

# A Methodology to Check Integrity of a Class Hierarchy

S. Chandra<sup>1</sup> and R. A. Khan<sup>2</sup>

<sup>1,2</sup>BBA University, Department of Information Technology, Lucknow, India

Email: {<sup>1</sup>nupur\_madhur, <sup>2</sup>khanraees}@yahoo.com

**Abstract**—To satisfy security requirements, it is essential to protect our data from unauthorized information disclosure and information alteration. The work is focused on integrity quantification using design hierarchy of object-oriented software. Several access control mechanisms are available to check or to restrict unauthorized access, but they only check place restriction and not the extension of sensitive data. A methodology has been proposed in this paper to check the integrity of a class hierarchy. An Integrity Risk Factor (IRF), metric to compute probability of integrity violation of a class hierarchy has been proposed.

**Index Terms**— Object-oriented, Design Phase, Software security, Security quantification, Security Attribute.

## I. INTRODUCTION

Information security metric plays an important role to make sound decisions from different aspects of security [2]. Metric should be able to depict the robustness of security architecture used to maintain the effectiveness and efficiency of software security. Several security metrics at system level and coding phase are available in literature [3]. Attackers identify weakness and exploit them and it is believed that without correct and effective design and development process correct and effective implementation is not possible [2]. So, it is required to identify security weakness in design phase. There is need to develop such a methodology, able to find out weak points in design phase and measure it that may be exploited.

In this project work, “Security Quantification of OO s/w in design phase”, three basic security requirements has been considered confidentiality, integrity and availability [4, 6]. This part of the work is focused on integrity quantification using OOD. The concept behind to measure security loop holes is to monitor the flow of sensitive data in the class hierarchy and identify weak points. Weak points are those points where sensitive data is used without its corresponding validity check methods. All authentication, authorization, and checking methods are considered as validity check method such as: login etc. Integrity of software in design hierarchy is maintained or not it depends how much sensitive methods are prone to attack. Because if once a sensitive method is not protected then it may get hamper and it may make other classes risky.

## II. DATA INTEGRITY

In distributed information access environment, it is a big problem that how to ensure that access of secure information is safe. To achieve the purpose how to design

secure software systems and how to check consistency and integrity of software systems is still an open problem [5]. OOD characteristics have impact on or contribute to security attributes. So, it is possible to control or improve security attributes by manipulating design characteristics [9]. For example, Integrity is an important attributes of security and design characteristic inheritance, coupling etc. contribute affect integrity of software [1].

Data integrity of software ensures that protected data has been modified only by authenticated and authorized persons. In today’s scenario wide range of application are used to maintain security of the software, several security attributes are required to be implemented.

## III. TERMINOLOGY USED

The following terminologies are used in the proposed methodology.

**Sensitive Method:** Methods altering sensitive information are called sensitive method. If there are n classes in a class diagram then set of sensitive methods of class  $C_i$  is represented as  $SM(C_i) = \{SM_1, \dots, SM_j\}$ , where  $j=1..n$ .

**Validity Check Method:** A method verifying the authenticity for accessing sensitive data by implementing a conditional check is known as validity check method. Set of validity check methods of class  $C_i$  is represented as  $VM(C_i) = \{VM_1, \dots, VM_j\}$ , where  $j=1..n$ . In order to identify these methods, some suggestive checks are proposed in Table I.

**Don’t Care Class:** If a class neither has any sensitive method nor validity check method, the class is said to be don’t care class. Set of these classes is represented as DC.

**Sensitive Class:** A class having sensitive methods and needs protection or having integrity violation is said to be sensitive class. The methods of the class may not be public or modifying information without any authentication, authorization and validity check methods. Set of sensitive class is represented as SC.

**Risky Class:** Classes having sensitive methods without its corresponding validity check method are considered as risky class. Set of risky classes are represented as RC. In order to verify risky class, a checklist has been proposed in Table II.

**Safe Class:** A class having sensitive methods with corresponding validity checks method of its own, inherited or coupled before allowing modification of information, is considered as safe class.

**Super Class:** The class from which any method or attribute has been taken or shared through coupling, inheritance, aggregation etc. is considered as super class/source class. Set of all super classes for class  $C_i$  is defined

as  $SP(C_i)$ . If there are overall  $k$  validity check methods in the set of super class of  $C_i$  then it is defined as  $SP(C_i)^{vm}=\{M_1, M_2, \dots\}$

TABLE I.  
CHECKS TO IDENTIFY VALIDITY CHECK METHODS

S. No.	Checks	Y	N
1.	The method has any attribute having any information (e.g. user id, user name, card id, card no. etc.)		
2.	The method has any attribute carrying encrypted code, password / passcode etc.		
3.	The method has any attribute having authorization code		
4.	The method has any conditional check (e.g. expiry date).		
5.	The method has any transaction id.		

TABLE II.  
CHECKS TO VERIFY RISKY METHODS

S. No.	Checks	Y	N
1.	Class has any self checking method.		
2.	Class is not inheriting or sharing any validity check method corresponding to their sensitive methods.		
3.	Previous classes are safe.		
4.	Any corresponding validity check method exists, before accessing the class.		
5.	Class is sharing with already declared risky class.		

*Sub Class:* The class where the methods and attributes of source class have been floated through inheritance, coupling, inheritance, aggregation etc. is called as subclass/target class. Set of all sub classes for class  $C_i$  is defined as  $SB(C_i)$ .

IV THE METHODOLOGY

It is possible that already existing defects or weaknesses may not be in ordinary use. Sometimes they really don't have any importance. But in a security context, defect or weakness do matters, because attackers may exploit these weaknesses and induced failures. Attackers may take advantage of it. During software development, it is required to consider that how an intruder might go about attacking it and then made changes at the design phase in order to prevent from any loss. This methodology is similar to the concept of threat modeling i.e. to scrutinize architecture to discover and correct design-level security problems [8].

During integrity check process, sensitivity of class and method is checked. They are not to be tampered. Reusability feature of OO paradigm is advantageous, but with information flow the feature authorization of access privilege extends. Therefore, it is necessary to verify the sensitive information not to be made public. If it is, some restriction or protection is required at the point otherwise it will act as loop hole.

In design phase, a class having any self checking method or inheriting any validity check method corresponding to the sensitive method, may be verified. The validity check method ensures that the user has right to get facility of the resources. Permission may be

granted to the user, depending on application security policy.

A. Integrity Check Process

In order to check integrity of sensitive class, every method of class need to go through the Integrity Check Process (ICP). For every sensitive method, its corresponding validity check method (authentication, authorization or validation, depending on the security context), is must. The validity check method may be of inherited, coupled or a method of its own class.

For every sensitive method in  $SM(C_i)$ , its corresponding validity check method should be there. It may be in the same class or in any of its superclass.  $SP(C_i)$ . If any method of  $SM(C_i)$  does not have its corresponding validity check method, the Integrity of that method may get violated easily, making the class risky.

Let,  $X=\{SM_j | SM_j \in SM(C_i)\}$ ; where,  $SM(C_i)$  is the set of sensitive methods of a class  $SC_i$ , and  $0 < i \leq n$ ,  $n$  is no. of classes in a class hierarchy.  $SM_j$  is the sensitive methods of Class  $SC_i$ , need protection from unauthorized access, where,  $0 < j \leq m$ ,  $m$  is no. of methods in class  $SC_i$ .

$$Y=\{VM_k | VM_k \in VM(C_i)\};$$

$VM(C_i) \in \{VM(C_i) \cup SP(C_i)^{vm}\}$ ; Where,  $VM(C_i)$  is the set of validity check methods of a class  $SC_i$ .  $SP(C_i)^{vm}$  is the set of validity check methods of all super classes of  $SC_i$  where  $0 < i < n$ ,  $n$  is no. of classes in a class hierarchy.

$VM_k$  is the validity check methods for class  $SC_i$ , where,  $0 < k \leq p$ ,  $p$  is no. of methods in class  $C_i$ .

$$\text{Set } A = \{1, \forall SM \in X\}$$

$$B = \left\{ \begin{array}{l} 1, \text{ if } Y \text{ exists corresponding to } X \\ 0, \text{ otherwise} \end{array} \right\}$$

$$\text{and, } Z = \left\{ \begin{array}{l} 1 \text{ if } A=1 \ \& \ B=1; \text{method is safe} \\ 0 \text{ if } A=1 \ \& \ B=0; \text{method is risky} \end{array} \right\}$$

*Interpretation:* If  $Z=1$ , means integrity of method is maintained. The Sensitive method and its corresponding validity check method both exist. If  $Z=0$ , means integrity of method may violated, the method becomes risky because the sensitive method exists but corresponding validity check method does not exist (or used by any other means).

The designed process illustrates the integrity check of a method based on the presence / absence of sensitive method and validity check method either its own or used by any means. If result of any input combination is false, then the whole class will be considered as risky making the complete hierarchy risky.

B. Integrity Check for Class Hierarchy ( $IC^2H$  Algorithm)

Even though the problem of Integrity checks of a class hierarchy is relatively trivial, an attempt has been made to address the problem in the following section. For Integrity confidentiality check of class hierarchy ( $IC^2H$ ) algorithm illustrating the same has been proposed.

*Procedure begin*

- Step 1: for each class  $C_i$  do
- Step 2: if class is sensitive then
- Step 3:  $SC_i[\text{status}] = \text{null}$
- Step 4: push  $SC$

```

Step 5:   else
Step 6:       push DC
Step 7:   for each class of SCi do
Step 8:       cflag=NULL
Step 9:       if SCi[status]=null
Step 10:          derive CiSM
Step 11:          for each SMj ∈ CiSM
Step 12:              If ICP = false then
Step 13:                  cflag= false
Step 14:              If cflag=false then
Step 15:                  Verify with checklist of RC
Step 16:                  push RC
Step 17:                  for each class of SBCi
Step 18:                      push RC
Step 19:                      SCi[status]=checked
Step 20:              else
Step 21:                  If count (SPCi ∈ RC) > 0
Step 22:                      push RC
Step 23:                  SCi[status]=checked
Step 24:   for each class DCi do
Step 25:       If count(SPCi ∈ RC)>0
Step 26:       push RC
End

```

The algorithm first identifies set of sensitive classes and don't care class. To check the integrity of sensitive class (SC<sub>i</sub>), it is required to check integrity of all sensitive methods (SM<sub>j</sub>) of the same class. If, for any sensitive method, integrity check process (ICP) returns false (Z=0) then after verifying with checklist, the class and its all subclass will be recognized as risky class. And even if any of super-class of the sensitive class SC<sub>i</sub>, belongs to risky class then the class SC<sub>i</sub> will go to RC.

From the integrity check process it get clear that don't care class does not require any protection but if it uses or shares method or attribute from any sensitive class, safe class or risky class, in the absence of validity check method, then it may also act as a leakage channel or risky class.

Using the algorithm, probability of integrity violation of class hierarchy may be quantified in terms of integrity risk factor (IRF) using the formulae:

$$IRF = \frac{count(RC)}{\sum_{i=0}^n Ci} \quad (1)$$

Where, RC is the set of Risky classes.

In OO s/w systems, through inheritance etc. classes are connected to each other. Because of sharing of methods and attributes, state of the class can be changed. As an advantage of the algorithm, dynamic behavior of class can be derived. From integrity point of view these states can be divided in four categories: don't care, sensitive, safe and risky.

A class having no any sensitive method will be at the state of Don't Care. On adding sensitive method, class at Don't Care state will be switched to sensitive state. A class at Sensitive state, on adding validity checks method reaches to safe state. A class at Safe state having any loss of validity checks method switches to risky state.

Generally, there is no need to protect class at Don't Care, but if the class share with any sensitive, safe and risky class without protection, it may also comes at risky

state and it may increase leakage channels as a result integrity of class may hamper.

## CONCLUSIONS

Data integrity is related to the access to objects. Objects are responsible for ensuring their own integrity in terms of concurrent updates and possible failures. Moreover, in order to maintain data integrity it is required to check each object flow. Using OO concepts classes may interact with each other but they must be responsible for their collective integrity [7]. For development of secure software, security considerations should be made at the early stage of the software. A methodology has been proposed in this paper to check the integrity of the class hierarchy. Integrity of class is maintained or not, it also depends on its super classes and subclasses.

## ACKNOWLEDGEMENT

This work is sponsored by University Grant Commission (UGC), New Delhi, India, under F.No.34-107/2008 (SR).

## REFERENCES

- [1] David L. Spooner, "The Impact of Inheritance on Security in Object-Oriented Database Systems", Computer Science Department, Rensselaer Polytechnic Institute, Troy, New York 12180.
- [2] Wayne Jansen, "Directions in Security Metrics Research", National Institute of standards and technology, NISTR 7564, March 2009.
- [3] P. Mandhata, J. M. Wing, "An Attack Surface Metric" Technical Report, CMU-CS-05-155, 2005, Carnegie Mellon University. Pittsburg.
- [4] S. Chandra, R.A.Khan and A. Agrawal, "Software Security Factors in Design phase", In Proceedings of International Conference on Information Systems, Technology and Management, ICISTM 2009, CCIS 31, 12-13 March, Springer, Delhi, pp:339-340, March 2009.
- [5] Yi Deng, Jiacun Wang, JefEey J.P. Tsai, "Formal Analysis of Software Security System Architectures", Proceedings of International Symposium on Autonomous Decentralized Systems. Dallas, TX, USA, pp: 426-434, march 2001
- [6] S. Chandra and R.A.Khan, "Software Security Metric Identification Framework (SSM)", In Proceedings of International Conference on Advances in Computing, Communication and Control (ICAC3' 09), 23-24 January, Mumbai, Maharashtra, India, ACM, pp: 725-731, 2009.
- [7] Ruth Breu, Klaus Burger, Michael Hafner, "Key Issues of a Formally Based Process Model for Security Engineering". Research Group "Quality Engineering".
- [8] Michael Howard and Steve Lipner, "The Security Development Lifecycle: SDL: A Process for Developing Demonstrably More Secure Software", Microsoft Press, 28 April 2006.
- [9] J. Bansiya and Carl G. Davis, "A Hierarchical Model for Object-Oriented Design Quality Assessment", IEEE Transaction on Software Engineering, 28(1), 2002,pp:4-17.