Security as a Theoretical Attribute Construct

Jennifer L. Bayuk

PII: S0167-4048(13)00051-5
DOI: 10.1016/j.cose.2013.03.006
Reference: COSE 686

To appear in: *Computers & Security*

Received Date: 21 July 2012
Revised Date: 11 February 2013
Accepted Date: 13 March 2013

Please cite this article as: Bayuk JL, Security as a Theoretical Attribute Construct, *Computers & Security* (2013), doi: 10.1016/j.cose.2013.03.006.

Security as a Theoretical Attribute Construct

Jennifer L. Bayuk
jennifer@bayuk.com
973-335-3530

Table of Contents

Abstract

This paper provides an overview of the field of security metrics and discusses results of a survey of security experts on the topic. It describes a new framework for developing security metrics that focuses on effectiveness measures while maintaining measures of correctness. It introduces a view of security as a theoretical concept which encapsulates multiple aspects of a system. Viewing security as a theoretical attribute construct promotes the recognition that multiple characteristics and features of a system are required to make it secure. The view also motivates a sharp focus on system aspects which exhibit a measurable security attribute. The framework is illustrated with a case study.

Keywords: security, metric, metrics, framework, verification, validation, survey, information security, cyber security, mobile security

1 Introduction

Today's security metrics are typically based on two assumptions: (i) there is a secure way to configure any system, and (ii) the task of security management is to maintain that configuration. However, today's cyber attacks typically do not exploit holes in configuration; they exploit application or system functionality. The most skilled engineers with the most sincere of intentions using state-of-the-art techniques may create application and platform designs intended to accomplish security objectives, yet it is nevertheless often the case that the resulting system architecture itself is vulnerable. So attacks are successful even though security is configured as designed.

The US National Institute of Standards and Technology (NIST) has documented this distinction as *correctness versus 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 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..” (Jansen 2009)

This distinction is not the same as one security professionals may remember from the Orange Book, which had made a distinction between security *function* and *assurance* (DoD 1985). The Orange Book distinction was criticized for merging functionality and assurance into one scale, creating a one-size-fits-all security model that changes in technology had made obsolete before it was ever even fully instantiated. To avoid the same criticism, Orange Book’s successor, the Common Criteria, focused on security devices and features rather than system-wide security requirements. The Common Criteria progressed the terms *function* and *assurance* through time to refer more and more exclusively to the *security functions* of information technology systems and not to the secure behavior of the system as a whole (CCRA 2012). The term assurance in the Orange Book and Common Criteria literature refers to the assurance that a security function is operating as designed, and both “function” and “assurance” definitions fall into the category of “correctness verification” side of the NIST distinction, which reserves “effectiveness” for the emergent secure behavior at the system level.

From a systems engineering perspective, an equivalent to the NIST distinction between correctness and effectiveness is: “building the system right” versus “building the right system.” (INCOSE 2011) The way to measure building the system right is called *verification*, while the measure of building the right system is called *validation*. Note that validation refers to the system’s ability to meet stakeholder needs in its target environment. Validation strategies are often based on models until the system has been verified to conform to specifications, but cannot be complete until the system is in fact being operated by its target user community (Buede 2009).

This paper presents a methodology for assessing the adequacy of system security using metrics. The methodology considers *security as a theoretical attribute construct* ("STAC") which encapsulates the multiple aspects of a system that render it secure. By definition, the construct must incorporate *both* verification and validation measures in a consistent and unified metric for system security. The benefits of the STAC framework are (i) that system security can be measured and (ii) that the security of similar systems may be compared using the same or similar measures.

2 Background

2.1 Related Work

Like the assurance measures of the Orange Book and the Common Criteria, most of the practical work in security metrics addresses not security, but compliance to some standard. In the case of the Orange Book and the Common Criteria, it is compliance with a design for security. Subsequent topics for security standards include not just design itself, but methodology for designing a secure system and methodology for operating a secure data center. These created additional set of metrics, for example, metrics designed to show compliance with the System Security Engineering Capability Maturing Model (SSE-CMM) and the Federal Computer Security Handbook (which evolved into Recommended Security Controls for Federal Information Systems) (NIST 1995; ISO/IEC 2002; NIST 2007). Variants on these exemplar methodologies have been adopted into enterprise security policies and standards, and corresponding security metrics programs internal to the enterprise have been established to demonstrate compliance with such enterprise standards.

In 1997, this journal published a then-innovative and also comprehensive application of management techniques, including a variety of statistical and other quantitative measurement techniques, to the enterprise information security management practices that were based on such standards (Kovacich 1997). In 2000, NIST began efforts to bring the information security community together to create standards for security metrics as its own field of study (Hancock 2000). The turn of the century also brought in the first Security Information (Event) Management (SIM or SIEM) products, which allowed automated collection of measurement data typically used by enterprise security metrics programs.

Despite this standardization in enterprise security management, there have also been a plethora of cyber security breaches in systems whose metrics had demonstrated such verified design and operation. In the case of certified Payment Card Industry Standard (PCIS) compliance, there are even metrics on incidents that occurred in PCIS-compliant systems (PCI 2008; Mogull 2009; Baker, Hutton et al. 2011). This situation is typically blamed on system security vulnerabilities in the form of software bugs and design flaws, and this situation led to a search for security metrics that could be used to certify that a system was free of known vulnerabilities.

This search in turn motivated the creation of the National Vulnerability Database (NVD) community of security metrics researchers (MITRE ongoing). The NVD is a common repository for security researchers to catalogue known software vulnerabilities in a structured format. It facilitates security software vendors' ability to create tests that security practitioners can use to verify that systems are not vulnerable to known security threats. These are called penetration tests ("pentests" for short), as the goal of the tester is typically to break into the target system. The NVD effort begat the Common

Vulnerability Scoring System (CVSS) (Mell, Scarfone et al. 2007), a security metric that quantifies the degree to which a system is vulnerable, given its technical composition and configuration compared to publicly reported vulnerabilities (assuming its technology is common enough to get reported to NVD). This approach, because it counts bad things and not good things, was mocked by McGraw as a "badness-ometer," a scale on which every measure is bad, so it is impossible to use it to tell if security it good (McGraw 2006). The NVD eventual answer to this criticism was the Common Configuration Enumeration (CCE) component of the Security Content Automation Protocol (SCAP) (NIST 2010), a method to measure whether a system is patched and configured to withstand the exploits enumerated in the NVD. Such configuration management methodology has long been a core component of the management metrics approach (Bayuk 2001).

These industry-wide activities begat security industry metrics analysts, and culminated in the seminal text *Security Metrics*, in which one such analyst thoroughly mapped the ISACA Control Objectives for Information Technology (ISACA 2007) to a recognizably relevant and available set of measures for verifying that security functions are operating as designed and also that systems are free of known vulnerabilities (Jaquith 2007). This blending of vulnerability and countermeasure into a single management view was intended to call attention to the role of metrics in security-related decision support, or security risk management.

The way that a system had traditionally been decided to exhibit a property called "security" is that its configuration corresponded to the security manager's design for security. The badness-ometers challenge that method of measuring security, because management's plan may have overlooked the presence of a known vulnerability. Moreover, zero-day threat metrics show that vulnerability exploits are active for an average of 312 days before the vulnerability is identified (Bilge and Dumitras 2012). These findings have called into question the entire process of maintaining patch levels against known threats as a first line of defense. The role of security metrics in risk management decision support, that is, in deciding how to secure systems, is now clearly independent of its role in deciding whether a system may be said to exhibit a property called "security."

Parallel to the efforts to use management metrics to achieve security goals, security researchers had begun trying to measure security as a system property. Scattered through the security literature are articles that discuss how to measure security in various system components, from network path weighting to attack severity analysis (Amran, Phan et al. 2009; Wang, Roy et al. 2010). There are also attempts to quantify security risk based on stochastic or experimental models of a system's ability to deflect attacks. One proposal measures security by charting out step-by-step attacks on behalf of potential adversaries, assigning probabilities of attack-step success given assumptions about adversary motives and skill sets, and then calculating the system security score based on the attack outcome (LeMay, Ford et al. 2011). Another attempts to assign values to security services based on their contribution to system value, and calculates a security value factor as the overall system security score (Jones and Horowitz 2012) Although these approaches should be commended for their holistic, system-level approach to security measurement, they have generally been considered impractical by practitioners because they are based on unfounded assumptions about system operations, and/or unknowable probabilities (Hubbard 2009).

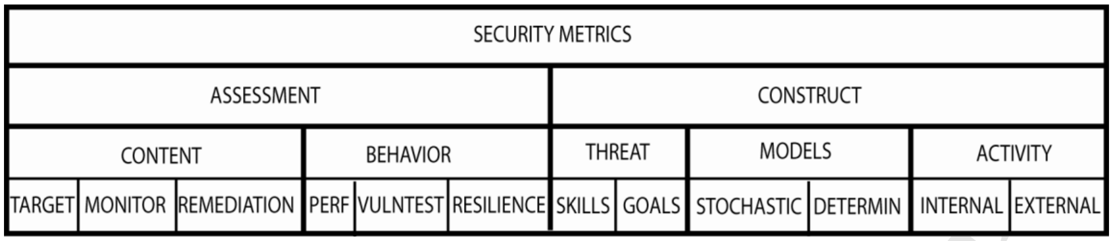
Practitioner conference presenters routinely reduced these approaches to absurdities using comparisons. For example, they tell the story of three blind men measuring an elephant when each has access to a different body part (India). Measures of tusk, tail, and trunk do not combine to measure “elephantness” even in combination. Similarly, security as a measurable system attribute is an elusive concept. If there is anything notable about these purely academic efforts to measure security, it is the assumption that there is a quality to be achieved, called security, and a conviction that this quality is a measurable attribute of systems. On the other hand, the distinction between measuring system security effectiveness and enterprise security program correctness is critical to understanding the field of security metrics. Going forward, the terms “construct decision” and “assessment decision” are used to differentiate the decision on *how to construct a secure system* from the decision on *whether a system exhibits security as an attribute*.

Suffice it to conclude from this summary of related work that the topic of security metrics per se is an object of study separate from the study of security itself. In fact, more thorough treatments of the subject are available (Herrmann 2007; Bayuk and Mostashari 2012). There are now numerous forums dedicated to various topics in security metrics (Forums ongoing). In addition to the topics discussed above, they include efforts to standardize security incident descriptions, efforts to quantify security risk, and how-tos on using various vendor tools to produce metrics reports. The *Metricon* workshop has drawn enthusiastic participants since 2006, and ACM launched an international version called MetriSec in 2009 (ACM 2009; Metricon ongoing). If there is a common view on security metrics, it is that the field is immature and in need of collaborative scrutiny. As one security practitioner-turned-researcher succinctly put it, “Current approaches do not generate all of the requisite measures and metrics, and those that are produced are insufficient to make appropriate decisions and take necessary actions.” (Axelrod 2012, p.152) A common joke is that the accounting profession has been working since 4000B.C. to come up with the Generally Accepted Accounting Principles, yet there is an unrealistic expectation that measures of control over technology will fully materialize in our lifetime.

2.2 Security Metrics Taxonomy

A common theme in security metrics literature is that taxonomies of security metrics tend to address technical configuration and operational process from the point of view of security management rather than attempt to measure progress toward business goals for security (Savola 2007). Even taxonomies that include governance in addition to management tend to focus on the security management tasks that are evidence of governance, and those metrics may also be considered part of the enterprise security management category (CISWG 2005). These taxonomies reflect a bias toward measuring whether management has correctly implemented mechanisms chosen to effect security, rather than a full assessment of whether those mechanisms are effective. A more comprehensive taxonomy of security metrics that includes all those discussed in section 2.1 is in Figure 1.

Figure 1. Taxonomy of Security Metrics

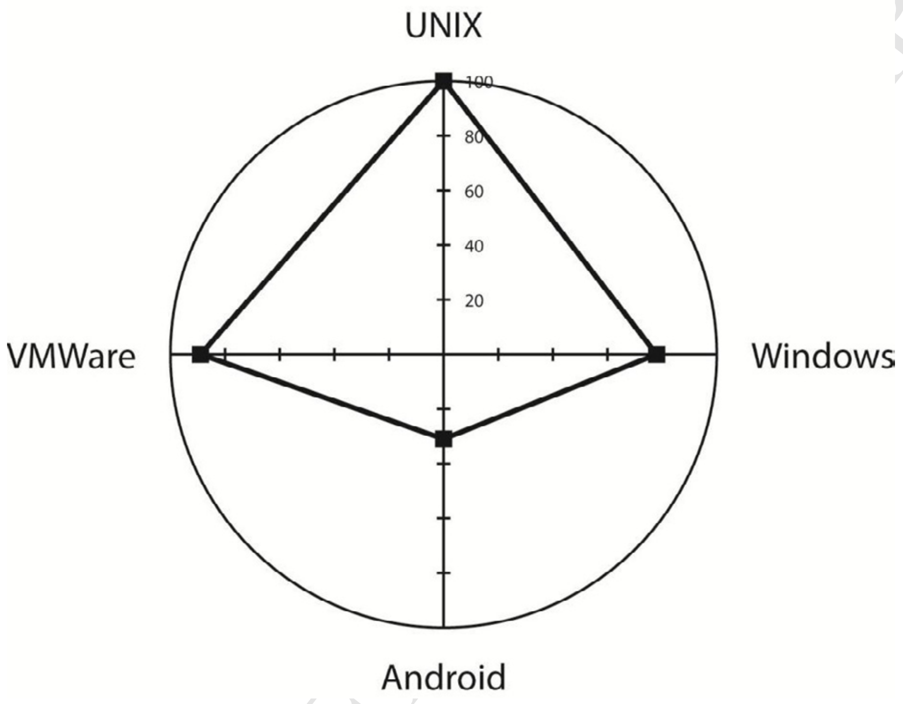


SECURITY METRICS												
ASSESSMENT						CONSTRUCT						
CONTENT			BEHAVIOR			THREAT		MODELS		ACTIVITY		
TARGET	MONITOR	REMEDIATION	PERF	VULNTEST	RESILIENCE	SKILLS	GOALS	STOCHASTIC	DETERMIN	INTERNAL	EXTERNAL	

Content metrics are the most common security metric among practitioners, and Metricon often includes presentations and panels wherein practitioners share their approaches to ensuring that systems have been implemented according to specifications tied to security requirements (Metricons ongoing). The label *content* denotes easily identifiable system artifacts, such as configuration and log files in system components. Content measures are typically automatically gathered based on an itemized system inventory, whose numbers may be in flux, yet each item contains objects whose security attributes are similar enough to be comparable, such as password strength and authorized user accounts. There are three types of content metrics: targets, monitors, and remediation.

Target metrics identify a set of devices or people, and test each member of the set to see that it conforms to a given security specification. For example, Figure 2 presents the result of configuration tests of 100% of the devices in inventory by device operating system, and shows the percentage of each that conform to secure configuration specifications. The secure configuration specification may be enterprise specific, or may follow a published specification such as the Security Consensus Operational Readiness Evaluation (SCORE) operating system checklists (SANS Institute ongoing). The number of devices in inventory running the same operating system provides the 100% target for each axis of Figure 3. Call the number of devices running Windows X and the number of devices running Windows that pass security configuration the SCORE Windows checklist Y. Figure 2 shows that Y/X is 0.8, achieving an 80% compliance rate for Windows, while the devices running UNIX are all compliant.

Figure 2. Example Operating System Security Target Metric



Monitor metrics also exhibit (in)correct content of security implementation, but measure whether a specified process designed to provide security is followed. For example, Figure 3 is a secure development process that may be measured in multiple dimensions in order to produce a metric that shows whether the process is followed. Each measure may be quantitative, as follows:

Measure DM:	identifier of development manager
Measure DMS:	software products delivered by DM
Measure D:	count of developers reporting to manager DM
Measure S:	count of security-trained developers reporting to DM
Measure V:	number of known vulnerabilities identified by pentests in quality assurance (QA) environment for DMS

Measure P:	number of known vulnerabilities identified by pentests in production environment for DMS

From these quantitative measures, an overall software security metric may be produced for each development manager, as in:

```
For each DM:
    If  (V = 0) and (P = 0) Then DM = Produces Good Code
    If  (D > S) Then DM = Takes Shortcuts
    If  (V > 0) Then  DM = Exhibits Negligence
    If  (P > 0) Then DM = Produces Bad Code
```

These numbers would typically be accumulated for all development managers in an organization, or for all development managers in an enterprise, as illustrated in Figure 4.

Figure 3. Example Secure Development Process

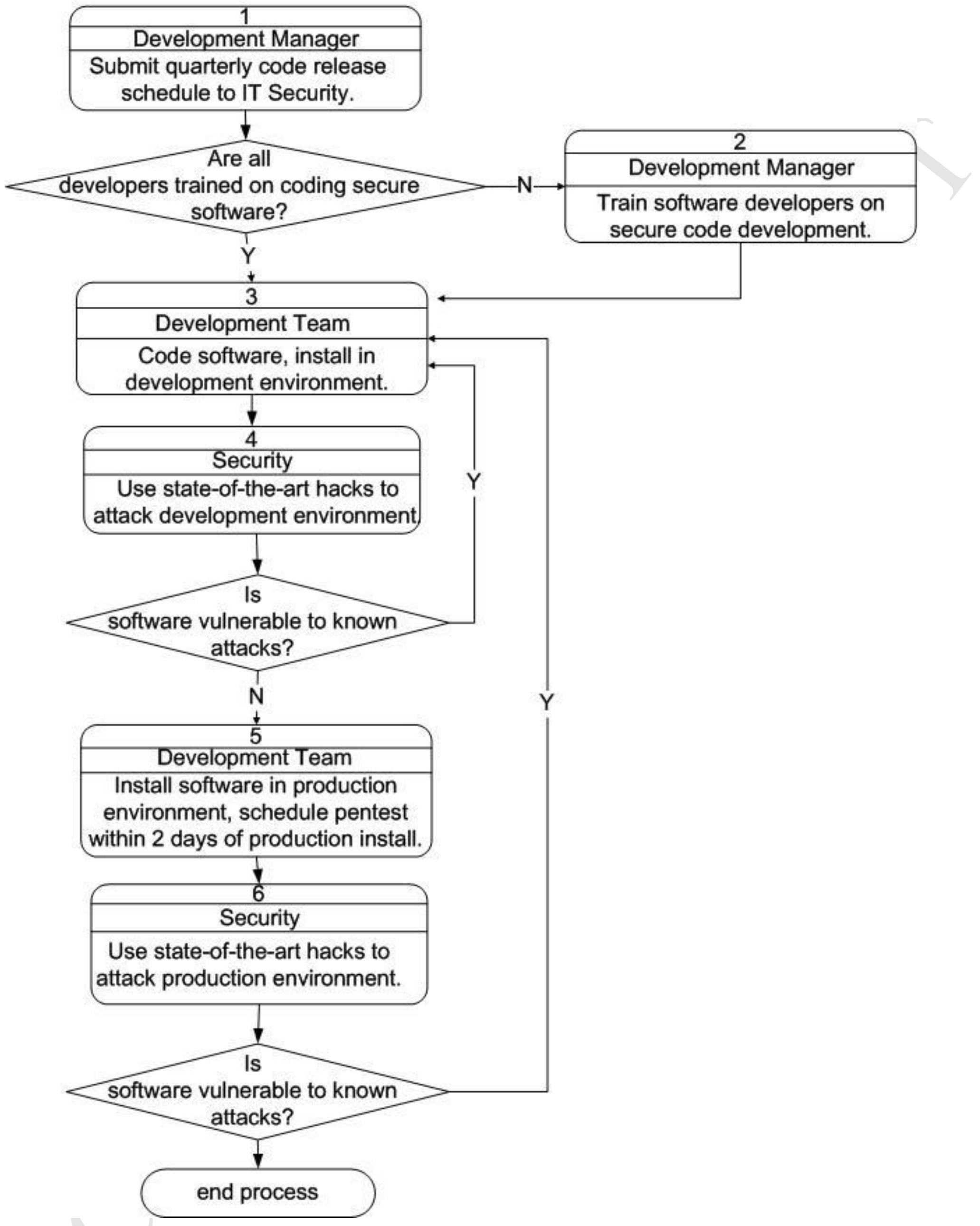
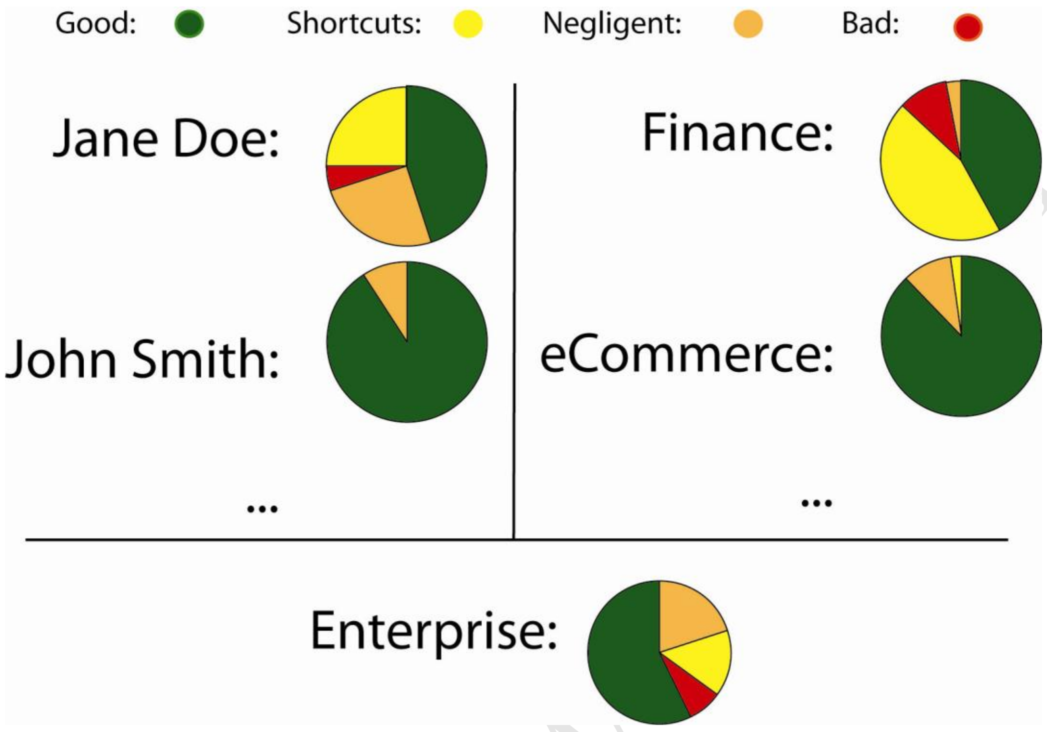

1
Development Manager
Submit quarterly code release schedule to IT Security.
Are all developers trained on coding secure software?
N
2
Development Manager
Train software developers on secure code development.
Y
3
Development Team
Code software, install in development environment.
4
Security
Use state-of-the-art hacks to attack development environment
Is software vulnerable to known attacks?
Y
N
5
Development Team
Install software in production environment, schedule pentest within 2 days of production install.
6
Security
Use state-of-the-art hacks to attack production environment.
Is software vulnerable to known attacks?
Y
end process

Figure 4. Secure Development Process Monitor Metric



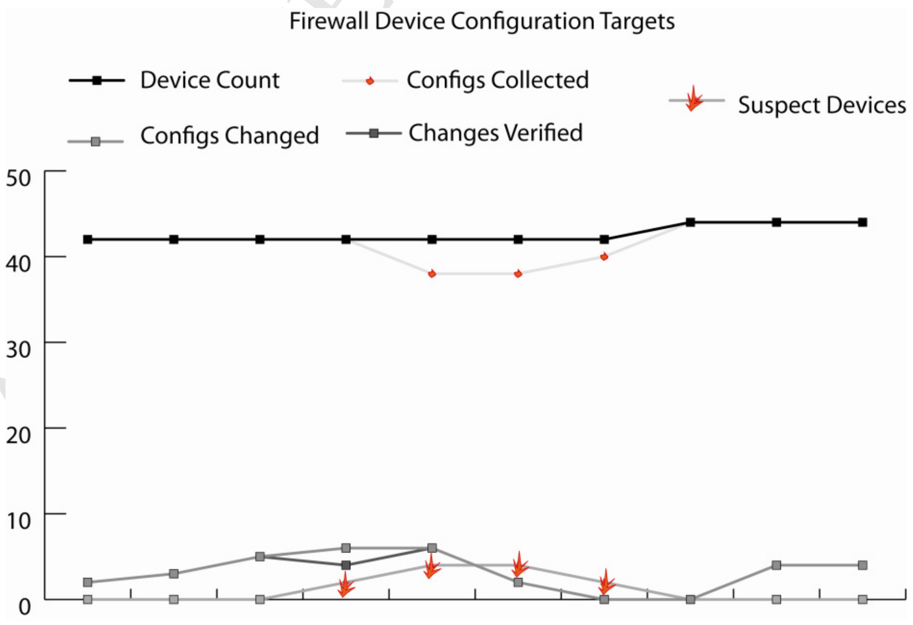
The reason why content metrics resolve to process metrics as well as target metrics is that targets are achieved by people, and if people do not behave in a manner dictated by process, then management does not know whether targets would have been achieved had process been followed. However, if management can verify that process was followed, and targets are still not achieved, then this situation could mean that processes are not working and should prompt consideration of changes in process. Hence in practice, target and monitor metrics are often combined. Figure 5 shows the combination in the service of firewall integrity monitoring. The measures contributing to the presentation are:

Device Count:	The number of firewall devices in operation.
Configs Collected:	The number of firewall devices whose configuration was retrieved in past 24 hours by network management system.

Configs Changed:	The number of firewall devices configurations that deviate from yesterday's configuration.
Changes Verified:	The number of deviant device configurations where operations network can confirm that deviations directly compare to authorized planned changes.
Verification Mistakes:	The number of false negative comparisons by network operations staff.

In the example of Figure 5, the daily metric comparison shows that the organization occasionally misses checking a configuration and also that some comparisons made by operations find that updates were incorrectly performed. Moreover, the added "Suspect Device" pointed to by the arrow was not part of the auto-generated graph, but added after a monitoring metric found a verification mistake by network operations staff. Such a metric presentation shows problems both with maintaining security and with metrics collection. As many such security metrics are not mature enough to be reliable by management control standards, it is important for decision-makers to understand the detail behind any high level security metrics presentation.

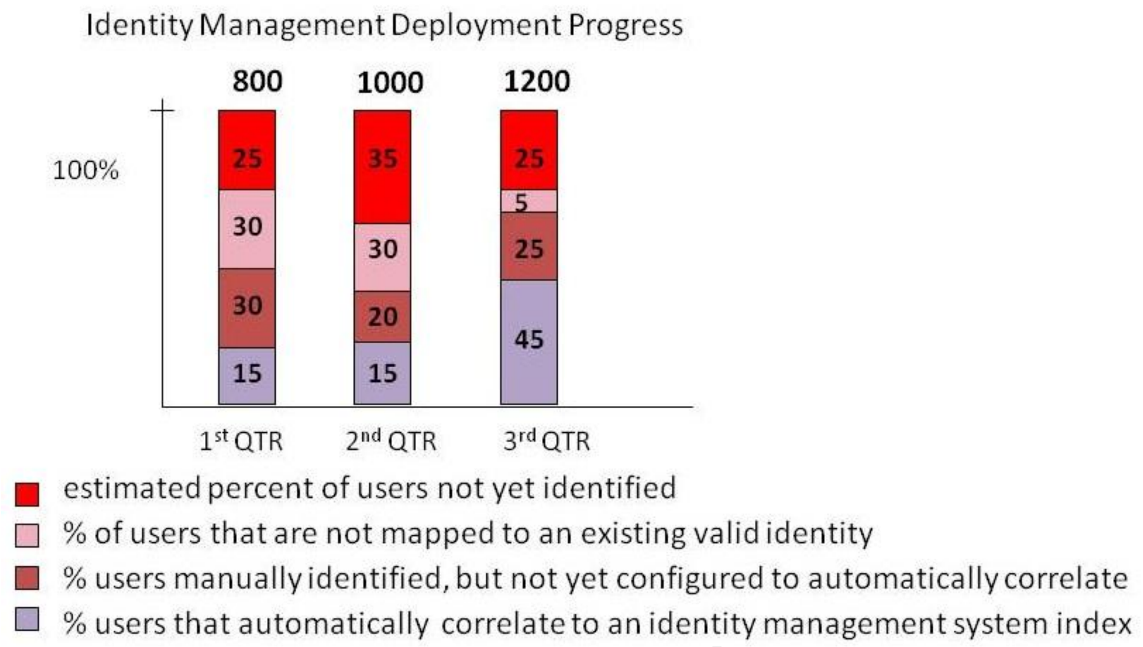
Figure 5. Example Firewall Combined Target and Monitor Metric



Where process to maintain secure configuration is broken or is yet to be developed, projects are typically developed to bring about a secure configuration and associated security maintenance process. These projects are tracked using standard project management tools and *security remediation metrics* may supplement such tracking. However, the extent of the deviation from the desired security specification is often unknown at the start of the project, and milestones often have to be revised in the course of the project execution. Figure 6 shows remediation metrics for a project designed to identify every user in an enterprise and correlate that user with a record for an authorized person in a central identity management system. At the start of the project, it was estimated that there were 800 authorized users in total, and that 15% had so far been correlated. As the project continued, however, it was discovered that the as-yet uncorrelated users required additional entries to the central identity management system, and the estimate of both the total number of users and the percentage of those as yet to be correlated increased. Remediation metrics are content measures because they provide information on known deviations in implementation from the specified secure design.

The security metrics taxonomy *assessment* category includes not only content metrics, but behavior metrics. Where content metrics show correct design configuration and operation, but systems fail due to cyber attacks, they are clearly not secure, and so the result of a security content measures cannot be the sole component of evaluation metric. Hence, security assessments should measure how systems behave while under attack, and compare the measures to criteria for secure behavior, which includes capability to accomplish system functionality while under attack.

Figure 6. Example Identity Management Remediation Metrics



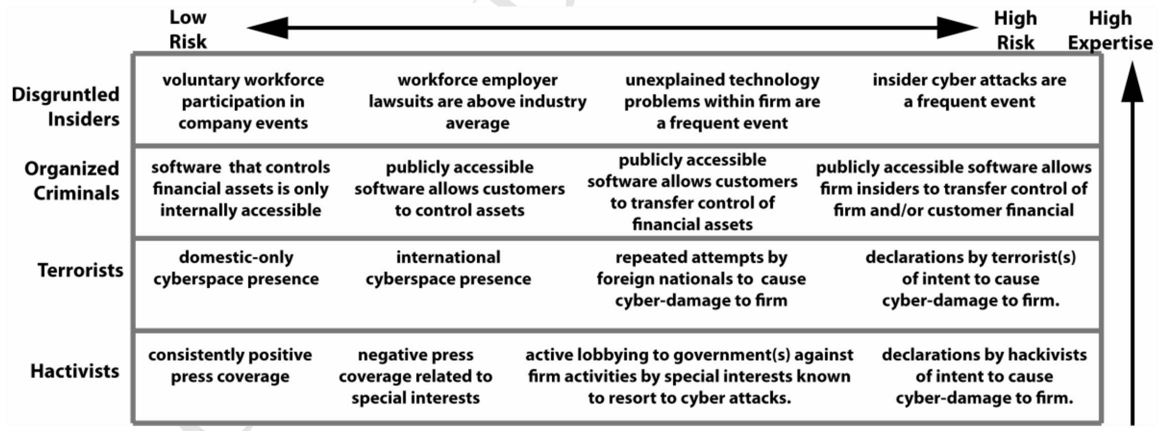
In order to know whether system performance has been adversely impacted by attacks, it is important to first understand what the system is supposed to accomplish whether or not it is under attack. Most organizations that invest in security metrics already have some program in place to measure system performance. These measures are not security-specific and are more thoroughly addressed in quality management literature dedicated to topic such as Six Sigma and Software Quality Assurance (e.g.: Fenton and Pfleeger 1997; Pande, Neuman et al. 2001). The reason they are components of security assessment metrics is that integrity and availability are major components of security, so when these metrics degrade, a secure system should respond to fortify itself against failure.

As the NVD contains readily available information on how systems have been attacked in the past, due diligence requires that security assessments measure whether exploit of these known system vulnerabilities will cause performance degradation. This is the province of penetration, or badness-o-meter testing (Wilhelm 2009). Such testing may measure, for example, the count and NVD-rated severity of vulnerabilities identified, the percent of vulnerable systems, the skill level required to execute the test, and/or the system impact (IDART 2008). Where attacks are not preventable, such as desktop attacks on customers, due diligence requires that response activities be put in place to detect and respond to security breaches (FS-ISAC 2011). Such response activities are typically measured by the time it takes to detect and/or thwart the attack.

Where the methods to be employed by adversaries are not yet known well enough to design corresponding vulnerability tests, behavioral assessments should include more generic resiliency measures. For example, Espenschied and Gunn describe adversary activities in the abstract using general military terms such as *reconnaissance*, *foothold*, *lateral movement*, and *acquire target*, and encourage security decision-makers to take a practical approach to constructing their enterprise security architecture to detect and respond to these generic activities rather than only to NVD-catalogued attacks (Espenschied and Gunn 2012). A secure system should have contingency plans that cover as yet unknown adversary methods. These might be automated reconfiguration in the form of patching, device failover, network path diversity, and/or diverse architecture alternatives. Such resilient behavior is measured with disaster recovery tests and/or table-top exercises of simulated scenarios. To truly measure response to unexpected events, these tests should be designed in the spirit of the Netflix chaos monkey, which randomly takes down system processes to measure systems capability to recover without impact to its mission (Izrailevsky and Tseitlin ongoing).

On the construct side of the security metrics taxonomy are all the metrics that practitioners have not yet figured out how to use, or perhaps did not find useful, in assessing an already-designed secure implementation. Many are nevertheless invaluable in creating requirements for a secure system. Measures of adversary motivation, skills, and justification, as well as intelligence on their goals, methods, and victim selection processes are of invaluable use in designing security measures. Although subjective and ordinal, such as the example in Figure 7, they serve to focus systems engineers on the types of attacks that a system is likely to experience so that appropriate safeguards may be built into the system. Studies such as the Carnegie Mellon Insider Threat (Cummings, Lewellen et al. 2012) and the Verizon Data Breach (Baker, Hutton et al. 2011) reports are of use here, as well as the stochastic methods and experimental models mentioned in section 2.1.

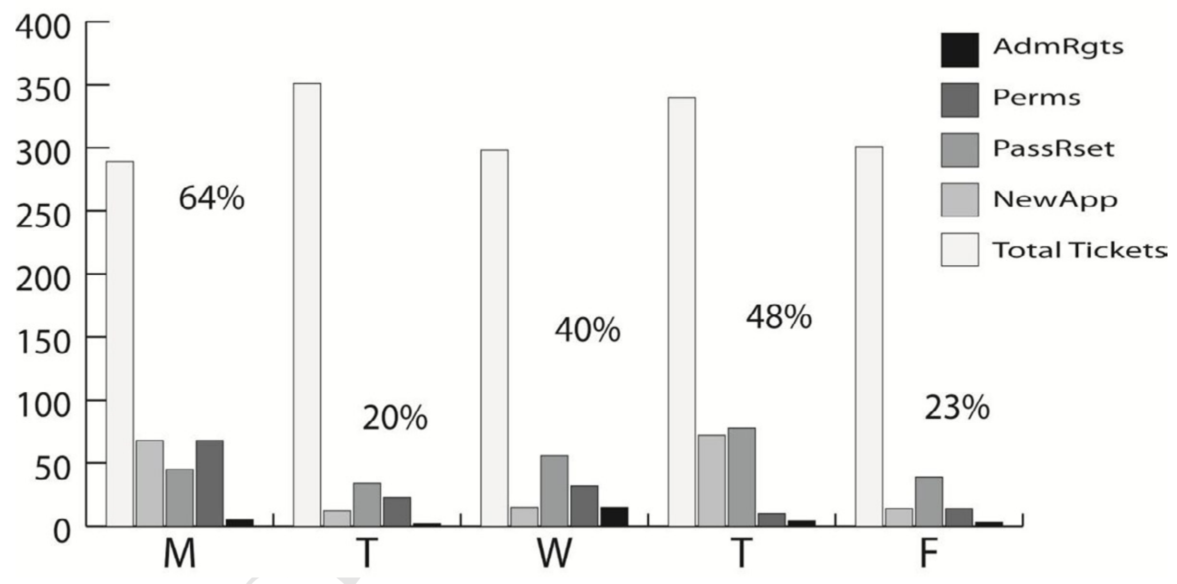
Figure 7. Example Threat Metrics



	Low Risk			High Risk
Disgruntled Insiders	voluntary workforce participation in company events	workforce employer lawsuits are above industry average	unexplained technology problems within firm are a frequent event	insider cyber attacks are a frequent event
Organized Criminals	software that controls financial assets is only internally accessible	publicly accessible software allows customers to control assets	publicly accessible software allows customers to transfer control of financial assets	publicly accessible software allows firm insiders to transfer control of firm and/or customer financial
Terrorists	domestic-only cyberspace presence	international cyberspace presence	repeated attempts by foreign nationals to cause cyber-damage to firm	declarations by terrorist(s) of intent to cause cyber-damage to firm.
Hactivists	consistently positive press coverage	negative press coverage related to special interests	active lobbying to government(s) against firm activities by special interests known to resort to cyber attacks.	declarations by hackivists of intent to cause cyber-damage to firm.

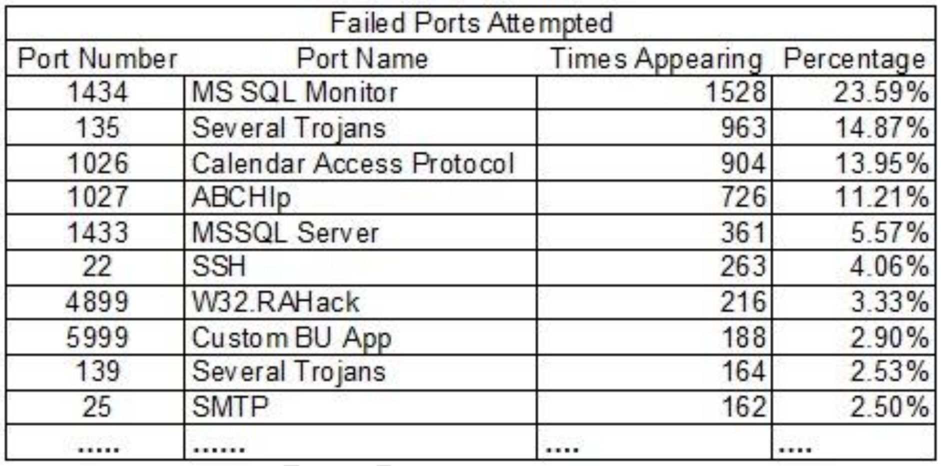
Another source of metrics frequently used in construct decisions are activity metrics. These do not measure any known security property of the threat or system under attack, but related aspects of the systems environment. When they are taken internally, they are sometimes derisively referred to as “busyness” metrics, because increases in the activity measured does not actually add measureable value to the business, nor make it more secure, yet it is important to observe how busy security staff may become as their time is devoted to routine tasks. Figure 8 is an example of such metrics related to the number and type of calls to an internal help desk. This does not immediately contribute to a decision on whether any given system is secure, but trends in this data may motivate decisions to change security constructs. Internal activity metrics also typically include, though of course not limited to, software version releases, customer complaint handling and third party service connection requests.

Figure 8. Security-Related Help Desk Internal Activity Metrics



When activity metrics are taken externally, they are sometimes referred to as "weather" metrics because they are observations of adversary-related activity that does not directly touch the system of interest, but nevertheless provide useful information about trends and potential storms on the system horizon. Dan Geer had a series of columns in *IEEE Security and Privacy* magazine whose content mostly fell into this category, for example, the popular "The Owned Price Index," which calculated the salable value of cybercrime plunder such as social security numbers (Geer and Conway 2009). Figure 9 illustrates an external activity metric, the relative frequency of failed attempts to penetrate an Internet firewall by port number. Again, this metric does not contribute to an immediate security assessment, but, as Amoroso describes in his history of SQL Slammer (Amoroso 2010), adversary activity is often evident on the Internet for months before successful attacks, and such observations may over time provide security managers with valuable information on where to place security resources.

Figure 9. Security-Related Firewall External Activity Metrics



Failed Ports Attempted			
Port Number	Port Name	Times Appearing	Percentage
1434	MS SQL Monitor	1528	23.59%
135	Several Trojans	963	14.87%
1026	Calendar Access Protocol	904	13.95%
1027	ABCHlp	726	11.21%
1433	MSSQL Server	361	5.57%
22	SSH	263	4.06%
4899	W32.RAHack	216	3.33%
5999	Custom BU App	188	2.90%
139	Several Trojans	164	2.53%
25	SMTP	162	2.50%
.....	......	....	....

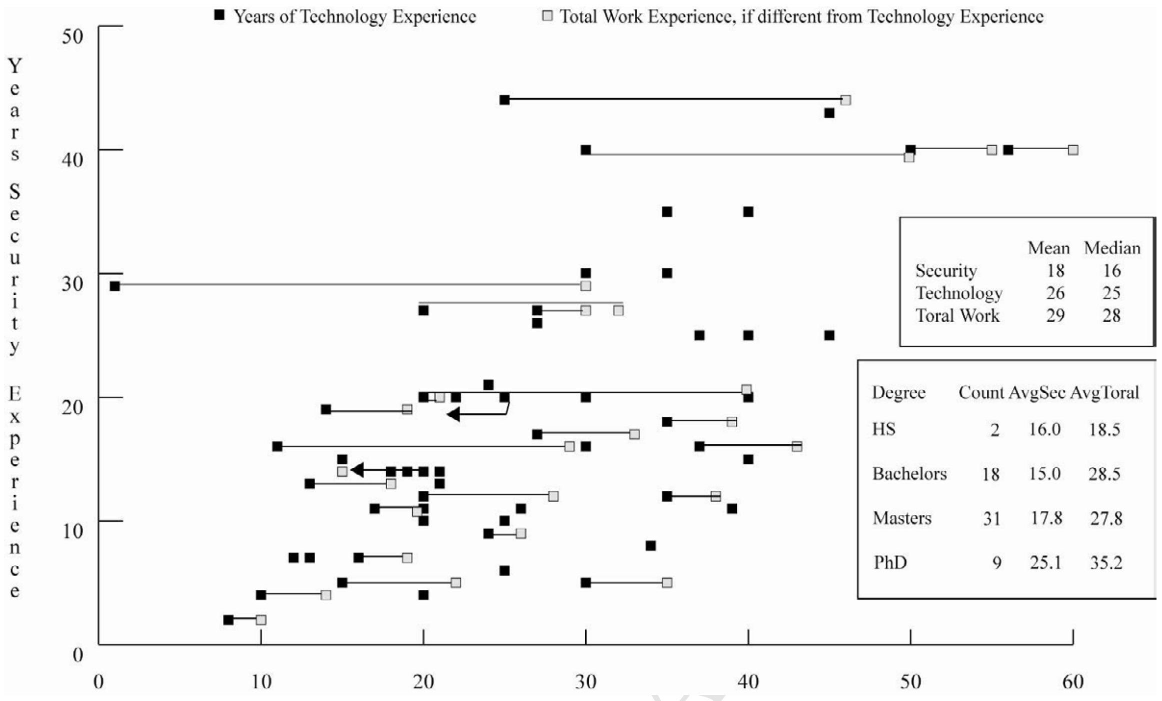
Note that this summary of existing types of metrics is not intended to include recommendations on how security metrics should be used in decision-making. By contrast, research in security risk management is rich with quantitative decision theory, and often replaces ordinal scales of the type in Figure 7 with estimates of corresponding numeric ranges. These studies typically also solicit expert opinions on probabilities of both attack and defense strength, and combine a wide variety of measurements in attempts to influence security spending decisions. These activities, some of which were referred to as models in section 2.1, typically present far more controversial material than the simple measurement-to-metric techniques that have been described in this section.

2.3 Security Metrics Survey

The NIST quote in the introduction refers to "security-enforcing mechanisms of the system" as if these mechanisms can be verified and validated distinct from the system operation itself. This seems to reflect an unstated assumption that any system's functionality may be constrained with security mechanisms, and if we can verify and validate that those mechanisms work, then we can measure security. Over the past half century, this search for a "security system" has created an industry of information security technology products and services.

However, security professionals are repeatedly quoted on record as denying there is any such thing as a Holy Grail. As a way of establishing an authoritative source for putting the Holy Grail search to rest, a survey of highly experienced security professionals was conducted. An illustrative display of the survey respondent's experience and education appears in Figure 10. Following is a brief summary of the survey and results, but interested readers will find both the raw survey data and detailed description of analysis methods at (Bayuk 2011).

Figure 10. Survey Demographics



	Mean	Median
Security	18	16
Technology	26	25
Toral Work	29	28

Degree	Count	AvgSec	AvgToral
HS	2	16.0	18.5
Bachelors	18	15.0	28.5
Masters	31	17.8	27.8
PhD	9	25.1	35.2

The survey was designed to capture the most important attributes of a system to measure in order to decide whether it was secure. However, as there is no established ranking or even comprehensive list of system security attributes, its first challenge was to answer prior studies’ concerns related to ambiguity and environment. Hence, the survey combined questions on the relative importance of a wide variety of potential security attributes, and mixed these with other, non-attribute related questions so that experts could not easily discern, and thus intentionally or unintentionally hinder, survey objectives. Attribute-related questions were ranked using three methods: Thurstone’s method post-initial ranking, where the positioning of items on the Thurstone scale can be found by averaging the percentiles of the standard normal distribution corresponding to the proportions of the respondents preferring one item over each of the others (Thurstone 1928), the One Number Method, which was designed to register strong opinions (Reichheld 2003), and a simple survey rating system based on proportionate number of respondent selections. These rankings were separated into four levels and sent to CISO-level survey respondents who volunteered to be asked follow-up questions.

Survey questions referred to security as a property of a system or as some aspect of a system's component technologies. The goal was to justify, or not, a claim that system-level security architecture attributes are more important than component security mechanisms in a determination of whether a system is secure. The claim was justified by the survey results. The seasoned security professionals who took the survey found this point obvious. The most important attributes selected to measure included:

- Ability to articulate, maintain, and monitor system mission.
- System interfaces accept only valid input.
- Capability for incident detection and response.
- Ability to withstand targeted penetration attacks by skilled attack teams.

The least important attributes to measure included:

- Percentage of systems or components that have passed security configuration tests.
- Ability to maintain values of standard security variables in system technical configuration.
- Ability to pass security audit.
- Security standards used to set requirements.

The results were not surprising because it is evident from the discussion in section 2.2 that assessment metrics typically used by practitioners are based on assumptions that the high level system design provides security. Nevertheless, that assumption is not currently supported with widespread practical metrics while practitioner assessment metrics are for the most part focused on what the survey revealed were the least important attributes of security. These findings reflect the reality that the security profession currently occupies a legislative landscape heavily weighted toward the metrics of low importance. The same set of detailed technology controls are recommended for all systems, albeit with consideration of perceived risk, but with no consideration for the higher level system architecture (Bayuk 2010). These “best practices” are actively lobbied for by security industry vendors, who have no interest in addressing higher-level system security metrics that may possibly make their products redundant. These vendors now create a substantial part of the security literature, which may be one reason why arguments for better security metrics at the system architecture level rarely surface.

3 Security as a Theoretical Attribute Construct (STAC)

After the survey results were analyzed and the ranking was complete, the results fell roughly into four segments. These were labeled, “Most important,” “Next Important,” “Less Important,” and “Least Important.” The labels were meant to facilitate a quick sanity check with the CISO-level subject matter experts to ensure that the survey results were meaningful to them. Most of the experts who reviewed the ranking result reshuffled the ranking of a few items, but a few also objected to the label “Least Important.” For example, *“I find your ranking labels confuse importance. All are of great importance. ‘Least important’ implies not important or immaterial. I would label them as Most Important, Next Most Important, and Important.*” (Parker 2011) This observation correctly reflects the attacker’s advantage and defender’s dilemma in that any weak link in the armor presents opportunity for attack (Howard 2002). Although measuring a detailed technology control such as the *percentage of systems or components that have* *passed security configuration tests* is not as useful a security metric as a system-level attribute *the ability to withstand targeted penetration attacks by skilled attack teams*, where the corresponding technology control contributes to the ability to withstand an attack, such percentage verification is nevertheless important to system security maintenance. The challenge then became to create a system security metric that emphasized system-level secure behavior over component configuration, but also included component measures.

The challenge required a deep dive into systems thinking, a systems engineering methodology that emphasizes structuring the problem and modeling potential solutions in order to test their efficacy in improving the problem situation (Checkland 1999). This exercise led to a view of security as a theoretical concept which encapsulates multiple aspects of a system. In physical theories, such as gravity, multiple measures, if consistent with the theory, do not disprove it, and increasing methods of producing independent measures that do not disprove it serve to make the theory stronger. Of course, it is also true that one failed theory prediction serves to disprove it. Viewing *security as a theoretical attribute construct* allowed the identification of multiple aspects of a system that made it secure, as well as focus on aspects which exhibit a measurable attribute. The survey conclusion that system-level aspects were most important established a priority to consider attributes of effectiveness over correctness, but not any reason to dismiss measurable attributes of correctness. Combining the measures of these attributes paved the way to produce a metric for *Security as a Theoretical Attribute Construct*, the methodology now called STAC.

In computer security (as opposed to physical security), we can comfortably refer to the thing to be secured as a given system of interest, where *system* is a construct or collection of different elements that together produce results not obtainable by the elements alone, and any given system, or a *system of interest*, specifies the one under scrutiny as distinct from its environment (INCOSE 2011). Security is not solely an attribute of a system, but the product of an interface between a system and its operating environment. Systems that have hostile adversaries need to devote more resources to deflecting threats than systems that do not in order to achieve the same attribute of security. Thus to be secure is an emergent property. It is not the sum of parts, but emerges from the interaction between a system of interest and its environment. This observation was reflected both in the literature on holistic security attributes described in section 2.1 and in the survey results described in section 2.3

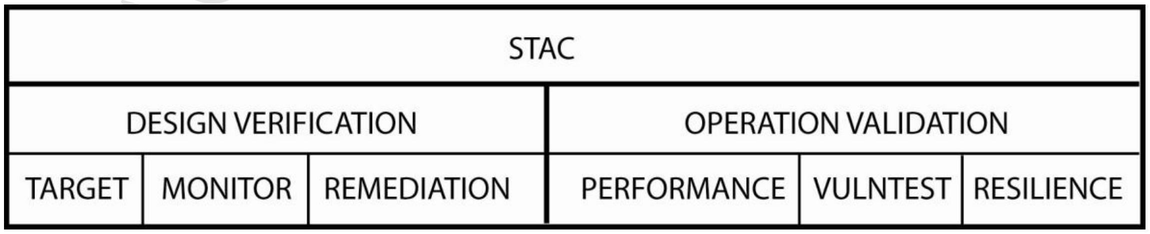
Metrics for security at the system level are combinations of measures of different focus. The elephant may be properly measured from multiple angles as long as there is a theory of the set dimensions outside which the elephantness property disappears (e.g., inch-length nose, tail, and ears means maybe it is a rhinoceros). Metrics that focus on verification that designs are properly implemented are required only because, in their absence, there is no way to know *what the system is* that is being measured. The content metrics used by today's enterprise security programs thus form the basis of the theory that a secure design enables a system to maintain security. However, a decision on whether or not the design does maintain security must also include information on the system's *ability to maintain its mission*, so any system security metric must include some evidence that the system can be used for its intended purpose. To be secure is to maintain functionality *in the face of hostile adversaries* who are expected to take full advantage of the latest vulnerabilities and crimeware technologies. Hence, testing for those vulnerabilities with penetration tests and other badness-ometers are also important components of a system-level security metric. Yet neither can such "*vulntests*" tell the whole story because there will always be vulnerabilities which have not yet been catalogued. Hence, another necessary component of a system-level security metric is evidence that the system does indeed fulfill its mission *despite constant potential for damaging impact*.

In summary, to construct a theory that any given system is secure requires identification of these types of attributes:

1. Correct configuration, to allow for *design verification.*
2. Effective operation, to exhibit:
    a. Ability to accomplish system purpose, or *performance* validation.
    b. Ability to deflect known threats, or *vulntest validation*.
    c. Ability to adapt to unexpected harmful impact, or *resiliency validation*.

The first of these criteria may be measured using content metrics, the remainder may be measured using behavior metrics. Figure 11 depicts the ontology of the STAC metrics formula.

Figure 11. STAC Ontology



STAC					
DESIGN VERIFICATION			OPERATION VALIDATION		
TARGET	MONITOR	REMEDIATION	PERFORMANCE	VULNTEST	RESILIENCE

Note that Figure 11 mirrors the left side of Figure 1. Both design specification verification and design goal validation techniques were discussed in section 2. Techniques for measuring the first attribute, ascertaining correct configuration, are the province of security content metrics. Techniques for measuring ability to accomplish system purpose include the same operational performance monitoring required for proactive maintenance. Techniques for measuring the ability to deflect known threats are the province of vulnerability testing, as well as measures of the time to respond and close detected intrusions and fraud. Techniques for measuring resiliency in the face of harmful impact have been motivated by availability requirements, and also include exercises of operational capacity to adapt to unanticipated threats.

What is new in the STAC approach to security metrics is that nothing can be said about the security of the system unless a combination of all these types of attributes have been constructed into a theory of what it means for a given system to be secure. Moreover, if measures of all these types of attributes simultaneously yield positive assessments of the system security, any security breach of the system means that the constructed theory was incorrect, and the analysis that led to it should be revisited. In effect, the process of creating security metrics should be viewed as creating a hypothesis that, if all security attribute measures return positive results, then the system is adequately secure.

For example, assume an enterprise risk assessment process results in the deployment of an enterprise security architecture comprised of logical access controls limiting data access to insiders, a firewalled periphery monitored with intrusion detection/prevention systems, fully encrypted network traffic, and state of the art security network operations processes to monitor and support the deployed design. Target and monitor metrics confirm the design is implemented correctly. Performance, vulntest, and resilience metrics confirm that the system is operating effectively. Assume further that an authorized insider steals confidential information and sold it to the highest bidder. Clearly, there was a breakdown not in the assessment process, but in the design process, and a forensic investigation in combination with construct metrics may be expected to yield information on which to base design improvements. Contrast this situation with one wherein the logical access controls and network traffic encryption capability were never fully functional nor supported with remediation metrics. In this case, it would be much harder to tell whether or not an insider was the culprit, and also difficult to determine if the design would have worked if it had been implemented. Without verification metrics, validation measures may be rendered meaningless, and visa versa.

System security should be treated like an empirical discovery process rather than a moving target. In the field of scientific validation, content validation implies that the full domain of content is measurable, criterion validation implies that the correspondence between the chosen behavioral criteria and the attribute to be measured is exact, while construct validation allows for a measure of the extent to which a set of measures are consistent with a theoretically derived hypothesis concerning the concepts being measured (Carmines and Zeller 1979). Using these standards of scientific validity, specification verification measures may be considered content valid because the full domain of system components are measurable, behavioral measures should be considered criterion valid when there is a direct correspondence between secure systems and those that withstand attacks, and a theoretical model of a secure system may be used to construct a combination of content and criterion measurements that provide evidence that a given system conforms to the model. Where there is evidence that a system conforms to a theoretical model of security, then lack of security breaches will support the theory, while a single security breach will either disprove the theory or call the integrity of the measurements into question.

A *system-level* security theory is important because it targets the goal of a secure system, which is easier to envision than an elusive property called security, which should somehow make availability, integrity, and confidentiality attributes visible. Rather than attempt to make a *security function* the target, it promotes a view of security as a property of the system of interest that emerges from the way the system is designed, configured, and operated. Moreover, assessments based on STAC include design verification as an *essential* component. Hence, incomplete verification does not simply inhibit one’s ability to measure security, it implies that the system is *not* secure because the definition of security includes the ability to measure the extent to which design specifications are met. Additionally, the validation measures in STAC allow two systems of different design with similar missions to be compared to see if one is architecture is more secure than the other.

This point is important because a more interesting topic to many researchers is how to use quantitative methods in security-related decision-making. The purpose of this paper is to bring recognition to the fact that there is much to be understood about how security is (and should be) measured prior to relying on security metrics to support decisions. There is a plethora of literature in security risk management that makes the explicit assumption that there is a security expert somewhere that can adequately assess the security stature of a system, as well as estimate the probability that it will survive an attack (e.g. some mentioned in section 2.1 as well as: Cavusoglu, Raghunathan et al. 2008; Ioannidis, Pym et al. 2009; Eskins and Sanders 2011). Verendel provides a good example of a disclaimer that typically appears in these publications: “Even if such modeling is clearly very challenging, in this paper we will assume that a decision-maker is provided with the result of security risk modeling.” (Verendel 2008) While the study of security decision theory is of course a worthy endeavor, until there is some consensus on a security metrics framework such as STAC, the security decision theory field is, in some sense, an orphan.

Note that STAC itself is not a security risk management framework, but a security measurement framework, which is a prerequisite to security risk management. Recent NIST guidance on the management of information security risk clearly differentiates how risk assessment information is framed and communicated from how risk assessment and response are conducted (NIST 2011, p.7). STAC addresses the former problem, paving the way for more informed solutions to the latter.

4 Results

4.1 STAC Metrics

Figure 12 shows a security metric derived from STAC for a system whose security has been thoroughly successfully measured. Assume the system is negatively impacted by a security breach. This means that the theory of system security relied on resiliency features that were previously successfully tested. The successful tests supported the theory at the time, but the counterevidence of the breach later proved the theory false. It means that, without any change in the system itself, the metric readings change to look more like Figure 13. That is, in the case where the verification was performed correctly and thus the theory is deemed false, a new theory should be formed, the system should be redesigned, and then measures must be redesigned as well before a new metric is derived. On the other hand, if a root cause of the investigation shows that verification of correctness was previously performed inadequately, then the system's security metric looks instead like Figure 14, and always should have. In that case, the theory may still be true, because if the system had been implemented correctly, then the resiliency features of the original design may have worked to thwart the attack.

Figure 12. STAC Security Metric

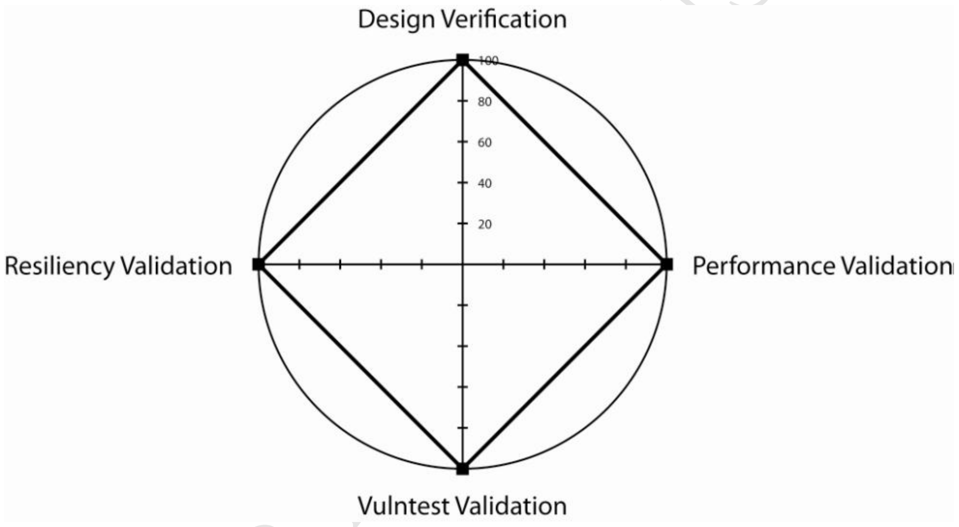


Figure 13. STAC Security Metric after Breach

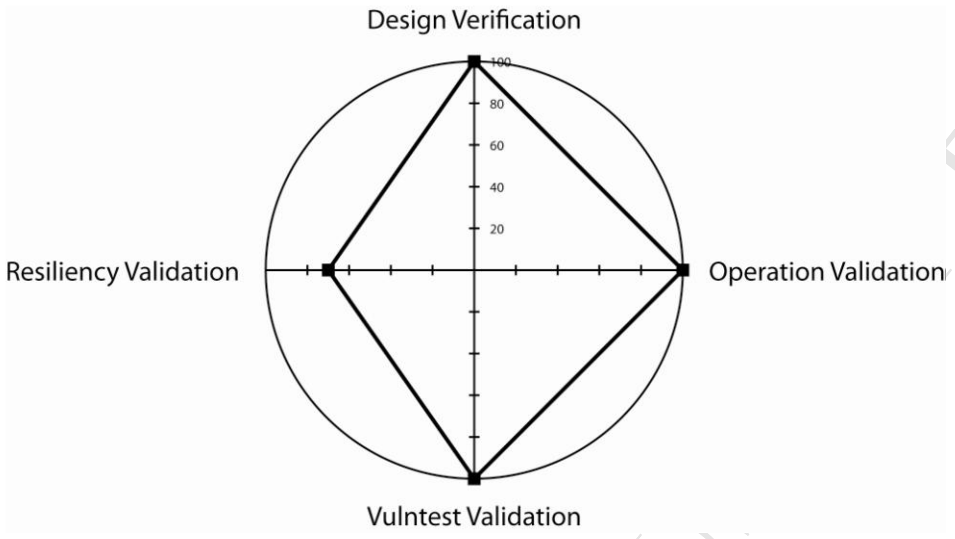
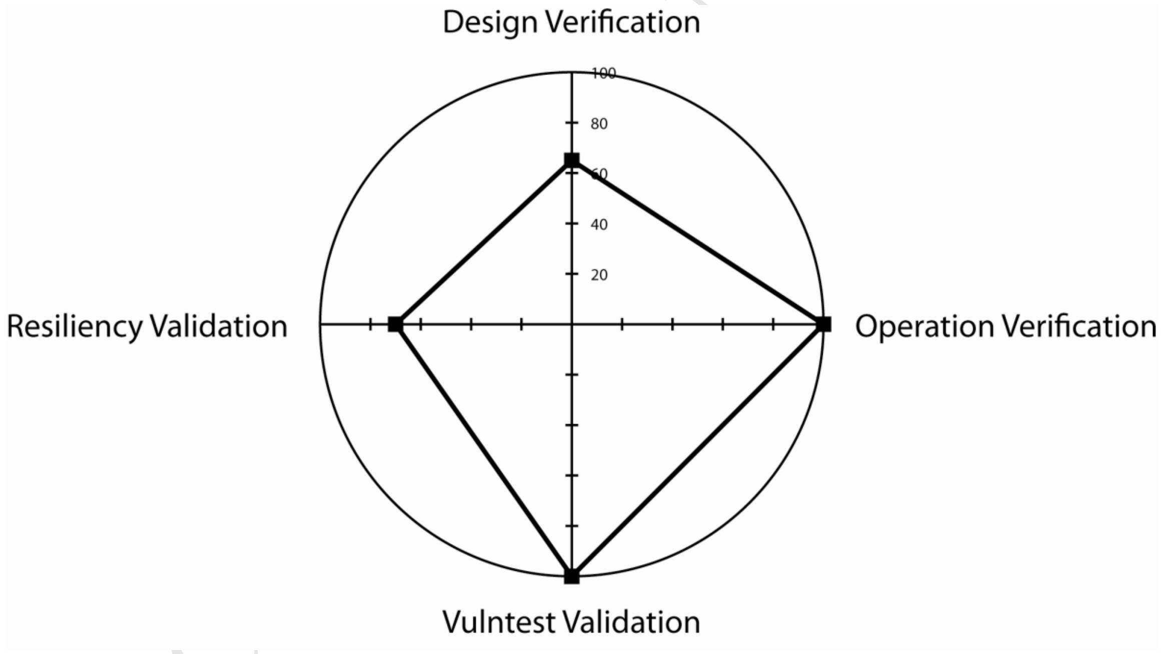


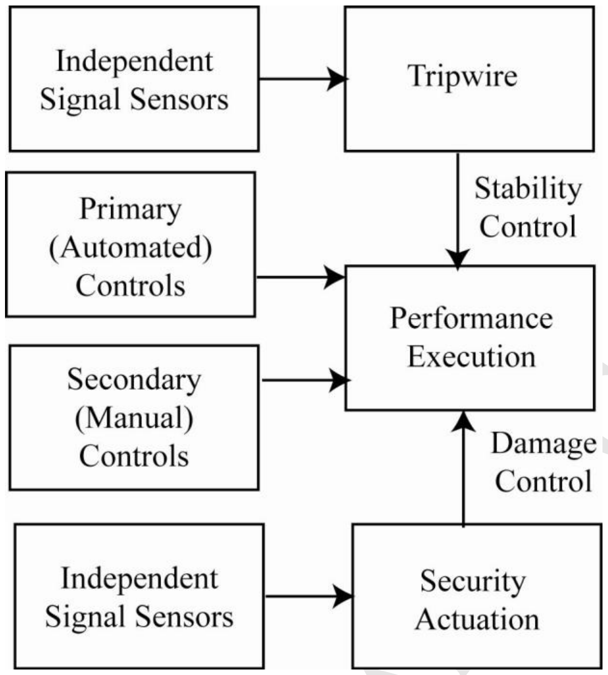
Figure 14. STAC Security Metric after Root Cause Analysis



Each of the four measurable components of the STAC will of course be based on its own construct. For instance, the *Design Verification* measure at the top of Figure 12 is a composite of an underlying set of measures that justifies the claim for 100% verification of technology design specifications. This would typically also include many of today's management metrics, customized to support the theory's concept of secure operation.

A demonstration of these concepts requires example architecture that is based on a theory that a set of system attributes are sufficient to protect the mission of a given system. One such system security theory has been derived from the nuclear regulatory model for maintaining system safety. In that architecture, more fully described in (Bayuk, Horowitz et al. 2012) and depicted in Figure 15, the primary system control functions are expected to maintain system stability, but signal sensors independent of the primary control functions also constantly measure stability using separate technology. If the independent sensors detect instability, they trigger a tripwire that automatically deploys resources to reinforce functions determined to be negatively impacted. Independent sensors may also activate a separate *security actuation* feature that detects negative impact and automated triggers damage control measures. In the event all of these mechanisms fail, manual monitoring (e.g. visual inspection) and manually activated controls provide secondary control functions independent of the primary control functions. These can combine with human acumen to recover system performance.

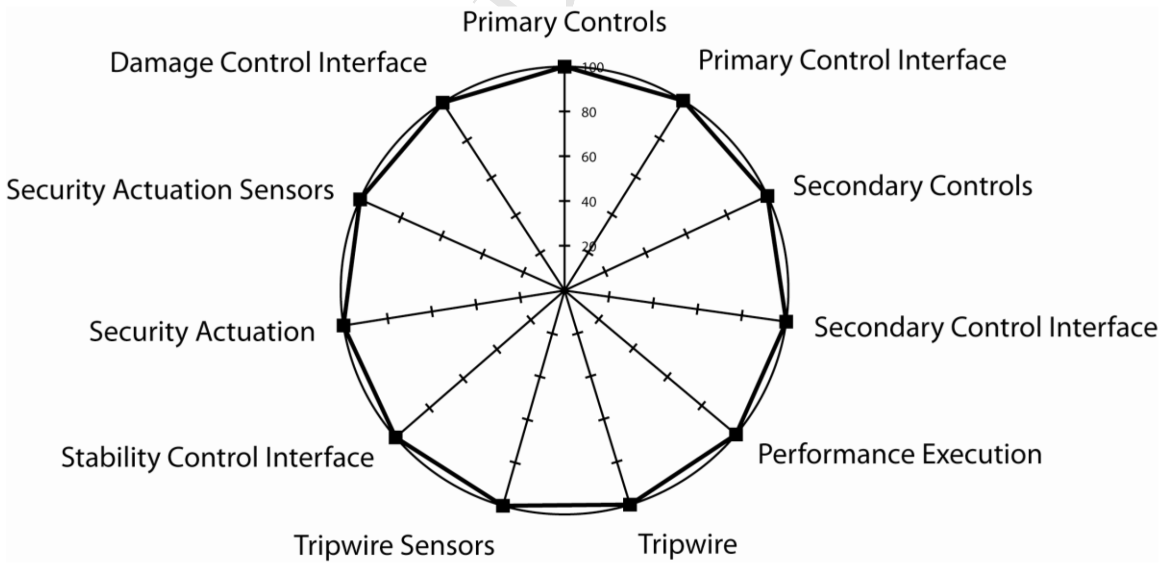
Figure 15. Example Secure System Architecture


Independent Signal Sensors
Tripwire
Stability Control
Primary (Automated) Controls
Performance Execution
Secondary (Manual) Controls
Damage Control
Independent Signal Sensors
Security Actuation

The security theory for the system is that both the primary controls and independent sensors will detect and respond to both known threats and damaging impact from unknown sources, the technology diversity will allow the system to continue to operate despite any subset of components succumbing to attack, and if the automation is defeated, human operators supported by human security forces will be able to safely shut down the system to avert damaging impact. If theory is valid, and the system is implemented correctly, then the system should be able to withstand both expected and unexpected attacks while maintaining a high level of performance.

In order for a STAC Design Verification measure to accurately reflect correctness, each of the system components must be independently measured to ensure they are configured to specification, and the interfaces between them and the system operating environment must also be thoroughly measured. These measures would have to verify manual as well as automated response features. Figure 16 illustrates the desired view when the measurement is complete. Anything less than 100% compliance with design specifications in any component or interface would be reflected in the Design Verification dimension of Figure 12. System level metrics presentations should be designed so that failed Design Verification dimensions such as those in Figure 14 would drill down into figures like Figure 16 so the component failure impact on system-level security may be analyzed and understood. In this case, the architecture is composed of functional blocks designed to avoid common mode failures, so the decomposition of verification falls also along those lines. The 100% targets of these functional blocks and interfaces should be supported by both target and monitor metrics commensurate with the verification requirements for the correct implementation of each component. This is a top-down approach that allows a decision-maker to mine the detail supporting the higher-level numbers.

Figure 16. Example System Architecture Design Verification Metrics



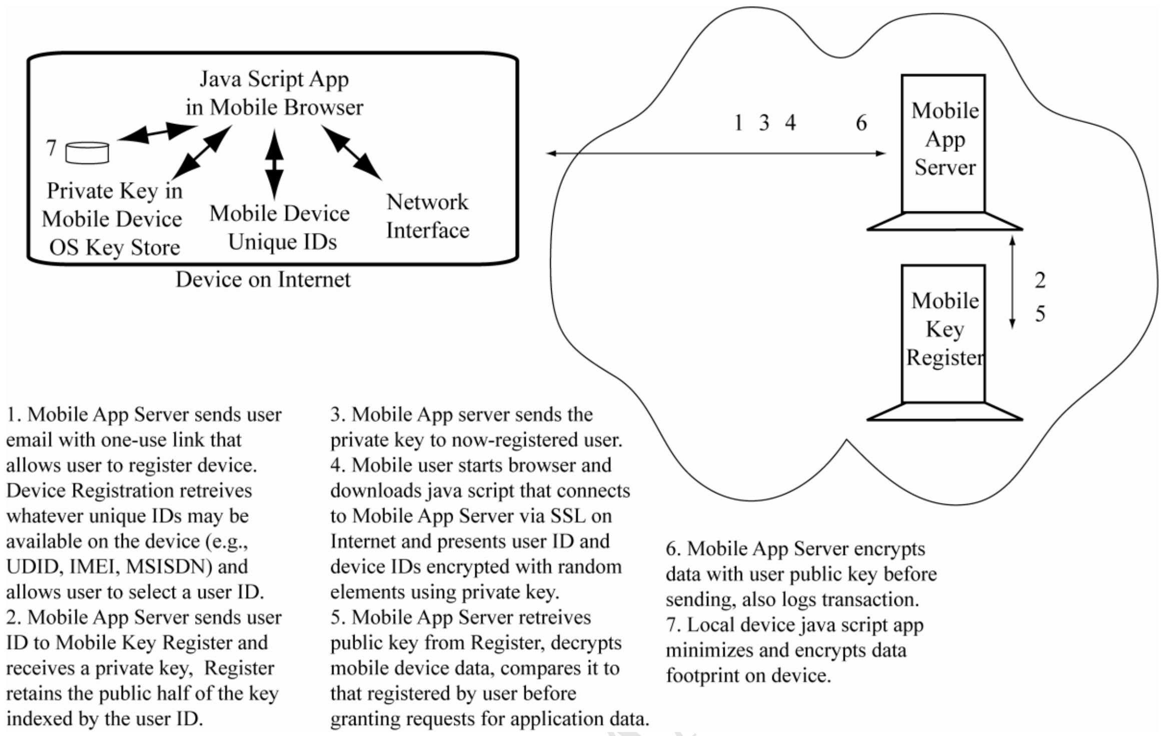
Where any target or monitor metric fails 100% testing, a remediation metric should be identified and its efficacy assessed as part of an overall system security assessment. Note that, although STAC does provide quantitative measures used for decisions concerning security assessment, the numbers generated by STAC, as in any metric, merely support a decision rather than replace it. Combined with an organizational structure for accountability, the details could be used to support decisions concerning security management as well as security assessment.

Operation Validation metrics, by contrast, would not be based on decomposition into component testing, but on system-wide measures of behavior in three very different types of scenarios: steady state, anticipated attack, and unanticipated attack. These correspond to STAC performance, vulntest, and resilience metrics, respectively. Performance metrics would be based on expectations for system functionality, in this case, the ability to deliver nuclear power. Vulntest metrics would be based on documented attack scenarios enacted by a red team. Resilience metrics would be based on a set of disaster recovery exercises that provide closure on diverse combinations of component failure, including total system failure.

4.2 Mobile Security Case Study

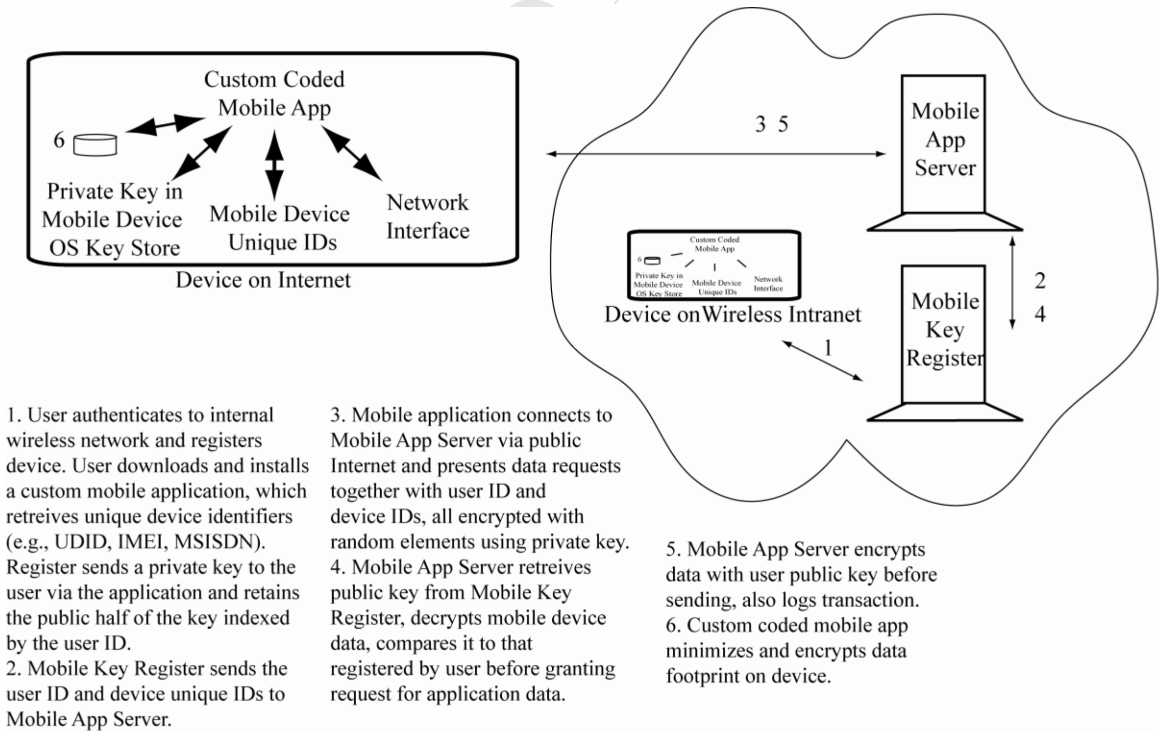
To see that STAC can be practical and useful, consider an enterprise mobile communications system designed to allow company staff to access confidential company information via a mobile device that is personally owned and operated by a company employee or contractor. The system of interest is the collection of different elements whose combined architecture allows staff to use mobile devices to access an application that is supported by the company, and to send and receive information to and from it. Figure 17 and Figure 18 illustrate alternative architectures designed to accomplish the system purpose.

Figure 17. Mobile System Architecture A



1. Mobile App Server sends user email with one-use link that allows user to register device. Device Registration retreives whatever unique IDs may be available on the device (e.g., UDID, IMEI, MSISDN) and allows user to select a user ID.
2. Mobile App Server sends user ID to Mobile Key Register and receives a private key, Register retains the public half of the key indexed by the user ID.
3. Mobile App server sends the private key to now-registered user.
4. Mobile user starts browser and downloads java script that connects to Mobile App Server via SSL on Internet and presents user ID and device IDs encrypted with random elements using private key.
5. Mobile App Server retreives public key from Register, decrypts mobile device data, compares it to that registered by user before granting requests for application data.
6. Mobile App Server encrypts data with user public key before sending, also logs transaction.
7. Local device java script app minimizes and encrypts data footprint on device.

Figure 18. Mobile System Architecture B



1. User authenticates to internal wireless network and registers device. User downloads and installs a custom mobile application, which retreives unique device identifiers (e.g., UDID, IMEI, MSISDN). Register sends a private key to the user via the application and retains the public half of the key indexed by the user ID.
2. Mobile Key Register sends the user ID and device unique IDs to Mobile App Server.
3. Mobile application connects to Mobile App Server via public Internet and presents data requests together with user ID and device IDs, all encrypted with random elements using private key.
4. Mobile App Server retreives public key from Mobile Key Register, decrypts mobile device data, compares it to that registered by user before granting request for application data.
5. Mobile App Server encrypts data with user public key before sending, also logs transaction.
6. Custom coded mobile app minimizes and encrypts data footprint on device.

Mobile System A is designed for maximum ease of deployment and use. It allows users to connect from the Internet to register their devices and run the application from any mobile device browser. Mobile System B requires that users connect their devices to the enterprise intranet to register for the application, and also to install a custom-built software application on their mobile device. In both cases, the theory of system security is that the ability for the application server to automatically recognize the registered mobile device will minimize risk that application data will be exposed to unauthorized individuals, and that at worst case, data exposure on lost or stolen devices would be limited to small quantities of data of relatively low sensitivity. However, in the first case, the theory considers user authentication from external networks a valid identification mechanism, and in the second case only user authentication from previously identified access points is considered valid. Translated into the high level security theoretical attributes described in section 3, Mobile System A would have these attributes:

1. Verified ability for the application server to automatically recognize only registered mobile device users minimizes risk that application data will be exposed to unauthorized individuals.

2a. Users shall have access to application anywhere, anytime, from any device.

2b. Vulntest shall reveal, in worst case, data exposure on lost or stolen devices would be limited to small quantities of data of relatively low sensitivity.

2c. Diverse Internet architecture and agile software support structure render system flexible enough to adapt to unexpected attack.

Mobile System B would have these:

1. Verified ability for the application server to automatically recognize only registered mobile device users minimizes risk that application data will be exposed to unauthorized individuals. Same basic attribute as A, though different components selected, based on difference in performance requirement of #2a.

2a. Users shall have access to application anywhere any time, from devices preregistered on the internal network. This differs from A's performance validation attribute (2a).

2b. Vulntest shall reveal, in worst case, data exposure on lost or stolen devices would be limited to small quantities of data of relatively low sensitivity. Same as A.

2c. Diverse Internet architecture and agile software support structure render system flexible enough to adapt to unexpected attack. Same as A.

The corresponding system architecture design verification metrics for these security theories might look like Figure 19. Like the example in the previous section, it assumes that an attempt has been made to ensure that functional block components do not have common mode failures, can be individually verified, and that interfaces between them are well-defined. Requirements for each component would be used to specify its configuration, and automated tests would be designed to verify that the configuration is correct. For example, in System B, the custom mobile app performs several security features. The configuration supporting each feature would combine with those of other features required to be performed by the device into component-level security metrics as in Figure 20.

Figure 19. Mobile System Design Verification Metrics

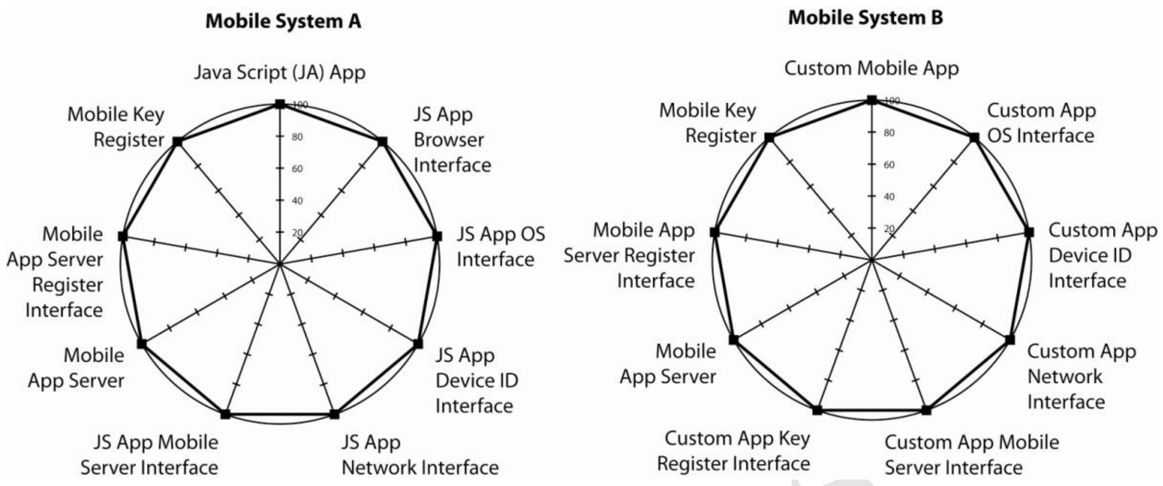
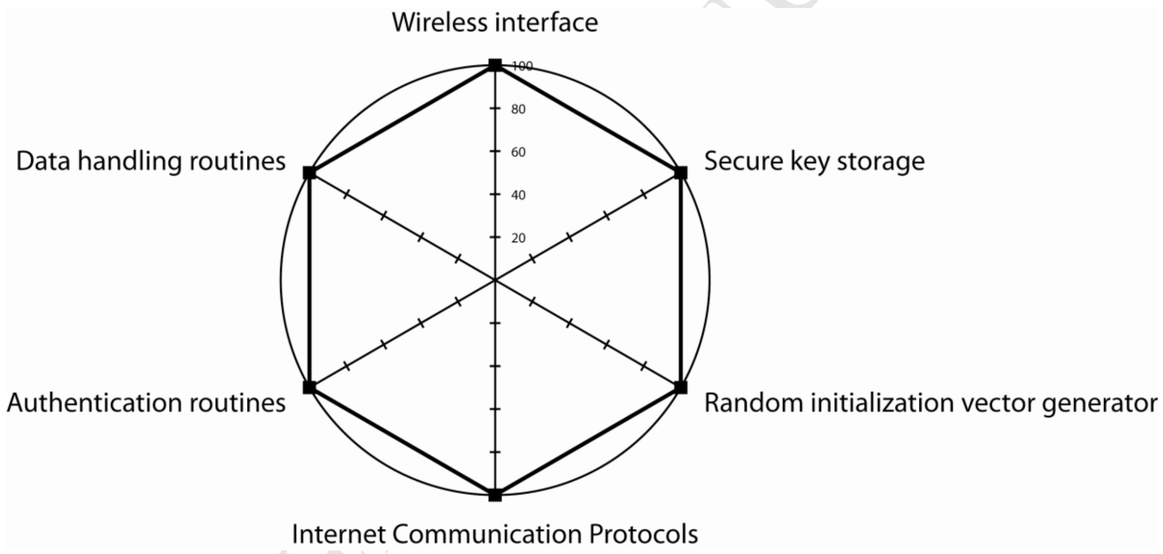


Figure 20. Custom Mobile App Verification Metrics



One of the features of Mobile System B is that a Custom Mobile App is to minimize and encrypt the data on the mobile device (Step #7 in Figure 18, allocated to the “data handling routines” in Figure 20). The requirements and associated configuration measures for the associated data handling routines may include a set like the one in Figure 21. Following STAC, positive results of each configuration comparison would contribute one-eighth, or 12.5%, of the total metric for the feature in the 100% aggregated display for its associated component configuration metric. In practice, security professionals often weight test results using some (subjective) assessment of the component's contribution to the overall objective of the feature.

Figure 21. Data Handling Routine Requirements

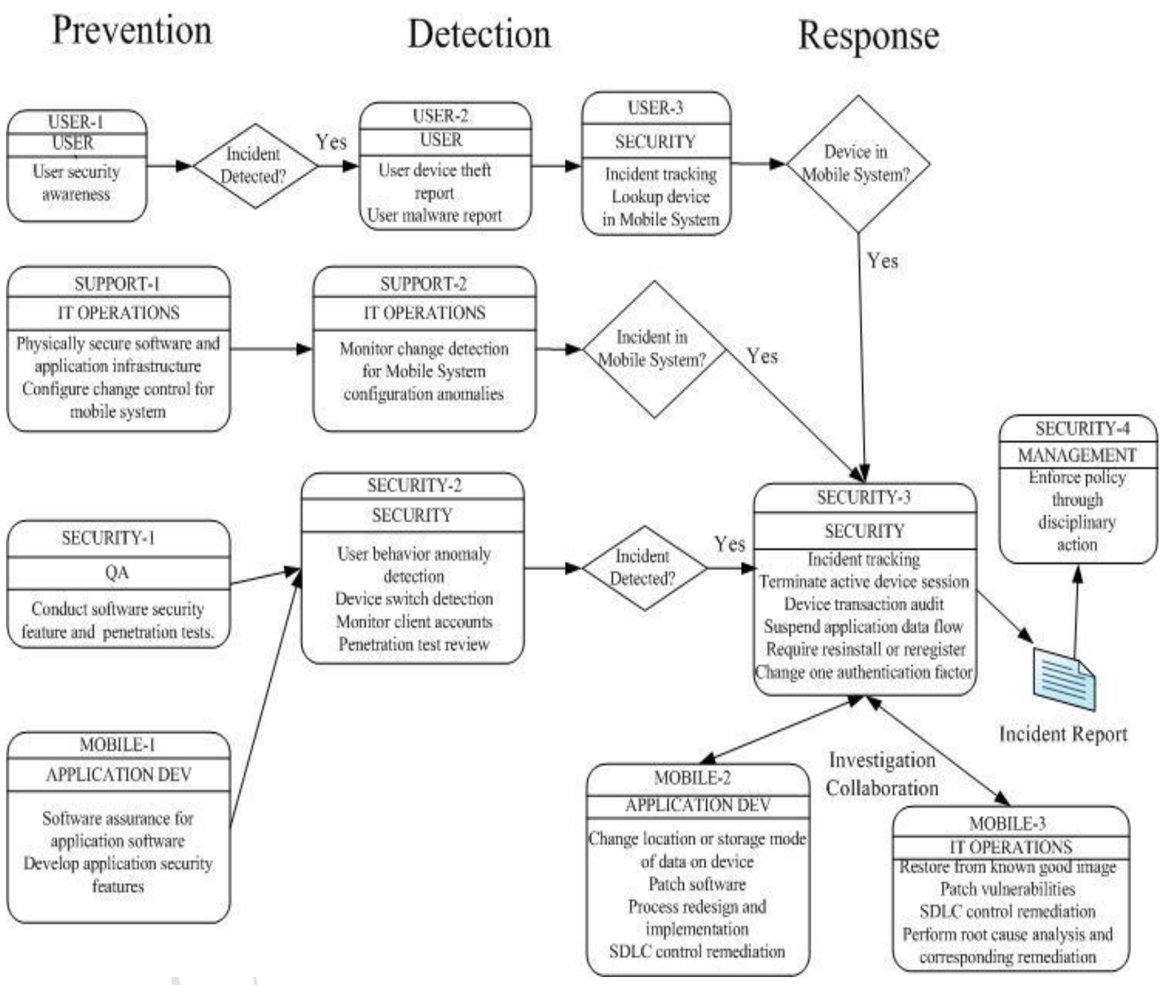
	Mobile Feature Requirement	Configuration Measure
1	Personally identifiable data (PII) shall not be sent to the mobile device.	As part of application QA, extract all data base queries from the mobile code installed on the device and compare to PII fields.
2	Application data entitlements shall be configured according to the principle of least privilege.	Periodically query application entitlement records by user and compare to user job function and scope of responsibility.
3	Applications shall encrypt all data destined for a mobile device.	Periodically during application operation, search the network traffic, memory cache, file system and databases used by the application on the device and compare to recognizable patterns of cleartext data.
4	Mobile application cryptographic algorithms shall have been certified US government cryptographic standards and coded by internationally reputable software vendors.	As part of application QA, use static and dynamic analysis on application source code to compare implementation to cryptography use cases and libraries previously approved as enterprise standards.
5	No data, either encrypted or decrypted, shall be stored on any mobile device, nor shall it remain in device cache or memory after an application session has terminated.	Before and after application operation, search the network traffic, memory cache, file system and databases used by the application and compare to storage parameters indicating null data sets.
6	No data shall reside in browser variables in an unencrypted state.	Periodically during application operation, query mobile browser variables and compare to recognizable patterns of cleartext data.
7	An encryption key and encrypted data shall not be simultaneously stored in non-volatile memory.	Periodically during application operation, examine network traffic, memory cache, file system and databases on the device and compare to encryption key field parameters and patterns of unencrypted data.
8	Application sessions shall automatically terminate if the user is idle for more than 15 minutes.	As part of QA, determine the location of idle time variables checked by application code. Periodically on device, examine idle time variables and compare to 15 minute standard.

Note that several of the configuration measures rely on a human to perform a task. In any enterprise security architecture, these tasks will be interdependent. Figure 22 is an example of how security process supports configuration targets to maintain system security for the mobile system. Each block of the process diagram has three elements. The first is a label meant to be a user-friendly way for those discussing the diagram to refer to each step. In Figure 22, the labels in the first element have been selected to correspond generally to the domain of activity workflow, and there should be no expectation that the domains or numbers in the top third of each block indicate order of execution, and indeed may occur simultaneously as different incidents arise. The second element of each block identifies the organization that is responsible for executing the activity described in the third element.

Measures in rows 1, 4, and 8 of Figure 21 are only as reliable as the correct performance of task labeled SECURITY-1 in Figure 22. Measures in rows 2,3,5,6, and 7 of Figure 21 rely on the correct performance of task SECURITY-2 in Figure 22. The triad "people, process, and technology," often repeated in security management literature, refers to the technology design, security process, and investment in human capital that are components of the security architect's theory that the overall system has been secured. Where human components were required to maintain correct configuration, monitor metrics for these critical tasks would also be devised. For simplicity's sake in this example, we assume that these measures yield no exceptions, so no remediation metrics are required.

Validation of operational performance is likely to be similar for the two different architectures. Measures of conformance to specifications such as ease of install and speed of data response may show some deviation, but it is also assumed that the system design meets requirements, so trade-offs, such as those between CPU cycles and confidentiality requirements expected to be met with encryption, have previously been worked out.

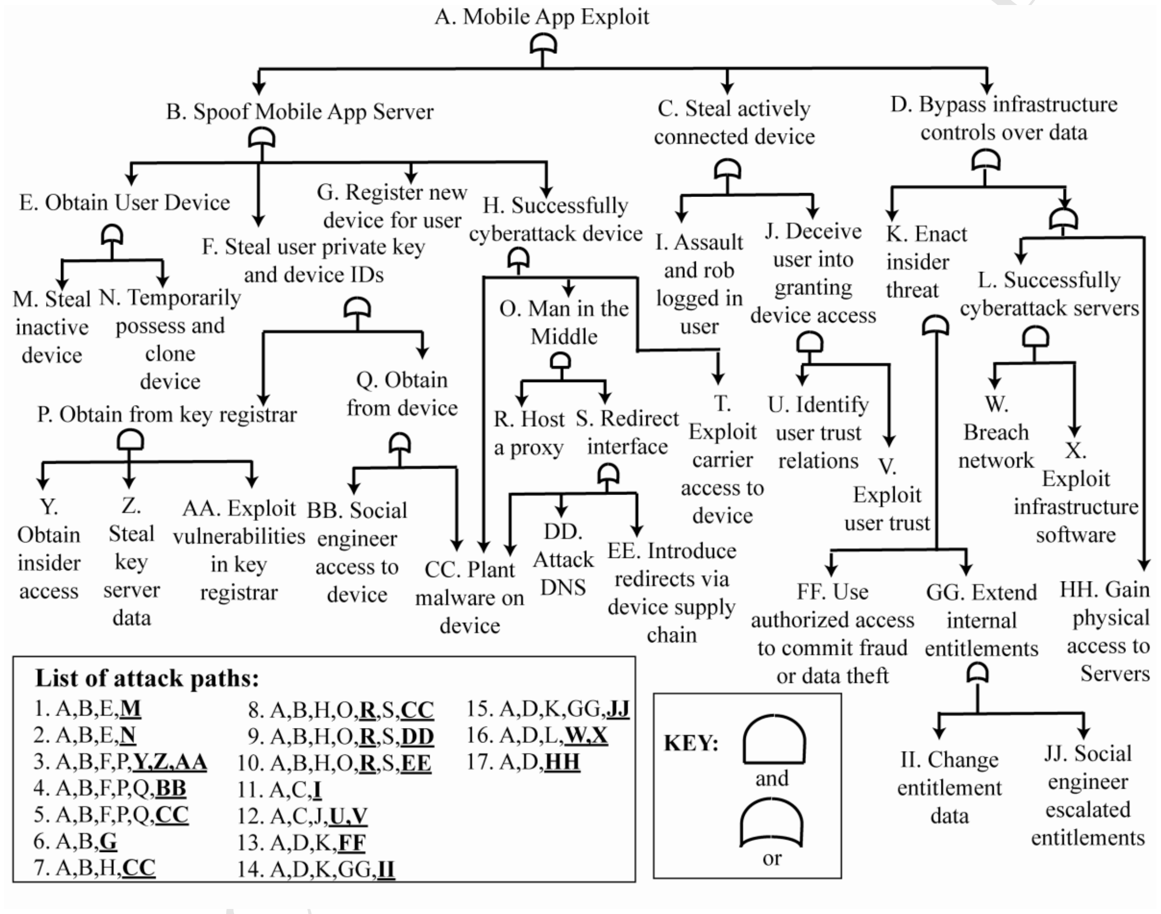
Figure 22. Mobile System Security Process


Prevention
Detection
Response
USER-1
USER
User security awareness
Incident Detected?
Yes
USER-2
USER
User device theft report
User malware report
USER-3
SECURITY
Incident tracking
Lookup device in Mobile System
Device in Mobile System?
Yes
SUPPORT-1
IT OPERATIONS
Physically secure software and application infrastructure
Configure change control for mobile system
SUPPORT-2
IT OPERATIONS
Monitor change detection for Mobile System configuration anomalies
Incident in Mobile System?
Yes
SECURITY-4
MANAGEMENT
Enforce policy through disciplinary action
SECURITY-1
QA
Conduct software security feature and penetration tests.
SECURITY-2
SECURITY
User behavior anomaly detection
Device switch detection
Monitor client accounts
Penetration test review
Incident Detected?
Yes
SECURITY-3
SECURITY
Incident tracking
Terminate active device session
Device transaction audit
Suspend application data flow
Require reinstall or reregister
Change one authentication factor
Incident Report
MOBILE-1
APPLICATION DEV
Software assurance for application software
Develop application security features
MOBILE-2
APPLICATION DEV
Change location or storage mode of data on device
Patch software
Process redesign and implementation
SDLC control remediation
Investigation Collaboration
MOBILE-3
IT OPERATIONS
Restore from known good image
Patch vulnerabilities
SDLC control remediation
Perform root cause analysis and corresponding remediation

Vulntest validation, by contrast, may be expected to be quite different. Vulntests may be expected to be based on attack trees that decompose goals for exploiting the mobile application into subgoals until the set of subgoals required to accomplish the main goals may be achieved by activity within the scope of adversary capabilities. Figure 23 is an example of an attack tree. It describes a general goal of “mobile application exploit,” that is labeled as goal “A,” and may be achieved if any one of three subgoals is achieved. The subgoals are thus joined by an “or” branch leading to “A.” They are labeled “B,” “C,” and “D.” In turn, each subgoal may be achieved by achieving the combination of subgoals below it. “And” and “Or” branches denote whether subgoals must be achieved in combination or present alternative methods of achieving the goal above them, respectively. A subgoal that has no subgoals is referred to as a “leaf” of the attack tree, and are intended to be well enough understood to form the basis for technical attack plans to achieve them. In Figure 23, each combination of leaf subgoals which will achieve the highest level goal “A” is identified by the path through the subgoals above it, and presented in bold. (For more thorough explanations and instructions on creating attack trees, see: Garcia 2008),

The attack tree in Figure 23 has been crafted to apply to both Mobile System Architecture A and Mobile System Architecture B. It shows that there are several attack paths that, if activities designed to achieve the subgoals identified in the (bold) leaves are executed successfully, will work on both. While some attacks present the same level of difficulty for an adversary in both architectures, a vulntest test designed to enact the threats at the leaves of System A would be different than one designed for System B. Results can be measured in terms of the known system vulnerabilities that were exploited, the number of leaves that were executed, the time to execute a successful attack, time that the test could continue without detection, or even using measures of the technical skill level required for attack. In the case of Path 6, Leaf G of Figure 23, for System A, the adversaries simply have to intercept an email. For System B, they have to gain access to the wireless Intranet. In the case of Paths 8 and 10, Leaves CC and EE (planting malware or tampering with the interface supply chain) could be made much more difficult to accomplish in System B than in System A. This is because malware or supply chains attacks would have to be created specifically for custom software in System B versus a mobile browser in System A.

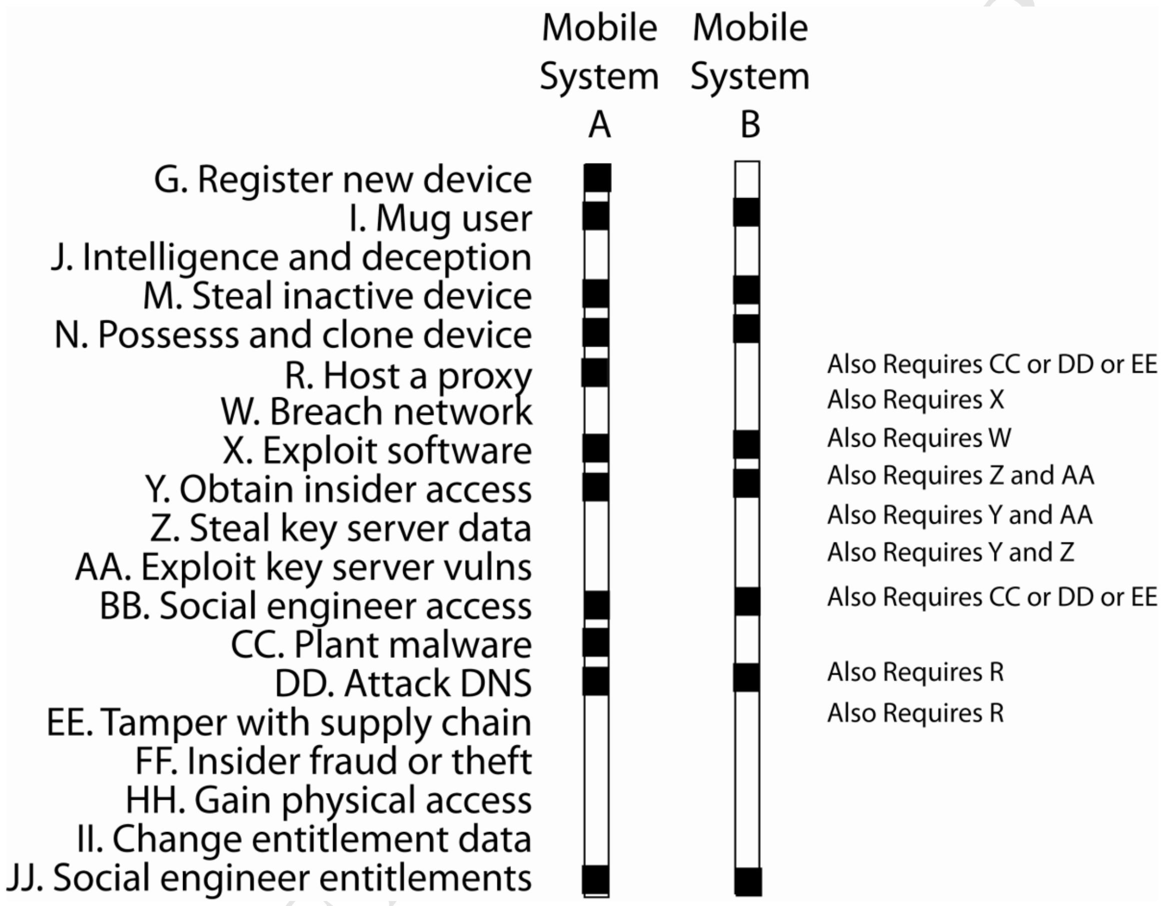
Figure 23. Mobile System Attack Tree


A. Mobile App Exploit
B. Spoof Mobile App Server
C. Steal actively connected device
D. Bypass infrastructure controls over data
E. Obtain User Device
F. Steal user private key and device IDs
G. Register new device for user
H. Successfully cyberattack device
I. Assault and rob logged in user
J. Deceive user into granting device access
K. Enact insider threat
L. Successfully cyberattack servers
M. Steal inactive device
N. Temporarily possess and clone device
O. Man in the Middle
P. Obtain from key registrar
Q. Obtain from device
R. Host a proxy
S. Redirect interface
T. Exploit carrier access to device
U. Identify user trust relations
V. Exploit user trust
W. Breach network
X. Exploit infrastructure software
Y. Obtain insider access
Z. Steal key server data
AA. Exploit vulnerabilities in key registrar
BB. Social engineer access to device
CC. Plant malware on device
DD. Attack DNS
EE. Introduce redirects via device supply chain
FF. Use authorized access to commit fraud or data theft
GG. Extend internal entitlements
HH. Gain physical access to Servers
II. Change entitlement data
JJ. Social engineer escalated entitlements
KEY:
and
or

List of attack paths:
1. A,B,E,**M**
2. A,B,E,**N**
3. A,B,F,P,**Y,Z,AA**
4. A,B,F,P,Q,**BB**
5. A,B,F,P,Q,**CC**
6. A,B,**G**
7. A,B,H,**CC**
8. A,B,H,O,**R**,S,**CC**
9. A,B,H,O,**R**,S,**DD**
10. A,B,H,O,**R**,S,**EE**
11. A,C,**I**
12. A,C,J,**U,V**
13. A,D,K,**FF**
14. A,D,K,GG,**II**
15. A,D,K,GG,**JJ**
16. A,D,L,**W,X**
17. A,D,**HH**

There is no requirement that the decomposed versions of the four high level STAC components be depicted in the same manner. Figure 24 shows the vulntest results as a simple checklist on whether or not the tests for the leaf activities of the attack tree were successful. It also includes information on whether that activity must be augmented with other leaves in order to achieve the attack goal. The result may also be a binary measure of defense in depth; that is, whether or not one vulnerability could allow the attack goal to be met. Figure 24 shows that Mobile System B scores better than Mobile System A because it is less vulnerable to known attacks.

Figure 24. Example Vulntest Metrics for Mobile System



Procedures for producing the fourth dimension of metrics required for STAC will be less straightforward because they measure response to events that are by definition unexpected. If resilience features work well, it should be possible to deliberately cause damaging system impact, and measure the ability of the system to recover. However, for fear that deliberate damage required to test response capability will cause stress rather than the peace of mind expected from successful test results, resilience metrics are often left out of assessment metrics. In these cases, construct stochastic or deterministic models are often used to supplement assessment decisions rather than use metrics that directly support them with measureable system attributes. An example metric to measure the resilience of the Mobile System is a Security Work Factor Ratio ("SWFR") (Arrott 2011). A SWFR is a ratio of two measurements, the time to protect (TTP) over the time to attack (TTA), defined as:

TTP:	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 the mobile system's response capability.
TTA:	the median lifetime of malicious activity emanating from a specific source. This measure may be taken from aggregated historical data on the number of months that zero-day attacks remained unreported (Bilge and Dumitras 2012).

To the extent the ratio TTP/TTA is minimized, the defenders may be sufficiently resilient in 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 zero.

To measure whether an adversary goal of mobile application exploit is thwarted, the TTP can be measured via timed recovery exercises where the leaves in given attack paths are declared to be accomplished. For example, attack path 8 in Figure 23 indicates that activities on leaves R and CC set the stage for the attack path to be utilized. If the attacker *hosts a proxy* to the mobile application site (leaf R) and *plants malware on the user device* (leaf CC) that directs the user to the proxy rather than the mobile application site, then the attacker may achieve the goal of mobile system exploit (node A) via the subgoals of *spoofing the mobile app server* (node B) and *successfully cyber-attacking the device* (node H) . To assign a TTP, the mobile support team may be presented with this scenario and asked to rearchitect the system in order to thwart it.

To assign a TTA, it may be possible to use historical data taken from similar exploits. If it is thought that the exploit would have to be especially innovative, it may be estimated using the average of paradigm-shifting exploits over the past decade.

If each known attack path can be assigned a SWFR based on the minimum TTP/TTA for attack recovery, then the resilience metric of the mobile system may be measured by the TTP against all so-far identified threats. Assume P_1 through P_n are the paths on a rigorously devised attack tree for Mobile System M, and P_1SWFR through P_nSWFR are the corresponding SWFR ratios that an attack would take on each path. The system-level M_{SWFR} is the longest of those minimum values, calculated as:

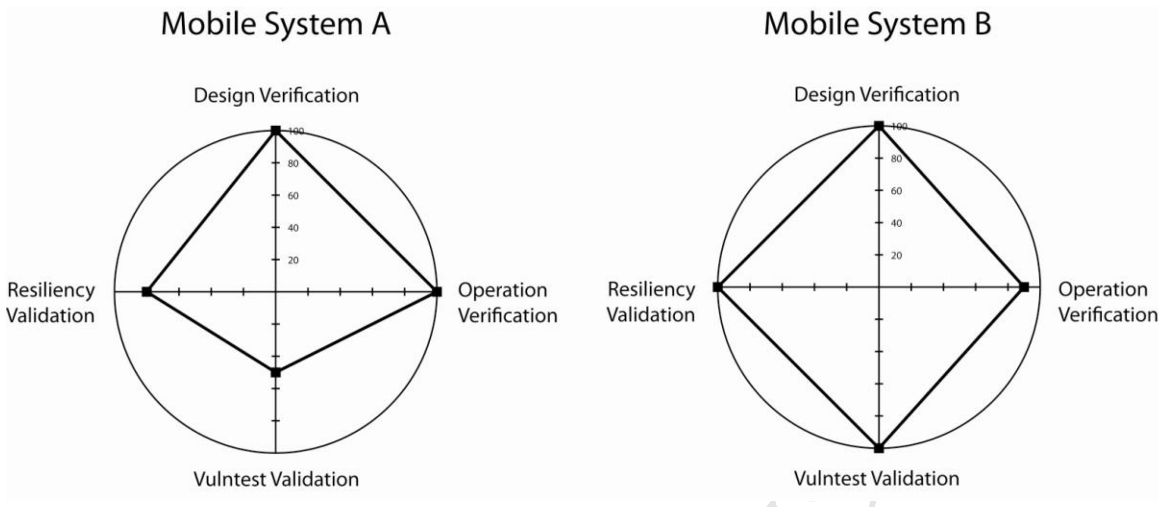
$$M_{SWFR} = \max\ (\ P_1SWFR \ldots P_nSWFR\ )$$

The aggregate resiliency metric for inclusion in a STAC presentation like Figure 12 might be the percentage of known attack paths for which the exercise results met a minimum time threshold. This measure differs from the vulnerability test in that the system support team is challenged with adapting the system operation to ensure that no successful leaf exploits will result in system damage, despite the fact that there is no known system vulnerability that would allow the exploit. It measures flexible response. The vulnerability test, by contrast, can only identify known vulnerabilities.

Trends will of course change continuously, so any security resilience validation metric based on it will have to be continuously monitored to ensure that the measures evolve in conjunction with changes in the threat environment. But in general, assuming equivalently thorough attack trees, the lower the M_{SWFR}, the stronger the security metric. Given two mobile environments with roughly equivalent threat surfaces, a system with a lower M_{SWFR} will be more resilient than one with a higher M_{SWFR}. For example, because Mobile System B registers devices only on the internal network rather than through the external interface used by Mobile System A, the system support team may find it easier and faster to change a breached device registration process, and so have a lower TTP for leaf G, which could bring down its overall M_{SWFR} compared to Mobile System A.

For two systems with the same mission and purpose, the performance, the vulntest and the resilience requirements may be expected to be similar enough such that the best metric score in each of these three areas would become the 100% mark for the purposed of STAC. Given that the Mobile System A’s performance is unconstrained by the internal network registration requirement, and that very feature makes it vulnerable to known vulnerability tests, and that B’s support team gains a resiliency advantage from this feature differential, the STAC metrics of Mobile System A versus B comparison would look like Figure 25. Where a system is measured in isolation, the performance, the vulntest and the resilience requirements may instead be set by stakeholder expectations.

Figure 25. Mobile System Metric Comparison



5 Discussion

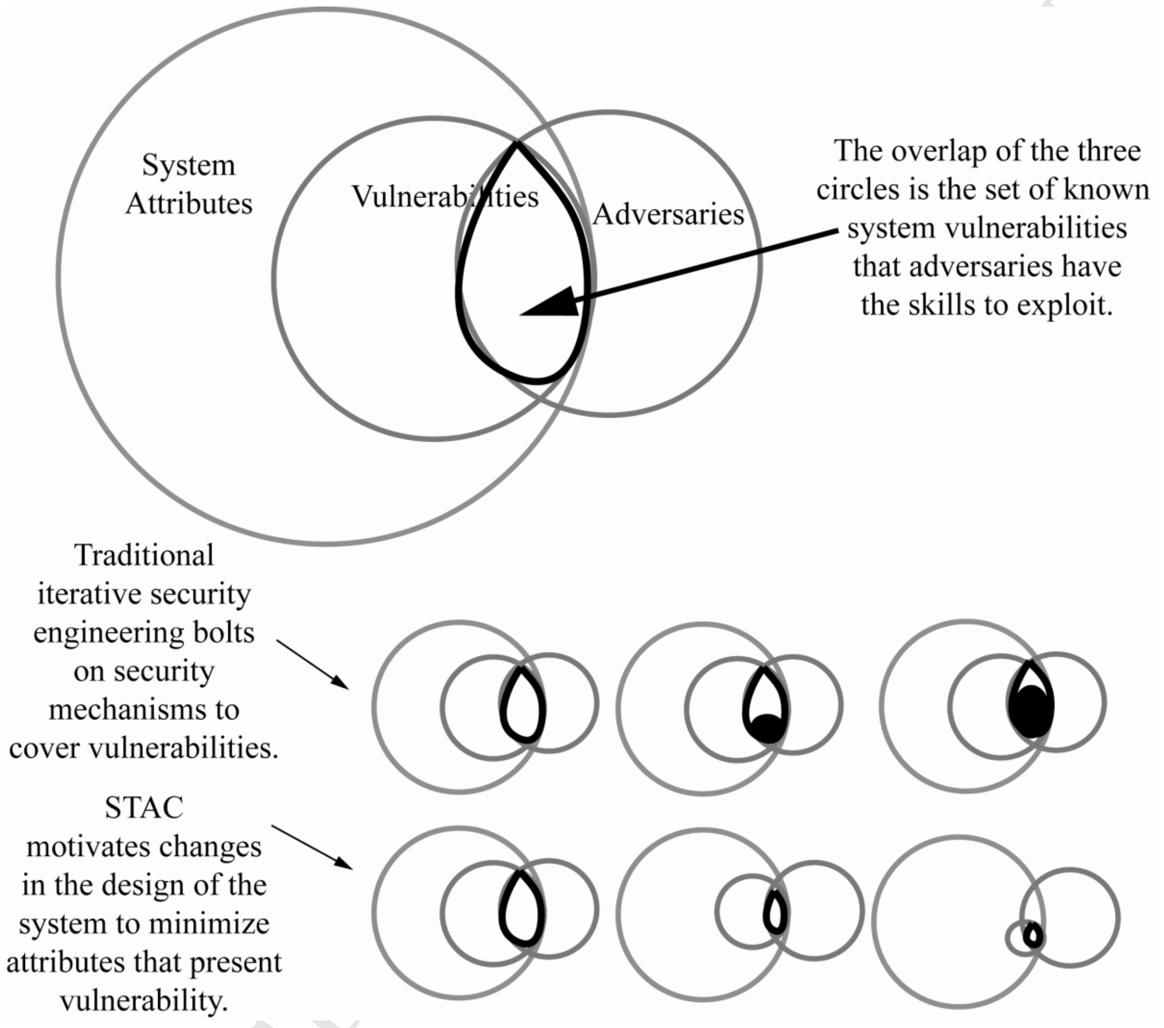
In *Security Metrics*, Jaquith described a tension between modelers and measurers that often appears in the securitymetrics.org mail list (Jaquith 2007). The modelers aim to model security risk by abstracting away unknowns. The measurers, who sometime humbly refer to themselves as muddlers, mine an endless supply of security-related data in search of clues with which to make progress. The modeler versus measurer debate should end. The answer is that we need to do both. Security metrics at the system level must include both verification and validation measures. The only way to do that is to construct both a theory of what it means to be secure and methods to measure whether a given hypothesis related to that theory is false.

The history of computer security is a search for security mechanisms that will minimize, if not eliminate, a newly discovered vulnerability in any system. Using STAC, the historical approach of adding performance-hogging tools to constrain system behavior is not an acceptable option. STAC instead requires that security be validated with respect to system mission and purpose, and performance-hogging tools reduce the operation validation component of the STAC.

Computer security history also demonstrates that target security metrics often are permitted to dip well below the 100% level. Using STAC, running at a 95% verification level and accepting the risk that some unverified design component creates a weak link is simply inadequate security management. If your theory is that you don’t need to verify your design in order to measure security, then you cannot also claim that the implementation of your secure design provides a measureable security attribute. That is, if you have planned a security feature, but find that you don't need to measure it to demonstrate system correspondence to your theory of security, then you did not need to include the security feature in your theory or design. Security metrics need to include verification only for the mechanisms you rely upon to secure your system mission.

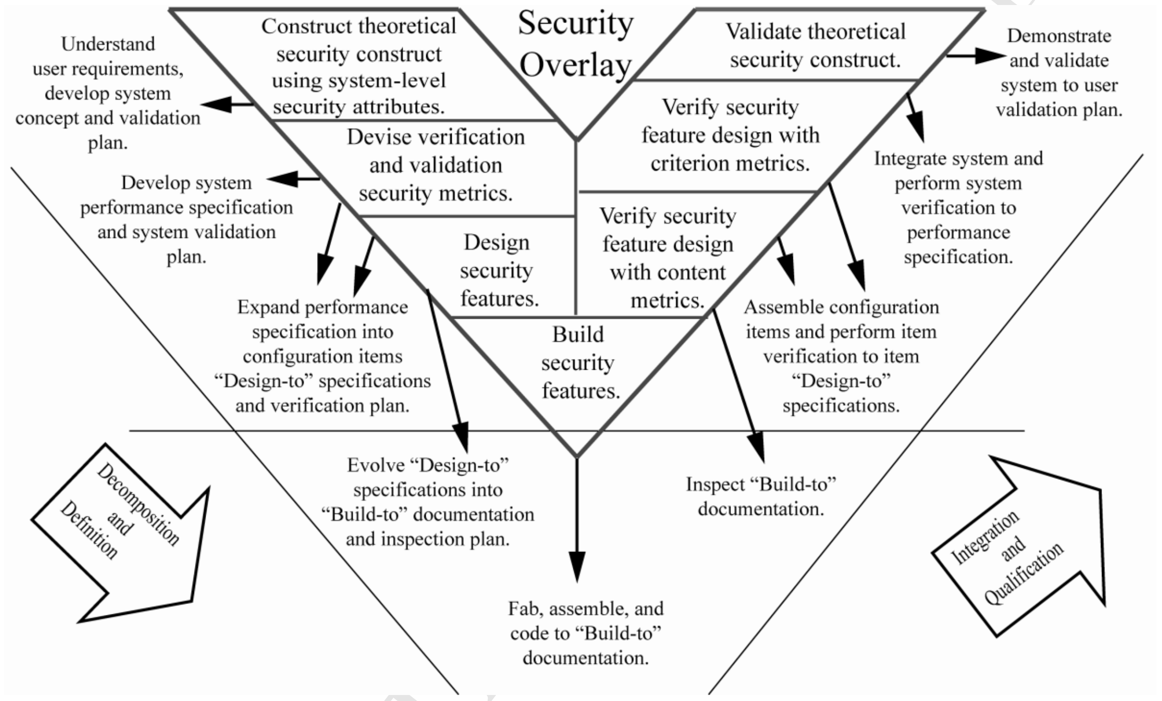
The requirement for relying on the design to produce measurable security attributes changes the nature of the problem space in systems security engineering. It becomes an issue of how to change the way the system functions under threat as opposed to the search for the next security fix. In the Mobile System example, adding a hand-held authentication factor to System A would not be an acceptable design change because having to know the whereabouts of a token constrains the system purpose of mobile communication. That would be a historical bolt-on approach. A design change in line with STAC would be to change the software architecture on the mobile device to eliminate dependency on the local browser. This design change could have a direct impact on the vulntest metrics in Figure 24 without a detrimental effect any operation verification metrics. Figure 26 illustrates the difference in mindset between the bolt-on approach to security engineering and the redesign approach.

Figure 26. Graphical Illustration of Change in Vulnerable Attributes


System Attributes
Vulnerabilities
Adversaries
The overlap of the three circles is the set of known system vulnerabilities that adversaries have the skills to exploit.
Traditional iterative security engineering bolts on security mechanisms to cover vulnerabilities.
STAC motivates changes in the design of the system to minimize attributes that present vulnerability.

STAC is not a risk metric. It measures security, not risk. Risk analysis should be performed during system design and specification, and should support construct decisions. Functionality trade-spaces and cost-benefit analysis should be part of design process, not operations process. The corresponding STAC theory and associated metrics should also be created at a very early stage of the system engineering process, and facilitate both construct and assessment decisions. Figure 27 illustrates how STAC should be integrated into the systems engineering process as described in a popular textbook (Buede 2009). Once the method for ensuring system security has been set, the metrics should be established, and any implementation that cannot achieve a 100% metric simply does not meet the design goals, that is, it fails validation.

Figure 27. STAC Overlaid onto Textbook System Engineering Lifecycle


Security Overlay
Construct theoretical security construct using system-level security attributes.
Devise verification and validation security metrics.
Design security features.
Build security features.
Verify security feature design with content metrics.
Verify security feature design with criterion metrics.
Validate theoretical security construct.
Understand user requirements, develop system concept and validation plan.
Develop system performance specification and system validation plan.
Expand performance specification into configuration items "Design-to" specifications and verification plan.
Evolve "Design-to" specifications into "Build-to" documentation and inspection plan.
Fab, assemble, and code to "Build-to" documentation.
Inspect "Build-to" documentation.
Assemble configuration items and perform item verification to item "Design-to" specifications.
Integrate system and perform system verification to performance specification.
Demonstrate and validate system to user validation plan.
Decomposition and Definition
Integration and Qualification

The recommendation that security as a theoretical attribute be constructed in the initial stages of a system lifecycle is not meant to imply that STAC cannot be applied to a system in operation. Instead, it means that, as a prerequisite to creating security metrics, the current mission and purpose of the system must be well understood, and the system security features must be evaluated on the basis of how well they fit that purpose. It also implies that, if a system design has been decided to be inadequate to achieve the attribute of security, then decisions need to be made on changes in either technology or operational process. The design change decisions should be based on security risk analysis using construct metrics, and the changes should be immediately and correctly implemented. One does not achieve security goals incrementally as a project plan moves components asymptotically toward a desired system configuration. The need for remediation metrics indicates failed verification. The current widely adopted approach of falling back on risk acceptance at the component level is not a substitute for sound security management.

6 Conclusions

"Building security in" is not just a software development issue. Given the escalating infrastructure dependence on software, it is a fundamental requirement for systems of all shapes and sizes: planes, power grids, and water treatment systems as well as cyber networks. Moreover, operational process and physical security have always been essential to systems security maintenance, and will become increasingly more important as technology evolves. Security professionals cannot in good conscience continue to present correct technology configuration measures in lieu of effective security metrics.

In order to measure security, one must first understand what it means for the system to be secure. The understanding should be articulated in the form of a theory. The theoretical attribute of security should be supported with a clearly defined configuration, and also performance specifications, expected threat environment, and clear criteria for mission success. Adherence to STAC means that security is measured using carefully selected metrics from each of these areas.

The contribution of this paper to existing literature in security is threefold: security metrics is a field distinct from security assessment, security metrics must be customized to consider system purpose, and security cannot be measured using verification techniques alone. Although none of the three statements should be particularly controversial, the presentation of these ideas in combination with a taxonomy of existing security metrics opens the door for a new thread in security research. The STAC framework equips a security researcher (or a systems engineer) to construct a theory of security for a given system of interest that can be also tested for *validity*. Because of the STAC focus on system security validation, STAC metrics are also comprehensible to executive decision-makers faced with trade-off decisions that affect system security. That is, where the STAC framework is correctly applied, resulting theories of system security are both construct and face valid. This research thereby provides a new paradigm for system security engineering. Using STAC metrics will drive the industry toward behavior that is necessary for security technologies to strengthen, rather than constrain, system operation.

References

ACM (2009). International Workshop on Security Measurements and Metrics. Metrisec was last hosted at http://metrisec2012.cs.nku.edu/history.html, last visited 11/1/12, Association for Computing Machinery.

Amoroso, E. (2010). Cyber Attacks, Elsevier, p159-161.

Amran, A. R., R. Phan, et al. (2009). Metrics for Network Forensics Conviction Evidence. International Conference for Internet Technology and Secured Transactions

Arrott, A. (2011). Work Factor Ratios, securitymetrics.org, post of June 9, 2011, last accessed 11/1/12.

Axelrod, C. W. (2012). Engineering Safe and Secure Software Systems, Artech House.

Baker, W., A. Hutton, et al. (2011). Data Breach Investigations Report, http://www.verizonbusiness.com/go/2011dbir. Verizon Business (Retrieved 11/1/12).

Bayuk, J. (2001). "Security Metrics." Computer Security Journal **XVII**(1).

Bayuk, J. (2010). The Utility of Security Standards. International Carnahan Conference on Security Technology (ICCST), IEEE.

Bayuk, J., J. Healy, et al. (2012). Cyber Security Policy Guidebook. Hoboken, NJ, Wiley.

Bayuk, J., B. Horowitz, et al. (2012). Security Via Related Disciplines. Conference on Systems Engineering Research (CSER) (errata corrected at http://www.bayuk.com/publications/Bayuk-CSER.pdf, Retrieved 11/1/12)

Bayuk, J. and A. Mostashari (2012). "Measuring Systems Security." Systems Engineering **16**(1-DOI: 10.1002/sys.21211).

Bayuk, J. and A. Mostashari (2012). "Measuring systems security." Systems Engineering **16**(1-10.1002/sys.21211).

Bayuk, J. L. (2011). "Security Subject Matter Expert Survey on Security Metrics." Retrieved 11/1/12, from http://www.bayuk.com/thesis.

Bilge, L. and T. Dumitras (2012). Before We Knew It: An Empirical Study of Zero-Day Attacks In The Real World. Conference on Computer and Communications Security. Raleigh, North Carolina ACM.

Buede, D. M. (2009). The Engineering Design of Systems, Models and Methods, Wiley.

Carmines, E. and R. Zeller (1979). Reliability and Validity Assessment. Thousand Oaks, California, SAGE Publications.

Cavusoglu, H., S. Raghunathan, et al. (2008). "Decision-Theoretic and Game-Theoretic Approaches to IT Security Investment." Journal of Management Information Systems **25**(2): 281–304.

CCRA (2012). Common Criteria for Information Technology Security Evaluation Version 4, Common Criteria Recognition Agreement, http://www.commoncriteriaportal.org.

Checkland, P. (1999). Systems Thinking, Systems Practice, John Wiley & Sons.

CISWG (2005). Report of the Best Practices and Metrics Teams. Corporate Information Security Working Group, US House of Representatives, Subcommittee on Technology, Information Policy, Intergovernmental Relations and the Census, Government Reform Committee.

Cummings, A., T. Lewellen, et al. (2012). Insider Threat Study: Illicit Cyber Activity Involving Fraud in the U.S. Financial Services Sector, Carnegie Mellon University.

DoD (1985). The Orange Book, Trusted Computer System Evaluation Criteria, Department of Defense. **(supercedes first version of 1983)**.

Eskins, D. and W. H. Sanders (2011). The Multiple-Asymmetric-Utility System Model: A Framework for Modeling Cyber-Human Systems. Proceedings of the 8th International Conference on Quantitative Evaluation of SysTems**:** 233-242.

Espenschied, J. and A. Gunn (2012). Threat Genomics. Metricon 7.0. Bellevue, WA, securitymetrics.org (Retrieved 11/1/12).

Fenton, N. E. and S. L. Pfleeger (1997). Software Metrics, a Rigorous and Practical Approach, Second Edition. Boston, MA, PWS Publishing Company.

Forums (ongoing). www.securitymetrics.org, metrisec2012.cs.nku.edu/history.html, samate.nist.gov, www.societyinforisk.org, www.veriscommunity.net, www.cisecurity.org, weis2013.econinfosec.org (all last visited 11/1/12).

FS-ISAC (2011). On-line Fraud Detection White Paper, Account Takeover Task Force, Financial Services Information Sharing and Analysis Center.

Garcia, M. L. (2008). The Design and Analysis of Physical Protection Systems. Amsterdam, Butterworth-Heinemann.

Geer, D. E. and D. G. Conway (2009). "The Owned Price Index." IEEE Security & Privacy **7**(1): 86-87.

Hancock, B. (2000). "US Government Board Setting Up Security Metrics." Computers & Security **19**(7): 580.

Herrmann, D. (2007). The Complete Guide to Security and Privacy Metrics. Boca Raton, FL, Auerbach Publications.

Howard, M. (2002). Writing Secure Code, 2nd Edition, Microsoft Press.

Hubbard, D. W. (2009). The Failure of Risk Management. Hoboken, NJ, John Wiley & Sons.

IDART (2008). "Information Design Assurance Red Team, Red Team Metrics " Sandia National Labortories, http://www.idart.sandia.gov/training/Metrics.html, last accessed 11/1/2012.

INCOSE (2011). INCOSE Systems Engineering Handbook, Version 3.2.1.

India. "Blind men and an elephant, From Wikipedia, the free encyclopedia." Retrieved 11/1/12, from http://en.wikipedia.org/wiki/Blind_men_and_an_elephant.

Ioannidis, C., D. Pym, et al. (2009). Investments and Trade-offs in the Economics of Information Security. Financial Cryptography and Data Security. R. Dingledine and P. Golle, Springer Berlin / Heidelberg. **5628:** 148-166.

ISACA (2007). Control Objectives for Information Technology (COBIT). Rolling Meadows, IL, Information Systems Audit and Control Association, IT Governance Institute.

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

Izrailevsky, Y. and A. Tseitlin. (ongoing, July 19, 2011). "The Netflix Simian Army, http://techblog.netflix.com/2011/07/netflix-simian-army.html, http://techblog.netflix.com/2012/07/chaos-monkey-released-into-wild.html." The

Netflix Tech Blog Retrieved 11/1/12, from http://techblog.netflix.com/2011/07/netflix-simian-army.html.

Jansen, W. (2009). Directions in Security Metrics Research, National Institute of Standards and Technology Interagency Report.

Jaquith, A. (2007). Security Metrics. Upper Saddle River, NJ, Pearson Education, Modeler versus Measurer debate p13-15.

Jones, R. A. and B. M. Horowitz (2012). "A System-Aware Cyber Security Architecture." Systems Engineering **15**(2).

Kovacich, G. (1997). "Information systems security metrics management." Computers & Security **16**(7): 610-618.

LeMay, E., M. D. Ford, et al. (2011). Model-based Security Metrics using ADversary VIew Security Evaluation. Proceedings of the 8th International Conference on Quantitative Evaluation of SysTems. Aachen, Germany**:** 191-200.

McGraw, G. (2006). Software Security. Upper Saddle River, NJ, Addison-Wesley.

Mell, P., K. Scarfone, et al. (2007). A Complete Guide to the Common Vulnerability Scoring System Version 2.0, Forum of Incident Response and Security Teams (FIRST).

Metricon (ongoing). Metricon is hosted at www.securitymetrics.org, a community website for security practitioners, last accessed 11/1/12.

Metricons. (ongoing). "Metricon 7.0, "What we know to work in security metrics," Panel led by Anton Chuvakin (2011), Metricon 6.5, "Data Mining Methods for Enterprise Level Security," Panel led by Scott Crawford (2010), Metricon 5.0, Seiersen, Richard, "Practical Security Metrics in the 4th Dimension" (2009), Metricon 4.5, Krikken, Ramon, "Security Metrics Field Research" (2008), Metricon 3.0, Wong, Caroline, "Metrics at Ebay," (2008)."   Retrieved 11/1/12, from www.securitymetrics.com

MITRE (ongoing). National Vulnerability Database, US National Institute of Standards and Technology, http://nvd.nist.gov/ last accessed 11/1/12.

Mogull, R. (2009). An Open Letter to Robert Carr, CEO of Heartland Payment Systems. Securosis Blog, Securosis. **2009**.

NIST (1995). An Introduction to Computer Security: The NIST Handbook. National Institute of Standards and Technology.

NIST (2007). Recommended Security Controls for Federal Information Systems, SP 800-53 Rev 2. National Institute of Standards and Technology. (authors: Ross, R. et.al.).

NIST (2010). The Technical Specification for the Security Content Automation Protocol (SCAP): SCAP Version 1.1, SP800-126 Rev.1, scap.nist.gov (last accessed 11/1/12). National Institute of Standards and Technology (US), Information Technology Laboratory (authors: Quinn, S, et.al.).

NIST (2011). Managing Information Security Risk, SP-800-39, National Institute of Standards and Technology, Joint Task Force Transformation Initiative Interagency Working Group,.

Pande, P., R. Neuman, et al. (2001). The Six Sigma Way, McGraw-Hill.

Parker, D. (2011) "Re: System Security Metric Survey Follow-up Request." Email Correspondance, September 1, 2011.

PCI (2008). Payment Card Industry (PCI) Data Security Standard, Version 1.2, Payment Card Industry (PCI) Security Standards Council.

Reichheld, F. F. (2003). The One Number You Need to Grow. Harvard Business Review.

SANS Institute (ongoing). "Security Consensus Operational Readiness Evaluation, http://www.sans.org/score/." e.g. UNIX Security Checklist, http://www.sans.org/score/checklists/AuditingUnix.pdf.

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

Thurstone, L. L. (1928). "Attitudes can be measured." American Journal of Sociology **33**(4): 529-554.

Verendel, V. (2008). A Prospect Theory Approach to Security, Chalmers University of Technology, Goteborg University.

Wang, H., S. Roy, et al. (2010). A Framework for Security Quantification of Networked Machines. 2nd International Conference on COMmunication Systems and NETworks, (COMSNETS) **1:** 5-9.

Wilhelm, T. (2009). Professional Penetration Testing, Syngress.

***Biographical Sketch**

Bayuk is an independent systems security consultant with professional experience in a wide variety of cyber security management, architecture, academic, and advisory roles. Her numerous publications and presentations on information security topics include 4 textbooks and 2 edited compilations. She has Masters degrees in Computer Science and Philosophy, a PhD in Systems Security Engineering, Certifications in Information Systems Audit, Information Systems Security, Information Security Management, and IT Governance (CISA, CISSP, CISM, CGEIT), and a NJ State Private Investigator's License.