TOWARDS A FRAMEWORK FOR SECURITY MEASUREMENT

Chenxi Wang, William A. Wulf
Department of Computer Science
University of Virginia
cw2e@virginia.edu, wulf@virginia.edu

1. Introduction

We are living in an era when computer technology constantly changes our lives. While placing unprecedented reliance on computers and digital systems, we continue to have a very poor grasp of the security aspects of the technology. As security plays an increasingly important role in many systems, it is essential that we have a better understanding and management of computer security.

This work aims to devise a framework for measuring system security. We can all agree that the ability to compare, to contrast, and to make quantifiable statements about system security is extremely valuable. It means that we will have a basis to determine where to put our limited resources, where to pay attention, and how to best secure our systems. We also believe that one can obtain a more complete and thorough understanding of a subject through measurements that may not otherwise be possible. "When you can measure what you are speaking about and express it in numbers you know something about it," wrote Lord Kelvin in 1883. "But when you cannot measure it, when you cannot express it in numbers, your knowledge is of a meager and unsatisfactory kind." This is exactly what motivated our research -- to develop the theory and practice of security measurements.

Quantifying security, however, is a difficult problem. The difficulties lie in that "computer security" is not a crisply defined term. We tend to know approximately what we mean by "security" and what we want it to do, but we seldom clearly state what security really means to us and how secure is "secure enough". Moreover, as computing systems grow larger and more complex, it is increasingly more difficult to make statements about any system-wide properties, even those better understood than security. The situation is worse for systems built from commercial off-the-shelf products and pre-existing components.

Despite these problems, we believe it is a worthy effort to explore ways to measure attributes that are of interest to us. In this paper, we propose a Security Measurement (SM) framework to overcome *some* of the problems. This framework should help us to:

- define the term of "computer security"
- define a measure that is acceptable to the definition of computer security
- define a methodology to make useful if not rigorous estimates of the measures
- validate the measures

This paper contains the first sketch of the framework. We realize that we are not going to solve the problem of quantifying security by proposing a universal measure. What we hope to achieve through this work is a systematic way which allows us to best approximate the security strength of a system or a family of closely related ones.

The rest of the paper is structured as follows: Section 2 presents each element in the SM framework and explains how it could be used to develop security measurements. Section 3 describes future research direction, and section 4 summarizes the paper.

2. The SM framework

In this section, we present the elements and the structure of the SM framework. Within this framework, one can define his or her own notion of an adequate security measure and evaluate the values of such measurements.

The SM framework is divided into the following elements:
1) Definition of computer security
2) Selection of units and scales
3) Definition of an estimation methodology
4) Validation of the measures

2.1 Definition of "Computer Security"

Different people will have different interpretations of what "computer security" means. For example, what are considered essential security issues for an academic environment such as the University of Virginia will certainly be different from the security issues of the CIA or that of a medical database system. This observation suggests that it is not sufficient to have a single definition of security.

Computer security is also a multi-dimensional attribute, and its many dimensions are not necessarily commensurate

properties. For example, a financial stock exchange network may define their security to be real-time availability and information privacy while an on-line newspaper will be primarily interested in the integrity of their information. In measuring such a multi-dimensional attribute, the many facets of the attribute must all be identified and adequately addressed.

Definition of security thus will be system dependent; it must identify a set of security-related attributes that are important to the use of the system. It must also decide whether the system security is to be represented as a vector or a single value. If a single value is desired, a model to relate the different attributes must also be defined. For example, when measuring standard of living, one may wish to consider the average salary level, the real estate prices, and the cost of everyday necessities, etc. In some cases, a simple addition of the various ratings can render a sufficient measure while others may require a more sophisticated model such as *weighted sum* to calculate the final measure.

In our framework, a security measure is represented as an *n* tuple of real numbers, each representing an aspect of the defined security. For example, if the system security is defined as the combination of confidentiality, integrity and availability, a possible security metric is the three tuple:

$$< f_1(\mathit{confidentiality}),\ f_2(\mathrm{int}\,\mathit{egrity}),\ f_3(\mathit{availability}) >$$

The values in this three-tuple indicate the measured strength of the confidentiality, integrity and availability of the system.

In the case that a single measure is desired, a model to derive the final measure from the measures of the different aspects must be defined. For example, the tuple

$$< f_1(\mathit{confidentiality}),\ f_2(\mathrm{int}\,\mathit{egrity}),\ f_3(\mathit{availability}),$$
$$g(f_1, f_2, f_3) >$$

where $g(f_1, f_2, f_3) = 0.65 f_1 + 0.25 f_2 + 0.1 f_3$

defines a measure whose value is dependent 65% on system confidentiality, 25% on system integrity, and 10% on system availability.

A good measure starts with knowing what to measure. Selecting the relevant security properties is an important first step. One of our ongoing efforts is to develop a set of guidelines to help researchers and practitioners understand and identify security-related concerns and translate them into specific security properties that will be measured later.

2.2 Selection of Units and Scales

An attribute can be measured in many different units and scale types. For example, length can be measured in feet and inches as well as meters and centimeters. Temperatures can be measured in interval scales [1] such as Celsius and Fahrenheit, or in a ratio scale [1] such as Kelvin. Units and scale types determine how we measure things as well as how we interpret the measured values.

There are many different types of measurement scales. Commonly used ones include nominal, ordinal, interval and ratio scales. We are primarily interested in the latter three because nominal scales do not establish orders hence are less useful for our purposes.

Ordinal scale preserves the ordering among classes or categories. The measured values are unique only up to orders. Addition, subtraction, and other arithmetic operations on the numerical values have no meaning. An example of ordinal scale is the Mohs scale of hardness for minerals [1].

Interval scales preserve not only the ordering but also the differences between classes. That is, we can compare the differences between any two of the ordered classes in the range of the mapping. Addition and subtraction are acceptable, but not computing of ratios. The commonly used Celsius and Fahrenheit scales are examples of interval scales.

Ratio scales preserve ordering, differences and ratios among classes. The measurement mappings must start at an absolute zero and increase at equal intervals, known as units. All arithmetic operations are meaningful in a ratio scale. The Kelvin scale for temperature is a ratio scale [1].

Units traditionally only apply to interval and ratio scales. Researchers have expanded the meaning of units as descriptions of categories for nominal and ordinal scales [2]. In the rest of the paper, we will use the broader definition of units.

Units and scale types determine how measurements can be achieved. For example, if we define the unit for availability as “requests served per hour”, we effectively stated that the measure of availability is to be derived from dividing the number of requests by a specified time frame.

In choosing appropriate units and scale types, the following issues must be considered:

- Plausibility: The richer the scale type, the more information the measures represent. However, sometimes it is simply not possible to use a rich scale. For example, if sufficient information or measurement tools are not available, a less ambitious scale may be more appropriate.

- Accuracy: Accuracy is an important criterion in selecting a unit and a scale type. Sometimes a good reason to choose one unit or scale type over others is the potential measurement errors caused by one measure as opposed to others.

2.3 The estimation methodology

When we measure things, sometimes a direct measure is not possible in which case a measuring instrument or an estimation method should be used. For example, scientists use the speed of light to measure distances between stars. Whether it is as simple as applying one's common sense to estimate a person's height, or as complex as measuring distances between stars, the key is to select an appropriate estimation method that best approximates the real value.

In the case of computer security, direct measurements of the end-to-end security properties are made impossible because of the scopes and structures of the modern computing systems. In these systems, security attributes are no longer functions of a single entity. They are more often functions of a host of objects and their interactions. To best approximate the security strength of large systems, an estimation method must be used.

There may be many estimation methods. For example, one can estimate system reliability by sampling the history of the entire system or by doing so on each component and integrate them in some manner. The following subsection describes a sample estimation methodology for information systems.

2.3.1 A Sample Estimation Methodology

In large systems, it is difficult to estimate system security. On the other hand, analyzing small, standalone components of the system is often an easier task. Assume that there is a way to estimate the security attributes of the individual components in a system. What we need is a model to relate the high-level security attributes to that of the low-level, more measurable components.

This methodology uses a decomposition method to develop such models -- starting with high-level security properties of the system, work our way down to the basic components of the system and their interactions.

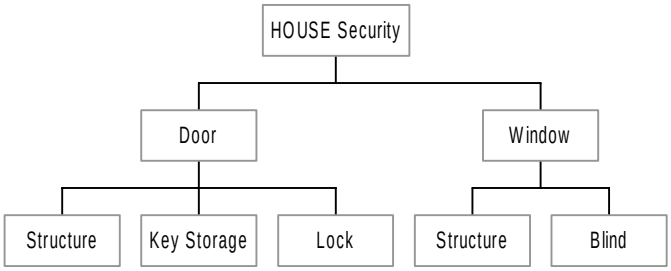
1. Decomposition

To see how it works, imagine we are analyzing the security of a house and how its various constructs work together to provide adequate services. The goal of the house is to provide privacy and protect from unauthorized break-ins. Starting from the goal, we will build a functional breakdown of the house system, always asking the question: "what needs to happen in order for the current goal not to fail?".

To simplify the problem, we assume that the house itself is perfectly constructed and is able to withstand arbitrary attacks. Furthermore, the house only has one window and one door.

Therefore, the main factors concerning the integrity and privacy of the house are the door and the window. We then repeat the decomposition process on the door and the window -- breaking them down to their functional contributions and, in this manner, a decomposition of the house system is created (see Figure 1).

Figure 1: House Security breakdown



It should be noted that there are many ways to decompose a particular system. As long as they are faithful interpretations of the original system, one isn't necessarily better than the others.

The decomposition process can be captured in the following steps:

1. Identify a set of security-related goal(s) for the system as the subject of the analysis.
2. Identify successive components that contribute to the success of the goal. These are the functions which have to succeed in order for the objective(s) not to fail.
3. Examine the subordinate nodes to see if further decomposition is needed. If so, repeat the process with the subordinate nodes as current goals, breaking them down to their functional components.
4. Terminate the decomposition process when none of the leaf nodes can be decomposed any further, or further analysis of these components is no longer necessary. In theory, when the decomposition terminates, all leaf nodes should be measurable components that are independent of each other.

Such a breakdown depicts the functional dependencies among the various system components. A tree structure is used to conveniently document the dependency relations. Note that in such a breakdown, a component can be either a

physical subsystem or a logical function which consists of a set of security properties.

2. Functional Relationships

Decomposing the system functions into their contributing factors is only the first step in constructing a complete model for the system. The many contributing factors must interact with each other to provide adequate functionality. A more important step is to analyze the relationships among the interacting factors and their composite effects.

The kinds of interaction among system components are far more complex than the classical logical relationships such as those used in fault tree analysis [5]. To permit a methodical building of representational models, we must be able to, at least conceptually, categorize these relationships using a set of well-defined rules.

The following list describes a set of logical relations among system components and the composite rules associated with them.

Weakest Link (WL): WL signifies that the functioning of the parent is ultimately bounded by the weakest of its children. "Weakest", in this context, refers to the measured security strength. In particular, a failure of any child node in a WL relation will cause the failure of the goal function. Mathematically, WL can be described as:

$$S(parent) = \min\left(S(child_1), S(child_2), \ldots, S(child_n)\right),$$

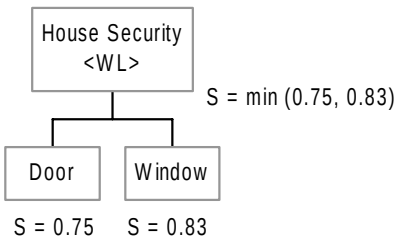
where S represents the assessment scores of the nodes and n is the number of children nodes.

Example 1: Consider the house example. The security of the house depends on two factors: the door and the window. It is easily seen that compromise of either one of the two factors may result in unauthorized break-ins and the consequences are equally detrimental to the house. Hence a WL relation exists between the door and the window (see Figure 2).

Supposing the assessment scores (either through direct measurement or calculation) for the door and the window are respectively 0.75 and 0.83. The score of the house is calculated as the following:

$$S(House) = \min(0.75, 0.83) = 0.75$$

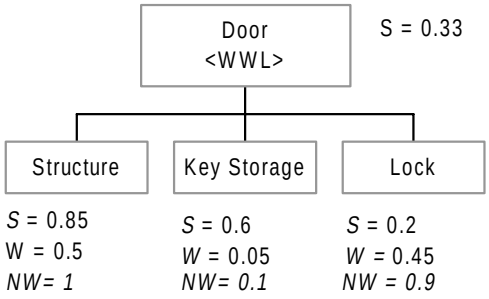
Figure 2: House security in relation to its factors



Weighted Weakest Link (WWL): WWL is a generalization of WL. While WL does not differentiate between trivial and important factors, WWL takes into account that different children nodes can have various degrees of impact on the parent node.

Example 2: Consider the functioning of the door in the house example, it is decided that three factors are of importance: the structure of the door, the key storage and the lock. However, security provided by the door maybe more heavily influenced by a subset of the factors than the rest. For instance, depending on what kind of neighborhood the house is located, it might be worthwhile to pay more attention to the sturdiness of the door and strength of the lock rather than the storage of the key. Hence, a WWL relation will appear to be more appropriate (see Figure 3).

Figure 3: Functioning of the door in relation to its factors



Mathematically, the output of a WWL relation is calculated following these steps:

1. Assumptions: each child node has an assessment score S and a weight W where S is a number between 0 and 1 and the sum of weights among all siblings is 1. In Figure 3, the three leaf nodes are weighted 0.5, 0.05, and 0.45. Their respective assessment scores are 0.85, 0.6 and 0.90.
2. Normalize the weights against the highest weight. The Normalized Weights (*NW*s) for the three leaf nodes in Figure 3 are 1, 0.1 and 0.9.
3. Select the weighted weakest child as $\min\left(\frac{S1}{NW1}, \frac{S2}{NW2}, \ldots, \frac{Sn}{NWn}\right)$, where n is the number of children nodes. In Figure 3, the weighted weakest child is the lock whose $\frac{S}{NW} = 0.222$.
4. Compute the weighted sum of the children nodes $WeightedSum = \sum_{i=1}^{n} (S_i \times W_i)$, where n is the number of children nodes.

Continue with the example in Figure 3, the weighted sum is:

$$0.85 \times 0.5 + 0.6 \times 0.05 + 0.2 \times 0.45 = 0.545$$

5. The score of the WWL parent node is computed as the square root of the product of the weighted sum with the score of the weighted weakest child.

$$\sqrt{0.545 \times 0.2} = 0.33$$

Prioritized Siblings (PS): This relation exists among siblings each contributing to an independent aspect of the parent function. For instance, the window element in the house example is decomposed into the "structure" and the "blind". It is easily seen that the blind and the structure provide independent functionality which collectively contribute to the functioning of the window element. However, the failure of a single element (e.g. the blind) will not necessarily cause the functional failure of the window. A PS relation also recognizes the relative importance among the sibling nodes (see Figure 4). Formally, PS can be described as:

$S(parent) = \sum_{i=1}^{n} (Si \times Wi)$, where S is the assessment score and W is the weight percentile, and n is the number of children nodes.

Supposing *S(structure)* is 0.98 and *S (blind)* is 0.63, and their respective weights are 0.85 and 0.15 (see Figure 4). The score of the window can be calculated as:

$$S(window) = 0.98 \times 0.85 + 0.63 \times 0.15 = 0.928$$

The above list of functional relationships captures only a few types of component interactions. It should be noted that a short list of relationships may never be enough to represent all possible types of interactions among digital system components. As a part of our ongoing research, we will continue to define new functional relationships to handle the most common cases.

3. Weighting and Priorities

While decompose, sometimes it is necessary to differentiate the relative importance or weights among components. The weights indicate the degrees with which children nodes influence their parent.

A correct weight assignment is critical because the weights are used to compute the combinatorial effect of the various elements on the overall system. There are no general rules for determining the relative priorities of the elements besides careful use of one's expertise and judgment. While it is not possible to obtain completely objective weight assignments, we recommend an exercise used in Analytic Hierarchy Process (AHP) [3] to help identify the relative importance among sibling components.

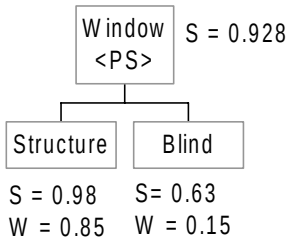
The technique of AHP is designed by Thomas Saaty. AHP elicits judgments in the form of pair-wise comparisons. To determine the weights among a set of n objects, AHP first performs $n(n-1)/2$ pair-wise comparisons of the objects with respect to a predefined ratio scale. Because the ratio scale is reciprocal, the results of the comparisons can be used to fill out an $n \times n$ matrix M where the entry Mij indicates the relative importance object i is to object j.

Once the matrix is constructed, AHP computes the eigenvalues of M and their corresponding eigenvectors. Assume that the largest eigenvalue is λ_{max}, Saaty has shown that the eigenvector associated with λ_{max} contains the most consistent weight assignment for the set of objects. A detailed description of AHP and the mathematics behind it can be found in Thomas Saaty's book "The Analytic Hierarchy Process"[3].

We use an example to illustrate the AHP weight judgment method. Example 4: A student wants to determine the relative importance of four activities in his life: A part-time job, study, personal activities and social activities. Using AHP, he will first make six pair-wise comparisons according to a pre-defined scale shown in Table 1. The results of the comparisons are shown in a matrix given in Table 2.

Table 1: Scale of pair-wise comparisons

Figure4: Functioning of the window in relation to its factors



Numerical Values	Definition
1	Equally important or preferred
3	Slightly more important or preferred
5	Strongly more important or preferred
7	Very strongly more important or preferred
9	Extremely more important or preferred

2, 4, 6, 8	Use as intermediate values to reflect compromise

Table 2: Results of pair-wise comparisons

	Job	Study	Personal	Social	weights using λ_{max}
Job	1	1/2	4	5	0.324
Study	2	1	5	6	0.508
Personal	½	1/5	1	2	0.103
Social	1/5	1/6	1/2	1	0.066

As indicated by the computation, the student considers "study" the most important activity, followed in order by his part-time job, personal activities and social activities.

Saaty's eigenvector method also yields a consistency measure, that is, it provides an indication of how consistent the entries in the comparison matrix are. Saaty argues that if the decision maker is perfectly consistent in making the pair-wise comparisons, then λ_{max}, the largest eigenvalue of *M*, should be equal to *n*. On the other hand, if he is inconsistent, then λ_{max} will be greater than *n*. The more inconsistent he is, the greater the value of λ_{max}. The proof behind Saaty's argument is beyond the scope of this paper. Interested readers should refer to Saaty's book [3].

AHP provides a way to formally deal with judgment errors through the use of the consistency measure. It has been used as a successful decision making tool in a wide range of applications.

On a final note, constructing such a functional decomposition requires a comprehensive and precise under-
standing of the system. Only when the functioning of the system is fully understood, can one build a model that is faithful to the system.

4. Basic Measurements

Previously, we posited that we could measure the end-to-end security attributes of a complex system given that there are ways to measure the security attributes of its basic components. In this subsection, we will explore the measurements of the basic components.

Measuring of the basic components are largely determined by the units and scale types. For example: if the measure of integrity is defined as "the probability of unauthorized alteration of information", then a potential basic measure of integrity can be the deployment of statistical methods on individual components to determine the value of such a probability measure.

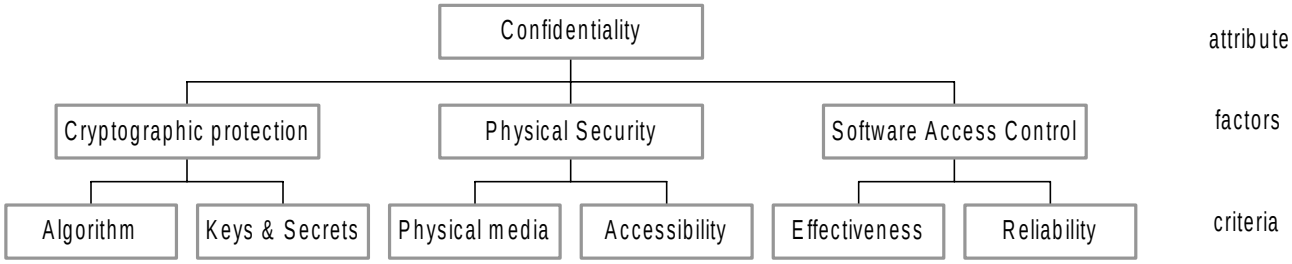
Most of the security attributes such as confidentiality and integrity are terms of qualities. In measuring such quality terms, an inherent difficulty is that there might be many different interpretations of what they really mean. Therefore, we must clearly articulate how these quality terms are to be defined. One way to do this is to define a model associated with the attribute to be measured.

For example, confidentiality of information has always played a central role in computer security. Unauthorized disclosure of information, if not prevented, may cause catastrophic results. In general, good cryptography combined with physical security is often considered to be our best answer to the problem.

We use a factor-criteria model to describe confidentiality. Figure 6 depicts such a model. It divides confidentiality into three main factors: cryptographic protection, physical security and software access control. These factors are then further broken into a set of lower level criteria.

Some of the criteria can be directly measured while others may need to be associated with a set of even lower level, directly measurable terms. For instance, Figure 7 shows how cryptographic protection can be described by two criteria and four basic metrics.

Figure 6. Confidentiality Model



A questionnaire of conditions can be used to solicit information about how rigorously the algorithm is tested, how long it has been used and what kind of cryptoanalysis was performed against it. A similar list of questions can be used to assess the implementation of the algorithm and the key storage mechanism.

There are many ways of transforming a questionnaire into a metric. One possible method is to use only "Yes" and "No" questions and assign a 1 to a "yes" answer and 0 to a "No" answer. The measure can be derived by computing the percentage of "yes" questions. For example, we can compute the level of cryptographic protection as the following to produce a measure that is a number between 0 and 1.

$$\frac{1}{2}\left(\frac{1}{2}\left(\frac{\text{Number of 1s for degree of testing}}{\text{Total Number of questions}}\right.\right.$$

$$\left.+\frac{\text{Number of 1s for implementation}}{\text{Total number of questions}}\right)$$

$$+\frac{1}{2}\left(\frac{\text{Key Length}}{\text{Minimum Length infeasible to break}}\right.$$

$$\left.\left.+\frac{\text{Number of 1s for key Storage}}{\text{Total Number of questions}}\right)\right)$$

Analogous measures can be calculated for physical security and software access control. Finally, by taking the mean of the three measures, we have the desired measure for "confidentiality":

Confidentiality = 1/3 (measure for cryptographic protection + physical security + software access control)

Because the overall estimate largely depends upon the basic metrics. Care must be taken in implementing them. Whether they are mathematical equations, diagrams, or questionnaires, they must be stated in a clear and unambiguous form to minimize the possibility of misinterpretation. They must also look for known weaknesses and security holes. Lastly, they should incorporate what is considered important to the organization or evaluator's needs.

In this example, all the questions and factors are weighted the same. Different weights should be used if special priorities are to be reflected.

Models can come in many different forms. A few examples of models are listed here for demonstration purposes. We envision that a set of general models will be developed by the community to handle the common cases. However, one does not have to accept any given models in his or her analysis, it is always a good exercise to develop one's own models to address specific concerns. We intent to lay out a set of basic principles in the SM framework for developing such models.

Integrity: The integrity model is similar to the one for confidentiality. They both build upon the same key factors which in turn depend on the same lower level criteria.

Figure 7: Cryptographic Protection

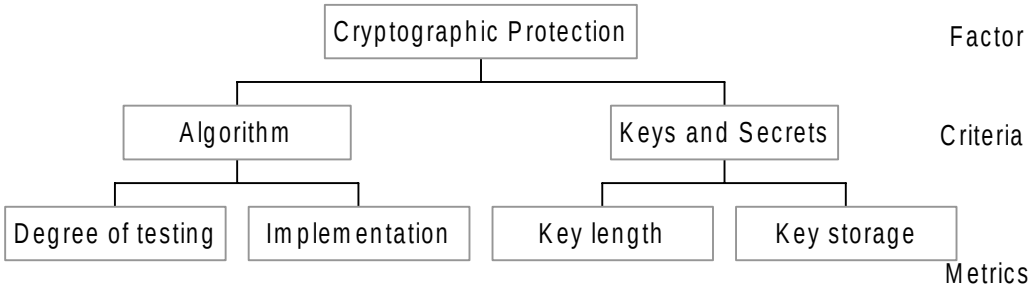
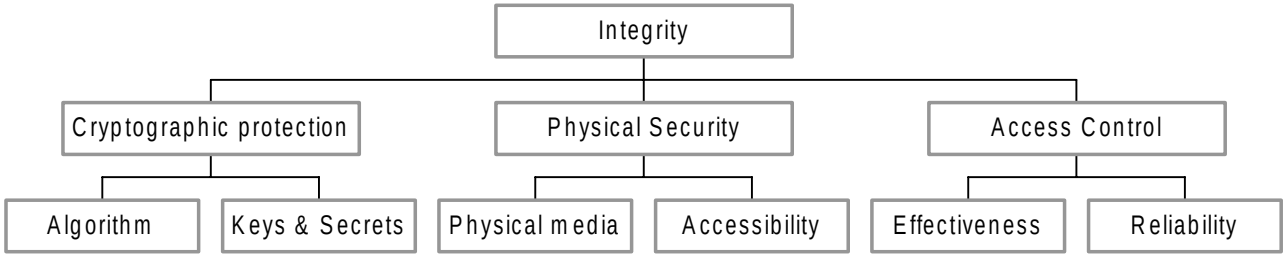


Figure 8. Integrity Model



Different questions may be used to assess the criteria in order to reflect unique integrity concerns.

Availability: There are plenty of examples of real time operations that can cause catastrophic results if denied of critical services. However, we know less about denial-of-service than the first two attributes. For example, how do one distinguish between an intentional, malicious flooding of the network from a simple degraded performance caused by occasional overloading?

While we do not have answers to all the questions, we prefer to study the subject through experimental measurement and observe its characteristic behaviors. One possible way of defining "Denial of service" in a measurable form is shown below:

Availability = the probability of a service request gets fulfilled

This probability can be determined through random samplings, statistics over a period of time, or specific testings. Using probabilistic measures are often useful in amortizing the effect of extreme and random events. Note that a maximum turnaround time should be defined for service requests.

Non-repudiation: Non-repudiation has become increasingly more important as electronic commerce quickly popularizes. Computer security must provide adequate mechanisms to support non-repudiation for security-related operations. We define the model for non-repudiation as follows:

To achieve non-repudiation, there must be an adequate proof-of-identify evidence and a robust underlying mechanism. For example, if the proof-of-identity is merely a four digit ATM PIN, the chance of that being compromised is a magnitude greater than biometric data such as finger prints or physical signatures. We must consider the integrity, reliability and consistency issues for the mechanism and the integrity, reliability of the evidence that is being used.

***Authentication*:** In many modern computer systems, authentication is an absolute necessity. Many security services are based on successful authentication. A sample model for authentication is defined in Figure 10.

It should be noted that high-level security attributes may also depend on attributes that are not purely security-oriented, such as reliability or predictability, in which case models for these attributes also need to be defined.

5. Component Sensitivity analysis

The last element of the estimation methodology is a component sensitivity analysis. A sensitivity analysis is performed to assess the impact of variations to the individual components. It allows us to identify a component or set of components' contribution to the overall system security.

If the overall measure varies dramatically with the change of

Figure 9: Non-Repudiation Model

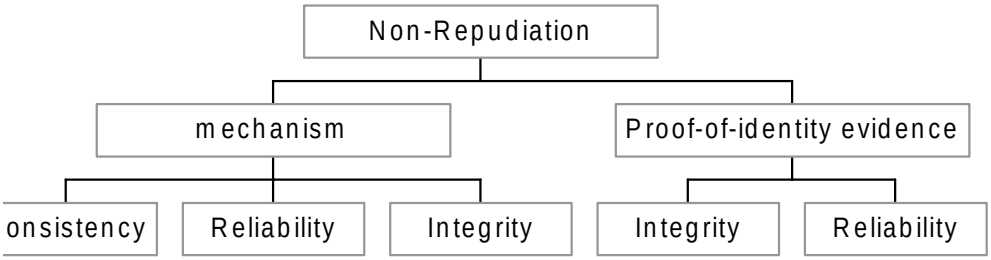
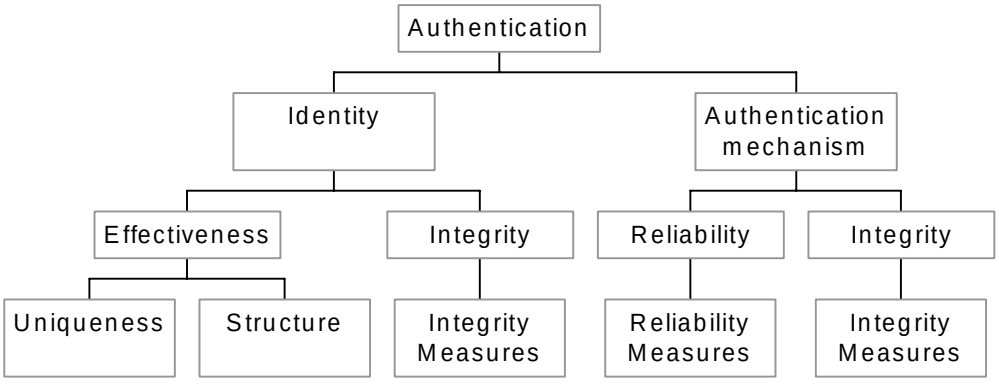


Figure 10: Authentication Model



a certain set of components, a closer second look should be performed on this particular set of components. Recommendations to improve the components' security strength can then be made based on the study.

Let us go back to the house example and briefly describe the process of a component sensitivity analysis. Using the composite rules defined by each functional relationship, measures of the high-level security attributes can be expressed in terms of that of the basic components. The contribution of each component is then calculated as the derivative of the high level measures with respect to the score of the basic components.

Figure 11 shows the previous example of the house system. The following list of equations is derived from the functional relationships used in the decomposition tree.

1. $S(house) = \min(S(door, S(window))$
2. $S(window) = 0.98S(structure) + 0.63S(blind)$

3. *S(door)*

$$= \sqrt{(0.5S(structure) + 0.45S(lock) + 0.05S(keystorage))} * \sqrt{\text{weighted min}(S(structure), S(lock), S(key))}$$

Once we have the basic measurements, the overall measure can be expressed as:

$$S(house) = S(door) = \sqrt{S(lock) * (0.5S(structure) + 0.45S(lock) + 0.05S(keystorage)}$$

Now we can calculate the component sensitivity indexes. Assuming that the other components' scores are kept constant, the Sensitivity Index (*S.I.*) of a particular component is defined as the derivative of the overall score with respect to the score of that component. For example, the *S.I.* of the door structure is computed as follows:

$$S.I._{door\_structure} = \frac{\partial S(house)}{\partial S(structure)} = \frac{0.5S(lock)}{2\sqrt{S(lock) * (0.5S(structure) + 0.45S(lock) + 0.05S(keystorage))}} = \frac{0.5 * 0.2}{2\sqrt{0.2(0.5 * S(structure) + 0.45 * 0.2 + 0.05 * 0.6)}} = \frac{0.5}{\sqrt{10S(structure) + 2.4}}$$

Substituting the value for S (structure), we have that $S.I._{door\_structure} = 0.1514$. For every unit of change in the performance of the door structure, the overall security strength of the house will change 0.1514 units. Table 3 shows the sensitivity indexes for all the leaf components of the house example:

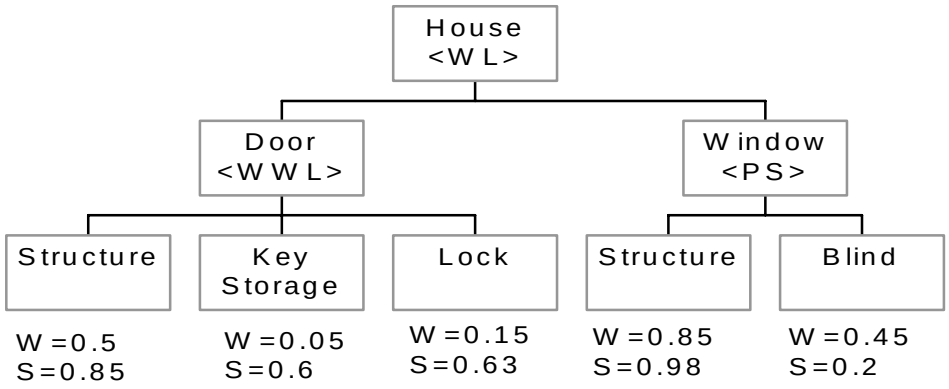
Table 3: S.I.s of the basic components of the house

	S.I. equation	*S.I. value*
Door Structure	$\frac{0.5}{\sqrt{10S(strucute)+2.4}}$	0.1514
Key Storage	$\frac{0.5}{\sqrt{S(keyStorage)+26.5}}$	0.096
Lock	$\frac{2.275+4.5S(lock)}{\sqrt{45.5+45S(lock)^2}}$	0.4617
Window Structure	0*	0
Window blind	0*	0

***:** S.I. of the window structure and the blind is only zero under the condition: S (window) >S (door)

The sensitivity analysis above shows us that improving the

Figure 11 : Decomposition Tree for a house



security strength of the lock has the biggest pay-off in increasing the overall security strength of the house. A component whose *S.I.* is greater than 0.1 is considered a nontrivial component.

Sensitivity analysis can also perform as a sanity check for the modeling effort. The decomposition may be with error if the *S.I.* of some component is drastically higher than the others. The decomposition process, coupled with adequate sensitivity analysis, can provide useful guidance as how to isolate the vulnerable areas of the system, identify the source of the problem and ultimately lead to the discoveries of security flaws or loopholes.

2.4 Validation of the measurement

If security measurement is to help us in our quest to better secure our systems, it is imperative that the measurements must be "valid".

Classical measurement theory [1] tells us that a "valid" measure is a mapping from an empirical domain to a numerical range such that the empirical relations are preserved and preserved by the numerical relations.

It is not always easy to prove the validity of a measurement, especially when the empirical relations cannot be readily identified. What we present here is a few thoughts toward validation of the security measurements. We do not claim it as a final solution, in stead, they are the research directions towards a more formal validation of the measurements.

Validation based on measurement theory

This step is a sanity check for the definition of the measurement. Once we have defined all the elements of the measure, we need to make sure that the definitions do not violate the basic axioms of the measurement theory.

If it is decided that the final measure of the system security is to be represented as a scalar value, our earlier discussion shows that a model to relate all the security aspects should be defined. Efforts must be made to ensure that such a model is justified. For example, any model utilizes arithmetic operations such as addition, subtraction and multiplication should not be used with ordinal-scaled measures.

In addition, care must be taken in interpreting the value of the measures. For example, to assess the system availability, one may use a probabilistic measure representing the average probability of a request being served. Probabilistic measures are ratio-scale measurements. A measured value of "0.8" indicates twice as much availability as a value of "0.4". On the other hand, if the system availability is categorized as *low*, *medium* and *high*, and the categories are mapped to the numbers 0, 1 and 2, a system measured "2" does not necessarily provide twice as much availability as one that is measured at "1".

Validation using empirical relations

It is true that in the context of computer security, extensive empirical studies are not readily available. However, there are still observed behaviors or relations that can be used to as a potential method to validate our measures.

The authors are currently engaged in an effort to apply the framework to a number of systems for which informal studies have previously observed relationships between certain system configurations and security flaws. Like the medical researchers have observed relationships between certain genes and obesity, similar correlations do exist in the context of computer security. The challenge then lies in how to filter out spurious correlations and discover meaningful relationships.

Validation using formal experiments

Another potential validation method is through the use of formal experiments. When carefully designed and executed, formal experimentation can be a useful vehicle in validating or disproving certain hypotheses.

First we have to carefully identify what the objectives or the hypothesis we set out to prove or disprove through the experiment. Then a meaningful plan must be designed to achieve the objectives. The plan must include applying different conditions and observing the change of behaviors in the subject.

Data used in the experiments should be collected in a carefully controlled environment. Statistical methods should be used to look for characteristic changes in both the experiment and the measurement. If the results indicate good correlation, we can at least increase our confidence in the measurement.

Besides careful, rigorous and complete experiment design, we must learn how to minimize extreme events and experiment errors. There are a host of studies on error handling techniques [5, 6, and 7]. How to design and conduct effective experiments is another ongoing effort of this research.

The downside of formal experiments is that they could be extremely time-consuming and costly to operate. In reality, it is extremely difficult to conduct formal experimentation on large systems. Therefore, it requires a great deal of design and preparation.

3. Future Work

There is a great deal to be learned about security measurements. In particular, the following issues need to be addressed:

1. A user guideline is needed to help ordinary user (not necessarily security-conscious ones) to identify and determine the relevant security properties.

2. The estimation methodology must be further developed. The near term tasks are:

a) Developing a set of guidelines to decompose a system and make visible the interactions between individual components
b) Further development of the functional relationships and the models that translate the security strength of lower level components into aggregated measures
c) Develop a set of automated tools for data gathering and quantitative analysis.

We envision incorporating the research results of other well-established areas such as software engineering and risk analysis to help us develop a sound process in which the assessment of complex, composite attributes can be translated into the measurements of more well-understood and measurable attributes.

3. A topic of great interest to us is vulnerability analysis in which we will utilize the estimation methodology and component sensitivity analysis to isolate and identify the sources of security vulnerabilities. The decomposition method lends itself well to a step-trace process, which can potentially be used to identify the component or a set of components that are the most responsible for the evaluation result. Efforts will be made to present the information in a descriptive form that is useful for risk identification and analysis. In particular, we will explore the following issues: Where and what things could go wrong? what is the likelihood that they would go wrong?, and what are the consequences?

4. Validation of the measurement is an interesting and challenging research question. One of our near term tasks is to identify suitable pilot systems on which a formal experiment can be executed. We also expect to identify a set of guidelines for running experiments as well as a procedure to analyze the results of the experiments.

4. Conclusion

We proposed the concept of a security measurement framework in this paper. The framework is based on the theory and practice of formal measurements. It is the analytic process of security measurement, not the final figures, that interests us.

One should be careful in utilizing such measurements. A study results in no visible security vulnerabilities does not indicate risk free. It may increase our confidence in the system if the study is conducted in a careful and adequate manner. To better facilitate the use of the measurement, we hope to develop a set of practical user guidelines in the near future.

There are a great deal more to be learned about computer security. We hope that this study is a step toward the right direction. The author will continue to engage in efforts to advance our knowledge of the subject. We need the support and feedback from the community to help us refine and improve the rigor of the measurements. We believe that through measurements and other formal studies, we can increase our understanding and ultimately lead to better management and practice of computer security.

5. Acknowledgments

This research is in part supported by a grant from the Corporate Information Security Office of Citicorp. The authors wish to thank Micky Lo, Steve Katz and Jan Jonak for their input and help with the project. Special thanks should go to Dr. Yacov Haimes, Dr. John Knight and Dr. Kevin Sullivan for stimulating conversations and useful insights.

6. References

[1] F. Roberts, "*Measurement Theory, with Applications to Decision-Making, Utility, and the Social Sciences*", Addison-Wesley, 1979.

[2] N. E. Fenton and S. L. Pfleeger, "*Software Metrics*", International Thomson Computer Press, 1996

[3] T. Saaty, "*The Analytic Hierarchy Process*", McGraw-Hill, 1980.

[4] B.L. Golden, E. A. Wasil, and P.T. Harder, "*The Analytic Hierarchy Process, Applications and Studies*", Springer-Verlag, 1989.

[5] W. E. Vesely, F. F. Goldverg, N. H. Roberts and D. F. Haasl, "*Fault-tree handbook*". U.S. Nuclear Regulatory Commission Rep. NUREG-0492, 1981.

[6] N. J. McCormick, "*Reliability and Risk Analysis, Methods and Nuclear Power Applications*", Academic Press, INC. 1981

[7] Y. Haimes, "*Risk: Its Modeling, Assessment, And Management"*, McGraw-Hill, 1998.

[8] J. R. Dunham, E. Kruesi, "*The measurement task area"* IEEE Computer, pp. 47-54, November 1983.

[9] S. MacDonnell, "*Rigor in software complexity measurement experimentation*", Journal of Systems and Software, Vol 16, pp. 141-149, 1991.

[10] E. J. Weyuker, *"Evaluating software complexity measures"* IEEE transactions on Software Engineering, SE-14(9), pp. 1357-1365, 1988.
[11] Department of Defense, *"Trusted Computer System Evaluation Criteria"*, Department of Defense Standard, December 1985.