Cloud Security Metrics

Jennifer Bayuk *(Author)*
School of Systems and Enterprises
Stevens Institute of Technology
Hoboken, NJ
jennifer.bayuk@stevens.edu

Abstract*— Cloud security had not yet distinguished itself as a field separate from information assurance. Its security metrics are currently synonymous with what a security professional would refer to as a third-party or vendor security audit. Where cloud services are viewed in a systems-of-systems context, any comprehensive security validation approach should rely on the ability of a cloud service to meet customer security requirements; that is, to provide the basis by which customers may assess the efficacy of their own security controls which may be dependent on those in the cloud. This requires a systems-level approach to security validation that is extensible to systems-of-systems environments. This paper describes such an approach)

Keywords- verification; validation; cloud computing; security; computer security; systems engineering; metrics

I. INTRODUCTION

Systems engineers divide metrics into two classes: verification and validation. Verification is confirmation by examination and provision of objective evidence that specified requirements have been fulfilled. Validation is confirmation by examination and provision of objective evidence that the particular requirements for a specific intended use are fulfilled [1]. In layman's terms, these correspond to the questions, "Did we build the system right?" and "Did we build the right system?" respectively.

System security measures used in the system engineering process are typically based on industry standards documents [2]. These began as technical target of evaluation checklists such as the Orange Book [3]. When it became apparent that measures of technical configuration of individual components were inadequate to meet holistic system operational challenges, additional measurements standards were devised to ensure that security was considered during system requirements and design [4, 5]. When it was observed that Systems Development Lifecycle Capability (SDLC) Maturity Models did little to ensure the security of a system in operations, security management standards were developed [6, 7]. In parallel with these approaches, the security profession has been accumulating a repository of known vulnerabilities, which provide the basis for security testing [8]. All of these approaches to security measurement have face-validity, in that the average layman may review them and think something that passes each test is secure [9]. They are accumulated practices from knowledgeable security professionals. However, none of these approaches have been tested in any scientific context, hence their application to engineering verification and validation problems is suspect.

Alternatives for scientific validation of security measures include content, criteria, and content validity [10]. Content tests in security are typically examinations of configuration variable settings that reflect compliance with standards, and these have already been described as inadequate to meet holistic system operational challenges. Criteria tests are those that check current capability against potential future behavioral requirements, such as those tests that determine whether a system can stand up to a test for the presence of Common Vulnerabilities and Exposures (CVE) [11]. Though these type of measures may prove systems are vulnerable, they cannot provide evidence that systems are free of vulnerabilities not on the current CVE list. They are tests that security is bad, but not tests that security is good [12]. Hence, a valid attribution of security therefore seems to lie in construct validity.

Construct validity as applied to security starts with a theory of what it is to be secure. A hypothesis is formed that if the attribute of the construct are found to exist in the system, the system may be judged to be secure. If any one attribute is not demonstrated to exist in a secure system, that failure may bring down the entire construct. Also, if a system is not secure despite the presence of all attributes, that finding will bring down the construct as well. This is the situation with SDLC and management metrics just described. Several systems have been found to exhibit all attributes identified by SDLC and security management standards, and yet are acknowledged not to be secure [13, 14]. Of course, one reason for this could be that a construct validity test may fail for reasons other than the hypothesis being false. It could fail testing because the test was not adequately designed. Nevertheless, compliance with today's standards is not equivalent with an construct-valid attribution of security.

II. SECURITY MODELS

In pursuit of a way to validate security design, engineers have adopted model-based approaches. For example, NIST has also introduced a systems security model wherein security services are functionally defined [15]. It distinguishes between security support services and security prevention, detection and recovery services. Security support services are system features that do not necessarily provide security, but provide hooks to form the basis for security features. For example, identification and naming services give labels to subjects and objects which facilitate systems features for provisioning access and transaction tracing. These features must exist for system functionality whether or not those access and transaction tracing functions are secured against potential misuse. This

allows security utility to be measured in terms of security features. However, the NIST model is not specific to any system architecture, and features required to secure one system may not necessarily be required to secure another. Hence, the usage of the NIST model for security metrics is limited to the definition of key security services terms rather than a construct theory of security for any specific system of interest.

A security model that comes closer to a construct theory of security is the International Telecommunications Union's (ITU) Data Networks and Open Communications Security Architecture for Systems Providing End-to-end Communications (the X.805 standard) [12]. It presents telecommunications architecture as composed of three hierarchical layers:

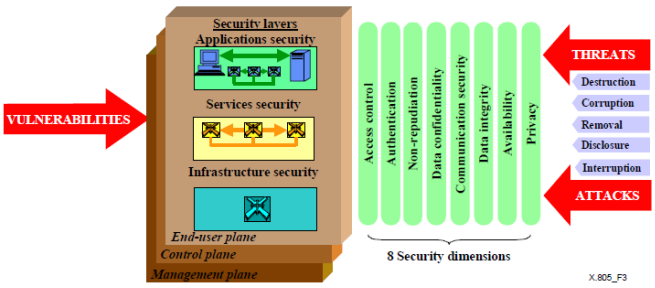
Infrastructure: The set of hardware and software components that provide telecommunications functionality.

Services: The billable customer traffic flow.

Applications: This is the layer that motivates users to pay for the control layer services.

In recognition that each of these functional components must continue to operate in the face of damage, the ITU standard specifies that that security be fully addressed at each layer with a segregation of duties for the management, traffic control, and end user planes that exist at each layer. In recognition that system interfaces cannot be completely controlled, it specifies a design basis threat that should be applied independently to each layer in order to come up with security dimensions, or features, that are required to support security requirements for each layer-plane combination. Figure 1 illustrates the approach.

Figure 1: ITU Security Model [16]



Security modules targeted at each layer and plane may be designed, verified to be designed correctly, and validated to achieve communications security goals. Reflecting the system mission and purpose, it is evident that resiliency of the service layer is paramount, and primary by comparison with the other layers, but they cannot be achieved without a corresponding level of control at the infrastructure layer. Security at the application layer is normally not configured by the telecommunications vendor, but the customer must rely on the service layer security to accomplish their security objectives, so requirements at the applications layer must be fully supported by the security architecture model for the system as a whole. The ITU security model may be considered a construct theory by which security is measured because telecommunications systems that exhibit all of its attributes are considered secure, and it would be hard to find a secure telecommunications systems that is missing attributes required by the model.

The ITU recommendation for full integration of recommended security features with a specific system mission and purpose is very unusual. Both the NIST and the IT security model examples illustrate that, in defining a security model, it is important to distinguish between what capabilities may provide the basis for security features and what capabilities strengthen the system against potential threat. It is also important to ensure that support service implementation strategies do not introduce vulnerabilities that impact system mission. Figure 2 lists some examples of systems of a given class (column 1) and corresponding system capabilities that are core functionality for systems of that class that are often considered security features, but because they are also core system requirements, can be considered security support functions (column 2). The third column of Figure 2 identifies security features that may enable the system to prevent, detect, and respond to threats in addition to those required simply to operate as intended. The key to creating meaningful security metrics is to recognize and devise value measures for this tradespace.

Figure 2: System Capability Overlap with Security

System	Overlap	Security
Sensor-enabled Monitoring	Data Continuity	Confidentiality
Telecommunications	Protocol integrity	Bandwidth utilization forensics
Financial Services	Identity management	Transaction Audit
Military	Confidential communications	Recovery and Reconstitution
Industrial Control	Incident detection and recovery	Protection against insider threat
SmartGrid	Accountability	Theft and Fraud investigation
Airspace	Situational Awareness	Software integrity
Cyberspace	Software integrity	Privacy

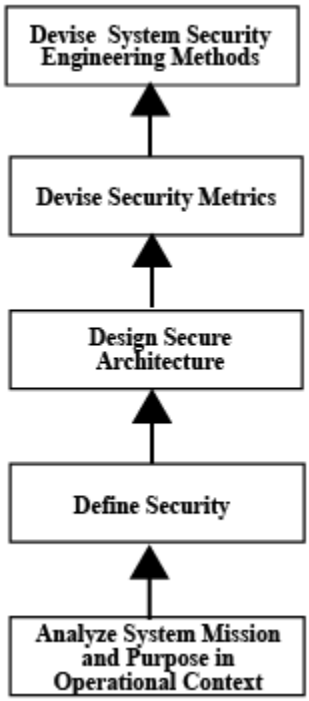
To follow this approach, the core system mission or capability must be defined, the security specific extensions need to be agreed, and a security architecture framework must be developed to allow a systems engineer to merge security requirements into system capabilities. This methodology may extend and enhance systems architecture to produce security requirements at the system level rather than at the security component level [17]. Of course, due to the possibility of threats that are unknown, no system will ever be 100% secure. Nevertheless, this approach should enable a new type of security metrics using this 3-step methodology:

1. Identifying security features that require system-level functions.

2. Evaluate the extent to which security features protect systems from deliberate damage that would cause system failure.

3. Devise verification and validation metrics at the system level that show security requirements are met.

Figure 3: Security Engineering Methodology



The approach is more fully described in the Systems Engineering Research Center Security Roadmap [18]. Figure 3 illustrates the recommendation of the roadmap. Security is defined in terms of the mission and purpose of the system of interest, it comes from the context within which a system operates. This view of security as an enabler allows security architecture to be functions of systems architecture, customized rather than bolted-on. Security architecture metrics may then measure whether security functional requirements are met. Where systems exhibit similar architecture patterns, it is expected that they will have similar security architecture requirements. The existence of common security architecture models should make it possible to develop tools that may be developed to guide future engineering efforts toward more secure solutions.

III. A Cloud Computing Example

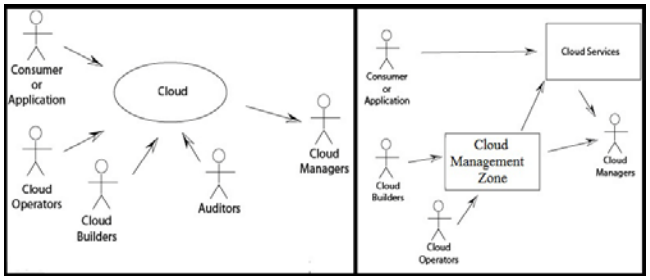
The current state of the practice in cloud security metrics follows the standards-based approaches describes in Section I. Although working committees of cloud security professionals have published cloud-specific security standards, they contain all of the same elements of those described in Section 1, though they may place more emphasis on vendor service level agreements [19]. However, for the purposes of this paper, we do not consider recommendations for contractual assurances or risk tolerance as security measures, but concentrate on cloud customer security goals and the sociotechnical context in which the cloud system of systems operates.

Cloud standards provide a rich taxonomy for cloud service delivery, for example, IaaS (Infrastructure as a Service), PaaS (Platform as a Service), and SaaS (Software as a Service). These service descriptions may form the basis for a demarcation in system level functions that will help identify the basis for security features. IaaS services may commit to provide secure network and storage services, but turn over the keys to the operating system to the customer. PaaS services may include secure web and database configuration, but leave application data flow to the customer. SaaS services may provide secure application services, but leave end user ID provisioning and auditing to the customer.

There are countless variations to these service offerings, and so any example will have to state assumptions. In this example, assume that the cloud is designed to be a private extension of the customer infrastructure, and a primary security requirement is to prevent data theft and loss via the connectivity to the cloud vendor. In addition, it is assumed that authorized data flow between the customer and cloud is adequately protected with firewalls and user authentication, and that managers on the customer side manage the user access lists and also receive reports on both customer end users and their activity.

To apply the above 3-step methodology in the example of PaaS, first identify security features that require system-level functions. In this scenario, several security features exist simply to provide the expected service. These are network periphery, server security, and identity management. However, the entire user community with access to customer data in the cloud must be somehow verified and validated as authorized customer users rather than vendor users or intruders, and so some identity management feature must be defined at the system rather than the component level. Figure 4 illustrates how the requirement for a top level function for identify management changes the input and output in the cloud use case diagram and introduces a segregation of duties in user communities from a scenario in which all customer users and vendors have equal cloud access to one in which cloud customer users have more access to cloud services than cloud customer managers, while cloud customer managers have more access to cloud management functions than customer end users. In addition, cloud vendor staff (builders and operators) end up with no access to the customer cloud services at all.

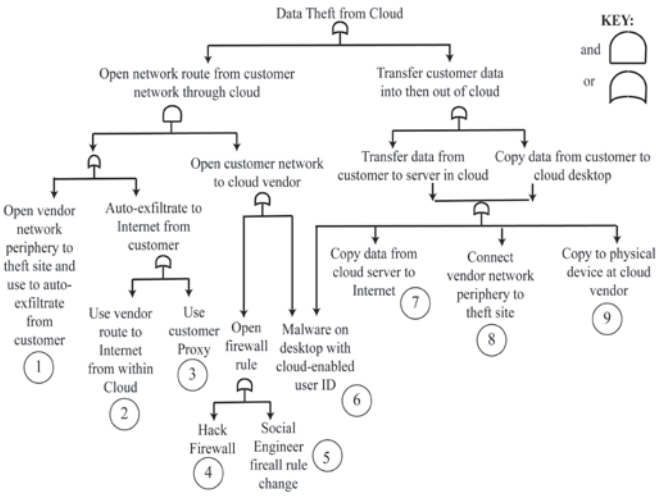
Figure 4: Cloud Use Cases



The next step is to evaluate the extent to which security features protect systems from deliberate damage that would cause system failure. To perform this step requires some analysis of potential adversary goals and some definition of a system failure mode. Figure 5 is an attack tree that presents an attack goal of using the cloud to steal information, and that event may be used as an example of cloud failure to achieve its

mission or purpose. In the figure, the attack goal is decomposed into components via and/or branches that would have to be combined to accomplish the goal identified in a higher layer. It identifies nine (9) possible activities that, if possible, would contribute to a situation in which customer data may be exfiltrated from the customer either through the cloud network or to the cloud network and then on to an external site. These introduce security requirements to incorporate controls against these activities while simultaneously allowing free flow of data to and from authorized cloud users.

Figure 5: Cloud Attack Tree

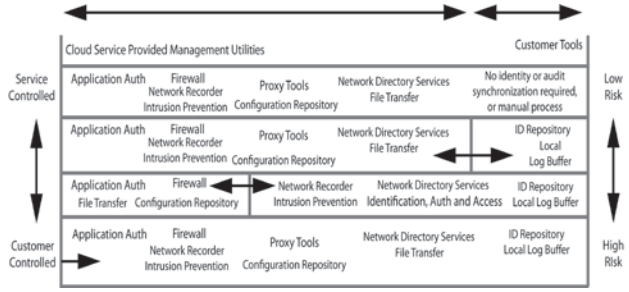


Thorough application of steps 1 and 2 in the systems requirements process will introduce both functional and control level requirements. It is still a design challenge to arrive at an overall system architecture that will combine both the functional breakdown requirements and technical control requirements to prevent, detect and respond to the activities at the attack tree leaf level, but these challenges are now put in the context of overall system architecture challenges that would exist whether or not security was being considered.

Where the structure of cloud service offerings render it impossible for an engineer to design a comprehensive cloud security model that is both manageable from an administrative standpoint and covers all anticipated threats, it may be necessary to supplement cloud services with either customized services or customer-run controls in order to fully develop a security model. Standard security principals such as failsafe defaults and least common mechanisms may be employed to facilitate the security design process and bring it to comprehensive completion [20].

Where it is recognized that cloud services will not consider all required security features part of their core offering, the additional security services relied upon by the customer may nevertheless depend heavily on the correct and effective implementation of the core security features that are provided. There may even be additional controls not required to provide the given service that may nevertheless be considered basic security features by the customer, in which case, the customer may have to pay customization fees. As depicted in Figure 6, customer risk calculations may introduce a sliding scale between the base service offering and higher level security controls that depend on a strong base, and various administrative components may be split between the vendor and the customer where the customer sees administrative controls as necessary to minimize risk. If the base features cannot be relied upon or satisfactorily monitored, the customer trade-space calculations may direct them away from the cloud and to internally-operated computing environments.

Figure 6: Customer-Cloud Provider Security Spectrum



Once the overall design is completed, the features and functions that can be traced to security requirements combine to form a theory of systems security that may be used as a basis for construct validity. In effect, the systems engineer's security validity test is the hypothesis that: if both the functional security features and the specified control activities are in place, the system will be able to withstand an environment of evolving threats. Step 3 of the approach is to devise verification and validation metrics at the system level that show security requirements are met. If all attributes of the construct are found to exist in the system, the system may be validated as secure. Note that this differs from security verification in that any one failed attribution of security either at system deployment or beyond may bring down the entire construct. Moreover, only constructs that stand the test of time should be considered patterns for architectural reuse

IV. Conclusion

This paper describes the current state of the practice in security metrics and applies it to cloud security issues. It provides examples of research in systems security metrics that may be used to extend the state of the art in Cloud security metrics. It describes research in systems security engineering that provides a framework that has the potential to improve the quality of both the state of the practice and the state of the art. It recommends avenue for further exploration of security metrics that may be used to measure a Cloud's ability to withstand attacks.

References

[1] International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC), "Systems and software engineering — System life cycle processes (ISO/IEC 15288)," 2008.

[2] J. Bayuk, "The Utility of Security Standards," presented at the International Carnahan Conference on Security Technology (ICCST), 2010.

[3] The Orange Book, "Trusted Computer System Evaluation Criteria," Department of Defense, 1985 (supercedes first version of 1983).

[4] International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC), "Information technology — Systems Security Engineering — Capability Maturity Model (SSE-CMM, ISO/IEC 28127)," 2002.

[5] Common Criteria Recognition Agreement, "Common Criteria for Information Technology Security Evaluation Version 3.1," 2009.

[6] R. Ross, *et al.*, "Recommended Security Controls for Federal Information Systems, SP 800-53 Rev 2," National Institute of Standards and Technology, 2007.

[7] International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC), "Information technology — Security techniques — Code of practice for information security management (ISO/IEC 27002)," 2005.

[8] *National Vulnerability Database*. Available: http://nvd.nist.gov/

[9] B. Nevo, "Face Validlity Revisited," *Journal of Educational Measurement*, vol. 22, pp. 287-293, 1985.

[10] E. Carmines and R. Zeller, *Reliability and Validity Assessment*. Thousand Oaks, California: SAGE Publications, 1979.

[11] MITRE, "Common Vulnerabilities and Exposures, dictionary of common names for publicly known information security vulnerabilities," see http://cve.mitre.org.

[12] G. McGraw, *Software Security*: Addison-Wesley, 2006.

[13] R. Mogull, "An Open Letter to Robert Carr, CEO of Heartland Payment Systems," in *Securosis Blog* vol. 2009, Securosis, 2009.

[14] G. Shipley. (2010, October 11, 2010) Epic Fail. *Information Week*. 26-38.

[15] G. Stoneburner, "Underlying Technical Models for Information Technology Security," National Institute of Standards and Technology, 2001.

[16] International Telecommunication Union, "Security architecture for systems providing end-to-end communications," vol. ITU-T Recommendation X.805, 2003.

[17] J. L. Bayuk and B. M. Horowitz, "An Architectural Systems Engineering Methodology for Addressing Cyber Security," *Systems Engineering*, vol. 14, 2011.

[18] J. Bayuk, *et al.*, "Systems Security Engineering, A Research Roadmap, Final Technical Report," Systems Engineering Research Center (www.sercuarc.org) SERC-2010-TR-005, 2010.

[19] Cloud Security Alliance, "Security Guidance for Critical Areas of Focus in Cloud Computing V2.1, see http://www.cloudsecurityalliance.org," 2009.

[20] P. G. Neumann, "Principled Assuredly Trustworthy Composable Architectures," SRI International, 2004.