2. REVIEW OF LITERATURE

This chapter presents a brief overview of some literature available pertaining to Security Metrics.

2.1 Operational Measures of Security

Littlewood et al. showcase the need for having “operational measures” of Security akin to those for Reliability (Littlewood et al., n.d.). They claim that the users of a System are more interested in having such operational measures rather than knowing about the process used to engineer the system. Unfortunately, many existing measures for Security seem to be more focused on the factors that are likely to influence the Operational Security. Such Operational Measures do exist in the case of Reliability defined as the “probability of failure free operation of the system for a specified length of time”. The “Orange Book Levels” (“NCSC”, 1985) take the notion of trust – which is an unquantified notion of Security. Since probabilistic approaches are used to yield sound quantifiers of reliability, Littlewood et al. explore the possibility of using the same with regard to Security. In this context, Lee uses probability distribution of the level of threat to model levels of security (Lee, 1989). Denning takes a statistical approach for the detection of transition of the operating environment from “non-threatening” to “threatening” (Denning, 1987).

Littlewood et al. attempt to draw various analogies between Security and Reliability and the deficiencies pertaining to these analogies. For example, parallels are drawn between the concepts of “System Failure” in the context of reliability and “Security Breach” in the case of Security. Just as “System Failures” are caused by faults, “Security Breaches” are caused by “Vulnerabilities”. Depending on whether the faults and failures are accidental or intentional there are 4 possible classes out of which Reliability is mainly concerned with accidental failures arising from accidental
faults. Security deals with intentional failures arising from accidental faults, (where an attacker comes to know about a vulnerability in a program unknown to the original developer), or intentional failures arising from intentional faults (of which, Trojan Horses are a popular example). The work of Littlewood et al. also shows how the notion of “execution time” in the context of Reliability cannot be applied to Security as easily. In Reliability, the concern is about a single stochastic process of failures in time. In Security, the notion of effort can play the role of time in Reliability and therefore it is possible to have “effort to breach” distribution analogous to the “time-to-failure” distribution. This leads to the contemplation of Operational Measures such as “Mean Effort to Next Security Breach” and “Probability of Successfully resisting an attack incurring a given expenditure of effort”. But in the case of Security there seems to be a doubly stochastic process – the process of Security Breaches as a function of effort and the process of effort as a function of time.

Littlewood et al. also present some deficiencies in the analogies between Security and Reliability that can be roadblocks in deriving Operational Measures for Security akin to those for Reliability. They argue that in the case of Reliability, the relationship between time and failure can be expressed as a step function. But such a notion when applied to Security, as the relationship between effort and breaches, fails to consider the rewards an attacker will accrue from a given attack. Just as Reliability is always defined in the context of an “Operating Environment”, Security is defined in the context of a “Threatening Environment”. But there is more variability in the threatening environment than that is in the operating environment as threatening environments can be deliberately created. In the case of Security, there is a need to grapple with more than one viewpoint – that of the owner and the attacker, while this is not the case in Reliability (Littlewood et al., n.d.).
In Summary, Littlewood et. al. present the case for the desirability of a probability based framework for Security similar to that for Reliability. But the practicality and the challenges of having such a framework have also been expressed.

2.2 Attack Surface Metric

Pratyusa Manadhata and Jeannette Wing present a metric to determine if one version of a Software System is more secure than another with respect to the system’s “attack surface” (Manadhata & Wing, 2005). The important caveat here is that they do not attempt to measure the absolute security of one system. The proposed measure is relative. The “attack surface” of a system is defined in terms of resources and it is assumed that the more resources a system has, the more susceptible it will be to attacks (having a large attack surface).

2.2.1 Model

Manadhata and Wing propose a model with a set $S$ of systems, having a user $U$ and a data store $D$. They define the environment of a system $s \in S, E$, as the triple $(U, D, T)$ where $T = S \setminus s$, the set of all systems excluding $s$. Every system is modeled as a state machine $M = (S, I, M, T)$ where $S$ is a set of states, $I \subseteq S$ is a set of initial states, $M$ is a set of methods, and $T \subseteq S \times A \times A$, is a transition relation. A state transition $s \times a \times s'$ is the invocation of method $m$ on state $s$, resulting in state $s'$. The method proposed is based on the notions of Entry Points and Exit Points. To understand these notions, Manadhata and Wing propose the following definitions (Manadhata & Wing, 2005):

- A Method $m$, receives a data item $d$, directly if a) $m$ receives $d$ as input or b) $m$ reads $d$ from data store or c) $m$ invokes a system $s_1$’s API and receives $d$ as result returned from $s_1$’s API.
A method \( m \) receives a data item \( d \) indirectly if a method \( m_1 \) of \( s \) receives \( d \) directly and passes \( d \) as input to \( m \).

A direct entry point of the system \( s \) is a method \( m \) of \( s \) such that a) the user \( U \) can invoke \( m \), or b) a system \( s' \in T \) can invoke \( m \), and \( U \) input-interacts with \( s' \), or c) \( m \) reads a data item \( d \) from the data store and the user can write to the data store \( D \), or d) \( m \) invokes the API of a system \( s' \) on which \( s \) relies, and the user input-interacts with \( s' \).

An indirect entry point of the system \( s \) is a method \( m \) of \( s \) such that (1) \( m \) is not a direct entry point of \( s \), and (2) a direct entry point \( m_1 \) of \( s \) invokes \( m \), and passes data items received by it to \( m \) as input, or (3) an indirect entry point \( m_2 \) of \( s \) invokes \( m \), and passes data items received by it to \( m \) as input.

The set of system entry points of the system \( s \) is the disjoint union of the set of direct entry points and the set of indirect entry points of \( s \).

The set of infrastructure entry points of the system \( s \) is the union of the set of direct entry points of all systems \( s' \) such that \( s \) relies on \( s' \), and the user \( U \) input-interacts with \( s' \).

A direct exit point of the system \( s \) is a method \( m \) of \( s \) such that (a) the user \( U \) can invoke \( m \), and receive data items as result returned from \( m \), or (b) a system \( s' \in T \) can invoke \( m \) and receive data items as result returned from \( m \), and \( U \) output-interacts with \( s' \), or (c) \( m \) writes a data item \( d \) to the data store and the user \( U \) can read \( d \) from the data store \( D \), or (d) \( m \) invokes the API of a system \( s' \) on which \( s \) relies and passes data items as input to \( s' \) API, and the user \( U \) output-interacts with \( s' \).

The set of system exit points of the system \( s \) is the set of direct exit points of \( s \). There is no notion of indirect exit point of a system.
• The set of infrastructure exit points of the system $s$ is the union of the set of direct exit points of all systems $s'$ such that $s$ relies on $s'$ and the user $U$ output-interacts with $s'$.

• An untrusted data input of the system $s$ is a persistent data item $d$ such that a direct entry point of $s$ receives $d$ and the user $U$ can write $d$ to the data store $D$.

• An untrusted data output of the system $s$ is a persistent data item $d$ such that a direct exit point of $s$ writes $d$ to the data store $D$ and the user $U$ can read $d$ from the data store $D$.

• The set of untrusted data items of the system $s$ is the disjoint union of the set of untrusted data inputs and the set of untrusted data outputs of $s$.

• An open channel of a system $s$ is a channel $c$ of $s$ such that a) the user $U$ can connect to $s$ through $c$ or b) a system $s' \in t$ can connect to $s$ through $c$ and $U$ either input interacts or output interacts with $s$.

The proposed method is based on the notion that a subset of resources of a system $s$, that the attacker can use, contribute to the system’s attack surface. A user can launch an attack by using the following resources: a) channels that the attacker can use to connect to the system b) entry points and exit points which are methods the user can invoke c) set of untrusted data items the user can send to and receive from the system. Hence the subset of resources that the attacker can use is determined by all the three listed above.

Because all resources are not likely to be used the model proposes the computation of the damage potential – which is the benefit accrued to the attacker by misusing the resource and effort – which is the cost incurred to the attacker in successfully using the resource to launch an attack. The Damage Potential of a method is determined by the method’s privileges (a method running with “root”
privileges will cause more damage that another running as “user”), while that of a
channel is determined by the channel’s type (socket allows exchange of raw bytes
while RPC doesn’t). The damage potential of a data item is determined by the data
item’s type (An executable file will cause more damage than an ordinary file).
Likewise the effort required to access a resource depends on the access rights of the
resource.

The attackability of a resource r, with damage potential, dp, and access rights,
ar, is \( \frac{dp}{ar} \). Hence the attackability of a method is \( \frac{\text{privilege}}{\text{accessrights}} \), the attackability of a
channel is \( \frac{\text{type}}{\text{accessrights}} \), and the attackability of a data item is \( \frac{\text{type}}{\text{accessrights}} \). The
cost-benefit ratio is symbolic rather than numeric.

The proposed method also groups the methods that have the same attackability
into an attack class. Same holds with respect to channels and data items. The
Attackability of an Attack Class A, is the total attackability of all the members of A.

Let SAC denote the set of all System Attack Classes obtained by grouping the
methods M having the same attackability and then taking the set of the sets thus
obtained. Likewise CAC denotes the set of all Channel Attack Classes and DAC
denotes the set of all data attack classes. Let SA denote the total system attackability
of all classes in SAC, and DA denote the total attackability of all classes in DAC, and
CA denote the total attackability of all classes in CAC. The System’s attack surface is
thus (SA, CA, DA).

To arrive at a numeric value, the method proposes devising a total ordering
among the set of privilege levels of methods, (Ex. jose<user< root), the set of channel
types (Ex. RPC < socket), the set of data items (ordinary-file<executable-file), and the
set of access rights of resources (unauthenticated < authenticated). Higher the
privilege level, channel type, and the data type, higher the damage potential. Likewise, higher the access rights level, higher the effort.

Substituting these numerical values in the formula for attackability defined above viz. \( \left( \frac{\text{privileges}}{\text{access} \, - \, \text{rights}} \right) \, \frac{\text{type}}{\text{access} \, - \, \text{rights}} \, \frac{\text{type}}{\text{access} \, - \, \text{rights}} \) a numeric 3-tuple (SA, CA, DA) is obtained which represents a quantified attackability (or Security) of the system.

Manadhata and Wing illustrate the method with examples and validate their findings by comparing the attackability of 2 IMAP server implementations (Manadhata & Wing, 2005).

As can be observed, this work represents a significant step in quantification of the security of a system. But the caveat suggested, applies to all security metrics proposed so far. Most of them are criticized for not quantifying the absolute security of the system. The discipline of Security Metrics is much in its infancy and such criticisms do not take away the values of the proposed metrics, which, even if do not quantify the security of the system in absolute terms, are valid when taken in a relative context. These can serve as milestones in the achievement of the Holy Grail – Absolute Quantitative Security Measures.

2.3 Directions in Security Metrics Research

Jansen claims that Security Metrics is not a novel discipline but one that gains sporadic interest. In his view, Security Metrics – which are objective and quantitative – can have several uses including (Jansen, 2009):

- Strategic Support – Aiding decision making
- Quality Assurance – tracking and correcting Security Flaws
• Tactical Oversight – identification of potential areas of improvement,
determination of compliance with security policies

In his work, Jansen quotes a definition of Security Metric from the SSE CMM Documentation (“SSE-CMM: Systems Security Engineering Capability Maturity Model”, 2008). “At a high-level, metrics are quantifiable measurements of some aspect of a system or enterprise. For an entity (system, product, or other) for which security is a meaningful concept, there are some identifiable attributes that collectively characterize the security of that entity. Further, a security metric (or combination of security metrics) is a quantitative measure of how much of that attribute the entity possesses. A security metric can be built from lower-level physical measures.” Security Metrics involve application of a method of measurement to a system that possesses assessable security properties. Jansen observes that to be useful the method of measurement applied should be reproducible – leading to the same results when applied by different evaluators – and repeatable – yielding the same results when the method is applied again by the same people (Jansen, 2009). He also notes that timeliness and relevance are also issues of paramount importance in the context of Security Metrics.

Jansen evinces a sense of optimism when he observes that while the measurement of many physical quantities like temperature started off with qualitative comparisons, like warmer and cooler, today they have reached a state of maturity and the same can be expected with respect to Security Measurement.

Technology Security Evaluation Criteria (ITSEC), 1991) have been proposed and applied but these methods, in the words of Jansen, have acquired only little success.

Jansen observes that the measurement of software properties has been a difficult problem in general, and quotes the criticisms against the Halstead Measures and McCabe Cyclomatic Complexity measure to substantiate the observation. Therefore, many of the measures seem to measure only some facets of the property they are expected to measure. He notes that Security Metrics can be Qualitative or Quantitative but the distinction in many cases is easily obscured. Qualitative indicators may be assigned to the observed numerical value, such as “low” for 0-5 vulnerabilities, “medium” for 5-10, and “high” for more than 10 vulnerabilities observed. In many cases, these Qualitative indicators are assigned numeric values, as for example, 1 for low, 5 for medium and 10 for high. The difference between the numerical values assigned does not have any significance as they are used solely for imposing an ordering (Jansen, 2009).

Two aspects of Security Measurement noted by Jansen are of particular interest. First, he observes that the measurement of some qualitative attributes such as beauty or perfume is inherently very subjective and therefore to objectively measure such properties, some measurable quality characteristics have to be assessed that have a high correlation to the attribute in question. Therefore, the discipline of Security Metrics can gain a lot from such similar techniques applied for these cases. Second, he notes that Security Measurements are likely to be more successful when the target of measurement of small. This intuitively makes sense as, when an attempt is made to measure the security of a single program, there is a high probability of success than when attempting to measure the security of a system composed of many programs.
Definitely, when programs secure by themselves are composed into a single system, the resulting system need not be necessarily secure.

2.3.1 Research Directions

Jansen presents five possible research directions with regard to Security Metrics (Jansen, 2009).

1. Formal Models of Security

Development of formal models that can adequately represent the security characteristics operational Systems can be of immense potential to the development of sound Security Metrics. Model checking, where a representation of a system is traversed to find out if desirable properties expressed as logical formulas hold, can also greatly aid the development of Security Metrics.

2. Analysis of Historical Data

Prediction of Security of Software Components can be done from data pertaining to the security of other similar components developed and deployed in the past. With regard to Security, every Security Attack opens up an avenue for learning about a new vulnerability and correcting the vulnerability. Ozment proposes a technique where the number of faults discovered in a body of code can be used to predict the number of faults remaining in the program (Ozment, 1986).

3. Artificial Intelligence

Dealing with inconsistency and uncertainty is an integral part of AI systems and AI systems that formulate, refine and test hypotheses from observed data have been proposed. Such Systems can be of great utility in the development of Security Metrics.
4. Concrete Measurements of the product

Many measures with regard to Security seem to focus on the process used to arrive at the product rather than at the product and Jansen suggests that measurements with respect to the products can be immensely useful. Several Source Code Analysis Tools that scan the source code and uncover potentially vulnerable patterns have been developed (Chandra, Chess & Steven, 2006; Michael, 2006) but most of these suffer from the drawback of generating a large number of false positives. Improvement is possible in this area, suggests Jansen. Various forms of Security Black-box testing techniques including fault injection can also help in the endeavor to develop sound Security Metrics.

5. Measurable Components

The probably best means, to ensure that sound security metrics can be formulated and used, is to develop modules that intrinsically attuned for measurement. For example, because cryptographic mechanisms lend themselves to measurement of the effort required to breach them, modules using the mechanisms can be easily measured for Security. Modules that rely on authentication mechanisms like passwords and biometric authentication, lend themselves to strength analysis that can be used to assess the security of the module.

2.4 OWASP Security Metrics

The Open Web Application Security Project (OWASP) lists the top ten vulnerabilities of web applications and Nichols and Peterson attempt to derive a number of metrics based on the top ten vulnerabilities and the life cycle phase (Nichols & Peterson, 2007).
2.4.1 Top Ten Items and related metrics

A) Unvalidated Input

This refers to the data from web requests that a web application uses without validation. A good metric to measure this vulnerability is “PercentValidatedInput”. Clearly, this is a design-time metric. The higher this percentage, lower the vulnerability (or, equivalently, higher the security). An organization, that observes a low value for this metric for most of the web applications it develops, should attempt to mandate the use of an input validation framework.

More formally, if $T$ represents the count of total number of data items received by a web application from web requests and $V$ represents the count of the number of data items validated, the ratio $V / T$, is the “PercentValidatedInput”.

B) Broken-Access Control

This refers to the situation where the application fails to restrict the operations of an authenticated user. A run-time metric in this context can be “AnomalousSessionCount” and this can be computed in two stages. First, the application server’s user log entries are scanned to determine the database tables accessed by a particular user in a particular session and a SessionTableAccessProfile consisting of the user ID and the list of tables accessed is prepared. In the second phase, the SessionTableAccessProfile is accessed to count the number of table accesses that do not fit into a user’s profile.

A high value for this metric can be indicative of a problem in the applications persistence layer.
C) Broken Authentication and Session Management

This problem occurs when the application does not properly maintain user credentials and session tokens and by exploiting this vulnerability, attackers can compromise passwords and a host of other sensitive information.

The metric “BrokenAccessCount” determined by counting the number of accounts not reporting any activity for more than 90 days and that will never expire. This metric is a run-time metric.

D) Cross-Site Scripting

In this case, an attack is transported to the web browser potentially leading to disclosure of session tokens or spoofing content to fool the user. A Penetration Testing Tool can be used to obtain a metric called “XsiteVulnCount” to capture this vulnerability.

E) Buffer-Overflow

Web applications can be crashed by Buffer-Overflow attacks that can largely take advantage from languages that fail to validate inputs such as C. A deployment time metric for this item, can be obtained by taking the mean, standard-deviation of the number of days it took to patch detected buffer flow vulnerability in a given period of time. A high value for the mean or standard-deviation is indicative of a requirement to correct the patching process that is either slow or inconsistent.

Another metric “OverflowVulnCount” can be obtained from tools that verify the patch levels of installed software against patch levels that repair known buffer overflow vulnerabilities.

F) Injection Flaws

This vulnerability occurs when a web application passes parameters without proper validation to external systems. A typical case is SQL Injection attack that
occurs when attackers embed malicious commands in SQL Commands passed to the database systems thereby getting access to sensitive information returned by the database system.

A metric can be “InjectionFlawCnt” – a run-time metric that can be obtained from a penetration testing tool that submits invalid parameters to external systems. Another metric is “ExploiedFlawCnt” which refers to the number of cases where an attacker has successfully used injection flaws to accomplish his mission. Both metrics have the potential of offering excellent feedback to developers on the extent to which their application can resist injection flaw attacks. Developers can use this feedback to improve their coding practices and thereby develop more secure applications in the future.

G) Improper Error Handling

When an exception occurs web applications have the tendency to disclose much information that can be exploited by attackers. For example, if a database accessed by the application is not reachable, the web application can output the IP address of the database server and the name of the database table accessed. Such information can be exploited by attackers to plot more severe attacks.

A Static Analysis tool can be used to count the number of function calls in the application that do not check the return values and this can be a good metric for indicating the Security of the Application. But caution must be exercised in not attempting to normalize the metric value by dividing it by the total number of function call as this can mask a serious problem (Nichols & Peterson, 2007).
H) Insecure Storage

Most web applications resort to cryptographic techniques to store sensitive credentials. A simple deployment time metric is a count of the number of servers with installed hard disk encryption facilities. Higher the metric value, higher the security.

I) Application Denial of Service

Here the attacker creates situations where resources are consumed in an abnormal fashion and therefore legitimate users are prevented from accessing the service. Penetration Testing tools can be used to yield metrics that can be indicative of the degree to which the application is secure against such attacks.

J) Insecure Configuration Management

Most web applications depend on many servers for achieving the desired function. The Security of these servers is vital to the security of the web application. As a simple deployment time metric, a simple count can be made of the number of servers with insecure configuration options like default passwords and guest users. A high value of this metric for a large number of applications developed by the organization warrants improvement in the deployment standards of the organization.

2.4.2 Security Scorecard

Nichols and Peterson envisage the development of an IT Security Scorecard for each web application like the one shown in the figure (Nichols & Peterson, 2007). After computing all the above metrics, through static analysis tools, penetration testing tools and manual code inspections, a score card can be prepared that can be of great aid to the project manager in assessing the security of the web application. The quantitative values are mapped to colors – for example, red for a range of undesirable values, yellow for a range of moderate values and green for a range of acceptable values. For example, the range of accepted values for “PercentValidatedInput” can be
values over 91, the moderate category can be for values in the range from 81-90 and values less than 81 fall in the undesirable category. Nichols and Peterson claim that several of their client organizations have benefited from the development and usage of such Security Scorecards. They also caution that political motives should not lead to the conversion of a “red” to a “yellow” or a “yellow” to a “green”.

To aid in the mapping from quantitative values to color-coded rating, Nichols and Peterson propose a set of steps inspired by the Six Sigma Framework (Nichols & Peterson, 2007).

i) Expression of each metric in terms of defects divided by opportunities.

ii) Mapping of values to colors by comparison of each value to the thresholds.

iii) Aggregation of all scores in a given vulnerability category to create a single summary code.

<table>
<thead>
<tr>
<th>OWASP Category</th>
<th>Rating</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unvalidated Input</td>
<td>![Green]</td>
<td>Design Standards in Place</td>
</tr>
<tr>
<td>Cross-Site Scripting</td>
<td>![Red]</td>
<td>Results of Penetration Testing this month</td>
</tr>
<tr>
<td>Injection Flaws</td>
<td>![Yellow]</td>
<td>Results of Penetration Testing this month</td>
</tr>
<tr>
<td>Authentication &amp; Session Management</td>
<td>![Yellow]</td>
<td>New password strength policy is giving good results</td>
</tr>
</tbody>
</table>

Figure 2.1 – IT Security Scorecard. Adapted from (Nichols & Peterson, 2007).

The OWASP top ten Security Metrics represent a significant effort on the part of the security community in the endeavor to create Secure Web Applications. Given
the prevalence of web applications of the day, such a move is highly desirable. The greatest strength of this set of metrics is that most of them are easily computable and do not require significant effort for computation. The simplicity of them does not in any way dismiss their usefulness. The “Top Ten Security Items” published by OWASP are based on real-time data collected and the adoption of this metrics framework and IT security scorecard development can greatly aid both the web developers and web application managers in their endeavor to engineer secure web applications.

2.5 SSE CMM Project Metrics

The System Security Engineering – Capability Maturity Model, discussed in Chapter 1, measures an organization’s capability to provide security products or services. To answer the question of many, as to whether the application of these process will result in a more secure system or operational capability, the SSE CMM Project Metrics Action Committee was constituted by the SSE CMM Project (Kormos, Givans, Gallagher, & Bartol, n.d.). The SSE CMM Project Metrics Action Committee categorizes the metrics as “Process Metrics” – that could serve as quantitative or qualitative evidence of the level of maturity of a particular process area or could serve as a binary indication of presence or absence of a mature process - and “Security Metrics” – which is defined by Kormos et al. as a Measurable attribute of the result of an SSE CMM Security Engineering Process that could serve as an evidence for its effectiveness.

The metrics have been developed within the boundaries of the SSE CMM Process Areas – Project, Organizational and Security Engineering. Kormos et al. identify the potential users of the metrics as Project Managers and leads, Chain of
command and funding sources, Community at large, Procurement, Hostile Entities, and Product Vendors.

Kormos et al. take the view that every process has inputs, activities, constraints and outputs. With this view, they show some process and security attributes that can be measured.

Figure 2.2 – Process Attributes that can be measured (from (Kormos et al., n.d.))
Kormos et al. claim that to achieve any meaningful measure of the effectiveness of the application of a security engineering process it is necessary to know the baseline posture so that measurements can be made “before” and “after” the application of the process to demonstrate the effectiveness of the process. They propose the following model:

**Figure 2.3 – Security Attributes that can be measured (from (Kormos et al., n.d.))**

Kormos et al. suggest a top-down approach of grouping metrics with the intent of discovering and categorizing metrics. Such an approach considers the views of
both – Security professionals interested in reducing vulnerabilities – and managers – interested in operational capabilities and low-cost. Their top-down approach is shown in Figure 2.5.

Figure 2.5 – Top-down approach of Kormos et al.

Kormos et al. also apply the process and security metrics for an organization – Booz-Allen and present one important finding – that it is easy for security engineering organizations to conduct a meaningful self-assessment using process metrics than using Security Metrics. They attribute the nature of measuring security effective in networks to this observation.

The SSE CMM Project Metrics Action Committee believes that the application of mature engineering process to a problem will definitely yield better outcomes. The committee is in its initial stages of work and believes that community and client participation is vital to the success of its endeavor.
2.6 Common Vulnerability Scoring System (CVSS)

The Common Vulnerability Scoring System (CVSS) is an open framework aimed at assisting the IT management in discovering and prioritizing vulnerabilities across desperate hardware and software platforms. Mell, Scarfone and Romanosky list the following merits of the framework (Mell, Scarfone & Romanosky, 2007).

2. Open Framework – Vulnerabilities are not given arbitrary scores. Instead the characteristics used to derive a score are visible to the user.
3. Prioritized Risk – CVSS includes an “environmental score”, thereby making the score contextual. Therefore, scores are representative of the actual risks faced by the organization.

CVSS defines three metrics groups – Base, Temporal and Environmental.

2.6.1. Base Metrics

The characteristics of a vulnerability that are constant with time and uniform across environments are represented by the Base Metrics – Access Vector, Access Complexity and Authentication.

2.6.1.1 Access Vector (AV)

This metric describes how the vulnerability is exploitable. Possible values and their meanings are:

A) Local (L) – A Vulnerability with “local” access vector is accessible only when the attacker has physical access to the system or a local account. Example – Local Privilege Escalation

B) Adjacent Network (A) – Here the attacker requires access to the broadcast or collision domain of the vulnerable program.
C) Network (N) – A Vulnerability under this category does not require a local network access or a local access. In other words, the vulnerable software is bound to the network stack.

2.6.1.2 Access Complexity (AC)

The complexity of the effort required to exploit a vulnerability after the attacker has obtained access to the target system is exemplified by this metric. Possible values include:

A) High (H) – Specialized access conditions exist. Examples – the attacker must already have elevated privileges and may have to spoof additional systems, the attacker must resort to social engineering methods easily detectable by experts etc.

B) Medium (M) – Access conditions are somewhat specialized. Example – Attacker must have obtained some additional information before launching an attack.

C) Low (L) – No access conditions exist. Example – attack can be performed manually with only little skills, default configurations etc.

2.6.1.3 Authentication (Au)

The number of times an attacker must authenticate to exploit a vulnerability is quantified by this metric. The main distinction between this and the Access Vector is that this quantifies authentications required after gaining access to the system. Possible values include:

A) Multiple (M) – 2 or more authentications are required

B) Single (S) – One authentication is sufficient

C) None (N) – No Authentication is required.
2.6.1.4 Confidentiality Impact (C)

Confidentiality refers to disclosure of information only to authorized users. The impact of an attack on confidentiality is represented by this metric.

Possible values are:

A) None (N) – No impact on confidentiality

B) Partial (P) – There is disclosure but the scope is restricted. For example, when the attacker gains access to information stored in some but not all database tables.

C) Complete (C) – Total information disclosure.

2.6.1.5 Integrity Impact (I)

The trustworthiness and guaranteed veracity of the information is referred to as its integrity. The Integrity Impact Metric captures the impact of an attack on integrity of the system. Possible values are:

A) None (N) – There is no impact on the integrity of the system.

B) Partial (P) – Modification of some information is possible but the scope is restricted.

C) Complete (C) – Total Compromise of Integrity.

2.6.1.6 Availability Impact (A)

Availability refers to the accessibility of the system services. Attacks such as abnormal consumption of processor cycles, excessive utilization of network bandwidth all have an impact on the Availability of the System. This impact is quantified by the Availability Metric. Possible values are:

A) None (N) – There is no impact on availability

B) Partial (P) – There is a reduced performance but not a complete degradation.

C) Complete (C) – There is a total shut down of the service.
2.6.2. Temporal Metrics

When the threat posed by a vulnerability may change with time, CVSS proposes the usage of Temporal Metrics. For example, the threat posed by a vulnerability that is not yet known is zero, but this may not be zero always. As the system is used successively by the attacker, he might gain knowledge of the vulnerability and then launch an attack exploiting the vulnerability. All the metrics in this category have an “optional” value which is used when the user feels that a particular metric does not apply and in this case the metric has zero effect on the overall score.

2.6.2.1 Exploitability (E)

The current state of exploit techniques and code do have an effect on the severity of a vulnerability. For example, if exploit code is publicly available and can be used with no modification, the vulnerability can be exploited even by an unskilled attacker thereby increasing the severity of the vulnerability.

This is grouped under “Temporal Metrics” because the exploitability may change with time. Initially, exploit code will not be available publicly. But with the progression of time as attacker becomes familiar with the system, Proof-Of-Concept Exploit code may be published followed by functional exploit code. This functional exploit code may also be made to exploit the vulnerability consistently and delivered as a worm. Possible values are:

A) Unproven (U) – The exploit is theoretical or no exploit code is available.

B) Proof-Of-Concept (POC) – Proof-of-Concept functional code is available. The code may required substantial modification before it can be made to exploit the vulnerability

C) Functional (F) – Functional Exploit Code is available.
D) High (H) – the vulnerability is exploitable by mobile autonomous code

E) Not Defined (ND) – the metric does not apply

2.6.2.2 Remediation Level (RL)

When prioritizing vulnerabilities, the available remediation for a vulnerability plays a crucial role. For example a severe vulnerability but for which an official patch is available need not be treated in par with another for which no remedies are available. Possible values for this metric include:

A) Official Fix (OF) – A Complete official vendor solution is available

B) Temporary Fix (TF) – Vendor issues a temporary hot fix or workaround

C) Workaround (W) – Unofficial, non-vendor solution is available.

D) Unavailable (U) – There is no solution available or the solution cannot be applied

E) Not Defined (ND) – Metric does not apply

2.6.2.3 Report Confidence (RC)

The confidence is in the existence of a vulnerability is captured by this metric. It is important to note that the existence of a vulnerability is “Confirmed” when either the author or the vendor of the affected technology acknowledges it or when Proof-Of-Concept Functional Code is published. Possible values include:

A) Unconfirmed (UC) – There is a single non-official source or multiple conflicting sources

B) Uncorroborated (UR) – There are multiple non-official sources

C) Confirmed (C) – Vulnerability is acknowledged by the author or vendor or proof-of-concept exploit code has been published

D) Not Defined (ND) – Metric does not apply
2.6.3 Environmental Metrics

The risk imposed by a vulnerability largely depends on the user’s environment. For this reason, CVSS introduces the notion of Environmental Metrics.

2.6.3.1 Collateral Damage Potential (CDP)

The damage that can be caused by the exploitation of the vulnerability is quantified by this metric. Possible values are:

A) None (N) – No potential for loss of life or assets
B) Low (L) – Potential for slight physical or property damage
C) Low-Medium (LM) – Potential for moderate damage
D) Medium-High (MH) – Potential for significant damage
E) High (H) – Potential for Catastrophe
F) Not Defined (ND) – Metric does not apply

2.6.3.2 Target Distribution (TD)

The proportion of Systems that would be affected by the exploitation of the vulnerability is exemplified by this metric. Possible values include:

A) None (N) – No target systems are available or only highly specialized ones
B) Low (L) – Targets exist on a small scale, say between 1%-25%
C) Medium (M) – Targets exist in a medium scale, say between 26% and 75%
D) High (H) – Targets exist on a large scale, say between 76% and 100%
E) Not Defined (ND) – Metric does not apply

2.6.3.3 Security Requirements (CR, IR, AR)

The importance of an affected IT asset measured in terms of confidentiality, integrity and availability is captured by these metrics. Thus CR, IR and AR represent
the degree to which the Confidentiality, Integrity or Availability of the affected asset is compromised. The weightings to the base impact metrics are assigned based on this. For example, if CR is set to high, then the weighting for the Confidentiality Impact base metric would be higher. Possible values are:

A) Low (L) – Loss of (Confidentiality or Integrity or Availability) will have only a low impact

B) Medium (M) - Loss of (Confidentiality or Integrity or Availability) will have a medium impact

C) High (H) - Loss of (Confidentiality or Integrity or Availability) will have a high impact

D) Not Defined (ND) – Metric does not apply

2.6.4 Scoring Guidelines

Mell, Scarfone and Romanosky list several guidelines that can assist the analyst in scoring the vulnerabilities (Mell, Scarfone & Romanosky, 2007). Some guidelines are listed below:

i) Vulnerability scoring should not take into account the interaction with other vulnerabilities.

ii) The direct impact to the target host should be taken into account. For example, in the case of Cross Site Scripting, the damage to the user’s system is far higher than the target host. In this case consideration is to be given only to the damage to the target host

iii) Most commonly used privileges should be considered when scoring.

iv) When there are multiple exploitable methods, preference is given to the one that causes the greatest impact and not the most common one
v) When a local vulnerability exists that can be exploited from a remote computer, Access Vector should be set to “Network”

vi) If a vulnerability exists in the authentication scheme itself, the value “None” should be assigned to the Authentication Metric

vii) Vulnerabilities that give root-level access should be scored with “Complete” loss of Confidentiality, Integrity and Availability.

2.6.5 Equations

The scoring equations follow.

2.6.5.1 Base Equations

BaseScore = round_to_1_decimal((0.6 * Impact)+(0.4 * Exploitability)-1.5)*f(Impact))

Impact=10.41 *(1-(1-ConfImpact)*(1-IntegImpact)*(1-AvailImpact))

Exploitability=20*AccessVector*AccessComplexity*Authentication

f(impact)= 0 if Impact=0, 1.176 otherwise

AccessVector=case AccessVector of

  Requires local access: 0.395

  Adjacent network accessible: 0.646

  Network Accessible: 1.0

AccessComplexity=case AccessComplexity of

  High: 0.35

  Medium: 0.61

  Low: 0.71
Authentication=case Authentication of
    Multiple: 0.45
    Single: 0.56
    None: 0.704

ConfImpact=case ConfidentialityImpact of
    None: 0.0
    Partial: 0.275
    Complete: 0.660

IntegImpact=case IntegrityImpact of
    None: 0.0
    Partial: 0.275
    Complete: 0.660

AvailImpact=case AvailabilityImpact of
    None: 0.0
    Partial: 0.275
    Complete: 0.660

2.6.5.2 Temporal Equation

The Temporal Equation combines the temporal metrics with the base score to yield a temporal score that is not higher than the base score and not lower than 33% of the base score.

TemporalScore=round_to_1_decimal(BaseScore*Exploitability*RemediationLevel*ReportConfidence)
Exploitability = case Exploitability of
    Unproven: 0.85
    Proof-of-concept: 0.9
    Functional: 0.95
    High: 1.00

RemediationLevel = case RemediationLevel of
    Official-fix: 0.87
    Temporary-Fix: 0.90
    Workaround: 0.95
    Unavailable: 1.00
    Not defined: 1.00

ReportConfidence = case ReportConfidence of
    Unconfirmed: 0.90
    Uncorroborated: 0.95
    Confirmed: 1.00
    Not Defined: 1.00

2.6.5.3 Environmental Equation

The Environmental Metrics if used are combined with Temporal Score to produce an environmental score no higher than the temporal score.

EnvironmentalScore = round_to_1_decimal((AdjustedTemporal + (10 - AdjustedTemporal) * CollateralDamagePotential) * TargetDistribution)

AdjustedTemporal = TemporalScore recomputed with the BaseScore’s impact sub-equation replaced with AdjustedImpact Equation

AdjustedImpact = min(10, 10.41 * (1 - (1 - ConfImpact * ConfReq) * (1 - IntegImpact * IntegReq) * (1 - AvailImpact * AvailReq))))
Collateral Damage Potential=Case CollateralDamagePotential of

None: 0
Low: 0.1
Low-Medium: 0.3
Medium-High: 0.4
High: 0.5
Not Defined: 0

TargetDistribution=Case TargetDistribution of

None: 0
Low: 0.25
Medium: 0.75
High: 1.00
Not Defined: 1.00

ConfReq=Case ConfReq of of

Low: 0.5
Medium: 1.0
High: 1.51
Not Defined: 1.00

IntegReq=Case IntegReq of

Low: 0.5
Medium: 1.0
High: 1.51
Not Defined: 1.00
AvailReq = Case AvailReq of

Low: 0.5
Medium: 1.0
High: 1.51
Not Defined: 1.0


In their concluding remarks, Mell, Scarfone and Romanosky also acknowledge that many other metrics could have been included and no one scoring system fits all. Their work is a “compromise” between completeness, ease-of-use and accuracy. They also predict the expansion or adjustment of the scorings with the maturity of CVSS.

2.7 A Stakeholder Based Model of Security Metrics

Patriciu, Priescu and Nicolaescu (Patriciu, Priescu & Nicolaescu, 2006) propose a stakeholder based model of security metrics because different people are interested in different properties (Patriciu, Priescu & Nicolaescu, 2006).

Executive Officers responsible for the overall performance of the enterprise are interested in the following metrics:

A) Systems Service Level – Percentage of time information system services are available for a given period of time
B) Network Service Level - Percentage of time network services are available for a given period of time
C) Number of Compromises – Number of incidents in a given period of time during which security was compromised

D) Organizational Impact of Compromises – For each compromise the number of hours and people affected

E) Costs and Benefits of Improvements – The direct and indirect costs and benefits of improvements in Security Practices

F) Peer Performances – Service level benchmarks of similar enterprises

Network and IT Systems operation groups responsible for infrastructure and production support may be interested in security at a more granular level. Metrics of interest to them may be:

A) Compliant Devices – Percentage of services and devices compliant with the security policy

B) Managed Devices – Percentage of devices and services under active management

C) Viruses detected in email messages – percentage of emails infected by viruses

D) Unauthorized access attempts – Percentage of attempts to access a service without proper authorization

E) Network Utilization – Bandwidth utilization at key gateways

The Network and Systems Security team responsible for organizations security policies and goals may be interested in:

A) Vulnerability Counts – Number of vulnerabilities exploited in security compliant services vs. in those that are not

B) Unauthorized access attempts – Percentage of attempts to access a service without proper authorization
C) Incident Forensics – Number of incidents attributable to failure in being policy compliant

D) Remediation Time – Time between compromise discovery and compromise remedy deliverance

If these measurements are automated, subjective interpretations and measurement errors can be avoided. The stakeholder based model is interesting because it takes into account the fact that different people in the organization have different perspectives on security and thus may be interested in different metrics. Patriciu at al. claim that these metrics are used in large enterprises.