CHAPTER-8

SYSTEM TESTING FOR
SECURITY MODEL FOR AN OBJECT ORIENTED
DESIGN (SMOOD)

Direct assessment of security is very intricate as it is a comprehensive nature, which fluctuates as per people, application and environment. Generally application level security is not adequate; security has to amalgamate with software development life cycle to endow with system level security. About 50 percent of the security flaws uncovered during Microsoft security push in 2002 were thoroughly associated to design level flaws. The direct measurement of security is complicated for the reason that there is no customary technique to compute security factors. For measuring this aspect, it is necessary to state in terms of metrics or model. In last chapter, we anticipated SMOOD model to assess the security of object oriented software at design stage of software development life cycle (SDLC). A variety of observations like the relationship the design properties were having with security properties were initiated by diverse appraisals of object oriented development allied literature. The proposed model could check the security in software architecture level which allows the designer to do the necessary action to deliver the secure software, which is safe by design. In this chapter attempt was made to validate the premeditated security model.

The validation of software product metrics means convincingly demonstrating that:
1. The product metric measures what it purports to measure. For example, whether a coupling metric is actually measuring coupling.
2. The product metric is associated with some important external metric (such as measures of maintainability or reliability).
3. The product metric is an improvement over existing product metrics. An improvement can mean, for example, whether it is easier to collect the metric or that it is a better predictor of faults (Devpriya Soni, 2009).

According to Fenton (N.Fenton and B.Kitchenham, 1990; N.Fenton, 1990), validating system in a given environment is the process of establishing the accuracy of the proposed system by empirical means. This can be done by comparing model performance with known data in the given environment. To estimate SMOOD was to assess how well the model is able to predict the “overall security” of an object oriented software design. The internal characteristics of a design can considerably
different based on the domain and the objectives of the design. Since the characteristics of design influence the security attributes and, therefore, the overall security, it was only significant to compare software designs that have been developed for similar or closely related objectives. Therefore, the validation of predictability of SMOOD required that several separate object oriented designs that had been developed. For the same set of requirement, can be evaluated and compared. This evaluation of the overall security of design determined by SMOOD required approving with the generally accepted requirements or characteristics of overall security designs. Based on these objectives the following procedure was adopted for validation of the overall software security assessments by SMOOD.

8.1 Investigation Techniques:
As per Fenton, (Fenton, 1990) there are two kinds of authentication like internal and external. Internal substantiation is a hypothetical application which assures that the metric is an appropriate statistical characterization of the property it asserts to determine. Describing that a metric determines what it professes to measure is a form of assumed authentication. External corroboration includes practically signifying points (2) and (3) above. Internal and external substantiations were also normally denoted as hypothetical and experimental authentication respectively (B. Kitchenham et al, 1995). Both these types of authentication were obligatory.

Three types of investigation were mentioned in empirical validation like experiment, case studies and surveys. A survey is a retrospective study that is done after an event has occurred. Both case studies and formal experiments were, usually, not retrospective. We need to decide in advance what we want to investigate and then plan how to capture data to support for respective investigation. A case study may be preferable to a formal experimentation if the process changes caused by the independent variables. The wide ranging, recurring the effects to the measured at a high level and across too many dependent variables to control and measure. In the proposed security model, it is observed that the changes in security occurred by different design properties, so case study approach is more appropriate than the survey and formal experiment. In baseline case study technique we can put side by side our new inspection approach with a general baseline. In this, data can be gathered from various projects, regardless of how different one project is from another one but should have general baseline. In present work, we have collected data from selected
open source software implemented with C++ having common baseline. Like pure
C++ programming language, the open source project of product is software
application, the product is released more than four stable products and the product has
long project age and big amount of download.

8.2 Stating Hypothesis:
Usually there were common reasons behind the new release of existing software like,
addition of new feature, changes in existing approach to boost potential, fitting of the
flaws found in the software. The early versions of the software were normally
unstable and undergo noteworthy redraft throughout initial releases. To improve
quality after stable release of a mature version of the software comes in the market
with improved robustness and reliability. So security of this mature version has been
observed to be significantly better than its predecessors. Not drastic changes but at
least it should have minor changes. After release of mature version, the approach is
expected to reserve the same for security and quality. It was anticipated that security
characteristics of each version of the software evaluated should match the expected
trends from one version to another version. To verify that the computed values of
SMOOD was in a valid range, the security attribute values were weighed against
predictable results.

8.3. Maintaining Control over Variables:
Once we have an explicit hypothesis, it is necessary to resolve on what variables can
affect its truth. In this study, we have used about ten different software’s on common
baseline as mentioned, five versions of 7zip, four versions of Keepass password safe,
sixteen versions of Bakery, eight versions of Interactive Visualization Framework
(IVF), six versions of GunWord, thirteen versions of logging framework for C++,
seventeen versions of Mock Objects for C++, thirteen versions of visual component
framework, nineteen versions of common CPP library, twelve versions of g3d. For
each adaptation of all software, in the first phase we had computed design properties
with the metric mentioned in Bansia quality model (Jagdish Bansia, 2002). In the
second part these computed design properties were used to compute the security
attributes like confidentiality, integrity and availability as mentioned in SMOOD
security model. The graphs plotted for individual software were shown in Fig. 8.1 to
Fig. 8.10. The discussion on obtained results were summarized in the next section.
8.4 SMOOD Testing on Open Source Software:

Fig. No. 8.1: Plot of computed security attributes values for 7z versions

Fig. No. 8.2: Plot of computed security attributes values for KeePass versions
Fig. No. 8.3 : Plot of computed security attributes values for bakery versions

Fig. No. 8.4 : Plot of computed security attributes values for ivf versions
Measuring the Impact of Software Design on Software Security

Chapter-8: System Testing for Security Model for an Object Oriented Design (SMOOD)

Fig. No 8.5: Plot of computed security attributes values for gunworld versions

Fig. No. 8.6: Plot of computed security attributes values for log4cplus versions
Fig. No. 8.7: Plot of computed security attributes values for mockpp versions

Fig. No. 8.8: Plot of computed security attributes values for vcf versions
Fig. No. 8.9 : Plot of computed security attributes values for commoncpp2 versions

Fig. No. 8.10 : Plot of computed security attributes values for g3d versions
8.5 Investigation over Hypothesis:

Validation is executed by correlating one measure with another. Fig. 8.1 to 8.10 shows the computed values for security attributes: confidentiality, integrity and availability for ten different software with their multiple versions. The detail descriptions of results were summarized in the following section.

For the 7 zips software (Fig. 8.1) we have five different releases, after second version software is viewing mature stable results for confidentiality, integrity, availability for next three releases. The effect of integrity was increased as compare to its previous versions. Whereas confidentiality and availability slightly decrease initially there after shows stable results compared to earlier versions. It is then matched with expected trends and our assumption.

For the keePass software (Fig. 8.2) there are four different releases, after second version software is shows mature stable results for confidentiality, integrity and availability for next three releases. The effect of integrity was higher compared to confidentiality and availability. The likely rise in value of security attributes as compared to its first version to the next one is seen in all versions of keepass software. It is then matched with expected trends and our assumption. The expected increase in the values of security attributes compatible with the hypothesis that the security attributes should improve with new releases.

For the Bakery software (Fig. 8.3) we have sixteen different releases, after tenth version software is viewing mature stable results for confidentiality, integrity, availability for next six releases. The integrity, confidentiality and availability increase as compared to its earlier version. But at the end slight decrease is observed. The expected increase in the values of security attributes is compatible with the hypothesis that the security attributes should improve with new releases.

In case of IVF software (Fig. 8.4) we have eight different releases, after fifth version software is viewing mature stable results for confidentiality, integrity and availability for next three releases. The effect of integrity was increased as compared to its previous versions. Whereas confidentiality and availability showing stable results as compared to earlier versions. It is then matched with expected trends and our assumption.

For the Gunworld software (Fig. 8.5) we have six different releases, after third version software shows mature stable results for confidentiality, integrity and availability for
next three releases. The effect of integrity was highly increased as compared to its previous versions. Whereas for confidentiality and availability shows stable results compared to earlier versions. It is then matched with expected trends and our assumption.

For the Logging framework C++ (Fig. 8.6) we have thirteen different releases, after seventh version software is viewing mature stable results for confidentiality, integrity, availability for next three releases. The effect of integrity, confidentiality and availability was higher and stable as compare to its earlier releases. The expected increase in the values of security attributes compatible with the hypothesis that the security attributes should improve with new releases.

For the Mock object for C++ (Fig. 8.7) about seventeen different releases, after tenth version software shows mature stable results for confidentiality, integrity and availability for next three releases. The effect of integrity was decrease initially then increases as compare to its earlier releases. Whereas confidentiality and availability remain stable as compare to its earlier releases. The expected increase in the values of security attributes compatible with the hypothesis that the security attributes should improve with new releases.

For the Visual component framework (Fig. 8.8) we have thirteen different releases, after fourth version software is viewing mature stable results for confidentiality, integrity and availability for next three releases. The effect of integrity was increase and decease so as showing mix trend. Whereas for confidentiality and availability shows stable results compared to earlier versions. It is then matched with expected trends and our assumption.

For the Common cpp library (Fig. 8.9) about nineteen different releases, after eleventh version software shows mature stable results for confidentiality, integrity and availability for next three releases. The effect of integrity, confidentiality and availability was higher and stable as compare to its earlier releases. The expected increase in the values of security attributes compatible with the hypothesis that the security attributes should improve with new releases.

For the G3d software (Fig. 8.10) we have twelve different releases, after third version software shows mature stable results for confidentiality, integrity and availability for next three releases. The effect of integrity was decreased for next five versions and then increase for last four versions. The effect of confidentiality and availability was higher and stable as compare to its earlier releases. The expected increase in the
values of security attributes compatible with the hypothesis that this security attributes should improve with new releases.

### 8.6 Discussion:

In this research work, we have efficaciously accomplished a security model SMOOD which measures the design security. Here, SMOOD model were tested on ten software with their all releases, so we can conclude that results shown by SMOOD model were match the predictable trend and our hypothesis that states that in first few releases software was undergoes the lots of changes, even software architecture have been changing and after a mature version of software comes in the market, which also keeps the same trend for its next releases. Security of these mature releases was observed either higher or same than its predecessors. As the hypothesized relationships were empirically validated, so the proposed security model (SMOOD) can be expended during the initial phases of software development to expand the eventual security of software products. Thus, the suggested modelling technique commendably seizures the conduct of both staunch and soft secure systems by employing the SMOOD model.