Technology, Hoboken, NJ. She has been a Wall Street CISO, a Big 4 Information Risk Management Consultant and Auditor, a Security Architect, a Manager of Information Systems Internal Audit, and a Bell Labs Security Software Engineer. Jennifer frequently publishes on IT governance, InfoSec, and audit topics, including three textbooks and two edited compilations on InfoSec Governance Issues. Jennifer has published numerous articles and lectured for organizations that include ISACA, NIST, and CSI. She has Certifications CISSP, CISA, CISM, and CGEIT. She has a Masters Degree in Philosophy from The Ohio State University, Columbus and a Masters Degree in Computer Science and a PhD in Systems Security Engineering from Stevens Institute of Technology.

Ali Mostashari is the Director for the Infrastructure Systems Program and an Associate Professor at the School of Systems and Enterprises at Stevens Institute of Technology, Hoboken, NJ. He also leads the Complex Adaptive Sociotechnological Systems (COMPASS) Research Center at Stevens Institute of Technology. He is an author of three books and author or coauthor of more than 40 academic papers on various aspects of sociotechnical systems. Prior to joining the School of Systems and Enterprises, Dr. Mostashari served as a strategic advisor to the Assistant Secretary General for Africa at the United Nations Development Programme (UNDP). Ali holds a PhD in Engineering Systems from the Massachusetts Institute of Technology (MIT), Cambridge, MA, a Master's in Civil Engineering from MIT, a Master's in Technology and Public Policy from MIT, a Master's in Chemical Engineering from the University of Nebraska, Linciln, and a Bachelor's in Chemical Engineering from Sharif University of Technology, Tehran, Iran.