Enhancing Data Integrity of Student Registration Input Using Integration of Secure Program Development Technique

Godwin. A. Otu[1], Stephen. E. Iheagwara[2], Akudo C. Okafor[3]

[1, 2, 3]Department of Computer Science, Air Force Institute of Technology Kaduna, Nigeria

Abstract: ***The act of developing secure program modules assist software owners to reduce maintenance cost and increase dependency on the software. Frequent attacks by hackers are mostly aim at corrupting data to undermine software reliability and distort input integrity. Inappropriate data can even result to denial of service to software users if such data crashes the system. In this research the use of secure program development approach will be deployed to generate reliable data for student registration in tertiary institutions. The data is transformed using object oriented approach (OOP) in conformity with software development using modular approach. OOP class construct, decision structure, loop, inheritance and interface will be employed to build the module. The interface is implemented to make sure there is conformity with the data specification. The program will ensure data such as name of student, level, matric number, department, age and grade point average (GPA) pass through a carefully formulated system that will ensure that these data conformed to required data laid down guidelines and conforms to format. This result from the module shows an output of range of references displaying either valid or invalid data depending on if either all the set of input data are either completely conforms to format or not. The module can be adopted by developers for a range of data checking activities during operations; it will also assist programmers to identify permissible data set for a given operation. A graphical interface for the module can be developed and then integrated as data integrity check in applications.***

Keywords: Data integrity, Inheritance, Interface, Secure program development. Software reliability

1. Introduction

According to Salagrama *et al. (*2022) data integrity is related to the serious threat of manipulation and alteration in the course storage and transmission. The data is being altered by malicious actors for getting the advantages which translates to destruction of trust and money. Data users regularly want to ensure that the consuming data is precise and not altered during processing and transmission. Also Nina *et al.* (2021) described secure software development in five development stages which are; software requirement security where elicitation and misuse issues are identified, design security where there is threat model and security patterns, construction security here issues like static code analysis and vulnerability detection are identified and finally testing security which includes penetration testing and vulnerability scanning. Managing the integrity of data is an intricate process for every type of data like data at rest, data at processing and data at transmit. Reply attack is very common with data in transit and this attack causes a serious harm to sender and receiver by losing the trust and money as well.

Many research work have been carried out to mitigate actions that will lead to data being compromised either in storages or during transmission. Krakowiak and Ziemba (2022) used SERM (Structure Entity Relationship Diagram) data model to design a structure for defining and processing rules ideal to the verification of data consistency and integrity. Son *et al*. (2013) evaluates android recovery mode variables that tend to compromise data integrity at the period of data acquisition. Based on the conducted analysis, an Android data acquisition tool that ensures the integrity of acquired data is developed, which is demonstrated in a case study to test tool's strength to preserve data integrity. Hussain *et al*. (2022) presented a flexible and formal methodology that adopts Model - Driven Engineering (MDE) to model closed - world integrity constraints for open - world reasoning. The proposed method offers semantic validation of data by describing integrity constraints at both the model and the code level. Barbaria *et al*. (2023) proposed a blockchain - based architectural model to ensure the integrity of healthcare - sensitive data in an AI based medical research context. The method will use the HL7 FHIR standardized data structure to ensure the interoperability of our approach with the existing hospital information systems (HIS). Ramadhan *et al.* (2022) infused blockchain technology into relational database to build a system with the FastAPI structure using Python programming language, React framework and JavaScript programming language. This system was tested using Katalon and Wireshark software to perform throughput testing and man - in - the - middle attack. Sukumaran and Misbahuddin (2018) designed a bio - computing solution approaches to address data security the proposed methodology is based on the principle concepts of polymerase chain reaction and primer generation for ensuring data confidentiality and integrity. The security analysis of the proposed cryptosystem is evaluated by theoretical analysis, complexity and probability analysis. Christensen *et al*. (2020) developed an architecture that allows for the formal verification and authentication of varied properties of the end - to - end system with a proof of correctness of the assembly - level implementation of the core algorithm in Coq, the integrity of trusted data via a non - interference proof, and a guarantee that the model achieves critical timing requirements. Liu *et al*. (2023) proposed auditing and authenticating scheme which is used to guarantee the correctness and the completeness of the stored data in 5G - enabled software - defined edge computing. The auditing results of the distributed data in the proposed scheme can be used as an important basis for evaluating the trustworthiness of the edge devices. Mittal *et al*. (2015) presented a hash function technique to test distributed environments against threats on data integrity.

The research studies critically looked at have proposed many techniques on how to maintain data integrity in the cloud, others the use of encoding algorithms and come the infusion of block - chain technology with relational databases in order to enforce data integrity. In this research decision structure, loop structure, interface and inheritance concepts will be used to enforce user input data integrity.

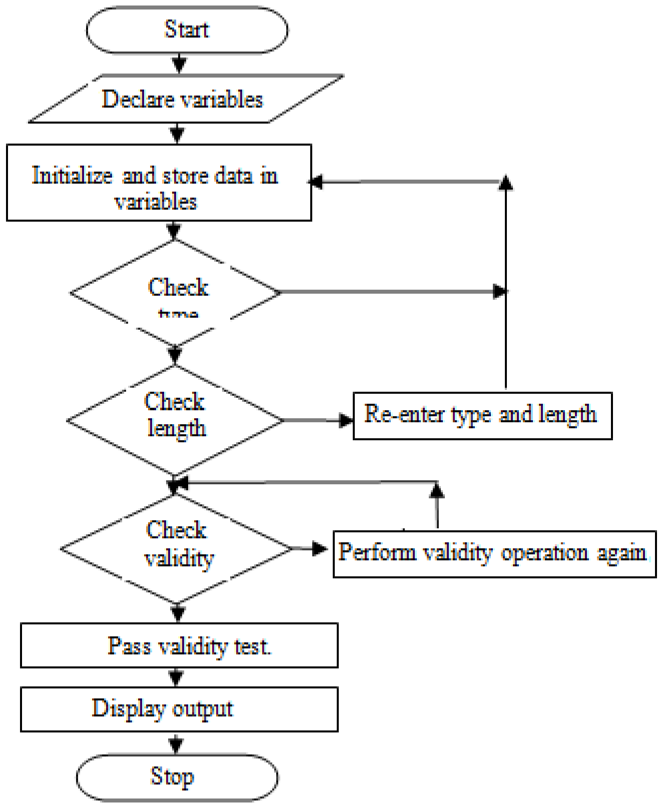
A class is created with student basic registration details and methods used to return the corresponding student types with respect to the details declarations. An interface is created with a check method which is implemented by another class that inherits the student details class. The implementation of the check method for each type is done in such a way as the type integrity is not compromised before the data is used for processing.

2. Methodology

Data Collection
The data used to test the module reliability is from the user input.

Algorithm
1) Start
2) Declare variables
3) Initialize and store data in variables
4) Check type
5) Check length
6) Else re - enter type and length
7) Check data validity
8) Pass validity test.
9) Else perform validity operation again
10) Display output
11) Stop

Flowchart



3. Implementation

```
publicclass studentDetail {
protected String name;
protected String level;
protectedint age;
protected String department;
protected String matricno;
protecteddouble gpa;
public studentDetail (String name, String level, int age, String department, String matricno, double gpa) {
this. name = name;
this. level = level;
this. age = age;
this. department = department;
this. matricno = matricno;
this. gpa = gpa;

 }
 public String getName () {
 return name;
 }
 public String getLevel () {
 return level;
 }
 publicint getAge () {
 return age;
 }
 public String getDepartment () {
 return department;
 }
 public String getMatricNo () {
 return matricno;
 }
 public double getGPA () {
 return gpa;
 }

}
publicclass Implementation extends student Detail
implements DataCheck {
 public Implementation (String name, String level, int age,
String department, String matricno, double gpa) {
 super (name, level, age, department, matricno, gpa);

 }
public boolean check () {
 for (int x = 0; x<name. length (); x++) {
 if (Character. isDigit (getName (). charAt (x))) {
 return false;
 }
 }
 if (getLevel (). isEmpty ()) {
 return false;
 }
 if (age<14) {
 return false;
 }

 if (getDepartment (). isEmpty ()) {
 return false;
 }
 for (int x = 0; x<department. length (); x++) {
 if (Character. isDigit (getDepartment (). charAt (x))) {
return false;

}
 }
 if (getMatricNo (). length () !=8) {
 return false;
 }
 if (gpa<= 0) {
 return false;
 }
 if (gpa> 5) {
 return false;
 }
 return true;

}
}
public interface DataCheck {
 public boolean check ();

}
public class Demonstration {
 public static void main (String [] args) {
 Implementation test1= new Implementation ("Godwin",
"100", 15, "computer science", "12345678", 8);
 Implementation test2= new Implementation ("Professor
Achimugu", "400", 17, "computer science", "12345678", 5);
 Implementation test3= new Implementation ("Oyebanji",
"300", 17, "software Engineering", "20345678", 2.5);
 Implementation test4= new Implementation ("Iheagwara",
"400", 20, "1omputer science", "13456891", 5);
 Implementation test5= new Implementation ("Okafor", "",
20, "computer science", "13456891", 1);
 Implementation test6= new Implementation (" Alcaraz", "
100", 0, "computer science", "13456891", 1.5);
 Implementation test7= new Implementation (" Samuel", "
400", 18, "computer science", "13456891", 4.88);
 Implementation test8= new Implementation ("Lucia", "
200", 20, "computer science", "13456891", 3);
 Implementation test9= new Implementation ("Modupe", "
Civil Engineering", 20, "computer science", "13456891", 3);
 Implementation test10= new Implementation ("Akong", "
400", 20, "computer science", "13456891", 3);
 System. out. println (checkIntegrity (test1));
 System. out. println (checkIntegrity (test2));
 System. out. println (checkIntegrity (test3));
 System. out. println (checkIntegrity (test4));
 System. out. println (checkIntegrity (test5));
 System. out. println (checkIntegrity (test6));
 System. out. println (checkIntegrity (test7));
 System. out. println (checkIntegrity (test8));
 System. out. println (checkIntegrity (test9));
 System. out. println (checkIntegrity (test10));
 }

 public static String checkIntegrity (DataCheckmyCheck) {
 if (myCheck. check ())
 return" valid data";
 else
 return"invalid data";
 }

}
```

4. Result

serial number	Object	Remark
1	test1	invalid data
2	test2	valid data
3	test3	valid data
4	test4	invalid data
5	test5	invalid data
6	test6	invalid data
7	test7	valid data
8	test8	valid data
9	test9	valid data
10	test10	valid data

5. Discussion

The method in the interface is overridden in the class where it is it implemented to give the required restrictions to the input data set in order to enforce data integrity and reliability. The range of reference parameters which are test1 to test2 point to their respective set of values which have been modeled from student details. While Barbaria *et al.* (2023) proposed a modeled block - chain method using artificial intelligence to secure health care data, this research employs a simplified object oriented approach to identify valid and invalid data from user input. Liu *et al.* (2023) also discussed the used of an authentication system to identify the correctness of data for 5G technology, but this system needs a complicated hardware and software implementation which take much more resources and time, but in this research there is a high level of flexibility with less programming time to implemented an abstract method from an interface to suite the chosen data restriction with less difficulty. Nina *et al.* (2021) describe a secure software development strategy which involves five stages of requirement security, design security, construction security, code analysis and testing. This elaborate and detail procedure using conventional software engineering approach has been summarized using object oriented approach. Result on the data clearly shows the objects and the corresponding remark from the module which shows whether a set of data is reliable based on user input or not.

6. Conclusion

The implementation in the research shows the used of inheritance and interface concepts to enforce that the input data conforms to the system specification during input. The module maintain data integrity by preventing the user from entering corrupt data which in return will give a faulty output. Once the system is incorporated into systems hackers with the intention of making the system susceptible to data integrity threat will find it difficult to corrupt the system with inappropriate data.

References

[1] Barbaria, S., Mahjoubi, H., &Rahmouni, H. B. (2023). A novel blockchain - based architectural modal for healthcare data integrity: Covid19 screening laboratory use - case. *Procedia Computer Science, 219*, 1436 - 1443. doi: https: //doi. org/10.1016/j. procs.2023.01.433

[2] Christensen, M., McMahan, J., Nichols, L., Roesch, J., Sherwood, T., &Hardekopf, B. (2021). Safe functional systems through integrity types and verified assembly. *Theoretical Computer Science, 851*, 39 - 61. doi: https: //doi. org/10.1016/j. tcs.2020.09.039

[3] Hussain, A., Wu, W., & Tang, Z. (2022). An MDE - based methodology for closed - world integrity constraint checking in the semantic web. *Journal of Web Semantics, 74*, 100717. doi: https: //doi. org/10.1016/j. websem.2022.100717

[4] Krakowiak, M., &Ziemba, P. (2022). A SERM based framework for defining and processing rules supporting the verification of data consistency and integrity. *Procedia Computer Science, 207*, 4227 - 4236. doi: https: //doi. org/10.1016/j. procs.2022.09.486

[5] Liu, D., Li, Z., &Jia, D. (2023). Secure distributed data integrity auditing with high efficiency in 5G - enabled software - defined edge computing. *Cyber Security and Applications, 1*, 100004. doi: https: //doi. org/10.1016/j. csa.2022.100004

[6] Mittal, M., Sangani, R., & Srivastava, K. (2015). Testing Data Integrity in Distributed Systems. *Procedia Computer Science, 45*, 446 - 452. doi: https: //doi. org/10.1016/j. procs.2015.03.077

[7] Nina, H., Pow - Sang, J. A., & Villavicencio, M. (2021). Systematic Mapping of the Literature on Secure Software Development. *IEEE Access, 9*, 36852 - 36867.

[8] Ramadhan, A. N., Pane, K. N., Wardhana, K. R., &Suharjito. (2023). Blockchain and API Development to Improve Relational Database Integrity and System Interoperability. *Procedia Computer Science, 216*, 151 - 160. doi: https: //doi. org/10.1016/j. procs.2022.12.122

[9] Salagrama, S., Bibhu, V., &Rana, A. (2022). Blockchain Based Data Integrity Security Management. *Procedia Computer Science, 215*, 331 - 339. doi: https: //doi. org/10.1016/j. procs.2022.12.035

[10] Son, N., Lee, Y., Kim, D., James, J. I., Lee, S., & Lee, K. (2013). A study of user data integrity during acquisition of Android devices. *Digital Investigation, 10*, S3 - S11. doi: https: //doi. org/10.1016/j. diin.2013.06.001

[11] Sukumaran, S. C., &Misbahuddin, M. (2021). PCR and Bio - signature for data confidentiality and integrity in mobile cloud computing. *Journal of King Saud University - Computer and Information Sciences, 33* (4), 426 - 435. doi: https: //doi. org/10.1016/j. jksuci.2018.03.008