SMBC: A Security Grading Tool for Accessing the Security at Design Phase of Software Development

Neha Mahendra¹, Mohammad Muqeem²

¹ Department of Computer Applications, Integral University, Lucknow, India.
² Department of Computer Applications, Integral University, Lucknow, India.

Abstract

Security testing during designing the software will be advantageous to reduce the rework and expenses required if it will be found insecure after the implementation. Through proper framework security testing can be achieved efficiently at design phase. There are various tools available for testing the security but normally tools is available that will run in implementation phase. At design phase through algorithms and other design materials security can be accessed and the secure design can be provided. If any loophole found at this stage then alteration is design will be easy in comparison to later stages of development cycle. Security grading tool is designed which is based on the security metric. Metric is calculated with using few of the object oriented features.

Keywords- Security grading tool, design phase, software development life cycle, security metric, object oriented paradigm.

1. INTRODUCTION

In the past years many approaches focused on including the security testing as a continuous process in all the phases of software development life cycle. Some emphasis is given on the early phases of software development life cycle even some attention is also drawn on the design phase but there is still a need of proper framework for security testing at design phase [1]. Tools are also available but the input to security tool is normally source code and if any changes are found and required at this stage then lots of work is needed to accommodate the change in the software. Therefore there should be mechanism that will execute the security tool without the need of source code. Now, in the present era most of the softwares are object oriented and in the design phase various features can be used for the purpose of testing the software [2]. Some of the features of the object oriented paradigm that affects the security of a software system at the design time are Abstraction, Encapsulation, Inheritance, Cohesion, Coupling, Complexity and Polymorphism [3]. To have a secure system it is necessary to start thinking and working on security during inception of the software development [4]. Security testing then be performed up to design phase of software development life cycle and the level of testing can be accessed. According to the security level needed, the changes can be easily accommodated and the design of the software can be altered to meet the requirements of the security. Efforts required for software security takes a huge amount of cost as if achieved in later stages of software development [5]. Security assessment in the early stages will reduce this cost. Some researchers worked in this direction but the techniques used were somewhat complex and needs lots of work. Techniques like UML, Tropos, and Scrums etc have been used. With UML there is a need to draw the UML diagrams first then apply the metrics that is a lengthy process [6]. For Tropos, agent oriented approach needs to be learned properly that is difficult to understand, and also diagrams need to be prepared [7]. With Scrums, various iterations need to be done for effective work; also need to learn new concepts and a team is required to accomplish the project [8]. With SMBC there are no such needs and security can be accessed easily and automatically.

2. SMBC: Security Grading Tool

For the implementation and analysis of the security testing at design phase, the algorithm or pseudocode can be used to access the security level. In the object oriented software the main thing around which everything is running is classes on which software is based. If all the classes and their interaction with the other classes are secured then software security can be easily achieved. There must be some criteria on which the classes can be checked for the level of security. Further classes can be categorized according to the level of security they posses. Therefore, select the features first on which categorization of classes will be done. Access specifiers are considered here for finding the security level of the software. This will cover the four main characteristics of the object oriented paradigm that are inheritance, coupling, cohesion and encapsulation. The details of the attributes, methods and access specifiers will be available for each class and given in the application logic. Access specifiers are keywords in object-oriented languages that set the accessibility of classes, methods, and other members. There are basically three access specifiers that are public, protected and private. Methods, variables, and constructors that are declared private can only be accessed within the declared class itself. Private access modifier is the most restrictive access level. A class, method, variable, etc. declared public can be accessed from any other class. Variables, methods, etc, which are declared protected in a class can be accessed only by the subclasses in other package or within the package. On the basis of the access specifiers one can find out whether the attributes and methods
are accessible from the outside world (here the default access modifier is assumed under the protected access modifier). Normally, the accessibility of a member is not greater than the accessibility of the type that contains it. However, a public member of a default class might be accessible from outside the package if the member implements interface methods or overrides virtual methods that are defined in a public base class. All the requirements must be first collected and refined in order to start evaluating the security of the software [9].

In Object Oriented design, count the each type of attributes and methods in each class as-

- Total number of private attributes in a class = \( A_{pri} \)
- Total number of protected attributes in a class = \( A_{pro} \)
- Total number of public attributes in a class = \( A_{pub} \)
- Total number of attributes in a class = \( A \)
- Total number of private methods not returning any values in a class = \( M_{pri} \)
- Total number of protected methods not returning any values in a class = \( M_{pro} \)
- Total number of public methods not returning any values in a class = \( M_{pub} \)
- Total number of methods returning values = \( M \)
- Total number of protected methods returning some value in a class = \( M_{pro} \)
- Total number of public methods returning some value in a class = \( M_{pub} \)
- Total number of private methods returning some value in a class = \( M_{pri} \)

The classification of classes can be done as follows-

- **Highly-Secured classes (HS)**
  - A class is said to be a Highly-Secured class if following conditions met-
    - i. All the attributes are declared as private.
    - ii. All the methods not returning any values are private.
    - iii. All the methods returning values are private or protected.

It can be stated as follows-

Condition-1: if \( \{A_{pub} = 0; A_{pro} = 0; A_{pri} = A_T \} \)
AND \( \{ M_{pub} = 0; M_{pro} = 0; 0 <= M_{pri} <= M_T \} \)
AND \( \{ M_{V_{pub}} = 0; M_{V_{pro}} = 0; 0 <= M_{V_{pri}} <= M_T \} \).

- **Secured classes (S)**
A class is said to be a Secured class if following conditions met-

  - i. All the attributes are declared as private or protected.

- **Moderate-Secured classes (MS)**
A class is said to be a Moderate-Secured if following conditions met-

  - i. All the attributes are declared as private or protected.
  - ii. All the methods not returning any values are private or protected.
  - iii. All the methods returning values are private or protected.

It can be stated as follows-

Condition-2: if \( \{ A_{pub} = 0; 0 <= A_{pro} <= A_T; 0 <= A_{pri} <= A_T \} \)
AND \( \{ M_{pub} = 0; 0 <= M_{pro} <= M_T; 0 <= M_{pri} <= M_T \} \)
AND \( \{ M_{V_{pub}} = 0; 0 <= M_{V_{pro}} <= M_T; 0 <= M_{V_{pri}} <= M_T \} \).

- **Less-Secured classes (LS)**
A class is said to be a Less-Secured if following conditions met-

  - i. All the attributes are declared as private or protected.
  - ii. All the methods not returning any values are private, protected or public.
  - iii. All the methods returning values are private, protected or public.

It can be stated as follows-

Condition-3: if \( \{ A_{pub} = 0; 0 <= A_{pro} <= A_T; 0 <= A_{pri} <= A_T \} \)
AND \( \{ M_{pub} = 0; 0 <= M_{pro} <= M_T; 0 <= M_{pri} <= M_T \} \)
AND \( \{ 0 <= M_{V_{pub}} <= M_T; 0 <= M_{V_{pro}} <= M_T; 0 <= M_{V_{pri}} <= M_T \} \).

- **Un-Secured classes (US)**
A class is said to be an Un-Secured class if following conditions met-

  - i. All the attributes are declared as private, protected or public.
  - ii. All the methods not returning any values are private, protected or public.
  - iii. All the methods returning values are private, protected or public.
It can be stated as follows-

Condition-5: if \(0 < A_{\text{pub}} \leq A_T; 0 \leq A_{\text{pro}} \leq A_T; 0 \leq A_{\text{pri}} \leq A_T\) \\
AND \(0 \leq M_{\text{pub}} \leq M_T; 0 \leq M_{\text{pro}} \leq M_T; 0 \leq M_{\text{pri}} \leq M_T\) \\
AND \(0 \leq MV_{\text{pub}} \leq M_T; 0 \leq MV_{\text{pro}} \leq M_T; 0 \leq MV_{\text{pri}} \leq M_T\).

Store the application logic in a file and write all access specifiers explicitly. It is assumed that access specifiers are well specified and mentioned at relevant places and there is no default access specifier. If default access specifier is used then it is assumed under the category protected.

For accessing the security of the software there must be some evaluation criteria on which the security grade can be assigned to the software. A metric should be defined on the basis of which a scale is provided to evaluate the level of security [10]. A metric called as Security Metric Based on Classes (SMBC), which is based on the categorization of the classes can be defined as follows:

\[
\text{SMBC} = \frac{C_1 \times CC_{\text{HS}} + C_2 \times CC_{\text{S}} + C_3 \times CC_{\text{MS}} + C_4 \times CC_{\text{LS}} + C_5 \times CC_{\text{US}}}{C_T}
\]

where

- \(CC_{\text{HS}}\) = Total Number of Highly-Secured classes
- \(CC_{\text{S}}\) = Total Number of Secured classes
- \(CC_{\text{MS}}\) = Total Number of Moderate-Secured classes
- \(CC_{\text{LS}}\) = Total Number of Less -Secured classes
- \(CC_{\text{US}}\) = Total Number of Un-Secured classes
- \(C_T\) = Total Number of classes
- \(C_1\) = Multiplier for the weightage of Highly- Secured classes in security
- \(C_2\) = Multiplier for the weightage of Secured classes in security
- \(C_3\) = Multiplier for the weightage of Moderate -Secured classes in security
- \(C_4\) = Multiplier for the weightage of Less -Secured classes in security
- \(C_5\) = Multiplier for the weightage of Un-Secured classes in security

The standard values for coefficients are taken as follows:

\[C_1 = 1, C_2 = 0.75, C_3 = 0.50, C_4 = 0.25\] and \(C_5 = 0\).

Higher the value of SMBC better is the security of the software. Security grades are decided according to the value of the SMBC.

The value of the SMBC ranges between 0 and 1. The range is divided in five grades of security as stated in table-1. It is shown graphically in figure-1.

<table>
<thead>
<tr>
<th>SMBC Range</th>
<th>Security Grade</th>
<th>Security Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMBC=1</td>
<td>O</td>
<td>System is said to be fully secured and safe.</td>
</tr>
<tr>
<td>0.75&lt;=SMBC&lt;1</td>
<td>A</td>
<td>System is said to be secured and safe at optimum level.</td>
</tr>
<tr>
<td>0.5&lt;=SMBC&lt;0.75</td>
<td>B</td>
<td>System is secured at satisfactory level and takes care of sensitive attributes.</td>
</tr>
<tr>
<td>0.25&lt;=SMBC&lt;0.5</td>
<td>C</td>
<td>System is less secured and satisfactory level is low.</td>
</tr>
<tr>
<td>0&lt;SMBC&lt;0.25</td>
<td>D</td>
<td>System is not secured and security features need to be altered.</td>
</tr>
<tr>
<td>SMBC=0</td>
<td>E</td>
<td>System is insecure and fully vulnerable and all the security features need to be redesigned to rectify it.</td>
</tr>
</tbody>
</table>

![Figure 1: SMBC versus Security Grades.](image)

3. IMPLEMENTATION OF THE SECURITY GRADING TOOL

The SMBC security tool is made on the NetBeans8.0 software using JAVA—

The following instructions should be carefully read and followed before running the SMBC security tool:-

i. It is assumed that access specifiers are well specified and mentioned at relevant places and there is no default access specifier. That is private, protected, public keywords are properly written in the application logic.

ii. If default access specifier is used then it is assumed under the category protected.
iii. Keyword “void” is mentioned properly wherever applicable.

iv. There should not be any blank line in between the application logic text written in the file.

The interface is shown in the screenshots. The path of the file containing the design logic either in the form of pseudocode or algorithm should be entered in the field provided.

![Screenshot of interface](image_url)

**Figure 2**

The following values have been initialized during coding in java-

| Table-2 |
|-----------------|-----------------|
| Total number of public methods returning some value in a class\(M_{\text{pub}}\) | int countpublic = 0; |
| Total number of private methods returning some value in a class\(M_{\text{pri}}\) | int countprivate = 0; |
| Total number of protected methods returning some value in a class\(M_{\text{pro}}\) | int countprotected = 0; |
| Total number of public attributes in a class\(A_{\text{pub}}\) | int countpublicatt = 0; |
| Total number of private attributes in a class\(A_{\text{pri}}\) | int countprivateatt = 0; |
| Total number of protected attributes in a class\(A_{\text{pro}}\) | int countprotectedatt = 0; |
| Total number of public classes | int countpublicclass = 0; |
| Total number of private classes | int countprivateclass = 0; |
| Total number of protected classes | int countprotectedclass = 0; |
| Total number of public methods not returning any values in a class\(M_{\text{pub}}\) | int countpublicvoid = 0; |
| Total number of private methods not returning any values in a class\(M_{\text{pri}}\) | int countprivatevoid = 0; |
| Total number of protected methods not returning any values in a class\(M_{\text{pro}}\) | int countprotectedvoid = 0; |
| Total Number of Highly-Secured classes | int class_hs = 0; |
| Total Number of Secured classes | int class_s = 0; |
| Total Number of Less-Secured classes | int class_ls = 0; |
| Total Number of Un-Secured classes | int class_us = 0; |
| Total Number of Moderate-Secured classes | int class_ms = 0; |
| Security Metric SMBC | double SMBC = 0; |
| Multiplier for the weightage of Highly-Secured classes in security | double c1 = 1; |
| Multiplier for the weightage of Secured classes in security | double c2 = 0.75; |
| Multiplier for the weightage of Moderate-Secured classes in security | double c3 = 0.5; |
| Multiplier for the weightage of Less-Secured classes in security | double c4 = 0.25; |
| Multiplier for the weightage of Un-Secured classes in security | double c5 = 0; |
First of all number of classes in each category is calculated then SMBC is calculated as follows-

```java
System.out.println("CATEGORIZATION OF CLASSES : ");
System.out.println("Number Of Highly-secured classes In the File : "+ class_hs);
System.out.println("Number Of Secured-classes In the File : "+ class_s);
System.out.println("Number Of Moderate-classes In the File : "+ class_ms);
System.out.println("Number Of Less-secured classes In the File : "+ class_ls);
System.out.println("Number Of Un-secured classes In the File : "+ class_us);
SMBC=(((c1*class_hs)+(c2*class_s)+ (c3*class_ms)+ (c4*class_ls)+ (c5*class_us))/i);
System.out.println("\nSMBC= "+ SMBC+"\n");
```
The value of the SMBC metric and security grade is displayed at the top. The detail information about the classes is also displayed. Security grades are calculated by the following logic:

```java
if (SMBC==1)
    {jTextArea1.append("Security Grade is 'O'.");
     System.out.println("Security Grade is 'O'.");}
else if (SMBC>=0.75)
    {jTextArea1.append("Security Grade is 'A'.");
     System.out.println("Security Grade is 'A'.");}
else if (SMBC>=0.5)
    {jTextArea1.append("Security Grade is 'B'.");
     System.out.println("Security Grade is 'B'.");}
else if (SMBC>=0.25)
    {jTextArea1.append("Security Grade is 'C'.");
     System.out.println("Security Grade is 'C'.");}
else if (SMBC>0)
    {jTextArea1.append("Security Grade is 'D'.");
     System.out.println("Security Grade is 'D'.");}
else if (SMBC==0)
    {
     jTextArea1.append("Security Grade is 'E'.");
     System.out.println("Security Grade is 'E'.");}
```
For each class, the number of public methods, private methods and protected methods returning some value and also number of methods not returning any value will be displayed. Also the number of public, private and protected attributes will be displayed. The type of class itself will be displayed at the top.

Next, the total number of private, public and protected classes will be displayed. After this the categorization of the classes will be shown and the total number of classes in each category will be counted and displayed.

This is an automated security tool which runs on the algorithm and gives the security grade. With this resultant security grade it can be decided whether the design of the software is secure and correct in reference to security required for the given application. In case the grade is not satisfactory then the design must be altered to meet the security requirements. At this stage changes are easy and less expensive in comparison to the implementation phase.

4. ANALYSIS OF THE SMBC TOOL

The SMBC tool grades any application software in one of six categories. It is object oriented and runs on algorithm or pseudocode of the module or software. It is automated and does not need the source code of the software. It is based on the categorization of the classes defined for any application software. The four main characteristics inheritance, coupling, cohesion and encapsulation of the object oriented paradigm are taken into account for the analysis of the SMBC tool. Relationship between the object oriented security characteristic and the category of classes can be defined as given in the table-1. For the first category that is Highly-Secured classes, the Cohesion and Encapsulation are very high and Inheritance and Coupling are very low. The level varies as we move to other categories. Inheritance and Coupling increase and Cohesion and Encapsulation decrease as we move towards the categories of comparatively less secured classes. For the Un-Secured classes, the Cohesion and Encapsulation are very low and Inheritance and Coupling are very high. The trends of the four main characteristics of the object oriented paradigm that are inheritance, coupling, cohesion and encapsulation with the class category changes are shown in figure-6.

<table>
<thead>
<tr>
<th>Object Oriented Characteristic</th>
<th>Category of class</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Highly-Secured classes (HS)</td>
</tr>
<tr>
<td>Inheritance</td>
<td>Very Low(0.1)</td>
</tr>
<tr>
<td>Coupling</td>
<td>Very Low(0.1)</td>
</tr>
<tr>
<td>Cohesion</td>
<td>Very High(1)</td>
</tr>
<tr>
<td>Encapsulation</td>
<td>Very High(1)</td>
</tr>
</tbody>
</table>
Category of Classes & Object Oriented Features

Some cases are explained for the proper understanding of the security grading as follows:

- If all the classes are Highly-secured then
  \[ \text{SMBC} = \frac{1 \times C_T + 0.75 \times 0 + 0.50 \times 0 + 0.25 \times 0 + 0 \times 0}{C_T} \]
  This implies SMBC=1 in the best case and the security is highest. This is the Best case (Highest Security, SMBC=1).

- If half of the classes are Highly-secured and half of the classes are secured then
  \[ \text{SMBC} = \frac{1 \times (C_T/2) + 0.75 \times (C_T/2) + 0.50 \times 0 + 0.25 \times 0 + 0 \times 0}{C_T} \]
  This implies SMBC=0.875. It is the average case and the security is high.

- If all the classes are secured then
  \[ \text{SMBC} = \frac{1 \times 0 + 0.75 \times C_T + 0.50 \times 0 + 0.25 \times 0 + 0 \times 0}{C_T} \]
  This implies SMBC=0.75. It is the good case and the security is very high.

- If half of the classes are Secured and half of the classes are secured then
  \[ \text{SMBC} = \frac{1 \times 0 + 0.75 \times (C_T/2) + 0.50 \times (C_T/2) + 0.25 \times 0 + 0 \times 0}{C_T} \]
  This implies SMBC=0.625. It is the above average case and the security is high.

- If half of the classes are Moderate-secured and half of the classes are Less-secured then
  \[ \text{SMBC} = \frac{1 \times 0 + 0.75 \times 0 + 0.50 \times (C_T/2) + 0.25 \times (C_T/2) + 0 \times 0}{C_T} \]
  This implies SMBC=0.375. It is the average case and the security is medium.

- If half of the classes are Moderate-secured and half of the classes are Unsecured then
  \[ \text{SMBC} = \frac{1 \times 0 + 0.75 \times 0 + 0.50 \times 0 + 0.25 \times (C_T/2) + 0 \times (C_T/2)}{C_T} \]
  This implies SMBC=0.125. It is the below average case and the security is very low.

- If all of the classes are Less-secured then
  \[ \text{SMBC} = \frac{1 \times 0 + 0.75 \times 0 + 0.50 \times 0 + 0.25 \times 0 + 0 \times 0}{C_T} \]
  This implies SMBC=0.25. It is the average case and the security is low.

- If half of the classes are Less-secured and half of the classes are Unsecured then
  \[ \text{SMBC} = \frac{1 \times 0 + 0.75 \times 0 + 0.50 \times 0 + 0.25 \times (C_T/2) + 0 \times (C_T/2)}{C_T} \]
  This implies SMBC=0.125. It is the below average case and the security is very low.

- If all the classes are unsecured then
This implies $\text{SMBC}=0$ in the worst case and the security is lowest. This is the Worst case (Lowest Security, $\text{SMBC}=0$).

5. Conclusion and Future Scope

The paper presents an approach for the security grading of the object-oriented software at design phase. This approach is used and implemented in the form of SMBC security grading tool. The main advantage of this security grading tool is that it does not require the source code of the software and it can run on the algorithm or pseudocode of the software. The security level can be automatically calculated using this security grading tool. It is very easy to use only the path of the file is required where the algorithm or pseudocode is saved for the software that is to be tested. All the details of the classes used in the software will be shown and the categorization of the classes according to the security level is done. The SMBC is calculated and the final grade of security is assigned. This grade is useful to understand the level of security that the software is acquired and to make the changes if required at the early stages of software development. This information can also be used further for the analysis of the security in the later stages of the software development cycle.

6. ACKNOWLEDGEMENT

This work is acknowledged through the Research and Development Cell, Integral University, Lucknow with the research manuscript communication number (MCN): IU/R&D/2017-MCN000112.

7. LIST OF ABBREVIATIONS

\[
\text{SMBC} = \frac{1*0+0.75*0+0.50*0+0.25*0+0* C_T}{C_T}
\]

REFERENCES


AUTHORS’ BIOGRAPHICAL SKETCH

Mrs. Neha Mahendra is pursuing a Ph.D. in Computer Application from Integral University, Lucknow, U.P., India. She is currently working as a Head Computers at Georgions academy Lucknow, UP, India. She worked as Assistant Professor in the department of Computer Application, Azad Institute of Engineering & Technology, Lucknow, UP, India for 14 years. Her research interests are in the areas of software security, security testing and software complexity. She is currently working in the area of software security testing. She is the member of IAENG.

Dr. Mohammad Muqeem has earned his doctoral degrees from Integral University, Lucknow. He is currently working as an Associate Professor in the department of Computer Application, Integral University, Lucknow, UP, India. Dr. Mohammad Muqeem is a young, energetic researcher. He has more than fourteen years of experience in the field of academics. He is currently working in the area of requirement engineering and software security. He is a member of CSI, ISTE, CSTA, IAENG and other societies.