International Journal of Computer Sciences and Engineering Open Access
Research Paper Volume-4, Issue-7 E-ISSN: 2347-2693

Security Model for Object Oriented Applications

Amanpreet Kaur

Department of Information Technology
Lovely Professional University, Jalandhar, Punjab, India

Available online at: www.ijcseonline.org

Received:20/Jun/2016 Revised: 30/Jun/2016 Accepted: 21/Jul/2016 Published: 31/Jul/2016

***Abstract* –Object oriented platform allow us to create 'n' number of objects of a class without imposing any constraints. By taking advantage of this feature, unknown objects of known and unknown classes can also be created in an application by the intruder. This paper presents a security model for object oriented platform to overcome such issues. In this paper, I have raised security issues related to object oriented applications. I have also discussed the needs of security for these applications.**

***Keywords* – Inheritance; Object; Unknown classes; Unknown Objects**

I. INTRODUCTION

Objects are the basic run time entities in object oriented system. [1] They can represent a person, an employee, a student, a table, or anything. Object oriented systems divides a problem into number of entities called objects and has data and functions associated with these objects. It has features like data encapsulation, polymorphism, message passing, inheritance, and dynamic binding. [2]

Object oriented platform makes it easier for a developer to convert a real life problem into a computer based application by using the 'reusability' feature. It can be used to develop business based applications effectively to manage data of an organization [3]

Example 1.1: 'emp.cpp'

```
class emp
{
 int  emp-id;    //represents the id of an employee
 char *name;    //represents the name of an employee

 public:

 emp()            //default constructor of class 'emp'
 {
 }

 void enter_id(int x,char *name)
 {                //methods of class 'emp'
  this->emp_id=x;
  this->name=name;
 }
 void show_id()
 {
  cout<<"\n id of an employee ="<<emp_id;
  cout<<"\n name of an employee="<<name;
 }
};


class salary:public emp
{
int salary;     //derived class 'salary' from 'emp'

 public:

 salary()        //default constructor of 'salary'
 {
 }
 void enter_salary(int x)
 {                //methods of class  'salary'
  salary=x;
 }
 void show_salary()
 {
  cout<<"\n salary of an employee    is="<<salary;
 }
};
```

Description:

In the above example, salary is derived from emp to add salaries with the names of employees. Two objects s1 and s2 of type salary are created to hold the data of employees.

```
salary s1;
salary s2;
s1.enter_id(1,aman);
s1.enter_salary(10,000);
s2.enter_id(2,garima);
s2.enter_salary (20,000);
```

TABLE 1
EMP DATA

Objects	ID	Name	Salary
s1	1	Aman	10,000
s2	2	Garima	20,000

"Objects holding data of employees"

In an application, known classes and objects are those which are created by an authorized person whereas unknown classes and objects are those which are created by an unauthorized person. Let us assume, s1 and s2 be the known objects of known class 'salary'.

In the below sections, security issues of object oriented applications are discussed. A proposed security model for object oriented applications is discussed.

II. SECURITY ISSUES OF OBJECT ORIENTED APPLICATIONS

If an unauthorized person has access to class definitions and methods of a defined class hierarchy then creating an unknown objects of it is an easy task for him. Through unknown objects of known classes, he will add unknown data in the database of an application.[4]

Inheritance is the special feature of object oriented system. It allows us to reuse a class 'n' number of times to derive new and more functional classes without bothering about any constraints. It can be used to develop effective business applications to manage data of an organization. If an intruder has access to class definitions and methods of an application then he can easily inherit a defined class hierarchy. He will add unknown classes to it. Through unknown objects of unknown classes, he will add unknown data in the database of an application.

Security issues are always different of different organizations. So, there is a need of such an embedded security model in object oriented platform which can be easily customized by developer for his application.

III. PROPOSED SECURITY MODEL TO IDENTIFY UNKNOWN CLASSES

The verification phase shown in Figure 5 should be included as an inbuilt feature of object oriented platform. Following should be the components of verification phase:

- An inbuilt string type variable should store type of an object before the object starts its inbuilt process of creation. Its value should be updated whenever a new object starts its process of creation. To understand the functioning of the security model, let's assume '**check_type'** to be that variable.

  salary s1; [check_type= "salary"]
  emp e1; [check_type= "emp"]
  department d1; [check_type= "department"]
- A file should be provided with object oriented platform which would be maintained by the developer (if he wishes to). The file should be designed to store the names of classes. If file is empty, then verification block will check nothing from it. In that file, developer can add the names of classes, he wishes to create in an application. The file permissions can be controlled by the developer. He can make it non-readable and non-editable for others to stop the addition of unknown classes by unauthorized people. Let's assume the name of file is '**type_database'**.

Verification phase will crosscheck the current value of 'check_type' with the data of 'type_database' file, whenever an object will start its process of creation. If the current value of 'check_type' matches with any entry of 'type _database' file then object should be created else it should be destroyed and an error on the output device should be shown. In this way, unknown objects of unknown classes will automatically got rejected.

Verification phase should be included as an inbuilt process at the time of creation of an object in object oriented platform. This phase will not only provide security to the application, in fact, the complete object-oriented platform on the system will be under the developer's control.

The object oriented platforms should impose restriction on creating more than one class with the same name. For example: if 'salary' is an existing class on system then another class designed to maintain the salaries of another department should be created with the different name. This approach will stop the entry of unknown classes which are created on the same name of known classes.

IV. SECURITY MODEL TO STOP UNKNOWN OBJECTS OF KNOWN CLASSES

An unauthorized person can create the unknown objects of those classes which will pass the verification phase. So, there is a need of protecting class definitions and methods from intrusion.

Class definitions and methods should be non-editable and non-readable for everyone except the authorized person.[7]Known objects should be protected by developer to avoid the modification of existing data in the database. Class definitions and methods should be protected to stop the addition of unknown classes in it.

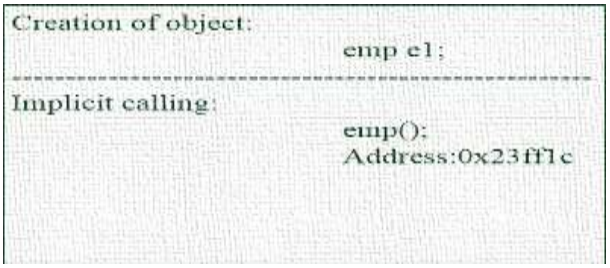

Fig.1. Inbuilt Process to Create an Object of Class 'Emp'

The inbuilt procedure of creating an object of the class is: an implicit call to its own class default constructor at run time i.e. calling emp ()[8].So, emp () should be modified in a way that it rejects unknown objects of known classes.

Example 4.1

int objects_created=0; //represents no. of objects created by the developer in application so far//


```
int size=5; //represent constraint on number of objects specified by developer//

void exit(void) //method to stop the further process of creation of an unknown object//
{
       cout<<"\n unauthorized access";
      getch();
 }
 class emp
{
      public:
        emp() //customized default constructor of 'emp'//
        {
            object_created++;
             if (objects_created>=size)
               {
                    exit();
                }
         }
};
```

Description:

When the object of emp class will call emp () at the time of its creation, 'objects_created' variable will be incremented. In the above example, developer will create five objects of emp class according to his requirement. After that, any object accessing emp () will be directed to exit () method, which will show an error of 'unauthorized access'. Class definitions should be non-editable and non-readable for everyone, except developer. So only he, can change the constraints on number of objects. In this way, an unknown object will be unable to access methods of the class.

A. Security of Inheritance

Inheritance is the special feature of object oriented system.[8] Inheritance allows us to reuse a class 'n' number of times to derive new and more functional classes without bothering about any constraints. The inbuilt process of creating the object of derived type class is different from the non-derived type class. To reject unknown objects, in case of inheritance, a different approach will be used.

1) Security of Single Level Inheritance

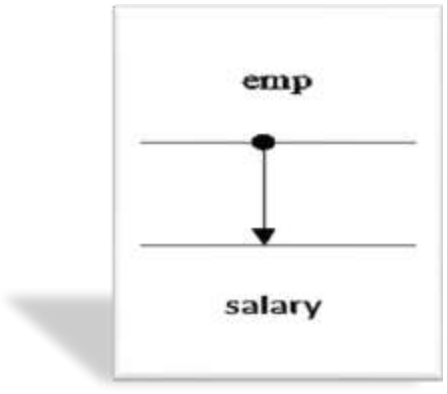

Fig.2. Single Level Inheritance

Description:

Figure 2 shows that emp is a base class for salary. These (emp and salary) are assumed as known classes of defined class hierarchy.

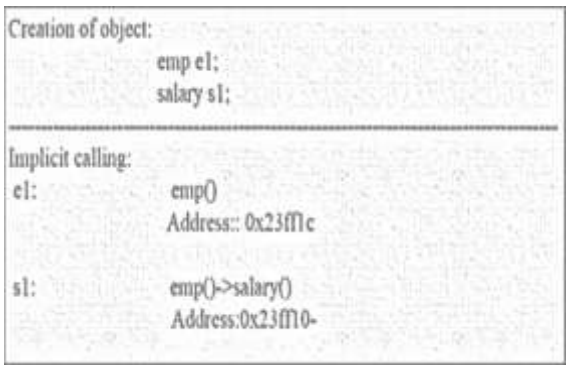

Fig.3. Inbuilt process to create an object of 'salary' ,and 'emp'

In single-level inheritance, the inbuilt process of creating a derived class object is: first implicit call to its base class default constructor, after that, an implicit call to its own class default constructor. At the time of creation, a memory address will be assigned to the object. The inbuilt process of creating 'salary s1' is: first call to emp (), after that, a call to salary ().Figure 3 shows that to stop the creation of unknown objects of salary and emp, 'emp ()' should be programmed in such a way that it rejects unknown objects of class salary.

Example 4.2: 'emp.cpp'

```
int objects_created=0; //represents number of objects created in an application so far//

int size=5;    //represents constraint on number of objects specified by developer//

void exit(void)
{    // method to stop the further process of creation of an unknown object//
  cout<<"\n unauthorized access";
  getch();
}
class emp
{
 emp()      //customized default constructor of 'emp'
 {
   if (objects_created>=size)
    {
     {
       exit();
     }
    }
 }
};
Class salary: private emp
{
```

```
 salary()
 {
 }
};
```

Description:

The above example shows class definitions and methods of emp and salary. It is able to create five known objects of either emp or salary or both because 'size' specified by developer is five. Only developer will be able to change the constraints on number of objects because class definitions are under his control. When unknown objects (after fifth object) of salary or emp will start their process of creation they will first access emp (), which will call exit () method for them. Exit () method will show an error of 'unauthorized access' and stop the further process of creation of the unknown objects of salary i.e. calling salary (). If unknown objects of class salary or emp will not be created they will be unable to access methods of these classes.

The verification phase is not used for identifying unknown objects of known classes because in emp () developer can easily use different type of operators (>=, <,>, <=) according to his requirements which would be tough in case of verification phase.

2) Security of Multilevel Inheritance

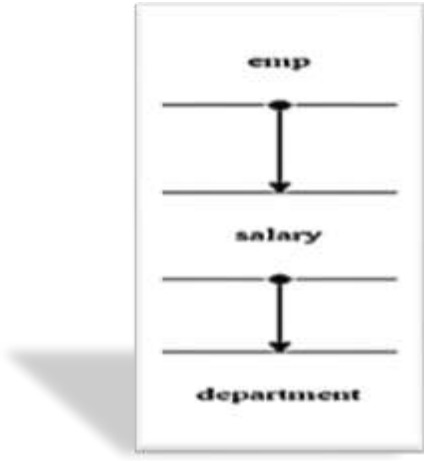

Fig.4. Multilevel Inheritance

Description:

In Figure 4, Department inherited salary which is derived from emp class. These are assumed as known classes of an application.

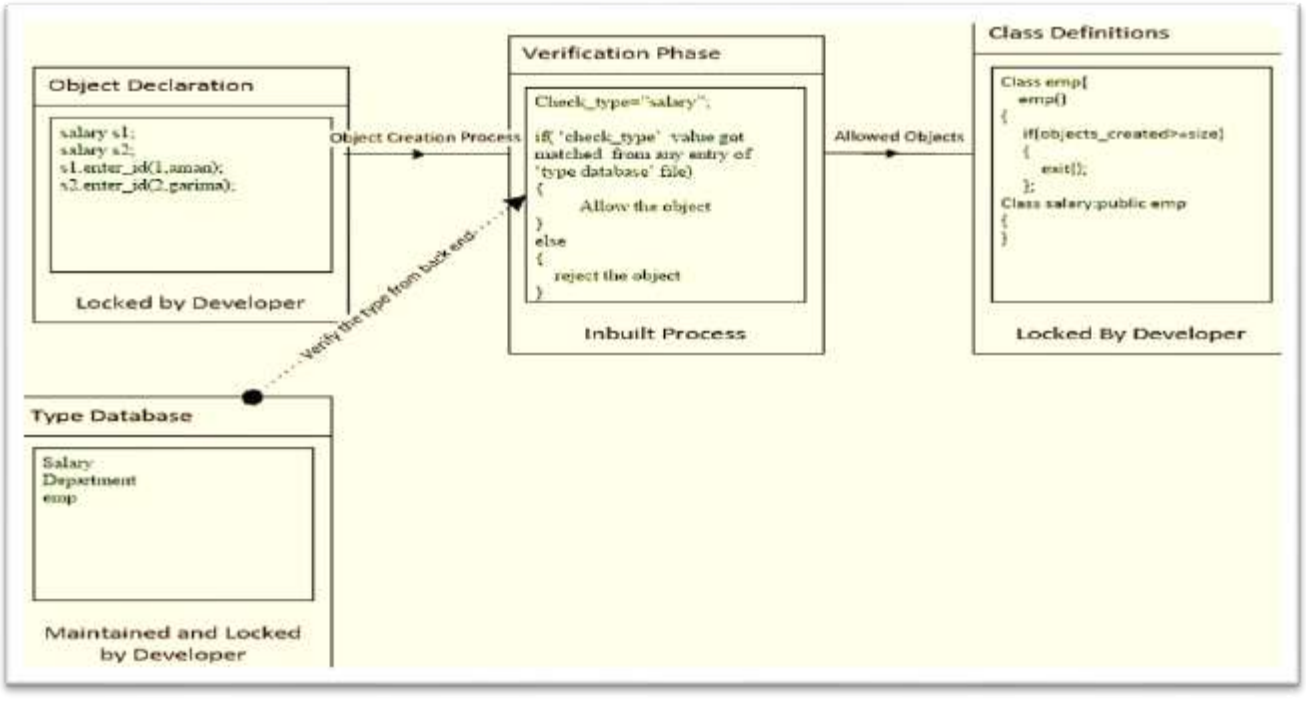

Fig.5. Working of security model in an application

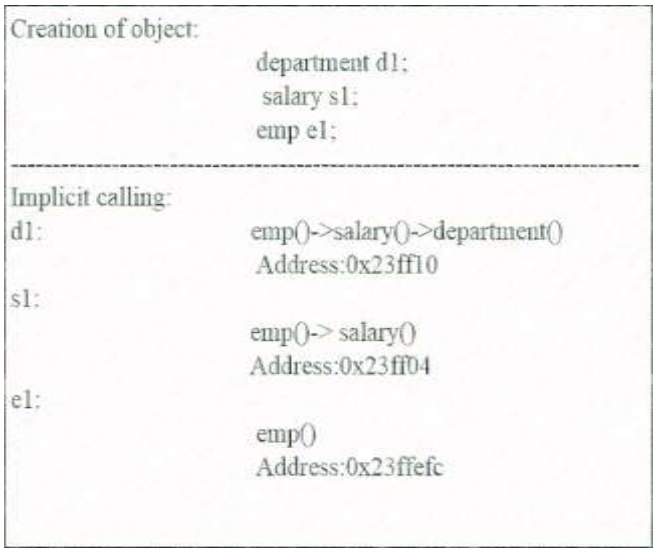

Fig.6. Inbuilt process to create an object of department, salary, and emp

Description:

In multi-level inheritance, the inbuilt procedure of creating a derived class object is: first implicit call to its base class's base class default constructor then an implicit call to its own base class default constructor, after that, an implicit call to its own default constructor. To complete the creation process, 'Department d1' will implicitly call emp() (default constructor of base class of 'salary') ,salary() (its own base class default constructor) and department() in order. Figure 6 shows that to stop the process of creation of unknown objects of department and salary, emp()(default constructor of base class **'emp'**) should be programmed like Example 4.2 according to requirements. Exit () method will stop the further process of creation of the unknown objects of department (i.e. calling salary () ->department) and salary (i.e. calling salary ()).

3) Security Issues of Multiple Inheritance

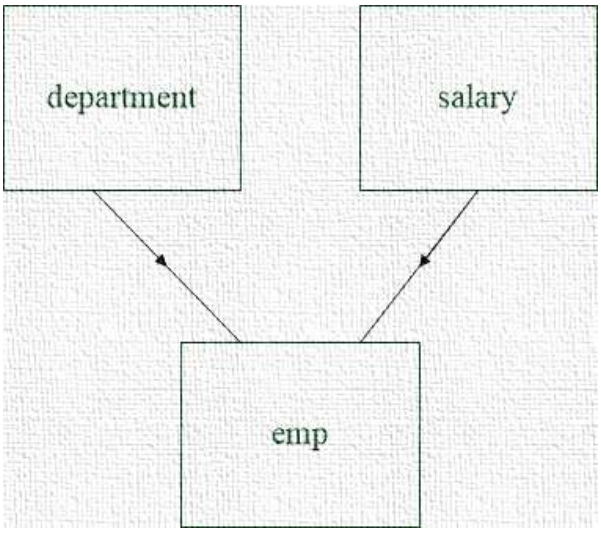

Fig.7. Multiple Inheritance

Description:

In Figure 7, department and salary are assumed as known base classes for emp. The inbuilt process of creation of an object of derived type class in multiple inheritance is different from that of multilevel inheritance. In multiple inheritance the order of implicit calling of base class default constructor follows the order of position of base class in derived class declaration.

For example class emp: public department, public salary.

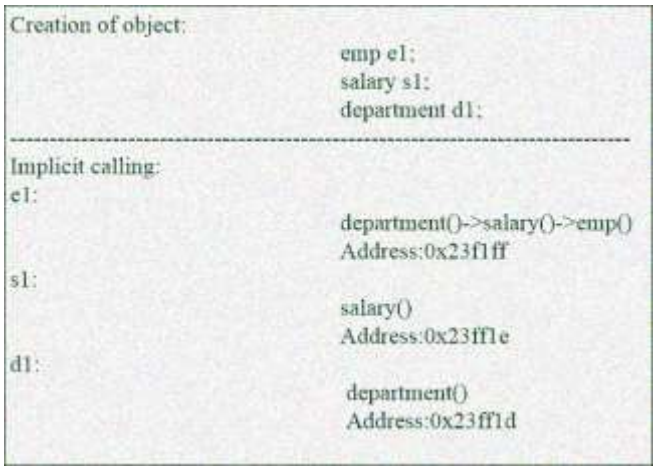

Fig.8. Inbuilt process to create an object of emp, salary, and department

Description:

Figure 8 shows that to complete the inbuilt process of Creation, 'emp e1' will implicitly call department () (the first inherited base class default constructor), salary () (the second inherited base class default constructor) and emp () (its own default constructor) in order. To put constraint on the number of objects of emp and department, department () should be customized according to developer's requirements like Example 4.2. Customized department () will not be able to impose restrictions on those objects which are of type ' salary' as it is an independent class and its objects will call salary() at the time of creation.

V. CONCLUSION

Due to different security requirements of organizations there is need of security model which can be easily customized by the developer or owner.[13] The verification phase as an inbuilt feature in object oriented platform will filter the unknown classes for a developer. The verification phase will protect the object oriented platform. The 'type_database' file can be easily protected by the developer by changing the permissions. Locking of class definitions and methods will stop the unknown additions of classes in it. Putting constraints in default constructor of a class will protect the methods of a class by stopping the unknown access to them. So, discussed methods will make object oriented platform more secure.

REFRENCES

[1] E.B. Fernandez, R.C. Summers, and C. Wood, "Database Security and Integrity," Addison-Wesley, February 1981.

[2] T.F. Keefe, "SODA: A Security Model For Object-oriented Management Systems", Proceedings of 15th International Computer Software and Applications Conference, Tokyo, Japan, September 1991.

[3] Jonathan K. Millen and Teresa F. Lunt, "Security for Object-Oriented Database Systems," IEEE 1992.

[4] Elisa Bertino, "Data Hiding and Security in Object-Oriented Databases", Eighth International Conference on Data Engineering, 338-347, February 1992.

[5] Juhnyoung Lee, Sang H.Son, Myung-Joon Lee, "Issues in devolving object-oriented databases systems for real-time applications," IEEE 1994.

[6] Y. Oki, T. Chikaraishi, T. Shimomura and T. Ohta, "A Design Method for Data Integrity in Object-Oriented Database Systems," IEEE 1995.

[7] Roshan K. Thomas and Ravi S. Sandhu, "A Trusted Subject Architecture for Multilevel Secure Object-Oriented Database," IEEE Transactions on Knowledge and Data Engineering, Vol. 8, No. 1, February 1996.

[8] Elisa Bertino and Ravi Sandhu, "Database Security-Concepts, Approaches, and Challenges", IEEE Transactions on Dependable and Secure Computing, Vol. 2, No.1, January-March 2005.

[9] Ni Xianjun, "A Logic Specification and Implementation Approach for Object Oriented Database Security," Workshop on Knowledge Discovery and Data Mining, IEEE 2008.

[10]Sohail IMRAN and Dr. Irfan Hyder, "**Security** Issues in Databases", Second International Conference on Future Information Technology and Management EngineeringIEEE 2009.

[11]Leon Pan, "A Unified Network Security and Fine-Grained Database Access Control Model," Second International Symposium on Electronic Commerce and Security, 2009.

[12]Mansaf Alam "Access specifiers Model for Data Security in Object Oriented Databases,"Eighth International Conference on Information Technology: New Generations, IEEE 2011.

[13]Hussain Al-Aqrabi; Lu Liu; Richard Hill; Zhijun Ding; Nick Antonopoulos"Buisness Intelligence Security on clouds:Challenges,Solutions and Future Directions",IEEE 2013

[14]Nitish Pathak; Girish Sharma; B. M. Singh "Designing of SPF based secure web application using forward engineering",IEEE 2015.

[15]S. Nagaparameshwara Chary, B.Ram "Analysis of Classification Technique Algorithms in Data mining- A Review",IJCSE 2016.

Author Profile

Amanpreet Kaur received the License degree Information Technology in 2013, then she received the M.S degree in Information Technology in 2014(Integrated), from Lovely Professional University, Jalandhar, Punjab, India. Her research interests are Object Oriented Databases.

.

