Volume 2 No. 7, July 2012 ISSN 2223-4985

International Journal of Information and Communication Technology Research

http://www.esjournals.org

A Conceptual Model for a Holistic Predictive Attack Ability Metric for Secure Service Oriented Architecture Software

[1]Samuel Mungai Mbuguah , [2]Waweru Mwangi , [1]Pang Chol Song [1]Geoffrey Muketha Muchiri
[1]Department of Computer Science
Masinde Muliro University of Science and technology ,Kakamega , Kenya
[2] ICSIT, Jomo Kenyatta University of Agriculture and Technology, Nairobi , Kenya

ABSTRACT

Software based systems are ubiquitous in modern day operations. There has been an increase in software based system attacks; leading to the need to equip the project managers, software designers and software developers with a better predictive attackability model at the architectural design stage. Attackability is a concept proposed recently in research literature to measure the extent that a software system or service could be the target of a successful attack. A literature survey of existing technical models was carried out to identify gaps in them. Also, a literature survey on, human traits that lead to human beings and the software system they man being subject to social engineering attacks was undertaken. Then a conceptual model has been proposed to extend the existing technical model and incorporate a social attackability model to produce a holistic predictive attackability model.

Keywords: *Attackability, Internal attributes, model, external attributes, metric*

[1]This study is part of an ongoing PhD thesis research work funded by the National Council for Science and Technology (NCST), Nairobi, Kenya.

1. INTRODUCTION

1.1 Background Information

The area of security metric is an active area of research. According to Wayne [1], there are several factors that impede progress in security metrics such as: The lack of good estimators of system security; the entrenched reliance on subjective, human, qualitative input; the protracted and delusive means commonly used to obtain measurements and the dearth of understanding and insight into the composition of security mechanisms.

In his paper Wayne, of the National Institute of Standard and Technology, in the USA department of commerce proposes several lines of research that address these factors and progress the state of the art in security metrics. The identified research areas are: Formal models of security measurement and metrics; Historical data collection and analysis; Artificial intelligence assessment techniques; Practicable concrete measurement methods; and intrinsically measurable components.

Software based system have internal attributes and external attributes. Internal attributes are easy to collect but difficult to interpret while external attributes are hard to collect but easy to interpret [2]. The major internal attributes include coupling, cohesion, and complexity. Some external attributes include safety, reliability, maintainability, efficiency, compatibility, portability and attackability. Predictive models allow mapping of hard to interpret internal measurement data into easily interpretable external measurement data.

Chowdhury et al. [3] carried out a research titled "Can Complexity, Coupling, and Cohesion Metrics be Used as Early Indicators of Vulnerabilities? In their work, they provide empirical evidence that complex, coupled, and non-cohesive software entities are often less secured.

The compelling rise of mobile and computer use across globe is not a passing cloud. Restraints for future growth do exist, while the power, accessibility, affordability of devices makes them an irresistible force in the coming decade. Many people will be using the mobile phones, with advanced capabilities for security critical applications. It is expected that there will be a tsunami of information insecurity. Software attacks will become more wide spread [4].

From a holistic security engineering point of view, real world systems are often vulnerable to attack despite being protected by elaborate technical safeguards. The weakest point in any security strengthened system is usually the human element; an attack is possible because the designers thought only about their strategy for responding to threats, without anticipating how real users would react. Some of the traits that make people vulnerable to social engineering attacks are: distraction, social compliance, herd mentality, dishonesty, kindness, time pressure, and need/greed [5].

2. RELATED WORK

2.1 Introduction

The issues of security and metrics have been studied for a long time, with continuous improvements on metrics and models being carried out, after analysis of existing models and analysis. The researcher highlights three recent literatures in this area and identifies the gaps in knowledge, which forms the bases of the proposed model.

2.2 Attack Surface

Manadhata et al.[6] carried out research on the attack surface, as an improvement what had be done at Microsoft. Measurement of security, both qualitatively and quantitatively, has been a long standing challenge to the research community, and is of practical importance to industry today. Industry has responded to demands for improvement in software security by increasing effort into creating "more secure" products and services. But how can industry determine if this effort is paying off? They suggest that their work was motivated by the questions faced by industry today: Has industry's effort made to make a system more secure paid off? Is the most recent release of a system more secure than the earlier ones? How can the results be quantified.

2.2.1 A New Metric

Manadhata et al.[6] proposed a new metric to determine whether one version of a system is more secure than another. Rather than measure the absolute security of a system, they measured its relative security: Given two versions, A and B, of a system, they measured whether version A is more secure than version B with respect to their attack surface. They did not use the attack surface metric to determine whether a version of a system is absolutely good or bad, rather to determine whether one version of a system is relatively better or worse than another.

Manadhata et al.[6] argued that a system's attack surface is the ways in which the system will be successfully attacked. They defined the attack surface of a system in terms of the system's resources; an attacker uses to attack a system. Intuitively, the more resources available to the attacker, the more exposed the attack surface. The more exposed the attack surface, the more ways the system can be attacked, and hence the more insecure it is.

Given two versions, A and B, of a system, they compared their attack surface to determine whether one is more secure than another. They admitted that attack surface measurements might be incomparable because of the way they defined attack surface along multiple dimensions. However, they opined that they could use the attack surface measurements along with the knowledge of the usage scenario of the system to determine whether version A is more secure than version B.

2.1.2 Contributions and Roadmap

Manadhata et al.[6] contributions to knowledge are : The introduction of the entry point and exit point framework to identify the resources that contribute to a system's attack surface; The introduction of the notion of attackability to determine the attack surface contribution of a resource and the definition of attackability as a cost-benefit ratio to the attacker; The introduction of the notion of attack classes for the convenience in defining the attack surface of a system.

2.1.3 Attack Surface Metric

In their prior work, they had defined the attack surface of a system in terms of the system's actions that are externally visible to its users and the system's resources that each action accesses or modifies[7]. Every system action can potentially be part of an attack, and hence contributes to attack surface. Similarly, every system resource also contributes to attack surface. Intuitively, the more actions available to a user or the more resources accessible through these actions, the more exposed the attack surface. Rather than consider all possible system resources, they narrowed their focus on a relevant subset of resource types. Attacks carried out over the years show that certain system resources are more likely to be used in an attack than others. Hence they did not treat all system resources equally.

They categorized the system resources into attack classes based on a given set of
properties associated with the resources. These properties reflect the attackability of a type of resource, i.e., some type of resource is more likely to be attacked than other types. They used the notion of attack class to distinguish between resources with different attackability. These attack classes together constituted the attack surface of a system. They measured the attack surface of four different versions of the Linux operating system[7].

Howard et al. measured the attack surface of seven different versions of the Windows operating system[8]. The results of both the Linux and Windows measurements confirmed perceived beliefs about the relative security of the different versions. They realized, however there is no systematic way of identifying the relevant subset of system resources that can be used in an attack, and the set of properties associated with each resource.

Hence it was difficult to identify the attack classes of a system. In their Linux attack surface measurement work, they used the history of attacks on Linux to identify the relevant subset of resources. Similarly, they relied on their knowledge of the Linux operating system to identify the properties of interest. They stated that Howard 's Windows attack surface measurement method suffers from a similar drawback: there is no systematic way of identifying the attack classes of the Windows operating system. Howard used the history of the attacks on Windows to identify twenty attack classes.

Their work was then motivated by the above mentioned difficulties in identifying the attack classes of a system. They defined the attack surface of a system in terms of the attackability of the system's resources. They used the entry point and exit point framework to identify the relevant subset of resources that contribute to the attackability of a system. They determined the attackability of each resource using a cost-benefit ratio to the attacker. They grouped the

resources into attack classes based on their attackability. The attackability of these attack classes constituted the attack surface of a system.

2.1.4 Entry Point and Exit Point Framework

Informally, entry points of a system are the ways through which data "enters" the system from its environment, and exit points are the ways through which data "exits" from the system to its environment. Manadhata et al.[6] argue that from their knowledge of the past that many attacks on software systems require the attacker to either send data into the system, or receive data from the system. Hence the entry points and the exit points of a system act as the entry points of attack on the system. They believe that the identification of the entry points and exit points of a system is the important first step towards the detection and prevention of such attacks. The entry point and exit point framework is a formal framework to define the entry points and exit points of a system. They use the framework to identify the resources that contribute to a system's attack surface.

They planned to implement a tool that will use the entry point and exit point framework to automatically identify the set of entry points and exit points, the set of open channels, and the set of untrusted data items from the source code of a system. The tool will help automate the attack surface measurement method described in the entry points and exit points of a system act as the entry points for attacks, hence the tool could be used to identify the parts of the code attention to make a system more secure. They needed to formalise their definition of attackability terms of a model. They proposed to use a state machine system model. A formal definition of attackability of a system's resources in terms of the execution traces of the state machine representing the system. They also planned to explore the usefulness of the entry point and exit point framework in the threat modelling process.

Threat modelling is a systematic way of identifying and ranking the threats that are most likely to affect a system[9]. By identifying and ranking threats, they could take appropriate countermeasures, starting with threats that present highest risk. The identification of entry points and exit points is an important step in the threat modelling process. The process, however, lacks a systematic way of performing this step. The users of the threat modelling process rely on their expertise and knowledge of a system to correctly identify the entry points and exit points. Manadhata et al.[6] proposed to use the entry point framework to formalize this step. Manadhata et al. attack surface measurement method assumes that the source code of a system is available while previous work on attack surface measurement [8] did not require the source code of a system. Manadhata et al.[6] planned to extend their method such that they could approximate the attack surface of a system, when the source code is not available.

Manadhata et al.[6] concluded by stating that there lacks a good metric for measuring a software system's security. They had proposed a metric to determine whether one version of a system is more secure than another. Their security metric was based on the notion of attack surface. They viewed their work as a first step towards a meaningful and practical metric for security measurement. They believed that the best way to begin is to start counting what is countable and then use the resulting numbers in a qualitative manner. They believed that their understanding over time would lead them to more meaningful and useful quantitative metrics for security measurement.

2.2 Complexity Measures for Secure Service-Oriented Software Architectures

Traore et al.[10] argue that software systems that run in open environments are facing more and more attacks or intrusions. This situation has brought security concerns into the software development process. Generally, software services are expected not only to satisfy functional requirements but also to be resistant to malicious attacks. Software *attackability* is defined as the likelihood that an attack on a software system will succeed;[8] in other words, it represents the ability of software services to resist malicious attacks

The main objective of Traore et al.[10] research was to develop a quantitative framework for predicting and mitigating software attackability in the early stages of the development process at the architectural level. They explored a quantitative approach for attackability analysis because the use of software metrics as cost effective quality predictors is widely accepted in the software community, both in academia and industry. It has been postulated that [11] there are no software systems that are immune to all kinds of attacks. Some may be strongly resistant to one form of software attacks, but seriously vulnerable to another kind. It is believed that some internal attributes of software products bear directly on the ability of software products to resist against specific forms of software attacks. Hence the need to affect these internal features in order to improve software security as an external quality.

Their objective of their research was to, study empirically software complexity as one such inner feature. They studied software complexity in the context of service-oriented architectures. They had shown in previous work how the User System Interaction Effect (USIE) paradigm can be used as measurement abstraction to derive various software security metrics[12]. They define a sample metric for service complexity based on the USIE paradigm, and use such metric to study the empirical relationship between service complexity and attackability. In their study, they use an open source web based software system as target application and focused on one form of security attack, namely the URL Jumping attack.

2.2.1 The Notions of Software Service Complexity

It has been argued that in the last several years, a rich body of research has been produced on software complexity. At the same time, with the growing interest in software security research there is a consensus that complexity has a negative impact on software security. Traore et al.[10] argue that, despite such consensus little effort has been made toward investigating empirically the link between complexity and security in software systems. Existing empirical works on software complexity have targeted for the most part traditional qualities such as correctness, reliability and maintainability, not security.

2.2.2 Measures

Using USIE abstraction[13], a software system can be represented as a collection of services structured hierarchically where the top service is a composite service representing the application itself. Furthermore, each of the services involved in the hierarchy can be described using a corresponding USIE graph that can be used to derive various metrics. In this case, the measurement targets are the software service entities.

2.2.4 Attackability Measure

Traore et al.[10] adopt an *effort reward* approach to assess the external attackability of software services in operational environments. Specifically, they define specific measurements for attack effort and reward with respect to specific software attack. The relative attackability among different software services (or different software applications) facing the same type of attack in identical operational environments can be captured by comparing the attack efforts required under the same reward, or by observing the attack rewards involved under the same effort. Generally, they compute the relative attackability by the ratio AttackReward/AttackEffort. Specifically for URL jumping attack, they quantify attack effort and reward of the URL jumping attack by defining the following metrics:

Measure of URL Jumping Attack Effort

AttackEffort service =The number of URLs exploited related to the service

Measure of URL Jumping Attack Reward:

$$Attack\,\mathrm{Re}ward_{URL-Jumping}(service_i)$$

Traore et al.[10] define the attack effort for URL Jumping attack on a given composite service as the total number of the URLs explored in finding a URL jumping vulnerability and they define the attack reward of URL jumping attack as a binary value which will be set to 1 if there is URL Jumping vulnerability in the given composite service and 0 otherwise. Since there is no universal interpretation for the notions of attack effort and reward for URL Jumping attack pattern, they defined such measurements based on their own understanding. Traore et al believe that their measurements capture the URL Jumping attackability to some extent, but they don't claim that these measures are definitive. They concluded that the URL Jumping attackability of a software service and the service's complexity metric value tended to strongly increase in the same directions. As the service complexity metric increased, the service URL Jumping attackability increased. Therefore, their hypothesis was supported by the experimental results.

2.3 Empirical Relation between Coupling Attackability in Software Systems

Traore et al[2] carried out research on the empirical relationship between coupling and attackability in software systems .Software attackability is defined as the likelihood that an attack on a software system will succeed [8], it represents the ability of software services to resist malicious attacks. Traore et al.[7] main objective in their research was to develop a quantitative framework for predicting and mitigating software attackability in the early stages of the development process at the architectural level. They explored a quantitative approach for attackability analysis because the use of software metrics as cost-effective quality predictors is widely accepted both in the research and industrial communities.

Traore et al. hypothesises that attackability increases as the amount of coupling increases in software systems. They stated their objective as to investigate the empirical relationship between coupling and attackability by presenting a case study of a Denial of Service (DoS) attack against a medical record keeping system.

2.31 Metrics

Objects and components of a software system are usually shared by different services underlying the software system. Sharing between services naturally involves some form of relationships between them[2] referred to this relationship as Service Coupling. A pair of services may share zero or more components. Some pairs of service may have a stronger coupling relationship than others with respect to specific criteria such as the number of shared components.

The degree of coupling can be captured using a coupling coefficient for each coupling relationship between services. Given two services ci and cj, the coupling coefficient between the two services is denoted by CoupCoefficient(ci,cj). A coupling coefficient should be non negative and equal to zero when there is no coupling. Various metrics can be used to generate the value of the coupling

coefficient given the coupling relationship. For instance, a possible coupling coefficient may consist of counting the number of shared persistent entities between two collaborations. This can be derived systematically from the USIE models[2].

2.3.2 DOS Attackability Measurement

Attackability is defined as the likelihood that a sofware system or service can be successfully attacked. However, due to the diversity of software attacks, it is difficult to capture software attackability universally. Traore et al are specific when referring to attackability of software systems or services. Their intention was to study the attackability of different software services with respect to a particular kind of DOS attack. More specifically, they proposed a reward-effort approach for measuring relative attackability among software services. Their basic strategy consisted of observing the different attack rewards under the same attack effort for different attack scenarios, and then capturing the relative service attackability specifically and objectively by the ratio between reward and *effort*.
Attackability =AttackReward / AttackEffort .

For instance, a typical flooding attack aims at either blocking or delaying regular software services by increasing massively workloads on software services. Therefore, the attack effort with respect to this type of DOS attack can be represented by the workload increase, and the attack reward can be defined by the delay in service execution.

Their ultimate goal was to identify the relationships between external software attackability and internal software attributes, and then use this information to improve security of software products at the early stage of software development. The specific purpose of the empirical study was to derive a statistic model to demonstrate the correlations between the DOS attackability and coupling measures for software services. In the study there was only one dependent variable and one independent variable, which are DOS attackability and coupling measures respectively. Their experimental hypothesis was simply that "DOS attackability increases as coupling increases" for software services[2].

2.3.3 Statistic Models

The correlation formula for the DOS attackability and the *ACOUC* metrics using statistical regression analysis is presented in this section. Based on the performance of the regression, Traore et al illustrated the strong relationship between the DOS attackability and the corresponding *ACOUC* metrics. Figure 3 shows a scatter plot between DOS attackability and *ACOUC* metrics of the medical record system application. According to Figure 3, there is a potential linear relationship between the mean DOS attackability and the corresponding ACOUC metrics, and there is also an increasing trend between the two variables of interest. The dependent variable *Y* was the DOS attackability and the software *ACOUC* metrics was the only *X* variable in the regression.

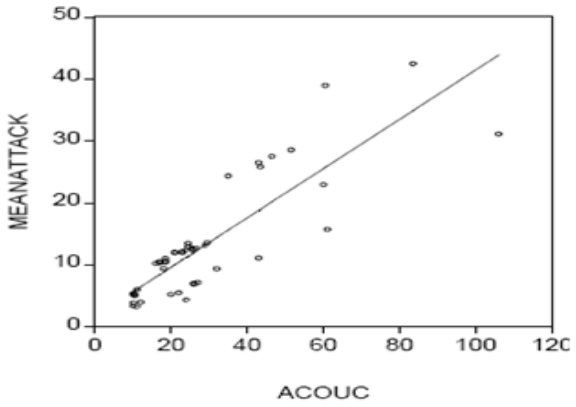

Figure 3: Scatter Plot between DOS Attackability and ACOUC metric[2](Traore at el. 2009)

The results obtained from the regression estimation using the statistic and econometric software package Eviews The model being estimated was:

$$MEANATTACK = a1 + a2ACOUCt + \epsilon t$$

Where *MEANATTACK* was the variable measuring the mean DOS attackability, *ACOUC* is the *ACOUC* metrics, and ϵt is the stochastic error or the random shock for this model; ϵt is usually assumed to be normally distributed with mean zero and σ^2 variance. The assumption could be tested by several means. The normality of ϵt also ensures the normality of MEANATTACK, because it is a linear function of the normally distributed variable t . The Now their estimation of model was

MEANATTACK=1.673+0.3983ACCOUCt

However Traore et al.[2] admit that even though their coupling metrics are shown to be good explanatory factors for DOS attackability, there might be other independent variables that should have been included in their regression analysis. They proposed in the future, to investigate the impact of other internal factors such as complexity on DOS attackability of software systems. Also, they were to study empirically, the relationships between internal software attributes and attackability using other forms of attacks.

2.4 Complexity, Coupling, and Cohesion Metrics as Early Indicators of Vulnerabilities?

Istehad Chowdhury and Mohammad Zulkernine carried out a research on the above topic. They argue that security failures in a software system are the mishaps we wish to avoid, but they could not occur without the presence of

vulnerabilities in the underlying software [3]. "Vulnerability is an instance of a fault in the specification, development, or configuration of software such that its execution can violate an implicit or explicit security policy" quoting Vulnerabilities are generally introduced during the development of software[14].

However, it is difficult to detect vulnerabilities until they manifest themselves as security failures in the operational stage of the software, because security concerns are not always addressed or known sufficiently early during the Software Development Life Cycle (SDLC). Therefore, it would be useful to know the characteristics of software artefacts that can indicate post-release vulnerabilities – vulnerabilities that are uncovered by at least one security failure during the operational phase of the software. Such indications can help software managers and developers take proactive action against potential vulnerabilities. Chowdhury et al.[3] use the term 'vulnerability' to denote post-release vulnerabilities only.

Software metrics are often used to assess the ability of software to achieve a predefined goal Fenton et al.[15. Software metric is a measure of some property of a piece of software. Complexity, coupling, and cohesion (CCC) related metrics can be measured during the software development phases (such as design or coding) and used to evaluate the quality of software. Because high complexity and coupling and low cohesion make understanding, developing, testing, and maintaining software difficult [15], these may lead to introduction of vulnerabilities.

The effect of cohesion on vulnerabilities has never been studied before. In their work, Chowdhury et al.[3] explore how vulnerabilities are related to all the three aforementioned aspects - complexity, coupling, and cohesion - both at the code and design level. Their objective being to investigate whether complex, coupled, and non-cohesive software entities are more vulnerable, and, if so, what CCC metrics can be used to indicate vulnerabilities in software? They postulate five hypotheses on how vulnerabilities are related to CCC metrics. The hypotheses are presented in Table 3. They argue that if the hypothesized relationships can be empirically validated, this information can be used during the early stages of software development to improve the ultimate security of software products. To validate the hypotheses, they conduct a case study on vulnerability data collected from fifty-two Mozilla Firefox [16] releases developed over a period of four years.

Table 3: the Hypotheses

	Hypotheses
H1	Complexities metrics positively correlate to the of vulnerabilities
H2	Coupling metrics positively correlate to the of vulnerabilities
H3	Cohesion metrics negatively correlate to the of vulnerabilities
H4	Code-level complexity, coupling, and cohesion metrics are better indicators of vulnerabilities than design-level metrics
H5	There is a subset of metrics that consistently correlate to vulnerabilities in all releases

In the case study, they first selected a set of standard metrics that measure complexity, coupling, and cohesion so that they could analyze their correlations with vulnerabilities. Then, they mined the vulnerability reports, bug repositories and software version archives of Mozilla Firefox to map the vulnerabilities back to their associated entities. An entity can be a function, file, class, module, or component. They then computed the CCC metrics for the entities and analyzed how these metrics correlate with the number of vulnerabilities so far fixed in those entities. The results of their case study showed that complexity, coupling, and cohesion metrics correlate to vulnerabilities at a statistically significant level. Since different metrics are available during different development phases, the correlations were further examined to determine which level (design or code) of CCC metrics were better indicators of vulnerabilities. They observed that the correlation patterns are stable across multiple releases of the software. These observations implied that the metrics could be dependably used as early indicators of vulnerabilities in software

They proposed for their future works the building vulnerability-prediction models from the CCC metrics. They also intend to analyze which level of granularity is more useful and feasible for vulnerability prediction; with the goal of proposing a framework, which can be used to obtain early prediction of vulnerabilities in software and as a basis for enhancing software security quality.

2.5 Principles to which Scam Victims Respond To

The principles to which victims respond have been studied by three researchers, namely: Cialdini, Lea et al and Stajano-Wilson. Their findings are listed Table 4.Wilson et al.[5] says that the finding support their thesis that systems involving people can be made secure only if designers understand and acknowledge the inherent vulnerabilities of the human factor. Their contributions are: They provided First hand data not otherwise available in literature; Second they abstracted seven principles; Third they applied the concept to more a general system point of view.

Table 4—Scam Victims

Principle	Cialdini (1985-2009)	Lea et al (2009)	Stajano-wilson (2009)
Distraction		~	X
Social	X	-	-

compliance(Authority)			
Herd (Social proof)	X		-
Dishonesty			X
Kindness	~		X
Need and greed (Visceral Triggers)	~	X	-
Scarcity (related Time)	X	-	~
Commitment and Consistency	X	-	
Reciprocation	X		~

~ -------Lists a related Principle
- Also lists this principle
X First identified this principle

They argue that behavioural patterns are not just opportunities for small scale hustlers but also of the human component of any complex system. They suggest that system –security architect should acknowledge the existence of these vulnerabilities as unavoidable consequence of human nature and actively build safeguards to prevent their exploitation [5].

2.6 Summary

The Table 5 is summary of literature review; list authors their contribution and gaps in their research.

Table 5 –Summary of Related work

Authors	Existing metrics frameworks	Limitations
Micheal Howard (2003)	Introduced the notion of attack surface.	Lack of systematic way of identifying attack surface , resources required and was based on prior knowledge.
Manadhata et al (2005)	Introduced the (i) entry and exit point framework for identifying the resources and attack surface. (ii) the concept of attackability as a cost benefit ratio (iii) the notion of attack surface classes for defining the attack surface.	The definition was based on intuition and hence the need to characterize attackability in a more formal way as a model.
Traore et al (2007)		Did not model the relationship.
Traore et al (2009)	Established a correlation between attackability and complexity using a URL jumping attac.	Did not model any other internal attributes such as complexity and cohesion.
Chowdhury et al (2010)	Established a correlation between attackability and coupling using DOS attack. Modelled the same.	He never did model the relationship. He relied on data mined from known vulnerability which means that data unknown vulnerabilities cannot be used to predict the future.
Ciadini, Lea et al and Sajano-Wilson (2011)	Established a correlation between CCC and Vulnerability Identified the principles that scam victims respond to and hypothesised that the same principles apply to software systems under social engineering attack.	They did not attempt to model the same nor did they link the same with technical based attacks.

2. PROPOSED CONCEPTUAL MODEL

The conceptual model constitutes two main blocks. The upper block illustrates how Technical attackability metric will be derived from the coupling, cohesion and complexity. The lower block illustrates the social attackability metric model. The seven human traits will be modelled and used to derive the mean social attackability metric which will be combined with Technical attackability metric to generate the system predictive attackability metric.

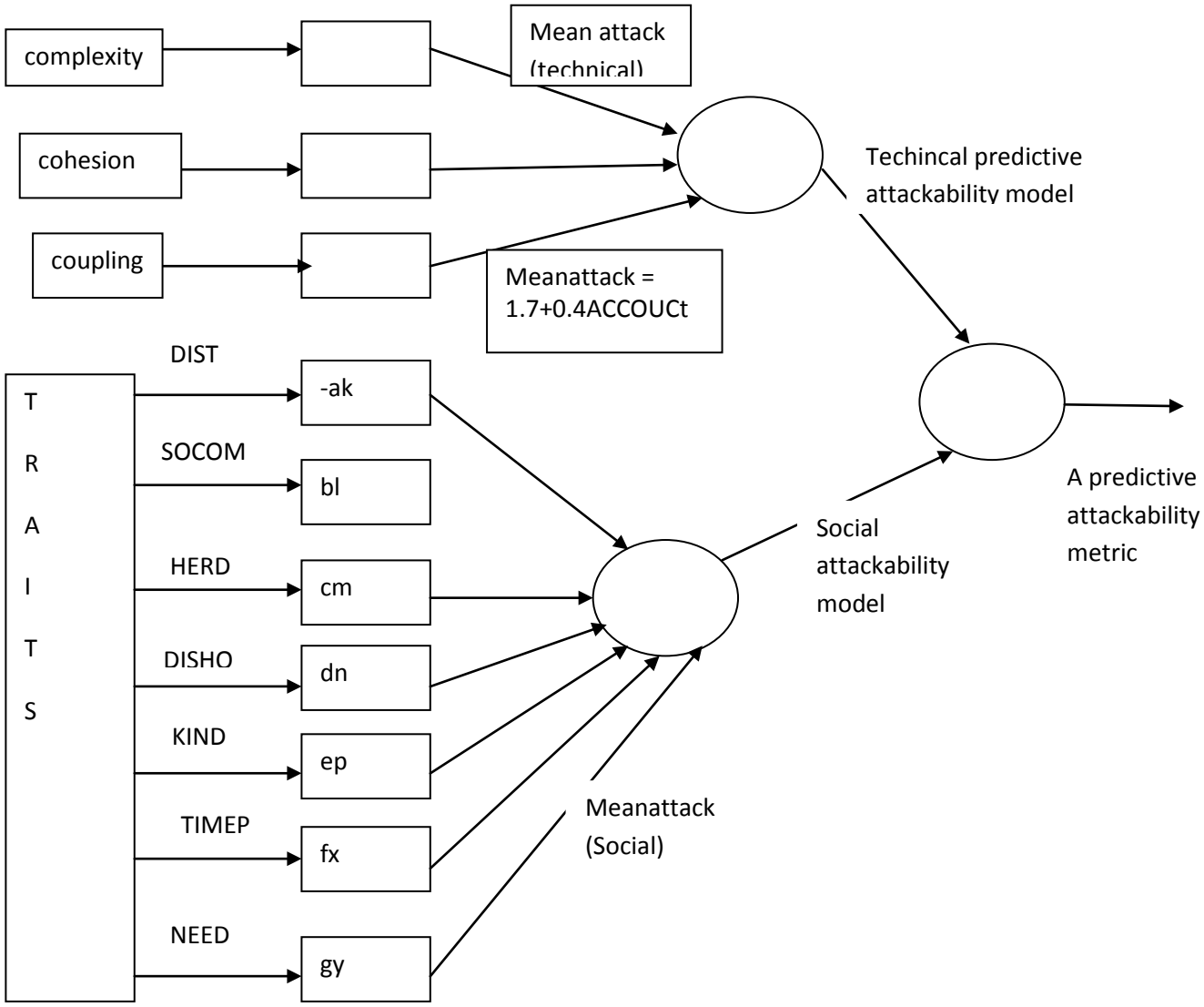

Figure 1 : Conceptual Diagram Source Self

3.1 Discussion

The coupling and attackability have been modelled as statistical model. The coupling and complexity are in the lower block, researchers have identified the seven human traits that lead human beings to be susceptible to social engineering attacks but no modelling has been done. The researchers proposes to do the modelling and combine it with technical predictive metric model.

The upper block indicates that the coupling versus attackability has been modelled. No modelling on complexity and cohesion has been done. The researchers propose to model for the two and then combine with coupling model to generate the Technical predictive model to produce a holistic attackability model, expected to be a statistical model. Hence they can all be combined to produce a technical complexity model and metric.

It can be assumed that each of the seven attributes occur in equal measure. If this be the case the factor labelled "a" to "g" will be 1/7.This can be verified by carrying out a research through, questionnaires, interviews, penetration tests and observations. If this is not the case then data collected can then be used to determine the frequency of occurrence of each which will be used to determine the factors "a" to "g". It can also be assumed that each of the trait contribute in equal measure to social attackability. This may not be the case the variables "k" to "y" assessment of weighting of the probable contribution to each them. It is assumed that the model of each will be statistical and all of them can be combined to generate a social attackability model and hence metric. To generate a holistic predictive

model and metric the technical model and social model should be combined to produce a metric. In the ideal case the metric should a numerical value. But there could be issues on whether this is justifiable. The researchers proposes to avoid this by proposing that the final metric M= mT + nS where small m represents the technical metric and n represents social metric.

REFERENCES

[1] Wayne, J. (2009). *Directions in security Metrics Research.* National Institute Of Standards and Technology USA Department Of Commerce.

[2] M.Y. Liu and I. Traore, ". R. (Jun. 2009). "Empirical Relations Between Attackabilityand Coupling: A case study on DoS,". *in ACM proc. SIGPLAN Workshop on Programming Languages and Analysis for security* (pp. pp. 57-64.). Ottawa, Canada: ACM.

[3] Instehad Chowdhury, M. Z. (March 22-26, 2010,). Can Complexity, Coupling, and Cohesion Metrics be Usedas Early Indicators of Vulnerabilities? *SAC'10,* . Sierre, Switzerland.: ACM.

[4] Harris, S. G. (2010). Emerging Markets:The coming African Tsunami of Information Insecurity. *Communications ACM* , 24-27.

[5] Wilson, F. S. (2011, march). Understanding Scam Victims:Seven Principles For system Security. *Communication Of ACM, Vol 54,No3* .

[6] Pratyusa Manadhata, J. M. (2005). *An Attack Surface Metric.* Pittsburgh,: Carnegie Mellon University.

[7] Wing, P. M. (2004). *Measuring a System's Attack Surface, Technical Report CMU.* Pittsburgh,: School of Computer Science, Carnegie Mellon University.

[8] Michael Howard, J. P. ((2003)). Measuring Relative Attack Surfaces,. *Proceedings of Workshop on Advanced Developments in Software and Systems Security* .

[9] F. Swiderski and W. Snyder, .. ((2004)). *Threat Modeling* Microsoft Press .

[10] Yanguo (Michael) Liu, I. Traore. (2007). Complexity Measures for Secure Service-Oriented Architectures. *Third International Workshop on Predictor Models in Software Engineering (PROMISE'07).* IEEE-COMPUTER SOCIETY.

[11] M. Andrews, J. A. (2005.). *"How to Break Web Software",*Addison-Wesley.

[12] M. Y. Liu, I. T. (June 2004). UML-based Security Measures of Software Products. *4th International Conference on Application International of Concurrency to System Design (ACSD-04),*. Hamilton,Ontario, Canada, : ACSD-04).

[13] Michael Yanguo Liu, I. T. (2005). *Measurement Framework for Software Privilege Protection based on User Interaction Analysis.* Victoria BC V8W 3P6, Canada: University of Victoria .

[14] Williams, Y. S. (Oct. 2008,). "Is Complexity Really the Enemy of Software Security?,. *" in the Proc. of the 4th ACM Workshop on Quality of Protection,* (pp. pp. 47-50). Virginia, USA: ACM.

[15] N. E. Fenton and S. L. Pfleeger. (1997,). *Software Metrics: A Rigorous andPractical Approach,* . Boston, MA, USA,: PWS Publishing Co.