Journal of Engineering Science and Technology
Vol. 13, No. 9 (2018) 2734 - 2749
© School of Engineering, Taylor's University

# UML 2.0 BASED ROUND TRIP ENGINEERING FRAMEWORK FOR THE DEVELOPMENT OF SPF BASED SECURE APPLICATION

NITISH PATHAK

Bharati Vidyapeeth's Institute of Computer Applications and Management (BVICAM),
Paschim Vihar (East) Metro Station, New Delhi-110063, India
E-mail: nitishforyou@gmail.com

## Abstract

This research paper proposes the UML 2.0 based framework for round-trip engineering and use of Security Performance Flexibility model to keep high security in web applications. This model allows system administrators to skip or disable some unnecessary security checks in trusted operating systems through which, they can effectively balance their performance needs without compromising the security of the system. For example, the system admin can tell that video on demand server is allowed to skip only security checks on reading files, while the database server is allowed to skip only security checks on seeking files. Which operation is needed to be skipped and, which operation is not needed to be skipped is very much subjective in nature, this will depend upon the user's requirement and the particular application's requirement. The selection of these operations for a particular application is the part of software requirement elicitation process. This UML 2.0 based research work proposes Object-Oriented class-based software development, source code generation in C++ and the integration of security engineering into a model-driven software development. On this source code, Halstead software science measures, etc., can be applied. This helps developers in code restructuring; identify probable bugs or deficiencies for probable improvements and helps from the analysis phase to the maintenance phase.

Keywords: Requirement elicitation, Round trip engineering, Security-enhanced version of linux (SELinux), Security performance flexibility (SPF), UML 2.0.

## 1. Introduction

In the last decade, there has been vast growth in the field of networking, sharing of data worldwide. Then comes the most extensively used thing Internet have made cybersecurity a very crucial aspect of research and development. It is a matter of concern for both the common users and researchers connected all over the world [1]. Despite a lot of works undergoing we are still unable to get something that reliable and silver bullet that it may provide us with complete security for our systems. Being so advanced we still lack the basic potential to create such a system that is capable of stopping viruses and accessing our confidential data from our systems [2]. The security methods developed, researched till yet are implemented in the application layer of the computers, which is making our systems more prone to data insecurity. These methods include encryption using a key, i.e., cryptography, using firewalls, access control using authentication and application layer access control [3]. The most two burning domains are cryptography and authentication techniques in which, max research is being done. Although these are something very difficult to crack, no one knows the dynamic minds making some of the probability of data insecurity [4].

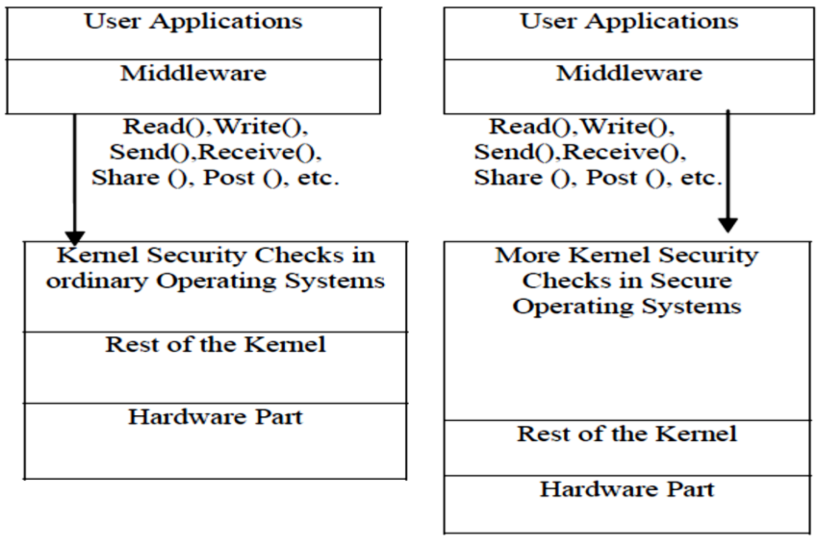
The biggest threat to our application layer is viruses and Trojan Horses. Once these two enters into our system they have the potential to access and even modify each and every data present in the system [5]. Now, these days, to overcome the threats operating system application layer and the network entry points are used to implement the security measures. Although no preventive measures are used inside the kernel of operating systems [6], it is believed that security measures in the kernel are much more effective than the application layer. In fact, after a lot of research such operating systems have been developed, which have much more mechanisms inside the OS kernel providing us with a very good level of security thus securing our systems [7]. In reality, Trusted Operating Systems (TOS) are a better choice for web applications to maintain the security concern, but this security will come at a cost. By using trusted systems, our web application will be more and more secure, but due to more security checks, the performance of the same system will disgrace in all respect [8].

The security is not something expected not only by big organizations but also by common consumers so now concerns are being there on this and many vendors are trying hard to fix the issue [9]. The companies, which came up with some promising operating systems with security features are Argus-Systems Group, AT&T (System V), Hewlett-Packard, Honeywell (Multics), Sun Microsystems, etc. [10]. These operating systems are a better choice for maintaining the security in web applications. In this paper, we are suggesting SPF model of SELinux trusted operating system for maintaining the high security in web applications. Proper definitions of the secure system vary from organization to organization. These secure Systems are more complex for computer administrators to handle and manage. As we know, that conventional operating systems can be managed by system admin easily [11]. Trusted operating systems generally refer to an operating system that provides sufficient support for multilevel security. Such secure Systems require much more extra effort and time to set up the desired security policy on the part of the administrator. The implementation of security policies, as per the requirement of the user, is very complex in such systems [12]. Programmers fluent in secure coding practices can avoid common security flaws in programming languages and follow best practices to help avoid the increasing number of targeted attacks that focus on application vulnerabilities [13]. In this research paper, we are suggesting SPF based SELinux trusted operating systems for maintaining the security concern in web applications.

The Unified Modelling Language (UML) is gradually more used for capturing conceptual object-oriented models of software, as it supports conceptual modelling of real-life domains. UML is used in many customs for expressing the concepts such as software specification, website structure and business modelling [14]. In the proposed design and development, we are using UML 2.0 based conceptual modelling for the development of SPF based secure application. It makes the programming simpler, more effective and manageable [15]. In this paper, we are concerned with the capture of abstract entities (or classes), the associations and relationship existing between them and adjacent ones, as represented in one or more object-oriented diagrams. To retain the balance between security and web application performance, we propose an amalgamation of security-based round-trip engineering [16]. This research work proposes SPF based secure software analysis, SPF based secure software design, SPF based secure software development and roundtrip engineering based software development. Far above the ground, quality of software design is necessary for the success of software [17].

## 2. Trusted Operating System Based Secure Web Applications

The essential structural design of this operating system is shown in Fig. 1. Just for a reminder to the readers; architecture is just a concept although implementation can be done in many ways [18]. The architecture of traditional operating systems is given in Fig. 1(a). System call interface helps the application and middleware interface to communicate with the operating system.


User Applications
Middleware
Read(),Write(), Send(),Receive(), Share (), Post (), etc.
Kernel Security Checks in ordinary Operating Systems
Rest of the Kernel
Hardware Part
User Applications
Middleware
Read(),Write(), Send(),Receive(), Share (), Post (), etc.
More Kernel Security Checks in Secure Operating Systems
Rest of the Kernel
Hardware Part

**(a) Ordinary operating systems.** **(b) Proposed structure of trusted operating systems.**

**Fig. 1. Structure of trusted operating systems and ordinary operating systems [18].**

Figure 1(a) illustrating a thin or slim security layer of operating systems kernel security checks. Now in order to provide higher security, many security checks are there in the kernel of trusted operating systems. Figure 1(b) demonstrates the additional security checks in the kernel [19]. This will cause trusted operating systems to be slower than standard operating systems. Figure 1(b) clearly depicts the thicker layer of kernel security checks. What all security measures are being taken in the kernel security check depends all on implementation and modelling [20]. But the disadvantage of having extra security check is that whenever user tries to do any useful work it need to undergo all the checks thus deteriorating the system overall performance [21].

## 3. Problem Explanation and Solution Methodology

Security of trusted operating systems (TOS) also affect the quality of multimedia and video streaming services. However, in the case of trusted operating systems for every read from disk system have to do repeated checks. Due to repeated security checks, the frame rate and quality of video streaming will be decreased. The result will be worst when the running system is heavily loaded with many programs [22]. In this case when the system is very loaded with multiple processing and multiple applications are in the running mode, definitely the streaming will be very slow and the quality of the video may be very poor [23]. This is not true for video on-demand server because they keep in mind about the security checks, so they care about the video quality rather than read accesses to the server [24]. This selection of operations and system calls will be different from one application to another. So, for the selection of these necessary and unnecessary operations pattern mining is required [25].

For example, admin of the system can disable all the read checks in web server because they are actually useless, which finally increases the throughput of the web server [26]. Web server deals with sole public data and public information. Since the majority of data is public on any web server, the task of checking it during a read from disk is something useless because this data is already readable by each and every user using internet [27]. The real task of security comes when it comes to writing access. For any web server integrity is the main issue rather than its confidentiality. As we had stated that there are many types of workloads that are continuously being checked by the security mechanisms of the kernel in which, many of them are very much useless or undesired in a trusted operating system [28]. The primary concern of these workloads is the quality as well as the integrity of data rather than the security of certain operations since the data they consist can be public. This let us conclude that by disabling security measures of some parts of OS performance can be increased.

The architectural consideration behind the SPF configuration is demonstrated in Fig. 2. This research paper proposes a concept of SPF, in order to gain improved performance and speed for particular system workloads for trusted operating systems. This particular proposed model allows system administrators to skip or disable some unnecessary security checks in these operating systems through which, they can effectively balance their performance needs without compromising the security of the system. Figure 2 provides the administrator's option of disabling security checks for some useless system processes or undesired process. By skipping these processes or system calls, the performance of the system will be improved. These SPF based improved version of secure systems can be used for the desired web application, for maintaining the desired level of security concern in particular application. The performance of the systems surely increased.

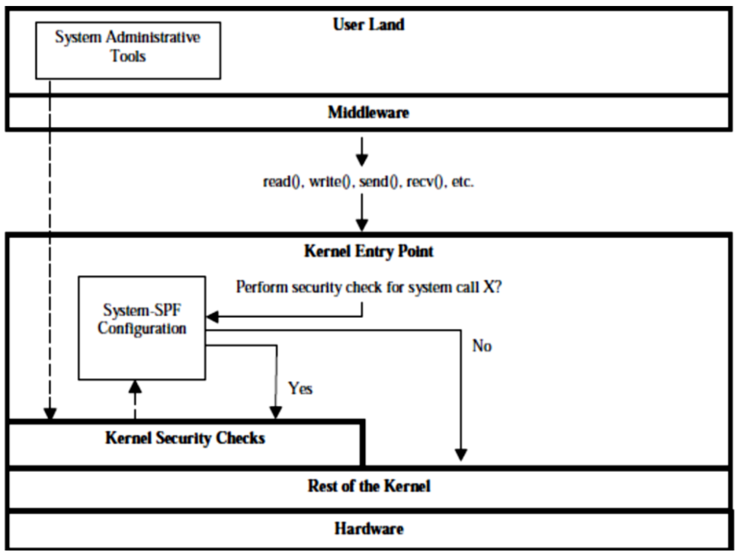


**Fig. 2. System-SPF structural design for stock control web application.**

## 4. Object-Oriented Implementation in C++ for Store Stock Control Based Web Application

The projected approach for the case study is based on the following phases of the development process:

- First of all, identify the pattern for the system calls, through requirement elicitation techniques.
- To represent the functional requirement of a web application, design the use case diagram. Explain almost every use case in a textual approach, i.e., description of major use cases is required for more refinement. If something is not clear through use case diagram then use case description will be very beneficial in other phases of the software development lifecycle.
- Development of class diagram for roundtrip engineering in C++ for the same case study.

Figure 3 shows the functional requirements of the system through use case diagram. A use case model is a business analysis presentation of the steps defining the interactions between a user (called an actor) and a system (usually a computer system). These functional requirements are implemented through functions, member functions, methods, procedures, subroutines, etc. During the analysis phase of the software development life cycle, for a case study, we can see that the store manager, sales assistant, customer, warehouse person, stock manager, etc., are the actor. These actors will be treated as object-oriented classes for the class diagram. Rest of the diagram indicates the use cases (display stock details, payment use case, order delivery use case and sales summary report use case, etc.). These use cases show the functional requirement of the software or web application. The use case diagram of Stock Control system is given in Fig. 3.

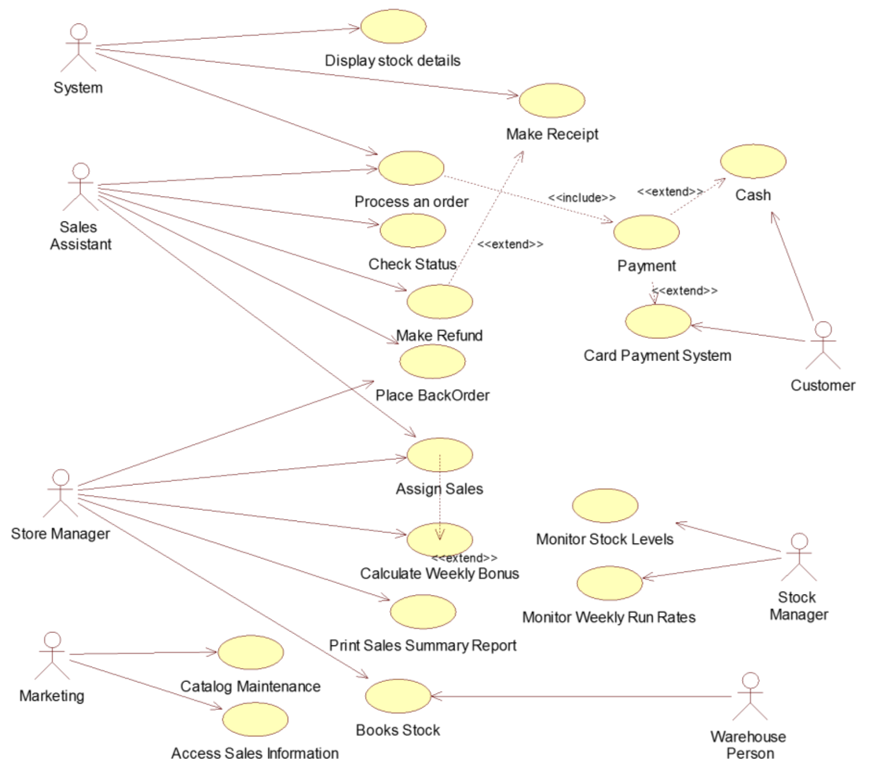


**Fig. 3. Use case model for store stock control based web application.**

## Description of major use cases for web application

A use case is a written description of how users will perform tasks on your website. It outlines, from a user’s point of view, a system’s behavior as it responds to a request. The use case description is a written account of the sequence of steps performed by an analyst to accomplish a complete business transaction. It is initiated by an actor, provides value to that actor, and is a goal of the actor working in that system. Tables 1 to 6 show the use case descriptions of major use cases.

**Table 1. Display stock details.**

| **Brief description** | **The system displays info to sales assistant** |
|---|---|
| **Actors** | System |
| **Flow of events** | • Sales assistant enters the product numbers and required quantities into system<br>• System displays the description of the product |
| **Alternative flow** | If the stocks control system not functioning, it will not start |
| **Precondition** | Product numbers must be entered |
| **Post condition** | Information regarding product is displayed |

**Table 2. Order delivery use case.**

| **Brief description** | **This use case enables the customer to get the delivery of their order from the stock control system** |
|---|---|
| **Actors** | Customer |
| **Flow of events** | • Places order<br>• Enter delivery details |
| **Alternative flow** | If the processing not successful, generate error report to the sales assistant |
| **Precondition** | The order should be successfully processed on the system |
| **Post condition** | If the use case successful, the order shall be delivered to the customer. If not, the system state is unchanged |

**Table 3. Payment use case description.**

| **Brief description of payment** | **Payment use case allows the customer to make payment for his order in the stock control system** |
|---|---|
| **Actors** | Customer |
| **Flow of events** | • Places order<br>• Enter delivery details<br>• Makes payment |
| **Alternative flow** | If the processing not successful, generate error report to the customer showing order is unsuccessful |
| **Precondition** | The order should be successfully processed on the system |
| **Post condition** | If the payment use case successful, the payment for the order shall be made by the customer. If not, the order is incomplete |

**Table 4. Credit card payment use case description.**

| **Brief description** | **The credit/debit card is the mode of payment for the customer.** |
|---|---|
| **Actors** | Customer, sales assistant |
| **Flow of events** | • The customer pays for the goods with credit card<br>• The card payment is verified using online transaction system |
| **Alternative flow** | If the processing not successful, generate error report to the customer showing order is unsuccessful |
| **Precondition** | The order should be successfully processed on the system |
| **Post condition** | If the credit card payment use case was successful, the payment shall be made by the customer. If not, the order is incomplete. |

**Table 5. Cash payment use case.**

| **Brief description** | **This use case enables the customer to make payment for his order in the stock control system through hard cash** |
|---|---|
| **Actors** | Customer |
| **Flow of events** | • Places order<br>• Enter delivery details<br>• Makes payment through cash |
| **Alternative flow** | If the processing not successful, generate error report to the customer showing order is unsuccessful |
| **Precondition** | The order should be successfully processed on the system |
| **Post condition** | If the use case executed successfully, the payment shall be made by the customer. If not, the order is incomplete |

**Table 6. Make refunds use case.**

| **Brief description** | **The sales assistant makes the refunds to the customer by initiating this use case** |
|---|---|
| **Actors** | Sales assistant, customer |
| **Flow of events** | • The customer produces a valid receipt<br>• The refunds are made by the sales assistant<br>• The use case ends |
| **Alternative flow** | Invalid receipt<br>• The customer not produce a valid receipt<br>• Customer will be asked for a valid receipt. If able to produce a valid receipt, the basic flow step "refund" is resumed. Otherwise, the use case ends |
| **Precondition** | Customer made all the payments |
| **Post condition** | Customer get refunds |

The analysis phase of the software development lifecycle is closed to design and design phase is closed to implementation. With the help of the use of case diagram, we develop the component based classes. In the object-oriented class diagram, the designer will identify the classes. These classes can be identified through a Software Requirement Specification (SRS). The actors, which have been identified in the use case diagram, for a specific web application, can be considered as classes in a class diagram. Use cases of use case diagram will be the member function or methods of object-oriented classes. As normal practices, actors of use case diagram are considered as classes and the use cases are considered as member functions or methods of the classes. When we want to model the structure of a system or a web application, we can make use of an object-oriented class diagram. Classes of applications are more or less like entities in entity relationship diagram. When we want to model the interaction among objects in runtime, and sequence of function calling and method invocation, we can design and draw the interaction or sequence diagram for major use cases. As we know, the analysis is close to the design phase of the software development life cycle and design is close to the development. These object-oriented languages are close to real-world mapping [28]. We develop the component based class diagram. The standard class diagram of store stock control is as follows in Fig. 4.

In this store stock control based web application, storing objects may be a sales clerk, inventory, credit card, cheque, store manager, payment, person, marketing, stock manager, person, warehouse person, invoice, system, customer, etc. (see Fig. 4). During the web development, we should note the point that the analysis phase of the Software Development Lifecycle is close to the design and design phase of software development life cycle is close to the implementation phase. The correct transition of analysis phase to design and design phase to development phase is essential. If the analysis is wrong, it means system requirement discovery is wrong. If the analysis and design phase not implemented correctly, it means final software and web application will not accomplish the purpose of users.

The class wise equivalent C++ source code of this case study is given in *Appendix A.*

With the help of above-mentioned software development process, developers can identify software Metrics like no. of data members, no. of data members per superclass, no. of data members per subclass, member functions, the length of the program, Volume, vocabulary of a program, average number of live variables, Count of executable statements, member functions per class, data structure metrics, and information flow, etc. These software metrics can be identified through halstead software science measures and data structure metrics. These metrics are therefore computed statically from the source code. The number of distinct operators, number of distinct operands, the total number of operators and the total number of operands can be identified through the C++ source code. Source Code review improves the overall quality of the software. With the help of the above-mentioned approach, software project planning will become easier for developers.

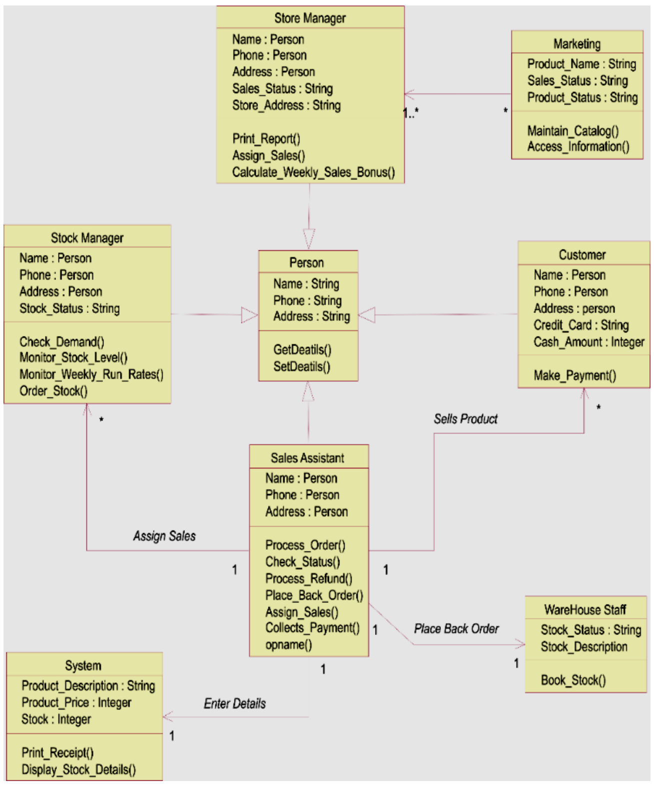


**Fig. 4. Class diagram for online store stock control web application.**

## 5. Transition Process of Roundtrip Engineering

Forward engineering starts with problem statements and reverses engineering start with existing software with source code. Reverse engineering is used to increase the understanding level of the software. Generate ANSI C++ source code from UML class model, and let the UML model reflect the change you made in the source code. Round-trip engineering helps keep your ANSI C++ source code and software design synchronized. Every time you generate code or update UML model, changes will be merged. Round trip engineering is also possible for C#, Java, PHP, XML, Ada95, PERL, Ruby, etc.

In this process, SRS, SDD any diagram or any document can be used, so that maintenance engineer can understand the existing software for the purpose of up gradation. Figure 5 shows the model of round-trip engineering process for complex software development. In this paper, we have used round-trip engineering implementation for the same stock control system web application. The recovered design of old software describes the existing software system, after that, we can design and develop a more secure system. Figure 5 indicates the model of round-trip engineering process for complex software development. In this paper, we have used IBM Rational Rose for the same purpose.

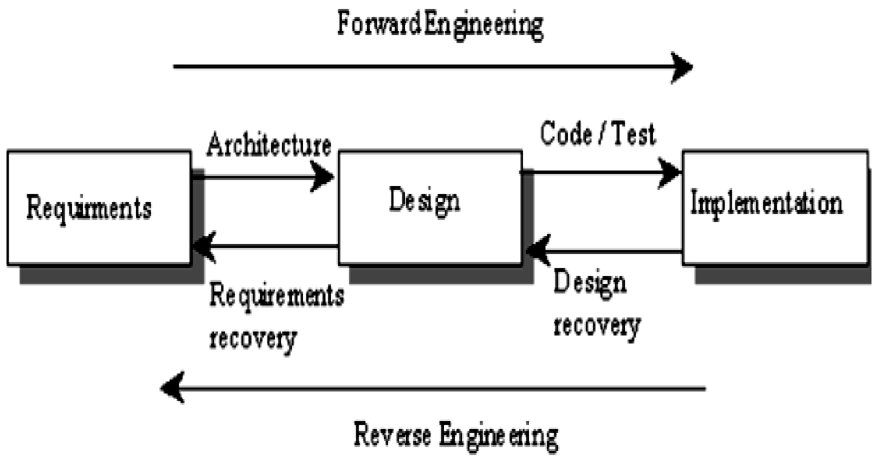


**Fig. 5. Model of round trip engineering process for complex software development.**

## 6. Results and Discussion

Tables 7 to 9 show the performance results that are appropriate to SPF. These tables showing the results with SPF, without SPF and showing the performance compression.

**Table 7. Security checks executed in SELinux trusted operating system.**

| File system tests | SELinux without SPF | SELinux with system-SPF model |
|---|---|---|
| **Random disk reads (K) per second** | 94167 | 93135 |
| **Random disk writes (K) per second** | 79188 | 79508 |
| **Sequential disk reads (K) per second** | 335527 | 325591 |
| **Sequential disk writes (K) per second** | 149616 | 153174 |
| **Disk copies (K) per second** | 102252 | 102744 |

**Table 8. Security checks skipped in SELinux trusted operating system.**

| File system tests | SELinux without SPF | SELinux with system-SPF model |
|---|---|---|
| **Random disk reads (K) per second** | 94167 | 99762 |
| **Random disk writes (K) per second** | 79188 | 84768 |
| **Sequential disk reads (K) per second** | 335527 | 363571 |
| **Sequential disk writes (K) per second** | 149616 | 159727 |
| **Disk copies (K) per second** | 102252 | 110315 |

**Table 9. Comparison of performance improvement after security checks skipped in SELinux trusted operating system.**

| File system tests | SELinux without SPF performance degradation | SELinux with system-SPF model improvement over SELinux no SPF |
|---|---|---|
| **Random disk reads (K) per second** | -6% | +5% |
| **Random disk writes (K) per second** | -6% | +6% |
| **Sequential disk reads (K) per second** | -9% | +7% |
| **Sequential disk writes (K) per second** | -5% | +6% |
| **Disk copies (K) per second** | -7% | +7% |

An added advantage of choosing SELinux is being open source thus allowing modification and change as per your requirement. SPF based SELinux trusted operating system is a better choice for maintaining the security in web applications. Just because of privacy and confidentiality in trusted operating systems, the source code of any software company, business and armed forces will not be available for the normal user. So obtaining such source code in a specific language is not as easy as we think. The privacy and security implementation for any system will vary from one development company to another.

## 7. Conclusions

This research paper presents an SPF based approach for web applications and the integration of security engineering into a model-driven software development. This research work showcases the effectiveness of UML 2.0 based object-oriented modelling with the primary focus of security through the system level SPF in web applications. In this paper, we suggested that the secure web application development should be enhanced by applying security checkpoints and techniques at early stages of development as well as throughout the Software Development Lifecycle. Special emphasis should be applied to the coding phase of development. SPF based model of trusted operating systems allowed system administrators to skip or disable some unnecessary security checks in trusted operating systems through which, they can effectively balance their performance needs without compromising the security of the system. In this paper, we described round trip engineering, source code structuring and restructuring of a secure software system.

Currently, our design is focusing on static, i.e., the fixed design models, which are relatively very close to the object-oriented implementation. Our future research work can focus on modelling security requirements and design information using the dynamic UML models.

## Abbreviations

| | |
|---|---|
| OO | Object-Oriented |
| SDD | Software Design Document |
| SDLC | Software Development Lifecycle |
| SELinux | Security-Enhanced Version of Linux |
| SPF | Security Performance Flexibility |
| SRS | Software Requirement Specification |
| TOS | Trusted Operating System |
| UML | Unified Modeling Language |

## References

1. Valderas, P.; and Pelechano, V. (2014). A survey of requirements specification in model-driven development of web applications. *ACM Transactions on the Web*, 5(2), 1-51.
2. Davis, J.P.; and Bonnel, R.D. (2009). Propositional logic constraint patterns and their use in UML-based conceptual modelling and analysis. *IEEE Transactions on Knowledge and Data Engineering*, 19(3), 427-440.
3. Marcus, A.; Poshyvanyk, D.; and Ferenc, R. (2011). Using the conceptual cohesion of classes for fault prediction in object-oriented systems. *IEEE Transactions on Software Engineering,* 34(2), 287-300.
4. Poblete, B.; Spiliopoulu, M.; and Baeza-Yates, R. (2012). Privacy-preserving query log mining for business confidentiality protection. *ACM Transactions on the Web*, 4(3), Article No. 10.
5. Comai, S.; and Mazza, D. (2013). A model-driven methodology to the content layout problem in web applications. *ACM Transactions on the Web*, 6(3), Article No. 10.
6. Bittar, T.J.; Fortes, R.P.M.; Lobato, L.L.; and Watanabe, W.M. (2009). Web communication and interaction modelling using model-driven development. *Proceedings of the 27th International Conference on Design of Communication (SIGDOC'09)*. Bloomington, Indiana, United States of America, 193-197.
7. Medeiros, I.; Neves, N.F.; and Correia, M. (2014). Automatic detection and correction of web application vulnerabilities using data mining to predict false positives. *Proceedings of the International World Wide Web Conference Committee* (WWW'14). Seoul, Korea, 63-73.
8. Andrea, D.L.; Carmine, G.; Rocco, O.; and Genoveffa, T. (2009). An experimental comparison of ER and UML class diagrams for data modelling. *Empiral Software Engineering*, 15(5), 455-492.
9. Selby, R.W.; Basili, V.R.; and Baker, F.T. (1987). Clean software development: An empirical evaluation. *IEEE Transactions on Software Engineering*, SE-13(9), 1027-1037.
10. Fernandes, J.M.; and Machado, R.J. (2001). From use cases to objects: An industrial information systems case study analysis. *Proceedings of the 7th International Conference on Object-Oriented Information Systems (OOIS '01)*. Calgary, Canada, 319-328.
11. Cheng, B.H.C.; and Wang, E.W. (2002). Formalizing and integrating the dynamic model for object-oriented modelling. *IEEE Transactions on Software Engineering*, 28(8), 747-762.
12. Ricci, L.A.; and Schwabe, D. (2006). An authoring environment for model-driven web applications. *Proceedings of the 12th Brazilian Symposium on Multimedia and the Web*. New York, United States of America, 11-19.
13. Bernardi, S.; Merseguer, J.; and Petriu, D.C. (2012). Dependability modelling and analysis of software systems specified with UML. *ACM Computing Surveys*, 45(1), Article No. 2.
14. Chauron, M.R.V.; Heijstek, W.; and Nughoro, A. (2012). How effective is UML modelling? An empirical perspective on costs and benefits. *Software and System Modeling*, 4(11), 571-580.
15. Pathak, N.; Sharma, G.; and Singh, B.M. (2015). Trusted operating system based model-driven development of secure web applications. *Proceedings of*

*the 50th Golden Jubilee Annual Convention, Computer Society of India (CSI-2015)*. New Delhi, India.

16. Runeson, P.; and Martin, H. (2008). Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering* 14(2), 131-164.
17. Pathak, N.; Sharma, G.; and Singh, B.M. (2017). Towards designing of SPF based secure web application using UML 2.0. *International Journal of Systems Assurance Engineering and Management*, 8(1), 208-218.
18. Siau, K.; and Lee, L. (2004). Are use case and class diagrams complementary in requirements analysis? An experimental study on use case and class diagrams in UML. *Requirements Engineering,* 9(4), 229-237.
19. Brambilla, M.; Ceri, S.; Fraternali, P.; and Manolescu, I (2006). Process modelling in web applications. *ACM Transactions on Software Engineering and Methodology,* 15(4), 360-409.
20. Kapitsaki, G.M.; Kateros, D.A.; Pappas, C.A.; Tselikas, N.D.; and Venieris, I.S. (2008). Model-driven development of composite web applications. *Proceedings of the 10th International Conference on Information Integration and Web-based Applications and Services* (iiWAS'08). Linz, Austria, 309-402.
21. Pathak, N.; Sharma, G.; and Singh, B.M. (2015). Designing of SPF based secure web application using forward engineering. *Proceedings of the 2nd International Conference on Computing for Sustainable Global Development-* (Indiacom). New Delhi, India, 464-469
22. Desnoyers, P.; Wood, T.; Shenoy, P.; Singh, R.; Patil, S.; and Vin, H. (2012). Modellus: Automated modelling of complex internet data center applications. *ACM Transactions on the Web*, 6(2), Article No. 8.
23. Pathak, N.; Singh, B. M.; and Sharma, G. (2017). UML 2.0 based framework for the development of secure web application. *International Journal of Information Technology*, 9(1), 101-109.
24. Kim, H.; Zhang, Y.; Oussena, S.; and Clark, T. (2009). A case study on model driven data integration for data centric software development. *Proceedings of the ACM First International Workshop on Data-Intensive Software Management and Mining (DSMM '09)*. Hong Kong, China, 1-6.
25. Pathak, N.; Sharma, G.; and Singh, B.M. (2015). Experimental analysis of SPF based secure web application. *International Journal of Modern Education and Computer Science* (*IJMECS*), 7(2), 48-55.
26. Kim, P.J.; and Noh, Y.J. (2013). Mobile agent system architecture for supporting mobile market application service in mobile computing environment. *Proceedings of the IEEE International Conference on Geometric Modelling and Graphics*. London, United Kingdom, 149-153.
27. Pathak, N.; Sharma, G.; and Singh, B. M. (2017). An empirical perspective of roundtrip engineering for the development of secure web application using UML 2.0. *International Journal of Intelligent Systems and Applications*, 9(5), 43-54.
28. Hernes, M.; Maleszka, M.; Nyuyen, N.T.; and Bytniewski, A. (2015). The automatic summarization of text documents in the cognitive integrated management information system. *Proceedings of the IEEE Federated Conference on Computer Science and Information Systems* (*FedCSIS*). Lodz, Poland, 1387-1396.

## *Appendix A*

**The class wise equivalent C++ sample code of the case study.**

```
Class Person
{
public:
//##ModelId=553C976E01C4
GetDeatils();
//##ModelId=553C97750038
SetDeatils();
private:
//##ModelId=553C971B02D3
String Name;
//##ModelId=553C97380226
String Phone;
//##ModelId=553C975B0001
String Address;
};
#include "Person.h"
//##ModelId=553C9B4C0295
class Customer : public Person
{
public:
//##ModelId=553C9B890326
Make_Payment();
private:
//##ModelId=553C9B5A003E
Person Name;
//##ModelId=553C9B5E0028
Person Phone;
//##ModelId=553C9B64019E
person Address;
//##ModelId=553C9B680189
String Credit_Card;
//##ModelId=553C9B6F0007
Integer Cash_Amount;
};
#endif /*
#include "Person.h"
//##ModelId=553C99010386
class Stock Manager : public Person
{
public:
//##ModelId=553C99340150
Check_Demand();
//##ModelId=553C99380262
Monitor_Stock_Level();
//##ModelId=553C9940011C
Monitor_Weekly_Run_Rates();
//##ModelId=553C994F005E
Order_Stock();
private:
//##ModelId=553C9912016D
Person Name;
//##ModelId=553C991601B4
Person Phone;
//##ModelId=553C991B0229
Person Address;
//##ModelId=553C9920005D
String Stock_Status;
};
#endif /*
WAREHOUSE_STAFF_H_HEADER_INCLUDED_AAC32
A1B */
//##ModelId=553C9C2202FC
class WareHouse Staff
{
public:
//##ModelId=553C9C4B0368
Book_Stock();
private:
//##ModelId=553C9C2B02DE
String Stock_Status;
//##ModelId=553C9C330009
Stock_Description;
};
#endif /*
Calculate_Weekly_Sales_Bonus();
private:
//##ModelId=553C97B30348
Person Name;
//##ModelId=553C97BD029E
Person Phone;
//##ModelId=553C97C5007C
Person Address;
//##ModelId=553C97D101AE
String Sales_Status;
//##ModelId=553C97E402A7
String Store_Address;
};
#endif /*
//##ModelId=553C981A01B5
Store Manager::Calculate_Weekly_Sales_Bonus()
{
}
#include "System.h"
//##ModelId=553C9ACD031A
System::Print_Receipt()
{
}
```

```
//##ModelId=553C9AD20266
System::Display_Stock_Details()
{
}
#include "WareHouse Staff.h"
//##ModelId=553C9C4B0368
WareHouse Staff::Book_Stock()
{
}
#include "Customer.h"
//##ModelId=553C9B890326
Customer::Make_Payment()
{
}
//##ModelId=553C9A3E007C
Sales Assistant::Place_Back_Order()
{
}
//##ModelId=553C9A4A0191
Sales Assistant::Assign_Sales()
{
}
//##ModelId=553C9A530207
Sales Assistant::Collects_Payment()
{
}
```